

University of Nottingham
UK | CHINA | MALAYSIA

INSTANCE SEGMENTATION OF FRONT DOORS IN MOBILE MAPPING SYSTEM IMAGES

Anna Maria Klimkowska

Nottingham Geospatial Institute, University of Nottingham, UK

Dissertation submitted to The University of Nottingham in partial fulfilment of the degree of Master of Research in Geospatial Systems

August, 2020

ABSTRACT

Scene understanding, through instance segmentation, is one of the most challenging and difficult tasks in computer vision. Instance segmentation is a combination of object detection and semantic segmentation, resulting in assigning each pixel in an image to a given class. This research aims to investigate the capabilities of the Mask R-CNN architecture to detect front doors from images delivered from a Mobile Mapping System. It was achieved by conducting a set of experiments focusing on training the Mask R-CNN with different combinations of training datasets. Front doors are important objects of interest in this work due to their significance in various applications like enriching information of building Level of Details, or recognising door locations for emergency services and people with disabilities. The data used in the training and testing phases included images collected from different locations (cities in the UK and Poland) and of various types (mobile phone, Mobile Mapping System, RGB and gray images). The results of the analysis suggest that increasing the size of the training dataset improves model capabilities for front door detection. Moreover, the analysis of the performance of models trained on datasets from different locations shows that models can be used to detect front door instances regardless of the origin of the training and test datasets. However, despite the satisfactory results, a comparative analysis of the Mask R-CNN and YOLOv3 approaches showed the superiority of the YOLOv3 over Mask R-CNN in the overall model performance. The best data combination used to train Mask R-CNN resulted in an F1-score equal to 0.7 which increased to 0.74 through the implementation of a post-processing step based on a height-to-width ratio threshold. The research carried out in this project shows the potential of the examined method for front door detection from Mobile Mapping Systems.

ACKNOWLEDGMENTS

I would like to thank the Ordnance Survey for supporting this work. Furthermore, dr. Stephen Grebby and dr. Stefano Cavazzi, for their knowledge, motivation, availability, patience and guidance, which were invaluable help given to me.

Additionally, I wish to express my sincere gratitude to my Family for their inspiration, lifelong support, and genuine belief that I could always achieve my greatest dreams, even if that includes moving to the other side of the world to follow my passion. And thank you my Friends. Being surrounded by such wonderful people makes me a fortunate person. I love you all.

Contents

ABSTRACT iv

ACKNOWLEDGMENTS iv

Contents v

List of Figures v

List of Tables vii

1 INTRODUCTION 1
1.1 Research question . . . 2
1.2 Scope . . . 2

2 THEORETICAL BACKGROUND AND RELATED WORK 3
2.1 RELATED WORK . . . 3
2.1.1 Door Detection . . . 3
2.1.1.1 2D imaging-based methods . . . 3
2.1.1.2 3D imaging based methods . . . 5
2.1.2 CNNs for computer vision . . . 5
2.1.2.1 Classification . . . 7
2.1.2.2 Object Detection . . . 8
2.1.2.3 Semantic Segmentation . . . 8
2.1.2.4 Instance Segmentation . . . 9
2.2 THEORETICAL BACKGROUND . . . 9
2.2.1 Introduction to Convolutional Neural Networks . . . 9
2.2.2 Deep Learning-Based Processing . . . 14
2.2.2.1 Instance Segmentation: Mask R-CNN . . . 14
2.2.2.2 Object Detection: YOLO . . . 16
2.2.3 Introduction to Mobile Mapping Systems . . . 18
2.2.3.1 Data available from Mobile Mapping Systems . . . 19
2.2.3.2 Waymo Open Dataset . . . 19
2.2.3.3 Cityscapes Open Dataset . . . 21

3 METHODOLOGY 23
3.1 Introduction . . . 23
3.2 Datasets . . . 25
3.2.1 Training and validation dataset . . . 25
3.2.2 Test Dataset . . . 27
3.2.3 Data Annotation . . . 28
3.3 Experiment Setup and Implementation . . . 29
3.3.1 Training Data Preparation . . . 29
3.3.2 Model Training . . . 30
3.3.3 Evaluation Metrics . . . 31
3.3.4 Software Implementation . . . 32

4 RESULTS 33
4.1 LOCATION AND DATA TYPE . . . 34
4.1.1 Data Type . . . 34
4.1.2 Location . . . 37
4.2 Size of training dataset . . . 37
4.3 Comparing Instance Segmentation and Object detection . . . 39

4.4 Sub-class analysis 41
4.5 Enhanced door detection using Mask R-CNN 42
4.6 Summary 44

5 DISCUSSION **45**

6 CONCLUSION **49**

A Project Management Plan **55**

B Data Management Plan **56**

List of Figures

1 Example of computer vision scene understanding tasks. Top row from left to right: Classification, Semantic Segmentation, bottom row: Object Detection , Instance Segmentation (Waleed, 2018). 6
2 Example illustrating the difference between human and computer cognition. What a human eye sees as a side mirror of a car, for a computer is just a grid of numbers, source: Bradski and Kaehler (2008). 7
3 Examples of different challenges when preforming image classification. 8
4 Example of CNN architecture. 10
5 Example of the convolution step. This step can also be implemented with 1D and 3D input 12
6 Zero-padding techniques. From left: no padding, padding where the feature map size is input/stride, maximum padding where end convolutions are applied on the edges of the input. 12
7 Example of stride hyperparameter where S=2. 12
8 Graphic representation of ReLU activation layer. 12
9 Example of a feature map and a corresponding rectified feature map 13
10 Example of a rectified feature map and the corresponding pooled feature map. The pooling layer implements different filters to help identify parts of the image such as corners, edges, eyes, etc. 13
11 Calculating a pooled feature map from a rectified feature map with parameters: 2x2 filter and stride 2. 13
12 Architecture schema of Mask R-CNN (He et al., 2017). 15
13 Feature Pyramid Network with predictions made independently at all levels (Lin et al., 2017a). 15
14 The RoIAlign schema. 16
15 YOLO system scheme to detect objects seen as a regression problem. 16
16 YOLOv3 architecture (Kathuria, 2018). 17
17 Example of a mobile mapping system: here the system used by Oxford Research Group (Maddern et al., 2017). Left: the sensors location on the RobotCar platform, right: sensor location diagram. Coordinate frames present the direction and origin of each sensor with the convention: x-forward (red), y-right (green), z-down(blue). 18
18 Sensor distribution and coordinate system used in the Waymo mobile mapping system. 20
19 Example of the image labels available in the Waymo dataset. 20
20 Example of the LiDAR labels available in the Waymo dataset. 21
21 Classes with number of annotated pixels (y-axis) and their associated categories (x-axis). 21
22 Example images with annotations. 22
23 Example with 3D bounding boxes for vehicles. 22
24 A flow diagram of the process developed for the purpose of this project. The white boxes refer to the processing steps, the gray with solid borders present the input/intermediate/output files and the boxes with dashed borders cover the evaluation points. 24
25 A map illustrating the locations of data used in this project. 25
26 The sample images used for model training and validation. Top row: Nottingham-mobile phone, second raw: Lublin - mobile phone, third row: Nottingham and Lublin -Google Street View, bottom row: Nottingham - glass doors. 26
27 Left: the Korec Mobile Mapping System; right: The Trimble® MX9 system. 27
28 Examples of images from Abingdon, © Crown copyright 2020 OS. 28
29 Sample images from Oxford, Oxford Copyright © 2016 Oxford Robotics Institute. 28

30 Results for different types of training images collected in Lublin city. Detection performance described based on the: recall, precision and F1-score. 35
31 Example of the detection results for experiments 1 (left column),3 (middle column) and 5 (right column) for selected images, top row: Abingdon © Crown copyright 2020 OS, bottom row: Oxford Copyright © 2016 Oxford Robotics Institute. . . . 35
32 Results for different types of training images collected in Nottingham city. Detection performance described based on the: recall, precision and F1-score. 36
33 Example of the detection results for experiments 2 (left column),4 (middle column) and 6 (right column) for selected images, top row: Abingdon © Crown copyright 2020 OS, bottom row: Oxford Copyright © 2016 Oxford Robotics Institute. . . . 36
34 Example of objects from background class classified as front doors © Crown copyright 2020. 36
35 Precision, recall and F1-score for all the six experiments. In experiments 1,3 and 5 model was trained with data from Lublin, whereas in 2,4 and 6 on data from Nottingham. 37
36 Example of different representations of the same front door instance. 38
37 The training model results based on the number of input training samples. 38
38 Evaluation metrics for experiments 4,7,8. 39
39 Evaluation metrics for the YOLOv3 experiments. 39
40 Example of the detection results for experiments 9 (top row) and 10 (bottom row), left and middle column: Abingdon © Crown copyright 2020 OS, Right column: Oxford Copyright © 2016 Oxford Robotics Institute 40
41 A comparison between Mask R-CNN and YOLOv3. 40
42 Evaluation metrics for experiment 7, experiment 7 after the application of the width-height ratio method and the YOLOv3 results from the same set of data. . . 43
43 Top row: examples of shop windows from test datasets (© Crown copyright 2020 OS) which features resemble doors from training datasets (bottom row). 47

List of Tables

1 A list of convolution neural network layers along with their parameters and hyperparameters (adapted from Yamashita et al. (2018)). 10
2 Main Mobile Mapping System sensors and its characteristics 19
3 Image statistics. The collection method (and type) and number of images used for the training and validation steps. Images illustrating glass doors were collected only in Nottingham, but used in both locations. The table also presents the distribution of the data between the test and validation steps. 26
4 Number of test images with respect to the location. 27
5 The dataset configuration used to train the Mask R-CNN model for each of the eight experiments 30
6 The hyperparameter setting for the Mask R-CNN model training. 31
7 Hyperparameter setting for YOLOv3 model training. 31
8 Summary results of all the experiments. 33
9 The Settings for tests 1-6 conducted to assess the influence of training data type and location on front door detection. 34
10 Detection results for each of the experiments with division into type of doors detected and type of objects from the background class which were incorrectly classified as doors. 41
11 Comparison of the evaluation metrics before and after implementing the width-height ratio screening (Experiment 7 enhanced). 43

Acronyms

ANN Artificial Neural Network.

CNN Convolutional Neural Network.

DL Deep Learning.

FPN Feature Pyramid Network.

GPS Global Positioning System.

GSV Google Street View.

INS Inertial Navigation System.

LiDAR Light Detection and Ranging.

ML Machine Learning.

MMS Mobile Mapping System.

RoI Region of Interest.

UAV Unmanned Aerial Vehicle.

1 INTRODUCTION

Thanks to recent advances in the field of data collection from street-level mapping systems, along with the continuing development of deep learning methods in the domain of computer vision, a growing number of new applications have appeared. One of the notable uses of the deep learning approach is object detection and the segmentation of images from Mobile Mapping Systems in self-driving and smart car applications. Visual scene understanding in the field of autonomous driving using semantic segmentation was investigated by Hofmarcher et al. (2019) and Ha et al. (Ha et al., 2017), while object detection for self-driving vehicles was presented in the works of Zhou et al. (2020) and Feng et al. (2020). The development of deep learning-based computer vision tasks and increased access to data has led to an expansion in the field of geomatics, allowing the development of a variety of deep learning method based applications in mapping and geographic information systems (GIS). One of the most common uses of deep learning methods and ground-level data in geospatial systems is building modelling and reconstruction (Bacharidis et al., 2020; Liu et al., 2017).

Front door detection and segmentation from mobile mapping platforms may help improve building models with higher Level of Detail (LoD), as the front door representation in these images is more visible compared with images acquired from airborne platforms. What is more, detection and later localization of front door instances can improve management and logistics for delivery services and waste management companies. Increased knowledge of types of doors (such as automatic or revolving doors), their width and information regarding the existence of steps/ramps in front, might also be helpful for disabled people to facilitate easier movement around a city or town. More traditional approaches of mapping front doors can be time consuming and laborious if manually surveyed or digitized on images. Therefore it is essential to find a method which allows for more efficient as well as more robust front door extraction from images.

Deep Learning (DL) on imagery owes its success to Convolutional Neural Networks (CNN) and is used in computer vision to help machines understand scenes obtained via one of these deep learning tasks: image classification, object detection or semantic and instance segmentation. Mask R-CNN is a well-established CNN architecture used for instance segmentation, which task is not only to performs object detection, but also to classify images on a pixel level, resulting in both a bounding boxes and a masks created for detected objects of interest. The advantage of instance segmentation over semantic segmentation is that the first method treats multiple objects of the same class as distinct individual instances and not as single entities. This is particularly useful when a segmentation product subsequently used in another process; for example, to determine the location of individual objects within one class.

This project investigated the feasibility of instance segmentation of front doors in images acquired by a MMS, by conducting various experiments assessing the influence of training data type and location on the door detection capabilities. The idea is to use a CNN based-approach in conjunction with MMS images in order to make door detection more efficient and automated. In this study, the term front door refers to external door instances which are part of a building (private ownership, students dormitories, shop doors), but does not include entry to a garage, patio doors or doors to a garden path.

1.1 Research question

The research question is:

To what extent is Mask R-CNN suitable for instance segmentation of doors from street-level images?

