

University of Nottingham
UK | CHINA | MALAYSIA

Ultra high-resolution segmentation of biological images using reduced-memory convolutional neural networks with inter-tile communication

Ezenwoko Nnabuike Onassis Obi Benson

Supervised by Dr Michael Pound

Computer Vision Laboratory
School of Computer Science
University of Nottingham

Thesis submitted for the degree of
Doctor of Philosophy

September 2021

Acknowledgements

I would like to thank Michael Pound, my supervisor, for listening to all my crazy ideas and always being available to help. His endless enthusiasm and positivity gave me the continued encouragement I needed throughout my PhD.

I'd also like to thank my other supervisor Tony Pridmore for his continued guidance, there were many situations I wouldn't have been able to navigate without his advice.

Abstract

Advances in imaging technology continue to outpace the innovations in computing hardware. For some tasks such as object classification this gap presents almost no problem, given that the results are largely unaffected if some pixel data is missing. This task can produce accurate results by shrinking the image since not all pixel information is required. As such, currently available hardware can be used to complete this task. Tasks like segmentation produce dense per-pixel classifications and benefit greatly from processing all of the pixels in an image, unchanged. In this task only a small amount of image shrinking can take place before the results are affected. Small images can be segmented however, the computational cost becomes prohibitive when applying the same methods to high resolution images. This hasn't been adequately addressed by current research because of the focus on performance on low resolution benchmark datasets. This thesis presents a novel solution which builds upon the commonly used technique of image tiling. Image tiling is where an image is divided into smaller sub-images called tiles. Processing tiles reduces the memory burden therefore, allowing for the segmentation of larger images. Segmenting a single tile only allows the method to view the context within a tile, which results in inaccurate segmentations when a wider context is required. In this work, context is shared between tiles using global filters. This allows corrections to be made to predictions based on the wider context of the image. The method is shown to be competitive on low resolution datasets against a baseline. It is also shown to outperform high resolution segmentation techniques on high resolution datasets.

Contents

Acknowledgements **i**

Abstract **iii**

1 Introduction **1**

1.1 Aims and Scope 1

1.2 Technical Challenges 2

1.3 Motivations 2

1.4 Contributions 4

2 Background **6**

2.1 Deep Learning 6

2.1.1 Overview 6

2.1.2 Dimensionality and Convolutions 8

2.2 Literature Review 9

2.2.1 Early Deep Learning and the Transition to Semantic Segmentation 9

2.2.2 Residual Learning 11

2.2.3 General Segmentation Literature 12

2.2.4 Efficient Semantic Segmentation 14

2.2.5 High Resolution Semantic Segmentation 16

2.3 Summary 18

3 Datasets **20**

3.1 Introduction 20

3.2 Lower Resolution Semantic Segmentation Datasets 21

3.3 High Resolution Semantic Segmentation Datasets 22

3.4 MRI Colon Dataset 24
3.4.1 Motivation 24
3.4.2 The Data 26
3.4.3 Tasks 28
3.5 Laser Ablation Tomography Root Cross Section Dataset 28
3.5.1 Motivation 31
3.5.2 Task 32
3.6 Summary 32

4 Investigating the effect of data dimensionality on deep learning 34

4.1 Introduction 34
4.2 Revisiting the data 35
4.3 Method 35
4.3.1 Model Architectures 36
4.3.2 Training 39
4.4 Quantitative Results 39
4.4.1 Single Vs Dual Channel 39
4.4.2 Generalisability across datasets 41
4.4.3 Sources of error 42
4.4.4 2.5D and 3D Data Ablation 43
4.4.5 Inference Time 45
4.5 Qualitative Analysis 46
4.5.1 Phillips Achieva Dataset 46
4.5.2 GE Dataset 48
4.6 Summary 51

5 Common Techniques for segmenting high resolution images 54

5.1 Introduction 54
5.2 Downsampling 55
5.2.1 Experimental Design 57
5.2.2 Results 58

5.3 Tiling 60
5.3.1 Experimental Design 61
5.3.2 Results 61
5.4 Conclusion 62

6 Revisiting Tiling - Global Context Aggregation Via Crosstalk 66

6.1 Introduction 66
6.2 Converting Networks 67
6.2.1 Selective Backpropagation 68
6.3 Post Tiling filters 69
6.4 Global Context Aggregation Via Crosstalk 71
6.4.1 Attention for selection 73
6.4.2 Pooling and Spatial Alignment 75
6.4.3 Global Image Filtering and Memory 76
6.4.4 Intergrating Crosstalk 76
6.5 Crosstalk Results 77
6.5.1 An Alternative Lightweight Tiling Computation 80
6.5.2 Crosstalk Module Placement in the Decoder 84
6.5.3 Varying Target Pooling Size 85
6.5.4 Varying Global Filter Hyperparameters 86
6.5.5 Adding Crosstalk Memory 88
6.5.6 Varying Tile Count in Crosstalk 89
6.5.7 Additional Lesser Ablation Studies 91
6.6 Crosstalk Final Comparison 93
6.7 Summary 95

7 Proving Crosstalk Across Datasets and Networks 97

7.1 Introduction 97
7.2 Cityscapes and DeepLab 98
7.2.1 Revisiting the Data 98
7.2.2 DeepLabV3+ Overview 99

7.2.3 Checkpointing Overview . . . 100
7.2.4 Adapting DeepLab . . . 101
7.2.5 Experimental Design . . . 102
7.2.6 Results . . . 103
7.2.7 Ablation Studies . . . 109
7.3 DeepGlobe . . . 114
7.3.1 Revisiting the Data . . . 114
7.3.2 Experimental Design . . . 114
7.3.3 Results . . . 115
7.4 Synthetic Dataset . . . 115
7.5 Summary . . . 117
8 Conclusion **121**
8.1 Contributions . . . 121
8.2 Limitations and Suggestions for Further Improvements . . . 123
8.2.1 Explicit Attention Between Tiles . . . 124
8.2.2 Competing With the Baseline . . . 125
8.2.3 A Multi-scale Crosstalk . . . 125
8.2.4 Incorporating More Global Information . . . 126
8.2.5 Guiding Crosstalk Through Intermediary Losses . . . 127
8.2.6 3D Crosstalk . . . 127
8.3 Summary . . . 127
Bibliography **129**

List of Tables

3.1 Popular, high resolution and ultra high resolution semantic segmentation datasets. HR denotes datasets which on average contain high resolution images. UHR denotes datasets which on average contain ultra high resolution images. The † symbol indicates the one dataset which is 3D and does not fit the category outlined categories. 21

4.1 Segmentation accuracy results, measured using Dice score, obtained on all three networks applied to test subset of the Philips dataset. WC denotes the average segmentation accuracy over all four individually segmented classes (including BG). AC denotes segmentation of the ascending colon, DC the descending colon, and TC the transverse colon. Binary segmentation refers to the task of segmenting the regions containing the colon from the BG. 41

4.2 Segmentation accuracy results, measured using Dice score, on all 3 networks on the GE dataset. WC denotes the average segmentation accuracy over all four individually segmented classes (including BG). AC denotes segmentation of the ascending colon, DC the descending colon, and TC the transverse colon. Binary segmentation refers to the task of segmenting the regions containing the colon from the BG. 42

4.3 Results obtained for 2.5D and 3D networks on ablated test images from the Philips Achieva scanner 45

4.4 Time taken for each network to make predictions on a single volume from a Philips Achieva scanner 46

5.1 The accuracy of the hourglass network when trained with various input resolutions. Each network is evaluated at the trained input resolutions as well as increasingly upsampled resolutions up to the native image resolution of 3072×3072. Accuracy is measured using Dice score. 59

5.2 The accuracy of the hourglass networks when trained with various tile sizes at various whole image resolutions. All networks are evaluated at the resolution they are trained at using the stitched tile outputs from the network. Accuracy is measured in dice score. '-' denotes an experiment where the tile size is equivalent to the image size and therefore is not conducted. . . . 62

6.1 Accuracy and VRAM consumption of hourglass when comparing a naive tiled network (No XT and no post filtering) against an hourglass with added bottleneck blocks post tile stitching. Filters are run over the stitched image after the prediction layer has been executed, this means the number of channels is equivalent to the number of classes. The image used in this example is an LAT image of dimensions 2048×2048 at a tile size of 512×512. 70

6.2 Accuracy and VRAM consumption of hourglass when comparing a naive tiled network (No XT and no post filtering) against an hourglass with added basic resblocks blocks post tile stitching. The image used in this example is an LAT image of dimensions 2048×2048 at a tile size of 512×512. 71

6.3 Results of two tiling networks on the LAT dataset. One network includes the XT module at each layer of the decoder, including the bottleneck. Results are given for 3 input image resolutions and 3 tile sizes. 78

6.4 Results of two tiling networks on the LAT dataset. One network includes the XT module at each layer of the decoder, including the bottleneck. Results are given for 3 input image resolutions and 3 tile sizes. Results here are the accompanying VRAM consumption data for 6.3. 80

6.5 The memory consumption of each of the tiling methods with and without XT. VRAM is given at all resolutions and tile sizes from $1024 \times 1024 - 3072 \times 3072$ and $128 \times 128 - 512 \times 512$ respectively. 82

6.6 The accuracy of each of the tiling methods with and without XT. VRAM is given at all resolutions and tile sizes from $1024 \times 1024 - 3072 \times 3072$ and $128 \times 128 - 512 \times 512$ respectively. . . 83

6.7 Accuracy results over a range of image sizes and tile sizes where the XT module placement is varied between processing featuremaps before the network decision layer and after it. Note the '-' symbol denotes an experiment which cannot be considered due to VRAM limitations. 84

6.8 VRAM consumption data over a range of image sizes and tile sizes where the XT module placement is varied between processing featuremaps before the network decision layer and after it. Note the '-' symbol denotes an experiment where the VRAM consumption exceeded the 12.3GB limit on the graphics card. . 84

6.9 The effect of XT modules placed at different levels of the decoder. Bottleneck is the region between encoder and decoder and such is before the first level of the decoder. All experiments are on images sized 3072×3072 at a tile size of 512×512 with a XT pooling target size of 4×4. 85

6.10 The effect of varying the pooling size on network accuracy. Results are given for images of resolution 3072×3072. 86

6.11 The effect on accuracy of different resblock configurations in the XT module. Block is an abbreviation for resblock whilst RFS is an abbreviation for receptive field size. The filters column gives the number of resblocks and the size of the kernels in them. Experimental results are given for images at resolution 2048×2048. The XT module uses a pooling size of 8×8, this is used instead of the usual 4×4 to allow for more experimentation with receptive field size. 87

6.12 The effect of adding memory to the XT module. Results are given for a network where XT modules are placed at every level of the decoder. Input images sizes are 3072×3072 with a tile size of 512×512. 89

6.13 The effect of varying the number of included neighbouring tiles in XT. All results are given for images of 3072×3072 at a tile size of 512×512. 91

6.14 The results of the best trained networks using methods from this chapter and from chapter 5. Naive tiling is given two rows because the results are very similar in the case of the higher resolution images. Results are given for 3072×3072 and 2048×2048 resolutions. Tiling XT uses a pooling size of 8×8 with a tile size of 512×512. It uses 2 resblocks with kernel size 3×3. Whilst Tiling-Light XT uses a tile size of 1024×1024 with a pooling size of 4×4. Only 1 3×3 kernel resblock is used here. Note the '-' symbol denotes an experiment which cannot be considered due to VRAM limitations. 94

7.1 Comparison across the Cityscapes dataset of the original implementation of DeepLabV3+ against our best performing tiled adaptation. VRAM usage at training and test time is given in GB for a batch size of 1, with relative memory usage shown in brackets as a percentage. We include results on networks implementing checkpointing showing additional memory savings at the cost of training time. Some test results are omitted due to submission rate limitation on the cityscapes evaluation server. '-' Denotes where values are missing due to the code being unavailable and the paper not specifying the values. 105

7.2 The accuracy of high resolution segmentation networks on each Cityscapes class in the test set. Note that the GLNet values are obtained through the the training implementation in this thesis as GLNet was not applied to cityscapes in the original works. . 106

7.3 Effect of tile configuration with and without XT, given as the meanIoU, along with required VRAM in GB. 111

7.4 Effect of target pooling size within XT on overall network accuracy. The network trained uses a single Resblock with a 5×5 kernel in XT and the tile configuration used for this setup is 2×2. 111

7.5 The effect of varying the number of consecutive XT modules on overall network accuracy. A tiling configuration of of 2×2 was used in this ablation study. 112

7.6 The effect of attention and its placement in XT on the accuracy of the final network. The network trained here uses a tiling configuration of 4×4. 113

7.7 The results on the DeepGlobe test set of each high resolution segmentation network. VRAM is given in GB for each network at training time, a '-' denotes where the consumption is unknown. This is for cases where authors did not specify the memory consumption and where code is not available to run. . . 115

List of Figures

3.1 (A) displays an example input image from the cityscapes training set. This image is taken in the city of Darmstadt, the ground truth annotation is shown in (B). 22

3.2 An example input image from the DeepGlobe dataset is shown on the left (A) and on the right (B) the ground truth mask with up to 7 classes categorising land coverage. 23

3.3 An example input image from the Inria dataset is shown on the left (A) and on the right (B) is the ground truth mask. The dataset consist of satelitte images with the goal of segmenting building coverage. 24

3.4 Examples of echo-1 and echo-2 contrasts in a single slice from the Philips Achieva dataset. 25

3.5 The dataset of 51 participants all of which received up to 7 scans over 2 or 3 visits. Scans acquired during the same visit are used in the same set. The training, validation and test sets are illustrated in beige, orange and red. If scans from a participant were used as part of the training set they were not included in the test set, this is to ensure the network does not learn any participant specific ques that could help performance. 27

3.6 (A) is an example Maize root cross section acquired through LAT. (B) is the accompanying annotation mask created by Penn State Plant Biologists using Cellset. A colour key for cell types (classes) is given in the bottom right of (B). 29

4.1 The hourglass model used for the 2D data 37

4.2 The 2.5D hourglass architecture. The asterisk on the displayed input image indicates which slice of the input is segmented. The 2.5D network only segments the middle slice but uses surrounding slices for context 37

4.3 The 3D hourglass architecture used during experiments 38

4.4 (a) Plot showing the change in segmentation accuracy during training time for input volumes with a single contrast (Echo-1 or Echo-2), and for dual contrast (combined) inputs. (b) Box plots showing the 10th , 25th, 50th , 75th, and 90th percentile of the Dice scores achieved by 3D models trained on dual-contrast images, echo-1, and echo-2 images respectively. Results obtained on the Philips Achieva test data. 40

4.5 Class confusion matrices for 2D (a), 2.5D (b) and 3D (c) for the GE Medical Systems HDxt scanner. Each has been normalized and shows the percentage of class confusion for each predicted and true label. 42

4.6 Class confusion matrices for 2D (a), 2.5D (b) and 3D (c) Philips Achieva scanner. Each has been normalized and shows the percentage of class confusion for each predicted and true label. . . . 43

4.7 Examples of network predictions on a volume with high accuracy. Coloured regions denote the ground truth with the blue, orange and green corresponding to AC, TC and DC respectively. Slices with purple labels are the corresponding binary segmentations. The network outputs are delineated by the thin contour lines. 47

4.8 Examples of network predictions on a volume with only average accuracy. Coloured regions denote the ground truth with the blue, orange and green corresponding to AC, TC and DC respectively. Slices with purple labels are the corresponding binary segmentations. The network outputs are delineated by the thin contour lines. 48

4.9 Examples of network predictions on a volume with a low accuracy. Coloured regions denote the ground truth with the blue, orange and green corresponding to AC, TC and DC respectively. Slices with purple labels are the corresponding binary segmentations. The network outputs are delineated by the thin contour lines.. 49

4.10 Examples of network predictions on a volume with the highest accuracy between networks on the GE scanner.Coloured regions denote the ground truth with the blue, orange and green corresponding to AC, TC and DC respectively. Slices with pink labels are the corresponding binary segmentations. The network outputs are delineated by the thin contour lines. 50

4.11 Noisy 2D network predictions in coronal slices. (a) and (b) do not contain any colon regions. All images show incorrect predictions whilst (a) and (c) show artefacts within the MRI which cause misclassification. Coloured regions denote the ground truth with the blue, orange and green corresponding to AC, TC and DC respectively. The network outputs are delineated by the thin contour lines. 50

5.1 The Stele cropped from a root cross section which has been downsampled by a factor of 4 and 6 resulting in an image size of 1024×1024 and 512×512. The stele cropped from the original image with no downsampling is also presented. The large black vessels forming a radial pattern are Xylem, this feature remains largely unaffected where only its borders are made blurry. Small cells, especially those around the xylem and outer most edge of the Stele are almost lost. 57

5.2 This figure shows cropped outputs from tiling networks. From left to right the images show the cropped input, the cropped ground truth (GT) and cropped outputs from Hourglass networks trained on images downsampled to 256×256 and 1024×1024. The input image shows the image at the original resolution of 3072×3072. 59

5.3 This figure shows cropped outputs from tiling networks. From left to right the images show the cropped input, the cropped ground truth (GT) and cropped outputs from Hourglass networks trained on tiles of sizes 256×256 and 512×512. All tiles are randomly selected from the input images at a native resolution of 3072×3072. 63

6.1 A high-level overview of in-network tiling. A tiling scheme of 4 tiles is shown in a 2×2 configuration. Forward passes are performed individually and sequentially. Gradient flow is highlighted using the red arrow. As shown backpropagation is only applied to a single tile on any given training iteration. 69

6.2 An overview of the XT module from a single tile perspective. Feature maps from each tile are passed through channel and spatial attention sequentially. These features are then average pooled to a pre-set size. The new low resolution features are passed through a global image filtering residual block to share information between tiles. The result of this global image filtering is combined with global image features from the XT block of the previous decoder level. Upsampling is used to return the features to the same size before pooling after which tiles are separated again. Spatial and channel attention is applied before finally combining the feature maps originally input to XT. . . . 73

6.3 The data from table 6.3 is plotted here. The tile size input image ratio is the tile size dimensions divided by the input image dimensions. 79

7.1 Images from the cityscapes validation set along with ground truth and network outputs from each high resolution segmentation technique. Left (A), showcases an example of the rider and motorcycle class. Right (B) showcases a difficult example which includes terrain and many small and distant objects. Note that the code for MagNet removes the ignored class from the images represented by the black pixels. This gives the image a more refined look but is not an actual network prediction. GLNet and XT display noise in these regions as a result of not replacing the pixels. 109

7.2 This figure displays 3 example scenarios from the randomly generated dataset. (A) shows the case where only the white circle is rendered resulting in it being segmented as a blue circle (class 1). If the grey square is rendered the circle should be segmented as a red circle (class 2) as shown in (B) and (C). White lines are rendered to illustrate tile boundaries in the 4×4 tiling scheme used. 117

Chapter 1

Introduction

1.1 Aims and Scope

This thesis aims to create a method for segmenting high and ultra high resolution images using deep learning. Segmenting these images is a difficult problem because larger images require excessive amounts of memory during training. Memory is not a barrier in the general use of deep learning since what must be stored is much less than at training time. As such the method aims to reduce the GPU memory cost during training alone and is not concerned with consumption in other stages. This method should be applicable to commonly used high performance convolutional neural networks (CNNs) and should directly solve the problem which is referred to as 'lack of global context'. Global context is the wider context available in the image which can be used to generate correct predictions. For some tasks position in the image or surrounding features can determine the class of an object or pixel. Therefore if a network processes a high resolution image by splitting it into sub images (tiling) it will lose global context. The method outlined in this thesis opts to use this technique and as such must tackle the problem of global context.

The thesis does not aim to create a new network but rather a method for modifying pre-existing networks. Modifying pre-existing networks allows the

technique to leverage the power of established state of the art networks on high resolution segmentation tasks. Moreover it allows other researchers to combine this method with their own networks when tackling new tasks. The method is accompanied by a new module called crosstalk. The module is specifically designed to resolve the problem of lack of global context since the method advocated for in this thesis is a variant of tiling.

1.2 Technical Challenges

The main technical challenge here is reducing the memory consumption during training, this opens up access to training on high resolution images but creates another technical challenge which must be overcome, lack of global context. The second technical challenge occurs because the method used to solve the first problem is tiling. Inherently tiling suffers from lack of global context.

1.3 Motivations

Over the past 2 decades massive strides have been made in the imaging technology world. So much so that the average smart phone is able to produce ultra high resolution images with ease. Processing such data is trivial for simple tasks, however for segmentation this causes many challenges. Smart phones are the consumer end of technological advancements, when the progress in satellite, biological and medical imaging is examined the resolution capabilities are even greater. These domains all have a multitude of use cases for segmentation, meaning it is a common problem for deep learning to tackle. Even though many state of the art segmentation networks have been devised very rarely do they attempt to focus on segmentation of high or ultra high resolution images. This is becuase most benchmark dataset use images below the high resolution mark. Moreover many state of the art methods do not concern themselves with resource consumption when they are designed and as

such exacerbate the problems associated with high resolution image segmentation by using excessive resources to achieve better results.

The use of deep learning is memory intensive but usually does not prohibit the processing of high resolution images. However this is not the case with regards to training CNNs on high resolution images, naturally the video memory (VRAM) cost at training time is higher due to the large number of stored activation maps needed for back propagation. High resolution images increase this GPU memory burden to a point where most state of the art CNNs can not be trained on them. People wishing to use these high performance networks are left with 3 options, custom networks, shrinking of images (downsampling) or splitting images into smaller sub-images (tiling). Creating a custom network is out of the scope of an applied machine learning scientist and is a complex and time consuming task for a deep learning researcher. This causes most to opt for downsampling or tiling. Downsampling may work if the problem is not complex but it often results in poor accuracy due to small object classes disappearing or object boundaries being eroded. Tiling allows for high fidelity network outputs but results in edge artefacts, a negative qualitative phenomena most end users and researcher find unacceptable. It also results in class confusion where a network gives the wrong class prediction due to limited context within a tile. Both of these problems can be summed up as being caused by the lack of information surrounding the objects in the tile.

Over the past half decade the average GPU used for deep learning has not changed in regards to memory. Most cards are within the 8-12GB range and specialist cards do exist with much larger amounts of memory in the 32-48GB range but these are prohibitively expensive for many researchers from smaller labs. The 8-12GB range will limit the average segmentation network to input images of resolution 1024×1024. Even if these specialist cards were not prohibitively expensive the solution of waiting for hardware to improve is not

an active solution. This is especially true when the advancement in GPU technology is compared against the advancement in imaging capabilities. This argument is reinforced when the issue of 3D volumes is addressed, volumetric convolutions compound the memory issue even for volumes which are small. Many imaging devices are now capable of producing volumes which are spatially high resolution and volumetrically high resolution, there is simply no way to segment these easily with current techniques.

1.4 Contributions

Chapter 2 gives a background on deep learning highlighting mainly key components of networks. A literature review is given which outlines how the first segmentation CNNs were devised and consequently how these were innovated upon. Two accompanying literature reviews are given for efficient segmentation and high resolution image segmentation since this thesis is mostly concerned with this subsection of the field.

Chapter 3 outlines the two datasets contributed in this thesis, each dataset is explained and a motivation for its segmentation is given. It also aims to outline other more commonly used high resolution datasets which are used later in this thesis, lower resolution datasets are briefly outlined for context purposes.

Chapter 4 aims to compare 2D, 2.5D and 3D segmentation on the MRI colon dataset. This evaluation includes accuracy, VRAM (our most important aspect), robustness to missing data and generalisability to other images in a similar domain.

Chapter 5 outlines the how downsampling and tiling are used for high resolu-

tion image segmentation. Each method is evaluated on VRAM and accuracy. More importantly their scalability to future more high resolution images is considered. This gives the basis for chapter 6.

Chapter 6 considers the lessons from chapter 4 and chapter 5 and uses them to devise a scalable technique for high resolution image segmentation. This is where the novel crosstalk module, selective back propagation and in network tiling are introduced. The three main concepts that underpin the method advocated for in this thesis.

Chapter 7 tests the method outlined in chapter 6 by applying it to a new network and two different high resolution datasets. Finally an interesting logic puzzle dataset is presented which is used to test the inner workings of the method.

Chapter 8 concludes the thesis by outlining the key takeaways. Some suggestions for future works are given along with an honest assessment of the method where limitations are mainly highlighted.

Chapter 2

Background

This chapter introduces Deep Learning by giving a small description of how neural networks (NNs) function and an explanation of how they came about. Only the most important layers are discussed at a high level along with how they fit together to make a generic reusable architecture. The literature is then discussed begining with classification networks and how they were adapted to create semantic segmentation networks. The main semantic segmentation architectures and inventions are discussed highlighting some of their drawbacks. Since the work presented in this thesis is on efficient semantic segmentation of high resolution images the efficient semantic segmentation literature is discussed along with why it does not address the needs of high resolution segmentation. Finally the small number of works on high resolution segmentation are reviewed.

2.1 Deep Learning

2.1.1 Overview

Deep learning is currently the most popular technique in machine learning especially within the field of computer vision. Deep learning has replaced almost all classical low level computer vision techniques in all computer vision sub fields such as classification, segmentation and object tracking.

Deep learning uses convolutional neural networks (CNNs) which evolved from artificial neural networks (ANNs) invented in the early years of artificial intelligence (AI) research. CNNs are comprised of 4 main components, convolutional layers, pooling layers, activation layers and decision layers. Convolutional layers are learnable NxM filters which are multiplied across an input image. The output of a convolutional layer can be successively fed into the next allowing compound filtering to take place. The output of each convolutional layer is known as a featuremap.

Usually after a small number of convolutions a pooling layer is used, the most common pooling layer used is the max pooling layer. An NxM grid is defined and the pixel with the highest activation is selected from this grid. Much like the convolutional layer this process is carried out in a sliding window approach across the whole featuremap. Pooling layers have multiple advantages, they reduce featuremap size allowing more convolutional layers to be used with a lower strain on resources. They allow translation invariance meaning an object's features will be found regardless of where that object is in the image. Finally they allow the most important pixels highlighted by the filters to progress to the next layer.

Activation layers allow the neural network to decide which neurons should fire. In the context of a CNN this is equivalent to deciding which filters should be used based on the incoming featuremaps. The most popular type use in CNNs is ReLU [1] but others such as the Softmax, sigmoid and Tanh exist, ReLU is prefered over many other options since it avoids issues related to vanishing gradients. It also requires lower computational power to compute when compared to a softmax this is because the ReLU is an adapted linear function with a clause to set all negative values to zero. In the case of ReLU this clause causes the function to be non-linear giving CNNs non-linearity. This allows

more complicated functions to be learnt when mapping from input to output.

The final component is the decision layer, in early CNNs a fully connected or dense layer was used which is equivalent to a small ANN. More modern papers use a global pooling layer as the decision layer, global pooling is a pooling operation which pools all featuremaps to the same resolution, usually at the decision layer the resolution is 1×1. Most networks opt to use average pooling here instead of min or max pooling. Global pooling is much less computationally expensive since there are no learnable parameters, the layer also provides more translational invariance as an added benefit. Decision layers vary depending on the task, a global pooling layer would be used in the case of a classification task and the output size of this layer directly relates to the number of classes. In the case of segmentation the final layer size also relates to the number of classes but the output is usually a dense prediction where a class is predicted per pixel on a featuremap the size of the input image.

When running neural networks there are two main passes the forward and backwards pass. The forward pass treats the neurons in a feed-forward fashion where a neuron fires and potentially activates the next connected neuron to also fire, at the end of this process a decision is made. The backwards pass takes this decision and compares it to the correct output and adjusts the weights in the network to move the prediction closer to the correct output. Neurons are adjusted from the decision layer backwards. The forward pass is executed both at test time and training time whereas the backwards pass is only performed at training time.

2.1.2 Dimensionality and Convolutions

Filters for 2D and 3D data in CNNs are both learnt through kernel convolutions. The fundamental concept is the same for both but are referred to

differently. 2D convolutions are referred to as spatial convolutions and 3D convolutions are referred to as volumetric convolutions. Each convolution processes tensors which are essentially library specific implementations of multi-dimensional arrays. 2D convolutions receive a tensor with dimensions 4D in the shape $B \times C \times H \times W$ where B is the batch size, C is the number of channels, H is the tensor height (image or featuremap height) and W is the tensor width (image or featuremap width). Whilst 3D convolutions receive the same tensor with an added dimension for the depth. These tensors take the form $B \times C \times D \times H \times W$ where the added term D is depth. Volumetric convolutions also differ to their spatial counterparts in their VRAM consumption. Volumetric convolutions are notorious for consuming large amounts of GPU memory because of the incredibly large featuremaps they produce due to their 3D nature, this theme is explored more in later chapters.

2.2 Literature Review

2.2.1 Early Deep Learning and the Transition to Semantic Segmentation

Early deep learning works all focused on classification taks, LeNet [2] was used for classification of digits, this structure was made deeper and applied to ILSVRC [3] by AlexNet [4]. ILSVRC is a general classification competition which AlexNet was the number 1 competitor, a year later it was followed by ZFNet [5]. ZFNET was very close to AlexNet with some hyperparameter tweaking. ZFNet was followed by GoogLeNet [6], this network won the same competition and introduced the Inception block which is a ensemble of convolutions at different kernel sizes. This is advantageous because having multiple kernel sizes allows filters for image features of different sizes to be learnt. Within the same competition VGGNet [7] was also introduced which followed the same path as ZFNet.

