

University of Nottingham
UK | CHINA | MALAYSIA

Denoising and Semantic Segmentation of 4D Datasets with Undersampled Micro X-ray CTs using Deep Learning

by

Dimitrios Bellos

Student ID number: 4294749

Supervised by: Andrew P. French
Mark Basham
Tony Pridmore

in

Computer Vision Laboratory (CVL)
School of Computer Science

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy

July 2021

Abstract

Over recent years, multiple imaging approaches has been proposed to depict the hidden inner structure of many different biological and non-biological systems in the form of 3D volumes. As time progressed due to multiple technological advances, these methods have become faster and more accessible, allowing in recent years the collection of time-resolved volumes that form 4D datasets. With 4D capturing techniques, the hidden inner structure of a system is collected repeatedly through time, allowing us to perceive not only its 3D structure, but temporal changes of this structure. Nevertheless, despite the technological advances in data collection there is still a major need for accurate, efficient and easy to use methods for 3D and 4D data analysis.

One of the most difficult tasks associated with 3D or 4D studies is their semantic segmentation. Unfortunately for such high-dimensional data, performing this task manually may take many days of human work, or even months in the case of 4D data. Moreover for the case of 4D data, there are other problems that make their semantic segmentation challenging. In particular, for time-series of micro X-ray Computed Tomograms (CTs), each tomogram is often undersampled in terms of projections due to time restraints. Specifically, in order to capture sufficient numbers of projections and perform the time-series collection quickly in order to capture temporal phenomena, the time allocated for each of its tomograms is low. Because of this only a limited amount of well-exposed projections can be collected. This sets an ill-posed reconstruction problem, and the resulting tomogram's reconstructions are low in signal-to-noise ratio. While computed tomography challenges grew, a new branch of computer vision approaches using deep learning have delivered promising approaches that address both the the problem of a low signal-to-noise ratio, and semantic segmentation. Inspired by these computed tomography challenges and the considerable promise of deep learning approaches, this thesis will propose approaches that address denoising and semantic segmentation of time-series, undersampled tomograms.

Initially, we propose a fast projection-upsampling method by upscaling a tomogram's sinograms using a deep learning-driven super-resolution approach. For our method works to upsample undersampled tomograms in a time-series with the use of a representative highly-sampled tomogram during training. These representative highly-sampled tomograms are usually collected either at the beginning or the end of a time-series, where their collection is not detrimental to the collection of the time-series. These tomograms are collected for spatial reference, providing essential spatial information for the restoration of the undersampled tomograms in our approach described here. Our method, named UDNN, designed to compete against analytical interpolation techniques with fast inference times is able to outperform them both on real-world and synthetic data.

Furthermore, we propose a second network for denoising and semantic segmentation of the undersampled tomograms' reconstructions using again the representative highly-sampled tomograms. Our proposed DenseUSeg approach expands upon Bui *et al.*'s DenseSeg architecture with a U-Net inspired decoder. Additionally, our proposed Stacked-DenseUSeg approach, by the stacking of two of our DenseUSegs, is able to produce both a denoising

and a segmentation output in an end-to-end fashion. Through this process, the denoising output can be utilised to offer more accurate segmentation. Both our methods are able to outperform other state-of-the-art deep learning approaches for volume segmentation, with our Stacked-DenseUSeg achieving the highest accuracy of all. They are also compared against the earlier state-of-the-art methods for the task of denoising. While the earlier state-of-the-art methods are designed for segmentation, we decided to test our proposed networks against them also for the task of denoising since state-of-the-art tomogram denoising methods primarily employ less sophisticated CNN architectures and comparison against them would not be fair. In our experiments, DenseUSeg performs either better or equal against them for the task of denoising, and while Stacked-DenseUSeg is only slightly less accurate than DenseUSeg, the visual quality of its denoised outputs is noteworthy compared to the state-of-the-art methods solely trained for denoising.

To help handle the scarcity of real-world annotated data, we propose a novel use of knowledge transfer using parameter initialisation with networks pretrained on *synthetic tomograms*. During experiments it is discovered that with the use of pretrained nets on realistically constructed synthetic tomograms, it is possible improve Stacked-DenseUSeg segmentation accuracy and at the same time reduce the amount of real-world annotated data needed. Moreover, due to the use of less real-world data, it possible for the training to require less time.

Finally, having addressed the semantic segmentation of undersampled tomograms in a time-series, the temporal dimension remains though underutilised. This because our proposed Stacked-DenseUSeg provides segmentation predictions for each tomogram in a time-series independently though time. For this reason we propose a novel use of hidden Markov models for the refinement of tomogram's segmentation using temporal information. Namely, we propose our HMM-T, that use temporal information from temporally adjacent segmentations of tomograms and our HMM-TC, which also utilises confidence (probabilities) from these segmentations as produced by the network that generated them (Stacked-DenseUSeg). During experiments our HMM-T improved the accuracy of the tomogram segmentation both qualitatively and quantitatively. On the other hand, our HMM-TC demonstrated only qualitative and visual improvements, which in specific areas are even better that the ones of HMM-T. Because of that we speculate that with a more precise fine-tuning of its parameters it may even possible to surpass HMM-T.

The combination of these methods enables the processing of significant quantities of time-series of undersampled tomograms collected at Diamond Light Source. The aim of this work is to enable researchers who use time-series of undersampled CT collections (4D datasets) to quickly and easily extract useful information from the torrent of 4D data that is routinely collected in synchrotron locations.

Acknowledgements

With the completion of this thesis, I would like to first thank Andrew P. French, Mark Basham and Tony Pridmore, who accepted me to for this PhD position and aided me with my research, with helpful ideas and suggestions and most of all by editing our publications as well as this thesis.

Furthermore, I would like to thank Diamond Light Source for access to beamline I13-2 (MT9396) that contributed to the results presented here, and Alison Davenport and her team at the University of Birmingham for the sample preparation and experimental method employed. I would also like thank Diamond Light Source and the School of Computer Science at Nottingham for jointly funding me with PhD STU0185 Studentship.

Finally, I would also like to thank my next of kin, for their immeasurable support during the years of my PhD.

Contents

Abstract **i**

Acknowledgements **iii**

List of Tables **xii**

List of Figures **xv**

1 Introduction **1**

1.1 Motivation, aims and objectives 4

1.2 Thesis overview 6

1.3 Contributions 9

1.4 Summary 11

2 Literature review **13**

2.1 Super-resolution 15

2.1.1 Frequency-domain-based SR methods 15

2.1.2 Interpolation-based SR methods 18

2.1.3 Regularisation-based SR methods 21

2.1.4 Learning-based or example-based SR methods 24

2.1.5 Deep learning SR methods 30

2.1.5.1 Deep learning SR methods for undersampled tomogram reconstructions 38

2.2 Semantic segmentation 41

2.2.1 Segmentation methods based on region discontinuities and boundaries 41

2.2.1.1 Edge detection segmentation methods 42

2.2.1.2 Contour-based segmentation methods 43
2.2.2 Segmentation methods based on region similarities 44
2.2.2.1 Segmentation methods based on thresholding 44
2.2.2.2 Segmentation methods based on growing, splitting or merging regions 45
2.2.2.3 Segmentation methods based on clustering 48
2.2.2.4 Learning-based segmentation methods 50
2.2.3 Deep learning semantic image segmentation 55
2.2.3.1 Fully convolutional networks (FCNs) 55
2.2.3.2 FCNs with graphical models 57
2.2.3.3 Region-based convolutional networks (R-CNNs) 60
2.2.3.4 Encoder-decoder architectures 63
2.2.3.5 Multi-scale, pyramid and dilated FCNs 69
2.2.3.6 Segmentation networks with attention mechanisms 73
2.2.3.7 Generative adversarial networks (GANs) for segmentation 76
2.2.3.8 Recurrent neural networks (RNNs) for segmentation . . . 79
2.2.4 Semantic segmentation of 3D volumes or high-dimensional data with deep learning 81
2.3 CT acquisition and reconstruction 89
2.3.1 Micro X-ray computed tomography 89
2.3.2 Sinogram space 96
2.4 Summary 99

3 Time-resolved undersampled X-ray CTs 101

3.1 Time-series X-ray micro CT datasets 101
3.2 The datasets used in our proposed approaches 106
3.2.1 Nature of real-world datasets-tomograms 106
3.2.2 Real-world datasets-tomograms 106
3.2.3 Synthetic datasets-tomograms 107
3.2.4 Real-world data challenges 109

3.3 Summary 112

4 UDNN: A CNN network for fast upsampling of X-ray CT time-series 114

4.1 The upscaling deep neural network 114

4.2 Methodology and network architecture 117

4.2.1 Feature extraction 117

4.2.2 Depth increase and non-linearity 118

4.2.3 Construction of the upscaled output 119

4.3 Formulation 121

4.4 Training, validation and testing 124

4.4.1 Loss function 124

4.4.2 Backpropagation and learning rate 125

4.4.3 Data allocation to training, validation and testing 126

4.4.4 Input sampling 128

4.4.5 Training and validation process 130

4.4.6 Testing process 132

4.4.7 Inference 134

4.4.8 The UDNN model and parameters 134

4.5 Experiments 135

4.5.1 Experiments for the hyperparameter selection and tuning 137

4.5.1.1 The effects of the number of features on performance . . . 138

4.5.1.2 The effects of the sizes of kernels on performance 140

4.5.1.3 Conclusion on hyperparameter tuning 142

4.5.2 Experiments with synthetic data 142

4.5.2.1 Sinogram upscaling of synthetic data 144

4.5.2.2 Reconstruction of synthetic data 147

4.5.3 Experiments with real-world data 150

4.5.3.1 Experimental complexities 150

4.5.3.2 Sinogram upscaling of real-world data 151

4.5.3.3 Reconstruction of real-world data 156

4.6 Discussion 161
4.7 Application to a time-series of tomograms 163
4.8 Conclusions 166

5 Denoising and segmentation networks of low SNR reconstructions of undersampled tomograms 168

5.1 Densely connected architectures for denoising and segmentation 168
5.2 Proposed networks' functionality and methodology 170
5.2.1 The Stacked-DenseUSeg's output and its functionality 172
5.2.2 DenseUSeg: Our proposed encoder-decoder network and its functionality 173
5.3 The network architectures 173
5.3.1 Densely connected convolutional layers 174
5.3.2 MSD: Mixed-scale dense network 178
5.3.3 3D DenseSeg 182
5.3.4 MSD and DenseSeg comparison 183
5.3.5 DenseUSeg 185
5.3.6 Stacked-DenseUSeg 189
5.4 Dataset and training 192
5.4.1 Highly-sampled representative tomograms of 4D datasets used for training 192
5.4.2 Tomogram reconstructions used for training 194
5.4.3 Hyperparameters and training settings 197
5.4.4 Validation and testing 199
5.4.5 Metrics and execution time 201
5.5 Experiments 202
5.5.1 Experiment 1: Denoising with DenseUSeg and Stacked-DenseUSeg . 203
5.5.2 Experiment 2: Segmentation with DenseUSeg using a high SNR reconstruction as the input 208

5.5.3 Experiment 3: Segmentation with Stacked-DenseUSeg and DenseUSeg with a low SNR reconstruction as the input 213
5.6 Conclusions 218

6 Denoising and segmentation knowledge transfer via the use of synthetically generated tomograms 220

6.1 Network parameter initialisation via trained networks on synthetic data . . 220
6.2 Knowledge transfer 223
6.2.1 Parameter initialisation using pretrained nets 223
6.2.2 Synthetic phantom used for knowledge transfer 226
6.2.3 Synthetic tomograms used for knowledge transfer 228
6.3 Training using knowledge transfer 234
6.3.1 Representative real-world and synthetic tomograms 234
6.3.2 Tomogram reconstructions used for training 235
6.3.3 Training, validation and testing 238
6.3.4 Hyperparameters and backpropagation 239
6.4 Experiments exploring use of differing amounts of annotated data 241
6.4.1 Experiment 1: Effects of knowledge transfer to the task of denoising 242
6.4.2 Experiment 2: Effects of knowledge transfer to the task of semantic segmentation 252
6.5 Discussion and time management 260
6.6 Conclusions 266

7 Temporal refinement of semantic segmentations in 4D datasets using hidden Markov models 269

7.1 Towards the segmentation of time-series of tomograms (4D datasets) . . . 269
7.2 Tomogram translation for spatial continuity through time 271
7.3 Methodology 274
7.3.1 HMM-T: Hidden Markov model that uses inter-tomogram information from segmentation predictions 276

7.3.2 HMM-TC: Hidden Markov model that uses inter-tomogram information and segmentation confidence from segmentation predictions 278
7.3.3 Selection of hidden Markov model parameters for the segmentation of time-series of tomogram reconstructions 279
7.4 Experiments 291
7.4.1 Experiment 1: Qualitative assessment of the HMM predictions . . . 291
7.4.2 Experiment 2: Behavioural experiments using HMM-T 294
7.4.3 Experiment 3: Quantitative experiments 299
7.4.4 Experiment 4: Sensitivity experiments for the elected transition probabilities 307
7.4.4.1 Sensitivity to the transition probabilities of Air 307
7.4.4.2 Sensitivity to the transition probabilities of Aluminium . . 309
7.4.4.3 Sensitivity to the transition probabilities of Magnesium . . 312
7.4.4.4 Sensitivity to the transition probabilities of Air Pockets . . 314
7.5 Discussions 316
7.6 Conclusions 319

8 Conclusions and future work 322
8.1 Overview of the main contributions 322
8.2 Limitations 326
8.2.1 Limitations of our UDNN approach 326
8.2.2 Limitations of our DenseUSeg and Stacked-DenseUSeg approaches . 329
8.2.3 Limitations of our knowledge transfer scheme for Stacked-DenseUSeg330
8.2.4 Limitations of our hidden Markov models and the refining of 4D segmentations 331
8.3 Future Work 334
8.3.1 Potential improvements on our UDNN approach 334
8.3.2 Potential improvements on our DenseUSeg and Stacked-DenseUSeg approaches 335

8.3.3 Potential improvements on our knowledge transfer scheme for Stacked-DenseUSeg . . . 336
8.3.4 Potential improvements our hidden Markov models and the refining of 4D network predictions . . . 337
8.4 Concluding remarks . . . 338

Bibliography **340**

Bibliography **340**

List of Tables

4.1 Sinogram upscaling validation PSNR scores of UDNN, UDNN-128, UNND-32 and cubic interpolation using real-world data 138

4.2 Sinogram upscaling validation PSNR scores of UDNN, UDNN-LK, UNND-SK and cubic interpolation using real-world data 140

4.3 Sinogram upscaling PSNR testing results of UDNN in comparison with the state-of-the-art using synthetic data 145

4.4 Reconstruction PSNR and SSIM testing results of UDNN in comparison with the state-of-the-art using synthetic data 148

4.5 Sinogram upscaling PSNR testing results of UDNN in comparison with the state-of-the-art using synthetic data 154

4.6 PSNR testing results of sinogram upscaling with multiplier x4 for UDNN and cubic interpolation using synthetic data 154

4.7 Reconstruction PSNR and SSIM testing results of UDNN-128 in comparison with the state-of-the-art using real-world data 158

4.8 Reconstruction PSNR and SSIM testing results with multiplier x4 for UDNN-128 and cubic interpolation using real-world data 160

4.9 Computational time of our UDNN-128 against the state-of-the-art Bertram's directional interpolation for the task of sinogram upscaling 163

5.1 Testing mean IoU and total training time for 3D U-Net, 3D DenseSeg_64 and MSD_100 185

5.2 PSNR and SSIM results from testing denoising using the 91-projection CGLS reconstruction as input and the 3601-projection FBP reconstruction as ground truth 205

5.3 IoU testing segmentation results using the 3601-projection FBP reconstruction . . . 210
5.4 Precision testing segmentation results using the 3601-projection FBP reconstruction . . . 211
5.5 Recall testing segmentation results using the 3601-projection FBP reconstruction . . . 211
5.6 IoU testing segmentation results using the 91-projection CGLS reconstruction 214
5.7 Precision testing segmentation results using the 91-projection CGLS reconstruction . . . 215
5.8 Recall testing segmentation results using the 91-projection CGLS reconstruction . . . 215

6.1 PSNR and SSIM results of the denoising output of Stacked-DenseUSeg during testing using progressively lower amounts of the real-world training samples post the use of knowledge transfer . . . 246
6.2 IoU results of the segmentation output of Stacked-DenseUSeg during testing using progressively lower amounts of the real-world training samples post the use of knowledge transfer . . . 256
6.3 Human and computational time required with or without knowledge transfer for the development of Stacked-DenseUSeg that can segment real-world data . . . 263

7.1 IoU segmentation results using as ground truth the annotations from the representative highly-sampled tomogram of the time-series. The comparison is performed in an area of the sample where temporal events are minimal 300
7.2 IoU segmentation results using as ground truth our manual 4D annotations. The comparison is performed in an area of the sample where temporal events are likely to have happened . . . 305
7.3 IoU segmentation results to measure the sensitivity of the empirically elected transition probabilities of air . . . 308

7.4 IoU segmentation results to measure the sensitivity of the empirically elected transition probabilities of aluminium 311
7.5 IoU segmentation results to measure the sensitivity of the empirically elected transition probabilities of magnesium 313
7.6 IoU segmentation results to measure the sensitivity of the empirically elected transition probabilities of air pockets 315

List of Figures

2.1 Graphical representations of the VDSR, DRCN, DRNN and SRResNet deep learning SR architectures 35

2.2 Graphical representations of the EDSR, MDSR, SRDenseNet and MemNet deep learning SR architectures 39

2.3 MRF with 4-connectivity 54

2.4 Graphic representation of FCN-VGG-16 architecture 58

2.5 Graphic representation of the Faster R-CNN architecture 61

2.6 Graphic representation of U-Net architecture 66

2.7 Example of a dilated convolution 70

2.8 Graphic representation of PSPNet architecture 71

2.9 Graphic representation of DANet architecture 76

2.10 Graphic representation of 3D U-Net architecture 83

2.11 Graphical representation of the 3D DenseSeg and its principal components 87

2.12 Graphical representation of a mixed-scale dense network (MSD) with depth=3 and width=2 89

2.13 Most common beam configurations for X-ray computed tomography 91

2.14 Axial view of the 3D reconstruction of the Shepp Logan phantom 95

2.15 Tomography reconstruction by performing inverse radon transformation on the tomogram's sinograms 97

2.16 The formation of a sinogram 98

2.17 Reconstruction of time-series of tomograms 98

3.1 Diamond Light Source's I13-2 and I12 beamline schematics 103

3.2 Diamond Light Source facilities 104

3.3 A coloured render of the datasets used by the respective methods presented in this thesis 108
3.4 Images of projections of the synthetic and real-world datasets 110
3.5 Images of sinograms of the synthetic and real-world datasets 111

4.1 Graphical representation of the upscaling deep neural network (UDNN) . . 120
4.2 Diagram of the input received from the upscaling deep neural network (UDNN) 122
4.3 Diagram of the 4x upscaled sinogram by interlacing pixel-rows/projections for 3 different output channels 130
4.4 Diagram of the output produced from the upscaling deep neural network (UDNN) 133
4.5 Training and validation PSNR graphs of UDNN variations with different feature map sizes 139
4.6 Training and validation PSNR graphs of UDNN variations with different kernel sizes 141
4.7 Sinogram upscaling results of UDNN in comparison with the state-of-the-art using synthetic data 146
4.8 Closer Look on the sinogram upscaling results of UDNN in comparison with the state-of-the-art using synthetic data 146
4.9 The absolute value of the difference in sinogram intensity for the upscaling results of UDNN in comparison with the state-of-the-art using synthetic data 147
4.10 Reconstruction results of UDNN in comparison with the state-of-the-art using synthetic data 149
4.11 Closer Look on the reconstruction results of UDNN in comparison with the state-of-the-art using synthetic data 149
4.12 Sinogram upscaling results of UDNN in comparison with the state-of-the-art using real-world data 153
4.13 Closer Look on the sinogram upscaling results of UDNN in comparison with the state-of-the-art using real-world data 153

4.14 The absolute value of the difference in sinogram intensity for the upscaling results of UDNN in comparison with the state-of-the-art using real-world data . . . 154

4.15 Sinogram upscaling results with upscale factor 4x for UDNN and Cubic Interpolation using real-world data . . . 155

4.16 Closer Look on the sinogram upscaling results with upscale factor 4x for UDNN and Cubic Interpolation using real-world data . . . 155

4.17 The absolute value of the difference in sinogram intensity for the upscaling results with upscale factor 4x for UDNN and cubic interpolation using real-world data . . . 156

4.18 Reconstruction results of UDNN in comparison with the state-of-the-art using real-world data . . . 159

4.19 Closer look on the reconstruction results of UDNN in comparison with the state-of-the-art using real-world data . . . 159

4.20 Reconstruction results with multiplier x4 for UDNN-128 and cubic Interpolation using real-world data . . . 160

4.21 Closer look on the reconstruction results with multiplier x4 for UDNN-128 and Cubic Interpolation using real-world data . . . 160

4.22 Bar charts showing the progression of the PSNR and SSIM metrics as the upscaling factor increases . . . 161

4.23 Reconstruction results of UDNN applied on a undersampled tomogram that is part of a time-series . . . 165

4.24 Closer look on the reconstruction results of UDNN applied on a undersampled tomogram that is part of a time-series . . . 165

5.1 Graphical representation of the Dense Module and the Dense Block . . . 176

5.2 Graphical representation of the residual block and the transitional layer . . . 177

5.3 Graphical representation of mixed-scale dense network (MSD) layouts . . . 181

5.4 Validation mean IoU for 3D U-Net, 3D DenseSeg_64 and MSD_100 . . . 186

5.5 Graphical representation of DenseSeg and our proposed DenseUSeg architecture . . . 187
5.6 Graphical representation of our proposed Stacked-DenseUSeg . . . 189
5.7 Cross section of *RealHS1* used for the DenseUSeg and Stacked-DenseUSeg experiments . . . 195
5.8 Cropping of the tomogram reconstruction to training samples . . . 198
5.9 The learning rate schedule of Stacked-DenseUSeg, both for its denoising part and its segmentation part . . . 199
5.10 Validation PSNR and SSIM for the denoising experiments . . . 206
5.11 Denoising results using CGLS reconstruction as input and the FBP reconstruction as ground truth . . . 207
5.12 Validation IoU for the segmentation experiments using the FBP reconstruction . . . 210
5.13 Segmentation results using FBP reconstruction as input . . . 212
5.14 Validation IoU for the segmentation experiments using the CGLS reconstruction . . . 214
5.15 Segmentation results using CGLS reconstruction as input . . . 217

6.1 Cross section of our synthetic tomogram reconstructions and annotations used during the knowledge transfer experiments . . . 231
6.2 Cross section of our synthetic tomogram reconstructions and annotations used during the knowledge transfer experiments . . . 232
6.3 Zoomed version of Figure 6.2, depicted cross section of our synthetic tomogram reconstructions and annotations used during the knowledge transfer experiments . . . 233
6.4 Learning rate schedule of Stacked-DenseUSeg under different amounts of annotated data . . . 241
6.5 Validation PSNR of segmentation experiments using pretrained nets and 100%, 50%, 10% 5%, 1% or none of training of the available real-world training samples . . . 244

6.6 Validation SSIM of segmentation experiments using pretrained nets and 100%, 50%, 10% 5%, 1% or none of training of the available real-world training samples 245

6.7 Denoising results of Stacked-DenseUSeg with progressively less real-world training samples and using the pretrained net trained on synthetic data with rotating artifacts 249

6.8 Denoising results of Stacked-DenseUSeg with progressively less real-world training samples and using the pretrained net trained on synthetic data without rotating artifacts 250

6.9 Denoising results of Stacked-DenseUSeg with progressively less real-world training samples and without the use of a pretrained net 251

6.10 Validation mean IoU of segmentation experiments using pretrained nets and 100%, 50%, 10% 5%, 1% or 0% of training of the available real-world training samples 255

6.11 Segmentation results of Stacked-DenseUSeg with progressively less real-world training samples and using the pretrained net trained on synthetic data *with* rotating artifacts 257

6.12 Segmentation results of Stacked-DenseUSeg with progressively less real-world training samples and using the pretrained net trained on synthetic data *without* rotating artifacts 258

6.13 Segmentation results of Stacked-DenseUSeg with progressively less real-world training samples and without the use of a pretrained net 259

6.14 Flowchart presenting the steps in order to obtain a real-world segmentation solution using knowledge transfer 265

7.1 Hidden Markov model for the refinement of 3D segmentation predictions using temporal information (HMM-T) 277

7.2 Hidden Markov model for the refinement of 3D segmentation predictions using temporal information *and prediction probabilities* of the CNN segmentation network who produced them (HMM-TC) 279

7.3 Segmentation predictions of the *RealTS1* time-series of tomograms after the use of Stacked-DenseUSeg and the confidence of the network for said predictions 283

7.4 Segmentation predictions of the *RealTS1* time-series of tomograms after the use of Stacked-DenseUSeg and Stacked-DenseUSeg+HMM-TC 287

7.5 Segmentation predictions of the *RealTS1* time-series of tomograms after the use of Stacked-DenseUSeg, Stacked-DenseUSeg + HMM-T, Stacked-DenseUSeg + HMM-TC 292

7.6 Comparison of the segmentation predictions of certain toxels and their spatially neighboring toxels in 3 directions and over time 293

7.7 Behavioral test of the HMM-T for the prediction of the restoration of information using information from adjacent time-steps 295

7.8 Behavioral test of the HMM-T for the prediction of the Magnesium (class=2) turning into air pockets (class=3) through time. Figure displaying the 11^{th} tomogram predictions 296

7.9 Behavioral test of the HMM-T for the expected transition of the magnesium toxels (class=2) to air pocket toxels (class=3) through time 297

7.10 Behavioral test of the HMM-T for the unexpected transition of the air pocket toxels (class=3) to magnesium toxels (class=2) through time 298

7.11 Dynamic IoU results for each tomogram in the time-series using as ground truth the annotations from the representative highly-sampled tomogram of the time-series for the different classes and for Stacked-DenseUSeg, Stacked-DenseUSeg+HMM-T and Stacked-DenseUSeg+HMM-TC 301

7.12 Comparison of the segmentation 4D predictions of Stacked-DenseUSeg, Stacked-DenseUSeg + HMM-T, Stacked-DenseUSeg + HMM-TC against the annotations from the corresponding representative highly-sampled tomogram to the time-series 303

7.13 Dynamic IoU results for each tomogram in the time-series using as ground truth our manual 4D annotations for the different classes and for Stacked-DenseUSeg, Stacked-DenseUSeg+HMM-T and Stacked-DenseUSeg+HMM-TC 304
7.14 Comparison of the segmentation 4D predictions of Stacked-DenseUSeg, Stacked-DenseUSeg + HMM-T, Stacked-DenseUSeg + HMM-TC against our manual 4D annotations 306

Chapter 1

Introduction

Over the recent decades, multiple imaging approaches has been proposed in order to depict the hidden inner structure of many different biological and non-biological systems in the form of 3D volumes. X-ray computed tomography (CT) [1], Magnetic Resonance Imaging (MRI) [2, 3] and Electron Microscopy (EM) [4, 5] are some of the methods that have been used by researchers and industrial users, in order to collect 3D information. At the same time, due to multiple technological advances, these methods have become progressively more accessible, allowing in recent years the collection of time-resolved volumes, that form 4D datasets. With 4D imaging techniques, the hidden inner structure of a system is collected repeatedly through time, allowing the researcher to perceive not only its 3D structure but also how this structure is modified through time, during an experiment for example.

Nevertheless, despite the advance of technology in the area of data collection there is still a major need for accurate, efficient and easy to use methods for 3D and 4D data analysis. Namely, one of the most difficult tasks associated with almost all 3D or 4D studies is the semantic segmentation of the resulting datasets, which if done manually requires many days of human work, or even months in the case of 4D data. This process is often the starting point for many of the analysis methods which are then of scientific relevance, such as object counting, or connectivity analysis. Moreover, 4D studies may suffer from additional issues that affect the quality of data, due to the less than optimal collection of volumes within the limited time-span over which several temporal phenomena occur. In particular, for the collection of time-series of micro X-ray CTs, the earlier limited time-

span means the time allocated for the collection of each of the time-series' tomograms is small, resulting in a significantly reduced Signal-to-Noise Ratio (SNR) of tomograms' reconstructions. As it is further discussed in chapter 3, tomograms of such a time-series have to be collected with a necessarily limited number of sufficiently-exposed projections, making their accurate reconstruction difficult and reducing their SNR.

While in recent years computed tomography challenges grew, a new branch of computer vision, deep learning [6], has matured, and may be able to address these challenges. Over the last few years promising semantic segmentation approaches [7, 8, 9, 10, 11] have been proposed for volume segmentation, as well as denoising approaches [12, 13, 14, 15, 16, 17] for the improvement of visual quality in volumes. Deep learning approaches achieve this denoising or segmentation by learning an end-to-end mapping between input volumes and the annotations (for semantic segmentation) or high quality data (for denoising). This mapping can later be applied to automatically segment or denoise new volumes. The success of the typical deep learning approaches is highly correlated with amount of data used during training, as well as the quality of that data. For this reason, despite their great performance and accuracy, their adoption is slow in areas such as computed tomography. This is because collected data can vary greatly from study to study, resulting in a very small amount of annotated data, while also in some cases collection challenges may reduce the data quality.

The idea behind deep learning and machine learning in general, is that researchers do not have to precisely design an algorithm and set multiple of carefully predetermined operations and parameters in order to attain the desirable output. Instead, they must construct a network with an intelligent design, which can "learn" how to process input in a certain manner. Specifically, the deep learning network "learns" how to perform a task, by fine-tuning its large number of parameters using multiple valid pairs of inputs and outputs (a training set). Usually, the parameters are initially selected randomly and its first predictions have no similarity to the expected output.

Nevertheless, the error between the network's predictions and the expected outputs is calculated using a predefined loss function. Then using a stochastic gradient descent

approach the parameters are iteratively updated. After multiple iterations of the earlier *training* process, the estimated error decreases and it eventually converges to a local minima in the high-dimensional parameter space. It is at this moment, that the network can be tested on new and independent data, in order to evaluate its accuracy. Naturally, since deep learning networks are supervised with the use of multiple examples (pairs), additional care has to be applied to ensure the annotated data is a representative subset of the entire data that network is trained to process.

Furthermore, the networks' architecture has a pivotal role towards the eventual success of deep learning methods. However, regardless of the networks' final architecture, those that are designed to process images, volumes or other data that encapsulate spatial information are usually processed by Convolutional Neural Networks (CNNs) [18, 19]. These networks derive their name from their constituent convolutional layers, which are image (or volume) filters with trainable kernels. The principal behind these is that for each output imaging element (pixels, voxels, etc.) the same parameters are used (stored in the kernels), where the input is the imaging element from the input in the same location as the output imaging element and its neighbouring imaging elements in the input. This is different to traditional multi-perceptron layers [20], where each output value is calculated using all the input imaging elements and a set of different parameters for each output value. This locality offers, an obvious reduction in the number of parameters (since each output shares the same parameters using a different input each time), and also the property of translational invariance to the convolutional neural layers. This means that a convolutional neural network using patch of input imaging elements would generate the same output irregardless of where this patch is located in the overall input imaging system (image, volume, etc.). Based on this principal, deep approaches are able to offer outputs with the same size (in regards to the pixel or voxel resolution) as their inputs, allowing the to solve various Computer Vision problems, including denoising and semantic segmentation.

1.1 Motivation, aims and objectives

Inspired by the earlier mentioned challenges of time-resolved tomography and the potential advantages of deep learning approaches, this thesis will present novel approaches designed to denoise and semantically segment time-series of undersampled tomograms in a time-sensible manner given the large amount data associated with them. The main motivation behind the development of new approaches is the scarcity of methods designed to be applied on this data format (4D datasets comprised by low SNR volumes since the tomograms are undersampled), where at the same time the amount of annotated data is minimal. The existence of small amount of annotated data, is particularly true for studies in biology and material science which utilise time-series of undersampled tomograms to study different physical experiment and where only a few time-series are collected per physical experiment. Since the time-series of undersampled tomograms can differ significantly for different physical experiments, it is not possible to construct a diverse enough training set of data in order to develop approaches that are universally accurate on any time-series regardless of the physical experiment that they depict. Additionally, new physical experiments may be studied in the future, where approaches already trained on older data may not be accurate. Therefore, researchers have to create manual annotations for every time-series that depicts a new physical experiment. Due to the large time requirement in order create multiple manual annotations for every time-series that depicts different physical experiment, manual annotations are very scarce. Because of this, any new approach is necessary to work using potentially a small amount of annotated data and while also having fast training and inference times.

On the other hand, it is indeed useful to develop approaches that relay on training with annotated data, especially for the use-case of time-series of undersampled tomograms, since after the training period they will be able to automatically denoise or segment large quantities of data. Their advantage over approaches that do not use annotated data, is that they do not require an expert user in order to modify them for time-series regarding different physical experiments. Instead researchers who are expert in biology and material science, can perform the feasible for them task of manual annotation and

create multiple trained approaches that can be applied on time-series of different physical experiments. Based on these challenges, for the denoising and segmentation of time-series of undersampled tomograms it is advantageous to use methods that can be trained with small amount of annotated data, can easily be applied on new data without the need for additional changes in their design apart from retraining, and their training and inference times are relatively fast. Especially, the fast training and inference is important since there is a large amount of data that has to be processed (multiple large tomograms in each time-series) and any reduction in the time requirements may facilitate the further utilisation of time-series of tomograms in biology and material science studies. Thusly, the development of methods based on the above conditions, may potentially permit more research on dynamic systems in biology and material science, which is prohibiting at the moment due to the large time costs of manual annotation.

Motivated by this we aimed to develop novel approaches that:

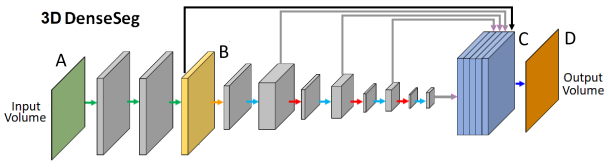
1. Increase the SNR of the reconstructions of time-series of undersampled tomograms.
2. Semantically segment the reconstructions of time-series of undersampled tomograms.
3. Offer time advantages either regarding the increase of the reconstructions' SNR or their semantic segmentation.
4. Be accurate using a small amount of annotated data, which is relevant for the task of semantic segmentation of the reconstructions of time-series of undersampled tomograms.

As mentioned in order to perform the above objectives, the proposed methods will utilise training on annotated data. An ally to this task is the existence of highly-sampled representative tomograms commonly captured before, after or either before and after the collection of each time-series. These tomograms and their respective high SNR reconstructions can provide sufficient training data for the development of approaches that can be used to increase the SNR of the reconstructions of the time-series' tomograms. Additionally, their high SNR reconstructions can allow the creation of spatially accurate semantic annotations that can be used in order to automatically segment the reconstruc-

tions of the time-series' tomograms. Based on this fact, our proposed approaches heavily relay on the utilisation of these highly-sampled tomograms. In the next Section we will provide an overview of this thesis and present ways with which our approaches perform the aforementioned objectives.

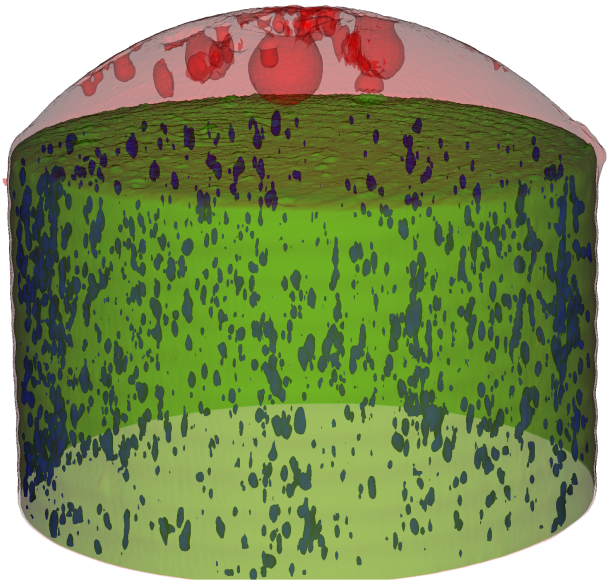
1.2 Thesis overview

Chapter 2: Literature review



Starting with a review of the literature, we present methods proposed over the recent decades for the tasks of super-resolution and semantic segmentation, with a special focus on deep learning approaches. Throughout this chapter, multiple approaches are presented and described along with how they influenced the development of our proposed approaches. Additionally, we describe their innate limitations regarding their application for the use-case of denoising and semantic segmentation of time-series of undersampled tomograms. Finally, we also present the methodology regarding the acquisition of X-ray CTs and how they are reconstructed.

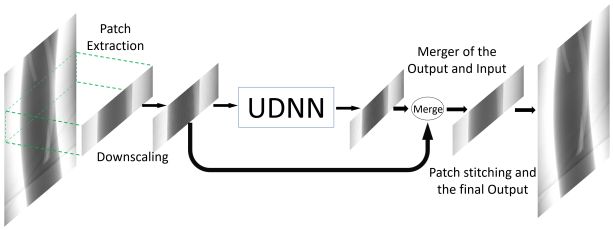
Chapter 3: Time-resolved undersampled X-ray CTs



In this chapter, we present the information regarding the time-resolved micro X-ray CT datasets used throughout this thesis. The collection of the tomograms' projections is described as well as their sinogram space. The data collection challenges of time-series are also described, with the largest of these being the projection undersampling due to temporal constraints. Additionally, we provide information regarding the synthetic tomograms created and used

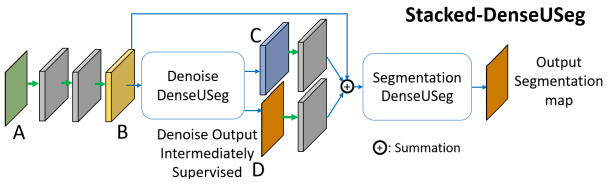
in this thesis.

Chapter 4: UDNN: A CNN network for fast upsampling of X-ray CT time-series



In this chapter we present our proposed UDNN deep learning approach for fast upsampling of undersampled tomograms. Our approach using representative highly-sampled tomograms is able to learn a mapping between low pixel resolution and high pixel resolution sinograms, essentially performing super-resolution. By upscaling the sinograms the tomograms' projections are upsampled, which leads to high SNR reconstructions. Our approach is test both on real-world and synthetic tomograms and against analytical interpolation approaches [21, 22]. Our objective for UDNN is to be faster for the use-case of time-series of undersampled tomograms than other state-of-the-art sinogram interpolation techniques and potentially more or equally accurate to them.

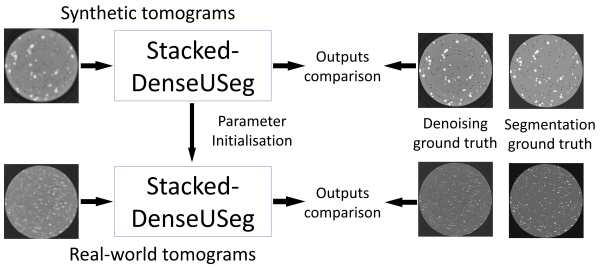
Chapter 5: Denoising and segmentation networks of low SNR reconstructions of undersampled tomograms



This chapter presents of our 2 proposed deep learning networks, DenseUSeg and Stacked-DenseUSeg for denoising and semantic segmentation of undersampled tomograms reconstructions present in time-series of tomograms. Similar to UDNN for training, a representative highly-sampled tomograms for the time-series is used, which can provides both high quality denoising and segmentation ground truths. For DenseUSeg our objective is to design a network which is more accurate than other 3D state-of-the-art segmentation networks. For Stacked-DenseUSeg, our goal is design a network that can offer an end-to-end fashion both a denoising and segmentation output for the component tomograms of a time-series. By performing this Stacked-DenseUSeg has a time advantage since both outputs are obtained simultaneously and the segmentation output can potentially be more accurate by internally utilising the

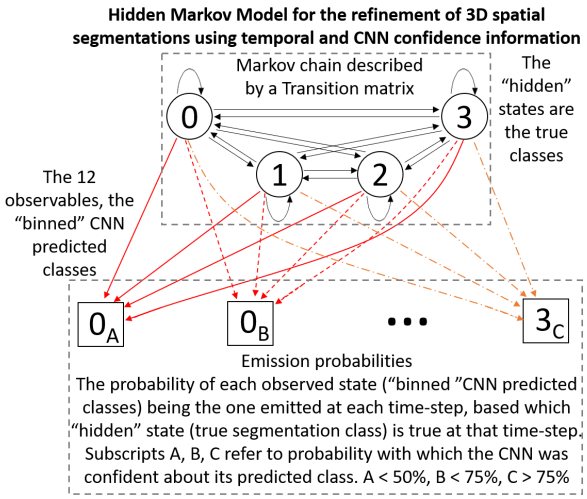
denoising output.

Chapter 6: Denoising and segmentation knowledge transfer via the use of synthetically generated tomograms



This chapter presents our proposed knowledge transfer scheme for Stacked-DenseUSeg using parameter initialisation with networks pre-trained on synthetic tomograms. For our knowledge transfer scheme, we use synthetic tomograms created by the TomoPhantom software [23]. Our knowledge transfer scheme's objective is to lead to the creation of more accurate semantic segmentation networks which also require less real-world manually annotated data. Furthermore, the need for less manually annotated data leads to time advantages since the user has to spend less time performing the time-consuming process of annotation.

Chapter 7: Temporal refinement of semantic segmentations in 4D datasets using hidden Markov models



This chapter presents two novel hidden Markov models, HMM-T and HMM-TC, for the refinement of semantic segmentations produced by our Stacked-DenseUSeg in individual component tomograms of time-series. The objective our hidden Markov models is to improve the segmentation accuracy of the overall time-series segmentations, by removing any temporal inconsistencies that can be attributed to misclassification by Stacked-DenseUSeg due to noise. Our models allow the utilisation of the temporal information in the time-series using an approach that does not require 4D manual annotations.

Chapter 8: Conclusions and future work

In the final chapter of this thesis we summarise the results and conclusions inferred by the previous chapters. Furthermore, we summarise the limitations associated with each our proposed approaches. Lastly, we propose future research ideas, for each of our proposed approaches and how they may be further be improved in the future.

1.3 Contributions

The contributions presented in this thesis are both publications and open source software. Below we present conference and journal publications and pending publications.

Chapter 4, will present our Upscale Deep Neural Network (UDNN) that is proposed in our publication with the title, "A convolutional neural network for fast upsampling of undersampled tomograms in X-ray CT time-series using a representative highly sampled tomogram" [24] in the *Journal of Synchrotron Radiation*. For our approach open source software [25] is available.

The main contribution of our UDNN approach is the SNR increase of the reconstructions in time-series of undersampled tomograms with accuracy comparable to other state-of-the-art sinogram interpolation techniques, with however faster execution times for the use-case of time-series of tomograms. Our approach is tested both on real-world and synthetic data. Due to its fastest execution times it may be preferable for use on time-series with numerous tomograms, since its most time demanding component is the UDNN's training which has to be performed only once before the inference-upscaling of all tomograms' sinograms in the time-series. Additionally, since our approach is trained using highly-sampled reference tomograms it is designed with the ability to handle temporal artifacts present in the sinograms of the tomograms (see Figure 3.5 Marker 1) that are associated with fast temporal events that occur during the capture of the undersampled tomograms.

Chapter 5, will present our DenseUSeg and Stacked-DenseUSeg network architectures. Our approaches were presented in the *Workshop on Visual Artificial Intelligence and Entrepreneurship (VAEI) in the British Machine Vision Conference 2019 (BMVC 2019)*

[26] and are also proposed in our publication with title "A Stacked Dense Denoising-Segmentation Network for undersampled tomograms and knowledge transfer using synthetic tomograms" in the *Journal of Machine Vision and Applications* [27]. For our approaches open source software [28] is available.

DenseUSeg's main contribution is that it can semantically segment the time-series' reconstructions with better accuracy than other state-of-the art 3D segmentation convolutional networks. Furthermore, our Stacked-DenseUSeg approach can offer additional segmentation accuracy. Stacked-DenseUSeg achieves this by creating a secondary output which is a denoised version of the time-series' 3D reconstructions. This secondary output is calculated in an end-to-end fashion along with the network's semantic segmentations and it may be used for visualisation purposes or potential further analysis.

Chapter 6, will present our knowledge transfer scheme proposed for our Stacked-DenseUSeg network. Our knowledge transfer scheme is also presented in our publication with title "A Stacked Dense Denoising-Segmentation Network for undersampled tomograms and knowledge transfer using synthetic tomograms" in the *Journal of Machine Vision and Applications* [27]. For our approaches open source software [28] is available.

Our knowledge transfer scheme main contribution is that it can potentially allow the fast training of our Stacked-DenseUSeg approach without sacrifices in the overall accuracy, using also less manually annotated data. Namely, this is achieved by training on synthetic tomograms and using the resulting nets for parameter initialisation of networks trained on real-world data. Furthermore, the reported accuracy is higher than the one reported with a Stacked-DenseUSeg network which is randomly initialised and trained with a fully annotated highly-sampled tomogram. The temporal advantage of our knowledge transfer scheme is because accurate Stacked-DenseUSeg networks trained on real-world data can be obtained with less annotated data, and thusly the user can spend less time performing the time-consuming task of manual annotation.

Chapter 7, will present our hidden Markov models HMM-T and HMM-TC, proposed in our publication with title "Temporal refinement of 3D CNN semantic segmentations on 4D time-series of undersampled tomograms using hidden Markov models" that is under

revision for *Nature's Scientific Reports* [29]. For our hidden Markov models open source software [28] is available.

The main contribution of our proposed hidden Markov models is that are designed to resolve temporal inconsistencies present in the segmentations of time-series' reconstructions, since those are obtained by trained Stacked-DenseUSeg that is applied on the time-series' reconstructions independently through time. Our hidden Markov models are designed in order to utilise the inter-tomogram information present the time-series, for the case of 3D segmentation networks applied on 4D (3D+time) data. Our models allow the removal of temporal inconsistencies due to misclassification by the network, but permitting physically viable temporal changes to manifest in the segmentations of the time-series's tomograms. By performing this operation, it allows for 3D segmentation networks to be used on 4D data, which is especially useful in cases where there is absence of annotated 4D data, which prohibits the development of 4D segmentation networks.

1.4 Summary

In this introductory chapter, chapter we described and elaborated on this thesis' motivations, aims and objectives. In summary, the segmentation of time-resolved X-ray tomography is associated with many challenges such as large quantities of data due to their high-dimensionality, undersampled tomograms with low SNR after their reconstruction and scarcity of semantic annotations. These challenges along with the scarcity of proposed methods for the denoising and semantic segmentation of X-ray CT time-series, motivated us to develop novel approaches towards this goal. The methods proposed in this thesis aims to denoise and segment the undersampled tomogram of X-ray CT time-series in a time-sensible manner given the large amount data associated with them. To address these challenges, our proposed approaches utilise many of recently proposed deep learning [6] techniques which have been introduced in order to resolve many problems of computer vision.

In particular, chapter 4, will present our proposed Upscale Deep Neural Network (UDNN), that is designed to upscale the sinograms of the constituent undersampled tomograms

in a time-series. This indirectly leads in the tomograms' upsampling in terms of their projections, and naturally in the high SNR of their reconstructions. As it was described earlier, our approach is faster than state-the-of-art sinogram interpolation techniques for the use-case of time-series with numerous tomograms. Additionally, it offers high upscaling accuracy that it is comparable to them.

In chapter 5, we present our DenseUSeg and Stacked-DenseUSeg network architectures aimed to denoise and semantically segment undersampled tomogram reconstructions in X-ray CT time-series. Namely, our proposed Stacked-DenseUSeg architecture is able to perform denoising and semantic segmentation in an end-to-end fashion, which allows it to achieve a high segmentation accuracy by utilising a denoised version of the input, which it also outputs.

Later in chapter 6, our proposed knowledge transfer scheme for Stacked-DenseUSeg is presented. The knowledge transfer scheme involves the utilisation of synthetic tomograms in order to create pretrained version of Stacked-DenseUSeg. Then using these pretrained versions for parameter initialisation, it allows the faster training/fine-tuning of our Stacked-DenseUSeg with real-world data without sacrifices in the overall obtained accuracy, using manually annotated data at the same time less.

Finally in chapter 7, we present our proposed hidden Markov models HMM-T and HMM-TC. Our hidden Markov models are designed to resolve temporal inconsistencies present in the segmentations of time-series' reconstructions since those are obtained by trained Stacked-DenseUSeg that are applied on the time-series' reconstructions independently through time.

In the next chapter, we will explore different approaches proposed for super-resolution, denoising and segmentation of images, volumes and other high-dimensional data.

Chapter 2

Literature review

Arguably the most common way in which people process information is by visualising it. By employing vision as our primary sense, the use of images and other visual elements is one of the most innate ways with which people understand the world. Imagery encompasses almost all aspects of our lives, ranging from art and entertainment and extending all the way to industry and science, where it is used to understand complex real-world systems. The use of images plays a pivotal role in multiple research fields. Ranging from medicine, pharmacology, chemistry, biology, physics, geology, material science, mechanical engineering and even sports, the collection and later processing of visual information is frequently a key component in the making of new discoveries.

These image processes steps on the abstract level are usually easy to convey even to people who are unfamiliar with the relative science, however there is an underlying difficulty in their execution. With tasks extending, from object recognition, classification and segmentation to image reconstruction, restoration and general image manipulation, while what their aim on the abstract level can be explained easily, i.e. find what pixels in an image depict an object (segmentation), or what an image displays (classification), it is their execution that becomes fairly complicated very rapidly. This can be attributed to the fact that humans usually neglect the significant role of their brain regarding their vision. As research indicates [30], human brains are responsible for the majority of the visual information we perceive, which is handled by a large number of neurons that equates to hardware with an extensive size of computational power.

For this reason, research and development of new image processing and analysis algorithms is complex, and an ongoing field of study. Over the last decades, improvements in technology and the discovery of novel learning-type approaches has allowed multiple previously impossible tasks to become part of our every-day lives. Face identification for security, augmented reality filters used in social media that intelligibly post-process human faces in real-time video (i.e. aging or de-aging filters), cancer detection in Magnetic Resonance Imaging [2, 3] datasets (MRIs), safer cars that operate semi-autonomously are some of the latest achievements that research on image processing algorithms has offered and it is expected that more will be possible in the future.

While plenty of new approaches have been emerged over the recent years, two visual processing tasks remain open for further improvement, super-resolution quality improvement, and semantic segmentation. The task of super-resolution involves the increase of the Signal-to-Noise Ratio (SNR) in imaging systems which is sometimes accompanied with an increase in the resolution of the imaging systems, in the words an increase in the number of imaging elements (i.e. pixels, voxels, toxels, etc.). Super-resolution is usually used in multimedia upscaling of films, video games, etc. and also numerous fields of study like biology, plant sciences, medicine and others, for the visual improvement of the collected data. Furthermore, the task of semantic segmentation is a very significant task in aforementioned research areas, where by segmenting interesting subregions in it is possible to obtain quantitative and qualitative information that can lead to new discoveries.

The following Sections, will present an overview of super-resolution and semantic segmentation over recent years, with emphasis on the latest deep learning approaches developed for both tasks. This is because the main goal of this thesis is to work towards improving the visual quality of the reconstructions of undersampled tomograms (more in Section 2.3) by increasing the resolution of their sinograms (method presented in chapter 4, more about sinograms in Section 2.3.2) and by increasing the SNR of their reconstructions (methods presented in chapters 5 and 6). Additionally, this thesis works towards the semantic segmentation of the earlier-described undersampled tomogram reconstructions (methods presented in chapters chapters 5 and 6), which are usually component tomograms of time-

series of tomograms.

The next Section 2.1 will provide an overview on super-resolution approached and in Section 2.2 an overview on Semantic Segmentation.

2.1 Super-resolution

The subject of super-resolution (SR), specifically the act of increasing the resolution (pixel count) of an image or its fidelity, has been explored by researchers over the last decades with the use a variety of techniques like Neighbour Embedding [31], Sparse Coding [32] and recently deep learning [33]. In general, SR algorithms can be divided into four categories.

Namely, there are algorithms that use:

1. Frequency-domain-based methods
2. Interpolation-based methods
3. Regularisation-based methods
4. Learning-based methods

The methods that belong in the first three categories typically, utilise multiple low-resolution (LR) (in pixel count) images in order to compose a single high-resolution (HR) output image. On the other hand, *learning*-based methods also named *example-based* methods, utilise a training dataset with multiple LR-HR pairs in order to learning a mapping between LR and HR images. Due to their ability, to train for the upscaling of images that have similar imaging characteristics (street images, medical images, images of natural, etc.) they are able to be exceptionally accurate and most state-of-the-art algorithms [33, 34, 35, 36] belong to this category.

2.1.1 Frequency-domain-based SR methods

Initially, introduced by Tsai and Huang [37] frequency-domain-based methods attempt to perform computations in the frequency domain of the LR images in order to increase HR outputs. In particular, Tsai and Huang [37] use the Discrete Fourier Transform (DFT), in order to obtain the aliased DFT coefficients of many noise-free LR input image

regarding the same scenery. Then using their proposed relationship between the aliased DFT coefficients and the Continuous Fourier Transform (CFT) coefficients of theoretical HR image of the same scenery, they are able to construct a HR output image transforming CTF coefficients back to the spatial domain. Later, this approach was extended by Kim *et al.* [38] for LR images that are blurry and contain noise. This is achieved by utilising a weighted least squares formulation. In case though, their approach assumes that the blurring effect in the LR images is the same and that they have similar noise characteristics. To account for this fact Kim and Su [39] use the Tikhonov regularisation method [40] for the solution of ill-posed problem created by considering each input LR created by a different blur effect. In order to further reduce memory requirements and computational times Rhee and Kang [41] proposed the use of Discrete Cosine Tranform (DCT) [42] instead of DFT. Furthermore, methods have been proposed that attempt to estimate finer-scale subband wavelet coefficients, resulting in a HR output image after the application of the inverse wavelet transform. in particular, Nguyen and Milanfar [43] proposed a wavelet-based interpolation approach that uses multiple LR images. Later El-Khamy *et al.*[44, 45] proposed a method that initially performs a registration of multiple LR images, restores their degradation, fuses their wavelet coefficients in order to produce a single image, which after interpolation generates the final HR output image. Ji and Fermüller [46, 47] proposed a more robust approach for wavelet transform in order to address errors during the registration and blur identification in the LR images. Lastly, Chappalli and Bose [48] investigate the optimal wavelet coefficient thresholding level in order to reduce the spatial domain noise present in wavelet-based SR approaches, but without severely increasing the blurring of the output HR image.

Frequency-domain-based methods have also been applied for the task of sinogram denoising (more regarding sinograms in Section 2.3.2). Sinograms in undersampled tomograms (undersampled in regards to their projections), have very few numbers of pixel-rows, since their number of pixel-rows is the same with the number of projections collected for respective tomogram. For this reason, sinograms of undersampled tomograms can be view as LR images that have low-resolution in regards to only one of their axis, since they

have a limited number of pixel-rows. During the collection of a CT scan, a sinogram is collected for every height position of the sample. By using all sinograms a reconstruction algorithm can create the 3D reconstruction of a sample and when sinograms with high number of pixel-rows are used, the final reconstruction has high SNR. Since only a single sinogram is collected for every height position of the sample, SR approaches that require multiple LR images regarding the same scenery can not be utilised in order to upscale the sinograms (increase their number of pixel-rows) and therefore upsample the number of projections of the respective tomogram. For this reason, the implementation of both the upscaling and denoising subtasks involved with SR can not easily be performed and therefore typically only sinogram denoising approaches are proposed. Jiao *et al.* [49] have proposed the use of stationary wavelet transform combined with a Bayesian estimation in order to reduce noise in undersampled tomograms. Additionally, approaches have been proposed [50, 51] that combine use of wavelet transform and MAP estimation in order to denoise sinograms of low-dose tomograms. However, as shown by Mehranian *et al.* [52], better denoising and/or de-artifact results can be obtained by working on the frequency-domain using a wavelet transform and also using a learning-based technique like sparse regularisation that utilises prior images. This is because a single LR sinogram as input can not offer sufficient information for the task of sinogram upscaling.

On the other hand, frequency-domain-based methods have been proposed to directly denoise and in general increase the SNR of undersampled tomogram reconstructions without applying sinogram upscaling. As it is explained in Section 2.3, reconstructions of undersampled tomogram have low SNR, as a result of the ill-posed reconstruction problem caused by the small number of projections. These reconstructions, typically have a large pixel-count and therefore usually the SR approaches applied to them attempt to increase the SNR of the reconstructions via denoising and/or the removal of imaging artifacts caused by the reconstruction algorithms when applied on undersampled tomograms. Chaudhari *et al.* [53] have proposed a frequency-domain-based denoising approach where different wavelets are used in order to decompose the reconstructions, denoise them in the frequency-domain and then perform the inverse wavelet transform. Borsdorf *et al.* [54]

have proposed to the creation of two reconstructions where one is created by odd numbered projections and another from even numbered projections. The resulting two low SNR reconstructions display the same spatial information, but they differ in respect to the noise. Then by detecting correlations in their wavelet coefficients the ones with small correlation can be suppressed and the ones with high correlation can be enhanced. The final reconstruction is created by averaging the weighted wavelet coefficients and performing the inverse wavelet transformation. Furthermore, Ghadrdan *et al.* [55] similar to Mehranian *et al.* [52] have proposed the combined use of wavelet transform and sparse regularisation in order to denoise the tomogram reconstructions directly.

There is a number of advantages regarding frequency-domain-based methods. Firstly, most of them are are theoretically simple and intuitive, working to introduce finer spatial details by predicting and introducing high-frequency information. Additionally, they can occasionally be easily parallelisable thus reducing hardware complexity. Nevertheless, frequency-domain-based methods when used exclusively are usually unfit for real-world applications, since they require the existence of global translation between the LR input images and to also have linear space invariant blur. In regards to the tasks of sinogram upscaling and CT reconstruction denoising, frequency-domain-based approaches have been proposed in conjunction with learning-based techniques like sparse regularisation. However, these approaches have considerably high training and inference times caused by performing the wavelet and inverse wavelet transformation for each input both during training and inference. Since we aim to develop fast approaches that can be implemented on multiple tomograms that are part of time-series, we focused more on creating fast and accurate state-of-the-art learning-based methods for sinogram upscaling and CT reconstruction denoising via the use of deep convolutional neural networks.

2.1.2 Interpolation-based SR methods

Interpolation-based methods use the most intuitive framework towards the estimation of a HR image through multiple LR images that refer to the same scenery. There are three steps typical present in interpolation-based methods. Firstly, the input LR images are

registered-aligned in order to negate any relative motion between them. Secondly, an interpolation step is performed that uses the pixel intensities of the LR images as data-points. This step then attempts via interpolation to estimate more densely spaced data-points that correspond to the pixels of the HR output. Finally, an image restoration step is performed in order to deblur and denoise the HR image made in the previous step. While the interpolation step can be performed using simple algorithms like nearest neighbour, bilinear interpolation, bicubic interpolation, etc. usually the increase of spatial detail that they offer is very small and a number of more sophisticated interpolation-based approaches has been proposed.

Ur and Gross [56] propose a nonuniform interpolation approach that uses multiple spatially shifted LR images and utilises Papoulis [57] and Brown [58] generalised multichannel sampling theorem. The advantage of their approach is that it is suitable for real-world applications due to its low computational cost, however the reconstructed HR images are not always certainly optimal since errors in the interpolation are not taken into account. Bose and Ahuja [59] employ the Moving Least Squares (MLS) method [60] in order to estimate pixel values for each pixel of the HR output image. For each pixel of the output, a different neighborhood is considered that contains the neighboring pixels of the registered LR input images. The estimation of each pixel value in the output HR image is performed via a polynomial approximation using the corresponding LR neighboring pixels. The order and the coefficients of the polynomial approximation is determined adaptively and it is different for each pixel of the HR output. Komatsu *et al.* [61] also proposed the use of the Landweber algorithm [62] regarding the reconstruction of the HR image from multiple LR images captured simultaneously by different cameras. As they point out, the use of multiple cameras that operate with the same aperture limits the possible camera configuration arrangements. For this reason, they later proposed the use [63] of different apertures per camera. The Landweber algorithm is also utilised by Shah and Zakhor [64] regarding the colour enchantment in videos. They also proposed the use of multiple candidate vectors for motion estimation of each pixel of the frames which uses both colour and luminance information in order to estimate them. Alam *et al.* [65]

proposed a method for SR and registration of infrared images in real-time. Registration is performed using a gradient-based algorithm and the interpolation is performed using a weighted nearest neighbour method.

Regarding the tasks for sinogram upscaling and CT reconstruction denoising, interpolation-based approaches have been proposed for the former but not for the latter. This is because interpolation-based focus on accurately increasing the pixel-count/resolution that results to high SNR outputs, but the denoising of CT reconstruction does not require any increase of the pixel-count/resolution. Köstler *et al.* [66] proposed an interpolation-based approach for sinogram upscaling by utilising optical flow [67] information along with nonlinear and anisotropic diffusion-based regularisers in order to accurately interpolate the sinusoidal traces present in the sinograms. Constantino and Ozanyan [68] propose a Hough transform [69] based approach in order to detect the sinusoidal traces present in the sinograms which aids to their accurate upscaling via interpolation. Bertram *et al.* [22] propose a directional interpolation technique where local orientation information is calculated in order to perform weighted interpolation of sinograms. Later Zhang *et al.* [70], further improved their approach in order to accurately interpolate sinusoidal traces with orientations that can not easily be detected based on a local neighborhood especially in very undersampled tomograms. Moreover, Li *et al.* [71] have also proposed a sinogram interpolation approach where for each interpolated pixel in the sinograms a local family of S-curves is estimated. Them combined with use of eigenvector-guided interpolation allows for the calculation of the output pixel intensities.

Overall, the benefits of the interpolation-based methods is that they are simple, fast and they can be easily be used for real-time applications. However, in general they are able to account for limited degradation effects, mainly for noise and blur which is similar amongst all input LR images. Furthermore, they may lead to non-optimal solutions, in cases where during the removal of degradation, the additional error introduced due to the interpolation step is not taken into account. Regarding the task of sinogram upscaling, interpolation methods are easy to implement and fast, but their accuracy is limited due to the presence of Poisson noise in the CT sinograms and other imaging artifacts (like

temporal artifacts, see Figure 3.5 Marker 1). For this reason and due to the existence of highly-sampled reference tomograms that can be used in order to train accurate and fast learning-based methods, our proposed approaches did not attempt to resolve the task of sinogram upscaling via an interpolation-based technique.

2.1.3 Regularisation-based SR methods

A different way to perform SR using multiple input LR images is to use a model that formulates the relationship between LR and HR images. Using the available input LR images, an ill-posed inverse problem can be created. The ill-posed status of the problem is due to the small number of available LR images and the unknown unblur operators that have to be applied to each LR image. Solutions to such ill-posed problems can be estimated using regularisation and so the respective SR methods are regularisation-based approaches. Regularisation-based methods typical attempt solve the inverse ill-posed problem either with a deterministic approach that utilises a priori knowledge regarding the relationship of the LR-HR images or with a stochastic/bayesian approach when the a posteriori probability density function (PDF) of the output HR image can be established based on the input LR images.

Regarding deterministic regularisation-based methods, Hong *et al.* [72, 73] proposed a multi-input stabilizing convex functional, which is created by the summation of multiple functionals for every LR input. Their multi-input stabilizing convex functional is used for the estimation of regularisation parameter, which regulates the tradeoff between the fidelity of the HR output image and its smoothness at each iteration step of their regularisation algorithm. Later Kang [74], proposed a multi-input functional again created by the summation of multiple functionals for every LR input which are though determined by within-input and cross-input information. Hardie *et al.* [75] in their approach incorporate knowledge regarding the optical system that is used in order to collect LR images and for estimation of the output HR image, a regularised cost function is minimised. For this task, they considered both the use of gradient descent and conjugate-gradient optimisation techniques. Bose *et al.* [76] also proposed the use of the L-curve method [77]

in order to systematically calculate the regularisation parameter and then a constrained least squares method is used to reconstruct the output HR image.

On the other hand, stochastic-bayesian methods have also been used to reconstruct the HR images, by estimating the probability density function of the output HR image using many input LR images. Tom and Katsaggelos [78] proposed a Maximum Likelihood (ML) estimation method that predicts the HR images using the Expectation Maximisation (EM) algorithm. Their approach estimates the HR image $\boldsymbol{X}^{ML}$ by maximising the conditional PDF $P(\boldsymbol{Y}|\boldsymbol{X})$ of the observed the input LR images $\boldsymbol{Y}$ based on the predicted HR image $\boldsymbol{X}$. Later, Hardie *et al.* [79] proposed a Maximum a Posteriori (MAP) estimation approach. Their approach determines the output HR image $\boldsymbol{X}$ so that the conditional PDF $P(\boldsymbol{X}|\boldsymbol{Y})$ is maximised. According to the Bayes rule $P(\boldsymbol{X}|\boldsymbol{Y}) = P(\boldsymbol{Y}|\boldsymbol{X})P(\boldsymbol{X})/P(\boldsymbol{Y})$. Therefore, the method has to estimate the $\boldsymbol{X}^{MAP}$ that maximises $P(\boldsymbol{X})$ based on a prior image model and the conditional PDF $P(\boldsymbol{Y}|\boldsymbol{X})$ similar to the ML estimation methods. In their approach the prior image model is formulated with a Gibbs distribution, which is equivalent to a Markov Random Field (MRF) [80] and the conditional PDF $P(\boldsymbol{Y}|\boldsymbol{X})$ is estimated using a gradient descent optimisation approach. Apart from a Gibbs distribution, other models have been proposed including, an edge preserving Huber-Markov Random Field (HMRF) [81, 82], a Markov Random Field with a discontinuity adaptive regulariser in order to be robust against motion and blur estimation errors [83] and a multi-input prior model [84]. Additionally, approaches have been proposed for the automatic selection of the regularisation parameters of the prior image model. Tipping and Bishop [85] used a scaled conjugate gradients algorithm in order to maximise a marginal likelihood function which also estimates the regularisation parameters of the prior image model. Pickup *et al.* [86], later further improved the prior image model both in terms of accuracy and by reducing its computational demands. Furthermore, their proposed prior image model is capable of handling illumination changes in addition to motion changes. He and Kondi [87] proposed an approach that estimates the regularisation parameters and also weight-coefficients that determine the level of fidelity in each of the input LR images. Woods *et al.* [88] also proposed both an ML and an MAP approach with an

efficient algorithm implementations in the frequency domain, thusly making a friendly application for large images. Finally, the use of a stochastic Markov chain Monte Carlo process [89, 90] has been proposed in order to estimate the statistics of the complicated implicated distributions (prior image model, conditional PDF of unknown HR images, etc.) using stochastic simulations.

Lastly, some alternative SR reconstruction methods have been proposed. Stark and Oskoui [91] were the first to proposed a Projection onto Convex Sets (POCS) [92] formulation of the SR problem, which was later extended by Tekalp *et al.* [93] in order to account for observation noise. According to the POCS formulation, based on priori knowledge and the input LR images for each of the pixels of the output HR image, a number of equations can be stated they restricts the HR pixel values-solutions within closed convex sets. If the intersection of these constraint sets is not empty, then the value-solution of each pixel of the HR image can be found by alternating projections onto these closed convex sets. Patti *et al.* [94] later proposed a more sophisticated POCS SR approach, able to take into consideration elements like the aperture time, the physical dimensions of the camera-sensor elements, focus related blurring, arbitrary sampling lattices and additional noise from the camera-sensor. Patti and Altunbasak [95] also by modifying the constraint sets we able to reduced ringing artifacts close to the edges in the output HR images. Additionally, another method that can be considered as a special-case of a POCS SR algorithm was introduced by Irani and Peleg [96] where they use an iterative backprobagation approach inspired by tomographic reconstruction in order to solve the SR problem.

Regularisation-based approaches have also been proposed for the task of sinogram upscaling or denoising and CT reconstruction. Namely, Karimi *et al.* [97] proposed an approach locally adaptive sinogram denoising approach that utilises maximum likelihood estimation in order to denoise projections for low-dose tomograms. Low-dose tomograms arise from the same CT collection problem like undersampled tomograms. In low-dose there is a need to reduce the overall exposure of the sample to X-rays and/or to collect the CT fast. For the case of low-dose tomograms, this performed by collecting many ill-exposed and therefore noisy projections quickly. On the other hand, undersampled tomograms collect

well-exposed projections, but not many of them as in low-dose tomograms. This means that the SR problem for low-dose tomograms can be performed either with sinograms denoising or denoising of the CT reconstructions. In comparison the SR problem for undersampled tomograms is regards either sinogram upscaling or CT reconstruction denoising. Additionally, regarding low-dose tomograms, Wang *et al.* [98, 99] have proposed a penalised weighted least-squares approach (PWLS) in order to either denoise sinograms or the CT reconstructions. Bayesian regularisation methods [50, 51] have already been mentioned in Section 2.1.1, where with the combined use of wavelet transform and MAP estimation sinograms of low-dose tomograms can be denoised. Furthermore, regarding to CT reconstructions, De Man *et al.* [100] has proposed an MRF based approach for reconstruction de-artification and Tabuchi *et al.* [101] developed an adaptive Wiener filter based on a Gaussian mixture distribution model for the denoising and de-artification of CT reconstructions.

Overall, the benefits of registration-based approaches is that they are able to model the SR problem with higher precision and complexity which leads to noteworthy SR results. Nevertheless, their complexity acts also as a limitation, since it increases their computational cost making them unsuitable for real-world applications or in cases where speed is essential. Due to this, the development of a registration-based method was not considered for the task of sinogram upscaling of undersampled tomograms. Since we are aiming to design an approach that would be applied on a time-series of multiple tomograms, the computational time of our approaches has to be fast in order to deal with the large amount of data within reasonable time (less or approximately within 24 hours). Additionally, the highly-sampled representative tomograms collected either at the beginning, end or both beginning and end of a CT time-series can be better utilised with the use of learning-based approaches that are described in the next Section.

2.1.4 Learning-based or example-based SR methods

Learning-based or example-based SR methods work by initially training in order to learn a mapping between LR images and their HR counterpart. Within this category a subset

of methods work internally using only information available by the input and utilizing the self-similarity property, they generate exemplar pairs of LR-HR patches. One such method introduced by Glasner *et al.* [102] effectively unites two methods of SR, the Classic Multi-Image SR [103] and the Example-based SR [31]. To adapt the Classic Multi-Image SR to work with single images, recurring patches within the same image are extracted and used together similar to how in Classic Multi-Image SR multiple LR images of the same imaged object are used together to create a HR image. Patches are extracted from lower resolution versions of the input image. For the Example-based part of their approach, using these lower resolution versions of the input image, exemplar pairs of LR-HR patches are extracted. If, for instance, a patch from a lower resolution of the input image is similar to a patch in the input image (of the original resolution) then a correlation has been found between a small region of the input image and a larger region of the input image. This larger region can then be used for the generation of the HR version of the small region as it would be displayed an HR version of the input image.

Nonetheless, most state-of-the-art Patch-based approaches work externally using a learned mapping between LR and HR images. One of the first proposed external Patch-based methods is from Freeman *et al.* [104]. For their approach the researchers used a database of LR-HR pairs of patches, with their low frequency components filtered out. The HR output images are constructed by reconstructing and upscaling individually, multiple patches of the LR input image. For the estimation of HR version of each input patch they proposed two different methods. The first uses a trained Markov Random Field (MRF) [105, 106] using the aforementioned LR-HR pairs of patches. The second uses stored feature vectors for each LR-HR pair of patches to find the nearest neighbour based on the input patch. In both cases, the HR patch with the high-frequency components is obtained. Then it is joined with the low-frequency components of an interpolated version of the LR patch. The final product is the HR patch. This operation is repeated for multiple input patches until the final HR image is obtained.

Later Chang *et al.* [31] introduced a neighbor embedding SR approach inspired by manifold learning methods, in particular the Locally Linear Embedding [107]. For their approach a

collection of low and HR patches is obtained by natural images (training images). These low and high resolution patches are represented by feature vectors, and form manifolds with similar local geometry in two different feature spaces that correspond to the low and high resolution. With the term "geometry" the authors refer to the ability of a patch (either low or high resolution) to be reconstructed by linearly combining its K neighbor's feature vectors. Since the "geometry" is assumed to be similar, the coefficients of this linear combination can then also be used with the feature vectors of corresponding HR training patches to reconstruct the HR output patch. This is possible if there is a sufficient amount of training low-high resolution pairs of patches. Then any patches can be reconstructed with a linear combination of the feature vectors of the K-nearest neighbors in the feature space, since they will be close to each other and the local neighborhoods (in both low and high resolution manifolds) would correspond to areas where the manifolds' geometry is linear. By repeating the earlier process for overlapping input patches and then combining the output patches by averaging on the overlapping areas the final HR image can be obtained.

Moving on, Yang *et al.* [32] proposed a sparse coding approach for SR. Their method used a large collection of image patches and a sparsity constraint to train together a low and a HR dictionary. These dictionaries represent LR patches and their respective HR components, using one sparse representation. Their algorithm then uses these dictionaries to search for a sparse representation for each input patch by solving the following optimisation problem:

$$\boldsymbol{\alpha}^* = \min_a \left\| \tilde{\boldsymbol{D}}\boldsymbol{\alpha} - \tilde{\boldsymbol{y}} \right\|_2^2 + \lambda \|\boldsymbol{a}\|_1 \tag{2.1}$$

Here $\tilde{\boldsymbol{D}}$ contains the low and HR dictionaries, $\tilde{\boldsymbol{y}}$ contains the input patch, $\boldsymbol{\alpha}$ is a weight matrix that works as a sparse selector of dictionary atoms and λ a weighting factor to balance the significance of the sparsity constraint. After that, each HR patch is estimated as $\boldsymbol{x} = \boldsymbol{D}_h\boldsymbol{\alpha}^*$ where $\boldsymbol{D}_h$ is the HR dictionary. The combined patches then synthesise an initial HR image $\boldsymbol{X}_0$, which however may not satisfy the reconstruction constraint $\boldsymbol{Y} = SH\boldsymbol{X}$. Here $\boldsymbol{Y}$ is the LR input image, H a matrix who represents a blurring filter and S a downsampling operator. The final HR image $\boldsymbol{X}^*$ is acquired through gradient

descent and is the closest image to $\boldsymbol{X}_0$, that satisfies the reconstruction constraint. This is achieved by solving the following optimisation problem:

$$\boldsymbol{X}^* = \arg\min_{\boldsymbol{X}} \|SH\boldsymbol{X} - \boldsymbol{Y}\|_2^2 + c\|\boldsymbol{X} - \boldsymbol{X}_0\|_2^2 \tag{2.2}$$

Here c is a weight factor that balances the requirement that the final HR image $\boldsymbol{X}^*$ should resemble the $\boldsymbol{X}_0$ image as much as possible, but at the same time satisfy the reconstruction constraint $\boldsymbol{Y} = SH\boldsymbol{X}$ as best as possible.

Inspired from the work of Yang *et al.* [32], Zeyde *et al.* presented a sparse coding approach [36], but they introduced several modifications. Firstly, numerical shortcuts, like interpolation of the LR image and dimensionality reduction via Principal Component Analysis (PCA) [108] are introduced, improving the algorithm efficiency and speed. Secondly, the training of the dictionaries is approached differently. They used K-SVD [109] for the training of the LR dictionary and for the HR dictionary they used a direct-approach using a Pseudo-Inverse expression [110]. Additionally, the ℓ^1-optimisation-based sparse coding algorithm is replaced with Orthogonal Matching Pursuit [111], which is considerably faster. Furthermore, redundant steps like the back-propagation in the post-processing stage are removed, simplifying the proposed approach. Finally, bootstrapping of the scale-up task is introduced allowing for the algorithm to operate without a prior training set.

Later Timofte *et al.* [34] introduced a SR approach using anchored neighborhood regression. Their approach based on the algorithm of Zeyde *et al.* [36], focused on achieving faster execution times, while retaining similar performance. This is achieved by reformulating the patch representation problem from equation 2.1 as a least squares regression regularised by the ℓ^2-norm of the coefficient matrix. Below is reformulated equation 2.1 :

$$\min_{\beta} \|\boldsymbol{N}_l\boldsymbol{\beta} - F\boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2 \tag{2.3}$$

Here F is a feature extraction operator, β similar to α in equation 2.1 and $\boldsymbol{N}_l$ is a local neighbourhood of the $\boldsymbol{D}_l$ LR dictionary. Respectively $\boldsymbol{N}_h$ is a local neighbourhood of the $\boldsymbol{D}_h$ HR dictionary. The advantage of working with ℓ^2-norm is that it transforms the optimisation problem to Ridge Regression [112] which has a closed solution. This means

that a projection matrix based on the considered neighborhood can be pre-calculated. Therefore, a LR input patch $\boldsymbol{y}$ can be projected to the HR space as:

$$\boldsymbol{x} = \boldsymbol{N}_h(\boldsymbol{N}_l^T\boldsymbol{N}_l + \boldsymbol{\lambda I})^{-1}\boldsymbol{N}_l^T F\boldsymbol{y} = \boldsymbol{P}_j\boldsymbol{y} \tag{2.4}$$

Here $\boldsymbol{P}_j$ is the stored projection matrix for the $\boldsymbol{N}_{l,h}$ neighborhood around the $\boldsymbol{d}_j$ dictionary atom of the joint $\boldsymbol{D}$ dictionary based on $\boldsymbol{D}_{l,h}$. For each of the dictionary atoms the $\boldsymbol{N}_{l,h}$ neighbourhoods are fixed (anchored). To obtain the high resolution patch $\boldsymbol{x_i}$ for each of the low resolution patches $\boldsymbol{y_j}$ of an input image, the nearest neighbour atom $\boldsymbol{d}_j$ has to be located in order to use its corresponding projection matrix $\boldsymbol{P}_j$. As a result, the whole process becomes a nearest neighbour search followed by a matrix multiplication for each of the input patches, making the approach considerably faster than Zeyde *et al.*'s [36].

Following that, Timofte *et al.* published an improved version [35] of their previous approach [34] using adjusted anchored neighborhood regression. In this the training samples are not used only for the obtaining the earlier $D_{l,h}$ dictionaries, but also for regression, since the new neighborhoods of each dictionary atom is not its neighboring atoms but its neighboring training samples. This is because dictionary atoms are generated in an attempt to sparsely sample the dictionary's feature space and its neighboring training samples, in terms of features, are closer than each of its neighboring dictionary atoms. The neighbourhood of atoms $\boldsymbol{N}_l$ in equation 2.3 gets replaced with a matrix $\boldsymbol{S}_l$ containing a number of training samples that lie closest to each dictionary atom.

$$\min_{\beta}\|\boldsymbol{S}_l\boldsymbol{\beta} - F\boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2 \tag{2.5}$$

This change allows the dictionaries atoms to sparsely sample the low and high resolution patches' feature space, resulting on fewer dictionary atoms being needed. This fact significantly contributed to the reduction of their method's execution time, while the accuracy of the method was still significantly better than the state-of-the-art.

Learning-based approaches that utilise sparse regularisation and dictionary learning have been proposed for the task of sinogram upscaling. As it is mentioned in Section 2.1.1, Meharanian *et al.* [52] have proposed a method that combines dictionary learning and

wavelet transform in order to upscale sinograms and therefore upsample a tomogram's projections. Namely, their method applies sparse regularisation for the estimation of missing projections in the wavelet domain, which is equivalent with sinogram upscaling. From their experiments they reported that their proposed approach suppresses streaking artifacts in the tomograms reconstructions and offers better results that linear interpolation of sinograms. Later, Zhang *et al.* [113] inspired by Wang *et al.* [98, 99] formulated a PWLS approach that also utilises dictionary learning for calculation of the PWLS cost function that has to be minimised. Their approach is able to offer noteworthy results for sinogram denoising in low-dose tomograms and increasing the SNR of their CT reconstructions. Additionally, Karimi and Ward [114] proposed the use of dictionary learning approach in order to denoise the stacked projections/sinograms of a tomogram. By stacking the projections of a tomogram a 3D volume can be form, which if processed in a 3D fashion it can allow the utilisation of information from spatially neighbouring pixels in the same projection and from neighboring projections. Namely, in their approach the stacked projections (that form a 3D volume) is partitioned in multiple overlapping 3D blocks which are then clustered. For each of the generated clusters a denoising dictionary is learned and after these dictionaries are training, they can used in order to denoise a tomogram's sinograms/projections. Apart from sinogram denoising and upscaling, learning-based approaches have also been used for the denoising of CT reconstructions. Bartuschat *et al.* [115] continuing the work of Borsdorf *et al.* [54] (as mentioned in Section 2.1.1), propose a method that uses two reconstructions of the same tomogram, one made by odd numbered projections and another made my even numbered projections. The two reconstructions depict the same sample, but the noise present in them is not correlated between them. By learning a dictionary in order to construct the first reconstruction with the second one as input, the resulting dictionary has effectively learn how to remove the noise from the reconstructions. By then creating a reconstruction from all projections and using the resulting dictionary, a denoised version of the reconstruction can be obtained. Later Yu *et al.* [116] proposed an approach for the denoising of CT reconstructions of low-dose tomograms using a single dictionary trained with K-SVD [109] and standard-dose CT reconstructions. Finally, as mentioned in Section 2.1.1, Ghadrdan *et al.* [55] have proposed

the combined use of wavelet transform and sparse regularisation in order to denoise the tomogram reconstructions more accurately.

The earlier mentioned learning-based approaches for image SR, sinogram upscaling and CT reconstruction denoising can preform their relative tasks with high accuracy by utilising HR exemplary images, sinograms or reconstructions. However as deep learning [6] became prominent, its potential for the development of fast and accurate learning-based SR approaches became apparent. Due to the large number of such proposed approaches of the recent years, the next Section will cover specifically learning-based SR approaches that utilise deep learning. Based on these findings, we decided to develop a deep learning based approach for the sinogram upscaling of undersampled tomograms in CT time-series and later a deep learning based approach for the denoising of CT reconstructions and their semantic segmentations.

2.1.5 Deep learning SR methods

One the first SR deep learning approaches was presented by Dong *et al.* [33, 117], who utilised a Convolutional Neural Network (CNN) [18, 19] in order to increase the resolution of "natural" coloured (RGB) images. In the proposed method, input patches are initially upscaled using bicubic interpolation [21] and then pass through a neural network, which outputs the desired HR patches. Due to its nature, a CNN is a very useful structure for SR as it can learn an end-to-end mapping between low and HR images, and also optimise simultaneously all the mapping layers during training. Despite that, different network architectures can lead to very different results. Because of that, the researchers experimented with different number of layers, feature maps and sizes of kernels. Their proposed model, SRCNN, is then compared with multiple state-of-the-art SR [31, 32, 34, 35, 118] algorithms. Based on the results SRCNN, achieves better performance across metrics such as PSNR and SSIM, even for higher scales of upscaling.

Our UDNN architecture [24] presented in chapter 4 is inspired by their method. We decided to base our approach's architecture on Dong *et al.*'s [33] (SRCNN) not only because of its great performance, but due to it is simplicity and fast execution and training times.

Dong *et al.*'s [33] approach, using only the LR-HR image pairs available, is able to learn an end-to-end mapping between them, without any prior assumptions as to how the low and HR images should relate. All of the aforementioned methods are designed with the assumption that LR images are produced from HR ones using an unspecified blurring filter followed by a downsampling operation, which is not the case for the downscaling of sinograms in undersampled tomograms (more about sinograms on Section 2.3.2). Therefore, we decided to base our network architecture upon SRCNN and modify it appropriately for the task of sinogram upscaling. This leads to projection upsampling of the undersampled tomograms and potentially a higher SNR in their reconstructions. In the rest of this Section, we will present more advanced deep learning SR approaches. These approaches are regarding heavyweight networks for the SR of *natural* coloured images and they are designed not to be frequently retrained. However, for the task of sinogram upscaling in time-series of undersampled tomograms, both fast training and inference times are important. Because of this, the lightweight SRCNN was considered as the architectural base for our UDNN network, and not the deeper SR network architectures, presented below.

In an attempt to further improve SR with convolutional neural networks, after the approach of Dong *et al.* [33] a number of novel deep learning approaches were proposed. Most of the approaches decided to address the error of pre-upscaling the input of the network with a bicubic interpolation when the downscaling operation is unknown. They also increase the complexity of their proposed networks (something necessary for SR of coloured *natural* images) and on occasion modify the standard architecture of sequential convolutional layers to something better tailored for the task of SR. Since our proposed approach [24] does not apply a pre-upscaling step using bicubic interpolation and does not require higher complexity due to the relatively easier task of sinogram upscaling compared to upscaling coloured *natural* images, later proposed approaches did not influence the development of our UDNN approach. Additionally, the element of fast training and inference times is important for our approach to be competitive against fast and analytical sinogram interpolation-based approaches. For this reason the more complex deep learning architectures which are presented in the following paragraphs were not considered during

the development of our UDNN approach.

One of the first of such approaches was proposed again from the Dong *et al.* [119], where they introduce deconvolutional layers [120, 121] in order to construct the output HR images. Thusly, they address the cases, when the downscaling operation is unknown. This downscaling operation is that one that creates the LRs inputs which should be upscaled. Before the introduction of deconvolutions, the inputs used to be upscaled with an interpolation technique and then processed by CNN networks to obtain the final HR output. This is because the typical convolutional layers present in CNNs can perform an upscaling of the input's resolution. Without the interpolation of the input before it is received by the networks, the resulting approaches are more computationally efficient, since interpolation may be a time consuming operation. Furthermore, with its absence, inaccurate estimations of the HR output are not introduced into the network, which is a positive characteristic since some reports has showed [122] that interpolation may create negative side-effect to the final results.

In an effort to eliminate the interpolation of the input and at the same time remove deconvolutions that are complicated and only slightly less time-demanding than interpolation, Shi *et al.* [123] proposed their network, ESPCN. In an approach which is similar to ours presented on chapter 4, HR information exits the network in multiple channels of the output which combined/interlaced using a predefined mapping, generates the final HR output. Their method is designed to upscale 2D *natural* images and the combination/interlace mapping of the different channels is performed in both dimensions, height and width.

Apart from approaches that aim to fix the error that input interpolation introduces, there are a number that attempt to deepen existing deep learning architectures for more accurate results. Naturally, this is sensible for SR applications where the nature of the input images remain roughly similar through time and the training network does not run the danger of becoming obsolete and requiring retraining. Because of that, deeper architectures which require large amounts of training time are not impractical since the resulting networks' lifespan would in comparison be much larger.

Such an approach, VDSR proposed by Kim *et al.* [124] and seen in Figure 2.1(**a**). For their approach they used a modified 20-layer VGG [125] network, which calculates the residual information between an upscaled version of the input with bicubic interpolation [21] and the desired upscaled output. Additionally, they also provided the network-interpolated version of the input with different upscaling rates. With the help of residual learning and the deeper architecture, their approach was able to outperform SRCNN [33].

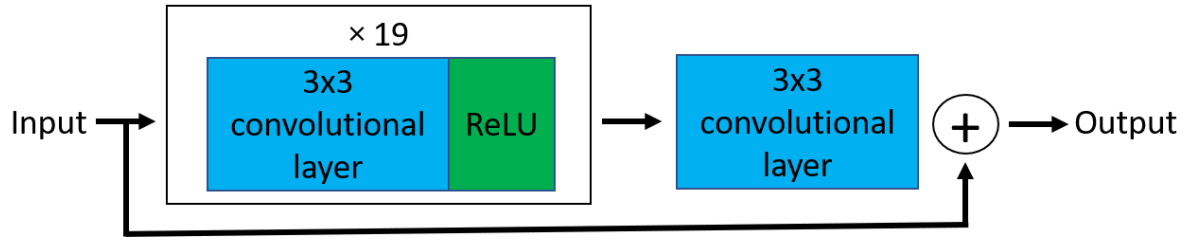
Attempting to further refine VDSR [124], Kim *et al.* later proposed DRCN [126] (see Figure 2.1(**b**)). Their new network utilises the fact that the kernels of the intermediate non-linear mapping convolutional layers tend to be very similar. For this reason, they proposed repeated use of convolutional layers with shared weights, therefore reducing the number of parameters. Additionally, they introduced deep supervision [127], since the repeated convolutional layers now produce 16 intermediate results and their weighted average provides the final result. Their approach offers similar results with VDSR [124], however with significantly fewer parameters. The advantage of their method is the reduction of the network's memory demands through the use of shared parameters, while keeping the SR accuracy high.

With the introduction of the ResNet [128, 129] architecture by He *et al.* (More in Section 2.2.3.4) to combat the problem of "degradation" that appears in deep networks when they start to converge, new SR approaches. This is because ResNet has proven to be a very effective architecture after examination [130, 131, 132]. SRResNet [133] (see Figure 2.1(**d**)) proposed by Ledig *et al.* modified the ResNet architecture for the task of SR, introducing residual modules and batch normalisation layers [134], managing to outperform earlier approaches. Additionally, they proposed an adversarial network that discriminates between genuine HR images and upscaled ones generated by SRResNet. Their presented generative adversarial approach, inspired by Goodfellow *et al.* [135], allows them to achieve more photo-realistic results according to human perceptual criteria and they used Mean-Opinion scores [136] to verify that. Similarly, influenced by He *et al.*, Yang *et al.* proposed the DEGREE architecture [137], where the residual blocks learn to predict high-frequency details and edge information. Then using these information like in [138], HR outputs can

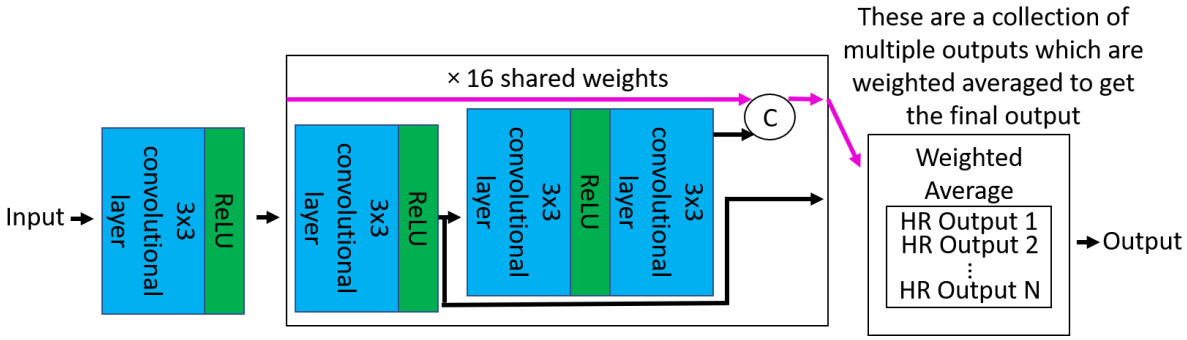
be generated. By focusing the learning on high-frequency details, the predictive strength of their proposed network is better directed for the recovery of the missing high-frequency information in the input LR images. Overall, the benefit of these methods is that by training convolutional layers which learn residual features, they are able to better utilise their depth and number of parameters, leading to higher SR accuracy.

Similar to DRCN [126], Tai *et al.* introduced DRRN [139] in an effort to further reduce the parameters in residual architectures for SR (see Figure 2.1(**c**)). Their proposed architecture using multiple residual blocks with shared weights is able to outperform earlier methods, while keeping, at the same time, the number of parameters low. Another architecture that also further improves the implementation of ResNets for SR is EDSR proposed by Lim *et al.* [140] (see Figure 2.2(**a**)). EDSR has two distinct changes, the first is the layer number increase and the increase of feature maps for each layer, and the second is the abolishment of the batch normalisation. The advantage of the first change is obvious, since deeper architectures have the potential to be more accurate; however, the second change requires further explanation. In cases like classification and segmentation, where batch normalisation is traditionally used, the inner representations of the input imprinted on the feature maps are highly abstract and may be insensitive to changes caused by batch normalisation. However for the task SR, the input and the output are highly correlated images and the operation of batch normalisation may be harmful. This is because when the network learns low-level features, batch normalisation may be obstructive, resulting in the pursuit of more abstract solutions, which are not necessarily linked to better SR results.

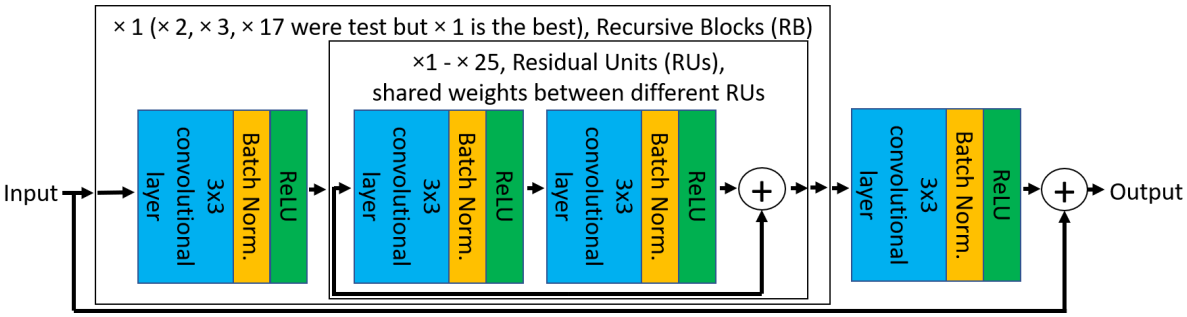
Furthermore, in effort to also utilise the kernel similarity in intermediate convolutional layers [126, 139], Lim *et al.* also proposed MDSR [140] (see Figure 2.2(**b**)) which produces multiple outputs with different upscale rates while using the same intermediate layers. Similar to that approach, Lai *et al.* proposed LapSRN [141], which is a deeply supervised [127] network comprised of consecutive normal and transposed convolutions, producing multiple outputs that represent progressively more upscaled versions of the input. Their architecture, inspired by Laplacian pyramids [142], uses two branches. The first calculates



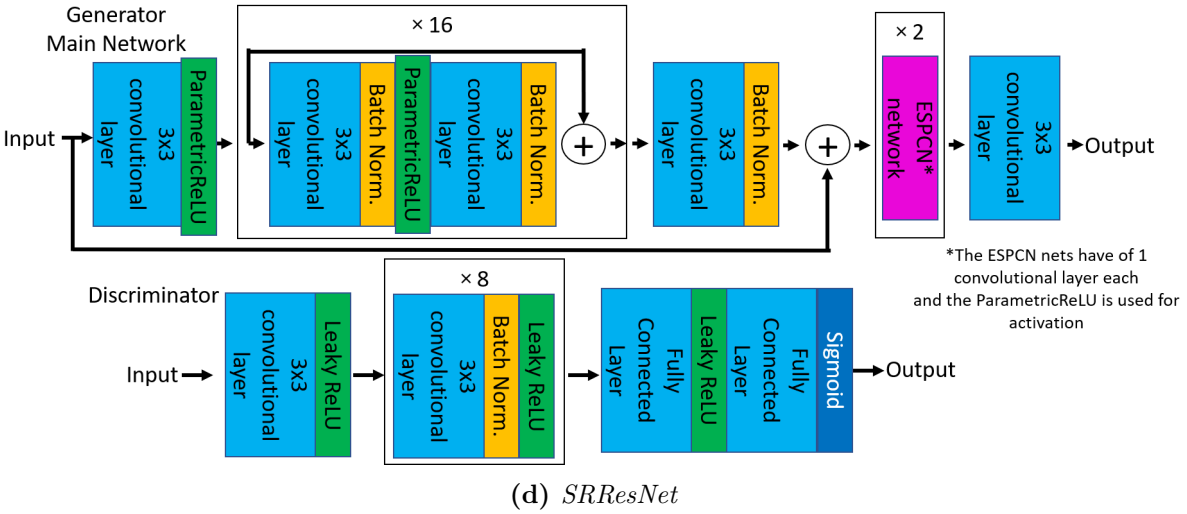
(a) *VDSR*



(b) *DRCN*



(c) *DRNN*



(d) *SRResNet*

Figure 2.1: *Graphical representations of the VDSR, DRCN, DRNN and SRResNet deep learning SR architectures. (**a**) the VDSR [124] approach learns residual information between the HR output and upscaled input version of the input (bicubic interpolation) using a sequence of convolutional layers. (**b**) DRCN [126] uses convolutional layers with shared weights, therefore reducing the number of parameters. From each convolutional layer intermediate outputs are created and their weighted average offers the final output, (**c**) DRNN [139] utilises multiple residual blocks with shared weights to achieve a higher SR accuracy, (**d**) the SRResNet [133] is a GAN network that improves the generator subnetwork (which performs the SR task) by using a discriminator that detects if its input is an upscaled image from the generator or a genuine HR image.*

the high-frequency details, while the second recursively upscales the input with transposed convolutions and adds the high frequency information in order to obtain the various multi-scale outputs. As also mentioned earlier for the DEGREE architecture [137], by focusing the predictive strength of their network on the recovery of the missing high-frequency information in the input LR images, high SR accuracy can be achieved with the network parameters used more efficiently.

Moving on with the proposal of Huang *et al.*'s DenseNet [143] the problem of "degradation" (see Section 2.2.3.4) is further reduced compared to ResNet, by replacing the operation of summation with concatenation and proposing the use of densely connected convolutions. It have been observed that while ResNet excels at feature re-usability, DenseNets are great for discovering new features from the data [144]. This fact lead to new proposals for SR, one of them named SRDenseNet by Tong *et al.* [145]. As can be seen in Figure 2.2(**c**), the authors combined multiple densely connected blocks, with a bottleneck layer and transposed convolution in order to elevate the networks performance. Taking inspiration by EDSR [140], they abstained from using batch normalisation operations, typically found on DenseNet-like architectures. By replacing the operations of summation found in ResNets with concatenations, the subsequent bottleneck layers (typically 1×1 convolutional layers) essentially perform a weighted summation of the different features using their parameters as weights. This additional degree of freedom, has been proven beneficial and as we will further explain in Section 2.2.4, lead us to the proposal of our DenseUSeg [26, 27] which is used to denoise CT reconstructions using densely convolutional layers.

Furthermore, Tai *et al.* [146] later proposed their MemNet architecture, which borrows elements from ResNets. According to the authors, the connections within MemBlocks (see Figure 2.2(**d**)) can be viewed as network's short memory and the dense skip connections as its long memory. Using the high capabilities of DenseNet, Haris *et al.* proposed their deep back-projection network DBPN [147], which densely connects parts of the network that iteratively upscale and downscale feature maps. Their approach with the use of error feedbacks by the upscaling and downscaling steps, is able to achieve better results even

for high upscaling factors.

Later Zhang *et al.* [148] proposed a similar architecture named RDN, where they addressed the input interpolation present in MemNet [146]. They also proposed the RCAN architecture [149] where they employ channel attention [150] and nested residual modules to improve performance. Finally, Jang *et al.* [151] reaffirmed Tai *et al.* [146]'s findings, and by also introducing channel attention like Zhang *et al.*, their method DRCA was able to achieve better performance than the state-of-the-art methods with fewer parameters.

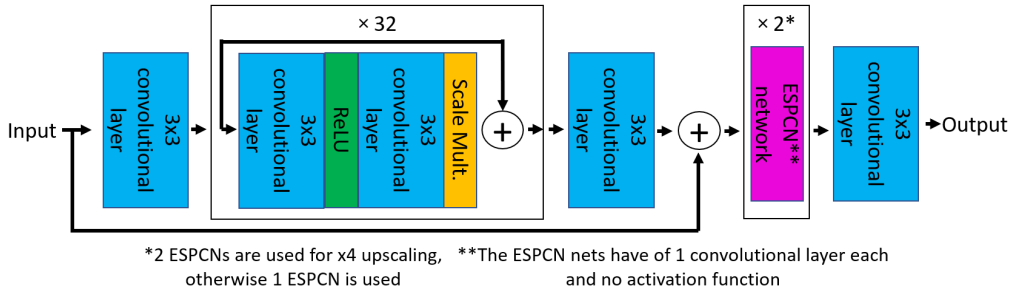
Overall, the networks mentioned in this Section though the utilisation of residual feature learning, densely connected convolutional layers, channel attention and in general deeper network architectures are able to achieve state-of-the-art SR accuracy of coloured *natural* images. Their main advantage is the high SR accuracy that they report, which is has lead to the increase of their popularity over the recent years. Nevertheless, there are a number of disadvantages associated with these approaches. One these drawbacks, is that the aforementioned networks relay on large amounts of training data, which may are not present in some use-cases. Furthermore they have high computational demands, which may lead to slow training and potentially inference times. Typically, for applications developed for the SR of images with similar *nature* through time, for instance the SR of images with human faces, SR of street scene images, SR of images take in indoor environments, etc., longer training times are not a disadvantage because the network has to be trained only once. However, for user-cases where a SR network has to be frequently retrained for new datasets, longer training times can be disadvantageous. In particular, for the task of sinogram upscale of undersampled tomograms in time-series, any network based approach would have to be frequently retrained for new time-series. This is because CT time-series used in biology and material science research may depict very different physical samples and a network trained to upscale the sinograms of the tomograms in one time-series may not be accurate in another. Additionally, due to the large number of sinograms that have to be upscaled for every time-series of tomograms, faster training and inferences times are desirable. For these reasons, we decided to base our UDNN [24] architecture upon the simpler SRCNN [33] architecture, allowing our proposed

approach to have fast training and inference times, while also utilising the highly-sampled representative tomograms that accompany each time-series in order to train our approach. In the next Section we will mention other deep learning SR approaches proposed for the task of sinogram upscaling or CT reconstruction denoising.

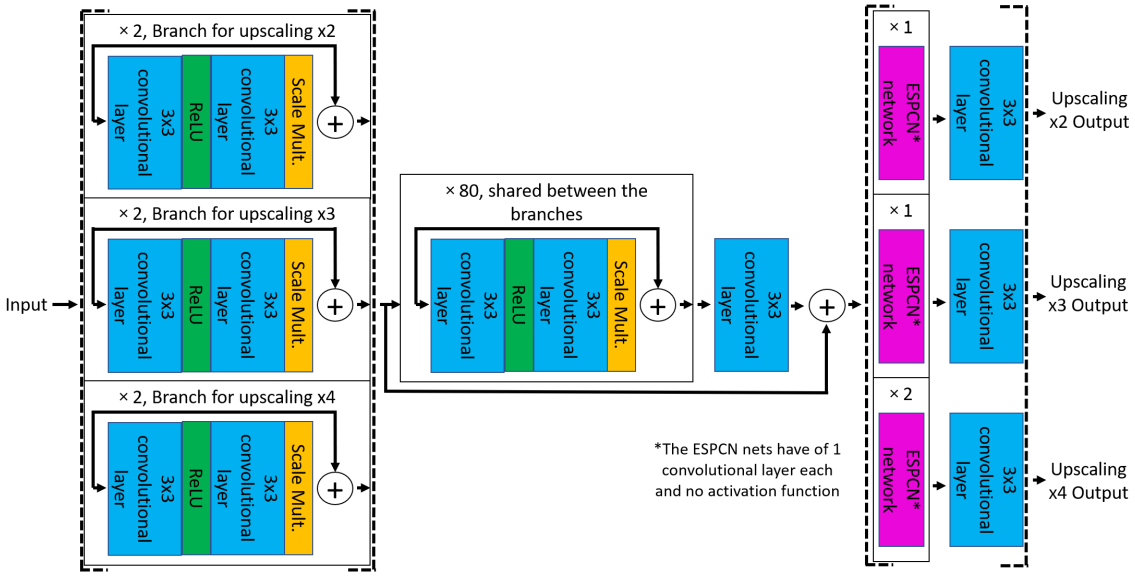
2.1.5.1 Deep learning SR methods for undersampled tomogram reconstructions

Naturally, with the introduction of deep learning SR techniques for *natural* images, deep learning approaches has also been proposed for the task of sinogram upscaling. Specifically, Dong *et al.* [152] and Lee *et al.* [153] proposed two U-Net-influenced sinogram interpolation-inpainting approaches. Their approaches aim to upscale a tomogram's sinograms in order to reduce noise and artifacts in the final tomogram reconstruction. Their methods are designed for medical tomograms that are undersampled in an effort to reduce X-ray dose received by the patient (low-dose CTs). These tomograms are common in practice, since they are preformed on many patients and on specific regions-organs. This means that after a successful training session, the resulting deep neural networks via sinogram upscaling highly improve the SNR of the tomogram reconstructions, that depict specific regions-organs. This results in the very deep trained networks that have a long lifespan without needing retraining. Therefore, the focus of these networks is to be very accurate by increasing their number of parameters and also the time required to train them. Because though it is expected to be trained once their large training times is not considered as a drawback.

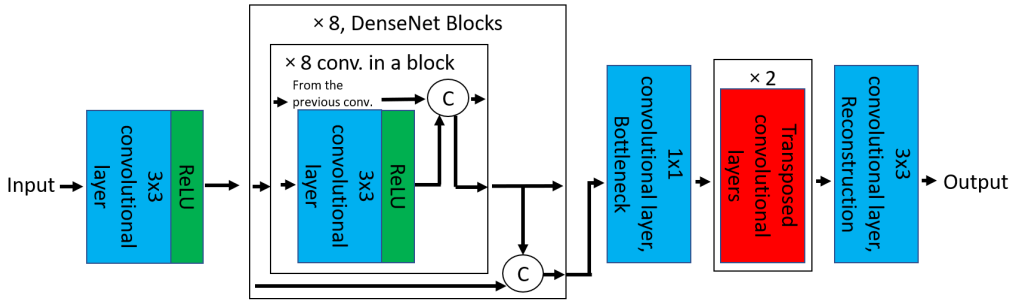
Opposite to that, undersampled tomograms produced by biologists and material scientists can be more diverse in nature, which means there can not be trained universal networks which can process all existing undersampled tomograms. Any proposed deep learning architecture would probably have to be retrained for new varieties of undersampled tomograms, making efficient training time requirements essential. Based on these conditions and inspired by Dong *et al.*'s approach [33], we proposed the UDNN [24] architecture in order to utilise the good performance of the deep learning approaches, while also keeping as low as possible the time requirements for training.



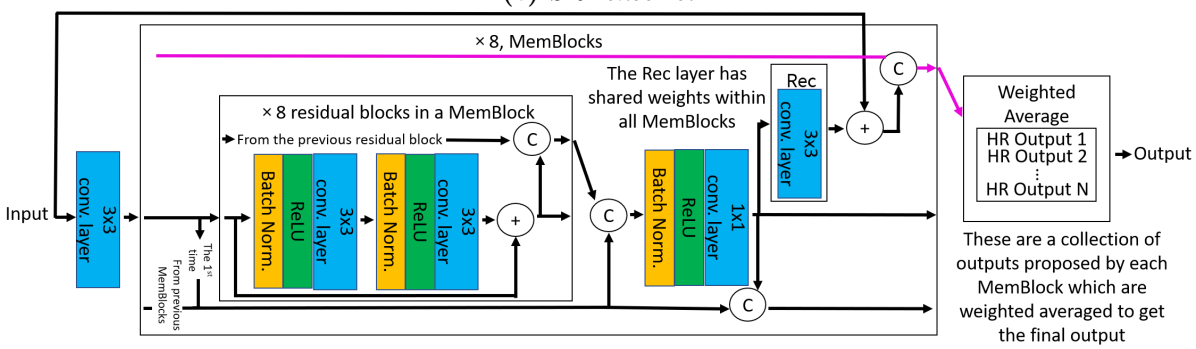
(a) *EDSR*



(b) *MDSR*



(c) *SRDenseNet*



(d) *MemNet*

Figure 2.2: *Graphical representations of the EDSR, MDSR, SRDenseNet and MemNet deep learning SR architectures. (**a**) the EDSR [140] architecture improves upon the SRResNet [133] architecture, by increasing the number of layers, increasing the feature maps that each layer creates and also removing the batch normalisation layers, (**b**) the MDSR [140] expands upon the EDSR by produces multiple outputs with different upscale rates, while using the same intermediate layers, (**c**) the SRDenseNet [145] proposes the use of densely connected layers instead of residual layers for higher SR accuracy, (**d**) MemNet [146] utilises both residual layers and densely connected layers in order to achieve more accurate SR results.*

Apart from deep learning SR algorithms that attempt to improve the SNR of the tomogram reconstruction via sinogram upscaling, others have been proposed [147, 154, 155] which directly process the tomogram reconstructions. Chen *et al.* [13] proposed a simple three layer approach, similar to Dong *et al.*'s [33] SRCNN for denoising reconstructions of undersampled tomograms, demonstrating that deep neural networks can successfully be implemented for improving CT reconstructions. Furthermore, Kang *et al.* [15, 16] proposed a more specialised architecture that process the wavelet transform coefficients of slices from reconstructions of undersampled tomograms. Their approach, utilising residual learning [128, 129] and an encoder-decoder scheme similar to U-Net [156], is able to efficiently target specific noise and artifacts present in reconstructions of undersampled tomograms.

Following this, Chen *et al.* [14] proposed a similar encoder-decoder architecture that was deeper (in terms of layers) and also utilised multiple skip connections. Their approach was able to outperform earlier proposed denoising approaches for tomogram reconstructions [13, 15, 157, 158, 159], setting a new standard for undersampled tomogram denoising. Roughly at the same time, in an effort to exploit non-locality for noise and artifact reduction, Lui *et al.* , inspired by approaches using stacked denoising autoencoders [160, 161, 162] and recent applications of them on undersampled tomogram reconstructions [163], proposed their Stacked Sparse Denoising Autoencoder (SSDA) method [12], which is also more accurate than the established denoising approaches [15, 157, 158, 159]. Finally, some of the recent proposals [17, 164] utilise generative adversarial networks to increase the denoising precision in applications, where spatial accuracy is crucial.

The methods mentioned above provide evidence that via the use of deep learning, accurate approaches can be developed regarding the task of denoising CT reconstruction. Additionally, they achieve noteworthy accuracy by adopting architectures initially proposed for the tasks of classification and segmentation (e.g. [14, 15, 16] preformed well by adopting the U-Net architecture [156]). Nevertheless, the adopted CNN architectures are not very contemporary, it is expected that higher accuracy can be obtained by adopting more recent deep learning segmentation architectures [8, 9, 10] which have yet to be tested for the

denoising of CT reconstructions. Inspired by this, we recently proposed a denoising CT reconstruction architecture [26, 27] that uses densely connected layers [10] within an encoder-decoder architecture [8]. Our proposed approach is able to accurately both denoise and semantically segment CT reconstructions of undersampled tomograms. Since our approach achieves high accuracy by utilising recent semantic segmentation architectures, we decided to compare it against the latest state-of-the-art 3D semantic segmentation networks. In the next section, we will explore approaches proposed for the task semantic segmentation and emphasising on state-of-the-art deep learning semantic segmentation methods.

2.2 Semantic segmentation

Since 1980's the problem of image segmentation has been approached through a plethora of different techniques. These methods can be categorised using many criteria, but in general they can be categorised to approaches that detect the discontinuities that form boundaries between the different segmentation instances and approaches that detect the continuities and similarities within different regions in the input, forming thusly the segmentation instances.

2.2.1 Segmentation methods based on region discontinuities and boundaries

One category of methods that performs image (or volume) segmentation achieves this by detecting discontinuities between different regions in the input, forming thusly the different segmentation instances. These methods essentially work by detecting the boundaries of the different segmentation instances. In certain use-cases, there are not strong imaging discontinuities within the semantic objects (e.g. in lungs, thorax and abdomen in medical CT images) that have to be segmented and therefore such boundary-based approaches can be used for semantic segmentation. In general, approaches based on discontinuities can be further split to two different subcategories. Firstly, regarding approaches that detect intensity edges between the different segmentation instances and secondly, approaches

that define the contours between the different segmentation instances.

2.2.1.1 Edge detection segmentation methods

Some of first and simplest techniques proposed for the task of image segmentation, revolve around the detection of intensity edges that form the boundaries of the regions, that have to be segmented. This is usually performed by either detecting the areas where first order derivatives of the input image reach their maximum/minimum or areas where second order derivatives of the image are zero. The above areas, can be detected using different operators such as Prewitt operator [165], Sobel operator [166], Canny operator [167], Laplacian/Laplacian of Gaussian operator, Difference of Gaussians, Roberts cross operator [168] or Kirsch compass kernel [169]. After the detection of the edges, these can be connected together to form the overall boundaries around the segmentation instances present in a given input. In Al-amri *et al.* [170], many of the above operators are utilised for the segmentation of satellite images. A common way to merge the detected edges and thusly perform segmentation is via the Hough transformation [69]. In Hough transformation through a voting process, lines or line segments can be found from the unmerged edges which can then guide their merger. Saha *et al.* [171] utilise Hough transformation in order to achieve text segmentation in images.

The advantages of edge detection segmentation methods is that they are fast and very easy to implement. However, they relay on the existence of high contrast in the input images/volumes and with small levels of noise. Furthermore, they are not inherently semantic. This means that areas of the same semantic object may be segmented as separate segmentation instances. Moreover, they require manual adjustment when they are to be applied on new data. For these reasons, they were not considered during the development of our approaches, since one of the main problems of with reconstructions of undersampled tomograms is the presence of low SNR.

2.2.1.2 Contour-based segmentation methods

Another way to detect the discontinuities in a given input image/volume is to attempt to define a contour around the different segmentation instances. This contour is selected in a way that simultaneously adheres to the intensity edges around the different segmentation instances while also being relatively smooth. One of the approaches that have been proposed to perform this task are active contour models [172]. Active contour models, employing a deformable spline which is defined by minimizing an energy function that compels the contour to adhere as accurately as possible to the intensity edges around the segmentation instances and at the same time be sufficiently smooth. Additionally, level set [173] approaches have been proposed in order to define contours around the segmentation instances. Level sets achieve this by defining a 3D surface where for $z = 0$ it offers the contours around the segmentation instances. Similarly, the 3D surface is determined by minimising an energy function.

Active contour models relay on intensity gradients for their accuracy, however in certain input images there is great gradient variation. To combat this problem Caselles *et al.* [174] proposed the combination of active contour model with minimal distance curves or geodesics, resulting in their proposed "geodesic active contours". In Atkins and Mackiewich work [175], active contour models are used for the accurate segmentation of brain MRIs. Later Loizou *et al.* [176] proposed an active contour segmentation approach for the segmentation of ultrasonic images of the common carotid artery. Regarding level set methods, Li *et al.* [177] proposed a level set method without periodic re-initialisation of level set function, which is typically performed in traditional level set methods [173]. In attempt to further improve the accuracy Rousson *et al.* [178] utilised shape priors to constrain the level set representations. Later Li *et al.* [179] proposed the use of distance regularised level sets, that do not require re-initialisation of level set function, and they can operate with large time-steps in order converge faster on the desired contour by being more computationally efficient. Level set have also been proposed for segmentation medical images, specifically MRI images [180, 181].

The benefits of contour-based segmentation methods is that they can be used on noisy

inputs and that the output contours are sufficiently smooth, which is true in many segmentation problems (brain segmentation, lung segmentation, etc.). However, their implementation on large quantities of data is slow compared to learning-based segmentation methods (see Section 2.2.2.4). Additionally, they are not inherently semantic, meaning that areas of the same semantic object may be segmented as separate segmentation instances. Based on these facts, these methods were not considered during the development of our approaches for segmentation of undersampled tomogram reconstructions, since we aim to develop automatic semantic approaches that will be applied on multiple undersampled tomograms of a time-series.

2.2.2 Segmentation methods based on region similarities

A second category of segmentation methods attempts to detect continuities and similarities between different regions of the input that form the different segmentation instances. In general, these can be split to methods that use thresholding, methods that use growing, splitting or merging regions, methods that use clustering and learning-based methods.

2.2.2.1 Segmentation methods based on thresholding

One of the oldest and simplest methods for performing image segmentation is by grouping pixels that have pixel intensity within a certain range, which is the operation of thresholding. Naturally, to improve accuracy the inputs may be preprocessed before thresholding is applied and in general thresholding can be split to two types global thresholding and local thresholding.

In global thresholding, the same thresholding ranges are applied throughout the whole input. This technique is feasible in cases, where certain pixel values are linked to distinct segmentation instances. Methods may apply thresholding with intensity ranges that are manually elected, or defined using characteristics of the input image. Popular global thresholding methods are Otsu's method thresholding [182], unimodal thresholding [183], bi-level image thresholding [184], minimal cross entropy thresholding [185] and thresholding using Tsallis entropy [186]. Additionally, in an approach proposed by Zhang *et*

al. [187] multiple regions are extracted by thresholding at different intensity ranges and then are appropriately joined in order to offer high accuracy solutions. One the other hand, local thresholding methods use in different portions of the input images different thresholding levels. Sauvola and Pietikäinen [188] proposed a local adaptive approach for document image binarisation. Later Yan *et al.* [189], proposed an approach that utilises both local and global information in order perform thresholding. Shi *et al.* [190] also utilised prior shape information and local thresholding to further improve their methods accuracy for document binarisation. Furthermore, Singh *et al.* [191] proposed a local adaptive thresholding method that utilises a local mean and mean deviation in order to perform thresholding.