The capability of Mask R-CNN for front door detection likely depends on many aspects, most notably the dataset used to train the model. In particular, this includes the size and nature (i.e., composition) of the training data. Moreover, the transferability of a model trained on one training dataset to detect doors in another location is also debatable. The objectives listed below should help to provide an answer to the main research question:

- *To determine the usefulness of deep learning based instance segmentation in extracting door instances from street-level images*
- *To determine the influence of the amount of training samples on the ability of a trained model to extract door instances*
- *To assess the effects of different types of training data on front door detection*
- *To evaluate the robustness of a trained model in detecting front doors in independent locations*
- *To compare the performance of instance segmentation and object detection for front door detection from street-level images*

1.2 Scope

The goal of this project is to determine whether instance segmentation can be used for front door detection and, if there is further potential in pursuing this avenue. The focus is to investigate and assess the effect of model training on the detection capabilities, and the optimisation of the hyperparameters is beyond the scope of the present study. The presented work offers three contributions to the field of research on mobile mapping image segmentation for front door detection. Firstly, it examines the influence of various configurations of training data on segmentation results. Secondly, it explores the suitability of Mask R-CNN for front door detection. Lastly, it compares the results of Mask R-CNN with results obtained from training the YOLOv3 architecture.

2 THEORETICAL BACKGROUND AND RELATED WORK

This chapter is divided into two sections. The first section provides an overview of the methods used for door detection from 2D and 3D data, and the applications of Convolutional Neural Networks (CNNs) in computer vision, with a focus on street level images. The second section then provides a theoretical description of the techniques used in this thesis.

2.1 RELATED WORK

2.1.1 Door Detection

Door detection is an important consideration for studies investigating both indoor and outdoor environments. In general, previous studies on this topic can be broken down into two main groups depending on the data acquisition method: 2D color imaging (digital cameras) and 3D imaging (photogrammetric methods or laser scanning). An undeniable advantage of using methods based on 2D color images is the low cost and versatility of the data acquisition. On the other hand, to increase detection precision and to improve the reliability of door detection, it is common to use solutions consisting of a combination of 2D and 3D data.

2.1.1.1 2D imaging-based methods

It is believed that no previous work exists on front door detection from images delivered by mobile mapping platforms although studies on front door detection from ground-level images collected from other 2D platforms are available. The most significant difference in how the front doors appear in these two cases is their size in the image and the amount of context surrounding the doors. Moreover, studies using images other than those collected by MMSs predominantly focus on whole building facade reconstruction and modelling, and include detection of different objects like windows or roofs, rather than specifically doors.

The only study that specifically addresses front door instances detection found within 6 feet of a camera, is a student report from Stanford University (Juenemann et al., 2010) in which a combination of Hough Transform, Canny edge detection and fuzzy logic algorithms was used to find objects of interest. The method was tested on a variety of door types, including rectangular, open, multiple or French. The highest detection score, 90%, was obtained for rectangular doors whereas the lowest detection score was obtained for French and multiple exterior doors, with a correctness rating of 38% and 66% respectively. However, it is important to mention that the number of samples was relatively small, with only 4 samples of French doors and 19 of multiple doors. Based on these findings, it is clear that the number of training images and composition of a training dataset can have a significant impact on door detection capability.

Whole building facade modelling and reconstruction, where front door detection is only a part of the process, is presented in Schmitz and Mayer (2016). Their work employed a convolution network-based approach for the semantic segmentation of facades. Their experiment was conducted on a dataset from eTRIMS (Korc and Förstner, 2009). The model training was performed on 50 images, and the accuracy metrics for door detection were very good (95%). However, a limitation of the method was the misclassification of windows as doors. Deep learning-based semantic segmentation of building facades was also the focus of a study by

Fathalla and Vogiatzis (2017). The methodology presented in this work consisted of implementing a VGG-model used for pixel-wise classification, followed by optimization based on the Restricted Boltzmann Machine. Even though the overall results obtained in this study were on-a-par with results achieved in related works, the authors drew attention to the fact that minority classes like doors or chimneys could be a bottleneck for these types of algorithms. These classes are sometimes excluded from processing, affecting the final accuracy. Gadde et al. (2017) developed another method for building facade segmentation . In contrast to previous work, where image segmentation was based on deep learning techniques, the approach presented in this study was based on the training of boosted decision trees using auto-context features. The method was tested on both 2D and 3D data. According to the paper, the method delivered superior results to other studies in this domain, even though it largely ignored domain knowledge, such as the relationship between balconies and windows.

A considerable body of literature on door detection and localization in indoor environments already exists. This has largely been driven by an increasing demand for a variety of applications such as navigational assistance (Tian et al., 2013) and 3D building reconstruction (Díaz-Vilariño et al., 2017). Many of these cases are associated with mobility restrictions for disabled people or intelligence systems allowing safe navigation through indoor spaces. In their research, Yang and Tian (2010) focused on detecting doors from color images collected from a wearable camera connected to a mini-laptop. This was done by extraction of corners and lines, followed by the detection of the vertical and horizontal segments that create a door frame. Their approach used a Canny (Canny, 1986) edge detector to locate edge candidates and a method based on global and local curvature properties (He and Yung, 2008) for corner identification. Despite demonstrating a good overall detection ratio, when corners or edges were occluded, retrieving a valid door model was found to be challenging. Several methods have been reported in the literature which use DL methods based on CNNs for internal door detection. The reasons for adopting a CNN approach for internal door detection is their ability to accommodate a certain degree of illumination, deformation and transformation variation. These properties led Chen et al. (2014) to solve the problem of false positives presented in Yang and Tian (2010) by employing a CNN-based approach. Nevertheless, they faced difficulties with the appearance of a large number of undetected doors from images acquired by a robot platform. In Llopart et al. (2017) the aim was to detect internal doors in real-time. For this reason they decided to employ a method based on CNNs. They developed a novel approach for detecting doors and handles from images acquired from a Kinect camera, by fusing CNN and several vision based techniques. The reason for implementing a CNN approach was to narrow down the search area in terms of detecting door candidates, while the vision based techniques were used to detect handles in new regions of interests. A recent study by Haque et al. (2020) implemented the You Only Look Once (YOLO) approach (Redmon et al., 2015) to detect doors and windows (macro-features) from Unmanned Aerial Vehicle (UAV) captured images.

A key limitation surrounding exterior front door detection from 2D images (as presented in this section) is the lack of focus on more detailed door detection-only techniques, which would aid the identification of the most appropriate method for this particular application. Moreover, none of the algorithms used in the presented literature were tested on images collected from MMSs, where external door instances expose different characteristics compared to those where the same door comprises part of a whole building facade. Additionally, all the data used in the processing pipelines presented in the literature review were taken from benchmark datasets and not tested on independent data.

The existing research presented in this section related to door detection in indoor environments from 2D images reveals the shortcomings of these methods, especially in delivering a significant number of false positives. The main reason for this lies in the lack of structural information related to the scene (i.e. wall location) and in the appearance of objects with sim-

ilar characteristics to doors, such as corresponding size or a rectangular shape (e.g. windows, furniture, paintings).

2.1.1.2 3D imaging based methods

Adding 3D information from laser scanning, point clouds generated from images or RGB-D (RGB-Depth) images is a potential way of increasing the detection precision or reliability of door detection. For this reason, systems operating in an indoor environment are often equipped with 3D sensors, sometimes supported by 2D color images. Moreover, in a complex urban environment, 3D data finds its application again, mainly in the modelling and reconstruction of buildings and their facades, and thus in the detection of front doors.

Front door detection as part of building facade modelling from LiDAR and image data is part of the research presented in Becker and Haala (2007). In their work, they used a cell decomposition method to extract door and window candidates from terrestrial LiDAR data and airborne images. Although it focused on the reconstruction of the entire facade, the processing and analysis presented in the paper described the detection of windows. A limitation was the problem of edge and corner detection from point clouds due to the laser beam splitting at the object edge. Bacharidis et al. (2020) implemented DL networks to reconstruct 3D building facades based on image segmentation and single image depth prediction. The proposed technique was developed to detect the structural elements of facades and generate a point-cloud. The results presented in this study seem promising in terms of the segmentation of the entire point cloud/image area for building facade reconstruction. Nalani and Perera (2014) presented a method used to recognize and reconstruct facade objects in 3D using MMS point clouds. The core of the approach was based on a primitive shape-based algorithm. To identify primitive shapes against other pre-defined primitive shapes, the Douglas-Peucker algorithm was implemented. However, the method suffered from a problem with occlusion of the lower parts of the building by objects such as cars or vegetation.

Door detection from 3D data is also a topic of research for indoor environments. In this regard, the literature covers a range of different scenarios, such as open, closed or semi-open doors. Quintana et al. (2018) proposed a method for detecting doors from 3D laser scanned data and a color camera. In the experiment, the color (i.e. RGB or HSV) and the door geometry (i.e. XYZ coordinates) were retrieved from a color camera and a set of 3D laser scanners respectively. The combination of the color and geometry allowed for object detection despite color differences and occlusion related to varying lightning conditions and different scanning locations. Based on their research, these authors highlighted limited detection ability when the doors and wall areas were coplanar and there was minimal color contrast.

2.1.2 CNNs for computer vision

Computer vision is a field which aims to teach machines to 'see', that is, to extract meaningful information from images or videos (Keiser, 2017). In a geospatial context, computer vision is used in many different applications, such as satellite imagery classification (Borra et al., 2019; Lu and Weng, 2007), the classification (Kyrkou and Theocharides, 2019) and segmentation (Zhao et al., 2019) of images from UAV or the multi-classification of CCTV images (Sirirattanapol et al., 2019). Methods of computer vision are utilised not only to retrieve information relating to images from different platforms, but also sensors with different spectral and spatial characteristics. In the geospatial field, fusion of computer vision processing tools with a wide spectrum of remote sensing data can be found in many applications like hyperspectral (Tu et al., 2019) and multispectral (Su et al., 2020) image classification or point cloud segment-

ation (He et al., 2019). This section briefly introduces computer vision tasks with emphasis on methods centered around DL models, as these are currently recognised as state-of-the-art in the computer vision field. Although many other applications exist, such as medical recognition, motion capture, fingerprint recognition, autonomous vehicles (and many more), this study focuses on cases using data collected from street-level for urban environments. For a in-depth review of computer vision techniques see Szeliski (2010) and Bradski and Kaehler (2008).

One of the tasks which remains challenging in the field of computer vision is scene understanding. The main aim of scene understanding is to extract semantic relationships and patterns through perception, analysis and interpretation of data. This task is divided into different subtasks based on the implemented technique and the expected results: Image Classification, Object Detection, Semantic Segmentation and Instance Segmentation. Figure 1 presents these subtasks:

- Image Classification: There is a balloon in the image. It predicts the probability of whether an input image belongs to a class, but does not provide pixel-level information.
- Object Detection: There are seven balloons in the image at these locations. It starts to account for objects that overlap, but does not provide pixel-level information.
- Semantic Segmentation: These are all the pixels that belong to the balloon class (class prediction for each pixel in an image)
- Instance Segmentation: There are seven balloons at these locations, and these are the pixels that represent each of them (a combination of the Object Detection and Semantic Segmentation subtasks).The objects are located with pixel-level accuracy.

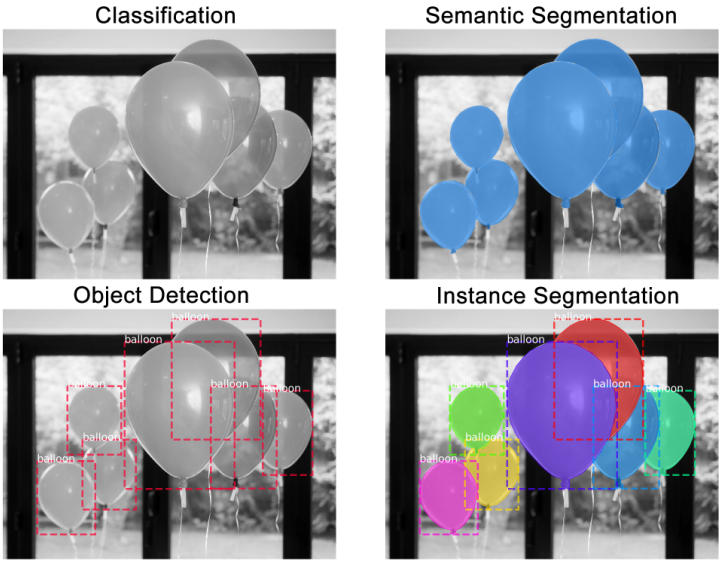


Figure 1: Example of computer vision scene understanding tasks. Top row from left to right: Classification, Semantic Segmentation, bottom row: Object Detection , Instance Segmentation (Waleed, 2018).

Before explaining the different tasks associated with Convolutional Neural Networks (CNNs) in computer vision, it is important to understand how a computer sees an object within an image (Figure 2). Where a human sees a car, a computer sees one large 3D (RGB image) or 2D (gray image) grid of numbers, which may differ significantly depending on the viewpoint, brightness, daytime, scale changes or background. For this reason it is important to carefully analyze the problem to be solved and the factors that may affect the test results.