All of these networks focused solely on classification and more specifically focused on the ILSVRC competition, FCN [8] took these networks and applied them to the task of segmentation for the first time in deep learning. FCN replaces the dense decision layer of the classification network with a convolutional layer with kernel 1×1, this is the first time it was used in such a way. FCN takes the output of this convolution and upsamples it significantly to begin recovering the spatial resolution of the featuremaps with the goal of creating a dense prediction approximately the size of the input image. This upsampling method takes place multiple times consecutively between which featuremaps, of the same size, from the classification network are combined with the current featuremap in order to help improve the prediction. This method of incorporating featuremaps from the classification network became known as skip connections or a skip architecture. Later on in the literature the classification network became known as the encoder and this resolution restoration process became known as the decoder.

Post FCN most segmentation networks use the newest classification network as the encoder, usually the decision layer is removed in order to preserve spatial resolution. In most cases the classification network is pre-trained meaning the network has already been trained on another task such as ImageNet and those weights are preserved until the segmentation training takes place. It is generally accepted that doing so increases the final accuracy of the network in most tasks because it allows the network to begin learning the new task in a favourable state. The state of the network is seen as favourable since the encoder has already been trained to filter generic features found in many real world objects because it has been trained on a dataset of 14M images with 1000 classes. If not for pre-training the encoder would begin training with it's weights randomly set. When beginning in this state, early training iterations are wasted on learning to filter generic features in images.

2.2.2 Residual Learning

The paper "Deep Residual Learning for Image Recognition" [9] introduced ResNets a class of networks which use residual learning to train allowing deeper networks to be trained. This paper and set of networks is given a special mention given that it is the most commonly used classification network and the blocks introduced are used in most segmentation networks.

The residual block is a set of convolutional layers with an identity mapping. More specifically there is an identity mapping from the input featuremaps to the output featuremaps of the final convolutional layer. They are combined using an elementwise addition which is an operation which combines featuremaps of the same size by simply adding them. Corresponding channels from each set of featuremaps are combined. This identity mapping is refered to as a shortcut connection. This shortcut connection allows the network to learn at least the identity mapping and at most the residual, meaning the left over features not yet learnt. Residual networks have been studied extensively and been found to avoid the vanishing gradient problem. The property of residual learning has also been identified by examining which paths during training are taken by gradient flow which resulted in the discovery that they act as an ensemble of networks [10].

For ease the two residual block defined in the original paper are defined here. The resblock also know as the basic block is two convolutions of size 3×3 with the same number of channels, a ReLU is also added post elementwise-addition. The bottleneck block is 3 convolutions, a 1×1 used to quarter the number of channels, a 3×3 at that number of channels primarily used for learning, then finally a 1×1 to restore the number of channels. Again a ReLU is added post elementwise addition. Bottleneck blocks were used to create deeper networks which are prohibited by resources when using the

basic block. Each of these blocks now also include batch normalisation layers. Batch normalisation is another common feature of modern CNNs and is used to normalise the featuremaps produced by convolutional layers which allows for faster training and network convergence.

2.2.3 General Segmentation Literature

After FCN many segmentation networks were invented in quick succession, the first of which was SegNet [11]. SegNet is very similar to FCN in structure but expands the size of the decoder and adds more convolutional layers to the decoder. Instead of using skip connections to copy entire featuremaps from the encoder to the decoder, SegNet uses the max pooling indices. More specifically it remembers the indices used to select the max values in the featuremap and copies these across to the appropriate location. The featuremap is then padded with zeros in the locations of the pixels which were not previously selected, finally this featuremap is fed to a convolutional layer. Copying the pixels in this way allows for the most important pixels to be remembered and is less computationally expensive since most values are not copied across. Arguably the most popular network after FCN is U-Net [12]. U-Net adapts FCN by matching the decoder architecture to the encoder architecture, each convolutional layer and pooling layer pair in the encoder is matched by a convolutional layer and upsampling layer in the decoder. Aside from this, U-Net also expands the general size of the decoder found in FCN.

DeepLab V1 and V2 [13] popularised the use of conditional random fields (CRFs) [14], atrous convolutions and atrous spatial pyramid pooling (ASPP) within the domain of segmentation. Fully connected CRFs were introduced in V1 to refine the output segmentation map of the network whilst atrous convolutions were used to increase the general receptive field size of the neurons in the decoder. Atrous convolutions allow a larger receptive field size with no

extra computational cost by leaving gaps between pixels in the convolutional kernel, this allows pixel information from further afield to be included. The ASPP block includes multiple atrous convolutions at different dilation sizes (the size of the gap between pixels in the kernel), this allows the encoder to process information at different scales in a single block.

PSPNet [15] brought the concept of pyramid pooling from the classification literature to segmentation. Pyramid pooling is identical to ASPP except it replaces atrous convolutions with global pooling at different kernel sizes for multi resolution context. Both PSPNet and DeepLab use minimal decoder architectures when compared to many other networks, usually the decoder consists of an upsampling rate of $\times 2$ followed by a convolutional block. Both PSPNet and DeepLab opt to upsample directly to the desired resolution post pyramid pooling or ASPP block.

Networks such as RefineNet [16] use multi path networks which take inputs at multiple resolutions to achieve a similar effect to the ASPP or pyramid pooling block. Lower resolution more global sections of the network are processed first using a very downsampled input image the prediction of these paths are refined consecutively by combining these outputs with a high resolution section of the network. This allows for a similar refinement process to that of a decoder which upsamples featuremaps and slowly filters them. If the inputs to each path of the network are significantly smaller than the original image this network can also be used as an efficient segmentation technique. Many networks take this approach to achieve efficient segmentation and are discussed in section 2.2.4.

Fully convolutional DenseNets [17] incorporate the DenseBlock [17] to replace normal convolutional layers or ResBlocks [9]. Although computationally expensive DenseBlocks achieve better accuracy than the popular Resblock by

building on the short cut connection concept using a concatenation operation instead of a elementwise addition. Elementwise addition is defined in subsection subsection:Resnetsection however concatenation is where two sets of featuremaps are stacked such that all featuremaps from both are kept in the output and remain unchanged. As an example if two featuremaps with the same number of channels are concatenated the resulting featuremap has doubled the number of channels. Fully convolutional DenseNets also used the deconvolution layer [18] which is also refered to as the tranposed convolution. This layer is only used in the decoder and allows the network to learn a filter and upsampling operation in one combined unit.

Papers such as "Large Kernels Matters" [19] support the use of larger kernels in the convolution and the concept of maximising the receptive field size of neurons just as DeepLab does. Most networks use 1×1 or 3×3 convolutions whereas this paper argues for larger kernels. It also introduces a boundary refinement block which it shows has a qualitative and quantitative effect on results.

2.2.4 Efficient Semantic Segmentation

Most efficient semantic segmentation techniques draw inspiration from the classification literature and often only apply to the encoder of the network. Moreover, many of these techniques do not address any issues relating to VRAM efficiency. Most works focus solely on floating point operations (FLOPs) or total number of network parameters. The most popular network in this domain is MobileNets [20] which introduced two new hyper parameters and a new convolutional unit called the depthwise separable convolution. This unit factorises normal convolutions into two constituent parts the depth wise convolution and pointwise convolution which results in a lower computational cost. The two hyperparameters introduced here are multipliers applied to the width of the

convolutions (number of filters in each layer) and a multiplier for the size of the input image. Although these two concepts were not new at the time this paper investigated the effect of reducing these multipliers on the number of parameters, computational cost and the accuracy of the network. The paper finds that reducing the width and resolution increases efficiency but largely decreases accuracy, the paper does not mention the decrease of VRAM here but these multipliers also decrease VRAM usage. VRAM is the local memory available on the video card or graphics processing unit (GPU), VRAM is used to store featuremaps both at inference and training time and can be a limited resource especially on embedded devices. In this case VRAM is reduced through multipliers because the size of the featuremaps stored is reduced.

ESPNet [21] continues on from mobilenets by applying the depthwise separable convolution technique to the spatial pyramid pooling block found in PSPNet, the number of network parameters is reduced, the processing speed and FPS are increased and finally the size and power consumption of the network are reduced. Although these are all significant benefits the total VRAM used is not targeted and likely not reduced by a significant margin.
Deep Compression [22] opts for a different route to efficiency by combining network pruning, quantization and huffman encoding to reduce network size on disk, number of parameters and computational cost. Although this technique was very successful it comes with several drawbacks. It requires the network to be retrained post pruning and quantization, it does not inherently change the network architecture therefore has very limited scope for future efficiency improvements. This method introduces an added bonus of decreasing VRAM at inference time but the initial training time VRAM is identical. Channel pruning [23] can further increase the efficiency of an encoder but again does not have the benefit of training time VRAM reduction.

Following on from RefineNet many works sought to achieve accurate real time

segmentation via more efficient segmentation networks. A lightweight RefineNet [24] was created by substituting layers with more parameter and FLOP efficient alternatives, in doing so they achieved a more computational efficient faster network. Although no assessment on VRAM usage was carried out it is certainly the case that VRAM consumption would be lower than SOTA counterparts such as PSPNET because the decoder is shallower and the images are processed at 1/4 the resolution and lower. Much like RefineNet, ICNet [25] uses the same input image at multiple lower resolutions where the lower resolution paths are deeper with more convolutions than the high resolution path. The network achieved an image processing speed up whilst reducing the VRAM used at both training and inference time in exchange for a reduction in accuracy. Similarly BiSeNet [26] utilises a context path and a spatial path where the spatial path processes the image at full resolution and the context path processes the image at a vastly downsampled rate. Again this network achieves a lower number of FLOPs but does not investigate the VRAM consumption at training time.

DSNet [27] focused again on road scene parsing and outperformed many of the aformentioned next works. It uses a network which adapts the Denseblock and trains on images at half resolution to improve segmentation speed. Again this network mostly focuses on real-time segmentation and does not achieve accuracies comparable to SOTA methods. The paper does not consider or address memory at training time or inference.

2.2.5 High Resolution Semantic Segmentation

High resolution images are defined in the literature as approximately 2048×1080 whilst ultra high resolution are defined as approximately 3840×2160 which are 2K and 4K respectively. Methods seldom explore segmenting images at these resolutions and usually opt for cropping, downsampling or tiling how-

ever some methods have attempted to solve this issue. Section 2.2.3 highlights many SOTA or previously SOTA methods all of which opt for image cropping due to their incredibly resource intensive architectures. Most methods in this domain focus on reducing VRAM at training time as this is the largest bottleneck to overcome in high/ultra high resolution semantic segmentation. Some works have shown that many classes in segmentation are segmented better when a resolution above native resolution is used. Particularly in situation where some classes only occupy a very small number of pixels in the image [28]. Although the network for this paper uses excessive hardware it supports the benefits of high resolution image segmentation and highlights the difference in results based on inference at different scales. Again it does not address the issues associated with training at these higher image resolutions.

Gradient checkpointing [29] is a training scheme that reduces the number of simultaneously stored activations by replacing the backwards pass with multiple forward passes. Effectively it allows GPU memory saving in return for computational efficiency, these savings allow researchers to train SOTA networks on high and ultra high resolution images. However the training time increase caused by this method can be prohibitive. The method also has the drawback of being difficult to apply to networks due to the complex set of rules around how the libraries often interact with specialist layers such as batch noramlisation and dropout. Efficient ladder-style DenseNets [30] were introduced by combining this gradient checkpointing with DenseNets.

GLNet [31] follows on from many efficient segmentation papers and uses a global context path and local context path. The global context path receives a downsampled input image whereas the local context path receives a random patch of the input image at full resolution. Branches share featuremaps and use a fusion layer to aggregate both contexts much like most multi path architectures. GLNet shows a reduction in VRAM on multiple datasets and shows that global and local context aggregated give better accuracy. Finally this

method showed it was possible to segment ultra high resolution images during training time. Patch proposal network (PPN) [32] follows on from GLNet by using a multi path approach where the local context path uses a proposal system to prioritise the most important patches in high resolution images. Prioritised patches are usually those which have many of the classes that are the segmentation targets whilst the left over patches usually consist of mostly unimportant background pixels.

MagNet [33] was recently introduced for progressive image refinement, however this method focuses on high resolution datasets. The authors produce results for datasets used by both PPN and GLNet and show that their method outcompetes these. MagNet trains a segmentation network on progressively less coarse image patches to refine the predictions, it focuses specifically on improving points with uncertain predictions. A refinement module is used to intergrate the output maps at the two levels of coarseness.

2.3 Summary

Deep learning provides the solution to many problems including but not limited to classification, object detection and location and segmentation. Much of the early segmentation literature directly adapted many early classification networks, this theme continues throughout the literature where the classification networks have become the first half of the common encoder-decoder network. Many modules designed to improve classification accuracy such as Resblocks or Denseblocks have been adopted for the same purposes in segmentation.

Although the segmentation literature is vast, much of it exists to address problems found in benchmark datasets. Nearly all of these datasets are low resolution. Since the literature focuses on these benchmark datasets, networks

produced are often very resource intensive since accuracy is prioritised. Many efficient segmentation methods exist but nearly all focus on reducing FLOPs, network parameters or size of disk. Although these are useful metrics that allow networks to be used on a wider range of hardware, VRAM is ignored despite its nature as a constrained resource during training. Some networks within this literature do address VRAM but only at inference time and mostly as a proxy for achieving real-time segmentation.

The literature which does address high resolution image segmentation is very limited and most methods use very similar network structures. Many of these networks are custom meaning they cannot leverage the accurate design patterns found in other segmentation networks. Many improvements can be made upon pre-existing networks and much can be done to expand the number of novel ideas in this area.

The next chapter contributes a volumetric dataset and a ultra-high resolution dataset. The chapter introduces them, giving motivations for why deep learning is needed to tackle their segmentation problems. The chapter also introduces many benchmarking datasets and some of the commonly used high and ultra-high resolution datasets. These exemplify the large gap between many common datasets and the size of images used in other domains. These high resolution datasets also act as a test for the method described later in the thesis.

Chapter 3

Datasets

3.1 Introduction

In this chapter the datasets for the proceeding chapters are introduced along with a description and motivating factors for the need to process such datasets. All datasets here are high resolution semantic segmentation datasets, two of which are part of the contribution of this thesis. As previously defined [31] high resolution images are at least size 2048×1080 with around 2.2M pixels. Ultra high resolution images can be defined as having a resolution of 3840×1080 or around 4.1M pixels. The datasets examined are a volumetric MRI dataset of human colons and a maize root cross section dataset acquired through laser ablation tomography (LAT). Other ultra high resolution and high resolution datasets such as cityscapes, a popular benchmark road scene dataset, are briefly introduced. Some popular datasets which are not used in the thesis are also covered here to give background and context to those which are covered.

The contribution of this chapter is to provide enough of an in-depth introductions to the datasets such that the following chapters can be read with ease. Table 3.1 contains a summary of the datasets covered in this section, it denotes which datasets fall into the categories of high resolution and ultra

high resolution. It also gives some basic information about image dimensions, subsequent pixel count and number of images included in the dataset.

Dataset	Average Resolution	Average Pixel Count	Images	HR Images	UHR Images
LAT	3072×3072	9.4M	96	✓	✓
MRI Colon	$256 \times 256 \times 24$	1.6M	930	†	†
DeepGlobe	2448×2448	6M	803	✓	✓
Inria	5000×5000	25M	180	✓	✓
Cityscapes	2048×1024	2M	25,000	✓	X
PASCAL VOC	500×500	0.25M	2,913	X	X
ADE20k	308×200	0.06	20,210	X	X

Table 3.1: Popular, high resolution and ultra high resolution semantic segmentation datasets. HR denotes datasets which on average contain high resolution images. UHR denotes datasets which on average contain ultra high resolution images. The † symbol indicates the one dataset which is 3D and does not fit the category outlined categories.

3.2 Lower Resolution Semantic Segmentation Datasets

Many semantic segmentation datasets exist and over the years some have become the standard used to compare the performance of CNNs. Datasets exist for nearly every domain of study from biological and biomedical to scene parsing. General scene parsing are seen as the most generalised type and as such are usually used as the general standard to compare. The most common three comparison datasets are within the scene parsing domain, these are, Cityscpaes [34], PASCAL-VOC 2012 [35] and ADE20k [36]. Note that many datasets exist which are not covered here for brevity. Important datasets not used are covered below:

PASCAL VOC 2012 is perhaps the most commonly used dataset in the literature, it consists of 2913 images of various sizes with an average resolution of around 500×500 and also includes 20 classes. The dataset focuses on segmenting foreground objects from the background in various indoor and outdoor scenes. Although this dataset is useful the resolution is too low to be considered in this thesis, the resolution is around 12.5% the size of a high

resolution dataset.

ADE20k is another lower resolution dataset which includes a variety of indoor and outdoor scenes including 150 different classes. It is a scene parsing dataset with around 20,000 images. Again this a good dataset with obvious real world use, just as with PASCAL VOC the images are too low resolution to provide useful studies within this thesis.

3.3 High Resolution Semantic Segmentation Datasets

Cityscapes is a road scene dataset with 30 classes annotated, 19 of these are used for evaluation and comparison purposes. The dataset captures various road scenes from German cities using a dash cam. Cityscapes is the most popular high resolution dataset and includes 20,000 coarsely annotated images and 5000 finely annotated ones. Despite falling well below the size of the ultra high resolution datasets most methods seeking to compete on this dataset do not attempt to segment the whole image. Figure 3.1 (A) displays an example of the input image from the Cityscapes dataset.Figure 3.1(B) shows corresponding annotation mask. Both are taken from the training set.

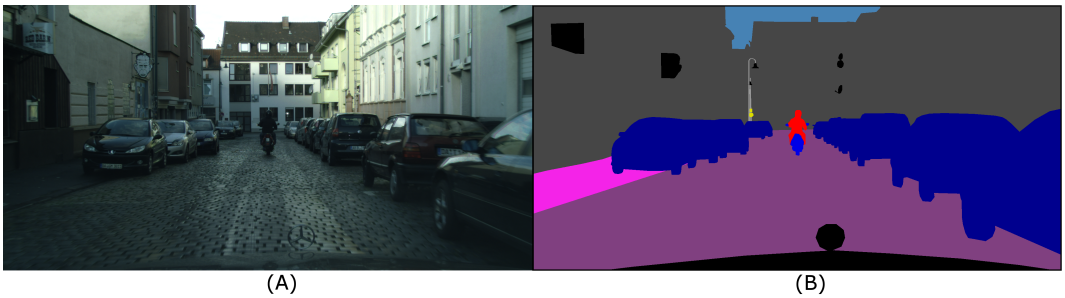


Figure 3.1: *(A) displays an example input image from the cityscapes training set. This image is taken in the city of Darmstadt, the ground truth annotation is shown in (B).*

DeepGlobe [37] is a satellite image dataset where the challenge is to segment the imagery into land cover classes. There are 7 classes which are, urban,

agriculture, rangeland, forest, water, barren, and unknown. The unknown class is largely ignored and is not evaluated during the official competition. The dataset consists of 803 images all of resolution 2448 × 2448 making it a high resolution dataset. Most papers in the literature [31–33] avoid using the official evaluation dataset and opt to split these 803 images into 455 training images, 206 validation images and 142 testing images. An example of an input image, ground truth mask pair can be found in figure 3.2.

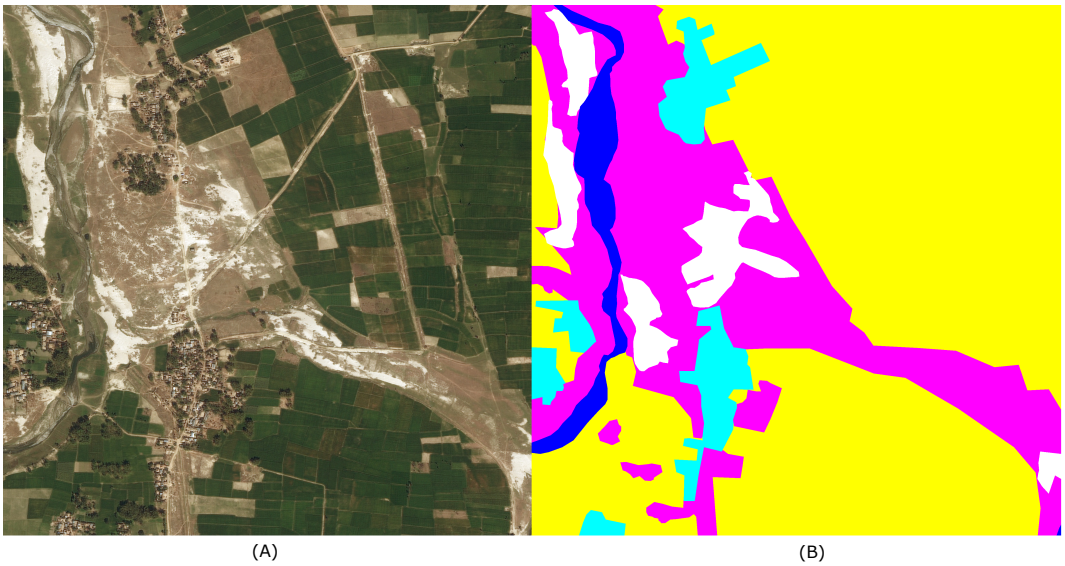


Figure 3.2: *An example input image from the DeepGlobe dataset is shown on the left (A) and on the right (B) the ground truth mask with up to 7 classes categorising land coverage.*

Inria [38] is a ultra high resolution dataset with images of resolution 5000 × 5000. This dataset was released as part of a challenge to segment buildings from the rest of the landscape and is therefore a binary classification dataset. There dataset includes 180 image of uniform size. Although it is not a popular dataset it is one of the largest ultra high resolution datasets in existence and is useful for testing the limits of deep learning methods seeking to segment this class of images. One example of these ultra high resolution images is shown in figure 3.3.



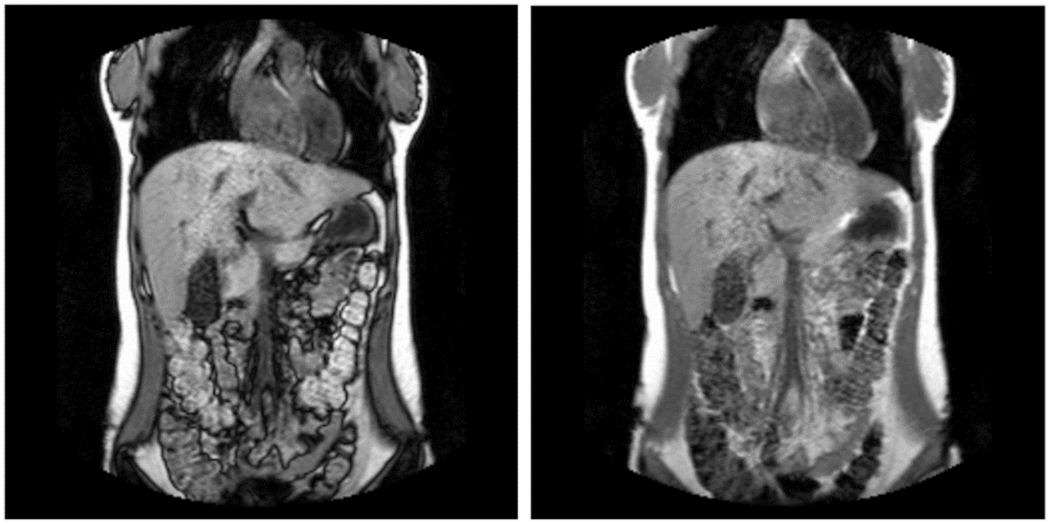
Figure 3.3: *An example input image from the Inria dataset is shown on the left (A) and on the right (B) is the ground truth mask. The dataset consist of satelitte images with the goal of segmenting building coverage.*

3.4 MRI Colon Dataset

3.4.1 Motivation

MRI is as a powerful tool for investigating GI physiology, it allows several variables of interest to be measured in a single study. The method can be utilized longitudinally throughout a day (e.g. to follow a meal) or on different days using multiple scans. This is because it involves no ionising radiation unlike other techniques. MRI assessments of colonic contents, transit, secretions and motility, provide novel insights into the pathophysiology of constipation [39], diarrhoea [40] and the symptoms associated with irritable bowel syndrome [41].

The effects of drugs and laxatives on opioid-induced constipation [42] have been studied through the measurement colonic volume using MRI. The effects of different dietary foods [43, 44] have been measured using similar means. Importantly the assessment of colonic volume is fundamental to interpreting other physiological measures including motor responses [39, 41]. These studies indicate that MRI has the potential to be a powerful tool in the clinical management of functional GI disorders. However the time-consuming nature



(a) Echo-1 (b) Echo-2

Figure 3.4: *Examples of echo-1 and echo-2 contrasts in a single slice from the Philips Achieva dataset.*

of the image analysis required, particularly for defining the colonic anatomy and measuring the volume of colonic contents is prohibitive.

Manual or semi automatic segmentation of large image datasets is time-consuming, and prone to uncertainty due to variability between observers [45]. Segmentation is particularly difficult for the colon since it extends over many imaging slices and varies considerably in position, orientation and size between people and throughout the day. The process is made even slower because the need for 3D visualization since the 3D context is an important aid to segmentation. Moreover changes in the composition of colon contents can lead to variations in MRI contrast and motility can lead to changes in image intensity and shape during a scan.

Complex manual or semi-automated data analysis limits the use of these techniques in research studies and prevents use in clinic settings where image analysis costs are factored into health economics assessments. Automating the analysis of colonic volume measurements will reduce the time and expertise needed for analyses, and reduce subjectivity and variability in the results. A

method of manual segmentation of regions of interest (ROI) around the colon along with unsupervised classification to refine said segmentation have been proposed [46]. Although interobserver variability is reduced the process is still time consuming taking around 20 minutes for an annotator to complete a single MRI. This time cost prohibits the methods use in datasets with more patients or more longitudinal studies. Other semi-automated methods [47] requiring only a few minutes of user control can achieve accurate colon segmentation but this is still far from fully automatic and is still subject to interobserver variability.

3.4.2 The Data

The dataset consists of volumetric MRI scans of the abdomen from healthy volunteers and patients with diarrhoea prominent irritable bowel syndrome (IBS). Each imaging volume is composed of 24 slices each of 256 x 256 pixels. The datasets were acquired using a dual-echo fast field echo sequence resulting in 2 modalities per volume which display different contrasts. For the networks these modalities are viewed as image channels. These are referred to as "echo-1" and "echo-2" see figure 3.4. The main dataset used for training and cross-validation was acquired on a 1.5 Tesla Philips Achieva MRI system which is referred to as the "Philips dataset" [41]. A secondary dataset, referred to as the "GE dataset", was acquired using similar scanning parameters on a 1.5 Tesla GE Medical Systems HDxt MRI scanner. This dataset is used to assess the generalisability of the trained networks to data from different hardware, a common problem in computer vision. The GE dataset is never used during training, it is part of a separate test set.

The colon regions were manually segmented as described in the original data collection study [40]. These regions were then manually segmented into the three foreground (FG) classes using Analyze9 software, the ascending colon

(AC), transverse colon (TC) and descending colon (DC). Although it is an important part of colon anatomy the sigmoid colon is not annotated in this dataset. Both echo-1 and echo-2 images were used in combination for the manual and automatic segmentation. The data were from 51 participants, providing up to 7 scans at 2 or 3 visits per subject and consisted of 930 MRI images split into 733 for training, 142 for validation and 55 for testing. This dataset split is illustrated in figure 3.5. Some participants appeared in both validation and test set, but with different scans being used for each set. Importantly if a scan appears in the training set it does not appear in either validation or test sets. The participants used in the test set do not appear in the training set, this is to avoid any potential issues of the network learning patient specific anatomy.

Figure 3.5 illustrates the datasets. No augmentation of the data is used prior to or during training. Dice score between manually segmented ground truth and network outputs are calculated and used as the comparison metric.

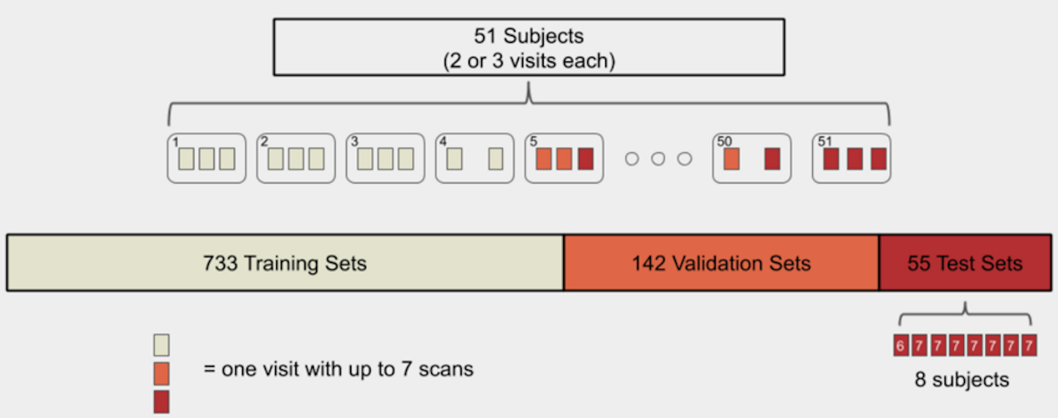


Figure 3.5: *The dataset of 51 participants all of which received up to 7 scans over 2 or 3 visits. Scans acquired during the same visit are used in the same set. The training, validation and test sets are illustrated in beige, orange and red. If scans from a participant were used as part of the training set they were not included in the test set, this is to ensure the network does not learn any participant specific ques that could help performance.*

3.4.3 Tasks

Two tasks are possible on this dataset, the first is to distinguish between background, AC, TC, and DC. The other is binary classification where the foreground is the combination of AC, DC and TC. Along with background background a whole colon (WC) metric can be calculated by the average score of each . When compared the two tasks can provide information around how much accuracy is lost through inter-colon class confusion and how much is caused by general confusion between colon and background.