Application of a thresholding technique for image segmentation has the benefit for being the most intuitive, simple and fast approach for image segmentation. This is especially true for user-cases where the segmentation instances of the same segmentation class has similar pixel intensity and different pixel intensity from background. However, similar to the edge detection segmentation methods (see Section 2.2.1.1) they are susceptible to noise. Similarly, they are not inherently semantic since in some use-cases semantic objects may be comprised by many regions that do not have similar intensity characteristics. Based on these facts and due to the low SNR in the undersampled tomogram reconstructions, these methods were not considered during the development of our approaches.

2.2.2.2 Segmentation methods based on growing, splitting or merging regions

Another category of methods that utilise continuities and similarities between pixels of the same segmentation instances are methods that use growing, splitting and/or merging regions. In region growing methods neighboring pixels are iteratively grouped together if they share similar characteristics starting from some initial starting points ("seeds") in the input. The methods stop when there are not any new pixels that can be grouped together in the same growing region. Alternatively, in methods that use splitting and/or merging regions, they iteratively split the input image into multiple regions and then check if the new formed region can be joined together due to common characteristics.

Then iteratively regions are split and merged until the formed regions of two subsequent iterations are the same.

Adams and Bischof [192] proposed an approach using growing regions that start from selected "seeds". Chang and Li [193] also proposed an approach that apart from growing regions, regions are also merging. Tremeau and Borel *et al.*[194] later proposed an approach for coloured images and Shih and Cheng [195] proposed a region growing approach for coloured images where the selection of the "seeds" is automatic. Region growing is commonly combined or performed via a watershed transformation. Zhang *et al.* [196] introduced an approach that utilised an improved watershed transformation for region growing while also applying a region merging step. Tang later utilised watershed transformation for the segmentation of coloured images using growing regions. Furthermore, Neubert and Protzel [197] later introduce a compact watershed approach for the segmentation of images into compact regions/superpixels.

On the other hand, methods have been used for region splitting and merging typically split the different regions using graph-based methods. Boykov and Jolly [198] proposed a graph-cut approach where their graph is simulated via an MRF and the MAP estimation is used. Later Boykov and Funka-Lea [199] improved it via the addition of several extensions. Both methods perform region splitting using a graph-cut, however they also require a tiny amount of user annotations (similar to "seeds" of the region growing methods), in order to separate the background from the foreground segmentation instances. In attempt to fix this issue, Rother *et al.* [200] proposed a graph-cut approach that it is iteratively optimised, which does not require some initial "seeds" provided by the user like in [198, 199]. Nevertheless, the user can optionally aid the method in the final step. Freedman and Zhang [201], improved the segmentation accuracy by introducing shape priors and Malcolm *et al.* [202] improved it further with the introduction of highly variable nonlinear shape priors. Later Price *et al.* [203], similar to *et al.* [174], introduced geodesic distance information into the formulation of their graph, allowing their method to be more accurate than other contour-based geodesic methods irregardless of the initial "seeds" provided by the user. Furthermore, Tang *et al.* [204] proposed the use of a

simpler energy term, that results in a more efficient approach in the use-case where only one object has to be segmented in the input image. Finally, Luengo *et al.* [205] proposed a region splitting and merging method for accurate oversegmentation of the input into many superpixels.

Methods based on growing and merging regions are designed to be robust against inputs that are noisy, multimodal and where illumination can vary in different areas of the input image. Additionally, their computation times are small, however it is not always certain the resulting regions will correspond to semantically meaningful objects. On the other hand, methods based on splitting and merging regions, provided that the user has carefully and accurately selected the initially "seeds", are able to achieve higher accuracy results even with noisy, multimodal or with inputs where the illumination varies greatly within them. This allows them to potentially offer semantically meaningfully results with a higher accuracy. Nevertheless, they are typically computationally expensive and they require user input in order to accurately segment each input image. In any case, if the output regions of the aforementioned methods are not semantically meaningfully, the user may be able to manually merge the resulting regions in order to obtain the semantic segmentation. This is possible provided that each output region contains partially only one semantically meaningful object. For this reason, some methods [197], [205] aim to accurately oversegment the input, meaning that each of the resulting output regions/superpixels contain partially or completely only one semantically meaningful object. It is then possible for the user to merge the different regions forming semantically meaningful objects. Alternatively, the output regions could be merged using a learned-based approach that relays on the manual annotation of some regions to certain segmentation classes [206]. Nevertheless, the aim of this thesis is to propose approaches for the automatic and semantic segmentation of undersampled tomograms in time-series of tomograms. Due to the large number of tomograms that typically comprise each times-series, it is highly important that the proposed methods are as automatic as possible. Additionally, due to the large size (in voxel-count) of the tomograms, fast inference times are important for the proposed approaches. Methods that use growing or splitting regions

can be accurate, however they are typically computationally expensive and so do not offer fast inference times. Based on these reasons, region growing or splitting methods and techniques were not considered during the development of our approaches.

2.2.2.3 Segmentation methods based on clustering

A different class of approaches that perform image segmentation use clustering in order to group the different pixels into the output segmentation instances. In clustering approaches, pixels or superpixels are grouped together into different classes in attempt to maximise both the intra-class similarity and inter-class dissimilarity. In general clustering approaches are split to hard clustering approaches, where a pixel or superpixel can belong only to only one class and soft or fuzzy clustering approaches where a pixel or superpixel can belong to multiple classes with a degree of confidence.

One of the first hard clustering approaches proposed for image segmentation was from Pappas and Jayant [207] where they used k-means clustering along with an MRF in order to induce spatial constraints on the k-means. Later Chen *et al.* [208] improved their approach by incorporating knowledge-based morphological operations. Ray and Turi [209], in attempt to remove the k-mean drawback of manually selecting the number of clusters, proposed the use of a validity metric in order to automatically determine the number of clusters. Luo *et al.* [210], also proposed a spatially constraint k-means approach combined with region merging in order segment coloured images. Achanta *et al.* [211], then proposed a Simple Linear Iterative Clustering (SLIC) approach based on k-means for the oversegemtation of input images. Recently Dhanachandra *et al.* [212], proposed a subtractive clustering technique for the selection of the centres that can be utilised from a k-means clustering in order to perform image segmentation. Apart from k-means also mean-shift clustering has been proposed for image segmentation [213]. The advantage of mean-shift is that it inherently does not require prior knowledge for the number of the segmentation instances in an input. Wang *et al.* [214] proposed an anisotropic kernel for the mean-shift method, in order to incorporate prior knowledge depending on the data which have to be segmented. Paris and Durand [215] also proposed the use of a

fast Gaussian mean-shift method that is applied within a hierarchical segmentation based approach.

On the other hand, soft or fuzzy clustering approaches have also been proposed for image segmentations. The key difference in soft or fuzzy clustering approaches is that each pixel or superpixel may simultaneously belong with a degree of confidence to multiple classes. Furthermore, one of the most popular methods for soft/fuzzy clustering is the Fuzzy C-Means (FCM) [216, 217], where has many similarities to the k-means method but membership of a pixel to class is calculated using fuzzy logic. In an attempt, to utilise FCM for image segmentation Tolias and Panas [218] proposed the use of adaptive spatially constrained membership functions thus improving the method's segmentation accuracy. Later Ahmed *et al.* [219] proposed a modified FCM method that can compensate for inhomogeneities in the input and allow for accurate segmentation even with the presence of noise. Chen and Zhang [220] proposed an improved FCM method with spatial constraints, that is robust to noise and image artifacts. Furthermore, Krinidis and Chatzis [221] aiming to increase robustness further, proposed an approach that incorporates local information and additionally does not employ any empirically chosen parameters.

Clustering-based segmentation methods have the advantage of being simple in implementation, have fast computational times and have noteworthy accuracy when they are spatially constrained. However, compared to other methods like region growing methods or region splitting methods they are not very robust. For these reason they may be combined with such approaches for additional accuracy [210]. However, the tradeoff is that the overall solution becomes computationally expensive. Furthermore, they are not inherently semantic and in cases where they are not spatially constrained very well, they may lead to output segmentation regions that partially contain many semantic objects. Since the aim of this thesis is to propose approaches for the automatic and semantic segmentation of undersampled tomograms of a time-series, methods based on clustering were not considered during the development of our approaches.

2.2.2.4 Learning-based segmentation methods

Finally, a category of methods based on region similarities are the learning-based segmentation methods. Learning-based segmentation methods annotated data in order "teaches" the approaches on how to segment pixels (or voxels) of images (or volumes). This is achieved by during a training step where several parameters of the methods are fine-tuned. Learning-based methods can be inherently semantic if trained on semantically meaningful ground truths. Nowadays the predominant way with which learning-based semantic segmentation is preformed is with the use of deep learning [6]. Deep learning has been able to provide us with methods that undoubtedly achieve extraordinary segmentation accuracy. Nevertheless before its arrival, learning-based semantic segmentation was performed with a variety of different methods, like machine learning and graphical models, and on occasion these methods are still being used today to address certain segmentation challenges.

One such method is Random Decision Forests (RDF) proposed by Ho *et al.* [222]. RDFs approach the problem of segmentation through a technique called ensemble learning. For this technique, multiple weak classifiers (decision trees) are trained, each on either a different subset of the annotated data or a different subset of the feature space. The final prediction can be made, through the union of their individual predictions. With RDFs, segmentation can be applied directly to pixels, where each pixel is represented by a descriptor of features and then used for the training of the multiple trees. Since each decision tree is weak, their individual predictions may not be by themselves sufficiently accurate. However, the fact that they are trained with limited information leads each of them to learn different discriminative rules, but their combined prediction achieves a substantial accuracy.

Decision forest are still being used in many applications, especially where fast execution times are highly significant. Schroff *et al.* [223] incorporated multiple globally learnt single-histogram class models into the Random Forest framework, resulting in accurate segmentation. Montillo *et al.*[224] proposes a new entangled decision forest that offers improved accuracy and it can capture long-range semantic context. Sommer *et al.* [225]

also proposed an interactive tool for volume segmentation using a random forest classifier. Zuo and Drummond [226] later proposed fast residual forests that utilises random forests and CNNs in order to reduce computational time for time critical applications like in robotic learning. Recently Kang and Nguyen [227] proposed a random forest framework for real-time semantic segmentation which learns flexible filters with unconstrained weights, shapes, and sparsities.

The advantages of Random Forest is their ability to learn segment or classify semantically meaningful objects, their easy implementation and their fast training and execution times. However, they are usually not very robust against noisy data and they are oftenly combined with other methods in order to improve their accuracy [226]. For this reason and due to the severely low SNR in reconstructions of undersampled tomograms, we decided to shift our focus on learning-based segmentation that employ deep learning during the development of our approaches.

Support Vector Machines (SVM) [228, 229, 230] have also been proposed for semantic segmentation. Each SVM works as a binary classifier, which aims to find a hyperplane in the feature space that separates the training set between two classes. However the sparsity of the training samples in the high-dimensional feature space does not insure their linearly separability. This means that there may not be a hyperplane that can separate the samples into two desired classes in the high-dimensional feature space. Nevertheless, there should be a non-linear mapping Φ that transforms the training set to a feature space with additional dimensions, where linearly separability can be achieved. After this transformation, the desired hyperplane can be found by solving an optimisation problem regarding the hyperplane's equation coefficients. The optimisation aims to find a hyperplane where the two classes are separated, but also the samples are as distant as possible from the hyperplane. Similar to RDFs, multiple classifiers can be used in order to achieve segmentation between multiple classes.

Kotropoulos and Pitas [231] proposed a SVM-based segmentation approach for the accurate segmentation of lesions in ultrasound images. Later Wang *et al.* [232] proposed an SVM method for pixel-wise classification/segmentation where the SVM is trained with

samples selected by a FCM. Bauer *et al.* [233] also introduced a SVM segmentation method that utilises a Conditional Random Field (CRF) for additional robustness. Similarly Wu *et al.* [234] proposed a method that combines a SVM and a MRF for both high accuracy and fast training and testing times. Bai and Wang [235] later introduce saliency information into their proposed SVM segmentation approach in order to further improve the segmentation accuracy. Qayyum and Basit [236], combined the SVM with other edge detection and thresholding techniques for the segmentation of cancerous cells in mammograms. Akram *et al.* [237] proposed a method that introduces multichannel saliency information to an SVM for the classification and segmentation of skin lesion. Recently, Kour and Arora [238] proposed a k-means and SVM segmentation/classification method for plant phenotyping images that is optimised using the particle swarm optimisation algorithm [239].

SVMs similar to RDFs, they are able to offer semantic segmentations, they are very easy to implement and they have fast computational times. However, when they are exclusively used for semantic segmentation they do not usually lead to very robust approaches. For this reason, they are typically combined with other methods such us CRFs [233], MRFs [234], edge detection and thresholding [236], and k-means [238] in order to perform the demanding task of segmentation. In these approaches the main focus of the applied SVMs is to contribute towards the classification of the segmentation instances and generally make the segmentation results semantically meaningful. While such approaches might have sufficient accuracy for specific use-cases (like in skin lesion segmentation [237] and plant phenotyping segmentation [238]) they are typically not robust enough for the segmentation of noisy multidimensional imaging systems like the reconstructions of undersampled tomograms in X-ray CT time-series. Based on this fact, SVMs were not considered during the development of our approaches, focusing instead on state-of-the-art learning-based semantic segmentation approaches that utilise deep learning.

Graphical models and specifically MRF [105, 106] have also been utilised for learning-based segmentation. MRFs are described by an undirected graph $G = (\mathcal{N}, \mathcal{E})$, where nodes $\mathcal{N}$ represent random variables and edges $\mathcal{E}$ the dependencies between them. Ad-

ditionally, the nodes obey the Markov constraint where for any two disjoint subsets of nodes A, B that are separated by a subset of nodes S, then the subset of variables Z_A (the subset of variables that A represents) is conditionally independent of Z_B given Z_S, $Z_A \perp\!\!\!\perp Z_B \mid Z_S$ (where Z_B and Z_S are represented by B and S respectively). Namely, a MRF with 4-connectivity is presented in Figure 2.3. Each of random variables X_i represent an input feature vector for each pixel and the random variables Y_i represent pixel classes, whose domain is from a set of labels $\mathcal{L} = \{l_1, l_2, \cdots, l_k\}$. According to the Hammersley-Clifford theorem [240] the joint probability distribution of an MRF $P(\boldsymbol{X}, \boldsymbol{Y})$ where $\boldsymbol{X} = X_1, X_2, ..., X_n$ and $\boldsymbol{Y} = Y_1, Y_2, ..., Y_n$ can be written as a Gibbs distribution. Then the prediction of each pixel's class is formulated into a MAP estimation, which involves the minimisation of an energy function. Verbeek and Triggs [241] combined an MRF with aspect models like probabilistic latent semantic analysis [242] and latent dirichlet allocation [243] in order to construct a learning-based semantic segmentation approach. Subbanna *et al.* [244] later combined Gabor decomposition and a MRF for the segmentation in brain MRI volumes. More recently, Ahmadvand *et al.* [245] proposed a dynamic classifier selection MRF. Their method is a supervised segmentation approach for brain MRI images, that combines an MRF along with an ensemble of multiple classifiers. While, MRFs can provide accurate segmentation results, typically as explained below, CRFs are more preferred in segmentation approaches due to their less complex MAP inference.

Conditional Random Fields (CRFs) [246] are a special case of MRFs, that are more frequently used in semantic segmentation approaches. In CRFs, observations $\boldsymbol{X} = X_1, X_2, ..., X_n$ which are again the input feature vectors for each pixel and outputs $\boldsymbol{Y} = Y_1, Y_2, ..., Y_n$ that represent the pixel classes can be divided into exactly two disjoint sets and the conditional distribution $P(\boldsymbol{Y}|\boldsymbol{X})$ can be modeled. In an early method proposed by He *et al.* [247] a multiscale CRF for image segmentation was introduced which learns prior label patterns from multiscale features of the training images. Later Wojek and Schiele [248] proposed a CRF methods that performs jointly both the tasks of object detection and semantic segmentation in order to achieve higher accuracy in both tasks. Ladický *et al.* [249] introduced a Hierarchical CRF that utilises features from different

levels based upon a learned hierarchy of pixel groupings. To further improve accuracy, Krähenbühl and Koltun [250] proposed a fully connected CRF, where all pixels in an image are nodes in the CRF, resulting in billions of edges. To address the computational overhead, they defined the pairwise potentials of the CRF as a linear combination of Gaussian kernels and they also proposed an efficient mean field approximate inference algorithm. Recently He and Gould [251] proposed a CRF for instance segmentation which is the task of detecting objects of a specific class in the input image and then also semantically segment them. To perform this, they created a CRF framework that jointly models object appearance and occlusions as well as shape deformations.

Both MRFs and CRFs can lead to robust segmentation approaches with high accuracy and they are still utilised in improving the accuracy of stated-of-the-art deep learning methods (More in Section 2.2.3.2). Nevertheless, they are typically associated with high computational and time costs during both training and inference. As we will explain in Section 2.2.3.2, the utilisation of a MRF or CRF along with a very deep neural network may increase the training and inference times to undesirable levels. This is because in this thesis we aim to design approaches that will be applied on time-series of multiple undersampled tomograms and it is expected our methods may be retrained frequently for the accurate segmentation of new time-series. For these reason during the development of our approaches, MRFs or CRFs were not considered. Due to the high number of proposed approaches using deep learning and since they present more interest to this thesis, the entire next Section will present the different categories of deep learning approaches that have been proposed for the task of semantic segmentation.

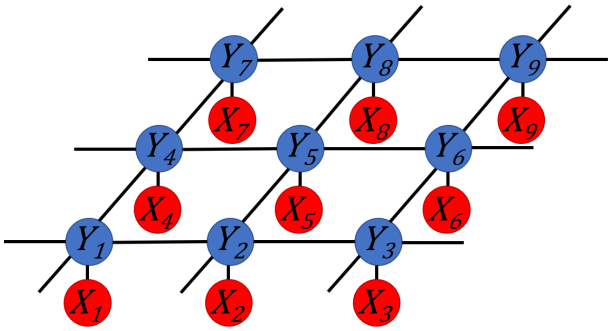


Figure 2.3: *MRF with 4-connectivity. Each node $\boldsymbol{X}_i$ represents a pixel's feature vector and each node $\boldsymbol{Y}_i$ represents a class.*

2.2.3 Deep learning semantic image segmentation

Since the arrival of Convolutional Neural Networks (CNN) [19] and especially deep learning [6] a variety of different architectures have been explored. Initially, the presented architectures were mostly used for classification and they were limited to a small sequence of convolutional layers [18, 19] separated by activation functions [252] and max pooling operations [253, 254]. However, with the introduction of AlexNet [255] the door was opened for more complicated architectures, leading to the development of ZFNet [256], VGG-16 [125], GoogLeNet (or Inception) [257], ResNet [129], DenseNet [143] and recently also efficient architectures for deployment in mobile devices like MobileNet [258, 259] and ShuffleNet [260, 261]. This increase in popularity of deep learning classification methods in turn inspired other researchers to adapt the classification CNN architectures to address the image analysis problem of segmentation. This resulted in the introduction of multiple novel segmentation methods described below, which achieve phenomenal performance and accuracy compared to the approaches presented in Sections 2.2.1 and 2.2.2. Most deep learning semantic segmentation approaches operate based on the same high level principle which is to fine-tune their parameters during training and then use the final parameters during inference. In general, most of them receive an image, volume or other high-dimensional input and then they output its semantic segmentation (with one notable exception the instance segmentation networks that output both bounding boxes and semantic segmentations. More about them in Section 2.2.3.3). Based on this principle, most of them are also tested using the same datasets/benchmarks [262, 263, 264]. For this reason there is more merit by categorising them based upon their architectural differences which allow them to achieve highly accurate semantic segmentations. Below we will explore some of the most known semantic segmentation deep learning methods, split into 8 different categories based upon their model's architecture.

2.2.3.1 Fully convolutional networks (FCNs)

One of the earlier attempts to use deep learning techniques for image segmentation, particularly for 2D Electron Microscopy (EM) [4, 5] images, was from Ciresan *et al.* [265]

in 2012. In their publication the researchers used a sliding-window network setup which consisted of a sequence of convolutional [18, 19], max-pooling [253, 254] and fully connected layers, which are similar to the layers in multilayered perception-like networks [20]. Their network works by receiving as input a local region (patch) around a pixel and calculating a number of features regarding the pixel being either part of the foreground or the background. Then, the final prediction about the class of the pixel is made through the use of a softmax function [266, 267] and thresholding, By repeating this process for all pixels of the image and zero-padding the input patches, the overall segmentation is acquired. Zero-padding in particular, is the act of upscaling the input of a convolutional layer with pixels with zero value around its border prior to the convolution. Zero-padding is usually performed when sliding windows request information outside the borders of an image and in cases where the size reduction caused by the convolution has to be negated, so that the input and output have the same size. During experiments, their network outperformed most state-of-the-art segmentation approaches of that time and created a precedent regarding image segmentation with deep learning. While being very accurate, the fact that it predicts one pixel at a time is not very convenient in terms of time both for training the network and inferring with it. For this reason *fully convolution networks* were developed as we will discuss below.

Long *et al.* [268] proved that fully convolutional neural networks (FCNs) could exceed existing state-of-the-art segmentation methods. For their approach they adapted popular classification networks (like VGG-16 [125] see Figure 2.4) into fully convolutional networks by converting their fully connected layers into convolutions. However, due to the downscaling that occurs gradually through the network (pooling operations), the final upscaled output is dissatisfyingly coarse. To remedy that, they introduce *skip* connections, that combines finer detail information from early shallower layers, with coarse ones from the deeper layers. Namely, the tensors from deeper layers are appropriately upsampled and then added to the ones from the shallow layers. Additionally, the upsampling is executed through transpose convolutional or deconvolutional layers [120, 121], which are trainable like typical convolutional layers. Finally, the researchers used classification *pretrained*

networks for parameter initialisation prior to the segmentation training, as they appear to offer a performance advantage over randomly initialised networks.

Nevertheless, in Long *et al.*'s network, as it forwards information from one layer to the next, it does not also forward global context information in an efficient way. This can affect the final segmentation accuracy, as in some segmentation problems global context information is essential (e.g. in pictures of natural scenes, whether a pixel's class label is grass or tree foliage may be very correlated to its global position in the image). For this reason Lui *et al.* proposed ParseNet [269], where by utilising global average pooling [270] it is able to efficiently utilise global context information. Specifically, ParseNet is a fully convolutional network, where in each layer after the convolution, the feature maps are globally average pooled, resulting in a vector with the same number of entries as the number of feature maps. Then both the feature maps and the vector are ℓ^2-normalised and the vector gets unpooled (replicated). This means that each of its entries becomes a feature map, forming the "global" feature maps. Finally, the initial feature maps and the "global" feature maps are concatenated before the handled to the next layer.

These advances in image segmentation, were later adopted to address problems such as retina vessel segmentation [271], segmentation of images with dermatological conditions such as skin lesions [272, 273] and segmentation of brain tumors [274].

FCNs are some of the first networks proposed for the segmentation of multiple pixels with in a single forward pass of the network. However, one of their drawbacks is that they mainly employ standard convolutional layers and pooling operations, which results in segmentations which are not be very accurate at the same pixel resolution as the input image. In the following Sections it will be shown how other network designs address this problem, which also influenced the design of our proposed approaches.

2.2.3.2 FCNs with graphical models

In an effort to further improve the performance of the fully convolutional networks, probabilistic graphical models such as MRFs and CRFs (More in Section 2.2.2.4) have been proposed as an additional and final part of the networks. Namely, Chen *et al.* [275] pro-

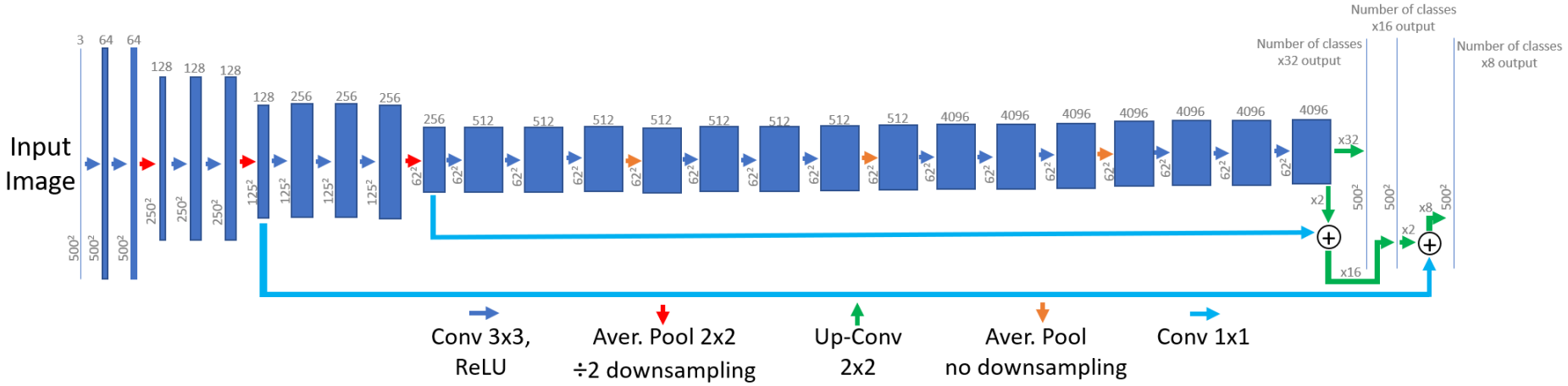


Figure 2.4: *FCN-VGG-16 architecture. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The height and width size is provided at the lower left edge of the box.*

pose a semantic segmentation algorithm that combines CNNs and fully connected CRFs. Their approach, based on Long *et al.*'s VGG-16 [268], highlights the additional accuracy that a Conditional Random Field can provide to the segmentation results. This is due to the difficulty of fully convolutional networks to offer very fine-grained segmentation predictions, as their translational invariance property makes them optimal for high level tasks such as classification. CRFs, by utilising the CNNs predictions, are able to solve their localisation deficiency and increase the segmentation accuracy.

Later Zheng *et al.* [276] proposed a Recurrent Neural Network (RNN) [277] that approximates the function of a CRF, allowing for the whole CNN-RNN network to train conjointly without requiring initial training of the CNN component. Their method achieved a new state-of-the-art standard in the Pascal VOC 2012 benchmark [262]. Similarly, Liu *et al.* [278] proposed a modification of VGG-16 [125] in order to model the unary terms of MRFs and added layers that approximate the mean field algorithm (MF) [279] for MRF's pairwise terms. However, their most important contribution was that the MRF approximated by their Deep Parsing Network (DPN) was able to offer deterministic end-to-end predictions, allowing training with a single iteration of the MF for each training image. Finally, instead of modeling CRFs with neural network, the option of enhancing them with CNNs is explored by Lin *et al.* [280]. Specifically, they formulated a CRF with CNN-based pairwise potential functions in order to model "patch-patch" semantic relations between them. Apart from collecting "patch-patch" context, they also utilised multi-scale inputs [281] and sliding pyramid pooling [282] for the collection of "patch-background" context, that improves the segmentation accuracy further.

As it was presented in this Section, the combined use of MRFs and CRFs with CNNs can lead to very accurate semantic segmentation techniques. Nevertheless, as it is briefly mentioned in end of Section 2.2.2.4, they are computationally expensive and they commonly have large inference times. Because of that, a typical use-case scenario for combining CNNs with graphical models is when network retraining in the near future is highly improbable and the main focus is to create specialised networks that are as accurate as possible for the segmentation of specific images or volumes with similar imaging charac-

teristics (e.g. for networks specialised only on lung CTs, thorax CTs, brain MRIs, etc). It is important to note that based on the reported experimental results, the addition of MRFs and CRFs increases the accuracy but not by a large degree. Therefore, for tasks where fast training and inference times are important, the additional accuracy offered by MRFs and CRFs may not justify the additional computational cost associated to them. For these reasons during the development of our approaches, deep learning methods that also utilise a graphical model were not considered. Instead, more focus was brought to the creation of efficient and accurate deep CNNs.

2.2.3.3 Region-based convolutional networks (R-CNNs)

An alternative way to improve the performance of deep learning segmentation networks was proposed by Girshick *et al.* [283] by combining the tasks of object detection with one of semantic segmentation. They discovered that a successful initial application of object detection can significantly aid the segmentation results. Additionally, this allows for great performance even when the labeled training samples are low in number. Their method named, region-based convolutional neural networks (R-CNN), later improved by the proposal of Fast R-CNN [284], lead to the proposal of applications that approach the task of semantic segmentation along with object detection which is conjointly named *instance segmentation*.

Particularly, Ren *et al.* proposed Faster R-CNN [285] which combines Fast R-CNN and their region proposal network (RPN) to offer potential bounding boxes for object detection, by utilising feature maps created by convolutional layers (from a VGG-16 as seen in Figure 2.5). Their RPN network calculates the probabilities of bounding boxing candidates. These candidates are 9 for each pixel, centered on each pixel with predetermined shapes and size (anchors). For each of the 9 in each pixel regression coefficients are also calculated for precise localisation. These 9 anchors are divided to 3 groups of different scale anchors and within each group each anchor has a different aspect ratio. In order for the RPN to obtain the final bounding box proposals, the earlier candidates are sorted based on the probability that contain an object and using non-maximum suppression

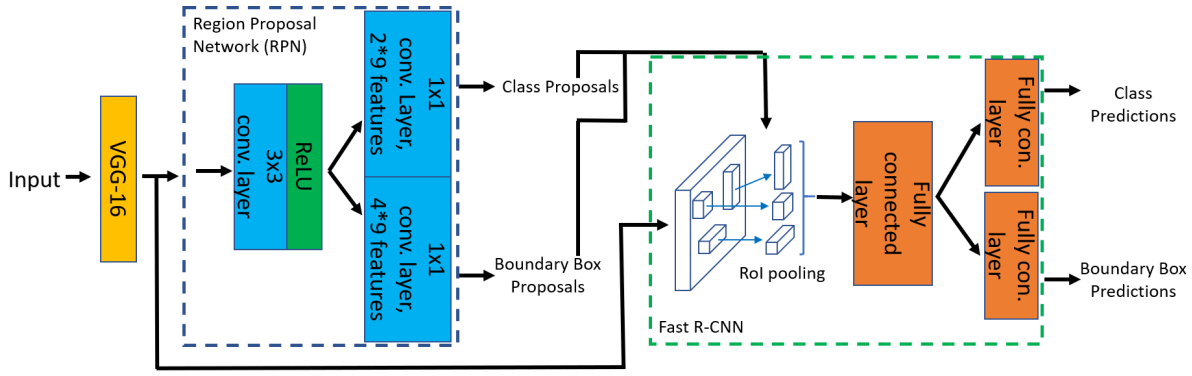


Figure 2.5: *Faster R-CNN architecture. The Region Proposal Network (RPN) calculates for every pixel the probability whether or not 9 boxes centered in each pixel capture an object. These 9 anchors are divided in 3 groups with different scales and within each group each anchor has a different aspect ratio. Regression coefficients are also calculated for each of the anchors in pixel position. The Fast R-CNN partition uses the bounding boxes and the class proposals by the RPN (reduced using non-maximum suppression [286] with 0.7 IoU threshold), performs a region of interest (ROI) pooling and with fully connected layers offers the final class and bounding boxes predictions.*

[286] (with threshold of 0.7 IoU) the final bounding box proposals are drawn. Then the bounding box proposals and feature maps (from the VGG-16) are processed by Fast R-CNN's region of interest (RoI) pooling layers in order to offer the final bounding box and class predictions.

Moving towards R-CNN approaches that offer more refined segmentations results, Dai *et al.* [287] proposed three networks that each utilises the output of the previous one and predicts progressively more refined outputs. Namely, the first outputs bounding boxes, then the second object masks and finally the third semantic segmentations. The whole approach is trained end-to-end and the outputs of all three networks is supervised, which means that the overall loss function tries to minimise the errors in all three outputs. Later Li *et al.* inspired by Dai *et al.*'s updated work [288], revisited the Faster R-CNN architecture, making it fully convolutional. They proposed a more sophisticated network [289] that jointly and simultaneously detects and segments input images.

In an attempt to retain the fast inference times of Faster R-CNN [285], while also offering more refined segmentations (masks) He *et al.* proposed their network Mask R-CNN [290], that forwards the final feature maps to a new branch with convolutional layers that predict refined segmentations (masks). All three outputs of the network (name of the class, bounding box and mask) are supervised by a loss function which is a combination

of the losses regarding the three outputs. A similar approach was proposed by Pinheiro *et al.* [291] where the bounding box output is omitted and the backbone of the architecture is based on VGG-16 [125] with some additional final layers that generate the class and mask predictions. Pinheiro *et al.* later revisited their work proposing an encoder-decoder network [292] capable of producing high-fidelity masks. The advantages of encoder-decoder architectures were also utilised by Liu [293], and by using an encoder-decoder-encoder scheme and feature maps across multiple scales of the final encoder, their network PANet offers very accurate bounding box, class and mask predictions.

Another interesting method, proposed by Hu *et al.* [294], introduces a novel parameter transfer function in the Mask R-CNN architecture. The parameter transfer function is applied from layers that detect the bounding boxes to the ones that detect the segmentation mask. Their proposal allows the network to learn how to segment a large category of objects, while all training samples have bounding box annotation, but only a few segmentation mask annotations. Further improvement towards the segmentation with only bounding box annotations is later proposed by Hsu *et al.* [295]. They solved the problem by cropping the training samples and forming "bags" for Multiple Instance Learning (MIL) [296, 297]. Specifically, in MIL, labels are known only for these "bags" and there are two versions of bags. "Positive bags" are crops within the labeled bounding boxes, thus containing at least one pixel that belongs to an object that has to be segmented. "Negative bags" are crop outside the bounding boxes containing only background pixels. Furthermore, Chen *et al.* [298] proposed a modification of the Faster R-CNN that also produces direction predictions. These direction predictions are estimations for each pixel's direction towards the center of the object which it is predicted to belong to. This additional output, is utilised for the prediction of the final segmentation masks and allows for better separation of different objects/instances in the same image that belong in the same semantic class. Moreover, Kai *et al.* using hybrid cascade networks [299] also improve the accuracy by effectively employing deep supervision [127]. Finally, Boyla *et al.* proposed a real-time method for instance segmentation [300] based on Lin *et al.*'s [301] object detection approach, with exceptionally good accuracy.

Based on the methods described above, R-CNNs by performing instance segmentation, which is the task of object detection and then semantic segmentation are able to achieve improved segmentation accuracy while at the same time producing a secondary output. Additionally, through MIL [296, 297] they are also able to offer segmentation results even with the absence of semantic annotations. A common element of R-CNN is that they all require bounding box annotations and that they perform really well in scenarios where specific objects in the input have to be detected and then segmented. However, in many occasions bounding box annotations are not provided (in X-ray CTs and MRIs). Furthermore, in some use-cases manual bounding box annotations and semantic annotations are roughly equally difficult to be obtained. If the end goal is the semantic segmentation, then the collection of only semantic annotations and the utilisation of approaches that only focus on semantic segmentation is more sensible. Moreover, using approaches that only perform semantic segmentation is more sensible in images or volumes that contain large continuous regions that have to be segmented (e.g. in tomograms that image continuous dendritic structures in metallic alloys as they are solidifying [302, 303, 304, 305]) and not multiple specific that depict certain objects. Based on these facts, R-CNN approaches were not considered during the development of our approaches for the semantic segmentation of undersampled tomogram reconstructions. This is primary because the generation of accurate bounding box annotations is equally difficult to the generation of semantic annotations in X-ray CTs, and considering the available tools for semantic annotation [306] manual semantic annotation might be easier. Additionally, we are aiming to develop, as best as possible data-generic approaches that can even be applied in cases where continuous regions have to be segmented (like in [302, 303, 304, 305]). Nevertheless, the general idea of jointly performing two relating tasks, influenced us to develop our Stacked-DenseUSeg [26, 27] denoising-segmentation network, where the network generates a denoised version of the input which aids its semantic segmentation.

2.2.3.4 Encoder-decoder architectures

As classification networks [255] became prominent, the issue of comprehending their exact function became apparent. To solve this issue, Zeiler *et al.* proposed an encoder-decoder

scheme with convolutions and deconvolutions or transpose convolutions [121, 256] in order to visualise the feature maps produced from the deeper convolutional layers and provide some understanding for the network's functionality. Nevertheless, this encoder-decoder layout inspired Noh *et al.* to utilise the architecture for the task of semantic segmentation [307] and used the convolutional layout of VGG-16 [125], with convolutions and transpose convolutions, for the encoder and decoder respectively. A similar architecture was also proposed from Badrinarayanan *et al.* [308]. The novelty of their SegNet architecture is that the decoder forwards the max pooling location-indices of the max pool layers in the encoder to the upscaling layers of the decoder. This eliminates the need for training the upscaling layers in the decoder, reducing parameters and making the network faster. Additionally, they also use knowledge transfer and initialise the encoder's parameters using a pretrained VGG-16 for classification, which further boosts their network's performance. A bayesian implementation of the network [309] was later proposed, using Monte Carlo sampling with dropout during testing in order to acquire the posterior distribution of the pixels' class labels.

At the same time, inspired from Ciresan *et al.*'s [265] and Long *et al.*'s [268] work, Ronneberger *et al.* [156] proposed a new encoder-decoder architecture, named U-Net, for the segmentation of biomedical images. The decoder's features are upsampled and concatenated with feature maps with the same resolution that are produced in the encoder. The concatenated feature maps then pass through convolutional layers that reduce their numbers before they are again upsampled and concatenated with feature maps of higher resolution from the encoder (through skip connections). This process is repeated until the feature maps reach the input's resolution and a 1×1 convolution with a softmax layer calculate the final predictions. In conjunction with their proposed data augmentation technique, their U-Net architecture performed well on many types of biomedical image segmentation problems. This made their architecture a great base for further research on encoder-decoder architectures.

In the meantime, He *et al.* [128, 129] published their Deep Residual Learning framework. In it the researchers propose a new configuration for convolutional neural layers aiming

to combat the problem of "degradation" that appears in deep networks when they start to converge. It has been shown that as a network's depth is increased, its accuracy gets saturated before it degrades rapidly. They counterintuitively noticed that this is not due to overfitting, since both the testing and training errors are increased, not just the former. According to the authors it is expected that with deeper network architectures the training errors should be similar or even lower. If a network A with N layers has completely learned the provided dataset (training error can not decrease further) then a deeper network B with $M > N$ layers should get the same performance by setting N layers the same as in network A and the extra $M - N$ layers as identity maps. However, this is not observed.

The authors also verified that it is unlikely the increased training errors are due vanishing gradients, and according to them the issue is probably the difficulty for the networks to learn identity mappings, and that it is potentially easier to learn zero mappings. Their configuration resolves this issue by adding (+) previous features with later ones. Therefore, if a part of the network had originally to learn a "mapping" denoted as $H(x)$, now it has to learn the residual between this and the input, $H^{'}(x) = H(x) - x$. If $H(x)$ is the identity mapping, $H^{'}(x) = H(x) - x$ is therefore the easier to learn zero mapping. For their experiments, the researchers created multiple residual network architectures (ResNets) with different numbers of layers and tested them on the ImageNet 2015 challenge [263]. From the results, it was observed that the deep residual learning principle is generic and that it has possible applications in a variety of deep learning applications. Furthermore, it is tested on the COCO 2015 [264] and Pascal 2015 challenge [262] for object detection using the Faster R-CNN [285] framework and replacing the VGG-16 [125] with ResNet-101, achieving remarkable results especially on the COCO dataset.

Naturally, the improvements of the ResNet architecture were later adopted for addressing the task of semantic segmentation. Chaurasia and Culurciello [310] proposed a modified U-Net, such that the forwarded feature maps in the skip connections are not concatenated with the decoder's upscaled feature maps, but summed. Pohlen *et al.* proposed an encoder-decoder architecture [311], where instead of skip connections there is a stream of feature

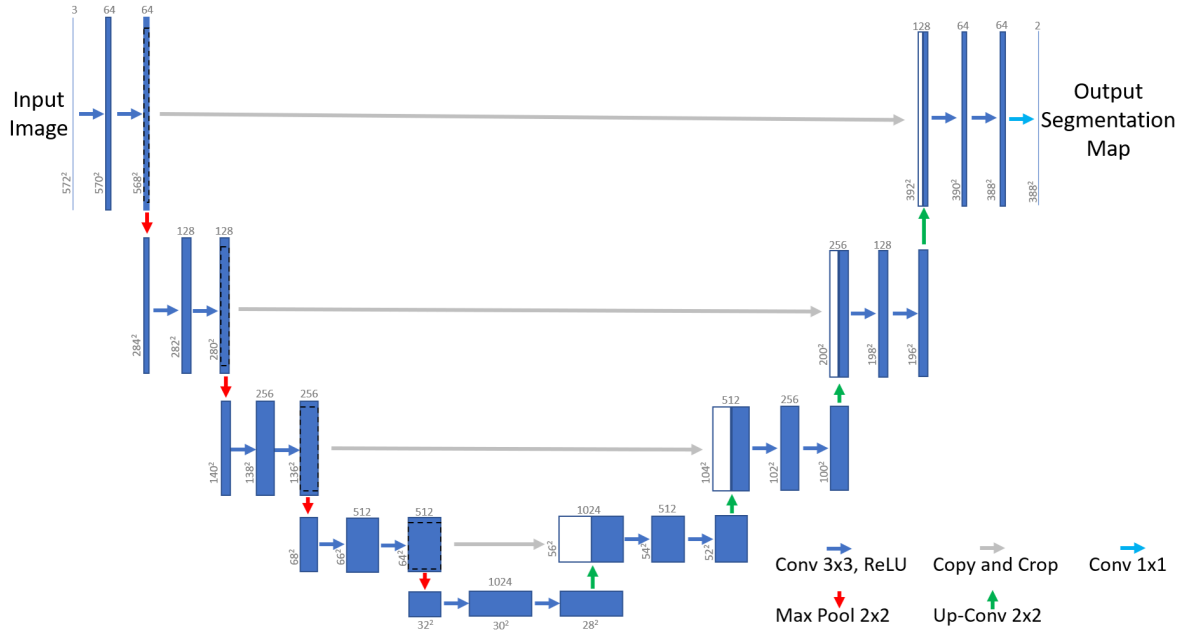


Figure 2.6: *U-net architecture. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The height and width sizes are provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations, as described in the figure.*

maps that retain the same pixel resolution as the input. These feature maps are then updated via summations by feature maps calculated from an encoder-decoder scheme with residual modules. Their approach achieved state-of-the-art results on the Cityscapes benchmark [312]. Moreover, Romera *et al.* proposed ERFNet [313], an encoder-decoder architecture that employs residual layers with factorised convolutions [314] that allow it to achieve great accuracy, while also be able to run in real-time.

Another important publication that influenced a series of proposed segmentation approaches is Newell *et al.*'s Stacked Hourglass Networks [315]. The authors' term "hourglass" refers to the encoder-decoder architectures, as their graphical representations resemble hourglass shapes. In their proposed approach, they arranged multiple *stacked* encoder-decoder networks, where the feature maps of the previous one are input to the next one. Additionally, each encoder-decoder network produces predictions which are supervised, resulting in a deeply supervised network [127]. These "intermediate" predictions do not contribute directly in the formation of the final predictions, but they are used as entry points for the additional supervision. By doing that, each network refines the predictions of the previous one and the number of *stacked* networks is selected

based on the point that the addition of more encoder-decoders offers diminishing returns. This architecture was later adopted for segmentation by Ghosh *et al.* [316, 317], where they proposed SUNets (which are "stacked" U-Nets) that calculate residual feature maps, meaning that at the end of each U-Net the output is summed with its input. For their approach the researchers did not use internal residual blocks in the U-Nets.

Huang *et al.* later proposed densely connected convolutional networks [143] as a better solution to the "degradation" problem in deep networks. In their architecture the residual modules of [129] are replaced by dense modules, which concatenate and do not sum previous feature maps with later feature maps. This means that the task of combining or "summing" the feature maps produced by a dense module is executed within the next dense module. This is usually performed via a bottleneck 1×1 convolutional layer with a ReLU [252] and a Batch Normalisation [134] for non-linearity and robustness respectively. Because of that design, it is easier to re-use the network with already-calculated feature maps for the generation of follow-up feature maps, and it also has a higher degree of freedom in the way it can utilise them. These two facts allow it to achieve highly accurate predictions and to reduce the number of parameters. However, due to the concatenation, the memory demands for storing the continuously growing numbers of feature maps is substantial. To deal with this, the authors proposed intermediate transitional layers, that halted the continuous concatenation, reduced the number of feature maps synthesised so far and condensed the information extracted up until that point.Their architecture was tested on the ImageNet challenge [263] for image classification, managing to outperform ResNet.

Their architecture was later modified for semantic segmentation by Jégou *et al.* [318], who replaced the sequential convolutions of U-Net's encoder and decoder (each responsible for processing different scales of the input) with densely connected convolutions, that each form a dense block. Their resulting approach further improved the state-of-the-art for challenging datasets such as CamVid [319] and Gatech [320]. Fu *et al.* later proposed the Stacked Deconvolutional Network (SDN) [321] borrowing elements from the Newell *et al.*'s architecture. In their proposed architecture a DenseNet [143] is used to acquire

some initial feature maps, which are then upscaled by a decoder with densely connected convolutions. Then the resulting feature maps are processed by 2 *stacked* encoder-decoder nets with densely connected convolutions, providing the final predictions. Similar to Jégou *et al.* [318] the convolutional layers in the encoders and decoders are densely connected; however, the dense blocks are slightly shallower since the architecture has already an enormous number of layers due to the *stacking* of multiple deep networks. Additional skip connections are employed, forwarding feature maps not only within the encoder-decoder pairs, but also between the different *stacked* encoder-decoder nets.

Deep supervision is implemented also similar to [315], however in a slightly different way, which the authors name *hierachical* supervision. Namely, additional supervision is performed for the different layers of the decoders that calculate multi-scale feature maps. Using the feature maps of the layers, pre-prediction feature maps are generated in the same scales as the feature maps via a 1×1 convolution. The final predictions of the i^{th} decoder for the j^{th} scale, are $P_j^i = S(I(F_j^{i-1} + F_j^i))$, where F_j^{i-1} are the pre-prediction feature maps of the previous $i-1$ decoder for the same j^{th} scale, F_j^i are the pre-prediction feature maps of the current i^{th} decoder for the j^{th} scale, $S(\cdot)$ is a softmax layer and $I(\cdot)$ is a bilinear interpolation layer that upscales the predictions to the ground truth resolution. Therefore, the construction of intermediate prediction is not only used as entry points for supervision like in [315], but also to directly enhance later predictions in deeper layers.

The encoder-decoder architecture is one of the most popular network segmentation architectures. Their main advantage of encoder-decoder segmentation networks is that they able to extract multi-scale information via the encoder and then accurately infer the semantic segmentation in the same resolution of the input via the decoder. Additionally, the encoder by reducing the resolution of the feature maps in deeper layers, is able to extract a high number of multi-scale features without highly increasing the network's memory demands. These benefits of the encoder-decoder architectures, motivated us to develop our Stacked-DenseUSeg [26, 27] denoising-segmentation network that is presented in chapter 5. Namely, its advantage of collecting a high number of features, while keeping its memory demands low is a very crucial element for the development of a 3D CNN

segmentation network. This is because, the collection of multiple high-dimensional features can alone increases the memory demands significantly. Furthermore, motivated by Newell *et al.*'s [315] approach, our Stacked-DenseUSeg performs jointly both the task of denoising and semantic segmentation, by stacking 2 DenseUSegs, which is a 3D encoder-decoder architecture also proposed by us. In the following Section 2.2.4, we will further elaborate on encoder-decoder architectures that specifically developed for volumes and high-dimensional data and how they motivated us for the creation of proposed methods.

2.2.3.5 Multi-scale, pyramid and dilated FCNs

As the previous encoder-decoder architectures have demonstrated, acquiring multi-scale context is an essential element for the accurate prediction of the segmentation labels in an image. For this reason, some sophisticated methods for acquiring multi-scale information have been incorporated in deep learning semantic segmentation approaches. One of the first, presented by Yu and Koltun [322] is the dilated convolution. As can be seen in Figure 2.7, in dilated convolutions a number of intermediate pixels of the input are skipped during the operation of convolution. Namely in Figure 2.7 one pixel is skipped for every two that there are not setting the dilation rate to one. This is mathematically the same as the the use of a larger convolution kernel, which is sparsely populated and a number of intermediate parameters constantly fixed to zero. Since neighboring pixels used in a dilated convolution are distanced based on the dilation rate, this rate also dictates the scale with which the dilated convolution "sees" the input. Therefore, using a series of dilated convolutions multi-scale context can be collected. Later Paszke *et al.* inspired by the ResNet [129] and dilated convolution, proposed their ENet architecture [323] for real-time semantic segmentation. Wolterink *et al.* [324] also used a dilated convolutional network for segmentation of cardiovascular MR images.

One other technique for the efficient utilisation of multi-scale context, is the use of *feature pyramids*, which build upon the *image pyramids* [325]. Namely, while *image pyramids* used a set of filtered copies of an image at different scales, each containing useful pattern information for image compression, enhancement, analysis, etc., *feature pyramids* use

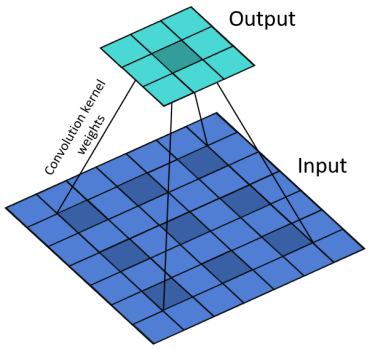


Figure 2.7: *Example of a dilated convolution, with a dilation rate of 2*

features maps of different scales generated by a decoder for the generation of the final predictions. One of the first who used such a technique is Ghiasi and Fowlkes [326], who similar to Lai *et al.* [141], use Laplacian Pyramids [142] to reconstruct the upscaled version of segmentation predictions. In particular, they implemented a network architecture similar to Long *et al.*'s [268], however they used the laplacian pyramid reconstruction method for upscaling the images with a $x8$ rate without transpose convolutions. For higher upscaling rates, they reconstruct/upscale the predictions based on earlier (shallower) feature maps that are already higher in scale. To enhance their precision, they use multiplicative gating with edge information collected from interpolated versions of the smaller scale predictions.

Lin *et al.* [327], who first coined the term *feature pyramids*, introduced a ResNet based encoder-decoder with skip connections which add feature maps together instead of concatenating them like in U-Net. Their network extracts object predictions across different scales, and it is proposed as a good replacement for the RPN or the Faster R-CNN network [285]. It also uses two multi-layer perceptrons [20] for generating segmentation masks; however, this was not the main focus of their publication. During experiments they found that despite the encoder-decoder's great scale representation and its implicit scale variation robustness, it is still highly advantageous to use pyramid representations for multi-scale problems. Later, Zhao *et al.* introduced [328] their Pyramid Scene Parsing Network (PSPNet) (see Figure 2.8). Their architecture uses various ResNet architectures [129] and add the new pooling module at the end of the network named Pyramid Pooling. The function of this module is to pool features maps from a ResNet across 4 different scales.

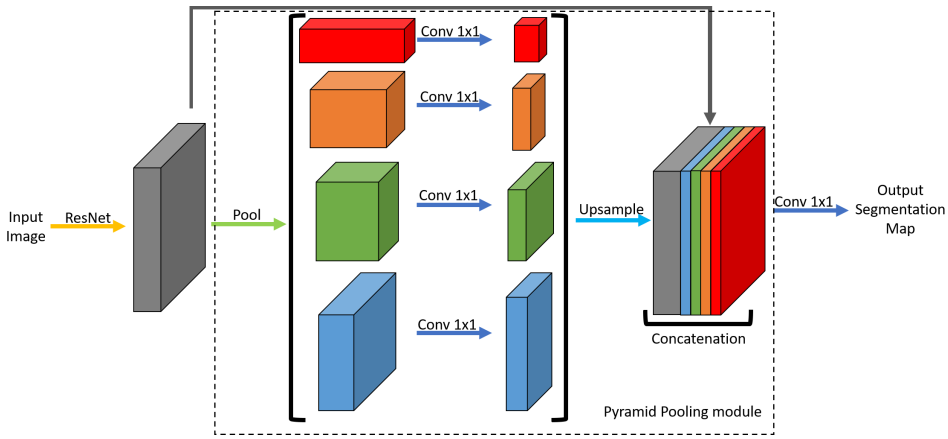


Figure 2.8: *PSPNet architecture. A ResNet variant generates feature maps across different scales and the pyramid pooling module collects and processes these feature maps. Finally, it upscales and concatenates the feature maps with the ResNet's initial feature maps to form the final feature maps. They are then processed by* 1×1 *convolution to generate the final segmentation map.*

Then they are processed by 1×1 convolutions to reduce their numbers, upsampled and concatenated with the initial features of the ResNet. Finally, a convolutional layer is used to generate the class predictions of each pixel. The pyramid pooling module is an important segmentation component as it provides both global and local contextual information in the layers that infer the final segmentation predictions. Moreover, in a manner similar to [326], Lin *et al.* presented RefineNet [329], which is a sophisticated decoder network that can receive as input multi-scale feature maps and produce upscaled feature maps. Namely, their proposed framework uses a ResNet to generate multi-scale feature maps and then a cascade of RefineNets to upscale them. Specifically, starting from the lower-scale feature maps, the first RefineNet upscales them and then the next RefineNet in the cascade uses the previous output and the same-scale feature maps of ResNet to produce further upscaled feature maps. The cascade of RefineNets work as a decoder architecture and the final feature maps are used to generate the segmentation predictions.

Furthermore, in follow-up publication of [275], where the authors already use one or two atrous/dilated convolutional layers (Figure 2.7), Chen *et al.* [330] managed to improve the accuracy of their method by introducing their Atrous Spatial Pyramid Pooling (ASPP). Namely, multiple dilated convolutions with different dilation rates are executed in parallel, creating features for different scales, and thus acquiring multi-scale context. Later, they proposed [331] the usage of cascaded and parallel atrous/dilated convolutions, with

the parallel atrous/dilated convolutions being in the form of ASPPs. Additionally, they introduce 1×1 convolutions and batch normalisation [134] in each branch of the ASPPs for further robustness. Finally, in 2018 they introduce a decoder branch, updating their architecture further [332]. They also changed the backbone of their encoder to Xception [333], that uses depthwise separable convolutions. Namely, for convolving 4 feature maps with a typical 3 convolution and producing 5 output feature maps, in practice this need 5 different $4 \times 3 \times 3$, $(Num\_feature\_maps \times Height \times Width)$ convolutional kernels. The same operation with depthwise separable convolutions is split into two parts. In the first 4 3×3 convolutions are applied to each of the 4 feature maps separately, producing 4 output feature maps. Then to generate the desired 5 output feature maps, a typical 1×1 convolution is applied, with 5 $4 \times 1 \times 1$ $(Num\_feature\_maps \times Height \times Width)$ convolutional kernels.

The advantages of Atrous Spatial Pyramid Pooling were later utilised by a number of publications. Mehta *et al.* [334] proposed ESPNet, that is more efficient thanks to using factorised convolutions [314] and *hierarchical feature fusion*, removing artifacts related to dilated convolutions ("gridding"). Attempting to remove these artifacts, Wang *et al.* [335] proposed hybrid dilated convolutions, that group sequences of dilated convolutions with gradually increasing dilation rate. The proposal of DenseNets [143] was also later utilised by Yang *et al.*, who proposed DenseASPP [336], an ASPP with densely connected dilated convolutions.

Overall, via the introduction of pyramid pooling operations and dilated convolutions, conventional FCNs are able to collect multi-scale information, improving their segmentation accuracy. Nevertheless, the way with which encoder-decoder architectures and pyramid pooling operations collect multi-scale information has a large number of similarities. Based on this and due to the relative success of recently proposed encoder-decoder architectures, we decided to utilise the encoder-decoder architecture for our proposed semantic segmentation solutions. Regarding, the operation of dilated convolutions, as presented in the earlier methods, dilated convolutions are able to collect multi-scale information without the need to reduce the resolution of the feature maps with a pool-

ing operations, which may lead to information loss. This lead to the proposal of some noteworthy approaches [337], and during the development of approaches an architecture that utilises dilated convolutions was investigated (more in Section 5.3.4). However as it was discovered, the extraction of multi-scale information for the task of segmentation is more beneficial via an encoder-decoder architecture. This is because the reduction of the resolution in feature maps of deeper layers keeps the network's overall memory demands low, thus allowing it to extract more features using the same memory quota.

2.2.3.6 Segmentation networks with attention mechanisms

Since the proposal of attention mechanisms in neural networks [338], many publications [339, 340] explored the idea of adding mechanisms which allow the networks to focus on specific areas of the input or features, for the improvement of their accuracy. In particular, regarding the semantic segmentation of images with deep learning, Chen *et al.* [341] proposed the introduction of a soft attention mechanism, that weights multi-scale feature maps for each pixel location. In experiments, they demonstrated that their mechanism by replacing operations such as average and max pooling, that do not have the attention property, is able to increase the network's accuracy. Later, in order to solve the problem of segmenting not only of different classes but also different objects/instances that belong to the same class, Ren *et al.* [342] proposed a recurrent network (RNN) with a visual attention mechanism that counts in a human-like manner different instances for the same class.

Moving on, Yu *et al.* proposed their Discriminative Feature network (DFN) [343]. Namely, a ResNet is used for the extraction of multi-scale features, which are provided to two sub-networks. The first, named Smooth Network, performs global pooling and channel attention. Channel Attention is the estimation of weights to which different channels of the feature maps are multiplied, and thus creating a discrimination between the different features. The second, named Border Network, extracts semantic boundary information in order to allow easier distinction between different classes. Apart from channel attention, self-attention [344] has also been used to relax the locality of CNNs. In particular,

with the use of matrix multiplication it is possible to create connections between all pixel positions of the feature maps. Zhao *et al.* [345] introduced such a self-attentive module at the end of a ResNet. Their Pixel-wise Spatial Attention module (PSA), predicts two global attention maps that connect each pixel position of the feature maps synthesised by the ResNet. After aggregating the information from these maps, long-distance contextual information can be effectively collected. Furthermore, Li *et al.* [346] proposed the Pyramid Attention Network (PAN). Their network introduces a *pyramid*, collecting multi-scale information which is used as weights for the modification of residual information responsible for upscaling feature maps in their network's decoder. Additionally, in their proposed network's decoder, they employ Global Attention Upsample modules (GAUs), which perform global pool and channel attention for the upscaling of low-scale feature maps.

Later, Fu *et al.* in attempted to merge the advantages of channel attention and self-attention by proposing their Dual Attention Network (DANet) [347] (see Figure 2.9). Their architecture uses the ResNet from [275], with its last feature maps shared between two parallel modules. The first is a position attention module, which in a selective manner aggregates features at each pixel position using a weighted sum from the provided feature maps at all pixel positions. After this operation similar features would be related to each other, independent from the distance between them. Moreover, the second module is a channel attention module, which selectively brings attention to interdependent channel maps. This is performed by integrating associated features among all channel maps. In attempt to make attention models not only accurate but also efficient, Huang *et al.* proposed their Criss-Cross Net (CCNet) [348]. Their architecture places consecutive Criss-Cross Attention modules (CCAs) in a sequence after a ResNet based on the one in [275]. The feature maps after the CCAs and the ResNet are then concatenated and used to produce the final segmentation predictions. The novelty of the CCAs, is that they find interdependencies between a pixel and all others that are either directly higher or lower than it or directly left or right of it (a cross pattern). Additionally, by repetitive placement of multiple CCAs in a sequence, non-local interdependencies can be found in

areas outside the cross pattern. By performing this, there in no need to perform traditional self-attention techniques, that are not memory efficient. Recently, Dosovitskiy *et al.* [349] proposed the use of a Vision Transformers (ViT) for image classification. In their approach a single image is split to multiple 16×16 pixel patches that are arranged in a sequence and its is considered as a "word". Then each patch is linearly embed similar to word embedding in Natural Language Processing (NLP) applications and offered as input to a transformer encoder [350] which is used in NLP applications. The advantage of using an transformer encoder for image classification is that it can reinforce the spatial interdependencies within the input via its self-attention mechanisms. Naturally, fairly soon after the proposal of [349] for image classification, an image segmentation approach that partially uses ViT was proposed. Namely, Chen *et al.* [351] proposed the TransUNet encoder-decoder architecture which utilises ViT as the encoder and U-Net's decoder.

In total, the introduction of attention mechanisms have allowed CNN segmentation networks to better focus on more important areas of the input or features, leading to improvements in their segmentation accuracy. Generally, attention is applied either as channel attention which allows networks to focus on specific features or self-attention that relaxes the CNN's locality and allow them to capture long-distance contextual information. Nevertheless, apart from their presented benefits, attention mechanisms have some limitations. Regarding the self-attention mechanism, it is commonly applied by utilising matrix multiplication between a number of features. This typically results in the generation of HR (in regards to the number of imaging elements) feature maps which highly increases the networks memory demands. Furthermore, when this is applied on high-dimensional data (3D and above), memory demands can increase to an unmanageable level. Due to this limitation we determined that the application of self-attention mechanism, despite its potential accuracy benefits, is not ideal for the development of semantic segmentation approaches for undersampled tomogram reconstructions. On the other hand, application of channel attention mechanisms may be more easy to implement for the development of 3D applications. Nevertheless, their application becomes more complex when dropout layers [352] are also applied. This is because dropout layers by randomly "eliminating"

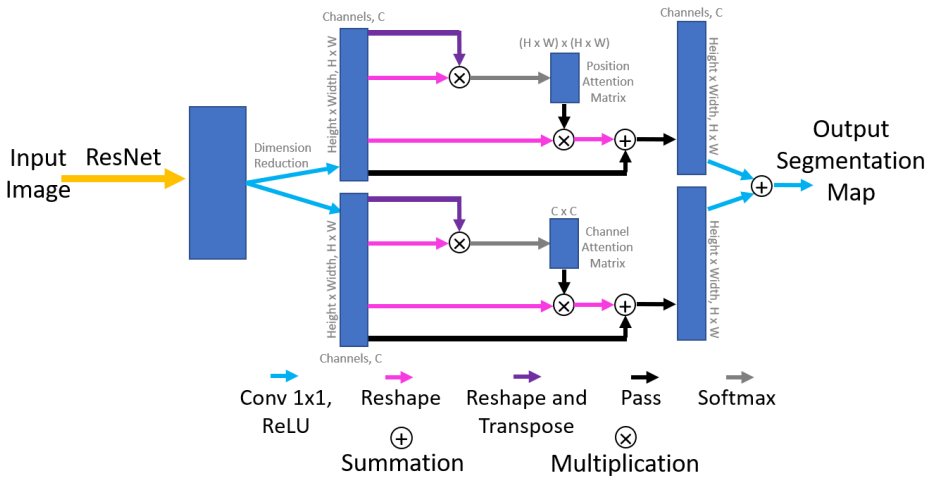


Figure 2.9: *DANet architecture. A ResNet based on [275], generates feature maps which are processed by two attention modules. The first attention module via self-attention reinforces spatial interdependencies, while the second using channel attention reinforces channel interdependencies. Finally, the new feature maps are processed by* 1×1 *convolutions and they are summed. A final* 1×1 *convolution generates the segmentation predictions.*

a number of channels, indirectly create an incompatibility between them the application of channel attention layers. The addition of dropout layers, however in applications that utilise very deep CNNs and at the same time use relatively limited amount of annotated data (e.g. for our application where a single highly-sampled tomogram is used in order to train our approach) can become more crucial for reducing the problem of overfitting than the addition of channel attention mechanisms. Based on this fact, during the development of our approach the addition of dropout layers was prioritised over the addition of channel attention mechanisms.

2.2.3.7 Generative adversarial networks (GANs) for segmentation

Apart from the previously mentioned mechanisms, like encoder-decoder schemes, attention modules and dilated convolutions, generative adversarial networks (GANs) [135] have also been used to for the task of semantic segmentations. The general architecture of a GAN is that they are comprised of two subnetworks, named the generator and the discriminator. The goal of the generator is to generate either an output like the segmentation map of the input, or generate additional training instances, which may be beneficial, as they will allow the network to train further and potentially become more accurate. On the other hand, the discriminator is a subnetwork that typically receives its input from

two sources, the real-world annotations and the annotations made by the generator. The discriminator may also be the subnetwork that performs the segmentation in some approaches; however, it receives its name by attempting to discern if its input originates from the generator or not. During training, the generator is optimised to "trick" the discriminator into believing that the input it provides comes from real-world annotations. At the same time during training the discriminator is optimised to better detect if the input it receives is from the generator. This adversarial training strategy eventually benefits both subnetworks and it is why GANs have the term *adversarial* as part of their name.

One of the first attempts to utilise GANs for semantic segmentation was proposed by Luc *et al.* [353]. Their generator network is a modified version of VGG-16 [125] presented in [322]. A few fully-connected layers are used as the discriminator network. The discriminator receives as input the original images, and also either the generator's predictions or the ground truths. Then it determines if what it received is a ground truths segmentation or the generator's predictions. Because of the discriminator, the generator is then trained to offer more "realistic" segmentations and it is able to remove higher-order inconsistencies between the predictions and the ground truths. Later, Souly *et al.* [354] proposed a semi-supervised segmentation GAN, with the discriminator used as the segmentation network and the generator to produce extra training examples. One additional class (fake class) is added to the ones that have to be segmented and the discriminator also receives unlabeled real-world images as input. The idea behind this implementation is to provide extra examples that may prevent overfitting, which would otherwise not occur with a small set of training data.

Hung *et al.* [355] also proposed a semi-supervised learning GAN approach, but in their implementation the generator is the segmentation network. Namely, their discriminator is supervised towards better detection of ground truth segmentations and predicted segmentations, and their generator is supervised by three loss functions. The first is a typical cross entropy loss, which improves the ability to segment between the different classes. The second is the adversarial loss, which helps to generate more convincing segmentation predictions with correct higher-order structure, and the third is a "semi-supervision" loss.

In particular for the "semi-supervision" loss, unlabeled data are inferred by the generator, and the discriminator produces a confidence map using the generator's predictions. By comparing the predictions and the confidence map it is possible to further train and refine the generator's parameters for more accurate segmentation.

Finally, Xue *et al.* [356] proposed a GAN-inspired approach for the segmentation of medical images. Their approach is end-to-end, using a single L_1 multi-scale loss (absolute errors), and it is comprised of two sub-networks, a segmentor and a critic. The segmentor's predictions are provided to the critic network, which uses them as masks to darken any other part of the input images. The same is also done with the ground truths as masks and the new feature maps are processed by a series of convolutions. After the convolutions, the resulting feature maps are then compared in order to calculate the L_1 loss and train both the critic and the segmentor sub-network.

Overall, generative adversarial approaches can be utilised for segmentation either by increasing the number of training samples and applying a form of weak supervision or by teaching the generator/segmentation network to create segmentation predictions that look similar to manual annotations. These applications can be very beneficial in use-cases where limited annotations are available or when segmentation networks can be benefited by learning some shape priors. Nevertheless, there is a number of limitations associated with GAN architectures. Namely, they have increased memory demands as they employ two networks simultaneously. Additionally, for the case where the generator/segmentation network is taught to generate predictions that look similar to manual annotations, these manual annotations might have bias. Naturally, this bias might be introduced to the network even in non-GAN architectures, however the GAN architectures might exacerbate its presence in the output predictions. This is because the generator/segmentation network is motivated to generate outputs that look as similar as possible to manual annotations in order to foul a discrimination network. Regarding the application of a GAN architecture for the generation of additional training samples, extra care has to placed into the formulation of the weak supervision that allows their utilisation. This is because generally the scarcity of training data is accompanied with scarcity of testing data,

which can be used in order to verify if the additional GAN-generated training samples improve the segmentation accuracy. In any case however, for use-cases where the additional training data can be created synthetically in a more controlled manner via other software, these data should be prioritised over GAN-generated training data. Based on the above reasons, we determined that the adoption of a GAN architecture is probably not beneficial for the segmentation of undersampled tomograms in X-ray CT time-series. This is because the aforementioned CT time-series depict scientific data and it is desired that their segmentations would contain as less as possible any bias present in the manual annotations. Namely, our proposed approach [26, 27] is a denoising-segmentation network formed by two stacked encoder-decoder subnetworks, the first is dedicated to denoising and the second to segmentation. This is inspired by [315], where multiple encoder-decoder subnetworks are stacked and all are dedicated to segmentation, meaning that they are all individually supervised for segmentation. One of the main reasons, however we decided not to stack multiple subnetworks for segmentation, is exactly in order to minimise the introduction of bias potentially present in the manual annotations. Since the ground truth used for the supervision of the denoising subnetwork is not made manually, but it is a high SNR reconstruction, it does not have the potential to reinforce into the trained network any bias present in the manual segmentation annotations. Furthermore, due to the existence of specialised software for the generation of synthetic tomograms, the utilisation of a GAN architecture for the generation of additional training data was not preferred.

2.2.3.8 Recurrent neural networks (RNNs) for segmentation

While they are usually implemented for time-related problems, Recurrent Neural Networks (RNNs) have also been used to semantically segment images. In general the modules that comprise each RNN, instead of only generating the feature maps that each provides to subsequent modules for the generation of an output prediction, also generate internal module states in the form of feature maps. These internal module states jointly compose the internal state for the overall RNN network. When the RNN receives the next input it also receives the previously generated internal state. Then, the RNN uses both the input

and the internal state, to produce an output prediction and the new internal state of the RNN. This RNN internal state forms a type of memory that makes RNNs exceptionally useful for time-related problems, where datasets are time sequences.

In particular, due to their ability to model short/long term connection between pixels, they are worthy of investigation for segmentation applications. Namely, Visin *et al.* proposed their network ReSeg [357], which is a RNN network for semantic segmentation, based on the ReNet architecture [358]. ReSeg is stacked upon the predictions of a pre-trained VGG-16 [125] and it is composed of 3 ReSeg layer each with 4 Gate Recurrent Units [359] sweeping the image both horizontally and vertically in both directions. ReSeg layers are followed by upscaling layers that generate the final predictions with the same resolution as the input. Byeon *et al.* [360] also proposed a classification and segmentation network operated by 2D Long-Short Term Memory blocks (2D LSTMs) [361, 362]. Later Liang *et al.* [363] proposed Graph Long-Short Term Memory blocks (Graph LSTMs) for the task of segmentation. Specifically, Graph LSTMs use as nodes not neighboring pixels but neighboring superpixels, forming an undirected graph where the spatial relationships between the different superpixels are the edges of the graph. Here, superpixels are groupings of multiple neighbouring pixels created by unsupervised segmentation approaches (see Section 2.2). In their implementation the superpixels are calculated by [211] and the Graph LSTM layers process feature maps generated by Long *et al.*'s FCN [268]. The feature maps are also used to determine the node updating sequence for each image as input. The final feature maps after the Graph LSTMs are processed by a 1×1 convolution to produce the segmentation predictions.

Lastly, a new recurrent architecture has been proposed by Alom *et al.* [364] named Recurrent Residual U-Net. In it the researchers use recurrent convolutional layers proposed by Liang and Hu [365] to replace the convolutional layers present in U-Net, and also introduce residual learning in the U-Net's layers. They tested their approach for retina blood vessel segmentation, skin cancer lesion segmentation and lung segmentation and achieved better performance than U-Net or a residual version of U-Net with the same number of parameters. This is potentially achieved because the recurrent units accumulate features

relevant to segmentation through time.

In general, the RNN's ability to use memory feature maps encapsulated in their internal state, can allow them to be utilised for accurate segmentation. Nevertheless, by storing these additional feature maps, the memory demands of the resulting networks can easily increase by a large amount with additional layers. Because of this limitation and due to their relative small adoption for the task of semantic segmentation, motivated us to not considered the introduction of recurrent layers in our proposed approaches. Nevertheless, in the future as explained in Section 8.3.4, with the potential existence of sufficient 4D segmentation annotations, a CNN-RNN network could be proposed for the segmentation of time-resolved X-ray CTs. In the next Section we will explore the state-of-theart deep learning architecture for the semantic segmentation of 3D volumes or other high-dimensional data.

2.2.4 Semantic segmentation of 3D volumes or high-dimensional data with deep learning

Apart from 2D images, significant attempts have also been made to semantically segment 3D volumes or other high-dimensional data. These data are commonly used in medicine, biology and material science through the use of Computed Tomography [366, 367] and Magnetic Resonance [2, 3], for example. Multiple applications of the techniques presented in Sections 2.2.1 and 2.2.2 have been proposed for the use-case of 3D volumes or high-dimensional data. Naturally, these correspond to modified versions of the previous approaches that can handle 3D volumes or high-dimensional data and utilise their properties. Some of these methods use:

1. Contour-based segmentation methods (More in Section 2.2.1.2) [368, 369, 370, 371, 372, 373, 374]
2. Segmentation methods based on thresholding (More in Section 2.2.2.1) [375, 376]
3. Segmentation methods based on growing, splitting or merging regions (More in Section 2.2.2.2) [377, 378, 379, 380, 381]
4. Learning-based segmentation methods (More in Section 2.2.2.4) [382, 383, 384, 385,

386, 387, 388, 389]

These methods, their advantages and disadvantages are covered in Sections 2.2.1 and 2.2.2 and their respective subsections. For this reason this Section will not further elaborate on them and instead focus on 3D volume or high-dimensional data deep learning segmentation approaches. The challenge with deep learning approaches is that they can be very computationally depending and when utilised for the semantic segmentation of 3D or high-dimensional data, additional attention is required in order to develop sufficiently accurate approaches with manageable computational and memory demands.

While applied on 2D images (slices of volumes), Ciresan *et al.*'s approach [265] and Ronneberger *et al.*'s U-Net [156] are some of the first proposed deep learning approaches in the field of medicine and biology, and inspired a large number of follow-up methods. Particularly, Çiçek *et al.* [8] proposed a 3D version of U-Net, where they used U-Net's [156] structure and replaced the 2D convolutional layers with 3D convolutional layers. Furthermore, in order to deal with the increased number of parameters, the depth of the network is reduced by one layer in both the encoder and the decoder (see Figure 2.6 and 2.10). Additionally, batch normalisation [134] is introduced before the ReLU layers [252]. The most important addition in their architecture is the weighted softmax loss function [266, 267], allowing their network to train even with sparse annotations. By setting the weights of the unlabeled voxels to zero, the network can learn using only the labelled voxels and then be able to segment to the rest of volume. During experiments they performed both semi-automated and fully-automated segmentation. In the semi-automated segmentation experiments, a number of slices out of a volume are annotated by a user, and then the network can use these to train and finally annotate the rest of the volume. On the other hand in the fully-automated experiments, a completely annotated volume is used for training, allowing the network to then annotate other similar volumes. In both sets of experiments, the network produced good segmentation results.

Later, in approach similar to Luc *et al.* [353], Dong *et al.* [390] proposed generative adversarial networks for multi-organ segmentation, named U-Net-GANs. Specifically, a U-Net-GAN is composed of a U-Net as the generator and a fully convolutional net

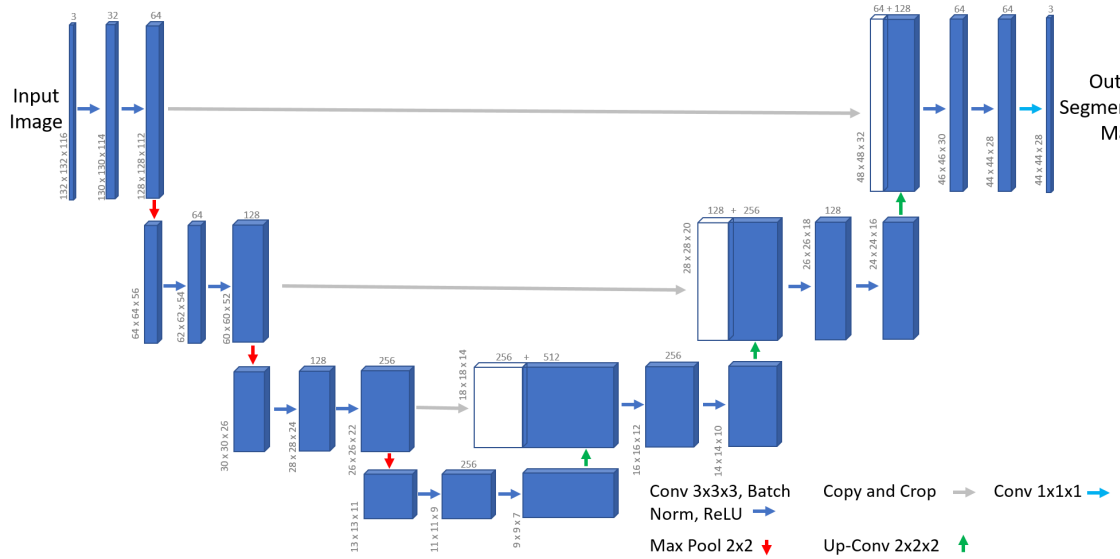


Figure 2.10: *3D U-net architecture. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of each box. The width, height and depth sizes are provided at the left of each box. White boxes represent copied feature maps. The arrows denote the different operations.*

as the discriminator. Namely the discriminator imposes a high-order structure to the predictions, similar to the one present in the ground truths. Their overall proposed approach uses 3 different U-Net-GANs for the accurate segmentation of multiple organs, connected in sophisticated manner based on nature of thorax CTs. All of the proposed U-Net-GANs predict using volumentric inputs; however, their first U-Net-GAN was 2.5D meaning that the channel dimension of a typical 2D U-Net is repurposed, and one of the spatial dimensions of the inputs is provided via the channel dimensions. This is possible since the input volumes are greyscale and there is a single input modality. This means that the channel dimension would otherwise be of size 1, and so it is possible one of the three spatial dimensions (width, height or depth) to be provided as the channel dimension and traditional 2D networks to be used for volume segmentation, hence 2.5D networks.

In an attempt to leverage the power of deep residual learning [129] for the 3D segmentation problem, Milletari *et al.* [391] proposed V-Net, their version for a 3D U-Net. In their architecture the 2D convolutional layers of U-Net are replaced with residual functions which perform 3D convolutions. Furthermore, instead of reducing the number of layers like Çiçek *et al.* the researchers decided to lower the number of feature maps in each layer by half compared to U-Net [156] and increase the kernel size from $3 \times 3 \times 3$ to $5 \times 5 \times$

5. Additionally, the max pooling layers are also replaced with convolutions, a technique which has recently shown similar results [392] with less complexity.

Apart from the changes in architecture, they also proposed a novel loss function based on the Dice coefficient [393]. Utilising residual learning, Chen *et al.* proposed VoxResNet [9], a ResNet-like [129] network with 3D convolutions. For their architecture, they implemented deep supervision [127] using predictions from feature maps at different scales and the final output is a fusion (summation) of these predictions and it is also supervised. Furthermore, using different modalities of MR images and the network's segmentation predictions, they further refine them using an auto-context algorithm [394].

Christ *et al.* [395] later proposed the use of *stacked* or *cascaded* fully convolutional networks, which they implemented using 2 *stacked* U-Nets, to segment volume slices which are then concatenated. Moreover, their approach uses 3D Conditional Random Fields (CRFs) [250] for further refinement of their predictions. Moreover, Wong *et al.* [396] proposed a residual 3D U-Net architecture with deep supervision [127] for feature maps of different scales in the decoder. Their network trained with their proposed exponential logarithmic loss is able to accurately segment 3D volumes even with highly unbalanced classes. Finally, Myronenko [397], trying to address the problem of limited annotations, proposed the addition of a second parallel decoder that receives the feature maps from the encoder and tries to reconstruct the input. In practice, this a variational auto-encoder [398] branch, which regularises the layers of the shared encoder and imposes additional constraints on them. Moreover, they replaced the use of Batch normalisation with Group Normalisation [399], which it proven to introduce less error when the batch size is small.

Chen *et al.* [400] in an effort to accurately segment anisotropic volume datasets, proposed the combination of convolutional and a recurrent neural networks in the same framework. Namely, this is because in anisotropic datasets a voxel's scale/resolution (expressed in units of length like cm) along 2 out of 3 axis might be the same (in the xy plane), while along the 3rd (z axis) might be different. Traditional 3D convolutional networks are not inherently designed for such datasets, since they assume that each voxel depicts a physical area that has the shape of a cube in real-world, meaning that each voxel's

physical dimensions (width, height, depth expressed with SI units) are equal. This is because traditional CNNs that use 3D convolutional layers attempt to be translationally invariant, along all three dimensions of the input. Chen *et al.*'s approach, splits the task of 3D segmentation into two parts. In the first, segmentation-worthy features are extracted from different 2D slices (xy slices) of the input volume along the z-$axis$. Then in the second part, these features are used to produce 3D segmentation-related features that eventually generate the 3D predictions. Similar to Christ *et al.* [395], for the convolutional part they *stack* multiple 2D U-Nets, which they name kU-Nets, also experimenting with way feature maps are forwarded from one to the next. For the second part, a new recurrent architecture is introduced called Bi-Directional Convolutional LSTM (BDC-LSTM), based on the Convolutional LSTM by Shi *et al.* [401]. Their overall recurrent network is a combination of multiple BDC-LSTMs and in conjunction with the kU-Net it forms their final architecture.

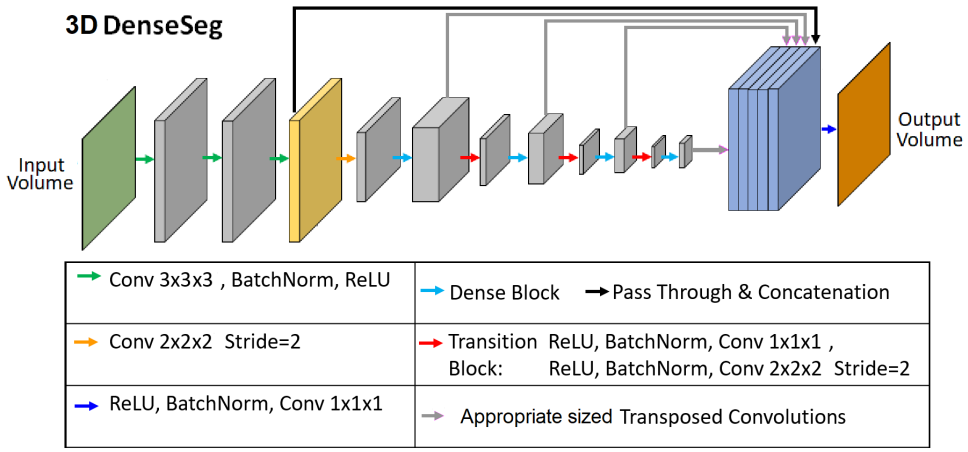
Later, Bui *et al.* and Yu *et al.* influenced by Huang *et al.*'s DenseNet [143], proposed 3D DenseSeg [10] and DenseVoxNet [402] respectively. Similar to the DenseNet, their architectures are comprised of some initial convolutions and then sequences of densely connected modules separated by transitional modules. The task of the transitional modules is to reduce the number of feature maps with a $1 \times 1 \times 1$ convolution and reduce their scale with the a pooling operations. In DenseVoxNet [402], there are 2 blocks of densely connected modules, separated by a transitional module. Each block contains 5 densely connected modules, and each module is a $3 \times 3 \times 3$ convolution, with ReLU and Batch normalisation. From the 2 blocks, two predictions are generated with the one one only used for additional supervision.

In 3D DenseSeg [10], there 4 dense blocks separated by 3 transition blocks. Each dense block has 4 densely connected modules, and each module has a $3 \times 3 \times 3$ convolution, with ReLU and Batch normalisation, and also an initial $1 \times 1 \times 1$ "bottleneck" convolution, like in the original DenseNet. Feature maps collected at different scales are upscaled, concatenated and processed by a final $1 \times 1 \times 1$ convolution that generates the 3D segmentation predictions. Likewise, Gibson *et al.* also proposed DenseVNet [403] with 3 consecutive

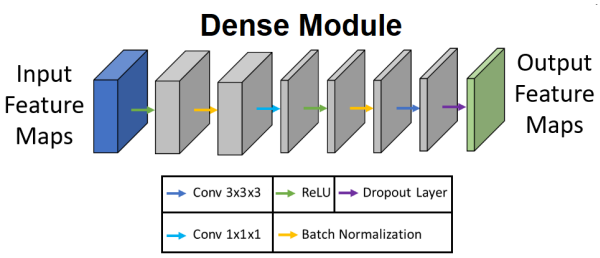
dense blocks that creates feature maps at 3 different scales. These are then upscaled, summed with each other and also summed with an upscaled spatial prior to adjusts the logit maps, since medical datasets are anatomically standard.

Li *et al.* [404] also used densely connected layers. Their approach is comprised of 2D and 3D U-Net subnetworks with DenseNet as the encoder, which they name DenseUNets. Namely, a 3D input is split into multiple 2D inputs and each 2D input is comprised by 3 slices/channels. For example, an $224 \times 224 \times 12$ input is split into 12, $224 \times 224 \times 3$ 2D inputs. Similar to the 2.5D U-Net-GANs mentioned earlier, these "2D" DenseUNets are in fact 2.5D, meaning that the channel dimension is repurposed for one of the spatial dimensions of the inputs. Again this is possible because the inputs are greyscale volumes with a single modality. The predictions of these "2D" DenseUNets have the same size as their inputs with three output channels, and after the prediction concatenation they form a 3D segmentation prediction. This prediction along with the original 3D input are handled by a 3D DenseUNet, which generates a second 3D prediction. The first and the second 3D predictions are summed and processed by an auto-context algorithm [394] to produce the final output. The reason they proposed the use of both 2D and 3D DenseUNet, is in order to mitigate the high computational cost of a 3D U-Net with a deep DenseNet as its encoder. Thusly, a 2D U-Net with a deep DenseNet encoder is used to extract intra-slice features and a 3D U-Net with a shallower DenseNet encoder extracts the inter-slice features.

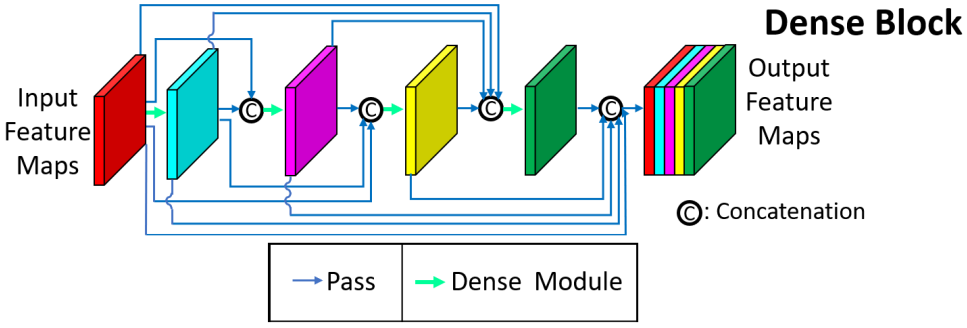
Similarly, Pelt and Sethian [337] also proposed a 2.5D network for the segmentation of 3D volumes. In particular, their network segments multiple slices of the input volume, and by concatenating the outputs, the final 3D predictions are formed. Again information regarding the 3rd dimension is offered via the channel dimension, however their approach segments only the middle slice. During inference a sliding window with stride of 1 along the 3rd dimension is used in order to generate multiple overlapping 2D inputs. Their networks are effectively a single large sequence of densely connected modules, and each module is comprised of multiple parallel dilated convolutional layers each with different dilation rates. An example with 3 densely connected modules (depth=3) each with two parallel



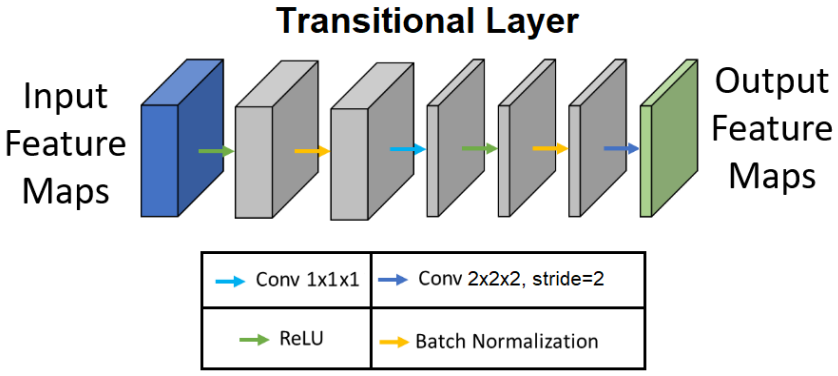
(a)



(b)



(c)



(d)

Figure 2.11: *A graphical representation of the 3D DenseSeg, Dense Module, Dense Block and Transitional layer. (**a**) The 3D DenseSeg architecture, (**b**) The building block of Dense Blocks, the Dense Module, (**c**) A Dense Block comprised of 4 Dense Modules, (**d**) A Transitional layer used after the a Dense Block.*

dilated convolutional layers (width=2) is presented in Figure 2.12. Their architecture is discussed in more detail in Section 5.3.2. We also propose a DenseNet-based architecture named 3D DenseUSeg that we discuss it further on chapter 5. Our architecture expands upon the 3D DenseSeg [10] architecture, by adding a decoder that manages to concatenate the feature maps in skip connections and not sum like in [404]. This is observed to improve the performance, however in order deal with the large numbers of feature maps both the transpose convolutions and plain convolutions of the decoder reduce the number of feature maps. Additionally, the growth rate of each dense block in our architecture is increased to $g = 64$ compared to 3D DenseSeg' original growth rate of $g = 16$ in order to further improve the model's accuracy.

Finally on chapter 5, we also propose our Stacked-DenseUSeg architecture that performs both the tasks of denoising (special case of SR) and semantic segmentation on undersampled (low-dose or sparse-angle) tomograms. Our architecture is designed by "stacking" sequentially two of our DenseUSegs, with the first performing the task of denoising the undersampled tomograms and the second utilising feature maps from the first in order to offer accurate semantic segmentation predictions. The two DenseUSeg subnetworks are joined similar to how the hourglass networks are joined in Newell *et al.*'s work [315]. We designed this architecture after observing performance of our DenseUSeg in both the task of denoising and segmentation on undersampled tomograms (see Section 5.5). Our Stacked-DenseUSeg is trained in an end-to-end fashion, which means that our network can offer both a denoising and a segmentation output within a single forward pass.

In the next Section we will provide more information regarding the acquisition of CT scans and how they are reconstructed in order to obtain their 3D representation.

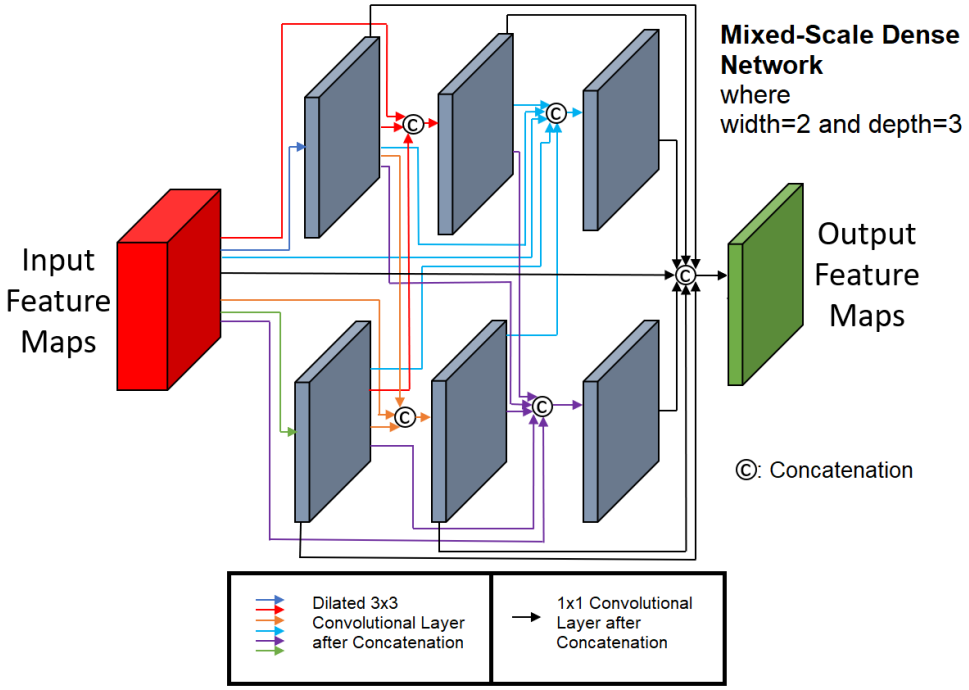


Figure 2.12: *A mixed-scale dense network (MSD) layout where the network's depth=3 and its width=2.*

2.3 CT acquisition and reconstruction

2.3.1 Micro X-ray computed tomography

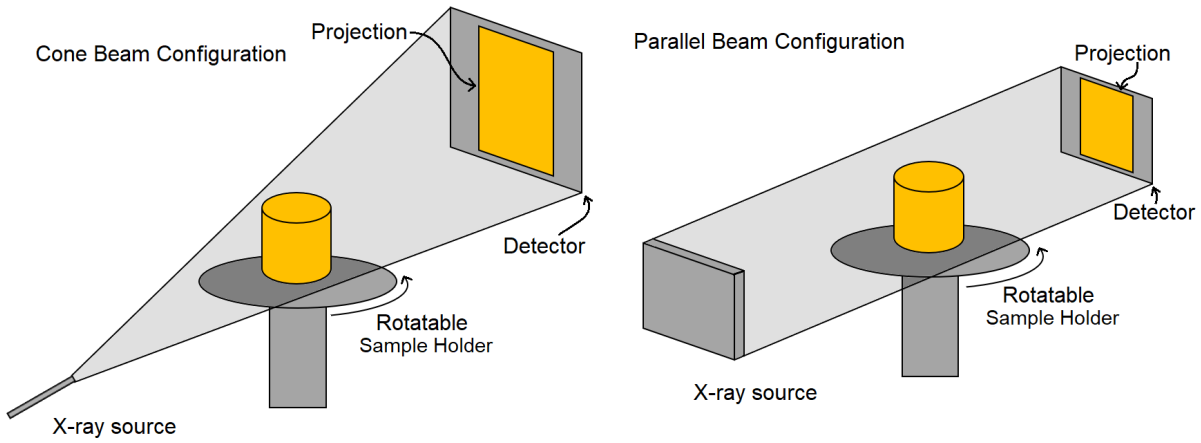
From the first mathematical description of the Radon transformation [405], [406] and the first commercial use of a Computed Tomography (CT) scanner in 1967 [1], Computed Tomography has become a prevalent tool and aid for multiple generations of scientists and commercial users in a plethora of scientific areas. These range from medicine, pharmacology and biology, to geology, material Science, mechanical and aerospace engineering, and multiple scientific discoveries over the last decades have been achieved by carefully and appropriately applying CT to reveal the inner structure and workings of an endless number of samples. Simultaneously, through the commercialisation of CT and its introduction to vital parts of the economy such as Medicine and Manufacturing, what it is described as quality of living advanced to new heights, either by saving countless human lives with accurate medical diagnosis or by ensuring the safety and upholding to certain standards of a myriad of different tools and equipment.

Through recent decades, as the technology around Computed Tomography was improving,

multiple approaches for the capture of projections were attempted using different types of imaging sources (electrons, positrons, X-rays, visible light). However, for the capture of the intricate structure of the samples in the micro meter scale, while also achieving high spatial pixel resolution, the most prevalent imaging source is X-ray light. This is due to the low attenuation of the X-ray light, as it passes through the sample. Naturally, the attenuation of the X-ray light also depends on the light's flux emitted by the source. High flux X-ray light attenuate less, and can penetrate deeper into the sample resulting in more accurate imaging results. However, this can potentially cause damage to the sample, extending the time needed to collect the data due to multiple reattempts.

Generally, there are two main X-ray configuration geometries, for the collection of projections presented in Figure 2.13. Cone beam configuration geometries are typically used for larger samples, where the source can be referred as a point relative to the sample and it is a common configuration for laboratory CT machines. As is shown in Figure 2.13(**a**), the rays of X-ray light, spread in a conical shape, penetrating the sample and reaching the detector, where a large projection relative to the sample is captured. On the other hand as is shown in Figure 2.13(**b**), in parallel beam configuration geometries that are common to synchrotron CT facilities, the rays of X-ray light are produced parallel to each other from the source, penetrate the sample and produce in the detector a one-to-one in scale projection relative to the sample. Due to this fact, later reconstruction of the collected projections can be achieved in a relatively easy fashion compared to the cone beam configuration geometry, since there is no need for extra care regarding the appropriate scale readjustment.

Additionally, since in the cone beam configuration the rays of X-ray light spread and separate slowly between each other as they travel towards the detector, the light density on the part of the sample closer to the source is higher compared to the part of the sample closer to the detector, which is translated as a partial loss of spatial information. Therefore, when the parallel configuration is possible, it is preferred since it offers more accurate projections, and that leads to more accurate reconstruction of the tomograms in a volume format. As it can be seen in Figure 2.13, the set of projections that comprise a tomogram



(a) *Cone Bean Configuration* **(b)** *Parallel Bean Configuration*

Figure 2.13: *Most common beam configurations for X-ray computed tomography. (**a**), cone beam configuration where the X-ray source is produced from a point relative to the sample and the X-rays spread in a conical shape, go through the sample and to the detector, producing the projection. (**b**) parallel beam configuration where the X-ray source is produced from a plane relative to the sample and the X-rays are parallel to each other, go through the sample and to the detector, producing the projection. The multiple projections needed for the tomography are produced by rotating the sample which is placed on a rotatable sample holder. As can be seen the projections in a cone beam configuration are larger in scale than the sample. In both configurations the sample is the one rotating, but the rotation of the source is more common in the cone beam configurations, mainly in medical uses where the patient (sample) stands still and X-ray source and the detector move in circular path around them.*

are captured by rotating the sample using a rotatable sample holder. Naturally, this is not the only way X-ray CT collection is performed using these configurations. For instance, in medical uses that are performed with a cone beam configuration, it is more common for the patient (sample) to be stationary when both the source and the detector rotate around them in a circular fashion. This is due to the technical ease of keeping the patient stationary, rather than attempting to accurately rotate them to capture the multiple projections. In conclusion, regardless of the geometry, X-ray computed tomography can be summarised as follows:

1. The sample that would undergo the X-ray tomography is appropriately processed (e.g. sliced in a precise way, frozen to a specific temperature, etc.), then placed and secured on the rotatable sample holder.
2. X-ray light passes through the sample and it is collected by the detector. The detector then calculates the absorption coefficient and synthesises a 2D projection for the current position of the sample.

3. Multiple 2D projections of the sample are captured, while the sample holder slowly rotates. Depending on the speed at which the detector registers a projection, the speed of the sample holder is appropriately set to ensure the collection of a specific number of projections. The slower the sample holder rotates the sample, the higher the number of overall collected projections would be .
4. In parallel beam configuration, the sample completes a rotation of 180° degrees around its height-axis; that allows the reconstruction of the 3D volume of the entire sample. On the other hand, in cone beam configuration the rotation has to be over 360 degrees. In both configurations, in cases where the sample's size or geometry prohibits its rotation or use with existing capturing apparatus, a smaller degree sweep can be used. However, this would most likely introduce artifacts into the final reconstruction.
5. Using the collected 2D projections, a reconstruction algorithm is applied, performing an inverse radon transformation, which synthesises a 3D volume that depicts the inner structure of the sample.

For a detector of size $H \times W$, where H is the height in pixel count and W is the width respectively, the final 3D volume would be of size $H \times W \times W$ with H and W being the pixel count of the height and width respectively, and W the pixel count of the depth of the volume.

Provided that the tomogram has been collected using the appropriate settings, conditions and operational guidelines, the final Signal-to-Noise Ratio (SNR) and imaged quality of its 3D reconstruction, is determined by the number of projections and reconstruction algorithm selected. Additionally, depending on the apparatus that is used to collect the tomogram, certain imaging artifacts may appear in the 3D reconstruction, such as ray artifacts, rotating artifacts, Poisson noise, zingers, etc. which are simulated upon reconstructions of the Shepp Logan phantom [407] in Figure 2.14. It is also important to note that during this process, the rotation axis of the sample holder might not necessarily align with the central pixel column of the projections. In more versatile setups, where researchers may change the detector between experiments and the imaged samples are in

micro and nano scales, the precise alignment of the source, the rotation axis of the sample holder and the detector, can not be guaranteed. To solve this issue, specific algorithms [408] are being used to calculate, with sub-pixel accuracy, the pixel column in the projections which the rotation axis passes through. This information is then appropriately used by the corresponding reconstruction algorithms, to ensure the accurate reconstruction of the tomograms.

Once a tomogram's projections are captured and the location of the rotation axis is known, reconstruction algorithms can be used in order to obtain the 3D representation of the tomogram, which is referenced throughout this thesis as the tomogram's reconstruction. The most common way to obtain a tomogram's reconstruction is via the use of Filtered Back Projection (FBP) [409] that employs an inverse Radon transformation [405, 406]. Nevertheless algebraic iterative reconstruction algorithms have been proposed for the calculation of a tomogram's reconstruction. Using the same detector example as before, the size of the final reconstruction for a detector with size $H \times W$ will be $H \times W \times W$. This reconstruction can be viewed as collection of H different $W \times W$ slices, which a reconstruction algorithm can reconstruct independently. Naturally for each of these slices, their projection is a row of $1 \times W$ pixels. Let's denote a slice in the reconstruction as x and one of its projections as b_1. Each of these W pixels in the b_1, is created by projecting (weightily summing) some of the $W \times W$ pixels of x. This means that b_1 can be calculated as a matrix multiplication of $\boldsymbol{x}$, which is the vectorised version of x, with a matrix A_1 that dictates how much each of the $W \times W$ pixels of x influence each of the pixels of b_1. This can be written as $A_1\boldsymbol{x} = b_1$. Naturally, more than one projections is captured and if $\boldsymbol{b}$ is the vectorised concatenation of all projection $b_1, b_2, ...$, then the projection problem of the slice x can be written as:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.6}$$

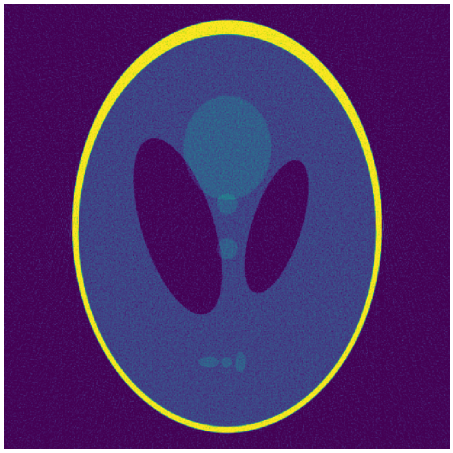
where $\boldsymbol{A}$ is a matrix with size $P \cdot W \times W^2$ with P being the number of projections. Based on this, the reconstruction problem of slice x is the inverse of problem of equation 2.6, which is:

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} \tag{2.7}$$

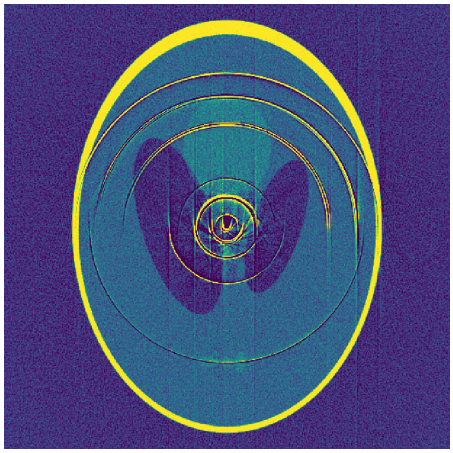
By solving equation 2.7, for every slice in a tomogram, the tomogram reconstruction can be obtained algebraically. Nevertheless, obtaining $\boldsymbol{A}^{-1}$ is not always possible, especially in ill-posed problems where only a small number of projections is collected. For these reasons, the proposed algebraic reconstruction approaches solve equation 2.7 with the use of iterative algorithms. Many iterative reconstruction approaches have been proposed so far like the Algebraic Reconstruction Technique (ART) [410], Simultaneous Algebraic Reconstruction Technique (SART) [411], the Simultaneous Iterative Reconstruction Technique (SIRT) [412] and the Conjugate Gradient Least Squares (CGLS) method [413]. Namely, the last one is also applied for the reconstruction of the undersampled tomograms used in chapters 5 and 6. The method utilises the conjugate gradient least squares method [414] in order to solve equation 2.7 for all slices in a tomogram. Below, the pseudocode 1 describes the iterative CGLS algorithm for the calculation of the reconstruction of one slice.

Algorithm 1: Conjugate Gradient Least Squares iterative solution of equation 2.7

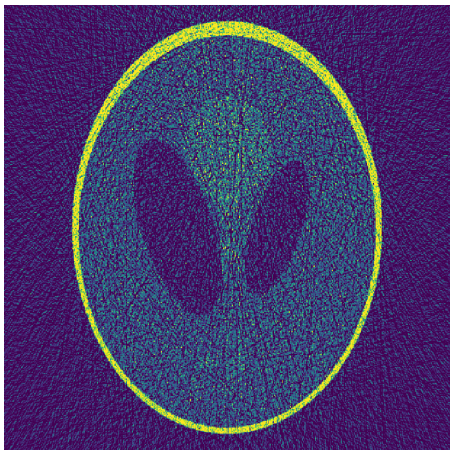
Input: $\boldsymbol{A}$, $\boldsymbol{b}$
Result: $\boldsymbol{x}_i$ after i iterations
// Calculate the initial residual $\boldsymbol{r}_0$ and conjugate vector $\boldsymbol{p}_0$
$\boldsymbol{x}_0 = 0$
$\boldsymbol{r}_0 = \boldsymbol{b}$
$\boldsymbol{p}_0 = \boldsymbol{A}^T \boldsymbol{r}_0$
for $k \leftarrow 0$ *to* i **do**
  // Iterative approach for the estimation of $\boldsymbol{x}_i$
  $\psi_k = \frac{\left\|\boldsymbol{A}^T \boldsymbol{r}_k\right\|^2}{\|\boldsymbol{A}\boldsymbol{p}_k\|^2}$
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \psi_k \boldsymbol{p}_k$
  $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \psi_k \boldsymbol{A}\boldsymbol{p}_k$
  $\beta_k = \frac{\left\|\boldsymbol{A}^T \boldsymbol{r}_{k+1}\right\|^2}{\left\|\boldsymbol{A}^T \boldsymbol{r}_k\right\|^2}$
  $\boldsymbol{p}_{k+1} = \boldsymbol{A}^T \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k$
end



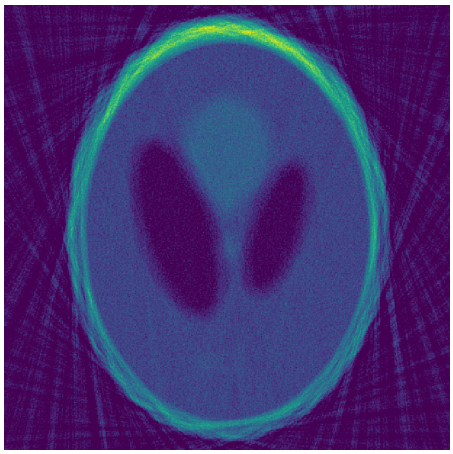
(a) *Axial View of the reconstruction of the Shepp Logan phantom with Poisson noise.*



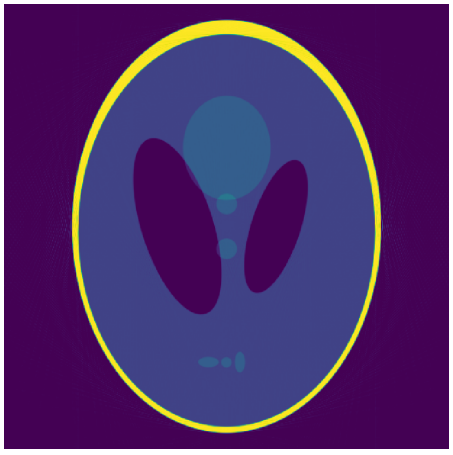
(b) *Axial View of the reconstruction of the Shepp Logan phantom with rotating artifacts.*



(c) *Axial View of the reconstruction of the Shepp Logan phantom with zinger artifacts.*



(d) *Axial View of the reconstruction of the Shepp Logan phantom with ray artifacts.*



(e) *Axial View of the reconstruction of the Shepp Logan phantom without noise or artifacts.*

Figure 2.14: *Axial view of the 3D reconstruction of the Shepp Logan phantom [407]. (**a**) Reconstruction with Poisson noise, (**b**) Reconstruction with rotating artifacts, (**c**) Reconstruction with zinger artifacts, (**d**) Reconstruction with ray artifacts, (**e**) Reconstruction without any noise or artifacts. The former axial views and artifacts are simulated using the TomoPhantom software [23].*

2.3.2 Sinogram space

In addition to the representation of a tomogram as a series of projections, it can also be represented as a series of sinograms. Each sinogram is created by stacking rows of pixels that correspond to a specific height in the sample from all the different projections, starting from the row captured first and ending with the row captured last, Figures 2.15.

For better understanding, let's denote the size of each projection as $y \times x$, where y and x are the projection's height and width respectively. Moreover, let's denote the number of projections as θ, since it's also the number of the different angles from which projections have been taken. After the transition from the projection space to the sinogram space, there will be y sinograms in total, with each of them having size $\theta \times x$. This representation receives its name from the sinusoidal lines present in the sinograms, a product of the trace that different internal components create as they are rotated with the sample (Figure 2.16). From the above, we can infer that tomograms with reduced number of projections, can be expressed as undersampled tomograms, which also have a small number of pixel rows in their sinograms. Following this point, we will refer to the tomograms with small number of projections as being undersampled.

When studying high-speed processes using time-series tomography, it is common either before, after or before and after the acquisition of a series of high-speed, undersampled tomograms to capture slow-speed, highly-sampled tomograms to help resolve spatial details and understand the system as a whole. We will refer to these tomograms as *highly-sampled tomograms*. The tomogram reconstruction of the entire time-series is shown in Figure 2.17, including both the undersampled tomograms and the potential highly-sampled ones, which may be captured either at the beginning of the collection, the end of the collection or both. It will be the aim of the approaches described in the later chapters, to use these highly-sampled tomograms to train convolutional neural network architectures (CNNs). This is because these tomograms can be reconstructed either with a high SNR or a low SNR. Their high SNR reconstructions can be used to perform an accurate manual segmentation of the tomograms and provide a "golden standard ground truth" for segmentation CNN architectures. Additionally, their high SNR reconstructions may also

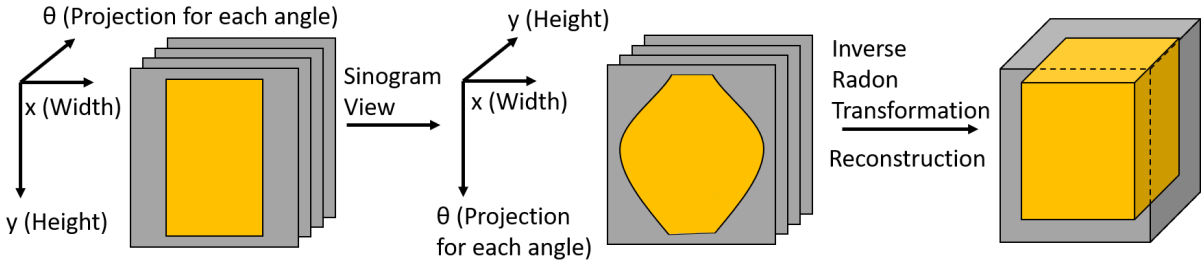


Figure 2.15: *A tomogram's projections can be seen as sinograms, each of them corresponding to specific pixel height in the tomogram. By performing inverse radon transformation in each of the sinograms and stacking them along the pixel height axis, the tomogram's reconstruction is obtained.*

be used as ground truths for denoising CNNs that would attempt to increase the SNR of noisy or low-quality tomogram reconstructions. In both segmentation and denoising CNN architectures, the input to the networks during training should be the corresponding low-quality tomogram reconstructions. This is where the low SNR reconstructions of the highly-sampled tomograms are used. By reconstructing them with low SNR, the results imitate the case in which the tomograms are undersampled in terms of projections. In conclusion, using the highly-sampled tomograms' reconstructions and annotations as inputs and/or ground truths, CNNs can be trained. These networks after training would have the ability to process reconstructions of genuine undersampled tomograms, and offer higher SNR version of them and/or their semantic segmentation.

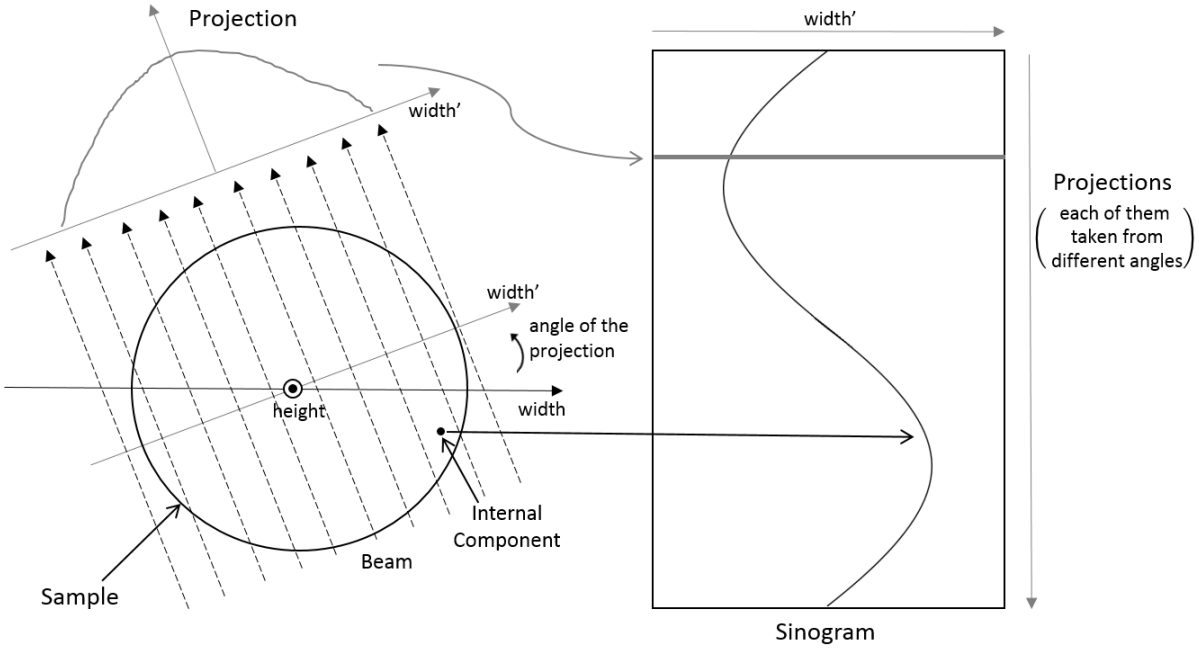


Figure 2.16: *The formation of a sinogram. Internal Components create sinusoidal lines in the sinogram as they rotate with the sample.*

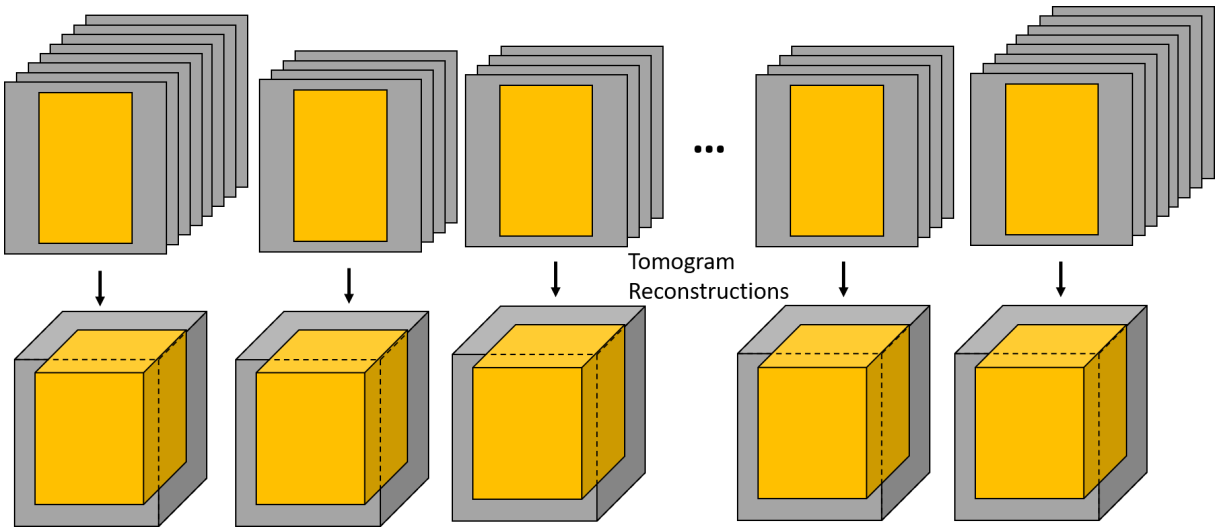


Figure 2.17: *By performing the reconstruction of the component tomograms of time-series of tomograms and aggregating them a 4D dataset is formed.*

2.4 Summary

In this chapter, we presented a literature review regarding both the tasks of super-resolution (SR) and semantic segmentation. For the task of SR, we presented 4 different categories of methods, frequency-domain-based methods, interpolation-based methods, regularisation-based methods and finally learning-based methods. As it is explained, frequency-domain-based methods and interpolation-based methods typically do not lead to very robust and accurate solutions, while regularisation-based methods can be very computationally expensive, which makes them not very suitable for applications like the sinogram upscaling of multiple undersampled tomograms in X-ray CT time-series. Learning-based methods, on the other hand, can lead to highly accurate and robust approaches with the main computational cost occurring during their training. As described state-of-the-art learning-based SR methods utilise deep neural networks and in particular, CNNs. Motivated by this fact our proposed approach for sinogram upscaling, that is described in chapter 4, utilises a CNN architecture. However, our approach utilises a lightweight network structure, since fast training and inference times are significant. This is because our approach is designed to be frequently retrained, in order to be easily implemented on new X-ray CT time-series.