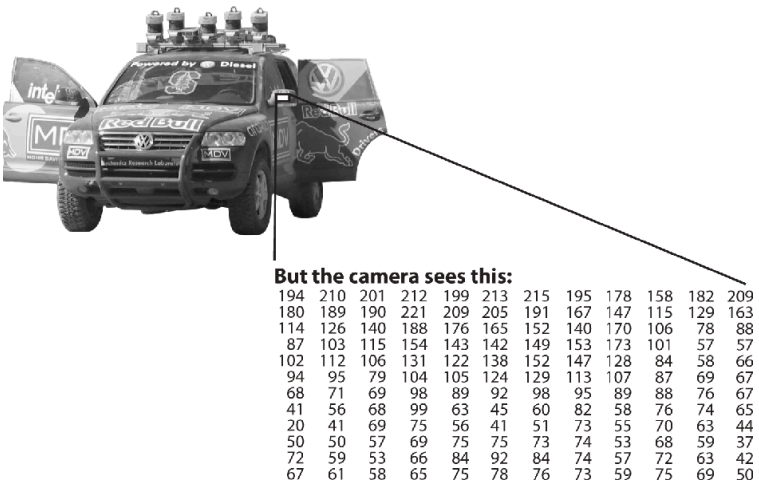


Figure 2: Example illustrating the difference between human and computer cognition. What a human eye sees as a side mirror of a car, for a computer is just a grid of numbers, source: Bradski and Kaehler (2008).

2.1.2.1 Classification

An image classification task requires the computer to assign one label (from a given fixed set of categories) to an input image. This is one of the basic computer vision tasks, which is used in wide range of applications, including urban tree species classification (Hartling et al., 2019) or land cover mapping (Zhang et al., 2019; Zhang et al., 2018). Even though the task appears to be simple for a human to perform, it has many challenges from the perspective of computer vision algorithms.

Below is a list of challenges related to image classification, along with the Figure 3 that visually represents each of them:

- Viewpoint variation. Each object can be mapped differently in the image, depending on its orientation with respect to the camera.
- Scale variation. The object of interest very often appears to be a different size in the real world and in the image.
- Occlusion. In many real world scenarios, many objects can be occluded and meaning only a very small part of it is visible (i.e., relatively few pixels).
- Deformation. Often the object of interest may be distorted in extreme ways.
- Background clutter. In some scenarios, the object of interest may blend with its background making it hard to distinguish.
- Illumination conditions. Illumination effects can have a drastic influence on the pixel level.
- Intra-class variation. The objects belonging to one class may vary significantly, such as a vehicle, which can be represented by a car, bus, truck, motorbike or bicycle.

Image classification is often referred to as a *data-driven approach*. This means that instead of writing an algorithm which specifies the appearance of objects in a certain type of class, the computer is fed with many examples of each class. It allows the machine to learn about the visual representation of each class based on these examples.

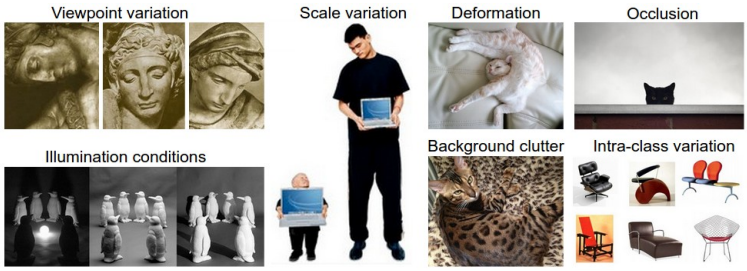


Figure 3: Examples of different challenges when preforming image classification.

Ground level image classification for urban environments is presented in Law et al. (2017). The authors of this work used Google Street View (GSV) images and 3D models generated from GSV images to train the Alexnet CNN (Krizhevsky et al., 2012) architecture to evaluate the street frontage quality of Greater London. The city of London was also the subject of investigation in Chen et al (2020) where the Places365 CNN model (B. Zhou et al., 2017) was fed with geotagged images from Flickr to gain knowledge of the characteristics of the local urban environment of the city. Ibrahim et al. (2019) used GSV images and proposed a method for recognising visual and weather conditions using a deep learning WeatherNet model. The resulting model allowed for the identification of foggy, rainy, snowy or clear weather conditions, as well as the time of the day.

2.1.2.2 Object Detection

Object detection is another task in the field of computer vision scene understanding. In general, object detection involves the classification of images using a localization method which makes a distinction between multiple classes. The output of this task is one or more bounding boxes along with a class label for each of them. A comprehensive survey on development of object detection tasks with an explanation of the components comprising this deep learning technique, detection frameworks, training strategies and evaluation metrics is presented in Liu et al. (2020) and Zhao et al. (2019).

Object detection is probably the most commonly used of the various CNN computer vision methods in scene understanding from street-level images. Zhang et al. (2018) applied deep learning to GSV images in order to identify utility poles with crossarms. In this experiment, RetinaNet, (Lin et al., 2017b), a one stage object detector, was used to detect utility pole candidates and a modified brute-force-based line-of-bearing approach was employed to calculate the position of objects. Hebbalaguppe et al. (2017) experimented with different methods to detect telecommunication infrastructure from GSV images, including deep learning-based Faster R-CNN (Ren et al., 2015). The detection step was followed by a location estimation, achieved by implementing a triangulation method from two consecutive locations. A very similar problem was addressed in Krylov et al. (2018). The approach consisted of three steps. Firstly, Faster R-CNN was applied to detect street lights and telegraph poles from GSV images. Next, a CNN based monocular depth estimation was calculated, according to the work of Laina et al. (2016). Finally, an object of interest location was retrieved from the custom Markov field model.

2.1.2.3 Semantic Segmentation

Semantic segmentation is a task in which each pixel in an image is given a label corresponding to a certain class, but without differentiation between instances of objects. Semantic segment-

ation is a crucial tool in scene understanding as it enables the decomposition of parts of an image into meaningful information and allows an understanding of a scene at a more granular level.

Yang et al. (2019) applied a highly fused convolutional network built from downsampling, fused upsampling and multiple predictions. This method involves the performance of semantic segmentation on different datasets, including CamVid which is The Cambridge-driving Labeled Video Database and ADE20K from MIT (this dataset includes scenes from the urban environment at street level). Lin et al. (2020) developed a method which implemented local and global contextual information in single network to perform semantic segmentation on images from CityScape. Huang and Yang (2017) demonstrated a context module using the Fully-Convolutional Network and its performance was evaluated on the Mapillary Vistas Dataset (MVD), a street-level imagery dataset.

2.1.2.4 Instance Segmentation

The last task described in this section is instance segmentation, which is a combination of object detection and semantic segmentation. This method differs from the latter in its ability to distinguish between individual candidates within one class (this does not occur in the case of semantic segmentation).

As a relatively new method, instance segmentation performance on ground level images has not yet been thoroughly investigated. In their study, Liu et al. (2018) introduced a path aggregating network (PANet) for the instance segmentation task. Their work indicated that information propagation in the Mask R-CNN (He et al., 2017) approach can be further improved by bottom-up path augmentation. The developed method was successfully trained and tested on two street level datasets, CityScape and MVD, to detect object representing classes provided by these benchmark datasets.

In this section the most important tasks of scene understanding have been presented. Although the described methods, especially those related to deep learning based image segmentation, are relatively new, since 2019 a further novel approach has come to the fore, which is a fusion of two of the segmentation methods described above. The idea behind the new method is to push the segmentation systems further so they are more useful for real-world applications. Kirillov et al. (2019) unified semantic and instance segmentation and proposed the so-called *panoptic segmentation* task. In this task, all of the pixels in an image need to be classified and assigned a class label. Moreover, it is necessary to identify to what instance of that class the pixels belong. As a result, each pixel has two values associated with it: a label and an instance number. Examples of panoptic segmentation on street-level images can be found in Xiong et al. (2019) and Mohan and Valada (2020).

2.2 THEORETICAL BACKGROUND

2.2.1 Introduction to Convolutional Neural Networks

This section introduces the fundamental concept of Convolutional Neural Networks (CNNs) and their application in object detection and image segmentation. Whilst it is assumed that the reader has basic knowledge of Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs), more details on these topics can be found in Goodfellow et al. (2016).

A CNN is a form of Artificial Neural Network (ANN) with the design inspired by the human brain. The inspiration for the implementation of CNNs was the work of the Nobel Laureates Hubel and Weisel, who investigated the vision systems of mammalians (starting with cats (Hubel and Wiesel, 1959)). Through their work they found that some neurons make strong responses to specific edge-like patterns and weaker responses to others. Human brains are composed of millions of neurons, which, in the case of ANNs are represented as weighted nodes which read an input, process it and generate an output, which is later used as input for another node.

The CNN was created to process data represented by multiple arrays. The network takes advantage of a natural signal by using shared weights, pooling, local connections and the multiple layers. A CNN, as its name suggests, uses mathematical operations of convolution in at least one layer. These networks have been widely used in computer vision due to their proven superior performance in object detection, image classification or image segmentation, in comparison to traditional methods (O'Mahony et al., 2019). The architecture of each CNN network can vary greatly and is often adjusted to solve specific problems. Although they do follow a general structure, due to the rapid development and constant improvement of CNN architecture it is difficult to find an optimal approach. A CNN architecture is built from many different layers, each responsible for a different action (Figure 4). A list of convolution neural network layers along with their parameters and hyperparameters is presented in Table 1.

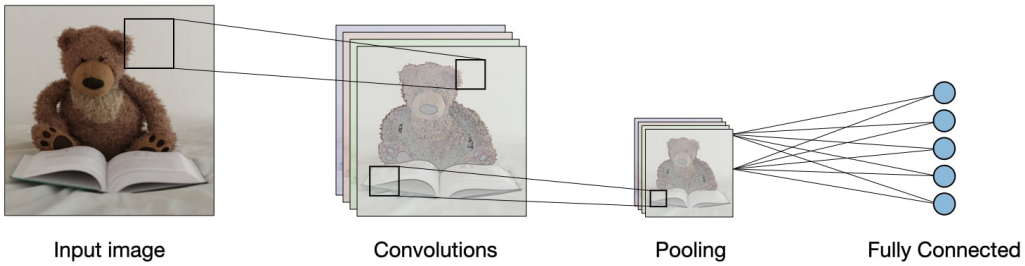


Figure 4: Example of CNN architecture.

Table 1: A list of convolution neural network layers along with their parameters and hyperparameters (adapted from Yamashita et al. (2018)).

Layer type	**Parameters**	**Hyperparameters**
Convolution layer	Kernels	Number of kernels, kernel size, padding, stride, activation function
Pooling layer	None	Filter size, padding, stride, pooling method
Fully connected layer	Weights	Activation function, number of weights
Others		Learning rate, optimize, loss-function, epochs, weight initialization, regularization, dataset splitting, model architecture

Before describing the layers composing the CNN architecture in more detail, it is necessary to define the related terminology. During the training phase, a process variable which is learnt automatically is called a *parameter*. Variables which are specified before the training process and can be tuned to improve training performance are named hyperparameters. A *kernel* (also called a filter) describes the sets of learnable parameters used in convolutional operations.

Convolution layer

First it is important to consider the convolution (Conv) layers through which the input is passed. These layers play a key role in the construction of the Convolutional Network block as they do most of the computational heavy lifting and are responsible for feature extraction. Typically these layers are a combination of linear and non-linear operations, which comprise the convolution and activation functions respectively.

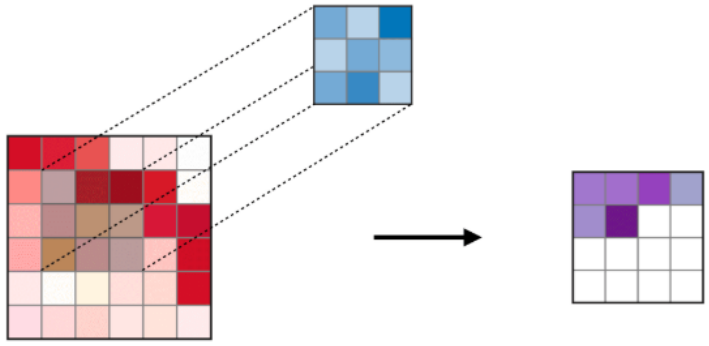
Convolution

The role of convolution is to detect the local conjunction of features from previous layers. This is achieved by convolving an input (an array of numbers also called a tensor) with small arrays of numbers called kernels. A kernel can be understood as a learnable filter (window) which slides (convolves) across all the pixels of the input image producing as an output a new feature map (also called an activation map). The output of the convolution step is presented in Figure 5 where input, *I*, of size 6x6 (red square) is convolved with a kernel of size 3x3 (blue square) with the resultant output, *O*, of size 4x4 (purple square). The convolution step is repeated by applying multiple kernels to create a number of feature maps. Using various kernels allows the detection of different features within the input image. Convolution operations are defined by two key hyperparameters: the number and size of the kernels. The size of a kernel is usually 3x3, but can sometimes have a size of 5x5 or 7x7. The number of kernels used in convolution operations is optional but determines the depth of the feature map output (Yamashita et al., 2018). Figure 5 demonstrates the fact that the center of a kernel does not overlay the elements located on the edges of the input. Therefore the output feature map is reduced in size compared to the width and height of the input data. This issue can be addressed by padding, *P*. In the majority of convolution operations, zero-padding is used. This techniques adds columns and rows of zeros on each side of the input so that the kernel center can align with the most distant element of input data. Such a method allows the preservation of the same in-plane dimension of the output. Figure 6 illustrates three versions of the zero-padding technique.