3.5 Laser Ablation Tomography Root Cross Section Dataset

This is a volumetric dataset which consists of root cross sections of Maize crops acquired through the process of shovelomics [48] after which images are acquired through the use of Laser Ablation Tomography (LAT) [49] and high end DSLR cameras fitted with large macros lens. More specifically the laser used in this system is a EdgeWave PX series 355nm laser with a 10 picosecond duration. A fast speed is used in order to achieve athermal ablation which reduces the probability that samples will ignite during the ablation process. Athermal ablation is desired to allow the sample to be intact and allow accurate measurement. It also allows more accurate segmentation by reducing artefacts and bright spots caused by thermal ablation. This laser is coupled with a Sony Alpha 7R camera which allows for up to 42MP images. These components are housed inside an enclosure which includes a sample holder for the roots and a stage to move the sample holder between laser cuts, the stage allows for effective 3D imaging of a sample.

LAT is a destructive technique that uses a laser to cut an object and expose the inner structures of said object. If the object is moved and imaged be-

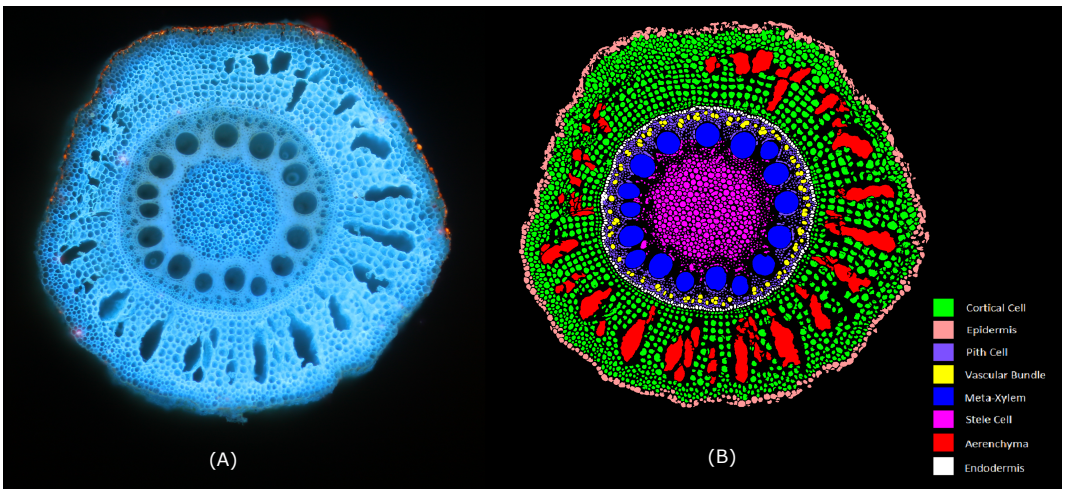


Figure 3.6: *(A) is an example Maize root cross section acquired through LAT. (B) is the accompanying annotation mask created by Penn State Plant Biologists using Cellset. A colour key for cell types (classes) is given in the bottom right of (B).*

tween movements a 3D structure can be recovered once the entire object has passed through the laser. Alternatively because most DSLR cameras include a video mode the, laser cuts and movement process can take place with the video setting instead of imaging between laser cuts. Although this method is easier and allows a faster throughput it also comes with the disadvantage of capturing visual artefacts associated with the laser ablation process. This makes segmentation harder and leaves us with the task of deciding when each slice has occurred.

Usually the side of the object with the smallest diameter is chosen to make the focusing of the camera easier however any side of an object can be chosen. The aim of this process for our dataset is to create root cross sections to recover the structure of the root as if travelling down the root system. As such the dataset does not include slices along the entire root commonly referred to as longitudinal sections.

After acquisition the dataset is annotated using a semi automated annotation tool called Cellset [50]. Cellset uses the watershed algorithm to find cells in an image. Although cellset was not designed for annotating these images

specifically the images lend themselves well to the software. This is because boundaries of a cell appear bright whilst the contents of the cell are almost entirely black, the watershed algorithm is designed to work effectively in such a scenario. Although cellset provides cell boundaries, which can be manually added, removed or adjusted, it does not provide cell type categorisation. Cellset may be accurate and provide some semi-autmated functions but it is still slow especially when compared to the potential speed of deep learning. Our dataset is annotated by the Penn State Root Department with 8 classes, I also helped to ablate 3 samples and annotate around one quarter of the dataset after receiving training from the department. The 8 classes are, Stele Cells (Purple), Pith cells (Dark Blue), Vascular Bundles (Yellow), Meta-Xylem (Light Blue), Cortical Cells (Green), Aerenchyma (Red), Epidermis (Peach) and Endodermis (White). Each image annotation gives us the central location of each cell and a mask as shown in figure 3.6(B), this section of the image also includes a colour key, in the bottom right, for the annotated cell types. Figure 3.6(A) displays the corresponding input image acquired through LAT.

Although this dataset is originally volumetric several pre processing steps are applied. Each image is cropped to remove part of the black background this reduces the images to a size of 3072×3072 and reduces the general computational cost of segmenting the images (including the time and cost of annotating images). Most importantly it reduces the VRAM requirement. A subset of images is selected which is varied thus making it a set of 2D slices and not a 3D volume. This is because segmenting a 3D volume proved too resource intensive in preliminary tests. Volumes varied from 100s to 1000s of slices in the Z domain whilst the images are 3072×3072 spatially. After this processing the dataset consists of 96 single images. In the next chapter we further elaborate on this decision by discussing the resource issues associated with volumetric segmentation and high resolution image segmentation.

3.5.1 Motivation

The extracted cell size information feeds into the larger task of plant phenotyping. Plant Phenotyping aims to measure the structural and functional properties of a plant, the field is often separated by the study of the different parts of the plant. Many consider the root to be the most important section of the plant due to its enormous impact on growth [51, 52]. There are several ways to study roots but most do not tackle how the cell traits effect root growth. Several papers exist showing the link between specific phtenotypes and growth. Large cortical cell size has been shown to improve drought tolerance in Maize [53] synergistically lower cortical cell file numbers provide the same effect [54]. Nitrogen uptake in low nitrogen soils is improved in roots with more cortical aerenchyma [55].
Segmenting the dataset effectively measures these cell traits which can then be linked to a genotype. From this genotype crops can be grown with the ability to thrive in harsher environments such as locations with deeper water tables or more nutrient deficient soils. It also allows crops to grow more effectively in more favourable conditions. This ability is especially important due to projected population increases by 2050 and the resultant projected food shortfalls [56]. Breeding crops with more favourable yields in locations previously inhospitable to certain crops will alleviate such shortfalls.

Prior to the introduction of LAT systems, root cross sections were processed using various forms of confocal microscopy [57] which allowed for the production of high resolution images. Although accurate the method is time consuming limiting it soley to the 2D domain. LATs allow for high throughput processing of root samples which can analysed manually or semi automatically using Rootscan [58]. The first method is subject to human errors but is largely more accurate than the later however it is too time consuming to be done at a large scale. Moreover when moving from singular slices to 3D it becomes impossible

given that even a small subsection of root imaged can be in excess of 1000 slices. The second method allows the semi automated processing of single root cross sections but is very error prone and does not handle variations in images. This method is based on low level computer vision techniques and as such is subject to the associated disadvantages of these.

Creating an automated method to segment these images not only allows accurate phenotypic traits to be measured it also allows the enormous volume of data being produced by the LAT to be processed. The LAT was constructed for high throughput plant phenotyping, the rate at which it produces data is far faster than a human or existing programs such as cellset or rootscan can process the image. Therefore an automated method with very fast inference is required to maximise the high throughput of the LAT.

3.5.2 Task

The task for this dataset is semantic segmentation of different cells in the root cross section images of maize. There are 8 classes in the dataset however because the epidermis can be algorithmically calculated as the outer most cortical cells in a concentric ring only 7 classes are used in the training set for the CNNs.

3.6 Summary

This chapter examined the available datasets and their characteristics in relation to resolution or pixel count. Multiple existing high or ultra high resolution datasets were presented proving that addressing the concerns around datasets of this magnitude is crucial. Most semantic segmentation datasets are of a low resolution especially the most popular such as ADE20k and Pascal VOC. Their popularity often captures the attention of much of the segmentation literature as shown in chapter 2, conversely the literature around high resolution images

and addressing their concerns is spare. This chapter combined with chapter 1 outline why these lower resolutions dataset will not be visited again in this thesis.

A ultra-high resolution LAT root cross sectional dataset and a colon MRI volume dataset are provided as contributions. Both are detailed along with their associated problems. A brief account of how computer vision or more specifically CNNs and Deep Learning can alleviate these concerns accompanies. The chapter also outlines the general motivations for segmenting each dataset. Field specific motivation and background information is provided where appropriate. These datasets are briefly revisited in later chapters when presenting experiments and technical works.

Chapter 5 reinforces what is seen in this chapter and what is outlined in chapter 1 by beginning to apply CNNs, used with low resolution segmentation datasets, to the novel LAT dataset presented here. However before this, chapter 4 explores segmenting the novel 3D colon dataset presented here. It goes on to show that many problems associated with volumetric segmentation are the same as those seen with high resolution image segmentation.

Chapter 4

Investigating the effect of data dimensionality on deep learning

4.1 Introduction

The effect of input dimensionality on deep learning accuracy is becoming clearer as the base of literature increases. Most comparisons take place in the area of biomedical imaging due to the 3D nature of the data. Many of these comparisons are indirect, through competitions such as KiTS, BRATS and many more, in these circumstances comparisons can be made indirectly by evaluating different teams' strategies where some teams opt for volumetric deep learning and others for spatial. Although these comparisons are useful in the competition setting the comparison is not totally fair due to the varying training parameters, training length, data augmentation, difference in hardware and a multitude of other factors.

This chapter aims to investigate the effect of 2D, 2.5D and 3D deep learning networks in the most fair comparison possible. To do this we use the same hardware, a single Nvidia GTX 1080 Ti and implement each network using the same version of PyTorch and Python. Most importantly all networks are limited to 6 million parameters. The accuracy and speed will be compared

along with resource consumption with a final discussion on how scalable these methods might be for larger volumes. Understanding the relationship between accuracy, speed and the various networks allows an evaluation of the weakness of current deep learning techniques, limitations caused by current hardware and the feasibility of applying similar technique to more difficult datasets of high resolutions.

4.2 Revisiting the data

In this chapter we use the MRI abdomen scans described in detail in chapter3. The scans consist of 24 slices in the coronal plane with a 256 x 256 pixel spatial resolution. Each scan consist of two contrasts which are labelled echo 1 and echo 2. Within the data the two contrasts are represented as two channels in a tensor. The task for each network is to distinguish between background, ascending colon (AC), transverse colon (TC) and descending colon (DC). A second task of discerning between background and the whole colon (WC Average) is also evaluated. The dataset consists of test sets from two scanners which simulate a real world scenario where training date is available from a single scanner and later a different scanner is used. In this scenario clinicians would like the network trained on data from the first scanner to segment images from the second, scanner without using new training data. Both scanners are tested on. The scanner the network is trained on is refered to as Philips Achieva and the second scanner, which is only tested on, is called GE Medical.

4.3 Method

In both the 2D and 2.5D network the MRI data is processed using 2D spatial convolutions which are less memory and computationally expensive than the volumetric equivalents. Using spatial convolutions allows for more features in each convolutional layer and a greater total number of convolutional layers.

The 3D network uses volumetric convolutions which are computationally expensive but more important GPU memory intensive.

The volumes are presented to the 2D and 2.5D networks as slices. To process the volumes in 2D, each volume is split into images corresponding to its constituent 24 2D slices. The 2.5D approach receives 3 images, the target image and also the posterior and anterior slices which are adjacent. Each of these include 2 channels from the echo 1 and echo 2 data, resulting in 6 greyscale channels in total. The 3D networks receives the entire volume.

4.3.1 Model Architectures

The 2.5D and 2D have almost identical networks, both use the stacked hourglass architecture [59]. The hourglass follows the literature closely in design and implementation. It consists of two main components, the encoder and decoder, this is shown in figure 4.1. The encoder contains resblocks [9] followed by max-pooling operations which downsample the featuremaps. At each level of the encoder, one resblock is used until the lowest resolution. The encoder and decoder contain 4 levels resulting in featuremaps at four different spatial resolutions.

The decoder segment of the network also contains resblocks but, conversely, each is followed by an upsampling operations. Just as in the encoder 4 are used in total due to there being 4 levels. Each slowly restores the output to the original input image's resolution. The hourglass uses skip connections to integrate encoder information to the decoder. 4 skip connections are used which help to refine the predictions at each stage of the decoder. A 1×1 convolution is used to the reduce the feature maps by half after, the featuremaps from the skip connection are integrated into the decoder. Using this convolution prevents the decoder from doubling in size.

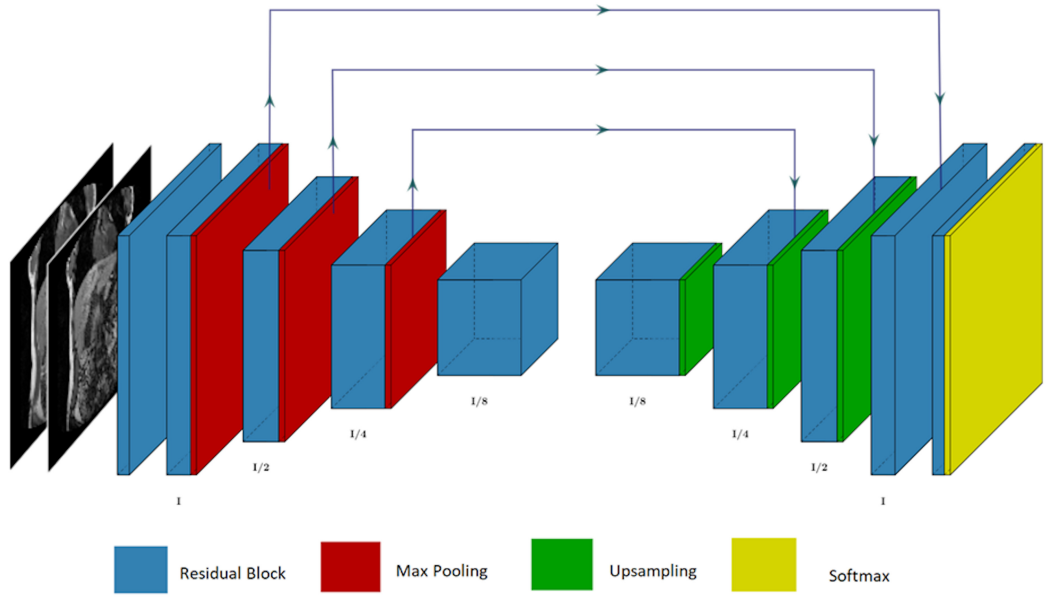


Figure 4.1: *The hourglass model used for the 2D data*

Figure 4.1 shows the 2D model architecture used in this section. Whilst figure 4.2 shows the 2.5D architecture, in this case the first convolution has an input of 6 channels unlike the 2D model which has an input of 2 channels.

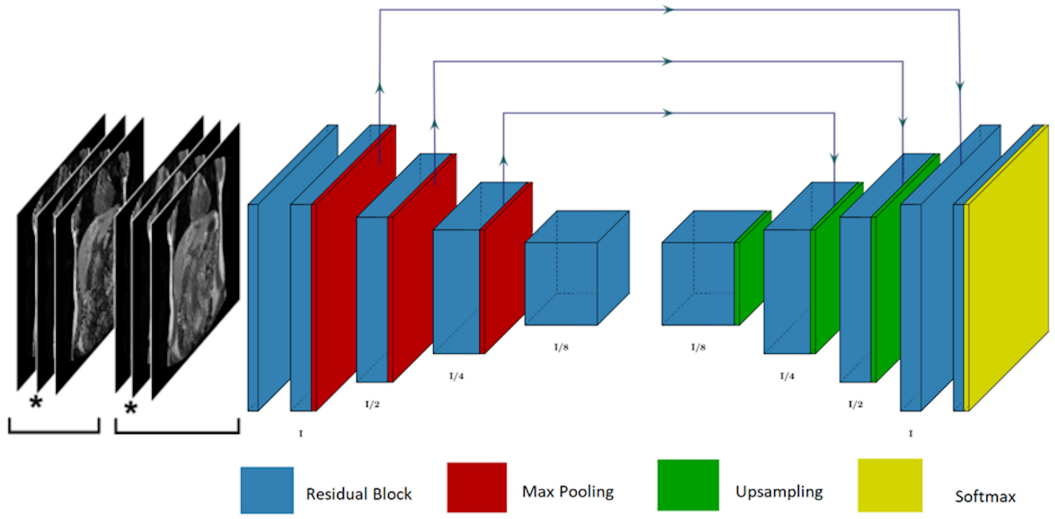


Figure 4.2: *The 2.5D hourglass architecture. The asterisk on the displayed input image indicates which slice of the input is segmented. The 2.5D network only segments the middle slice but uses surrounding slices for context*

The 3D network is also a stacked hourglass following the same encoder-decoder structure. Spatial convolutions are replaced with volumetric convolutions. Due

to memory constraints, skip connections use element-wise additions to integrated skip connection information into the decoder. These replace the 1×1 convolution and concatenation operations seen in the 2D and 2.5D networks. The 3D network is also shallower containing 3 levels in the encoder and decoder. At the lowest resolution multiple resblocks are used to maximise the network's ability to learn. More resblocks can be added at this level since the GPU memory consumed is lower due to the lower size of the featuremaps. The added resblocks do not cause the network to exceed VRAM capacity of parameter limit, which was set to enforce fairness between networks. More downsampling isn't used due to the depth dimension constraint and VRAM resource l imitations. In this volume the third dimension is $10\times$ smaller. The 3D architecture described above is shown in Figure 4.3.

All architectures are kept as similar as possible but due to novel constraints of implementing a 3D network they cannot be identical. However they are made comparable due to the introduced parameter limit.

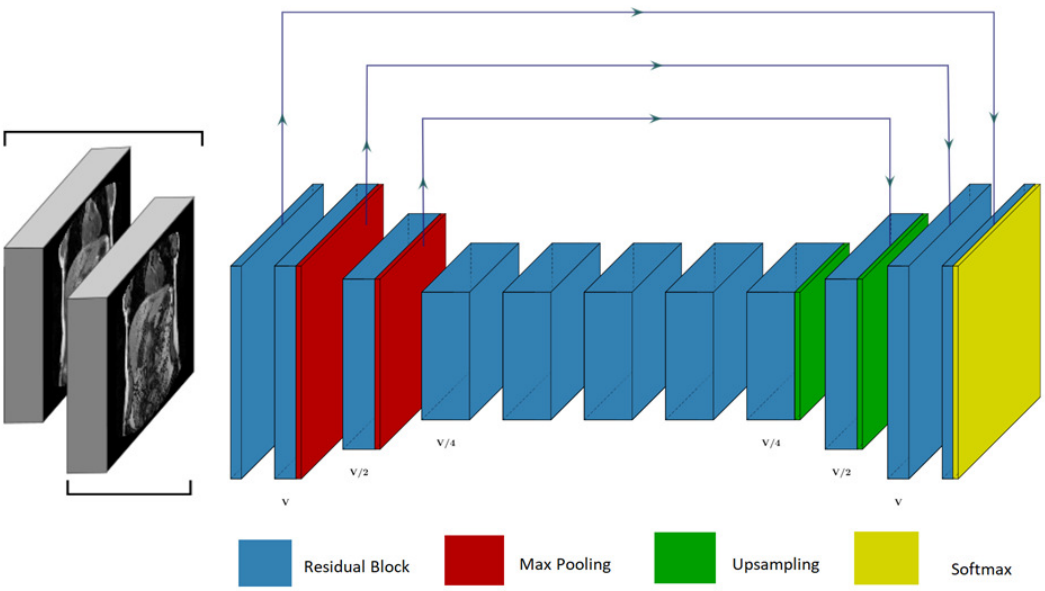


Figure 4.3: *The 3D hourglass architecture used during experiments*

4.3.2 Training

For 2D and 2.5D, the MRI datasets were split into slices, and the network was trained on a large set of slices. A mini-batch is a very small subset of the training data which is presented to the network. Presenting the network with more than a single instance of the training data allows the network to learn features across the training data more reliably. In separate training instances, mini-batch sizes were varied and a mini-batch size of six was found to offer the best performance. ADAM [60], a stochastic gradient descent algorithm variant, was used to train the network at a learning rate of 1×10^{-5}, which was selected empirically.

GPU memory limitations for the 3D models resulted in a mini-batch size of 1 along with a learning rate of 10-6. The test set was used to compute the final results for all networks which are shown in the quantitative analysis section.

4.4 Quantitative Results

4.4.1 Single Vs Dual Channel

Given the two available contrasts, separate 3D models were trained using either echo-1 or echo-2 images as input using WC ground truths. Figure **??**(a) shows the Dice score, during the training, of the single contrast modalities and the combined contrasts. When only using a single contrast, echo-2 obtained a higher accuracy than echo-1, whilst using both contrasts provided the best accuracy and trained the fastest Figure **??**(b) demonstrates that, the model using echo-2 contrast was more successful at segmenting the colon than the model using echo-l. However the highest Dice score was achieved when the 3D image volumes with different contrasts were combined. Given this result, all subsequent models are trained using dual-contrast image volumes as training data. The same rule is applied to both the 2D and 2.5D networks.

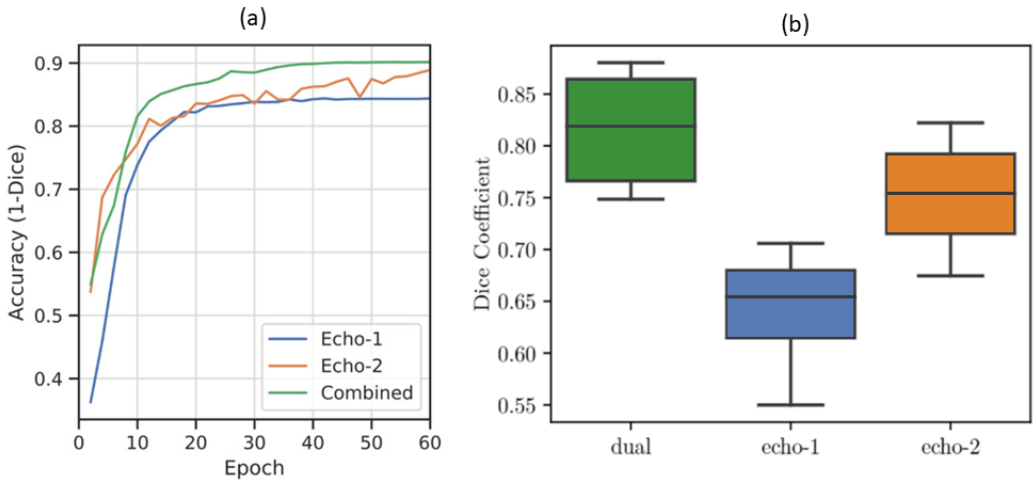


Figure 4.4: *(a) Plot showing the change in segmentation accuracy during training time for input volumes with a single contrast (Echo-1 or Echo-2), and for dual contrast (combined) inputs. (b) Box plots showing the 10th , 25th, 50th , 75th, and 90th percentile of the Dice scores achieved by 3D models trained on dual-contrast images, echo-1, and echo-2 images respectively. Results obtained on the Philips Achieva test data.*

Table 4.1 breaks down the tasks based on the segmentation target. WC refers to the segmentation of individual colon sections and is calculated as the average accuracy of the 4 classes BG, AC, DC and TC. A measure of accuracy is also provided for each section of the colon. The first row, Binary, presents the accuracy of the network when trained to perform binary segmentation into two classes, Colon or BG. The 3D network is the best performing network with an accuracy of 89.8% and a dice whole colon binary classification accuracy of 93.3%. When looking at WC average (overall segmentation) the 2D network was outperformed by the 2.5D network by 4.0% and the 3D network outperforms the 2.5D by 6.4%. For binary segmentation of FG and BG the 3D network performs the best followed by 2D and 2.5D which perform very similarly. The 3D network outperforms the 2.5D network by 6.2%, a similar margin to before and the 2.5D is outperformed by the 2D network by 1.6%. It is unclear why this is the case. The 3D network, also achieves the best results when segmenting the AC, TC and DC sections individually.

	Accuracy		
Class	**2D**	**2.5D**	**3D**
Binary	88.7	87.1	**93.3**
WC	79.4	83.4	**89.8**
AC	78.5	81.2	**89.0**
DC	66.9	74.7	**82.3**
TC	72.5	77.9	**88.2**

Table 4.1: Segmentation accuracy results, measured using Dice score, obtained on all three networks applied to test subset of the Philips dataset. WC denotes the average segmentation accuracy over all four individually segmented classes (including BG). AC denotes segmentation of the ascending colon, DC the descending colon, and TC the transverse colon. Binary segmentation refers to the task of segmenting the regions containing the colon from the BG.

4.4.2 Generalisability across datasets

The results for the GE data set are shown in Table 4.2. In general, running models on the GE data led to lower segmentation accuracies. The models trained to segment the colon as a whole (Binary) performed significantly better than the models trained to segment individual colon sections. The 3D network achieves a binary segmentation accuracy of 88.2% which is 12.6% than the 2.5D network's performance at 75.5%. In turn this network outperformed the 2D network by 0.9%, giving the 2D network an accuracy of 74.7%. The 2.5D network outperforms the 2D network on the binary task on this scanner unlike previously. Each network's relative ability to segment sections of the colon remained the same between scanners. The 2D, 2.5D and 3D networks achieved accuracies of 52.5%, 58.6% and 77.0% for WC Average segmentation respectively. The difference in contrast information appears to affect the 2D and 2.5D networks the most. BG-FG class confusion is very high amongst the networks which probably accounts for the large decrease in performance, this is seen in Figure 4.5. However, the confusion between BG and TC is very similar to the DC which was not seen with the original Philips scanner data as shown in Figure 4.6.

Class	Accuracy		
	2D	**2.5D**	**3D**
Binary	74.7	75.6	**88.2**
WC	52.5	58.6	**77.0**
AC	44.3	58.6	**78.5**
DC	32.5	55.8	**67.4**
TC	25.8	29.1	**62.7**

Table 4.2: Segmentation accuracy results, measured using Dice score, on all 3 networks on the GE dataset. WC denotes the average segmentation accuracy over all four individually segmented classes (including BG). AC denotes segmentation of the ascending colon, DC the descending colon, and TC the transverse colon. Binary segmentation refers to the task of segmenting the regions containing the colon from the BG.

Example output images in the qualitative analysis section show noisy prediction where BG pixels were predicted as colonic regions - this could be due to artefacts and change in contrast information between the scanners. Figure 4.11 (a) shows an example of a ringing artefact being wrongly classified as being part of the transverse colon. Given that a similar texture of parallel dark and light stripes also delineates the folding tissue between the colon's haustra (pouch-like structures forming the colon), it could be possible that this texture was learned by the network as a useful feature for colon segmentation.

4.4.3 Sources of error

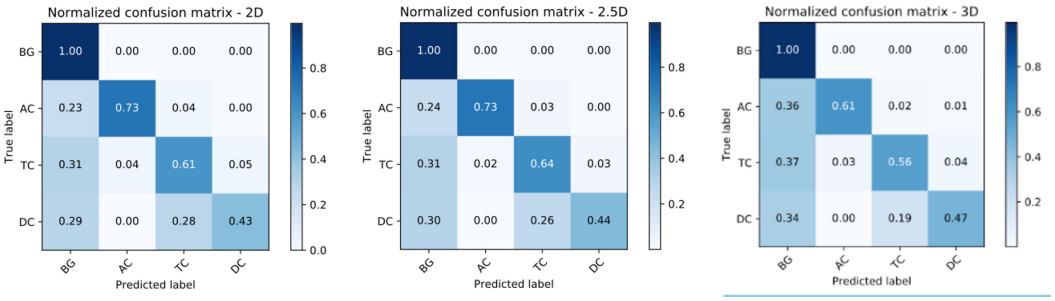


Figure 4.5: *Class confusion matrices for 2D (a), 2.5D (b) and 3D (c) for the GE Medical Systems HDxt scanner. Each has been normalized and shows the percentage of class confusion for each predicted and true label.*

The class confusion pattern between colon sections remains the same between the two scanners. Misclassification between DC and TC is the highest error es-

pecially in the GE dataset. Figures 4.7,4.8 and 4.9 show that the largest source of error is likely to be FG-BG class confusion caused by inaccurate boundaries, but there are instances of inter-class confusion which can account for individual class errors as can be see in Figure 4.7 with TC-DC class confusion. The confusion matrices in Figure 4.5 indicate that the most likely inter-colon class confusion is between the TC and DC sections. This is closely followed by misclassifying the TC as the AC and vice versa. These results are expected because the TC intersects both of the other two colon regions whereas the ascending and descending sections appear on opposite sides of the body. All networks have very few confusions between AC and DC. The remaining error comes from the misclassification of BG as FG and vice-versa. The section most commonly confused with the BG is the DC which accounts for 25% in each dataset.