Furthermore this chapter presents a literature review on segmentation approaches, focusing more on semantic segmentation methods. As described segmentation methods can be split to methods design to detect edges and discontinuities within the input and to methods design to define continuous regions with internal similarities. Based on the presented approaches, the most robust solutions can be found utilising either growing, splitting and merging regions or learning with an existing dataset. However, methods learning-based have the additional advantage that once their training session is over, usually they are able to provide segmentation predictions quicker than methods utilising growing, splitting and merging regions. Based on this fact, more focus was brought on learning-based segmentation approaches, where most state-of-the-art methods apply deep learning. After exploring various network architectures designed for semantic segmentation, it became apparent that encoder-decoder architectures are the more advantageous for the development

of new approaches for the segmentation of undersampled tomograms. This is because they have the ability to extract multi-scale information, while having small memory demands due to the reduction of their feature maps' resolution in deeper layers. This characteristic, makes them especially advantageous for the design of 3D CNN applications, where memory demands are already high due to the high dimensionality of the input and the subsequent feature maps generated by the network. Motivated by this fact, we designed an encoder-decoder architecture for the segmentation of undersampled tomograms that is presented in chapter 5. Furthermore, in chapters 5 and 6, we are going to explore a denoising-segmentation architecture that is created by stacking two of our proposed encoder-decoder networks, inspired by an approach proposed by Newell *et al.* [315].

Finally, in the end of this chapter the acquisition and reconstruction of micro X-ray CTs is explored, providing further details and explanations for the terminology used throughout this chapter (e.g. sinograms). In the following chapter we will provide information regarding micro X-ray CT time-series and their challenges towards both the increase of their reconstructions' SNR and the semantic segmentation of their reconstructions. Furthermore, it will present more details regarding the datasets used in this thesis.

Chapter 3

Time-resolved undersampled X-ray CTs

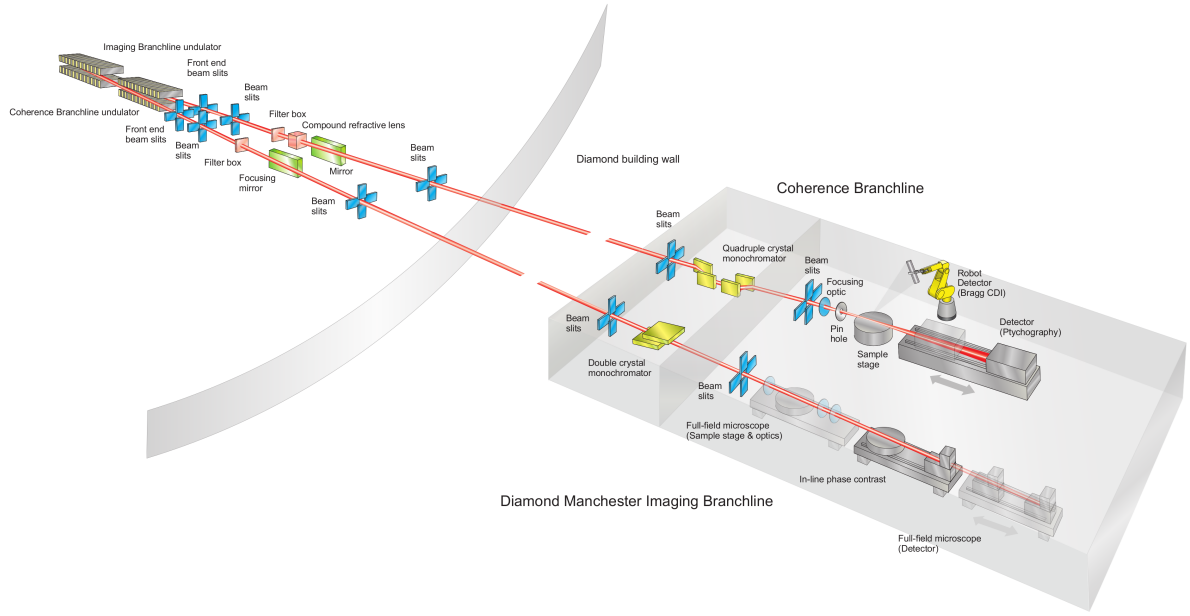
3.1 Time-series X-ray micro CT datasets

With recent advances in X-ray source technology, reducing collection times, a new approach of performing X-ray computed tomography has been utilised by researchers and industry users, the time-resolved CT data collection. In an attempt to obtain the temporal changes of a given sample's structure and geometry, multiple tomograms are captured at discrete intervals of time. By collecting these time-series of tomograms, valuable temporal information is captured with volumetric context. This allows capture of a time-related events taking place on the imaged sample, such as corrosion, deformation or failure of materials. Since they contain both temporal and volumetric information, these datasets are 4D datasets. Naturally, these types of collections can be performed on events that develop slowly or quasi-static processes where speed is not an issue. Nonetheless researchers are also attempting to use them on fast-occurring temporal events. This is achieved by either reducing the exposure time of each projection or by significantly reducing the number of projections collected for each of the time-series tomograms, with the latter usually preferred over the first since the visual quality of underexposed projections may be unable to accurately reconstruct the tomograms despite the large number of projections. However, this comes at a cost. During the application of a reconstruction algorithm that produces the 3D volume representation of the tomograms, the high noise of the underexposed projections or the lower number of projections sets an ill-posed re-

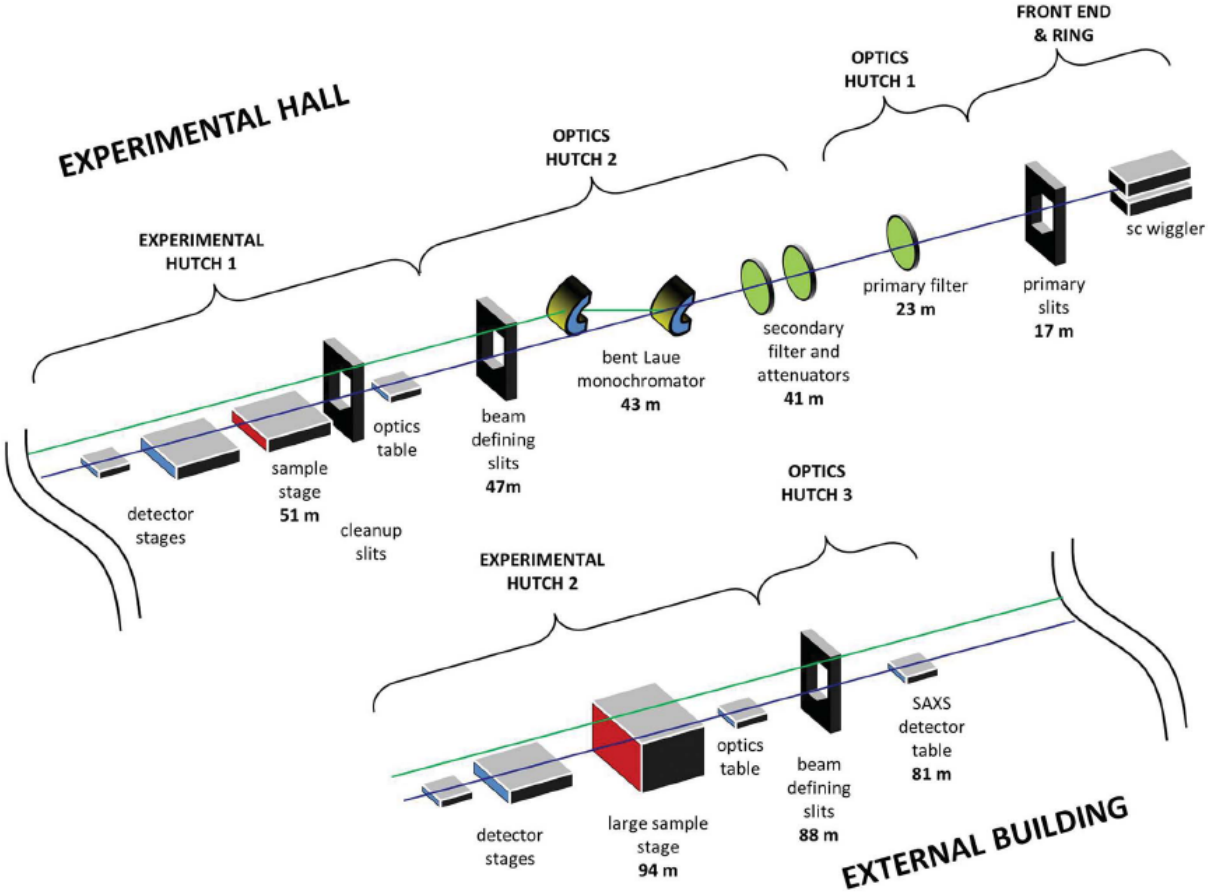
construction problem. This eventually leads to poorer quality reconstructions, where the level of the SNR is very low and imaging artifacts, such as ray artifacts are prevalent; making the clear identification and detection of their internal components exceedingly difficult. This can sometimes be addressed using iterative reconstruction methods such as the Simultaneous Algebraic Reconstruction Technique (SART) [411], the Simultaneous Iterative Reconstruction Technique (SIRT) [412] and the Conjugate Gradient Least Squares (CGLS) method [413]. Unfortunately though, these are often incredibly computationally expensive and the quality of their reconstructions can fluctuate greatly depending on a number of projections and the nature of the sample.

The datasets used in the later chapters have been produced at Diamond Light Source. They are micro-computed tomographic time-resolved datasets captured in the Diamond Manchester Imaging Branchline (I13-2) (see Figure 3.1(**a**)). In addition to beamline I13-2, time-resolved tomographic data are also produced from Diamond's I12 Joint Engineering, Environmental and Processing (JEEP) beamline (see Figure 3.1(**b**)). The I12 and I13-2 are located on short straight sections of the Diamond storage ring (see Figure 3.2) and they are powered by the 3 GeV (eV: Electronvolt $\approx 1.6\ 10^{-19}$J) source of light produced from Diamond's Storage Ring. The X-ray light is produced when the accelerated electrons in the polygonal-shaped Storage Ring, change direction in order to continue their course within the synchrotron. The electrons are forced to change direction at each of the synchrotron's angles, however due to the change in momentum powerful light in the X-ray frequencies is emitted in the opposite direction. The light then travels through the various beamlines which are tangential to the corresponding synchrotron angles, and in the case of a ongoing experiment, the light passes through the sample and to the detector, allowing for the collection of tomographic datasets. The collection of the aforementioned dataset is executed as follows:

1. Similar to the process described in Section 2.3.1 the sample is appropriately processed, and placed and secured on a rotatable sample holder, which is not yet in line between the source and that detector.
2. The X-ray source is covered and a number of projections is collected, that creates



(a) *The Diamond Manchester Imaging Branchline (I13-2) schematic.*



(b) *Diamond's I12 Joint Engineering, Environmental and Processing (JEEP) beamline schematic.*

Figure 3.1: *(**a**) The Diamond Manchester Imaging Branchline (I13-2) schematic and (**b**) Diamond's I12 Joint Engineering, Environmental and Processing (JEEP) beamline.*

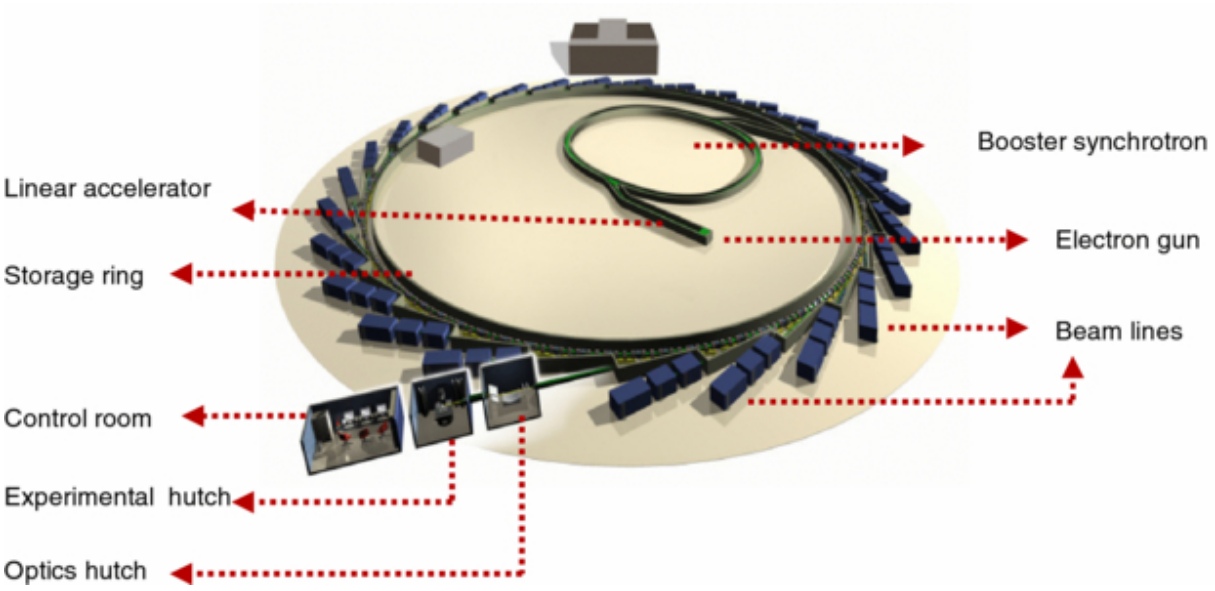


Figure 3.2: *The Diamond Light Source storage ring. In it an electron gun fires electrons which are then accelerated by the linear accelerator and the booster synchrotron before they enter the main synchrotron. At the points where the electrons velocity is shifted by magnets to stay within the synchrotron, light is emitted in the tangent. The light continues its path through the various beamlines that as it can be seen are tangential to the storage ring. Optical instruments adjust the light in the optical hutches which is then used for experiments in the experimental hutches. In each of the beamlines, control rooms the data from the experiments are monitored and/or processed.*

references for the most dark conditions possible within the collection area. These are the dark field projections.

3. The X-ray source is fully uncovered and a number of projections is collected, that creates references for the most bright conditions possible within the collection area. These are the flat field projections.
4. X-ray light passes through the sample and is collected by the detector. The detector then calculates the absorption coefficient and synthesises a 2D projection for the current position of the sample.
5. Again, similar to the process described in Section 2.3.1 multiple 2D projections of the sample are captured, and the sample holder rotation speed dictates the number of the overall collected projections.
6. The sample completes a rotation of 180° degrees around its height-axis. By doing that the tomogram acquires information for the complete angle space $[0, \pi)$ (due to symmetry the range $[\pi, 2\pi)$ is not needed). After the inverse Radon transformation (reconstruction), this provides information for the complete depth space $[0, 1)$ (and

it is up to 1 since the depth space is normalised). In the end, this allows the entire 3D volume reconstruction of the sample.

7. For the collection of a time-series of tomograms, steps 2-6 are repeated multiple times for precise time intervals. It is common at the start, or after the time-series collection, to obtain tomograms with a high number of projections (by slowly rotating the sample holder), which are captured outside the time critical portion of the collection. While the rest of the time series tomograms are collected with a small number of projections (quickly moving sample holder).
8. Using the dark and flat field projections, every tomogram in the time series, including both the highly-sampled tomograms and the undersampled tomograms of the series are normalised. The normalisation is independent between each of the pixels, and provides the removal of false attenuation due to other conditions, which are not caused by the sample.
9. As described in Section 2.3.1, for each of the tomograms a reconstruction algorithm is applied, generating a 3D volume that depicts the inner structure of the sample, during the time that the corresponding tomogram was collected (Figure 2.17. The aggregation of a time-series of reconstructions forms a 4D dataset.

The Diamond Manchester Imaging Branchline (I13-2) can perform real space imaging and tomography in the 8-30 keV energy range, and the imaging can be conducted with both pink-beam and monochromatic light. Pink light includes all synchrotron's available wavelengths and is incredibly bright, while monochromatic light represents a single wavelength which is chosen accordingly to the sample. The luminous flux of I13-2 is over a million times higher than that of a laboratory X-ray source or "X-ray tube". On the other hand, I12 using its 4.2 Tesla superconducting wiggler can provide polychromatic ("white beam") and monochromatic X-rays in the 53–150 keV energy range, allowing for the performance of static and time-resolved radiography and tomography. The sufficient beam energy in both beamlines, combined with the beams' large size enable good light penetration even in large or dense samples. It also permits the study of materials without the issue of an unacceptable beam attenuation or the need to reduce the size of the samples. The design,

layout and operational capabilities of the I12 JEEP beamline at Diamond Light Source are articulately described by Drakopoulos *et al.* [415]. Similar information about I13-2 can be found on [416], [417]. We want to acknowledge Diamond Light Source for the time on beamline I13-2 under the proposal mt9396.

3.2 The datasets used in our proposed approaches

3.2.1 Nature of real-world datasets-tomograms

In the following chapters, we are using both real-world and synthetic datasets/tomograms. The real-world datasets were captured at Diamond Light Source to observe a corrosion process on a metal test sample [418]. This was initiated by applying a droplet of salt-water on top of a 500 micron aluminum pin with magnesium deposits (see example images in Figures 3.3, 3.4, 3.5). The sample is mounted on the beamline sample stage, and the salt-water drop is added to the sample, starting the corrosion process. As soon as this has started, a series of high-speed tomography data collections (91 projections per collection over 180 degrees) are acquired capturing the corrosion process as it occurs. This results in a time-series of undersampled tomograms. Once the corrosion process has finished, or slowed to a stable level, and before the sample is removed, a final high number of projections scan (3601 projections) is collected. The resulting highly-sample tomograms are collected as spatial reference for its respective time-series of undersampled tomograms.

3.2.2 Real-world datasets-tomograms

There are two available datasets/tomograms, which are highly-sampled tomograms (as described in Section 3.2.1) captured at the end of two distinct time-series of undersampled tomograms, depicting thusly two different physical samples. For convenience, we are going to name these real-world highly-sampled tomograms as *RealHS1* and *RealHS2*, while their corresponding time-series as *RealTS1* and *RealTS2*, respectively. Both were collected at Diamond Light Source I13-2 beamline [416] and *RealHS1* and *RealHS2* (their projections) are available at [419]. Additionally, all 21 reconstructed undersampled tomograms of

RealTS1 are available at [420], and they are reconstructed using the CGLS reconstruction algorithm [413].

All projections of *RealHS1*, *RealHS2*, *RealTS1* and *RealTS2* have a size of 2160×2560 pixels. It takes approximately 30 seconds for the capture of each of the undersampled tomograms in the time-series and 5 minutes for the final highly-sampled scan. The data is stored using the NeXuS data format [421] which is built upon the HDF5 format (.h5) [422], that permits the researchers at Diamond to easily process the datasets and keep a detailed history of the data being processed by various steps [423].

Both *RealHS1* and *RealHS2* are used for the training and testing of our UDNN [24], presented in chapter 4 as explained in Section 4.4.3. For our approaches described at chapters 5, 6 and 7, the *RealHS1* is used after it is manually annotated. The resulting annotations are available at [424]. *RealHS1* high SNR reconstruction is available at [425], produced using the Filter Back Projection method (FBP) [409] algorithm and all 3601 available projections. Additionally, its low SNR reconstruction is available at [426], produced using the Conjugate Gradient Least-Squares method (CGLS) [413] and a downsampled version of the tomogram with only 91 projections. The utility of *RealHS1's* reconstructions and annotations is described further at Section 5.4.1.

3.2.3 Synthetic datasets-tomograms

Apart from the use of real-world tomograms, it is helpful to also perform our approaches using synthetic data which can be generated in predictable ways. Therefore, synthetic data are created using the TomoPhantom toolbox [23]. They are synthesised through the use of simple geometric objects like cylinders and ellipsoids in an effort to simulate their real life counterparts, broadly representing the real-world datasets. Because of the current limitations of the TomoPhantom toolbox the size of the synthetic projections is 1080×1280 pixels, again having a total of 3601 projections. Specifically, the pin is simulated via a cylinder primitive, and the droplet of salt-water via an ellipsoid, cut in half. The hydrogen bubbles, a product of the corrosion in the real-world data, are simulated with randomly-placed spheres of random radii.

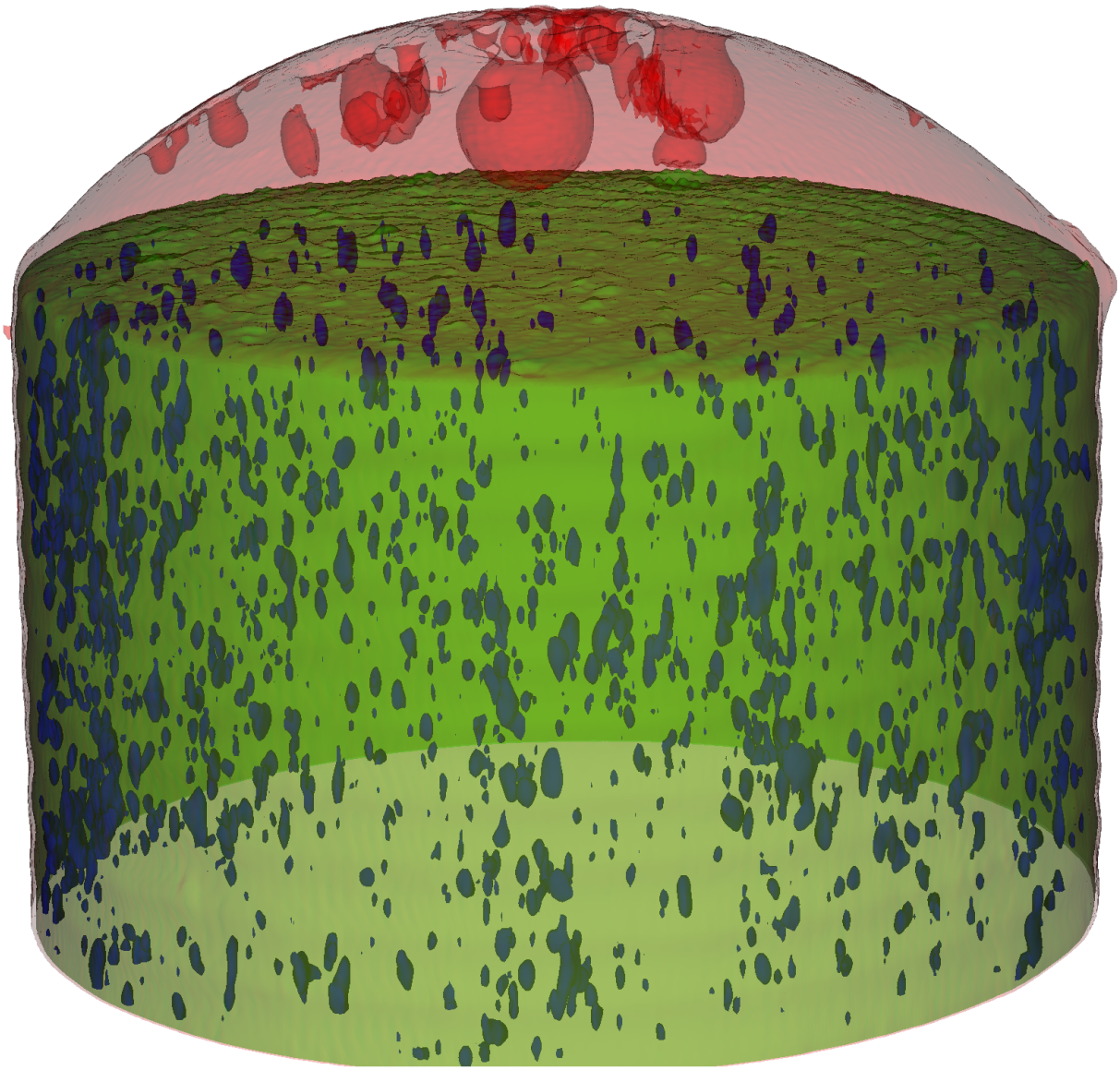


Figure 3.3: *A coloured render of the datasets used by the respective methods presented in this thesis. The render was produced using the VisIt software [427]. It is a drop of salt water to the top of a 500 micron aluminum pin with magnesium deposits within it. The green is the aluminum, the blue are the magnesium deposits, and the red is the salt water on top with some hydrogen bubbles produced from the ongoing corrosion.*

In the following chapter 4, two synthetic tomograms are used, *SynthHS1* and *SynthHS2*, that are created based on two distinct phantoms (the synthetic equivalent of a physical sample). Both simulate the magnesium deposits that are present in the pin with forty randomly-placed ellipsoids and with random elliptic radii for each in both training and testing datasets. To these tomograms Poisson noise is also added, in an attempt to emulate the noise present in the real-world tomograms. Both their noisy and noiseless version of these tomograms are available at [428].

Nevertheless, in the following chapter 6 that presents a knowledge transfer scheme where neural networks are trained on synthetic tomograms, two more sophisticated synthetic tomograms, *SynthHS3* and *SynthHS4* are used. Both are based on the same phantom (the synthetic equivalent of a physical sample) and both have a higher number of ellipsoids which simulate magnesium deposits and also additional ellipsoids that simulate air pockets presents in real-world tomograms. Reconstructions of *SynthHS3* are available at [429, 430] and reconstructions of *SynthHS4* are available at [431, 432]. More details about their generation and utility is described at Section 6.2.2.

3.2.4 Real-world data challenges

In the real-world data, however temporal events occur during the capture and in this case, several corrosion products are created. Such events produce abrupt changes in the sinogram (e.g. in Figures 3.5(**a**) and 3.5(**b**) see Marker 1). This produces an extra challenge when handling these tomograms. For instance for the case of sinogram upscaling, analytical interpolation methods are not able to account for this step changes; compared to our UDNN method (more in chapter 4) that may be able to learn from the provided highly-sampled tomograms.

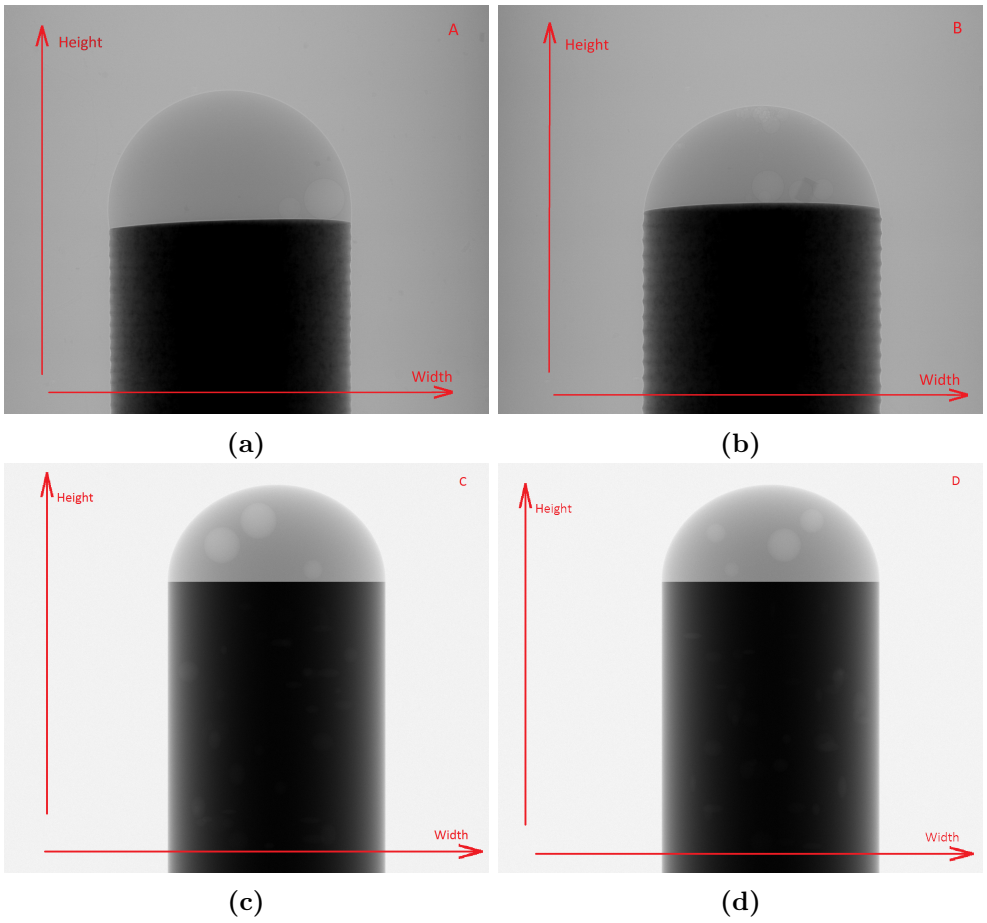
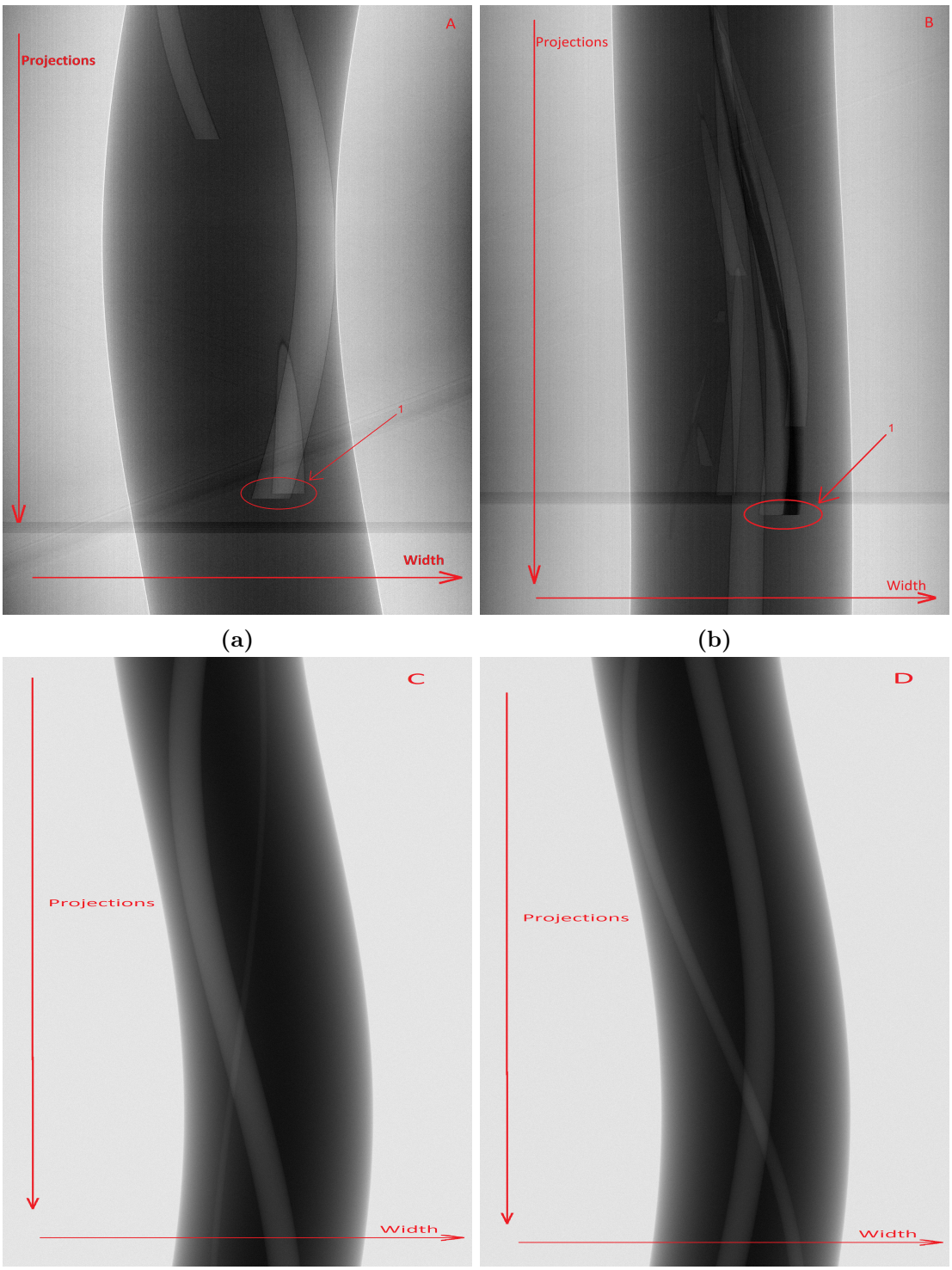


Figure 3.4: *Images of projections (the* 1801^{th} *projection out of* 3601*) from the different tomograms used in this thesis and specifically for the approach described at chapter 4. (**a**) From RealHS1, the real-world data used for training, (**b**) From RealHS2, the real-world data used for testing, (**c**) From SynthHS1, the synthetic data used for training, (**d**) From SynthHS2, the synthetic data used for testing. The sample in the real-world data is an aluminium pin with magnesium deposits that reacts with the droplet of salt-water on the top of the pin which produces hydrogen air bubbles. The synthetic data were constructed trying to simulate the real-world world counterpart.*



(a) (b)

(c) (d)

Figure 3.5: *Images of sinograms (the 1001st pixel row counting from the top for the real-world data, the 201^{th} pixel row for the synthetic data) from the different dataset used in this thesis and specifically for the approach described at chapter 4. (**a**) From the RealHS1, real-world data used for training, (**b**) From RealHS2, the real-world data used for testing, (**c**) From SynthHS1, the synthetic data used for training, (**d**) From SynthHS2, the synthetic data used for testing. These sinograms correspond to height were hydrogen bubbles are expected to be found. As can be seen in the real-world data the sinograms suddenly stop (Marker 1) as the bubbles surface on the top of the sample and pop. This was purposely avoided in the synthetic data in order to check the accuracy of our method when the conditions are nominal. The image ratio is different from the original for presentation purposes.*

3.3 Summary

In this chapter, we presented how computed tomography has enabled the study of the inner systems of various biological and non-biological systems. Namely, X-ray CTs due to the low attenuation of X-ray light have facilitated the imaging of the intricate structure of samples in the micro meter scale. However, high doses of radiation can be damaging to the samples, requiring potentially multiple attempts to collect the data.

There are two configuration geometries for collecting X-ray CTs, the cone beam configuration and the parallel beam configuration. The parallel beam configuration is commonly used at synchrotron sites and it is preferable for non-medical experiments (biological and material science samples) as in conjunction with the high flux of the X-ray light produced in synchrotrons, it can lead to more accurate tomogram reconstructions. To obtain a tomogram using the parallel beam configuration, the sample is placed on a rotating tray and multiple projections are captured as it completes 180° degrees around its height-axis. Then using an inverse Radon transformation a 3D tomogram reconstruction can be acquired.

A common way to study temporal changes in the inner structure of imaging samples is with time-resolved CT data collections. During these collections, multiple tomograms are captured sequentially forming a time-series of tomograms. In an attempt to increase the temporal resolution of these time-series, the number of collected tomograms has to be high, leading to a time limitation for the collection of each tomogram. Each of the resulting tomograms will have a small number of collected projections, which sets an ill-posed problem during their reconstruction and their final reconstructions will have low SNR. Fortunately though, it is common either before, after or both before and after the collection of each time-series, highly-sampled tomograms to be collected as a spatial reference.

It will be the aim of the approaches described in the later chapters, to utilise these highly-sampled tomograms to train CNNs. This is because an undersampled version of them can be made by downsampling their projections (downscaling their sinograms). This allows

deep learning approach to learn a mapping between the undersampled version of these tomograms and their original highly-sampled version. By then apply this mapping to the undersampled tomograms of the time-series, which the highly-sampled tomograms accompany, can lead to upsampled versions of the time-series' tomograms with high SNR reconstructions. This idea is explored in the following chapter 4, where the highly-sampled tomograms are used to train a CNN network which learns to upscale the sinograms of the undersampled tomograms in a time-series, therefore upsampling their projections and leading to high SNR reconstructions.

Chapter 4

UDNN: A CNN network for fast upsampling of X-ray CT time-series

4.1 The upscaling deep neural network

As the previous chapter 3 mentions, the component tomograms in micro X-ray CT time-series suffer from a severely low Signal-to-Noise Ratio (SNR), while also contains noticeable artifacts produced from the X-ray CT imaging procedure (see Section 3.1). In this chapter we will explore our proposed Upscaling Deep Neural Network [24] published in Journal of Synchrotron Radiation, that aims to increase the SNR of the reconstructions of the earlier tomograms. As it also mentioned earlier, the low SNR is cause by the high speed in which the time-series' tomograms are captured, in order to ensure a sufficient time-resolution and collect time sensitive events. The higher rate at which data is captured allows only the capture of a limited number of projections for each of the respective tomograms of a time-series. During the reconstruction of these tomograms from a set of projections to a final volume, the low number of projections translates to a low SNR reconstruction. In addition, prevalent imaging artifacts like ray artifacts, are introduced and its overall spatial quality of the reconstruction volume is decreased. Since the accuracy of segmentation is heavily dependent on the initial quality of the data, a potential increase of the SNR (taking care not to introduce features not present in the original data) in the aforementioned tomograms would be greatly beneficial during the segmentation process. This is the motivation behind the work in this chapter.

A common response to improve the quality of the tomogram's reconstruction, iterative

reconstruction algorithms are usually applied in such cases. Examples include the Simultaneous Algebraic Reconstruction Technique (SART) [411], the Simultaneous Iterative Reconstruction Technique (SIRT) [412] and the Conjugate Gradient Least Squares method (CGLS) [413]. Nevertheless, these iterative algorithms require a considerable amount of compute time, and do not scale linearly with the size of the data. For the case of tomogram time-series this additional time is also multiplied by the length of the time-series, since each respective algorithm would be applied individually to each tomogram. As demonstrated in Section 2.3.2, an undersampled tomogram with respect to the number of projections can be interpreted as a tomogram with low resolution sinograms along their θ-$axis$. Therefore, an upscaling of the individual sinograms of these undersampled tomograms can effectively lead to an increase in the number of projections, and in turn improve the tomogram's reconstruction quality after the application of a reconstruction algorithm. For this purpose, approaches toward upscaling the tomogram's sinograms have been proposed in the literature. Those range from analytical interpolation techniques, like cubic interpolation [21], and directional interpolation techniques [22], [70] to more sophisticated approaches using deep learning [153]. However, despite the fact that the latter ones are able to provide a remarkable increase in SNR, as well as noteworthy quality improvement of tomogram reconstruction, they require a sufficient amount of highly-sampled representative tomograms combined with an ample time spend for training. Additionally, as a trade-off for their unquestionable performance, they are only becoming proficient at upscaling sinograms of tomograms that only depict samples of similar nature.

Motivated from the fact that in material science and biology, datasets can differ greatly between different studies, we developed a novel lightweight network that can easily and quickly be *retrained* and therefore applied to different classes of samples. For the training of our approach a bare minimum of one representative highly-sampled tomogram is required. Fortunately though, during studies that utilise time-series of tomograms, it is common practice to collect tomograms that contain a large number of projections either before, after or both before and after the time-critical portion of the experiment, without detrimentally affecting the time-series. The prior provided by these highly-sampled

tomograms allows the application of an upscaling process with a high accuracy. Finally, since one of our approach's main goals is to be lightweight and quick for inference or retraining, it is compared to analytical approaches [21, 22] that are respectively fast and do not require an extensive training phase. On the contrary, heavyweight methods [153] are not designed for frequent retraining and they require many hours to train. For this reason, we do not compare them to our approach, since such a comparison would not be fair.

As described in 2.3.2, each pixel row of the *theta-axis* in each sinogram corresponds to different projection of the tomogram, and since an increase of the number of projections will increase the SNR, our proposed approach is designed to upscale the sinograms along their *theta-axis* only. Subsequently, this requires the estimation of the intermediate projections that are omitted during the tomogram's collection. It is also important to point out that each sinogram corresponds to a particular height in the tomogram projections, since a sinogram is the concatenation of all pixel-rows the have the same y index value from all projections. Since, for the capture of individual tomogram of a time-series, the sample rotates around its height *y-axis* 3.4, there is not any transfer of information along this axis within the collected projections. This combined with the earlier fact, that each sinogram corresponds to a particular height in the tomogram, sets the task of attaining the omitted projections more easier performed as the upscaling of tomograms' sinograms. This is because a direct projection upsampling would require image synthesis for the new projections, which is more complicated that the addition of new pixel-rows in sinograms. Furthermore, the upscaling of the tomograms' sinograms that are part of a time-series (4D dataset) can be performed separately (independently through time) to each of the time-series tomograms, therefore setting our approach applicable on time-series of tomograms.

The proposed network architecture is based on Dong *et al.* [33] and is comprised of three convolutional layers separated by two Rectified Linear Units (ReLUs) [433]. The network layers jointly learn the underlying mapping between the input sinograms and the missing intermediate pixel-rows/projections. Because of that, the prediction which our network outputs is the concatenation of these missing intermediate pixel-rows/projections. The

desired upscaled sinogram is obtained by interlacing the pixel-rows/projections from the initial input sinogram with the predicted intermediate pixel-rows/projections (more details in Section 4.3).

The remainder of the chapter is organised as follows. Section 4.2 will present details for the network's architecture and design. Section 4.3 will provide comprehensive information for the network's internal components. Section 4.4 will offer details regarding the training procedure employed by the network as well as the way at which the final upscaled sinograms are formulated from the network's predictions. Section 4.5 will present the experiments that took place, specifically experiments regarding the hyperparameter selection (Section 4.5.1), the approach's application on synthetic data (Section 4.5.2), on real-world data (Section 4.5.3), and concluding with a discussion at Section 4.6 about the quantitative results produced from the earlier experiments. Lastly, in Section 4.7 our approach is qualitatively evaluated on a different tomogram that is part of time-series and Section 4.8 concludes the chapter.

4.2 Methodology and network architecture

As it is described earlier the proposed approach's goal is to predict the intermediate pixel-rows/projections that have to be placed between those of the input sinogram in order to upscale it. To do so, the network consists of 3 Convolutional Layers separated by two Rectified Linear Units. In the following subsections, we will go through the functionality of each of the network's layers and the special role that each of them has towards the accurate prediction of the aforementioned pixel-rows/projections. A graphical overview of the network and its operation is presented in Figure 4.1.

4.2.1 Feature extraction

Starting with the first layer in the Upscaling Deep Neural Network, its role is to extract significant features for upscaling from the input sinogram regarding the intermediate pixel-rows/projections using a convolutional layer. A number of high-dimensional vectors are constructed for each pixel of the layer's output, using a local neighbourhood from the

input sinogram. The concatenation of these vectors comprise the set of feature maps for this layer, equal in number to the dimensionality of the vectors. As it will be further explored in Section 4.4.8, emphasis it is given to the direction of the *x-axis* in the local neighbourhood, due to the angle at which sinusoidal traces of internal components are captured in the sinograms. This is expressed by rectangular kernels with their longer side aligned along the *x-axis*. This allows for the collection of features that are significant for the accurate upscaling of the sinograms, especially sinograms which have few rows (projections) along the *y-axis*, which is a feature of undersampled tomograms, where the angle of the sinusoidal traces with the *x-axis* is very acute. The rectangular shape of the kernels is also inspired by the directional interpolation proposed by Bertram *et al.*'s approach [22], where the typical interpolation is modified to become more asymmetric and emphasise on information present along the *x-axis*.

4.2.2 Depth increase and non-linearity

Continuing the exposition of the network, the network next uses a Rectified Linear Unit (ReLU) as an activation function, and a second convolutional layer is introduced. With the use of a Rectified Linear Unit (ReLU) all the negative elements in the feature maps are replaced with zeros, thusly introducing non-linearity into the system. The introduction of non-linearity is very significant, as without it the relationship between the input and the output would only be linear, which is almost never the case. After the inclusion of the non-linearity, the rest of the parameters are appropriately adjusted during training, to learn the desirable input/output mapping. The use of ReLU as an activation function is very common in convolutional neural networks, as it has been proven to yield better results [433], [434] compared to other traditional activation functions, like sigmoid and tanh. Furthermore, with the introduction of second convolutional layer and of course with the third one following it, the network acquires additional depth which aids it to learn the aforementioned underlying mapping at a much higher precision. Specifically, the layer works as a non-linear mapping between the vectors of the previous layer and new vectors which have size a smaller size. The additional depth introduced by the layer

increases also the receptive field for each of the new vectors. This means that the vectors comprising this layer's feature maps, would contain higher levels of information derived from the previous features. These would then provide the network with the necessary complexity to achieve more accurate predictions.

4.2.3 Construction of the upscaled output

Another ReLU unit is used as an activation function for the similar purpose of introducing a non-linearity, and following that is the third and final convolutional layer of the network. The functionality of this layer is to calculate the intermediate pixel-rows/projections using the previous layer's vectors. The precision of this layer is very crucial, as for every pixel of the output, it condenses information from multiple vectors into a single scalar value. This significance is further explored in Section 4.4, in the form of carefully selecting this layer's learning rate during training. At the final stage, the intermediate pixel-rows/projections predicted by the network are interlaced with the ones for the input to produce its upscaled version. Additionally, as it is described in Section 4.4.4, the network is able to upscale only *chunks* of the input sinogram at each forward propagation, therefore the final upscaled sinograms are produced by combining the output upscaled chunks (More in Section 4.4.7). After all sinograms of a tomogram have been upscaled, the upsampled tomogram can then be fed to a reconstruction algorithm that reconstructs it into a volume format. In the case of time-series, this process is repeated respectively for each of the time-series tomograms.

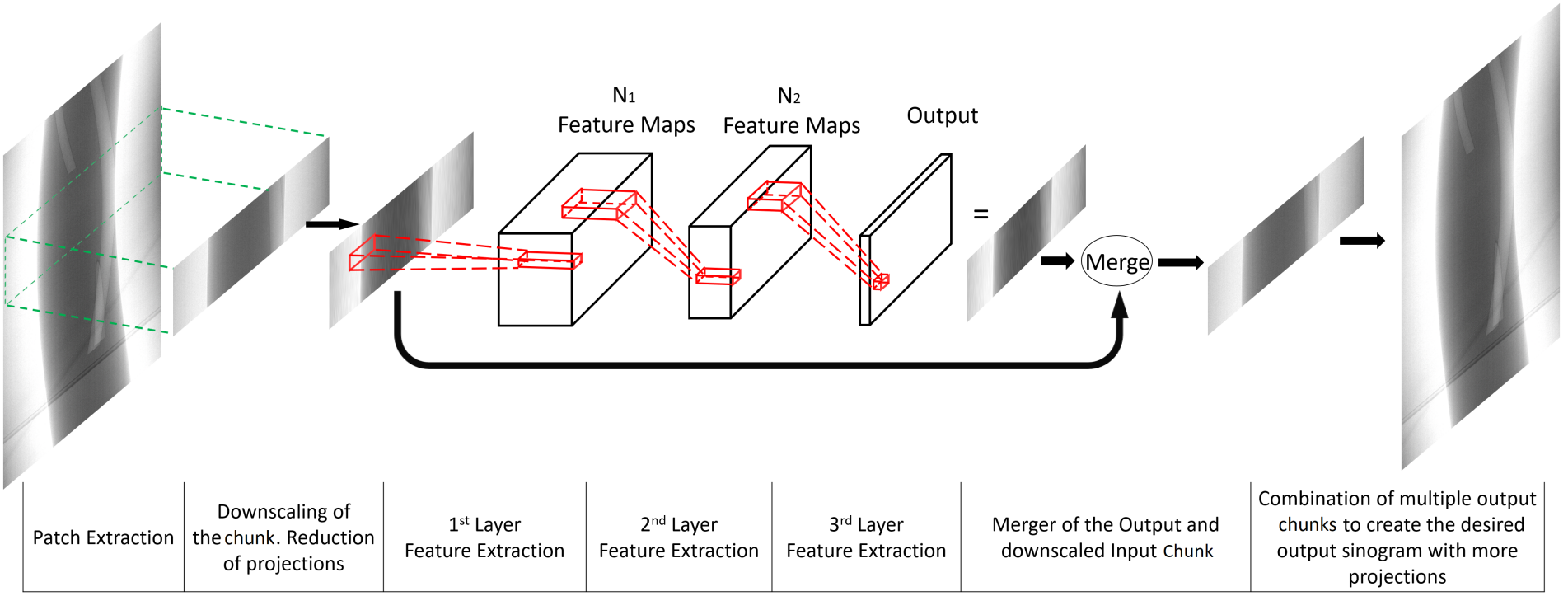


Figure 4.1: *Graphical representation of the functionality of the Upscaling Deep Neural Network. The downscaling step is necessary during the network's training with a set of high sampled sinograms, but not for using the network after the training has finished.*

4.3 Formulation

Let's denote the input sinogram as $\boldsymbol{I}$ and the desired learned mapping as $\boldsymbol{F}$. The goal is to recover from $\boldsymbol{I}$ an image $F(\boldsymbol{I})$ that would be as similar as possible to the ground truth image $\boldsymbol{O}$. Here $\boldsymbol{O}$ is the concatenated intermediate pixel-rows/projections (see Figure 4.2) extracted from a ground truth highly-sampled sinogram $\boldsymbol{G}$.

The first layer of the network can be expressed as:

$$F_1(\boldsymbol{I}) = ReLU(W_1 * \boldsymbol{I}' + B_1) = max(0, W_1 * \boldsymbol{I}' + B_1) \tag{4.1}$$

Here $\boldsymbol{I}'$ represents the input sinogram $\boldsymbol{I}$ appropriately padded with zeros, W_1 are the weights and B_1 the biases of the first layer (identified by subscript 1). The symbol of '$*$' denotes the operation of convolution. In equation (1) the weights W_1 represents N_1 kernels of size $1 \times f_{1h} \times f_{1w}$, where f_{1h}, where f_{1h} and f_{1w} are the height and width of the kernels of this layer respectively. In this first layer, N_1 convolutions are applied on the input $\boldsymbol{I}$ resulting in an output of N_1 feature maps. Additionally, B_1 is a N_1-dimensional vector containing the biases of each convolution. It is important to mention here, that the input sinograms are appropriately padded with zeros before the convolutions. The act of padding with zeros is common in CNNs as it is used to dictate the size of the output. In this layer the padding is used so each of its feature maps have the same size as $\boldsymbol{O}$. Lastly, the function of ReLU is represented as a max function with the second argument being zero. ReLU's functionality is to replace all the negative elements in the feature maps with zero.

Next in the architecture, there is another convolutional layer which provides the necessary depth and non-linearity to the network. The mathematical expression for it is as follows:

$$F_2(\boldsymbol{I}) = ReLU(W_2 * F_1(\boldsymbol{I})' + B_2) = max(0, W_2 * \boldsymbol{I}' + B_2) \tag{4.2}$$

Here $F(\boldsymbol{I})'$ represents the feature maps $F(\boldsymbol{I})$ appropriately padded with zeros, so that the input and output feature maps have the same size. Moreover, in this second layer (identified by subscript 2), W_2 corresponds to N_2 kernels of size $N_1 \times f_{2h} \times f_{2w}$, where f_{2h}

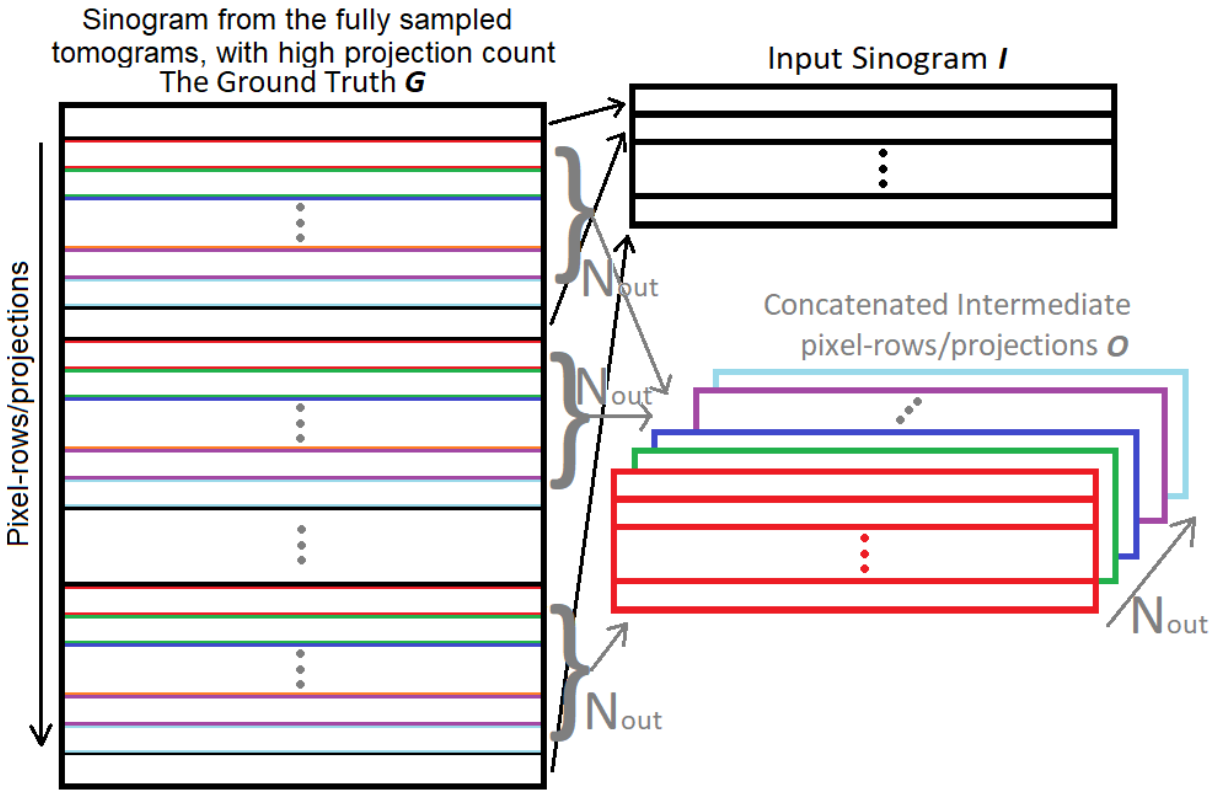


Figure 4.2: *Diagram presenting how the input sinogram and the concatenation of the intermediate pixel-rows/projections are formed. Intermediate pixel-rows that are between two neighbouring pixel-rows/projections of the input sinogram are placed in different output channels.*

and f_{2w} are the height and width of the kernels of this layer respectively. The biases for this layer are represented by the N_2-dimensional vector B_2. In this layer the N_1 previous vectors are replaced with N_2 new ones, replacing also consecutively the corresponding feature maps. This is achieved by applying N_2 convolutions with kernels of W_2.

At this stage, it be should noted that it is possible to add additional layers to further deepen the network for additional accuracy. However, as it is explored in Dong *et al.*'s [33] publication, it has shown to have diminishing returns. For that reason, and since our proposed network is designed to upscale only grey-scale intensity images in only one direction (along the *y-axis*; versus 2D colour upscaling), the idea of adding additional layers is considered unnecessary. It is also significant to point out that similar to Dong *et al.*'s [33] publication the kernel's dimensions f_{2h} and f_{2w} are not going to be elected to be small numbers. On the contrary, as it would be pointed out in Section 4.4.8 they are going to be fairly large compared to the 3x3 kernels that are common in most CNN applications [153, 268]. This translates to large increases of additional training time, for each extra

convolutional layer added to UDNN's architecture. Furthermore, due to the large size of the kernel dimensions, the receptive field of each consecutive layer increases drastically and to levels that are not necessary for the accurate interpolation of the sinusoidal traces within the sinograms; therefore supporting the decision of not going further than 3 convolutional layers.

Additionally, deeper network architectures perform accurately by learning in depth, meaningful features that are prevalent in their training dataset. In particular, they are able to detect highly abstract features. For instance in *natural* images, they are able to discern the appearance of a human noise for the appearance specific edges in a location. Due to this effect, training datasets have to be as representative as possible of the datasets to which the network is intended to be applied upon. It is therefore advantageous to apply neural networks in situations where they can be trained once and with datasets that will not be outdated soon, meaning the training datasets will remain representative to the datasets used for inference for a long period of time (even forever). In the same example as before, the training dataset of *natural* images may contained human noises and the network is training to relay on their presence in order to be accurate. If the follow-up images (datasets), in which the network will be applied on cease to contain this element (human noises), the accuracy of the network may be noticeably lower for these images, since the training set was not representative to them. Regarding now not *natural* images but tomograms in time-series, while representative tomograms exist, post-training the number of tomograms at which the resultant network can be applied to is limited and it requires retraining in order to be applied on new tomograms. Based on this fact, the design of our approach is purposely selected to be lightweight, to allow for a network that can easily be retrained and applied on a particular tomograms' sinograms.

Finally, the last layer constructs the output $F(\boldsymbol{I})$ using the feature maps of the previous layer. The layer can be expressed as follows:

$$F_3(\boldsymbol{I}) = (W_3 * F_2(\boldsymbol{I})' + B_3) \tag{4.3}$$

Here $F_2(\boldsymbol{I})'$ represents the feature maps $F_2(I)$ appropriately padded with zeros, so that

F(I) have the same size as $\boldsymbol{O}$. Additionally in the third layer (identified by subscript 3), W_3 contains N_{out} kernels of size $N_2 \times f_{3h} \times f_{3w}$, where f_{3h} and f_3w are the height and width of the kernel respectively. B_3 are the biases for this layer and are represented in an N_{out}-dimensional vector. In this layer the N_2 feature maps are being condensed using N_{out} convolutions to the desired output. As will be further explained in Section 4.4.3, N_{out} is the number of intermediate pixel-rows/projections between two neighboring pixel-rows/projections of the input. Therefore, N_{out} is equal to the upscaling factor minus one and by increasing it, the network can be set to upscale the sinogram at higher rates. Despite being present in the earlier layers, for this layer there is not a subsequent ReLU unit. As mentioned in Section 4.4.3, input sinograms are normalised and so using a ReLU unit which sets to zeros all negative values would eradicate important information, since the range of possible output pixel values due to normalisation can be negative. Moreover, the parameters for this layer must be precisely fine-tuned as the estimation of each output pixel is the linear combination of multiple feature vectors, thus making it very error sensitive. This sensitivity plays an important role during the design of the training, which is further described in Section 4.4.3.

4.4 Training, validation and testing

4.4.1 Loss function

In order for the network to adjust its weights and biases $\boldsymbol{\Theta} = \{W_1, W_2, W_3, B_1, B_2, B_3\}$, it has to train using already known input and ground truth pairs. This is achieved by attempting to minimise a loss function which calculates the error between the networks predictions $F(\boldsymbol{I}; \boldsymbol{\Theta})$ and the ground truth $\boldsymbol{G}$. For our approach, the loss function that is selected is the Mean Square Error (MSE), described in equation 4.4. MSE is a common loss function used to calculate the error in applications where the task is to predict a visual output, such in image denoising and image generation. It is commonly used function for estimating error between an imaging system (image, volume, etc.) with low imaging quality and another with high imaging quality which is considered as "golden standard ground truth". This is because MSE is differentiable and it is balanced, meaning that it

does not attribute extra error to specific inclinations of the input from the ground truth than others.

Given the concatenated intermediate pixel-rows/projections $\boldsymbol{O}$ extracted from ground truth sinograms $\boldsymbol{G}$ of the highly-sampled tomogram, and the available input sinogram $\boldsymbol{I}$, the mean square error can expressed as:

$$MSE\big(F(\boldsymbol{I};\boldsymbol{\Theta}),\boldsymbol{O}\big) = \frac{1}{N}\big\|F(\boldsymbol{I};\boldsymbol{\Theta}) - \boldsymbol{O}\big\|_2^2 \tag{4.4}$$

where N is the total number of imaging elements, pixels in our case, present in the concatenated intermediate pixel-rows/projections $\boldsymbol{O}$.

4.4.2 Backpropagation and learning rate

The minimisation of the loss function is typically achieved through a stochastic gradient-based optimisation. In the experiments described later (Section 4.5) we use Kingma and Ba's Adaptive Momentum estimation (Adam) method [435] to perform the task of back-propagation while training the network. The Adam method has been shown to outperform other stochastic gradient descent methods, like AdaGrad [436] and RMSProp [437]. Their method calculates adaptive learning rates individually for each of the network's parameters (weights and biases) using estimates of first and second moments of the gradients. Adam has 4 main hyperparameters:

1. α or η, The stepsize, which is also the baseline learning rate.
2. $\beta_1 \in [0, 1)$, which is the exponential decay rate of the moving average of the gradient.
3. $\beta_2 \in [0, 1)$, which is the exponential decay rate of the moving average of the squared gradient
4. ϵ, a very small number to eliminate the case of dividing by zero; its default value is set to 10^{-8}

As is suggested by the authors and observed by [435], good settings for the decay rates β_1, β_2 are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. As regard to the learning rate η, following the design option of Dong *et al.* [33] and supported by preliminary experiments, the network has two different learning rates; η_1 for the first and second convolutional layer and η_2 for the

third layer. As it is mentioned earlier both in Section 4.2 and 4.3 the third layer has the sensitive job of calculating the networks predictions and thus large adjustment on its parameters are counterproductive. Because of the fact, while the learning rate for the first and second convolutional layers is set to $\eta_1 = 0.01$ the one for the third is set to $\eta_2 = 0.001$. Furthermore during the training, based on the validation error's trend the learning rates are appropriately decreased (by a factor of 10) to enable the network to converge.

4.4.3 Data allocation to training, validation and testing

As is described in 3.2, one synthetic dataset/tomogram (*SynthHS1*) is used for training/validation and a separated one (*SynthHS2*) is used for testing. Similarly, a real-world dataset/tomogram (*RealHS1*) is used for training/validation and a separated one (*RealHS2*) is used for testing. Regarding both the real-world and synthetic tomograms respectively, the tomograms used for training/validation and the tomograms used for testing are similar in nature but have a different structure, meaning that they depict two different physical samples. Specifically, for the synthetic data the placement, size and orientation of the particles that simulate the magnesium deposits and hydrogen bubbles are different between the tomogram used for training and the one used for testing and both tomograms are obtained using the same parameters and the TomoPhantom toolbox [23]. Similarly for the real-world data, the tomogram used for training/validation and the tomogram used used for testing depict different pins of aluminium with magnesium deposits. This means that any hydrogen bubbles, magnesium deposits and air-pockets have different locations, sizes and shapes between the tomogram used for training/validation and the tomogram used for testing. Additionally, as it can be seen from Figure 3.4 (**a**,**b**) the location of the pins in relation to the whole tomogram is different between the two real-world tomograms and both tomograms are collected using the same experimental conditions. The synthetic and real-world dataset/tomograms both contain 3601 projections, with the synthetic projections having a size of 1080×1280 pixels and the real-world 2160×2560 pixels. Moreover, there are two other datasets/tomograms used to calculate the testing error and

they are respectively a synthetic tomogram and a real-world tomogram. Regarding the tomograms used for training/validation, their set of sinograms is split into two halves, one for the network's training and the other for validation. Specifically for the real-world data, from the training/validation tomograms, 1079 sinograms are reserved for training and 1079 for validation. Correspondingly from the synthetic tomogram, 540 sinograms are reserved for training and 540 for validation.

However, it is important to point out that the 2 halves in both the synthetic and real-world tomograms, do not split the respective tomograms into two separate spatial halves. This means that sinograms are not split to two groups each corresponding to two different height ranges. Instead, the split happens with an odd-even separation, where the synthetic sinograms used for training are the $1^{st}, 3^{rd}, \cdots, 1079^{th}$ and the $2^{nd}, 4^{th}, \cdots, 1080^{th}$ are used for validation. Likewise, the real-world sinograms used for training are the $2^{nd}, 4^{th}, \cdots, 2158^{th}$ and the $3^{rd}, 5^{th}, \cdots, 2159^{th}$ are used for validation. In the real-world data, the 1^{st} sinogram is omitted since the Wadeson and Basham's Savu Framework [438] used to produce the real-world tomograms inserts as first a sinogram containing only metadata. The reason sinograms are separated in this alternating fashion, for the network's training and validation purposes is exactly because spatial separation must be avoided and the halves have to be representative of the entire tomograms' volume. As it is mentioned, each sinogram correspond a different value in the *y-axis* of the projections (see Figures 2.15 and 2.16) and an non-alternating split would separate a tomogram into two spatial different portions, not necessarily representative of the entire tomogram's volume (Figure 3.4 shows that our tomograms change a lot along the height *y-axis*). It could be claimed that this might cause the training and validation sets to have a high correlation with each other and so generate misleading results. Nevertheless, as will be later explored in Section 4.5 during testing (where entirely different tomograms are used respectively for the real-world and the synthetic experiments) UDNN's testing accuracy close to the network's validation accuracy, providing evidence that the unwanted scenario of overfitting is avoided during the training.

4.4.4 Input sampling

Since the datasets/tomograms that are being used are highly-sampled tomograms collected at the end of time-series capture (More in Section 3.2) they contain 3601 projections, compared to only 91 present in the intermediate time-series' tomograms. For this reason, in order for sinograms to be used for training, they have to initially be downsampled along their θ-$axis$ and discard some of their pixel-rows/projections, while keeping some as ground truth. The downsampled sinograms will then be fed into the network to predict a number of intermediate pixel-rows/projections which would be compared with the ones kept as ground truth. By predicting only one pixel-row/projection between each two neighbouring input pixel-rows/projections ($N_{out} = 1$, More in Section 4.3), the upscaling factor is set to 2x. This means that the output sinogram, which is the input sinogram interlaced with the network's predictions, will be almost double in size along the y-$axis$ (almost, because if the input has a length of $n + 1$ in the y-$axis$, the output will have a length of $2n + 1$). Namely, after the training sinogram of 3601 pixel-rows/projections is downsampled to 91 pixel-rows/projections, the network is able to double the pixel-rows/projections to 181. This downscaling of the sinograms by a factor of 40, sets the angle difference between the pixel-rows/projections of input and ground truth similar to if they were a part of an undersampled tomogram (More in Section 3.2).

Since the rate of downsampling is that large (from 3601 to 91 projections), in an effort to utilise the available sinograms as much as possible, the network does not use as input an entire sinogram, and downsampling it before using it, but rather chunks of it. This increases the number of training instances, which is important for the sufficient training of the network and it also keeps the memory demands of the network low. For the selection of the chunks, first the height of the input has to be determined. Let's denote the size of the input's height is r and f_d is the downscaling factor in the projections between the highly-sampled representative tomogram and the undersampled tomograms. Then the height of the selected chunk should be $f_d \cdot (r-1)+1$, and the width would be same as the width of the input, which is equal to the width of the sinograms. These chunks are then downscaled by a factor of f_d/f_u, where f_u is the upscaling factor. From the downscaled

chunks, a number of rows is assigned to the input and the others to the ground truth. For our data $f_d = 40$ and we select $r = 11$. Therefore each chunk is 401 pixels high and 2560 pixels wide. For $f_u = 2$, the chunks are then downscaled by a factor of $f_d/f_u = 20$, and from the 21 pixel-rows/projections left, these are split to 11 pixel-rows/projections used as input and another 10 as ground truth. The pixel-rows/projections used as input are the $1^{st}, 41^{st}, 81^{st}, \cdots, 401^{st}$ rows from the chunk and the $21^{st}, 61^{st}, \cdots, 381^{st}$ are those used as ground truth.

Each sinogram chunk is selected by a sliding window with a stride of 1, resulting in 3200 chunks per sinogram. Through the use of a stride smaller than 401 (in this case, 1), the chunks overlap with each other. This is chosen in order to diversify the input data and imitate the case of the same sample being captured under different starting positions, utilizing therefore all 3601 different projections.

It is also possible to increase the upscaling factor higher than 2x by adding more output channels ($N_{out} > 1$) in the final convolutional layer as it can be shown in Figure 4.2. For example if there were three output channels instead of one, the upscaling factor would be 4x. This is because for $N + 1$ pixel-rows/projections in the input sinogram, N intermediate pixel-rows/projections would be created for each of the output channels, therefore the final count of them would be $N + 1 + 3 * N = 4N + 1$, and so the upscaling rate= $\frac{4N+1}{N+1} \approx 4$. In this case, each channel would try to predict different intermediate rows in the chunk. The first channel would be for the $11^{th}, 51^{st}, \cdots, 371^{st}$ rows, the second for the $21^{st}, 61^{st}, \cdots, 381^{st}$ and the third the $31^{st}, 71^{st}, \cdots, 391^{st}$. This means that the angle difference between the pixel-rows/projections in each of the output channels and the input sinogram remains the same (see Figure 4.3). The reason we decided to output the intermediate pixel-rows/projections in this fashion, with multiple channels, is for ease since the upscaling factor can be easily change by changing the output channels. Otherwise, each time a new upscaling factor would to be tested, the kernel sizes in the network would have to appropriately change so that the network output one channel with all the needed intermediate pixel-rows/projections.

In the following experiment Section 4.5, we shall mostly test our network using an up-

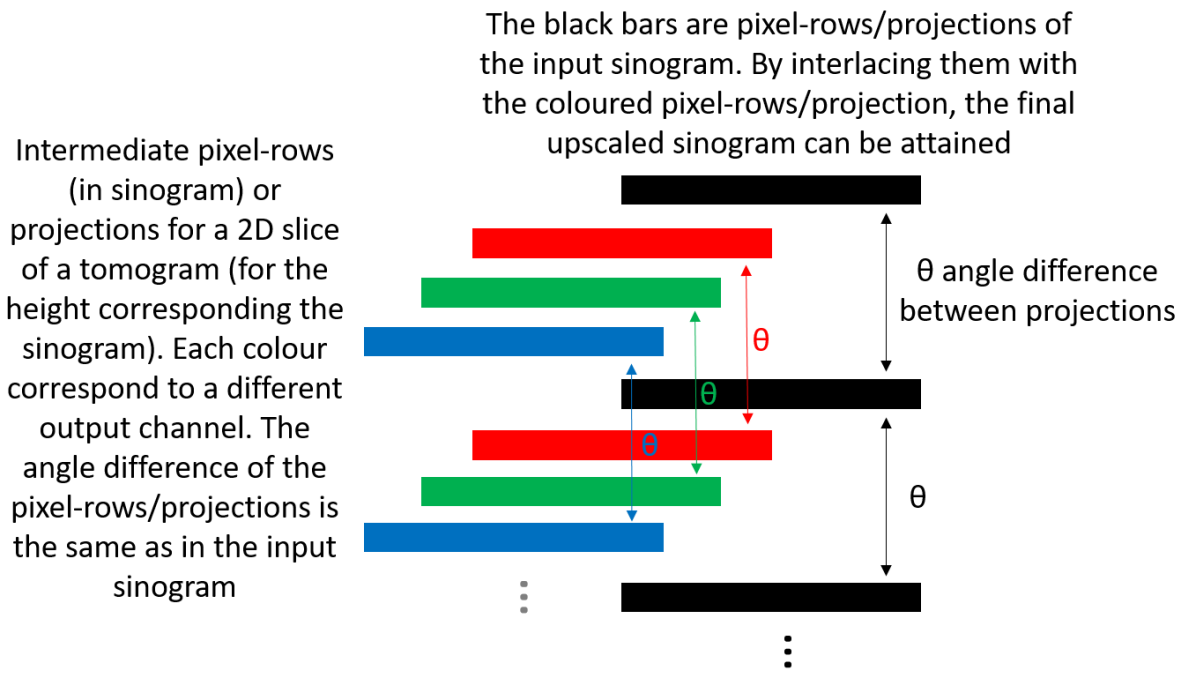


Figure 4.3: *Diagram of the 4x upscaled sinogram by interlacing pixel-rows/projections for 3 different output channels. Intermediate pixel-rows that are between two neighbouring pixel-rows/projections of the input sinogram are placed in different output channels (see also Figure 4.2). The angle difference between the pixel-rows/projections in the input sinogram and the output channels is the same.*

scaling factor of 2x, but will demonstrate an upscaling factor of 4x in the real-world data. In principle, the upscaling rate could be set to 40x in order to predict all the pixel-rows/projections that are skipped during the downsampling, however this was not attempted. As is verified by the results in the experiment Section 4.5, an accurate upscaling operation of this magnitude would not be possible with our lightweight network. Additionally, any upscaling of the sinograms and subsequent improvement in the SNR in the tomogram reconstructions, is preferable compared the low SNR in the undersampled tomograms. Due to the high amount of information lost during the downsampling, any inference from a neural network would effectively be an effort to produce a tomogram that just looks like a highly-sampled tomogram.

4.4.5 Training and validation process

Based on the above, the number of chunks reserved for training is $3,452,800$ (1079 real-world sinograms $\times$ 3200 chunks per sinogram) and $1,728,000$ (540 synthetic sinograms

$\times$ 3200 chunks per sinogram) respectively for the real-world and synthetic data, while there is the exact same number of chunks reserved for validation. Obviously, using all the chunks available for validation would result in an enormous amount time needed, as every time we would have to estimate the corresponding validation error. Adding to the fact that validations have to be frequently performed during training, it is easy to understand how much this would lengthen the training process. For this reason, 10 random chunks are selected from each sinogram resulting in $10,790$ (1079 real-world sinograms $\times$ 10 chunks per sinogram) and $5,400$ (540 synthetic sinograms $\times$ 10 chunks per sinogram) chunks used for validation, respectively for the real-world and synthetic data experiments.

During the training of the network, the sequence of input chunks used for training are chosen at each training instance/backpropagation from a different sinogram out of those available. A sinogram will be "visited" again to select and use one of its chunks only after all other sinograms have been "visited" and one of their chunks has been used. The use of a chunk from all the available training sinograms, marks one "cycle". The sequence of sinograms from which chunks are drawn in each "cycle" is selected randomly for each "cycle". Additionally, the chunks are provided in mini-batches of 10 for each iteration of a forward and backwards propagation.

The validation is selected to occur after every 16 of the earlier mentioned "cycles" have been passed, meaning that each validation occurs after every 17280 and 8640 iterations, again respectively for the real-world and synthetic data. After every validation, a snapshot of the network parameters is saved and depending on the trend of the validation error measured by the loss function, the learning rates are appropriately decreased (by a factor of 10) to enable the network to converge.

Finally, it is shown in early experiments that the network performs better when the pixels in each real-world data chunk are normalised. Therefore, each real-world data input is normalised before it is fed into the network and then the output is accordingly denormalised before it is compared with the ground truth in the evaluation stage. The denormalisation part is highly significant, since the values present in the real-world tomograms correspond to the real absorption values of the materials present in the imaged sample. In stud-

ies where chemical reactions take place during the collection of tomograms (e.g. in 4D datasets), the captured tomograms might be used to identify materials that temporally appear in the imaged sample. This means that the detection of certain materials based on their adsorption values, indicated by the pixel values of tomogram's projections, would not be possible. By denormalisating the output sinograms, the pixel values in resulting projections correspond to adsorption values of real materials and therefore unexpected materials can be discovered. The normalisation-denormalisation process is not needed for the synthetic datasets, because they are already normalised when generated.

4.4.6 Testing process

The testing of the network is conducted on completely separate, isolated datasets (More in Section 3.2) in both the real-world and synthetic data experiments to ensure the validity of the results. From these testing datasets 10% of the sinograms are downscaled by a factor of 40 (using 91 projection out of the 3601) and then upscaled with the trained network. These sinograms are the $1^{st}, 11^{th}, 21^{st}, ..., 1271^{st}$ for the synthetic data experiments and the $2^{nd}, 12^{th}, 22^{nd}, ..., 2152^{nd}$ for the real-world data ones (the first of the sinogram of the real-world datasets contains metadata making it unusable). This selection is in order so the subsets of selected sinograms to be representative to all sinograms present in the testing tomograms. The hypothetical selection of second subsets of sinograms, respectively the $2^{nd}, 12^{th}, 22^{nd}, ..., 1272^{nd}$ sinograms from the synthetic testing tomogram and the $3^{rd}, 13^{th}, 23^{rd}, ..., 2153^{rd}$ from the real-world testing tomogram, for a secondary experiment in order to estimate the standard deviation for the reported testing accuracy would not be a valid approach. This because by being part of the same testing tomograms the resulting subsets of sinograms would be highly correlated. This is also the reason why we use different tomograms for training/validation and testing. Therefore, multiple highly-sampled representative tomograms of the same or similar time-series of undersampled tomograms would be needed for testing, which are not available to us.

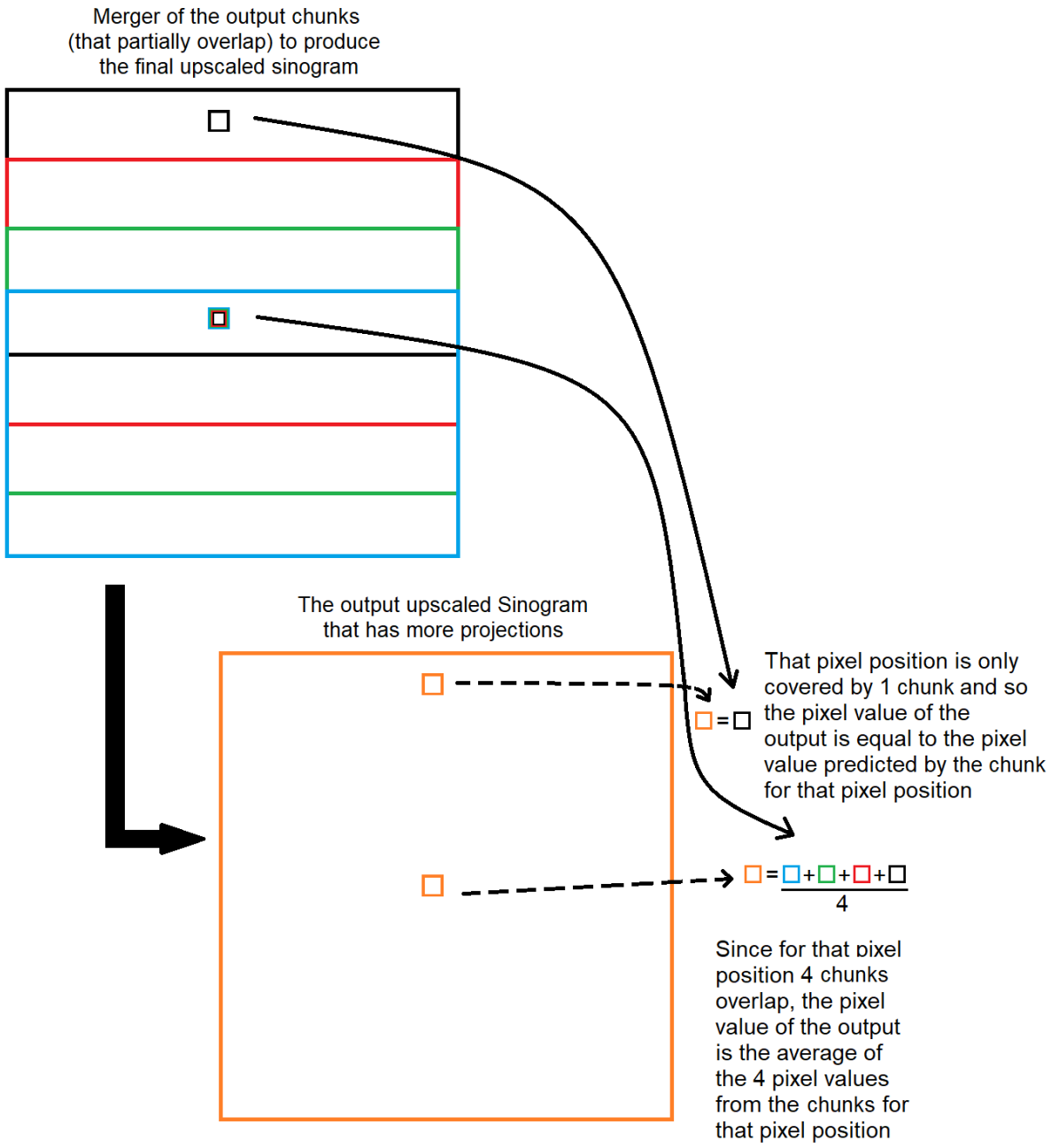


Figure 4.4: *Diagram presenting how the output chunks are merged to produce the output sinogram. At the end of the process each pixel of the output sinogram will be the average value of the pixels from the chunks that overlap in each corresponding pixel-position.*

4.4.7 Inference

After the interlacing of the network's *predictions*, (the intermediate pixel-rows/projections) with the *input* pixel-rows/projections, the upscaled version of the input is generated. However, as is mentioned earlier, the network is able to upscale sinogram chunks, therefore the final upscaled sinogram is constructed through combining the network's output chunks. Their merger could have been a concatenation of the upscaled chunks, provided that pixel-rows/projections in each of the chunks correspond to different pixel-rows/projections of the sinogram; in other words they do not overlap with each other. However, this is not the case here, and instead *overlapping* chunks are selected and then joined to form the output sinogram, using a voting strategy. This is applied by setting the value of each pixel of the output sinogram to the average value of the pixels from the different chunks that overlap in the corresponding pixel-position (Fig. 4.4).

4.4.8 The UDNN model and parameters

Apart from the hyperparameters associated with the stochastic gradient-base descent, two other important ones, the network's kernel dimensions and the feature map sizes per layer, have to be determined. Using the notation present in Section 4.3, the basic model of the network UDNN (Upscaling Deep Neural Network), has for its first layer a convolution with kernel size $f_{1h} \times f_{1w} = 10 \times 17$, with $N_1 = 64$ feature maps as well as padding (4, 8) accordingly for the height and the width. Its second layer has a kernel size of $f_{2h} \times f_{2w} = 7 \times 13$, $N_2 = 32$ feature maps and $(3, 6)$ for height and width padding. Finally, its last layer has kernel size of $f_{3h} \times f_{3w} = 7 \times 13$, $N_{out} = 1$ output channels (for an upscaling factor of $2x$), and $(3, 6)$ for padding. The number of feature maps for the UDNN model are set based on the network proposed by Dong *et al.* [33]; however, the size of the kernels are changed here. They are selected in an effort to be adequately large, so that network can accurately upscale the sinograms, but not excessively so, as that would affect the training and execution speed. They are also purposefully rectangular, as the presence of more significant contextual information is highly likely to be present along the x-$axis$ compared to the θ-$axis$ of the sinogram; which is true in a higher degree for the

case of our data, since in very undersampled sinograms the angle between the *x-axis* and the sinusoidal traces within the sinograms is acute. Additionally, the padding is selected appropriately so that the output feature maps in each layer have the correct size 4.3. The first layer the feature maps have the same number of columns as the input chuck and number of rows in reduced by one (from 11 goes to 10). Subsequently, the other two layers keep the size of their output feature maps the same as the first one. Based on this, it can be conjectured that further performance might be gained by exploring different feature numbers, sizes of kernels or training strategies. In an effort to explore some of these options, we perform in the following Section 4.5.1 experiments that alter two basic sets of hyperparameters, the number of feature maps in each layer and their kernel sizes respectively. The effects on the overall performance, after modifying these sets of hyperparameters are presented in the following Section 4.5.1.

4.5 Experiments

In this Section 3 different categories will be presented.

1. Experiments using real-world data examining the effects of feature map sizes and the kernel sizes on the network's accuracy and performance.
2. Experiments examining the network's accuracy on sinogram upscaling and tomogram reconstruction in synthetic datasets
3. Experiments examining the network's accuracy on sinogram upscaling and tomogram reconstruction in real-world datasets, both with 2x and 4x upscaling factors.

As metrics for the experiments we use the Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarity index metric (SSIM) [439]. The PSNR is a logarithmic representation of the mean square error, which is also the loss function selected for our network, displayed in equation 4.4. The PSNR, as shown in equation 4.5, represents the ratio between the maximum possible power of the signal, in our case the upscaled sinograms or the reconstructions of the undersampled tomograms, and the power of noise that affects its fidelity and distance it from the ground truth provided by the highly-sampled tomogram (More in Section 3.2). PSNR as its name implies, is a metric of the SNR of an image, but it

is also widely used to evaluate the image restoration quality of an image using a second image that due to its high quality it is considered as a ground truth. High PSNR values, which are expressed in a logarithmic decibel scale, signify higher restoration qualities. Below equation 4.5 describes how the PSNR metric is calculated.

$$PSNR\big(F(\boldsymbol{I};\boldsymbol{\Theta}),\boldsymbol{O}\big) = 20\log_{10}(MVV) - 10\log_{10}\big(MSE\big(F(\boldsymbol{I};\boldsymbol{\Theta}),\boldsymbol{O}\big)\big) = 20\log_{10}(MVV) - 10\log_{10}\big(\big\|F(\boldsymbol{I};\boldsymbol{\Theta}) - \boldsymbol{O}\big\|_2^2\big) \quad (4.5)$$

where $F(\boldsymbol{I};\boldsymbol{\Theta})$ is the networks predictions given an input $\boldsymbol{I}$, its parameters $\boldsymbol{\Theta}$, $\boldsymbol{O}$ is the ground truth and MVV is the maximum possible voxel value in the tomograms.

Additionally, as it is stated earlier we also use the SSIM [439] metric shown in equation 4.5. Contrary to PSNR and MSE metrics, which estimate images' SNR and visual quality based on absolute errors, the SSIM [439] metric is a perception based metric. Different to them, SSIM [439] considers image degradation as a cause of quality decrease, but ignores illumination and contrast alterations that that do not cause structural changes on what it is imaged. The structural term, refers to the strong inter-dependencies between spatially close pixels. SSIM [439] by estimating how much these inter-dependencies remain unchanged between the network's predictions and the ground truth, provided from the highly-sampled tomogram (More in Section 3.2), is able to measure the perceptual improvement in the quality. Below equation 4.6, describes how the SSIM [439] metric is calculated.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + (k_1L)^2)(2\sigma_{xy} + (k_2L)^2)}{(\mu_x^2 + \mu_y^2 + (k_1L)^2)(\sigma_x^2 + \sigma_y^2 + (k_2L)^2)} \quad (4.6)$$

Where $x = F(\boldsymbol{I};\boldsymbol{\Theta})$, $y = \boldsymbol{O}$, μ_x is the mean of $F(\boldsymbol{I};\boldsymbol{\Theta})$, μ_y is the mean of $\boldsymbol{O}$, σ_x is the variance of $F(\boldsymbol{I};\boldsymbol{\Theta})$, σ_y is the variance of $\boldsymbol{O}$, σ_{xy} is the covariance of $F(\boldsymbol{I};\boldsymbol{\Theta})$ and $\boldsymbol{O}$, L is the dynamic range of the pixel-values, and by default $k_1 = 0.01$ and $k_2 = 0.03$.

Both metrics are commonly used to quantitatively measure the visual quality of an imaging system (image or volume) compared to another that is held as the ground truth in terms of quality. In the experiments with the synthetic data in Section 4.5.2 and with the real-world data in Section 4.5.3 these metrics are used; additionally Figures 4.7 - 4.24 provide also a qualitative indication of our approach's performance.

4.5.1 Experiments for the hyperparameter selection and tuning

As it is mentioned earlier in Section 4.4.8, we perform experiments to examine the effects of different sets of hyperparameters on the accuracy of the network. In the experiments of this Section, we use real-world datasets, as they are described in Section 3.2 using the training setup described in Sections 4.4.3-4.4.7. For these experiments, Bertram's Directional Interpolation [22] is not used as a comparison, because the aim of these experiments is to compare the accuracy of different versions of the network (with different hyperparameters) between each other. Instead, cubic interpolation[21] is used in order to provide a comparative baseline. Additionally, only training and validation is performed, since the validation results can provide ample evidence for the performance of each versions of our network. Furthermore, in the following experiments we are only going to calculate the validation PSNR score(equation 4.5), since the calculation of SSIM [439] (equation 4.6) is reserved for the tomogram reconstructions, where the perseverance of structural information has to be accessed. Namely, we examine the:

1. Effects of the number of features on accuracy, where we examine if a larger or smaller number of features in the hidden layers can improve the accuracy.
2. Effects of the sizes of kernels on accuracy, similarly to examine if larger or smaller kernels can improve the overall accuracy.

The reason we choose to experiment with these particular sets of hyperparameters is because they are the most fundamental hyperparameters in a CNN architecture, and they can have a powerful effect on its accuracy. It can be speculated that additional accuracy might be gained by experimenting with other hyperparameters (e.g. more extensive range of features map sizes, kernel sizes, different learning rates, etc) or network architectures, but the exploration of these options would take a very long time, and we believe that the ones explored in this section are already substantial. For this reason these questions are left for future research.

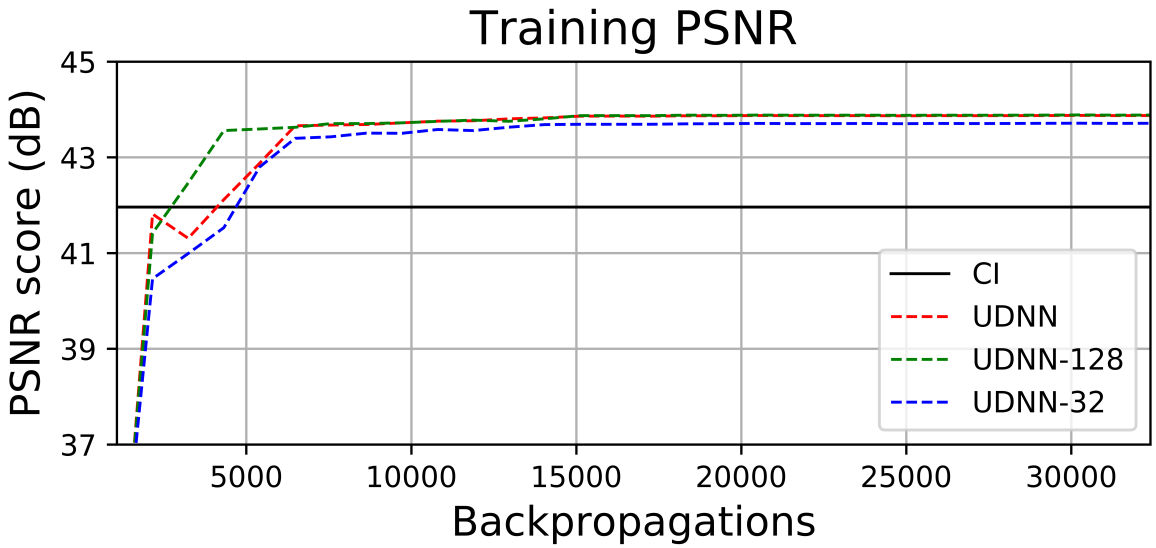
	UDNN (default)	UDNN-128	UDNN-32	CI
Validation PSNR	43.88 dB	**43.89** dB	43.71	42.76 dB

Table 4.1: Sinogram upscaling validation PSNR scores of UDNN (64-32 numbers of feature maps), UDNN-128 (128-64 numbers of feature maps), UNND-32 (32-16 numbers of feature maps) and cubic interpolation using real-world data. The validation PSNR scores is for the networks at their $32,400^{th}$ iteration using 10 random chunks from the $3^{rd}, 5^{th}, ..., 2159^{th}$ sinogram of the training/validation real-world tomogram. These chunks are downscaled to a height of 11 rows/projections from 401 (close to 40x downscale factor, imitating the 3601-to-91 downscaling if they were a whole sinogram) and then they are upscaled using the available methods to 21 rows/projections (2x upscaling), therefore creating the predictions for the rows $21, 61, ..., 381$ of the original chunk (intermediate row/projections). The PSNR for these predictions is then calculated using the row/projections from the original chunks as ground truth. Best results are shown in bold.

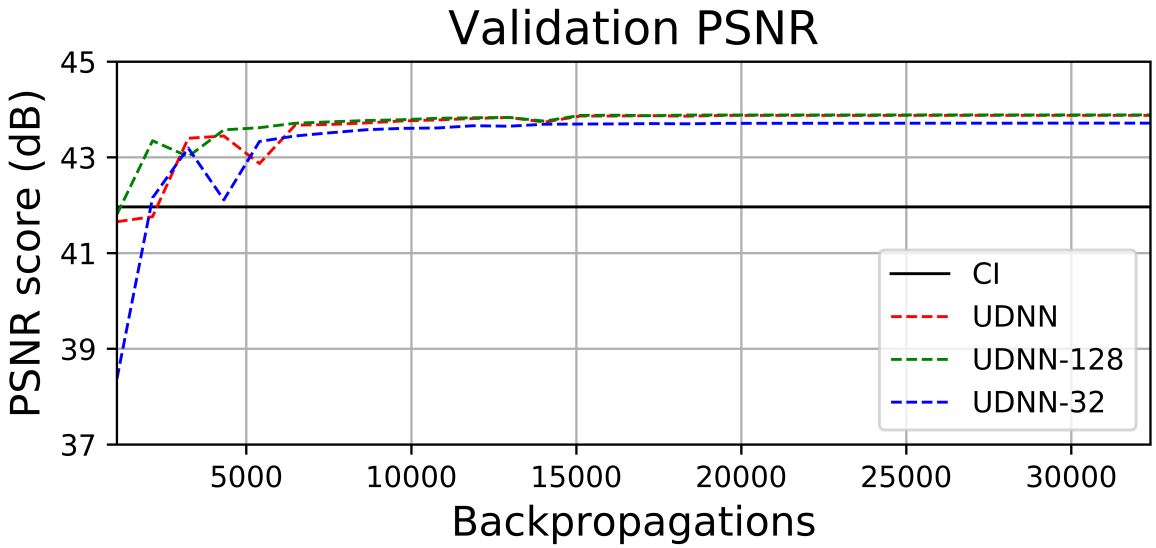
4.5.1.1 The effects of the number of features on performance

One of the most significant sets of a neural network's hyperparameters during it designing phase is the number of feature maps and therefore channels in each of the network's layer. For our UDNN model, as mentioned in Section 4.4.8, the default number of feature maps selected for the first layer is 64, and for the second 32. In this experiment, we will explore how the network performs when the number of its feature maps is doubled to 128 and 64 (UDNN-128) or it is halved to 32 and 16 (UDNN-32) respectively for the first two layers. Results are shown in Figure 4.5 .

Based on the validation PSNR scores (equation 4.5) present in Table 4.1 and the dynamic response in Figures 4.6(**a**),(**b**), we can conclude that the addition of more feature maps helps the network to converge to a higher PSNR value, but with diminishing returns. By diminishing returns, we imply that while the network's accuracy does increase (from $43.88dB$ of the default to 43.89 of the UDNN-128), but it does not reflect the increase in amount of the added feature maps in each of the layers (only a $0.01dB$ increase for doubling the feature maps). Because of that, in the later experiments we use the UDNN-128 and UDNN model, but we are not going to use or test a model with a higher number of feature maps. Furthermore, as it is expected the UDNN-32 model with the fewer number of feature maps converged with accuracy lower than the UDNN model. Therefore, we can also conclude that using less than 64-32 feature maps does not lead to satisfactory results.



(a)



(b)

Figure 4.5: *The training (**a**) and validation (**b**) PSNR graphs for the Cubic Interpolation (CI), the default UDNN using 64-32 feature maps, the UDNN using 128-64 feature maps (UDNN-128) and the UDNN using 32-16 (UDNN-32). The graphs depict the PSNR score from the initial 32,400 iterations; after this point the network has converged and the PSNR score plateaus. Use of less feature maps as seen by the UDNN-32 model, decreases the achieved models validation accuracy slightly and the additional feature maps used to create the UDNN-128 model offer diminishing returns. While standard deviation bars can not be presented due to the unavailability of multiple highly-sampled tomograms, the differences in accuracy (both training and validation) between UDNN-32, UDNN and UDNN-128 are very small, considering also the fact that between UDNN-128 and UDNN-32 there are 4 times more parameters used. Based on this, we conclude that there is no merit, in terms of the accuracy, to increase the feature maps per layer further than the one used for UDNN-128.*

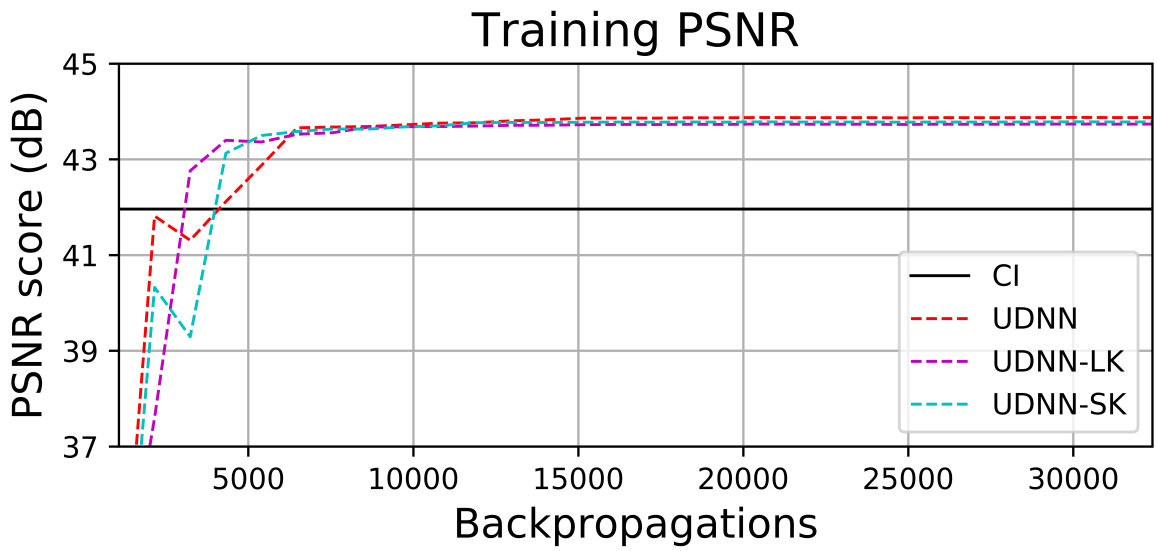
4.5.1.2 The effects of the sizes of kernels on performance

Here, we investigate another essential set of the network's hyperparameters: the size of the kernels in each of the network's layers. In the default UDNN model, as is displayed in Section 4.4.8, the kernels have size of 10×17, 7×13, 7×13, respectively for the first, second and third convolutional layer. In this experiment, two variations of the model are tested. One with larger kernel sizes (UDNN-LK), specifically 12×19, 9×15, 9×15 respectively for each of the three layers and one where the kernels are smaller (UDNN-SK), specifically 8×15, 5×11, 5×11 respectively for each of the three layers. The kernel sizes for UDNN-LK and UDNN-SK are set by adding or subtracting 2 columns and rows from those of the default UDNN model. The addition or subtraction by 2 is in order to preserve the parity of the default kernel sizes. Similar to 4.5.1.1, the purpose of this experiment is to observe the network's response and performance with larger or smaller kernels sizes, or in other words with larger or smaller perceptive fields in all of the layers. Results are shown in Figure 4.6 .

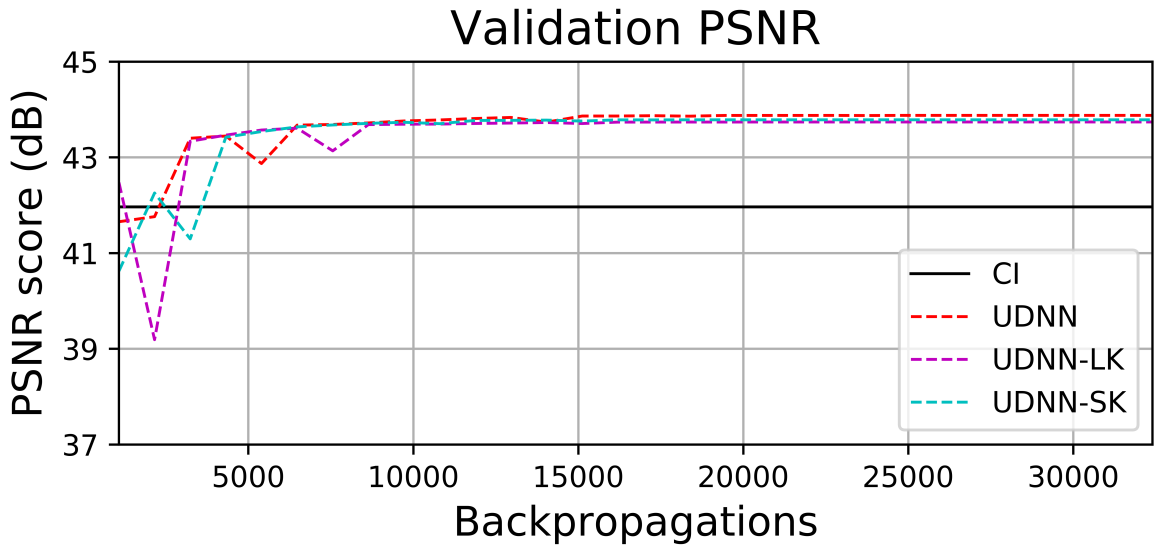
Based on the validation PSNR scores (equation 4.5) of the $32,400^{th}$ iteration, present in Table 4.2 and the Figures 4.6(**a**),(**b**) we can infer that the enlargement or shrinkage of the sizes of the kernels does not offer improved results as regards to the network's accuracy. This could perhaps be explained because with an increased kernel size the network receptive field increase, but further away elements don't contain valuable information for the task of upscaling the central pixel. However, considering that our sinograms contain

	UDNN(default)	UDNN-LK	UDNN-SK	CI
Validation PSNR	**43.88** dB	43.74 dB	43.78 dB	42.76 dB

Table 4.2: Sinogram upscaling validation PSNR scores of the default UDNN, UDNN-LK, UNND-SK and cubic interpolation using real-world data. The validation PSNR scores is for the networks at their $32,400^{th}$ iteration using 10 random chunks from the $3^{rd}, 5^{th}, ..., 2159^{th}$ sinogram of the training/validation real-world tomogram. These chunks are downscaled to a height of 11 rows/projections from 401 (close to 40x downscale factor, imitating the 3601-to-91 downscaling if they were a whole sinogram) and then they are upscaled using the available methods to 21 rows/projections (2x upscaling), therefore creating the predictions for the rows $21, 61, ..., 381$ of the original chunk (intermediate row/projections). The PSNR for these predictions is then calculated using the row/projections from the original chunks as ground truth. Best results are shown in bold.



(a)



(b)

Figure 4.6: *The training (**a**) and validation (**b**) PSNR graphs for the Cubic Interpolation (CI), the default UDNN, the UDNN using Larger Kernel sizes (UDNN-LK) and the one with Smaller Kernel sizes (UDNN-SK). The graphs depicts the PSNR score from the initial 32,400 epochs. The graphs doesn't show any further iterations as the network has converged and the PSNR score plateaus. Use of smaller or larger kernels (presented by UDNN-SK and UDNN-LK respectively) than the ones used for the UDNN model decrease the validation accuracy slightly. While standard deviation bars can not be presented due to the unavailability of multiple highly-sampled tomograms, the differences in accuracy (both training and validation) between UDNN-SK, UDNN and UDNN-LK are very small despite the large difference in kernel sizes especially between UDNN-LK and UDNN-SK. Based on this we conclude that there there is no merit, in terms of the accuracy, to increase or decrease UDNN's kernel sizes per layer.*

noise, it possible that extra noise propagating through the additional neurons added of the larger kernels, acts damaging for the overall accuracy of the network. On the other hand, smaller kernels perhaps miss information, which occasionally can be crucial for the upscaling of the central pixel, especially in our case where the angle difference between neighboring rows/projections of the input is large (in only 91 projections the sample is rotated 180 degrees).

4.5.1.3 Conclusion on hyperparameter tuning

Based on the above Sections 4.5.1.1 and 4.5.1.2, we can conclude that the additional feature maps or channels created in each of the network's convolutional layers can improve the network's overall accuracy, but with diminishing returns. Furthermore, it seems that the default sizes of kernels from the default UDNN model of Section 4.4.8 are already close their "ideal" and they are not going to be modified further. Naturally, there are preliminary experiments, that aided the selection of values of other hyperparameters and training approaches, but they are omitted from this chapter for purposes of being concise. Based on these experiments and those in Sections 4.5.1.1 and 4.5.1.2 we continue our experiments using the UDNN-128 version. Namely, in Section 4.5.2 our networks accuracy will be compared against state-of-the-art analytical approaches using synthetic data (More in Section 3.2) and in Section 4.5.3 again using real-world data.

4.5.2 Experiments with synthetic data

Having concluded with the experiments regarding the hyperparameter selection and tuning of the network, we proceed here with testing our approach against state-of-the-art analytical approaches using the synthetic data presented in Section 3.2. As is described in Section 4.1, due to our network's lightweight structure and emphasis on fast training and execution times that allow its re-usability on new varieties of datasets/tomograms, its comparison against analytical approaches is more appropriate that the comparison too heavyweight neural network approaches [153] methods that are not designed for frequent retraining and are designed for a narrow variety of imaged samples. For this purpose

we are going to compare our approach against Bertram *et al.*'s directional interpolation method [22] and also cubic interpolation [21].

In order to assess our approach on controlled conditions we initially perform some experiments using synthetic data created using Kazantsev *et al.*'s the TomoPhantom toolbox [23] (More in Section 3.2). With the term controlled conditions, we refer to the absence of Poisson noise and more significantly to the absence of temporal artifacts in the sinograms. (See Figure 3.5 Marker 1). Namely, during the collection of highly-sampled real-world tomograms it is possible that a number of temporal events to occur during their collection. This is because on occasion, temporal events might begin prematurely, when the high-sampled tomogram is captured before the collection of the time-series or not have seized completely, when the high-sampled tomogram is captured after the collection of the time-series. When this is true, it is also more likely that temporal events would occur during the collection of high-sampled tomograms, since their collection is slower than the collection of undersampled tomograms and some fast-occurring temporal events may manifest during their collection (See Figure 3.5 Marker 1). Since the state-of-the-art analytical interpolation approaches, which our method is compared against, are not designed to handle this type of sinogram artifacts, it possible to that they may underperform on real-world data. Therefore if the approaches where to be tested only on real-world data, it would be impossible to determine if any accuracy advantage of our method is only caused due to the special nature of the real-world data. By testing on synthetic data, it may offer an environment for more fair comparisons between the accuracy of our approach and the other state-of-the-art analytical interpolation approaches. Tests regarding synthetic datasets are only performed for x2 upscaling. Further upscaling factors are not tested for synthetic data, since the earlier scientific question regarding the absence of temporal artifacts in the sinograms can be answered by the x2 upscaling factor. Additionally, at it is stated by the authors the Bertram *et al.*'s [22] directional interpolation method is not designed for upscaling by using it iteratively. Therefore for x4 upscaling factor, our method can only be tested against cubic interpolation. While comparison only against the cubic interpolation for x4 upscaling is an interesting question that should be addressed,

we decided to perform it only on real-world data. This is also reasonable since the available ground truth from the real-world sinograms is as valid as the ground truth in the synthetic sinograms. Commonly, in deep learning applications, experiments that utilise synthetic data can be very helpful, since they may provide a more accurate ground truth. However, for the case of sinogram upscaling using highly-sampled real-world tomograms, the high number of projections present in them provide sufficient and accurate information, which can be considered equally valid ground truth both regarding their sinograms and their reconstructions, compared to the ground truth offered by synthetic tomograms. For these experiments, as stated earlier, we use the UDNN-128 version of our approach. Results regarding the sinogram upscaling are shown in Section 4.5.2.1 and regarding the improvement of the reconstructions' SNR and quality in Section 4.5.2.2.

4.5.2.1 Sinogram upscaling of synthetic data

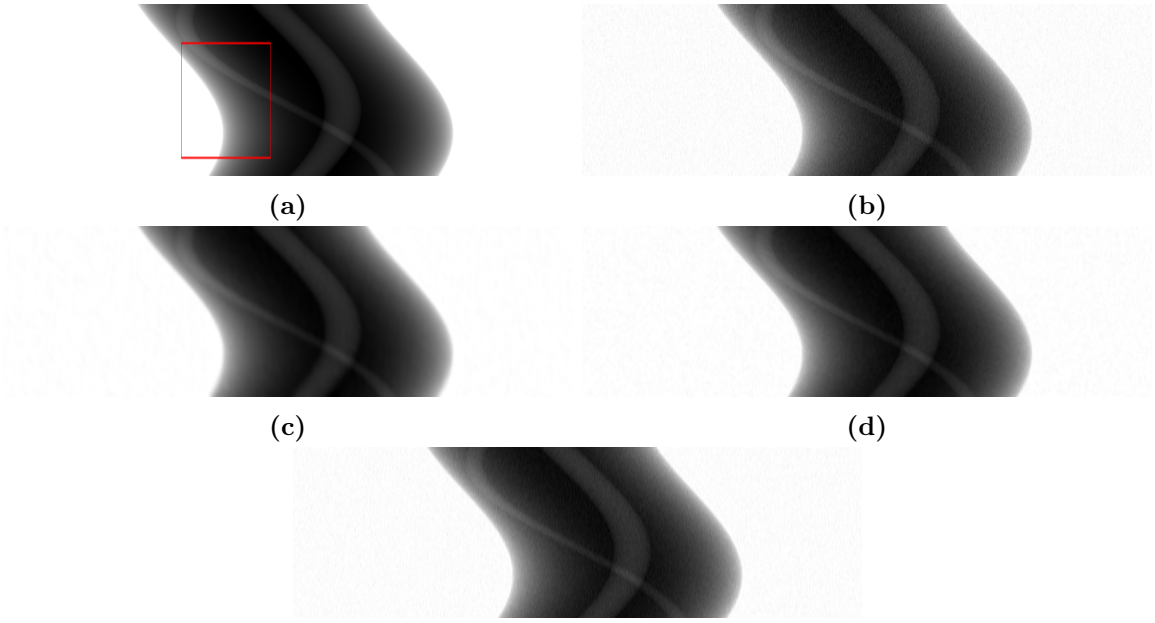
Following the training and testing procedure described in Section 4.4, we perform tests of our approach compared to Bertram *et al.*'s directional interpolation method [22] and cubic interpolation [21] using the synthetic data as they are described in Section 3.2. Visual results regarding the sinogram upscaling can be seen in Figures 4.7, 4.8, 4.9, and PSNR (equation 4.5) scores from testing can be seen in Table 4.3 . For the case of sinogram upscaling, only the PSNR metric is being used to compare the approaches. This due to the fact, that while high accuracy in sinogram upscaling is a good indication for the success of our approach, it is the improvement of the reconstruction quality that constitutes the criterion of its success. For this reason, we are going to only calculate both the PSNR and SSIM [439] (equation 4.6) metric during the reconstruction of the tomograms (synthetic or real-world) after the corresponding methods have been applied on their sinograms.

As it can be seen from Table 4.3 all sinogram upscaling approaches perform better than the noisy ground truth with the added Poisson noise, since they are designed to predict the signal and not the noise. Naturally, this is due to the simple structure of the synthetic data which are constructed using some simple geometrical objects, and it doesn't mean that a potential improvement in reconstruction quality can always be found by de-

2x Sinogram Upscaling	Noisy Proj.	CI	UDNN-128	DI
Testing PSNR	40.88 dB	42.53 dB	**48.43** dB	48.15 dB

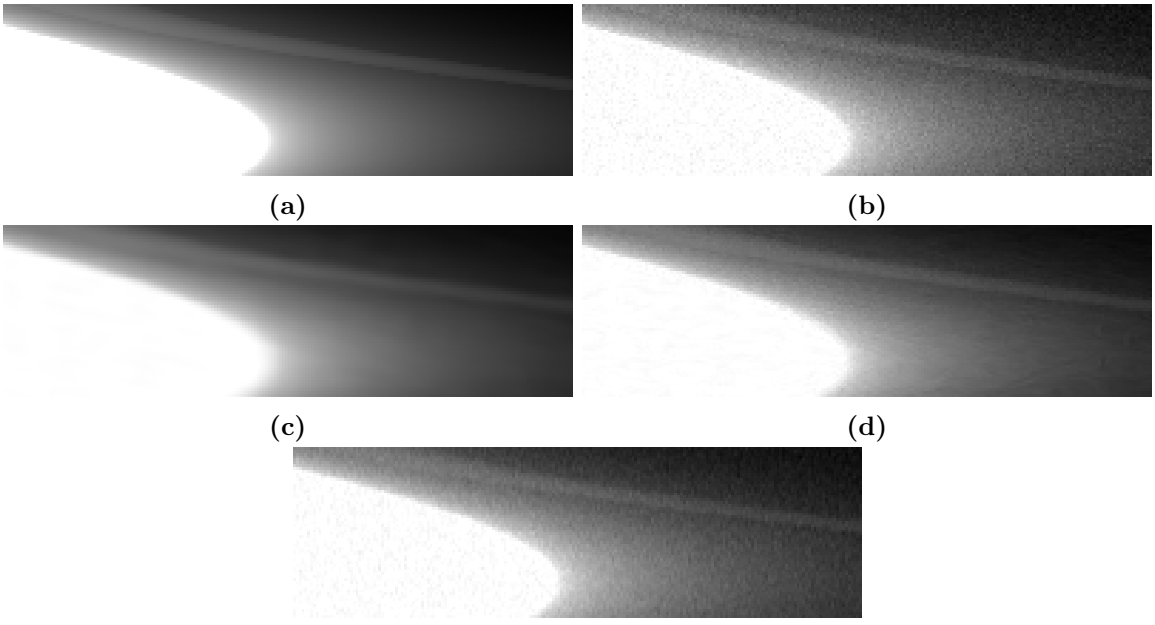
Table 4.3: PSNR score for upscaling the sinograms $1, 11, \cdots, 108$ of the synthetic dataset used for testing (Section 3.2, Figure 3.5). For the calculations, the intermediate row/projections (rows $21, 61, \cdots, 3581$) of the original sinograms without noise are used as ground truth. The Noisy Proj. is the PSNR from the error between the downscaled noisy sinograms (sinograms $1, 11, \cdots, 108$ and for rows) and the downscaled noiseless sinograms (Again sinograms $1, 11, \cdots, 108$ and rows $21, 61, \cdots, 3581$). For the other columns the sinograms $(1, 11, \cdots, 108)$ where downscaled to a height of 91 (or 91 projections) and then upscaled using the different methods to 181 (2x upscaling), therefore creating the predictions for the rows $21, 61, \cdots, 3581$ of the original (intermediate row/projections). The PSNR for these predictions is then calculated against the downscaled noiseless sinograms. Best result is shown in bold.

liberately downscaling and upscaling noisy sinograms. As it would be further discussed in Section 4.5.3.3, the interpolation approaches introduce structural errors to the tomograms, which after the use of an reconstruction algorithm can easily be detected using the SSIM [439] metric (equation 4.5). Besides this, it also can be observed that our UDNN-128 performs marginally better Bertram *et al.*'s directional interpolation method [22] as well as cubic interpolation [21]. While multiple executions of the same experiment with different testing tomograms are needed in order to calculate the standard deviation and verify that our method is always the one that performs better, the fact that Bertram *et al.*'s approach and ours are very close in terms of accuracy, offers a good indication about the comparability of the approaches' accuracy. As it will be explored in the later Section 4.6, our method is also faster than Bertram *et al.*'s for the use-case of time-series of undersampled tomograms. Therefore our approach is more preferable considering that it can offer accuracy comparable to the one of Bertram's. In Figures 4.7, 4.8 and 4.9 it can be seen that especially for our approach (Figure 4.7(**c**), 4.8(**c**), 4.9(**b**)) and slightly less for Bertram *et al.*'s approach (Figure 4.7(**d**), 4.8(**d**) and 4.9(**c**)) noise has been reduced significantly, without the introduction of any noticeable artifacts.