The position for two successive kernels is defined by a hyperparameter called stride, *S* (Figure 7). Usually, the stride value is 1, nonetheless strides of greater size are also used to attain downsampling of a feature map. Downsampling can also be achieved by pooling operations (these are described in more detail below). Once the feature map is calculated, it is next passed through a non-linear activation function.

Non-linear activation function

In order to introduce non-linearity to the network, that is, to allow it to learn the complex relationships in the data, the activation function is needed. In the past, functions such as sigmoid or hyperbolic tangents were utilised. Commonly, a Rectified Liner Unit (ReLU) is used due to its rapid training and the fact that it mitigates issues related to the vanishing gradient



.

Figure 5: Example of the convolution step. This step can also be implemented with 1D and 3D input

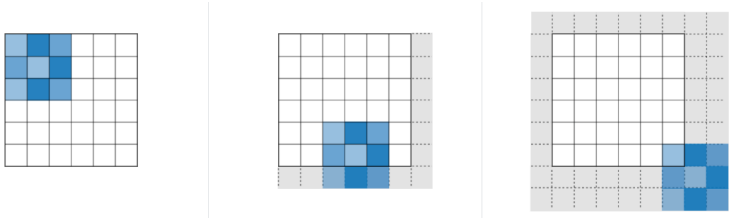


Figure 6: Zero-padding techniques. From left: no padding, padding where the feature map size is input/stride, maximum padding where end convolutions are applied on the edges of the input.

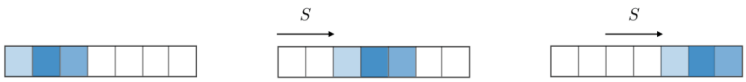


Figure 7: Example of stride hyperparameter where S=2.

(Xu et al., 2015). The definition of the output of ReLU activation function (g) is:

$$g(z) = max(0, z) \tag{1}$$

where *z* represents the input. Its visual representation is illustrated in Figure 8.

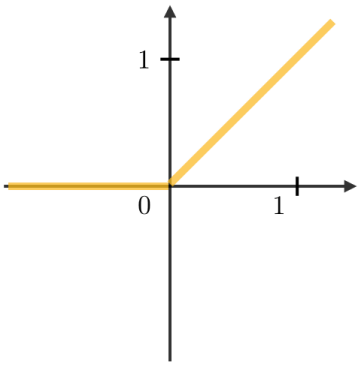
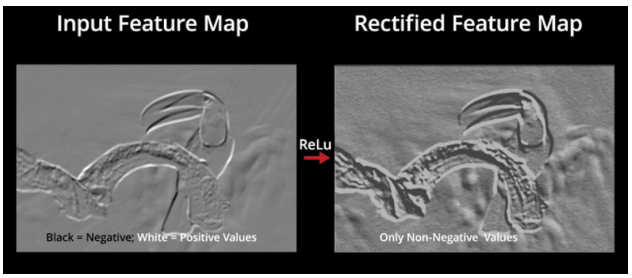


Figure 8: Graphic representation of ReLU activation layer.

By introducing the ReLU function, all the negative pixels are set to 0 via an element-wise operation. The output of this calculation is called *rectified feature map* and an example can be seen in Figure 8



.

Figure 9: Example of a feature map and a corresponding rectified feature map

Pooling layer

A pooling layer is introduced to the model architecture to perform a downsampling operation. This step reduces the in-plane dimensionality of the feature maps. Such a solution reduces the number of parameters and amount of computation in the network, therefore helping to control for overfitting. It is important to note that pooling layers have no learnable parameters, which means that the filter size, padding and stride are all hyperparameters in a similar way to convolution operations. The most commonly used form of pooling layer is represented by a filter (kernel) of size 2x2 and stride 2 (Teuwen and Moriakov, 2020). This downsamples every depth slice giving an output of half size in two dimensions (the third dimension is preserved). In connection with the above, this reduces the number of parameters by 75%. Additionally, adding pooling layers makes the network invariant to small distortions, transformations and translations. Figure 10 presents an example of calculating a pooling layer from a rectified feature map while Figure 11 demonstrates an output image where pooling has been performed on a rectified feature map.

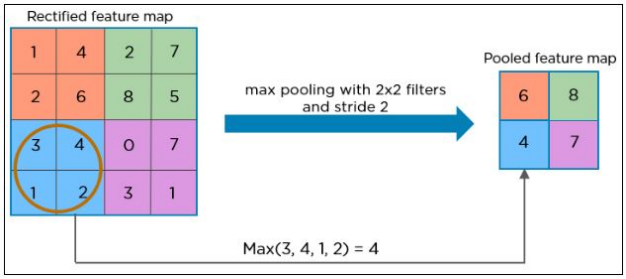


Figure 10: Example of a rectified feature map and the corresponding pooled feature map. The pooling layer implements different filters to help identify parts of the image such as corners, edges, eyes, etc.

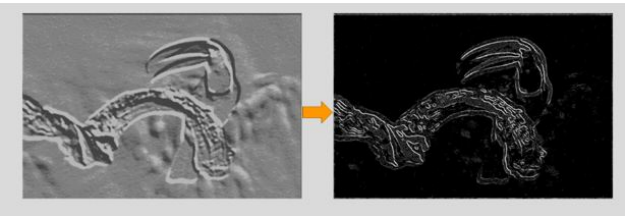


Figure 11: Calculating a pooled feature map from a rectified feature map with parameters: 2x2 filter and stride 2.

The most commonly used functions are max-pooling [1], average pooling (a comparison of max and average pooling can be found in (Boureau et al., 2010)), Stochastic pooling (Zeiler and Fergus, 2013) or *L2* (Le, 2013).

It is important to mention that pooling operations have their opponents. In their work, Springenberg et al. (2014) propose the use of architecture built solely on repeated convolution layers. In this case the subsampling is achieved through the increased size of the stride.

Fully connected layer

The fully connected layer works on a flattened input and their neurons have full connection to all activations in previous layers. The aim of the fully connected layer is to return a classification vector obtained from high-level feature map. Using a popular softmax classifier, the input provided is a vector of scores while the received output is a vector of probabilities.

2.2.2 Deep Learning-Based Processing

Deep learning is a field within machine learning which uses algorithms inspired by the functions and structure of the brain. Compared to conventional machine-learning methods, which are restricted in their ability to process natural data in their raw form, deep learning uses models built from multiple processing layers to learn representations of raw data including multiple levels of abstraction (LeCun et al., 2015).

2.2.2.1 Instance Segmentation: Mask R-CNN

Mask R-CNN is a multi-task deep learning network that combines the idea of Fully Convolutional Networks (FCNs) and Feature Pyramid Networks (FPNs) (He et al., 2017), and is based on the Faster R-CNN (Ren et al., 2015) (Figure 12). The purpose of the Mask R-CNN is not only to effectively detect objects of interest, but also to produce a high-quality segmentation mask for the objects. According to the developers of the Faster R-CNN architecture, the method outperformed the majority of other architectures in the field of object detection. However, it was insufficient for users looking for a deep learning-based semantic segmentation method. Having created a very good network for object detection, it was decided to extend the functionality of the Faster R-CNN, but instead of implementing semantic segmentation the designers went one step further and worked on instance segmentation. To achieve their goal, a few adaptations were implemented in the Faster R-CNN architecture. In line with the original terminology used by He et al. (2017), the remaining part of this section will discuss the use of backbone architecture for feature extraction, head architecture for classification, mask prediction and bounding box regression.

Backbone architecture

The original implementation of Mask R-CNN, which was proposed in 2016 by the Microsoft Research team (He et al., 2017), used ResNet as a backbone architecture. Additionally, experiments on developing Mask R-CNN included the extension of the ResNet architecture by including an FPN. The idea of an FPN is to perform subsampling on a single-scale image of arbitrary size to obtain a different spatial resolution feature hierarchy with a multi-scale, pyramidal

[1] The biggest value from those overlaid by a kernel is chosen

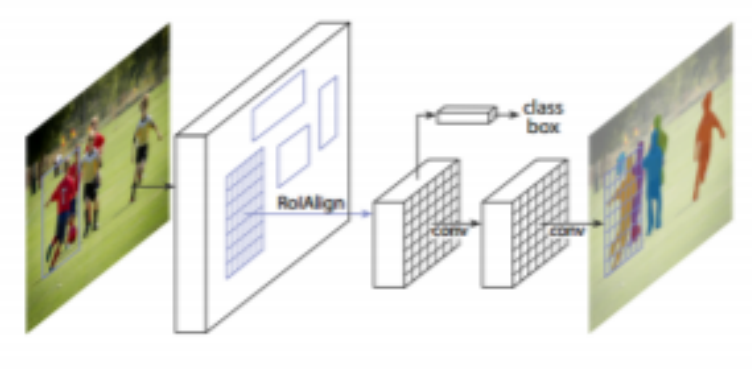


Figure 12: Architecture schema of Mask R-CNN (He et al., 2017).

shape (Figure 13). This method does not depend on the backbone convolutional architectures and uses a bottom-up pathway, a top-down pathway and lateral connections to construct a pyramid. The bottom-up pathway is constructed to compute feature maps at various scales, with a scaling step of 2. In the described method, many layers produce feature maps of the same size, and therefore they are included in the same network stage. In an FPN, one level is obtained from the output of the previous layer of the stage, excluding the first stage due to its large memory demands. The top-down pathway is performed by the application of the upsampling method on feature maps from higher levels, which are then enhanced with features acquired in the bottom-up pathway via lateral connections. Lateral connections are used to merge feature maps from the bottom-up and top-down pathways belonging to the same spatial size stage.

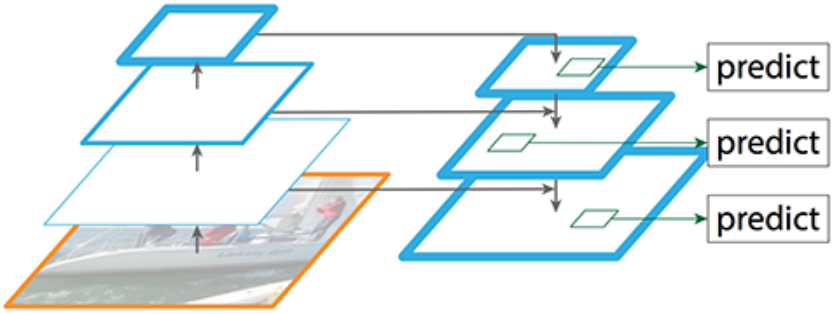


Figure 13: Feature Pyramid Network with predictions made independently at all levels (Lin et al., 2017a).

Head architecture

The most important change made to Faster R-CNN to achieve instance segmentation was in the head architecture, by including branches for classification, bounding box regression and the mask branch. The mask branch is the result of applying the FCN to each region of interest (RoI), therefore predicting a segmentation mask at a pixel-to-pixel level. The mask is represented by a binary matrix made of ones (where the object is located) and zeros (remaining locations).

RoIAlign

The RoIAlign layer used in Mask R-CNN differs from that used in Fast R-CNN (called RoI). The adaptation was needed because the spatial accuracy of RoI pooling (RoIPool) was not designed to be used for pixel-to-pixel alignment. This inaccuracy was the result of the harsh quantization of RoIPool, an operation commonly used to extract small feature maps for each RoI (Girshick, 2015). The problem was mitigated by introducing RoIAlign that removed harsh quantization and properly aligned the extracted features with the input. The input values were calculated by implementing bilinear interpolation at regular sampled locations. Figure 14 presents the RoIAlign used in Mask RCNN, where the feature map is represented by the dashed grid, the dots represent 4 sampling points in each bin and the solid lines represent an RoI.

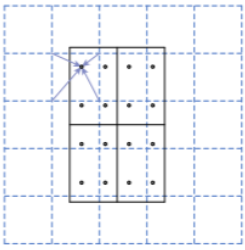


Figure 14: The RoIAlign schema.

2.2.2.2 Object Detection: YOLO

The Mask R-CNN method introduced in 2.2.2.1 was developed to allow object detection and localization at a pixel level. The information retrieved from instance segmentation allows for informed reasoning about depth layering and occlusion (Z. Zhang et al., 2016). Unfortunately, due to the complex nature of the method, the segmentation step is limited by the detection stage (Ruiz-Santaquiteria et al., 2020). To assess the detection capabilities of Mask R-CNN, in this study a comparative analysis between instance segmentation and object detection was performed. Object detection was achieved by training the You Only Look Once (YOLO) model.

The YOLO, is a family of end-to-end deep learning models developed by Redmon et al. (2016) and used for fast object detection (often in real-time applications). This model uses a one-stage detector strategy. YOLO sees object detection as a single regression problem where an input image is fed once through the network, resulting in score and bounding box coordinates of detected objects of interest (Figure 15).