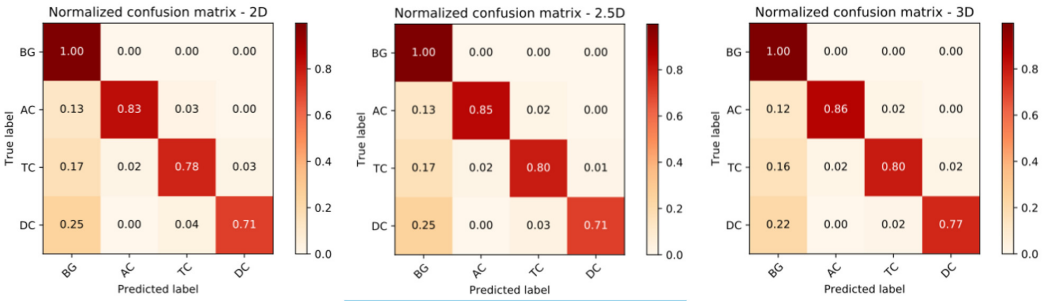


Figure 4.6: *Class confusion matrices for 2D (a), 2.5D (b) and 3D (c) Philips Achieva scanner. Each has been normalized and shows the percentage of class confusion for each predicted and true label.*

4.4.4 2.5D and 3D Data Ablation

To assess the trained models' performance and to gain insight into the inner workings of the network, ablations studies are performed. Regions of the image volumes are 'greyed out' using the mean intensity of the dataset. These ablation experiments could show to what extent the models have learnt to use 3D context information. The networks were not trained on these ablated sequences, these sequences are only seen at test time. The aim is to examine

how effective the networks are at filling in missing data based on surrounding context in the volume. As a result the 2D model is excluded from this ablation study because it does not use any depth context. Data used in the experiments were taken from the test set to provide a fair evaluation. First, the middle slice in every triplet in the test set is ablated for the 2.5D network.

The 3D network received multiple instances of the same volume where only one slice is ablated, but the slice which was ablated is shifted a single place between instances. Overall the same slices which were ablated in the 2.5D network input were also ablated here. In a separate experiment, the 3D network received a single instance of each test set volume where every other slice is ablated. To ensure there was no bias, the experiment is repeated with an offset of 0 and 1 so that all slices in a volume are ablated at least once. The two 3D experiments are referred to as single ablation and multi-ablation. The score for each volume in the test set is calculated as the average dice score on the ablated image in the volumes. In the multi-ablation experiments the results are averaged across the two offsets.

Table 4.3 shows the results of ablating images on the test set acquired from the Philips Achieva scanner. The 3D network outperformed the 2.5D network during single ablation by a significant Dice margin of 25.8% when carrying out binary classification. On the task of individual colon section segmentation, the 2.5D network is outperformed by 44.0%. Overall, the accuracies obtained by the 2.5D network indicate that there was not a large amount of context information being used to segment the slices, and the 2.5D network appears to heavily rely on the main slice being segmented. The 3D network can use context from further afield in the volume, so potentially more context information is available to help fill in the ablated data.

For the task of segmenting individual colon sections a minimal difference is

found between the multi-ablation and single ablation experiments. Overall single ablation only yielded an accuracy 0.2% higher for WC Average labelling, however on the task of binary segmentation it achieved a score which was 1.3% lower than its multi-ablation counterpart. Single ablation achieved a higher accuracy for all colon sections except the descending colon. It is not clear why this occurred it is expected that the single ablation results to be higher for all sections of the colon. However, the difference is only small. Comparing the performance of the network 3D network with and without ablation in Tables 4.1 and 4.3 shows that, ablation only decreased the WC average accuracy of the 3D network by 6.9%, whereas for the task of binary segmentation ablation decreases the dice score by 6.4%.

	Accuracy		
Class	**2.5D ablation**	**3D Single-slice ablation**	**3D multi-slice ablation**
Binary	59.8	85.6	**86.9**
WC Average	38.9	**82.9**	82.7
AC	26.6	**82.7**	81.0
DC	15.5	69.6	**70.3**
TC	14.8	79.7	**80.2**

Table 4.3: Results obtained for 2.5D and 3D networks on ablated test images from the Philips Achieva scanner

4.4.5 Inference Time

The run time of the networks are compared, the forward pass is measured which is the equivalent to having the network give segmentation predictions. The fastest network at inference time was the 2D network, followed by 2.5D and lastly 3D. Table 4.4 shows that the same trend was found when the data loading is included in the inference time. Although the trend is the same it is much less pronounced. Distinguishing the effect of data load is important, typically for the 2D network only the current batch is loaded into memory. Since the number of slices in a volume is larger than the batch size, processing the whole volume requires several loading sequences. This gives the 2D and 2.5D network a speed disadvantage.

Timing Method	2D	2.5D	3D
Including Data Load(s)	0.205	0.225	0.253
Excluding Data Load(s)	0.095	0.106	0.216

Table 4.4: Time taken for each network to make predictions on a single volume from a Philips Achieva scanner

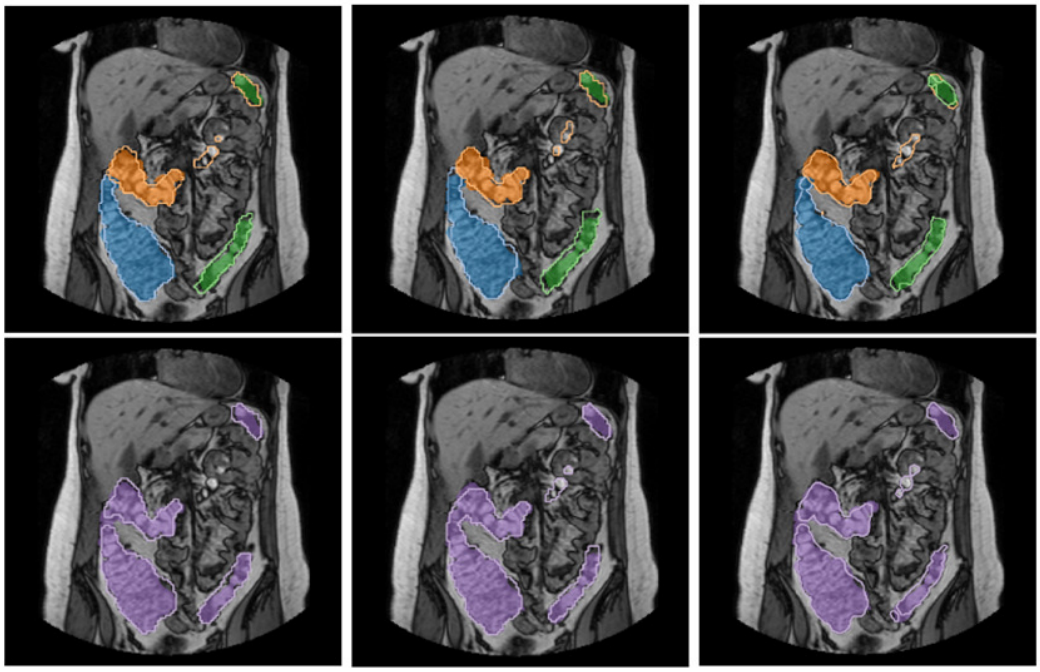
4.5 Qualitative Analysis

Quantitative metrics provided in the section proceeding this one, provide a useful method for comparing competing architectures, but give few insights into the semantic strengths and weaknesses of a network. In the following sub sections, visual results are shown from each network and common issues that arise from the network predictions are discussed. In each Figure, the contour indicates the network prediction, whilst the solid overlay represents the ground truth. Only a single channel of the dual echo input is shown. For each input image, The corresponding network prediction for binary as well as individual colon sections is shown The binary and individual colon segmentations are separated in the figures given where binary is in the row below individual colon segmentations.

4.5.1 Phillips Achieva Dataset

Figure 4.7 displays examples from a well-segmented case, the average Dice score achieved across the three networks for this image was 90%. In such cases, the networks were able to produce segmentations with only a small amount of class confusion and FG-BG confusion. The contours were relatively smooth and most colonic regions were captured.

In Figure 4.8 we present network outputs on a volume where all networks perform closed to the mean. The average Dice score for this volume was 83.6%. In these cases large contoured regions of FG-BG misclassification were common. Class confusion between colon segments was also common. Well defined regions with high contrast which fall between the AC and DC were

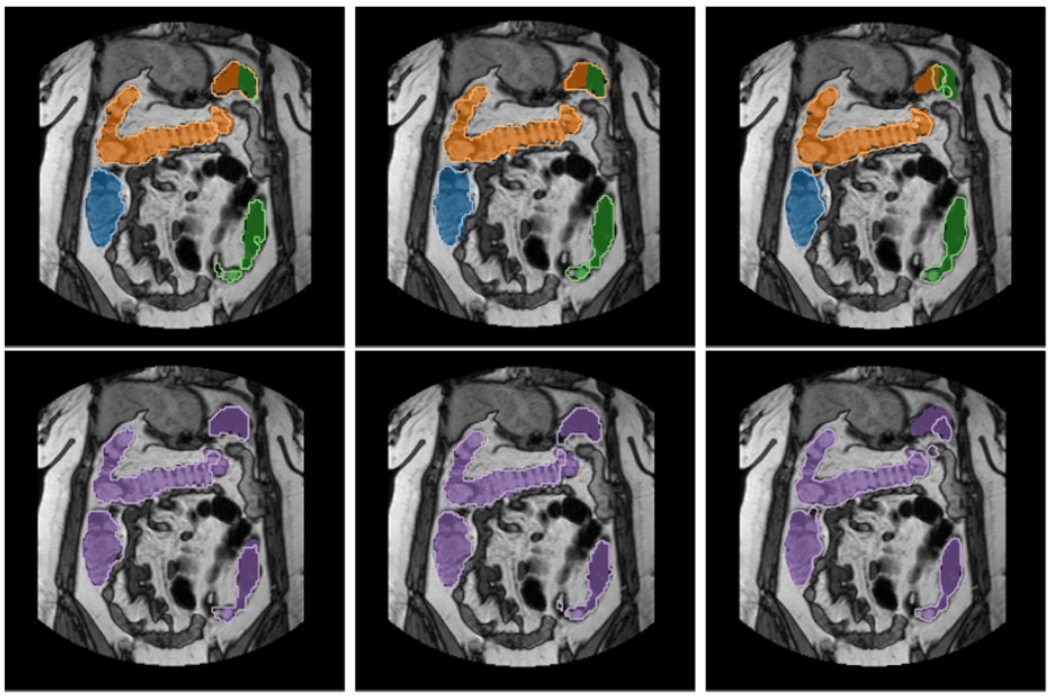


(a) 2D (b) 2.5D (c) 3D

Figure 4.7: *Examples of network predictions on a volume with high accuracy. Coloured regions denote the ground truth with the blue, orange and green corresponding to AC, TC and DC respectively. Slices with purple labels are the corresponding binary segmentations. The network outputs are delineated by the thin contour lines.*

often misclassified as TC.

Finally, in Figure 4.9 low accuracy network predictions are provided. The contours of each network are less smooth, with significant regions of the colon being missed altogether. Contours often overlapped with BG pixels or exist as a sub-region of the ground truth. In these more difficult cases, the networks has differing disadvantages. The 3D network often makes predictions around regions which do not show the contrast expected in the colon, as seen in Figure 4.9(c). Upon further investigation these regions often contained colon further along the volume or in preceding slices. It appears that the 3D network attempts to segment colon regions it expects to appear in certain positions. In the more difficult cases, the 2D and 2.5D network suffered similar disadvantages. Both networks missed many colon regions which present in each slice whilst the 2.5D network seemed to be more prone to misclassification upon inspection of the images.



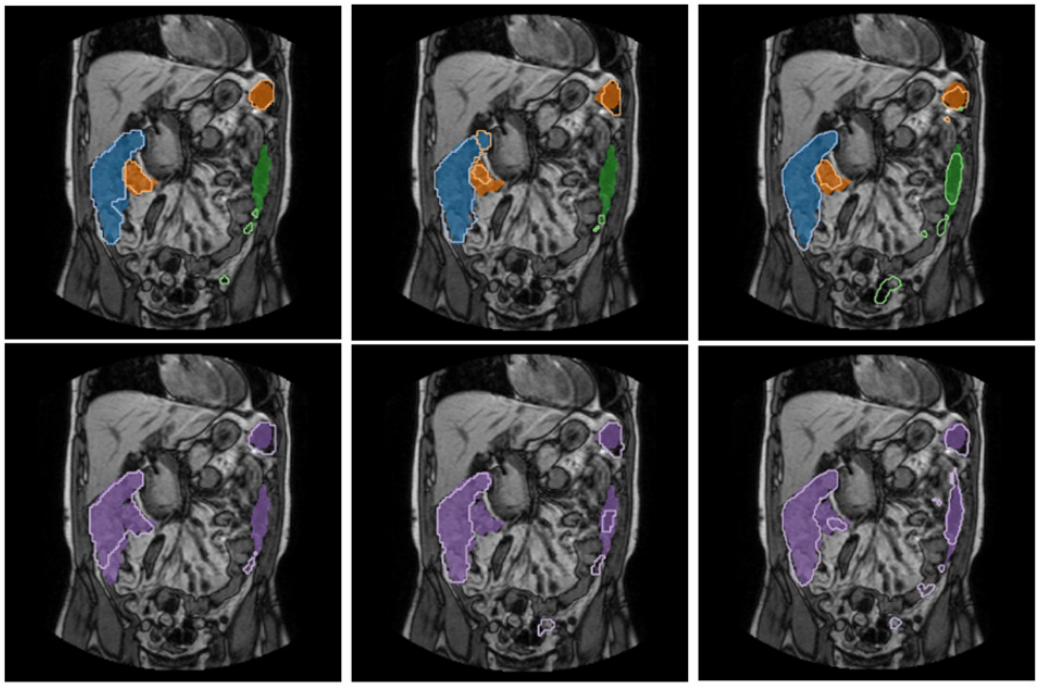
(a) 2D (b) 2.5D (c) 3D

Figure 4.8: *Examples of network predictions on a volume with only average accuracy. Coloured regions denote the ground truth with the blue, orange and green corresponding to AC, TC and DC respectively. Slices with purple labels are the corresponding binary segmentations. The network outputs are delineated by the thin contour lines.*

The colon region which was most often misclassified as the BG was the descending colon. Below the descending colon exists the sigmoid colon which has very similar contrast information to the descending colon, but the sigmoid colon was not defined in the ground truth manual segmentations, and the endpoints of each section are ill-defined. Since the sigmoid colon was treated as BG we hypothesise that the networks struggled to differentiate between the two regions causing a BG misclassification.

4.5.2 GE Dataset

Since this dataset did not form part of the training set, and transfer learning was not used, we expected results here to be worse than those above. Overall results look acceptable, but class confusion between the colon regions is high

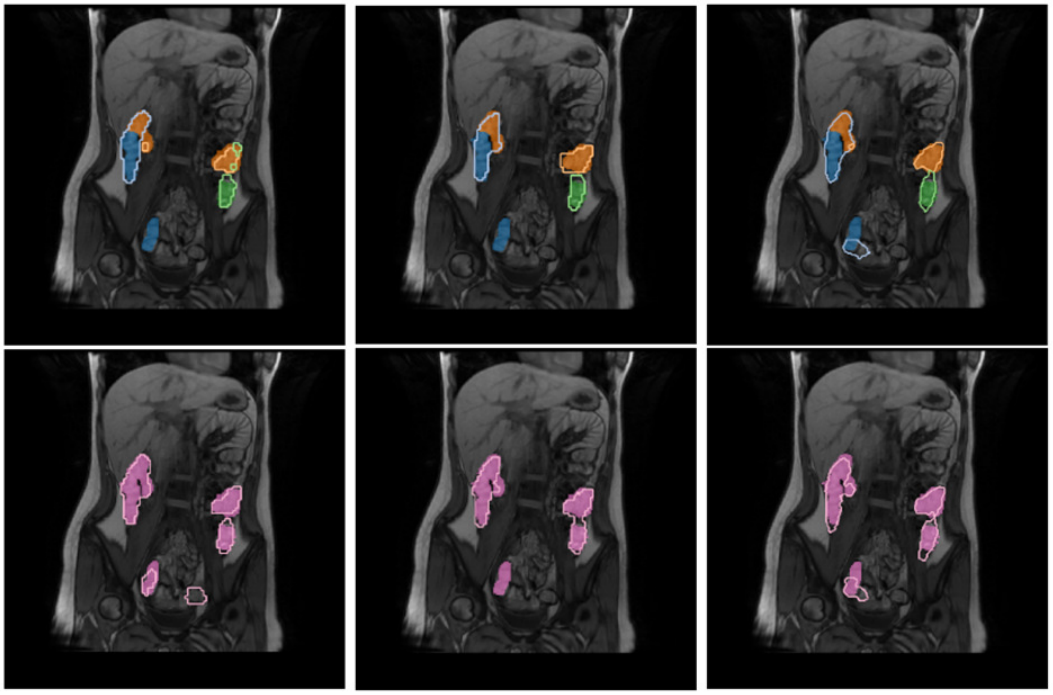


(a) 2D (b) 2.5D (c) 3D

Figure 4.9: *Examples of network predictions on a volume with a low accuracy. Coloured regions denote the ground truth with the blue, orange and green corresponding to AC, TC and DC respectively. Slices with purple labels are the corresponding binary segmentations. The network outputs are delineated by the thin contour lines..*

and most notably FG-BG class confusion is common, as shown in row 2 of Figure 4.10

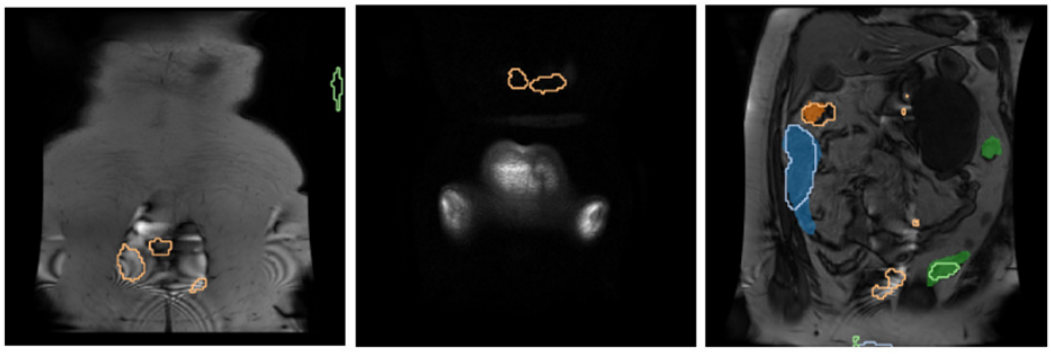
Misclassified voxels of FG and BG occurred in all segmentation scenarios; however, it is found that regions which clearly have a different appearance to that found in the training set (referred to as the Phillips dataset) contained significantly more misclassified voxels (Figure 4.11). This could occurr due to the difference in contrast information between scanners. Artefacts which were only present in the unseen dataset (GE scanner dataset) are often misclassified as regions of colon. Since they are not present in the Philips scanner dataset the network was not trained to interpret this noise correctly. This is important since different scanning methodologies are likely to generate their own unique artefacts, but with transfer learning it may be possible to reduce the effect



(a) 2D (b) 2.5D (c) 3D

Figure 4.10: *Examples of network predictions on a volume with the highest accuracy between networks on the GE scanner.Coloured regions denote the ground truth with the blue, orange and green corresponding to AC, TC and DC respectively. Slices with pink labels are the corresponding binary segmentations. The network outputs are delineated by the thin contour lines.*

of these. The effect of these artefacts on accuracy are evident in the results shown in Table 4.2 of the quantitative analysis section.



(a) (b) (c)

Figure 4.11: *Noisy 2D network predictions in coronal slices. (a) and (b) do not contain any colon regions. All images show incorrect predictions whilst (a) and (c) show artefacts within the MRI which cause misclassification. Coloured regions denote the ground truth with the blue, orange and green corresponding to AC, TC and DC respectively. The network outputs are delineated by the thin contour lines.*

4.6 Summary

This chapter reviewed the effect of data dimensionality on the accuracy of the network. A 2D, 2.5D and 3D network were compared where we found that the 3D network performed better than the 2.5D network which in turn outperformed the 2D network for the task of segmenting individual colon sections. This indicates that depth information provides important context for a CNN. However on the task of binary segmentation context is less important, and we observes that the difference in performance between networks is much lower, with the 3D network performing best followed by 2D, and 2.5D networks. Although this is a slight anomaly the differences between the 2D and 2.5D network results are marginal.

Predictably 2D networks are the fastest to run at inference time followed by 2.5D and then 3D networks, this result remains the same even when data loading times are factored in. 2D and 2.5D networks are impacted far more by data loading where it causes a 2.15× and 2.12× increase in computational time. The 3D networks run time is only increased by around 1.17×, the difference is likely caused by the repeated loading required for each slice in the 2D and 2.5D networks when processing a single volume. Conversely the 3D network only requires a single load as the entire volume is held together in one file, this is a big advantage especially when dealing with larger datasets. It is unclear whether the data loading associated with 2D and 2.5D networks scales poorly as higher dimension datasets are used.

The chapter shows it is possible to use the 3D network successfully on images from an entirely different scanner without the use of transfer learning. In comparison the 2D and 2.5D network perform very poorly showing that there is some inherit robustness garnered from using volumetric filters. Once more this is important for the LAT dataset as there exist volumes from two differ-

ent machines giving different colour and intensity characteristics. Our data shows the 3D network doesn't only rely on local pixel intensity of a given slice it gathers information from further afield unlike the 2D network. The 2.5D network appears to use some surrounding information but does not rely on it heavily as in the 3D network.

The 3D network is the clear choice if the computational power given the dataset is available to the user. This is key because the 3D network scales poorly due to the VRAM consumption, the 2.5D network only uses 5.6GB whilst the 3D network consumes 11GB. The performance benefits are a good trade off for the increased accuracy but further work is needed to decrease the VRAM usage when scaling up to larger resolutions or large 3D volumes. This is especially the case in situations such as the super resolution LAT volumes dataset presented in 5.

Parallels can be drawn between the work here and those in 5 both suffer the same VRAM shortage related to using CNNs on ever larger images and volumes. Future work should investigate the commonality with aims to exploit this to produce a generic solution for both problems. Part of this will require investigating the minimum surrounding depth required to accurately segment volumes in order to maximise accuracy and minimise the resource usage. Again parallels are likely to be seen with high resolution image segmentation where we would seek to minimise tile size, reduce VRAM usage and maintain accuracy.

3D networks demonstrate that having greater context available to any slice in a volume gives better results.

Next chapter 5 explores the spatial domain and the associated issues with current techniques when image size is scaled up. This is a crucial aspect to

solving the high resolution problem presented there but also the ever growing number of high resolution volumetric datasets that are being produced.

Chapter 5

Common Techniques for segmenting high resolution images

5.1 Introduction

This section explores common techniques used to segment high resolution images in deep learning. This section only considers methods related to modifying the images such as tiling and downsampling. These methods are most common due to the ease of implementation, network specific modifications are much more complex and time consuming causing most developers to opt for the methods discussed here. Usually network specific adaptions are only explored when results from these techniques are not sufficient. Often state of the art segmentation networks produce high accuracy outputs but are so resource intensive they cannot be used for images with resolutions above those seen in standard datasets. Using most of the commonly available segmentation networks on super resolution images is not possible without first tiling or downsampling mostly due to VRAM constraints.

As seen in chapter 4 using volumetric convolutions to segment 3D volumes pro-

duces the most accurate results but is far more resource intensive especially in regards to VRAM. Infact VRAM is the resource bottleneck for both high resolution image segmentation and 3D volume segmentation. The LAT dataset we discuss in chapter 3 is both volumetric and high resolution in the spatial domain. In such a case volumetric convolutions cannot yet be considered and instead solving the problem in 2D first is best. To solve the problem in 2D the issue of high resolution image segmentation must be tackled before moving on.

Image downsampling and tiling are evaluated, both the advantages and disadvantages of implementing the techniques are discussed. Further to that each method's inherit drawbacks are discussed along with any impacts to qualitative results. Experiments are conducted to investigate the effect of progressively downsampling the input image on the final result. The results are evaluated at the downsampled resolution and at the native resolution to see the final impact of the technique. The effect of tile size in comparison to whole image resolution is explored using 3 tile sizes at 3 different whole image resolutions.

Finally each technique is qualitatively and quantitatively evaluated and compared. Network specific future uses are discussed for each technique along with a small discussion of whether each technique can be adapted to solve the task in a better way. In particular the scalability of each technique in regards to VRAM consumption is considered given that it is the current resource bottleneck for high resolution image segmentation.

5.2 Downsampling

Downsampling or subsampling is a specific image re-scaling technique where a higher resolution image is transformed into a lower resolution representation. There are various algorithms for carrying out downsampling the most common of which are bilinear and nearest neighbour. The former is more popular for use

in situations where data is continuous such as when dealing with intermediary featuremaps or an input image. Nearest neighbour is popular when the data is categorical such as when dealing with an image mask or classmap ground truth.

Although it is very easy to implement, downsampling inevitably results in loss of information. Where an image contains redundant information this may not be an issue however there are various drawbacks associated with this technique. Some tasks, such as classification, make sparse per image predictions and therefore suffer less accuracy losses from this technique when compared to segmentation. The sparse prediction requires just enough pixels to recognise objects, for most objects this a small number of pixels as seen in the latent space representations produced by a CNN which are low resolution featuremaps. Segmentation attempts a dense per pixel prediction meaning loss of information around a small number of pixels effects the accuracy profoundly caused by the loss of information around object borders.

If downsampling takes place at continuously larger scales important image features are lost, these features can no longer be segmented because the pixel data is not fed to the network. Large features which remain in the image have their borders eroded away resulting in less accuracy in these regions. This effect is made worse if upsampling is used to return the network output to the same size as the original image. Upsampling has the drawback that it is difficult to know which pixels in the higher resolution image correspond with the output, once the corresponding output pixels have been chosen a simple copying of values occurs resulting in a blurry output. Figure 5.1 shows the effect of downsampling on input image features both small and large. The figure shows a cropped Stele from an root cross section acquired from the LAT process downsampled by a factor of 4 and 6. The large black vessels (Xylem) remain largely unaffected by the downsampling however the small cells surrounding these vessels are almost completely eroded at a resolution of

512×512. At this resolution even the Xylem vessels have their borders eroded.

Finally lower resolution segmentation outputs cannot easily be used in real world scenarios without heavy post processing. They are rendered useless in tasks requiring high precision. One work around commonly used is to train on lower resolution downsampled images and infer on the higher resolution images at run time. This may solve the issue for some images however for 3D volumes or very large spatial resolution images it does not due to growing size difference between the learnt features and those presented at inference. Moreover both techniques are susceptible to the aforementioned drawback where small important features are not segmented at all.

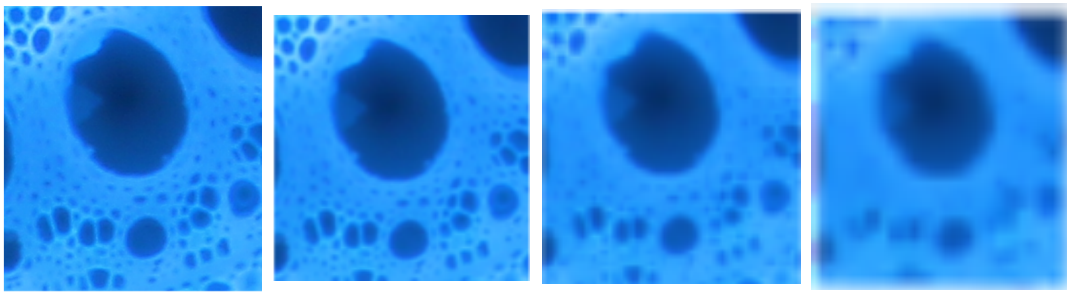


Figure 5.1: *The Stele cropped from a root cross section which has been downsampled by a factor of 4 and 6 resulting in an image size of* 1024×1024 *and* 512×512*. The stele cropped from the original image with no downsampling is also presented. The large black vessels forming a radial pattern are Xylem, this feature remains largely unaffected where only its borders are made blurry. Small cells, especially those around the xylem and outer most edge of the Stele are almost lost.*

5.2.1 Experimental Design

Figure 5.1 shows how the input image is effected by downsampling, in order to prove its knock-on effect on results a series of experiments are conducted. Firstly the original images are downsampled to the resolutions of 1024×1024, 512×512 and 256×256 before input, a CNN is then trained to segment the images at these resolutions. Accuracy at these resolutions is measured by downsampling the ground truth to the same resolutions as the network

outputs. The network is trained in a consistent way with the same parameters across all experiments in order to avoid any confounding variables. We train a similar stacked hourglasses with the same structure as the network presented in chapter 4. Each hourglass is stacked only once with a depth of 4, at each depth a residual block is place on both the encoder and decoder side. Each network is trained for 600 epochs with a batchsize of 1 using a learning rate of $1 \times 10^{-}4$. 80 images are used in the training set and 8 images are used in the test set, the number of and image sets are kept consistent across experimental runs. For each experiment the most accurate state of the network over the 600 epochs is selected where accuracy is measured by Dice score.
The network outputs are also upsampled, these network outputs and compare them to the ground truth downsampled to 2048×2048 and the ground truth at the native resolution of 3072×3072. We also train the networks at the aforementioned two input resolutions and infer directly on images at 2048×2048 and 3072×3072 to show the results of this method.

This experiment simulates the results we should expect in a real scenario where the output is required to be the same or near to native resolution, it explores both possible methods for achieving this when downsampling is required at training time. The networks are not trained using higher resolution images due to memory limits.

5.2.2 Results

The results in table 5.1 show that training at a lower resolution yields a lower accuracy than training at any higher resolution. This is true when the evaluated network output is upsampled to a higher resolution and when the network output is evaluated as is. When upsampling the network output to any resolution beyond the training resolution accuracy is lost. Most importantly network outputs upsampled to the native ground truth resolution have low-

ered accuracy regardless of the training resolution used. Networks trained at larger resolutions have their accuracy impacted less than those trained at lower resolutions. 256×256 loses 6.5% accuracy when upsampling to 3072×3072 whereas the network trained at 1024×1024 only incurs a loss of 1.2%.