(a) **(b)** **(c)** **(d)** **(e)**

Figure 4.7: *The* 201^{th} *sinogram presenting only the intermediate row/projections (Rows* $21, 61, \cdots, 3581$*, pixel-value [e-1:1]), in order to compare what is generated by our UDNN approach with the other approaches. (**a**) From the original dataset without Noise, (**b**) From the original dataset with Noise, (**c**) Prediction for these rows using UDNN-128 for upscaling, (**d**) Prediction for these rows using the Bertram's Directional Interpolation, (**e**) Prediction for these rows using Cubic Interpolation. Each image ratio is different from the original for presentation purposes. The red region in A is shown zoomed in in Figure 4.8.*



(a) **(b)** **(c)** **(d)** **(e)**

Figure 4.8: *The selected area marked by the red window in Figure 4.7(**a**) enlarged. (width window [401:600] out of [1:1280] and for rows* $821, 861, \cdots, 3221$*, pixel-value [e-1: 1]). (**a**) From the original dataset without Noise, (**b**) From the original dataset with Noise, (**c**) Prediction for these rows using UDNN-128 for upscaling, (**d**) Prediction for these rows using the Bertram's Directional Interpolation, (**e**) Prediction for these rows using Cubic Interpolation. Note the clarity of the proposed method (panel (**c**)). Each image ratio is different from the original for presentation purposes.*

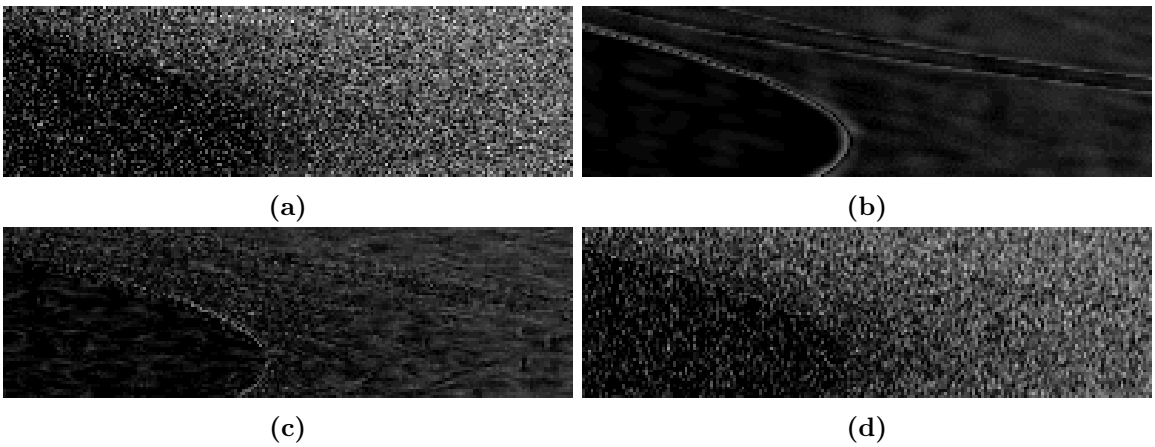


Figure 4.9: *The absolute value of the difference in sinogram intensity of Figure 4.8(**b**-**e**) against Figure 4.8(**a**). (**a**) The absolute value of the difference between the noisy synthetic sinogram of Figure 4.8(**b**) and the noiseless synthetic sinogram of Figure 4.8(**a**), (**b**) The absolute value of the difference between the UDNN-128 predictions of Figure 4.8(**c**) and the noiseless synthetic sinogram of Figure 4.8(**a**), (**c**) The absolute value of the difference between Bertram's Directional Interpolation predictions of Figure 4.8(**d**) and the noiseless synthetic sinogram of Figure 4.8(**a**), (**d**) The absolute value of the difference between the Cubic Interpolation predictions of Figure 4.8(**e**) and the noiseless synthetic sinogram of Figure 4.8(**a**). Each image ratio is different from the original for presentation purposes. The brighter the regions in the figures, the more they differ from the ground truth sinogram of Figure 4.8(**a**). As can be seen our approach has the less bright regions compared to the other approaches, meaning that it upscales the sinograms with less error, but as seen it is less accurate along the edges of the sinusoidal traces that the projected objects create in the sinogram.*

4.5.2.2 Reconstruction of synthetic data

Moving on, we then reconstruct the upscaled sinograms produced by the corresponding approaches in the previous Section 4.5.2.1, using the Filter Back Projection (FBP) [409] reconstruction algorithm provided in the Astra toolbox [440]. The reason we are not using more elaborate iterative reconstruction algorithms like SART [411], SIRT [412] or CGLS [413], is because we wanted to observe how the different upscaling approaches contribute in the quality improvement, without the aid of any sophisticated reconstruction algorithm, as they are by their very nature adaptive, and can hide changes in the underlying data. FBP is simply a numerical transform to the data, and therefore allows an accurate representation of the improvements made by the upsampling algorithm. Visual results can be seen in Figures 4.10 and 4.11, while PSNR and SSIM [439] (equations 4.5, 4.6) scores from testing can be seen in Table 4.3.

As it can be seen from Table 4.4, similar to Table 4.3 all approaches perform better in

2x Sinogram Upscaling	181 Noisy Proj.	CI	UDNN-128	DI	91 Noisy Proj.
Testing PSNR	14.82 dB	15.67 dB	**17.67** dB	17.40	11.67 dB
Testing SSIM [439]	0.0391	0.0454	**0.0625**	0.0607	0.0211

Table 4.4: PSNR and SSIM [439] score of the reconstructions using the sinograms $1, 11, \cdots, 108$ of the synthetic dataset used for testing (Section 2.3, Figures 2D and 3D). For the calculations, a reconstruction of the original sinograms without noise (3601 projections) is used as ground truth. The 181 and 91 Noisy Proj. correspond to reconstructions made from sinograms that are only downscaled accordingly from the original. The other columns correspond to reconstructions made by these sinograms, where they are initially downscaled to a height of 91 (or 91 projections) and then upscaled to 181 (2x upscaling) with the available methods. Best result for each metric highlighted in bold.

the PSNR metric, than the reconstruction of the 20x and not 40x downsampled (181 projections) noisy sinograms, following the pattern of Section 4.5.2.1. This fact is also true for the SSIM [439] metric, but it would be later shown in Section 4.5.3.3 that this is due to the simple and easily predictable structure of the synthetic data. Moreover, it can be seen that for both metrics, our approach UDNN-128 performs marginally better than both Bertram *et al.*'s directional interpolation method [22] and cubic interpolation [21]. While multiple executions of the same experiment with different testing tomograms are needed in order to calculate the standard deviation and verify that our method is always the one that performs better, the fact that Bertram *et al.*'s approach and ours are very close in terms of accuracy, offers a good indication about the comparability of the approaches' accuracy. As it will be explored in the later Section 4.6, our method is also faster than Bertram *et al.*'s for the use-case of time-series of undersampled tomograms. Therefore our approach is more preferable considering that it can offer accuracy comparable to the one of Bertram's. In Figures 4.10 and 4.11, there is a noticeable noise reduction in both for our approach (Figure 4.10(**c**) and 4.11(**c**)) and Bertram *et al.*'s directional interpolation approach (Figure 4.10(**d**) and 4.11(**d**)); however our approach contains notably less rotating artifacts, which appear here when the data jitter along curved traces, and are not properly reconstructed.

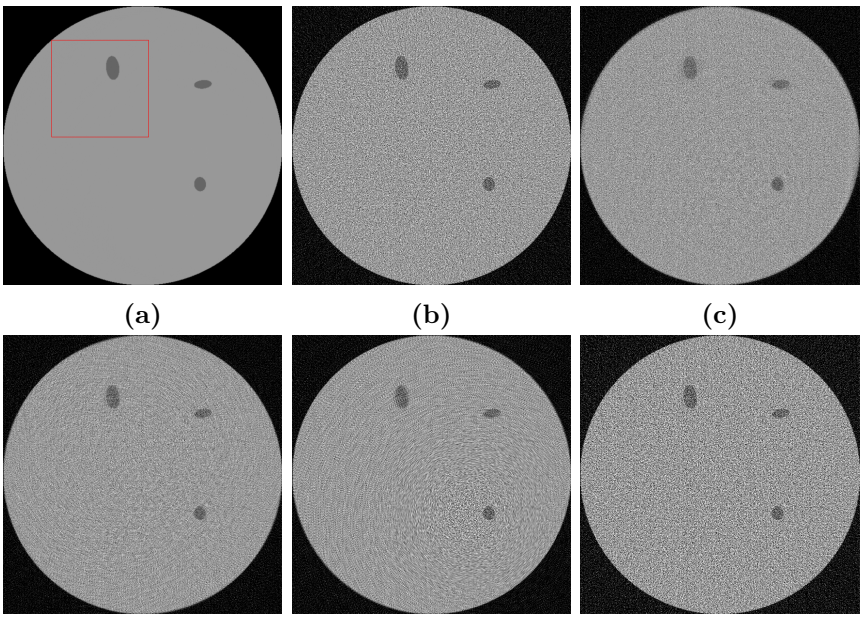


Figure 4.10: *The reconstruction of the 601^{th} sinogram (area [273:848, 273:848] of [1:1280,1:1280], pixel-value [0:0.5]) (**a**) From the original dataset without Noise and 3601 projections, (**b**) From the original dataset with Noise and 181 projections, (**c**) Using UDNN-128 for upscaling (181 projections), (**d**) Using the Bertram's Directional Interpolation (181 projections), (**e**) Using Cubic Interpolation (181 projections), (**f**) From the original dataset with Noise and 91 projections. Region of interest shown zoomed in Figure 4.11 .*

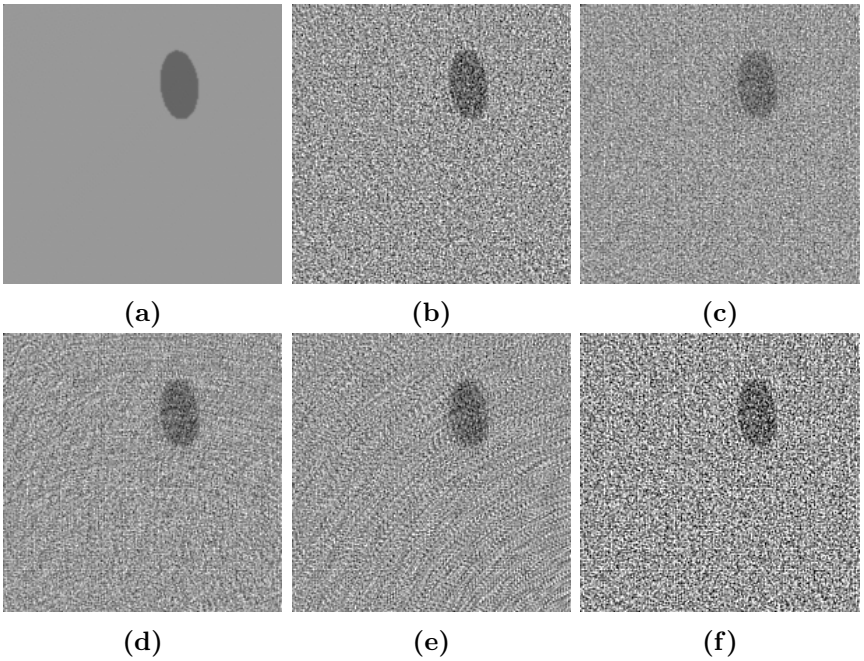


Figure 4.11: *The selected area marked by the red window in Figure 4.10(**a**) enlarged. (area [373:573, 343:543] of [1:1280,1:1280], pixel-value [0:0.5]). (**a**) From the original dataset without Noise and 3601 projections, (**b**) From the original dataset with Noise and 181 projections, (**c**) Using UDNN-128 for upscaling (181 projections), (**d**) Using the Bertram's Directional Interpolation (181 projections), (**e**) Using Cubic Interpolation (181 projections), (**f**) From the original dataset with Noise and 91 projections. With the proposed method (panel (**c**)) the presence of noise is decreased and there are less rotating artifacts, however there is also a slight decrease of contrast.*

4.5.3 Experiments with real-world data

In addition to synthetic data experiments, we also tested our method against Bertram *et al.*'s directional interpolation method [22] and cubic interpolation [21], using data provided by the Diamond Light Source (more on the data in Section 3.2). Again in these experiments we are going to use the UDNN-128 version of our model.

4.5.3.1 Experimental complexities

During the upscaling of the sinograms of real-world tomograms, especially when they are part of a tomogram time-series, there are two important issues which can arise.

The first issue is the translation of the rotation axis between the different tomograms that are part of a tomogram time-series. In sinogram space, this is expressed as a shift of the collected data along the *x-axis*, which for the case of our datasets collected at Diamond Light Source is within a margin of 5 pixels. However, due to the nature of CNNs, data translations across the *x-axis* do not pose any threat for the network's overall accuracy. This is due to the kernels of the respective convolutional layers of the network, operating like sliding windows during the operation of convolution, which sets the whole network translationally invariable.

The second issue, is that the sample might be shifted perpendicular to the orientation of the detector, the *z-axis*. As a result it can change the shape and tilt of the sinusoidal lines within the collected sinograms. While this could pose a problem for the correct upscaling of the sinograms, since our approach uses a learned prior provided from one only tomogram, something like this is not observed during the experiments. Moreover, the tomogram used for testing during the experiments with real-world data is a completely different tomogram, with different position for its rotation axis both along the *x-axis* and the *z-axis*, from the one used in training. We speculate that this behavior is due to the light-structure of our network, which prohibits overtraining despite using just one tomogram as the source for training. This could leave the door open for potential recycling of the trained network in tomograms of time-series that the training tomogram is not part of, as long as these tomograms depict samples of similar nature.

Regardless of the previous issues though, in most cases researchers during the collection of time-series of tomograms (4D datasets) strive to minimise the movement of the sample as it can jeopardise the accuracy of the later processing algorithms applied on the tomograms.

4.5.3.2 Sinogram upscaling of real-world data

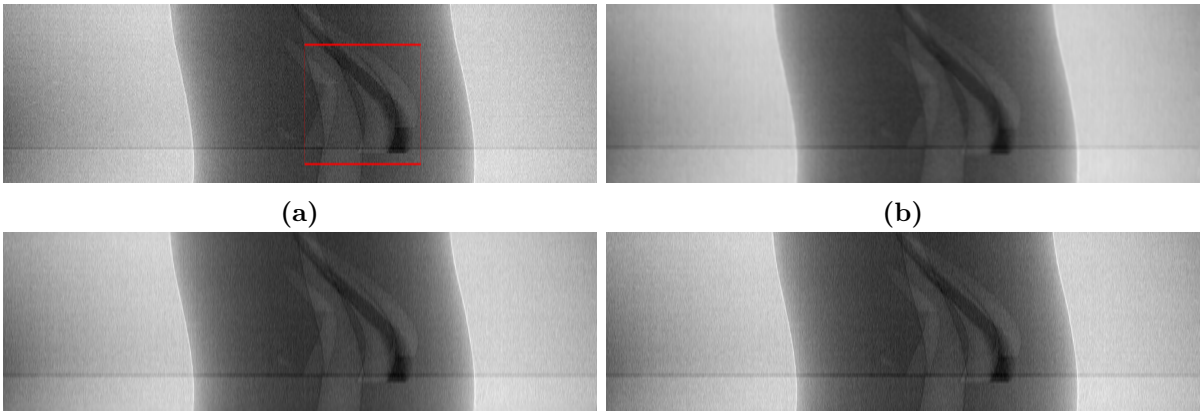
Moving forward in this Section, we perform sinogram upscaling of the real-world tomograms described in Section 3.2, while following the training and testing procedures described in Section 4.4. Our approach is compared to Bertram *et al.*'s directional interpolation method [22] and cubic interpolation [21], and similar to the Section 4.5.2.1 we only use of the PSNR metric (equation 4.5) for the upscaling of the sinograms. Visual results regarding the sinogram upscaling can be seen in Figures 4.12, 4.13 and 4.14, and PSNR scores from testing can be seen in Table 4.5.

Furthermore, using the modifications mentioned in Sections 4.4.5 we also test our network using a higher upscaling factor (of 4x), by adding additional output channels in the final convolutional layer our network, $N_{out} = 3$. The reason we jump from 2x to 4x and skip the 3x, is because it is operationally easier based on the choices made in Section 4.4.4. This due to the fact that, the chunks' height is 400+1 rows, which is divisible by 4 (ignoring the +1). For the case of a 3x upscaling factor, a different chuck height would have to be selected, and then new chunks would have to be elected and shuffled before their use for the network's training. In this 4x upscaling experiment, we would not apply Bertram *et al.*'s [22] directional interpolation method as it is not designed for upscaling by using it iteratively. The results are presented in Table 4.6 and in Figures 4.15, 4.16, 4.17.

As can be seen from Table 4.3 all sinogram upscaling approaches perform better than the synthetic ground truth with noise, which is the intermediate pixel-rows/ projections from the highly-sampled synthetic tomogram with Poisson Noise (More in Section 3.2). However, unlike Section 4.5.2.1, for the real-world data there is not a denoised version of the highly-sampled tomogram, hence Table 4.5 is one column shorter than Table 4.3. As can be seen in Figures 4.12, 4.13 and more easily in Figure 4.14, Bertram *et al.*'s [22] approach and the cubic interpolation [21] method appear to have predictions with slightly

more noise compared to our approach. This is true, since Figure 4.14(**a**) appears to be slightly darker than 4.14(**b**,**c**) and therefore the absolute value of the difference from the ground truth is smaller.

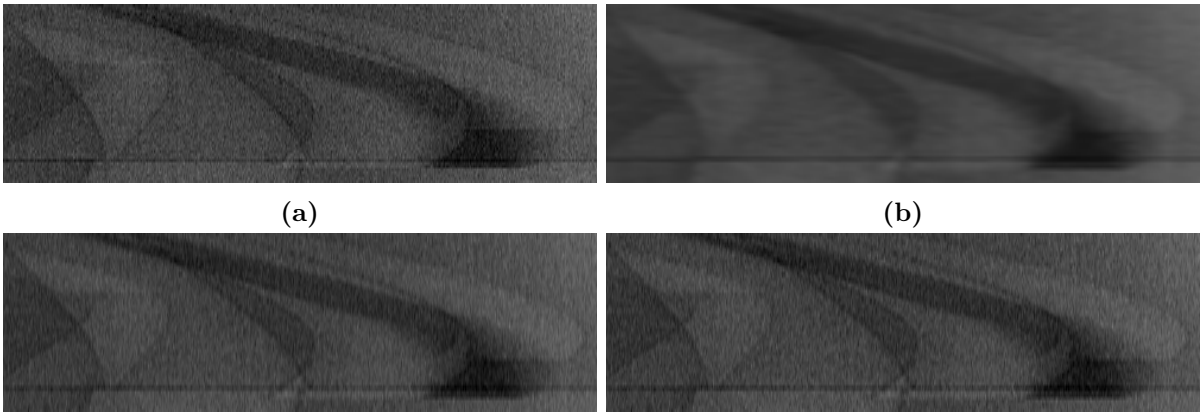
Additionally, by observing Table 4.6 against Table 4.3, the accuracy difference between our approach and Bertram's *et al.*'s [22] is still small, but slightly higher for the real-world experiments. This can potentially be attributed to the additional challenge presented in our datasets, where apart from noise there are also imaging artifacts (the line artifact in the lower part of Figures 4.12(**a**)-(**d**)) and abrupt changes due to temporal events occurring during capture (sinusoidal traces suddenly stop in the lower part of Figures 4.12(**a**)-(**d**)). Naturally, multiple executions of the same experiment with different testing tomograms are needed in order to calculate the standard deviation and verify that our method is always the one that performs better, the fact that Bertram *et al.*'s approach and ours are very close in terms of accuracy, offers a good indication about the comparability of the approaches' accuracy. As it will be explored in the later Section 4.6, our method is also faster than Bertram *et al.*'s for the use-case of time-series of undersampled tomograms. Therefore our approach is more preferable considering that it can offer accuracy comparable to the one of Bertram's. Regarding, the upscaling with a 4x upscaling factor, as it can be seen in Table 4.5 and Table 4.8, our network's accuracy decreases only by a slight amount (versus 2x upscaling), which in conjunction with the qualitative results provided from Figures 4.15, 4.16 and 4.17, encourages the notion that our network could be further used for upscaling with higher rates, however to truly access the scalability of our approach the corresponding tomogram reconstructions have to also be accessed, which is presented in the next Section.



(a) **(b)**

(c) **(d)**

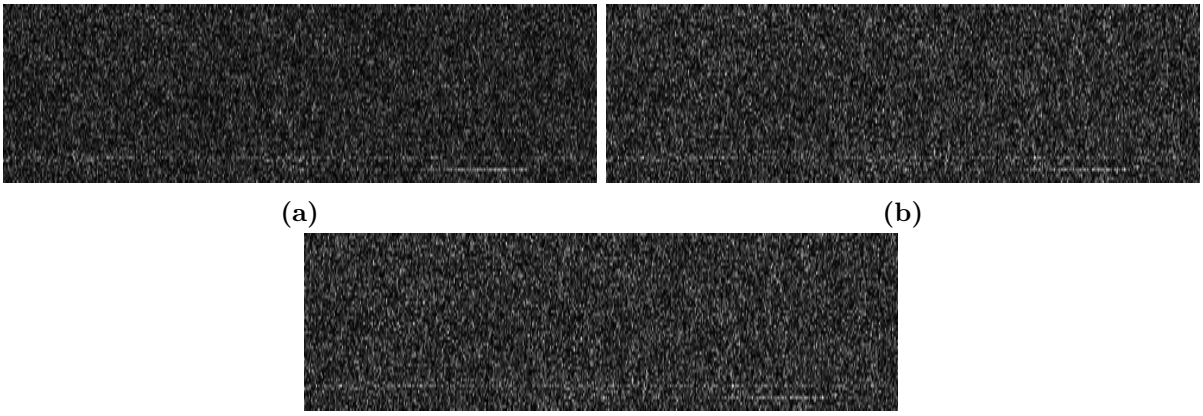
Figure 4.12: *The* 1002^{th} *sinogram presenting only the intermediate row/projections (Rows* $21, 61, \cdots, 3581$*, pixel-value [0.59:0.74]), in order to compare what is generated by our UDNN approach with the other approaches. (**a**) From the original dataset, (**b**) Prediction for these rows using UDNN-128 for upscaling, (**c**) Prediction for these rows using the Bertram's Directional Interpolation, (**d**) Prediction for these rows using Cubic Interpolation. Each image ratio is different from the original for presentation purposes. Region of interest is shown zoomed in Figure 4.13 .*



(a) **(b)**

(c) **(d)**

Figure 4.13: *The selected area marked by the red window in Figure 4.12(**a**) enlarged. (width window [1301:1800] out of [1:2560] and for rows* $821, 861, \cdots, 3221$*, pixel-value [0.59:0.74]). (**a**) From the original dataset, (**b**) Prediction for these rows using UDNN-128 for upscaling, (**c**) Prediction for these rows using the Bertram's Directional Interpolation, (**b**) Prediction for these rows using Cubic Interpolation Similar to Figure 4.12. Similar to Figure 4.8 with the proposed method (panel **b**) there is less noise introduced into the sinograms. Each image ratios is different from the original for presentation purposes.*



(a) **(b)**

(c)

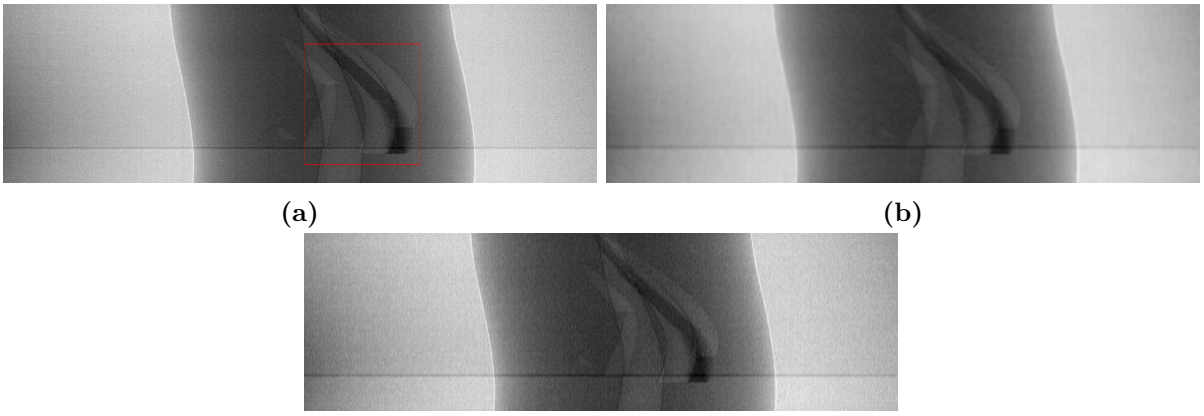
Figure 4.14: *The absolute value of the difference in sinogram intensity of Figure 4.13(**b**-**e**) against Figure 4.13(**a**). (**b**) The absolute value of the difference between the UDNN-128 predictions of Figure 4.13(**c**) and the real-world sinogram of Figure 4.13(**a**), (**c**) The absolute value of the difference between Bertram's Directional Interpolation predictions of Figure 4.13(**d**) and the real-world sinogram of Figure 4.13(**a**), (**d**) The absolute value of the difference between the Cubic Interpolation predictions of Figure 4.13(**e**) and the real-world sinogram of Figure 4.13(**a**). Each image ratio is different from the original for presentation purposes. The brighter the regions in the panels above the more they differ from the ground truth sinogram of Figure 4.13(**a**). Similar to Figure 4.9, from the above panels it can be seen that our approach is slightly less bright than Bertram's Directional Interpolation and the cubic interpolation approaches, meaning that it upscales the sinograms with slightly less error.*

2x Sinogram Upscaling	CI	UDNN-128	DI
Testing PSNR	42.20 dB	**43.90** dB	42.59 dB

Table 4.5: PSNR score for upscaling the sinograms $2, 12, \cdots, 217$ of the real-world dataset used for testing (Section 2.3, Figures 2B and 3B). For the calculations the intermediate row/projections of the original sinograms are used as ground truth. Namely, the sinograms $(2, 12, \cdots, 217)$ where downscaled to a height of 91 (or 91 projections) and then upscaled using the available methods to 181 (2x upscaling), therefore creating the predictions for the rows $21, 61, \cdots, 3581$ of the original (intermediate row/projections). The PSNR for these predictions is then calculated against the downscaled noiseless sinograms. Best results are shown in bold.

4x Sinogram Upscaling	CI	UDNN-128
Testing PSNR	42.01 dB	**43.89** dB

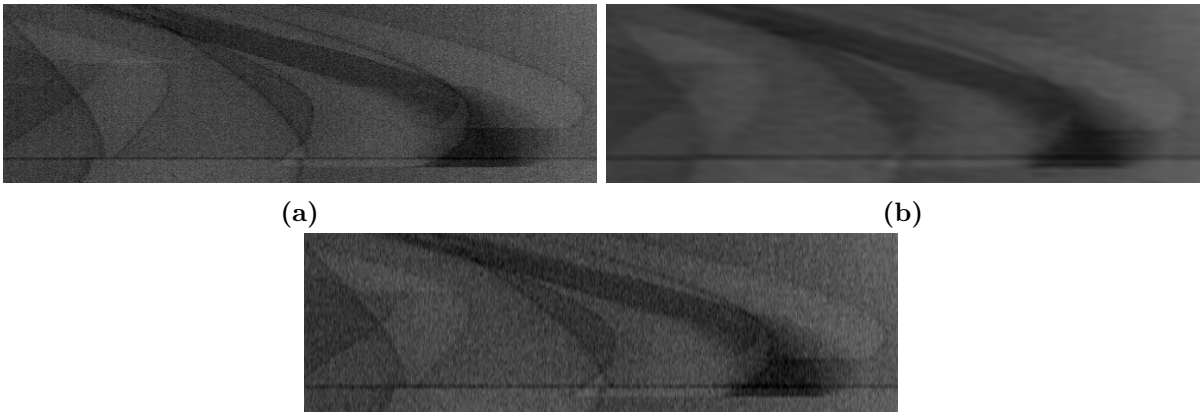
Table 4.6: PSNR score for upscaling the sinograms $2, 12, \cdots, 217$ of the real-world dataset used for testing (Section 2.3, Figures 2B and 3B). For the calculations the intermediate row/projections of the original sinograms are used as ground truth. Namely, the sinograms $(2, 12, \cdots, 217)$ where downscaled to a height of 91 (or 91 projections) and then upscaled using the available methods to 361 (x4 upscaling), therefore creating the predictions for the rows $21, 61, \cdots, 3581$ of the original (intermediate row/projections). The PSNR for these predictions is then calculated against the downscaled noiseless sinograms. Best result shown in bold.



(a) **(b)**

(c)

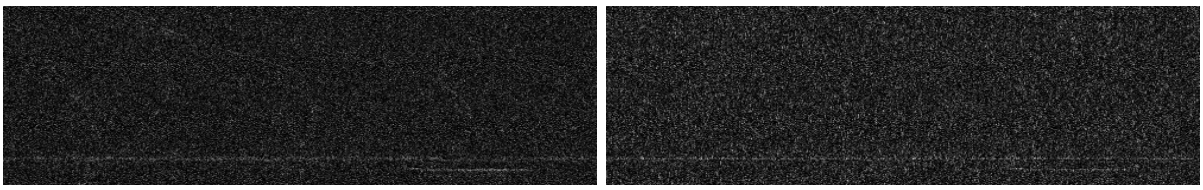
Figure 4.15: *The* 1002^{th} *sinogram presenting only the intermediate row/projections (Rows* $11, 21, 31, 51, 61, 71, \cdots, 3571, 3581, 3591$*, pixel-value [0.59:0.74]), in order to compare what is generated by our UDNN approach with the other approaches. (**a**) From the original dataset, (**b**) Prediction for these rows using UDNN-128 with 3 output channels, (**c**) Prediction for these rows using using Cubic Interpolation. Region of interest is shown zoomed in Figure 4.16 .*



(a) **(b)**

(c)

Figure 4.16: *The selected area marked by the red window in Figure 4.15(**a**) enlarged. (width window [1301:1800] out of [1:2560] and for rows* $811, 821, 831, 851, \cdots, 3211, 3221, 3231$*, pixel-value [0.59:0.74]). (**a**) From the original dataset, (**b**) Prediction for these rows using UDNN-128 with 3 output channels, (**c**) Prediction for these rows using using Cubic Interpolation. Again with the proposed method (panel (**b**)) there is less noise introduced into the sinograms. Each image ratio is different from the original for presentation purposes.*



(a) **(b)**

Figure 4.17: *The absolute value of the difference in sinogram intensity of Figure 4.16(**b**-**c**) against Figure 4.16(**a**). (**a**) The absolute value of the difference between the UDNN-128 predictions of Figure 4.16(**b**) and the real-world sinogram of Figure 4.16(**a**), (**b**) The absolute value of the difference between the Cubic Interpolation predictions of Figure 4.16(**c**) and the real-world sinogram of Figure 4.16(**a**). Each image ratio is different from the original for presentation purposes. The brighter the regions in the figures, the more they differ from the ground truth sinogram of Figure 4.16(**a**). As seen from approach from the above figures our approach is slightly less bright than Bertram's Directional Interpolation and the cubic interpolation approaches, meaning that it upscales the sinograms with slightly less error.*

4.5.3.3 Reconstruction of real-world data

Concluding with the real-world experiments, we reconstruct the upscaled sinograms produced from the corresponding approaches in the previous Section 4.5.3.2, using the Filter Back Projection (FBP) [409] reconstruction algorithm, provided by the Savu Framework [438] and its AstraReconGpu plug-in [440]. Similar to Section 4.5.2.2, we are not going to use an elaborate iterative reconstruction algorithm like SART [411], SIRT [412] or CGLS [413], for the tomogram reconstruction. For the calculation of the SSIM [439] metric for the real-world tomogram, we only used a window of pixels [501:1500, 501:1500], where the two ranges correspond respectively to the *x-axis* and *z-axis* of the tomogram reconstructions. Regarding the *y-axis*, since in Section 4.4.6 it is stated that the $2^{th}, 12^{th}, 22^{nd}, ..., 2152^{nd}$ sinograms are used for testing, and so the PSNR and SSIM metrics are calculated only for the pixels with *y-index* equal to $2, 12, 22, ..., 2152$. This reason we elect to use this window is because the AstraReconGpu plug-in [440], produces reconstruction data for a tomogram only within a cylindrical area inscribed to the output reconstruction volume, which is a rectangular cuboid. Since the voxels outside this area are occupied by NaN (Not a Number; missing data) elements, we elect the earlier window, which does not contain any NaN elements and it is large enough to be representative of the entire volume. Visual results can be seen in Figures 4.18 and 4.19, while PSNR and SSIM [439] (equations 4.5, 4.6) scores from testing can be seen in Table 4.7.

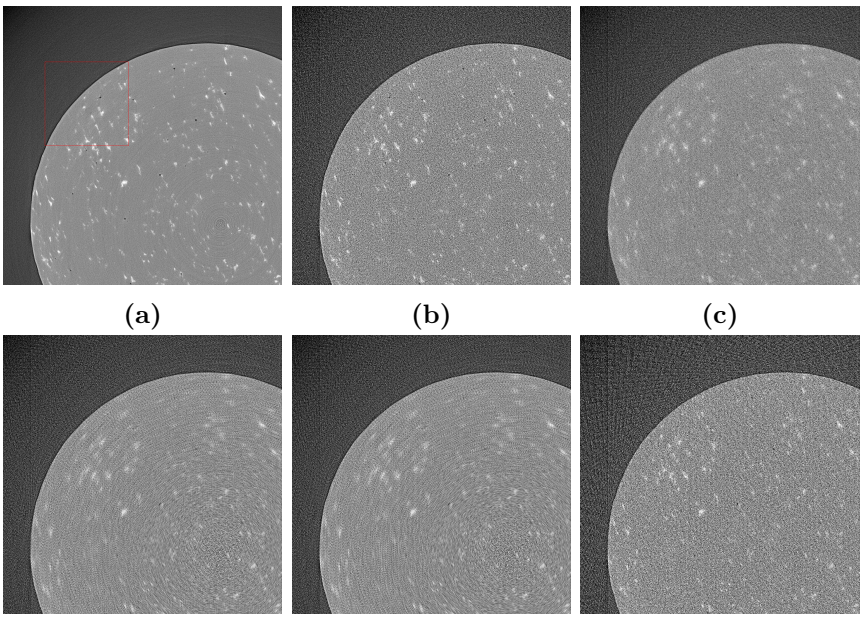
Additionally, we also perform the reconstruction of the upscaled sinograms that use the higher upscaling factor 4x, calculated in Section 4.5.3.2. Similarly, we do not apply Bertram *et al.*'s [22] directional interpolation method as it is not designed for upscaling by using it iteratively. The results are presented in Table 4.8 and in Figures 4.20, 4.21. Finally, we conclude this Section with the Figure 4.22, that presents the previous testing PSNR and SSIM [439] scores presented in Tables 4.7 and 4.8 using two bar charts. The PSNR scores are presented in Figure 4.22(**a**) and the SSIM [439] scores in Figure 4.22(**b**). By presenting the earlier PSNR and SSIM scores from testing as bar charts, our network's effect on quality improvement can be visually quantified and assessed, as the upscaling rate and therefore the number of projections increases.

As it can be seen from Table 4.7, similar to Table 4.5 all approaches perform better than the FBP reconstruction using 181 real-world projections for the same reason as in Section 4.5.3.2; however as it is stated in Section 4.5.2.2 this is only true for the PSNR scores and not the SSIM [439] scores. Since in real-world tomograms, there are challenging elements in addition to the Poisson noise, such as imaging artifacts and abrupt changes due to temporal events (more in Section 4.5.3.2). Analytical methods are not designed to perform well in these conditions, potentially resulting in loss of structural information which cannot be predicted by them. Despite the challenging nature of the data though, our approach is able to perform equally good to these approaches, by utilising a learned prior that aids better adjustment for these conditions. This allows it to improve the reconstruction's SNR, both in terms of absolute error and perceptual quality. While multiple executions of the same experiment with different testing tomograms are needed in order to calculate the standard deviation and verify that our method is always the one that performs better, the fact that Bertram *et al.*'s approach and ours are very close in terms of accuracy, offers a good indication about the comparability of the approaches' accuracy. As it will be explored in the later Section 4.6, our method is also faster than Bertram *et al.*'s for the use-case of time-series of undersampled tomograms. Therefore our approach is more preferable considering that it can offer accuracy comparable to the one of Bertram's.

2x Sinogram Upscaling	181 Proj.	CI	UDNN-128	DI	91 Proj.
Testing PSNR	19.03 dB	19.69 db	**21.79** dB	20.13 dB	15.94 db
Testing SSIM [439]	0.1737	0.1504	**0.2122**	0.1627	0.0936

Table 4.7: PSNR score for reconstruction the upscaled sinograms $2, 12, \cdots, 217$ of the dataset with real-world data used for testing (Section 2.3, Figures 2B and 3B). For the calculations, a reconstruction of the original sinograms (3601 projections) is used as ground truth. The 181 and 91 Noisy Proj. correspond to reconstructions made from sinograms that are only downscaled accordingly from the original. The other columns correspond to reconstructions made by these sinograms, where they are initially downscaled to a height of 91 (or 91 projections) and then upscaled to 181 (2x upscaling) with the available methods. Best result for each metric highlighted in bold.

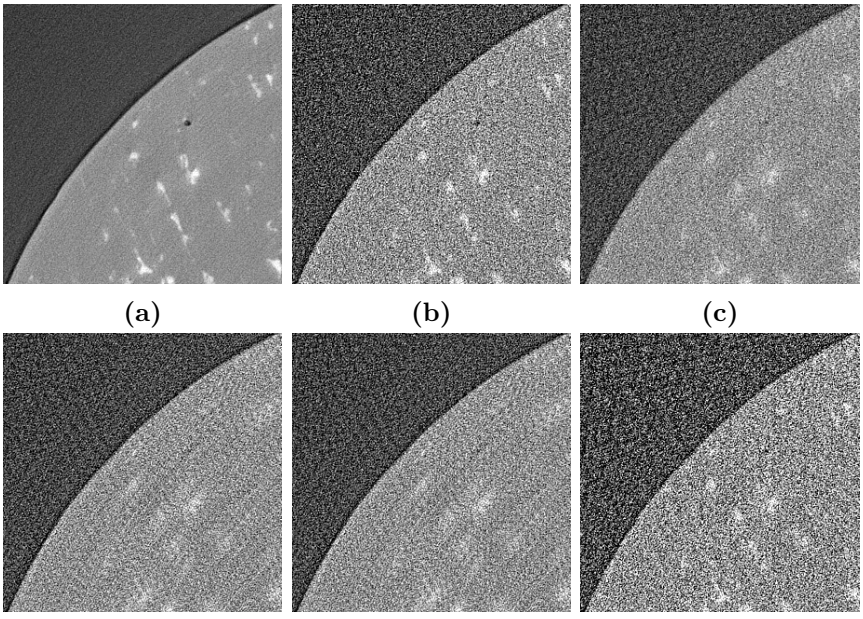
In Figure 4.19 it can observed that our approach provides a solution, both better in terms of noise and the appearance of rotating artifacts. As regards to the upscaling with a 4x upscaling factor, as it can be seen in Table 4.7 and Table 4.8, that the additional projections predicted either by the UDNN-128 or the cubic interpolation [21], increased even further the reconstruction quality both for the PSNR and SSIM [439] metrics, despite the slightly worse performance of the PSNR metric for the same transition (2x to 4x) during the sinogram upscaling. This can be attributed to the high importance of additional projection for acquiring tomogram reconstructions with higher SNR. However, as it can be seen in Figures 4.20 and 4.21 and specifically by comparing Figures 4.19(**c**) and 4.21(**b**), in the reconstruction of 4x upsampled tomograms, despite the improved metrics, we can detect the existence of more rotating artifacts combined with reduced contrast for the internal components of the imaged sample. Therefore, the improvement in both PSNR and SSIM [439] testing scores is mainly attributed to the reduction of noise. Finally, Figure 4.22 visually quantifies the improvement of the reconstruction quality both for the PSNR and the SSIM [439] metrics, by performing sinogram upscaling using our approach.



(a) **(b)** **(c)**

(d) **(e)** **(f)**

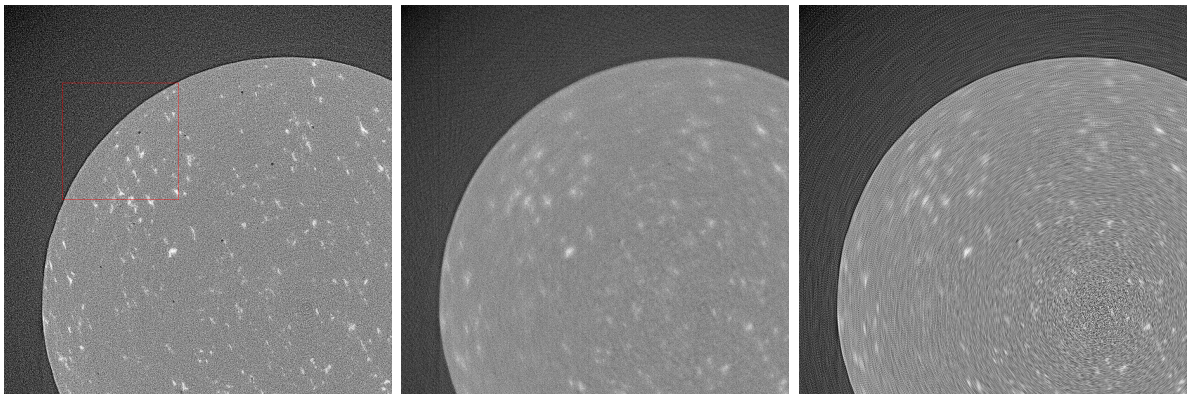
Figure 4.18: *The reconstruction of the* 1502^{th} *sinogram (area [501:1500, 501:1500] of [1:2560,1:2560], pixel-value [-1/2560:1/2560]) (**a**) From the original dataset with 3601 projections, (**b**) From the original dataset with 181 projections, (**c**) Using UDNN-128 for upscaling (181 projections), (**d**) Using the Bertram's Directional Interpolation (181 projections), (**e**) Using Cubic Interpolation (181 projections), (**f**) From the original dataset with 91 projections. Region of interest shown zoomed in Figure 4.19 below.*



(a) **(b)** **(c)**

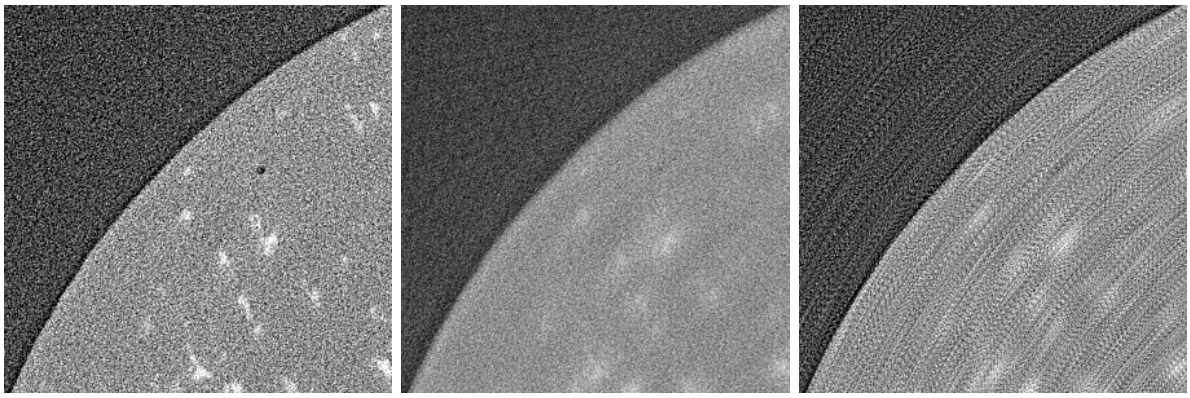
(d) **(e)** **(f)**

Figure 4.19: *The selected area marked by the red window in Figure 4.18(**a**) enlarged. (area [651:951, 701:1001] of [1:2560,1:2560], pixel-value [-1/2560:1/2560]). (**a**) From the original dataset with 3601 projections, (**b**) From the original dataset with 181 projections, (**c**) Using UDNN-128 for upscaling (181 projections), (**d**) Using the Bertram's Directional Interpolation (181 projections), (**e**) Using Cubic Interpolation (181 projections), (**f**) From the original dataset with 91 projections. Similar to Figure 4.11, in the proposed method (panel **c**) the level of noise is lower compared to the other methods, there are less rotating artifacts, but image contrast is slightly decreased.*



(a) **(b)** **(c)**

Figure 4.20: *The reconstruction of the* 1502^{th} *sinogram (area [501:1500, 501:1500] of [1:2560,1:2560], pixel-value [-1/2560:1/2560]) (**a**) From the original dataset with 361 projections, (**b**) Using UDNN-128 and an upscaling factor of 4x (361 projections), (**c**) Using Cubic Interpolation (361 projections). Region of interest shown zoomed in Figure 4.21 below.*



(a) **(b)** **(c)**

Figure 4.21: *The selected area marked by the red window in Figure 4.20(**a**) enlarged. (area [651:951, 701:1001] of [1:2560,1:2560], pixel-value [-1/2560:1/2560]). (**a**) From the original dataset with 361 projections, (**b**) Using UDNN-128 and an upscaling factor of 4x (361 projections).), (**c**) Using Cubic Interpolation (361 projections). Again, the proposed method (panel **b**) introduces less noise and rotation artifacts, however the contrast reduction is more noticeable.*

4x Sinogram Upscaling	361 Proj.	CI	UDNN-128 ($N_{out} = 3$)
Testing PSNR	22.28 dB	25.59 db	**26.89** dB
Testing SSIM [439]	0.3043	0.2163	**0.3925**

Table 4.8: PSNR score for reconstruction the upscaled sinograms $2, 12, \cdots, 217$ of the dataset with real-world data used for testing (Section 2.3, Figures 2B and 3B).). For the calculations, a reconstruction of the original sinograms (3601 projections) is used as ground truth. The 361 Noisy Proj. correspond to reconstructions made from sinograms that are only downscaled to 361 projection from the original. The other columns correspond to reconstructions made by these sinograms, where they are initially downscaled to a height of 91 (or 91 projections) and then upscaled to 361 (4x upscaling) with the available methods. Best result for each metric highlighted in bold.

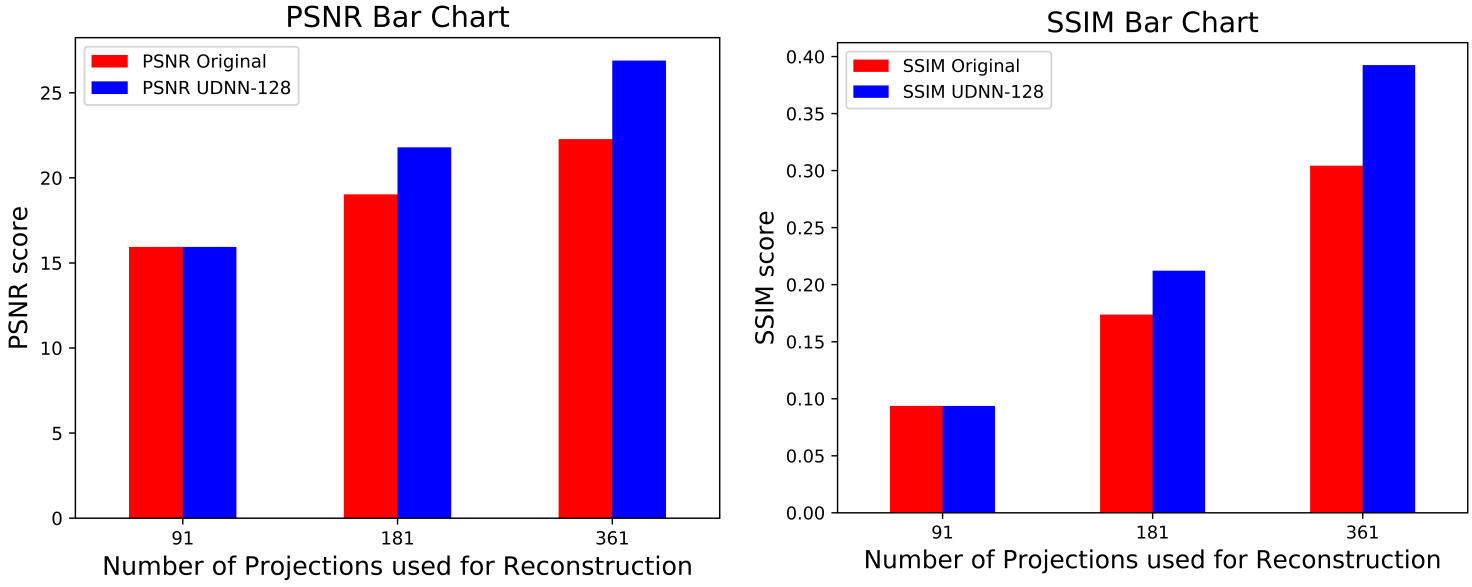


Figure 4.22: *Bar Charts showing the two metrics (PSNR and SSIM [439]), for the UDNN-128 network as the sinogram upscaling factor increases (and therefore also the number of projections).*

4.6 Discussion

Based on the experiment results in Section 4.5, we can deduce that the UDNN-128 version of our network is able to slightly but consistently outperform both the directional [22] and cubic interpolation [21] in both the PSNR and the SSIM [439] metrics (Tables 4.3-4.8). In addition, as can be seen in both synthetic and real-world data experiments (Figures 4.10, 4.11, 4.18, 4.19, 4.20, 4.21), the tomogram reconstructions after the use of our sinogram upscaling method contain less rotating artifacts, and have lower levels of noise. Additionally, by performing well not only for the case of the well defined *synthetic data*, but also for *real-world* tomograms, our approach confirms its ability to be stable even when the data have extra challenges as a result of real-life imaging. However, as it can be seen in Figures 4.20, 4.21 and Table 4.8, for increased levels of the upscaling factor (4x and more) due to the loss of information in the undersampled tomograms (by the extreme low sampling frequency), it is ever more difficult to predict and restore spatial details, especially for small internal components of the sample; though this is a limitation for all upscaling techniques. For this reason, we consider that upscaling factors larger than 4x will not be essentially helpful, despite probably UDNN-128 achieving higher PSNR

and SSIM than Bertram *et al.*'s directional interpolation [22] and cubic interpolation [21] due to the noise reduction (Figure 4.21). A potential solution for this issue, would be the introduction of weights associated with these smaller internal components in the loss function during training; but we will leave this to be explored by future research.

Apart from the improved restoration quality over other interpolation techniques, an argument can be made that UDNN also offers fast execution times though not to individual tomograms, but for the sinogram upscaling of time-series of tomograms as a whole. For the upscaling of one sinogram of size 91×2560 to a size of 181×2560, using UDNN-128, it requires approximately 1.7 seconds, when Bertram *et al.*'s directional method [22] requires approximately 5.5 seconds. Nonetheless, it would not be candid to claim that the above proves our approach's faster application, since the additional time required for training it is not taken into account. However, for the UDNN-128 model, training essentially is concluded after approximately 4 hours, when the validation error has eventually plateaued and the network's parameters do not change any further by backpropagation, since the loss function has converged to a local minima. It is important to note though, that our networks training is performed at worse once per time-series of tomograms, using a representative highly-sampled tomogram captured either before, after or both before and after the collection of the time-series. As presented in Table 4.9, if we consider the example of a time-series of 20 tomograms each composed by 2160 sinograms, it would take approximately 66 hours for our workstation to upscale all sinograms using Bertram *et al.*'s directional method [22], but roughly 25 hours with our method including the training session. This is a reduction of time required, by a factor of 62% and it could allow for faster upscaling and reconstruction of time-series, which is very beneficial for research centres like Diamond Light Source that store multiple terabytes of data that have to be processed.

As mention in Section 4.4.6, training and testing of our method is performed at different tomograms in order to avoid using sinograms from the same tomograms for both training and testing. This is because there is high correlation between sinograms of the same tomogram, and by use of sinograms from the same tomogram for both training and testing

	Inference time for one tomogram with 91 projections of size 2160 × 2560	Training time using a highly-sampled representative tomogram	Inference and Training times for a time-series of 20 tomograms each with 91 projections of size 2160 × 2560
UDNN-128	**36 minutes**	4 hours	**25 hours**
Bertram *et al.*'s Directional Interpolation [22]	3 hours 18 minutes	**0**	66 hours

Table 4.9: Computational time of our UDNN-128 against the state-of-the-art Bertram's directional interpolation for the task of sinogram upscaling. Bold text indicate the shortest computation times for each column respectively. As seen Bertram *et al.*'s Directional Interpolation does not require training prior to the inference. However due to its slower inference times it is faster to train and infer using UDNN-128 in order to upsample the projections of typical a time-series which is comprised by 20 undersampled tomograms.

may lower the validity of the obtained testing results. Nevertheless as mentioned in the previous paragraph, in some rare occasions highly-sampled representative tomograms are collected both at the beginning and the end of the collection of a time-series of tomograms. We argue that for future collections of time-series of tomograms, the collection of 2 highly-sampled tomograms both at the beginning and the end of a time-series collection should be preferred by biologists, material scientist and other researchers who implement studies with time-series of tomograms. This is because, it would be possible to use one highly-sampled tomogram for training and the other for testing. Therefore, the collection of 2 highly-sampled tomogram would be beneficial for the application of approaches like ours, which attempt to learning a mapping between the highly-sampled and undersampled sinograms. Additionally, if multiple time-series of tomograms are collected for similar physical samples along with their respective highly-sampled tomograms, it may allow the use of multiple highly-sampled tomograms for testing and thus allow a valid estimation of the standard deviation of our method's accuracy.

4.7 Application to a time-series of tomograms

Apart though from the experiments in Section 4.5, which provide quantitative results for the accuracy and effectiveness of our approach, our method has to be tested on ac-

tual undersampled tomograms that are part of time-series of tomograms. The reason, our algorithm has to be applied on undersampled tomograms is because our methodology explained in Section 4.4, of discarding intermediate projections from highly-sampled tomograms to synthesise an undersampled tomogram equivalent is not 100% physically the same, as running the capture procedure faster and collecting fewer projections. Obviously, for Section 4.5 since the undersampled tomograms used have a highly-sampled versions, it allows for the usage of the highly-sampled versions as a ground truth during training, validation and testing. Nonetheless, it is wise to at least acquire qualitative results by performing our approach on actual undersampled tomograms from a time-series; and it is only qualitative since for quantitative results we would require a highly-sampled tomograms to be used as ground truth which obviously does not exist.

In the Figures 4.23 and 4.24, we can see our approach is performed on the 1101^{th} and 1601^{th} sinogram using both a 2x and 4x upscaling factor, from a tomogram that is part of a time-series. Conveniently, this time-series' representative and highly-sampled tomogram is the tomogram used during training of the real-world experiments in Section 4.5. Because of this fact, there is not need to retrain our network since it it is already trained for tomograms that belong to this time-series. Based on the two Figures 4.23 and 4.24, and by comparing them with the Figures 4.18, 4.19, 4.20, 4.21, we can visually verify that our method does not behave obviously differently or unexpectedly, when it is applied for the upscaling of sinograms of tomograms that are collected with low number of projection, as opposed to tomograms that are downsampled artificially to resemble the first. This is more evidence of our approach's stability, and effectiveness under the use of real-world data.

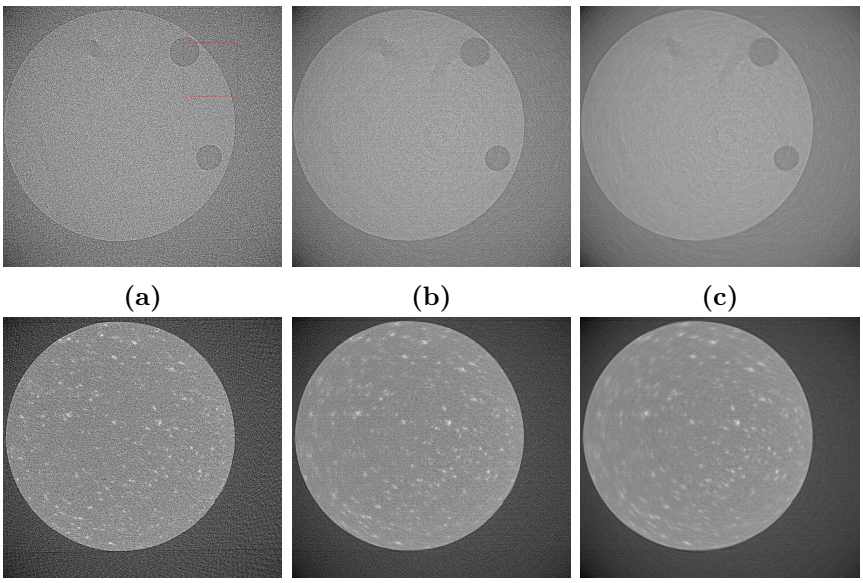


Figure 4.23: *Applying our upscaling method on a undersampled tomogram that is part of a time-series. (**a**), (**b**), (**c**) Reconstruction of the* 1101^{th} *sinogram (area [550:2000, 550:2000] of [1:2560,1:2560], pixel-value [-4.501E-4:1.097E-4]), (**d**), (**e**), (**f**) Reconstruction of the* 1601^{th} *sinogram (area [550:2000, 550:2000] of [1:2560,1:2560], pixel-value [-1/2560:1/2560]). (**a**), (**d**) From the original dataset with 91 projections, (**b**), (**e**) Using UDNN-128 for upscaling resulting to 181 projections, (**c**), (**f**) Using UDNN-128 for upscaling resulting to 361 projections. Region of interest presented zoomed in Figure 4.24 .*

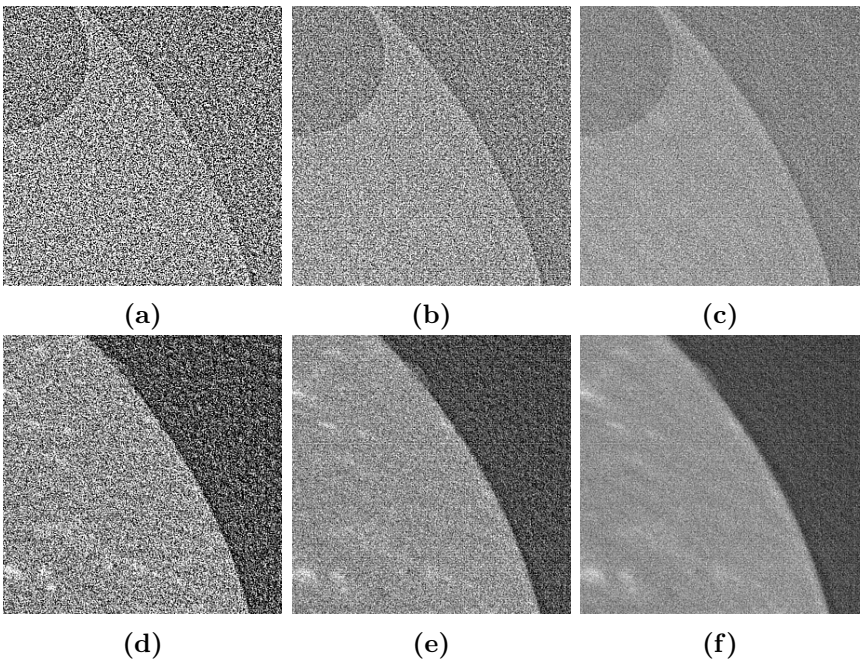


Figure 4.24: *The selected area marked by the red window in Figure 4.23 enlarged. (area [1501:1800, 701:1000] of [1:2560,1:2560]). (**a**), (**b**), (**c**) Reconstruction of the* 1101^{th} *sinogram (area [550:2000, 550:2000] of [1:2560,1:2560], pixel-value [-4.501E-4:1.097E-4]), (**d**), (**e**), (**f**) Reconstruction of the* 1601^{th} *sinogram (area [550:2000, 550:2000] of [1:2560,1:2560], pixel-value [-1/2560:1/2560]). (**a**), (**d**) From the original dataset with 91 projections, (**b**), (**e**) Using UDNN-128 for upscaling resulting to 181 projections, (**c**), (**f**) Using UDNN-128 for upscaling resulting to 361 projections. Similar to the earlier Figures 4.11, 4.19 and 4.21 the proposed method (panels (**b**), (**c**), (**e**) and (**f**)) produces more clear reconstructions with less noise.*

4.8 Conclusions

To conclude, in this chapter we have proposed a novel approach for sinogram upscaling of tomograms, specifically where highly-sampled representative scans are available, such as in time-series tomograms. Our method using a super-resolution approach based on deep learning is able to upscale the sinograms along their θ-*axis*, effectively providing predictions for the intermediate projections that are omitted to be collected, for instance during the capture of a time-series of tomograms. The latter is a common feature of time-series tomography, as it ensures a sufficient time-resolution through the time-series, needed for the capture of time sensitive events.

The proposed approach, UDNN [24] (code available at [25]), using a convolutional neural network architecture and a highly-sampled representative tomogram learns an end-to-end mapping between the sinograms with low and high number of projections. Based on the prior provided by the highly-sampled dataset, the network is able to train effectively and achieve comparable accuracy against state-the-art analytical interpolation techniques for sinogram upscaling. Qualitatively, it also leads to a quality improvement of the tomograms' reconstructions, and the appearance of fewer artifacts and noise, enabling the easier processing of the tomograms for further analysis.

Based on the experiments in Section 4.5, the UDNN-128 version of UDNN is able to achieve comparable interpolation accuracy against the state-of-the-art Bertram *et al.*'s directional interpolation [22] method, in both PSNR and SSIM [439] metrics. However, our approach is faster compared to the other analytical interpolation techniques (even including training times), for the use-case of sinogram upscaling of undersampled tomograms in a time-series of tomograms. This means that there is argument for the use of our approach, when also considering the fact that our method's accuracy is comparable to the one of other state-the-art analytical interpolation techniques.

For upscaling factor higher than 2x, due to observed reduced contrast of the internal components of the imaged sample in Figures 4.21 and 4.24, we consider that the application of our approach for upscaling factors larger than 4x, will not be beneficial since a significant

portion of the spatial details is expected to be lost.

Finally, we conjecture that additional accuracy or faster training/execution times can be further achieved by the introduction of weights associated with the smaller internal components with tomograms, that could be expressed as additional terms added to the network's loss function. Such additions could allow for the use of higher upscaling factors, resulting in the higher SNR and quality of the tomogram's reconstructions.

The software used for our method can be found on [25].

Chapter 5

Denoising and segmentation networks of low SNR reconstructions of undersampled tomograms

5.1 Densely connected encoder-decoder architectures for denoising and semantic segmentation

Chapter 3 explained the challenging nature of the undersampled tomograms (that are used in the collection of tomography time-series, amongst other places. We also demonstrated that it is possible improve the reconstruction quality of the undersampled tomograms (in Chapter 4), using representative highly-sampled tomograms. However, as it can be seen in Figures 4.23 and 4.24 despite the notable improvement in quality, it is still either extremely difficult or almost impossible for an annotator to semantically segment the datasets, or even potentially use software [306] to segment it faster and semi-automatically. Nonetheless as is described earlier in Section 3, it is common practice for the researchers who will execute a time-series tomography collection, to capture either at the beginning of the collection, the end of the collection or both, tomograms with a high number of projections. These tomograms captured outside the time critical portion of the collection, are able to provide additional spatial details for the sample. Additionally, they can be considered representative to the time-series' tomograms, since they depict all existing classes that are also present in the undersampled tomograms of the time-series. This is also what determines when they are captured (either before or after or both), so that all existing classes in the time-series are also present in the representative tomograms.

Inspired by Newell *et al.*'s [315] approach of "Stacked Hourglasses", in this chapter our proposed deep learning approach, named Stacked-DenseUSeg, presented at the British Machine Vision Conference (BMVC) 2019 Workshop on Visual Artificial Intelligence and Entrepreneurship (VAEI) [26] and also published with additional contributions (more on them on chapter 6), in the Machine Vision and Applications journal [441]. Stacked-DenseUSeg is an end-to-end approach designed for the task of denoising and segmentation of reconstructions of undersampled tomograms, also referred to as low-dose or sparse-angle(view) [12, 153] tomograms. This is achieved by stacking two consecutive networks, with the first one used for denoising the input and the second for its semantic segmentation. Stacking two networks creates an intermediate result that acts as an entry-point for the application of an intermediate loss function. This result is the denoised output of the combined network. Additionally, the stacking of the networks enables the denoised output to be utilised by the follow-up network to achieve a more accurate result regarding the semantic segmentation of the input. Furthermore, since the approach is an end-to-end solution it means that only one training session is necessary. This decreases the otherwise lengthy time training the two networks sequentially, and additional allows us to use the already trained denoising network to train the segmentation network for more accurate results.

As part of the same work [26, 441], we will also present our proposed a novel encoder-decoder network architecture 3D DenseUSeg, "DenseUSeg" in short, which expands the 3D DenseSeg architecture [10] by introducing a decoder portion comprised of consecutive transpose convolutional layers and skip connections similar to Ronneberger *et al.*'s U-Net [156]. Naturally, since our method deals with volumetric inputs, and not images, and also utilise densely connected convolutional layers [143], there is a massive increase in memory demands from the network's feature maps. This is due to the large number of feature maps constructed at the end of the encoder part, which is the product of the densely connected convolutional layers and also due to the size of the volumetric feature maps. To handle the large number of feature maps, while also employing a decoder part similar to Ronneberger *et al.*'s [156], appropriate measures have to be applied so that the

storage demands remain within reasonable boundaries. To achieve this, our approach, unlike previous proposed approaches [404], reduces the size of the feature maps in both the transpose convolutional layers and in the follow-up convolutional layers, that process the output feature maps of the earlier transpose layer concatenated with those brought through by skip connections.

The next Sections of this chapter are organised as follows. Section 5.2 will present the proposed network architectures, purpose and functionality. Section 5.3 will provide comprehensive information for the networks' structure and architectural design. Namely, Section 5.4 will offer details regarding the procedure employed for the training of our networks, as well as the way the predictions are inferred. Section 5.5 will present the experiments that took place regarding the denoising of undersampled tomograms, the segmentation under ideal conditions using highly-sampled tomograms and the more difficult segmentation of undersampled tomograms. Lastly, Section 5.6 will conclude the chapter.

5.2 Proposed networks' functionality and methodology

As introduced earlier, inspired both by Newell *et al.*'s [315] approach and the Bui *et al.*'s 3D DenseSeg architecture [10] we propose an end-to-end deep learning approach for the task of denoising *and* semantic segmentation. After training, our network is able to offer two different outputs. The first is a denoised version of the input and the second a semantic segmentation map of the classes present. Since our network is end-to-end, and the denoised output is provided into the following segmentation subnetwork, meaning that the accuracy of the resulting segmentation is improved.

In general our approach, Stacked-DenseUSeg, can be applied for denoising and semantic segmentation tasks in volumes, when there are representative high Signal-to-Noise Ratio (SNR) versions of the input volumes available, and sufficient available semantic annotations necessary for training. One of the cases where these conditions are met, is for time-resolved tomographic data collections. As is mentioned in Section 5.4, it is common either

before, after or both before and after of such time-series collections to obtain representative highly-sampled tomograms, which when they are reconstructed to a volume format wield high-quality denoised reconstructions; in other words with high SNR. Additionally, due to their high-quality compared to the ones of undersampled tomograms mid time-series, annotations produced using these reconstructions can be considered ground truth approximations with a high level of certainty, forming a "golden standard ground truth". As a result, it is very advantageous to use these tomograms and their corresponding annotations for the training of networks, especially for our proposed approach that predicts both a denoised, high in quality version of the input tomography reconstruction and a semantic segmentation map of the same input.

It is possible of course, to improve the reconstruction quality of the undersampled tomograms of time-series, either with the use of our proposed UDNN network presented in chapter 4 or iterative reconstruction algorithms such as the SART [411], SIRT [412] or CGLS [413], however due to the extremely low number of projections in these tomograms, there is a lot of information that the methods can not infer and restore. It can be deduced in Figures 4.23, 4.24 5.7(**c-d**) that these improved reconstructions could be used for visual inspection of the data and potentially allow the users to better discern areas that contain high levels of noise and are not very well imaged in the initial undersampled tomograms. Unfortunately though, it is not possible for them to be used for tasks that require significant high SNR such as a manual or semi-manual (with the use of special software [306]) segmentation that can be considered faithfully and sufficiently accurate. Naturally, without the creation of even a limited number of user-annotated data using these tomograms, it is not possible to make the leap to automatically segmenting the tomograms using machine learning techniques. Based on this fact our approach is dual purposed, for denoising and segmentation. Using the aforementioned highly-sampled tomograms, it is trained to segment the undersampled tomograms while it also offers a denoised, high-quality version of these tomograms for later use by approaches that require high SNR, similar to segmentation.

5.2.1 The Stacked-DenseUSeg's output and its functionality

Since the training of our network uses the earlier-mentioned highly-sampled tomograms, but its goal is it to offer both denoising and segmentation to *undersampled* tomogram reconstructions, the highly-sampled tomograms are artificially downsampled to synthesise an appropriate input for the network. As it was mentioned earlier, the improved reconstructions of undersampled tomograms either produced by our UDNN method or by iterative reconstruction algorithms [411] [412] [413], can be proven beneficial to discern certain features. which can be helpful for segmentation. Based on this, the reconstructions of the artificially downsampled tomograms are produced using one of the former approaches before they are used as inputs to the network. We could use our UDNN method to accomplish this, however in order to test our networks with input reconstructions constructed with standard methods, we used the CGLS [413] algorithm for the reconstruction quality improvement of the artificially downsampled tomograms.

Our network, "Stacked-DenseUSeg", is comprised of two consecutive encoder-decoder networks of our proposed 3D DenseUSeg architecture. The way these networks are connected is similar to Newell *et al.*'s [315], however different to their approach ours predicts a denoised, high-quality counterpart of the input instead of additional segmentation predictions which only aid for the application of an intermediate supervision. This means that contrary to Newell *et al.*'s approach ours offers a useful secondary output of the network. This output can be particularly helpful as it can potentially be used to manually correct inaccurate segmentation results in difficult areas. This is true for the case of our data (see Figure 5.7), since the reconstructions of undersampled tomograms are so low in fidelity that they can not even be used to infer manual or semi-manual corrections to segmentation predictions. Therefore, with the existence of a denoised, high-quality version of these reconstructions, researchers can correct potential mistakes in the segmentation results. Apart from that, they might also be used to benefit other image analysis approaches that are usually performed on reconstructions of highly-sampled tomograms, but here the low SNR in the case of undersampled tomograms (even with the help of UDNN, CGLS, or others) prohibits it.

5.2.2 DenseUSeg: Our proposed encoder-decoder network and its functionality

Our proposed encoder-decoder architecture employed by the two consecutive subnetworks within Stacked-DenseUSeg, is inspired by the recent Bui *et al.*'s 3D DenseSeg architecture [10] which contains densely connected convolutional layers, first proposed by Huang *et al.* [143]. Similar to He *et al.*'s [129] Residual Modules, Huang *et al.*'s Dense Blocks work to further combat the gradient "degradation" that appears in deeper architectures (More in Section 2.2.3.4). Nonetheless, in Bui *et al.*'s [10] volumetric solution, due to the high computational cost involved with networks that work with volumes, it cannot employ a sophisticated decoder part. To overcome this issue, in our decoder all convolutional layers including the transposed ones are decreasing the output feature maps by half. Therefore, our transpose convolutional layers have the added goal of condensing the information while they also upscale the feature maps. Contrary to other approaches [404], our network does not reduce the memory demands by summing feature maps in multiple places in the decoder. Instead similar to U-Net [156] it concatenates them, and the memory demands are kept low by the decoder's transposed convolutions producing less feature maps. This gives our network an advantage, since the optimal upscaled feature maps in the decoder part of the network are not necessarily the result of the summation between the feature maps produced by the transposed convolutions and the ones passed through skip connections. By producing less feature maps from the transposed convolutions, but concatenating them with the ones that pass from skipped connections it offers flexibility to the network to reach a better local minima for the loss function during training. This is because the subsequent convolutional layers have access to all input layers since they are concatenated and not summed.

5.3 The network architectures

In this section, we will present in detail the architecture of our DenseUSeg and Stacked-DenseUSeg networks as well as other relative architectures. In the next section we will explore the densely connected convolutional layers that are utilised in our architectures.

5.3.1 Densely connected convolutional layers

Similar to DenseSeg, our architecture DenseUSeg is composed of densely connected convolutional layers, as proposed by Huang *et al.* [143]. These layers form what is known as Dense Blocks. These blocks are comprised by stacking multiple Dense Modules, with a continuous concatenation approach for the produced feature maps. Specifically, the output feature maps of the l^{th} Dense Module within a Dense Block are:

$$x_l = H_l([x_0, x_1, \cdots, x_{l-1}]) \tag{5.1}$$

where $[x_0, x_1, \cdots, x_{l-1}]$ is the concatenation of the feature maps formed in the earlier $0, 1, \cdots, l-1$ layers and H_l the composite function of the l^{th} Dense Module. Based on the above and Figure 5.1 (**b**) each Dense Module receives as input the concatenation of the all the previous Dense Modules' feature maps. Through this process, the number of feature maps after l Dense Modules is $k_0 + l \cdot g$ where k_0 is the number of the initial feature maps of the input to the Dense Block and g the growth rate, is the number of feature maps produced by each Dense Module.

Each of the Dense Modules in our proposed architecture contains a bottleneck $1 \times 1 \times 1$ convolutional layer and a $3 \times 3 \times 3$ layer. Essentially the $1 \times 1 \times 1$ bottleneck convolutional layers and the concatenation of the Dense Blocks work similar to the summation and identity layers that appear to Residual Blocks (Figure 5.2 (**a**)). However, with the Dense architecture there is a continuous concatenation allowing further convolutional layers to access previous feature maps. This large increase of connections allows for better reuse of earlier feature maps, without the need of using following feature maps just to pass information that was previously calculated. Additionally, because of this and due to the removal of the identity layers ($1 \times 1 \times 1$ convolutional layers) which would exist if the architecture followed the residual approach, the number of parameters is also decreased, which is a very welcome feature especially for dealing with volumetric data, that have high memory demands. Furthermore, batch normalisation [134] and dropout layers [352] are introduced to combat the overfitting that can arise by the use of such a deep architectural building elements. The batch normalisation layers complement the Dropout layers as

they allow for lower dropout levels (less channels are dropped out), without overfitting appearing and thus allowing the network to reach higher accuracy. On the other hand, the dropout layers, by randomly deactivating some of the feature maps during training, prevent the network from learning statistical noise that is present in the training dataset and so reducing the risk of overfitting.

Moreover, in large architectures that utilise multiple sequences of Dense Blocks, usually for learning multi-scale contextual information, due to the continuous concatenation the number of feature maps can easily reach prohibitive sizes regarding the total memory required. To address this issue and to adjust at the same time the feature maps which subsequent learn smaller scale information, transitional layers are commonly employed. As presented, in Figure 5.2 (**b**), similar to Dense Modules they initially utilise a $1 \times 1 \times 1$ bottleneck convolutional layer, which decreases the number of feature maps and achieves the desirable reduction of feature maps. Following that, a $2 \times 2 \times 2$ with stride=2 convolutional layer decreases the scale by half, similar to a pooling operation. Naturally with different sizes of kernels and strides, different rates of scale reduction can be achieved; however, it is common in architectures that learn multi-scale information using a form of a pooling operation to decrease the scale by half in each step. As is demonstrated in [392], in large networks pooling operations can essentially be replaced with strided convolutions, which allows for simpler architectures where the search of the most advantageous form of pooling (average, max, etc.) can be skipped since convolutions are able to approximate it.

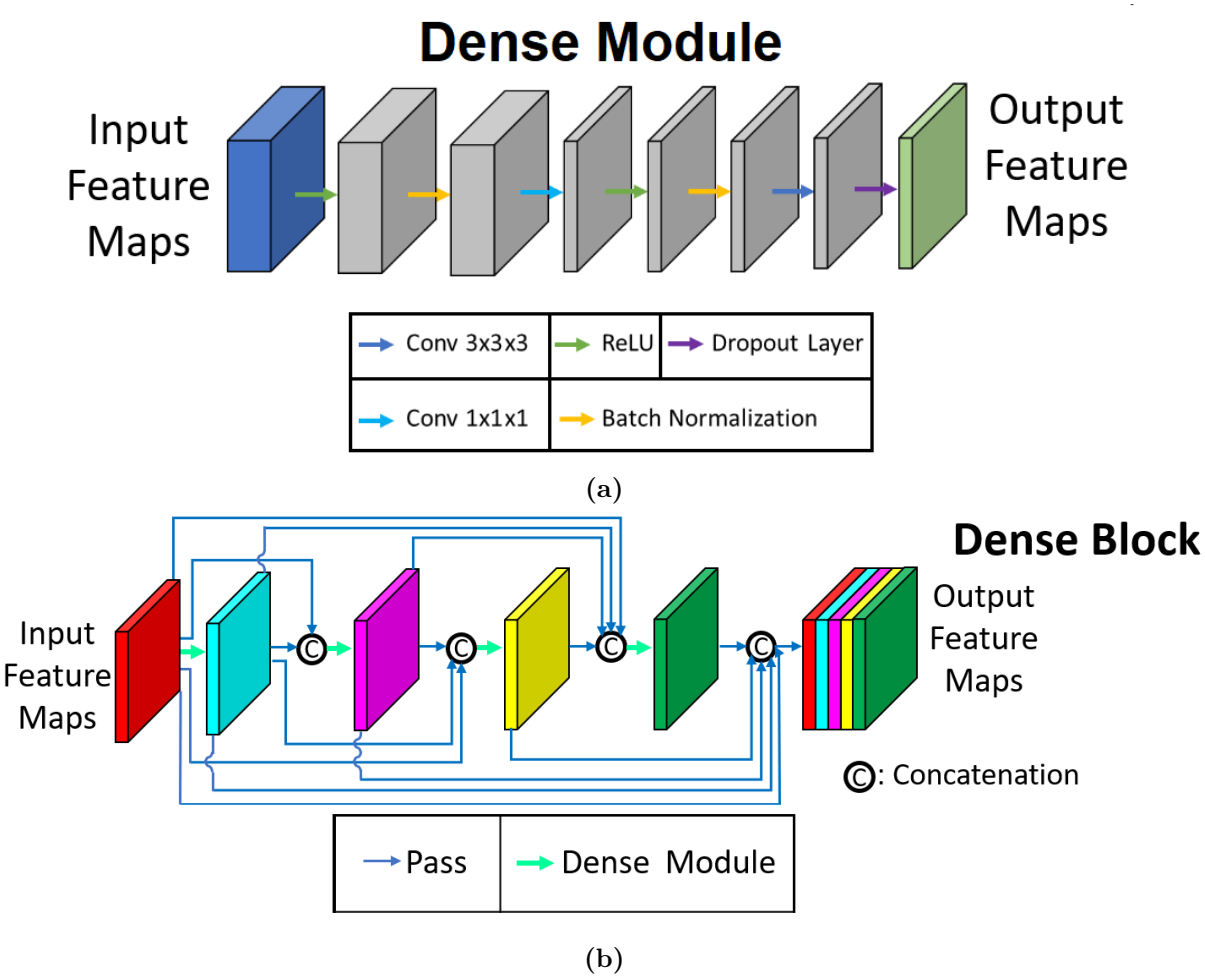
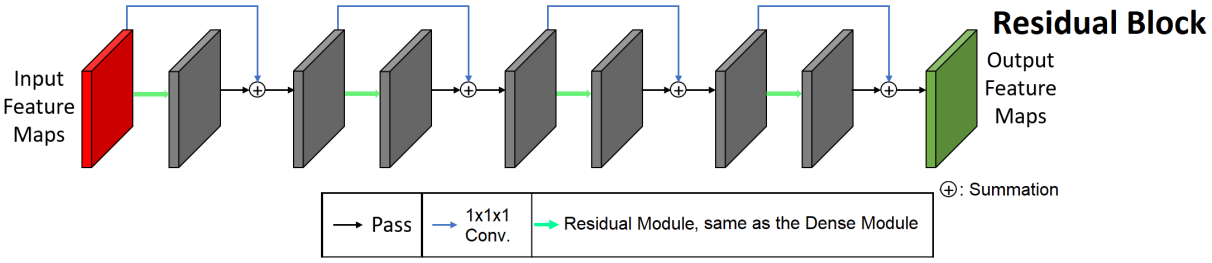
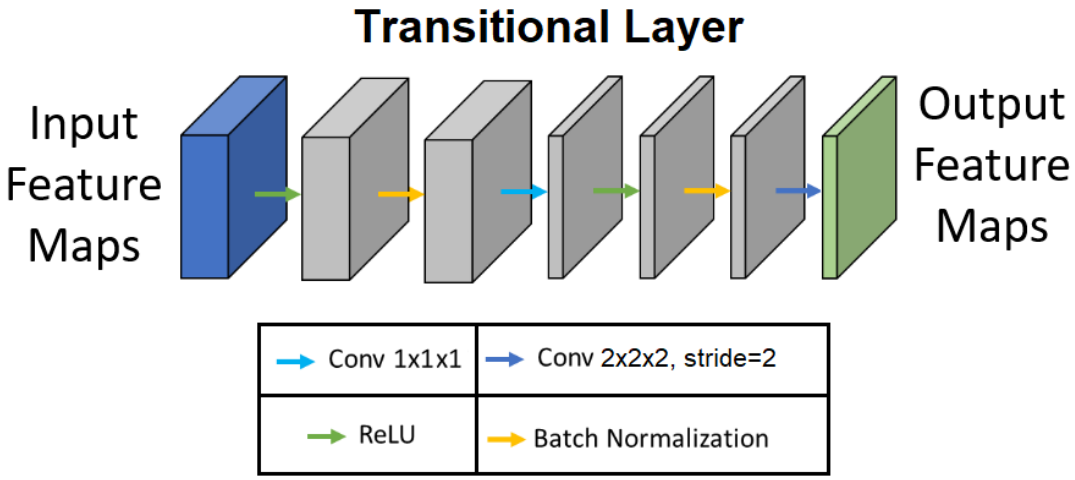


Figure 5.1: *A graphical representation of the Dense Module and the Dense Block. (**a**) The building block of Dense Blocks, the Dense Module. Each is comprised of one* $1\times1\times1$ *convolutional layer which acts as bottleneck and a* $3 \times 3 \times 3$*. ReLU and Batch Normalisation layers are placed between the convolutions for non-linearity and robustness purposes respectively. Finally a Dropout layer is added at the end as it mitigates the risk of overfitting. (**b**) A Dense Block comprises of 4 Dense Modules. Within this sequence of Dense Modules each input to the Dense Modules is the concatenation of all the feature maps that originate from the previous Dense Modules in the sequence. The Dense Block's output is the concatenation of all the feature maps produced from the individual Dense Modules.*



(a)



(b)

Figure 5.2: *(**a**) A Residual Block equivalent of the Dense Block of figure 5.1 (**b**). Naturally, the number of connections is smaller than in densely connected convolutional layers. (**b**) A Transitional layer used after the a Dense Block. Using a bottleneck* $1 \times 1 \times 1$ *convolutional layer the number of feature maps are decreased and with a subsequent use of a* $2 \times 2 \times 2$ *convolution with stride 2, the scale of the feature maps is reduced in preparation for the network to learn features from smaller input scales.*

5.3.2 MSD: Mixed-scale dense network

One of the early architectures that was designed and implemented for the goal of denoising and semantically segmenting undersampled tomograms or sparse angle tomograms was the Mixed-Scale Dense (MSD) architecture proposed by Pelt and Sethian [337]. The architecture consists of multiple consecutive convolutional layers that are densely connected as described in Section 5.3.1. For their proposed architecture, instead of the more complex Dense Module, 3×3 dilated convolutional layers are used with ReLU layers for non-linearity [433]. Regarding the connectivity, the inputs in each of the layers are the feature maps created from the previous layers plus the original input. One key characteristic of their proposed architecture is that the final network is customizable by the user based on two parameters. The first one is the depth of the network, which defines the number of consecutive convolutional layers, where each receives as input all the feature maps of the previous layers, thus forming a "chain" of convolutional layers. The second parameter is the width of the network, which describes how many of those "chains" exist in parallel. Feature maps of a "chain" are also used by other "chains" in a way that is clearly shown in Figure 5.3 (**a**), which offers a schematic presentation of an MSD network with depth of 3 and width of 2. As can be seen, a convolutional layer receives as input the concatenation of the feature maps of all previous layers and the initial input, irregardless of which "chain" they belong (being either from the one the convolutional layer belongs to or any the others).

Additionally, it is important to mention that across all "chains" each convolutional layer has a different value of dilation [322]. The reason behind this is to extract features across different scales, hence the mixed-scale part in the architecture's name. This is an essential part of the network, since it enables it to accurately segment classes with features of different sizes. This property is also prevalent in network architectures that do not use dilation [8, 9, 156, 268, 308, 391] where features from different scales are attained by $2 \times 2 \times 2$ convolutional layers with a stride of 2, or pooling layers. However, since these operations progressively reduce the resolution of the feature maps; upscaling operations have to be used in the final part of the network to increase the resolution of the

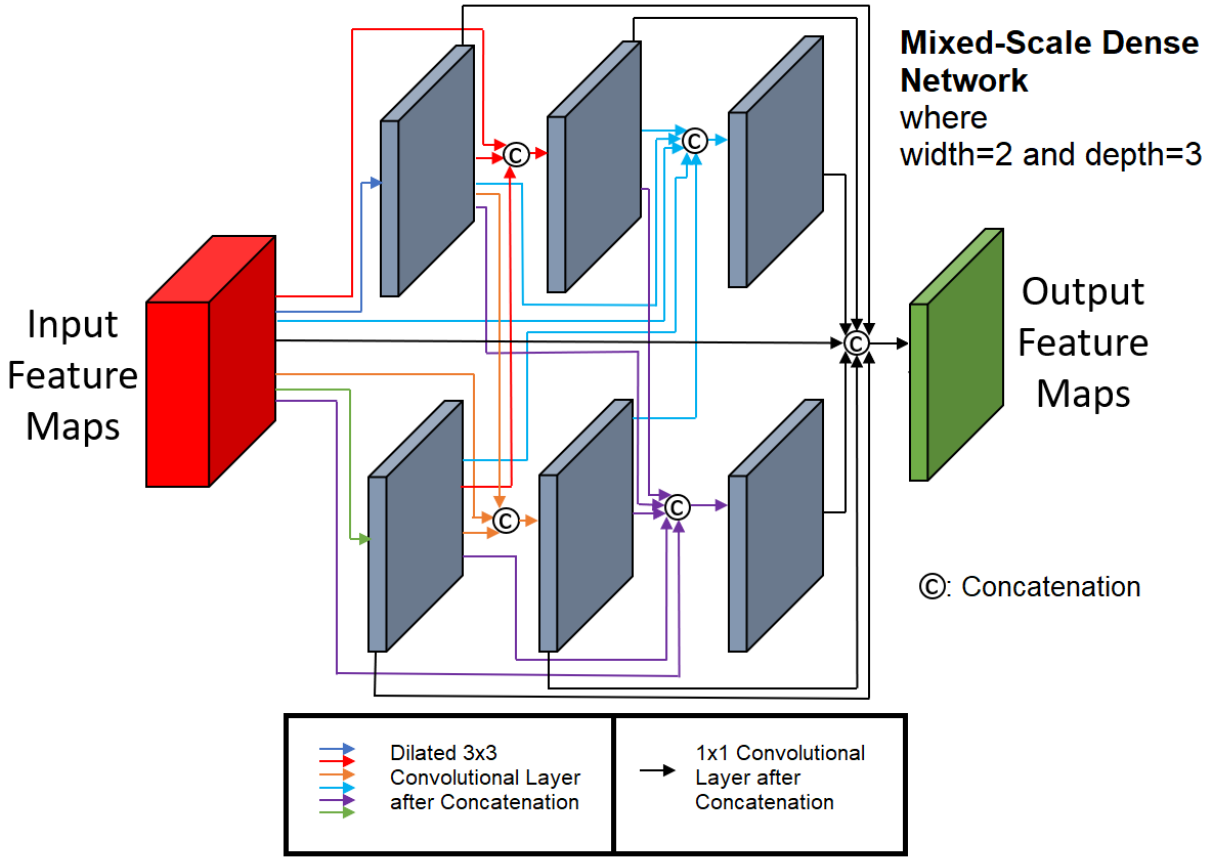
feature maps in order for the network to produce an output of the same resolution as the input. These upscaling operations that are either an interpolation method (e.g. bicubic interpolation) or transposed convolutional layers can potentially increase the training and inference time significantly. The use of dilated convolutions has been explored in the past and most prominently by Chen *et al.* [330], who showed that they can increase the accuracy of the final segmentation. Furthermore, alongside the dilated convolutions the researchers also use reflective padding to counteract the introduction of zero pixels by the use of a traditional zero-padding. The reason padding is used is because in order to maintain the resolution of the produced feature maps equal to the one of the input image/volume, appropriate padding has to be applied to each of the feature maps before the operation of convolution is performed. Nonetheless, the use of a traditional zero-padding would introduce a lot of error for the prediction of the pixels around the edges and combined with the use of dilated convolutions, that may use padded pixels even for the prediction of the pixels that are far from the edges, this can potentially adversely affect the overall attained segmentation. For the growth rate of each of the convolutional layers, the researchers choose it to be just one feature map; therefore each convolutional layer produces one new additional channel. Based on that, for the case that the width is higher than one, the multiple "chains" can be viewed as one "chain" but each of the convolutional layer produces a number of new feature maps/ channels equal to the width. However, each of the produced channels in each of the layers have different dilation settings. Since this version of convolutional layers is uncommon to most popular deep learning frameworks (Pytorch [442], Caffe [443], Tensorflow [444]) researchers had to develop one of their own using PyCUDA [445].

Despite however the theoretical introduction of a more flexible architectural format, the research during experiments resulted in a network architecture in which w=1 and various sizes of d were used (d=25, 50, 100, ..., etc.). Figure 5.3 (**b**) displays how much the architecture is simplified by such choices. The selection of the dilation value in each layer was made according to

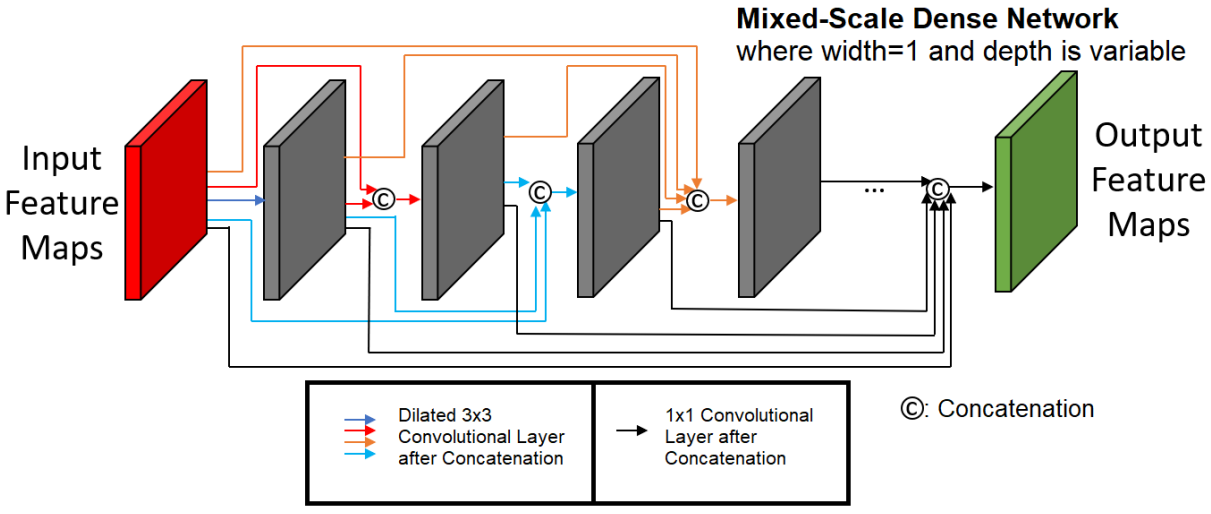
$$d_i = ((i) mod 10) + 1 \tag{5.2}$$

where i is the index of convolutional layer starting from 0. The choice of width=1 apart from removing a lot of complexity that would affect the necessary training times and computing resources, is also a convenient option from another point of view: it allows for the operation of the network without the need of a convolutional layer that applies dilation at different rates for each of the layer's output channels. Subsequently, this version of MSD network can be coded using traditional deep learning frameworks; for our case we used PyTorch [442]. Finally, in order to apply the network to 3D data, the researchers do not use 3D convolutions, but instead a multi-channel 2D input where each channel refers to a different level of depth in the $z\text{-}axis$. The final prediction is a 2D output, and it is the segmentation or the denoising of the middle channel (for example if there are 5 channels, the prediction is regarding the voxels of the 3^{rd} channel). By employing a sliding window scheme and concatenating the 2D predictions, the final 3D prediction is formed.

One of the main reasons for our investigation of this particular network architecture is because during the experiments, the researchers are able to demonstrate its good performance not only regarding the semantic segmentation of tomograms, but also the denoising of undersampled tomograms. Nevertheless, since it uses the dilated convolutions to collect multi-scale information instead of gradually reducing the scale of the feature maps, the network's memory requirement may be very high for highly deep MSD nets.



(a)



(b)

Figure 5.3: *(**a**) A mixed-scale dense network (MSD) layout where the network's depth=3 and its width=2 (therefore 2 "chains"). As can be seen, each convolutional layer receives as input the concatenation of the feature maps from all preceding convolutional layers irregardless to "chain" they belong. (**b**) A Mixed-Scale Dense Network (MSD) layout where the width is just 1 (1 "chain") and the depth can vary (d=25, 50, 100, . . . , etc.). According to Pelt and Sethian, this version of the MSD has the least complexity while it achieves noteworthy performance.*

5.3.3 3D DenseSeg

Apart from the Mixed-Scale Dense network, a contemporary architecture that we also considered for the denoising and segmentation of undersampled tomograms was Bui *et al.*'s [10] 3D DenseSeg, or "DenseSeg" for short. As is presented in Figure 5.5 (**a**), the architecture is comprised of 3, $3 \times 3 \times 3$ convolutional layers that produce $k_0 = 32$ feature maps followed by a $2 \times 2 \times 2$ convolution with stride 2 to reduce their scale. Following that, the architecture continues with 4 Dense Blocks each with 4 Dense Modules (see Figure 5.1 (**a**-**b**)) separated by 3 Transitional layers (see Figure 5.1 (**b**)). The growth rate for each of the Dense Blocks is set to $g = 16$ and the rate at which the Transitional layers decrease the feature maps channel's number is $\theta = 0.5$, while they also decrease the scale of the feature maps with a $2 \times 2 \times 2$ convolution with stride 2. Batch normalisation is employed between the layers to improve robustness [134], as well as ReLU modules for non-linearity [433]. Furthermore, a Dropout layer [352] with rate $p = 0.2$ is used after each Dense Module, to help with over-fitting problems. The final feature maps from the Dense Blocks are then appropriately upscaled to the same pixel resolution as the input using transposed convolutions. The upscaled feature maps and the one that is produced after the initial 3 convolutions are then concatenated and processed by a $1 \times 1 \times 1$ convolution and a softmax layer [266, 267] to produce the probabilities each voxel belongs to each class. By assigning to each voxel the class that has the highest probability to belong to, the final segmentation map is attained. For the use-case of denoising instead of segmentation, the softmax layer is skipped and the $1 \times 1 \times 1$ convolution offers the final output.

The DenseSeg, embodying a classic encoder-decoder architecture, utilises multiple Dense Blocks to attain rich information for different scales of the input. Additionally, the reduction of the scales of feature maps using Transitional Blocks allows the deeper layers to use higher numbers of feature maps without increasing the computational complexity. Nevertheless, this potentially may come with an accuracy cost, since the feature maps from deeper layers have to be upscaled using transposed convolutions in order to contribute to the final predictions and during this process error may be introduced.

5.3.4 MSD and DenseSeg comparison

During the initial phases of designing our final architecture for denoising and segmentation of undersampled tomograms, the two aforementioned architectures had to be tested in order to compare which approach is more advantageous. On the one hand, the MSD network does not progressively reduce the scale of the feature maps and attains multi-scale information using dilated convolutions. This is potentially ideal for the cases where the final segmentation or denoising predictions have the same pixel resolution as the input to the network, since information is not lost due to downsampling and potential error is not introduced during the upsampling. On the other hand, DenseSeg, using a classic encoder-decoder layout, is able to operate using a higher number of parameters, when its computational footprint is the same as MSD's. As the network's feature maps are processed by deeper layers, the network's transitional blocks limit both the scale and the number of feature maps. This naturally may result in partial information loss, however, by using the final feature maps of all the Dense Blocks for the prediction of the final output, the network has an opportunity regain any information that was lost.

In order to compare the networks, two variations of them were evaluated against each other. In our case, MSD_100 is used, which is an MSD architecture with 100 layers ($d = 100$) and $w = 1$ (See Section 5.3.2) proposed by Pelt and Sethian. Moreover, its reflective padding is replaced with a zero-padding, as the PyTorch implementation of reflective padding increases the GPU memory requirements to an unreasonable amount. Moreover, the modified 3D DenseSeg named 3D DenseSeg_64 uses a growth rate of $g = 64$, as the author's recommended growth rate of $g = 16$ was far to small to offer noteworthy results and it had scores even lower than 3D U-Net [8]. Additionally, it was observed that this modification does not increase the network's computational demands to unreasonable levels. The two networks were compared for the task of semantically segmenting undersampled tomograms (specifically the one described in Section 5.4). The modified MSD and 3D DenseSeg architectures are also compared along with Çiçek *et al.*'s [8] 3D U-Net, which is used as a baseline.

The reason we conducted this experiment was in order to determine which architecture

is the most promising for the task of segmentation of undersampled tomograms, valuing not only the performance but also the speed during training/inference. Subsequently this experiment determined which architecture will be the base for our proposed approach of segmenting undersampled tomograms. For the experiment, the 3D U-Net is selected as the baseline because it is a noteworthy method that has influenced multiple state-of-the-art approaches [9, 10, 391]. Based on Figure 5.4 and Table 5.1, it can be concluded that the 3D DenseSeg_64 architecture is a better fit for the task of segmentation of undersampled tomograms, which have low SNR and poor visual quality as can be seen in Figure 5.7. This is due to its structure, as it collects multi-scale information through the gradual reduction of the feature maps' resolution, which indirectly reduces the ill effects attributed to the presence of noise in the input. Furthermore, as can be seen in Table 5.1 the 3D DenseSeg_64 architecture offers faster training times as the large number of feature maps in the deeper layers is compensated by the low voxel resolution of the feature maps in those layers.

The MSD_100, on the other hand, sadly does not perform as well as it was expected, potentially because of the absence of reflective padding. The exemption of this feature, it seems, significantly limits the performance of the network most likely because of the zero elements in the zero-padding, which is more prevalent in convolutional layers that have a high value of dilation in their kernels. In particular, in convolutional layers with dilation rate of 1, only the border imaging elements (pixels, voxels, etc.) are affected by the zero elements creating border effects (ill-predicted features due to the lack of pixel information) that are eventually negated during inference using overlapping input patches. However, performing zero padding in convolutional layers with high dilation rates (2, 3, etc.) affects more border imaging elements leading in lower accuracy. Additionally, since the feature maps do not gradually reduce in scale in deeper layers, the network requires additional training time to process not only multiple, but also high-resolution (high in pixel count) feature maps.

In conclusion, based on the above, the 3D DenseSeg_64 architecture seems the most promising architecture for tackling the task of segmentation of undersampled tomograms,

	3D U-Net	3D DenseSeg_64	MSD_100
Testing mean IoU	0.64	**0.65**	0.58
Training time for 100 epochs (HH:MM:SS)	5:10:47	**4:10:10**	11:32:27

Table 5.1: Testing mean IoU and total training time for 3D U-Net, 3D DenseSeg_64 and MSD_100. As mentioned in Section 5.4.2 10% of the annotated data is used for validation at each epoch, while 70% and 20% of the annotated data is reserved for training and testing respectively. Each input instance is a $64 \times 64 \times 64$ volume and the earlier percentages equates to 3723 input instances reserved for training, 532 input instances reserved for validation and 1065 input instances reserved for testing. As it can be observed, MSD_100 accuracy is disappointing and 3D DenseSeg_64 is only marginally better than the older 3D U-Net. Additionally, MSD_100 requires an unreasonable amount of time to train, due to not reducing progressively the scale of the feature maps in deeper layers.

both in terms of accuracy and training time. However, as it can be seen in Table 5.1, 3D DenseSeg_64 accuracy is marginally larger than the one of 3D U-Net. Based on this fact, and some preliminary testing it was discovered that 3D U-Net's approach in upscaling the feature maps is crucial for achieving an elevated accuracy. Based on this fact, in the following section we will describe our proposed *DenseUSeg* that expands upon the DenseSeg architecture proposed by Bui *et al.* [10] by adding an expansive path or a decoder similar to U-Net [156]. While our findings provide answers regarding the segmentation performance of MSD_100 and 3D DenseSeg_64, their denoising performance is an open question. However, due to their distinctly different use of dimensionality (MSD uses 2D data and DenseSeg uses 3D), their connection to a hybrid network for denoising and segmentation would be very difficult, complicated and inefficient regarding the time required to predict an output. Therefore, our desired goal of achieving an end-to-end network for both denoising and segmentation has to be obtained by stacking the same network architecture (either the MSD, DenseSeg or our proposed DenseUSeg) twice. This leads us to discard of the MSD as a potential candidate not only for the task of segmentation, but also the task of denoising.

5.3.5 DenseUSeg

Our proposed architecture DenseUSeg, following a traditional encoder-decoder architecture uses a number of initial layers in order to extract information from multiple scales and

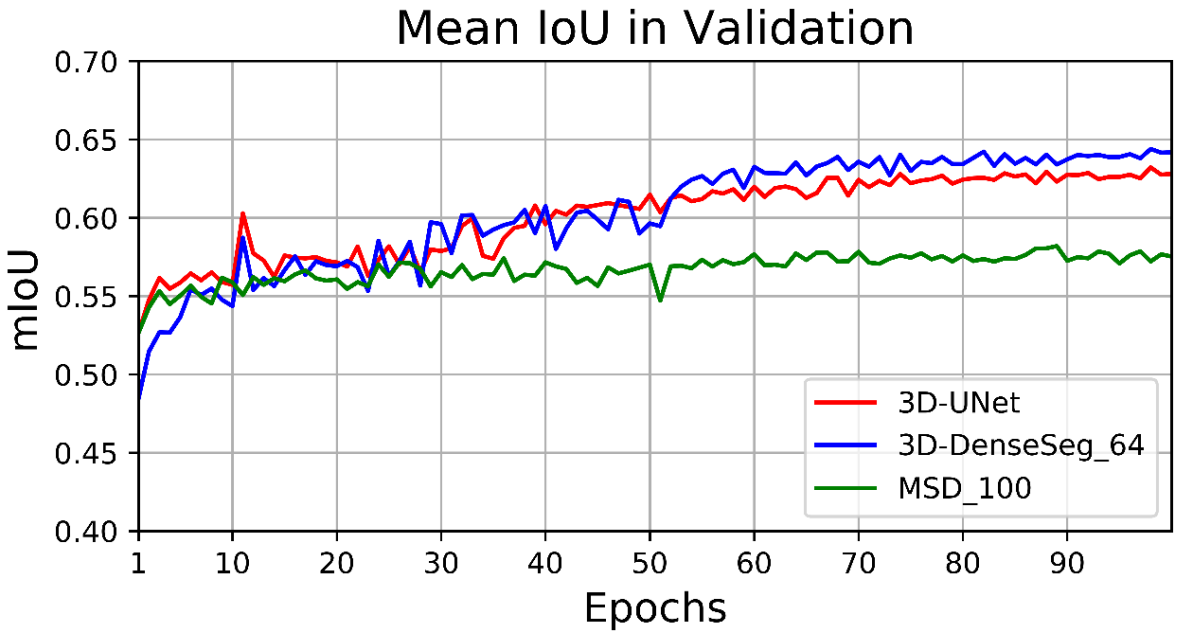


Figure 5.4: *Validation mean IoU for 3D U-Net, 3D DenseSeg_64 and MSD_100. As mentioned in Section 5.4.2 10% of the annotated data is used for validation at each epoch, while 70% and 20% of the annotated data is reserved for training and testing respectively. Each input instance is a* $64 \times 64 \times 64$ *volume and the earlier percentages equate to 3723 input instances reserved for training, 532 input instances reserved for validation and 1065 input instances reserved for testing. As seen in the graph, the MSD's accuracy is far smaller even compared to older 3D U-Net. On the other hand, the 3D DenseSeg_64's accuracy is marginally better than 3D U-Net's, even after the increase of the growth rate to* $g = 64$*. This is an indication for the beneficial effects of utilising a decoder in the architecture regarding the task of semantic segmentation.*

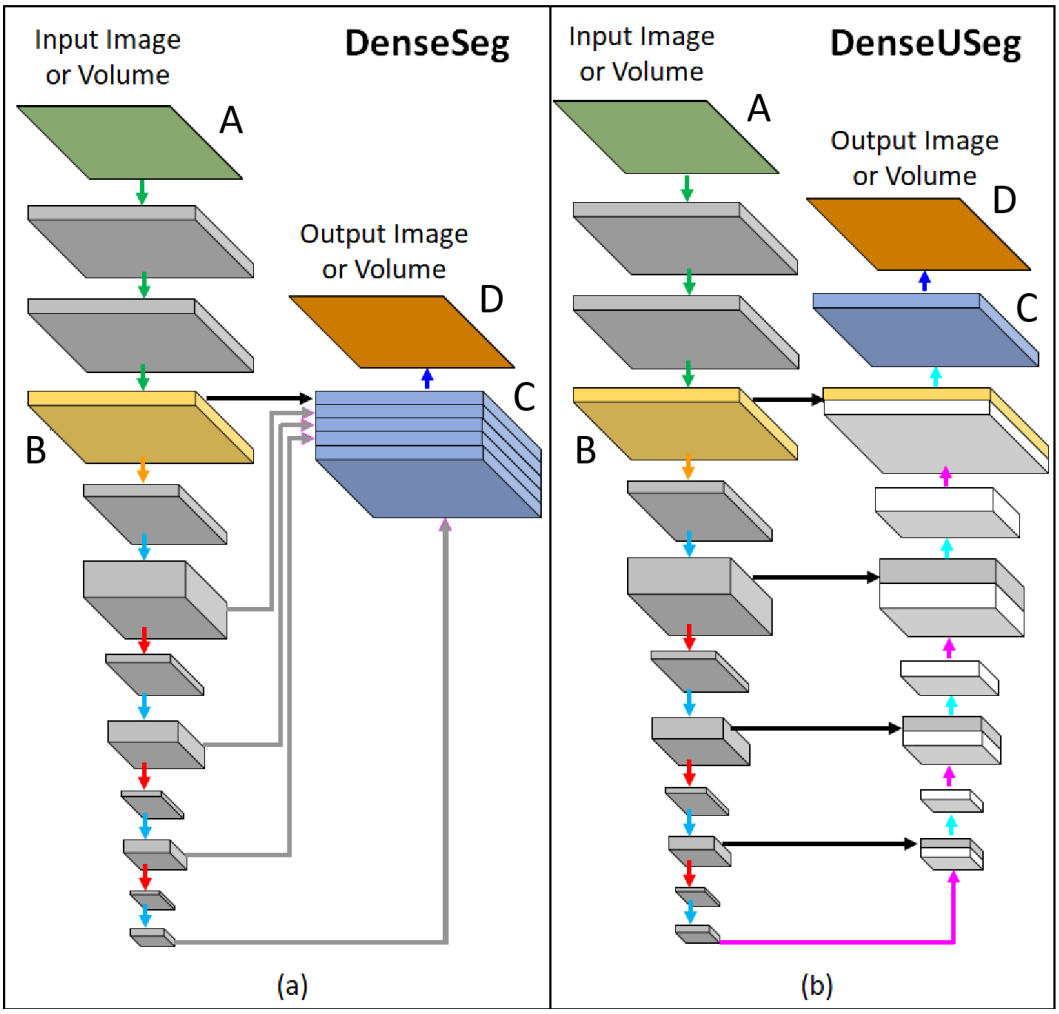


Figure 5.5: *Graphical representation of Bui et al.'s [10] DenseSeg and of our proposed DenseUSeg architecture. Our proposed network introduced an "expansive" path inspired from U-Net [156].*

then using upscaling operations it processes this information to obtain the final output. Namely, similar to the Bui *et al.* [10] approach there are 3 initial $3 \times 3 \times 3$ convolutional layers that produce $k_0 = 32$ feature maps. These feature maps compose the entry feature maps, that contain essential information about significant features that have the same scale as the input. In Figure 5.5(b) they are marked as **B** and the black arrow next to them signifies that they pass through to the decoder part without any upscaling via a transposed convolution. After these first feature maps are produced, a $2 \times 2 \times 2$ convolution with stride 2 reduces their scale, and a sequence of dense blocks extract valuable information about various scales of the input. In total, similar to DenseSeg the encoder part of the architecture consists of 4 Dense Blocks, each consisted by 4 Dense Modules and separated by 3 Transitional layers. Each of the Dense Modules contains a bottleneck $1 \times 1 \times 1$ convolutional layer and a $3 \times 3 \times 3$ convolutional layer (see Figure 5.1 (**a**)). To mitigate the extensive increase of feature maps as the network becomes deeper because of Dense Blocks, Transitional Blocks are placed between each of the Dense Blocks. These include a $1 \times 1 \times 1$ convolution that decreases the number of feature maps by a factor of $\theta = 0.5$, continued by a $2 \times 2 \times 2$ convolution with stride 2 for scale reduction. This way the network condenses the extracted information from each scale of the input, before continuing with the extraction of information for smaller scales. It is also important to mention, that before the application of each convolutional layer either in the Dense or Transitional Blocks, batch normalisation and the Rectified Linear Unit (ReLU) are employed to improve robustness [134] and introduce the necessary non-linearity [433] respectively. Similarly, these layers are also present in the initial three convolutional layers, but they are placed after the convolutions. Finally, a Dropout layer [352] with rate $p = 0.2$ is placed at the end of each Dense Module, since it has been proven to help with overfitting problems.

Moving forward to the architecture's expansive path, DenseUSeg uses a scheme similar to U-Net [156]. The network upscales the feature maps of the last Dense Block using a $4 \times 4 \times 4$ transpose convolutional layer with stride 2 and padding 1 and then concatenates them with the feature maps of the earlier Dense Block that have the same resolution

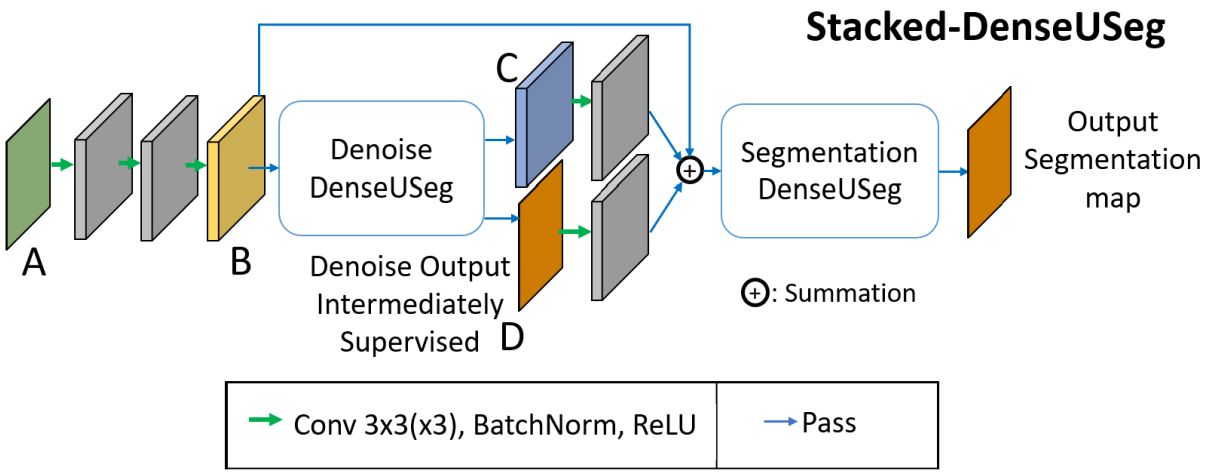


Figure 5.6: *Graphical representation of our proposed Stacked-DenseUSeg, that is composed by two DenseUSeg. The first DenseUSeg is oriented towards denoising the input and the second towards semantically segmenting it.*

(same number of voxels for the height, width and depth). The feature maps then pass through two consecutive convolutional layers, and then the upscaling is repeated. Since the number of feature maps is very extensive after the Dense Blocks, both the transpose convolutions and the first of the two consecutive convolutions in the expansive path reduce the number of convolutions by half. When the feature maps reach the same resolution as the input volume, they pass through a $1 \times 1 \times 1$ convolution and a Softmax Layer which produces the probabilities for each of the voxels belonging in each of the classes. By assigning each voxel to the class which has the highest probability, the final segmentation map is formed. Naturally, when the network is used for the task of denoising the Softmax layer is redundant and the pixel values of the final feature map consist of the prediction for the denoised output.

5.3.6 Stacked-DenseUSeg

Having introduced our proposed DenseUSeg network, we continue with our proposed end-to-end denoising and segmentation approach. As Figure 5.6 shows, when combining two DenseUSeg nets, in our Stacked-DenseUSeg architecture the initial 3, $3 \times 3 \times 3$ convolutional layers (the ones between the feature maps **A** and **B**) are common. The denoising part of our architecture then produces the final feature maps of the denoised output (feature maps **C**), followed by the denoised prediction for the input (layer **D**).

These then pass through $3 \times 3 \times 3$ convolutions (with batch normalisation and ReLU layers) and the resulting feature maps are summed with feature maps (**B**) to provide the input for the segmentation DenseUSeg. Since the input to the second DenseUSeg, tasked with segmenting the input, receives information from the 3 convolutional layers (feature maps **C**) of the denoising DenseUSeg, the 3 convolutional layers are not repeated in the segmentation DenseUSeg. Furthermore, as will be also shown in section 5.5, the existence of a Dropout layer in DenseUSeg and DenseSeg [10] improves the performance only in the case where the input is noisy and has no effect when the input is noiseless. Therefore, while for the denoising DenseUSeg a Dropout layer with rate $p = 0.2$ is selected for the respective segmentation network, the rate is set to $p = 0$ as it already receives a denoised input.

The connection between the networks is inspired by Newell *et al.* [315], where the authors stack multiple hourglass networks with intermediate supervision in order to improve the network's final predictions. However, the goal of our architecture is to produce a denoised output which assists in the accuracy of the final segmentation output. The stacked approach is essential, as it allows for the creation of an intermediate output of the same size (number of voxel dimensions) as the input. This enables the application of an intermediate loss function, to the network in order to obtain the desired denoised output. This is possible in the case that the available annotated data used during training have both a high SNR version that can be used in the intermediate supervision, and a segmented map for the final supervision. This is generally true for undersampled tomograms, which are part of time-resolved studies as representative highly-sampled tomograms usually collected at the end or the begin of a capture of multiple undersampled tomograms (a 4D dataset). By semantically annotating these highly-sampled tomograms and producing two reconstructions of them, one having high SNR and the other low, it is possible to use the high SNR reconstruction and the annotations as "ground truth" during the training while the low SNR reconstruction is the input. Then during inference of *undersampled* tomograms, where both the segmentation map and a high-quality (high SNR) version of the tomograms are desired, our network is able to offer two distinctive outputs using the

low-quality reconstructions.

The order in which the two networks are connected is significant. By placing the denoising network first it is possible to create a better quality version of the input, which can then be utilised by the second segmentation network in order to offer more precise predictions. The way which the networks are connected is also very important, since it is not just a simple passing of the denoising predictions. There are three distinctive elements that provided from the previous network and to the next. The first is the initial feature maps produced by the first network, which are created by 3, $3 \times 3 \times 3$ consecutive convolutional Networks marked in Figure 5.6 as **B**. These contain information essential for the task of segmentation, that may not be featured in the final predictions or the final feature maps of the first network, because they may not contribute towards the denoising effort. The other two elements are the final predictions of the denoising network and the its last feature maps. These contain refined information by the first network, which may enhance the accuracy of the segmentation network. These last two feature maps also pass through $3 \times 3 \times 3$ convolutional layers and then they are summed with the feature maps marked as **B**. In the end, this link between the networks allows for crucial information necessary for the segmentation of the input to be introduced into the second network, and allow it to offer better predictions. While, the operation of summation, may not be the most ideal compared to the most favorable operation of concatenation [143], a bottleneck functionality is needed in order to reduce the number of feature maps before they are used as input in the next network. This is because, the densely connected layers in the next network will increase the number of feature maps proportionally to the number of input feature maps. Without a bottleneck functionally, the number of feature maps will highly increase within the next network, making the overall memory demands of Stacked-DenseUSeg unmanageably high. The use of the operation of summation is a parameter-wise inexpensive way to introduce a feature map bottleneck at this point. Nevertheless as it was latter considered, a more optimised way to perform this bottleneck, would be to concatenate and then use a $1 \times 1 \times 1$ convolution to reduce the number of feature maps, which we may attempt to address this in a future research.