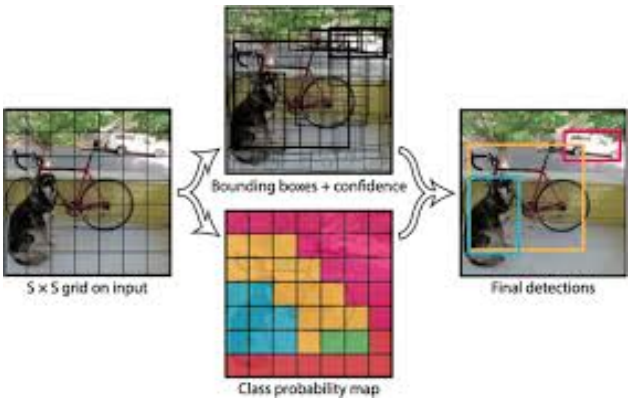


Figure 15: YOLO system scheme to detect objects seen as a regression problem.

YOLO divides the input image into a SxS grid. Next, for each grid cell model, it calculates

predictions for the bounding box (B) along with confidence and class scores. The confidence score signalizes how confident the model is that the object is within the bounding box, and how accurate the prediction of the box itself is. Formally, confidence is defined as $Pr(Object) * IOU_{pred}^{truth}$ (where IOU stands for Intersection Over Union). If there is no object in the cell, the confidence score should be zero, otherwise it should be equal to the IOU between truth and the predicted box. Additionally, for each cell box the model predicts C class probabilities, $Pr(Class_i|Object)$. Regardless of the number of bounding boxes (B) in the grid cell, the model only calculates one set of class probabilities. During the testing phase for each box the model calculates class-specific confidence scores from the equation:

$$[h!]Pr(Class_i|Object) * Pr(Object) * IOU_{pred}^{truth} = Pr(Class_i) * IOU_{pred}^{truth} \quad (2)$$

Each of these scores informs the probability of the class within the box and how accurately the predicted box fits the object.

YOLOv3

Since its first release in 2016, the YOLO model has evolved through a few modified and improved versions. The one used in this dissertation project is called YOLOv3 and was released in 2018 (Redmon and Farhadi, 2018). In general the YOLOv3 architecture can be described as a system built from two major components: a Feature Extractor and a Detector, both multi-scale. Feature extraction in YOLOv3 uses Darknet-53 backbone architecture, which, as the name suggests, uses 53 convolutional layers. The whole YOLOv3 network consists of 106 layers where 75 of them are convolutional (Figure 16). The model architecture does not include a pooling layer and downsampling is performed by the convolutional layers with stride 2.

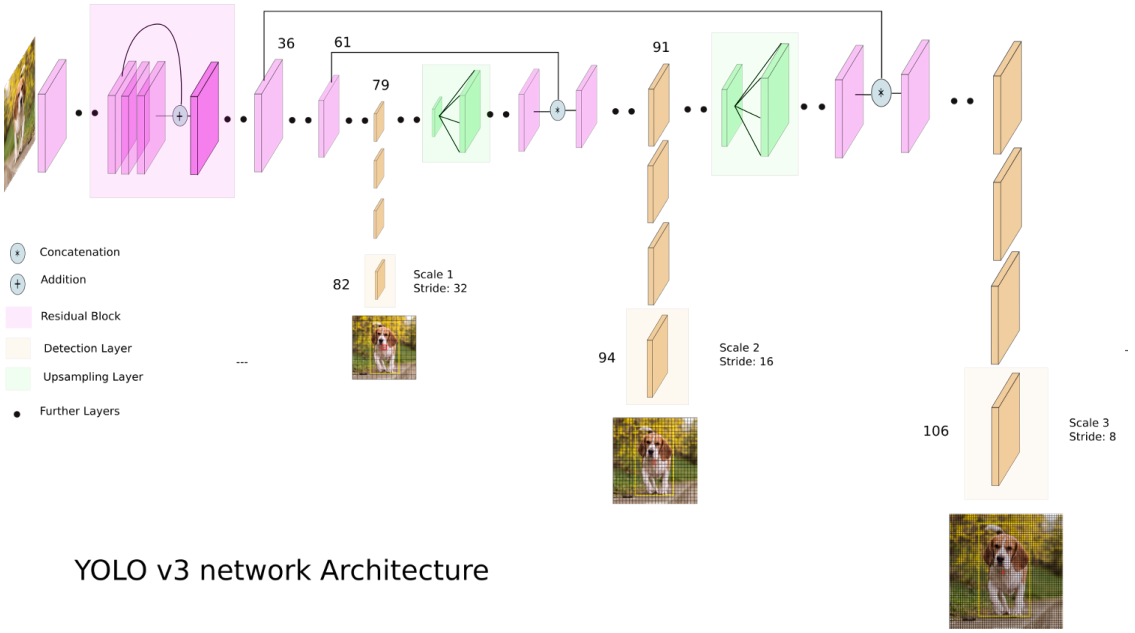


Figure 16: YOLOv3 architecture (Kathuria, 2018).

One of the improvements of YOLOv3 is its box prediction at 3 different scales, allowing detection of small, medium and large objects. This is achieved by applying a detection kernel of size $1x1$ at three different places in the network and on feature maps of three different sizes.

The detection kernel shape is calculated from the equation:

$$1x1x(Bx(4+1+C)) \tag{3}$$

where B refers to the number of bounding boxes, C stands for the number of classes, '4' refers to the 4 bounding box coordinates, '1' represents the object confidence and C refers to the number of classes.
Another improvement in YOLOv3 is its shift from multi-class to multi-label classification. This change was due to the fact that some datasets contain labels of hierarchical or related structures, e.g. datasets with woman and person classes. Therefore, the authors implemented logistic regression to predict class scores and binary cross-entropy to predict multiple labels for an object.

YOLO has been applied in various areas like real-time object detection (Laroca et al., 2018), urban application (Cai et al., 2020),vehicle and vessel detection (Zhang and Zhang, 2019) and vegetation detection (Tian et al. ,2019).

2.2.3 Introduction to Mobile Mapping Systems

In general, Mobile Mapping Systems (MMS) are built on the basis of combining electronic imaging with high-performance georeferencing. In ground-based mapping applications, sensors are mounted on the roof of a vehicle. Both images and point clouds are usually captured while the vehicle commutes through the area of interest. At the same time, with each exposure, the positioning system records the orientation and location of the imaging systems to enable accurate georeferencing. Figure 17 presents an example of a typical Mobile Mapping System. Ground-based imagery provides a high-resolution viewpoint which offers great potential for enriching the information collected from overhead images to map and monitor human and natural activities on Earth. Additionally, they are less prone to typical issues related to remote sensing data. For instance they are not limited to nadir or oblique view and avoid the negative impact of cloud coverage. On the other hand, street level images pose challenges in terms of acquisition conditions (illumination changes), motion or common object occlusions. This section will introduce a selection of mobile mapping systems. The table 2 presents the basic functions of the most important components of the MMS.

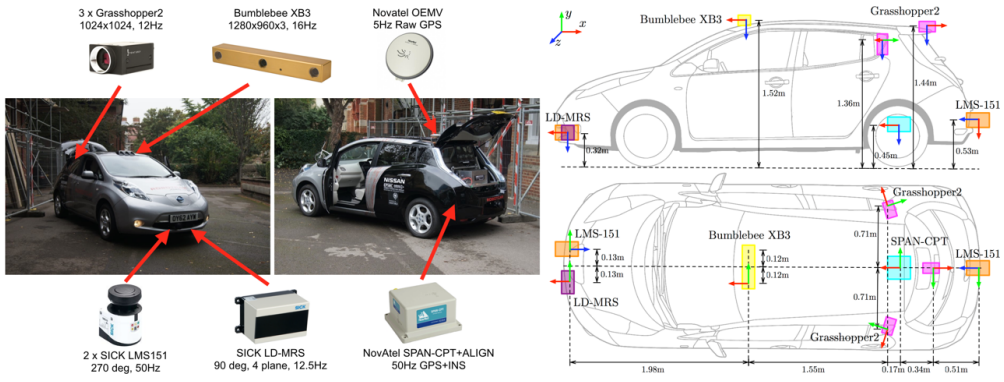


Figure 17: Example of a mobile mapping system: here the system used by Oxford Research Group (Maddern et al., 2017). Left: the sensors location on the RobotCar platform, right: sensor location diagram. Coordinate frames present the direction and origin of each sensor with the convention: x-forward (red), y-right (green), z-down(blue).

Table 2: Main Mobile Mapping System sensors and its characteristics

Sensor	Type of Information	Characteristics
Cameras	Descriptive/Geometric	Images; geometric accuracy correlated with sensor resolution
Laser Scanner	Descriptive/Geometric	Determine distance between the object and the sensor, scanning angle
GPS	Geometric	Provides the senor position in 3D space. Global Reference, high position accuracy and medium attitude accuracy
INS	Geometric	Provides the senor orientation in 3D space. Low-to-high relative position and attitude accuracy

2.2.3.1 Data available from Mobile Mapping Systems

Due to the growing interest and recognition of many applications for data from ground-based mobile systems, this section introduces datasets collected by MMS which are available for public use. These datasets are very often collected to either help researchers to work with autonomous and smart cars or to allow for better understanding of complex urban environments through the implementation of numerous different computer vision, machine learning and deep learning methods.

While analyzing the existing literature on computer vision applications for street furniture detection based on ground level data, it became clear that the vast majority of these applications use images or point cloud data. An overview of the data provided by two typical, freely-available street-level MMS datasets is summarised below.

2.2.3.2 Waymo Open Dataset

Waymo LLC in an autonomous driving technology development company from America, which offers a high-resolution dataset collected by its self-driving cars in a wide variety of conditions (Sun et al., 2020). The dataset comprises 1950 segments obtained in diverse conditions and geographies. Figure 18 presents the layout of sensors mounted to the Waymo car. Figure 19 and Figure 20 presents examples of labeled data: images and scans, respectively. Sensor data of Waymo Mobile Mapping System includes:

- mid and short range lidars
- camera images
- lidar to camera projections

- synchronised camera and lidar data
- as well as vehicle poses and sensor calibration data

Data in the set include:

- 11.8M bounding box labels for images for four classess (vehicles, pedestrians, cyclists and signs)
- 12.6M 3D bounding box labels for lidar data
- 1,200 segments with high quality labels for lidar data
- 1,000 segments with high quality labels for camera data

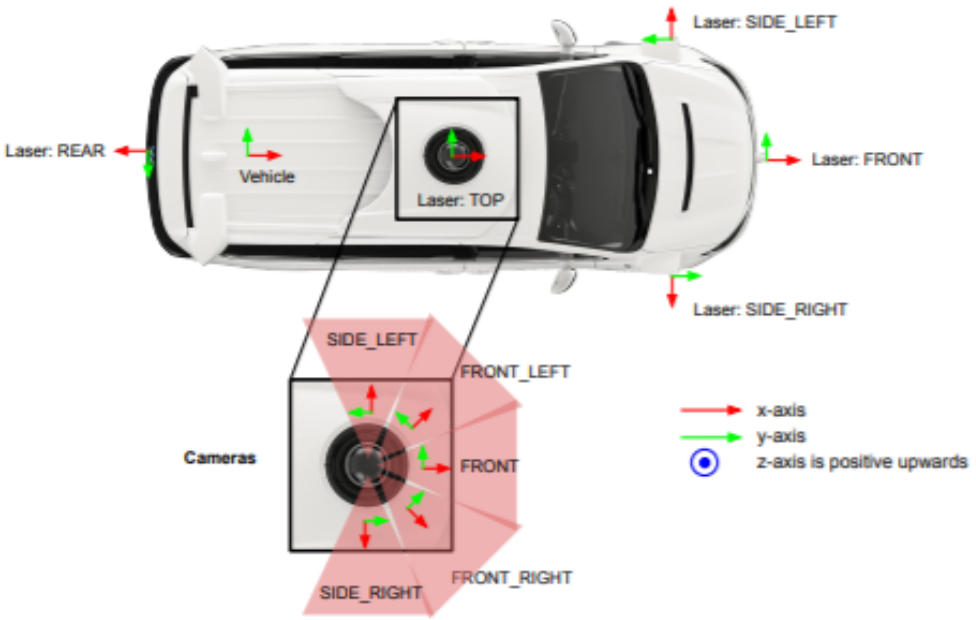


Figure 18: Sensor distribution and coordinate system used in the Waymo mobile mapping system.

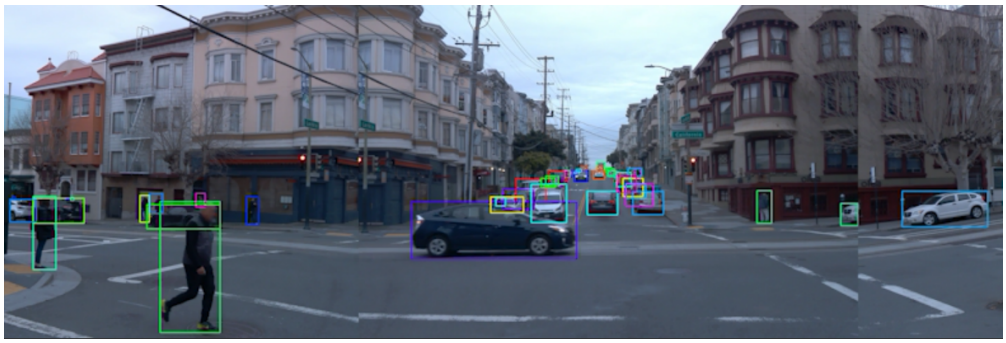


Figure 19: Example of the image labels available in the Waymo dataset.