	Upsampled Resolution				
Training Resolution	**256 × 256**	**512 × 512**	**1024 × 1024**	**2048 × 2048**	**3072 × 3072**
256 × 256	78.0	71.0	71.0	70.5	69.6
512 × 512	-	81.3	78.1	77.7	77.2
1024 × 1024	-	-	**81.6**	**79.6**	**79.0**

Table 5.1: The accuracy of the hourglass network when trained with various input resolutions. Each network is evaluated at the trained input resolutions as well as increasingly upsampled resolutions up to the native image resolution of 3072×3072. Accuracy is measured using Dice score.

Figure 5.2 shows the results of the aformentioned experiments, more specifically we highlight the results when the input images are downsampled to 256×256 and 1024×1024 and where the outputs are both upsampled to 3072×3072. The figure shows a large amount of pixelation in cells and erosion around cell borders for the 256×256 case. For the 1024×1024 case pixelation does not occur but on closer inspection cell boundaries are eroded in comparison to the ground truth. Class confusion seldom occurs in both cases and when it does it is usually a small cluster of pixels within a cell. Due to the downsampled the 256×256 case misses many cells and gives the incorrect shape for most small cells. This does not occur in the other case. The qualitative differences here are also reflected in the quantitative results in the previous subsection.

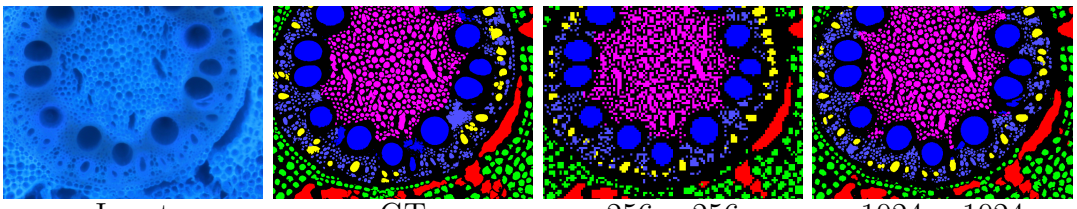


Figure 5.2: *This figure shows cropped outputs from tiling networks. From left to right the images show the cropped input, the cropped ground truth (GT) and cropped outputs from Hourglass networks trained on images downsampled to* 256×256 *and* 1024×1024*. The input image shows the image at the original resolution of* 3072×3072*.*

5.3 Tiling

Tiling, commonly referred to as patching or cropping, is the method of splitting an input image into multiple sub-images along predefined boundaries. Usually the sub-images are all of equal size and are referred to as patches or tiles. During training tiles are passed to the network in a random order so that the network learns filters applicable to the tiles. When a segmentation output of the entire input image is needed, each tile in turn is passed through the network. The results are stitched together in order resulting in an output the same size as the initial image.

This technique reduces the VRAM usage by approximately n times where n is the number of tiles because the network only views x/n pixels of the original image where x is the number of pixels in the original image. This memory reduction applies to both training and inference time.

Although tiling drastically reduces the VRAM cost of executing a CNN it suffers from several drawbacks. The CNN can only see the local context of a tile which may be enough to accurately contour the desired object but may not be enough to classify the object correctly. These scenarios occur when different objects share visual characteristics but their classification is also determined by some other characteristic such as location. In biology, cells share similar appearances but their classification may be determined by their position within a larger structure. Another such situation is when a tile resides inside the boundaries of large object causing similar class confusion. An example of this is when a tile resides within a vehicle that we wish to segment, it is impossible to say whether the door or vehicle panel belongs to a truck or a car.

Another drawback of this approach is that it leads to edge artefacts. This phenomena occurs when objects are segmented around the edges of two cor-

responding tiles and the segmentation outputs of each tile do not align. The network can only see each tile independently from the rest and therefore cannot adjust the segmentation output such that an object across multiple tiles is classified and segmented coherently.

5.3.1 Experimental Design

The LAT image dataset, both ground truth and input image, is taken and downsampled to 2048×2048 and 1024×1024. The stacked hourglass network mentioned in the downsampling section of this chapter is used for all experiments in this section also. The hourglass only contains one stack, with a depth of four containing a single residual block at each level of the encoder and decoder. Multiple different networks are trained on tiles of resolution 128×128, 256×256, 512×512 and 1024×1024 for each of the aforementioned full image resolutions including the native resolution of 3072×3072. This gives a total of 11 different experiments. One experiment is omitted since the tile size is equivalent to the input image size. The tiles in the batch are randomly selected from the randomly selected training set images on the fly at each training iteration. Each training session concludes after 600 epochs using a batch size of 1 at a learning rate of 1×10^{-4}. As before 80 images are used for training and 8 are used for the test set. Again the most accurate network state in the 600 epochs is selected, Dice score is used to measure the accuracy. The network and training set up allows for a fair comparison between the downsampling and tiling techniques.

5.3.2 Results

The results in table 5.2 show that at all resolutions using a smaller tile size yields worse results, the lowest accuracy at all resolutions are achieved by the networks trained using the lower tile size. Conversely the highest accuracy

achieved is by networks trained with the highest resolution tiles. More specifically the network trained at a tile size of 128×128 only achieves an accuracy of 54.9% when segmenting an image of resolution 3072×3072. However a network trained at a tile size of 1024×1024 achieves an accuracy that is 21.0% higher at an accuracy of 75.9%. As a network of a given tile size is trained on larger images, performance is lost due to the loss of global context. The greatest performance decrease is seen at the lowest tile size of 128×128, when trained on images of resolution 1024×1024 it achieves an accuracy of 73.9% however this drops by 19% to 54.9% when training on images of resolution 3072×3072. Conversely the network trained on tile size 512×512 only loses 4.8% accuracy dropping from 80.6% to 75.8% given the two resolutions.

	Image Resolution		
Tile Size	1024×1024	2048×2048	3072×3072
128×128	73.9	62.4	54.9
256×256	78.9	72.2	68.0
512×512	80.6	79.0	75.8
1024×1024	-	79.9	75.9

Table 5.2: The accuracy of the hourglass networks when trained with various tile sizes at various whole image resolutions. All networks are evaluated at the resolution they are trained at using the stitched tile outputs from the network. Accuracy is measured in dice score. '-' denotes an experiment where the tile size is equivalent to the image size and therefore is not conducted.

Figure 5.3 shows the outputs from networks when input images of 3072×3072 are tiled into 256×256 and 512×512. The outputs are cropped to better highlight differences. Both networks show noticeable edge artefacts and class confusion, the tiling network at 256×256 shows these issues much more clearly. Aside from these qualitative issues the outputs do not suffer from pixelation or boundary erosion as shown in the downsampling cases.

5.4 Conclusion

Downsampling leads to poor accuracy and often the complete loss of features of interest before the image is even processed. Moreover when examining the qualitative results from the upsampled outputs of a network trained on down-

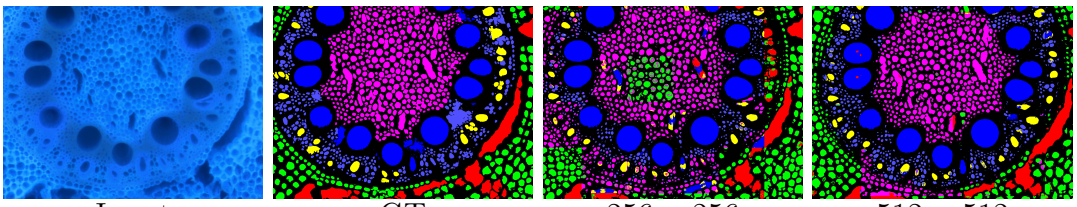


Figure 5.3: *This figure shows cropped outputs from tiling networks. From left to right the images show the cropped input, the cropped ground truth (GT) and cropped outputs from Hourglass networks trained on tiles of sizes* 256×256 *and* 512×512*. All tiles are randomly selected from the input images at a native resolution of* 3072×3072*.*

sampled input images we find a significant degradation in the boundaries of objects. This gives the final masks a pixelated look. Not all tasks using segmentation networks require accurate object boundaries to be used in the real world but many tasks do rely on it such as robotics where a robot must interact with objects or tasks where shape analysis takes place (such as in the case of cell anatomic). Although this technique may be used in other computer vision tasks such as classification without a significant degradation in performance this chapter has shown the significant impact of downsampling images in relation to the task of segmentation. The results presented in this chapter have shown that processing an image at the highest resolution possible yields the highest accuracy even when those outputs are upsampled to the desired resolution.

Although downsampling is an easy solution requiring minimal software engineering (given that any image processing library has a native implementation of both downsampling and upsampling) and no extra deep learning network adjustments it scales poorly for the task of deep learning. As the resolution of images increase the performance decreases but the available VRAM remains consistent, it does not provide a scalable solution to larger images especially if those larger images have small objects of interest. The chapter does not provide an experimental analysis of this technique on 3D volumes but intuitively the issue of scalability only becomes worse in such a case. This is further compounded by the extra VRAM requirement of volumetric convolutions as

shown in chapter 4 The scalability issue with downsampling means it will not be evaluated or used further in this thesis as a technique for efficient segmentation of high resolution images or 3D volumes. Further future work outside of the scope of this thesis may explore some combination of downsampling and network specific adaptations to combine the ease of downsampling with a scalable network.

This chapter also explored the relatively easy technique of tiling, which although more difficult to implement provides a more scalable and efficient solution to segmenting higher resolution images in comparison to downsampling. The best performing network is 3.2% less accurate than the best performing downsampling network but also requires 4x less VRAM due to training on tiles half the resolution in both dimensions. Another advantage of tiling is that the images are processed as native resolution meaning object boundaries are accurately preserved unlike with downsampling. This makes the technique suitable for tasks where objects are relatively small compared to the image resolution and where object boundaries are important.

Tiling does suffer from multiple disadvantages. The qualitative results show issues with edge artefacts which cannot be resolved without a technique which passes over the whole output mask or having a network with such high accuracy that adjacent tiles agree on segmentation boundary and classification of pixels. Tiling also suffers from misclassification of cells which is mostly unseen in the downsampling qualitative results. These misclassifications exist in cases where two cells have similar appearances and are defined by their location in the image. Both the problem of edge artefacts and misclassification is caused by the same issue, lack of global context.

In chapter 6 the technique of tiling is taken and adapted such that global context can be maintained whilst the advantages of lower VRAM at training

time are preserved. This allows a network to train on super resolution images whilst reducing issues with edge artefacts and misclassifications. It also allows the network to train on smaller tile sizes, reducing VRAM usage, whilst maintaining most of the accuracy.

Chapter 6

Revisiting Tiling - Global Context Aggregation Via Crosstalk

6.1 Introduction

Chapter 4 showed that volumetric networks achieve much greater accuracy on 3D datasets but are incredibly VRAM intensive even at low spatial resolutions. Even with a small number of slices, volumetric convolutions almost exceed the VRAM capacity of commonly used higher end GPUs. Larger volumes already exist both spatially and volumetrically, these volumes are already inaccessible without using a technique from chapter 5. Chapter 5 already showed that these techniques should be avoided due to their drawbacks. Given that our LAT dataset is volumetric we must address this issue. This issue is further compounded by the high resolution nature of the dataset. In chapter 5 we showed the common techniques used for high resolution image segmentation. These techniques are also often used for 3D volumes as they both share VRAM consumption issues. Chapter 5 showed that downsampling is not a viable option because as the image scales the inaccuracy associated with the method increases. It also showed that tiling can be effective at retaining accuracy but

introduces the problem of diminishing global context as tile size decreases. Moreover the method introduces edge artefacts which may not show a significant different quantitatively, but qualitatively can effect the network outputs to an unacceptable level. The chapter also discussed the surrounding issues of pre- and post-processing of the images required for tiling.

This chapter abandons the use of downsampling as an option and instead explores how tiling can be improved in order to preseve the VRAM savings whilst alleviating the problem associated with lack of global context. The chapter solely focuses on solving the problem of high resolution spatial segmentation and does not address volumetric segmentation. The chapter covers how the pre and post processing steps of tiling can be integrated into the forward pass of a network, the advantages of doing so and how crosstalk (XT) a novel module can be used to share global context between tiles. A number of ablation studies on the XT module are carried out to show which parts of the module contribute accuracy increases. An extensive analysis of how XT effects accuracy over different tile sizes is performed. This is used to ascertain how effective XT can be when sharing global context. Comparisons are made between a naive tiling network and a tiling network which includes the new XT module. A summary comparison is made between the previous techniques in chapter 5 an the technique introduced here.

6.2 Converting Networks

Pre-processing of images into tiles is fairly simple however it must be made universal to work with any network. First a class is defined which acts as a wrapper around any preexisting network, the class receives as attributes the desired network and the number of tiles for both the X and Y dimensions. The number of tiles therefore determines the tile resolution E.g. a 4×4 tile scheme on a 2048×2048 resolution image would result in 16 tiles of resolution

512×512. To achieve this tiling effect the class includes a definition for a forward function which in turn calls the forward function of the desired network. Before this call takes place the narrow function native to PyTorch is used to split up the incoming image into the desired number of tiles. This results in a list of tensors whose size is calculated using the method above.

Each tile can be passed through the network fully and in order. The resulting transformed list of tensors can be spatially aligned to create a network output with equivalent dimensions to the input image. From here this image can be returned and passed to the loss function.

6.2.1 Selective Backpropagation

The solution outlined in the previous subsection works but is missing a key component. In its current form the gradients accumulated from this process would cause the VRAM cost to be similar to simply passing the full resolution input image to the native network. This is because gradients accumulate each time the network executes a forward pass on a tile. Intermediary featuremaps are stored for each tile to compute the gradients, when summed these are similar to parsing the whole image. To combat this gradients are only calculated over a single tile, a random tile is chosen such that when this tile is passed to the network the gradients are calculated. All other tiles are run such that no gradients are accumulated. This process is refered to as selective backpropagation since only the selected tile is included in the backpropagation step during training.

Although this approach does not provide a large benefit over simple tiling code outside of the network it lays the ground work in adding further elements to allow tiles to communicate within the network. It also allows the ability to apply post network filters to share information between tiles and remove edge

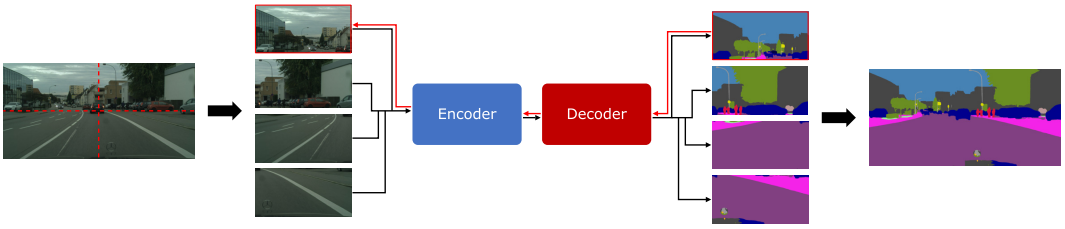


Figure 6.1: *A high-level overview of in-network tiling. A tiling scheme of 4 tiles is shown in a 2×2 configuration. Forward passes are performed individually and sequentially. Gradient flow is highlighted using the red arrow. As shown backpropagation is only applied to a single tile on any given training iteration.*

artefacts.

6.3 Post Tiling filters

Given the converted network outlined in the previous section the easiest way to provide global context to the featuremaps and correct edge artefacts is to add convolutional layers that are applied over the entire image. Adding these layers after the tiles have been stitched into a single image is effective. In this experiment the hourglass, defined in chapter 4, is converted into a tiling hourglass with additional post processing filters added. These post processing filters are a bottleneck block given in the original resnet works [9]. This is comprised of 3 convolutional layers separated by batch normalisation layers and relu functions. The convolutional layers have a kernel size of 1×1, 3×3 and 1×1 where the first layer reduces halves the number of channels, the second continues with that number and final convolution restores the number of channels to the same as those pre bottleneck block. Note that all experiments in this section are conducted on the LAT dataset downsampled to size 2048×2048 at a tile size of 512×512. This is kept consistent to allow for fair comparisons.

Table 6.1 shows the result of bottleneck blocks being added after the decision layer in the network. In this case the decision layer is a convolutional layer that predicts the class of every pixel and as such it reduces the number of

channels to the number of classes. In the case of the hourglass the number of channels is reduced from 32 down to 7 classes. The table shows that the naive tiling network with no XT and no post filters achieves an accuracy of 79% whilst adding one bottleneck block increases that accuracy by 0.2% percentage points. Conversely adding 2 bottleneck blocks decreases the accuracy by 0.4% percentage points. We hypothesize that it is difficult for the network to learn these filters, and that adding too many bottleneck blocks creates a network that is difficult to train.

The added post filtering block increases the accuracy but includes the drawback of an added 0.4GB of VRAM. Adding additional blocks increases the VRAM by a similar amount with no advantage.

Bottleneck Blocks	**Accuracy (Dice)**	**VRAM (GB)**
0 (No post filters)	79.0	**2.57**
1 Block	**79.2**	3.04
2 Blocks	78.6	3.44

Table 6.1: Accuracy and VRAM consumption of hourglass when comparing a naive tiled network (No XT and no post filtering) against an hourglass with added bottleneck blocks post tile stitching. Filters are run over the stitched image after the prediction layer has been executed, this means the number of channels is equivalent to the number of classes. The image used in this example is an LAT image of dimensions 2048×2048 at a tile size of 512×512.

A similar experiment is conducted where bottleneck blocks are replaced with the basic resblock from the original Resnet. The basic resblock uses 2 3×3 convolutional layers in a row both with relu and batch norm. The results of the experiment are shown in table 6.2, where it is clear that adding basic res blocks has a detrimental impact on accuracy. Adding a single basic resblock as a post filter reduces the dice accuracy by 0.6% whilst adding 2 reduces the dice score by 1.3%. Adding the basic resblocks incurs an additional 0.4GB per block. Despite incurring a similar VRAM cost the basic resblock is not advantageous especially when compared to the bottleneck block. Overall, these experiments show that resblocks do not provide enough to share context between different areas of the image.

Experiments were also conducted with adding simple convolutional layers, with batch norm and relu however this also resulted in a decrease in network performance. We theorise that the lack of elementwise summation or concatenation, which can be seen as the lack of being residual, causes training to be unstable. This in turns results in the network being difficult to train. This effect is seen in the training of networks before the introduction of the residual block.

This small set of experiments indicates that resblocks or just convolutions cannot provide the complex information sharing required to increase global context. At most, these resblocks will share pixels around the boundary of tile. This is because the kernel sizes of the convolutions are only 3 pixels at most and the tiles are at a resolution of 512×512.

Basic Resblocks	Accuracy (Dice)	VRAM (GB)
0 (No post filters)	**79.0**	**2.57**
1 Block	78.4	3.05
2 Blocks	77.7	3.45

Table 6.2: Accuracy and VRAM consumption of hourglass when comparing a naive tiled network (No XT and no post filtering) against an hourglass with added basic resblocks blocks post tile stitching. The image used in this example is an LAT image of dimensions 2048×2048 at a tile size of 512×512.

6.4 Global Context Aggregation Via Crosstalk

Global context is the wider context of the image and the information associated with the general theme of the image. Global context is not required for all problems, as in some cases local pixels provide enough information to segment an area. For the LAT dataset many different cell types have similar visual representations and are defined better by their location in the image. They are also defined in respect to other surrounding cell classes. To know the location in the image, a wider more global context is required. If the entire image is visible to the network the entire global context is available and these ambiguities quickly disappear, however when tiling the available context

shrinks as the tile size decreases. To address this issue information must be shared across tiles so that the network can learn positional information and share tile predictions to correct neighbours' predictions.

Given that the goal is to preserve the memory savings from using a tiled approach, the XT module must not increase the VRAM usage substantially and most importantly it's VRAM usage should scale well with increases in image resolution. Ideally an effective module would also preserve accuracy as tile size decreases rapidly. Figure 6.2 outlines the final XT module design from the perspective of a single tile. XT includes 3 main Attention in, filtering and attention out. The first stage is subdivided into channel and spatial attention which allow XT to filter the most important channels and pixels. This is important as it determines what is shared from each tile later. Spatial attention is especially important because pooling is used after this stage. Selecting and highlighting the most important pixels is critical since only a few will be selected.

The second stage comprises of pooling, spatial alignment, global image filtering and XT memory. The steps are applied respectively. Pooling preserves the efficiency of the crosstalk module by making sure only important are processed and shared. Whilst spatial alignment allows the global image filtering to share information between tiles whilst they are in the correct location. XT memory allows previously global images from previous XTs to help guide the module.

The final stage uses upsampling, spatial and channel attention. Upsampling restores the featuremap resolution before each tile is separated again. Attention before a tile exits the module is used to selected which learnt, shared features are most important.

In figure 6.2 three dots denote processes that take place individually for each

tile, this applies for attention, pooling and upsampling. Spatial alignment and global image filtering take place over all tiles simultaneously. The key steps are explained in the following subsections.

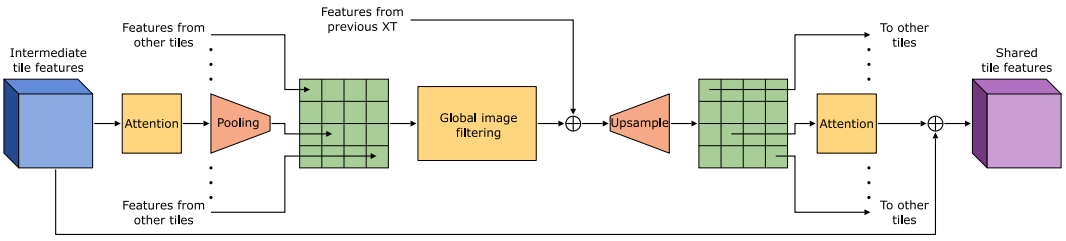


Figure 6.2: *An overview of the XT module from a single tile perspective. Feature maps from each tile are passed through channel and spatial attention sequentially. These features are then average pooled to a pre-set size. The new low resolution features are passed through a global image filtering residual block to share information between tiles. The result of this global image filtering is combined with global image features from the XT block of the previous decoder level. Upsampling is used to return the features to the same size before pooling after which tiles are separated again. Spatial and channel attention is applied before finally combining the feature maps originally input to XT.*

The XT module is made residual to avoid destabilising training or adding difficulty to the training process of the network. Early experiments with XT modules which were not residual gave some promising results but often caused difficulty in training. The residual design is similar to that found in the original Resnet paper, an elementwise addition is used with the output of XT and the incoming featuremaps. This addition is symbolised by the plus sign encased in a circle found in figure 6.2.

6.4.1 Attention for selection

After featuremaps enter the XT module, channel and spatial attention are applied which are denoted by the yellow boxes labelled ‘attention’ in figure 6.2. The key use for channel attention is to learn which channels in the incoming tensor are most important and therefore which should be the focus for the global filters. Spatial attention’s use is similar except it focuses on pixel data and weights the most important pixels. Spatial attention is helpful within in

the XT module because it proceeds a pooling operation. Using spatial attention prior to the pooling allows the block to increase the contribution of more important pixels causing them to influence the output of the pooling operation.

The attention mechanism in the module is identical in design to the convolutional block attention (CBAM) [61]. Channel attention is applied first, the result of which is fed into the spatial attention. For channel attention the spatial resolution is first average pooled down to 1×1 such that the tensor takes the form $B \times 1 \times 1 \times C$ where B is batch size and C is the number of channels. The original paper follows this process twice, once for maxpooling and once for average pooling. Experiments were conducted using both together and each alone, it was found that average pooling alone provided the best performance.

Next the newly resized featuremaps are sent through a multi-layer perceptron (MLP) or fully connected layer. A reduction factor is applied to reduced how many units are in the MLP. Experimentally only 4 and 8 were tried, it was found that 4 provided the best performance without producing too much VRAM overhead. The result of the MLP layer is fed to a sigmoid function resulting in all values being scaled to between 0 and 1. This is useful because the output will become a weightmap with equivalent dimensions to the input of channel attention, multiplying this calculated weightmap will essentially dim less important channels. All pixels for the same channel are multiplied by the same weight.

The result of the channel attention is fed into the spatial attention. Much like channel attention, spatial attention uses pooling. However, the pooling operation is applied along the channels. Again the original paper uses a combination of max and average pooling however we found that average pooling gave the best results and as such we only used a single pooling strategy. Post channel pooling the tensor takes the shape $B \times H \times W \times 1$ where B is batch size, H

is featuremap height and W is featuremap width. The result of this operation is fed to a simple convolutional layer with a kernel size of 3×3, this layer learns to filter the most important pixels. Note the simple convolutional layer is accompanied by a batch normalisation layer and a relu as in the original CBAM implementation. Similar to channel attention this final featuremap is converted into a weight matrix using a sigmoid function the result of which is multiplied with the original incoming featuremaps. Each channel receives the same multiplication value however pixels have differing values between 0 and 1.

We found that using attention prior to pooling and after upsampling in our XT module gave the best results. Ablation studies are presented later to show the effect of different configurations. Attention at the begining of the module allows for selection of the most important feauturemaps in each tile and allows the module to focus on the most important pixels in each tile. Doing so ensures the correct information is shared between tiles to maximise the global context shared. Similarly attention post global filtering allows each tile to focus on which pixels and featuremaps are most important for it.

6.4.2 Pooling and Spatial Alignment

Before the filtering can be applied pooled tiles are spatially aligned to create an intermediate representation of the entire image therefore allowing for global context. Aligning the tiles in this way causes the filters to share information using simple convolutions. Spatially aligned tiles are represented by the green boxes in figure 6.2, in this diagram all tiles are used during XT. Pooling the featuremaps is a crucial part of reducing the VRAM of this added module. The size to which featuremaps will be pooled is predetermined during the network definition, multiple ablation studies are presented later determining which pooling size is best and how the result of pooling can effect global filtering parameters. This module can be added at every level of the decoder and as

such would double the VRAM cost of the decoder if the global image filters are assumed to be basic blocks. Pooling allows for a low resolution representation of the state of each tile and allows the maintenance of all featuremaps.

6.4.3 Global Image Filtering and Memory

Global image filtering is a simple set of basic resblocks that are applied over the spatially aligned intermediary image representation. The global filters must have a receptive field size large enough to share information between tiles, this is mostly determined by the pooling parameters in the previous step. Ablation studies are conducted on changing the filter size from 3×3 up to 7×7 along with the number of these filters and whether the convolutional layers should include strides for maximum receptive field size.

Many computer vision tasks are solved by continually refining an image from a low resolution representation upwards to a high resolution output. This idea is adapted and employed here through the use of XT memory. XT memory is where the latent global image representation from the previous XT at a lower level of the decoder is upsampled and added to the current latent global image representation. This allows for a slow refinement of the global image in step with the refinement of the network predictions for each tile. In practice we found that this increased performanced, ablation studies for this are presented in the quantitative results section.

6.4.4 Intergrating Crosstalk

Theoretically the XT module can be added to any network, the most intuitive point to add it to the network is post decoder. The XT module can be added anywhere along the decoder including the bottleneck of the network between the encoder and decoder. Here, we add XT at each level of the decoder to maximise results. This means the module outlined in figure 6.2 is placed

at each image scale in the decoder which is represented by the red box in figure 6.1. More specifically the module is placed after the upsampling, skip connection and convolutional layers in any common decoder. Figure 6.1 only serves to illustrate how an image can be tiled up and processed by a tiling network whilst figure 6.2 shows the exact module which will be added to the decoder. Since it can be added to any decoder no specific decoder is shown in figure 6.1.

6.5 Crosstalk Results

All results below are gathered through training with identical setups to those in chapter 5 to allow fair comparison. These experiments use the same Hourglass [59] with the same depth, number of resblocks at each level and number of stacks. These are 4, 1 and 1 respectively. The learning rate used here is identical before and is set at 1×10^{-4}. Finally the network is trained for 600 epochs using the same training, validation and test set. Both networks are trained at 3 input resolutions of 1024×1024, 2048×2048 and 3072×3072. At each resolution 3 tile sizes are used 512×512, 256×256 and 128×128.

Table 6.3 shows the first experiment comparing a tiling network with XT against its identical counterpart without XT. XT is naively added at each level of the decoder including before the decision layer and at the bottleneck point of the network between the encoder and decoder. The pooling for XT is set such that the featuremaps are pooled to a resolution of 4×4. A single resblock is used within XT with a kernel size of 3. The results show that an identical network including XT achieves a greater accuracy at all tile sizes for all input image resolutions. In some cases the XT network at a lower tile size outperforms the non-XT network at a higher tile size. One such example is a tile size of 512×512 on images of resolution 3072×3072. At this resolution by 2.1%. Crucially the XT network performs best at the native ultra resolution

of 3072 × 3072 achieving a result of 79.3% compared to 77.2%.

	Accuracy (DICE)								
	1024			2048			3072		
Method	512 × 512	256 × 256	128 × 128	512 × 512	256 × 256	128 × 128	512 × 512	256 × 256	128 × 128
Tiling	80.6	78.9	73.9	79.0	72.2	62.4	77.2	68.0	54.9
Tiling XT	**81.6**	**80.9**	**77.9**	**80.5**	**79.6**	**75.3**	**81.3**	**79.3**	**62.7**

Table 6.3: Results of two tiling networks on the LAT dataset. One network includes the XT module at each layer of the decoder, including the bottleneck. Results are given for 3 input image resolutions and 3 tile sizes.