Furthermore, our Stacked-DenseUSeg architecture is able to offer two different outputs within one forward pass. This is translated into a potential reduction of training time of two networks, where the second uses the output of the first. While training the two networks could occur in parallel, this would require extra computational resources and the segmentation output would not benefit from the denoised output. Furthermore, in applications where only the denoising or the segmentation is desired, it is possible both the intermediate and the final output could be used either for segmentation or denoising. This means that the intermediate output would not be a secondary output, but just an entry point for additional supervision. As shown in [127, 315] additional supervision increases the accuracy of the final output.

5.4 Dataset and training

5.4.1 Highly-sampled representative tomograms of 4D datasets used for training

The highly-sampled tomogram (*RealHS1* as described in Section 3.2.2) used to train our networks is collected at the Diamond Light Source I13-2 beamline [416], and is used in [24]. As it is described in Chapter 3, the tomogram depicts a droplet of salt water on top of a 500 micron aluminium pin with magnesium deposits. A projection and a sinogram is depicted in Figures 3.4(**a**) and 3.5(**a**) respectively, and in Figure 5.7 there are example reconstructions with a segmentation map that is manually annotated. It is captured at the end of a time-series of undersampled tomograms that forms a 4D dataset, itself part of a study to measure the corrosion of the metallic pin by the salt-water droplet over time [418]. Since the tomograms captured from Diamond Light Source are $2160 \times 2560 \times 2560$ in size after reconstruction, even a single tomogram can in practice provide plenty of training instances after cropping. The highly-sampled tomogram, used in Section 5.5 is captured at the end of a time-series, and as such it has been possible to capture it with a high number of projections (namely 3601 instead of the 91 projections in the 4D dataset tomograms). This allows a high-quality reconstruction and so the acquisition of more precise annotations. Collection of these "book end" highly-sampled tomograms is

common practice for researchers dealing with 4D datasets, especially where the speed of the event to be captured precludes detailed scans at every time point. Depending on what is studied, highly-sampled tomograms are attained either before, after or both before and after the capture of a 4D dataset, providing tomograms that can be used as reference, but without interfering with the time-critical part of the study. These tomograms, even though they are not a part of the 4D dataset, contain valuable information about the features of the samples under study, which the researchers want to identify and label. This makes them representative to the 4D datasets allowing not only operations like the projection upsampling of time-series' tomograms as mentioned by Chapter 4, but also the denoising and semantic segmentation of the time-series' tomogram reconstructions.

The motivation for using a highly-sampled tomogram for the training of our networks is two-fold. Firstly, with this type of tomogram a high-quality reconstruction is achievable, which can be used as "ground truth" when training to denoise low SNR or low-quality reconstructions of undersampled tomograms. While the difference between low-quality and high-quality reconstructions is not limited to the existence of noise, the term "denoise" is used as an umbrella term that describes the improvement of visual quality. It is also possible through the use of denoising network to regain spatial information that is obscured in low-quality reconstructions, based on the prior provided by the representative tomograms and their high-quality reconstructions. Naturally, the high-quality reconstructions are not devoid of imaging artifacts, however due to their different and random placement between the low-quality and high-quality reconstruction of the same tomogram, the employed networks can not infer where to place artifacts on the denoised output based on the input. This means that the networks mainly use the parts of the input that are useful to better predict a high-quality output, and not how to predict the noise or the artifacts.

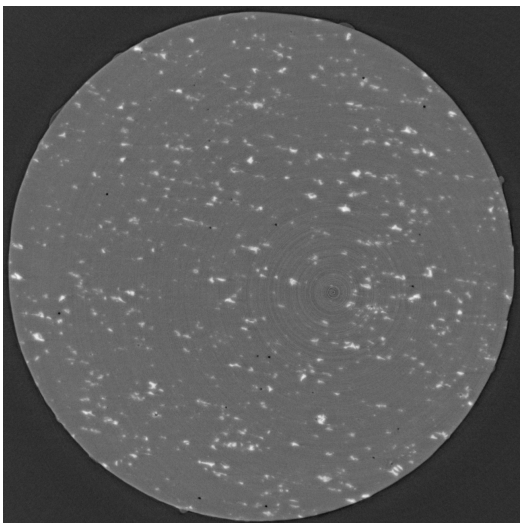
The second reason we train our networks using these representative highly-sampled tomograms is because their high-quality reconstructions can enable high-quality manual annotations, that can be considered a "golden standard ground truth" in terms of segmentation. Nevertheless, their high-quality reconstructions are not perfect acquisitions of the 3D structure of the imaged sample, but a very good approximation. Unfortunately,

sometimes due to the fact they are approximations, human annotation may unintentionally introduce bias to the training, validation and testing data, which in turn may reduce the true accuracy of the final segmentation [446]. This may be alleviated by annotating the same tomograms using multiple annotators, but when this bias is due to common human intuition there is not a practical way to address it using just real-world data. In a later chapter 6, we will demonstrate how this issue can be alleviated using synthetic tomograms.

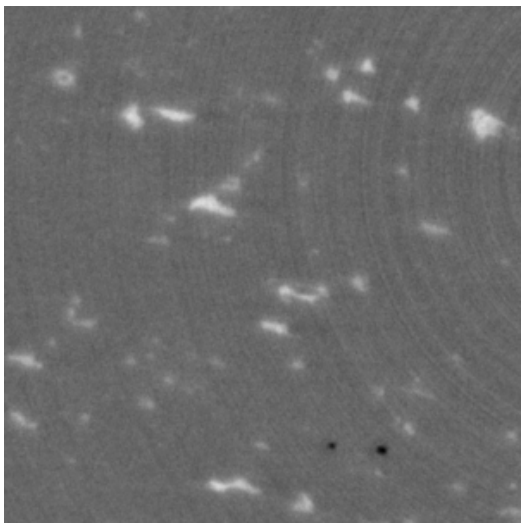
5.4.2 Tomogram reconstructions used for training

For training, a highly-sampled real-world tomogram is used, reconstructed using the Savu Python package [438]. From the tomogram, two reconstructions are produced. The first is made using the Filter Back Projection method (FBP) [409] algorithm and all 3601 available projections. The resulting high-quality reconstruction [425] provides the ground truth used in the denoising experiments and in the denoising part of Stacked-DenseUSeg. The second reconstruction is made with the Conjugate Gradient Least-Squares method (CGLS) [413] reconstruction algorithm [413] and uses only 91 projections ($1^{st}, 41^{st}, \ldots, 3601^{st}$ projections) ignoring some of the intermediate projections. This low-quality reconstruction [426], which is provided as input during training has the same visual quality as the reconstructions from the undersampled tomograms in the 4D datasets, when only a small number of projections are used. The annotations [424] of the tomogram were obtained using Luengo *et al.*'s SuRVoS [306] software tool using the high-quality reconstruction [425]. Figure 5.7 provides cross sections of the two reconstructions for the tomogram, as well as its annotations, which depict the classes that the network aims to segment.

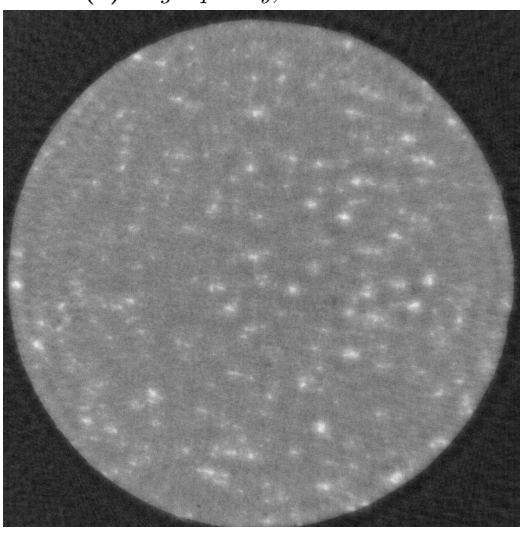
The reason two different algorithms are used for each of the two reconstructions is because the iterative CGLS method improves the reconstruction quality when applied to undersampled tomograms, while it offers worse results that the simpler FBP method when applied to highly-sampled tomograms. Alternatively, our proposed UDNN network [24] may be used to upscale the sinograms of the tomograms and therefore in conjunction



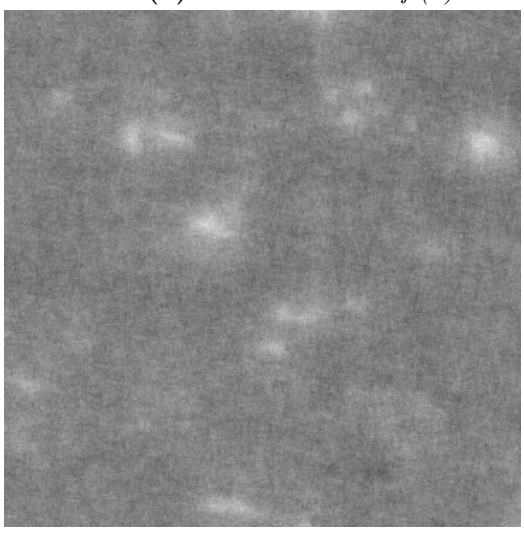
(a) *High-quality, FBP reconstruction*



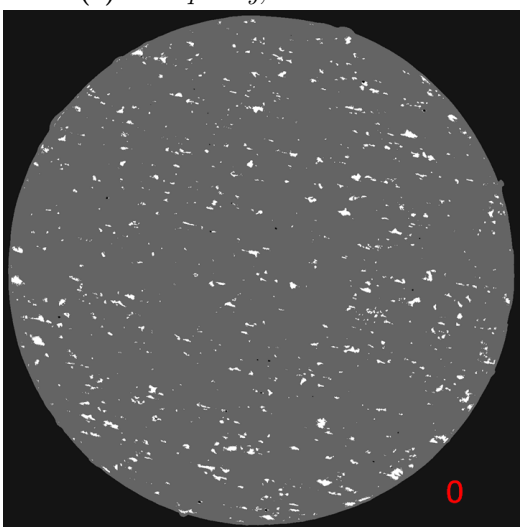
(b) *Zoomed version of (a)*



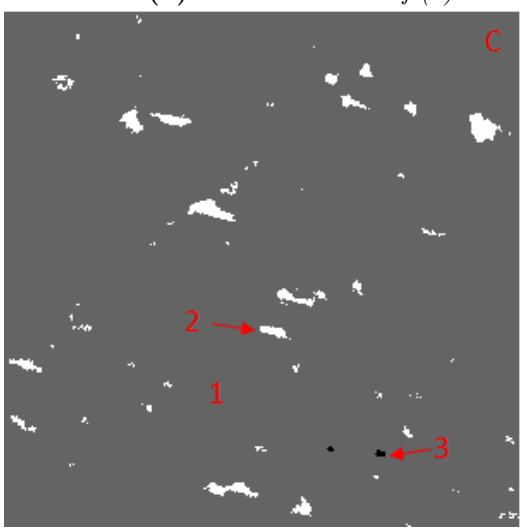
(c) *Low-quality, CGLS reconstruction*



(d) *Zoomed version of (c)*



(e) *Annotations*



(f) *Zoomed version of (e)*

Figure 5.7: *Cross section of RealHS1 tomogram reconstructions and annotations. (**a**) FBP reconstruction from 3601 projections (these are cropped and centered around the metallic pin of Label 0), (**b**) CGLS reconstruction from 91 projections (these are cropped and centered around the metallic pin of Label 0), (**c**),(**d**) are zoomed versions of (**a**),(**b**) respectively and (**e**) are annotations for the same area. Label 0 in (**b**) refers to the air outside the base material, Label 1 in (**e**) refers to the base material, Label 2 are the magnesium deposits and Label 3 are the air pockets*

with an FBP reconstruction to offer an elevated reconstruction of to which our proposed networks would train. This is especially, significant when our proposed DenseUSeg and Stacked-DenseUSeg will be used on 4D datasets, which are time-series of undersampled tomograms. This is due to the time CGLS [413] or other iterative reconstructive algorithms [411, 412] require to reconstruct a single undersampled tomogram, and as we presented in Section 4.6, when the time-series is composed of a large number of tomograms, training a UDNN model and then reconstructing all undersampled tomograms is faster. However, for the experiments presented in Section 5.5 we reconstructed the undersampled tomograms using the CGLS method, in order to ensure the reader that there is not a correlation between the results reported and the way which undersampled tomograms are reconstructed.

There are four classes to be segmented: the air outside the base material (Label 0 in Figures 5.7 (**c**), the base material which is the water and aluminium (Label 1 in Figures 5.7 (**f**), the magnesium deposits within the base material (Label 2 in Figures 5.7 (**f**) and lastly the air pockets within the base material (Label 3 in Figures 5.7 (**f**). While the base material class (Label 1) does contain both water and aluminium, water appears in very small traces as condensation around the aluminium pin and this class is practically mostly aluminium. Since the training of the network is performed using a single tomogram at time, they are normalised and split into multiple non-overlapping samples of $64 \times 64 \times 64$. Additionally, before entering the network each of the samples is either randomly rotated between 90°, 180°, 270° degrees, or it is horizontally or vertically mirrored.

From these non-overlapping samples of $64 \times 64 \times 64$ samples, 70% are randomly chosen for training, 10% for validation and 20% for testing. This results in, for the real-world tomogram, 3723 samples used for training, 532 used for validation and 1065 used for testing. The samples are cropped from the area of the tomograms that contain the desired classes to segment, because most voxels in the tomograms belong to Label 0 and sampling from the tomograms as a whole would create a great class imbalance. Thankfully, in most datasets (4D datasets or single tomograms) of this type [418] when the classes which have to be segmented are located in a single area of the tomogram, simple low-level imaging

analysis techniques (edge detection, thresholding, etc.) can be used to isolate the area that the networks have to be applied, allowing for faster and easier inference.

In order to further reduce any potential correlation between the samples, apart from being non-overlapping, samples are cropped as is shown in Figure 5.8. Namely, different "slabs" of samples are not perfectly aligned and their centers' x and z coordinate are different between the imaged object central *y-axis* and between themselves. With the term "slabs" we refer to different groupings of multiple $64 \times 64 \times 64$ samples which centres have the same y coordinate. In Figure 5.8 the red line is the central *y-axis* of the imaged object and the green spot is the center of the first "slab", and as it can be seen the different "slabs" have centers with different x and z coordinates. The "dislocation" of each "slab" along the x and *z-axis* is randomly chosen based on a normal distribution (one for each coordinate) with a standard deviation of 12 voxels. The mean is equal to zero since the position $x = 0, z = 0$ is the center of the imaged object. The dislocation in each coordinate may be positive or negative. Additionally, during inference on undersampled tomograms, they are separated to multiple samples (again $64 \times 64 \times 64$) which are though overlapping, and the final prediction occurs by averaging the predictions in the areas that overlap. The predictions, which are the probabilities of what class each voxel belongs to, after averaging are then used to infer the class for each voxel. Non-overlapping samples are used clear separation and they are assigned randomly for use during training, validation or testing, reducing any correlations between these 3 subgroups that would hinder the accurate evaluation of the network's performance. On the other hand, during inference separating the input volumes (tomogram reconstructions) to overlapping samples, eliminates potential border artifacts in the output subvolumes and helps resolve potential uncertainties for the class of some voxels.

5.4.3 Hyperparameters and training settings

For backpropagation, the stochastic gradient-based optimisation method Adam [435] is used, with mini-batches of 8. For DenseUSeg the learning rate is set to 10^{-5}, while for the Stacked-DenseUSeg there are two learning rates, with the first one set to 10^{-5} for the

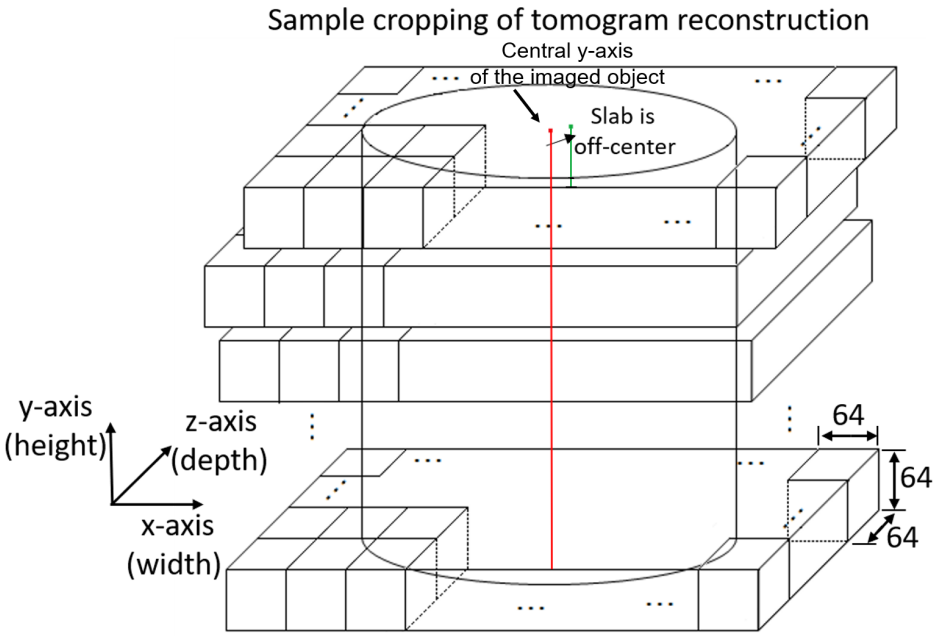


Figure 5.8: *Cropping of the tomogram reconstruction to provide training samples. As can be seen in the figure, the slabs' centers have different x and z coordinates. The slabs are groupings of multiple $64 \times 64 \times 64$ samples which centres have the same y coordinate. The red spot is the center of the imaged object along the y-axis and the green spot is the center of the first slab, and as it can be seen the different slabs' centers are not the same as the imaged object center or with each other.*

segmentation part of the network (2^{nd} DenseUSeg in the stack), and the second one set to 10^{-4} for the denoising part (1^{st} DenseUSeg in the stack). The reason there is such duality of learning rates present in the Stacked-DenseUSeg is because its first part, the denoising part, has to training faster than the second segmentation part which is partially dependent on the denoising output. In all tested networks (ours and the state-of-the-art), a weight decay [447] of 0.0005 is used, that acts as a L2 regularisation which applies penalties when parameters collectively get too large, which can lead to overfitting. It is also significant to mention that during training the learning rates are adjusted based on how many epochs have passed. In most experiments, based on preliminary testing, it is observed that a reduction of the learning rate by a factor of 10, at the middle of the training process (50^{th} epoch) is advantageous. This setting offers a sufficient time for the network to increase its accuracy in the first half of the training session, and in the second half to steadily converge to a local minima. This event, depending on the network that is used, may occur earlier than the last epoch and this is why after training, the final network parameters are set

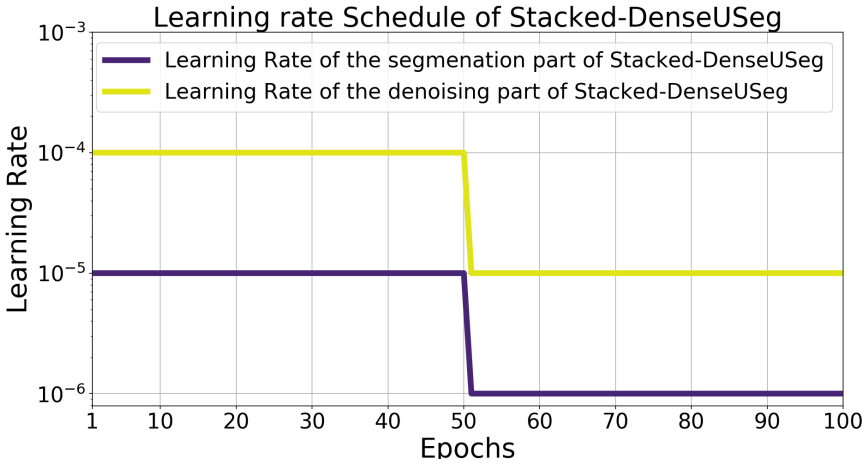


Figure 5.9: *The learning rate schedule of Stacked-DenseUSeg, both for its denoising part and its segmentation part*

to be the same as the ones during the epoch where the validation score was the highest. Subsequently, testing is performed using this network's image, and not the one during the last epoch of training. The learning rates and their change through time is visually displayed in Figure 5.9. While, a more complex learning rate schedule may have offer better results, based on preliminary testing we predict that the improvement would not be significant and may lead to a very data-dependant solution. Additionally, based on the fact that other important publications [129, 143] employ step-wise adjustments to the learning rate, this increases our confidence that more complex solutions are not necessary. Nevertheless the polynomial decay [275, 328] approach was tested, however regardless of the exponent value selected, the learning rate was either decaying very fast in the early epochs or to slow on the later epochs.

5.4.4 Validation and testing

Each of epoch lasts for 300 minibatches and the network passes through 100 epochs during training. The reason not all 3723 training samples are used within one epoch is because this would increase the time between each validation. Since the minibatch is 8, there are $300 \cdots 8 = 2400$ training samples used in each epoch. By keeping the number of training samples lower for each training epoch, each validation is performed more frequently and thusly the risk of the network overfitting by a high degree within one epoch is reduced.

As it was mentioned earlier the final weights of the network are the ones for the epoch that has the best validation score. Therefore, by increasing the frequency of validation, or in other words by decreasing the length of each epoch, it is more probable that a good state of the network would not be skipped due to overfitting that may occur in a longer epoch. Furthermore, in order to verify the accuracy of the network variations presented in Section 5.5, every training session is repeated 3 times in a 3-fold fashion. For each of these iterations the 70-10-20 split between training, validation and testing samples respectively is kept, however the samples for validation and testing are different without having any common samples between each iteration. The Weighted Cross Entropy loss criterion is used for the segmentation output and the Mean Square Error (MSE) for the denoising one. For a volume of size $W \times H \times D$, where W, H and D are the width, height and depth respectively, the Weighted Cross Entropy loss is calculated as such

$$WCE(\boldsymbol{x}, \boldsymbol{c}) = \frac{-1}{W \cdot H \cdot D} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \sum_{k=0}^{D-1} w(c(i,j,k)) \cdot \log(F(\boldsymbol{x})_{c(i,j,k),i,j,k}) \tag{5.3}$$

where $\boldsymbol{x}$ is the input volume, $\boldsymbol{c}$ is the segmentation map of the input that contains the ground truth class of all input voxels, $c(i,j,k)$ is the ground truth class of the voxel in position with indexes i,j,k, $F(\boldsymbol{x})_{c(i,j,k),i,j,k}$ is the output predicted probability that the voxel in position with indexes i,j,k belongs to the ground truth class $c(i,j,k)$ and $w(c(i,j,k))$ is the weight that have been attributed to class $c(i,j,k)$. Namely, the weights for the Cross Entropy are calculated as

$$\boldsymbol{w} = \frac{median(\boldsymbol{P}/\boldsymbol{S})}{\boldsymbol{P}/\boldsymbol{S}} \tag{5.4}$$

where $\boldsymbol{P}$ is a vector of the pixel counts of the different classes and $\boldsymbol{S}$ is a vector that contains the number of samples of each class is present. The combined criterion for Stacked-DenseUSeg is

$$\mathcal{L}_{Combined} = \mathcal{L}_{Cross\_Entropy} + \lambda \cdot \mathcal{L}_{Mean\_Square\_Error} \tag{5.5}$$

where λ balances the two losses and it is empirically set to 10. The $\mathcal{L}_{Cross\_Entropy}$ loss uses as ground truth the segmentation annotations while the $\mathcal{L}_{Mean\_Square\_Error}$ the high-quality reconstruction. The parameter λ attempts to balance the two losses, so that

they will have approximately the same magnitude and that they would not be some orders of magnitude different between each other. During experiments, by increasing the $\mathcal{L}_{Mean\_Square\_Error}$ by one order of power (10), we observed that the two losses were relatively in balance. Alternatively, a statistical approach could be used to dynamical adjust the two loss, however we leave this to be explored by future research.

5.4.5 Metrics and execution time

As a segmentation metric, the mean Intersection over Union (mIoU) is used. Specifically,

$$IoU_l = \frac{|\{u|u \in V_l\} \wedge \{p|p \in V_l\}|}{|\{u|u \in V_l\} \vee \{p|p \in V_l\}|} = \frac{TP_l}{TP_l + FP_l + FN_l} \tag{5.6}$$

where $u \in V_l$ are the voxels that belong to class l, and $p \in V_l$ are the predicted voxels that belong to class l. The IoU of class l can be calculated also from IoU's second definition as seen in the second fraction of equation 5.6. In it the TP_l is the total number of voxels predicted to belong to class l when they actually do belong, FP_l is the total number of voxels predicted to belong to class l but they do not belong in reality, and FN_l the total number of voxels predicted to belong to any other class other than l, but they belong to class l.

As shown by equation 5.6, the IoU of class l is the count of pixel/voxel predicted correctly to belong to class l divided by the same count plus the number of the voxels wrongfully predicted to belong to class l and the number voxels that belong to class l but are wrongfully predicted to belong to other classes. This means that the IoU of a class is penalised both by predicting pixels/voxels to belong to other classes when they belong to this class and by predicting pixels/voxels to belong to this class when they belong to others. This means that it is a good metric for segmentation, since it does not get high values by overpredicting or underpredicting the segmentation classes. Because of this the IoU, also known as the Jaccard index, is a popular metric for the quantitative measurement of segmentation quality as seen in other publications [268, 308, 328, 330].

For denoising, the Peak Signal-to-Noise Ratio (PSNR) metric (which is a logarithmic representation of the mean square error, see Equation 4.5), and the Structural Similar-

ity Index (SSIM) [439] (see Equation 4.6) are used to quantitatively evaluate the image restoration quality compared to the ground truth (in our case the high-quality reconstruction [425]). Training for 100 epochs takes approximately 18 hours on PyTorch [442]. The workstation that was used had 4 NVIDIA Tesla V100s GPUs, a 20-core 2.5GHz CPU and 380GB of RAM. Naturally, a termination condition can be employed when overfitting is detected and the validation error gets progressively worse. This in principle can reduce the time needed for training even further, however, the size of 100 epochs was chosen so that so that our proposed networks may display signs of overfitting during the last 10 epochs, therefore the advantages of such termination condition will probably not be especially noticeable.

5.5 Experiments

In this Section we present and compare the results obtained from our Stacked-DenseUSeg and DenseUSeg architectures to the state-of-the-art methods of DenseSeg [10], 3D U-Net [8] and VoxResNet [9]. In the following experiments, we increase the growth rate of DenseSeg [10] to $g = 64$, as with the use of the original growth rate we observed very low performance and additionally this setting allows us to produce results comparable to DenseUSeg (which also has $g = 64$).

There are three experiments presented in this Section:

1. Experiments regarding the **denoising** accuracy of Stacked-DenseUSeg and DenseUSeg in *low* SNR reconstructions of undersampled tomograms.
2. Experiments regarding the **segmentation** accuracy of DenseUSeg network under *optimal* conditions, in high SNR reconstructions. This experiment in conjunction with the results of the following experiment aids in measuring the accuracy gap between the high and low-quality reconstruction which is provided as input.
3. Experiments regarding **segmentation** accuracy of Stacked-DenseUSeg and DenseUSeg in low SNR reconstructions of undersampled tomograms. The results from this and the first experiment provide a full insight regarding the Stacked-DenseUSeg's two outputs which are generated simultaneously by the network.

Furthermore, while [8, 9, 10] are state-of-the-art approaches for the semantic segmentation of volumes, there are not tested for their denoising capabilities. However, while there are deep learning methods for denoising the reconstructions of undersampled tomograms, most [12, 13, 14, 15, 16, 17] utilise simple deep learning architectures, as the recent advantages of deep learning have only been recently showcased. Because of that, the recent and deeper [8, 9, 10] architectures have yet to be adopted for this task. Nevertheless, based on the general trend in deep learning network architectures, deeper architectures tend to offer higher performances regardless of the task (classification, segmentation, etc.). For this reason in this Section, we compare our proposed architectures against the more recent and deeper [8, 9, 10] approaches for both the task of denoising and segmentation.

5.5.1 Experiment 1: Denoising with DenseUSeg and Stacked-DenseUSeg

Regarding the task of denoising, we trained the aforementioned networks using the Mean Square Error (MSE) as the loss criterion except for Stacked-DenseUSeg which is trained based on the combined loss described in equation 5.5 in Section 5.4. The input to the networks is the low in quality (and SNR) reconstruction [426] (see Figure 5.11 (**a**)), and the high-quality reconstruction [425] (see Figure 5.11 (**b**)) acts as ground truth. As can seen from Figures 5.11 (**a**-**b**) the task of denoising the low SNR reconstruction is a strenuous task, due to the amount of excess noise that has to be removed and also the spatial information that has to be restored based on the prior provided by the representative highly-sampled tomograms. For this task, the final Softmax layer in the networks is removed or replaced with a $1 \times 1 \times 1$ convolution if there is not one present before it.

This experiment is performed in order to determine which architecture is best for the task of denoising. As shown in Table 5.2 our DenseUSeg architecture performs similar to VoxResNet [9] regarding the SSIM metric, however VoxResNet [9] outperforms it regarding the PSNR metrics. Nevertheless, since in literature [439] SSIM has proven to be a reliable indicator of the restored perceptual quality, while PSNR can be potentially sensitive to noise, we can deduce that both DenseUSeg and VoxResNet have almost identical

performance. Additionally, as can be seen in Figure 5.11 from mark $\boldsymbol{D_2}$, the DenseUSeg is able to restore the air pocket with the correct intensity and size compared to mark $\boldsymbol{D_5}$ from VoxResNet (Figure 5.11 (**g**)). As for the other networks from marks $\boldsymbol{D_3}$, $\boldsymbol{D_4}$. Furthermore, from mark $\boldsymbol{D_7}$ it can be seen that it is able to restore the difficult case where the base material consists of water and not aluminum compared to the other networks (marks $\boldsymbol{D_8}$, $\boldsymbol{D_9}$) and especially the VoxResNet (mark $\boldsymbol{D_{10}}$).

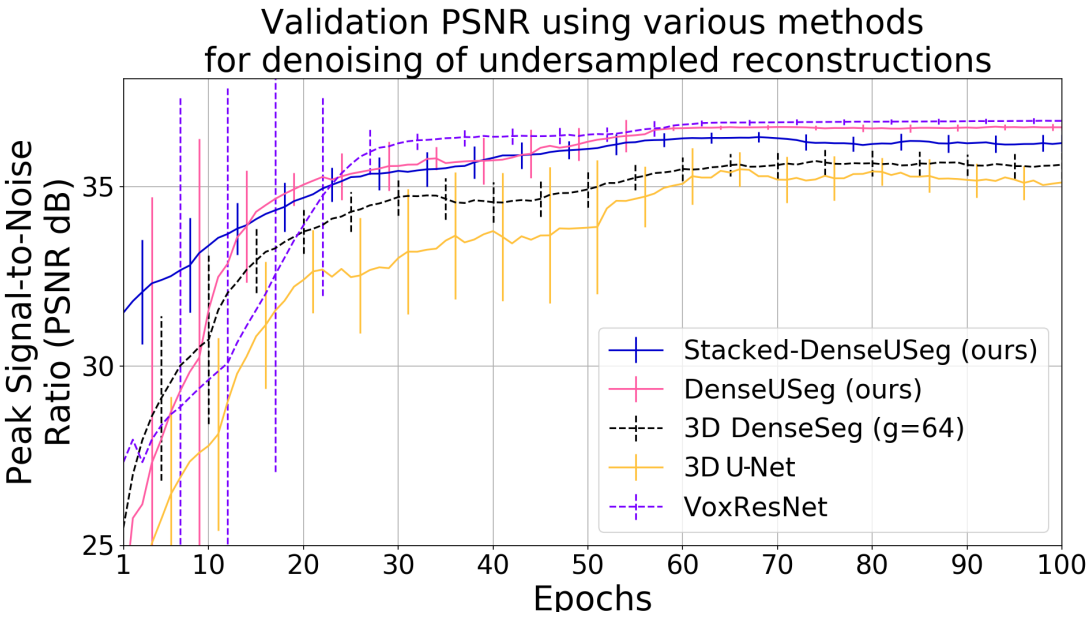
As will be shown later in Sections 5.5.2 and 5.5.3, DenseUSeg outperforms the other state-of-the-art methods in terms of segmentation accuracy and this is why it is selected as the segmentation part of Stacked-DenseUSeg. Since stacking of networks is easier when the networks are of the same type and in regards of denoising, DenseUSeg has better or similar results compared to the state-of-the-art; this justifies its selection as the denoising part of Stacked-DenseUSeg. The name of Stacked-DenseUSeg is then derived from the fact that it is the product of stacking two DenseUSeg networks. Table 5.2 displays the performance of Stacked-DenseUSeg's denoised output, and while it is not as accurate as a single DenseUSeg trained specifically for the task of denoising, it still outperforms DenseSeg [10] on PSNR and 3D U-Net [8] on both PSNR and SSIM [439]. However, considering the fact that Stacked-DenseUSeg's denoised output is its secondary, and not primary, output, this still sets its performance as noteworthy. Based on Figure 5.11 (**c**), it can be observed that the Stacked-DenseUSeg's denoised output is reasonably good in quality and perceptually comparable to the other networks. Namely, from mark $\boldsymbol{D_1}$ it can be observed that Stacked-DenseUSeg's denoised output, in a similar way to to DenseUSeg, has accurately denoised the difficult case of the air pocket compared to the other networks (marks $\boldsymbol{D_3} - \boldsymbol{D_5}$) and has also correctly denoised the water in mark $\boldsymbol{D_6}$ compared to the other state-of-the-art approaches (marks $\boldsymbol{D_8} - \boldsymbol{D_{10}}$).

Finally, it can be seen in Figures 5.10 (**a**-**b**) that both our networks (Stacked-DenseUSeg, DenseUSeg) converge stably and, based on the standard deviation in Table 5.2, the standard deviation our network's accuracy is similar to the other state-of-the-art methods, which provides evidence for their stability. Nonetheless, it is important to point out that by comparing Figures 5.11 (**c**-**g**) to Figure 5.11 (**b**) the exact structure of internal

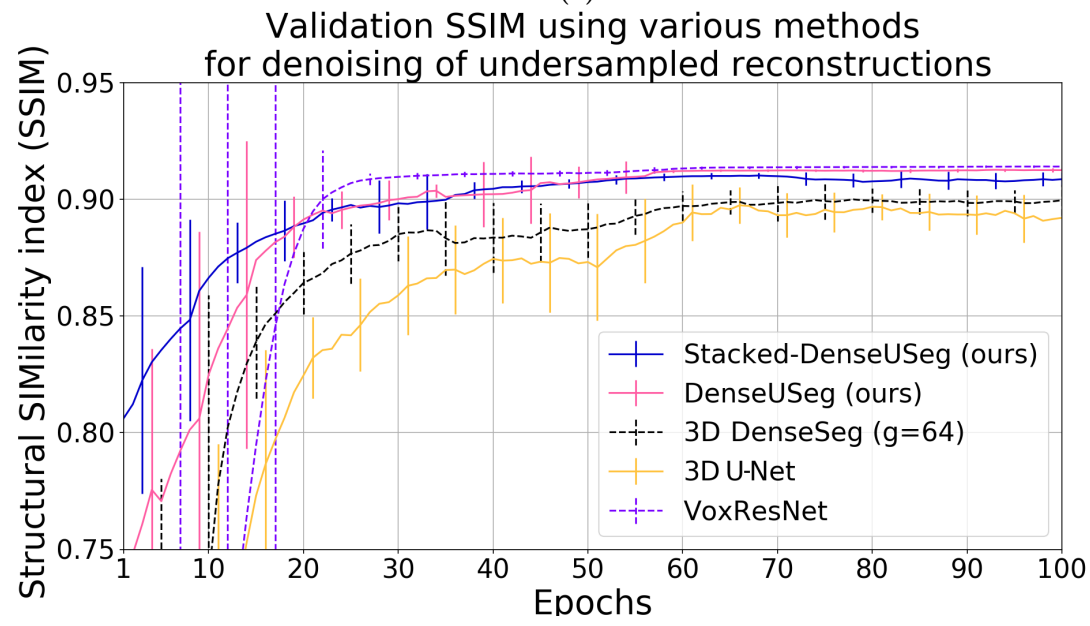
Method	PSNR (dB)	SSIM
Stacked-DenseUSeg (ours)	36.33 ± 0.4	$0.910 \pm 2 \cdot 10^{-3}$
DenseUSeg (ours)	$36.79 \pm 4 \cdot 10^{-2}$	$\mathbf{0.914} \pm 4 \cdot 10^{-4}$
DenseSeg [10] ($g = 64$)	$36.24 \pm 5 \cdot 10^{-3}$	0.906 ± 10^{-3}
3D U-Net [8]	35.98 ± 0.2	0.905 ± 10^{-3}
VoxResNet [9]	$\mathbf{36.86} \pm 7 \cdot 10^{-2}$	$\mathbf{0.914} \pm 4 \cdot 10^{-4}$

Table 5.2: PSNR and SSIM results from testing denoising using the 91-projection CGLS reconstruction as input and the 3601-projection FBP reconstruction as ground truth. The growth rate of DenseSeg is increased from 16 to 64 in order to offer more comparable results. The values presented are averages between the 3 iterations of the 3-fold validation/testing and the ones after the $\pm$ sign refer to the standard deviation that the approaches achieved in 3-fold validation/testing. For both metrics higher is better, while for standard deviation smaller is a proof for higher consistency and stability.

components cannot be recovered. This is due to large undersampling of projections in the input (the undersampled tomogram), and as can be observed in Figure 5.11 (**a**), there is permanent lose of spatial information which is impossible to be inferred by any method, due to how low the sampling rate is (from 3601 projections to 91 projections).

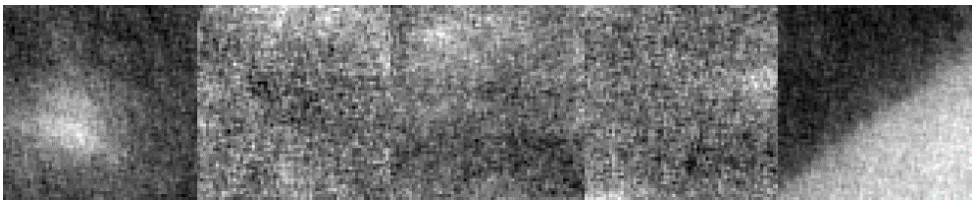


(a)

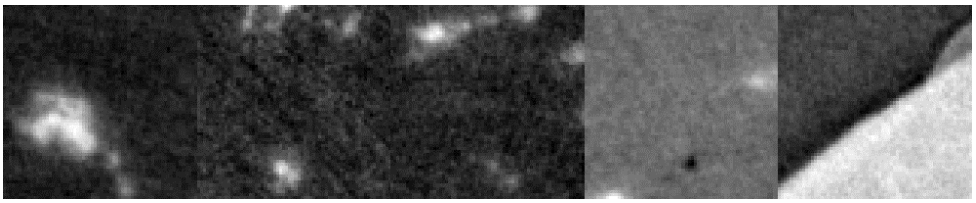


(b)

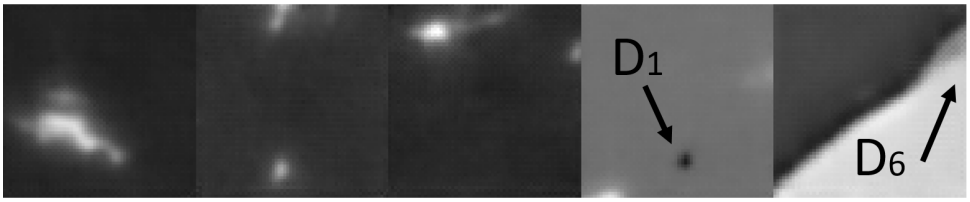
Figure 5.10: *(**a**) Validation PSNR results for the denoising experiments, (**b**) Validation PSNR results for the denoising experiments. All values are averages between the 3 iterations of the 3-fold validation/testing, which are also averaged based on a 10 epoch moving average. The error bars refer to the standard deviation that the approaches achieved based om a (3x)10 epoch moving standard deviation.*



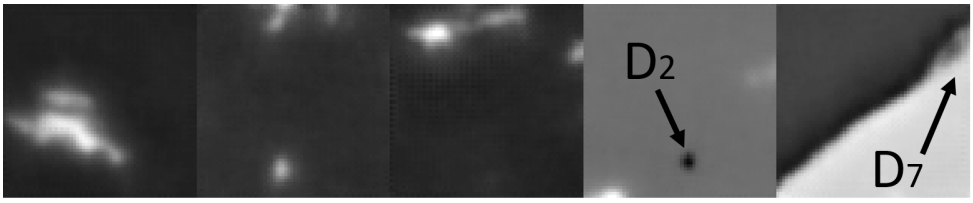
(a) *low-quality reconstruction from CGLS and 91 projections*



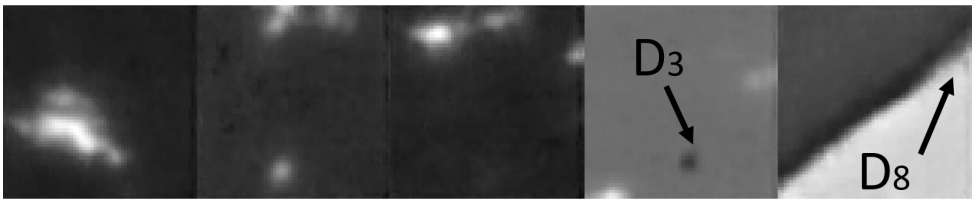
(b) *high-quality reconstruction from FBP and 3601 projections*



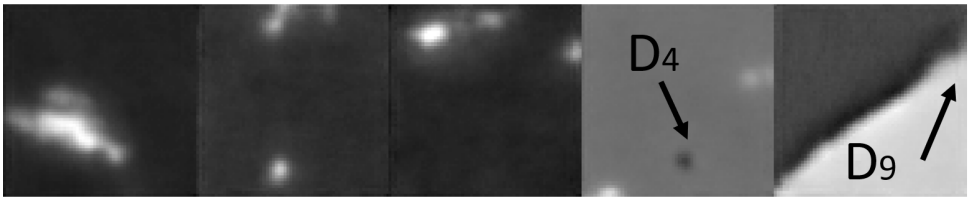
(c) *Stacked-DenseUSeg denoise output*



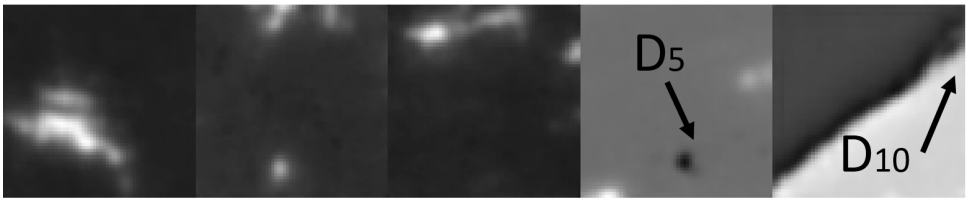
(d) *DenseUSeg denoise output*



(e) *DenseSeg (g=64) denoise output*



(f) *3D U-Net denoise output*



(g) *VoxResNet denoise output*

Figure 5.11: *Denoising Results. (**a**) Input to the denoising networks, the CGLS reconstruction from 91 projections [426], (**b**) the FBP reconstruction from 3601 projections that is ground truth for the denoising networks, (**c**)-(**g**) are the denoised outputs of the respective networks when they work to denoise (**a**) using (**b**) as ground truth. marks $\boldsymbol{D_1} - \boldsymbol{D_{10}}$ are described in text.*

5.5.2 Experiment 2: Segmentation with DenseUSeg using a high SNR reconstruction as the input

For the task of segmentation, DenseUSeg and the existing state-of-the-art methods are trained using as a loss criterion the Weighted Cross Entropy described in equation 5.3 in Section 5.4. As an initial first part of the segmentation experiments, we test DenseUSeg's segmentation accuracy, when the input is the high quality and high SNR reconstruction of the highly-sampled tomogram. As it can be seen from Figures 5.13 (**a**-**b**) it's relatively easy to detect the one-to-one correspondence between what is imaged in the tomography and its segmentation map. Naturally, the segmentation of this reconstruction is not without difficulties as the network has to perform accurately despite the presence of artifacts, specifically ring artifacts which can be observed in the Figure 5.7 (**b**). Furthermore, the exact edges of the classes with Labels 2 and 3 has to be determined by the network, which is a difficult task considering that their transition from class with Label 2 or 3 to the class with Label 1 is smooth and gradual transition.

The goal of this segmentation experiment is to learn the accuracy of the different networks under optimal conditions provided by the high-quality reconstruction, with the absence of noise and potentially less imaging artifacts which appear when using a smaller number of projections. For this experiment, it is important to mention that both DenseUSeg and DenseSeg are set with a Dropout rate of $p = 0$, as in preliminary tests it was seen that Dropout was not helpful when noise is absent. Thankfully, with the addition of batch normalisation [134] and weight decay [447] as was mentioned in Sections 5.3.5 and 5.4.3 respectively, there are enough prevention mechanisms that aid to reduce potential overfitting. In addition, Stacked-DenseUSeg is not listed in Tables 5.3, 5.4 and 5.5 as this network is designed to upscale the low-quality reconstruction [426] by using the high-quality reconstruction [425] and the annotations of Figure 5.7 (**a**-**b**) and (**e**-**f**) respectively as ground truths; in this first segmentation experiment, we are already given *optimal* data as input.

Based on Table 5.3 and the Figure 5.12, it can be deduced that under optimal data input conditions, DenseUSeg is able to outperform or match the performance of the other

methods in the Intersection over Union (IoU) metric of individual classes and in the mean IoU. Specifically, by observing Table 5.4 DenseUSeg works well not to oversegment the difficult classes with Labels 2 and 3, since it achieves the highest precision for these classes compared to the other approaches. Nevertheless as observed by the recall Table 5.5 it tends to slightly undersegment these classes.

In Figure 5.13 it can be seen from mark $\boldsymbol{O_1}$ that it is more precise on the edges compared other state-of-the-art methods marks $\boldsymbol{O_4}$, $\boldsymbol{O_{11}}$. Moreover, from marks $\boldsymbol{O_2}$ and $\boldsymbol{O_3}$, it is clear that apart from VoxResNet, DenseUSeg is the only network which is able to correctly segment the difficult case of a small particulate of class with Label 2, and when the base material is water and not aluminum the other methods fail to do that) respectively. By observing marks $\boldsymbol{O_6}$, $\boldsymbol{O_8}$ in for DenseSeg and 3D U-Net, it can be seen in that they either segment non existing particulates or wrongly segment the one present. Similarly, from marks $\boldsymbol{O_5}$, $\boldsymbol{O_7}$ it is clear that both are not able to correctly segment the water present at the edge, which is part of the class with Label 1, the base material.

Nonetheless VoxResNet does not present such a problems in marks $\boldsymbol{O_9}$ and $\boldsymbol{O_{10}}$, however from mark $\boldsymbol{O_{11}}$ it does not correctly segment the items that belong to the class with Label 2. By generally observing its segmentation output, it can be deduced that it can only offer segmentations with smooth edges, and while it is accurate in certain areas, the earlier fact decreases its accuracy for the difficult classes with Labels 2 and 3, which have more complex edges. Based on the above observations, DenseUSeg can be argued to be the best performing network under the conditions present in micro X-ray CT tomograms, and that fact justifies our selection of DenseUSeg for the segmentation part of Stacked-DenseUSeg. This, in conjunction with our findings in Section 5.5.1, is the reason our proposed Stacked-DenseUSeg is composed by two sequentially stacked DenseUSegs. Additionally, as can be seen in Figure 5.12, DenseUSeg is similarly stable during training as the rest of the state-of-the-art methods.

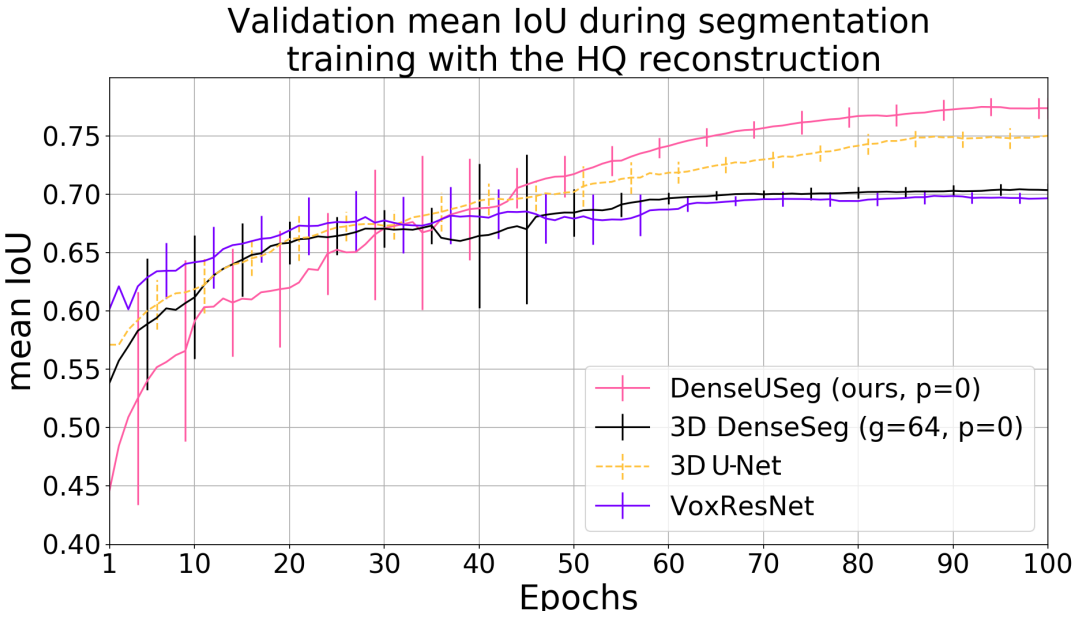


Figure 5.12: *Validation mean IoU for the segmentation experiments where the input is the low-quality FBP reconstruction using 3601 projections. All values are averages between the 3 iterations of the 3-fold validation/testing, and they are also averaged based on 10 epoch moving average. The error bar heights refer to the standard deviation that the approaches achieved based om a (3x)10 epoch moving standard deviation.*

Input: High-quality, 3601-projection FBP reconstruction [425]					
Testing IoU	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
DenseUSeg (ours, p=0)	**0.99**± $8 \cdot 10^{-4}$	**0.98**± $3 \cdot 10^{-4}$	**0.60**± $4 \cdot 10^{-3}$	**0.56**± $2 \cdot 10^{-2}$	**0.78**± $6 \cdot 10^{-3}$
DenseSeg [10] (g=64, p=0)	**0.99**± $6 \cdot 10^{-4}$	0.97± $2 \cdot 10^{-3}$	0.51± $2 \cdot 10^{-2}$	0.37± $4 \cdot 10^{-2}$	0.71± $4 \cdot 10^{-3}$
3D U-Net [8]	**0.99**± $2 \cdot 10^{-4}$	**0.98**± 10^{-3}	0.57± $2 \cdot 10^{-2}$	0.53± 10^{-2}	0.77± $6 \cdot 10^{-3}$
VoxResNet [9]	**0.99**± $3 \cdot 10^{-3}$	**0.98**± $5 \cdot 10^{-3}$	0.60± $6 \cdot 10^{-2}$	0.32± $2 \cdot 10^{-2}$	0.72± $9 \cdot 10^{-3}$

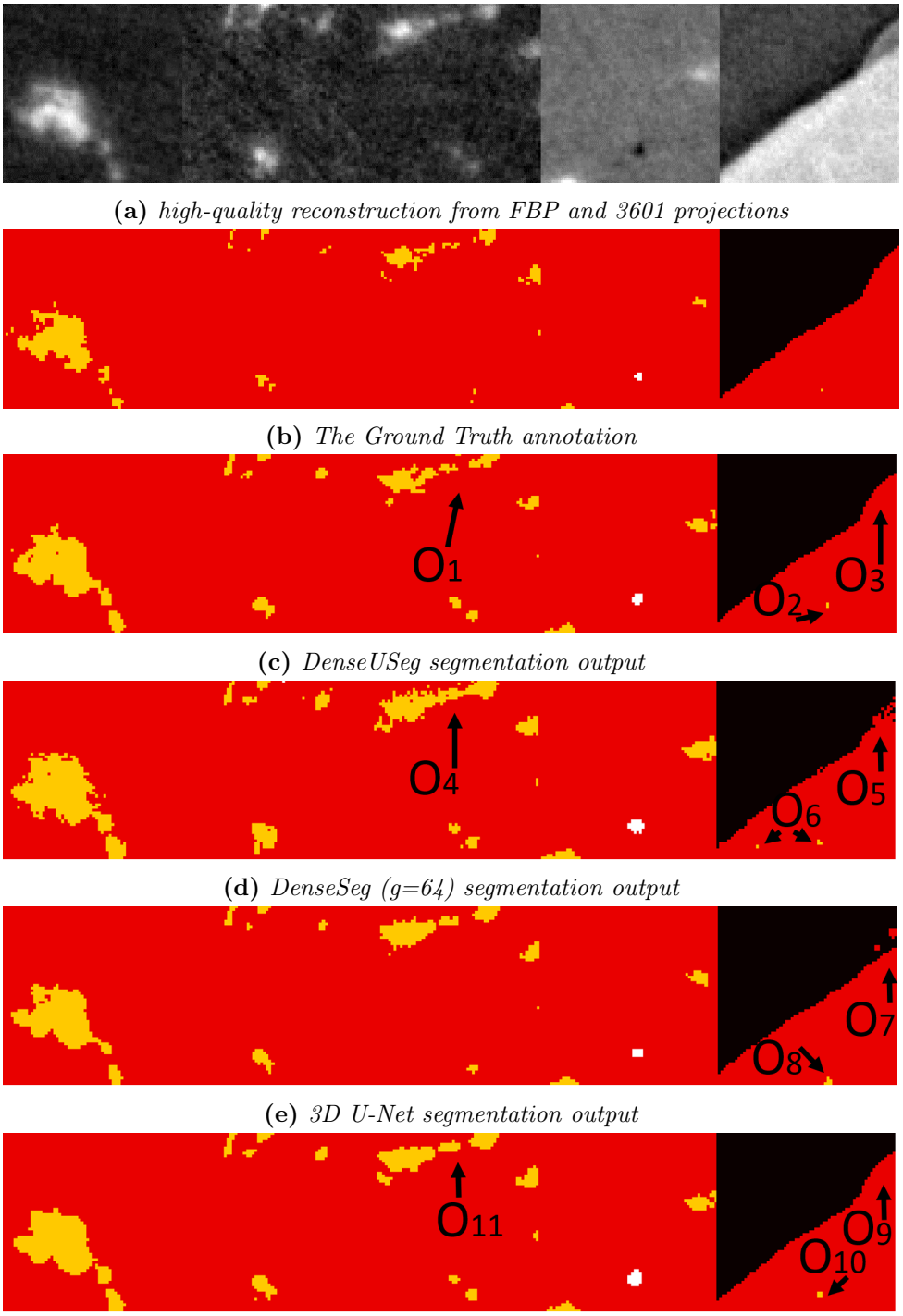
Table 5.3: Intersection over union (IoU) testing segmentation results using the 3601-projection FBP reconstruction [425]. The DenseSeg and DenseUSeg architectures are set to a Dropout rate $p = 0$ since from preliminary experiments it was shown that Dropout is not helpful when there is an absence of noise in the data. The growth rate of DenseSeg is set to 64 to offer more comparable results. The values presented are averages between the 3 iterations of the 3-fold validation/testing and the ones after the ± sign refer to the standard deviation that the approaches achieved in 3-fold validation/testing. Higher IoU is better, while for standard deviation smaller is a proof for higher consistency and stability. As it can be seen, our approaches and especially Stacked-DenseUSeg achieves an increased accuracy in the challenging Labels 2 and 3 over the other state-of-the-art methods.

Input: High-quality, 3601-projection FBP reconstruction [425]					
Testing Precision	Precision of Label 0	Precision of Label 1	Precision of Label 2	Precision of Label 3	mean Precision
DenseUSeg (ours, p=0)	**0.995**± $2 \cdot 10^{-4}$	0.997± $1 \cdot 10^{-4}$	**0.64**± $6 \cdot 10^{-3}$	**0.59**± $3 \cdot 10^{-2}$	**0.80**± $8 \cdot 10^{-3}$
DenseSeg [10] (g=64, p=0)	0.99± $9 \cdot 10^{-4}$	**0.999**± $3 \cdot 10^{-4}$	0.52± $2 \cdot 10^{-2}$	0.37± $4 \cdot 10^{-2}$	0.72± $6 \cdot 10^{-3}$
3D U-Net [8]	0.99± $6 \cdot 10^{-4}$	0.998± $2 \cdot 10^{-3}$	0.58± $2 \cdot 10^{-2}$	0.54± 10^{-2}	0.78± $6 \cdot 10^{-3}$
VoxResNet [9]	0.99± $4 \cdot 10^{-3}$	0.998± $8 \cdot 10^{-4}$	0.63± $7 \cdot 10^{-2}$	0.32± $2 \cdot 10^{-2}$	0.73± 10^{-2}

Table 5.4: Precision testing segmentation results using the 3601-projection FBP reconstruction [425]. The DenseSeg and DenseUSeg architectures are set to a Dropout rate $p = 0$ since from preliminary experiments it was shown that Dropout is not helpful when there is an absence of noise in the data. The growth rate of DenseSeg is set to 64 to offer more comparable results. The values presented are averages between the 3 iterations of the 3-fold validation/testing and the ones after the ± sign refer to the standard deviation that the approaches achieved in 3-fold validation/testing. Higher precision is better, while for standard deviation smaller is a proof for higher consistency and stability. As it can be seen, our approaches and especially Stacked-DenseUSeg achieves an increased accuracy in the challenging Labels 2 and 3 over the other state-of-the-art methods.

Input: High-quality, 3601-projection FBP reconstruction [425]					
Testing Recall	Recall of Label 0	Recall of Label 1	Recall of Label 2	Recall of Label 3	mean Recall
DenseUSeg (ours, p=0)	0.995± $8 \cdot 10^{-4}$	**0.98**± $3 \cdot 10^{-4}$	0.91± $4 \cdot 10^{-3}$	0.94± 10^{-2}	0.96± $4 \cdot 10^{-3}$
DenseSeg [10] (g=64, p=0)	0.997± $5 \cdot 10^{-4}$	0.97± $2 \cdot 10^{-3}$	**0.98**± $8 \cdot 10^{-3}$	**0.99**± 10^{-2}	**0.98**± $5 \cdot 10^{-3}$
3D U-Net [8]	0.997± $3 \cdot 10^{-4}$	**0.98**± 10^{-3}	0.95± $6 \cdot 10^{-3}$	0.96± $7 \cdot 10^{-3}$	0.97± $2 \cdot 10^{-3}$
VoxResNet [9]	**0.999**± $7 \cdot 10^{-4}$	**0.98**± $6 \cdot 10^{-3}$	0.94± $3 \cdot 10^{-2}$	0.998± $7 \cdot 10^{-4}$	**0.98**± $6 \cdot 10^{-3}$

Table 5.5: Recall testing segmentation results using the 3601-projection FBP reconstruction [425]. The DenseSeg and DenseUSeg architectures are set to a Dropout rate $p = 0$ since from preliminary experiments it was shown that Dropout is not helpful when there is an absence of noise in the data. The growth rate of DenseSeg is set to 64 to offer more comparable results. The values presented are averages between the 3 iterations of the 3-fold validation/testing and the ones after the ± sign refer to the standard deviation that the approaches achieved in 3-fold validation/testing. Higher recall is better, while for standard deviation smaller is a proof for higher consistency and stability. As it can be seen, our approaches and especially Stacked-DenseUSeg achieves an increased accuracy in the challenging Labels 2 and 3 over the other state-of-the-art methods.



(a) *high-quality reconstruction from FBP and 3601 projections*

(b) *The Ground Truth annotation*

(c) *DenseUSeg segmentation output*

(d) *DenseSeg (g=64) segmentation output*

(e) *3D U-Net segmentation output*

(f) *VoxResNet segmentation output*

Figure 5.13: *Segmentation Results. (**a**) Input to the segmentation networks, the FBP reconstruction from 3601 projections [426], (**b**) Annotations: Label 0 in black refers to the air outside the base material, Label 1 in red refers to the base material, Label 2 in yellow are the magnesium deposits and Label 3 in white are the air pockets, (**c**)-(**g**) Segmentation output of the respective networks, marks* $\boldsymbol{O_1 - O_{10}}$ *are described in text.*

5.5.3 Experiment 3: Segmentation with Stacked-DenseUSeg and DenseUSeg with a low SNR reconstruction as the input

Moving on from the second segmentation experiment, both Stacked-DenseUSeg and DenseUSeg are here tested against the state-of-the-art methods using as input the low-quality reconstruction [426] of the tomogram. As a loss criterion for the networks, DenseUSeg uses Weighted Cross Entropy described in equation 5.3 and the Stacked-DenseUSeg uses the combined criterion described in equation 5.5. With this final segmentation experiment both our networks are tested under the conditions that undersampled tomograms provide, where the resulting reconstructions have a very poor SNR and are very low in quality. As can seen from Figures 5.13 (**a**-**b**) the task of segmenting the low in SNR reconstructions of undersampled tomograms is extremely hard, considering the fact that the more simple task of denoising the same reconstruction is already very hard. This is the motivation why we developed the Stacked-DenseUSeg network, which utilises high-quality reconstructions from highly-sampled representative tomograms as a prior to train itself to initially denoise and then segment the low-quality reconstructions of the undersampled tomograms.

As can be seen from Table 5.6 and Figure 5.14, both DenseUSeg and Stacked-DenseUSeg achieve higher IoU compared to the other state-of-the-art methods. Stacked-DenseUSeg performs particularly well compared to the other approaches, presumably because of the denoised output. As can be seen in Table 5.7, our DenseUSeg and especially our Stacked-DenseUSeg approach, does not oversegment the difficult 2^{nd} and 3^{rd} label classes, which due to the small size of items that they represent, lose a lot of their spatial details (see Figures 5.11 (**a**-**b**)) during the projection downsampling (from 3601 to 91 projections). As seen in Table 5.8 DenseUSeg undersegments these classes. However, Stacked-DenseUSeg improves on these classes, by achieving not only a higher precision but also a much higher recall than DenseUSeg.

In terms of qualitative results, examining marks $\boldsymbol{S_1}$ and $\boldsymbol{S_2}$ in Figure 5.15 shows that Stacked-DenseUSeg can offer more accurate segmentation predictions regarding the difficult class with Label 2, presumably due to utilising its denoising part. Moreover, it is the only approach that accurately separates objects of the same class that are in close proximity

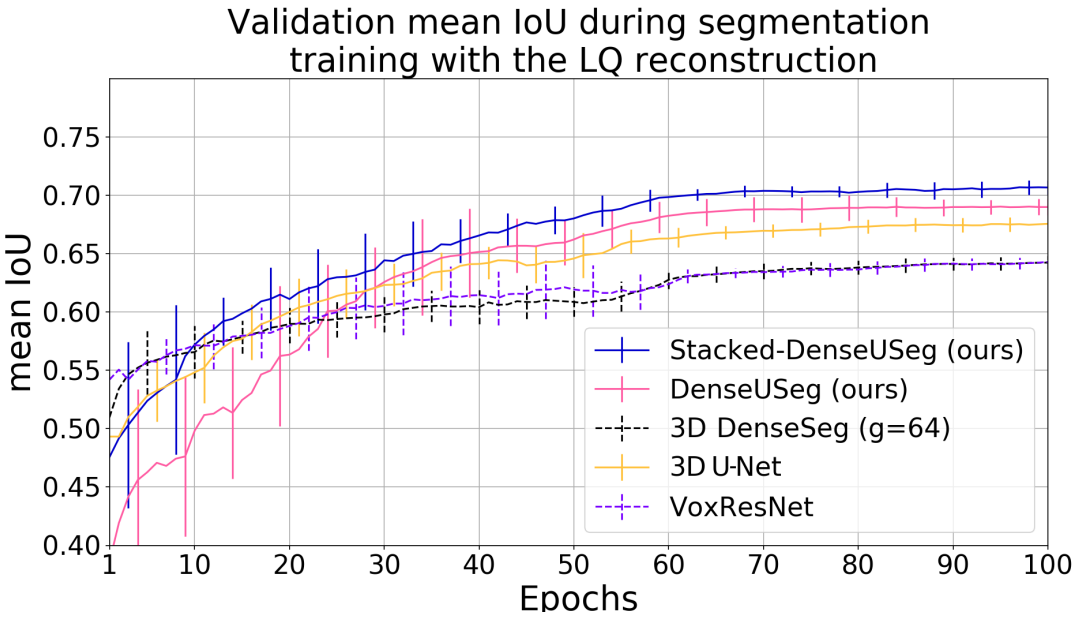


Figure 5.14: *Validation mean IoU for the segmentation experiments where the input is the low-quality CGLS reconstruction using 91 projections. All values are averages over the 3 iterations of the 3-fold validation/testing, and they are also averaged based on a 10 epoch moving average. The error bars refer to the standard deviation that the approaches achieved based on a (3x)10 epoch moving standard deviation.*

Input: Low-quality, 91-projection CGLS reconstruction [426]					
Testing IoU	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
Stacked-DenseUSeg (ours)	**0.99**± $2 \cdot 10^{-4}$	**0.97**± $3 \cdot 10^{-4}$	**0.45**± $5 \cdot 10^{-3}$	**0.46**± 10^{-2}	**0.72**± $4 \cdot 10^{-3}$
DenseUSeg (ours)	0.98± $5 \cdot 10^{-3}$	**0.97**± $5 \cdot 10^{-3}$	0.43± $2 \cdot 10^{-2}$	0.41± $2 \cdot 10^{-2}$	0.70± $2 \cdot 10^{-3}$
DenseSeg [10] (g=64)	**0.99**± $7 \cdot 10^{-5}$	0.95± $2 \cdot 10^{-3}$	0.36± 10^{-2}	0.30± $2 \cdot 10^{-2}$	0.65± $7 \cdot 10^{-3}$
3D U-Net [8]	**0.99**± 10^{-4}	0.96± $3 \cdot 10^{-3}$	0.37± 10^{-2}	0.43± $5 \cdot 10^{-3}$	0.69± $3 \cdot 10^{-3}$
VoxResNet [9]	0.98± $2 \cdot 10^{-4}$	0.96± $3 \cdot 10^{-3}$	0.42± $2 \cdot 10^{-2}$	0.27± $2 \cdot 10^{-2}$	0.66± $3 \cdot 10^{-4}$

Table 5.6: Intersection over union (IoU) testing segmentation results using the 91-projection CGLS reconstruction [426] as input. The growth rate of DenseSeg is set to 64 to offer more comparable results. The values presented are averages between the 3 iterations of the 3-fold validation/testing and the ones after the ± sign refer to the standard deviation that the approaches achieved in 3-fold validation/testing. Higher IoU is better, while for standard deviation smaller is a proof for higher consistency and stability. As it can be seen, our approaches, especially Stacked-DenseUSeg, achieve an increased accuracy in the challenging Labels 2 and 3 over the other state-of-the-art methods.

Input: Low-quality, 91-projection CGLS reconstruction [426]					
Testing Precision	Precision of Label 0	Precision of Label 1	Precision of Label 2	Precision of Label 3	mean Precision
Stacked-DenseUSeg (ours)	**0.99**± 10^{-3}	0.992± $5 \cdot 10^{-4}$	**0.53**± 10^{-2}	**0.60**± $2 \cdot 10^{-2}$	**0.78**± $8 \cdot 10^{-3}$
DenseUSeg (ours)	**0.99**± $2 \cdot 10^{-3}$	0.991± 10^{-3}	0.51± $5 \cdot 10^{-2}$	0.57± $2 \cdot 10^{-2}$	0.77± $2 \cdot 10^{-2}$
DenseSeg [10] (g=64)	**0.99**± $5 \cdot 10^{-4}$	**0.996**± $7 \cdot 10^{-4}$	0.37± $2 \cdot 10^{-2}$	0.29± $4 \cdot 10^{-3}$	0.66± $3 \cdot 10^{-3}$
3D U-Net [8]	**0.99**± $3 \cdot 10^{-3}$	0.994± $5 \cdot 10^{-4}$	0.40± 10^{-2}	0.50± 10^{-2}	0.72± $3 \cdot 10^{-3}$
VoxResNet [9]	**0.99**± $3 \cdot 10^{-3}$	**0.996**± $2 \cdot 10^{-4}$	0.45± $2 \cdot 10^{-2}$	0.27± $2 \cdot 10^{-2}$	0.68± $6 \cdot 10^{-4}$

Table 5.7: Precision testing segmentation results using the 91-projection CGLS reconstruction [426] as input. The growth rate of DenseSeg is set to 64 to offer more comparable results. The values presented are averages between the 3 iterations of the 3-fold validation/testing and the ones after the ± sign refer to the standard deviation that the approaches achieved in 3-fold validation/testing. Higher precision is better, while for standard deviation smaller is a proof for higher consistency and stability. As it can be seen, our approaches, especially Stacked-DenseUSeg, achieve an increased accuracy in the challenging Labels 2 and 3 over the other state-of-the-art methods.

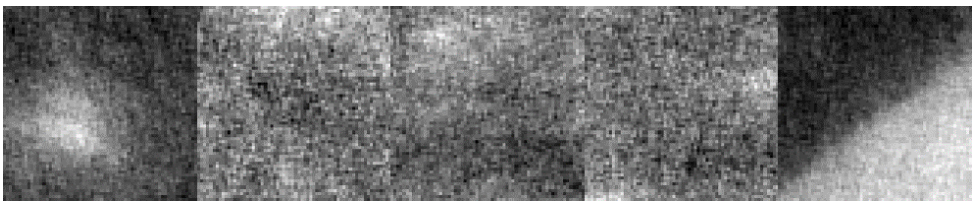
Input: Low-quality, 91-projection CGLS reconstruction [426]					
Testing Recall	Recall of Label 0	Recall of Label 1	Recall of Label 2	Recall of Label 3	mean Recall
Stacked-DenseUSeg (ours)	0.995± $8 \cdot 10^{-4}$	**0.98**± $7 \cdot 10^{-4}$	0.75± $8 \cdot 10^{-3}$	0.66± $3 \cdot 10^{-2}$	0.85± $8 \cdot 10^{-3}$
DenseUSeg (ours)	0.991± 10^{-3}	**0.98**± $7 \cdot 10^{-3}$	0.75± $6 \cdot 10^{-2}$	0.61± $7 \cdot 10^{-2}$	0.83± $3 \cdot 10^{-2}$
DenseSeg [10] (g=64)	0.993± $3 \cdot 10^{-3}$	0.96± $3 \cdot 10^{-3}$	**0.90**± $9 \cdot 10^{-3}$	**0.96**± $4 \cdot 10^{-3}$	**0.95**± $1 \cdot 10^{-3}$
3D U-Net [8]	0.993± $3 \cdot 10^{-3}$	0.96± $4 \cdot 10^{-3}$	0.84± $9 \cdot 10^{-3}$	0.75± 10^{-2}	0.89± $2 \cdot 10^{-3}$
VoxResNet [9]	**0.998**± $6 \cdot 10^{-4}$	0.97± $3 \cdot 10^{-3}$	0.88± 10^{-2}	**0.96**± $5 \cdot 10^{-3}$	**0.95**± $3 \cdot 10^{-3}$

Table 5.8: Recall testing segmentation results using the 91-projection CGLS reconstruction [426] as input. The growth rate of DenseSeg is set to 64 to offer more comparable results. The values presented are averages between the 3 iterations of the 3-fold validation/testing and the ones after the ± sign refer to the standard deviation that the approaches achieved in 3-fold validation/testing. Higher recall is better, while for standard deviation smaller is a proof for higher consistency and stability. As it can be seen, our approaches, especially Stacked-DenseUSeg, achieve an increased accuracy in the challenging Labels 2 and 3 over the other state-of-the-art methods.

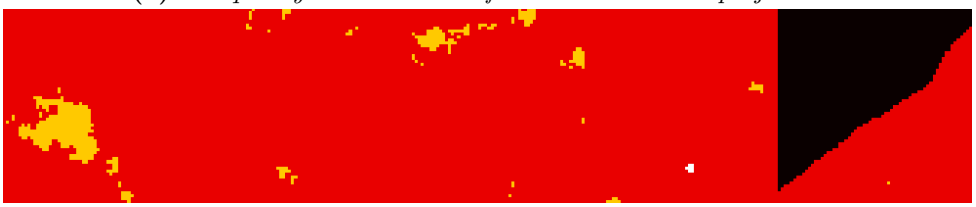
(mark $\boldsymbol{S_2}$) and according to mark $\boldsymbol{S_2}$ and Table 5.7 it does not overestimate the size of the internal components in the tomogram. Additionally apart from VoxResNet, it is the only method that accurately segments the water in mark $\boldsymbol{S_4}$ as class with Label 1. Finally, with the exception of 3D U-Net, the segmentation output of Stacked-DenseUSeg in mark $\boldsymbol{S_3}$ is not oversized like in results of the other approaches, which can be attributed to the high precision of Stacked-DenseUSeg and 3D U-Net (see Table 5.7) for the class of Label 3 (Air Pockets).

As regards to the segmentation output of DenseUSeg, it is evident that it not quite as accurate compared to the one of Stacked-DenseUSeg, which is expected due to absence of denoising components. However, by observing Figure 5.15(**d**) and specifically the area where mark $\boldsymbol{S_6}$ it can be observed that it is the only network which detected the existence of an item that belongs to the class with Label 2. Moreover, based on mark $\boldsymbol{S_5}$, VoxResNet and this (mark $\boldsymbol{S_9}$) are the only networks that more accurately segmented the water which is part of the base material. Their solutions are slightly less accurate than the one of Stacked-DenseUSeg (mark $\boldsymbol{S_4}$), however they are not as inaccurate as the ones of 3D U-Net and DenseSeg (marks $\boldsymbol{S_7}$, $\boldsymbol{S_8}$).

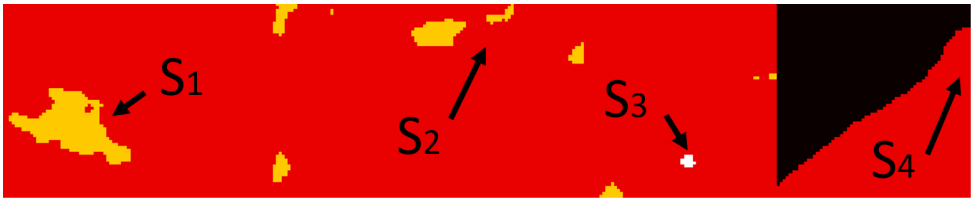
Finally, by examining Figure 5.14 and the standard deviations presented in the Tables 5.6, 5.7, 5.8, both of our approaches are similar in terms of consistency and stability of their accuracy to other state-of-the-art methods. Since also similar findings are present in Figure 5.12 and Table 5.3, we can deduce that the successful training and testing of our approaches are not due to irregularities in the distribution of data between these two tasks, but rather because of their structure.



(a) *low-quality reconstruction from CGLS and 91 projections*



(b) *The Ground Truth annotation*



(c) *Stacked-DenseUSeg segmentation output*



(d) *DenseUSeg segmentation output*



(e) *DenseSeg (g=64) segmentation output*



(f) *3D U-Net segmentation output*



(g) *VoxResNet segmentation output*

Figure 5.15: *Segmentation Results. (**a**) Input to the segmentation networks, the CGLS reconstruction from 91 projections [426], (**b**) Annotations: Label 0 in black refers to the air outside the base material, Label 1 in red refers to the base material, Label 2 in yellow are the magnesium deposits and Label 3 in white are the air pockets, (**c**)-(**g**) Segmentation output of the respective networks, marks* $\boldsymbol{S_1} - \boldsymbol{S_9}$ *are described in text.*

5.6 Conclusions

In this chapter, we propose the DenseUSeg [26, 441] (code available from [28]) architecture which is able to outperform the state-of-the-art accuracy of DenseSeg [10], 3D U-Net [8] and VoxResNet [9] networks regarding the volumetric semantic segmentation of highly-sampled or undersampled tomograms (sampling refers to the number of projections). Our approach expands upon and further improves, the DenseSeg [10] architecture which employs densely connected convolutional layers in order to perform volumetric semantic segmentation. Initially there was a number of network architectures that were tested, however as mentioned in Section 5.3.4 the DenseSeg architecture was selected as as template. Our approach expands upon the DenseSeg, by utilising a "expansive" path of repeated transposed convolutions and normal convolutions similar to 3D U-Net [8]. From preliminary experiments and the ones present in Section 5.5, it is evident that the addition of a decoder similar to one employed by U-Net, in conjunction with the densely connected layers present at the encoder part of the network results in an elevated accuracy of segmentation.

Furthermore, from the experiments in Section 5.5.1, DenseUSeg is also tested for the task of denoising tomogram reconstructions that have low SNR where their imaging quality is low. These reconstructions are a product of undersampled tomograms (low numbers of projections), as they are usually components of time-series of tomograms forming 4D datasets. The reason these 4D datasets are composed of undersampled tomograms is that in order to collect a sufficient number of tomograms within a limited time window, when time-critical events occur, the capture process must be fast, meaning the number of projections captured is limited. This limits the time for the collection of each tomogram and for the collection of its projections with the appropriate exposure time; only a small number of projections can be collected per tomogram. Therefore, the resulting reconstructions of the tomograms have a low SNR and their imaging quality is correspondingly low. Nevertheless, highly-sampled representative tomograms are usually captured either at the beginning of the collection, the end of the collection or both, offering high-quality reconstructions. Using these reconstructions it possible to train networks to denoise

the low-quality reconstructions of the undersampled tomograms of the 4D dataset. As is shown from the experiments in Section 5.5.1, DenseUSeg is capable of offering denoising results that are either better or equal to other state-of-the-art networks. Additionally, as it presented in Sections 5.5.2 and 5.5.3 the same network is able to outperform the other state-of-the-art approaches. Inspired from this fact, we also propose a second network.

In order to better utilise the highly-sampled tomograms that are usually collected with 4D data collections, we also proposed Stacked-DenseUSeg [26, 441] (citation for code [28]), a network which produces *two* different outputs, a denoised and a segmented version of the input. The network is trained to first denoise the input, and then segment it in an end-to-end fashion. This results in an improved accuracy in segmentation while an additional denoised output is attained. The way in which Stacked-DenseUSeg is formed is by stacking two DenseUSeg's sequentially similar to the way Newell *et al.* [315] proposed. Since the first part of the network calculates a denoised version of the input, through the passing of the relevant feature maps, the network is able to aid the second part of the stack, and achieve higher segmentation accuracy compared to use a single DenseUSeg dedicated for segmentation.

As can be seen in Tables 5.6 and 5.7 and Figure 5.14, the Stacked-DenseUSeg outperforms the state-of-the-art in the IoU and the precision metrics, along with our DenseUSeg approach. Nonetheless, our methods from Tables 5.8 rank low on recall. In total, however according to the IoU results of Table 5.6 our approaches overall (accounting equally for both precision and recall) outperform the state-of-the-art in terms of segmentation accuracy. Furthermore, from Tables 5.3, 5.4, 5.5 and Figure 5.12, similar findings can be observed for DenseUSeg regarding the segmentation using as input the a high-quality reconstruction of the representative highly-sampled tomogram. Based on the above, we conclude that Stacked-DenseUSeg is a well-placed approach to accurately denoise and segment undersampled CT tomograms, and therefore the 4D datasets collected at Synchrotron facilities such as Diamond Light Source.

Chapter 6

Denoising and segmentation knowledge transfer via the use of synthetically generated tomograms

6.1 Network parameter initialisation via trained networks on synthetic data

In the previous chapter, we presented our end-to-end deep learning approach to segment and denoise undersampled tomograms, which also referenced to in the literature as low-dose or sparse-angle(view) [12, 153] tomograms. These tomograms are often part of time-series of tomograms as showcased in Chapter 3. By utilising highly-sampled representative tomograms that usually accompany such time-series, we demonstrated that it is possible to train deep learning networks and acquire both a *denoised* version of the input along with accurate *segmentation* predictions. Nevertheless, in order to be able to achieve the former, these highly-sampled tomograms have to be manually annotated and vetted by experts. Either with the annotation of multiple small tomograms or fewer large ones, the demand of manual annotation of an adequate number of training samples is a major bottleneck in the process of applying deep network approaches. Additionally, between different 4D datasets, the type and segmentation classes of the imaged object may differ greatly, which requires the training process to be repeated with new representative highly-sampled tomograms that have to be annotated. Because of this, an effective approach needs to be developed in order to reduce the amount of required annotated data, while not compromising the accuracy of both denoising and segmentation.

As was presented in Section 3.2.3, it is feasible to create synthetic tomograms using specialised software [23]. Since the software operates by using a known model and simulates the physical process of an X-ray tomography, it is possible to use the resulting reconstructions as training inputs, and the model/phantom as the segmentation ground truth for deep learning applications. Provided that the synthetic tomograms are sufficiently similar to the real-world tomograms, which is often possible with materials science samples, training with synthetic data may enable the use of less real-world data. One of the proposed ways that these synthetic tomograms can be utilised is to initially train deep learning networks and then retrain these networks using real-world data. This methodology has also been used in the other work, especially in applications where classification networks are modified for segmentation [308]. This method has been proven to be an effective way to increase the overall accuracy or to reduce the amount of real-world annotated training data while keeping the accuracy at the same levels.

Inspired by the earlier, this chapter will present our proposed *knowledge transfer scheme* for achieving high levels of segmentation accuracy with our Stacked-DenseUSeg approach (see chapter 5) in situations where the available amount of real-world annotated data is small, which has been published to the Machine Vision and Applications journal [27]. We achieve the above *knowledge transfer scheme* by employing Kazantsev *et al.*'s [23] TomoPhantom software to construct synthetic tomograms, using randomly-placed simple virtual geometrical objects (e.g. cylinders, ellipsoids, etc.) to approximate the different classes present in the real-world data. The synthetic tomograms are then used to train networks. Through this process, key knowledge such as the attenuation levels of different classes, descriptive shape features and so on, can be transferred to the parameters of these networks' convolutional layers, using procedurally-synthesised tomograms that are produced already annotated. These networks are then used as a stepping stone, initialising the parameters of new networks before the latter are further trained/fine-tuned using a small amount real-world data. Provided that the synthetic tomograms approximate a broader family of different real-world tomograms with the same segmentation classes, these "pretrained" networks can be used multiple times for the fine-tuning of new net-

works targeted towards the segmentation and/or denoising of different real-world datasets. Therefore, the additional time cost for manufacturing synthetic tomograms and training "pretrained" networks is not repeated for every new real-world tomogram, the user would like to segment or denoise. Furthermore, the CT capturing conditions provided by a synchrotron always allow for only a very limited time window due to operating costs, thus any additional time spent in preparation (generating synthetic data or pretrained models) is preferable than more time spent on the CT collection. In summary in this chapter:

1. We propose a scheme that transfers knowledge from procedurally generated synthetic tomograms to instances of our proposed networks, referred to as the *pretrained* nets. These pretrained nets are then used for parameter initialisation of new networks, which after fine-tuning with real-world data may potentially attain higher accuracy compared to the non-use of knowledge transfer.
2. We present the effects of knowledge transfer to the demand for real-world annotated data, in order for our Stacked-DenseUSeg approach to be sufficiently accurate on denoising and segmenting real-world undersampled tomograms.
3. We present the effects of knowledge transfer to the time-demands for acquiring a trained Stacked-DenseUSeg that can denoise and segment real-world undersampled tomograms.
4. We compare the denoising and segmentation accuracy of the Stacked-DenseUSeg after using knowledge transfer using 2 different pretrained nets. One is generated for a synthetic tomogram *with* rotating artifacts and another for a synthetic tomogram *without* rotating artifacts.

The next Sections of this chapter are organised as follows. Section 6.2 is going to describe the methodology that our knowledge transfer follows, as well as the synthetic tomograms used for the generation of the pretrained nets. Section 5.4 will describe the training process for the creation of the pretrained nets and the further training/fine-tuning using real-world data which performed after the application the parameter initialisation on new networks. Section 5.5 will present experimental results and the effects of knowledge transfer both to the denoising and segmentation accuracy of Stacked-DenseUSeg on real-world

undersampled tomograms. Additionally, it investigates the effects of knowledge transfer to the amount of real-world annotated data needed to successfully train an accurate network on real-world data. Section 6.5 details how our proposed knowledge transfer scheme affects the time required to develop a Stacked-DenseUSeg that can denoise and segment real-world undersampled tomograms. Finally Section 6.6 concludes this chapter.

6.2 Knowledge transfer

6.2.1 Parameter initialisation using pretrained nets

Using Kazantsev *et al.*'s [23] TomoPhantom software it is possible to create synthetic tomograms, accompanied with their ground truth segmentation map. Specifically, the synthetic tomograms are generated using their pre-calculated segmentation map, also know as the phantom of the synthetic tomograms. It is then possible to generate various reconstructions that have different Signal-to-Noise Ratio (SNR) values, and can therefore be used as either input or ground truth when training denoising networks. This sets the synthetic tomograms as exceptional candidates for training deep learning networks, making them a potentially useful tool for increasing the overall accuracy when inferring on real-world data, reducing the amount of annotated real-world data needed and reducing the overall time required for training. We explore this potential in the rest of this chapter.

Generally there are two different ways that synthetic tomograms can be used in deep learning applications. The first approach is to train a deep network using solely synthetic data, or a mixture of real-world and synthetic data. This method of "doping" the training sample pool has been successfully be used in other works [448, 449], however it overly depends on the fact that the synthetic and real-world data have to be similar enough in order for the network to achieve a reasonably good accuracy, especially when training solely with synthetic data. This means that the main element of success of this approach relies in the quality of the synthetic data and their similarity to the real-world data. This approach may prove to be very effective in problems of computer vision where generating really accurate synthetic data is feasible and worthy of the potentially high number of

hours required to generate them. Especially, the last part is true when future imaging datasets are expected to be similar, and therefore the prospect of retraining the network is small.

Nevertheless, there are applications where the nature of the future datasets changes frequently and the proposed networks may need to be retrained multiple times, each time for a new variety of data. In such cases, a second approach of using synthetic data may be employed. In this method, an instance of a network is *initially* trained using *solely* synthetic data, then the training is continued using a small amount of real-world data. The instance of the network trained only with synthetic data is referred to in the literature [308] as a "pretrained" net, or "pretrained" map. This "pretrained" net, can be viewed as an "initialiser" of the parameters for each of the layers of the final "real-world" network. This means that starting state of the "real-world" network parameters (which is meant for real-world applications), is substantially closer to the final state of its parameters, where the loss function converges to a local minima. In other words, there is a knowledge which is transferred before training with real-world data. Naturally, for this to occur synthetic data must resemble the real-world data, and the higher the resemblance, the closest these starting and final parameter states will be. Nonetheless, with this approach potential dissimilarities between the synthetic and real-world data are not detrimental to accuracy of the final network the same way as when training with a mixture of real-world and synthetic data. This is due to the "correction" phase, where after the parameter initialisation of a network it is trained solely with real-world data to assure that it's accurate upon them.

Furthermore, there is another advantage through the utilisation of synthetic data using pretrained nets compared to the joint training with real-world data, which is the reduction of the required time for training. This is achieved by reusing pretrained nets and only fine-tuning/training them to different real-world datasets using a small amount of real-world data. This way, training is split into two parts, the first one is the more time-consuming creation of the pretrained net and the second is the less time consuming fine-tuning using real-world data.

While the fine-tuning must be repeated each time a network has to be deployed on a new variety real-world data, the recreation of the pretrained nets may not be necessary. This is because the synthetic data used for generating the pretrained nets may be general enough to describe multiple varieties of real-world datasets. Training/fine-tuning with real-world data may require the same amount of time irregardless of the quantity of real-world data used. It is possible by using the same small amount of real-world training data repeatedly for multiple consecutive epochs. However, as is commonly known in machine learning applications, this would result in overfitting after a few epochs and the accuracy would get further decrease in the following epochs after that point. For this reason validation is commonly used to detect this overfitting point and trigger an early termination, which reduces the amount of time spent for fine-tuning. This is the underlying reason why fine-tuning with less real-world annotated data requires less time. While the use of less real-world training data is also possible in the case of joint training with synthetic data, since there are not two different training sessions (one with synthetic and one with real-world data), the overall training with a mixture of real-world and synthetic data has to occur every time a network has to be deployed on a new variety of the real-world tomograms.

Regarding the time-series of tomograms described in chapter 3, new varieties real-world data are collected in each data collection because of that, we propose a knowledge transfer scheme for our Stacked-DenseUSeg using synthetic data created using Kazantsev *et al.*'s [23] TomoPhantom software and then use pretrained nets as a parameter initialisation technique. By making use of this scheme we are potentially aiming not only for the increase of the overall accuracy and the reduction of the real-world annotated data that are required, but also the reduction of the overall time required so that our approach can be applied. The last is true for 2 reasons. The first is that less annotated real-world data are used, which reduces the amount manual labour needed for annotation. The second reason is through the potential reuse of pretrained nets multiple time and the quick fine-tuning process with real-world data because it is not useful to be prolonged due to overfitting. In the next section, we will describe the synthetic data that are used during the experiments.

6.2.2 Synthetic phantom used for knowledge transfer

For the generation of the pretrained nets, two different highly-sampled synthetic tomograms are constructed using Kazantsev *et al.*'s [23] TomoPhantom software, both based on the same phantom. As described in Section 3.2.3, they are *SynthHS3* and *SynthHS4*. *SynthHS4* apart from some Poisson noise it does not contain any additional artifacts usually found in real-world tomograms. *SynthHS3* however is a synthetic tomogram, which contains Poisson noise, but it also some simulated rotating artifacts (ring artifacts) that are usually present in real-world tomograms. These artifacts appear in real-world tomograms and they are a byproduct of the dark and flat field correction applied to the real-world tomograms, as it is described in Section 3.1. Both synthetic tomograms are presented in the Figures 6.1, 6.2, 6.3.

These synthetic tomograms are constructed by TomoPhantom [23], using an *imaging* phantom. This phantom is essentially a volume modelled by the user, where each of its voxels contain an absorption index the virtual material should. The common way to construct such a phantom is generate via code a predetermined inner structure, which can be viewed as a segmentation map. This early stage of the phantom's voxels contains labels corresponding to a material that occupies their space. Then based on these labels, the labels are swapped for the corresponding absorption rates of the each label, which correspond to the absorption rates of the respective materials. The approach with which the TomoPhantom allows the user to design the aforementioned inner structure of the phantom, is to let them purposefully place virtual objects in a virtual stage, which is the area that the phantom depicts. These virtual objects can be any from a predetermined list of the geometrical shapes, such as cylinders, ellipsoids, cuboids, gaussians, paraboloids, etc. Additionally using constructive solid geometry [450, 451] (addition, subtraction, union or intersection of virtual solid objects that partially overlap), it is possible to construct more complicated shapes.

Based on this fact, we created a phantom that closely resembles the real-world data depicted in Figures 3.4(**a**-**b**) and 6.1. The phantom we used for knowledge transfer experiments is similar in structure to the phantom responsible for the synthetic tomograms of

Figure 3.4(**c-d**), with a crucial difference. Our updated phantom (Figures 6.1(**e**), 6.2(**e**), 6.3(**e**)) is more faithful in regards to the class-balance with our real-world tomogram (Figures 3.4(**a-b**) and 6.1). This is achieved by increasing the number of virtual objects that belong to classes with Labels 2 and 3, which correspond to the magnesium deposits and the air pockets.

To construct the phantom, we used 5000 randomly placed, sized and oriented ellipsoids for each of the two classes. Naturally there are some restrictions regarding where these ellipsoids are placed, what is their size and what is their orientation. For the class of label 2 specifically, the virtual objects have to be bounded within a virtual cylinder of label 1, which a diameter equal to 45% of the phantom's width/length (the phantom is a cube volume and so it has the same width, length and depth). This is in an attempt to keep the proportions of the virtual cylinder similar to the aluminium bin imaged in the real-world tomogram. The 3 diameters of the virtual ellipsoids of label 2 are can randomly fluctuate from 0.2% to 2.2% of phantom's width/length and the ellipsoids are free to be oriented at any direction. Similarly the 0.2% to 2.2% are elected to roughly approximate the size of magnesium deposits in the real-world tomogram

On the other hand, the virtual ellipsoids of the class with label 3 have their x and $z-axis$ diameters exactly at 0.5% of the phantom's width/length and only the ellipsoids $y-axis$ diameter is allowed to randomly fluctuate from 0.2% to 2.2% of the phantom's width/length. Additionally their orientation is not altered, which means that the axis' directions of their local coordinate system are the same as the ones of the phantom. This is in order to imitate the air pocket behaviour in the real-world data, where they are larger along the $y-axis$. Percentages are again elected to keep the scales of air pockets roughly similar to the ones present in the real-world tomogram. The reason we developed such detailed synthetic tomograms, for the prospect of using them for knowledge transfer, is because this high level of detail benefits the tasks of segmentation and denoising. Additionally, by introducing such a large number of virtual objects that belong in the class with labels 2 and 3, it allows for potential accidental overlapping which diversifies the shapes present in the tomograms. As can be seen in Figure 6.3, some of the resulting

visual objects that belong to the class of label 2 are not simple ellipsoids. This fact allows the pretrained nets to potentially attain more generalised knowledge regarding the classes of label 2 and 3, that are difficult to denoise or segment in the real-world data. This in turn may improve the overall accuracy after knowledge transfer has been performed.

The way which TomoPhantom allows the user to model their desired phantom is by reading a user-made file that contains a list of all the virtual objects that have to be placed, and their desired attributes. However, in cases where the phantom of the tomograms (like the one presented in the Figures 6.1, 6.2, 6.3) contains a large amount of virtual objects that have to be placed, an alternative method of producing this file is via code. Additionally apart from allowing the user to place only simple geometrical shapes as virtual objects, another limitation of TomoPhantom is that size of the output tomograms can be larger than $1280 \times 1280 \times 1280$ in voxel-size. This is because of the software not being optimised to handle larger data sizes. Nevertheless, in order to overcome this limitation and create synthetic tomograms which are close to the voxel resolution of the real-world micro X-ray tomograms, the upscaling-interpolation functionalities of the PyTorch deep learning framework [442] are utilised. The use of deep learning frameworks is also preferable, because they are coded for GPU parallelisation which allows the task of upscaling to happen relatively quickly.

6.2.3 Synthetic tomograms used for knowledge transfer

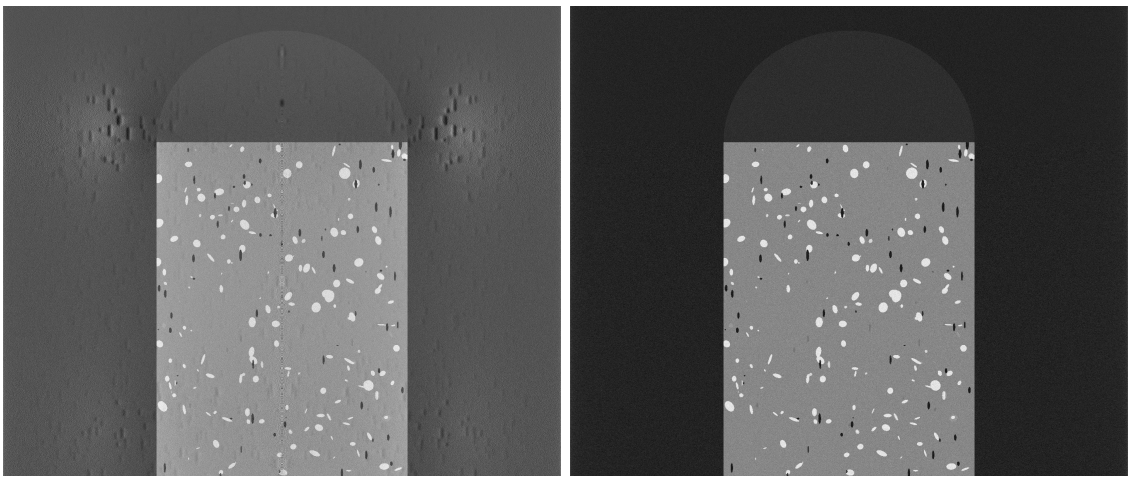
Once the synthetic phantom is constructed, it is saved as a segmentation map with each voxel containing a class label. The TomoPhantom software then continues with the creation of the projections/sinograms using this phantom. Firstly, the labels of the classes in the phantom are replaced with the corresponding absorption rates, which depend on the material that its class represents. The way with which the absorption rates can be acquired is either by using their nominal values based on literature, or by using a real-world dataset as reference. For our synthetic tomograms we used the fully-sampled real-world tomogram as a reference, and since there are manual annotations for this tomogram it was generally easy to acquire the average abortion rates for each of the classes.

In general, if the user has even a small portion of annotated real-world data, these can be used to extract the average absorption rates. After this point, the phantom that now contains the absorption rates is processed with the use of Astra toolbox [440]. Astra toolbox has utilities which effectively simulate the processes that the real-world tomogram go through during their capture. After this simulated capture the result is a collection of several projections. Similar to the real-world tomograms used for training in chapter 5, we create highly-sampled synthetic tomograms which means that the synthetic tomograms contain 3601 projections. Following that, in a process similar to the Savu Python package [438] these projections are handled again by the Astra toolbox [440] and with specific reconstruction algorithms two distinctive reconstructions are attained. The first using a small number of the provided projections is a reconstruction that simulates the collection of an undersampled tomogram, while the second using all the available projections provides a reconstruction that simulates a highly-sampled tomogram. The first of these reconstructions are seen in Figures 6.1 (**a-b**), 6.2 (**a-b**), 6.3 (**a-b**), while the second ones in Figures 6.1 (**c-d**), 6.2 (**c-d**), 6.3 (**c-d**).

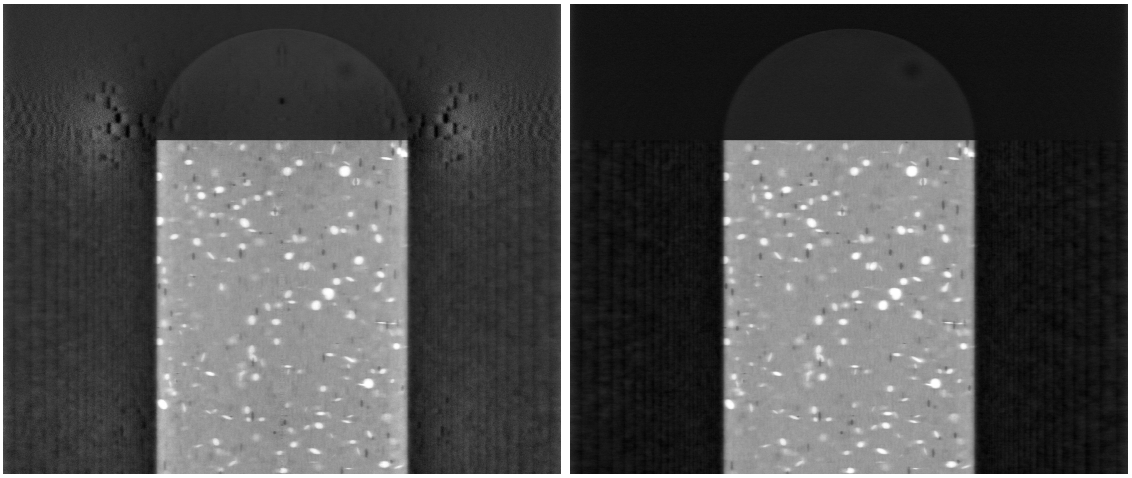
As it was mentioned earlier, using the same phantom there are two different synthetic tomograms that are created, and for each there are two reconstructions synthesised. As it is displayed in Figure 2.14 of Section 2.3.1, TomoPhantom [23] is able to generate synthetic tomograms with simulated Poisson noise and artifacts like rotating, zinger and ray artifacts. However, in attempt to create tomograms that better resemble the real-world tomograms, one of the synthetic tomogram we use contains Poisson noise, while the other contains both Poisson noise and simulated rotating artifacts. The reason we used two synthetic tomograms is because we wanted to evaluate the "quality" of the pretrained nets produced using synthetic tomograms with different levels of resemblance to the real-world tomograms. Our goal is to determine if the addition of rotating artifacts in the second synthetic tomogram would result in the generation of a better "quality" pretrained net. It is important to mention here that with the term better "quality" pretrained net we signify a pretrained net which lead to networks with high denoising and segmentation accuracy on real-world data. This is of course after the use of this pretrained net for parameter

initialisation and the completion of a fine-tuning session with real-world annotated data. Additionally, this better "quality" pretrained net may enable the reduction of real-world annotated data used during the fine-tuning session without though this fact affecting the accuracy of the resulting networks.

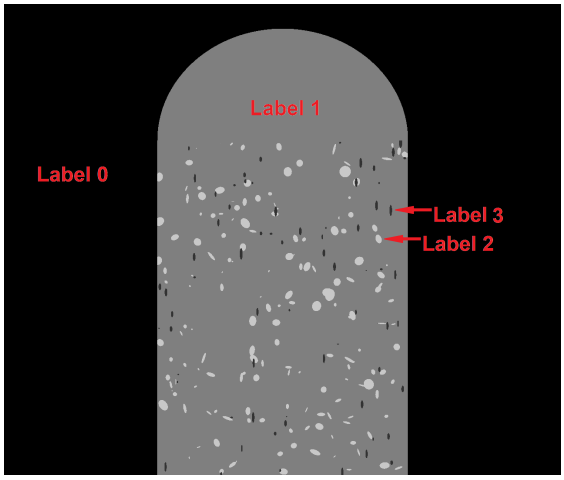
The reason these rotating artifacts appear in real-world data is due to the dark and flat field corrections. As is described in Section 3.1, during the capture of real-world tomograms several projections are collected either with the detector looking directly to the X-ray source or the detector being blocked by a lid. These bright and dark projections regularise the pixel values, and correct for problems in the experimental setup. However, a usual byproduct of that of this process is the addition of rotating artifacts (ring artifacts) in the final reconstruction of the tomogram. By including such rotating artifacts in the synthetic tomograms, it is possible during the training of the pretrained nets to encapsulate essential information on how to ignore such artifacts for the tasks of denoising and segmentation. Transfer of this knowledge to the networks that would infer on real-world data can potentially be highly significant, as the networks are ready to anticipate such artifacts and their accuracy would not be affected by their appearance.



(a) *FBP recon. with rotating artifacts* **(b)** *FBP recon. without rotating artifacts*

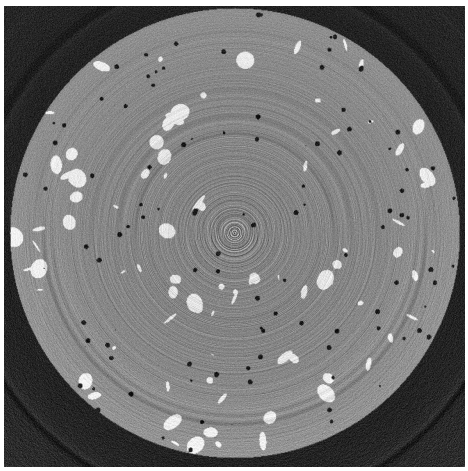


(c) *CGLS recon. with rotating artifacts* **(d)** *CGLS recon. without rotating artifacts*

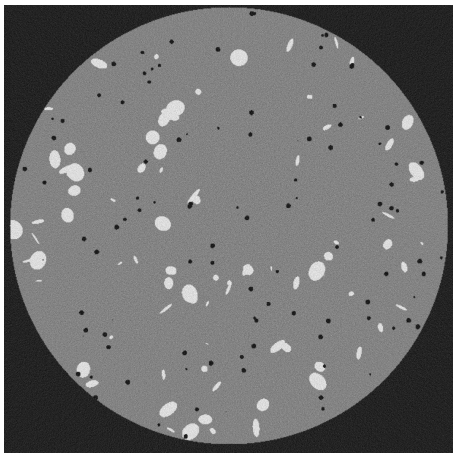


(e) *Annotations*

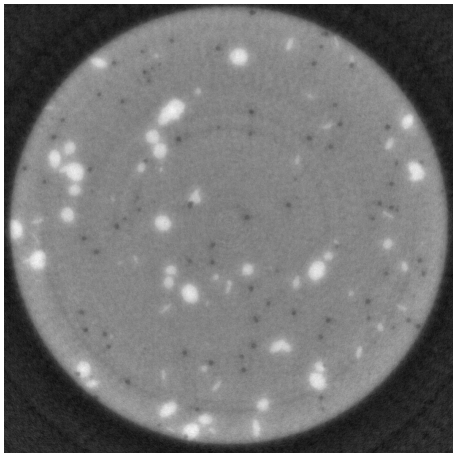
Figure 6.1: *Cross section of the SynthHS3 and SynthHS4 tomogram reconstructions and annotations used during the knowledge transfer experiments. The cross sections are cropped and centered around the pin the simulates the aluminium pin of the real-world tomograms. (**a**), (**b**) FBP reconstruction of SynthHS3 and SynthHS4 from 3601 projections respectively with [430] and without [432] rotating artifacts, (**c**), (**d**) CGLS reconstruction of SynthHS3 and SynthHS4 from 91 projections respectively with [429] and without [431] rotating artifacts, (**e**) the common annotation/phantom of SynthHS3 and SynthHS4 [452]. Similar to Figure 5.7, Label 0 in (**e**) refers to the air outside the base material, Label 1 in (**e**) refers to the base material, Label 2 are the magnesium deposits and Label 3 are the air pockets.*



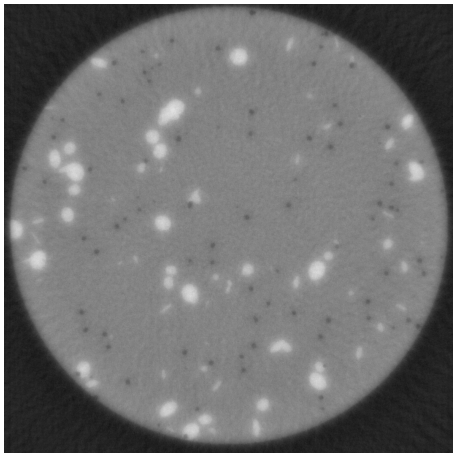
(a) *FBP reconstruction with rotating artifacts*



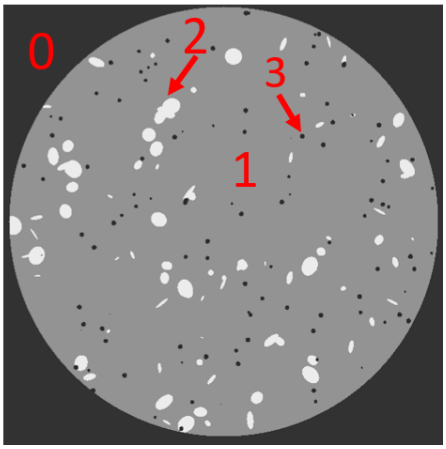
(b) *FBP reconstruction without rotating artifacts*



(c) *CGLS reconstruction with rotating artifacts*

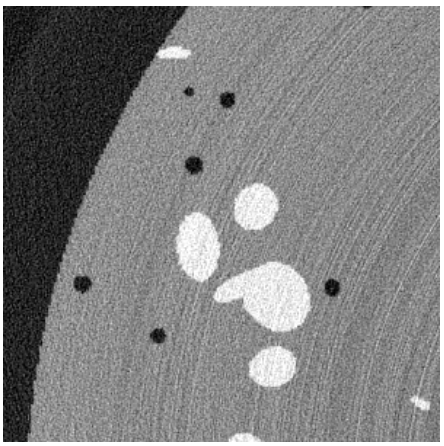


(d) *CGLS reconstruction without rotating artifacts*

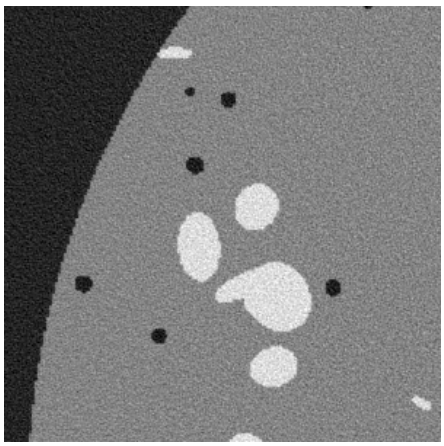


(e) *Annotations*

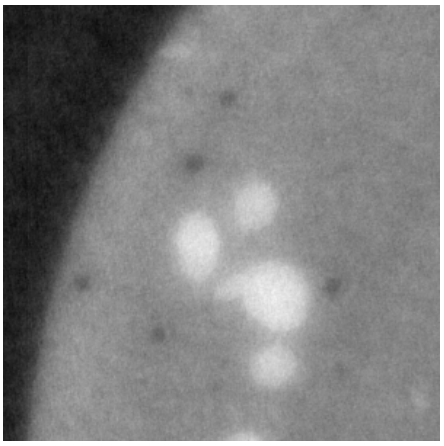
Figure 6.2: *Cross section of SynthHS3 and SynthHS4 tomogram reconstructions and annotations used during the knowledge transfer experiments. The cross sections are cropped and centered around the metallic pin of Label 1. (**a**), (**b**) FBP reconstruction of SynthHS3 and SynthHS4 from 3601 projections respectively with [430] and without [432] rotating artifacts, (**c**), (**d**) CGLS reconstruction of SynthHS3 and SynthHS4 from 91 projections respectively with [429] and without [431] rotating artifacts, (**e**) the common annotation/phantom of SynthHS3 and SynthHS4 [452]. Similar to Figure 5.7, Label 0 in (**e**) refers to the air outside the base material, Label 1 in (**e**) refers to the base material, Label 2 are the magnesium deposits and Label 3 are the air pockets.*



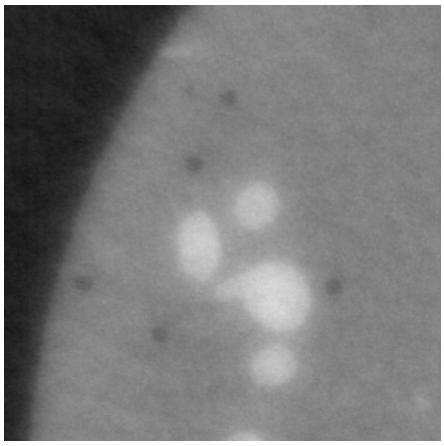
(a) *FBP reconstruction with rotating artifacts*



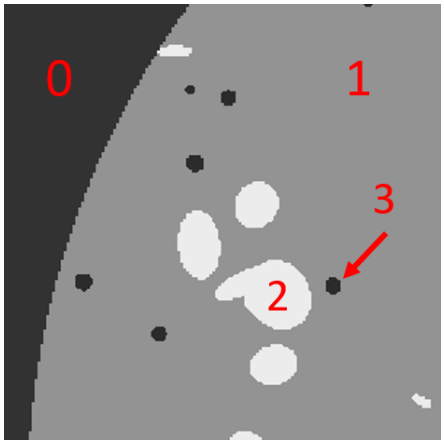
(b) *FBP reconstruction without rotating artifacts*



(c) *CGLS reconstruction with rotating artifacts*



(d) *CGLS reconstruction without rotating artifacts*



(e) *Annotations*

Figure 6.3: *Zoomed version of Figure 6.2, depicted cross section of SynthHS3 and SynthHS4 tomogram reconstructions and annotations used during the knowledge transfer experiments. (**a**), (**b**) zoomed version of 6.2(**a**) and (**b**) showing a FBP reconstruction of SynthHS3 and SynthHS4 from 3601 projections respectively with [430] and without [432] rotating artifacts, (**c**), (**d**), zoomed version of 6.2(**c**) and (**d**) showing a CGLS reconstruction of SynthHS3 and SynthHS4 from 91 projections respectively with [429] and without [431] rotating artifacts, (**e**) zoomed version of 6.2(**e**) of the common annotation/phantom of of SynthHS3 and SynthHS4 [452]. Similar to Figure 5.7, Label 0 in (**e**) refers to the air outside the base material, Label 1 in (**e**) refers to the base material (water droplet on top and aluminium pin), Label 2 are the magnesium deposits and Label 3 are the air pockets.*

6.3 Training using knowledge transfer

6.3.1 Representative real-world and synthetic tomograms

Continuing the work that is presented in chapter 5, we are here going to use a single, real-world, highly-sampled tomogram (*RealHS1* as presented in Section 3.2.2) to train our networks *after* the use of knowledge transfer. This tomogram is produced at the Diamond Light Source I13-2 beamline [416], and is as used in [24]. It depicts a droplet of salt water on top of a 500 micron aluminium pin with magnesium deposits. It is captured at the end of a time-series of undersampled tomograms that forms a 4D dataset, itself part of a study to measure the corrosion of the metallic pin by the salt-water droplet over time [418]. The size of the tomograms captured from Diamond Light Source are $2160 \times 2560 \times 2560$ in voxel resolution after their reconstruction, which, provided that it is fully annotated, can provide sufficient training samples for application on deep learning approaches.

Based on this fact, we also designed the synthetic tomograms to have same the voxel resolution. This ensures that the scale between the real-world and synthetic tomograms remains the same, which facilitates the knowledge transfer. Additionally by constructing synthetic tomograms of this size, it is possible after cropping to attain the necessary amount of training data for the creation of a pretrained net. Provided that there is a sufficient amount of annotated synthetic training samples, the resulting pretrained net may offer better generalisation for the tasks of denoising and segmentation in networks that infer on real-world data. This generalisation in turn elevates the accuracy of the networks and also allows them to train with a limited amount annotated real-world data, which would otherwise prevent the development of highly accurate networks. Finally, and importantly, these artificial datasets have a literal "ground truth" segmentation map associated with them, produced for free, rather than a "golden standard" approximation provided by careful manual annotation of real-world data.

Both synthetic and real-world, highly-sampled tomograms, used in Section 6.4 are captured with a high number of projections, namely 3601 projections. This results in high-

quality reconstructions and for the real-world tomogram, it allows the manual annotations to be very accurate approximations of the ground truth, but not the ground truth as with the synthetic tomograms. For the reasons explained in Section 5.4.1, the highly-sampled real-world tomogram can be considered representative to the time-series' tomograms (4D dataset), which justifies their use for training segmentation or denoising networks. Furthermore, the synthetic tomograms depicted in Figures 6.1, 6.2, 6.3 are also produced using 3601 projections in order to allow the two different reconstructions to be produced for each of them. In addition, while they can not be considered representative of the 4D dataset's tomograms like the real-world highly-sampled tomograms, they are constructed it in an effort to *resemble* these tomograms and in turn allow for a greater transfer of knowledge between the pretrained nets and the networks meant to infer the real-world data.

Due to this effort of creating synthetic tomograms that are as faithful as possible to the real-world tomograms, there is an argument against the practicality of using them for knowledge transfer, since new networks may have to be trained for new varieties of 4D datasets. However, based on the available geometrical virtual objects that can simulated by TomoPhantom and the use of clever coding in order to tackle the high numbers of simulated virtual objects, it is not very complicated or time-consuming to generate synthetic tomograms that attempt to be faithful for materials data. In some areas, such as manufacturing, detailed plans of the components to be scanned already exist and can be used to create the phantoms reasonably easily.

6.3.2 Tomogram reconstructions used for training

Namely, there are three highly-sampled tomograms that are used in the following Section 6.4. Two of them are the synthetically constructed tomograms for the generation of pretrained nets (*SynthHS3* and *SynthHS4* as presented in Section 3.2.3) and the other one is a real-world tomogram used in the later fine-tuning/training (*RealHS1* as presented in Section 3.2.2). The real-world tomogram is reconstructed using the Savu Python package [438] and the synthetic tomograms are created by Kazantsev *et al.*'s [23] TomoPhantom

software while their reconstructions are produced using the Astra toolbox [440]. For each of the tomograms, two reconstructions are produced. The first is made using the Filter Back Projection method (FBP) [409] algorithm and all 3601 available projections. The resulting high-quality reconstructions [425, 430, 432] provides the ground truth used in the denoising experiments and in the denoising part of Stacked-DenseUSeg. The second reconstruction is made with the Conjugate Gradient Least-squares method (CGLS) [413] reconstruction algorithm [413] and uses only 91 projections ($1^{st}, 41^{st}, \cdots, 3601^{st}$ projections) ignoring some of the intermediate projections. This low-quality reconstruction [426, 429, 431], which is provided as input during training has the same visual quality as the reconstructions from the undersampled tomograms in the 4D datasets, when only a small number of projections are used.

The reason two different algorithms are used for the two reconstructions is because the iterative CGLS method improves the reconstruction quality when applied to undersampled tomograms, while it offers worse results that the simpler FBP method when applied to highly-sampled tomograms. Figures 5.7, 6.2 and 6.3 provides cross sections of the two reconstructions for the real-world and synthetic tomograms, as well as their annotations, which depict the classes that the network aims to segment. The annotations [424] of the real-world tomogram were obtained using Luengo *et al.*'s SuRVoS [306] software tool using the high-quality reconstruction [425].