Waymo is one of the TensorFlow Datasets, ready to use with TensorFlow or other machine learning pipelines available in Python for object detection applications. The undeniable advantage of this dataset is its diversity. Data acquisiton took place in multiple cities and suburbs, during day and night, with various constructions along the road and under diverse weather conditions. This diversity makes it a very powerful asset for the research community.

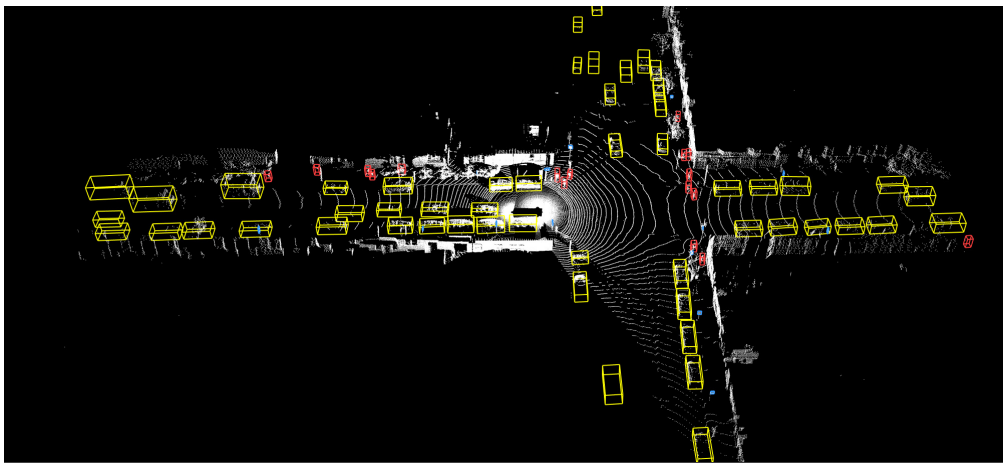


Figure 20: Example of the LiDAR labels available in the Waymo dataset.

2.2.3.3 Cityscapes Open Dataset

Second on the list of publicly available datasets representing city environments is the Cityscapes benchmark suite. The motivation to create this dataset was the lack of large-scale datasets with an understanding of urban scenes (Cordts et al., 2016). The Cityscape dataset includes data collected from 50 different cities. 5,000 of the images included in the set have high quality pixel-level annotations. Additionally, coarse annotations have been prepared for 20,000 images, for the benefit of methods that require large amounts of weakly-labeled data. The dataset is characterised by the following features:

- data were collected over several months in spring, summer and fall (images were intentionally not collected during conditions with heavy snow or rain)
- initially, images were acquired in Germany but then extended to neighbouring countries
- dense pixel-level annotations were created on 5,000 images from 27 cities while
- coarse annotations were delivered from the remaining 23 cities

The images used in the dataset are annotated with 30 visual classes (Figure 21) which are grouped into eight categories: human, object, vehicle, nature, flat, construction, sky and void. The selection was dictated by the importance from an application point of view, the frequency of occurrence, and the effort involved in the annotation process. For the evaluation, classes with only very rare appearances were excluded. Figure 22 presents sample images with annotations.

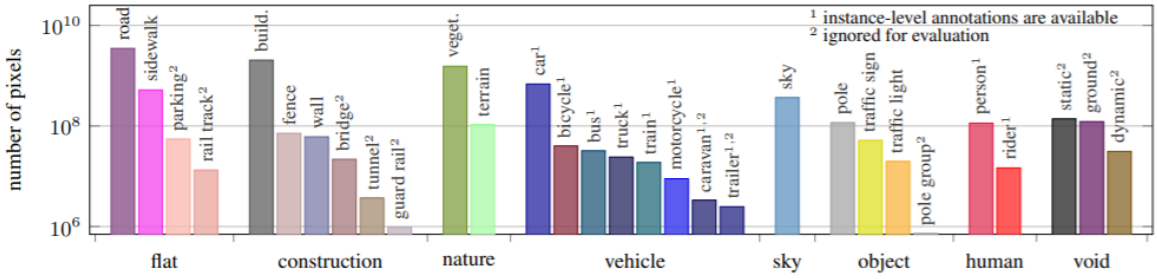


Figure 21: Classes with number of annotated pixels (y-axis) and their associated categories (x-axis).



Figure 22: Example images with annotations.

The creators of this benchmark suite plan to release an extended version of their work in the near future. The latest version will be enriched with 3D bounding box annotations for all types of vehicles. Most often 3D annotations are acquired from LiDAR point clouds. In contrast to this approach, the Cityscape designers used stereo RGB images which allowed for only a pixel-accurate reprojection in the RGB images. This resulted in a higher range of annotations compared to the LiDAR-based approach (Gählert et al., 2020). An example of the image with 3D bounding boxes is presented in Fig 23.



Figure 23: Example with 3D bounding boxes for vehicles.

3 METHODOLOGY

3.1 Introduction

The main goal of this project was to investigate the capabilities of deep learning-based instance segmentation for front door detection using images obtained from a Mobile Mapping System. Deep learning instance segmentation involves training data to develop a predictive model. Thus, the collection of suitable images for training is the most crucial step. This requirement is different to the work of Juenemann et al. (2010) who, in their study combined image processing and fuzzy logic techniques, used only 30 images out of 196 and achieved 80% rate of detection of front door instances. This study therefore investigates how different input image combinations influence the predictive model performance. When developing the method, the following aspects were considered:

- To what extent does the location of the data collected for the model training influence front door detection in images collected at different test sites? In this case, data collected in Nottingham (United Kingdom) and Lublin (Poland) were used to train the model, whereas data from Abingdon and Oxford (both in United Kingdom) were used in the testing datasets.
- Is there an improvement in the front door detection rate when new types of images are added to the training dataset? These new images refer to training data with glass doors (e.g. French doors, glass shop doors) and data from Google Street View, which presents doors collected by a camera at different angles compared to data collected from mobile phones. This process not only increased the number of training samples, but also enriched the variety of door types available in the training samples.
- How does the number of images used for the model training influence the detection rate? Deep learning processing is known for being a data driven approach. For this reason, the model is trained using all of the available training data as well as using a reduced amount of data. The reduction involves the removal of images that represent the same type door, so that each door appears only once. Some of the images were zoomed in or out, or door representatives were located in different places within the training image frames (these types of duplicated data were removed).

Image instance segmentation was carried out using Mask R-CNN, a well investigated and established DL model. As outlined in Chapter 2, Mask R-CNN is a modified version of the Faster R-CNN object detection framework, which has been extended with additional branches being used to perform the segmentation of the detected RoIs.

As demonstrated in Section 2.1.1, there is a clear lack of previous research focusing on front door detection. This makes it difficult to directly compare the results obtained in this study with those achieved using other methods. For this reason, a second model, named YOLOv3, was also evaluated for its detection capabilities. YOLOv3 is a deep learning architecture used for object detection. It is commonly adopted due to its rapid performance and the ease of implementation. It is a one stage method which looks at the input image only once. For the purpose of this project, the YOLOv3 model was trained on two datasets comprising the one that offered the best overall performance and the one that achieved the highest ratio of correctly identified front doors for the Mask R-CNN instance segmentation approach.

The methodological process was an iterative approach. Below are listed the milestones of the project development:

1. Data were collected via a mobile phone in Lublin and Nottingham and used to train the model, which was then tested on MMS images from Oxford and Abingdon. These preliminary results steered the project towards more in-depth research and analysis of the effect of the data type and localization origin on final front door detection. Additionally, these results also highlighted a problem with incorrect classification, where windows were sometimes classified as doors. Accordingly, it was decided to perform model training with an extended number of images including glass doors. The final dataset included the extraction of images from Google Street View for both Nottingham and Lublin. This was mainly due to an issue with the testing images from Oxford, as some of them showed doors which were very tilted (with the camera facing the direction of the ride).

2. The model was trained on various combinations of data, distinguished by the location and sensor from which they were captured to determine how these factors influence the detection ability of the trained model. The additional dataset containing glass doors was collected in Nottingham only, but was also used in the model training along with images from Lublin. Furthermore, two tests were carried out with images from both Lublin and Nottingham.

3. Lastly, in order to compare the results obtained from the Mask R-CNN with an alternative method, two datasets were used to train an another model commonly used for deep learning based object detection, YOLOv3.

Figure 24 outlines the final methodology pipeline, including the files and points of evaluation for this project.

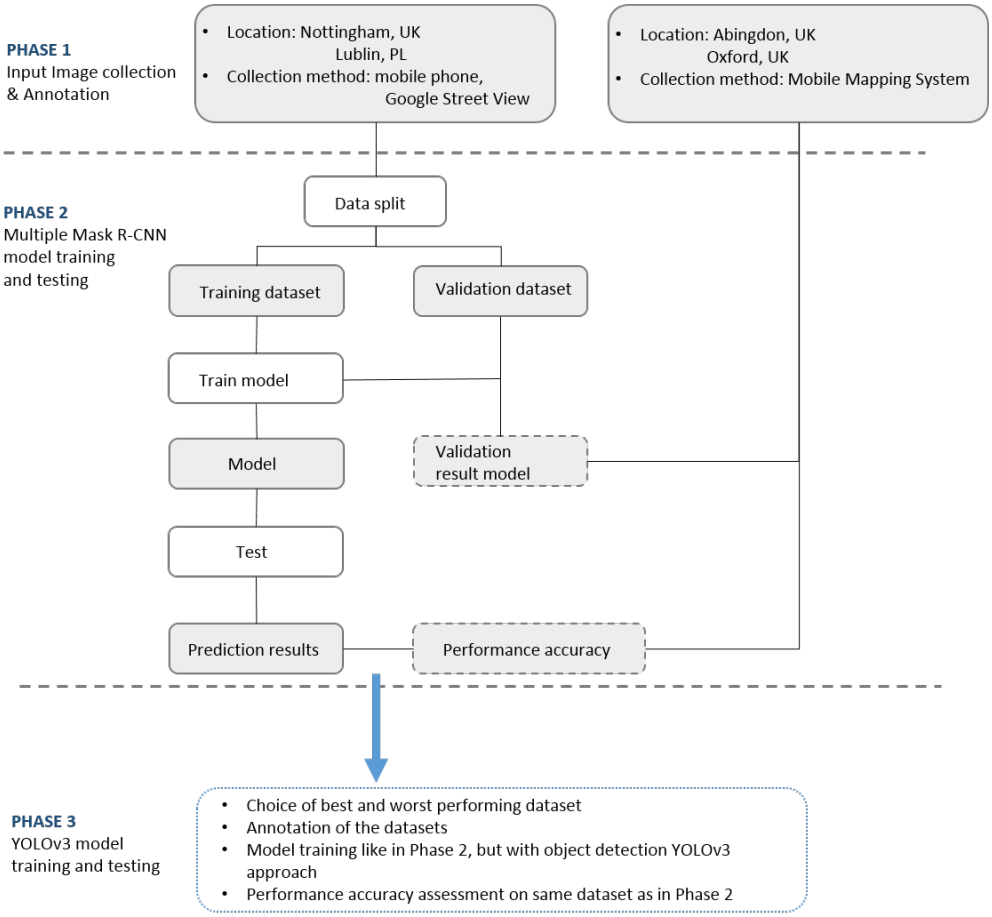


Figure 24: A flow diagram of the process developed for the purpose of this project. The white boxes refer to the processing steps, the gray with solid borders present the input/intermediate/output files and the boxes with dashed borders cover the evaluation points.

3.2 Datasets

The images used in this project were collected in different locations and using different platforms. The reason for this was to assess the robustness of the Mask R-CNN method for detecting front doors. Despite the diversity found within the images, they had one thing in common: they were all collected at street level. Figure 25 illustrates the locations of the data in respect to their function in the processing pipeline:

- Magenta dots - data used for training and validation, collected in Nottingham (United Kingdom) and Lublin (Poland); data collected by mobile phone and from Google Street View.
- Green dots - data used for testing, collected in Abingdon and Oxford (both cities in the United Kingdom); data acquired from the Mobile Mapping System.



Figure 25: A map illustrating the locations of data used in this project.

3.2.1 Training and validation dataset

The images used to build the datasets for the model training and validation were collected both by mobile phone and from Google Street View. The difference in the location of the origin of the photographs was in order to investigation whether it is possible to use images from locations independent from the testing area and still achieve satisfactory results. The sets of data included in this stage of processing included images with glass doors, collected in Nottingham only, but used for both locations. Table 3 presents the image statistics based on the location and data collection method. The total number of images used in this work was 525 for the training dataset and 96 for the validation dataset. Figure 26 presents the sample images used for training and validation.