Results at the two downsampled resolutions are included to illustrate 2 key points, the tile size as a percentage of the input image determines global context and has a strong effect on the performance of a tiling network. The lower the tile size, as a percentage, the smaller the global context and in turn the worse the performance. Secondly XT helps preserve the accuracy of the network as the gap between tile size and input resolution increases. Specifically in the case of the 2048 × 2048 image we see that the no XT network accuracy decreases by 6.8% then 9.6% between tile schemes. However the XT network's accuracy only decreases by 0.9% and 4.3% over the same tiling schemes. Moreover the gap between the networks at these two tiling schemes of 256 × 256 and 128 × 128 is 7.4% and 12.9% respectively. In cases where the tile covers large regions of the image the effect of crosstalk is less pronounced. At a tile size of 512 × 512 on images of size 1024 × 1024 only an accuracy increase of 1% is observed when using XT. XT only provides a small benefit in this scenario because each tile includes most of the global context required to segment images accurately.

This effect is shown in 6.3 where the table is plotted except the X axis is tile size as a percentage of the input image size. The plot shows that the lines that represent the networks which include XT have slightly more favourable gradients as the accuracy drops less. Overall in all cases where one point from each network share the same X axis value (tile size percentage) the XT network obtains a higher performance. Interestingly, points which share the same X

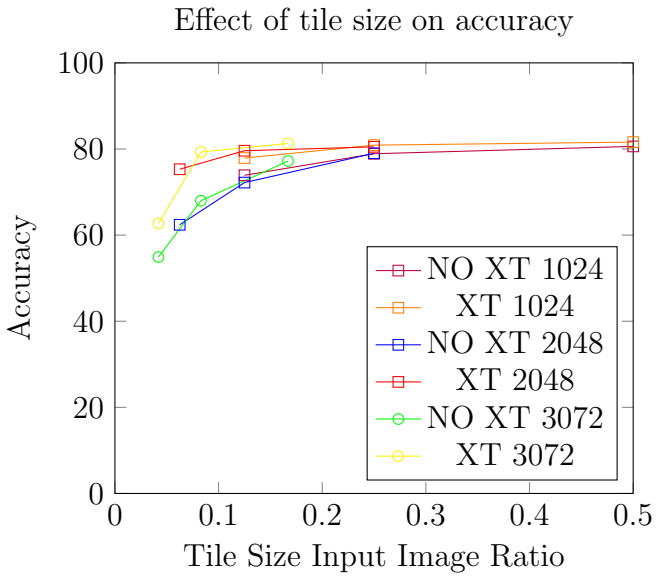


Figure 6.3: *The data from table 6.3 is plotted here. The tile size input image ratio is the tile size dimensions divided by the input image dimensions.*

axis value within the network e.g. a tile size of 256×256 at an input resolution of 1024×1024 and a tile size of 512×512 at an input resolution of 2048×2048 (In this case the tile size is 1/4 of the input image for both) share very similar accuracies and as such the plot overlaps.

Next, VRAM consumption of each method is explored, table 6.4 shows the VRAM consumption of the experiments shown in table 6.4. For all tile sizes and input resolutions the tiled network without XT uses less VRAM, this is to be expected given XT provides the performance boost seen in 6.4 with an overhead.

For both networks the VRAM savings given by using a small tile size display a diminishing returns effect. When calculating the VRAM cost of training for a tiled network we add the cost of the forward pass of all tiles without gradient accumulation, then add the cost of the forward and backwards pass for the one selected tile. Fundamentally the forward pass cost remains the same regardless of tile size since we are processing the whole image as tiles, only the backwards pass cost is reduced. As an example, moving from a tile size of 512×512 to

256×256 creates $4\times$ the number of tiles. Every new tile still requires a forward pass but now only 1 tile, 1/4 the previous size requires a backwards pass.

	VRAM (GB)								
	1024			2048			3072		
Method	512×512	256×256	128×128	512×512	256×256	128×128	512×512	256×256	128×128
Tiling	**2.18**	**1.55**	**1.42**	**4.22**	**3.67**	**3.29**	**7.49**	**7.18**	**6.53**
Tiling XT	3.43	2.53	2.32	7.70	6.42	6.41	12.1	11.9	12.0

Table 6.4: Results of two tiling networks on the LAT dataset. One network includes the XT module at each layer of the decoder, including the bottleneck. Results are given for 3 input image resolutions and 3 tile sizes. Results here are the accompanying VRAM consumption data for 6.3.

The network with XT suffers a greater diminishing returns effect due to the overhead of the added module. This is because the module applies the global filters over all the spatially aligned tiles. Given that the target resolution for the pooling is the same for each tile, a greater number of tiles will make for a larger spatially aligned image. If the number of tiles is quadrupled, as in the case outlined above, the spatially aligned image will also quadruple.

The diminishing returns effect is shown most obviously in table 6.4 where the network is trained on images of resolution 2048×2048 at tiles sizes of 512×512 and 256×256. Only a saving of 10MB is made whilst the accuracy suffers a 4.3% loss. When examining the same tile sizes on input images of size 3072×3072 the VRAM consumption actually increases by 10MB whilst dropping 16.6% accuracy.

The overhead scaling less favourably with excessive tile numbers is because XT is is applied over all tiles. If only immediate neighbours of a selected tile are considered, the VRAM scales much more favourably. An ablation study is included in a later subsection to explore this in greater depth.

6.5.1 An Alternative Lightweight Tiling Computation

In the converting networks chapter a process was outlined for adapting any network to a tiling version of itself. This subsection outlines changes that can

further reduce VRAM consumption with some added limitations. In the previous method all tiles were run sequentially however if XT modules are placed throughout the decoder many intermediary featuremaps must be stored for all tiles. The lightweight version of tiling passes a tile through the encoder and decoder entirely before moving onto the next tile. Once all tiles are processed they can be combined before or after a prediction layer although the forward and backwards pass VRAM cost remains the same as before VRAM is saved due to the decrease in stored intermediary featuremaps. A XT module can be placed at the end of the decoder however the main limitation of this method is XT blocks cannot be placed throughout the decoder.

This alternative tiling computation scheme is refered to as XT-Light and is compared with the previous scheme for networks both with and without XT, results for VRAM and accuracy are given in tables 6.5 and 6.6 respectively. To make comparisons fair we used the same pooling parameters and resblock parameters inside XT, these are a pooling target size of 4×4 and a single resblock with kernel size 3.

When comparing the Tiling-Light and Tiling-Light XT to their counterparts an immediate VRAM saving is shown. At an input size of 3072×3072 tiling consumes 2.2×, 3.1× and 3.0× the VRAM when compared to its Light counterpart. This is at the 3 given tile sizes from left to right. Almost no changes in accuracy are seen in this case because the network has not changed only the way intermediary featuremaps are stored has been changed. In some cases, such as such as a tile size of 256×256 on images of size 1024×1024 there is a very small difference in accuracy. This is due to the non deterministic nature of training.

Similarly tiling XT consumes 4.2× and 4.6× the VRAM when comparing to Light-XT. Again this is for the two tiling schemes from left to right. In nearly all cases Tiling XT achieves a higher accuracy than Tiling-Light XT, intu-

	VRAM (GB)								
	1024			2048			3072		
Method	512 × 512	256 × 256	128 × 128	512 × 512	256 × 256	128 × 128	512 × 512	256 × 256	128 × 128
Tiling	2.18	1.55	1.42	4.22	3.67	3.29	7.49	7.18	6.53
Tiling-Light	**2.09**	**1.17**	**0.93**	**2.52**	**1.63**	**1.38**	**3.39**	**2.35**	**2.16**
Tiling XT	3.43	2.53	2.32	7.70	6.42	6.41	12.1	11.9	12.0
Tiling-Light XT	2.10	1.18	0.97	2.68	1.78	1.58	3.70	2.84	2.60

Table 6.5: The memory consumption of each of the tiling methods with and without XT. VRAM is given at all resolutions and tile sizes from $1024 \times 1024 - 3072 \times 3072$ and $128 \times 128 - 512 \times 512$ respectively.

itively this is expected due to the larger overall number of XT blocks placed throughout the network. On the most important input size of 3072×3072 Tiling-Light XT achieves the highest accuracy overall when considering all tiling schemes. Notably Tiling-Light Xt is able to easily run on the GPU at the tile size of 512×512. whereas Tiling XT is very close to exceeding GPU memory limits.

At an input size of 3072×3072 table 6.5 also shows that Tiling-Light XT only consumes 9.1%, 21% and 20% more VRAM compared to Tiling-Light. The small increase in VRAM is accompanied by a 2.6%, 5.0% and 4.1% increase in accuracy. A final examination of table 6.6 finds that Tiling-Light XT displays the same diminishing returns effect reported in the previous subsection. It is able to retain accuracy at all tile sizes better than its non-XT counterparts but performs poorly when compared to the original Tiling-XT network. Overall these results indicate that Light tiling should be used over the original tiling if no XT is to be implemented. If XT is used Tiling Light should be used for larger tile sizes on larger images. In a later subsection a compromise between the two extremes is explored by limiting the tiling computation to very close neighbours.

A final ablation study is conducted for the Tiling-Light XT network to ascertain where the best XT module placement post decoder. Two different positions are tried, before and after the decision layer. The previous experiment discovered that Tiling-Light XT should only be used with larger tile sizes

	Accuracy (DICE)								
	1024			2048			3072		
Method	512 × 512	256 × 256	128 × 128	512 × 512	256 × 256	128 × 128	512 × 512	256 × 256	128 × 128
Tiling	80.6	78.9	73.9	79.0	72.2	62.4	77.2	68.0	54.9
Tiling-Light	80.6	79.0	73.9	79.0	72.2	62.4	77.2	68.0	54.9
Tiling XT	**81.6**	**80.9**	**77.9**	80.5	**79.6**	**75.3**	**81.3**	**79.3**	**62.7**
Tiling-Light XT	80.7	80.2	75.8	**80.8**	78.5	66.7	79.8	73.0	59.0

Table 6.6: The accuracy of each of the tiling methods with and without XT. VRAM is given at all resolutions and tile sizes from $1024 \times 1024 - 3072 \times 3072$ and $128 \times 128 - 512 \times 512$ respectively.

on larger images. As a result only images of size 2048×2048 and 3072×3072 at tile sizes 1024×1024-256×256 are considered. Note that the tile size of 1024×1024 was not included in the previous tables 6.5 and 6.6 for brevity. When considered, the additional information does not change the main table standings. Tiling-Light XT is still outperformed by Tiling-XT in all cases except one.

To allow for a fair comparison no training parameters or network parameters are changed except where the XT module is placed. The XT module used in each experiment is also identical. Accuracy results are shown in table 6.7 whilst VRAM results are shown in table 6.8. Which network performs best is split between the two module placements. Firstly VRAM becomes a limitation for the 'before' network at the large tile size of 1024×1024. This is because the resolution of the featuremaps are at their largest point at this part of the network. Moreover the number of featuremaps is 32 whilst after the prediction layer there are only 7.

The before prediction layer network performs better overall at the lower tile sizes whilst being outperformed at the higher tile sizes. This indicates that more featuremaps are needed to correct tile predictions when the context available in a given tile is small. It also indicates that more featuremaps can be obstructive to learning when a large amount of context is available, in this case very simple small errors are likely corrected that only need a few featuremaps to be shared between tiles. Overall the best performing network at each reso-

lution is after prediction layer network. It performs better by 0.2% and 0.5% when any tile scheme is considered at resolutions 2048×2048 and 3072×3072 respectively.

	Accuracy (DICE)					
	2048			3072		
XT Module Placement	1024×1024	512×512	256×256	1024×1024	512×512	256×256
Before Prediction Layer	-	80.6	**78.5**	-	**80.2**	**75.1**
After Prediction Layer	**80.4**	**80.8**	**78.5**	**81.4**	79.8	73.0

Table 6.7: Accuracy results over a range of image sizes and tile sizes where the XT module placement is varied between processing featuremaps before the network decision layer and after it. Note the '-' symbol denotes an experiment which cannot be considered due to VRAM limitations.

Larger numbers of featuremaps being processed by XT required noticeably more memory, table 6.8 shows that placing the XT module before the prediction layer requires $2\times$ or more the VRAM in nearly every experimental setup. Whilst no accuracy increase is observed when considering real world scenarios.

	VRAM (GB)					
	2048			3072		
XT Module Placement	1024×1024	512×512	256×256	1024×1024	512×512	256×256
Before Prediction Layer	-	4.41	3.75	-	7.61	7.14
After Prediction Layer	**6.22**	**2.68**	**1.78**	**7.56**	**3.70**	**2.84**

Table 6.8: VRAM consumption data over a range of image sizes and tile sizes where the XT module placement is varied between processing featuremaps before the network decision layer and after it. Note the '-' symbol denotes an experiment where the VRAM consumption exceeded the 12.3GB limit on the graphics card.

6.5.2 Crosstalk Module Placement in the Decoder

The XT module can be placed at any level of the decoder and between resblocks or convolutional layers. Empirically it was found that placing the XT module before or after the resblock in the decoder resulted in no difference in performance. Conversely ablation studies were conducted where the XT module was omitted from differing levels of the decoder. A performance difference was found here and is shown in table 6.9. Placing a single XT module at the bottleneck of the network, at the end of the final level of the decoder and adding one to each level of the decode all yield the same result. Each achieve an accuracy of 81.3% when conducting the experiment on a tile size of

512×512 with images of resolution 3072×3072. Adding a single XT module to levels 1, 2 and 3 of the decoder yields accuracies of 80.1%, 80.4% and 80.1% respectively. These results clearly demonstrate that the most important points of the network to place a XT module is at the bottleneck and at the final stage of the decoder. Naturally, adding a module to all levels yeields the same result however results in more GPU memory usage. The most memory efficient XT module placement is at the bottleneck of the network, this configuration consumes 8.79GB whilst adding a module to each level consumes an additional 3.41GB.

XT Module Placement	Accuracy (%)	VRAM (GB)
Bottleneck	**81.3**	8.79
Level 1	80.1	**8.75**
Level 2	80.4	9.35
Level 3	80.1	10.4
Level 4	**81.3**	11.7
All levels	**81.3**	12.2

Table 6.9: The effect of XT modules placed at different levels of the decoder. Bottleneck is the region between encoder and decoder and such is before the first level of the decoder. All experiments are on images sized 3072×3072 at a tile size of 512×512 with a XT pooling target size of 4×4.

By considering the results in table 6.6 and table 6.9 it is possible to conclude that the best scheme for XT module placement is situation dependent. If a large tile size is chosen XT in the final level of the decoder or even the bottleneck should be considered. However if smaller tile sizes are chosen XT modules should be placed throughout the decoder to maximise global context sharing and increase accuracy as shown in table 6.6.

6.5.3 Varying Target Pooling Size

Pooling size is a crucial hyperparamter for the XT module and it is unclear what the optimal target pooling size should be. The target pooling size should be kept small so that the module remains memory efficient. An ablation study is conducted to determine the optimal pooling size for the XT module. Pooling sizes of 1×1, 4×4 and 8×8 are used at tile sizes of 512×512, 256×256 and 128×128. Using multiple tile sizes allows for a relationship to be established

between tile size, global context and pooling size. Each experiment is conducted at an image resolution of 3072×3072. Table 6.10 shows that the best results are obtained a tile size of 512×512. On this tile size the best results are obtained when the largest pooling size of 8×8 is used. This is likely because global context is not required as much and as such small refinements around tile boundaries occur. These refinements are most optimal when a higher spatial resolution is used. The 8×8 pooling achieves an accuracy of 82.1% which is 1.2% higher and 2% higher than the 4×4 and 1×1 pooling respectively.

For the middle tile size of 256×256 the best results are found when a tile size of 4×4 is used. It achieves an accuracy of 79.3% whilst 8×8 and 1×1 only achieve an accuracy of 74.4% and 73.3% resectively. Finally the 1×1 and 8×8 pooling schemes achieve similar accuracies of 70.3% and 70.4% whilst the 4×4 scheme only achieves an accuracy of 62.7%. It is unclear why this large decrease in accuracy is seen. In general the results show that a larger tile size should be combined with a large pooling size and a lower tile size should be combined with a lower pooling size.

	Accuracy (%)		
Pooling Size	512×512	256×256	128×128
1×1	80.1	73.3	70.3
4×4	81.3	**79.3**	62.7
8×8	**82.1**	74.4	**70.4**

Table 6.10: The effect of varying the pooling size on network accuracy. Results are given for images of resolution 3072×3072.

6.5.4 Varying Global Filter Hyperparameters

Changing the kernel size of the convolutions inside a resblock will impact the filters learned by the network. Doing so can lead to more favourable filters leading to higher final accuracies because the network may need to learn filters over a larger scale.

Different kernel sizes also result in different receptive field sizes (RFS). A con-

volution's receptive field size is the region over which a kernel can receive information from the pixels. For a single convolution this is equivalent to the kernel size but for larger structures, such as the XT module, the RFS is the sum total of all the contained convolutions' RFS'. Increasing the RFS of the XT module can be achieved using larger kernel sizes or adding more convolutions through the addition of resblocks.

The XT module's RFS can be seen as it's ability to see over the whole image represented by the spatially aligned featuremaps. This ability to see also determines how far information may travel, if the RFS is smaller than the size of a tile (after it has been pooled) information will only be passed between neighbouring tiles around border regions. If the RFS is larger than the size of a tile, information may pass between multiple tiles or larger amounts of information inside a single tile may be passed to it's neighbours.

		512×512		256×256	
Filters	**RFS (px)**	**Accuracy (%)**	**VRAM (GB)**	**Accuracy (%)**	**VRAM (GB)**
1 3×3 Block	5	80.7	**6.24**	78.7	**6.45**
2 3×3 Block	9	**80.8**	7.62	**79.4**	6.48
4 3×3 Block	17	80.6	7.81	78.8	6.62
1 5×5 Block	9	80.4	7.91	79.2	6.66
2 5×5 Block	17	80.2	8.07	79.2	6.84

Table 6.11: The effect on accuracy of different resblock configurations in the XT module. Block is an abbreviation for resblock whilst RFS is an abbreviation for receptive field size. The filters column gives the number of resblocks and the size of the kernels in them. Experimental results are given for images at resolution 2048×2048. The XT module uses a pooling size of 8×8, this is used instead of the usual 4×4 to allow for more experimentation with receptive field size.

Table 6.11 shows the result of investigating the impact of varying kernel size and number of resblocks. The target pooling size of XT is increased from the usual 4×4 to 8×8. This is to allow a single experiment where the RFS of the XT module is smaller than the size of a tile. The results in the table are for images of size 2048×2048, this resolution is used instead of 3072×3072 because it is easier to control the total size of spatially aligned featuremaps which results in an easier experimental design. Additionally experiments with larger tile sizes and larger kernel sizes on the higher resolution result in out of

memory errors.

The XT module with the highest performance for both tile sizes uses 2 resblocks with a kernel size of 3×3, it achieves an accuracy of 80.8% and 79.4% for tile sizes of 512×512 and 256×256 respectively. The RFS of this XT module is 9 which is larger than the size of a tile in this experiment. For the larger tile size a configuration with an RFS of 5 only achieves an accuracy of 0.1% lower than the highest achieved accuracy. At most the XT module will share border information of a single tile with its neighbours or share small amounts of information from two tiles together. This implies that information sharing is less important due to the larger context available within a tile. Conversely for the smaller tile size this same XT module achieves the lowest performance, this implies that global context is more important in this case and that this RFS may lead to this result.

The XT modules containing 1 resblock with kernel size 5×5 and 2 resblocks with kernel size 5×5 achieve an accuracy of 79.2%. This is just 0.2% lower than the highest accuracy achieved for the lower tile size. The RFS of these two modules is 9 and 17 respectively which are both values above the size of a single tile. Memory usage is very similar for all XT module designs in the smaller tile size are very similar, the range is only 0.39GB. The range for the larger tile size is 1.83GB where the highest memory usage is from the 5×5 kernel resblocks, particularly where 2 resblocks are used. These XT modules are less accurate and more memory intensive therefore they should be overlooked in future in favour of the 3×3 kernel size resblocks.

6.5.5 Adding Crosstalk Memory

Adding XT memory allows spatially aligned featuremaps from previous XT modules to be combined with the current XT module being considered. Doing

so allows guidance from previous XT blocks and incorporates featuremaps from previous scales. An element-wise addition is used to combined previously spatially aligned featuremaps. This process occurs post global filtering. An ablation study is conducted to see how much of a benefit to accuracy XT memory is. Table 6.12 shows the improved accuracy of adding XT memory to all XT modules in the decoder. Only the first XT module in the decoder does not include memory since at this point in the decoder there are no previous XT modules.

XT Module Type	Accuracy (%)	VRAM (GB)
XT No Memory	78.4	**11.7**
XT w/ Memory	**79.3**	11.9

Table 6.12: The effect of adding memory to the XT module. Results are given for a network where XT modules are placed at every level of the decoder. Input images sizes are 3072×3072 with a tile size of 512×512.

Removing XT memory from the module decreases the network's final accuracy by 0.9% and results in a 0.2GB decrease in GPU memory consumption.

6.5.6 Varying Tile Count in Crosstalk

Not all tiles must be used in XT, if a tile only requires its immediate neighbours to achieve sufficient global context, including the additional tiles further afield results in more memory consumption. The problem is more difficult to learn and potentially produces no performance benefits. An experiment is conducted to investigate the effect of the inclusion of neighbouring tiles on the accuracy of the network. To facilitate this neighbouring scheme, the loss function must first be changed such that it is only calculated over the selected tile. Currently the entire image is used to calculate the loss.

A whole image loss function takes the outputs of all tiles and calculates a loss over them, however the gradients only flow through the single tile, this single tile is the selected tile for that iteration. This allows the single tile to update using the average performance of the network over the whole image. Alterna-

tively the loss over the selected tile can be calculated and used to update the weights in the network.

The first method outlined here requires a forward pass over all tiles which is more costly but also simplifies and speeds up whole image segmentation at inference time, something classical out of network tiling methods don't benefit from. They require multiple single passes through the entire network before image stitching can take place outside of the network. Similarly the second method outlined suffers from the same problem but allows for a significant amount of memory to be saved. When compared to the out of network counterpart this second method can sequentially forward pass multiple tiles without calculating a loss immediately. One such case where this is needed is if XT modules are placed throughout the decoder and a local neighbours XT scheme is being used. This requires the featuremaps of just the 4 or 8 neighbours and therefore requires their forward passes. Although this is more memory intensive it could allow for greater accuracy since information from neighbours Incorporated. It could still however, be less accurate than the first method which allows for XT across all tiles.

An ablation study was conducted to investigate whether whole image loss results in a more accurate network than a tile loss. No difference in accuracy was observed between the two loss types. Using the above method an ablation study may take place to examine the 4 and 8 neighbour tile schemes.
A 4 neighbour tiling scheme is used first which only includes one tile in each direction. Left, right, above and below are used to make the spatially aligned image inside the XT module, in this scheme global filters are learnt over this region and cannot gather context from other tiles. In the 8 tile neighbour scheme 4 extra tiles are included, these are diagonally above and below in each direction, left and right. The default tile scheme which includes all tiles in XT is compared against these two in table 6.13. The experiments take

place on images of resolution 3072 × 3072 at tile size 512 × 512. All other XT module hyperparameters are kept the same to allow for a fair comparison. In this experiment Tiling-Light and tile loss is used to take advantage of the memory savings. The smaller tile size is used to allow for an adequate number of tiles for the 8 neighbour tile scheme.

Included Tiles	Accuracy (%)	VRAM (GB)
4	78.7	**3.58**
8	79.1	3.61
All	**79.8**	3.70

Table 6.13: The effect of varying the number of included neighbouring tiles in XT. All results are given for images of 3072 × 3072 at a tile size of 512 × 512.

The results show that including all tiles in the XT global spatially aligned image yields the best result. Including all tiles gives an accuracy of 79.8%, using 8 neighbours gives an accuracy of 79.1% and finally using 4 neighbours gives an accuracy of 78.7%. The memory consumption of each is very similar which is shown by the range. The range is only 0.12GB, the most memory efficient network is the 4 neighbouring tiles network.

6.5.7 Additional Lesser Ablation Studies

This subsection includes experiments and ablation studies which are small in nature and do not warrant a subsection of their own.

Overlapping Tiles as an Alternative Baseline Overlapping tiles were considered as an alternative baseline to non-overlapping tiles. Although overlapping tiles improve qualitative and quantitative results slightly, on a local level, several issues were found empirically. Firstly overlapping regions in the tiled segmentation masks require extra computation to resolve how masks should be combined. There are several methods including, averaging, selecting the tile with the highest softmax probability for given pixels in the overlapped regions or choosing the overlapped tile to take precedence over other tiles. Each has advantages and disadvantages and efficacy may vary between problem domains and even between images in problem domains. Overlapping tile

schemes also require computational overhead due to the extra tiles being segmented. A tile is required wherever tiles border, resulting in a method which scales poorly as the number of tiles increases. Empirically it was found that the difference between overlapped and non-overlapped tiles only differed slightly because overlapping tiles do not offer global context. The overlapping regions only provide a greater local context which resolves very local ambiguities but does not solve issues where context is required from a tile beyond the immediate neighbours. One such case is where the tiles are very small due to very large input image resolution. Another such case is on the root dataset where cell classifications are made within the Stele (inner concentric ring of the root). An overlapping tile does not provide enough global context for the network to disambiguate whether cells are within the Stele or Cortex (outer concentric ring of the root). To do so would require context from tiles multiple positions over. However if a tile is on the border of the Cortex and Stele overlapping tiles may provide some context from either Stele or Cortex to help with cell classification. Edge artefacts will also be improved by overlapping tiles if the overlapping tile prediction takes precedence over non-overlapping tiles.

Batch Size The VRAM savings acquired through tiling allowed for higher batch sizes than before. Many experiments at different batch sizes were carried out but no extra performance benefit was seen. The literature indicates [62] that changing the learning rate with batch size can increase performance therefore three different learning rate schemes were used. Firstly a square root increase in learning rate was used. Then a linear increase in learning rate was used and finally no increase in learning rate was used. In all three schemes no increase in performance was seen, the linear increase in learning rate yielded the best results of the three but often still performed worse than a simple batch size of 1. However increasing the batch size did speed up the time taken to train the network. Note that the batch size available is limited by which network, tile size and input size was used. Batch sizes of 1-8 were tried al-

though not all configurations were experimented with due to the large number of configurations available.

Encoder XT Adding XT to the encoder is an intuitive step which was experimented with. Many configurations were tried ranging from a single XT module in the encoder to one at each level of the encoder, much like other experiments. No performance increase was seen, in fact in many cases a performance decrease was seen. A negative correlation between accuracy and number of XT modules in the encoder was observed as such XT was not included in the encoder in experiments thereafter.

Non-local Blocks Non-local blocks [63] were designed to capture longer range dependencies in images and videos as such they are essentially an alternate attention mechanism to modules such as CBAM [61]. As such these were inserted as a replacement the attention mechanism. No performance increase was found in fact a performance decrease was found. Additionally the GPU memory consumption increased by a large amount causing XT to no longer be lightweight.

6.6 Crosstalk Final Comparison

In-network tiling and XT were designed to out compete the common methods for handling high resolution image semantic segmentation, namely downsampling and out-of-network tiling. This hypothesis is tested by doing a comparison of all the methods on the LAT dataset at all high resolutions. The best performing scheme of each method is given and its efficiency is considered through its VRAM usage, this allows us to see the potential of the method for use on even larger images.

Both tiling XT and Tiling-Light XT are included to show the two main techniques described in this chapter. The best results achieved in any ablation study are shown in table 6.14. The XT module in the Tiling XT method uses

	Image Size			
	2048 × 2048		3072 × 3072	
Method	**Accuracy (Dice %)**	**Train VRAM (GB)**	**Accuracy (Dice %)**	**Train VRAM (GB)**
Downsampling	79.6	8.43	79.0	8.43
Tiling (Naive) 1024 × 1024	79.9	8.42	75.9	8.42
Tiling (Naive) 512 × 512	79.0	**2.64**	75.8	**2.64**
Tiling-Light XT 512 × 512	80.7	2.68	79.8	3.70
Tiling-Light XT 1024 × 1024	80.7	6.17	81.4	6.99
Tiling XT 512 × 512	**80.8**	7.62	**82.1**	12.1
Tiling XT 1024 × 1024	-	-	-	-

Table 6.14: The results of the best trained networks using methods from this chapter and from chapter 5. Naive tiling is given two rows because the results are very similar in the case of the higher resolution images. Results are given for 3072 × 3072 and 2048 × 2048 resolutions. Tiling XT uses a pooling size of 8 × 8 with a tile size of 512 × 512. It uses 2 resblocks with kernel size 3 × 3. Whilst Tiling-Light XT uses a tile size of 1024 × 1024 with a pooling size of 4 × 4. Only 1 3 × 3 kernel resblock is used here. Note the '-' symbol denotes an experiment which cannot be considered due to VRAM limitations.

a target pooling size of 8 × 8 with 2 resblocks at a kernel size of 3 × 3. A XT module is placed at every level of the decoder. Conversely the Tiling-Light XT method uses a a target pooling size of 4 × 4 and a single resblock with kernel size 3×3. The methods use a tile size of 512×512 and 1024×1024 respectively.