The synthetic tomograms' annotations [452], are the same as the phantom referenced in Section 6.2.2. They are are created before the synthetic tomograms by TomoPhantom based on a file that contains the list of virtual objects that have to be simulated. This file/list can be created directly by the user or for simplicity in cases where there is a large number of virtual objects that have to be simulated via code. Using these annotations/phantom the respective tomogram projections and then reconstructions are produced.

In contrast to the real-world annotations, the synthetic annotations are not human estimations of the ground truth, rather they are the literal ground truth. This is very significant since due to imaging artifacts, present even in highly-sampled tomograms, human anno-

tators may unintentionally introduce bias/errors to the data that will later be used for training, which in turn may reduce the accuracy of the final segmentation [446]. Synthetic datasets lack this potential bias, and by transferring knowledge using networks trained using them to networks that would infer on real-world data, the latter may offer better predictions in difficult cases and potentially be more accurate than human annotators.

Two synthetic tomograms are used during experiments: one more realistic, with simulated rotating artifacts (Figures 6.1 (**a**)(**c**), 6.2 (**a**)(**c**), 6.3 (**a**)(**c**)) that are also present in real-world data, and another less realistic tomogram, without such artifacts (Figures 6.1 (**b**)(**d**), 6.2 (**b**)(**d**), 6.3 (**b**)(**d**)). This effectively gives us two levels of physical simulation quality to compare. For both tomograms, in a similar way to the real-world tomogram, there are two subsequent reconstructions: a high SNR reconstruction (using the FBP method and all projections) that provides the ground truth used in denoising experiments and the denoising part of Stacked-DenseUSeg (Figures 6.1 (**a-b**), 6.2 (**a-b**), 6.3 (**a-b**)), and a low SNR reconstruction (using the CGLS method and 91 projections) used as input during training, since it has the same visual quality as the reconstructions of real-world undersampled tomograms (Figures 6.1 (**c-d**), 6.2 (**c-d**), 6.3 (**c-d**)).

There are four classes to be segmented: the air outside the base material (Label 0 in Figures 5.7(**e-f**), 6.2(**e**) and 6.3(**e**)), the base material which is the water and aluminium (Label 1 in Figures 5.7(**e-f**), 6.2(**e**) and 6.3(**e**)), the magnesium deposits within the base material (Label 2 in Figures 5.7(**e-f**), 6.2(**e**) and 6.3(**e**)) and lastly the air pockets within the base material (Label 3 in Figures 5.7(**e-f**), 6.2(**e**) and 6.3(**e**)). Since the training of the network is performed using a single tomogram at time, they are normalised (if real data; synthetic tomograms are produced pre-normalised) and split into multiple non-overlapping samples of $64 \times 64 \times 64$. Additionally, before entering the network each of the samples is either randomly rotated between 90 180 270degrees, or it is horizontally or vertically mirrored. As explained in Section 6.2.3, the segmentation instances/objects present in the synthetic tomograms are set so that they will have approximately the same absorption rates and therefore voxel intensities as their real-world counterparts. Furthermore, their placement and size is elected in a manner that emulates the segmentation classes in the

real-world tomogram (more in Section 6.2.2).

6.3.3 Training, validation and testing

As is mentioned in Section 5.4.2 from these non-overlapping $64 \times 64 \times 64$ samples, 70% are randomly chosen for training, 10% for validation and 20% for testing. This results in, for the real-world tomogram, 3723 samples used for training, 532 used for validation and 1065 used for testing. For the synthetic tomograms, 4646 samples are used for training, 663 used for validation and 1329 used for testing. The samples are cropped from the area of the tomograms that contain most of the desired classes to segment, in order to avoid high class imbalance, since the rest of the tomogram volume belongs to the background class of Air (Label 0). By design, the synthetic datasets, similar to the real-world data [418], are also concentrated in a smaller area of the tomogram. This is in order to ensure that the segmentation instances in the synthetic tomograms to have the same size/scale as in the real-world data. This smaller area in the synthetic datasets is the one that simulates the aluminium pin and its size was elected in order to approximate the size of the real-world pin, relative to the size of the dataset (More in Section 6.2.2). While the location of this area in synthetic data is known because the data are artificially constructed, for real-world datasets it can be determined by thresholding the voxel values and then using morphological operations like region filling. Typically, the surrounding background of an imaged physical sample in an X-ray tomography is air, which appear darker than any other imaged materials in a tomogram's reconstruction. Therefore with use of thresholding and morphological operations, regions of interest can be easily detected. The resulting networks during inference are applied only in these specific areas, allowing for faster and easier inference. Additionally, during inference the samples are chosen to be overlapping, in order to negate potential border artifacts, while during training the samples are non-overlapping samples in order for clear separation between training, validation and testing samples.

Our knowledge transfer scheme is evaluated using different amounts of annotated real-world data. During these evaluations, only the amount of data used for training is de-

creased, while the amounts of data used for validation and testing remain the same. This means that the number of validation and testing samples does not fluctuate, which allows for compatible results which are not dependent on the size of annotated data used for training.

Furthermore, similar to the experiments of Section 5.5, the ones presented in following section 6.4 are repeated three times, assigning different training, validation and testing samples each time. Extra care has been taken so that each validation and testing sample is not assigned again to the validation or testing sets respectively, between the 3 iterations. This principle is similar to the 10-fold cross-validation, however in our case there is both a testing and a validation set which are "folded". Moreover, our "fold" is not rigorous in the sense that *every* available annotated sample must be used at least once for validation or testing. The reason a more rigorous approach was not attempted, is because it would require extra "folds" and since each training session/iteration usually takes numerous hours to complete, due to the 3D nature of the network, the overall time and computational resources required would be prohibiting.

6.3.4 Hyperparameters and backpropagation

Similar to Section 5.4.3, for backpropagation the stochastic gradient-based optimisation method Adam [435] is used, with mini-batches of 8. The two learning rates of Stacked-DenseUSeg are set to 10^{-5} for the segmentation part of the network and 10^{-4} for the denoising part. A weight decay [447] of 0.0005 is also used, to act as a L2 regularisation which penalises when parameters collectively get too large, which can lead to overfitting.

It is also significant to mention that during training the learning rates are adjusted based on how many epochs have pasted. In the experiments where a large number of training samples are used it is observed based on preliminary testing, that a reduction of the learning rate by a factor of 10 at the middle of the training process (50^{th} epoch) is more advantageous. This setting offers a sufficient time for the network to increase its accuracy in the first half of the training session, and in the second half to steadily converge to a local minima. However, for the experiments that use less than half of the available training

samples, specifically when using 10%, 5% and 1% of the available training samples, higher starting learning rates have to be adopted (10^{-4} for the segmentation part and and 10^{-3} for the denoised part) and different learning rate adjustment strategies. Namely, the learning rates are reduced by a factor of 10 at two different times during the training session. The first one is after the 10^{th} epoch and other after at the 40^{th} epoch. The first adjustment comes to correct the initial higher learning rates, and during this short period (1^{st} to 10^{th} epoch) the network learns at a very high pace. This is more beneficial than the network learning with a slower pace for a longer duration, since the gradients in each backpropagation are more erroneous when the amount of training data is small, and error may accumulate during this longer duration. This is also the reason the second adjustment is performed at the 40^{th} epoch and not at the 50^{th} epoch.

After the second adjustment, convergence to a higher accuracy is possible, however in most occasions the network would probably overfit (this is also confirmed later in the validation graphs in Figures 6.10 (**c**-**e**), 6.5 (**c**-**e**)). Because of this, after training, the network at the epoch which had the higher validation score is considered the final version of the network and testing is applied to it. This also signifies that the training process can be stopped prematurely if the validation score is in decline, which can reduce the time required for training. Figure 6.4 displays the learning rate schedule regarding both the cases of high amounts of training data used (Figure 6.4(**a**)) and the low amounts of training data used (Figure 6.4(**b**)).

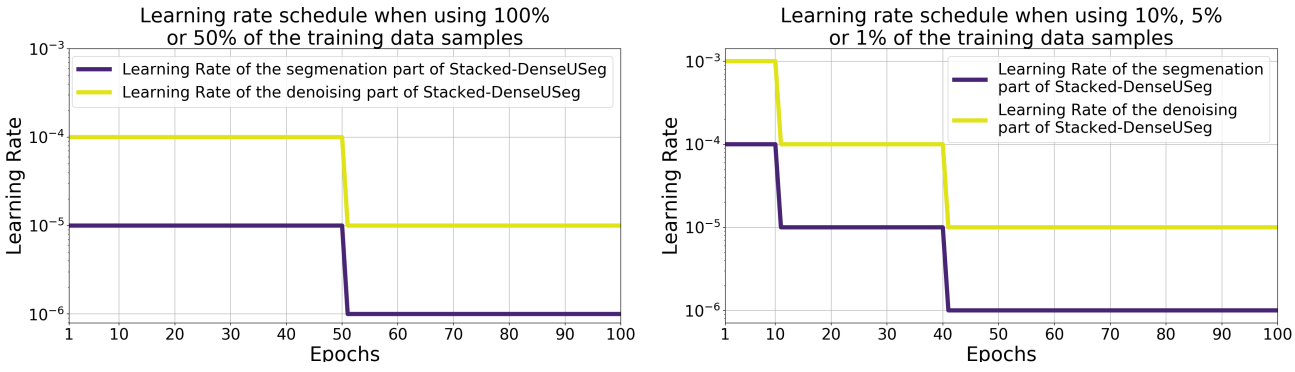


Figure 6.4: *The learning rate schedule of Stacked-DenseUSeg, both for its denoising part and its segmentation part. (**a**) The learning rate schedule when all or 50% of the training data is used for training (the later case probably for the case of knowledge transfer) and (**b**) The learning rate schedule when 10%, 5% or 1% of the training data is used for training or fine-tuning of a pretrained net to real-world applications following a knowledge transfer scheme.*

6.4 Experiments exploring use of differing amounts of annotated data

In the earlier chapter 5, we demonstrated that our approaches, DenseUSeg and Stacked-DenseUSeg have similar, if not better, results to the state-of-the-art regarding the denoise of low SNR reconstructions of undersampled tomograms, and that they outperform them in regards to the semantic segmentation of undersampled tomograms. However, despite the demonstrated success of our networks, typical to deep learning approaches they require a sufficient (sizeable) amount of annotated real-world data for successful training before they can be applied to real-world applications. This creates a challenge as such data is hard to come by, expensive to produce, and may very well be formed from biased estimations of the ground truth, as the labels are the best effort of the researcher conducting the manual annotation, which may offer segmentation with reduced accuracy [446].

Nevertheless, as discussed, through the development of software tools like the TomoPhantom by Kazantsev *et al.*'s [23] it is possible to create realistic synthetic tomograms that are able to approximate real-world tomograms. These synthetic tomograms, described in Section 6.2 are generated with their corresponding annotations, actual ground truths, automatically. Specifically their annotations (phantoms) [452] are first designed and created

by a user, and then their simulated tomographic counterparts are produced by the software [429, 430, 431, 432], similar to real-world X-ray tomograms. This is a physical-based rendering process, which, as we shall see (Figures 6.1(**a**)(**c**), 6.2(**a**)(**c**) and 6.3(**a**)(**c**)), can even include imaging artifacts that also appear in real-world CTs.

In this Section, we will present experiments using three parameter initialisation settings and a fine-tuning/training using various amounts of the available training data. Namely, there is the initialisation using a pretrained net generated by a synthetic tomogram *with* rotating artifacts, the initialisation with a pretrained net generated by a synthetic tomogram *without* rotating artifacts and finally the random initialisation which would act as baseline. Additionally, the subsequent fine-tuning/training with real-world data is tested using different amounts of training samples. The training samples are set to 100%, 50%, 10%, 5%, 1% of the initially available real-world training samples. In addition, the pretrained nets are also tested without the use of any fine-tuning/training.

Based on the above, in this section we will examine:

1. The advantages of employing different parameter initialisation settings under different amounts of training data for the task of denoising. These are regarding the intermediate *denoising* output of Stacked-DenseUSeg.
2. The advantages of employing different parameter initialisation settings under different amounts of training data for the task of semantic segmentation. These are regarding the final *segmentation* output of Stacked-DenseUSeg.

6.4.1 Experiment 1: Effects of knowledge transfer to the task of denoising

As it is demonstrated in the previous chapter 5, our proposed Stacked-DenseUSeg architecture has two distinctive outputs that are generated simultaneously when it is used for inference. The first output is a denoised version of the input, with a higher SNR as well as some its spatial information restored, using representative highly-sampled tomograms for reference. With the aid of these tomograms, our Stacked-DenseUSeg network is able to tackle the task of denoising noisy, undersampled tomogram reconstructions, in a process

that is easier that the semantic segmentation as these tomograms do not have to be annotated manually. The reason these highly-sampled tomograms can be used for this task is because they can be reconstructed either as a low SNR reconstruction similar to the ones of undersampled tomograms, or as a high SNR reconstruction using all the available projections acquired during their collection. This naturally provides both the input *and* the ground truth, that can be used in a deep learning approach respectively. Nevertheless their high SNR reconstructions, on occasion are far from providing the ideal ground truth in terms of denoising, and this is due to some unavoidable real-world limitations of X-ray tomography introducing imaging artifacts. In extreme cases the presence of imaging artifacts might be detrimental and subvolumes of the high SNR reconstruction of highly-sampled tomograms might not be suitable for use in deep learning as a spatial references for undersampled tomograms. Naturally, this reduces the available amount of training data. For this reason, the knowledge transfer from synthetic tomograms can potentially be a powerful tool towards the goal of the denoising undersampled tomograms. With their reconstructions simulated under highly-controlled conditions, it is possible for the developers of deep learning approaches to more accurately guide the network towards its desired behaviour.

Based on the earlier argument and by implementing the knowledge transfer scheme presented in Section 6.2, we applied two different pretrained nets as "initialisers" of the network's parameters. The first is constructed by initially training on a highly-sampled synthetic tomograms *with* rotating artifacts, which enhance its relation to the real-world tomograms, while the second one is constructed through training with a synthetic tomogram that *does not* contain any additional artifacts apart from Poisson noise. The effects that the different pretrained nets have are shown in the Table 6.1 and in the Figures 6.5 and 6.6. Similar to Section 5, the metrics used to evaluate the denoising experiments are the Peak Signal-to-Noise Ratio (PSNR) metric (which is a logarithmic representation of the mean square error, see Equation 4.5), and the Structural Similarity Index (SSIM) [439] (see Equation 4.6) are used to quantitatively evaluate the image restoration quality compared to the ground truth (in our case the high-quality reconstruction [425]).

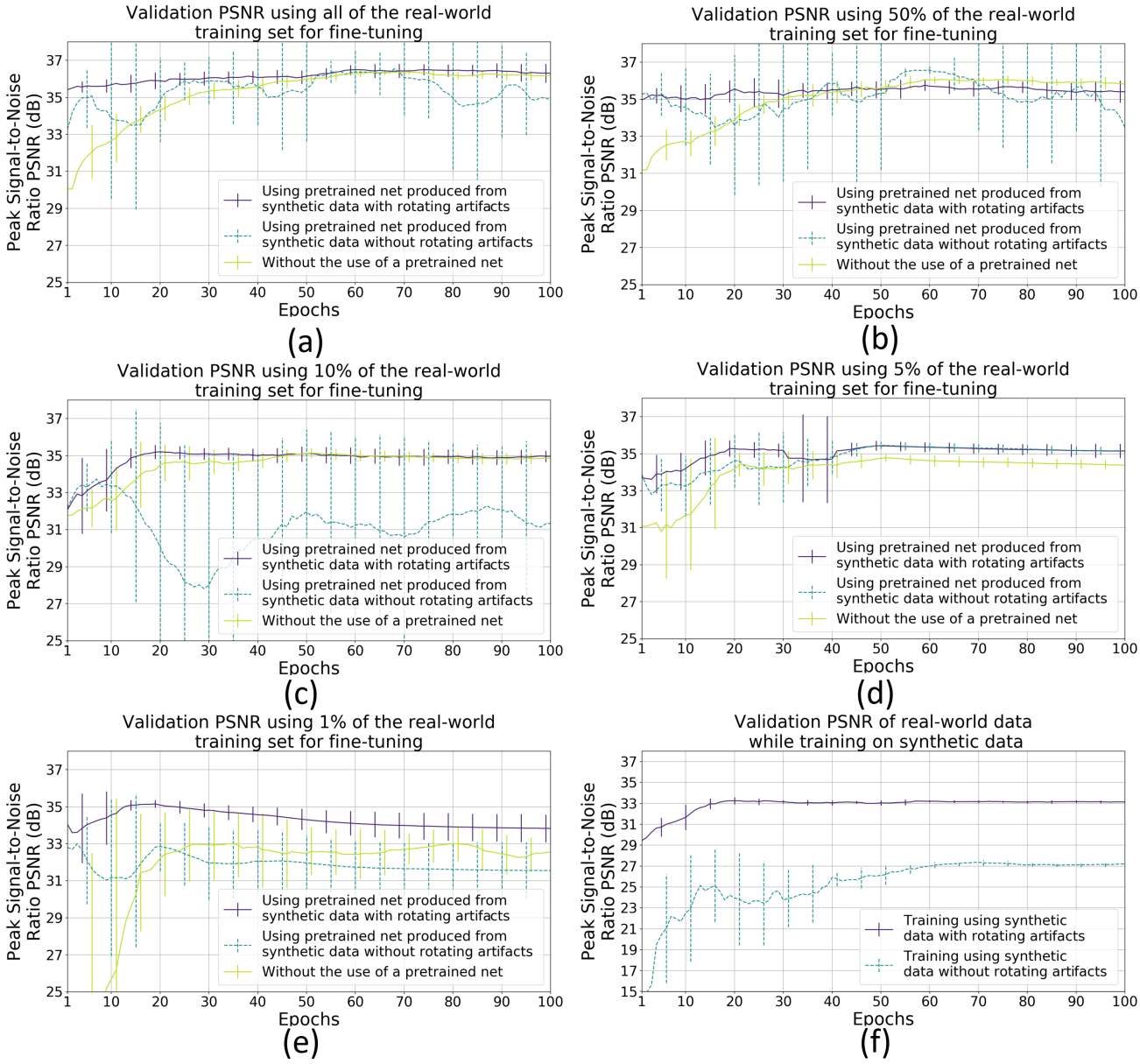


Figure 6.5: *Validation PSNR for the denoising experiments where the input is the CGLS reconstruction of the real-world tomogram (more in Section 6.3), and by initialising the parameters using pretrained nets produced from synthetic data with rotating artifacts, without rotating artifacts or without the use of a pretrained net. The network is then fine-tuned using (**a**) 100%, (**b**) 50%, (**c**) 10%, (**d**) 5%, (**e**) 1%, or (**f**) 0% of available real-world training samples . As can be seen, using the pretrained net produced from training with a synthetic tomogram with rotating artifacts offers overall higher accuracy, even when using very small amounts of real-world data for training/fine-tuning, while it is also very stable based on the error bars. All values are averages between the 3 iterations of the 3-fold validation/testing, averaged based on a 10 epoch moving average. The error bar heights refer to the standard deviation that the approaches achieved based om a (3x)10 epoch moving standard deviation.*

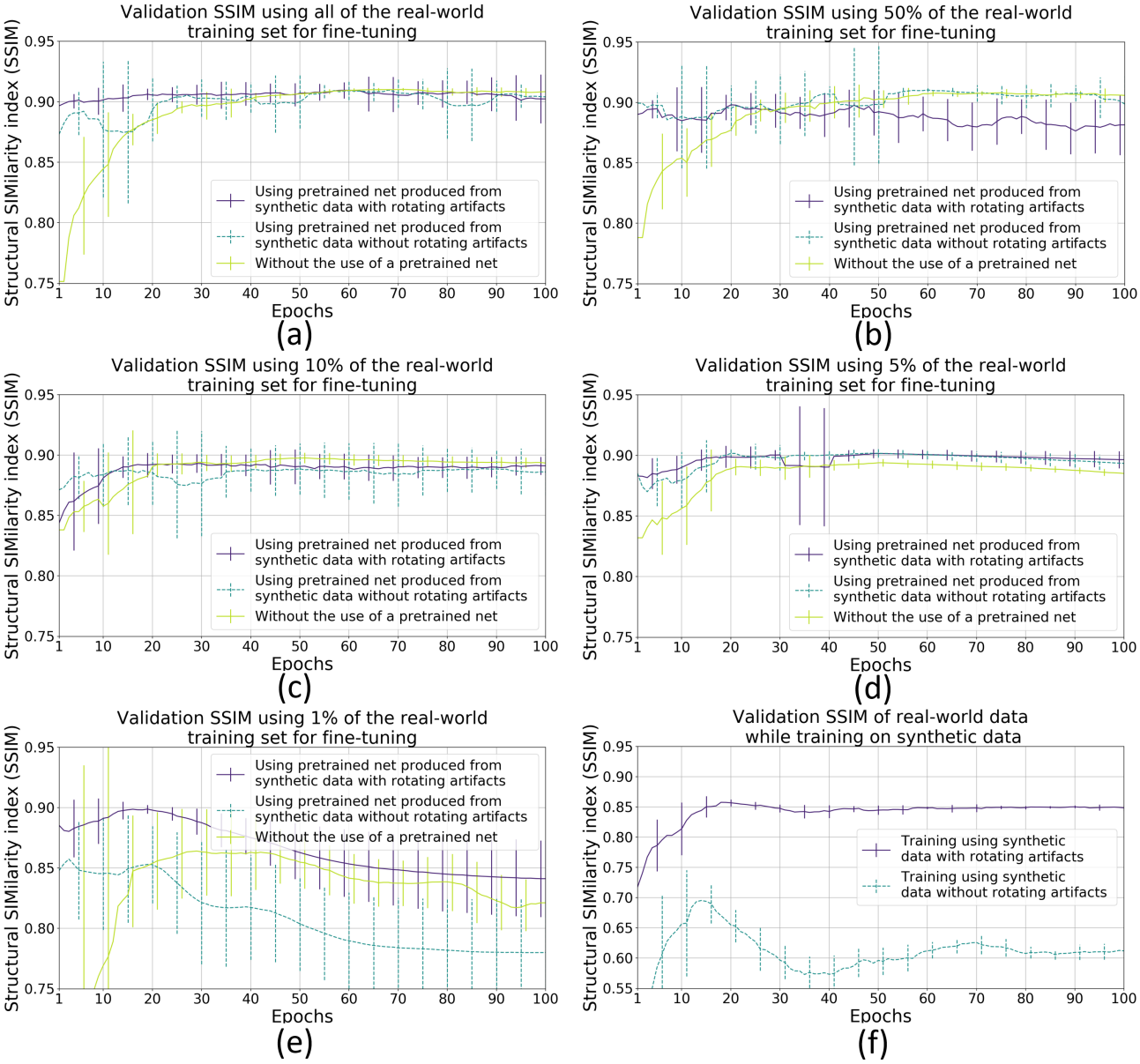


Figure 6.6: *Validation structural similarity index (SSIM) for the denoising experiments where the input is the CGLS reconstruction of the real-world tomogram (more in Section 6.3), and by initialising the parameters using pretrained nets produced from synthetic data with rotating artifacts, without rotating artifacts or without the use of a pretrained net. The network is then fine-tuned using (**a**) 100%, (**b**) 50%, (**c**) 10%, (**d**) 5%, (**e**) 1%, or (**f**) 0% of available real-world training samples. As it can be seen using the pretrained net produced from training with a synthetic tomogram with rotating artifacts offers overall higher accuracy, even in the when using very small amounts of real-world data for training/fine-tuning, while it is also very stable based on the error bars. All values are averages between the 3 iterations of the 3-fold validation/testing, also averaged based on a 10 epoch moving average. The error bar heights refer to the standard deviation that the approaches achieved based om a (3x)10 epoch moving standard deviation.*

Table 6.1: PSNR and SSIM results of the denoising output of Stacked-DenseUSeg during testing using progressively lower amounts of the real-world training samples, and by initialising the parameters using pretrained nets produced from synthetic data *with* rotating artifacts, *without* rotating artifacts or without the use of a pretrained net. While the training samples are reduced, the testing samples are not, setting the results comparable between themselves and the ones on Table 5.2. The values presented are averages between the 3 iterations of the 3-fold validation/testing, with standard deviation calculated over 3-fold validation/testing. Based on the results, it can be deduced that the absence of rotating artifacts, aid the obtaining of more accurate denoising results, when the amount of training real-world annotated data is high (100%-50%). For smaller amounts of real-world annotated data (10%, 5%) the addition of rotating artifacts offers slightly better denoising accuracy and for 1% and below it offer a more noticeable improvement in accuracy. For both metrics higher is better, while for standard deviation smaller is better.

	Pretrained net *with* rotating artifacts	Pretrained net *without* rotating artifacts	Without a pretrained net
Using 100% training dataset			
PSNR (dB)	36.34 ± 0.3	$\mathbf{36.70} \pm 0.1$	36.33 ± 0.4
SSIM	$0.905 \pm 1 \cdot 10^{-2}$	$\mathbf{0.911} \pm 9 \cdot 10^{-4}$	$0.910 \pm 2 \cdot 10^{-3}$
Using 50% training dataset			
PSNR (dB)	35.42 ± 0.4	$\mathbf{36.56} \pm 3 \cdot 10^{-2}$	$36.11 \pm 7 \cdot 10^{-2}$
SSIM	$0.875 \pm 2 \cdot 10^{-2}$	$\mathbf{0.910} \pm 10^{-3}$	$0.909 \pm 6 \cdot 10^{-4}$
Using 10% training dataset			
PSNR (dB)	$\mathbf{35.47} \pm 0.3$	34.45 ± 1	34.68 ± 0.5
SSIM	$\mathbf{0.903} \pm 4 \cdot 10^{-3}$	$0.900 \pm 4 \cdot 10^{-3}$	$0.895 \pm 4 \cdot 10^{-3}$
Using 5% training dataset			
PSNR (dB)	$\mathbf{35.47} \pm 0.3$	35.44 ± 0.3	34.65 ± 0.4
SSIM	$\mathbf{0.903} \pm 4 \cdot 10^{-3}$	$\mathbf{0.903} \pm 4 \cdot 10^{-3}$	$0.895 \pm 4 \cdot 10^{-3}$
Using 1% training dataset			
PSNR (dB)	$\mathbf{34.92} \pm 0.2$	32.98 ± 0.2	33.09 ± 1
SSIM	$\mathbf{0.895} \pm 8 \cdot 10^{-3}$	$0.871 \pm 9 \cdot 10^{-3}$	$0.868 \pm 2 \cdot 10^{-2}$
Using only the pretrained nets			
PSNR (dB)	$\mathbf{33.27} \pm 7 \cdot 10^{-3}$	$25.54 \pm 2 \cdot 10^{-2}$	-
SSIM	$\mathbf{0.848} \pm 2 \cdot 10^{-4}$	$0.569 \pm 7 \cdot 10^{-4}$	-

As we can see from the figures and the table, the use of pretrained nets increased the SNR and visual quality of the real-world tomogram reconstructions. Nevertheless, by inspecting Table 6.1 we can observe that the improvement is only slight, and is dependant on the amount of real-world training data used, and sensitive to the nature of the pretrained net utilised for parameter initialisation. It seems that for *low* amounts of real-world training data, both PSNR and SSIM [439] reach slightly higher scores with the use of the pretrained net from the synthetic tomogram *with* rotating artifacts. For the use of 1% or 0% fine-tuning this improvement becomes more noteworthy, potentially due to the synthetic data *with* supplementing features absent in the very limited real-world data.

When higher amounts of real-world training data are available for fine-tuning, the pretrained net from the synthetic tomogram *without* rotating artifacts achieves slightly higher performance. By remembering the fact that the ground truth for the network's denoised output is the high SNR reconstruction of the provided tomogram, it offers a hint as to why the pretrained net from the synthetic tomogram *without* rotating artifacts has the advantage. The probable reason behind it is that its high SNR reconstruction (see Figures 6.1(**b**)(**d**), 6.2(**b**)(**d**), 6.3(**b**)(**d**)) better represents the ground truth (since there are no additional artifacts) than the high-quality reconstruction of the synthetic tomogram *with* rotating artifacts.

In the future, it could therefore be potentially a good idea to combine the two synthetic tomograms during the training of a pretrained net. For that the low SNR reconstruction of the synthetic tomogram that has rotating artifacts would be used as input, and the high SNR reconstruction of the one that does not would be used as the ground truth for the denoising part of the network. Even so, the potential improvement would be more prevalent, when higher amount of training data are available for training/fine-tuning. This might only be an occasional use-case anyway though, since the absence of fully annotated real-world tomograms is a common phenomenon and was the main motivation for use of synthetic data in this chapter in the first place.

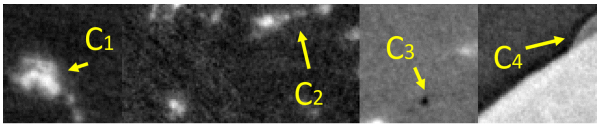
Furthermore, by observing Figure 6.5 it can be seen that initialising with the pretrained net from the synthetic tomogram *with* rotating artifacts tends to train the network in a

more stable fashion than initialising with the pretrained net from the synthetic tomogram *without* artifacts. In the graph, the moving standard deviation for the networks initialised with the pretrained net from the synthetic tomogram *without* rotating artifacts is almost an order of magnitude larger than the one initialised with the pretrained net from the synthetic tomogram *with* rotating artifacts. Nevertheless, regarding the perceptual quality, which is more accurately measured by SSIM, it seems that both have the similar behaviours.

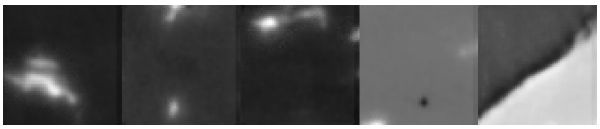
Regarding the denoising results presented in Figures 6.7, 6.8, 6.9, one of the immediate differences that can be identified regarding the performance of the pretrained net from a synthetic tomogram *with* rotating artifacts compared to the one without, is when they are applied without fine-tuning. By comparing marks $\boldsymbol{C_1}$-$\boldsymbol{C_4}$ to the $\boldsymbol{V_9}$-$\boldsymbol{V_{12}}$, it is clear that the pretrained net from a synthetic tomogram *with* rotating artifacts it is able to present a noteworthy accuracy, despite the fact that it has never trained with real-world data. Compared to the Figure 6.8(**h**), the pretrained net from a synthetic tomogram *without* rotating artifacts is nowhere close to that accuracy. Additionally, by comparing marks $\boldsymbol{U_1}$-$\boldsymbol{U_{12}}$ and $\boldsymbol{W_1}$-$\boldsymbol{W_{12}}$ to the $\boldsymbol{V_1}$-$\boldsymbol{V_8}$ marks, we can infer that either with the use of a pretrained net from a synthetic tomogram *without* rotating artifacts or without any use of a pretrained net, the resulting network struggles to be accurate in conditions where the provided training samples are very few. One the other hand, with the use of a pretrained net from a synthetic tomogram that has rotating artifacts remains to be fairly accurate even in the difficult cases shown in the marks $\boldsymbol{V_3}$, $\boldsymbol{V_7}$, $\boldsymbol{V_{11}}$.



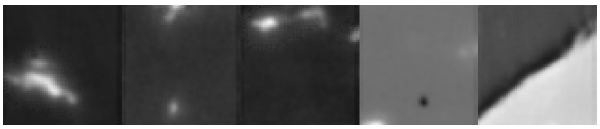
(a) *Low-quality reconstruction from CGLS and 91 projections*



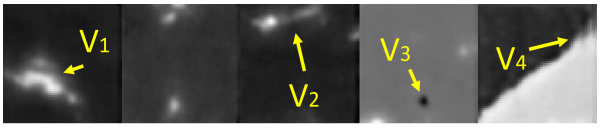
(b) *High-quality reconstruction from FBP and 3601 projections*



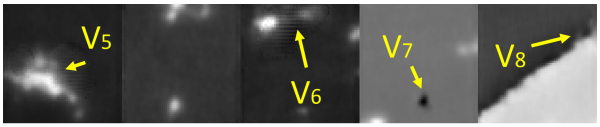
(c) *Using 100% of available training data*



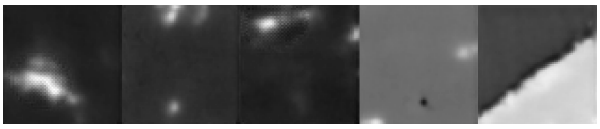
(d) *Using 50% of available training data*



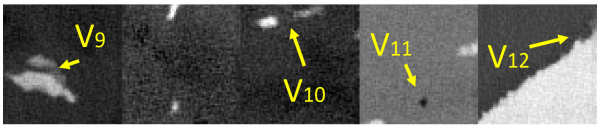
(e) *Using 10% of available training data*



(f) *Using 5% of available training data*

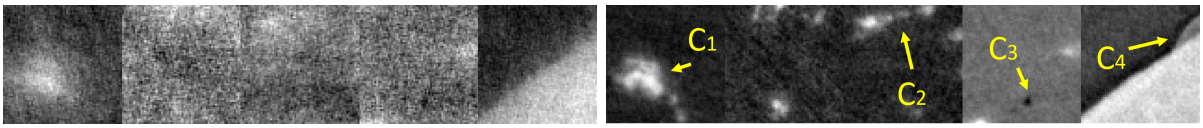


(g) *Using 1% of available training data*



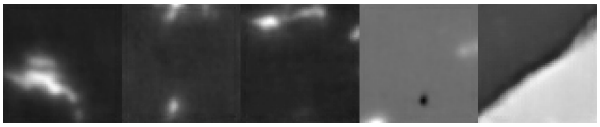
(h) *Inferring using the pretrained nets without any training/fine-tuning*

Figure 6.7: *Denoising results of Stacked-DenseUSeg with progressively less real-world training samples and using the pretrained net trained on synthetic data* ***with*** *rotating artifacts [429, 430]. (**a**) The CGLS, 91 projection reconstruction [426] that is the networks input, (**b**) the FBP, 3601 projection reconstruction [425] that is ground truth for the denoising part of Stacked-DenseUSeg, (**c**) Using 100% of training samples, (**d**) 50%, (**e**) 10%, (**f**) 5%, (**g**) 1%, (**h**) Inferring directly using the pretrained nets without any training/fine-tuning. Marks (**C_1-C_4**, **V_1-V_{12}** are described in text.*

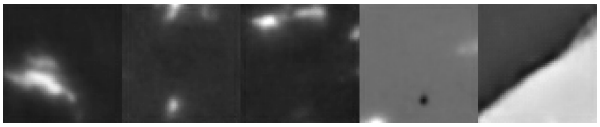


(a) *Low-quality reconstruction from CGLS and 91 projections*

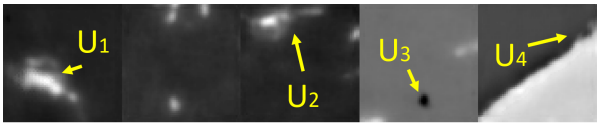
(b) *High-quality reconstruction from FBP and 3601 projections*



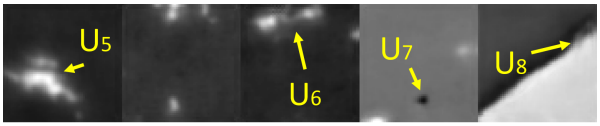
(c) *Using 100% of available training data*



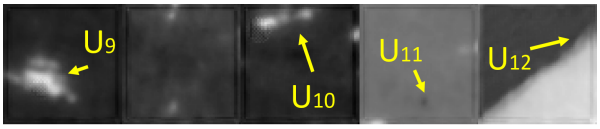
(d) *Using 50% of available training data*



(e) *Using 10% of available training data*



(f) *Using 5% of available training data*

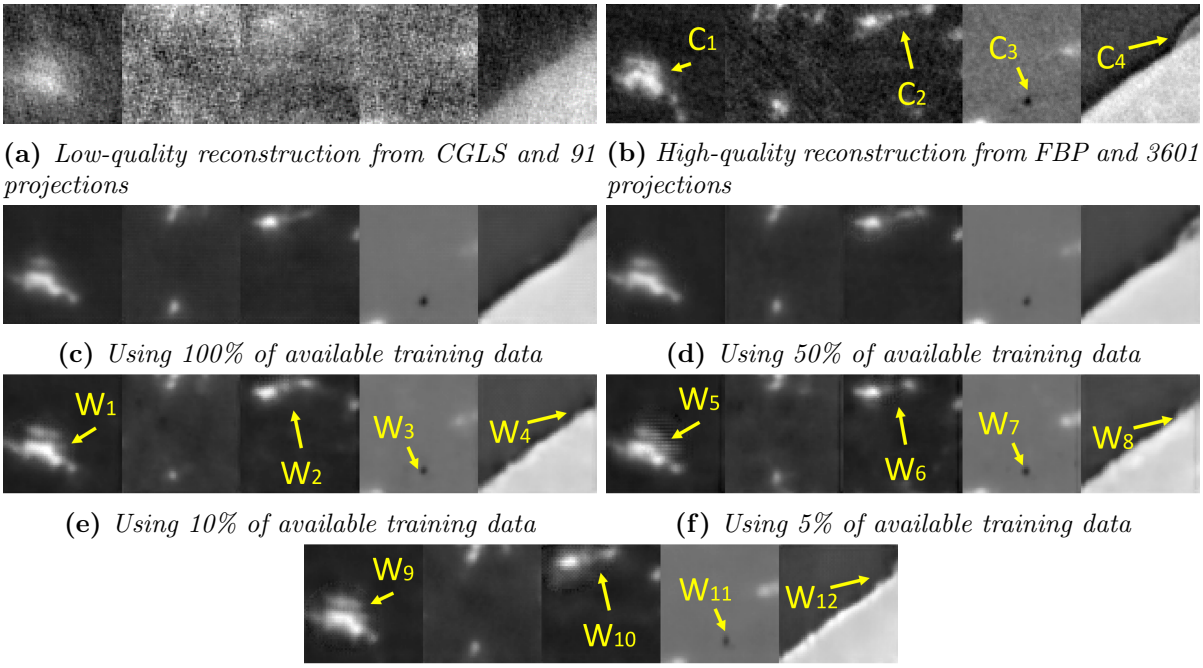


(g) *Using 1% of available training data*



(h) *Inferring using the pretrained nets without any training/fine-tuning*

Figure 6.8: *Denoising results of Stacked-DenseUSeg with progressively less real-world training samples and using the pretrained net trained on synthetic data* ***without*** *rotating artifacts [431, 432]. (**a**) The CGLS, 91 projection reconstruction [426] that is the networks input, (**b**) the FBP, 3601 projection reconstruction [425] that is ground truth for the denoising part of Stacked-DenseUSeg, (**c**) Using 100% of training samples, (**d**) 50%, (**e**) 10%, (**f**) 5%, (**g**) 1%, (**h**) Inferring directly using the pretrained nets without any training/fine-tuning. Marks* $\boldsymbol{C_1}$-$\boldsymbol{C_4}$, $\boldsymbol{U_1}$-$\boldsymbol{U_{12}}$ *are described in text.*



(a) *Low-quality reconstruction from CGLS and 91 projections*

(b) *High-quality reconstruction from FBP and 3601 projections*

(c) *Using 100% of available training data*

(d) *Using 50% of available training data*

(e) *Using 10% of available training data*

(f) *Using 5% of available training data*

(g) *Using 1% of available training data*

Figure 6.9: *Denoising results of Stacked-DenseUSeg with progressively less real-world training samples and without the use of a pretrained net. (**a**) The CGLS, 91 projection reconstruction [426] that is the networks input, (**b**) the FBP, 3601 projection reconstruction [425] that is ground truth for the denoising part of Stacked-DenseUSeg, Label 2 in yellow are the magnesium deposits and Label 3 in white are the air pockets, (**c**) Using 100% of training samples, (**d**) 50%, (**e**) 10%, (**f**) 5%, (**g**) 1%. Marks* $\boldsymbol{C_1}$-$\boldsymbol{C_4}$, $\boldsymbol{W_1}$-$\boldsymbol{W_{12}}$ *are described in text.*

6.4.2 Experiment 2: Effects of knowledge transfer to the task of semantic segmentation

Moving on to Stacked-DenseUSeg's final output, the semantic segmentation of undersampled tomograms, there are a few expected advantages of using a knowledge transfer via the parameter initialisation employed by pretrained nets trained on synthetic tomograms. One of them is that it would allow the network to perform better at generalisation in the way it infers its predictions, leading to higher accuracy.

The pretrained nets would allow for low-level information regarding the physics of X-ray tomography, which is easily simulated by synthetic analogues, to pass into the new networks and permit the use of less annotated real-world data during their training. The latter is actually highly significant as it is usually difficult to obtain large amounts of annotated data, they are expensive to produce and the resulting labels are human estimations of the ground truth which may have some bias. Specifically, it is possible due to human limitations and tendency to resolve ambiguities, to introduce bias to the training dataset which may in turn reduce the accuracy of the final segmentation [446]. Therefore, the introduction of knowledge based on synthetic data, that have genuine segmentation ground truth is highly welcomed.

By applying the aforementioned transfer of knowledge, Figure 6.10 and Table 6.2 shows that by initialising the network's parameters using a Stacked-DenseUSeg already trained on a synthetic tomogram *with* rotating artifacts, the initialised network (Stacked-DenseUSeg as well), is able to achieve a higher accuracy score in segmentation of real-world data, compared to when there is no parameter initialisation. Additionally, as it can be observed the latter network, is also able to offer accurate real-world predictions in use-cases, where only small amounts of real-world annotated samples are available, without any accuracy sacrifices as the amount of annotated data decreases. Specifically, it is only after training/fine-tuning with 1% of the real-world training samples, namely 37 $64 \times 64 \times 64$ samples (approximately $213 \times 213 \times 213$ voxels of annotated volume), that the accuracy is comparable to when all the available real-world data is used, and that includes the challenging classes with labels 2 and 3. Even in the case where no fine-tuning with real-world

data is employed, it is still able to achieve a noteworthy segmentation accuracy, which further proves the importance of rotating artifacts in synthetic data. Furthermore, it is important to point out that in the case where 100% of that real-world training samples are used, the network with the highest accuracy remains to be the one that is initialised with the pretrained net from the synthetic tomogram *with* rotating artifacts. This is true due to its highest accuracy in the difficult classes with labels 2 and 3 compared to the other initialisation settings. Finally, by observing the standard deviation in the Table 6.2 and the error bars in Figure 6.10, it can be deduced that the latter network achieves more stable predictions, compared to the other initialisation settings.

Regarding the parameter initialisation using the pretrained net derived from the synthetic tomogram *without* rotating artifacts [431, 432], it is can be observed both in the Table 6.2 and in the Figure 6.10 that its use is not as advantageous as the other pretrained net [429, 430]. This is especially true by examining the last rows of Table 6.2. When using the pretrained nets directly without fine-tuning, the performance of the pretrained net from the synthetic *without* rotating artifacts is especially poor. This is to be expected as the physical rendering of the synthetic tomograms that is trained upon is less realistic without these artifacts. However, in cases where the available amount of real-world training data is fairly small, the parameter initialisation using *any* pretrained net is essential. In any case, the improved performance via the use of *any* pretrained net (*with* or *without* rotation artifacts) compared to random parameter initialisation (not pretrained), signifies that the creation of synthetic tomograms and the training the pretrained nets is worthy of the additional time that both processes require. The topic of time investment is especially significant and it is discussed further in Section 6.5.

By viewing, Figures 6.11, 6.12, 6.13, we can qualitatively infer that with the use of a pretrained net from a synthetic tomogram *with* rotating artifacts, the accuracy with only 10% and 5% of the real-world training data is comparable to the use of 100% of the available training data, but without the use of any pretrained net. Namely, by comparing marks $\boldsymbol{P_1}$-$\boldsymbol{P_6}$ to marks $\boldsymbol{T_1}$-$\boldsymbol{T_3}$ and $\boldsymbol{G_1}$-$\boldsymbol{G_3}$, it can be seen that the network that used a pretrained net from synthetic data *with* rotating artifacts is able to accurately segment the

difficult classes with Labels 2 and 3. Moreover, in the case of $\boldsymbol{P_7}$-$\boldsymbol{P_9}$ it can seen that the latter pretrained net can be relatively accurate even if it has not been trained with real-world data. While, this can be said for the pretrained net from the synthetic tomogram *with* rotating artifacts, sadly by examining marks $\boldsymbol{R_1}$-$\boldsymbol{R_6}$ the other pretrained net can be seen to deviates noticeably for the ground truth data. Similarly, by examining Figure 6.12(**i**) it can be seen that the pretrained net from a synthetic tomogram *without* rotating artifacts is not in a position to infer directly on real-world data. This is potentially due to the pretrained net being over-reliant on the good quality of the input (synthetics *without* any rotating artifacts) in order to be accurate, and therefore it. Nevertheless, by comparing the marks $\boldsymbol{R_1}$-$\boldsymbol{R_6}$ to the $\boldsymbol{T_4}$-$\boldsymbol{T_9}$ ones there is some slight improvement. This fact advocates in favor of using *any* pretrained net, when the amount of training data decreases substantially.

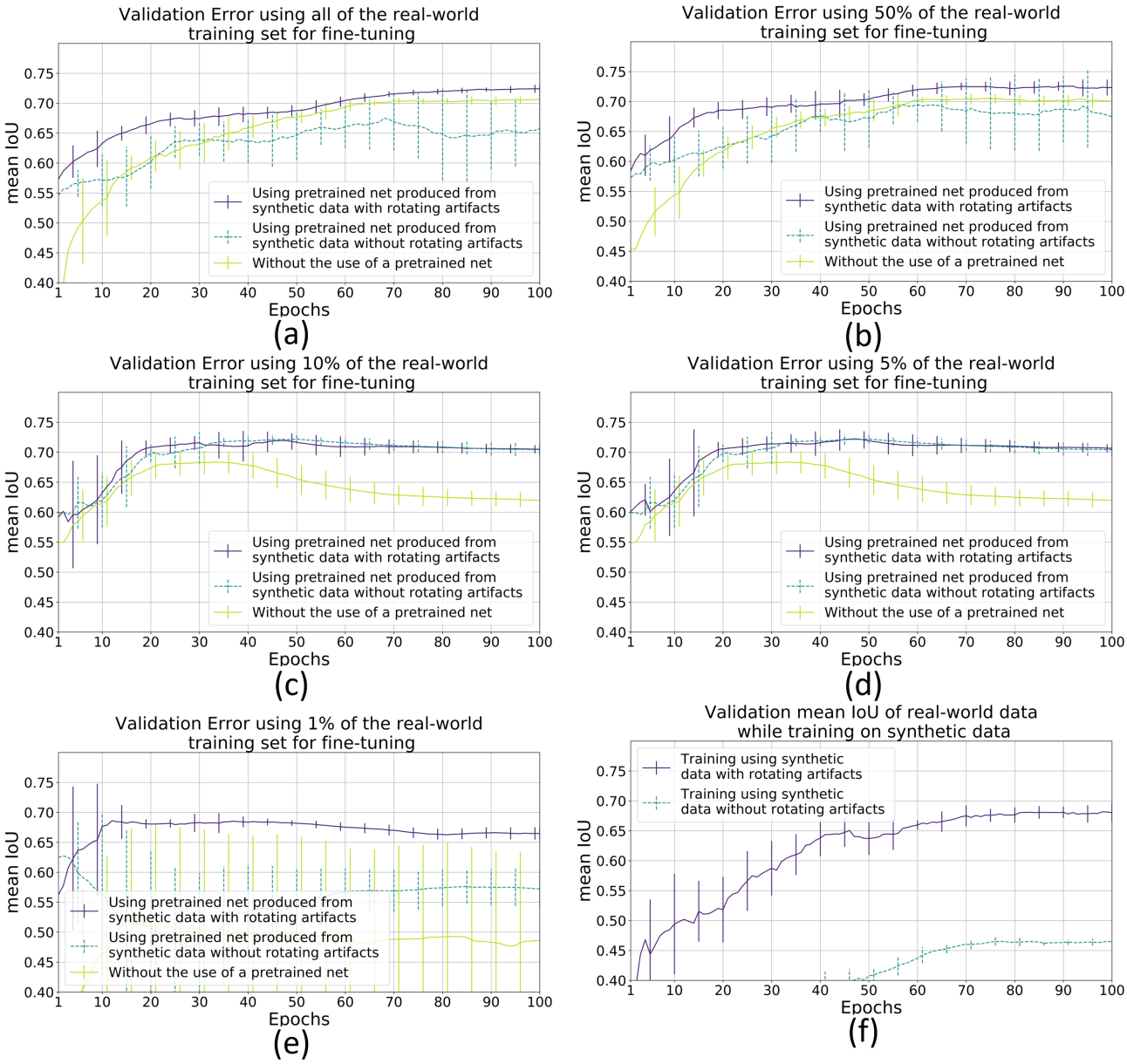
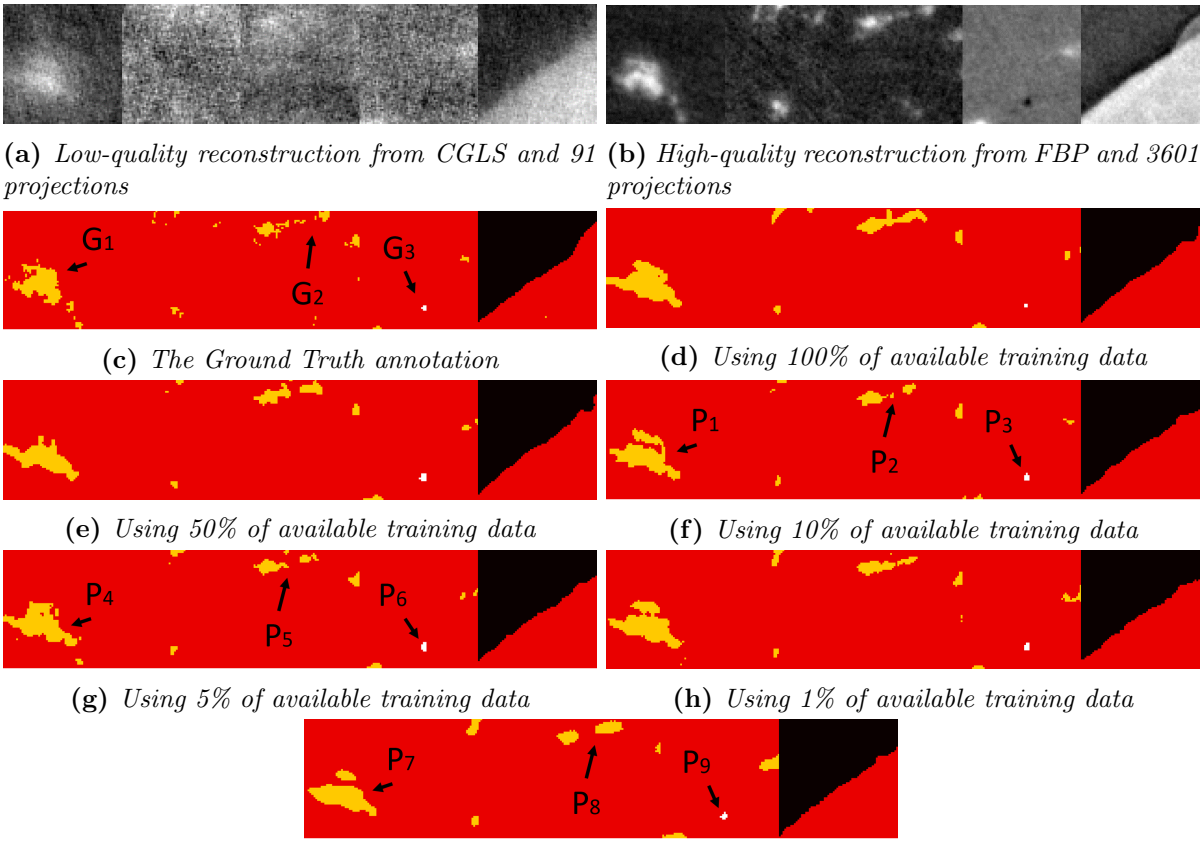


Figure 6.10: *Validation mean IoU for the segmentation experiments where the input is the CGLS reconstruction of the real-world tomogram (more in Section 6.3), and by initialising the parameters using pretrained nets produced from synthetic data* ***with*** *rotating artifacts, without rotating artifacts or without the use of a pretrained net. The network is then fine-tuned using (**a**) 100%, (**b**) 50%, (**c**) 10%, (**d**) 5%, (**e**) 1%, or (**f**) 0% of available real-world training samples.of available real-world training samples. As can be seen, using the pretrained net produced from training with a synthetic tomogram* ***with*** *rotating artifacts offers overall higher accuracy, even when using very small amounts of real-world data for training/fine-tuning. We can also note it is stable based on the error bars. All values are averages between the 3 iterations of the 3-fold validation/testing, averaged over a 10 epoch moving average. The error bars refer to the standard deviation that the approaches achieved based om a (3x)10 epoch moving standard deviation.*

Table 6.2: IoU results of the segmentation output of Stacked-DenseUSeg during testing, using progressively lower amounts of the real-world training samples and by initialising the parameters using pretrained nets produced from synthetic data **with** rotating artifacts, **without** rotating artifacts or without the use of a pretrained net. While the training samples are reduced, the testing samples are not, making the results comparable between themselves and to the results on Table 5.6. The values presented are averages between the 3 iterations of the 3-fold validation/testing, followed by the standard deviation that the approaches achieved in 3-fold validation/testing. As it can be seen, training/fine-tuning using a pretrained net produced from training with a synthetic tomogram **with** rotating artifacts offers better segmentation accuracy even for the occasion when there are sufficient real-world annotated data available for training. For IoU higher is better, while for standard deviation smaller is better.

	Pretrained net *with* rotating artifacts	Pretrained net *without* rotating artifacts	Without a pretrained net
Using 100% training dataset			
IoU of Label 0	$\mathbf{0.99} \pm 8 \cdot 10^{-4}$	$\mathbf{0.99} \pm 7 \cdot 10^{-4}$	$\mathbf{0.99} \pm 2 \cdot 10^{-4}$
IoU of Label 1	$\mathbf{0.97} \pm 5 \cdot 10^{-4}$	$\mathbf{0.97} \pm 4 \cdot 10^{-3}$	$\mathbf{0.97} \pm 3 \cdot 10^{-4}$
IoU of Label 2	$\mathbf{0.47} \pm 3 \cdot 10^{-3}$	$0.42 \pm 2 \cdot 10^{-2}$	$0.45 \pm 5 \cdot 10^{-3}$
IoU of Label 3	$\mathbf{0.50} \pm 6 \cdot 10^{-3}$	$0.45 \pm 3 \cdot 10^{-2}$	0.46 ± 10^{-2}
mean IoU	$\mathbf{0.73} \pm 4 \cdot 10^{-4}$	$0.71 \pm \cdot 10^{-2}$	$0.72 \pm 4 \cdot 10^{-3}$
Using 50% training dataset			
IoU of Label 0	$0.98 \pm 6 \cdot 10^{-4}$	$\mathbf{0.99} \pm 3 \cdot 10^{-4}$	$0.98 \pm 5 \cdot 10^{-4}$
IoU of Label 1	$\mathbf{0.98} \pm 10^{-3}$	$0.97 \pm 9 \cdot 10^{-5}$	$\mathbf{0.98} \pm 10^{-3}$
IoU of Label 2	$\mathbf{0.47} \pm 4 \cdot 10^{-3}$	$\mathbf{0.47} \pm 5 \cdot 10^{-3}$	$0.46 \pm 5 \cdot 10^{-3}$
IoU of Label 3	$\mathbf{0.49} \pm 5 \cdot 10^{-3}$	$0.47 \pm 2 \cdot 10^{-2}$	$0.43 \pm 2 \cdot 10^{-2}$
mean IoU	$\mathbf{0.73} \pm 2 \cdot 10^{-3}$	$\mathbf{0.73} \pm 4 \cdot 10^{-3}$	$0.71 \pm 6 \cdot 10^{-3}$
Using 10% training dataset			
IoU of Label 0	$\mathbf{0.98} \pm 10^{-3}$	$\mathbf{0.98} \pm 2 \cdot 10^{-3}$	$\mathbf{0.98} \pm 3 \cdot 10^{-3}$
IoU of Label 1	$\mathbf{0.98} \pm 10^{-3}$	$0.97 \pm 3 \cdot 10^{-3}$	$0.97 \pm 2 \cdot 10^{-3}$
IoU of Label 2	$\mathbf{0.48} \pm 10^{-3}$	0.46 ± 10^{-2}	$0.46 \pm 2 \cdot 10^{-2}$
IoU of Label 3	$\mathbf{0.48} \pm 10^{-2}$	$0.44 \pm 3 \cdot 10^{-2}$	$0.40 \pm 9 \cdot 10^{-3}$
mean IoU	$\mathbf{0.73} \pm 3 \cdot 10^{-3}$	0.72 ± 10^{-2}	$0.70 \pm 4 \cdot 10^{-3}$
Using 5% training dataset			
IoU of Label 0	$\mathbf{0.98} \pm 10^{-3}$	$\mathbf{0.98} \pm 2 \cdot 10^{-3}$	$\mathbf{0.98} \pm 3 \cdot 10^{-3}$
IoU of Label 1	$\mathbf{0.98} \pm 10^{-3}$	$\mathbf{0.98} \pm 3 \cdot 10^{-4}$	$\mathbf{0.98} \pm 10^{-3}$
IoU of Label 2	$\mathbf{0.48} \pm 10^{-3}$	$\mathbf{0.48} \pm 4 \cdot 10^{-3}$	$0.47 \pm 6 \cdot 10^{-3}$
IoU of Label 3	$0.47 \pm 2 \cdot 10^{-2}$	$\mathbf{0.48} \pm 9 \cdot 10^{-3}$	$0.37 \pm 4 \cdot 10^{-2}$
mean IoU	$\mathbf{0.73} \pm 6 \cdot 10^{-3}$	$\mathbf{0.73} \pm 2 \cdot 10^{-3}$	0.70 ± 10^{-2}
Using 1% training dataset			
IoU of Label 0	$\mathbf{0.97} \pm 9 \cdot 10^{-4}$	0.95 ± 10^{-2}	0.90 ± 0.1
IoU of Label 1	$\mathbf{0.98} \pm 5 \cdot 10^{-4}$	$0.97 \pm 4 \cdot 10^{-3}$	$0.97 \pm 7 \cdot 10^{-3}$
IoU of Label 2	$\mathbf{0.48} \pm 2 \cdot 10^{-3}$	$0.42 \pm 2 \cdot 10^{-2}$	$0.41 \pm 5 \cdot 10^{-2}$
IoU of Label 3	$\mathbf{0.42} \pm 10^{-2}$	$0.25 \pm 8 \cdot 10^{-2}$	0.13 ± 0.1
mean IoU	$\mathbf{0.71} \pm 4 \cdot 10^{-3}$	$0.65 \pm 3 \cdot 10^{-2}$	$0.60 \pm 6 \cdot 10^{-2}$
Using the pretrained nets directly for inferring			
IoU of Label 0	$\mathbf{0.97} \pm 3 \cdot 10^{-4}$	$\mathbf{0.97} \pm 3 \cdot 10^{-4}$	-
IoU of Label 1	$\mathbf{0.98} \pm 3 \cdot 10^{-4}$	$0.46 \pm 6 \cdot 10^{-4}$	-
IoU of Label 2	$\mathbf{0.48} \pm 2 \cdot 10^{-3}$	$0.06 \pm 4 \cdot 10^{-4}$	-
IoU of Label 3	$\mathbf{0.35} \pm 3 \cdot 10^{-3}$	$0.003 \pm 5 \cdot 10^{-6}$	-
mean IoU	$\mathbf{0.69} \pm 4 \cdot 10^{-4}$	$0.37 \pm 2 \cdot 10^{-4}$	-



(a) *Low-quality reconstruction from CGLS and 91 projections*

(b) *High-quality reconstruction from FBP and 3601 projections*

(c) *The Ground Truth annotation*

(d) *Using 100% of available training data*

(e) *Using 50% of available training data*

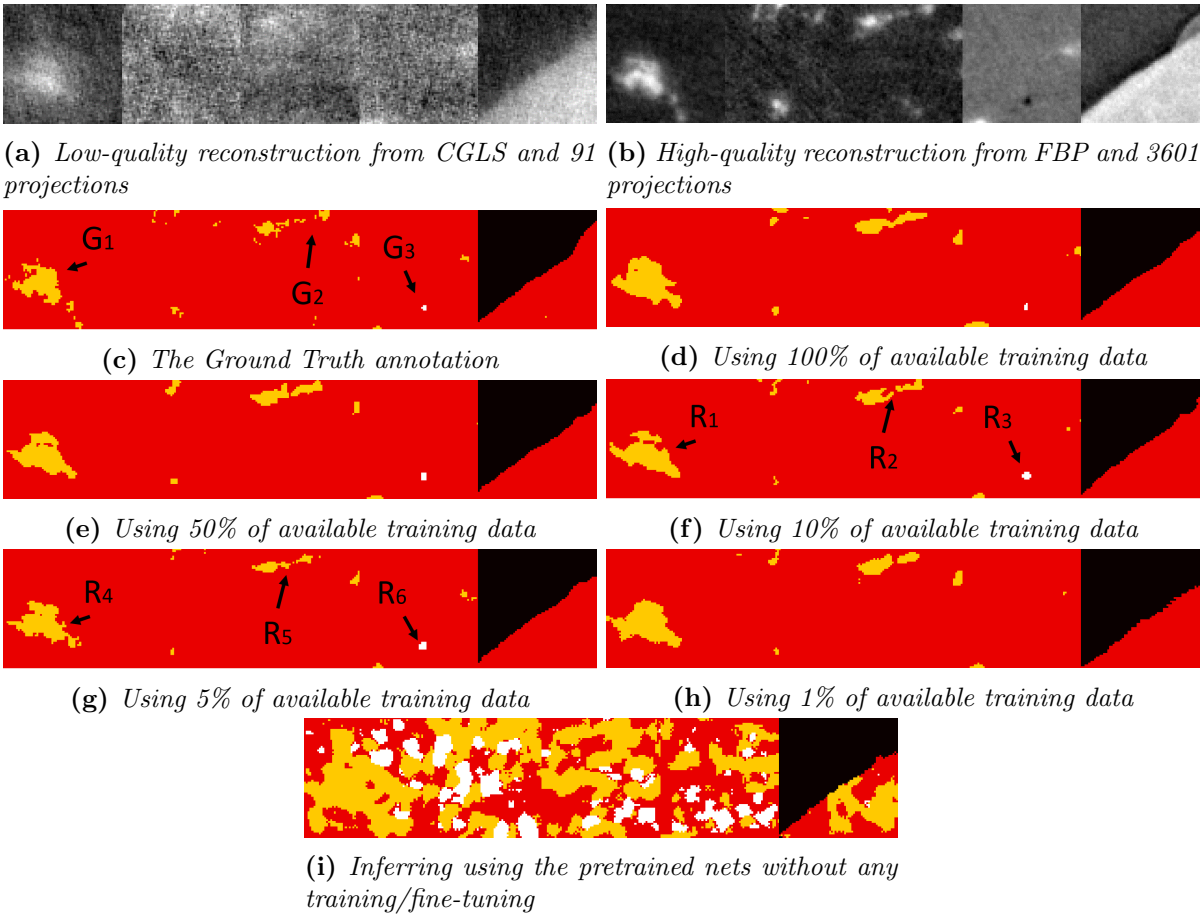
(f) *Using 10% of available training data*

(g) *Using 5% of available training data*

(h) *Using 1% of available training data*

(i) *Inferring using the pretrained nets without any training/fine-tuning*

Figure 6.11: *Segmentation results of Stacked-DenseUSeg with progressively less real-world training samples, and using the pretrained net trained on synthetic data* ***with*** *rotating artifacts [429, 430]. (**a**) The CGLS, 91 projection reconstruction [426] that is the networks input, (**b**) the FBP, 3601 projection reconstruction [425] that is ground truth for the denoising part of Stacked-DenseUSeg, (**c**) Annotations: black refers to the air outside the base material, red refers to the base material, yellow are the magnesium deposits and white are the air pockets, (**d**) Using 100% of training samples, (**e**) 50%, (**f**) 10%, (**g**) 5%, (**h**) 1%, (**i**) Inferring directly using the pretrained nets without any training/fine-tuning. Marks* $\boldsymbol{G_1}$-$\boldsymbol{G_3}$, $\boldsymbol{P_1}$-$\boldsymbol{P_9}$ *are described in text.*



(a) *Low-quality reconstruction from CGLS and 91 projections*

(b) *High-quality reconstruction from FBP and 3601 projections*

(c) *The Ground Truth annotation*

(d) *Using 100% of available training data*

(e) *Using 50% of available training data*

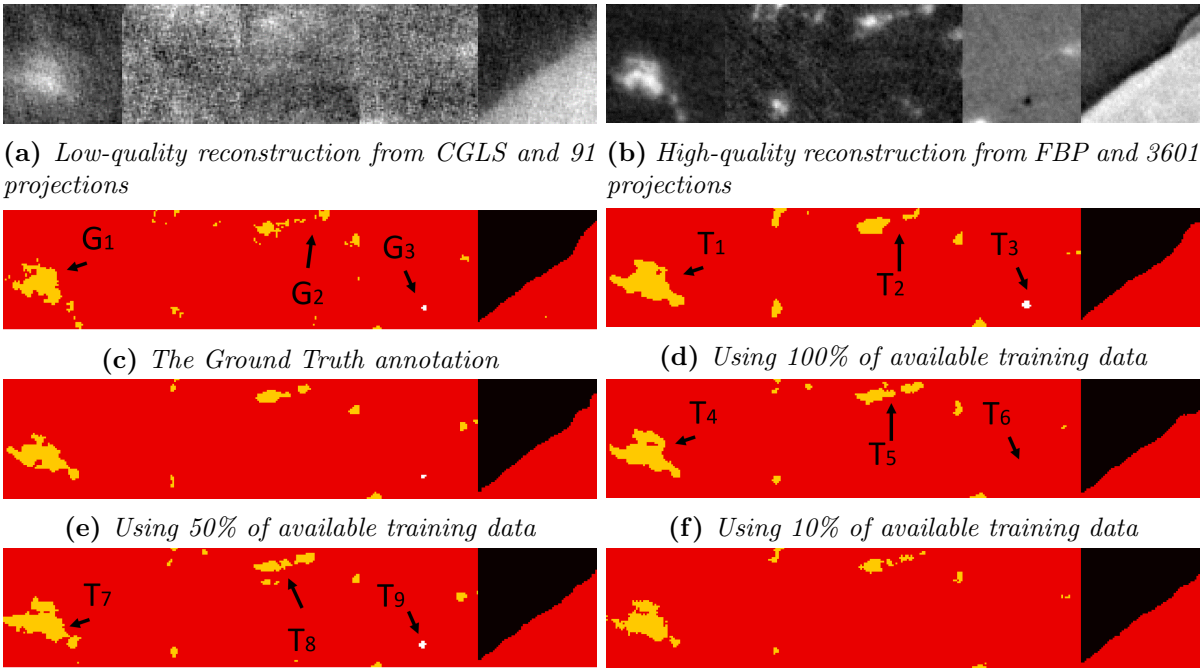
(f) *Using 10% of available training data*

(g) *Using 5% of available training data*

(h) *Using 1% of available training data*

(i) *Inferring using the pretrained nets without any training/fine-tuning*

Figure 6.12: *Segmentation results of Stacked-DenseUSeg with progressively less real-world training samples and using the pretrained net trained on synthetic data* ***without*** *rotating artifacts [431, 432]. (**a**) The CGLS, 91 projection reconstruction [426] that is the networks input, (**b**) the FBP, 3601 projection reconstruction [425] that is ground truth for the denoising part of Stacked-DenseUSeg, (**c**) Annotations: black refers to the air outside the base material, red refers to the base material, yellow are the magnesium deposits and white are the air pockets, (**d**) Using 100% of training samples, (**e**) 50%, (**f**) 10%, (**g**) 5%, (**h**) 1%, (**i**) Inferring directly using the pretrained nets without any training/fine-tuning. Marks* $\boldsymbol{G_1}$*-*$\boldsymbol{G_3}$*,* $\boldsymbol{R_1}$*-*$\boldsymbol{R_6}$ *are described in text.*



(a) *Low-quality reconstruction from CGLS and 91 projections*

(b) *High-quality reconstruction from FBP and 3601 projections*

(c) *The Ground Truth annotation*

(d) *Using 100% of available training data*

(e) *Using 50% of available training data*

(f) *Using 10% of available training data*

(g) *Using 5% of available training data*

(h) *Using 1% of available training data*

Figure 6.13: *Segmentation results of Stacked-DenseUSeg with progressively less real-world training samples and without the use of a pretrained net. (**a**) The CGLS, 91 projection reconstruction [426] that is the networks input, (**b**) the FBP, 3601 projection reconstruction [425] that is ground truth for the denoising part of Stacked-DenseUSeg, (**c**) Annotations: black refers to the air outside the base material, red refers to the base material, yellow are the magnesium deposits and white are the air pockets, (**d**) Using 100% of training samples, (**e**) 50%, (**f**) 10%, (**g**) 5%, (**h**) 1%. Marks $\boldsymbol{G_1}$-$\boldsymbol{G_3}$, $\boldsymbol{T_1}$-$\boldsymbol{T_9}$ are described in text.*

6.5 Discussion and time management

The use of knowledge transfer schemes have been shown to be advantageous in regards to the overall accuracy and the use of less real-world annotated data. Additionally, by examining step-by-step the additional time needed in order to implement our knowledge transfer scheme versus the time that is saved because of it, it can be deduced that the application of a knowledge transfer scheme is also advantageous in terms of time. Namely, the time required to synthesise synthetic tomograms, obtain pretrained nets and subsequently fine-tune/train an accurate network with a limited amount of annotated real-world data, is less than the time required to annotated a substantial amount of annotated real-world data and then train an accurate network with them. Additionally, the creation of synthetic tomograms and their respective pretrained nets can occur parallelly with the *partial* manual annotation of real-world data, thus removing any additional time delays.

In particular, in the case of a materials science experiment, the dictation of a list-file that describes how TomoPhantom [23] should generate a synthetic tomogram may take a only few minutes from the user to completed it. Alternatively, for more complex synthetic tomograms that are comprised by multiple virtual objects in the phantom, the earlier file could be partially generated via code. For instance, for the generation of the numerous virtual objects that emulate the magnesium and air pockets in the synthetic tomograms described in Section 6.2.2, we created a small Python script. Our script then iteratively selected the random in most cases positions, sizes and orientations of the virtual objects that were placed in the final synthetic phantom. Overall, the time required to generated the list-file, which TomoPhantom [23] uses to produce the synthetic tomograms may require one hour at most. As presented in Table 6.3 the dictation of our small Python script and the generation of the list-file from the script required approximately 40 minutes. After the creation of the list-file, it is given as input to TomoPhantom to generate the synthetic phantom and then the synthetic phantom is given as input to the Astra toolbox [440] to generate tomograms (our *SynthHS3* and *SynthHS4*), with the whole process from list-file to tomograms requiring close to 24 hours (as reported in Table 6.3) depending of course on the available hardware. In the future (as mentioned in Table 6.3) it is expected

that this time maybe reduced further since an MPI version of TomoPhantom is under consideration, which may reduce the computational time using distributed computing. Finally the generation of the pretrained nets with the hardware available to us required 18 hours. Our workstation as mentioned in Section 5.4.5, had 4 NVIDIA Tesla V100s GPUs, a 20-core 2.5GHz CPU and 380GB of RAM. As mentioned in Table 6.3, it is possible to reduce the time required to train the pretrained nets further, using a termination strategy that utilises the validation error. For example a rule can be inplace, where if for 10 consecutive epochs there is not a new minimum recorded validation error then the training is terminated.

The next step after the creation of the pretrained nets, is the necessary *partial* annotation of a real-world tomogram, using special software like SuRVoS [306], which may take an experienced user from a few hours to a day in order to annotate, depending of course on the complexity of the tomogram. For our data it is estimated that the manual annotation of 5% of the high-sampled tomogram from an experienced user would require one working day (see Table 6.3). In comparison, the *complete* annotation of a tomogram $2160 \times 2560 \times 2560$ in voxel resolution, similar to ones used here, may require multiple days, to weeks for a single annotator, or even multiple annotators in order to finish. For our data approximately 10 working days were needed in order to manually annotate the representative high-sampled tomogram (see Table 6.3). It is important to mention here, that the manual annotation can be performed simultaneously with the generation of synthetic data and pretrained nets. Namely, a user can spend at maximum an hour in order to write code that will generate a list-file of virtual objects that TomoPhantom uses as input (in our case as reported in Table 6.3 it took approximately 40 minutes). Then, the generation of the synthetic data and pretrained net is handled by software, not human supervision is needed and simultaneously the user could perform manual annotation of the real-world dataset. If an already constructed pretrained net is considered for parameter initialisation, it is recommended that user to be certain that given TomoPhantom's functionality a more realistic synthetic analog can not be created. This is because the user can save time by creating a new synthetic tomogram and pretrained net while he/she performs a *partial*

manual annotation of the highly-sampled representative tomogram. At the final step, an instance of Stacked-DenseUSeg is trained with parameter initialisation using a pretrained net. As seen by Table 6.3, this require 18 hours with our hardware and as mention in the previous paragraph and it is possible to reduce this time using a termination strategy based on monitoring the validation error.

Overall, the human involvement using knowledge transfer is the dictation of code that creates the list-file and the *partial* manual annotation of a real-world tomogram, which may require just one day of human labor. In contrast, weeks of manual work are required for a *complete* tomogram annotation when no knowledge transfer is employed. As can be seen in Table 6.3 for our data, even when *half* of the real-world tomogram is annotated and knowledge transfer is used, it is still faster than the *complete* annotation of the real-world tomogram without knowledge transfer. Furthermore, since the synthetic datasets can be general enough to simulate a variety of similar real-world tomograms, the aforementioned additional time required for rendering new synthetic tomograms and creating new pretrained nets, might occur less frequently. Typically, when a particular temporal phenomenon is studied using a time-series of undersampled tomograms multiple time-series are collected for different physical samples that have the same chemical composition and similar physical structure. The time-series used throughout this thesis, is one of two time-series (*RealTS1* and *RealTS2* as presented in Section 3.2.2) that were collected in order to study the corrosion caused by salt-water in aluminium pins with magnesium deposits [418]. In other studies, time-resolved tomograms are used to study the nucleation and growth dynamics of certain materials when they are solidifying [453], magma crystallisation [454] and also the formation of dendritic structures as metallic alloys are solidifying [302, 303, 304, 305]. Namely, for the last type of studies due to large similarities in their structure, already constructed pretrained nets could be used with little to none modification in order to completely or partially segment the respective tomogram time-series. Alternatively, new pretrained could be created faster training, without the need to re-render new synthetic tomograms, but just using slightly modified old synthetic tomograms (e.g after making changes in intensity of certain classes, merging

	Dictation of code that creates a list-file of virtual objects which is used to construct the synthetic data	Generation of synthetic data using the list-file + Pretrained net with synthetic data	Manual annotation of real-world data	Training with real-world data (using parameter init. with the pretrained net in KT)	Total human time	Total computation time	Total time
KT and man. annotation of 5% of data	~ 40 minutes$^{(2)}$	$24\text{h}^{(1,2,4)} + 18\text{h}^{(1,5)} \approx 1.75$ days	~ 1 working day$^{(3)}$	18 hours$^{(1,5)}$	~ 1 working day + 40 minutes	2.5 days	$2.5 - 3.5$ days$^{(6)}$
KT and man. annotation of 10% of data	~ 40 minutes$^{(2)}$	$24\text{h}^{(1,2,4)} + 18\text{h}^{(1,5)} \approx 1.75$ days	~ 2 working days$^{(3)}$	18 hours$^{(1,5)}$	~ 2 working days + 40 minutes	2.5 days	$2.75 - 4.5$ days$^{(6)}$
KT and man. annotation of 50% of data	~ 40 minutes$^{(2)}$	$24\text{h}^{(1,2,4)} + 18\text{h}^{(1,5)} \approx 1.75$ days	~ 5 working days$^{(3)}$	18 hours$^{(1,5)}$	~ 5 working days + 40 minutes	2.5 days	$5.75 - 7.5$ days$^{(6)}$
KT and man. annotation of 100% of data	~ 40 minutes$^{(2)}$	$24\text{h}^{(1,2,4)} + 18\text{h}^{(1,5)} \approx 1.75$ days	~ 10 working days$^{(3)}$	18 hours$^{(1,5)}$	~ 10 working days + 40 minutes	2.5 days	$10.75 - 12.5$ days$^{(6)}$
No KT and man. annotation of 100% of data	-	-	~ 10 working days$^{(3)}$	18 hours$^{(1,5)}$	~ 10 working days	18 hours	10.75 days$^{(6)}$

Table 6.3: Human and computational time required with or without Knowledge Transfer (KT) for the development of Stacked-DenseUSeg that can segment real-world data. The table presents the human and computational time required with knowledge transfer and with the representative highly-sampled tomogram being manually annotated by various amounts. As can be seen even when knowledge transfer is used with manual annotation of 50% of the highly-sampled tomogram it is still faster than not implementing knowledge transfer and manually annotating 100% of the highly-sampled tomogram.
$^{(1)}$: This time may vary with different hardware. $^{(2)}$: This time may vary for synthetic data with different complexity and number of virtual objects. $^{(3)}$: This time may vary for real-world data with different complexity and number of segmentation classes. $^{(4)}$: Generation of synthetic phantoms and tomograms is expected to improve in the future with an implementation of TomoPhantom that uses MPI and distributed computing. $^{(5)}$: This time may be reduced with an implementation of a termination strategy using the validation error. For example a rule can be inplace where if for 10 consecutive epochs there is not a new minimum recorded validation error the training is terminated. $^{(6)}$: The first time is when the generation of synthetic data and the respective pretrained net occurs in parallel with the manual annotation of real-world. The second time is when none of the relative tasks occurs simultaneously.

certain segmentation classes, etc.).

In a typical use-case scenario, either a pretrained net that has already been created can be used for parameter initialisation or a new one has to be created using a synthetic tomogram. Synthetic tomograms are created as synthetic analogs to real-world tomograms using the TomoPhantom's capabilities. If the synthetic tomograms used to create the pretrained net, are analogs to different real-world tomograms that have different segmentation classes (in terms of number, type and relative geometry) than the ones in the current real-world tomogram that has to be segmented, then a new synthetic tomogram

and pretrained net should be generated. Since the *partial* annotation of the representative highly-sampled tomogram (it is recommended at least 5% of the total) would require 1-2 days (see Table 6.3), it is wise to generate a new synthetic tomogram and pretrained net simultaneously, provided that a new synthetic tomograms that better resembles the real-world data can be generated given TomoPhantom's capabilities. It is important to mention here, that the user should *partially* annotate a portion of real-world data that is representative to the whole highly-sampled tomogram. This probably means that multiple smaller unconnected regions should be annotated in an attempt to have all segmentation classes be represented in the *partial* annotation. After using the partial annotations to fine-tune the pretrained net, if the achieved segmentation accuracy is high enough, according to specifications that set by the user, he/she may stop at this step.

Alternatively the user may continue with the annotation of further real-world data. If the additional annotated data do not increase the segmentation accuracy after the fine-tuning of the pretrained net, it is very likely that due to the low SNR of the real-world reconstructions further accuracy can not be achieved. If however the additional annotated data do increase the segmentation accuracy, the user may continue with the annotation of more real-world data until he/she observes after the fine-tuning of the pretrained net diminishing returns. Naturally, the earlier is recommended with the assumption that the synthetic dataset used to generate the pretrained net already resembles the real-world data as best as possible given TomoPhantom's capabilities. If the user believes that more realistic synthetic data can be created (using different settings in TomoPhantom or using an additional software that does not require human supervision), it is recommended to create them while he/she simultaneously performs further manual annotation of the real-world data. This is because there is higher certainty that further manually annotations will potentially offer high accuracy solutions. Therefore precious time will not be lost, while the user waits for a new synthetic tomogram and pretrained net to be generated. The user then can test if the new pretrained net (or the original one if he/she did not make a new one while manually annotating) performs similarly good with less or more annotations or if more annotations offer better results. If the accuracy does not improve

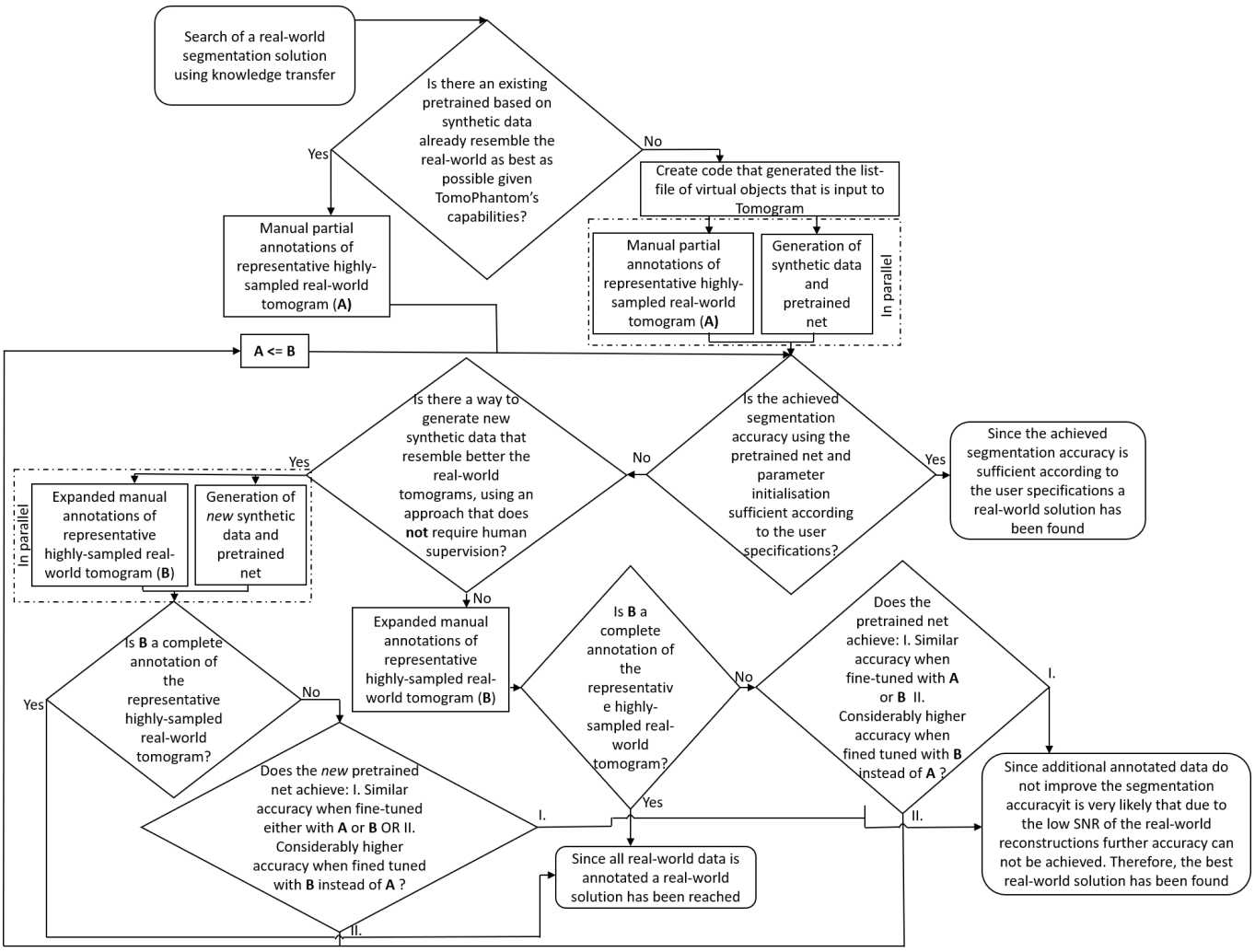


Figure 6.14: *Flowchart presenting the steps in order to obtain a highly accurate version of Stacked-DenseUSeg that segments real-world data using knowledge transfer. As presented it is highly recommended to perform the manual partial annotation of the highly-sampled representative tomogram in parallel with the generation of the synthetic data and the pretrained net.*

with more annotations it is highly likely that the best segmentation solution has been found and the accuracy can not improve further due to the low SNR of the real-world reconstructions. If accuracy does improve with more annotations, then he/she should proceed with more manual annotation. Figure 6.14 presents in detail using a flowchart all steps mentioned above.

In any case it is recommended to generate the synthetic data and pretrained nets simultaneously with the manual data annotation. By performing this, even when knowledge transfer is unable to offer a high accuracy solution using a *partial* annotation, the user would be able to obtain one using a *complete* annotation without knowledge transfer and without extra delays.

6.6 Conclusions

In this chapter we proposed a novel use of knowledge transfer in deep learning networks, using synthetic tomograms [429, 430, 431, 432, 452]. Our proposed knowledge transfer method works to improve the denoise and segmentation accuracy of our proposed Stacked-DenseUSeg [26, 27] (code available from [28]) by allowing the network to better generalise the way with which it accomplices its tasks. This transfer of knowledge is achieved by initially training a Stacked-DenseUSeg solely with synthetic data and then use its parameters to initialise the parameters of a new Stacked-DenseUSeg which will be fine-tuned/trained with real-world data. These initial Stacked-DenseUSegs, also referenced as "pretrained" nets, are the preferred way with which denoising and segmentation information obtained from synthetic data can be utilised for the same task in real-world data. This is because upon their generation, they can be reused to initialise the parameters of new instances of Stacked-DenseUSeg that will work on new varieties of real-world data, which can be approximated by the same synthetic data.

Furthermore, in the experiments Section 6.4 there are two different pretrained nets that are compared with each other, generated after training with two different synthetic tomograms. One of the synthetic tomograms is more simplistic and contains only noise, while the other also includes rotating artifacts similar to the ones present in the real-world data. The pretrained nets are also tested under different amounts of real-world data used for fine-tuning/training.

Based on the results, the use of pretrained nets is always generally beneficial compared to not using pretrained nets especially for the cases where fewer real-world data are available for fine-tuning/training. In terms of semantic segmentation, the pretrained net from the synthetic tomogram *with* rotating artifacts offers noticeably higher accuracy compared to the one trained with the synthetic tomogram *without* rotating artifacts. Additionally, when using 100%, 50%, 10% or only 5% of the real-world training samples, the overall accuracy remains unwavered. This means that the real-world training samples can be reduced to only 5% of their original count, without any trade off to the accuracy. It is

important to note here however, that performed experiments are regarding the dataset of a single physical sample/object. This means that the proposed knowledge scheme may reduce the amount of the real-world annotated data used for training, that otherwise would be needed for an accurate segmentation of different datasets that image different physical samples/objects. Although, independent of the sample/object, the annotation of the reduced real-world training samples/segmentation-instances should be performed in a manner that will result in a representative training set. In order to perform the earlier, it may necessary not to annotate a continuous small region in the available highly-sampled tomogram reconstruction, but multiple smaller ones that jointly construct a representative training set.

Similarly, in the task of denoising, the use of pretrained nets is also beneficial. The pretrained net from the synthetic tomogram *without* rotating artifacts has a slight advantage when high amounts of real-world training data are available for fine-tuning/training; however the pretrained net from the synthetic tomogram *with* rotating artifacts has a more prevalent advantage when low amounts of real-world training data are used. The performance of the latter in both cases of semantic segmentation and denoising, can potentially be attributed to the fact that the pretrained net offers significant information regarding the handling of rotating artifacts in the input, which can lead to major errors if they are network is not equipped for them. Moreover, the slight advantage of pretrained from the synthetic tomogram *without* rotating artifacts in denoising is also related to the rotating artifacts, but this time is potentially caused by their absence in the high SNR reconstruction that is used as the denoising ground truth. Based on this, a potential future improvement of the knowledge transfer process may arrive from a pretrained net generated by a synthetic tomogram with a low SNR reconstruction *with* rotating artifacts used as input and a high SNR reconstruction *without* rotating artifacts used as the denoising ground truth. Nevertheless, the potential improvement is not expected to be significant and it would only likely be effective when high amounts of training data are available for training/fine-tuning, which an infrequent use-case.

To summarise, there are three major improvements that are achieved through the use of

our proposed knowledge transfer scheme. The first is that the overall obtainable accuracy is increased for both the tasks of denoising and the semantic segmentation. Secondly, and most importantly, the amount of real-world annotated samples necessary for the use in real-world applications is reduced dramatically. With only the use of 186 $64 \times 64 \times 64$ training samples (that equates to 5% of the available 3723 training samples), our final network is able to offer the same accuracy in terms of semantic segmentation compared to when all the existing training samples are used. As it can be seen in Table 6.2, for the pretrained net from the synthetic tomogram *with* rotating artifacts, when 100%, 50%, 10% or only 5% of the real-world training samples are used for fine-tuning/training the mean IoU remains unwavered at 0.73. Similarly, in terms of denoising the accuracy achieved gets only slightly decreased and it remains comparable in terms of the perceptual quality with the denoising result achieved when all available training data is used.

Finally, the third improvement is in terms of time. As is raised before, there are less real-world annotations that are needed for training, which translates to numerous hours previously spent on manual annotation now no longer need. Additionally, since the synthetic tomograms used for the generation of pretrained nets are approximating analogues of real-world data, a single one of them can potentially simulate multiple varieties of real-world data. This potentially prompts the reuse of the pretrained nets, which reduces the time demands of the knowledge transfer. In any case, the additional time required for the generation of the synthetic data and their corresponding pretrained nets corresponds to some additional computational time and not human effort. This means that these processes can be completed during off hours or in parallel with the manual annotation of real-world data, thus potentially not generating additional time delays.

Chapter 7

Temporal refinement of semantic segmentations in 4D datasets using hidden Markov models

7.1 Towards the segmentation of time-series of tomograms (4D datasets)

In the previous chapters, we presented different approaches for improving the Signal-to-Noise Ratio (SNR) in reconstructions of undersampled tomograms, denoising said tomograms, and culminating in semantic segmentation. Nevertheless, the overall goal of these approaches and this thesis is the segmentation of *time-series* of tomograms, also referred as 4D datasets. These are composed of a series of undersampled tomograms captured over time. Naturally, with the repeated application of our proposed Stacked-DenseUSeg (chapter 5 and 6) on each tomogram of a time-series, it possible to acquire 4D segmentation predictions, but this ignores the similarities in the 4^{th} dimension as it treats each 3D volume independently. Unfortunately, due to unavailability of high quality 4D annotations, similar to the difficulties associated with 3D annotations of the representative highly-sampled tomogram, it is not possible to implement supervised approaches which implicitly utilise temporal information. On the other hand, segmentations that are generated using only the spatial information present in each individual tomogram may not be temporally consistent; hence they are not always suitable for the analysis of temporal phenomena. For instance, in scene segmentations (video segmentations of a car driving along a road) there might be cases where a 2D segmentation CNN for some frames may

detects two cars in the far distance and in others it may falsely detects only one car (especially if the cars have the same colour, the distance is great, etc.). If the segmentation CNN could also use inter-frame information, it may be possible to discern that there are indeed two cars in all frames without failing for some. For these reasons in our data, after procuring segmentations for each tomogram of a time-series, extra refinement is needed using the temporal information of the temporal adjacent tomograms.

In this chapter we will present our two proposed novel applications of hidden Markov models (HMM) [29] submitted to Nature Scientific Reports, that are designed to refine the segmentation predictions of tomograms generated by our Stacked-DenseUSeg network [26], that compose a time-series. In particular, HMMs [455] derive their name from their hidden states, that are modeled by a Markov chain. Namely, these hidden states are the model's output prediction classes. Based on the hidden state or class at a specific time-step, there are transition probabilities regarding the hidden state at the next time-step. Additionally, in HMMs in each time-step there is a provided input that belongs to range of observable states. These states are correlated to the hidden states through emission probabilities, meaning that specific observations certain hidden states more or less probable in each time-step. Finally, HMMs have starting probabilities for each of their hidden states. Based on the above, the HMMs are essentially Bayesian networks comprised by the transition, emission and starting probabilities as its parameters. Specifically, our proposed HMMs use the temporally consecutive 3D segmentation predictions generated by our Stacked-DenseUSeg [26] as input and their output is "true" 4D segmentations. Furthermore, they work without the need for annotated data and model training since their parameters are partially determined empirically and also by metrics like the confusion matrix of our Stacked-DenseUSeg generated during its testing. The first of our proposed HMM, named HMM-T (Hidden Markov Model using only Temporal information) uses as input/observations the tomogram segmentations generated independent through time by our Stacked-DenseUSeg. While, this approach suffices for the introduction of temporal coherence between the segmentations our second HMM, named HMM-TC (Hidden Markov Model using both Time and Confidence information) expands upon the first by also util-

ising segmentation probabilities generated by our Stacked-DenseUSeg. To the best of our knowledge this is the first time 3D CNN networks and HMMs have been jointly been used to generate and then refine multiple 3D segmentations that constitute a 4D segmentation.

In summary in this chapter: :

1. We present a hidden Markov model approach, named HMM-T, that uses temporal information from temporally adjacent segmentations of tomograms in a time-series. Our model uses partially empirically determined parameters, as well as parameters set by the confusion matrix during testing of the CNN segmentation network that generated the predictions.
2. We present a hidden Markov model able to utilise label confidence information, named HMM-TC. Unlike HMM-T, this uses not only temporal information but also confidence information in the form of segmentation likelihood probabilities generated by the network. Its parameters are again partially determined empirically and from the confusion matrix during testing of the CNN segmentation network that generated the predictions.
3. We present experiments regarding our two models, HMM-T and HMM-TC. These provide qualitative proof for their good performance, test their behaviour under a number of conditions and also attempt to measure the accuracy of their predictions.

In the following Section 7.2, we will present an algorithm that aligns the tomograms of the time-series in case of translation error during the collection of the time-series. Then in Section 7.3, we will present the methodology behind our proposed hidden Markov models and in Section 7.4 the aforementioned experiments with our proposed hidden Markov models. Finally Section 7.6, concludes the chapter.

7.2 Tomogram translation for spatial continuity through time

Before the segmentation of a time-series of tomograms using a 3D segmentation network, the misalignment of the tomograms in the time-series, also referenced in Section 4.5.3.1,

has to be addressed. In brief, one of the major issues regarding the collection strategy used for this real-world time-series of tomograms is the potential translation of the imaged object, affecting the spatial continuity through time. Namely, in each tomogram of the time-series the following erroneous translation transformation may occur.

$$\boldsymbol{v}_t = \boldsymbol{P}_t \boldsymbol{v}' = \begin{bmatrix} 1 & 0 & 1 & p_{x,t} \\ 0 & 1 & 0 & p_{y,t} \\ 0 & 0 & 1 & p_{z,t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x^{'} \\ v_y^{'} \\ v_z^{'} \\ 1 \end{bmatrix} = \begin{bmatrix} v_x^{'} + p_{x,t} \\ v_y^{'} + p_{y,t} \\ v_z^{'} + p_{z,t} \\ 1 \end{bmatrix} \tag{7.1}$$

where $\boldsymbol{v}_t$ is the new position of what the voxel $\boldsymbol{v}'$ depicts in $t = 0$, $\boldsymbol{P}_t$ is the translation matrix for the tomogram of time t. Therefore, in order for the voxels $\boldsymbol{v}_{1,2,\cdots,n}$ to depict the same location as the $\boldsymbol{v}_0 = \boldsymbol{v}'$ voxel, each tomogram in the series must be undo its translation using a reverse translation described by $\boldsymbol{P}^{-1}_{1,2,\cdots,n}$. Thankfully during this capture of a time-series tomograms, there is not any significant translation along the y and so $p_{y,t} = 0$ for $t = 0, 1, \cdots, n$.

To calculate the translation present in each tomogram of the time-series along the x and z axis, we perform the following algorithm:

The algorithm sums 100 *xz-slices* with the highest y value, that are in the bottom of the imaged object in the height dimension. This is because the bottom slices are usually unaffected by temporal events in our datasets, and so should (imaging artifacts aside) remain unchanged throughout the sequence. Additionally, due the sample's structure that is constant in each tomogram through time (there is an aluminum pin and any temporal events happen only towards the top of the pin where the chemical reactions are taking place), the element-wise summation of multiple slices may produce a 2D image that is unaffected by temporal events, meaning that it is similar for all tomograms of a time-series. This is because imaging information is merged from multiple slices, leading to an 2D image where two features can be distinguished, the background (Air, class=0) and the foreground (classes=1-3). Since all temporal events occur in the foreground, this image is a temporally invariant feature for each of the tomograms. Moving on, for our

Algorithm 2: Tomogram translation for spatial continuity through time

Input: Time-series of tomogram reconstructions I of size $T \times D \times H \times W$
Result: Align the tomogram reconstructions for t=$1, 2, \cdots, T$ to the one for t=0

for $i \leftarrow$ *0 to* T **do**
    // Addition of multiple slices to negate temporal changes in each tomogram
    slices $\leftarrow$ I[i, : ,-100:, :] // Crop the last 100 *xz-slices*
    s $\leftarrow$ sum(slices, axis=1) // Sum all 100 slices to one slice
    x_line $\leftarrow$ sum(s, axis=0) // Sum s along its columns
    z_line $\leftarrow$ sum(s, axis=1) // Sum s along its rows
    // Thresholding to negate any noise
    x_range $\leftarrow$ max(x_line) - min(x_line)
    z_range $\leftarrow$ max(z_line) - min(z_line)
    range $\leftarrow$ mean(x_range, z_range)
    x_line[x_line <(0.3*range)] $\leftarrow$ 0
    z_line[z_line <(0.3*range)] $\leftarrow$ 0
    // Calculating the center of ''brightness'' for the x and z axis
    x_mean $\leftarrow$ weighted_average(x_line, weights $\leftarrow (1, 2, \cdots, W)$)
    z_mean $\leftarrow$ weighted_average(z_line, weights $\leftarrow (1, 2, \cdots, D)$)
    X[i] $\leftarrow$ round(x_mean)
    Z[i] $\leftarrow$ round(z_mean)
    // Performing the appropriate translation of the tomogram reconstructions
    X[i] $\leftarrow$ X[i] - X[0]
    Z[i] $\leftarrow$ Z[i] - Z[0]
    Translate the i^{th} tomogram reconstruction in the time-series, by -X[i] along the *x-axis* and by -Z[i] along the *z-axis*
end

datasets in particular, foreground voxels (classes=$1, 2, \cdots, n$) are on average brighter than background voxels (class=0). By calculating the centers of "brightness" for the x and z axis in the summation of the 100 slices in each time, the inverse translation matrices $\boldsymbol{P}^{-1}_{1,2,\cdots,n}$ can be calculated. Since the summation of the 100 slices and the centers of "brightness" are temporally invariant, by tracking the changes of the last ones through time, it can lead to the calculation of the erroneous translations. After the calculation of the $\boldsymbol{P}^{-1}_{1,2,\cdots,n}$ they can be used in order to align all the tomograms of the time-series. In other datasets, depending on the characteristics of their foreground pixels (classes=$1, 2, \cdots, n$), other temporally invariant features might be worth detecting (like the center of "darkness" if the reverse is true, etc.). In general in order to alignment reconstructions of tomograms that are part of a time-series, "anchor" features has to be selected in order to measure the erroneous translations and then reverse them. Alternatively, using better experimental capturing techniques in the future that can overcome the observed random translations in the tomogram reconstructions may also resolve this issue.

After this operation all toxels (voxels through time) that have the same spatial coordinates, depicts the same location of the input sample through time, and therefore any class predictions for these groups of toxels depict meaningful temporal changes. In the next Section we will describe an unsupervised approach for improving 4D segmentations after individual 3D predictions in a time-series have been calculated. This approach, to the best of our knowledge, is a novel use of hidden Markov models (HMM) [455] in conjunction with 3D segmentation predictions generated by a CNN.

7.3 Methodology

As discussed in chapters 5 and 6, it is possible using a time-series' representative highly-sampled tomograms to create denoising and semantic segmentation networks that can individually handle the reconstruction of tomograms in a time-series. These representative, highly-sampled tomograms are collected either at the beginning, end or both the beginning and the end of a time-series of tomograms, when their slow process of collection is not detrimental on the temporal events the time-series tries to collect. Since they are not

captured during the progression temporal events, they can only offer a spatial prior for the denoising and segmentation of the undersampled tomograms in the time-series.

Naturally, the application of the alignment algorithm described in the previous Section 7.2 and the iterative application of the above networks to all tomograms of a time-series can offer semantically meaningful segmentation predictions for the whole time-series, represented as a 4D dataset. We hypothesise that additional improvement can be attained by comparing temporally previous and subsequent spatial predictions, since there must be a temporal continuity for the segmentation classes based on a high-order structure. Namely, due to the physical properties of the temporal events that take place, there are a number of outcomes that are expected and others that are not. For instance, the physical process of corrosion present in our datasets (see Section 3.2.1) postulates that the traces of magnesium within the sample naturally corrode due to the presence of salt-water. This leads to the formation of hydrogen bubbles, the disappearance of the magnesium (class=2) and its replacement with holes full of air (air pockets, class=3). On the other hand, temporal events are not expected for the background air (class=0); the air class would not be expected to be replaced with a different class throughout these time series. In other words, the transition from class 2 to class 3 is expected (ie. $P(X_t = 3|X_{t-1} = 2)$ is high, where X_t is a random variable representing a toxel's class at the t time-step), while the transition of class 0 to any other class is not expected (ie. $P(X_t = 1, 2, 3|X_{t-1} = 0)$ is low).

Additionally, the segmentation predictions after the application of a segmentation network have different levels of accuracy (certainty) for each class. For a toxel that belongs to a "difficult" class $X_t = n$ (that is comprised by spatially objects/instances, which have intricate shapes and are therefore difficult to accurately segment by a 3D CNN), the emission probability $P(X_t = n|E_t = k)$ that the CNN predicted class should in fact be a different one (ie. where $k \neq n$), might be high. On the other hand, for a toxel that belongs to a "easy" class $X_t = m$ (one that is not "difficult"), the emission probability $P(X_t = m|E_t = k)$ (where $k \neq m$) might be lower.

7.3.1 HMM-T: Hidden Markov model that uses inter-tomogram information from segmentation predictions

Based on the earlier mentioned transition and emission probabilities, we propose that the problem of refining spatial 3D segmentation predictions using previous and subsequent predictions can be modelled using a Dynamic Bayesian Network, namely a hidden Markov model (HMM) [455]. The final segmentation classes are derived from the HMM's (hidden) states, and the predicted classes from a segmentation CNN are provided as the observations/input to the HMM. As can be seen in Figure 7.1, a HMM is composed of two sets of probabilities, transition probabilities $P(X_t|X_{t-1})$ that determine the probabilities of (hidden) states transitioning from one to the other through time, and emission probabilities $P(X_t|E_t)$ that determine the probabilities of each (hidden) state being "emitted" for each of the observable states.

The $P(X_t|X_{t-1})$ probabilities that denote the transition probabilities from each (hidden) state to another through time form what is know as a Markov chain. They can be expressed using a $N \times N$ matrix $\boldsymbol{T}$, named the transition matrix, where N is the number of segmentation classes. On the other hand, the $P(X_t|E_t)$ probabilities express the likelihood of a segmentation class to be predicted as any other class by a segmentation network, forming a $N \times M$ matrix $\boldsymbol{O}$, named the emission matrix. Transition probabilities allow the model to capture the likelihood of toxels with the same spatial coordinates switching from one class label to another through time. Emission probabilities capture the likelihood of a toxel belonging to any output segmentation class based on the input segmentation class (output classes are the HMM hidden states for each time-step and input classes are the CNN predicted classes).

For the simple case, where the predicted classes from a segmentation CNN are the same in number as the (hidden) states (the segmentation classes inferred by the HMM), then $M = N$ and $\boldsymbol{O}$ is also $N \times N$. Furthermore, a HMM model also requires starting probabilities for each of its (hidden) states. They express the probabilities for each state to appear at $t = 0$ (assuming indexing starts with 0, t=$0, 1, 2, \cdots$), compensating for the absence of a previous state (there is no X_{-1}) and so the transition probabilities

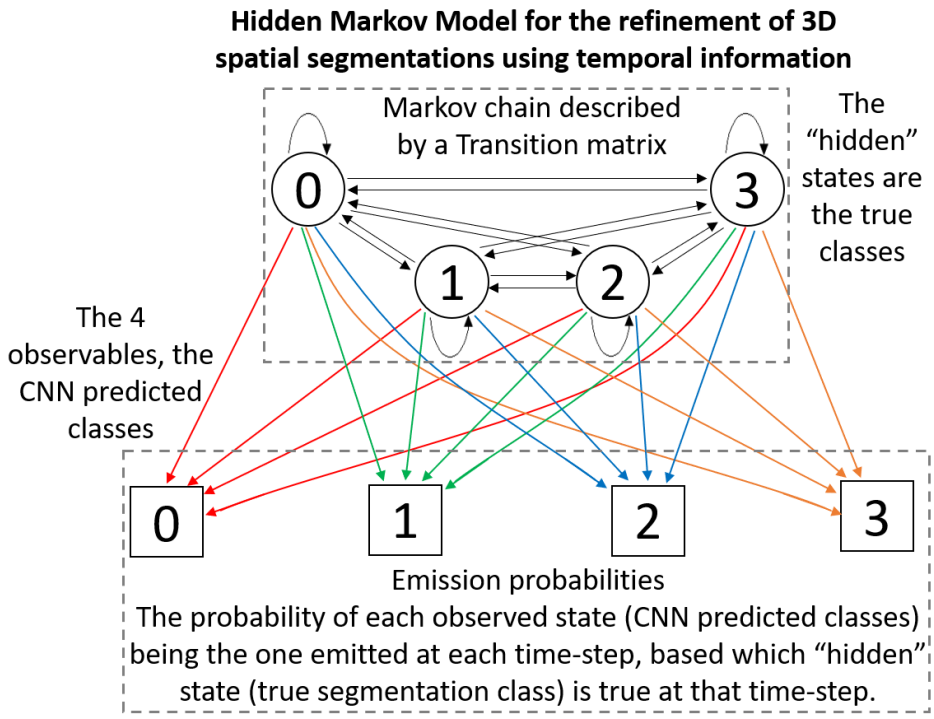


Figure 7.1: *Hidden Markov model for the refinement of 3D segmentation predictions after the application of a 3D segmentation CNN on each tomogram reconstruction of a time-series. The hidden Markov model (HMM-T) corrects the predictions of a time-step using temporal information; in other words using earlier and later 3D predictions. The black arrows represent transition probabilities from one class to the next through one time-step and the coloured arrows the probabilities of each true sample state to be inferred by our Stacked-DenseUSeg as any class prediction, including itself. The emission probabilities can be determined by the networks confusion matrix during testing, while the former transition probabilities can be empirically determined by an expert user (as in our case) or determined using a sufficient amount of user-annotated 4D data.*

cannot be used. These probabilities can be expressed by a vector $\boldsymbol{S}$ with size $1 \times N$. Provided that all HMM parameters are determined, using a forward prediction algorithm like the Maximum a posteriori estimation (MAP) or the Viterbi algorithm [456], the segmentation classes of each group of toxels with the same spatial coordinates can be determined. Namely, with the Viterbi algorithm that is used for our HMM models the most likely sequence of hidden states is being calculated the observations. The algorithm using dynamic programming and progressing calculations finds the most likely "path" or sequence of hidden states, therefore in our case the most likely segmentation class for each toxel. Since our proposed hidden Markov model use only temporal information from previous and subsequent segmentation predictions, we will refer to it as HMM-T in the following sections. This will be in contrast to our second proposed method described later,

which apart from temporal information also uses the CNN certainty for its predictions in other words how probable is each of the observations/input to the HMM.

7.3.2 HMM-TC: Hidden Markov model that uses inter-tomogram information and segmentation confidence from segmentation predictions

We also propose an alternative application of a hidden Markov Model, where the observable states are not directly the predicted classes of a 3D segmentation network. Instead, each prediction of the CNN network is further re-classified by the confidence of the network for said prediction. This confidence is the probability of the leading class, meaning that it the probability of the segmentation class that it has the highest probability according to the network. Since we are using the probability of the leading class for each of the networks predictions, this can be within the range $(\frac{1}{N}, 1)$, where N is the number of the segmentation classes. This is because in the case where the network is not confident about its prediction, the probability distribution between the different classes is close to uniform and so the $P(X = \textit{leading class}) \rightarrow \frac{1}{N}$. On the other hand, when it is very confident regarding its prediction $P(X = \textit{leading class}) \rightarrow 1$. Naturally, the probability of the leading class cannot be below $\frac{1}{N}$, since for the rest class probabilities to add up to 1, it would mean that there is another class with higher probability making it the leading class.

In order to introduce this confidence information to the emission matrix $\boldsymbol{O}$, the network's segmentation probability for the leading class is quantised based on a number of "bins" that equally split the range $(\frac{1}{N}, 1)$. Then each original observable states (same as in HMM-T) is split to new observable states that are equal to the number of "bins". For instance using b "bins" to split the $(\frac{1}{N}, 1)$ range and M observable states the total of the new observable states is bM. Figure 7.2 depicts the case where 4 observable classes (the network's prediction classes) are split by 3 "bins" $((25\%, 50\%), (50\%, 75\%), (75\%, 100\%))$ to create 12 new observable states. In the next sections we will refer to these 3 "bins" as low-confidence, mid-confidence, and high-confidence.

In order to differentiate the workings of this model from from HMM-T, we will refer to it as HMM-TC (with C representing Confidence information).

7.3.3 Selection of hidden Markov model parameters for the segmentation of time-series of tomogram reconstructions

As described in Section 3.1, a time-series of X-ray micro CTs comprises a collection of multiple tomograms that capture temporal events. These tomograms are undersampled, meaning that their SNR is small, making it very difficult to manually segment their reconstructions. With the use of our proposed methods in chapters 4, 5, 6 we introduced supervised methods that increase of their SNR and provide semantic segmentation indi-

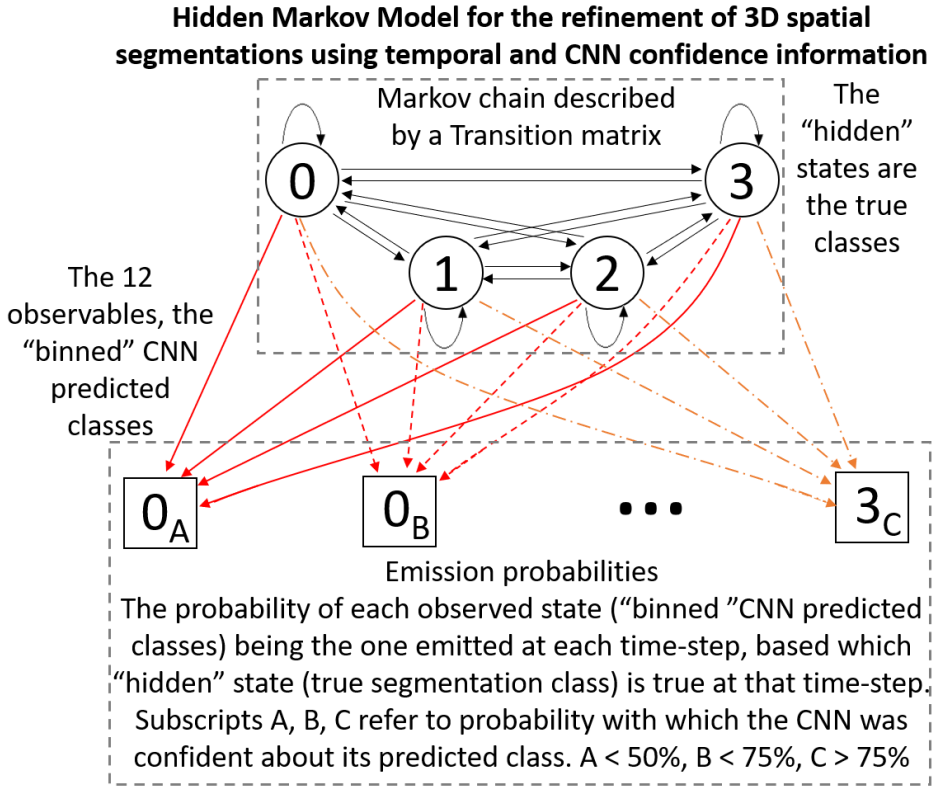


Figure 7.2: *Hidden Markov model for the refinement of 3D segmentation predictions after the application of a 3D segmentation CNN on each tomogram reconstruction of a time-series. The hidden Markov model corrects the predictions of a time-step using temporal information, in other words using earlier and later 3D predictions, and also using the prediction probabilities offered by the CNN segmentation network which produced the 3D predictions (HMM-TC). The black arrows represent transition probabilities from one class to the next through one time-step and the coloured arrows the probabilities of each true class to be inferred by our Stacked-DenseUSeg as any other including itself, with 3 levels ("bins") of confidence. The latter ones can be determined by the networks confusion matrix during testing, while the former are could be empirically determined by the user like in our case or determined using a small amount of user-annotated 4D data.*

vidually for each reconstruction of the tomograms in the time-series. However, during this process the supervision provided to our methods utilised annotations of highly-sampled tomograms and not fully or partial annotations of tomogram time-series. This means that inter-tomogram information is not utilised since there are not 4D annotations for supervision or training of our approaches. Additionally, such annotations with the same level of quality as annotations of highly-sampled tomograms are practically impossible to acquire due to the extremely low level of the SNR of the time-series' tomogram reconstructions. Sadly, our approaches, though able to increase their SNR, they are still not able to increase to the same level as in highly-sampled tomograms. Therefore, it is a preventing factor for the implementation of supervised segmentation approaches with utilisation of inter-tomogram information, since any attempt to fully or partially annotate the time-series of tomograms will potentially lead to erroneous and biased annotations.

Based on the above, the proposed hidden Markov models presented in Sections 7.3.1 and 7.3.2 are used without the need for annotated data and model training. Values for their transition probabilities described by matrix $\boldsymbol{T}$ are empirically determined based on the expected physical behaviour of our datasets (see Section 3.2 and [418]). We empirically selected the transition probability of magnesium to air pockets to be approximately 10%, and we also introduce some small transition probabilities between all the other classes. The value of 10% is derived from observations of the low SNR tomogram reconstructions of the time-series, where the population difference of magnesium and air pocket instances was approximated. This minor transitions are in place for the very rare cases where the transition between two classes is a product of minor misalignments between the tomograms in the time-series or thermal expansions and contractions. Matrix 7.2 displays the final transition matrix $\boldsymbol{T}$ for the HMM-T and HMM-TC models. It is important to note that as we explained in Section 5.4.2, class with Label 1, the base material while is comprised of both water and aluminium, water appear in very small traces around the edge of the aluminium pin as it imaged in our data. For this reason in the following Section we are going to refer to it simply as aluminium.

$$\boldsymbol{T} = \begin{array}{c} \begin{matrix} \textit{Air} & \textit{Aluminium} & \textit{Magnesium} & \textit{Air Pockets} \end{matrix} \\ \begin{bmatrix} 0.998 & 0.001 & 0.001 & 0 \\ 0.001 & 0.996 & 0.001 & 0.002 \\ 0.001 & 0.001 & 0.8982 & 0.0998 \\ 0 & 0.001 & 0.001 & 0.998 \end{bmatrix} \end{array} \begin{array}{c} \textit{Air} \\ \textit{Aluminium} \\ \textit{Magnesium} \\ \textit{Air Pockets} \end{array} \tag{7.2}$$

To set the emission matrix $\boldsymbol{O_1}$, the confusion matrix $\boldsymbol{C_1}$ (see matrix 7.3) produced during the testing of the segmentation CNN is used. The confusion matrix can provide a clear picture regarding emission distributions, the probabilities for each class to be misinterpreted by the network as any other class. Namely, every row of the Confusion Matrix is divided by the sum of its cells, and matrix 7.4 is the emission matrix of the HMM-T.

$$\boldsymbol{C_1} = \begin{array}{c} \begin{matrix} \textit{Predicted Air} & \textit{Predicted Aluminium} & \textit{Predicted Magnesium} & \textit{Predicted Air Pockets} \end{matrix} \\ \begin{bmatrix} 27171100 & 265330 & 35 & 4 \\ 148082 & 239760000 & 1747349 & 18310 \\ 115 & 4118587 & 5080109 & 633 \\ 10 & 32301 & 227 & 52233 \end{bmatrix} \end{array} \begin{array}{c} \textit{Air} \\ \textit{Aluminium} \\ \textit{Magnesium} \\ \textit{Air Pockets} \end{array} \tag{7.3}$$

$$\boldsymbol{O_1} = \begin{array}{c} \begin{matrix} \textit{Predicted Air} & \textit{Predicted Aluminium} & \textit{Predicted Magnesium} & \textit{Predicted Air Pockets} \end{matrix} \\ \begin{bmatrix} \mathbf{9.946 \cdot 10^{-1}} & 5.420 \cdot 10^{-3} & 4.209 \cdot 10^{-6} & 3.660 \cdot 10^{-7} \\ 1.087 \cdot 10^{-3} & \mathbf{9.819 \cdot 10^{-1}} & 1.687 \cdot 10^{-2} & 1.323 \cdot 10^{-4} \\ 5.126 \cdot 10^{-6} & 2.559 \cdot 10^{-1} & \mathbf{7.440 \cdot 10^{-1}} & 3.325 \cdot 10^{-5} \\ 5.620 \cdot 10^{-5} & 2.572 \cdot 10^{-1} & 8.893 \cdot 10^{-3} & \mathbf{7.338 \cdot 10^{-1}} \end{bmatrix} \end{array} \begin{array}{c} \textit{Air} \\ \textit{Aluminium} \\ \textit{Magnesium} \\ \textit{Air Pockets} \end{array} \tag{7.4}$$

As can be seen in the bold text, for each of the observable states (derived from the

predicted classes from the CNN), the state/class that has the higher probability to emit them is the same as the observable states. This essentially, means that if the network predicts a certain class for a toxel this has the highest probability of being its actual "true" class.