In addition to the aforementioned data, images available on the internet were also taken into consideration for the purpose of the study. However, these often lacked context, i.e. the doors covered majority of the image. What is more, resized images were distorted in the door width to height ratio. There were also attempts to use images from the websites of various retail agencies (e.g. Rightmove), but the images uploaded were very small, making it unfeasible to distinguish door instances.



Figure 26: The sample images used for model training and validation. Top row: Nottingham-mobile phone, second raw: Lublin - mobile phone, third row: Nottingham and Lublin -Google Street View, bottom row: Nottingham - glass doors.

Table 3: Image statistics. The collection method (and type) and number of images used for the training and validation steps. Images illustrating glass doors were collected only in Nottingham, but used in both locations. The table also presents the distribution of the data between the test and validation steps.

	Nottingham		**Lublin**	
	Training	Validation	Training	Validation
Mobile Phone	204	36	166	30
Glass Doors (Nottingham only)	69	13	69	13
Google Street View	41	8	45	9

3.2.2 Test Dataset

An independent set of images acquired from the Mobile Mapping System was used to evaluate the performance accuracy of the trained models. These data were chosen for testing purposes due to the angle at which they mapped the street (facing buildings perpendicularly), their applications in real world scenarios, as well as their usefulness for further study. The data used for testing were delivered by the Ordnance Survey and Oxford RobotCar group, and were collected in Abingdon and Oxford respectively. Table 4 presents the number of images from each of the sets used for testing purposes.

Table 4: Number of test images with respect to the location.

	Abingdon	**Oxford**
MMS RGB	65	-
MMS Gray (mono and stereo)	-	13

Abingdon Ordnance Survey dataset

The data provided by Ordnance Survey were collected by Korec[2], who used the Trimble® MX9 Mobile System System as their solution (Figure 27). The system is equipped with:

1. An MX9 laser scanner
2. One spherical camera
3. Two side looking cameras (optional)
4. One backward/downward looking camera (optional)
5. The Trimble GNSS-Inertial system



Figure 27: Left: the Korec Mobile Mapping System; right: The Trimble® MX9 system.

The images used for the project were obtained from the MX9's spherical camera, which captured a 360° view of the scene around the vehicle at approximately 3m intervals along the route. Each spherical image was made up of six frames, each consisting of 5 megapixels. The spherical images were decomposed into their six frames and examined to discover which ones offered the best view of the building facades and doorways. The resulting images were used to compose the dataset used to examine the model performance. Figure 28 presents sample photos from Abingdon, Oxfordshire.

[2]https://www.korecgroup.com/



Figure 28: Examples of images from Abingdon, © Crown copyright 2020 OS.

Oxford Open dataset

The Oxford RobotCar dataset (Maddern et al., 2017) was chosen due the distinct characteristics it displayed although being far from perfect (data collected with focus on investigating long-term localisation and mapping and not instance segmentation problems) in comparison with the dataset provided by Ordnance Survey or the benchmark datasets presented in section 2.2.3.1. The Oxford RobotCar data were provided in mono and stereo composition so their quality is inferior compared to other datasets; for example: the edges of the doors appeared slightly blurred and indistinct. Moreover, they appeared in grayscale, while the others appeared in Red, Green, Blue (RGB) color space. Lastly, the Oxford RobotCar group provided a 1GB sample which was easier to maintain than the other datasets (over 15 GB), taking into consideration the remote work, insufficient internet connection and time constraints. Sample testing images from this dataset are presented in Figure 29. The platform that collected the data was equipped with the following sensors:

- one Point Grey Bumblebee XB3 trinocular stereo camera
- three Point Grey Grasshopper2 monocular cameras
- two SICK LMS-151 2D LIDARs
- one SICK LD-MRS 3D LIDAR
- 1 x NovAtel SPAN-CPT ALIGN inertial and GPS navigation system



Figure 29: Sample images from Oxford, Oxford Copyright © 2016 Oxford Robotics Institute.

3.2.3 Data Annotation

Annotating images is an integral part of deep learning model training. In this study, images were annotated using following the software:

- VGG Image Annotator[3] (Dutta and Zisserman, 2019) - this software was used to draw and annotate masks around doors, with the output used in Mask R-CNN training.
- VoTT Visual Object Tagging Tool created by Microsoft[4] - the software was used to draw bounding boxes and annotate doors, with the output used in the YOLOv3 training.

The choice of different annotation tools for the Mask R-CNN and YOLOv3 image annotation was due to the different nature of the annotation shapes, polygons and bounding boxes utilised by the two techniques. Furthermore, these were the same tools that were used in the previous studies involving the deep learning solutions implemented in this project. This reduced the time needed to annotate the data and the necessary processing of the files with bounding box/mask coordinates in order to suit the implementation requirements. Both softwares used for the image annotation were able to labeling the full number of objects in a given photo. The difference is however, in the file format storing annotation information. VGG Image Annotator creates one file which can be imported in one of the following formats: *.csv, *.json or *.coco. VoTT tool, on the other hand, first creates a separate *.txt file for each corresponding image and later these files are combine into one when exported (exported file formats include *.csv, Pascal VOC, Tensorflow Records).

3.3 Experiment Setup and Implementation

The aim of this section is to present the full implementation details of the data and the methods used in this study.

3.3.1 Training Data Preparation

The training images used in the process were prepared according to the following pipeline:

1. Image cropping to 512X512 pixels. This was done to reduce the size of the training data and therefore speed up the training process.
2. Data annotation.
   (a) To maintain the shape of the masks created to train Mask R-CNN models, all of the data were labelled once and then selected along with the images based on the data configuration necessary for each of the model training experiments.
   (b) The annotation process for the YOLOv3 method was performed after analyzing the results of the Mask R-CNN training.

Based on the information presented in Table 5, the training of the Mask R-CNN model was conducted using a several different data configurations to evaluate the method's front door detection performance depending on:

- different location and data type - Data Combination 1,3, 5 and Data Combination 2,4, 6
- different size of training dataset - Data Combination 7 and 8

After analyzing the obtained results for Mask R-CNN the training data representing the combinations that achieved the highest number of correctly detected doors and best overall performance were subsequently used to train the YOLOv3 model.

[3]http://www.robots.ox.ac.uk/ vgg/software/via/via.html
[4]https://github.com/microsoft/VoTT

Table 5: The dataset configuration used to train the Mask R-CNN model for each of the eight experiments

Dataset	Data combination							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Lublin-Phone	+		+		+		+	+
Nottingham-Phone		+		+		+	+	+
Nottingham-Glass			+	+	+	+	+	+
Lublin-GSV					+		+	+
Nottingham-GSV						+	+	+

3.3.2 Model Training

Both deep learning approaches used in this study were trained using their settings, as evaluating the model performance with respect to their hyperparameters was not within the scope of this study. The only exception relates to the Mask R-CNN minimum probability value used to accept a detected instance, which refers to a threshold below which Regions of Interest are skipped. In the default settings this value was set to 0.7. However, the initial analysis of the preliminary results indicated that all objects with a threshold below 0.8 were incorrectly classified; for instance, windows were classified as doors. Therefore, the parameter was changed to 0.8, which improved the final results. Table 6 and Table 7 present the hyperparameter settings for the training of the Mask R-CNN and YOLOv3 architecture respectively. While training the YOLOv3 model, the process did not run through all of the set 102 epochs. This was due to a setting in the code which used a Keras[5] function called *EarlyStopping*. This function allowed the establishment of an arbitrary number of epochs to train the model and ended the process when the model performance stopped improving based on the validation dataset (function used to avoid model overfitting). In this study, the YOLOv3 model training stopped between 75 and 82 epochs, depending on the training dataset.

Along with the training step, each model was evaluated using a separate validation dataset. The validation datasets were composed of images from Nottingham and Lublin (unlike the test data originating from Abingdon and Oxford) and their division is presented in Table 3. The main reason for using the validation step, was to measure the quality of the model, whether it overfits or underfits. The models did not show any alarming behaviour indicating overfitting or underfitting. Since the Mask R-CNN model was trained on 30 epochs, 30 weights for each experiment were generated. Analysis of the validation graph helped to determine, which model weight to used to test the model performance. The model weights chosen for the testing phase were generated between 25 to 30 epoch.

[5]https://keras.io/

Table 6: The hyperparameter setting for the Mask R-CNN model training.

Configuration	Description
Learning Rate	0.001
Learning Rate Momentum	0.9
Image Min Dimension	512
Image Max Dimension	512
Detection Min Confidence	0.8 (originally it is 0.7)
Number of Epoch	30
Steps per Epoch	1000
Mask Shape	28x28
No. of classes	2 (door and background)

Table 7: Hyperparameter setting for YOLOv3 model training.

Configuration	Description
Image Min Dimension	512
Image Max Dimension	512
Batch size	32
Number of Epochs	102
Steps per Epoch	Depends on no. of input training images. It is equal to (no. of images/batch_size)
No. of classes	2 (door and background)

3.3.3 Evaluation Metrics

The models were assessed based on the following metrics:

- Precision
- Recall
- F1-score

The performance of each trained model was evaluated using precision (P), recall (R) and F1-

score (Sokolova et al., 2006). For the purpose of the models comparison, value of F1-score and precision equal to 0.6 is set as the baseline, and everything above is considered as a good result. Recall below 0.7 is set as a baseline, and everything below is considered as a good result. Definition of precision, recall and F1-score are presented in equations 4, 5 and 6.

$$P = \frac{T_P}{T_P + F_P} \tag{4}$$

$$R = \frac{T_P}{T_P + F_N} \tag{5}$$

$$F1 = 2 * \frac{P * R}{P + R} \tag{6}$$

where: T_P (true positive) represents the number of correctly identified front door instances, F_P (false positive) appears when the network identified a door instance where there is none. Finally, F_N (false negative) refers to situations when a front door was missed by the network.

Precision is a measure of the correctly identified true positive cases from amongst all of the positive cases. Its value decreases as the number of false positives increases. Recall refers to the number of positive predictions from amongst all of the actual positive class values. It demonstrates the impact of false negatives on the performance of the model. The F1 score conveys the harmony between the precision and the recall as its penalizes extreme values.

3.3.4 Software Implementation

Due to hardware limitations, all of the processing was conducted using the Google Colaboratory Platform[6] which offers free access to its GPU; a crucial factor in deep learning processing. The graphics card offered by Google Colab is the NVIDIA Tesla T4 Tensor Core. The training time for each epoch differed from 45s to 300s, possibly because the full capacity of the offered GPU was not utilised. The best way to use Google Colab is to download all of the necessary data onto Google Drive despite the limited storage capacity of the drive. Another limitation of the Colab is the maximum processing time, which is restricted to 12h. This is sufficient for smaller projects with relatively low amounts of input data, but can cause issues if more time is needed for model training. Solutions in this work were adopted from:

- Mask R-CNN[7] from Matterport (Abdulla, 2017)
- YOLOv3[8] from Anthon Muehlemann (2019)

[6]https://colab.research.google.com/notebooks/intro.ipynb
[7]https://github.com/matterport/Mask$_RCNN$
[8]https://github.com/AntonMu/TrainYourOwnYOLO

4 RESULTS

This section reports the results obtained from training the Mask R-CNN and YOLOv3 models with different combinations of training image samples. Every experiment conducted varied in terms of the data type or deep learning model and the results presented in this section show the evaluation of these experiments. The assessment helped to address the main aim of establishing *To what extent is Mask R-CNN is suitable for instance segmentation of doors from street level images?*.

Each model performance was assessed using the metrics described in section 3.3.3. The evaluation of the results was made on the basis of the following criteria, and, in order, with the first being the most important and the third being the least, although still crucial:

1. F1-score - the higher the value the better overall performance of the model.
2. Precision - the higher the value the better. Important as it says how well the model detects front doors.
3. Recall - the lower the value the better. Indicates the number of objects falsely classified as doors.

The table 8 presents a summary of the results for all the experiments conducted in this research.

Table 8: Summary results of all the experiments.

	Mask R-CNN									**YOLOv3**	
	1	**2**	**3**	**4**	**5**	**6**	**7**	**7'**	**8**	**9**	**10**
F1-score	0.44	0.65	0.47	0.66	0.62	0.63	0.70	0.74	0.65	0.79	0.81
Precision	0.56	0.57	0.48	0.63	0.52	0.5	0.69	0.83	0.59	0.94	0.91
Recall	0.37	0.76	0.48	0.69	0.76	0.82	0.71	0.67	0.71	0.67	0.72

The characteristics used to evaluate the model performance in the testing phase are listed below:

- Number of test images: 78. 65 images from Abingdon and 13 from Oxford.
- Number of images without door instances: 6.
- Number of classes: 2 - door or background.
- Total number of doors: 101 (on some images door instances occured more than once). No. of house doors - 57, no. of glass doors (French, shop doors) - 30, no. of open doors - 11, no. of other doors (lattice) - 3.
- Classified objects were taken into account if they were within 75% of the photo (central part). This is because doors near edges may have be distorted or the full shape may not have appeared. As each object appeared on at least one photo, despite the reduction of the search area, each was classified at least once. Therefore the area of the photo in which the object was searched for was narrowed down.

- Location of images: training and validation - Lublin and Nottingham, testing - Abingdon and Oxford.