At the lower resolution of 2048 × 2048 the best performing method is Tiling XT which achieves an accuracy of 80.8% only 0.1% higher than Tiling-Light XT. It also consumes 1.45GB than the Tiling-Light XT method. Downsampling and Naive Tiling achieve just 79.6% and 79.9% accuracy respectively. The most memory efficient network is the naive tiling network which consumes just 2.64GB of GPU memory. For a resolution of 3072 × 3072 the best performing method is the Tiling XT method which achieves an accuracy of 82.1%, this method is the most memory intensive and consumes 12.1GB of GPU memory. This method outperforms the Tiling-Light XT method by a larger margin of 0.7%. However it consumes 5.11GB more GPU memory in this case. Downsampling and Naive tiling only achieve 79.0% and 75.9% respectively. Again the most memory efficient method is the naive tiling which consumes just 2.64GB. The memory consumption of this technique is equivalent across resolutions because the memory cost is associated with the tile size alone a variable we did not change.

6.7 Summary

This chapter introduced in-network tiling which is a way of adapting pre-existing or new networks to tile up images, segment them and realign the tiles to output a whole image. The purpose of this is to allow for the segmentation of high resolution images where the main barrier is excessive VRAM consumption during training. Selective backpropagation is introduced to keep memory cost low by only training on a single tile at a time. Splitting images into tiles reduces this cost but has drawbacks. In network tiling reduces the global context available to each tile resulting in class confusion even if segmentation boundaries are accurate. Crosstalk is introduced to share information between tiles making global context available to all tiles.

A second more memory efficient version of tiling is introduced where the tile passes the entire encoder and decoder before the next tile is processed, this results in very few intermediary featuremaps being stored during training. The method suffers from the drawback of only allowing a single crosstalk module at the last level of the decoder just before or after the decision layer. Ablation studies show that it competes well with the previously defined crosstalk but only at large tile sizes, as tile sizes decrease this method attains lower accuracies due to its limited capacity to share global context. The main advantage of this version of crosstalk is that it is much more scalable than the original, this may allow it to segment even larger images in future.

Crosstalk modules are added to the decoder because it is found that adding them to the decoder results in networks that do not train well and have a lower final accuracy. The chapter explored where the XT module should be placed and shows that the bottleneck of the network or at the final level of the decoder results in the best accuracy. Adding a module at each level of the decoder gives the best final accuracy especially when tile sizes are reduced. A

larger tile size is shown to have a higher overall accuracy but also results in more memory consumption.

The chapter shows that crosstalk achieves better accuracy than a tiling network which uses resblocks as global filters around the network decision layer. This global filtering alternative is also shown to scale poorly in comparison since it scales based on the size of the input image and not the size of tiles.

Smaller tiles need more global context and as such this chapter has shown that a larger receptive field size in the crosstalk module results in a better accuracy. Ablation studies were presented for pooling size, and tile count where it was shown that the optimal pooling size is related to the tile size. A bigger tile size and bigger image require bigger pooling sizes to preserve spatial resolution. Including all tiles in the spatially aligned featuremap used for global filtering yields better results than just the neighbouring tile in each direction. When including neighbouring corner tiles results are more accurate than just 4 neighbours but less accurate than including all tiles.

This chapter compares the methods outlined with the common methods for segmenting high resolution images presented in chapter 5. The chapter shows that both Tiling Crosstalk and Tiling-Light XT result in higher performing networks when compared to downsampling and naive out of network tiling. In the case of Tiling Crosstalk this results in a higher consumption of GPU memory, Tiling Light Crosstalk is more memory efficient and scalable than either downsampling or naive tiling.
In the next chapter this technique is applied to an additional commonly network and other high resolution datasets to show it can be applied universally.

Chapter 7

Proving Crosstalk Across Datasets and Networks

7.1 Introduction

This chapter takes the generic tiling approach outlined in chapter 6 and applies it to other datasets and networks. Applying the method to other networks and showing minimal performance loss proves the universal nature of the XT module meaning the results shown in chapter 6 are not from network specific adaptations. Applying the method to other high resolution datasets shows that the method is not specific to the LAT dataset. In turn this shows it is generalisable. Comparing on other datasets also allows comparisons with other methods seeking to segment ultra high resolution images.

Details are outlined around adapting the popular semantic segmentation network DeepLabV3+ and results are given for the Cityscapes dataset. Multiple ablation studies are carried out to explore whether themes from chapter 6 hold true across datasets and networks using the tiled XT approach. The network is compared against a baseline without tiling and checkpointing, an alternative method of training networks to reduce VRAM consumption. Checkpointing is not a CNN and therefore provides an interesting challenge for XT which

can also be applied to any network. In theory this method could be used to train high performance semantic segmentation networks on higher resolution images. GLNET, an alternative high resolution semantic segmentation CNN, is trained on Cityscapes to allow for comparisons on a standard dataset. One final high resolution segmentation technique, namely MagNet, is compared on this dataset also.

Another high resolution dataset, DeepGlobe, is used to test XT against 3 other state of the art methods. These methods are GLNet, PPN and Magnet, these are given in order of ranking on this dataset. Results are presented for each network comparing accuracy and VRAM in order to evaluate how XT compares to the best networks for this problem area.

Lastly, a synthetic dataset is created to test whether XT is truly sharing information between tiles. The synthetic dataset is a semantic segmentation logic puzzle which can only be solved with context from the entire image, which XT should provide. Predictions are made around what the accuracies should be, given the probabilities associated with the puzzle. These predicitions are then compared to what actually occurs.

7.2 Cityscapes and DeepLab

7.2.1 Revisiting the Data

Cityscapes is a road scene dataset where 19 classes are evaluated, the classes include cars, buses, trees, street signs, humans and much more. It includes 20,000 coarsely annotated images and 5,000 finely annotated ones, in the experiments used in this chapter only the finely annotated images are used. The coarsely annotated images are omitted because their effect to increase performance is well known and as such increasing training time to include them

does not yield any novel knowledge. The finely annotated images are split into 2,750 training, 500 validation images and 1,525 test images, the split given is standardised and the 1,525 images are evaluated using a test server where the annotations are kept secret from the public.

7.2.2 DeepLabV3+ Overview

DeepLab is a popular semantic segmentation network that is considered close to state of the art on many standard datasets (outlined in chapter 3). It has seen many iterations from DeepLab to DeepLabV3+ where the simplicity of the decoder has remained throughout. The following subsections use DeepLabV3+ for experimentation so a brief outline of the network is given below.

As with all semantic segmentation networks DeepLab follows an encoder-decoder structure where the encoder can be almost any classification network. Popular choices range from mobilenet to the set of Resnets with varying depths. The decoder consists of 2 levels in which convolutional layers are applied, an atrous spatial pyramid pooling (ASPP) block is also used at the bottleneck of the network. The bottleneck of the network is the point between the encoder and the decoder where the featuremaps are smallest. The ASPP block takes encoder features one level above the deepest point of the network and parses them using convolutions at various dilation (atrous) rates. Dilations are gaps between which pixels a kernel selects when filtering a featuremap. Larger gaps result in a greater context in the resulting featuremap therefore varying the dilation rate produces featuremaps containing image information at multiple scales.

A 1×1 convolution is included at the output of the block which acts as a selection layer. It bottlenecks the returned featuremaps from ASPP, this

increases computational efficiency (reducing VRAM and FLOPs) and allows the network to learn which featuremaps are important at each scale. The ASPP block allows the network to consider information at multiple scales just as a multi-level decoder with skip connections does. However it does so without using as many levels. As a standard for cityscapes the ASPP block uses dilation rates of 12, 24 and 36. DeepLabV3+'s decoder only contains two levels which pull featuremaps from the ASPP block, between each level upsampling at a rate of $4\times$ is used to restore image resolution.

7.2.3 Checkpointing Overview

Checkpointing is a universal non-network specific method to save VRAM. It works by reducing the backpropagation VRAM consumption (much like the tiling scheme outlined in this thesis) by limiting the storage of intermediary activation maps from the forward pass. In a standard scheme all intermediary activation maps are stored during a single forward pass and used during the backwards pass to calculate gradients. Saving them during the forward pass rather than computing them on the fly saves time. Checkpointing recomputes the necessary activation maps by rerunning the forward pass for each node. A checkpoint is a stored activation map, if multiple checkpoints are used VRAM cost increases but computation time is decreased. This is because only a subset of forwarded passes is required. The subset is the forward pass between two checkpoints which will generate the missing activation maps. In the most extreme case no checkpoints would be used and multiple forward passes would be needed for each activation map. Checkpointing was originally devised to allow for the training of large memory expensive networks. However given that it saves memory, normal networks can use this method to segment high resolution images since the barrier is also excessive VRAM use.

Checkpointing is effective at reducing VRAM but comes with two drawbacks.

Firstly the computational time during training is increased and secondly implementing checkpointing for a network is complicated, requiring in-depth knowledge of CNNs. Some commonly used layers such as dropout and batch normlisation (The most popular layer along with convolutions and ReLU) are effected by forward passes. It cannot be used with dropout since dropout randomly ignores a node upon every forward pass. Careful consideration must be used when using checkpointing to ensure checkpointing is placed around such layers. Although it seems trivial to do so. if a network is defined in a monolithic function this is seldom the case.

Good programming guidelines result in large sections of networks being modularised and within those modules there is often further modularisation. This makes the implementation of checkpointing even more difficult, and often results in the need to completely rewrite code where these special layers are used. Batch normalisation was designed for a single forward pass. It is updated upon each pass and must be modified when using checkpointing since multiple forward passes are used. All of these considerations and complexities are a barrier to use this results in the method rarely being used.

Checkpointing is included in comparisons because it allows a baseline network to save VRAM and maintain accuracy. Moreover the method could theoretically allow more computational expensive CNNs to segment larger images.

7.2.4 Adapting DeepLab

DeepLabV3+ is adapted using the same method outlined in chapter 6, firstly the network is changed so that it splits the input image into a set of $X \times Y$ tiles which are processed by the encoder and ASPP block separately. Both the encoder and ASPP block remain unchanged. The decoder is extended slightly such that it includes 3 levels rather than 2. An additional resblock is

added at each level, this process is carried out for the tiling and tiling with XT variation to discern how much performance increase is being provided by XT. When adding XT the module is placed after each level of the decoder. The upsampling rates are changed to accommodate this additional decoder level, the first upsampling rate of 4 is now 2. Finally the ASPP block's 1×1 convolution is adapted to include twice as many output featuremaps. The network now receives featuremaps at a reduced scale so the dilation rate used in the ASPP block is proportionally reduce to 4, 8 and 12.

7.2.5 Experimental Design

Training of DeepLabV3+ baseline, tiling and tiling XT variants is done with parameters very close to the original paper. Stochastic gradient descent is used with a poly learning rate schedule [64]. The baseline learning rate is 0.1 and the learning rate power used in the poly scheme is set to 0.9. The baseline is trained on random crops of 768×768 whilst the tiling networks are trained at crops of 1024×512. The baseline uses the random crop from the original paper whilst the tiling networks used the most performant crop size. The baseline was trained at a crop size of 1024×512 but a lower performance was observed. A maximum of 30,000 iterations is used in the experiments. All DeepLabv3+ networks are trained using the ResNet50 as the encoder.

Checkpointing is applied to both the baseline and tiling XT. For the baseline network, 2 checkpointing segments are used for the decoder. Similarly the XT network uses 3 checkpointing segments. Both networks use identical checkpointing for the encoder, a segment is used for each level of the encoder resulting in 4 segments total. The outlined checkpointing implementation is more aggressive than others, this is done to save as much VRAM as possible.

The local branch of GLNet uses a crop size of 512×512 in the original works, as

such this implementation is copied here. GLNet also includes a global branch where the entire image is downsampled to the same resolution as the crop size of the local branch. The training settings are copied from the original paper and as such here a learning rate of $1 \times 10^{-4}$is used for the global branch whilst 2×10^{-5} for the local branch. Adam [60] is also used for optimisation instead of stochastic gradient descent. The global to local implementation of the network is trained where the global network is first trained followed by the local branch in tandem at the second step of training.

7.2.6 Results

Table 7.1 contains accuracy, VRAM and training time statistics for the experiments run on the cityscapes validation and test dataset. The best performing network is the DeepLabV3+ baseline which outperforms the tiling XT implementation by 1.2% on the validation set and 1.5% on the test set. Although it obtains a higher accuracy it takes 1.33× longer per iteration to train and consumes 2.77× the memory. Similarly tiled deeplab without XT consumes 3.23× less memory than the baseline whilst losing 1.9% accuracy on the validation set. Compared to XT the tiled approach is 0.3% less accurate on the validation set whilst only saving 1.43% VRAM. However the tiled approach is 1.08× and 1.44× faster than tiled XT and baseline deeplab. This makes it the second fastest network per iteration when training, only MagNet is faster by 1.37×. PPN does not provide traintime stats and therefore the memory used or the time taken per iteration can not be compared.

Checkpointing, one of the competing VRAM saving techniques, performs 1.0% higher in accuracy than tiled XT but consumes 71% more VRAM. Moreover the time taken per iteration during training is 3.45× that of tiled XT. Tiling XT and checkpointing are combined to prove the XT method can be combined with checkpointing for further memory saving. Through this combination an

extra 14% VRAM is saved making it the most VRAM efficient method, this comes with a drawback of increasing the time per iteration by 2.1×. Lastly adding checkpointing to XT results in a loss of 0.7% accuracy. This is likely due to the slight change in batch normalisation's function due to the multiple forward passes. It should be noted that even though XT combined with checkpointing is the most memory efficient network, adding checkpointing to the tiling network would result in a slightly more memory efficient network. However this is not implemented because it does not garner any novel knowledge.

Finally GLNet, a network specifically designed for high resolution image segmentation, is evaluated. This method achieves the lowest accuracy on the validation set, its accuracy is 16.4%, 15.5% and 15.2% lower than DeepLab baseline, tiling XT and tiling respectively. The memory consumption is 30% lower than the baseline but 94% higher than tiling XT. This network is by far the slowest during training, it is slower than checkpointing the baseline deeplab (previously the slowest method) by 4.5× and is slower than tiling XT by 15.6×. On the test set the worst performing network is MagNet which achieves an IoU of 67.6% which is lower than GLNet by 2.3%, lower than Tiling XT by 6.5% and lower than the baseline deeplab by 8%.

PPN achieves the highest accuracy on the validation set when considering high resolution segmentation methods. It achieves an accuracy of 0.2% higher than tiling XT but its results are not given on the test set, therefore it unclear whether PPN outperforms XT in the final evaluation.
The results from table 7.1 indicate that using checkpointing is a viable technique for a dataset of this resolution and using it on an ultra high resolution dataset such as the LAT dataset in chapter 6 would likely also work. However the time per iteration during training would likely increase proportionally. The technique already produces a noticeable slow down causing it to be a far

DeepLab V3+	Accuracy(IoU)		Traintime Statistics	
	Val (%)	**Test (%)**	**VRAM (GB)**	**Time/iter (s)**
Standard	**76.2**	**75.6**	1.94 (1.0)	2.26
Standard Checkpointing	76.0	75.1	1.20 (0.478)	5.88
XT	75.0	74.1	0.70 (0.361)	1.70
XT Checkpointing	74.3	73.5	**0.60 (0.312)**	3.60
Tiled	74.7	72.7	0.69 (0.356)	1.57
PPN	75.2	-	-	-
MagNet	68.2	67.6	0.93 (0.479)	**1.15**
GLNet	59.8	69.9	1.36 (0.703)	26.5

Table 7.1: Comparison across the Cityscapes dataset of the original implementation of DeepLabV3+ against our best performing tiled adaptation. VRAM usage at training and test time is given in GB for a batch size of 1, with relative memory usage shown in brackets as a percentage. We include results on networks implementing checkpointing showing additional memory savings at the cost of training time. Some test results are omitted due to submission rate limitation on the cityscapes evaluation server. '-' Denotes where values are missing due to the code being unavailable and the paper not specifying the values.

slower technique than tiling or tiling XT. However the technique causes minimal accuracy loss whilst bringing down the VRAM consumption noticeably. The time would likely be 4.5× longer in this example. Moreover deeplab has been shown not to perform well on ultra high resolution images [31]. Tiling XT shows the greatest compromise between training speed, VRAM consumption and accuracy. Despite saving memory it does not suffer from the increased training time drawbacks like GLNet or checkpointing. On further observation combining checkpointing with tiling XT does improve memory saving but the accuracy and training time trade off make this choice questionable in real world applications.

MagNet shows promising VRAM and training time savings when compared to DeepLab baseline but its accuracy falls far below all other methods on this dataset. Similarly GLNet's accuracy also falls far below the baseline and tiling XT, When combined with the extremely prohibitive training time the method is not a viable choice on this dataset. Note that the training time per iteration stats for GLNet do not include the initial phase of training the local branch, therefore in a real world scenario GLNet training is even slower. Additionally its memory savings are notable when compared to the baseline but easily outcompeted by all other methods in this metric.

To further examine why tiling XT outperforms other networks designed for high resolution image segmentation the IoU obtained for each class is presented in table 7.2.

Method	**road**	**sidewalk**	**building**	**wall**	**fence**	**pole**	**trafficlight**	**trafficsign**	**vegetation**	**terrain**	**sky**	**person**	**rider**	**car**	**truck**	**bus**	**train**	**motorcycle**	**bicycle**
Tiling XT	97.9	80.9	**91.6**	**53.0**	**53.8**	60.2	**65.6**	72.4	**93.0**	**70.3**	**94.8**	**82.0**	**60.3**	94.8	60.0	74.2	75.4	**57.0**	70.2
GLNet	96.9	**81.3**	90.8	34.5	51.9	**60.9**	63.1	**73.1**	90.8	58.7	93.5	81.5	27.7	93.2	60.9	71.4	**76.6**	50.4	**70.8**
MagNet	**98.3**	80.7	89.8	48.0	51.6	49.4	44.0	66.1	90.5	55.7	90.0	72.2	47.1	91.8	**67.4**	**78.0**	56.4	45.2	61.8

Table 7.2: The accuracy of high resolution segmentation networks on each Cityscapes class in the test set. Note that the GLNet values are obtained through the the training implementation in this thesis as GLNet was not applied to cityscapes in the original works.

For most classes Tiling XT achieves a higher accuracy than both GLNet and Magnet but there are a few exceptions. For the road, sidewalk, pole, trafficsign, train and bicycle classes XT achieves an IoU of 0.4%, 0.4%, 0.7%, 0.7%, 1.2% and 0.6% lower than the best performing network. For the classes truck and bus it achieves an IoU of 7.4% and 3.8% lower than the best network. Given this data it is possible to conclude that in most cases where tiling XT is not the best method it is only marginally outcompeted aside from the two outlined cases.

Similarly for many classes Tiling XT only achieves a slightly higher accuracy. With the exception of the wall, terrain, rider and motorcycle class. For these classes the method achieves an IoU of 5.0%, 11.6%, 13.2% and 6.6% greater than the next best method. Additionally for the pole, trafficlight, train and person classes MagNet achieves an accuracy much lower than either GLNet or tiling XT. Both MagNet and GLNet appear to perform disproportionately badly on the wall, terrain, rider and motorcycle classes where the majority of

accuracy loss is seen. These classes are rarer and smaller than many of the other larger classes which appear in more images with a greater number of pixels.

Figure 7.1 gives examples of the classes where XT performs well above GLNet and MagNet. Figure 7.1 (A) shows an example of the rider and motorcycle class, MagNet overestimates the size of the bike on the left and gives a noisy and low resolution segmentation for the rider. For the rider on the right side of this image it estimates the shape well with no noise however it fails to segment the back seat of the motorcycle. Similarly GLNet overestimates the size of the motorcycle on the left but gives a good segmentation on the rider. Unlike MagNet, GLNet overestimates the size of the motorcycle on the right side of the image and the resulting segmentation map for the rider is impacted negatively. Tiling XT manages to give a better estimation for the motorcycle on the left and consequently gives the best shape for the rider. For the motorcycle on the left the backseat is segmented correctly and consequently the shape of the rider is qualitatively better. All methods miss subtle details around the front of the motorcycle on the left and fail to segment the rider's arms.

Figure 7.1 (B) showcases a particularly difficult image for all methods, it gives an example where the terrain class features heavily in an image. MagNet and tiling XT manage to segment some of the terrain in the distant background whilst GLNet does not. For the terrain in the foreground GLNet and MagNet underestimate the size of the terrain and segment it as pavement instead. XT gives a better estimation for this area but displays a noisy segmentation.

In both images in figure 7.1, tiling XT displays a better ability to segment small objects or distant objects. MagNet and GLNet often miss or produce small noisy segmentations for distant road signs, traffic lights and poles. Distant bikes and pedestrians are often missed or display low accuracy around

boundaries. All methods display this characteristic but XT appears to display it to a lesser degree.

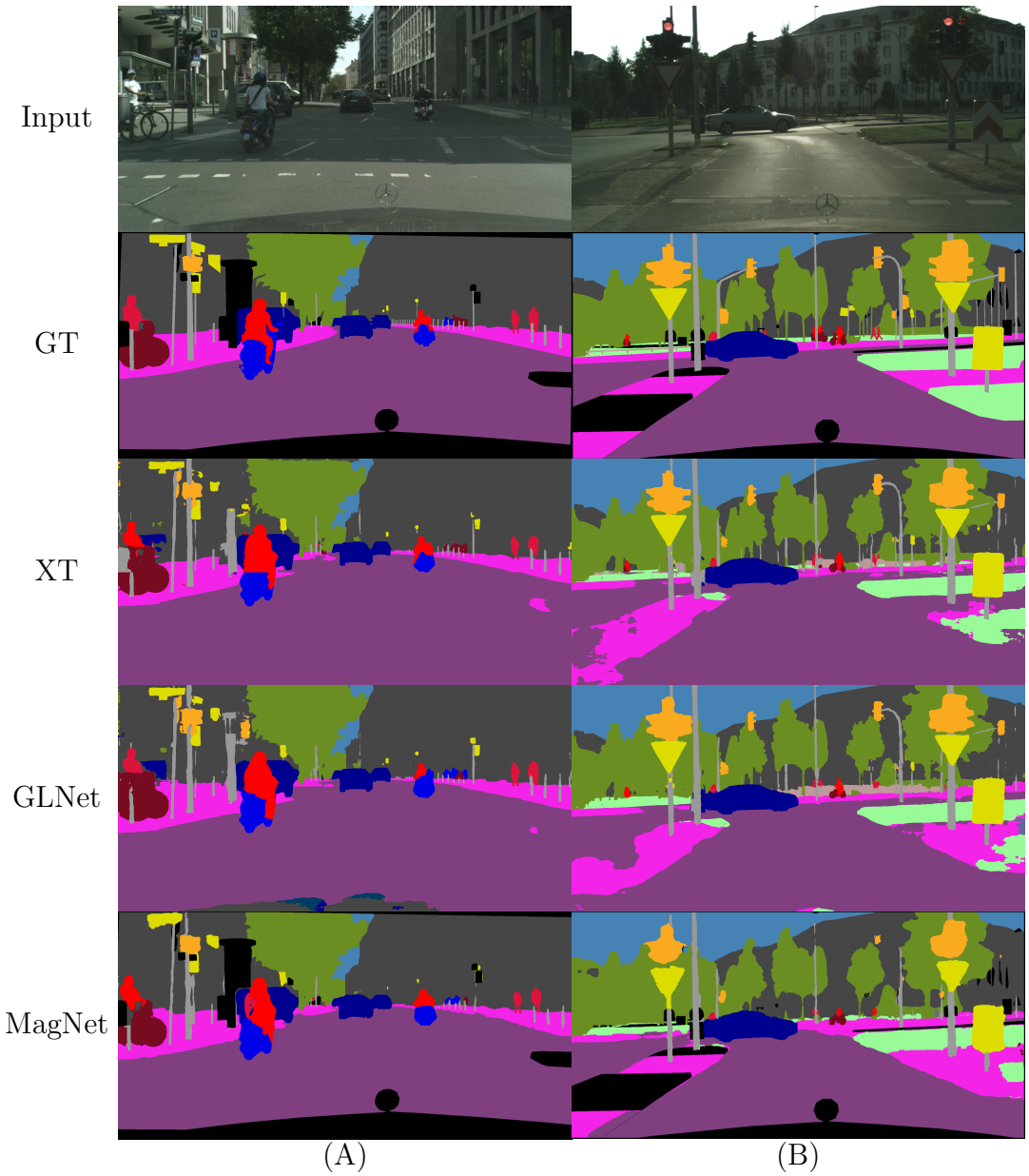


Figure 7.1: *Images from the cityscapes validation set along with ground truth and network outputs from each high resolution segmentation technique. Left (A), showcases an example of the rider and motorcycle class. Right (B) showcases a difficult example which includes terrain and many small and distant objects. Note that the code for MagNet removes the ignored class from the images represented by the black pixels. This gives the image a more refined look but is not an actual network prediction. GLNet and XT display noise in these regions as a result of not replacing the pixels.*

7.2.7 Ablation Studies

This section showcases ablation studies which are used to give a better understanding of XT. The studies show which parameters work best for XT

and consequently give the best final accuracy. They also allow exploration of which components of XT contribute accuracy improvements. Comparisons can be made to the ablation studies in chapter 6, which give insight into whether the relationship between the parameters and accuracy hold across datasets.

Varying Tile Number In chapter 6 the relationship between accuracy, XT and tile configuration was examined through ablations studies, an identical experiment is conducted here. The results are provided in table 7.3. The results show a diminishing returns effect as the tile size is decreased (tile number increases) in regards to VRAM saving. Moving from a tile configuration of 2×2 to 4×4 saves 36% memory however moving from 4×4 to 8×8 saves 14% memory. This effect is more pronounced with the XT variation where the memory use only decreases by 20MB despite quadrupling the tile number. This diminishing returns effect is observed for the hourglass network on the LAT dataset in chapter 6. This appears to be a common property of tiling networks, an explanation of why this is can be found in that chapter.

At lower tile numbers the difference in accuracy between tiling XT and tiling is very small, at the 2×2 tile configuration only a 0.3% accuracy difference is observed. Similarly XT only consumes 10MB more at the same tile configuration. Moving to increasingly small tile sizes (higher tile numbers) shows that XT is able to maintain accuracy better than its naive counterpart, whilst not increasing the VRAM consumption substantially. XT maintains a better accuracy because increasingly small tiles require more context to produce accurate predictions. XT provides this context regardless of tile size because it always shares context between neighbouring tiles. Again an identical effect is observed in chapter 6 for a different network and dataset. The effect observed here is much less pronounced when compared to the LAT dataset.
Overall this ablation study shows that using large tile numbers does save memory but the savings are limited. Given the accuracy results it is easy to conclude

Tile Configuration	mIoU (%)	Train VRAM (GB)
2x2 XT	**75.0**	0.70
2x2	74.7	0.69
4x4 XT	67.6	0.47
4x4	64.3	0.44
8x8 XT	51.4	0.45
8x8	41.0	**0.38**

Table 7.3: Effect of tile configuration with and without XT, given as the meanIoU, along with required VRAM in GB.

that using smaller tile sizes on this dataset is not a good alternative the rate of accuracy lost appears to be higher than those observed in chapter 6.

Varying Target Pooling Size Target pooling size is a key hyperparameter within the XT module because it determines the level of spatial compression that occurs on the featuremaps. The target pooling size is the size of the featuremaps from each tile which is used for global filtering. To explore the relationship between this, pooling size and final accuracy, an ablation study is performed where the target size is varied from 1×1 to 8×8. This size represents the size of each tile post pooling operations. Table 7.5 displays the results.

Target Pooling Size	mIoU (%)
8×8	72.3
4×4	**74.2**
2×2	73.3
1×1	73.3

Table 7.4: Effect of target pooling size within XT on overall network accuracy. The network trained uses a single Resblock with a 5×5 kernel in XT and the tile configuration used for this setup is 2×2.

It appears that the size at which tiles are pooled to is a compromise between conserving enough spatial resolution and not overburdening the resblock within the module. Pooling sizes of 1×1 and 2×2 appear too small and perform equally well whilst a size of 8×8 is too large and performs the worst of all setups. The 4×4 size out competes each by 0.9%, 0.9% and 1.9% respectively. Recalling the results of chapter 6 allows for parallels to be drawn. That chapter showed that a larger tile size (smaller tile number) favours larger target pooling sizes. A similar but not identical relationship is found here, it is likely the case

that the results are slightly different here due to the actual size of the tiles. The network used here is deeper and the tile size at the beginning of the network is smaller. This results in overall smaller tiles being processed by XT.

Consecutive Crosstalk Modules Intuitively it could be reasoned that increasing the number of XT modules used consecutively would also increase VRAM consumption in return for more overall accuracy. Additionally it could be hypothesised that multiple XT modules would result in multiple rounds of sharing global context, which would allow context to travel further across the image. It would also allow context to be shared back and forth between tiles. To explore this relationship an ablation study is carried out to increase the number of modules used consecutively at each level of the decoder. The number is varied from 1-3.

Number of consecutive XT modules	mIoU (%)
1	**75.0**
2	74.2
3	70.9

Table 7.5: The effect of varying the number of consecutive XT modules on overall network accuracy. A tiling configuration of of 2×2 was used in this ablation study.

The results in table 7.5 oppose initial intuition and show that increasing the number of XT modules reduces accuracy. Moreover increasing the number linearly appears to have a non-linear effect on decreasing accuracy. Adding an extra XT module decreases accuracy by 0.8%, whilst adding an extra 2 decreases this by a further 3.3% totalling for a 4.1% accuracy decrease. It is unclear why adding XT blocks creates this effect but once again it could be related to how difficult a network becomes to train by adding a complex module multiple times.

Effect of Attention Understanding how attention placement in the XT block effects accuracy can give further insight into the network's inner workings or at least give clues around them. An ablation study is conducted to investigate the overall accuracy contribution of attention and then the con-

tribution of Pre XT attention and Post XT attention. The contribution of spatial and channel attention separately is not investigated. The results are presented in table 7.6.