For the second proposed hidden Markov model that incorporates the CNN's confidence (HMM-TC) via the increase of the observable states, the emission matrix is also derived from the confusion matrix of the CNN during testing. During the network's testing, the calculation of the confusion matrix is coded to distinguish between new observable states and produce a rectangular confusion matrix. This means that the confusion matrix contains the instances that any class (air, aluminium, etc.) is predicted by the network as any of new predicted classes (low-confidence air, mid-confidence air, $\cdots$, high-confidence air pockets) that are created after the quantisation of the leading class probability to 3 "bins". Figure 7.3 displays the new observations classes (quantised CNN predictions) and matrix 7.6 displays $\boldsymbol{O_2}$ the new emission matrix based on matrix 7.5 which is the new confusion matrix $\boldsymbol{C_2}$.

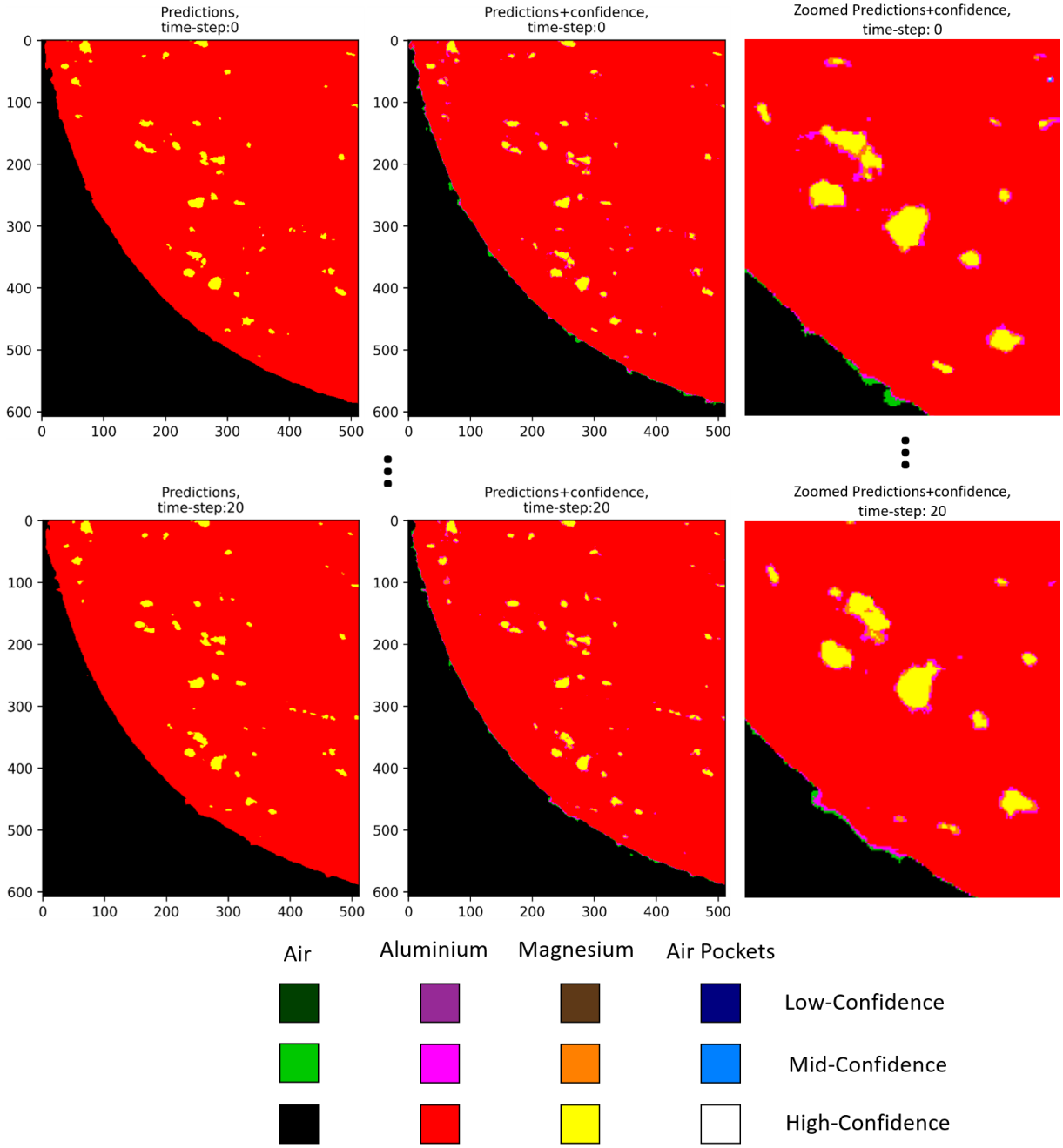


Figure 7.3: *Segmentation predictions of the RealTS1 time-series of tomograms after the use of Stacked-DenseUSeg and the confidence of the network for said predictions. The predictions are a time-series of 21 tomograms for the slice of height 1205 (out of 2160) (y-axis) and for the ROI [1256:1864, 450:962] (xz-plane). For the right figures there is a colour palette that displays the different classes and confident of the network these classes.*

$$
\begin{array}{ccc|c}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Air} & \textit{Air} & \textit{Air} & \\
\hline
615 & 58203 & 27112284 & \textit{Air} \\
1055 & 64192 & 200083 & \textit{Aluminium} \\
5 & 23 & 7 & \textit{Magnesium} \\
2 & 1 & 1 & \textit{Air Pockets}
\end{array}
$$

$$
\begin{array}{ccc|c}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Aluminium} & \textit{Aluminium} & \textit{Aluminium} & \\
\hline
650 & 45731 & 101700 & \textit{Air} \\
5503 & 1001348 & 238755654 & \textit{Aluminium} \\
998 & 207744 & 1538605 & \textit{Magnesium} \\
63 & 2552 & 15695 & \textit{Air Pockets}
\end{array}
$$

$$
\boldsymbol{C_2} = \qquad (7.5)
$$

$$
\begin{array}{ccc|c}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Magnesium} & \textit{Magnesium} & \textit{Magnesium} & \\
\hline
17 & 55 & 43 & \textit{Air} \\
4407 & 910667 & 3203510 & \textit{Aluminium} \\
1016 & 269887 & 4809208 & \textit{Magnesium} \\
26 & 235 & 372 & \textit{Air Pockets}
\end{array}
$$

$$
\begin{array}{ccc|c}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Air Pockets} & \textit{Air Pockets} & \textit{Air Pockets} & \\
\hline
2 & 7 & 1 & \textit{Air} \\
134 & 5614 & 26553 & \textit{Aluminium} \\
8 & 74 & 145 & \textit{Magnesium} \\
42 & 2354 & 49837 & \textit{Air Pockets}
\end{array}
$$

Using matrix 7.6 as the emission matrix of HMM-TC, during preliminary testing we received unsatisfactory results as they are shown in Figure 7.4, as the model wrongfully added new Air Pocket toxels.

As it can be seen from matrix 7.6, the observable states, sometimes are emitted with higher probability by classes that are not the leading class. For example Mid-Confidence Aluminium is emitted with higher probability by Air Pockets. This because the probability of Air Pocket being predicted as any observable class (CNN predicted classes), is the result of how many times this occurred divided by the number of Air Pocket voxels present in the ground truth during testing (voxels not toxels because we refer to the testing dataset of Stacked-DenseUSeg that it is a 3D volume). The denominator of course (the sum of all occurring Air Pocket voxels) is significantly smaller than the denominators in other states/classes (the sum of all occurring Air, Aluminium, etc voxels) since Air Pockets are more rare.

$$
\boldsymbol{O_2} = \begin{array}{ccc|c}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Air} & \textit{Air} & \textit{Air} & \\
\hline
2.251 \cdot 10^{-5} & \mathbf{2.130 \cdot 10^{-3}} & \mathbf{9.924 \cdot 10^{-1}} & \textit{Air} \\
4.321 \cdot 10^{-6} & 2.629 \cdot 10^{-4} & 8.194 \cdot 10^{-4} & \textit{Aluminium} \\
7.323 \cdot 10^{-7} & 3.369 \cdot 10^{-6} & 1.025 \cdot 10^{-6} & \textit{Magnesium} \\
\mathbf{2.810 \cdot 10^{-5}} & 1.405 \cdot 10^{-5} & 1.405 \cdot 10^{-5} & \textit{Air Pockets}
\end{array}
$$

$$
\begin{array}{ccc|c}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Aluminium} & \textit{Aluminium} & \textit{Aluminium} & \\
\hline
2.379 \cdot 10^{-5} & 1.674 \cdot 10^{-3} & 3.723 \cdot 10^{-3} & \textit{Air} \\
2.254 \cdot 10^{-5} & 4.101 \cdot 10^{-3} & \mathbf{9.778 \cdot 10^{-1}} & \textit{Aluminium} \\
1.462 \cdot 10^{-4} & 3.043 \cdot 10^{-2} & 2.253 \cdot 10^{-1} & \textit{Magnesium} \\
\mathbf{8.851 \cdot 10^{-4}} & \mathbf{3.585 \cdot 10^{-2}} & 2.205 \cdot 10^{-1} & \textit{Air Pockets}
\end{array}
$$

$$
\begin{array}{ccc|c}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Magnesium} & \textit{Magnesium} & \textit{Magnesium} & \\
\hline
6.223 \cdot 10^{-7} & 2.013 \cdot 10^{-6} & 1.574 \cdot 10^{-6} & \textit{Air} \\
1.805 \cdot 10^{-5} & 3.730 \cdot 10^{-3} & 1.312 \cdot 10^{-2} & \textit{Aluminium} \\
1.488 \cdot 10^{-4} & \mathbf{3.953 \cdot 10^{-2}} & \mathbf{7.044 \cdot 10^{-1}} & \textit{Magnesium} \\
\mathbf{3.653 \cdot 10^{-4}} & 3.301 \cdot 10^{-3} & 5.226 \cdot 10^{-3} & \textit{Air Pockets}
\end{array}
$$

$$
\begin{array}{ccc|c}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Air Pockets} & \textit{Air Pockets} & \textit{Air Pockets} & \\
\hline
7.321 \cdot 10^{-8} & 2.562 \cdot 10^{-7} & 3.660 \cdot 10^{-8} & \textit{Air} \\
5.488 \cdot 10^{-7} & 2.299 \cdot 10^{-5} & 1.087 \cdot 10^{-4} & \textit{Aluminium} \\
1.172 \cdot 10^{-6} & 1.084 \cdot 10^{-5} & 2.124 \cdot 10^{-5} & \textit{Magnesium} \\
\mathbf{5.901 \cdot 10^{-4}} & \mathbf{3.307 \cdot 10^{-2}} & \mathbf{7.002 \cdot 10^{-1}} & \textit{Air Pockets}
\end{array}
\tag{7.6}
$$

Naturally, the high probabilities are also intensified by the numerators, namely the Air Pocket voxels predicted as low, mid or high confidence voxels of other classes. The misi-classification of "difficult" classes (Magnesium, Air Pockets), is high because the ground truths used during the CNN's testing are high quality annotations from a highly-sampled tomogram which details them very accurately. Due to the undersampling of projections in the input, it is difficult for the network to accurately segment these "difficult" classes, since they are spatially small, intricate and there is a significant loss of their spatial information by the undersampling. During the CNN's training, this is not an issue since

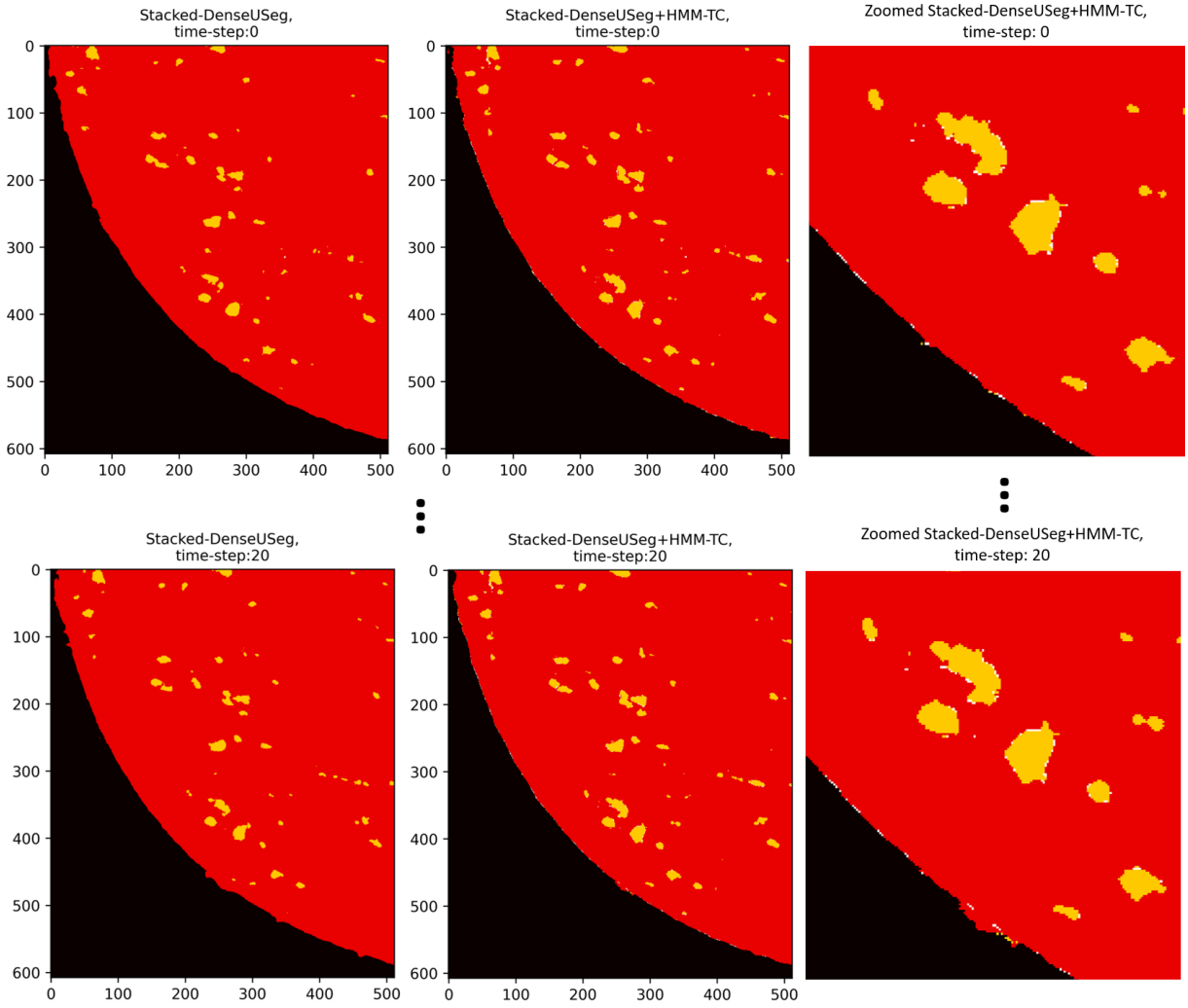


Figure 7.4: *Segmentation predictions of the RealTS1 time-series of tomograms after the use of Stacked-DenseUSeg and Stacked-DenseUSeg+HMM-TC. The predictions are a time-series of 21 tomograms for the slice of height 1205 (out of 2160) (y-axis) and for the ROI [1256:1864, 450:962] (xz-plane). As it can be seen in the right figures, HMM-TC using the emission matrix 7.6 directly based on Stacked-DenseUSeg confusion matrix, attempts to erroneously restore missing Air Pockets that were removed from Stacked-DenseUSeg's input due to projection downsampling.*

the network is pressured to not only segment easily distinguishable objects, but to also potentially restore missing information from the input, and get even higher accuracy. Unfortunately, all missing information between the undersampled and highly-sampled version of the input cannot be restored completely, and this is also true for very large and deep CNNs. CNNs perform predictions after training with a spatial prior and using the provided input during inference. In the provided tomograms, as mentioned earlier there are segmentation instances that are spatially small and have intricate shapes which do not follow any specific underlying pattern. Their spatial geometry would be easily distinguishable if the tomograms were highly-sampled, but due to the low number of projections their reconstructions cannot provide sufficient information the CNNs, which irregardless of their size, cannot predict with a very high level of accuracy the spatial geometry of the earlier segmentation instances in order to facilitate their accurate segmentation. An extreme version of this is when the undersampling essentially erases such instances in the resulting reconstructions and networks cannot restore them via prediction since such act would essentially be guesswork. In general, there is a level of information that cannot be restored after the undersampling of projections and the remaining information is not sufficient to enable the accurate prediction and restoration of the missing information. This means that there is a number of wrongful network predictions that are not caused by the networks' ineffectiveness, but due to the extreme informational loss. Therefore, there is a number toxels that are "correctly" misclassified (some Air Pockets predicted as Mid-Confidence Aluminium) since there is not sufficient information in the provided input to the network to restore missing information. For these reason, the earlier probabilities of the emission matrix are manually altered to facilitate the existence of these "correctly" misclassified toxels. Matrix 7.7 displays the changed emission matrix $\boldsymbol{O_2^*}$. During preliminary testing, conversion of the confusion matrix to the emission matrix using balancing factors were also attempted. However, the results were heavily biased towards the "easy" classes, even for toxels that after visual inspection should definitely be classified differently.

$$
\begin{array}{cccc}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Air} & \textit{Air} & \textit{Air} & \\
\left[\begin{array}{ccc|c}
\mathbf{2.251 \cdot 10^{-5}} & \mathbf{2.130 \cdot 10^{-3}} & \mathbf{9.924 \cdot 10^{-1}} & \textit{Air} \\
4.321 \cdot 10^{-6} & 2.629 \cdot 10^{-4} & 8.194 \cdot 10^{-4} & \textit{Aluminium} \\
7.323 \cdot 10^{-7} & 3.369 \cdot 10^{-6} & 1.025 \cdot 10^{-6} & \textit{Magnesium} \\
2.810 \cdot 10^{-7} & 1.405 \cdot 10^{-5} & 1.405 \cdot 10^{-7} & \textit{Air Pockets}
\end{array}\right. & & &
\end{array}
$$

$$
\begin{array}{cccc}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Aluminium} & \textit{Aluminium} & \textit{Aluminium} & \\
\left[\begin{array}{ccc|c}
2.141 \cdot 10^{-5} & 1.674 \cdot 10^{-3} & 3.723 \cdot 10^{-3} & \textit{Air} \\
\mathbf{2.254 \cdot 10^{-5}} & \mathbf{4.101 \cdot 10^{-3}} & \mathbf{9.778 \cdot 10^{-1}} & \textit{Aluminium} \\
2.193 \cdot 10^{-5} & 4.077 \cdot 10^{-3} & 2.253 \cdot 10^{-1} & \textit{Magnesium} \\
8.851 \cdot 10^{-6} & 3.585 \cdot 10^{-4} & 2.205 \cdot 10^{-2} & \textit{Air Pockets}
\end{array}\right. & & &
\end{array}
$$

$$
\boldsymbol{O_2^*} = \tag{7.7}
$$

$$
\begin{array}{cccc}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Magnesium} & \textit{Magnesium} & \textit{Magnesium} & \\
\left[\begin{array}{ccc|c}
6.223 \cdot 10^{-7} & 2.013 \cdot 10^{-6} & 1.574 \cdot 10^{-6} & \textit{Air} \\
1.805 \cdot 10^{-5} & 3.730 \cdot 10^{-3} & 1.312 \cdot 10^{-2} & \textit{Aluminium} \\
\mathbf{1.488 \cdot 10^{-4}} & \mathbf{3.953 \cdot 10^{-2}} & \mathbf{7.044 \cdot 10^{-1}} & \textit{Magnesium} \\
7.305 \cdot 10^{-7} & 3.301 \cdot 10^{-3} & 5.226 \cdot 10^{-3} & \textit{Air Pockets}
\end{array}\right. & & &
\end{array}
$$

$$
\begin{array}{cccc}
\textit{Low-Conf} & \textit{Mid-Conf} & \textit{High-Conf} & \\
\textit{Air Pockets} & \textit{Air Pockets} & \textit{Air Pockets} & \\
\left[\begin{array}{ccc|c}
7.321 \cdot 10^{-8} & 2.562 \cdot 10^{-7} & 3.660 \cdot 10^{-8} & \textit{Air} \\
5.488 \cdot 10^{-7} & 2.299 \cdot 10^{-5} & 1.087 \cdot 10^{-4} & \textit{Aluminium} \\
1.172 \cdot 10^{-6} & 1.084 \cdot 10^{-5} & 2.124 \cdot 10^{-5} & \textit{Magnesium} \\
\mathbf{5.901 \cdot 10^{-4}} & \mathbf{3.307 \cdot 10^{-2}} & \mathbf{7.002 \cdot 10^{-1}} & \textit{Air Pockets}
\end{array}\right. & & &
\end{array}
$$

Finally regarding the starting probabilities, temporally there is not a cause that sets any state/class more plausible at $t = 0$. While some states/classes are more frequent due to their population in the tomogram reconstructions, this is a class imbalance true for every moment and not specifically for $t = 0$. For this reason, we elected starting probabilities that have a uniform distribution between the different states/classes. Finally we selected the Viterbi algorithm [456] as the one used to refine the input predictions. It is important to mention here that algorithm receives as input 1D sequences of toxels that have the same spatial coordinates. This means that our proposed HMM models take into account only temporal information, but ignores the spatial information. Potentially, spatial information could be introduced to the HMMs by further increasing the number of observable states. The new observable states would also encode information about the predicted class of a voxel's spatial neighbors. However, this can potentially massively increase the number of observable states in our HMM models, leading to large inference times and making the HMM models impractical in a real-world setting. Fortunately, spatial information is heavily utilised by the 3D CNN (Stacked-DenseUSeg in our case) that generated the segmentation predictions. This allows us to develop HMM models that only refine the segmentations along the time-axis, without detrimental effects to the accuracy of the refined segmentations.

Furthermore, in an effort to speed up our HMM-T and HMM-TC models we use a computational shortcut. Namely, for groups of toxels through time where the observable state is the same, which is very likely for large 4D datasets with temporal events that transpire slowly, we use to the emission matrix to assign the same class to these toxels. Specifically, we look up using the Emission matrix which class emits this observable state with the highest probability. Then we assign the relevant class to these toxels. This shortcut allows skipping the Viterbi algorithm for numerous toxels, saving a lot of computational time. We can then focus the Viterbi algorithm on toxels which are known to change state over time. After having described both the methodology and the elected parameters for our proposed HMM models, in the following Section we proceed with the experiments that we preformed on our models.

7.4 Experiments

In this Section, we will present experiments using our proposed HMM models, regarding the refinement of multiple undersampled tomogram segmentations that are part of a time-series (*RealTS1* as presented in Section 3.2.2) [420] predicted by our Stacked-DenseUSeg network. Namely, we will present:

1. Qualitative results for the HMM-T and HMM-TC models by visually assessing how they refined the input predictions.
2. Behavioural experiments regarding the refinement performed by our HMM models. Specifically, we will only use the HMM-T model as it easy to understand its behaviour (less parameters) under the tested conditions.
3. Quantitative experiments, that provide quantifiable evidence regarding the accuracy improvement after the use of our HMM models. These experiments will measure their ability to correct temporal inconsistencies and improve the segmentation predictions.
4. Sensitivity experiments, that investigate the sensitivity of the manually elected transition probabilities. Namely, we will only use the HMM-T model as it easy to understand its behaviour (less parameters) under the tested conditions.

7.4.1 Experiment 1: Qualitative assessment of the HMM predictions

As it can be seen in Figure 7.5 after the execution of our HMM models the refined predictions remain relatively similar to our Stacked-DenseUSeg's predictions. One indication of their refinement can be seen in mark $\boldsymbol{I_1}$ of the Figure, where the jagged edges around the main structure in the Stacked-DenseUSeg's predictions are refined using the temporal information of previous and subsequent segmentation predictions in the time-series, providing a more coherent boundary.

To probe deeper in to the effects of our HMMs, we looked into the Stacked-DenseUSeg's predictions for certain toxels through time. The toxels examined do not belong in one class

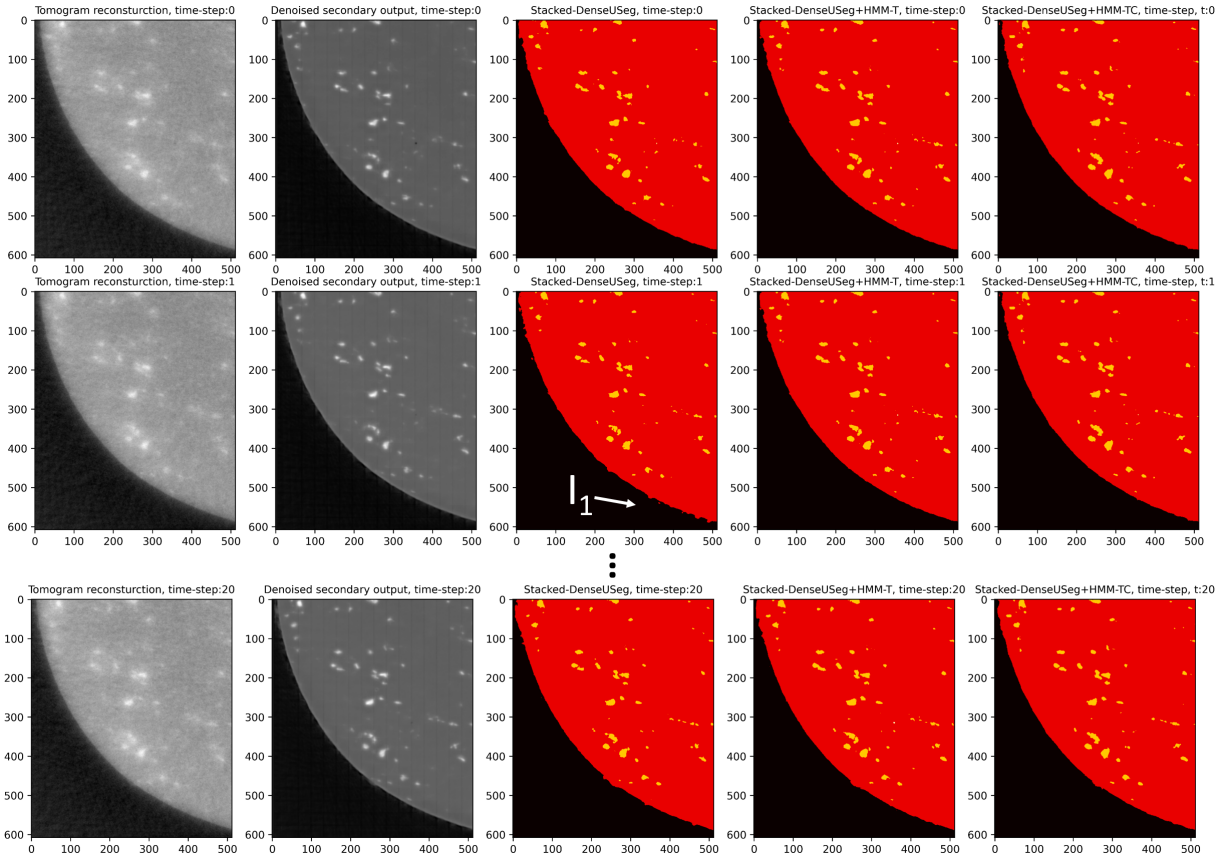


Figure 7.5: *Segmentation predictions of the RealTS1 time-series of tomograms after the use of Stacked-DenseUSeg, Stacked-DenseUSeg + HMM-T, Stacked-DenseUSeg + HMM-TC. The predictions are a time-series of 21 tomograms for the slice of height 1205 (out of 2160) (y-axis) and for the ROI [1256:1864, 450:962] (xz-plane). As it can be seen from mark* $\boldsymbol{I_1}$*, the HMM models corrected the jagged edges of the Stacked-DenseUSeg predictions. In general the changes are very subtle, as expected, meaning that they work to refine the predictions rather than dramatically change them.*

through time nor they have a simple and single transition between two classes. Examples of such "high transitioning" toxels have been selected for analysis and can be seen in Figure 7.6. In the Figure, there are two examples of "edge toxels", meaning that they are close to an edge of an object that belongs to one class, while its background belongs to another class. As can be seen with the help of spatially neighboring toxels through time, the class predictions for these toxels oscillate between two classes; the edge is effectively noisy and boundary toxels switch class over time accordingly. The segmentation network is uncertain (segmentation probabilities are close to 50-50 between two classes) and random noise present may eventually dictate the final prediction at each time-step. However, after the application of our HMM models the predictions between the classes are much more consistent over time. Furthermore, as it can be seen the predictions after the use of the

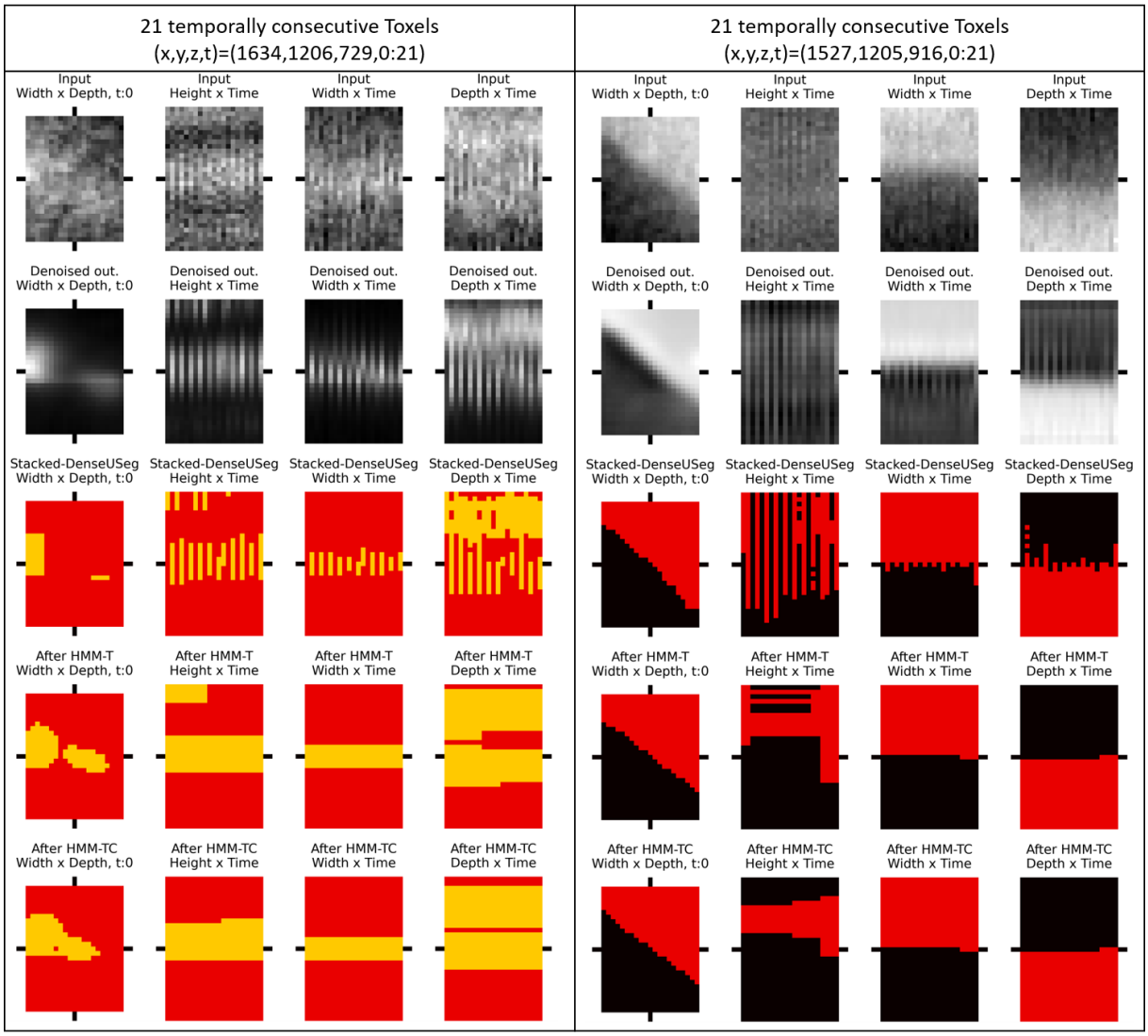


Figure 7.6: *Comparison of segmentations of 2 groups of temporally consecutive toxels and their neighboring toxels (both spatially and temporally). The subfigures display predictions through the Width-Depth plane ($xz-$plane), Height-Time plane ($yt-$plane), Width-Time plane ($xt-$plane) and Depth-Time plane ($zt-$plane) respectively, after the use of Stacked-DenseUSeg (**3rd row**), Stacked-DenseUSeg+HMM-T (**4th row**) and Stacked-DenseUSeg+HMM-TC (**5th row**). The black bars indicate that the toxels in question are the ones in the middle and the black crosshair point to the central toxel that belongs to first time-step ($t = 0$). From the subfigures, it can be deduced that the HMM models solve the temporal inconsistencies in the Stacked-DenseUSeg's predictions. HMM-TC seems that it does not allow many transitions, potentially due to the utilisation of Stacked-DenseUSeg's prediction confidence in its emission matrix.*

HMM-TC are even more "strict", which can potentially be attributed to the incorporated confidence information from the Stacked-DenseUSeg, allowing the HMM-TC to have more informed output predictions. In the next Section, we will present the results of particular behavioral test scenarios that we ran on our HMM-T model to examine its behaviour under specific conditions.

7.4.2 Experiment 2: Behavioural experiments using HMM-T

In an effort to further examine the accuracy and behaviour of our HMM models we performed a number of experiments where specific toxels have been manually assigned to certain classes through time. Then we investigate how our HMM models handled the input predictions. However, due to the similar structure of the HMMs and the ease of understanding HMM-T behaviour due to less parameters, we only used this model for the behavioural tests.

As can be seen in Figure 7.7 for our first experiment we replaced all the predictions in a 2D slice for one time-step with a class 1 label (Aluminium). After the application of the HMM-T, it can be seen that the model recovered almost all the information that was missing. Naturally, this task becomes easier with high temporal resolution, or in other words, when for each time-step any temporal changes progress slowly. Nevertheless, this experiments provides an indication about the benefits of using temporally adjacent predictions to better inform and refine the current predictions. As well as being informative in its own right towards the understanding of the model, this synthesised scenario could in fact occur in real life if data capture fails entirely for one time step, for example.

We next decided to investigate the effects of our empirically determined transition probabilities on the HMM-T segmentation behaviour. For this experiment, toxels are deliberately assigned to either one of two classes, Magnesium or Air Pockets, and at different time points they transition between them. The elected transition probabilities (see matrix 7.2) favours Magnesium to Air Pockets class changes, but not the opposite. This is relevant since according to the expected chemical reactions that occur in out datasets [418], Magnesium corrodes and escapes as gas, leaving in its place Air Pockets. The ultimate goal of the imaging is to measure the rate at which corrosion progresses under different conditions. Because of this, accurate prediction of transitions between the two classes is essential. As it can be seen in Figure 7.8, the assigned toxels are placed in the time-series predictions made by the Stacked-DenseUSeg. Furthermore by observing Figure 7.9, toxels of Magnesium (yellow label) are slowly changing into Air Pocket toxels. The spatial geometry of the assigned toxels is not significant since the HMM models process

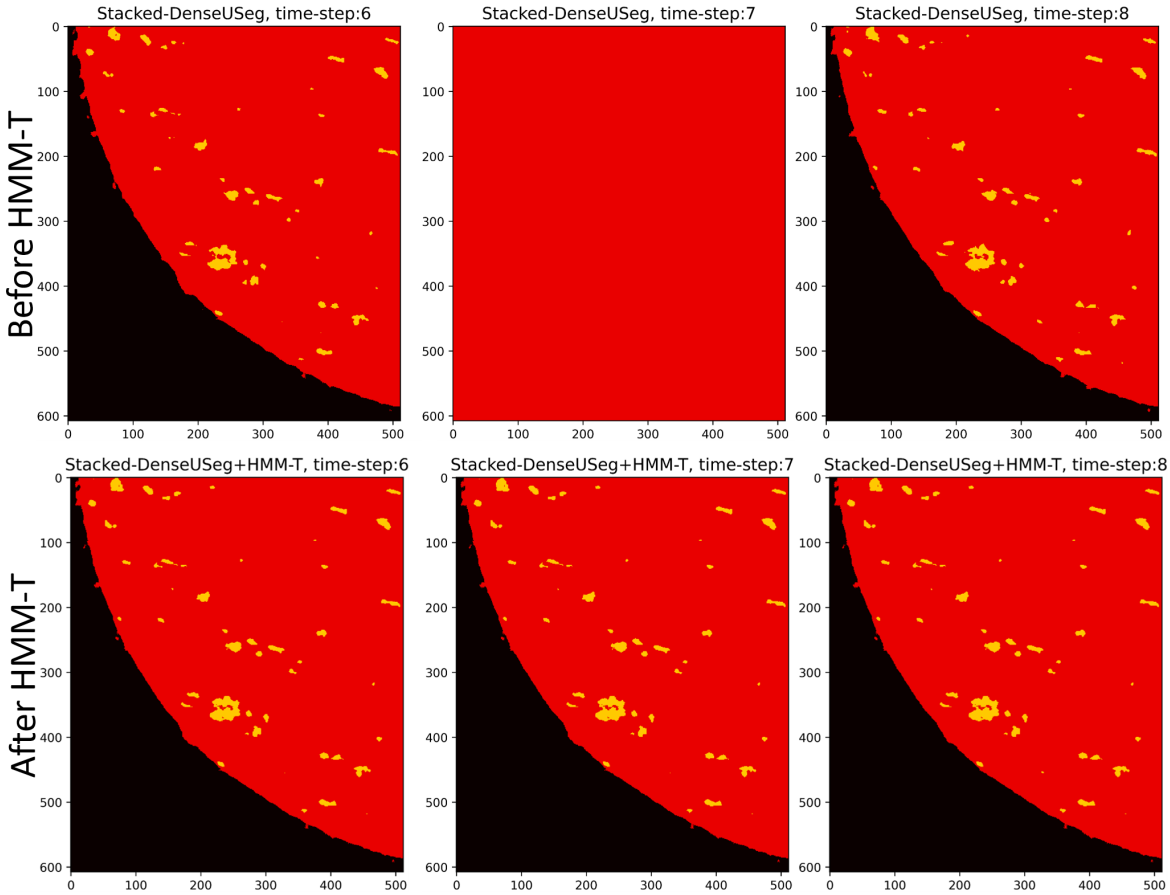


Figure 7.7: *Behavioral test of the HMM-T for the restoration of information missing from a time-step. In this figure the predictions of the slice from the* 7^{th} *tomogram in the time-series have been replaced completely with toxels of Aluminium (red label). The slice displayed is the* 1190^{th} *(out of 2160) of the height dimension (y-axis), for the ROI [1256:1864, 450:962] (xz-plane). As can be seen, annotations for missing time-steps can be recovered with what looks to be a good level of accuracy*

separately groups of toxels with the same spatial coordinates through time (this means spatial information is not included) as a 1D time-sequence of states. In Figures 7.9(**i**) and 7.9(**q**) it can be seen that in cases where the toxels' change from Magnesium to Air Pockets occurs in the unexpected direction, the HMM-T corrects these predictions and keeps the toxels assigned as Air Pockets which is the expected behaviour.

In addition, we also tested HMM-T behaviour with the reverse assignments of toxels and the results are shown in Figure 7.10. This means that HMM-T on the one hand should deter changes from Air Pockets (class=3) to Magnesium (class=2), but on the other should allow them if based on the observations there is overwhelming evidences that such transitions happened. As can be seen from the Figure, due to the existence of multiple observations and the high emission probabilities when the emitted/predicted class and the

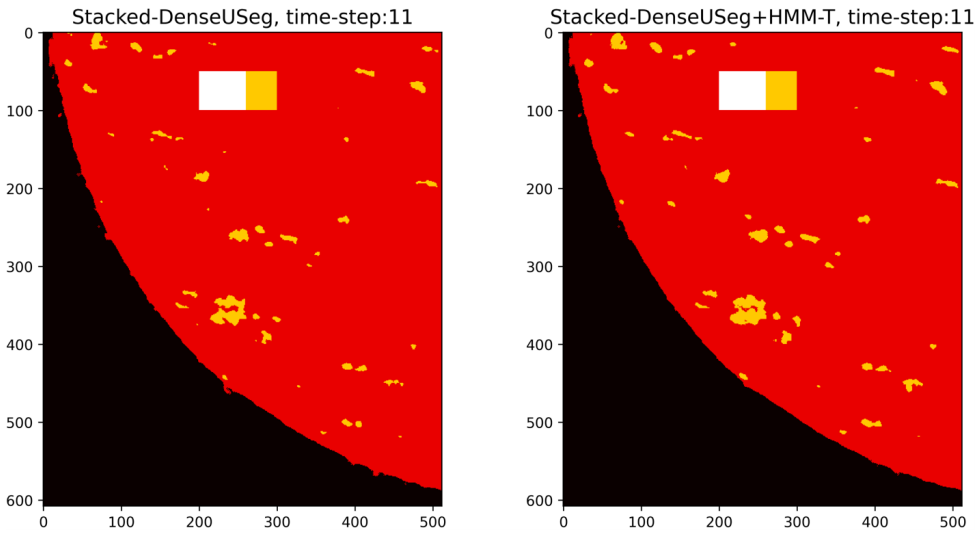


Figure 7.8: *Behavioral test of the HMM-T for the prediction of the Magnesium (class=2, yellow) turning into air pockets (class=3, white) through time. For this test the HMM-T is being used. In this figure the predictions of the* 11^{th} *tomogram in the time-series are being displayed. The slice displayed is the 1190 (out of 2160) of the height dimension (y-axis), for the ROI [1256:1864, 450:962] (xz-plane)*

same with the output classes, the HMM-T model eventually allows improbable transitions (Magnesium to Air Pockets has probability 0.0998 but Air Pockets to Magnesium has 0.001, see matrix 7.2). Additionally, it is important to mention that the probability of a toxel actually being Magnesium when it is emitted/predicted to be Magnesium, is higher compared to when the same thing happens with Air Pockets (see matrix 7.4). This is due to our Stacked-DenseUSeg be able to better predict Magnesium compared to Air Pockets, which is expressed via its confusion matrix and eventually on the model's emission matrix. Nevertheless, Figures 7.10(**e**), (**l-n**), (**u**), display the effort of our model to prevent the Air Pocket to Magnesium transition. Especially in Figures 7.10(**l-m**), the transition of Air Pockets to Magnesium is delayed/prevented since in the later Figure 7.10(**n**) there are still evidence for Air Pockets. This means that the Viterbi algorithm, based on the transition probabilities, will allow the improbable transition of Air Pocket to Magnesium, but in a later time (at $t = 16$ as shown from Figure 7.10(**p**)) because there still evidences/observations of Magnesium (at $t = 14, 15$ as shown in Figure 7.10(**n-o**)). It will also not allow 3 transitions (Air Pocket to Magnesium then to Air Pocket and then again to Magnesium, at $t = 13-16$ as shown in Figures 7.10(**m-p**)) since this sequence of events is highly unlikely (two improbable transitions and one probable are less likely than just

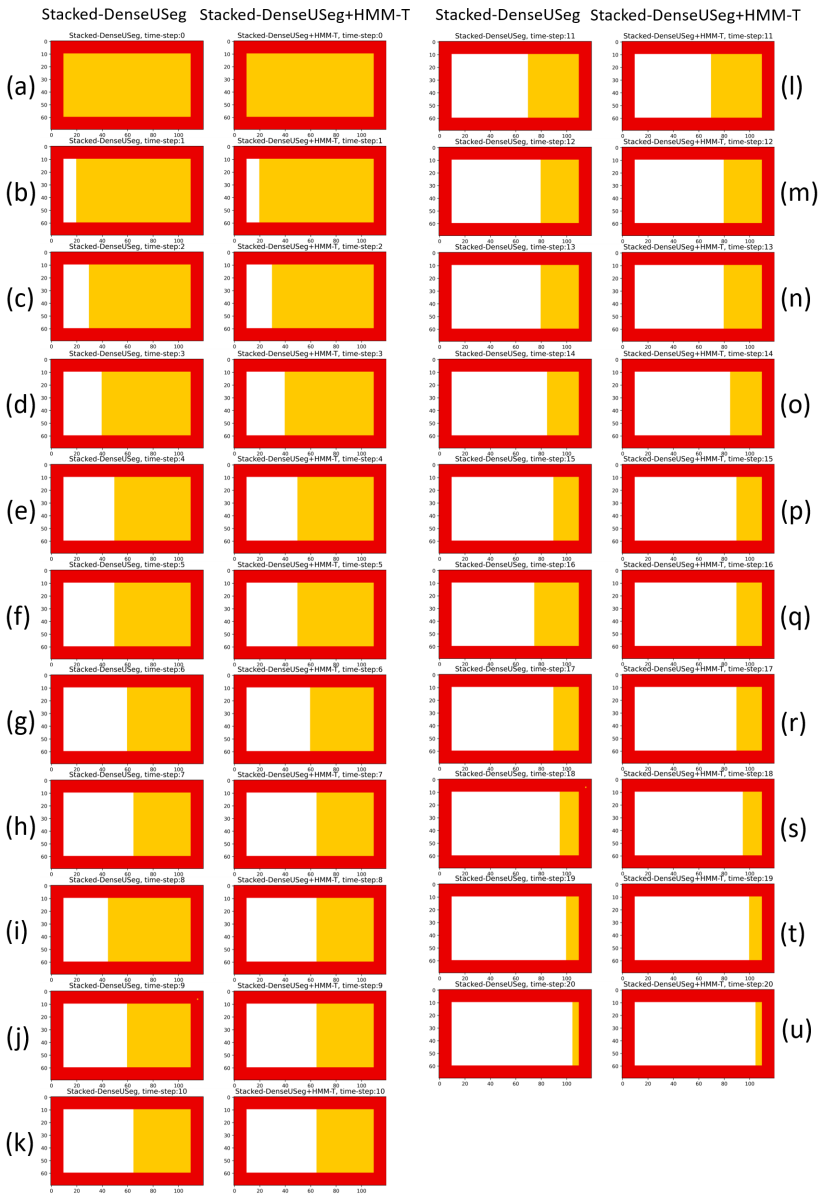


Figure 7.9: *Behavioral test of the HMM-T for the expected transition of the magnesium toxels (class=2) to air pocket toxels (class=3) through time. For this test the HMM-T is being used. Subfigures of displaying a ROI around the testing area (see Figure 7.8) and (**a**-**u**) refer to the time-steps* $0-20$ *respectively. The slice displayed is the 1190 (out of 2160) of the height dimension (y-axis), for the ROI [1256:1864, 450:962] (xz-plane). As it can be seen in (**i**) and (**q**) a number of the air pocket prediction (white) remained in place, which is expected since magnesium (yellow) has high probability to turn into air pockets (white) but not the opposite*

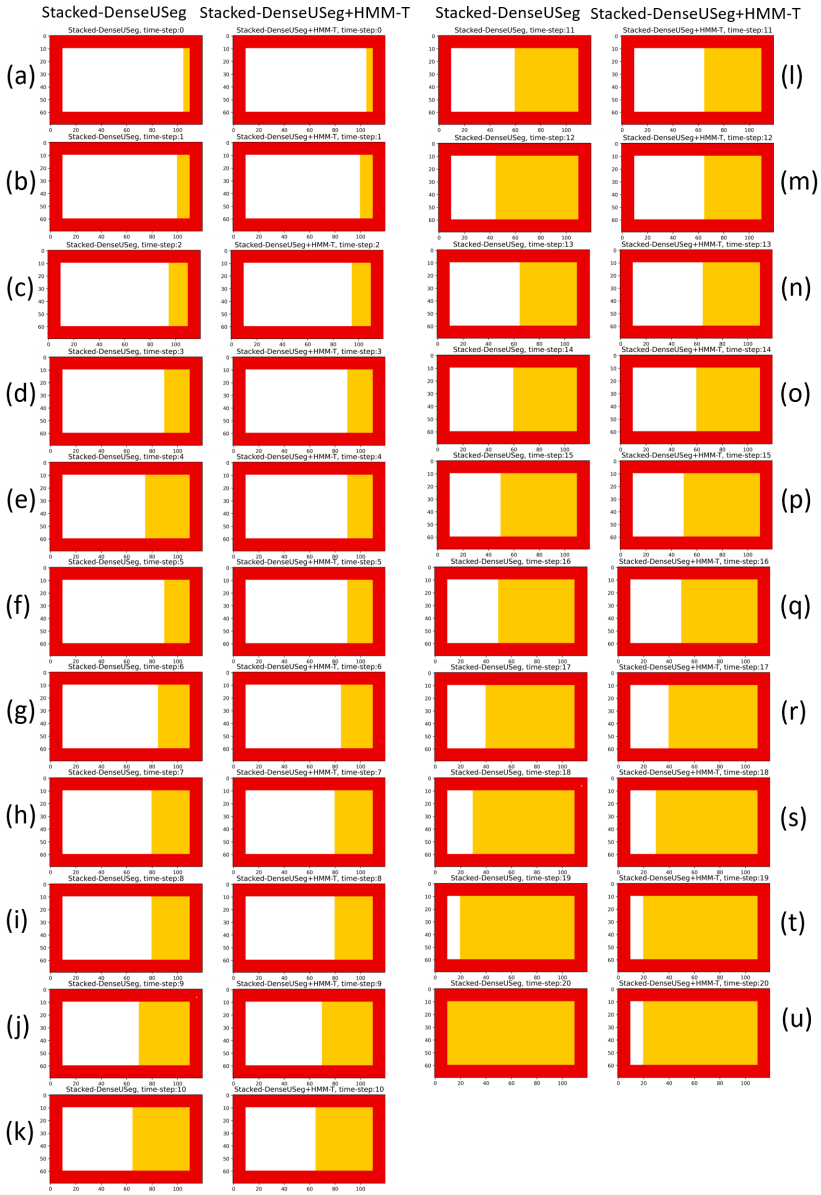


Figure 7.10: *Behavioral test of the HMM-T for the unexpected transition of the air pocket toxels (class=3) to magnesium toxels (class=2) through time. For this test the HMM-T is being used. Subfigures of displaying a ROI around the testing area (see Figure 7.8) and (**a**-**u**) refer to the time-steps* $0 - 20$ *respectively. The slice displayed is the 1190 (out of 2160) of the height dimension (y-axis), for the ROI [1256:1864, 450:962] (xz-plane). As it can be seen in (**e**), (**l**-**n**), (**u**) a number of air pocket predictions (white) persisted, which is expected since air pockets (white) has low probability to turn into magnesium (yellow). Specifically (**t**-**u**) shows that without a follow-up observation to provide support for what (**u**) displays (no air pockets), HMM-T kept the predictions of the previous time step.*

one improbable transition). Finally, in Figures 7.10(**t-u**) it seem that without a follow-up observation to provide support and further evidence for what Figure 7.10(**u**) displays (no Air Pockets), the HMM-T does not allow the transition. The inability of the network to further prevent or stop unexpected transitions at first might appear to be a negative property, however it is not a bad behaviour in practice. It is important to remember that the purpose of the time-series is to investigate temporal physical processes and unexpected results if valid should be welcomed; no predictive model can be perfect. With an approach designed to aggressively alter a segmentation to a certain direction, there is a high risk of incorporating bias and altering not only the incorrect noise-related annotations but also the unexpected correct ones. Therefore, we believe that allowing improbable transitions when there is an abundance of confirmatory observations is positive and that the level of refinement that our model executes is proper. In the following Section the results of quantitative experiments investigating HMM performance will be presented.

7.4.3 Experiment 3: Quantitative experiments

After testing the quality of the model's outputs and their behaviour in specific cases (via the HMM-T model), this Section we will now measure the segmentation accuracy. The best way to measure their accuracy would be with the existence of accurate 4D annotations with a testing time-series. However, each tomogram in the time-series is undersampled, which means that is practically impossible to manually segment their reconstructions with the same accuracy as the reconstructions of highly-sampled tomograms. This due to their low SNR, making it extremely difficult for human annotators to discern exact edges in what they have to segment, let alone restore missing information due to the downsampling of projections. As we explained in Section 7.3.3, the impossible task of restoring missing information also prompted us to alter certain probabilities in the emission matrix (see matrices 7.6 and 7.7) due to the "correctly" misclassified toxels. In this scenario, there are two practical options available for measuring accuracy. The first is to measure it in an area of the sample, where temporal events are unlikely to have taken place. Then for this area, it would be valid to copy the annotations from a representative highly-sampled

Resulting IoU	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
Stacked-DenseUSeg	0.987	0.966	0.405	0.062	0.605
Stacked-DenseUSeg+HMM-T	**0.990**	**0.967**	**0.431**	**0.122**	**0.628**
Stacked-DenseUSeg+HMM-TC	**0.990**	**0.967**	0.430	0.085	0.618

Table 7.1: Intersection over union (IoU) segmentation results using the annotations from the representative to the time-series highly-sampled tomogram. The comparison is performed in an area of the sample where temporal events are minimal. The tested approaches are Stacked-DenseUSeg, Stacked-DenseUSeg + HMM-T, Stacked-DenseUSeg + HMM-TC. Labels $0-3$ are respectively Air, Aluminium, Magnesium and Air Pockets. The results are from 10 slices (slices 1660-1670 height-wise) located at the bottom of the imaged sample in the time-series. This is to ensure that temporal events did not take place and that the annotations from the representative highly-sampled tomogram of the time-series are valid though time. As it can be seen both HMM models performed better than the simple use of Stacked-DenseUSeg, which is a strong indication that our HMM models corrected time inconsistencies due to the Stacked-DenseUSeg using only spatial and not temporal information.

tomogram (*RealHS1* as presented in Section 3.2.2) captured at the beginning, end or both the beginning and the end of the collection of the time-series through time. Since temporal events are unlikely to have happened, the annotations should have remained the same through time. The second approach would be to manually annotate 4D data, as best as we can, focusing on areas where temporal events are likely to have happened. Obviously, due to the segmentation difficulties mentioned earlier, these annotations may contain bias and errors. This is because there would be areas where it is not humanly possible to restore and infer missing information to the same level as a trained network. Nevertheless, measuring the accuracy against these annotations may give a hint regarding the accuracy of our models, especially if used as a relative comparison.

Starting with the annotations copied through time from a highly-sampled tomogram, Table 7.1 presents the Intersection over Union achieved for each class and the mean IoU. For the comparison only 10 slices (slices 1660-1670 height-wise) are used from the bottom of the imaged sample in the time-series, increasing the probability that temporal events did not occur. This is because temporal events in our datasets are caused by a droplet of salt-water triggering corrosion in the areas that it comes in contact with the Magnesium (at the upper slices in the height dimension). Therefore, it is not anticipated that corrosion will occur lower down in the sample. As can be seen, HMM-T is the best performing model overall, but HMM-TC is also better than the individual segmentations

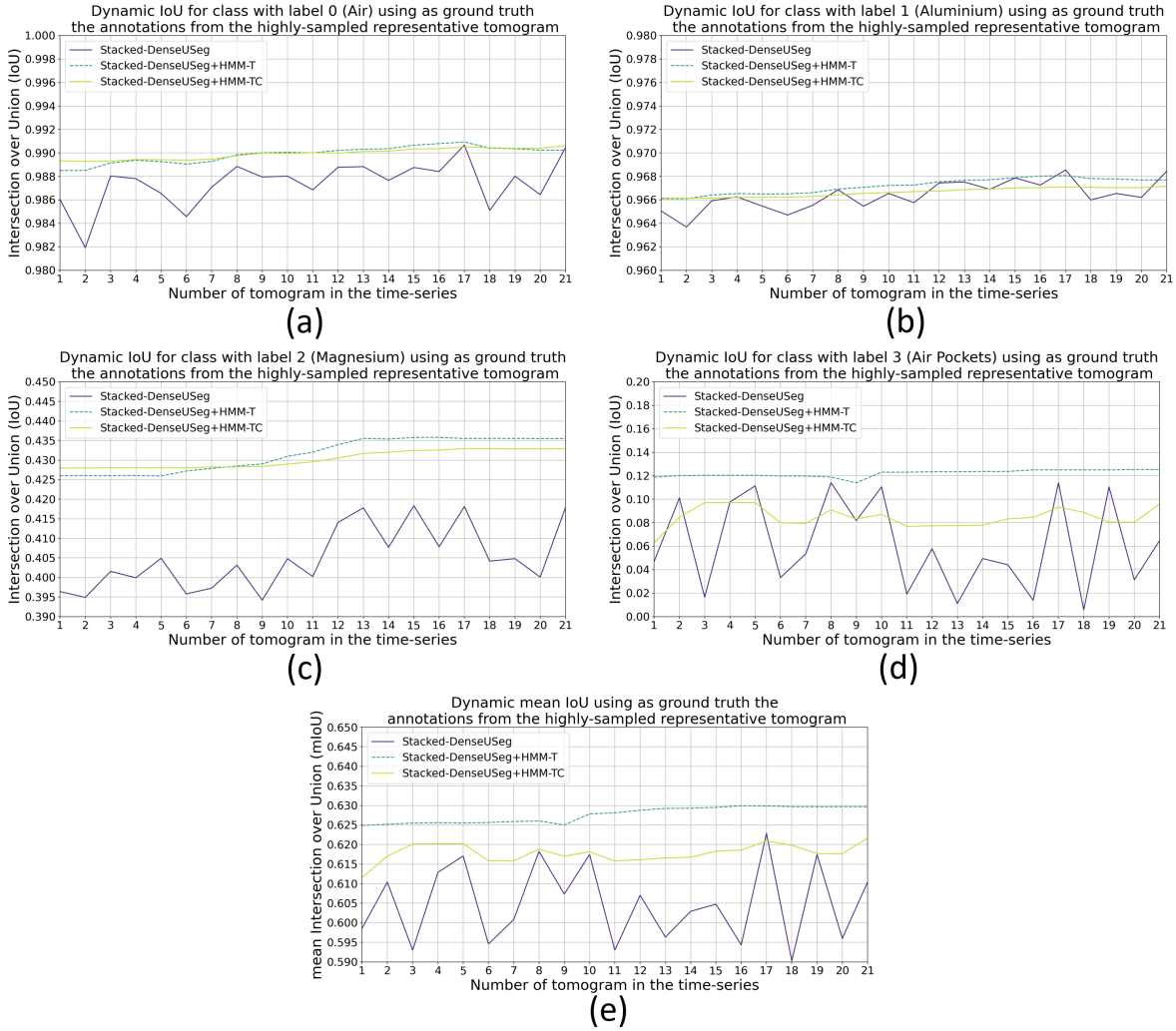


Figure 7.11: *Dynamic testing IoU for each tomogram in the time-series using as ground truth the annotations from the representative to the time-series highly-sampled tomogram regarding Stacked-DenseUSeg, Stacked-DenseUSeg+HMM-T and Stacked-DenseUSeg+HMM-TC. (**a**) Dynamic testing IoU for class with label 0, Air, (**b**) Dynamic testing IoU for class with label 1, Aluminium, (**c**) Dynamic testing IoU for class with label 2, Magnesium, (**d**) Dynamic testing IoU for class with label 3, Air Pockets, (**e**) Dynamic testing mean IoU for all the segmentation classes. Both HMM-T and HMM-TC work to correct temporal inconsistencies, but do not remove all temporal changes. If they were removed all temporal changes since the ground truth is static, the IoU result would also be static, however the IoU results through time for "Stacked-DenseUSeg+HMM-T" and "Stacked-DenseUSeg+HMM-TC" are not horizontal lines. Overall it seems that HMM-T is performing better that the HMM-TC especially for the class of Air Pockets.*

of our Stacked-DenseUSeg network alone. This showcases the advantages of the use of hidden Markov Model for the refinement of 3D segmentation predictions through time. Based on Figure 7.11, both models correct temporal inconsistencies, but do not remove all temporal changes. If the case was that all temporal changes are removed, meaning that the segmentations after the application of our models are static through time, since the employed ground truth is static though time (annotations of a highly-sampled tomogram copied through time), then the IoU should also be static though time. However as can be seen in Figure 7.11, the IoU results through time for "Stacked-DenseUSeg+HMM-T" and "Stacked-DenseUSeg+HMM-TC" are not horizontal lines, which means that they are not static and thus not all temporal changes have been removed. The elevated performance of HMM-T over HMM-TC is potentially attributed to the modification of the probabilities (for the reasons explained in Section 7.3.3), making HMM-TC not as well-tuned as HMM-T. Namely, by examining Figure 7.11(**d**) it can be seen that it does not perform well in the refining of the segmentations of Air Pockets, which is the class for which most observation probabilities were modified.

In Figure 7.12 the resulting annotations are presented. As it can be seen in marks $\boldsymbol{P_1}$-$\boldsymbol{P_4}$ and the zoomed images, the improvement that the HMM models offer is noticeable. Here it is seems that the HMM-TC model visual performance is slightly better due to resembling more closely the highly-sampled annotations. Certainly, $\boldsymbol{P_1}$ (Stacked-DenseUSeg output) would seem to be the worst of the three predictions.

After the use of annotations from a representative highly-sampled tomogram in the areas that do not experience temporal events, we continue with comparisons using our manual 4D annotations as ground truth. For this test we annotated a sub-region only, the two top-most slices (slices 1190-1191 height-wise), paying special effort to adhere to the input's boundaries and correctly segment objects where the other methods did not segment well. Only a small region was annotated due to the labour costs involved. Naturally, this is a difficult task due to the low SNR of the input reconstructions. Unavoidably, there may be some errors and bias incorporated in the annotations, however there is not an alternative for the measurement of segmentation accuracy in areas that temporal events occurred.

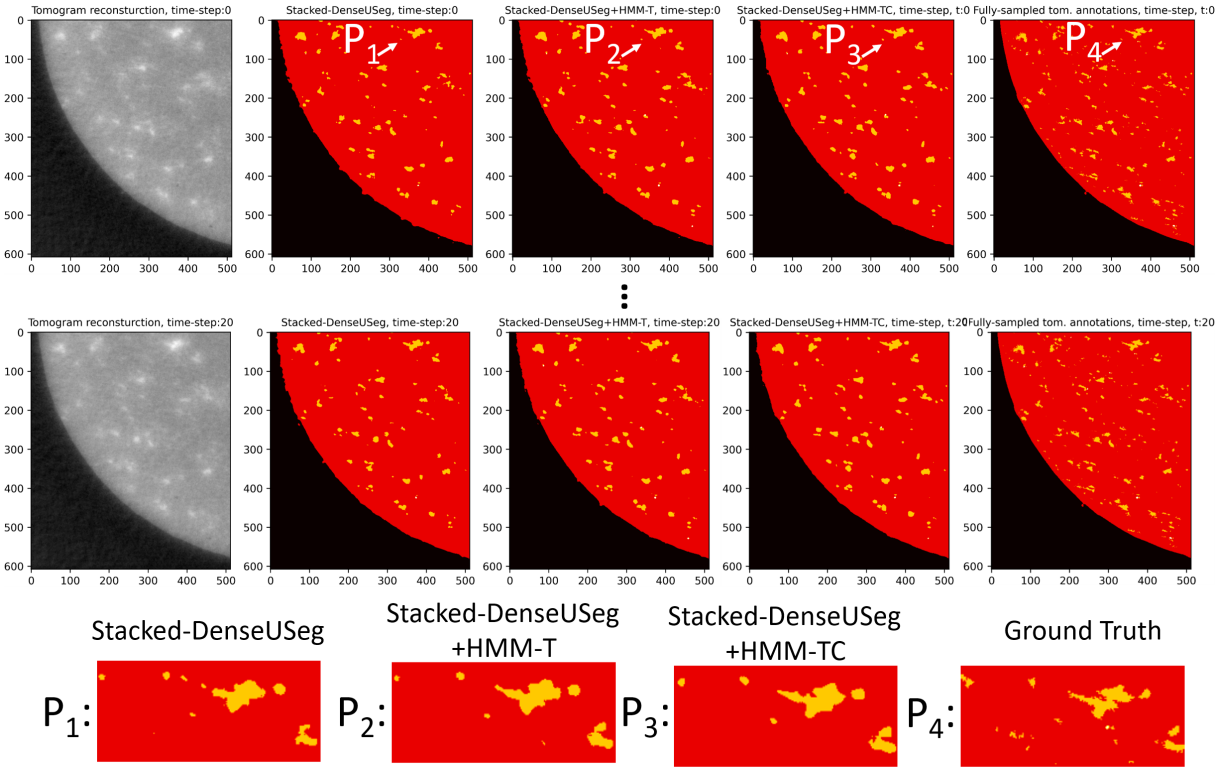


Figure 7.12: *Comparison of segmentation predictions in the RealTS1 time-series of tomograms after the use of Stacked-DenseUSeg, Stacked-DenseUSeg + HMM-T, Stacked-DenseUSeg + HMM-TC against the annotations from the corresponding representative highly-sampled tomogram to the time-series. For the comparison 10 slices (slices 1660-1670 height-wise) are used at the bottom of the imaged sample in the time-series. This is to ensure that temporal events did not take place and that the annotations from the representative highly-sampled tomogram of the time-series are valid though time. The figure displays the* 1660^{th} *slice out of 2160 (y-axis) and for the ROI [1256:1864, 450:962] (xz-plane). Marks* $\boldsymbol{P_1}$-$\boldsymbol{P_4}$*, and their zoomed versions provide a qualitative indication of the advantages of the HMM models.*

For the production of the annotations, the predictions of the HMM-T model were first used as a template, which was then carefully refined manually. As can be seen in Table 7.2 and Figure 7.13, HMM-T is again the best performing method with the other methods not performing as well. However, their results are almost identical in regards to the mean IoU. Naturally, the high accuracy of the HMM-T can be attributed to the fact that it was used as a template for the 4D manual segmentations. In Figure 7.14, mark $\boldsymbol{L_5}$ suggests that that HMM-TC was able to better capture finer details in the manual annotations for difficult areas comparing $\boldsymbol{L_1}$-$\boldsymbol{L_4}$. We would hope that with further modifications and better tuning of its parameters, the HMM-TC may surpass the HMM-T in visual accuracy.

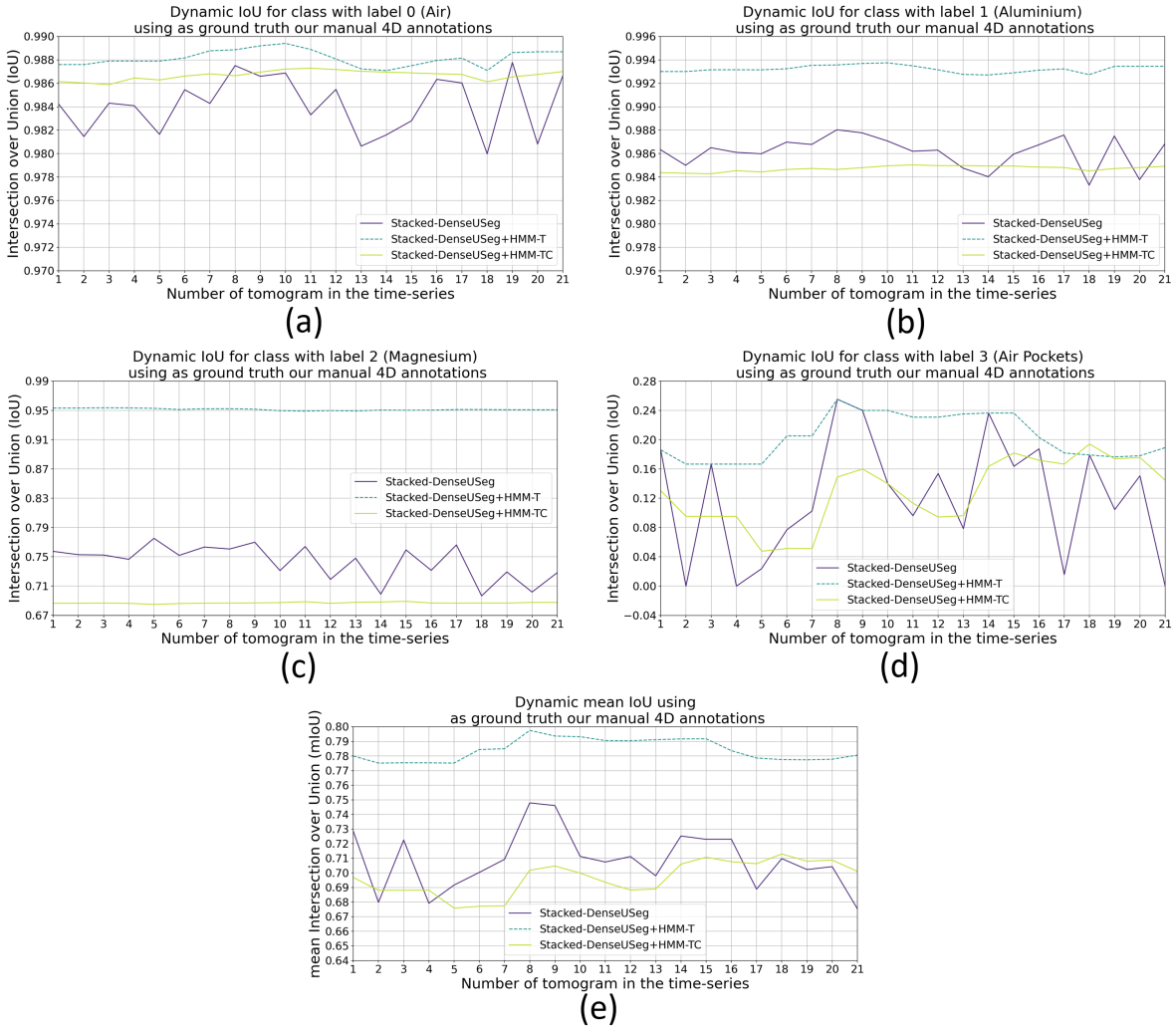


Figure 7.13: *Dynamic testing IoU for each tomogram in the time-series using as ground truth our manual 4D annotations regarding Stacked-DenseUSeg, Stacked-DenseUSeg+HMM-T and Stacked-DenseUSeg+HMM-TC. (**a**) Dynamic testing IoU for class with label 0, Air, (**b**) Dynamic testing IoU for class with label 1, Aluminium, (**c**) Dynamic testing IoU for class with label 2, Magnesium, (**d**) Dynamic testing IoU for class with label 3, Air Pockets, (**e**) Dynamic testing mean IoU for all the segmentation classes. From the results it seems HMM-T performs with better IoU accuracy than HMM-TC, however we report this with some reservations due to the high correlation between the manual annotations used and the predictions of HMM-T.*

Resulting IoU	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
Stacked-DenseUSeg	0.984	0.986	0.743	0.123	0.709
Stacked-DenseUSeg+HMM-T	**0.988**	**0.993**	**0.951**	**0.203**	**0.784**
Stacked-DenseUSeg+HMM-TC	0.987	0.985	0.687	0.135	0.698

Table 7.2: Intersection over union (IoU) segmentation results using our manual 4D annotations. The comparison is performed in an area where temporal events are likely to occurred. The tested approaches are Stacked-DenseUSeg, Stacked-DenseUSeg + HMM-T, Stacked-DenseUSeg + HMM-TC. Labels $0-3$ are respectively Air, Aluminium, Magnesium and Air Pockets. For the comparison 2 top slices (slices 1190-1191 height-wise) are used in the imaged sample of the time-series. This is to ensure that temporal events occurred in these areas and that the comparison would provide some evidence about the temporal behaviour of our HMM models. As it can be seen the HMM-T model performs better than the simple use of Stacked-DenseUSeg and HMM-TC. However it is highly likely that this is due to the our manual 4D annotations being highly correlated with HMM-T output, which were used as our annotations' starting point. Nevertheless, the fact that HMM-TC is very close in accuracy with the unrefined segmentations, provides a soft indication about its good behaviour in areas where temporal events have occurred. Finally, the fact that the HMM-T predictions are so good that they are chosen as a starting point for a more refined annotations, is an other indication for the quality and the accuracy of the results.

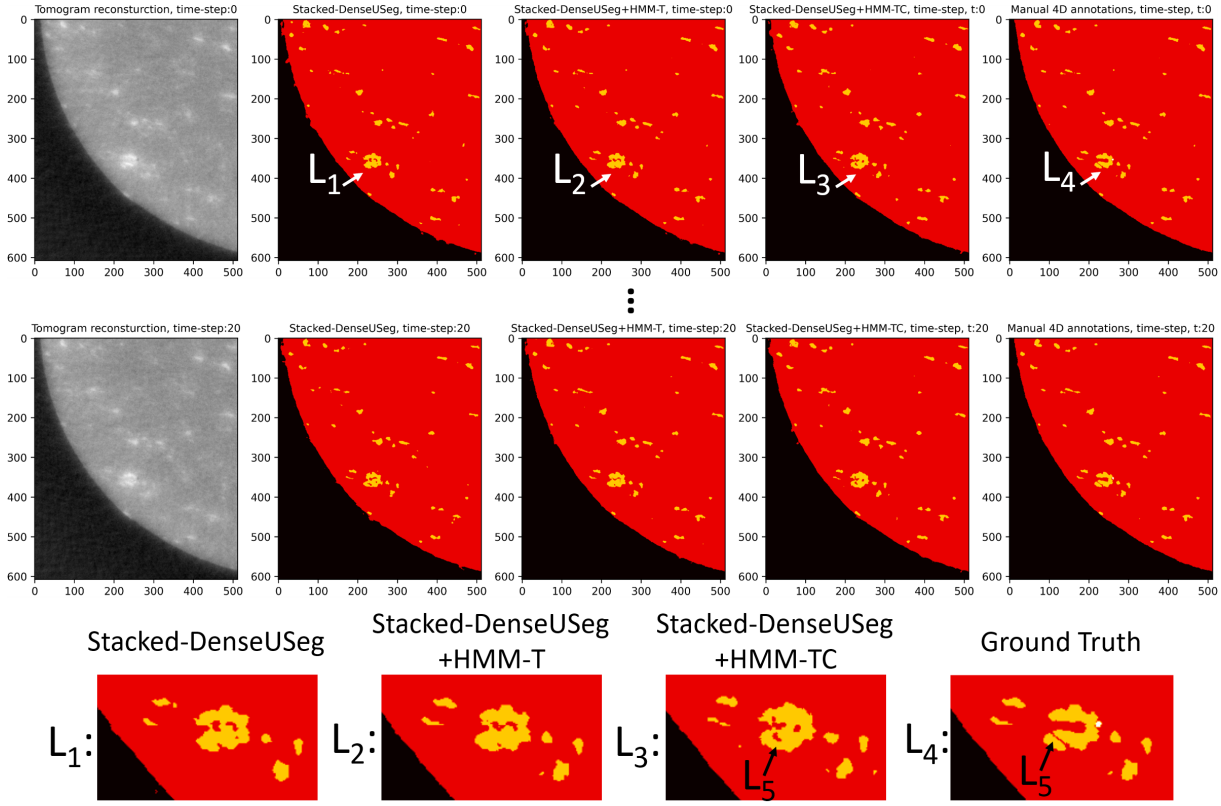


Figure 7.14: *Comparison of segmentation predictions in the RealTS1 time-series of tomograms after the use of Stacked-DenseUSeg, Stacked-DenseUSeg + HMM-T, Stacked-DenseUSeg + HMM-TC against our manual 4D annotations. For the comparison 2 top slices (slices 1190-1191 height-wise) are used in the imaged sample of the time-series. This is to ensure that temporal events occurred in these areas and that the comparison would provide some evidence about the temporal behaviour of our HMM models. The figure displays the* 1190^{th} *slice out of 2160 (y-axis) and for the ROI [1256:1864, 450:962] (xz-plane). Marks* $\boldsymbol{L_1}$*-*$\boldsymbol{L_5}$*, and zoomed versions provide a qualitative indications for the advantages of the HMM models.*

7.4.4 Experiment 4: Sensitivity experiments for the elected transition probabilities

After testing both our proposed models qualitatively, behaviourally and quantitatively, this Section we will investigate the sensitivity of the segmentation results to the empirically elected transition probabilities. However, similar to the behavioural experiments in Section 7.4.2, due to the similar structure of the HMMs and the ease of understanding HMM-T behaviour due to less parameters, we only used HMM-T for the sensitivity tests. Again, the presented results are regarding two areas of the sample. In the first area, temporal events are unlikely to have taken place and in the second, it is likely that they have. For the first area, the ground truth used is the annotations of the representative highly-sampled tomogram of the time-series repeated though time (since for this area temporal events are unlikely to have occurred). For the second area, the ground truth used is our manual 4D annotations, mentioned in Section 7.4.3.

As presented in Section 7.3.3, the transition probabilities of our HMMs form the transition matrix 7.2, where each row corresponds to the transition probabilities of each class transitioning to all other. Because of this, the summation of the probabilities of each row in matrix 7.2 is equal to 1. For the sensitivity results presented in the following subsections, the transition probabilities of only one class are altered at each time (only one row is altered at each time), while the rest transition probabilities have the same values as in presented in matrix 7.2.

7.4.4.1 Sensitivity to the transition probabilities of Air

Starting with the transition probabilities of Air (1^{st} row of matrix 7.2), Table 7.3 presents the IoU results for Stacked-DenseUSeg+HMM-T for two areas (where temporal events probably did not occur or where probably did occur) after altering the transition probabilities of Air accordingly. In the presented experiments, the transition probability of Air remaining as Air through time is gradually decreased from 0.998 to 0.6, when at the same time the transition probabilities of Air turning into other classes get increased. Nevertheless, the transition probability of Air turning into Air Pockets is kept at 0. This

A: IoU results using as ground truth the annotations from the representative highly-sampled tomogram repeated through time					
Stacked-DenseUSeg +HMM-T	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
$99.8\% - 0.1\% - 0.1\% - 0\%$ (same as in matrix 7.2)	0.990	0.967	0.431	0.122	0.628
$90\% - 5\% - 5\% - 0\%$	0.990	0.967	0.431	0.122	0.628
$80\% - 10\% - 10\% - 0\%$	0.990	0.967	0.431	0.122	0.628
$70\% - 15\% - 15\% - 0\%$	0.990	0.967	0.431	0.122	0.628
$60\% - 20\% - 20\% - 0\%$	0.990	0.967	0.431	0.122	0.628
$60\% - 15\% - 15\% - 10\%$	0.990	0.967	0.431	0.122	0.628
B: IoU results using as ground truth our manual 4D annotations					
Stacked-DenseUSeg +HMM-T	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
$99.8\% - 0.1\% - 0.1\% - 0\%$ (same as in matrix 7.2)	0.988	0.993	0.951	0.203	0.784
$90\% - 5\% - 5\% - 0\%$	0.988	0.993	0.951	0.203	0.784
$80\% - 10\% - 10\% - 0\%$	0.988	0.993	0.951	0.203	0.784
$70\% - 15\% - 15\% - 0\%$	0.988	0.993	0.951	0.203	0.784
$60\% - 20\% - 20\% - 0\%$	0.988	0.993	0.951	0.203	0.784
$60\% - 15\% - 15\% - 10\%$	0.988	0.993	0.951	0.203	0.784

Table 7.3: Intersection over union (IoU) segmentation results using, **A** as ground truth the annotations from the representative highly-sampled tomogram and **B** our manual 4D annotations in order to measure the sensitivity of the empirically elected transition probabilities of Air. The comparison is performed, **A** in an area of the sample where temporal events are minimal (slices 1660-1670 height-wise) and **B** in an area where temporal events are likely to have occurred (slices 1190-1191 height-wise). The tested approach is Stacked-DenseUSeg + HMM-T. The four percentages in the first column represent the various examined transition probabilities for Air and correspond to the transition probability of Air turning into Air, Aluminium, Magnesium and Air Pockets respectively. While the transition probabilities of Air are altered the transition probabilities for the other classes are kept the same as in matrix 7.2. The IoU of Labels $0 - 3$ are respectively the IoU of Air, Aluminium, Magnesium and Air Pockets. Since Air and Air-Pockets correspond to the same material, but one present outside the Aluminium pin and the other inside, it is not possible to transition between the two and this is why for most experiments the transition probability of Air turning into Air Pockets is set to 0. However, in the final experiment this probability is set to 10% in order to examine the HMM-T response. As seen potentially due to the high accuracy of Stacked-DenseUSeg to detect Air (see the observation matrix 7.4), changes in the transition probabilities of Air did not change HMM-T response both in areas **A** and **B**.

is logical given the fact that Air and Air-Pockets correspond to the same material (Air), but one present outside the Aluminium pin and the other inside and so it is physically not possible to transition between the two. Despite this fact though, an experiment with a non-zero transition probability of Air turning into Air Pockets is tested (last row of Table 7.3(**A**) and 7.3(**B**)).

By observing Table 7.3, it is seen that potentially due to the high accuracy of Stacked-DenseUSeg to detect Air (see the observation matrix 7.4) changes in the transition probabilities of Air did not change the HMM-T response both in areas **A** and **B**. This means that as long as the highest transition probability for Air is to remain as Air through time, HMM-T will offer the same results.

7.4.4.2 Sensitivity to the transition probabilities of Aluminium

Regarding the transition probabilities of Aluminium (2^{nd} row of matrix 7.2), Table 7.4 presents the IoU results for Stacked-DenseUSeg+HMM-T for two areas (where temporal events probably did not occur or where probably did occur) after altering the transition probabilities of Aluminium accordingly. In the presented experiments, the transition probability of Aluminium remaining as Aluminium through time is gradually decreased from 0.996 to 0.6, when at the same time the transition probabilities of Aluminium turning into other classes get increased.

As it can be seen the transition probabilities of Aluminium elected in matrix 7.2 offer the highest segmentation IoU results in Table 7.4(**A**) (for the area where temporal events probably did not occur). The same can be seen in Table 7.4(**B**) (for the area where temporal events probably occur) for the mean IoU and the IoU of all classes, except the IoU of Air Pockets. However, the improvement of the IoU of Air Pockets, when the transition probabilities of Aluminium are $7.5\% - 70\% - 7.5\% - 15\%$ and $10\% - 60\% - 10\% - 20\%$, is marginal and when these are used, the IoU of Magnesium is decreased significantly. Overall, it can be seen that the different transition probabilities of Aluminium mostly affect the IoU of Magnesium in Table 7.4(**B**), due to the existence of temporal events. This is potentially due to the increased probability of Aluminium toxels falsely transitioning into

Magnesium, since the IoU for Magnesium becomes worse when the transition probability of Aluminium turning into Magnesium gets increased.

A: IoU results using as ground truth the annotations from the representative highly-sampled tomogram repeated through time					
Stacked-DenseUSeg +HMM-T	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
$0.1\% - 99.6\% - 0.1\% - 0.2\%$ (same as in matrix 7.2)	**0.990**	**0.967**	**0.431**	**0.122**	**0.628**
$2.5\% - 90\% - 2.5\% - 5\%$	**0.990**	**0.967**	**0.431**	0.114	0.625
$5\% - 80\% - 5\% - 10\%$	**0.990**	**0.967**	**0.431**	**0.122**	0.627
$7.5\% - 70\% - 7.5\% - 15\%$	**0.990**	0.966	**0.431**	0.121	0.627
$10\% - 60\% - 10\% - 20\%$	**0.990**	0.966	0.429	0.121	0.627
B: IoU results using as ground truth our manual 4D annotations					
Stacked-DenseUSeg +HMM-T	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
$0.1\% - 99.6\% - 0.1\% - 0.2\%$ (same as in matrix 7.2)	**0.988**	**0.993**	**0.951**	0.203	**0.784**
$2.5\% - 90\% - 2.5\% - 5\%$	**0.988**	**0.993**	0.923	0.203	0.777
$5\% - 80\% - 5\% - 10\%$	**0.988**	0.992	0.907	0.203	0.772
$7.5\% - 70\% - 7.5\% - 15\%$	**0.988**	0.991	0.891	**0.208**	0.770
$10\% - 60\% - 10\% - 20\%$	**0.988**	0.991	0.866	**0.208**	0.763

Table 7.4: Intersection over union (IoU) segmentation results using, **A** as ground truth the annotations from the representative highly-sampled tomogram and **B** our manual 4D annotations in order to measure the sensitivity of the empirically elected transition probabilities of Aluminium. The comparison is performed, **A** in an area of the sample where temporal events are minimal (slices 1660-1670 height-wise) and **B** in an area where temporal events are likely to occurred (slices 1190-1191 height-wise). The tested approach is Stacked-DenseUSeg + HMM-T. The four percentages in the first column represent the various examined transition probabilities for Air and correspond to the transition probability of Aluminium turning into Air, Aluminium, Magnesium and Air Pockets respectively. While the transition probabilities of Aluminium are altered the transition probabilities for the other classes are kept the same as in matrix 7.2. The IoU of Labels $0-3$ are respectively the IoU of Air, Aluminium, Magnesium and Air Pockets. As it can be seen the transition probabilities of Aluminium elected in matrix 7.2 offer the highest segmentation IoU results for **A**. The same is observed in **B** for the mean IoU and the IoU of Air, Aluminium and Magnesium. Overall, Aluminium's transition probabilities mostly affect the IoU of Magnesium in **B**, making them range from 0.951 to 0.866.

7.4.4.3 Sensitivity to the transition probabilities of Magnesium

Regarding the transition probabilities of Magnesium (3^{rd} row of matrix 7.2), Table 7.5 presents the IoU results for Stacked-DenseUSeg+HMM-T for two areas (where temporal events probably did not occur or where probably did occur) after altering the transition probabilities of Magnesium accordingly. In the presented experiments, the transition probability of Magnesium remaining as Magnesium through time is gradually decreased from 0.8982 to 0.525, when at the same time the transition probabilities of Magnesium turning into other classes get increased.

By observing Table 7.5 it is seen that for 7.5(**A**) the optimal transition probabilities of Magnesium might be $0.05\% - 0.05\% - 94.95\% - 4.95\%$ (which are the transition probability turning into Air, Aluminium, Magnesium and Air Pockets respectively). This is because when they are used, the IoU of Magnesium is increased without the IoU for the other of classes getting decreased. Nevertheless, this improvement is extremely marginal and in general it can be concluded from 7.5(**A**) (regarding the area where temporal events probably did not occur), that the reported IoU change only marginally for the various tested transition probabilities of Magnesium.

In Table 7.5(**B**), on the other hand (regarding the area where temporal events probably occur) it seems that the optimal transition probabilities are the elected for matrix 7.4. However, this is reported with some reservations due to the high correlation between the manual annotations used as ground truth and the predictions of HMM-T. In general, based upon Table 7.5(**B**), the elected Magnesium transition probabilities play an important role towards an overall high mean IoU accuracy. This is logical, since the *RealTS1* time-series studies the corrosion of Magnesium (Magnesium turning into Air Pockets) and the results of Table 7.5(**B**) are regarding the area of the sample where temporal events probably occurred.

A: IoU results using as ground truth the annotations from the representative highly-sampled tomogram repeated through time					
Stacked-DenseUSeg +HMM-T	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
$0.05\% - 0.05\% - 94.95\% - 4.95\%$	**0.990**	0.967	**0.432**	**0.122**	**0.628**
$0.1\% - 0.1\% - 89.82\% - 9.98\%$ (same as in matrix 7.2)	**0.990**	0.967	0.431	**0.122**	**0.628**
$2.5\% - 2.5\% - 77.5\% - 17.5\%$	**0.990**	0.967	0.427	**0.122**	0.627
$5\% - 5\% - 65\% - 25\%$	**0.990**	0.967	0.425	**0.122**	0.626
$7.5\% - 7.5\% - 52.5\% - 32.5\%$	**0.990**	**0.968**	0.422	**0.122**	0.625
B: IoU results using as ground truth our manual 4D annotations					
Stacked-DenseUSeg +HMM-T	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
$0.05\% - 0.05\% - 94.95\% - 4.95\%$	**0.988**	**0.993**	0.943	**0.203**	0.782
$0.1\% - 0.1\% - 89.82\% - 9.98\%$ (same as in matrix 7.2)	**0.988**	**0.993**	**0.951**	**0.203**	**0.784**
$2.5\% - 2.5\% - 77.5\% - 17.5\%$	**0.988**	0.992	0.907	**0.203**	0.773
$5\% - 5\% - 65\% - 25\%$	**0.988**	0.992	0.889	**0.203**	0.768
$7.5\% - 7.5\% - 52.5\% - 32.5\%$	**0.988**	0.991	0.871	**0.203**	0.763

Table 7.5: Intersection over union (IoU) segmentation results using, **A** as ground truth the annotations from the representative highly-sampled tomogram and **B** our manual 4D annotations in order to measure the sensitivity of the empirically elected transition probabilities of Magnesium. The comparison is performed, **A** in an area of the sample where temporal events are minimal (slices 1660-1670 height-wise) and **B** in an area where temporal events are likely to occurred (slices 1190-1191 height-wise). The tested approach is Stacked-DenseUSeg + HMM-T. The four percentages in the first column represent the various examined transition probabilities for Magnesium and correspond to the transition probability of Magnesium turning into Air, Aluminium, Magnesium and Air Pockets respectively. While the transition probabilities of Magnesium are altered the transition probabilities for the other classes are kept the same as in matrix 7.2. The IoU of Labels $0-3$ are respectively the IoU of Air, Aluminium, Magnesium and Air Pockets. For **A** it can be seen that potentially an performance improvement can be gained by using $0.05\% - 0.05\% - 94.95\% - 4.95\%$ as the transition probabilities of Magnesium, since the IoU of Magnesium is increased without the IoU for the rest of classes getting decreased. For **B**, it seems that the optimal transition probabilities are the ones of matrix 7.2, however this reported with some reservations due to the high correlation between the manual annotations used as ground truth and the predictions of HMM-T.

7.4.4.4 Sensitivity to the transition probabilities of Air Pockets

Regarding the transition probabilities of Air Pockets (4^{th} row of matrix 7.2), Table 7.6 presents the IoU results for Stacked-DenseUSeg+HMM-T for two areas (where temporal events probably did not occur or where probably did occur) after altering the transition probabilities of Air Pockets accordingly. In the presented experiments, the transition probability of Air Pockets remaining as Air Pockets through time is gradually decreased from 0.998 to 0.6, when at the same time the transition probabilities of Air Pockets turning into other classes get increased. Nevertheless, the transition probability of Air Pockets turning into Air is kept at 0. This is for the same reason described at Section 7.4.4.1. Despite this fact though, an experiment with a non-zero transition probability of Air Pockets turning into Air is tested (last row of Table 7.6(**A**) and 7.6(**B**)).

As it can be seen in Table 7.6, the transition probability of Air Pockets remain Air Pockets is very significant both for experiments in the areas, where temporal events probably did not occur (Table 7.6(**A**)) and areas where temporal events probably occur (Table 7.6(**B**)). This is potentially because continuous observations of Air Pockets through the time-series are rare, This means that if Stacked-DenseUSeg segments a toxel as Air Pocket, is rare that the temporally subsequent toxel will also be segmented as Air Pocket. This is probably due to noise and the difficulty of detecting and segmenting the spatially small class of Air Pockets. For these reason, low transition probabilities of Air Pockets turning into other classes are needed in order to make them continuous through time once they are detected and segmented by the Stacked-DenseUSeg. Since this leads to better IoU results in both Table 7.6(**A**) and Table 7.6(**B**), it can be concluded the optimal elected transition probabilities for Air Pockets are the ones selected in matrix 7.2.

A: IoU results using as ground truth the annotations from the representative highly-sampled tomogram repeated through time					
Stacked-DenseUSeg +HMM-T	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
$0\% - 0.1\% - 0.1\% - 99.8\%$ (same as in matrix 7.2)	**0.990**	**0.967**	**0.431**	**0.122**	**0.628**
$0\% - 5\% - 5\% - 90\%$	**0.990**	**0.967**	**0.431**	0.065	0.613
$0\% - 10\% - 10\% - 80\%$	**0.990**	**0.967**	**0.431**	0.055	0.611
$0\% - 15\% - 15\% - 70\%$	**0.990**	**0.967**	**0.431**	0.05	0.609
$0\% - 20\% - 20\% - 60\%$	**0.990**	**0.967**	**0.431**	0.047	0.609
$10\% - 15\% - 15\% - 60\%$	**0.990**	**0.967**	**0.431**	0.035	0.606
B: IoU results using as ground truth our manual 4D annotations					
Stacked-DenseUSeg +HMM-T	IoU of Label 0	IoU of Label 1	IoU of Label 2	IoU of Label 3	mean IoU
$0\% - 0.1\% - 0.1\% - 99.8\%$ (same as in matrix 7.2)	**0.988**	**0.993**	**0.951**	**0.203**	**0.784**
$0\% - 5\% - 5\% - 90\%$	**0.988**	**0.993**	0.950	0.124	0.764
$0\% - 10\% - 10\% - 80\%$	**0.988**	**0.993**	0.943	0.101	0.756
$0\% - 15\% - 15\% - 70\%$	**0.988**	**0.993**	0.937	0.091	0.752
$0\% - 20\% - 20\% - 60\%$	**0.988**	**0.993**	0.936	0.075	0.748
$10\% - 15\% - 15\% - 60\%$	**0.988**	**0.993**	0.937	0.064	0.745

Table 7.6: Intersection over union (IoU) segmentation results using, **A** as ground truth the annotations from the representative highly-sampled tomogram and **B** our manual 4D annotations in order to measure the sensitivity of the empirically elected transition probabilities of Air Pockets. The comparison is performed, **A** in an area of the sample where temporal events are minimal (slices 1660-1670 height-wise) and **B** in an area where temporal events are likely to occurred (slices 1190-1191 height-wise). The tested approach is Stacked-DenseUSeg + HMM-T. The four percentages in the first column represent the various examined transition probabilities for Air Pockets and correspond to the transition probability of Air Pockets turning into Air, Aluminium, Magnesium and Air Pockets respectively. While the transition probabilities of Air Pockets are altered the transition probabilities for the other classes are kept the same as in matrix 7.2. The IoU of Labels $0-3$ are respectively the IoU of Air, Aluminium, Magnesium and Air Pockets. As it can be seen the transition probability of Air Pockets remain as Air Pockets is very significant both for **A** and **B**. This is potentially because, continuous observations of Air Pockets through the time-series are rare and low transition probabilities of Air Pockets turning into other classes are needed, in order to make them continuous through time once they are detected.

7.5 Discussions

Based on the results presented in Section 7.4, we demonstrate that it is possible to use our proposed hidden Markov models in order to refine the segmentations of 3D CNN networks such as the Stacked-DenseUSeg, when applied on 4D datasets (3D+time) by segmenting their component volumes independently through time. This operation is necessary, when only sufficiently accurate 3D ground truth is available for 4D datasets, which is the case in time-series of undersampled tomograms. Using highly-sampled representative tomograms captured before, after or before and after the collection of a time-series, it is possible to train accurate 3D CNN networks and then refine their segmentations on the time-series using our HMM models. The advantage of our HMM models is that they do not require 4D annotated data. The observation matrix of our HMM models is determined from the confusion matrix of the segmentation network, while their transition matrix is empirically determined by the user.

Naturally, since there are not any highly accurate 4D annotated data our HMM models face two limitations. Firstly, as mentioned in Section 7.4.3, the accuracy of our HMM models is not possible to be accurately accessed on regions of the data where temporal events occur. In Section 7.4.3, we were able to access it using our 4D manual annotations, however since they are obtained using the low SNR tomograms of time-series, they can not be considered as accurate as the annotation created using a highly-sampled tomogram. Additionally, the lack of accurate 4D annotated data, limits the user's ability to discover the most optimal transition probabilities for the transition matrix.

In order to address the previous to two limitations, two potential solution that could be employed, are either the use of synthetic tomograms and/or the use of a pair of highly-sampled tomograms that are collected respectively before and after the collection of the time-series. As regard to the use of synthetic data, it is possible to generate a time-series of synthetic tomograms with simulated temporal events in order to quantitatively access the accuracy of our HMM models. Since the data is synthetic, accurate ground truth of the time-series is known, allowing the accurate assessment of our HMM models

in regions where temporal events occur. However, use of synthetic data introduces a number of difficulties and restrictions. Firstly, with using a software such as Kazantsev *et al.*'s TomoPhantom toolbox [23] as presented by Table 6.3, generation of a synthetic tomogram requires approximately 24 hours. Therefore the generation of a time-series with 20 synthetic tomograms would require approximately 20 days of computation time. Furthermore, additional time will be needed for the creation of code that generates the synthetic time-series, which will contain temporal events that simulate real-world temporal events. Another restriction of synthetic time-series, is that due to the absence of tomogram misalignments (as explained in Section 7.2) and the less complex structure of the synthetic tomograms, a 3D network like Stacked-DenseUSeg is expected to segment them very accurately and with minimal temporal inconsistencies. This means that our proposed HMM models are also expected to offer very minimal improvements in the segmentation accuracy, and thus making more difficult to present their effectiveness and how they would compare when used on real-world data. Finally, while synthetic data may allow a more accurate estimation of the optimal transition probabilities between the classes, this advantage cannot be transferred to the real-world data since the synthetic and real-world time-series may have different optimal transition probabilities. Based on these facts, we decided to explore the testing our HMM approaches using synthetic tomograms with simulated temporal events in a future research.

On the other hand, a pair of highly-sampled tomograms collected respectively before and after the collection of a time-series, could be used to aid the performance of our HMM models. As mentioned also in Section 4.6, collection of 2 representative highly-sampled tomograms both before and after a time-series should be preferred by researchers who collect time-series of undersampled tomograms. Namely, using both tomograms it may be possible to train a 3D segmentation network using one of them, while testing on the other. This may result in more accurate testing of the 3D segmentation network and the more accurate estimation of its confusion matrix. Since the confusion matrix of the segmentation network is used in order to generate the observation matrix of our proposed HMM models, more a accurate confusion matrix will improve the accuracy of our HMMs

observation matrix. Additionally, for the highly-sampled tomogram used for testing, its refined segmentations after the application of Stacked-DenseUSeg and our hidden Markov models can be compared against accurate manual annotations. Therefore, this will allow an accurate estimation of the accuracy of our models on real-world data and in areas that temporal events probably occurred. Finally, by comparing the two highly-sampled tomograms, it may be possible to accurately estimate the transition probabilities of our HMM model's transition matrix.

Lastly, we would like to address the reported mean IoU score of all methods in Table 7.1 being close to 0.6. An IoU score of 0.6 is very low, however after considering of all relative factors and the fact that the ground truth used in Table 7.1 are annotations based on the highly-sampled tomogram, the achieved score is not unexpected. Specifically, as explained by Section 7.3.3 there is a number of "correctly" misclassified segmentations, due to the loss of spatial information by undersampling (collecting less projections) of the tomograms in the time-series. While our network Stacked-DenseUSeg may be able to restore/predict some spatial information, after it is trained on a representative highly-sampled tomogram, it is almost certain that a large portion of spatial information can not be recovered.

Nevertheless, the low IoU score reported cannot solely be attributed to the poor accuracy of Stacked-DenseUSeg combined with our HMM models and given the bad SNR of the tomograms in time-series. Moreover, the algorithm for tomogram alignment proposed in Section 7.2, does not align the tomograms in the time-series using subtoxel accuracy, thusly introducing additional error. In the future, a more accurate algorithm for tomogram alignment could be implemented, however it is expected that future acquisitions of time-series of tomograms would not suffer from this effect. This is because, it will not be necessary to move the sample in order to collect the flat field projections (see Section 3.1) for every tomogram in the time-series, thus eliminating the tomogram misalignments.

Finally, as explained in Section 7.4.3, the results of Table 7.1 are regarding an area where temporal events are unlikely to have occurred. However, it is possible that small scale temporal events (like thermal expansion and contraction, etc.) might occurred, which in-

troduces a very small error when measuring the IoU score. Based on the above, it becomes apparent that multiple factors are responsible for the reported low IoU score. Despite the small IoU score however, the final 4D segmentation are still very useful in practice. As of now there are hardly any annotated time-series of undersampled tomograms. As can be seen in Table 7.1, the poor IoU score of the spatially small and difficult to segment classes of Magnesium and Air Pocket are mainly responsible for the small mean IoU score. By utilising the final 4D segmentations and manually correcting the segmentations of Magnesium and Air Pocket, it is now possible by the researchers who collected the time-series, to obtain in a very fast manner accurate 4D annotations. Namely, the manual annotations used in Table 7.2, are created using this approach. Additionally, since our data and code is publicly available our approach can serve as a baseline for future approaches that attempt to automatically and semantically segment time-series of undersampled tomograms.

7.6 Conclusions

In this chapter we have presented a novel use of hidden Markov models for the refinement of multiple tomogram segmentations that comprise a time-series of tomograms. Namely, we presented a hidden Markov model, named HMM-T [29] (code available at [28]), that receives as input segmentation predictions generated by a 3D CNN (our Stacked-Dense-USeg). Our model then refines these predictions based on the expected transition probabilities of the classes through time. Additionally, it uses the emission/prediction accuracy of the network in the form of a confusion matrix to set its emission probabilities. HMM-T, using these probabilities and the temporally adjacent tomogram segmentations is able to produce output predictions with a high temporal consistency and accuracy. Secondly in addition to HMM-T, a second hidden Markov model is presented, named HMM-TC [29] (code available at [28]), that takes advantage not only of the prediction classes of the network, but also from certain probabilities it attributes to the segmentations, in other words its confidence for its predictions. This is performed by further splitting each of network's output classes to 3 different "bins" of low-confidence, mid-confidence and high-confidence, resulting in $3N$ new input classes where N is the number of the original classes

predicted by the network. For our models, the transition probabilities are empirically determined based on expert knowledge, and the emission probabilities are derived for the confusion matrix of the segmentation network during testing. Their overall execution is unsupervised, as there are not any high quality 4D annotations to train our HMMs.

As it can be seen in Section 7.4, the quality of the segmentation outputs after the use of the hidden Markov models is improved compared to the individual segmentations of the tomograms using a 3D segmentation network. Namely, the most profound outcome is the removal of sudden unexpected transitions of the classes through time, which is product of noise in the inputs (see Figure 7.6). Moreover, through the use of manually assigned annotation scenarios, the behaviour of the hidden Markov models is investigated (via the use of the HMM-T). Based on our observations it seems that the models are able to halt non-expected transitions to a degree. Naturally, the existence of multiple observations to the contrary, and the high emission probabilities that are a consequence of the good accuracy of the 3D CNN, do eventually allow for improbable transitions (see Figure 7.10). This reduced level of aggressiveness is a desired property of our models. Despite the expected transitions of classes, our models should not force the data towards a certain direction and potentially misclassify correct annotations.

Furthermore, we attempted to measure the accuracy of the segmentations after the use of our models. In areas that did not probably experience any temporal events, we see that both of our models are able to remove temporal inconsistencies and improve the input segmentations. From this test it is our HMM-T model that performs better. HMM-T does also performs better in areas that it is likely temporal events occurred, however we report this with some reservations due to the high correlation between the manual annotations used and the predictions of HMM-T. Nevertheless, the good quality of the HMM-T predictions are enough to prompt us to use them as templates for our manual annotations, which is itself an argument for the good performance of the model. On the other hand, our HMM-TC does not show the same level of accuracy in terms of IoU, however in Figures 7.12 and 7.14 its outputs do show desirable visual features. We suspect that with further investigation and better fine-tuning of the emission probabilities, it may

even surpass the accuracy of HMM-T due to the inclusion of confidence.

Finally, we also measured the sensitivity of the empirically elected transition probabilities, both on areas of the data where temporal events probably did not occurred or where they probably did. Based on the results, the elected transition probabilities of matrix 7.2, are the optimal ones. As expected, the accuracy of our HMMs segmentation refinements is more sensitive to the elected transition probabilities of the spatially small and more difficult to segment classes of Magnesium and Air Pockets. For this reason, it is recommended that future collections of time-series of undersampled tomograms would be accompanied by highly-sampled tomograms collected both at the beginning and the end of the collection of the time-series. This would then enable before-and-after comparisons between the highly-sampled tomograms, leading to more accurate estimations of the transition probabilities.

Concluding, after the generation and the inspection of 4D refined segmentations made by our models, 4D ground truths with an acceptable accuracy may be attainable. These then may be used to develop full 4D deep learning segmentation methods, making the semantic segmentation of time-series of X-ray CTs fully automatic.

Chapter 8

Conclusions and future work

8.1 Overview of the main contributions

Concluding the thesis, this last chapter summarises the overall achievements and contributions of the earlier presented work. As is mentioned throughout this thesis, the main goal is to address the problems of denoising and semantic segmentation of time-series of tomograms, also referred as in this work as 4D datasets (three spatial dimensions and a temporal dimension). As described in chapter 3, the time-series used during the experiment sections of the previous chapters are micro X-ray CTs collected with a reduced number of projections. The reason these time-series of tomograms are collected with undersampled (in regards to the projections) tomograms is due to time constraints during the collection. In each time-series, the tomograms have to be collected fast enough to ensure a good time-resolution, and collecting a large number of projections is very time-consuming. On the other hand, undersampled tomograms bring complications to the spatial resolution, in particular a low Signal-to-Noise Ratio (SNR), after they are reconstructed to 3D volumes. Based on these challenging conditions the focus of our work has a dual purpose, first to increase the SNR in reconstruction (denoising) and then to semantically segment them. Furthermore, as deep learning [6] have become widely adopted towards the solution of a plethora of imaging problems, our approaches utilise deep learning to achieve their goals.

Chapter 2 reviews previously proposed methods regarding denoising and semantic segmentation imaging data. In the literature, the relative terms for these tasks are Super-

Resolution and Semantic Segmentation respectively, and our review mainly focuses on methods that perform them using deep learning. Some of these approaches are the main inspiration for our contributions, offering helpful guidance and research information for the successful implementation of our approaches. Moreover, chapter 3 offers a detailed description regarding the data challenges in time-series computed tomography and essential information regarding the datasets used for our experiments.

Based on the literature research in chapter 2, in chapter 4, we describe our UDNN approach for fast upsampling of undersampled tomograms in time-series using a representative highly-sampled tomogram [24]. Our approach, applies a super-resolution network to upscale the tomograms' sinograms and increase their reconstructions' SNR. As Section 2.3.2) describes, the act of sinogram upscaling is translated to projection upsampling and, therefore, to a higher SNR due to the additional projections. Namely, our approach also has a light-weight structure allowing fast inference, and also fast training time. This is essential for the case of tomogram time-series as they are used in multiple studies and their imaging subjects may vary. Therefore, a universal example-based approach that can be obtain with a single training session, might be less helpful than a data-specific approach, which can be applied easily and retrained quickly. Additionally, in conjunction with the fast Filter Back Projection (FBP) [409] reconstruction algorithm, our approach can replace the use of time-consuming iterative reconstructive methods [411, 412, 413]. During experiments, UDNN is able to increase the reconstructions' SNR in both synthetic and real-world undersampled tomograms and achieve higher performance compared to analytical interpolation techniques [21, 22].

Building on this initial development, in chapter 5 we present our proposed network for both denoising and semantic segmentation of these reconstructions. While our fast UDNN approach improves the tomograms prior to their reconstruction, our Stacked-DenseUSeg [26, 27] approach not only further increases their SNR after their reconstruction, but at the same time offers their semantic segmentation. This is achieved by stacking two subnetworks of our proposed DenseUSeg architecture, that extends the previously-proposed DenseSeg [10] architecture with a decoder similar to that of U-Net [156]. The first subnet-

work in Stacked-DenseUSeg denoises the input (undersampled tomogram reconstructions) and increases its SNR while the second offers its semantic segmentation. Our approach, in a similar way to [315], "bridges" the two subnetworks with feature map skip connections, allowing the second subnetwork to achieve more accurate segmentations from the denoising feature maps. During experiments, both DenseUSeg and Stacked-DenseUSeg reported higher respective accuracy than state-of-the-art approaches in 3D semantic segmentation. DenseUSeg also performs either better or equal to the earlier state-of-the-art networks for the task of denoising the tomogram reconstructions. The denoised output of Stacked-DenseUSeg is only slightly less accurate than the one of DenseUSeg, it is visual quality is noteworthy compared to the state-of-the-art methods.

Then chapter 6, addresses the issue of limited annotations in representative highly-sampled tomograms and the difficulty to generate more. Naturally, for each new variety of time-series of tomograms, new representative highly-sampled tomograms would have to be annotated manually. This is of course a very time-consuming process, that can halt the application of automated semantic segmentation approaches. For this reason, we proposed a knowledge transferring scheme [27] for our Stacked-DenseUSeg, utilising synthetic tomograms that are generated pre-segmented. In particular, an initial Stacked-DenseUSeg (pretrained net) is trained with synthetic data, and its parameters are used to initialise the parameters of a second Stacked-DenseUSeg that will train/fine-tune on real-world tomograms. During experiments, we discovered the characteristics of the synthetic tomograms play a pivotal role in the success of this scheme. Particularly, the inclusion of rotating artifacts is essential for obtaining improved segmentation results. Additionally with this method, the overall segmentation accuracy is increased, while at the same time the amount of real-world annotated data can be significantly reduced without an ill-effect to the accuracy. Furthermore, by performing the manual annotation in parallel with the creation of synthetic tomograms and the pretrained nets, our proposed knowledge transfer does not add any additional time delays. On the other hand, it may lead to the development of accurate Stacked-DenseUSeg networks in less time, by avoiding to perform the time-consuming operation of annotating an entire real-world highly-sampled tomogram

and instead only *partially* annotating it.

Finally, chapter 7 discusses the issue of "uniting" multiple 3D segmentation of tomograms that are part of a time-series to provide a true 4D segmentation of the entire time-series. Obviously simply concatenating the individual segmentations of the time-series' tomograms is the simplest approach. However, inter-tomogram information is still not utilised. For this reason, we propose a novel use of hidden Markov models (HMMs) to refine the multiple 3D segmentations using temporal information (HMM-T) [29]. Namely, we propose our HMM-T model that uses temporal information through the time-series of tomogram segmentations, and our HMM-TC model [29] which also utilises the confidence (prediction probabilities) of the segmentation network (Stacked-DenseUSeg) for these segmentations. We do not set our hidden Markov model parameters using training examples, since there is an absence of high-quality 4D annotations. Instead, our models' parameters are partially hand-picked, and partially set by metrics output from our Stacked-DenseUSeg. Their transition matrix is hand-picked, and the physical phenomena that are expected to occur are used as guideline when doing this. On the other hand, the emission matrix is derived by using Stacked-DenseUSeg's confusion matrix during testing.

In the experiment section, we observed that the visual quality of the segmentations is improved and that time inconsistencies in the segmentations are removed. The models' behaviour is also tested under various segmentation conditions, showing no signs of unexpected reactions. Additionally, the accuracy of their segmentation refinement is measured both in areas where real-life temporal phenomena potentially occurred during data capture, and in areas where they potentially did not. For the areas potentially unaffected by temporal phenomena, the annotation of the representative highly-sampled tomogram is copied for each tomogram of the time-series. In the areas potentially affected by temporal phenomena, we use a small amount of 4D annotations which are collected manually and with our best effort to be accurate. In the areas potentially unaffected by temporal phenomena areas, both HMM models improved the segmentation accuracy, and in the areas potentially affected by temporal phenomena, only our HMM-T demonstrated better results. However, for our manual 4D annotations, there is probably a high correlation

between these and the HMM-T results, since the latter was used as a starting point for the former. Nevertheless, the fact that the HMM-T refined segmentations after visual inspection are accurate to the point they are best possible starting point for manual annotation, is itself an argument for the accuracy of our model. Regarding our HMM-TC, is seems it is not measurably better than our HMM-T from the calculated metrics, however visual results suggest otherwise. For this reason we argue that with better tuning of its parameters it should be possible to surpass the HMM-T in accuracy.

Based on the above, with the combined application of our approaches, we can achieve the denoising and the semantic segmentation of tomogram time-series, also referred as 4D datasets. Namely, our UDNN and Stacked-DenseUSeg can offer denoising solutions respectively, before and after the reconstruction of the time-series's tomograms. Additionally, our Stacked-DenseUSeg with parameter initialisation and with our hidden Markov models, can offer very accurate semantic segmentations for entire time-series. In the next Section, we will describe some of the limitations that exists regarding our proposed approached.

8.2 Limitations

8.2.1 Limitations of our UDNN approach

While our proposed methods lead to many contributions, there is a number of limitations associated with them that we would like to address in this Section. Starting with our first proposed approach UDNN, described in chapter 4, there are some limitations, regarding the testing of our approach, that exist due to low amount of available data. Namely, when testing on real-world data, multiple highly-sampled tomograms are needed in order to validly estimate the standard deviation of our approach. As explained in Section 4.4.6, during testing, the sinograms selected from a testing tomogram, equally sample the height dimension. This allows the testing subset of sinograms to be representative to all sinograms in the testing tomogram. If a different subset of testing sinograms where selected in the same way from the same testing tomogram, then the two testing subsets

would be highly correlated, and the estimation of the standard deviation would not be valid. For this reason multiple highly-sampled tomograms that capture the same physical experiment, but on different physical samples are required in order to accurately estimate the standard deviation of our approach. However, this is not usually the case and oftenly for every physical experiments only one or two time-series of undersampled tomograms are collected. For each of the time-series, usually one highly-sampled reference tomogram is collected either at the beginning or the end of the time-series' collection and very rarely two tomograms are collected both at the beginning and the end. For the experiments presented in Section 4.5, two real-world highly-sampled tomograms are used, both captured at the end of a time-series and performed on different samples with similar characteristics (both being aluminium pins with magnesium deposits holding a droplet of salt-water). One of the real-world tomograms (*RealHS1*) is used for training while the other (*RealHS2*) for testing. If however, as we mention in Sections 4.6 and 7.5, the common practice was to always collect a pair of highly-sampled reference tomograms, then there would be four available highly-sampled tomograms, allowing three independent tomograms to be used for testing and thus permitting an accurate estimation of the standard deviation of the accuracy our approach. Apart from that, even in cases where only one time-series is collected per physical experiment, the collection of two highly-sampled tomograms (both at the beginning and the end) may always allow the accurate measurement of the accuracy of our approach on an independent highly-sampled tomogram that was not used for training.

Another limitation of our UDNN approach is that we decided to test only a 3-layer CNN architecture, leaving the testing of deeper architectures for future research. Based on the findings of Dong *et al.* [33], deeper architectures, that use more sequential convolutional layers, offered diminishing returns regarding the upscaling of colored *natural* images. Apart from that, our approach compared to the one of Dong *et al.* [33] is regarding less complex data (tomograms' sinograms) with one modality compared to the three modalities of coloured images that Dong *et al.*'s approach is designed to upscale. Additionally, our approach is designed to upscale tomograms' sinograms in one direction, along their θ-

axis, which is a less complex task compared to the upscaling of colored images in both their dimensions (height and width). Finally, our approached is designed to operate using large kernels in order to increase the receptive field of each layer, which is very useful regarding the upscaling of undersampled tomograms (see Section 4.4.8). Therefore the addition of more layers would significantly increase the training and inferring time, making our approach less competitive in terms of time compared to the other state-of-the-art analytical interpolation techniques. In summary, the diminishing returns of deeper architectures reported by *et al.* [33], the smaller complexity of sinogram upscaling compared to the upscaling colored *natural* images and the higher computational time of additional layers due to the use of large kernels per layer, discouraged us from testing deeper architectures. Nevertheless, the potential use of more layers may allow major improvements in accuracy and we believe it should be further examined in the future.

Furthermore, due to the downscaling rate between the highly-sampled representative tomograms and the undersampled tomograms of the respective time-series, we decided to downsample the highly-sampled tomograms by a factor of x40, therefore matching the number of projections (from 3601 projections to 91 projections) present in the time-series tomograms. We then investigated upsampling these projections by a factor of x2 and x4. Therefore, the testing our approach is limited to the use of just these specific number of projections. For this reason, we believe that in the future, more downsampling (and potentially more upsampling rates) should be used in order to test the behaviour our approach regarding the upsampling of tomograms with number of projections other than 91. This is because for instance, the task of upsampling by a factor x2 of a tomogram with 1001 initial projections (therefore increasing it to 2001 projections), might be performed more or less accurately, compared to the upsampling by a factor x2 of a tomogram with only 91 initial projections (therefore increasing it to 181 projections). Thusly, the accuracy of our approach on upsampling undersampled tomograms with other than 91 initial projections is something that should be explored in the future.