Attention Placement in XT	**mIoU (%)**
No Attention	62.5
Pre CA + SA	64.8
Post CA + SA	67.1
Pre and post CA + SA	**67.6**

Table 7.6: The effect of attention and its placement in XT on the accuracy of the final network. The network trained here uses a tiling configuration of 4×4.

Intuitively the least performant network contains no attention in the XT module and achieves an accuracy of 62.5% which is 2.3% lower than adding pre XT attention. Pre XT attention appears to choose important featuremaps and pixels which are shared between tiles in the global filtering stage. Adding attention after the XT module increase the baseline accuracy by 4.6% which is twice that of adding pre attention alone. Adding both together only further increases the accuracy by 0.5%. It might be expected that adding both would at least increase the accuracy to around the sum of the parts (which would sum to 6.9%) however it does not. It is possible that post attention, when used alone, provides some of the effect of the pre attention to filter out less important featuremaps. If powerful enough the attention module can perform this task since it is given all of the processed featuremaps unfiltered by XT. More importantly this ablation study shows that post attention, which should be seen as filtering which featuremaps leave XT, is more important than pre attention. This supports the theory that creating a XT module that is non-residual, too complicated or too disruptive to the flow of gradients causes lower accuracy. Which in-turn could explain why multiple consecutive XT modules reduces accuracy as opposed to increasing it.

7.3 DeepGlobe

In this section the hourglass, found in chapter 4, is applied to a more standard high resolution dataset, DeepGlobe. This tests the hypothesis that Tiling XT can perform well across high resolution datasets and allows it to be compared to other high resolution segmentation networks that run on the same dataset.

7.3.1 Revisiting the Data

DeepGlobe is a challenge dataset of ultra high resolution size, namely it consists of 803 images at a resolution of 2448×2448. The dataset includes land cover satelite images of 7 classes, one of which is not used for official evaluation (as dictated by the authors of the dataset) as this class represents an unknown class.

7.3.2 Experimental Design

The hourglass from chapter 2 trained using similar parameters from chapter 6. The network is trained for 400 epochs with a batch size of 1 at a tile size of 612×612. The learning rate is set to 1×10^{-3} and halved at epoch 200, and halved again at epoch 300. XT is used with a target pooling size of 4×4 as it is shown to be the most consistent so far across tile sizes, networks and datasets.

The dataset is split into 455 training images, 206 validation images and 142 test images to follow the protocol of previous works [31–33]. Although an evaluation server exists, it is not typically used due to the time constraints around waiting for results to be manually accepted. The images in each split are copied from previous works [31–33] to allow fair comparison on the test set. The masks used as groundtruth include the 7^{th} class but the crossentropy loss function is set to ignore this label.

7.3.3 Results

Tiling XT is compared against other state of the art methods in table 7.7, where it is shown that the method presented in this thesis outperforms all other high resolution segmentation methods (at the time of writing). Tiling XT outperforms the most popular high resolution segmentation network, GLNet, by 1.54% and another competitor PPN, by 1.24%. The previous best high resolution segmentation method, MagNet (released 2021) is beaten by a marginal 0.18%.

Method	mIoU (%)	VRAM (GB)
GLNet	71.60	2.72
PPN	71.90	-
MagNet	72.96	**1.97**
Tiling XT	**73.14**	3.46

Table 7.7: The results on the DeepGlobe test set of each high resolution segmentation network. VRAM is given in GB for each network at training time, a '-' denotes where the consumption is unknown. This is for cases where authors did not specify the memory consumption and where code is not available to run.

The most memory efficient network is MagNet which consumes 72.4% and 57.9% less memory than GLNet and Tiling XT respectively. Tiling XT is the most memory inefficient network despite its high accuracy.

7.4 Synthetic Dataset

Simply applying tiling and XT to networks and seeing an increase in accuracy does not demonstrate conclusively that XT is sharing information between tiles. Although chapter 6 shows XT networks don't simply perform better due to more parameters, (by comparing with extra resblocks) it is still unclear whether XT is improving the context available to each tile. Previous sections of this chapter explored the inner workings of XT more through ablation studies. The effect of in and out attention was explored by removing each component at a time and measuring the accuracy. The pooling size was varied to demonstrate its effect on accuracy however, these ablation studies do

not go far enough. This section sets out a study to demonstrate that the XT module enables sharing between tiles.

A logic puzzle is designed in the form of a semantic segmentation problem, the dataset is generated by drawing an image of size 256×256 which is split into a 16×16 grid. A single shape is drawn in any tile of the grid. This shape can either by a circle or a square. In every image at least one shape is drawn, this is a white circle and is the target for segmentation. In the case that there in just one shape drawn the white circle is to be segmented as class 1. A second shape can be draw in an alternate tile in the grid, this occurs with a probability of 0.5. The second shape is a grey square, when it is drawn the white circle is to be segmented as class 2. Figure 7.2 displays multiple example scenarios under this ruleset, blue represents class 1 and red represents class 2.

The image is generated in a way so that no shape information crosses the boundary of the tile it is rendered on, this is to stop any information leaking across. If information were to leak across a boundary it is possible that the grey square could be rendered next to the white circle and inform a tiling network of the presence of the other shape. The experimental design here means that only a network in which information can be passed between tiles, can segment correctly and recognise class 2. A network which can only see a single tile at a time should always score 50%.

Three hourglass networks are trained on the problem. One hourglass receives the entire image as input and the other two are tiling networks, one with XT and one without. The tiling networks use a tiling scheme of 4×4 to match the image tiling scheme and avoid information leak. A results table is not given since the results are very brief, although it is a logic puzzle it is given as a segmentation problem therefore the results are measured using the Dice score metric.

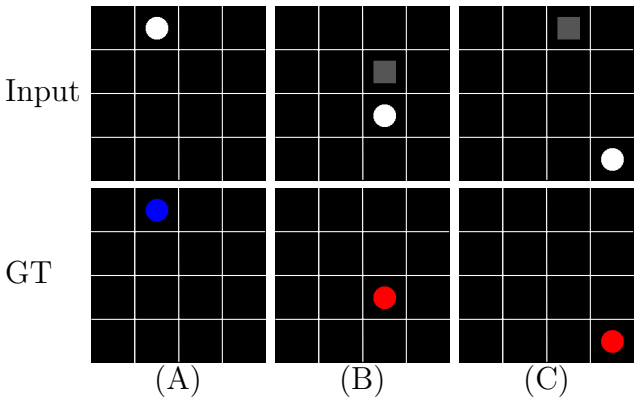


Figure 7.2: *This figure displays 3 example scenarios from the randomly generated dataset. (A) shows the case where only the white circle is rendered resulting in it being segmented as a blue circle (class 1). If the grey square is rendered the circle should be segmented as a red circle (class 2) as shown in (B) and (C). White lines are rendered to illustrate tile boundaries in the* 4×4 *tiling scheme used.*

For the first hourglass the problem is trivial and training produces a network that can segment the problem with 100% accuracy. The second network, tiling alone, performs with 50% accuracy as predicted. It is always able to segment the white circle alone (7.2(A)) but continues to segment the white circle as the blue class even when a grey square appears in a separate tile (7.2(B) and 7.2(C)). Adding XT to create the third network results in an increase of accuracy to 81.3%, although it is not the expected value of 100% it is a big improvement over tiling alone. It is not known why the network does not reach maximum accuracy, but this may be due to segmentation masks not being produced perfectly or simply the rules not being learnt wholly. A future identical study could be carried out, using a network classification, to eliminate the possibility that posing it as a segmentation problem is affecting the results.

7.5 Summary

In this chapter, XT was tested across multiple datasets to evaluate whether the method has applicability across domains. The generic scheme to tile networks and add XT from chapter 6 was applied to a new more popular network, DeepLabv3+, to again evaluate the generalisability of the method.

DeepLabv3+ a popular segmentation network, was trained on the cityscapes dataset using tiling XT. The accuracy and resource statistics were compared to an unmodified baseline DeepLabV3+, which showed a 1.5% decrease in accuracy. In return for 64% reduction in VRAM, and 25% reduction in training time per iteration. Checkpointing was explored and was seen to reduce VRAM consumption by a noticeable margin for very little loss in accuracy, however using this method always at least doubles training time. Another network, GLNet, created specifically for high resolution segmentation was evaluated against and was outperformed in all metrics. XT achieved an accuracy 5.2% higher whilst the VRAM consumption was 49% lower and the training time was 15.6× faster. Similarly, the results from MagNet trained on cityscapes were included where it is seen that XT outperforms this state of the art method by 7.5%. Additionally XT consumes 64.5% the memory of MagNet but is 1.48× slower. A final alternative method was compared against, PPN. This method achieved a validation accuracy of 0.2% higher than XT but due to lack of information or code from the original paper, more comparisons cannot be made. The training speed, VRAM consumption and results on the test set are not provided.

XT maintains accuracy well compared to the baseline but is better suited to high resolution images. Despite providing a lower performance than the competing method checkpointing a strong case is made for avoiding the technique. Implementing the method is technically difficult with many pitfalls, it also includes an accompanying training time increase. Two drawbacks which XT does not suffer from due to its simplicity. Future work should be done to increase the accuracy of XT in this domain, other networks should be adapted to explore whether DeepLab is a difficult use case for XT.

The relationship between optimal target pooling size was explored through

ablation studies, where it was found that larger tile sizes benefit from larger pooling to an extent. This is a similar relationship to that explored in chapter 6. That chapter also found a diminishing returns effect between VRAM saving and tile size reduction, the ablation studies in this chapter reinforce this. The same relationship is found between reductions in tile size and XTs ability to maintain accuracy. Curiously it was shown that adding consecutive XT modules actually decreases accuracy, a linear increase in consecutive blocks appears to have a non-linear effect in decreasing accuracy. It is not obvious why but a possible explanation is that the added modules disrupt learning. Multiple XT modules could result in the network being required to learn a much more complex function. Future work should aim to solve this issue.

Another high resolution dataset called DeepGlobe was evaluated on, using an hourglass network modified with tiling and XT. Results showed tiling XT outperformed the best state of the art high resolution segmentation network, MagNet, by 0.18% and the most popular network of this type, GLNet, by 1.54%.

Lastly, an attempt was made to explore the inner workings of XT, first through an ablation study then through a logic puzzle. The ablation study was carried out in concordance with the cityscapes DeepLab work, the effect of attention was explored by removing it and changing its placement. Removing attention yields the lowest accuracy whilst adding spatial and channel attention to the start of XT increases accuracy by 2.3%. Alternatively adding attention to the end of XT increases accuracy by 4.6%. Using both forms of accuracy increases the accuracy by 5.1%. Combining both does not meet an accuracy value equivalent to the sum of the parts. Nevertheless this ablation study showed post attention is the most important factor indicating that controlling featuremaps leaving XT is most important.

The logic puzzle used in this chapter was designed such that a tiling network does not have sufficient information to score above 50% accuracy. This is because it is a two part problem where the task is to segment a shape as a given class, where the class changes if another shape exists in another tile. This is a simple problem for a network which receives the entire image and therefore has total global context. As predicted the tiling network achieves an accuracy of 50% since it can only see one tile at a time and is unaware of the other shape at the time of segmenting that tile. The network receiving the whole image achieves 100% and XT achieves 81.3%. Although this is a large improvement over tiling alone it is not a 100% score. Future work could examine another experiment where a classification version of the problem is used, this would eliminate the possibility that this lower accuracy is being caused by the logic puzzle being posed as a segmentation problem.

Overall this chapter provides a large amount of evidence that crosstalk does share information between tiles, providing each tile more context. It also proves crosstalks' ability to work as an easy to implement module with other networks. Finally the chapter shows XT is not limited to working on a single dataset, it maintains efficacy across image datasets.

Chapter 8

Conclusion

8.1 Contributions

The main contribution of this thesis is a high resolution image segmentation method, this is achieved through the reduction of GPU memory, the main limitation faced in this task. The thesis adapts the method of tiling and solves its main drawback, reduced image context, through crosstalk. Crosstalk is designed specifically to share information between tiles, resulting in a greater context being provided to each tile. Chapter 3 to 7 provide different contributions which are outlined below.

Chapter 3 contributes 2 datasets, one a volumetric human colon dataset and the other also volumetric but converted to slices. The later dataset is of ultra high resolution in the spatial domain making it a more important contribution since there are only a small number in the high resolution image domain. Each dataset is also given with an accompanying motivation and importance. The chapter contributes knowledge on other high and ultra high resolution dataset as well as popular semantic segmentation datasets These popular low resolution datasets give context around why many CNNs are developed without consideration for memory at training time.

Chapter 4 contributes a study of how data dimensionality effects the performance of a CNN on 3D data, 3 methods are compared. Two networks process slices from the 3D volume however one receives the surrounding 2 slices of a selected slice effectively making it 2.5D. The third and final network is volumetric making it able to process the whole volume at once. This study found that segmenting in 2D is less resource intensive, especially when concerning GPU memory, but achieves a substantially lower accuracy when compared to 3D. 2.5D segmentation performs better than 2D but is not robust when removing slices. Conversely 3D networks are robust to slice removal even when significant parts of the volume are removed. The 3D network demonstrated the ability to generalise to images from other scanners. Overall the 3D networks achieve higher accuracies, display robustness to data removal and generalise the best. For this reason they are an obvious choice, but nevertheless consume excessive GPU memory.

Chapter 5 contributes a study on common techniques used to handle high resolution image segmentation, downsampling and tiling (patching). An evaluation of both methods is given which then becomes a baseline to be used in the next chapter. This chapter shows why the use of these techniques may work in some lower resolution cases, but not in high and ultra high image segmentation.

Chapter 6 contributes the main novel work found within this thesis. The chapter outlines a new scheme for generically converting any common semantic segmentation network into a tiling version of itself. The problems associated with tiling remain and as such a novel module, crosstalk, is introduced. The aim of this module is to share information between tiles. It is designed such that edge artefacts, a negative qualitative phenomena caused by tiles giving differing segmentation at the boundaries, can be eliminated. This is achieved by increasing tile agreement at boundaries, and providing global context which

rectifies incorrect class predictions. This module is designed to reduce VRAM consumption and therefore allows high and ultra high resolution images to be segmented.

Multiple ablation studies are contributed that explore the inner workings of the module and show how it can best be used. A second version of crosstalk is also contributed which is more lightweight but is less performant at lower tile sizes. Finally an evaluation against the techniques in chapter 5 is given.

Chapter 7 uses the technique outlined in chapter 6 to adapt DeepLabv3+ a popular segmentation network. Crosstalk is added as a final change to the network. This network is then trained on cityscapes, a high resolution popular dataset for semantic segmentation. It is shown that XT outcompetes other state of the art high resolution semantic segmentation networks. Although it does not perform better than checkpointing, an alternative method of saving VRAM, it does provide significant benefits over this method. It is faster to train and does not suffer from an overly difficult implementation problem.

A test using an artificial logic puzzle showed that crosstalk does share more information between tiles, as hypothesised. Finally a study on another high resolution satellite dataset, DeepGlobe, is contributed. In this example all other high resolution semantic segmentation networks are outperformed, including the most state of the art.

8.2 Limitations and Suggestions for Further Improvements

Even though crosstalk was shown to share global context and allow the segmentation of high resolution images, it can be further improved. Firstly it

uses an attention mechanism which is effective but inefficient when compared to the newest attention mechanisms. Despite the fact that non-local blocks were used and showed a negative impact on performance, the newest most efficient version of the method has not been experimented with. More ablation studies on the use of modern non-local blocks can be carried out to see the effect.

Even if non-local blocks prove to be sub optimal other more modern versions of the attention mechanism used now exist. Of particular interest are those which eliminate the use of an MLP for channel attention, this mechanism is the most inefficient part of attention. As a final note on attention, the newest literature shows that transformers [65], a network type invented for NLP that has recently moved to computer vision, act as an attention mechanism. Of particular interest is the fact that they take image patches as an input natively, this fits well with the design of the method in this thesis, which uses a tiled structure. Transformers compute affinity between patches, which could act much like an attention mechanism for information sharing between tiles.

8.2.1 Explicit Attention Between Tiles

Attention in the module is never applied directly between tiles, it is only ever applied on a per tile basis to improve the utility of features passed into and out of crosstalk. All information sharing is then conducted using a global filter block, comprising of convolutional layers applied over the aligned tiles. The convolutional layers have the complex task of learning which information should then be shared between which tiles. A more explicit tile attention mechanism could be used to learn which tiles should share information with each other and limit the sharing to these tiles alone. In a trivial case this will be neighbours however in cases where tiles are very small and positional informa-

tion is required i.e. where that tile is in the image, distant tiles will be required.

The between tile attention can be implemented using current normal attention mechanism or perhaps even a transformer, again transformers lend themselves well to this task because they learn dependencies between image patches.

8.2.2 Competing With the Baseline

Although crosstalk outperforms other state of the art high resolution segmentation networks on cityscapes it struggles to match the accuracy of the baseline network it is adapting, on regular resolution data. More work should be done to understand the best methodology for adapting networks with tiling and XT so that they maintain accuracy even at lower resolutions. Further to that more networks should be adapted to a tiling XT scheme to better understand if there is a class of networks for which this solution is not compatible.

When conducting the artificial dataset study found in chapter 7 crosstalk did not reach 100% accuracy as predicted. In future a classification network should be adapted to use tiling and XT and the study should be repeated. Posing the problem as a segmentation problem could be affecting the final results, therefore conducting the outlined work would give insights. These insights would determine whether crosstalk is not performing as intended.

8.2.3 A Multi-scale Crosstalk

Solving the problem of edge artefacts and class correction require somewhat different solutions. Class correction requires very low resolution information which is related to either, positional information with regard to the wider image or, information related to the class predictions in surrounding tiles. Edge artefact removal requires higher resolution spatial information around how the

segmentations have been carried out in surrounding tiles. The higher resolution information is used to align segmentation contours at tile boundaries. In some cases edge artefacts are caused by incorrect class predictions, in such cases the first solution is required.

A more effective crosstalk strategy might process tiles at differing spatial resolutions. This concept already exists in atrous spatial pyramid pooling and pyramid pooling blocks. Modifying one of these blocks and creating a crosstalk which processes multiple resolution could lead to higher accuracy networks since the module would include explicit paths for solving the different problems.

8.2.4 Incorporating More Global Information

Most of the recent papers in the field of high resolution image segmentation use a network which combines a global shallow path and a local deep path [31–33]. The global path usually takes the entire input image at a downsampled rate whilst the local path takes a patch from the input image at native resolution. The global path is given the task of providing global context to each local patch. This strategy is very effective and produces some of the most high performance networks.

The strategy of adding a global path could be added to the method outlined in this thesis, to keep the method memory efficient, this global shallow sub-network would need to be trained with an alternating pattern. An alternate training scheme could be used where every X iterations (Where X is a value that could be derived empirically) the global path is trained. Where one sub-network is being trained the other sub-network would require it's weights to be frozen. Using an alternate training scheme would allow the network to train without increasing the overall VRAM consumption at training time, enabling

higher resolutions.

8.2.5 Guiding Crosstalk Through Intermediary Losses

The goal of crosstalk is to share information such that the globally aligned tiles make a prediction that is coherent. From the experiments in this thesis we can assert that crosstalk improves this through indirect supervision. Downsampling the ground truth and adding an intermediary loss in the middle of each crosstalk module could improve results. This would force the global filters in crosstalk to produce spatially aligned featuremaps which shared information explicitly based on the ground truth.

8.2.6 3D Crosstalk

Chapter 4 shows the effectiveness of 3D networks in comparison to their 2D counterparts, as such maximising their use is key for future success. In order to segment the ever growing size of imaged volumes, a 3D crosstalk is required, which in theory should provide the same benefit for volumes as it has for high resolution images. Beginning with medical image volumes would be the first step since they have smaller dimensions in general. Eventually the plant root cross section dataset can be revisited where the hope is to segment the volumes in 3D provide more segmentation consistency across depth. Of all future works this is the most important, due to the increasing number of volumes being produced by the community and because there are few techniques being developed to address memory limitations in this domain.

8.3 Summary

This thesis contributes a novel method to segment ultra high resolution images by saving GPU memory through tiling. Tiling causes a lack of global context and leads to edge artefacts but does save significant amounts of GPU memory

during training. A module is designed to share information between tiles therefore resolving class confusion and removing edge artefacts with the new found global context. The module is lightweight which results in the overhead being low with regard to memory consumption. Finally this new method is used to train networks on and segment high and ultra high resolution datasets now that the memory constraint has been resolved. Doing so displays state of the art results for some high resolution datasets. The ability to segment high resolution data will increase in importance as imaging devices, from domains such as biomedical imaging, continue to improve.

Bibliography

[1] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Journal of Machine Learning Research*, vol. 15, 2011.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, 1998.

[3] 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*, vol. 115, no. 3, 2015.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 2, 2012.

[5] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8689 LNCS, no. PART 1, 2014.

[6] 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*, vol. 07-12-June-2015, 2015.

[7] 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*, 2015.

[8] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, 2017.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, 2016.

[10] A. Veit, M. Wilber, and S. Belongie, "Residual Networks are Exponential Ensembles of Relatively Shallow Networks," *arXiv preprint*, 2016.

[11] 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, vol. 39, no. 12, 2017.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9351, 2015.

[13] 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*, vol. 40, no. 4, 2018.

[14] l. john, M. Andrew, and C. P. Fernando, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, vol. 2001, no. June, 2001.

[15] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, 2017.

[16] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, 2017.

[17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, 2017.

[18] 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.

[19] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters - Improve semantic segmentation by global convolutional network," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, 2017.

[20] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arxiv.org*, 2017.

[21] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11214 LNCS, 2018.

[22] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[23] Y. He, X. Zhang, and J. Sun, "Channel Pruning for Accelerating Very Deep Neural Networks," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, 2017.

[24] V. Nekrasov, C. Shen, and I. Reid, "Light-weight refinenet for real-time semantic segmentation," in *British Machine Vision Conference 2018, BMVC 2018*, 2019.

[25] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for Real-Time Semantic Segmentation on High-Resolution Images," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11207 LNCS, 2018.

[26] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11217 LNCS, 2018.

[27] P. R. Chen, H. M. Hang, S. W. Chan, and J. J. Lin, "DSNet: An efficient CNN for road scene segmentation," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA ASC 2019*, 2019.

[28] Tao Andrew, Sapra Karan, and Catanzaro Bryan, "Hierarchical Multi-Scale Attention for Semantic Segmentation," 2020.

[29] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin, "Training Deep Nets with Sublinear Memory Cost," *arXiv*, 2016.

[30] I. Krešo, J. Krapac, and S. Šegvić, "Efficient ladder-style densenets for semantic segmentation of large images," 2019.

[31] W. Chen, Z. Jiang, Z. Wang, K. Cui, and X. Qian, "Collaborative global-local networks for memory-efficient segmentation of ultra-high resolution images," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, 2019.

[32] T. Wu, Z. Lei, B. Lin, C. Li, Y. Qu, and Y. Xie, "Patch Proposal Network for Fast Semantic Segmentation of High-Resolution Images," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020.

[33] Huynh Chuong, Tran Anh, Luu Khoa, and Hoai Minh, "Progressive Semantic Segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 755–16 764.

[34] 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 *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, 2016.

[35] M. Everingham, S. M. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, 2015.

[36] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ADE20K dataset," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, 2017.

[37] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raska, "DeepGlobe 2018: A challenge to parse the earth through satellite images," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2018-June, 2018.

[38] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark," in *International Geoscience and Remote Sensing Symposium (IGARSS)*, vol. 2017-July, 2017.

[39] C. Lam, G. Chaddock, L. Marciani, C. Costigan, J. Paul, E. Cox, C. Hoad, A. Menys, S. Pritchard, K. Garsed, S. Taylor, D. Atkinson, P. Gowland, and R. Spiller, "Colonic response to laxative ingestion as assessed by MRI differs in constipated irritable bowel syndrome compared to functional constipation," *Neurogastroenterology and Motility*, vol. 28, no. 6, 2016.

[40] S. E. Pritchard, L. Marciani, K. C. Garsed, C. L. Hoad, W. Thongborisute, E. Roberts, P. A. Gowland, and R. C. Spiller, "Fasting and postprandial volumes of the undisturbed colon: Normal values and changes in diarrhea-predominant irritable bowel syndrome measured using serial MRI," *Neurogastroenterology and Motility*, vol. 26, no. 1, 2014.

[41] G. Major, S. Pritchard, K. Murray, J. P. Alappadan, C. L. Hoad, L. Marciani, P. Gowland, and R. Spiller, "Colon Hypersensitivity to Distension, Rather Than Excessive Gas Production, Produces Carbohydrate-Related Symptoms in Individuals With Irritable Bowel Syndrome," *Gastroenterology*, vol. 152, no. 1, 2017.

[42] J. L. Poulsen, E. B. Mark, C. Brock, J. B. Frøkjær, K. Krogh, and A. M. Drewes, "Colorectal transit and volume during treatment with prolonged-release oxycodone/naloxone versus oxycodone plus macrogol 3350," *Journal of Neurogastroenterology and Motility*, vol. 24, no. 1, 2018.

[43] T. J. Sloan, J. Jalanka, G. A. Major, S. Krishnasamy, S. Pritchard, S. Abdelrazig, K. Korpela, G. Singh, C. Mulvenna, C. L. Hoad, L. Marciani, D. A. Barrett, M. C. Lomer, W. M. De Vos, P. A. Gowland, and R. C.

Spiller, "A low FODMAP diet is associated with changes in the microbiota and reduction in breath hydrogen but not colonic volume in healthy subjects," *PLoS ONE*, vol. 13, no. 7, 2018.

[44] R. A. Bendezú, M. Mego, E. Monclus, X. Merino, A. Accarino, J. R. Malagelada, I. Navazo, and F. Azpiroz, "Colonic content: effect of diet, meals, and defecation," *Neurogastroenterology and Motility*, vol. 29, no. 2, 2017.

[45] D. Withey and Z. Koles, "A review of medical image segmentation: methods and available software," *International Journal of Bioelectromagnetism*, vol. 10, no. 3, 2008.

[46] T. H. Sandberg, M. Nilsson, J. L. Poulsen, M. Gram, J. B. Frøkjær, L. R. Østergaard, and A. M. Drewes, "A novel semi-automatic segmentation method for volumetric assessment of the colon based on magnetic resonance imaging," *Abdominal Imaging*, vol. 40, no. 7, 2015.

[47] B. Orellana, E. Monclús, P. Brunet, I. Navazo, Bendezú, and F. Azpiroz, "A scalable approach to T2-MRI colon segmentation," *Medical Image Analysis*, vol. 63, 2020.

[48] S. Trachsel, S. M. Kaeppler, K. M. Brown, and J. P. Lynch, "Shovelomics: High throughput phenotyping of maize (Zea mays L.) root architecture in the field," *Plant and Soil*, vol. 341, no. 1-2, 2011.

[49] C. F. Strock, H. M. Schneider, T. Galindo-Castañeda, B. T. Hall, B. Van Gansbeke, D. E. Mather, M. G. Roth, M. I. Chilvers, X. Guo, K. Brown, J. P. Lynch, and M. Gifford, "Laser ablation tomography for visualization of root colonization by edaphic organisms," *Journal of Experimental Botany*, vol. 70, no. 19, 2019.

[50] M. P. Pound, A. P. French, D. M. Wells, M. J. Bennett, and T. P. Pridmorea, "CellSeT: Novel software to extract and analyze structured networks of plant cells from confocal images," *Plant Cell*, vol. 24, no. 4, 2012.

[51] S. C. Grossnickle, "Importance of root growth in overcoming planting stress," in *New Forests*, vol. 30, no. 2-3, 2005.

[52] J. Lynch, "Root architecture and plant productivity," 1995.

[53] J. G. Chimungu, K. M. Brown, and J. P. Lynch, "Large root cortical cell size improves drought tolerance in maize," *Plant Physiology*, vol. 166, no. 4, 2014.

[54] ——, "Reduced root cortical cell file number improves drought tolerance in maize," *Plant Physiology*, vol. 166, no. 4, 2014.

[55] P. Saengwilai, E. A. Nord, J. G. Chimungu, K. M. Brown, and J. P. Lynch, "Root cortical aerenchyma enhances nitrogen acquisition from low-nitrogen soils in maize," *Plant Physiology*, vol. 166, no. 2, 2014.

[56] N. Alexandratos and J. Bruinsma, "World agriculture towards 2030/2050. Revision 2012." *Land Use Policy*, vol. 20, no. 4, 2012.

[57] C. Conrad, A. Wünsche, T. H. Tan, J. Bulkescher, F. Sieckmann, F. Verissimo, A. Edelstein, T. Walter, U. Liebel, R. Pepperkok, and J. Ellenberg, "Micropilot: Automation of fluorescence microscopy-based imaging for systems biology," *Nature Methods*, vol. 8, no. 3, 2011.

[58] A. L. Burton, M. Williams, J. P. Lynch, and K. M. Brown, "RootScan: Software for high-throughput analysis of root anatomical traits," *Plant and Soil*, vol. 357, no. 1, 2012.

[59] 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)*, vol. 9912 LNCS, 2016.

[60] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[61] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11211 LNCS, 2018.

[62] P. Goyal, Dollar Piotr, Girshick Ross, Noordhuis Pieter, Wesolowski Lukasz, Kyrola Aapo, Tulloch Andrew, Jia Yangqing, and He Kaiming, "accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," *arxiv.org*, 6 2017.

[63] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local Neural Networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[64] W. Liu, A. Rabinovich, and A. Berge, "ParseNet: Looking Wider To See Better," *Arxiv.org*, 6 2015.

[65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 2017-December, 2017.