8.2.2 Limitations of our DenseUSeg and Stacked-DenseUSeg approaches

Moving on, there is a number of limitations regarding our DenseUSeg and Stacked-DenseUSeg approaches described in chapter 5. Namely, both of our approaches are tested using a single highly-sampled tomogram that is collected as a reference to a time-series of undersampled tomograms. Because of that, the behaviour of our approaches is not tested on reference highly-sampled tomograms of time-series that capture temporal events of other physical experiments. Despite that however, since major elements of our proposed architecture (like the encoder of DenseUSeg being similar to the DenseSeg architecture, DenseUSeg's decoder also being used in architectures like 3D-UNet, etc.) are similar to the ones present in other 3D state-of-the-art semantic segmentation architectures, it is expected that the accuracy of our approaches will remain comparable to them on different datasets. Nevertheless, testing on different highly-sampled tomograms of time-series which capture different physical experiments should be explored in the future. This could further be aided, by the increase of the publicly available real-world time-series of undersampled tomograms (with their respective highly-sampled tomograms). For this reason, we have made the real-world data used throughout this thesis publicly available [424, 425, 426].

Another limitation of our approach is the use of one highly-sampled representative tomogram for both for training and testing. This is because, usually a single highly-sampled representative tomograms is collected for each time-series either before or after its collection. While the use of non-overlapping training instances derived by our single highly-sampled tomogram reduces the risk of high correlation between the training and testing set, it is not as ideal as the use of two different reference tomograms. For this reason, as we mention in Sections 4.6 and 7.5, we believe that the collection of a pair of highly-sampled representative tomograms before and after the collection of a time-series of undersampled tomograms can be very beneficial in the future.

Additionally, as mentioned on Section 5.3.6, the way with which feature maps are shared between the denoising DenseUSeg subnetwork and segmentation DenseUSeg subnetwork

in the Stacked-DenseUSeg architecture, may potentially be further improved. This is because, only the methodology proposed by Newell *et al.*'s [315] is tested, which sums a number of feature maps in order to form the input for the second DenseUSeg subnetwork. However, similar to the replacement of the summation operations with concatenation ones in DenseNets [143], this summation of feature maps could be replaced with a concatenation and a subsequent "bottleneck" $1 \times 1 \times 1$ convolution layer that reduces the number of feature maps. Essentially, the concatenation and the "bottleneck" convolution layer, work to create new feature maps with a weighted summation and the weights are learned during the Stacked-DenseUSeg's training. Apart from that, the contribution of the feature maps that are summed (marked as **B**, **C** and **D** in Figure 5.6) could be investigated and potentially be summing or concatenating only the ones that are the most useful, execution times may be reduced without sacrifices to the segmentation accuracy.

8.2.3 Limitations of our knowledge transfer scheme for Stacked-DenseUSeg

Regarding our proposed knowledge transfer scheme described in chapter 6, the aforementioned limited testing using a single real-world highly-sampled tomogram also applies here. Furthermore, due to the use of synthetic tomograms for the training of pretrained nets (used for parameter initialisation of networks that will be trained on real-world data), the similarity between synthetic and real-world data plays an important role. In principal, good synthetic analogs of real-world data used to generate pretrained net, should be obtainable with the use of the available geometrical shapes in the TomoPhantom toolbox [23] and constructive solid geometry [450, 451]. However, this is something that it should be verified in the future, on different real-world highly-sampled tomograms of time-series that capture temporal events of different physical experiments. This may lead to better estimations of the time benefits, associated to the fact that pretrained nets may allow networks to achieve high segmentation accuracy without the need to manually annotate large amounts of real-world data.

Moreover, another limitation of our proposed knowledge transfer scheme, is that the

generation of synthetic datasets and pretrained nets requires a significant amount of time. As described in Section 6.5, it is important for the user to simultaneously perform the manual annotation of the real-world data while the synthetic data and pretrained nets are generated, in order reduce or nullify any potential time delays associated with our knowledge transfer scheme. However, if the user omit to start the rendering of synthetic data and generation of pretrained nets, prior to him/her start performing the manual annotation of real-world data, this will result in unnecessary time delays. In addition, for the testing of our knowledge transfer scheme, a fully annotated real-world highly-sampled tomogram is used while progressively less training data are selected. Nevertheless, during the application of our knowledge transfer scheme, the user will progressively annotate more and more data. One important aspect of this operation is, which portion of the real-world highly-sampled tomogram he/she should annotate first. As explained in Section 6.5, effort should be made in order to the annotate a small, but representative portion of the real-world data. Nonetheless, the applicability of this general guideline should be tested in the future, on multiple real-world highly-sampled tomograms of time-series that capture temporal events of various physical experiments.

8.2.4 Limitations of our hidden Markov models and the refining of 4D segmentations

Lastly, there is also a number of limitations associated with the refining of the 4D segmentations, described in chapter 7, that are generated by the application of our 3D Stacked-DenseUSeg on a time-series of undersampled tomograms independently through time. As described in Section 7.5, the algorithm for tomogram alignment proposed in Section 7.2 does not align the tomograms in the time-series using subtoxel accuracy. This means that a small number of toxels, do not consistently through time contain information regarding the same spatial position, and therefore after the use of our hidden Markov models error may be introduced. In the future, it is expected that new acquisitions of time-series of tomograms would not suffer from this effect. This is because movement of the sample will not be necessary in order to collect the flat field projections (see Section 3.1) for every tomogram in the time-series.

Another limitation regarding the testing of our hidden Markov models, is the estimation of their accuracy on areas of the sample that experience temporal events. The main motivation behind the utilisation of hidden Markov models for the refinement of the 4D segmentations (made by our 3D Stacked-DenseUSeg after applying it on a time-series of tomograms independently through time), is their ability to be applied without the need for 4D annotated data. This is because the hidden Markov models' two major components are determined without the use of annotated data. Their observation matrix is be derived by Stacked-DenseUSeg's confusion matrix and their transition matrix is determined empirically. The lack of 4D annotated data is because the tomograms collected within a time-series are undersampled. This leads to their reconstruction having low SNR, which severely hinders the ability to create accurate manual annotations. Thusly without the existence of accurate 4D annotations, our hidden Markov models can not be tested accurately for areas of the sample that experience temporal events. In Section 7.4.3, we describe the use of manually annotated 4D data for the areas of the sample that experience temporal events. To obtain manual annotations that are as accurate as possible, the predictions of our HMM-T model are modified using the low SNR reconstruction of the time-series as guidance. As we mention, since the predictions of our HMM-T model are used as a starting point, our manual annotations have high correlation with the HMM-T predictions and any result that we report is with some reservations. On the other hand, for areas that probably do not experience temporal events, the accurate annotations of the reference highly-sampled tomograms are used.

Alternatively, synthetic time-series of tomograms could be used to test the accuracy of our approaches, but this option has itself some inherent limitations. As explained in Section 7.5 using Kazantsev *et al.*'s TomoPhantom toolbox [23], the generation of a synthetic tomogram requires as presented by Table 6.3 approximately 24 hours. Therefore the generation of a time-series with multiple synthetic tomograms would be very computationally demanding, and it would require the allocation of useful computational resources for a prohibiting long amount of time. Furthermore, the synthetic time-series due to the absence of tomogram misalignments (solved for real-world data using the algorithm

of Section 7.2) and due to its simplified structure constructed by TomoPhantom's [23] capabilities, is highly likely that it will be segmented with a very high accuracy by our proposed Stacked-DenseUSeg, even if applied it is to the time-series' tomograms independently through time. This will lead to existance minimal temporal inconsistencies in the 4D segmentations, and potentially very minimal improvements after the applications of our proposed hidden Markov models. Because of this, the results reported on a synthetic time-series would not be representative to the use-case of real-world time-series. Finally, on synthetic time-series the optimal transition probabilities can easily be found. However, the real-world time-series that the synthetic time-series is an analog to, may have very different optimal transition probabilities. This means that the optimal transition probabilities for the synthetic time-series, may not be useful in order to determine the optimal transition probabilities of the real-world time-series.

On the other hand, a pair of highly-sampled tomograms collected respectively before and after the collection of a real-world time-series may be helpful. As mentioned in 4.6, these tomograms may allow the accurate estimation of the accuracy of the segmentation refinements made by our HMMs, in areas that temporal events probably occurred. Additionally, they may offer an accurate estimation of Stacked-DenseUSeg's confusion matrix and therefore an accurate estimation of our HMMs' observation matrix. Finally, by comparing the two highly-sampled tomograms, it may be possible to obtain better estimations for the transition probabilities of the different classes in the time-series.

Concluding, as explained in Section 7.3.3, for our proposed HMM-TC model a number of its observation probabilities has to modified due to existence of "correctly" misclassified toxels in the segmentations provided by the Stacked-DenseUSeg. As we point out, there is a portion of spatial information that can not be recovered due to the low SNR of the input tomogram reconstructions. This inevitably leads to toxel misclassification, as our HMM-TC informed by its observation matrix that is derived by Stacked-DenseUSeg's confusion matrix, predicts commonly misclassified classes (in our case mostly Air Pockets) in areas of low confidence of other classes (like low confidence aluminium or magnesium). Nevertheless, the areas of low confidence of the other classes, may be caused due to the

proximity to the boundaries between different class instances, as shown in Figure 7.3 where, low confidence Magnesium toxels typically surround areas of high confidence Magnesium. This behaviour and the erroneous predictions of commonly misclassified classes (in our case mostly Air Pockets) is shown in Figure 7.4. In order to address this issue, we manually altered certain entries of HMM-TC's observation matrix. While this approach resolved the erroneous prediction of commonly misclassified classes, it potentially affects the overall accuracy of HMM-TC. For this reason, alternative ways of determining HMM-TC's observation matrix should be explored in the future.

8.3 Future Work

During the development of this thesis' contributions, a lot of research was conducted in certain areas of our methods allowing them to achieve the reported results. Nevertheless, there are still some areas where further research may lead to future improvements.

8.3.1 Potential improvements on our UDNN approach

In particular, for our UDNN, as we discuss in Section 4.6 a future idea would be the introduction of weights to the small components of the sinograms allowing our method to be accurate with higher upscaling factors. This may be accomplished by applying a projection algorithm, similar to the one applied to synthetic phantoms, on the real-world tomogram annotations, and thus generate "segmentations" in the sinogram space. These could then be used to weight the error of each output pixel of UDNN differently. Alternatively, if annotated real-world, highly-sampled tomograms are not available, a solution similar to the one presented in chapter 6 could be used. Namely, an initial UDNN, could be first trained using synthetic data which are generated with annotations and then a new UDNN could be trained on real-world data using the previous for parameter initialisation.

Finally, our UDNN architecture could be potentially to be improved with the introduction of residual learning [129], densely connected layers [143] and/or dilated convolutions [322]. Especially for the latter, they might be able to replace the large kernels of UDNN, allowing

for inexpensive deeper architectures with a larger final receptive field. Naturally since for sinogram upscaling useful information is stored in the sinusoidal traces mostly placed along the *x-axis*, dilated convolutions with different dilation rates for each axis might be useful.

8.3.2 Potential improvements on our DenseUSeg and Stacked-DenseUSeg approaches

Furthermore apart from UDNN, there are future research ideas for our DenseUSeg, and Stacked-DenseUSeg too. As is described in Section 5.3.6, one of the improvements that may implemented in order to improve Stacked-DenseUSeg's performance is by replacing the summation operation used to link the two *stacked* DenseUSegs, with a concatenation followed by a $1 \times 1 \times 1$ convolutional layer that act as a "bottleneck". This substitution, may reduce the number of feature maps in a more trainable fashion. This is because, the concatenation and the follow-up $1 \times 1 \times 1$ convolution, essentially generate new feature maps by weightily summing the previous-concatenated feature maps and these weights can be learned during the training of Stacked-DenseUSeg.

Furthermore, as mentioned in Section 5.4.4 by equation 5.5, the overall loss of our Stacked-DenseUSeg is the combination of the Mean Square Error for its denoising subnetwork and the Cross Entropy Loss for its segmentation subnetwork. Specifically, this combination is achieved with a weighting parameter λ that balances the two losses. Alternatively to manually setting it, a statistical approach could be used to dynamically adjust it. For instance, moving averages could be used to measure the two losses during training to adjust the λ, however extra care has to be applied in the initial epochs. During the initial epochs, faster converging of the denoising part of Stacked-DenseUSeg is significant and without sufficient data to accurately estimate the averages of the two losses, problems might occur. Additionally, apart from Stacked-DenseUSeg, our DenseUSeg's accuracy might improve by increasing the number of convolutional layers that process feature maps at different scales in both the encoder and the decoder. Naturally, this would increase the memory demand of the network, as well as its training and inference times. A potential adjustment

to alleviate this, would be to reduce the growth rate ($g = 64$) of the Dense Blocks, and at the same time decrease the Dropout rate $p = 0.2$ to keep the number of affect channels the same. Additionally, an adjustment that may improve the accuracy without potentially increasing its memory needs, would be to replace the convolutional layers in the decoder with residual convolutional layers. After experimenting with the above variants of DenseUSeg, different combinations of them in Stacked-DenseUSeg could be investigated in order to obtain a higher denoising and segmentation accuracy. Finally, as technology progresses and older hardware limits will become obsolete, deeper architectures could be applied to increase the accuracy further.

8.3.3 Potential improvements on our knowledge transfer scheme for Stacked-DenseUSeg

Regarding our knowledge transfer scheme described in chapter 6, as discussed on Section 6.4.1, one future idea would be to use a synthetic undersampled tomogram *with* rotating artifacts as input and a highly-sampled tomogram *without* such artifacts as a denoising ground truth. This combination might improve the denoising accuracy of Stacked-DenseUSeg and potentially even its segmentation accuracy. Likewise, it can be speculated that by testing more combinations using new synthetic data that are more realistic with new artifacts, higher accuracy in both denoising and segmentation might be possible for the network fine-tuned with real-world data. In any case, as software solutions like Tomo-Phantom [23] would become more sophisticated in the future, synthetic tomograms would become more realistic, with more accurate physical effects and would likely be faster and easier for a user to generate.

A potential deep learning application that would increase the synthetic tomogram's realism further would be with a use of a Generative Adversarial Network [135]. Namely, the input to the generator would not be noise but a pair of a synthetic tomogram reconstruction and its phantom (segmentation annotations). The generator would then attempt to modify the pair and to appear more "realistic" compared to real-world tomogram that would be provided to the discriminator. Obviously, this generator should be designed to

offer output phantom/annotations that are accurate segmentations of its output synthetic tomogram reconstruction. Potentially the application of a CycleGAN [457] might help towards this goal.

8.3.4 Potential improvements our hidden Markov models and the refining of 4D network predictions

To conclude this Section, there are some future ideas for our proposed hidden Markov models in chapter 7. As it was mentioned in Section 7.6, our HMM-TC model even though reported lower IoU scores than our HMM-T, has very noteworthy visual results. Based on this, we believe that with a better fine-tuning of the emission matrix (having in mind the "correct" misclassified voxels mentioned in Section 7.3.3), further improvement can be achieved. Moreover, new configurations of how to "bin" the predicted classes may offer better result. For instance, instead of quantising the input predicted classes to three "bins" according to the probability of the input class with the higher probability, the following split could be performed. There could be a low-confidence and high-confidence "bin" as before, but the mid-confidence "bin" could be split into three new "bins" according to which class is the second most probable (based on the Stacked-DenseUSeg's predictions).

Apart from new configurations for HMM-TC, the collection of a small portion of valid 4D annotations may improve the HMMs even further. In particular, they may be used to obtain a more accurate transition matrix. In any case though, for the segmentation of multiple similar time-series, where retraining of the Stacked-DenseUSeg is not necessary, the collection of a few 4D annotations may allow more complex approaches. One of them could be the application of Bi-Directional Convolutional LSTMs (BDC-LSTMs) [400] in conjunction with our Stacked-DenseUSeg, resulting in a 4D network that automatically infers the segmentations for entire tomogram time-series.

Finally, apart from segmentation refining models, denoising refining models could also be developed. Namely, using the segmentation annotations it may be possible to find normal distributions for the output intensity of each class in the denoised output. Then these distributions could be used to alter the intensity (pixel value) of each pixel in the denoised

output, based on the refined classes from HMM-T or HMM-TC.

8.4 Concluding remarks

Concluding the thesis, the combined use of the methods presented here enables the processing of significant quantities of time-series of undersampled tomograms which are collected at synchrotron locations, like the one of Diamond Light Source (Harwell, Oxford). The aim of this work is to enable researchers who use time-series of undersampled CT collections (4D datasets) to quickly and easily both denoise and semantically segment these collection, facilitating the extraction of useful information from the torrent of 4D data.

With help of our proposed UDNN approach, described in chapter 4, it is now possible to increase the SNR of tomogram reconstructions in time-series, using an approach that is comparable in terms of the accuracy with other state-of-the-art sinogram interpolation techniques. However, our approach is faster for the use-case of time-series of tomograms. Additionally, since our approach is trained using highly-sampled reference tomograms, it may be able to handle temporal artifacts present in the sinograms of the tomograms (see Figure 3.5 Marker 1) that are associated with fast occurring temporal events occurring during the capture of the undersampled tomograms.

Moving on, with the use of our DenseUSeg approach, described in chapter 5, it is now possible to semantically segment the time-series' reconstructions with better accuracy than other state-of-the art 3D segmentation convolutional networks. In addition, further accuracy can be obtained with the use of our proposed Stacked-DenseUSeg approach, which additionally outputs a denoised version of the time-series' 3D reconstructions. This secondary output is calculated in an end-to-end fashion along with the network's semantic segmentations. Potentially it may be used for visualisation purposes or for further analysis of the time-series, using algorithms that require high SNR reconstructions as input.

Moreover, our knowledge transfer scheme, described in chapter 6, may allow fast training of our Stacked-DenseUSeg approach without sacrifices in the overall accuracy, using less

manually annotated data. Namely, this is achieved by training on synthetic tomograms and using the resulting nets for parameter initialisation of networks trained on real-world data. As we report, using only 5% of the annotations of a fully annotated highly-sampled tomogram to fine-tune a pretrained net trained on synthetic data, it is possible to obtain networks that have high segmentation accuracy. In particular, the reported accuracy is higher than that of Stacked-DenseUSeg network randomly initialised and trained on a fully annotated highly-sampled tomogram. Based on this, and following the flowchart presented by Figure 6.14, it is now possible to obtain in potentially less time and with less human involvement, highly accurate Stacked-DenseUSeg networks that can segment real-world data.

Finally, with the use of our proposed hidden Markov models in chapter 6, it is now possible to resolve temporal inconsistencies present in Stacked-DenseUSeg's segmentations of time-series' reconstructions. These temporal inconsistencies are generated, because Stacked-DenseUSeg is used to segment the time-series' reconstructions independently through time. As we present with the use of behavioural, qualitative and quantitative experiments, after the application of our hidden Markov models, inter-tomogram information is utilised within the time-series. This leads to the removal of temporal inconsistencies due to misclassification by Stacked-DenseUSeg, but physically viable temporal changes are allowed to manifest in the segmentations of the time-series's tomograms.

Bibliography

[1] C. Richmond, "Sir Godfrey Hounsfield," *BMJ : British Medical Journal*, vol. 329, no. 7467, p. 687, 9 2004. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC517662/

[2] P. C. LAUTERBUR, "Image Formation by Induced Local Interactions: Examples Employing Nuclear Magnetic Resonance," *Nature*, vol. 242, no. 5394, pp. 190–191, 1973. [Online]. Available: https://doi.org/10.1038/242190a0

[3] P. Mansfield and P. K. Grannell, "NMR 'diffraction' in solids?" *Journal of Physics C: Solid State Physics*, vol. 6, no. 22, pp. L422–L426, 11 1973.

[4] M. Knoll and E. Ruska, "Das Elektronenmikroskop," *Zeitschrift für Physik*, vol. 78, no. 5, pp. 318–339, 1932.

[5] R. Rüdenberg, "Elektronenmikroskop," *Naturwissenschaften*, vol. 20, no. 28, p. 522, 7 1932.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[7] H. R. Roth, A. Farag, L. Lu, E. B. Turkbey, and R. M. Summers, "Deep convolutional networks for pancreas segmentation in CT imaging," in *Medical Imaging 2015: Image Processing*, S. Ourselin and M. A. Styner, Eds., vol. 9413. SPIE, 3 2015. doi: 10.1117/12.2081420 pp. 378–385. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2081420

[8] O. Cicek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net : Learning Dense Volumetric," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2016*, S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, Eds. Springer International Publishing, 2016. doi: 10.1007/978-3-319-46723-8. ISBN 9783319467238. ISSN 10636919 pp. 424–432.

[9] H. Chen, Q. Dou, L. Yu, J. Qin, and P. A. Heng, "VoxResNet: Deep voxelwise residual networks for brain segmentation from 3D MR images," *NeuroImage*, vol. 170, pp. 446–455, 2018.

[10] T. D. Bui, J. Shin, and T. Moon, "3D Densely Convolutional Networks for Volumetric Segmentation," *arXiv:1709.03199 [cs]*, 9 2017. [Online]. Available: http://arxiv.org/abs/1709.03199

[11] R. Zhang, L. Zhao, W. Lou, J. M. Abrigo, V. C. Mok, W. C. Chu, D. Wang, and L. Shi, "Automatic Segmentation of Acute Ischemic Stroke

from DWI using 3D Fully Convolutional DenseNets," *IEEE Transactions on Medical Imaging*, vol. 37, no. 9, pp. 2149–2160, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8328863/

[12] Y. Liu and Y. Zhang, "Low-dose CT restoration via stacked sparse denoising autoencoders," *Neurocomputing*, vol. 284, pp. 80–89, 2018.

[13] H. Chen, Y. Zhang, W. Zhang, P. Liao, K. Li, J. Zhou, and G. Wang, "Low-dose CT denoising with convolutional neural network," in *Proceedings - International Symposium on Biomedical Imaging*, 2017. doi: 10.1109/ISBI.2017.7950488. ISBN 9781509011711. ISSN 19458452 pp. 143–146.

[14] H. Chen, Y. Zhang, M. K. Kalra, F. Lin, Y. Chen, P. Liao, J. Zhou, and G. Wang, "Low-Dose CT with a Residual Encoder-Decoder Convolutional Neural Network (RED-CNN)," *IEEE Transactions on Medical Imaging*, vol. 36, no. 12, pp. 2524–2535, 2017.

[15] E. Kang, J. Min, and J. C. Ye, "A deep convolutional neural network using directional wavelets for low-dose X-ray CT reconstruction," *Medical Physics*, vol. 44, no. 10, pp. e360–e375, 2017.

[16] E. Kang, W. Chang, J. Yoo, and J. C. Ye, "Deep Convolutional Framelet Denosing for Low-Dose CT via Wavelet Residual Network," *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1359–1369, 2018.

[17] C. You, Q. Yang, H. Shan, L. Gjesteby, G. Li, S. Ju, Z. Zhang, Z. Zhao, Y. Zhang, W. Cong, and G. Wang, "Structurally-sensitive multi-scale deep neural network for low-dose CT denoising," *IEEE Access*, vol. 6, pp. 41 839–41 855, 2018.

[18] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980. [Online]. Available: https://doi.org/10.1007/BF00344251

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 11 1998.

[20] A. G. Ivakhnenko, "Polynomial Theory of Complex Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-1, no. 4, pp. 364–378, 10 1971.

[21] C. de Boor, "Bicubic Spline Interpolation," *Journal of Mathematics and Physics*, vol. 41, no. 1-4, pp. 212–218, 4 1962. [Online]. Available: http://doi.wiley.com/10.1002/sapm1962411212

[22] M. Bertram, J. Wiegert, D. Schafer, T. Aach, and G. Rose, "Directional View Interpolation for Compensation of Sparse Angular Sampling in Cone-Beam CT," *IEEE Transactions on Medical Imaging*, vol. 28, no. 7, pp. 1011–1022, 2009. [Online]. Available: https://ieeexplore.ieee.org/document/4738374/?arnumber=4738374

[23] D. Kazantsev, V. Pickalov, S. Nagella, E. Pasca, and P. J. Withers, "TomoPhantom, a software package to generate 2D–4D analytical phantoms for CT image reconstruction algorithm benchmarks," *SoftwareX*, vol. 7, pp. 150–155,

6 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711018300335

[24] D. Bellos, M. Basham, T. Pridmore, and A. P. French, "A convolutional neural network for fast upsampling of undersampled tomograms in X-ray CT time-series using a representative highly sampled tomogram," *Journal of Synchrotron Radiation*, vol. 26, no. 3, 5 2019. [Online]. Available: http://scripts.iucr.org/cgi-bin/paper?S1600577519003448

[25] D. Bellos, M. Basham, A. P. French, and T. Pridmore, "DimitriosBellos/UDNN: UDNN, https://zenodo.org/record/1299347," 10 2018. [Online]. Available: https://zenodo.org/record/1299347#.W9csG5P7SUk

[26] D. Bellos, A. P. French, M. Basham, and T. Pridmore, "A Stacked Dense Denoising-Segmentation Network for low-projection CT Volumes," in *British Machine Vision Conference (BMVC) 2019 Workshop on Visual Artificial Intelligence and Entrepreneurship (VAEI)*, 2019. doi: Pending DOI from BMVA. [Online]. Available: https://dculibrk.github.io/bmvc-vaie/2-programme.html

[27] D. Bellos, M. Basham, T. Pridmore, and A. P. French, "A stacked dense denoising–segmentation network for undersampled tomograms and knowledge transfer using synthetic tomograms," *Machine Vision and Applications*, vol. 32, no. 3, p. 75, 2021. [Online]. Available: https://doi.org/10.1007/s00138-021-01196-4

[28] D. Bellos and D. Bellos, "DimitriosBellos/Stacked-DenseDenSeg: Knowledge transfer and HMMs," 1 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4416013#.X_Nath28S1k.mendeley

[29] D. Bellos, M. Basham, A. P. French, and T. Pridmore, "Temporal refinement of 3D CNN semantic segmentations on 4D time-series of undersampled tomograms using hidden Markov models," *Nature Scientific Reports, under revision*, 2021.

[30] J. K. O'Regan and A. Noë, "A sensorimotor account of vision and visual consciousness," *Behavioral and Brain Sciences*, 2001.

[31] H. Chang, D. Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 2004. doi: 10.1109/cvpr.2004.1315043. ISSN 10636919 pp. I–I.

[32] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[33] C. Dong, C. C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[34] R. Timofte, V. De, and L. V. Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *2013 IEEE International Conference on Computer Vision*, 2013. doi: 10.1109/ICCV.2013.241. ISBN 9781479928392 pp. 1920–1927.

[35] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Computer Vision – ACCV 2014*, vol. 9006. Springer, Cham, 2015. doi: 10.1007/978-3-319-16817-3_8. ISSN 16113349 pp. 111–126. [Online]. Available: http://link.springer.com/10.1007/978-3-319-16817-3_8

[36] R. Zeyde, M. Elad, and M. Protter, "On Single Image Scale-Up Using Sparse-Representations," in *Curves and Surfaces*, J.-D. Boissonnat, P. Chenin, A. Cohen, C. Gout, T. Lyche, M.-L. Mazure, and L. Schumaker, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN 978-3-642-27413-8 pp. 711–730.

[37] R. Tsai and T. S. Huang, "Multiframe Image Restoration and Registration," *Advances in Computer Vision and Image Processing*, vol. 1, no. 2, 1984.

[38] S. P. Kim, N. K. Bose, and H. M. Valenzuela, "Recursive reconstruction of high resolution image from noisy undersampled multiframes," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 6, pp. 1013–1027, 6 1990.

[39] S. P. Kim and W.-Y. Su, "Recursive high-resolution reconstruction of blurred multiframe images," *IEEE Transactions on Image Processing*, vol. 2, no. 4, pp. 534–539, 10 1993.

[40] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, F. John, Ed. V. H. Winston and Sons, 1979, vol. 21, no. 2. [Online]. Available: https://doi.org/10.1137/1021044

[41] S. Rhee and M. G. Kang, "Discrete cosine transform based regularized high-resolution image reconstruction algorithm," *Optical Engineering*, vol. 38, no. 8, pp. 1348–1356, 1999. [Online]. Available: https://doi.org/10.1117/1.602177

[42] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, 1 1974.

[43] N. Nguyen and P. Milanfar, "A wavelet-based interpolation-restoration method for superresolution (wavelet superresolution)," *Circuits, Systems and Signal Processing*, vol. 19, no. 4, pp. 321–338, 2000. [Online]. Available: https://doi.org/10.1007/BF01200891

[44] S. E. El-Khamy, M. M. Hadhoud, M. I. Dessouky, B. M. Salam, and F. E. A. El-Samie, "Regularized super-resolution reconstruction of images using wavelet fusion," *Optical Engineering*, vol. 44, no. 9, pp. 1–10, 2005. [Online]. Available: https://doi.org/10.1117/1.2042947

[45] S. E. EL-KHAMY, M. M. HADHOUD, M. I. DESSOUKY, B. M. SALAM, and F. E. A. B. D. EL-SAMIE, "WAVELET FUSION: A TOOL TO BREAK THE LIMITS ON LMMSE IMAGE SUPER-RESOLUTION," *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 04, no. 01, pp. 105–118, 2006. [Online]. Available: https://doi.org/10.1142/S0219691306001129

[46] H. Ji and C. Fermüller, "Wavelet-Based Super-Resolution Reconstruction: Theory and Algorithm," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33839-0 pp. 295–307.

[47] H. Ji and C. Fermüller, "Robust Wavelet-Based Super-Resolution Reconstruction: Theory and Algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 649–660, 4 2009.

[48] M. B. Chappalli and N. K. Bose, "Simultaneous noise filtering and super-resolution with second-generation wavelets," *IEEE Signal Processing Letters*, vol. 12, no. 11, pp. 772–775, 11 2005.

[49] J. Chun, W. Dongming, L. Hongbing, Z. Zhu, and Z. L. Jerome, "Multiscale noise reduction on low-dose CT sinogram by stationary wavelet transform," in *IEEE Nuclear Science Symposium Conference Record*, 2008. doi: 10.1109/NSSMIC.2008.4774439. ISBN 9781424427154. ISSN 10957863 pp. 5339–5344.

[50] F. Ye, Z. Yabin, G. Dongwei, and Z. Zhan, "De-noising based on wavelet analysis and Bayesian estimation for low-dose X-ray CT," in *ICEMI 2009 - Proceedings of 9th International Conference on Electronic Measurement and Instruments*, 2009. doi: 10.1109/ICEMI.2009.5274424. ISBN 9781424438624 pp. 2944–2947.

[51] A. Mehranian, M. R. Ay, A. Rahmim, and H. Zaidi, "Sparsity constrained sinogram inpainting for metal artifact reduction in x-ray computed tomography," in *IEEE Nuclear Science Symposium Conference Record*. Institute of Electrical and Electronics Engineers Inc., 2011. doi: 10.1109/NSSMIC.2011.6153697. ISBN 9781467301183. ISSN 10957863 pp. 3694–3699.

[52] A. Mehranian, M. R. Ay, A. Rahmim, and H. Zaidi, "X-ray CT metal artifact reduction using wavelet domain L0 sparse regularization," *IEEE Transactions on Medical Imaging*, vol. 32, no. 9, pp. 1707–1722, 2013.

[53] A. Chaudhari, P. Chaudhary, A. N. Cheeran, Y. Aswani, and Radiologist, "Improving Signal to Noise Ratio of Low-Dose CT Image Using Wavelet Transform," *International Journal on Computer Science and Engineering*, vol. 4, 2012.

[54] A. Borsdorf, R. Raupach, T. Flohr, and J. Hornegger, "Wavelet based noise reduction in CT-images using correlation analysis," *IEEE Transactions on Medical Imaging*, vol. 27, no. 12, pp. 1685–1703, 12 2008.

[55] S. Ghadrdan, J. Alirezaie, J. L. Dillenseger, and P. Babyn, "Low-dose computed tomography image denoising based on joint wavelet and sparse representation," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2014*. Institute of Electrical and Electronics Engineers Inc., 11 2014. doi: 10.1109/EMBC.2014.6944334. ISBN 9781424479290 pp. 3325–3328.

[56] H. Ur and D. Gross, "Improved resolution from subpixel shifted pictures," *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 2, pp. 181–186, 3 1992.

[57] A. Papoulis, "Generalized sampling expansion," *IEEE Transactions on Circuits and Systems*, vol. 24, no. 11, pp. 652–654, 11 1977.

[58] J. Brown, "Multi-channel sampling of low-pass signals," *IEEE Transactions on Circuits and Systems*, vol. 28, no. 2, pp. 101–106, 2 1981.

[59] N. K. Bose and N. A. Ahuja, "Superresolution and noise filtering using moving least squares," *IEEE Transactions on Image Processing*, vol. 15, no. 8, pp. 2239–2248, 8 2006.

[60] M. Fenn and G. Steidl, "Robust local approximation of scatterred data," in *Geometric Properties for Incomplete data*, R. Klette, R. Kozera, L. Noakes, and J. Weickert, Eds. Dordrecht: Springer Netherlands, 2006, pp. 317–334. ISBN 978-1-4020-3858-7. [Online]. Available: https://doi.org/10.1007/1-4020-3858-8_17

[61] T. Saito, T. Komatsu, and K. Aizawa, "A signal-processing based method for acquiring very high resolution images with multiple cameras and its theoretical analysis," in *1992 International Conference on Image Processing and its Applications*, 4 1992, pp. 365–368.

[62] L. Landweber, "An Iteration Formula for Fredholm Integral Equations of the First Kind," *American Journal of Mathematics*, vol. 73, no. 3, pp. 615–624, 1951. [Online]. Available: http://www.jstor.org/stable/2372313

[63] T. Komatsu, T. Igarashi, K. Aizawa, and T. Saito, "Very high resolution imaging scheme with multiple different-aperture cameras," *Signal Processing: Image Communication*, vol. 5, no. 5-6, pp. 511–526, 12 1993.

[64] N. R. Shah and A. Zakhor, "Resolution enhancement of color video sequences," *IEEE Transactions on Image Processing*, vol. 8, no. 6, pp. 879–885, 6 1999.

[65] M. S. Alam, J. G. Bognar, R. C. Hardie, and B. J. Yasuda, "Infrared image registration and high-resolution reconstruction using multiple translationally shifted aliased video frames," *IEEE Transactions on Instrumentation and Measurement*, vol. 49, no. 5, pp. 915–923, 10 2000.

[66] H. Kostler, M. Prummer, U. Rude, and J. Hornegger, "Adaptive variational sinogram interpolation of sparsely sampled CT data," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, 2006. doi: 10.1109/ICPR.2006.225 pp. 778–781. [Online]. Available: https://ieeexplore.ieee.org/ielx5/11159/35819/01699641.pdf?tp=&arnumber=1699641&isnumber=35819https://ieeexplore.ieee.org/document/1699641/

[67] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 185–203, 8 1981.

[68] E. P. A. Constantino and K. B. Ozanyan, "Sinogram recovery for sparse angle tomography using a sinusoidal Hough transform," *Measurement Science and Technology*, vol. 19, no. 9, p. 94015, 6 2008. [Online]. Available: http://stacks.iop.org/0957-0233/19/i=9/a=094015http://iopscience.iop.org/article/10.1088/0957-0233/19/9/094015/pdf

[69] P. V. C. Hough, "Method and means for recognizing complex patterns," pp. 1–2, 1960. [Online]. Available: https://patents.google.com/patent/US3069654

[70] H. Zhang and J.-J. Sonke, "Directional sinogram interpolation for sparse angular acquisition in cone-beam computed tomography," *Journal of X-Ray Science and Technology*, vol. 21, no. 4, pp. 481–496, 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/24191986

[71] Y. Li, Y. Chen, Y. Hu, A. Oukili, L. Luo, W. Chen, and C. Toumoulin, "Strategy of computed tomography sinogram inpainting based on sinusoid-like curve decomposition and eigenvector-guided interpolation," *Journal of the Optical Society of America. A, Optics, Image Science, and Vision*, vol. 29, no. 1, pp. 153–163, 2012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22218362

[72] M.-C. Hong, M. G. Kang, and A. K. Katsaggelos, "Regularized multichannel restoration approach for globally optimal high-resolution video sequence," in *Visual Communications and Image Processing '97*, J. Biemond and E. J. D. III, Eds., vol. 3024, International Society for Optics and Photonics. SPIE, 1997. doi: 10.1117/12.263211 pp. 1306–1316. [Online]. Available: https://doi.org/10.1117/12.263211

[73] M.-C. Hong, M. G. Kang, and A. K. Katsaggelos, "An iterative weighted regularized algorithm for improving the resolution of video sequences," in *Proceedings of International Conference on Image Processing*, vol. 2, 10 1997. doi: 10.1109/ICIP.1997.638811 pp. 474–477.

[74] M. G. Kang, "Generalized multichannel image deconvolution approach and its applications," *Optical Engineering*, vol. 37, no. 11, pp. 2953–2964, 11 1998.

[75] R. C. Hardie, K. J. Barnard, J. G. Bognar, E. E. Armstrong, and E. A. Watson, "High-resolution image reconstruction from a sequence of rotated and translated frames and its application to an infrared imaging system," *Optical Engineering*, vol. 37, no. 1, pp. 247–260, 1998. [Online]. Available: https://doi.org/10.1117/1.601623

[76] N. K. Bose, S. Lertrattanapanich, and J. Koo, "Advances in superresolution using L-curve," in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196)*, vol. 2, 5 2001. doi: 10.1109/ISCAS.2001.921100 pp. 433–436.

[77] P. C. Hansen and D. P. O'Leary, "The Use of the L-Curve in the Regularization of Discrete Ill-Posed Problems," *SIAM Journal on Scientific Computing*, vol. 14, no. 6, pp. 1487–1503, 1993. [Online]. Available: https://doi.org/10.1137/0914086

[78] B. C. Tom and A. K. Katsaggelos, "Reconstruction of a high-resolution image by simultaneous registration, restoration, and interpolation of low-resolution images," in *Proceedings., International Conference on Image Processing*, vol. 2, 10 1995. doi: 10.1109/ICIP.1995.537535 pp. 539–542.

[79] R. C. Hardie, K. J. Barnard, and E. E. Armstrong, "Joint MAP registration and high-resolution image estimation using a sequence of undersampled images," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1621–1633, 12 1997.

[80] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721–741, 11 1984.

[81] R. R. Schultz and R. L. Stevenson, "A Bayesian approach to image expansion for improved definition," *IEEE Transactions on Image Processing*, vol. 3, no. 3, pp. 233–242, 5 1994.

[82] R. R. Schultz and R. L. Stevenson, "Extraction of high-resolution frames from video sequences," *IEEE Transactions on Image Processing*, vol. 5, no. 6, pp. 996–1011, 6 1996.

[83] K. V. Suresh and A. N. Rajagopalan, "Robust and computationally efficient superresolution algorithm," *J. Opt. Soc. Am. A*, vol. 24, no. 4, pp. 984–992, 4 2007. [Online]. Available: http://josaa.osa.org/abstract.cfm?URI=josaa-24-4-984

[84] S. P. Belekos, N. P. Galatsanos, and A. K. Katsaggelos, "Maximum a posteriori video super-resolution with a new multichannel image prior," in *2008 16th European Signal Processing Conference*, 8 2008. ISSN 2219-5491 pp. 1–5.

[85] M. Tipping and C. Bishop, "Bayesian Image Super-Resolution," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer, Eds., vol. 15. MIT Press, 2003. [Online]. Available: https://proceedings.neurips.cc/paper/2002/file/88bfcf02e7f554f9e9ea350b699bc6a7-Paper.pdf

[86] L. Pickup, D. Capel, S. J. Roberts, and A. Zisserman, "Bayesian Image Super-resolution, Continued," in *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman, Eds., vol. 19. MIT Press, 2007. [Online]. Available: https://proceedings.neurips.cc/paper/2006/file/10ce03a1ed01077e3e289f3e53c72813-Paper.pdf

[87] H. He and L. P. Kondi, "Resolution enhancement of video sequences with adaptively weighted low-resolution images and simultaneous estimation of the regularization parameter," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, 5 2004. doi: 10.1109/ICASSP.2004.1326519. ISSN 1520-6149 pp. iii–213.

[88] N. A. Woods, N. P. Galatsanos, and A. K. Katsaggelos, "Stochastic methods for joint registration, restoration, and interpolation of multiple undersampled images," *IEEE Transactions on Image Processing*, vol. 15, no. 1, pp. 201–213, 1 2006.

[89] J. Tian and K.-K. Ma, "Stochastic super-resolution image reconstruction," *Journal of Visual Communication and Image Representation*, vol. 21, no. 3, pp. 232–244, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047320310000027

[90] J. Tian and K.-K. Ma, "Edge-adaptive super-resolution image reconstruction using a Markov chain Monte Carlo approach," in *2007 6th International Conference on Information, Communications Signal Processing*, 12 2007. doi: 10.1109/ICICS.2007.4449569 pp. 1–5.

[91] H. Stark and P. Oskoui, "High-resolution image recovery from image-plane arrays, using convex projections," *J. Opt. Soc. Am. A*, vol. 6, no. 11, pp. 1715–1726, 11 1989. [Online]. Available: http://josaa.osa.org/abstract.cfm?URI=josaa-6-11-1715

[92] M. R. Banham and A. K. Katsaggelos, "Digital image restoration," *IEEE Signal Processing Magazine*, vol. 14, no. 2, pp. 24–41, 3 1997.

[93] A. M. Tekalp, M. K. Ozkan, and M. I. Sezan, "High-resolution image reconstruction from lower-resolution image sequences and space-varying image restoration,"

in *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, 3 1992. doi: 10.1109/ICASSP.1992.226249. ISSN 1520-6149 pp. 169–172.

[94] A. J. Patti, M. I. Sezan, and A. Murat Tekalp, "Superresolution video reconstruction with arbitrary sampling lattices and nonzero aperture time," *IEEE Transactions on Image Processing*, vol. 6, no. 8, pp. 1064–1076, 8 1997.

[95] A. J. Patti and Y. Altunbasak, "Artifact reduction for set theoretic super resolution image reconstruction with edge adaptive constraints and higher-order interpolants." *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 10, no. 1, pp. 179–186, 2001.

[96] M. Irani and S. Peleg, "Improving resolution by image registration," *CVGIP: Graphical Models and Image Processing*, vol. 53, no. 3, pp. 231–239, 5 1991.

[97] D. Karimi, P. Deman, R. Ward, and N. Ford, "A sinogram denoising algorithm for low-dose computed tomography," *BMC medical imaging*, vol. 16, p. 11, 1 2016.

[98] J. Wang, T. Li, H. Lu, and Z. Liang, "Penalized weighted least-squares approach to sinogram noise reduction and image reconstruction for low-dose X-ray computed tomography," *IEEE Transactions on Medical Imaging*, vol. 25, no. 10, pp. 1272–1283, 10 2006.

[99] J. Wang, H. Lu, J. Wen, and Z. Liang, "Multiscale penalized weighted least-squares sinogram restoration for low-dose X-ray computed tomography," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 3, pp. 1022–1031, 3 2008.

[100] B. De Man, J. Nuyts, P. Dupont, G. Marchai, and P. Suetens, "Reduction of metal streak artifacts in x-ray computed tomography using a transmission maximum a posteriori algorithm," *IEEE Transactions on Nuclear Science*, vol. 47, no. 3 PART 3, pp. 977–981, 2000.

[101] M. Tabuchi, N. Yamane, and Y. Morikawa, "Adaptive Wiener filter based on Gaussian mixture model for denoising chest X-ray CT image," in *Proceedings of the SICE Annual Conference*, 2007. doi: 10.1109/SICE.2007.4421069. ISBN 4907764286 pp. 682–689.

[102] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proceedings of the IEEE International Conference on Computer Vision*, 2009. doi: 10.1109/ICCV.2009.5459271. ISBN 9781424444205 pp. 349–356.

[103] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar, "Fast and robust multiframe super resolution," *IEEE Transactions on Image Processing*, vol. 13, no. 10, pp. 1327–1344, 2004.

[104] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[105] P. L. Dobruschin, "The Description of a Random Field by Means of Conditional Probabilities and Conditions of Its Regularity," *Theory of Probability & Its Applications*, vol. 13, no. 2, pp. 197–224, 1968. [Online]. Available: https://doi.org/10.1137/1113026

[106] F. Spitzer, "Markov Random Fields and Gibbs Ensembles," *The American Mathematical Monthly*, vol. 78, no. 2, pp. 142–154, 1971. [Online]. Available: http://www.jstor.org/stable/2317621

[107] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 12 2000. [Online]. Available: http://science.sciencemag.org/content/290/5500/2323

[108] K. P. F.R.S., "LIII. On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. [Online]. Available: https://doi.org/10.1080/14786440109462720

[109] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[110] J. Wang, S. Zhu, and Y. Gong, "Resolution enhancement based on learning the sparse association of image patches," *Pattern Recognition Letters*, vol. 31, no. 1, pp. 1–10, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865509002311

[111] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, 1993. doi: 10.1109/acssc.1993.342465. ISBN 0818641207. ISSN 10586393 pp. 40–44.

[112] R. Timofte and L. Van Gool, "Adaptive and Weighted Collaborative Representations for image classification," *Pattern Recognition Letters*, vol. 43, pp. 127–135, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865513003140

[113] H. Zhang, L. Zhang, Y. Sun, and J. Zhang, "Projection domain denoising method based on dictionary learning for low-dose CT image reconstruction," *Journal of X-Ray Science and Technology*, vol. 23, pp. 567–578, 2015.

[114] D. Karimi and R. Ward, "Reducing streak artifacts in computed tomography via sparse representation in coupled dictionaries," *Medical Physics*, vol. 43, no. 3, pp. 1473–1486, 6 2016. [Online]. Available: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.4942376https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1118/1.4942376

[115] D. Bartuschat, A. Borsdorf, H. Köstler, R. Rubinstein, and M. Stürmer, "A parallel K-SVD implementation for CT image denoising," in *Lehrstuhl für Informatik 10 (Systemsimulation)*, F. Erlangen, Ed., no. 09-1, 2009, p. 26. [Online]. Available: https://www10.cs.fau.de/publications/reports/TechRep_2009-01.pdf

[116] F. Yu, Y. Chen, and L. Luo, "CT image denoising based on sparse representation using global dictionary," in *2013 ICME International Conference on Complex Medical Engineering, CME 2013*, 2013. doi: 10.1109/ICCME.2013.6548279. ISBN 9781467329699 pp. 408–411.

[117] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a Deep Convolutional Network for Image Super-Resolution," in *European Conference on Computer Vision (ECCV) 2014*, ser. Lecture Notes in Computer Science. Springer, Cham, 9 2014. doi: 10.1007/978-3-319-10593-2_13. ISBN 978-3-319-10592-5 978-3-319-10593-2 pp. 184–199. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-10593-2_13

[118] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 6, pp. 1127–1133, 2010.

[119] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European Conference on Computer Vision (ECCV) 2016*, 2016. doi: 10.1007/978-3-319-46475-6_25. ISBN 9783319464749. ISSN 16113349 pp. 391–407.

[120] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 6 2010. doi: 10.1109/CVPR.2010.5539957. ISSN 1063-6919 pp. 2528–2535.

[121] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011. doi: 10.1109/ICCV.2011.6126474. ISBN 9781457711015 pp. 2018–2025.

[122] N. Efrat, D. Glasner, A. Apartsin, B. Nadler, and A. Levin, "Accurate blur models vs. image priors in single image super-resolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013. doi: 10.1109/ICCV.2013.352. ISBN 9781479928392

[123] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. doi: 10.1109/CVPR.2016.207. ISBN 9781467388504. ISSN 10636919

[124] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. doi: 10.1109/CVPR.2016.182. ISBN 9781467388504. ISSN 10636919

[125] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 9 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[126] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. doi: 10.1109/CVPR.2016.181. ISBN 9781467388504. ISSN 10636919

[127] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-Supervised Nets," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Lebanon and S. V. N. Vishwanathan, Eds., vol. 38. San Diego, California, USA: PMLR, 2015, pp. 562–570. [Online]. Available: http://proceedings.mlr.press/v38/lee15a.html

[128] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision (ECCV) 2016*, B. Leibe, , J. Matas, , N. Sebe, , and M. Welling, Eds., 2016. doi: 10.1007/978-3-319-46493-0_38. ISBN 9783319464923. ISSN 16113349 pp. 630–645.

[129] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 6 2016. doi: 10.1109/CVPR.2016.90. ISBN 978-1-4673-8851-1 pp. 770–778. [Online]. Available: http://ieeexplore.ieee.org/document/7780459/

[130] A. Veit, M. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Advances in Neural Information Processing Systems*, 2016. ISSN 10495258 p. 550–558. [Online]. Available: http://papers.nips.cc/paper/6556-residual-networks-behave-like-ensembles-of-relatively-shallow-networks.pdf

[131] D. Balduzzi, M. Frean, L. Leary, J. P. Lewis, K. W. D. Ma, and B. McWilliams, "The shattered gradients problem: If resnets are the answer, then what is the question?" in *34th International Conference on Machine Learning, ICML 2017*, D. Precup and Y. W. Teh, Eds., 2017. ISBN 9781510855144 pp. 342–350.

[132] F. He, T. Liu, and D. Tao, "Why ResNet Works? Residuals Generalize," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2020.

[133] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. doi: 10.1109/CVPR.2017.19. ISBN 9781538604571 pp. 105–114.

[134] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds. PMLR, 7 2015. doi: 10.1007/s13398-014-0173-7.2. ISBN 9780874216561. ISSN 0717-6163 pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

[135] I. J. Goodfellow, J. Pouget-Abadie, B. X. Mehdi Mirza, D. Warde-Farley, S. Ozairy, A. Courville, and Y. Bengioz, "Generative Adversarial Nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2014. doi: 10.5555/2969033.2969125. ISBN 9781139058452. ISSN 01652427 p. 2672–2680.

[136] R. C. Streijl, S. Winkler, and D. S. Hands, "Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives," *Multimedia Systems*, vol. 22, no. 2, pp. 213–227, 2016. [Online]. Available: https://doi.org/10.1007/s00530-014-0446-1

[137] W. Yang, J. Feng, J. Yang, F. Zhao, J. Liu, Z. Guo, and S. Yan, "Deep Edge Guided Recurrent Residual Learning for Image Super-Resolution," *IEEE Transactions on Image Processing*, vol. 26, no. 12, pp. 5895–5907, 12 2017.

[138] A. Singh and N. Ahuja, "Sub-band Energy Constraints for Self-Similarity Based Super-resolution," in *2014 22nd International Conference on Pattern Recognition*, 8 2014. doi: 10.1109/ICPR.2014.761. ISSN 1051-4651 pp. 4447–4452.

[139] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.298. ISBN 9781538604571 pp. 2790–2798.

[140] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017. doi: 10.1109/CVPRW.2017.151. ISBN 9781538607336. ISSN 21607516 pp. 1132–1140.

[141] W. Lai, J. Huang, N. Ahuja, and M. Yang, "Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7 2017. doi: 10.1109/CVPR.2017.618. ISSN 1063-6919 pp. 5835–5843.

[142] P. J. Burt and E. H. Adelson, "The Laplacian Pyramid as a Compact Image Code," in *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987, p. 671–679. ISBN 0934613338

[143] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.243. ISBN 9781538604571. ISSN 0022-4790 pp. 2261–2269.

[144] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017. ISSN 10495258 pp. 4467–4475. [Online]. Available: http://papers.nips.cc/paper/7033-dual-path-networks.pdf

[145] T. Tong, G. Li, X. Liu, and Q. Gao, "Image Super-Resolution Using Dense Skip Connections," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017. doi: 10.1109/ICCV.2017.514. ISBN 9781538610329. ISSN 15505499 pp. 4809–4817.

[146] Y. Tai, J. Yang, X. Liu, and C. Xu, "MemNet: A Persistent Memory Network for Image Restoration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017. doi: 10.1109/ICCV.2017.486. ISBN 9781538610329. ISSN 15505499 pp. 4549–4557.

[147] M. Haris, G. Shakhnarovich, and N. Ukita, "Deep Back-Projection Networks for Super-Resolution," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6 2018. doi: 10.1109/CVPR.2018.00179. ISSN 2575-7075 pp. 1664–1673.

[148] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual Dense Network for Image Super-Resolution," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. doi: 10.1109/CVPR.2018.00262. ISBN 9781538664209. ISSN 10636919 pp. 2472–2481.

[149] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., 2018. doi: 10.1007/978-3-030-01234-2_18. ISBN 9783030012335. ISSN 16113349 pp. 294–310.

[150] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. doi: 10.1109/CVPR.2018.00745. ISBN 9781538664209. ISSN 10636919 pp. 7132–7141.

[151] D. W. Jang and R. H. Park, "Densenet with deep residual channel-attention blocks for single image super resolution," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2019. doi: 10.1109/CVPRW.2019.00230. ISBN 9781728125060. ISSN 21607516 pp. 1795–1803.

[152] X. Dong, S. Vekhande, and G. Cao, "Sinogram interpolation for sparse-view micro-CT with deep learning neural network," in *Medical Imaging 2019: Physics of Medical Imaging*, T. G. Schmidt, G.-H. Chen, and H. Bosmans, Eds., vol. 10948, International Society for Optics and Photonics. SPIE, 2019. doi: 10.1117/12.2512979 pp. 692–698. [Online]. Available: https://doi.org/10.1117/12.2512979

[153] H. Lee, J. Lee, H. Kim, B. Cho, and S. Cho, "Deep-neural-network based sinogram synthesis for sparse-view CT image reconstruction," *IEEE Transactions on Radiation and Plasma Medical Sciences*, vol. 3, no. 2, pp. 109–119, 3 2018. [Online]. Available: http://arxiv.org/abs/1803.00694

[154] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning Deep CNN Denoiser Prior for Image Restoration," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7 2017. doi: 10.1109/CVPR.2017.300. ISSN 1063-6919 pp. 2808–2817.

[155] T. Meinhardt, M. Moeller, C. Hazirbas, and D. Cremers, "Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 10 2017. doi: 10.1109/ICCV.2017.198. ISSN 2380-7504 pp. 1799–1808.

[156] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015. ISBN 978-3-319-24574-4 pp. 234–241.

[157] E. Y. Sidky and X. Pan, "Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization." *Physics in medicine*

and biology, vol. 53, no. 17, pp. 4777–4807, 9 2008. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2630711/pdf/nihms-85944.pdf

[158] Y. Chen, X. Yin, L. Shi, H. Shu, L. Luo, J.-L. Coatrieux, and C. Toumoulin, "Improving abdomen tumor low-dose CT images using a fast dictionary learning based processing." *Physics in medicine and biology*, vol. 58, no. 16, pp. 5803–5820, 8 2013.

[159] K. Sheng, S. Gou, J. Wu, and S. X. Qi, "Denoised and texture enhanced MVCT to improve soft tissue conspicuity." *Medical physics*, vol. 41, no. 10, p. 101916, 10 2014.

[160] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in Neural Information Processing Systems*, 2012. ISBN 9781627480031. ISSN 10495258

[161] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *J. Mach. Learn. Res.*, vol. 11, p. 3371–3408, 12 2010.

[162] R. Wang and D. Tao, "Non-Local Auto-Encoder With Collaborative Stabilization for Image Restoration," *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 2117–2129, 5 2016.

[163] L. Gondara, "Medical Image Denoising Using Convolutional Denoising Autoencoders," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 12 2016. doi: 10.1109/ICDMW.2016.0041. ISSN 2375-9259 pp. 241–246.

[164] Q. Yang, P. Yan, Y. Zhang, H. Yu, Y. Shi, X. Mou, M. K. Kalra, Y. Zhang, L. Sun, and G. Wang, "Low-Dose CT Image Denoising Using a Generative Adversarial Network With Wasserstein Distance and Perceptual Loss." *IEEE transactions on medical imaging*, vol. 37, no. 6, pp. 1348–1357, 6 2018.

[165] J. Prewitt, "Object enhancement and extraction," in *Picture processing and Psychopictorics*, 1970, vol. 10, no. 1.

[166] I. Sobel, "An Isotropic 3x3 Image Gradient Operator," *Presentation at Stanford A.I. Project 1968*, 2014.

[167] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[168] L. Roberts, *Machine Perception of Three-Dimensional Solids*, 1963. ISBN 0-8240-4427-4

[169] R. A. Kirsch, "Computer determination of the constituent structure of biological images," *Computers and Biomedical Research*, vol. 4, no. 3, pp. 315–328, 6 1971.

[170] M. Al-amri, D. Kalyankar, and D. S.D, "Image segmentation by using edge detection," *International Journal on Computer Science and Engineering*, vol. 2, 2010.

[171] S. Saha, S. Basu, M. Nasipuri, and D. K. Basu, "A Hough Transform based Technique for Text Segmentation," *Journal of Computing*, vol. 2, no. 2, pp. 134–141, 2010. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1002/1002.4048.pdf

[172] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, 1988.

[173] J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, 1999, vol. 3.

[174] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic Active Contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, 1997. [Online]. Available: https://doi.org/10.1023/A:1007979827043

[175] M. Stella Atkins and B. T. Mackiewich, "Fully automatic segmentation of the brain in MRI," *IEEE Transactions on Medical Imaging*, vol. 17, no. 1, pp. 98–107, 1998.

[176] C. P. Loizou, C. S. Pattichis, M. Pantziaris, T. Tyllis, and A. Nicolaides, "Snakes based segmentation of the common carotid artery intima media," *Medical and Biological Engineering and Computing*, vol. 45, no. 1, pp. 35–49, 1 2007. [Online]. Available: https://link.springer.com/article/10.1007/s11517-006-0140-3

[177] C. Li, C. Xu, C. Gui, and M. D. Fox, "Level set evolution without re-initialization: A new variational formulation," in *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, vol. I. IEEE Computer Society, 2005. doi: 10.1109/CVPR.2005.213. ISBN 0769523722 pp. 430–436.

[178] M. Rousson and N. Paragios, "Shape priors for level set representations," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2351. Springer Verlag, 2002. doi: 10.1007/3-540-47967-8_6. ISBN 9783540437444. ISSN 16113349 pp. 78–92. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-47967-8_6

[179] C. Li, C. Xu, C. Gui, and M. D. Fox, "Distance regularized level set evolution and its application to image segmentation," *IEEE Transactions on Image Processing*, 2010.

[180] A. Tsai, A. Yezzi, W. Wells, C. Tempany, D. Tucker, A. Fan, W. E. Grimson, and A. Willsky, "A shape-based approach to the segmentation of medical imagery using level sets," *IEEE Transactions on Medical Imaging*, vol. 22, no. 2, pp. 137–154, 2 2003.

[181] C. Li, R. Huang, Z. Ding, J. C. Gatenby, D. N. Metaxas, and J. C. Gore, "A level set method for image segmentation in the presence of intensity inhomogeneities with application to MRI," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 2007–2016, 7 2011.

[182] N. Otsu, "A threshold selection method from gray level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, 1979.

[183] P. L. Rosin, "Unimodal thresholding," *Pattern Recognition*, vol. 34, no. 11, pp. 2083–2096, 11 2001.

[184] A. Anjos and H. Shahbazkia, "Bi-Level Image Thresholding - A Fast Method," in *BIOSIGNALS*, 2008.

[185] C. H. Li and C. K. Lee, "Minimum cross entropy thresholding," *Pattern Recognition*, vol. 26, no. 4, pp. 617–625, 4 1993.

[186] M. Portes de Albuquerque, I. A. Esquef, A. R. Gesualdi Mello, and M. Portes de Albuquerque, "Image thresholding using Tsallis entropy," *Pattern Recognition Letters*, vol. 25, no. 9, pp. 1059–1065, 7 2004.

[187] Y. Zhang, H. Qu, and Y. Wang, "Adaptive Image Segmentation based on Fast Thresholding and Image Merging," in *Proceedings - 16th International Conference on Artificial Reality and Telexistence - Workshops, ICAT 2006*, 2006. doi: 10.1109/ICAT.2006.32. ISBN 076952754X pp. 308–311.

[188] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2 2000.

[189] F. Yan, H. Zhang, and C. R. Kube, "A multistage adaptive thresholding method," *Pattern Recognition Letters*, vol. 26, no. 8, pp. 1183–1191, 6 2005.

[190] J. Shi, N. Ray, and H. Zhang, "Shape based local thresholding for binarization of document images," *Pattern Recognition Letters*, vol. 33, no. 1, pp. 24–32, 1 2012.

[191] T. R. Singh, S. Roy, O. Singh, T. Sinam, and K. M. Singh, "A New Local Adaptive Thresholding Technique in Binarization," *International Journal of Computer Science Issues*, vol. 8, no. 6, pp. 271–277, 2011.

[192] R. Adams and L. Bischof, "Seeded Region Growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994.

[193] Y. L. Chang and X. Li, "Adaptive Image Region-Growing," *IEEE Transactions on Image Processing*, vol. 3, no. 6, pp. 868–872, 1994.

[194] A. Tremeau and N. Borel, "A region growing and merging algorithm to color segmentation," *Pattern Recognition*, vol. 30, no. 7, pp. 1191–1203, 7 1997.

[195] F. Y. Shih and S. Cheng, "Automatic seeded region growing for color image segmentation," *Image and Vision Computing*, vol. 23, no. 10, pp. 877–886, 9 2005.

[196] X. Zhang, Y. Shan, W. Wei, and Z. Zhu, "An image segmentation method based on improved watershed algorithm," in *Proceedings - 2010 International Conference on Computational and Information Sciences, ICCIS 2010*, 2010. doi: 10.1109/ICCIS.2010.69. ISBN 9780769542706 pp. 258–261.

[197] P. Neubert and P. Protzel, "Compact watershed and preemptive SLIC: On improving trade-offs of superpixel segmentation algorithms," in *Proceedings - International Conference on Pattern Recognition*, 2014. doi: 10.1109/ICPR.2014.181. ISBN 9781479952083. ISSN 10514651

[198] Y. Y. Boykov and M. P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 1, 2001. doi: 10.1109/iccv.2001.937505 pp. 105–112.

[199] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient N-D image segmentation," *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109–131, 11 2006. [Online]. Available: https://link.springer.com/article/10.1007/s11263-006-7934-5

[200] C. Rother, V. Kolmogorov, and A. Blake, ""GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts," *ACM Trans. Graph.*, vol. 23, no. 3, p. 309–314, 8 2004. [Online]. Available: https://doi.org/10.1145/1015706.1015720

[201] D. Freedman and T. Zhang, "Interactive graph cut based segmentation with shape priors," in *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, vol. I. IEEE Computer Society, 2005. doi: 10.1109/CVPR.2005.191. ISBN 0769523722 pp. 755–762.

[202] J. Malcolm, Y. Rathi, and A. Tannenbaum, "Graph cut segmentation with nonlinear shape priors," in *Proceedings - International Conference on Image Processing, ICIP*, vol. 4. IEEE Computer Society, 2007. doi: 10.1109/ICIP.2007.4380030. ISBN 1424414377. ISSN 15224880 pp. 365–368.

[203] B. L. Price, B. Morse, and S. Cohen, "Geodesic graph cut for interactive image segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010. doi: 10.1109/CVPR.2010.5540079. ISBN 9781424469840. ISSN 10636919 pp. 3161–3168.

[204] M. Tang, L. Gorelick, O. Veksler, and Y. Boykov, "Grabcut in one cut," in *Proceedings of the IEEE International Conference on Computer Vision*. Institute of Electrical and Electronics Engineers Inc., 2013. doi: 10.1109/ICCV.2013.222. ISBN 9781479928392 pp. 1769–1776.

[205] I. Luengo, M. Basham, and A. P. French, "SMURFS: superpixels from multi-scale refinement of super-regions," in *Proceedings of the British Machine Vision Conference (BMVC) 2016*, York, UK, 9 2016, pp. 1–12. [Online]. Available: http://www.bmva.org/bmvc/2016/papers/paper004/index.html

[206] I. Luengo, M. Basham, and A. P. French, "Fast global interactive volume segmentation with regional supervoxel descriptors," in *Medical Imaging 2016: Image Processing*, 2016. doi: 10.1117/12.2216382. ISBN 9781510600195. ISSN 16057422 p. 97842D.

[207] T. N. Pappas and N. S. Jayant, "Adaptive clustering algorithm for image segmentation," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 3. Publ by IEEE, 1989. doi: 10.1109/icassp.1989.266767. ISSN 07367791 pp. 1667–1670.

[208] C. W. Chen, J. Luo, and K. J. Parker, "Image segmentation via adaptive K-mean clustering and knowledge-based morphological operations with biomedical applications," *IEEE Transactions on Image Processing*, vol. 7, no. 12, pp. 1673–1683, 1998.

[209] S. Ray and R. Turi, "Determination of number of clusters in k-means clustering and application in colour image segmentation," *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques*, 1999.

[210] M. Luo, Y. F. Ma, and H. J. Zhang, "A spatial constrained K-means approach to image segmentation," in *ICICS-PCM 2003 - Proceedings of the 2003 Joint Conference of the 4th International Conference on Information, Communications and Signal Processing and 4th Pacific-Rim Conference on Multimedia*, vol. 2. Institute of Electrical and Electronics Engineers Inc., 2003. doi: 10.1109/ICICS.2003.1292554. ISBN 0780381858 pp. 738–742.

[211] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 11 2012.

[212] N. Dhanachandra, K. Manglem, and Y. J. Chanu, "Image Segmentation Using K-means Clustering Algorithm and Subtractive Clustering Algorithm," in *Procedia Computer Science*, vol. 54. Elsevier, 1 2015. doi: 10.1016/j.procs.2015.06.090. ISSN 18770509 pp. 764–771.

[213] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[214] J. Wang, B. Thiesson, Y. Xu, and M. Cohen, "Image and video segmentation by anisotropic kernel mean shift," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3022, pp. 238–249, 2004. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-24671-8_19

[215] S. Paris and F. Durand, "A topological approach to hierarchical segmentation using mean shift," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007. doi: 10.1109/CVPR.2007.383228. ISBN 1424411807. ISSN 10636919

[216] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, 1973. [Online]. Available: https://doi.org/10.1080/01969727308546046

[217] J. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*, 1981. ISBN 978-1-4757-0452-5

[218] Y. A. Tolias and S. M. Panas, "Image segmentation by a fuzzy clustering algorithm using adaptive spatially constrained membership functions," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, vol. 28, no. 3, pp. 359–369, 1998.

[219] M. N. Ahmed, S. M. Yamany, N. Mohamed, A. A. Farag, and T. Moriarty, "A modified fuzzy C-means algorithm for bias field estimation and segmentation of MRI data," *IEEE Transactions on Medical Imaging*, vol. 21, no. 3, pp. 193–199, 3 2002.

[220] S. Chen and D. Zhang, "Robust image segmentation using FCM with spatial constraints based on new kernel-induced distance measure," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 4, pp. 1907–1916, 8 2004.

[221] S. Krinidis and V. Chatzis, "A robust fuzzy local information C-means clustering algorithm," *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1328–1337, 5 2010.

[222] T. K. Ho, "Random decision forests," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1995. doi: 10.1109/ICDAR.1995.598994. ISBN 0818671289. ISSN 15205363

[223] F. Schroff, A. Criminisi, and A. Zisserman, "Object Class Segmentation using Random Forests," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2008. doi: doi:10.5244/C.22.54. ISBN 1-901725-36-7 pp. 1–54.

[224] A. Montillo, J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas, and A. Criminisi, "Entangled Decision Forests and Their Application for Semantic Segmentation of CT Images," in *Information Processing in Medical Imaging*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 7 2011. doi: 10.1007/978-3-642-22092-0_16. ISBN 978-3-642-22091-3 978-3-642-22092-0 pp. 184–196. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-22092-0_16

[225] C. Sommer, C. Straehle, U. Kothe, and F. A. Hamprecht, "Ilastik: Interactive learning and segmentation toolkit," in *Proceedings - International Symposium on Biomedical Imaging*, 2011. doi: 10.1109/ISBI.2011.5872394. ISBN 9781424441280. ISSN 19457928 pp. 230–233.

[226] Y. Zuo and T. Drummond, "Fast Residual Forests: Rapid Ensemble Learning for Semantic Segmentation," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 2017, pp. 27–36. [Online]. Available: http://proceedings.mlr.press/v78/zuo17a.html

[227] B. Kang and T. Q. Nguyen, "Random Forest With Learned Representations for Semantic Segmentation," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3542–3555, 7 2019.

[228] V. VAPNIK and A. LERNER, "Pattern recognition using generalized portrait method," *Automation and Remote Control*, vol. 24, pp. 771–780, 1963.

[229] V. VAPNIK and A. CHERVONENKIS, "A note on one class of perceptrons," *Automation and Remote Control*, vol. 25, 1964.

[230] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "Training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 1992. doi: 10.1145/130385.130401. ISBN 089791497X pp. 447–152.

[231] C. Kotropoulos and I. Pitas, "Segmentation of ultrasonic images using Support Vector Machines," *Pattern Recognition Letters*, vol. 24, no. 4-5, pp. 715–727, 2 2003.

[232] X. Y. Wang, T. Wang, and J. Bu, "Color image segmentation using pixel wise support vector machine classification," *Pattern Recognition*, vol. 44, no. 4, pp. 777–787, 4 2011.

[233] S. Bauer, L. P. Nolte, and M. Reyes, "Fully Automatic Segmentation of Brain Tumor Images Using Support Vector Machine Classification in Combination with Hierarchical Conditional Random Field Regularization," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6893 LNCS, no. PART 3. Springer, Berlin, Heidelberg, 2011. doi: 10.1007/978-3-642-23626-6_44. ISBN 9783642236259. ISSN 03029743 pp. 354–361. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-23626-6_44

[234] T. Wu, M. H. Bae, M. Zhang, R. Pan, and A. Badea, "A prior feature svm-mrf based method for mouse brain segmentation," *NeuroImage*, vol. 59, no. 3, pp. 2298–2306, 2 2012.

[235] X. Bai and W. Wang, "Saliency-SVM: An automatic approach for image segmentation," *Neurocomputing*, vol. 136, pp. 243–255, 7 2014.

[236] A. Qayyum and A. Basit, "Automatic breast segmentation and cancer detection via SVM in mammograms," in *2016 International Conference on Emerging Technologies (ICET)*, 10 2016. doi: 10.1109/ICET.2016.7813261 pp. 1–6.

[237] T. Akram, M. A. Khan, M. Sharif, and M. Yasmin, "Skin lesion segmentation and recognition using multichannel saliency estimation and M-SVM on selected serially fused features," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–20, 2018. [Online]. Available: https://doi.org/10.1007/s12652-018-1051-5

[238] V. P. Kour and S. Arora, "Particle Swarm Optimization Based Support Vector Machine (P-SVM) for the Segmentation and Classification of Plants," *IEEE Access*, vol. 7, pp. 29 374–29 385, 2019.

[239] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4. IEEE. doi: 10.1109/ICNN.1995.488968. ISBN 0-7803-2768-3 pp. 1942–1948. [Online]. Available: http://ieeexplore.ieee.org/document/488968/

[240] J. M. Hammersley and P. E. Clifford, "Markov random fields on finite graphs and lattices," *Unpublished manuscript*, 1971.

[241] J. Verbeek and B. Triggs, "Region classification with Markov field aspect models," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007. doi: 10.1109/CVPR.2007.383098. ISBN 1424411807. ISSN 10636919

[242] T. Hofmann, "Unsupervised learning by probabilistic Latent Semantic Analysis," *Machine Learning*, vol. 42, no. 1-2, pp. 177–196, 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1007617005950

[243] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *J. Mach. Learn. Res.*, vol. 3, no. null, p. 993–1022, 3 2003.

[244] N. K. Subbanna, D. Precup, D. L. Collins, and T. Arbel, "Hierarchical Probabilistic Gabor and MRF Segmentation of Brain Tumours in MRI Volumes," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*, K. Mori,

I. Sakuma, Y. Sato, C. Barillot, and N. Navab, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40811-3 pp. 751–758.

[245] A. Ahmadvand, M. R. Daliri, and S. M. Zahiri, "Segmentation of brain MR images using a proper combination of DCS based method with MRF," *Multimedia Tools and Applications*, vol. 77, no. 7, pp. 8001–8018, 2018. [Online]. Available: https://doi.org/10.1007/s11042-017-4696-8

[246] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. ISBN 1558607781 p. 282–289.

[247] X. He, R. S. Zemel, and M. A. Carreira-Perpinan, "Multiscale conditional random fields for image labeling," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2004. doi: 10.1109/cvpr.2004.1315232. ISSN 10636919

[248] C. Wojek and B. Schiele, "A Dynamic Conditional Random Field Model for Joint Labeling of Object and Scene Classes," in *Computer Vision – ECCV 2008*, D. Forsyth, P. Torr, and A. Zisserman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. ISBN 978-3-540-88693-8 pp. 733–747.

[249] L. Ladický, C. Russell, P. Kohli, and P. H. Torr, "Associative hierarchical CRFs for object class image segmentation," in *2009 IEEE 12th International Conference on Computer Vision*, 2009. doi: 10.1109/ICCV.2009.5459248. ISBN 9781424444205 pp. 739–746.

[250] P. Krähenbühl and V. Koltun, "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 109–117. ISBN 9781618395993. [Online]. Available: http://papers.nips.cc/paper/4296-efficient-inference-in-fully-connected-crfs-with-gaussian-edge-potentials.pdf

[251] X. He and S. Gould, "An exemplar-based CRF for multi-instance object segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 9 2014. doi: 10.1109/CVPR.2014.45. ISBN 9781479951178. ISSN 10636919 pp. 296–303.

[252] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–951, 2000. [Online]. Available: https://doi.org/10.1038/35016072

[253] J. Weng, N. Ahuja, and T. S. Huang, "Cresceptron: a self-organizing neural network which grows adaptively," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 1, 6 1992. doi: 10.1109/IJCNN.1992.287150 pp. 576–581.

[254] M. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun, "Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 6 2007. doi: 10.1109/CVPR.2007.383157. ISSN 1063-6919 pp. 1–8.

[255] A. Krizhevsky and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012. doi: 10.1016/j.protcy.2014.09.007. ISBN 9781627480031. ISSN 10495258 pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[256] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014. ISBN 978-3-319-10590-1 pp. 818–833.

[257] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015. doi: 10.1109/CVPR.2015.7298594. ISBN 9781467369640. ISSN 10636919 pp. 1–9.

[258] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *CoRR*, vol. abs/1704.0, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[259] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6 2018. doi: 10.1109/CVPR.2018.00474. ISSN 2575-7075 pp. 4510–4520.

[260] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6 2018. doi: 10.1109/CVPR.2018.00716. ISSN 2575-7075 pp. 6848–6856.

[261] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018. ISBN 978-3-030-01264-9 pp. 122–138.

[262] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4

[263] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[264] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014. ISBN 978-3-319-10602-1 pp. 740–755.

[265] D. Ciresan, A. Giusti, L. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012. doi: 10.1016/S1673-8527(09)60051-5. ISBN 9781627480031. ISSN 10495258 pp. 2843–2851. [Online]. Available: http://papers.nips.cc/paper/4741-deep-neural-networks-segment-neuronal-membranes-in-electron-microscopy-images.pdf

[266] J. S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," in *Neurocomputing*, F. F. Soulié and J. Hérault, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990. ISBN 978-3-642-76153-9 pp. 227–236.

[267] J. S. Bridle, "Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed. Morgan-Kaufmann, 1990, pp. 211–217. [Online]. Available: http://papers.nips.cc/paper/195-training-stochastic-model-recognition-algorithms-as-networks-can-lead-to-maximum-mutual-information-estimation-of-parameters.pdf

[268] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 3431–3440, 2015.

[269] W. Liu, A. Rabinovich, and A. C. Berg, "ParseNet: Looking Wider to See Better," *4th International Conference on Learning Representations, ICLR 2016 - Workshop Contribution*, 2016. [Online]. Available: http://arxiv.org/abs/1506.04579

[270] M. Lin, Q. Chen, and S. Yan, "Network in network," in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014. [Online]. Available: https://arxiv.org/abs/1312.4400

[271] A. Oliveira, S. Pereira, and C. A. Silva, "Retinal vessel segmentation based on Fully Convolutional Neural Networks," *Expert Systems with Applications*, vol. 112, pp. 229–242, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418303816

[272] Y. Yuan, M. Chao, and Y. Lo, "Automatic Skin Lesion Segmentation Using Deep Fully Convolutional Networks With Jaccard Distance," *IEEE Transactions on Medical Imaging*, vol. 36, no. 9, pp. 1876–1886, 2017.

[273] L. Bi, J. Kim, E. Ahn, A. Kumar, M. Fulham, and D. Feng, "Dermoscopic Image Segmentation via Multistage Fully Convolutional Networks," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 9, pp. 2065–2074, 2017.

[274] G. Wang, W. Li, S. Ourselin, and T. Vercauteren, "Automatic Brain Tumor Segmentation Using Cascaded Anisotropic Convolutional Neural Networks," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi,

S. Bakas, H. Kuijf, B. Menze, and M. Reyes, Eds. Cham: Springer International Publishing, 2018. ISBN 978-3-319-75238-9 pp. 178–190.

[275] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015. [Online]. Available: https://arxiv.org/abs/1412.7062

[276] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional Random Fields as Recurrent Neural Networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 12 2015. doi: 10.1109/ICCV.2015.179. ISSN 2380-7504 pp. 1529–1537.

[277] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. [Online]. Available: https://doi.org/10.1038/323533a0

[278] Z. Liu, X. Li, P. Luo, C. Loy, and X. Tang, "Semantic Image Segmentation via Deep Parsing Network," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 12 2015. doi: 10.1109/ICCV.2015.162. ISSN 2380-7504 pp. 1377–1385.

[279] M. Opper and D. Saad, "From Naive Mean Field Theory to the TAP Equations," in *Advanced Mean Field Methods: Theory and Practice*. MIT Press, 2001, pp. 7–20. ISBN 9780262281058. [Online]. Available: https://ieeexplore.ieee.org/document/6281797

[280] G. Lin, C. Shen, A. van den Hengel, and I. Reid, "Efficient Piecewise Training of Deep Structured Models for Semantic Segmentation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2016. doi: 10.1109/CVPR.2016.348. ISSN 1063-6919 pp. 3194–3203.

[281] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning Hierarchical Features for Scene Labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 8 2013.

[282] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 6 2006. doi: 10.1109/CVPR.2006.68. ISSN 1063-6919 pp. 2169–2178.

[283] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 6 2014. doi: 10.1109/CVPR.2014.81. ISSN 1063-6919 pp. 580–587.

[284] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 12 2015. doi: 10.1109/ICCV.2015.169. ISSN 2380-7504 pp. 1440–1448.

[285] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 6 2017.

[286] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[287] J. Dai, K. He, and J. Sun, "Instance-Aware Semantic Segmentation via Multi-task Network Cascades," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2016. doi: 10.1109/CVPR.2016.343. ISSN 1063-6919 pp. 3150–3158.

[288] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-Based Fully Convolutional Networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016. ISBN 9781510838819 p. 379–387.

[289] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully Convolutional Instance-Aware Semantic Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7 2017. doi: 10.1109/CVPR.2017.472. ISSN 1063-6919 pp. 4438–4446.

[290] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 10 2017. doi: 10.1109/ICCV.2017.322. ISSN 2380-7504 pp. 2980–2988.

[291] P. O. Pinheiro, R. Collobert, and P. Dollár, "Learning to Segment Object Candidates," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 1990–1998.

[292] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to Refine Object Segments," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016. ISBN 978-3-319-46448-0 pp. 75–91.

[293] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6 2018. doi: 10.1109/CVPR.2018.00913. ISSN 2575-7075 pp. 8759–8768.

[294] R. Hu, P. Dollár, K. He, T. Darrell, and R. Girshick, "Learning to Segment Every Thing," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6 2018. doi: 10.1109/CVPR.2018.00445. ISSN 2575-7075 pp. 4233–4241.

[295] C.-C. Hsu, K.-J. Hsu, C.-C. Tsai, Y.-Y. Lin, and Y.-Y. Chuang, "Weakly Supervised Instance Segmentation using the Bounding Box Tightness Prior," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 6586–6597. [Online]. Available: http://papers.nips.cc/paper/8885-weakly-supervised-instance-segmentation-using-the-bounding-box-tightness-prior.pdf

[296] J. D. Keeler, D. E. Rumelhart, and W. K. Leow, "Integrated Segmentation and Recognition of Hand-Printed Numerals," in *Advances in Neural Information*

Processing Systems 3, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds. Morgan-Kaufmann, 1991, pp. 557–563. [Online]. Available: http://papers.nips.cc/paper/397-integrated-segmentation-and-recognition-of-hand-printed-numerals.pdf

[297] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1, pp. 31–71, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370296000343

[298] L. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam, "MaskLab: Instance Segmentation by Refining Object Detection with Semantic and Direction Features," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6 2018. doi: 10.1109/CVPR.2018.00422. ISSN 2575-7075 pp. 4013–4022.

[299] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "Hybrid Task Cascade for Instance Segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2019. doi: 10.1109/CVPR.2019.00511. ISSN 2575-7075 pp. 4969–4978.

[300] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-Time Instance Segmentation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 10 2019. doi: 10.1109/ICCV.2019.00925. ISSN 2380-7504 pp. 9156–9165.

[301] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 10 2017. doi: 10.1109/ICCV.2017.324. ISSN 2380-7504 pp. 2999–3007.

[302] B. Cai, J. Wang, A. Kao, K. Pericleous, A. B. Phillion, R. C. Atwood, and P. D. Lee, "4D synchrotron X-ray tomographic quantification of the transition from cellular to dendrite growth during directional solidification," *Acta Materialia*, vol. 117, pp. 160–169, 9 2016.

[303] E. Guo, A. B. Phillion, B. Cai, S. Shuai, D. Kazantsev, T. Jing, and P. D. Lee, "Dendritic evolution during coarsening of Mg-Zn alloys via 4D synchrotron tomography," *Acta Materialia*, vol. 123, pp. 373–382, 1 2017.

[304] B. Cai, A. Kao, P. D. Lee, E. Boller, H. Basevi, A. B. Phillion, A. Leonardis, and K. Pericieous, "Growth of β intermetallic in an Al-Cu-Si alloy during directional solidification via machine learned 4D quantification," *Scripta Materialia*, vol. 165, pp. 29–33, 5 2019.

[305] T. Nelson, B. Cai, N. Warnken, P. D. Lee, E. Boller, O. V. Magdysyuk, and N. R. Green, "Gravity effect on thermal-solutal convection during solidification revealed by four-dimensional synchrotron imaging with compositional mapping," *Scripta Materialia*, vol. 180, pp. 29–33, 4 2020.

[306] I. Luengo, M. C. Darrow, M. C. Spink, Y. Sun, W. Dai, C. Y. He, W. Chiu, T. Pridmore, A. W. Ashton, E. M. Duke, M. Basham, and A. P. French, "SuRVoS: Super-Region Volume Segmentation workbench," *Journal of Structural Biology*, vol. 198, no. 1, pp. 43–53, 2017.

[307] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 12 2015. doi: 10.1109/ICCV.2015.178. ISSN 2380-7504 pp. 1520–1528.

[308] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[309] V. B. Alex Kendall and R. Cipolla, "Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding," in *Proceedings of the British Machine Vision Conference (BMVC)*, G. B. Tae-Kyun Kim Stefanos Zafeiriou and K. Mikolajczyk, Eds. BMVA Press, 9 2017. doi: 10.5244/C.31.57. ISBN 1-901725-60-X pp. 1–57. [Online]. Available: https://dx.doi.org/10.5244/C.31.57

[310] A. Chaurasia and E. Culurciello, "LinkNet: Exploiting encoder representations for efficient semantic segmentation," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, 12 2017. doi: 10.1109/VCIP.2017.8305148 pp. 1–4.

[311] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7 2017. doi: 10.1109/CVPR.2017.353. ISSN 1063-6919 pp. 3309–3318.

[312] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[313] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 1 2018.

[314] J. M. Alvarez and L. Petersson, "DecomposeMe: Simplifying ConvNets for End-to-End Learning," *CoRR*, vol. abs/1606.0, 2016. [Online]. Available: http://arxiv.org/abs/1606.05426

[315] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016. doi: 10.1007/978-3-319-46484-8_29. ISBN 9783319464831. ISSN 16113349

[316] A. Ghosh, M. Ehrlich, S. Shah, L. Davis, and R. Chellappa, "Stacked U-Nets for Ground Material Segmentation in Remote Sensing Imagery," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 6 2018. doi: 10.1109/CVPRW.2018.00047. ISSN 2160-7516 pp. 252–2524.

[317] S. Shah, P. Ghosh, L. S. Davis, and T. Goldstein, "Stacked U-Nets: A No-Frills Approach to Natural Image Segmentation," *CoRR*, vol. abs/1804.1, 2018. [Online]. Available: http://arxiv.org/abs/1804.10343

[318] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 7 2017. doi: 10.1109/CVPRW.2017.156. ISSN 2160-7516 pp. 1175–1183.

[319] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and Recognition Using Structure from Motion Point Clouds," in *ECCV*, vol. 1, 2008, pp. 44–57.

[320] S. H. Raza, M. Grundmann, and I. Essa, "Geometric Context from Videos," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 6 2013. doi: 10.1109/CVPR.2013.396. ISSN 1063-6919 pp. 3081–3088.

[321] J. Fu, J. Liu, Y. Wang, J. Zhou, C. Wang, and H. Lu, "Stacked Deconvolutional Network for Semantic Segmentation," *IEEE Transactions on Image Processing*, p. 1, 2019.

[322] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 11 2016. [Online]. Available: http://arxiv.org/abs/1511.07122

[323] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation," *CoRR*, vol. abs/1606.0, 2016. [Online]. Available: http://arxiv.org/abs/1606.02147

[324] J. M. Wolterink, T. Leiner, M. A. Viergever, and I. Išgum, "Dilated Convolutional Neural Networks for Cardiovascular MR Segmentation in Congenital Heart Disease," in *Reconstruction, Segmentation, and Analysis of Medical Images*, M. A. Zuluaga, K. Bhatia, B. Kainz, M. H. Moghari, and D. F. Pace, Eds. Cham: Springer International Publishing, 2017. ISBN 978-3-319-52280-7 pp. 95–102.

[325] E. H. Adelson, P. J. Burt, C. H. Anderson, J. M. Ogden, and J. R. Bergen, "PYRAMID METHODS IN IMAGE PROCESSING." *RCA engineer*, 1984.

[326] G. Ghiasi and C. C. Fowlkes, "Laplacian Pyramid Reconstruction and Refinement for Semantic Segmentation," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016. ISBN 978-3-319-46487-9 pp. 519–534.

[327] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7 2017. doi: 10.1109/CVPR.2017.106. ISSN 1063-6919 pp. 936–944.

[328] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 7 2017. doi: 10.1109/CVPR.2017.660. ISBN 978-1-5386-0457-1 pp. 6230–6239. [Online]. Available: http://ieeexplore.ieee.org/document/8100143/

[329] G. Lin, F. Liu, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-Path Refinement Networks for Dense Prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 5, pp. 1228–1242, 5 2020.

[330] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[331] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," *CoRR*, vol. abs/1706.0, 2017. [Online]. Available: http://arxiv.org/abs/1706.05587

[332] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018. doi: 10.1007/978-3-030-01234-2_49. ISBN 9783030012335. ISSN 16113349

[333] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7 2017. doi: 10.1109/CVPR.2017.195. ISSN 1063-6919 pp. 1800–1807.

[334] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018. ISBN 978-3-030-01249-6 pp. 561–580.

[335] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell, "Understanding Convolution for Semantic Segmentation," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 3 2018. doi: 10.1109/WACV.2018.00163 pp. 1451–1460.

[336] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, "DenseASPP for Semantic Segmentation in Street Scenes," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6 2018. doi: 10.1109/CVPR.2018.00388. ISSN 2575-7075 pp. 3684–3692.

[337] D. M. Pelt and J. A. Sethian, "A mixed-scale dense convolutional neural network for image analysis," *Proceedings of the National Academy of Sciences*, vol. 115, no. 2, pp. 254–259, 1 2018. [Online]. Available: http://www.pnas.org/lookup/doi/10.1073/pnas.1715832114

[338] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[339] J. Driver and G. C. Baylis, "Attention and visual object segmentation." in *The attentive brain.* Cambridge, MA, US: The MIT Press, 1998, pp. 299–325. ISBN 0-262-16172-9 (Hardcover)

[340] J. Driver, G. Davis, C. Russell, M. Turatto, and E. Freeman, "Segmentation, attention and phenomenal visual objects." *Cognition*, vol. 80, no. 1-2, pp. 61–95, 6 2001.

[341] L. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to Scale: Scale-Aware Semantic Image Segmentation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2016. doi: 10.1109/CVPR.2016.396. ISSN 1063-6919 pp. 3640–3649.

[342] M. Ren and R. S. Zemel, "End-to-End Instance Segmentation with Recurrent Attention," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7 2017. doi: 10.1109/CVPR.2017.39. ISSN 1063-6919 pp. 293–301.

[343] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Learning a Discriminative Feature Network for Semantic Segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6 2018. doi: 10.1109/CVPR.2018.00199. ISSN 2575-7075 pp. 1857–1866.

[344] Z. Tan, M. Wang, J. Xie, Y. Chen, and X. Shi, "Deep semantic role labeling with self-attention," in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018. ISBN 9781577358008

[345] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. C. Loy, D. Lin, and J. Jia, "PSANet: Point-wise Spatial Attention Network for Scene Parsing," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018. ISBN 978-3-030-01240-3 pp. 270–286.

[346] H. Li, P. Xiong, J. An, and L. Wang, "Pyramid attention network for semantic segmentation," in *British Machine Vision Conference 2018, BMVC 2018*, 2019. [Online]. Available: https://arxiv.org/pdf/1805.10180.pdf

[347] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual Attention Network for Scene Segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2019. doi: 10.1109/CVPR.2019.00326. ISSN 2575-7075 pp. 3141–3149.

[348] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "CCNet: Criss-Cross Attention for Semantic Segmentation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 10 2019. doi: 10.1109/ICCV.2019.00069. ISSN 2380-7504 pp. 603–612.

[349] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[350] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[351] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, "TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation," Tech. Rep. [Online]. Available: https://github.com/Beckschen/

[352] I. Sutskever, G. Hinton, A. Krizhevsky, and R. R. Salakhutdinov, "Dropout : A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, 2014.

[353] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic Segmentation using Adversarial Networks," *Advances in Neural Information Processing Systems - Workshop on Adversarial Training*, 11 2016. [Online]. Available: http://arxiv.org/abs/1611.08408

[354] N. Souly, C. Spampinato, and M. Shah, "Semi Supervised Semantic Segmentation Using Generative Adversarial Network," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 10 2017. doi: 10.1109/ICCV.2017.606. ISSN 2380-7504 pp. 5689–5697.

[355] W. C. Hung, Y. H. Tsai, Y. T. Liou, Y. Y. Lin, and M. H. Yang, "Adversarial learning for semi-supervised semantic segmentation," in *British Machine Vision Conference 2018, BMVC 2018*, 2019.

[356] Y. Xue, T. Xu, H. Zhang, L. R. Long, and X. Huang, "SegAN: Adversarial Network with Multi-scale L1 Loss for Medical Image Segmentation," *Neuroinformatics*, vol. 16, no. 3, pp. 383–392, 2018. [Online]. Available: https://doi.org/10.1007/s12021-018-9377-x

[357] F. Visin, A. Romero, K. Cho, M. Matteucci, M. Ciccone, K. Kastner, Y. Bengio, and A. Courville, "ReSeg: A Recurrent Neural Network-Based Model for Semantic Segmentation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 6 2016. doi: 10.1109/CVPRW.2016.60. ISSN 2160-7516 pp. 426–433.

[358] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. C. Courville, and Y. Bengio, "ReNet: A Recurrent Neural Network Based Alternative to Convolutional Networks," *CoRR*, vol. abs/1505.0, 2015. [Online]. Available: http://arxiv.org/abs/1505.00393

[359] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing ({EMNLP})*. Doha, Qatar: Association for Computational Linguistics, 10 2014. doi: 10.3115/v1/D14-1179 pp. 1724–1734. [Online]. Available: https://www.aclweb.org/anthology/D14-1179

[360] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki, "Scene labeling with LSTM recurrent neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2015. doi: 10.1109/CVPR.2015.7298977. ISSN 1063-6919 pp. 3547–3555.

[361] W. Byeon, M. Liwicki, and T. M. Breuel, "Texture Classification Using 2D LSTM Networks," in *2014 22nd International Conference on Pattern Recognition*, 8 2014. doi: 10.1109/ICPR.2014.206. ISSN 1051-4651 pp. 1144–1149.

[362] W. Byeon and T. M. Breuel, "Supervised texture segmentation using 2D LSTM networks," in *2014 IEEE International Conference on Image Processing (ICIP)*, 10 2014. doi: 10.1109/ICIP.2014.7025887. ISSN 2381-8549 pp. 4373–4377.

[363] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic Object Parsing with Graph LSTM," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016. ISBN 978-3-319-46448-0 pp. 125–143.

[364] M. Z. Alom, C. Yakopcic, M. Hasan, T. M. Taha, and V. K. Asari, "Recurrent residual U-Net for medical image segmentation," *Journal of Medical Imaging*, vol. 6, no. 1, pp. 1–16, 2019. [Online]. Available: https://doi.org/10.1117/1.JMI.6.1.014006

[365] Ming Liang and Xiaolin Hu, "Recurrent convolutional neural network for object recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2015. doi: 10.1109/CVPR.2015.7298958. ISSN 1063-6919 pp. 3367–3375.

[366] G. N. Hounsfield, "Computerized transverse axial scanning (tomography). 1. Description of system." *The British journal of radiology*, vol. 46, no. 552, pp. 1016–1022, 12 1973.

[367] K. R. Hill and R. W. Joyner, "Computerized x-ray tomography." *Science progress*, vol. 62, no. 246, pp. 237–262, 1975.

[368] C. Baillard, P. Hellier, and C. Barillot, "Segmentation of brain 3D MR images using level sets and dense registration," *Medical Image Analysis*, vol. 5, no. 3, pp. 185–194, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1361841501000391

[369] A. E. Lefohn, J. E. Cates, and R. T. Whitaker, "Interactive, GPU-Based Level Sets for 3D Segmentation," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003*, R. E. Ellis and T. M. Peters, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. ISBN 978-3-540-39899-8 pp. 564–572.

[370] Yan Yang, A. Tannenbaum, and D. Giddens, "Knowledge-based 3D segmentation and reconstruction of coronary arteries using CT images," in *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 1, 2004. doi: 10.1109/IEMBS.2004.1403502 pp. 1664–1666.

[371] P. A. Yushkevich, J. Piven, H. C. Hazlett, R. G. Smith, S. Ho, J. C. Gee, and G. Gerig, "User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability," *NeuroImage*, vol. 31, no. 3, pp. 1116–1128, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1053811906000632

[372] M. de Bruijne, B. van Ginneken, M. A. Viergever, and W. J. Niessen, "Adapting Active Shape Models for 3D Segmentation of Tubular Structures in Medical Images," in *Information Processing in Medical Imaging*, C. Taylor and J. A. Noble, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. ISBN 978-3-540-45087-0 pp. 136–147.

[373] M. Rousson, N. Paragios, and R. Deriche, "Implicit Active Shape Models for 3D Segmentation in MR Imaging," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004*, C. Barillot, D. R. Haynor, and P. Hellier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. ISBN 978-3-540-30135-6 pp. 209–216.

[374] T. Heimann, I. Wolf, and H.-P. Meinzer, "Active Shape Models for a Fully Automated 3D Segmentation of the Liver – An Evaluation on Clinical Data," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2006*, R. Larsen, M. Nielsen, and J. Sporring, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. ISBN 978-3-540-44728-3 pp. 41–48.

[375] K. H. Höhne and W. A. Hanson, "Interactive 3D Segmentation of MRI and CT Volumes using Morphological Operations," *Journal of Computer Assisted Tomography*, vol. 16, no. 2, 1992. [Online]. Available: https://journals.lww.com/jcat/Fulltext/1992/03000/Interactive_3D_Segmentation_of_MRI_and_CT_Volumes.19.aspx

[376] B. Bouraoui, C. Ronse, J. Baruthio, N. Passat, and P. Germain, "3D segmentation of coronary arteries based on advanced mathematical morphology techniques," *Computerized Medical Imaging and Graphics*, vol. 34, no. 5, pp. 377–387, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895611110000121

[377] J. Sijbers, P. Scheunders, M. Verhoye, A. Van der Linden, D. van Dyck, and E. Raman, "Watershed-based segmentation of 3D MR data for volume quantization," *Magnetic Resonance Imaging*, vol. 15, no. 6, pp. 679–688, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0730725X97000337

[378] X. Zhang, J. Tian, D. Xiang, X. Li, and K. Deng, "Interactive liver tumor segmentation from ct scans using support vector classification with watershed," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 8 2011. doi: 10.1109/IEMBS.2011.6091484. ISSN 1558-4615 pp. 6005–6008.

[379] A. A. Duquette, P.-M. Jodoin, O. Bouchot, and A. Lalande, "3D segmentation of abdominal aorta from CT-scan and MR images," *Computerized Medical Imaging and Graphics*, vol. 36, no. 4, pp. 294–303, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895611111001480

[380] R. Beichel, A. Bornik, C. Bauer, and E. Sorantin, "Liver segmentation in contrast enhanced CT data using graph cuts and interactive 3D segmentation refinement methods," *Medical Physics*, vol. 39, no. 3, pp. 1361–1373, 2012. [Online]. Available: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.3682171

[381] A. M. Ali and A. A. Farag, "Automatic lung segmentation of volumetric low-dose CT scans using graph cuts," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008. doi: 10.1007/978-3-540-89639-5_25. ISBN 3540896384. ISSN 03029743

[382] R. A. Heckemann, J. V. Hajnal, P. Aljabar, D. Rueckert, and A. Hammers, "Automatic anatomical brain MRI segmentation combining label propagation and decision fusion," *NeuroImage*, vol. 33, no. 1, pp. 115–126, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1053811906006458

[383] J. M. P. Lötjönen, R. Wolz, J. R. Koikkalainen, L. Thurfjell, G. Waldemar, H. Soininen, and D. Rueckert, "Fast and robust multi-atlas segmentation of brain magnetic resonance images," *NeuroImage*, vol. 49, no. 3, pp. 2352–2365, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1053811909010970

[384] X. Artaechevarria, A. Munoz-Barrutia, and C. Ortiz-de Solorzano, "Combination Strategies in Multi-Atlas Image Segmentation: Application to Brain MR Data," *IEEE Transactions on Medical Imaging*, vol. 28, no. 8, pp. 1266–1277, 8 2009.

[385] H. Wang, J. W. Suh, S. R. Das, J. B. Pluta, C. Craige, and P. A. Yushkevich, "Multi-Atlas Segmentation with Joint Label Fusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 611–623, 3 2013.

[386] A. Akselrod-Ballin, M. Galun, M. J. Gomori, R. Basri, and A. Brandt, "Atlas Guided Identification of Brain Structures by Combining 3D Segmentation and SVM Classification," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2006*, R. Larsen, M. Nielsen, and J. Sporring, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. ISBN 978-3-540-44728-3 pp. 209–216.

[387] A. M. Santos, A. O. de Carvalho Filho, A. C. Silva, A. C. de Paiva, R. A. Nunes, and M. Gattass, "Automatic detection of small lung nodules in 3D CT data using Gaussian mixture models, Tsallis entropy and SVM," *Engineering Applications of Artificial Intelligence*, vol. 36, pp. 27–39, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197614001754

[388] D. Anguelov, B. Taskarf, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative learning of Markov random fields for segmentation of 3D scan data," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, 6 2005. doi: 10.1109/CVPR.2005.133. ISSN 1063-6919 pp. 169–176.

[389] M. S. Hassouna, A. A. Farag, S. Hushek, and T. Moriarty, "Cerebrovascular segmentation from TOF using stochastic models," *Medical Image Analysis*, vol. 10, no. 1, pp. 2–18, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1361841505000186

[390] X. Dong, Y. Lei, T. Wang, M. Thomas, L. Tang, W. J. Curran, T. Liu, and X. Yang, "Automatic multiorgan segmentation in thorax CT images using U-net-GAN," *Medical Physics*, vol. 46, no. 5, pp. 2157–2168, 2019. [Online]. Available: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1002/mp.13458

[391] F. Milletari, N. Navab, and S. A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 4th International Conference on 3D Vision, 3DV 2016*, 2016. doi: 10.1109/3DV.2016.79. ISBN 9781509054077 pp. 565–571.

[392] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for Simplicity: The All Convolutional Net," in *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Contribution*, 12 2015. [Online]. Available: http://arxiv.org/abs/1412.6806

[393] L. R. Dice, "Measures of the Amount of Ecologic Association Between Species," *Ecology*, 1945.

[394] Zhuowen Tu, "Auto-context and its application to high-level vision tasks," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 6 2008. doi: 10.1109/CVPR.2008.4587436. ISSN 1063-6919 pp. 1–8.

[395] P. F. Christ, M. E. A. Elshaer, F. Ettlinger, S. Tatavarty, M. Bickel, P. Bilic, M. Rempfler, M. Armbruster, F. Hofmann, M. D'Anastasi, W. H. Sommer, S.-A. Ahmadi, and B. H. Menze, "Automatic Liver and Lesion Segmentation in CT Using Cascaded Fully Convolutional Neural Networks and 3D Conditional Random Fields," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, Eds. Cham: Springer International Publishing, 2016. ISBN 978-3-319-46723-8 pp. 415–423.

[396] K. C. L. Wong, M. Moradi, H. Tang, and T. Syeda-Mahmood, "3D Segmentation with Exponential Logarithmic Loss for Highly Unbalanced Object Sizes," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, A. F. Frangi, J. A. Schnabel, C. Davatzikos, C. Alberola-López, and G. Fichtinger, Eds. Cham: Springer International Publishing, 2018. ISBN 978-3-030-00931-1 pp. 612–619.

[397] A. Myronenko, "3D MRI Brain Tumor Segmentation Using Autoencoder Regularization," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi, S. Bakas, H. Kuijf, F. Keyvan, M. Reyes, and T. van Walsum, Eds. Cham: Springer International Publishing, 2019. ISBN 978-3-030-11726-9 pp. 311–320.

[398] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[399] Y. Wu and K. He, "Group Normalization," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018. ISBN 978-3-030-01261-8 pp. 3–19.

[400] J. Chen, L. Yang, Y. Zhang, M. Alber, and D. Z. Chen, "Combining fully convolutional and recurrent neural networks for 3D biomedical image segmentation," in *Advances in Neural Information Processing Systems 29*, D. D. L. Garnett, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016. ISSN 10495258 pp. 3036–3044. [Online]. Available: http://papers.nips.cc/paper/6448-combining-fully-convolutional-and-recurrent-neural-networks-for-3d-biomedical-image-segmentation.pdf

[401] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems 28*, 2015. ISSN 10495258 pp. 802–810. [Online]. Available: http://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting.pdf

[402] L. Yu, J.-Z. Cheng, Q. Dou, X. Yang, H. Chen, J. Qin, and P.-A. Heng, "Automatic 3D Cardiovascular MR Segmentation with Densely-Connected Volumetric ConvNets," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2017*, M. Descoteaux, L. Maier-Hein, A. Franz, P. Jannin, D. L. Collins, and S. Duchesne, Eds. Cham: Springer International Publishing, 2017. ISBN 978-3-319-66185-8 pp. 287–295.

[403] E. Gibson, F. Giganti, Y. Hu, E. Bonmati, S. Bandula, K. Gurusamy, B. Davidson, S. P. Pereira, M. J. Clarkson, and D. C. Barratt, "Automatic Multi-Organ Segmentation on Abdominal CT With Dense V-Networks," *IEEE Transactions on Medical Imaging*, vol. 37, no. 8, pp. 1822–1834, 8 2018.

[404] X. Li, H. Chen, X. Qi, Q. Dou, C. W. Fu, and P. A. Heng, "H-DenseUNet: Hybrid Densely Connected UNet for Liver and Tumor Segmentation from CT Volumes," *IEEE Transactions on Medical Imaging*, vol. 37, no. 12, pp. 2663–2674, 2018.

[405] RADON and J., "On the Determination of Functions from their Integrals along Certain Manifolds," *Ber. Verh, Sachs Akad Wiss.*, vol. 69, pp. 262–277, 1917. [Online]. Available: https://ci.nii.ac.jp/naid/10006638717/

[406] J. Radon, "On the determination of functions from their integral values along certain manifolds," *IEEE Transactions on Medical Imaging*, vol. 5, no. 4, pp. 170–176, 12 1986. [Online]. Available: http://ieeexplore.ieee.org/document/4307775/

[407] L. A. Shepp and B. F. Logan, "FOURIER RECONSTRUCTION OF A HEAD SECTION," *IEEE Transactions on Nuclear Science*, 1974.

[408] N. T. Vo, M. Drakopoulos, R. C. Atwood, and C. Reinhard, "Reliable method for calculating the center of rotation in parallel-beam tomography," *Optics Express*, vol. 22, no. 16, p. 19078, 8 2014. [Online]. Available: https://www.osapublishing.org/oe/abstract.cfm?uri=oe-22-16-19078

[409] D. E. Dudgeon and R. M. Mersereau, "Multidimensional Digital Signal Processing," 1985.

[410] R. Gordon, R. Bender, and G. T. Herman, "Algebraic Reconstruction Techniques (ART) for three-dimensional electron microscopy and X-ray photography," *Journal of Theoretical Biology*, vol. 29, no. 3, pp. 471–481, 12 1970.

[411] A. H. Andersen and A. C. Kak, "Simultaneous Algebraic Reconstruction Technique (SART): A superior implementation of the ART algorithm," *Ultrasonic Imaging*, vol. 6, no. 1, pp. 81–94, 6 1984. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0161734684900087

[412] P. Gilbert, "Iterative methods for the three-dimensional reconstruction of an object from projections," *Journal of Theoretical Biology*, vol. 36, no. 1, pp. 105–117, 7 1972.

[413] J. A. Scales, "Tomographic inversion via the conjugate gradient method." *Geophysics*, vol. 52, no. 2, pp. 179–185, 2 1987. [Online]. Available: https://library.seg.org/doi/abs/10.1190/1.1442293

[414] M. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, 1952.

[415] M. Drakopoulos, T. Connolley, C. Reinhard, R. Atwood, O. Magdysyuk, N. Vo, M. Hart, L. Connor, B. Humphreys, G. Howell, S. Davies, T. Hill, G. Wilkin, U. Pedersen, A. Foster, N. De Maio, M. Basham, F. Yuan, and K. Wanelik, "I12: the Joint Engineering, Environment and Processing (JEEP) beamline at Diamond

Light Source," *Journal of Synchrotron Radiation*, vol. 22, no. 3, pp. 828–838, 5 2015. [Online]. Available: http://scripts.iucr.org/cgi-bin/paper?S1600577515003513

[416] A. J. Bodey and C. Rau, "Launch of the I13-2 data beamline at the Diamond Light Source synchrotron," *Journal of Physics: Conference Series*, vol. 849, no. 1, p. 012038, 2017. [Online]. Available: http://stacks.iop.org/1742-6596/849/i=1/a=012038

[417] B. Singh, R. Bartolini, R. Fielder, E. Longhi, I. Martin, C. Rau, and U. Wagner, "Double Mini-Beta-Y Plus Virtual Focussing Optics for the Diamond Storage Ring," in *Particle Accelerator Conference (PAC 09)*, 2010, p. TH6PFP033.

[418] F. Eckermann, T. Suter, P. J. Uggowitzer, A. Afseth, A. J. Davenport, B. J. Connolly, M. H. Larsen, F. D. Carlo, and P. Schmutz, "In situ monitoring of corrosion processes within the bulk of AlMgSi alloys using X-ray microtomography," *Corrosion Science*, vol. 50, no. 12, pp. 3455–3466, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010938X08003739

[419] D. Bellos, M. Basham, A. P. French, and T. Pridmore, "UDNN_Real_Datasets, https://zenodo.org/record/1299359," 10 2018. [Online]. Available: https://zenodo.org/record/1299359#.W9crKpP7SUk

[420] D. Bellos, A. P. French, M. Basham, and T. Pridmore, "Micro X-ray CT time-series (4D dataset)," 11 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4293394#.X9IpX3PW7vg.mendeley

[421] M. Könnecke, F. A. Akeroyd, H. J. Bernstein, A. S. Brewster, S. I. Campbell, B. Clausen, S. Cottrell, J. U. Hoffmann, P. R. Jemian, D. Männicke, R. Osborn, P. F. Peterson, T. Richter, J. Suzuki, B. Watts, E. Wintersberger, and J. Wuttke, "The NeXus data format," *Journal of Applied Crystallography*, vol. 48, no. 1, pp. 301–305, 6 2015. [Online]. Available: http://journals.iucr.org/j/issues/2015/01/00/po5029/po5029.pdf

[422] T. H. D. F. Group, "Hierarchical Data Format, version 5," 1997. [Online]. Available: http://www.hdfgroup.org/HDF5/

[423] R. C. Atwood, A. J. Bodey, S. W. Price, M. Basham, and M. Drakopoulos, "A high-throughput system for high-quality tomographic reconstruction of large datasets at diamond light source," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 373, no. 2043, p. 20140398, 6 2015. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rsta.2014.0398

[424] D. Bellos, M. Basham, A. P. French, and T. Pridmore, "Stacked Dense Denoise-Segmentation Volumetric Annotations," 4 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2646117

[425] D. Bellos, M. Basham, A. P. French, and T. Pridmore, "Stacked Dense Denoise-Segmentation FBP reconstruction from 3601 projection," 4 2019. [Online]. Available: https://zenodo.org/record/2645965#.XLiEKnVKiA0

[426] D. Bellos, M. Basham, A. P. French, and T. Pridmore, "Stacked Dense Denoise-Segmentation CGLS reconstruction from 91 projection," 4 2019. [Online]. Available: https://zenodo.org/record/2645838#.XLiEKHVKiA0

[427] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rübel, M. Durant, J. M. Favre, and P. Navrátil, "VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data," in *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, 10 2012, pp. 357–372.

[428] D. Bellos, M. Basham, A. P. French, and T. Pridmore, "UDNN_Synthetic_Datasets, https://zenodo.org/record/1306243," 10 2018. [Online]. Available: https://zenodo.org/record/1306243#.W9crtZP7SUk

[429] D. Bellos, A. P. French, M. Basham, and T. Pridmore, "Stacked Dense Denoise-Segmentation CGLS synthetic reconstruction from 91 projection with ring artefacts," 8 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3986506#.Xzq2YTp4xNY.mendeley

[430] D. Bellos, A. P. French, M. Basham, and T. Pridmore, "Stacked Dense Denoise-Segmentation FBP synthetic reconstruction from 3601 projection with ring artefacts," 8 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3986502#.Xzq2kjpcFEo.mendeley

[431] D. Bellos, A. P. French, M. Basham, and T. Pridmore, "Stacked Dense Denoise-Segmentation CGLS synthetic reconstruction from 91 projection without ring artefacts," 8 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3986508#.Xzq2SfAiX4E.mendeley

[432] D. Bellos, A. P. French, M. Basham, and T. Pridmore, "Stacked Dense Denoise-Segmentation FBP synthetic reconstruction from 3601 projection without ring artefacts," 8 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3986504#.Xzq2xIJZsp0.mendeley

[433] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, vol. 15, 2011. doi: 10.1.1.208.6449. ISBN 1532-4435. ISSN 15324435 pp. 315–323.

[434] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning*, ser. ICML'10, no. 3. USA: Omnipress, 2010. doi: 10.1.1.165.6419. ISBN 9781605589077. ISSN 1935-8237 pp. 807–814. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104425

[435] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, 6 2014. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf

[436] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*,

vol. 12, no. Jul, pp. 2121–2159, 6 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.htmlhttp://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf

[437] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," 2012.

[438] N. Wadeson and M. Basham, "Savu: A Python-based, MPI Framework for Simultaneous Processing of Multiple, N-dimensional, Large Tomography Datasets," *arXiv:1610.08015 [cs]*, 6 2016. [Online]. Available: https://www.arxiv.org/pdf/1610.08015.pdf

[439] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. [Online]. Available: https://ieeexplore.ieee.org/ielx5/83/28667/01284395.pdf?tp=&arnumber=1284395&isnumber=28667

[440] W. van Aarle, W. J. Palenstijn, J. De Beenhouwer, T. Altantzis, S. Bals, K. J. Batenburg, and J. Sijbers, "The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography," *Ultramicroscopy*, vol. 157, pp. 35–47, 10 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304399115001060?casa_token=t4Qr0HI-4m8AAAAA:uJ971NrHx2ayNuLgddHPRl-3R5czAQ2Eu3YV72GIUElHjpw5OyFo1xZRSpTYhX-PhRTg3x7pLw

[441] D. Bellos, M. Basham, A. P. French, and T. Pridmore, "A Stacked Dense Denoising-Segmentation Network for undersampled tomograms and knowledge transfer using synthetic tomograms," *Machine Vision and Applications, revised manuscript submitted, acceptance pending*, 2020.

[442] A. Paszke, S. Chintala, G. Chanan, Z. Lin, S. Gross, E. Yang, L. Antiga, Z. Devito, A. Lerer, and A. Desmaison, "Automatic differentiation in PyTorch," in *31st Conference on Neural Information Processing Systems*, 2017. doi: 10.1017/CBO9781107707221.009. ISBN 9788578110796. ISSN 1098-6596

[443] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, 2014. doi: 10.1145/2647868.2654889. ISBN 9781450330633

[444] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, 2016. ISBN 9781931971331

[445] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih, "PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation," *Parallel Computing*, 2012.

[446] C. W. Hecksel, M. C. Darrow, W. Dai, J. G. Galaz-Montoya, J. A. Chin, P. G. Mitchell, S. Chen, J. Jakana, M. F. Schmid, and W. Chiu, "Quantifying Variability of Manual Annotation in Cryo-Electron Tomograms." *Microscopy and microanalysis : the official journal of Microscopy Society of America, Microbeam Analysis Society, Microscopical Society of Canada*, vol. 22, no. 3, pp. 487–496, 6 2016.

[447] A. Krogh, J. a. Hertz, and a. K. Connect, "A Simple Weight Decay Can Improve Generalization," *Advances in Neural Information Processing Systems*, vol. 4, pp. 950–957, 1992. [Online]. Available: http://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization.pdf

[448] C. J. Tsai, Y. W. Tsai, S. L. Hsu, and Y. C. Wu, "Synthetic Training of Deep CNN for 3D Hand Gesture Identification," in *Proceedings - 2017 International Conference on Control, Artificial Intelligence, Robotics and Optimization, ICCAIRO 2017*, 2017. doi: 10.1109/ICCAIRO.2017.40. ISBN 9781509065363

[449] K. Lindgren, N. Kalavakonda, D. E. Caballero, K. Huang, and B. Hannaford, "Learned Hand Gesture Classification Through Synthetically Generated Training Samples," in *IEEE International Conference on Intelligent Robots and Systems*, 2018. doi: 10.1109/IROS.2018.8593433. ISBN 9781538680940. ISSN 21530866

[450] D. H. Laidlaw, W. B. Trumbore, and J. F. Hughes, "Constructive Solid Geometry for Polyhedral Objects," in *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '86. New York, NY, USA: Association for Computing Machinery, 1986. doi: 10.1145/15922.15904. ISBN 0897911962 p. 161–170. [Online]. Available: https://doi.org/10.1145/15922.15904

[451] D. H. Laidlaw, W. B. Trumbore, and J. F. Hughes, "Constructive Solid Geometry for Polyhedral Objects," *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, p. 161–170, 8 1986. [Online]. Available: https://doi.org/10.1145/15886.15904

[452] D. Bellos, A. P. French, M. Basham, and T. Pridmore, "Stacked Dense Denoise-Segmentation Synthetic Annotations," 8 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3986494#.Xzq1ipd9Ms8.mendeley

[453] Z. Song, O. V. Magdysyuk, L. Tang, T. Sparks, and B. Cai, "Growth dynamics of faceted Al13Fe4 intermetallic revealed by high-speed synchrotron X-ray quantification," *Journal of Alloys and Compounds*, vol. 861, p. 158604, 4 2021.

[454] M. Polacci, F. Arzilli, G. La Spina, N. Le Gall, B. Cai, M. E. Hartley, D. Di Genova, N. T. Vo, S. Nonni, R. C. Atwood, E. W. Llewellin, P. D. Lee, and M. R. Burton, "Crystallisation in basaltic magmas revealed via in situ 4D synchrotron X-ray microtomography," *Scientific Reports*, vol. 8, no. 1, p. 8377, 2018. [Online]. Available: https://doi.org/10.1038/s41598-018-26644-6

[455] A. Markov, *Wahrscheinlichkeitsrechnung*. Leipzig-Berlin: B. G. Teubner, 1912.

[456] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 4 1967.

[457] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 10 2017. doi: 10.1109/ICCV.2017.244. ISSN 2380-7504 pp. 2242–2251.