4.1 LOCATION AND DATA TYPE

Six experiments on Mask R-CNN architecture were carried out to provide insight into the effect of the data type and the location of the training data on the front door detection. Tests were conducted in the order presented in Table 9.

Table 9: The Settings for tests 1-6 conducted to assess the influence of training data type and location on front door detection.

Experiment no.	Location	Data type
1	Lublin	Mobile photos only
2	Nottingham	Mobile photos only
3	Lublin	Mobile photos, glass door images
4	Nottingham	Mobile photos, glass door images
5	Lublin	Mobile photos, glass door images, Google Street View
6	Nottingham	Mobile photos, glass door images, Google Street View

4.1.1 Data Type

The first set of the conducted tests provided insight into the influence of data type on front door detection. To achieve this, the training dataset with images acquired only by mobile phone was supplemented first with data representing glass doors and subsequently with data from Google Street View. The assumption was that the more diverse the data in the training dataset, the more precise the detection results should be, due to the wider variety of front door representatives.

Figure 30 illustrates the door detection performance for models training using the three datasets built from images with front doors located in Lublin, Poland (Figure 31). The results were evaluated using recall, precision and F1-score metrics to show the performance when the trained model was applied to the combined Abingdon-Oxford test dataset. The detection precision for experiments 1 and 3 was similar and equal to 0.44 and 0.47 respectively, however the recall in test 3 was higher by 0.11 (0.37 for experiment 1 and 0.48 for experiment 3), which means that the model detected more objects as doors, even though they actually belonged to the background class. The number of correctly detected doors increased with the addition of other types of images with correctly detected front door instances for experiments 1,3 and 5 equal to 37, 48, 77 respectively. However the number of background features classified as doors also increased, resulting in 29, 52 and 70 background features for experiments 1, 3 and 5 respectively. Nevertheless, the model performance estimated on the basis of the value of the F1-score, suggested that enriching the training datasets with different door representatives

produces the best overall model performance, which in this case was an F1-score equal to 0.62 for experiment 5.

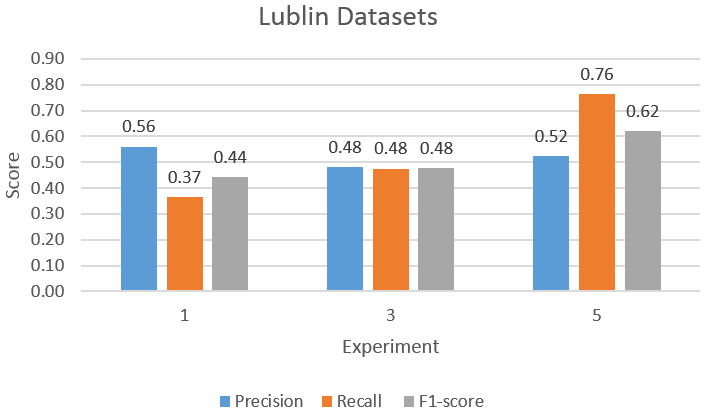


Figure 30: Results for different types of training images collected in Lublin city. Detection performance described based on the: recall, precision and F1-score.



Figure 31: Example of the detection results for experiments 1 (left column),3 (middle column) and 5 (right column) for selected images, top row: Abingdon © Crown copyright 2020 OS, bottom row: Oxford Copyright © 2016 Oxford Robotics Institute.

The experiments for training data collected in Nottingham, United Kingdom were carried out in the same order as the photos from Lublin, and their results are presented in Figure 32. An example of the front door detection results acquired from models trained in experiments 2,4 and 6 is illustrated in Figure 33.

Among these experiments, the results obtained by model 4 (data from the mobile phone and images with glass doors) delivered the best trade-off between precision, recall and F1-score, demonstrating the highest values for precision (0.63) and F1-score (0.66) among all of the experiments. In general, the addition of further images to the datasets improved the number of cases in which doors were detected (except case number 4), but also introduced more cases in which the background was incorrectly classified as a door instance (Figure 34). In the majority of cases, the background feature incorrectly detected as a door was a window instance with few cases of buss stop and trash bin falsely detected as doors.

Based on the presented results, it is difficult to unequivocally state which data type had the strongest influence on the final results. Nevertheless, there is a trend suggesting that introducing new data types improved the front door detection ratio but with the cost of a

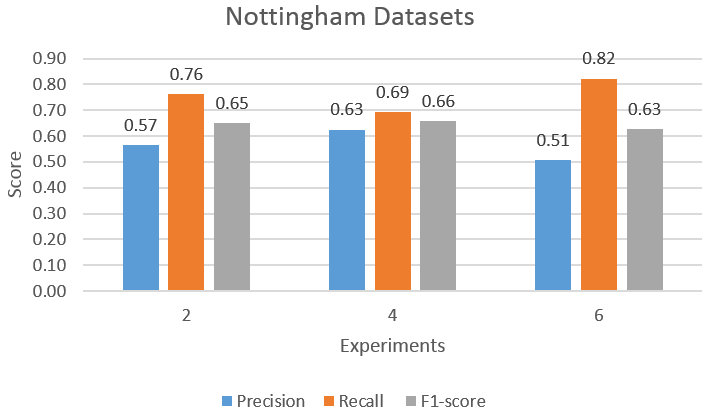


Figure 32: Results for different types of training images collected in Nottingham city. Detection performance described based on the: recall, precision and F1-score.



Figure 33: Example of the detection results for experiments 2 (left column),4 (middle column) and 6 (right column) for selected images, top row: Abingdon © Crown copyright 2020 OS, bottom row: Oxford Copyright © 2016 Oxford Robotics Institute.



Figure 34: Example of objects from background class classified as front doors © Crown copyright 2020.

higher rate of incorrect classification of backgrounds as door instances. This occurred in all cases except experiment 4, where adding more data (from Google Street View) did not improve the model performance.

4.1.2 Location

The experiments investigated in section 4.1.1 are also analysed in this section, but to answer a different question. In this case the focus was on determining the impact of the location of the training data on the detection rate. Figure 35 illustrates precision, recall and F1-scores for all six experiments, where in experiments 1,3 and 5 the model was trained on data from Lublin, and in experiments 2,4 and 6 on data from Nottingham.

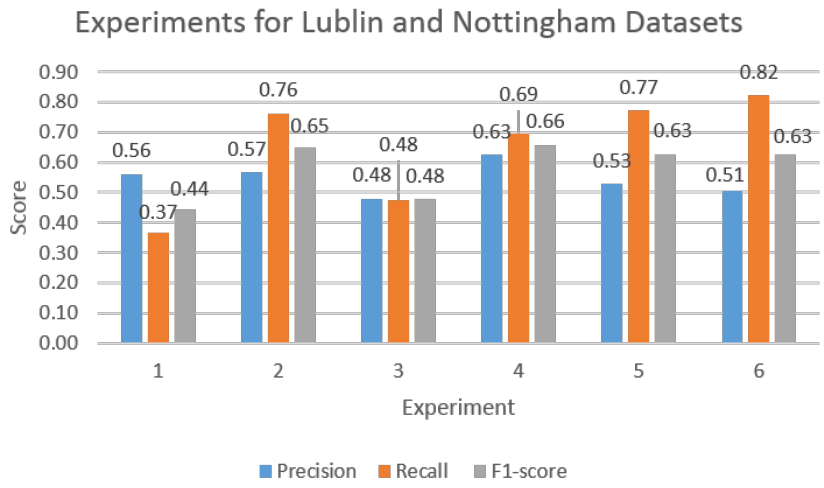


Figure 35: Precision, recall and F1-score for all the six experiments. In experiments 1,3 and 5 model was trained with data from Lublin, whereas in 2,4 and 6 on data from Nottingham.

The results presented in Figure 35 indicate that the best trade-off between precision, recall and F1-score was delivered by the model trained in experiment 4 (F1-score: 0.66, precision: 63 and recall: 0.69), the model trained on data from Nottingham (including images from mobile phone and with glass doors). Experiments 2 and 4, both using data from Nottingham, delivered the best results for precision (for experiment 2: 0.57) and F1-score (for experiment 2: 0.65) with the recall lower for experiment 4 (0.69) by 0.07 compared to recall for experiment 2 (0.76). Interestingly, the experiments conducted on specific combinations of data types for images from Nottingham demonstrated inferior recall scores to the corresponding experiments for images from Lublin (experiments 1-2, 3-4, 5-6), which means that there were more cases where the background was classified as a door for Nottingham training datasets. This suggests that training data from Lublin allows the model to better learn what is the background class, while models trained on Nottingham datasets learns faster how the front door instances looks like. This is presumably related to the fact that the architectural style within different parts of the UK is more similar than that between the two countries. Nevertheless, F1-score for all models trained on Nottingham dataset is higher than the one achieved by the models trained on Lublin datasets, implying that the location where the training data are collected is an important factor in the construction of training datasets.

4.2 Size of training dataset

The experiments described in section 4.1.1 focused on investigating how the location and type of training data impacted the front door detection capability. This section describes the results of experiments carried out to determine whether the number of training images impacted front door detection using the Mask R-CNN approach. In order to evaluate this, two further tests were carried out. In the first, the combined data from experiments 1-6 were used for training the model (525 images from both locations and all types). Then, size of the dataset was

reduced (size of this new dataset had 410 samples) by discarding photos showing the same door but at a different location inside the photo frame, or generated by the zooming function (Figure 36).



Figure 36: Example of different representations of the same front door instance.

Figure 37 presents the results obtained for experiment 7 (full set of data) and experiment 8 (reduced dataset). In both experiments the number of correctly detected doors was the same and equal to 72, while the number of background features classified as doors decreased from 50 to 33, for experiments 8 and 7 respectively. Therefore, the combination of data used in experiment 7 outperformed all others, with the smallest number of false positives (excepting experiment 1). The results obtained in experiment 7 had the best F-1 score (0.70) and precision (0.69) among all the eight tests and the recall (0.71) was still one of the best achieved (with the best recall equal to 0.37 delivered by a model trained in experiment 1). Based on the results of these two experiments, it can be concluded that the amount of training data is important for the correct determination of front door instances as well as reduction of the number of false positive, hence, the overall performance of the model.

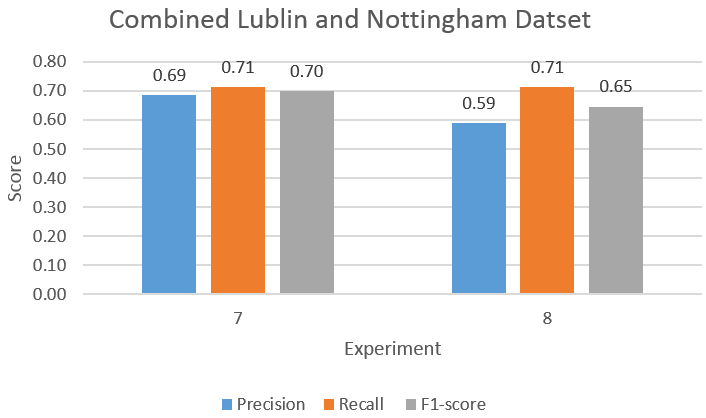


Figure 37: The training model results based on the number of input training samples.

Along with experiments 7 and 8, the model that gave the best overall results from experiments 1-6 was chosen, to investigate how these three models compared to the others. Both the datasets used in experiments 7 and 8 were composed of data from different locations, while the model used in experiment 4 was trained only with images from Nottingham. Figure 38 collates the results for the selected three experiments. While experiment 7 delivered the best results (F1-score: 0.7, precision:0.69 and recall:0.71), the superior performance of model 4 (F1-score: 0.0.66, precision:0.63 and recall:0.69) compared with model 8 (F1-score: 0.65, precision:0.59 and recall:0.71) suggests that the location where the training data were collected was more important factor in the correct detection of front doors using Mask R-CNN.

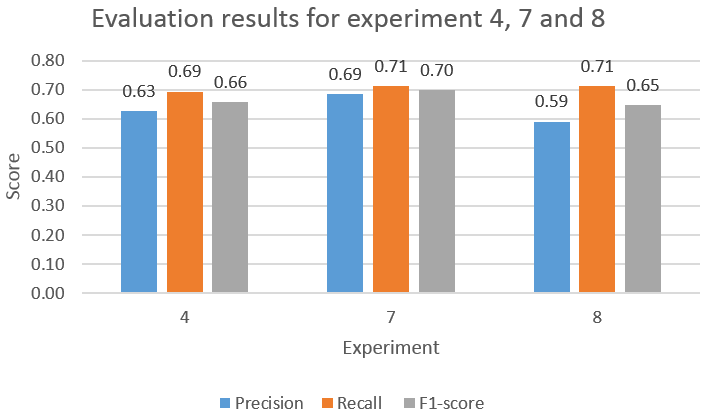


Figure 38: Evaluation metrics for experiments 4,7,8.

4.3 Comparing Instance Segmentation and Object detection

As mentioned before, it is believed that no research has been carried out on any CNN based technique to detect front doors from MMS images. This posed difficulties for drawing conclusions on the performance and usability of the method given the lack of comparable data. To overcome this, two experiments were conducted, with the application of a well known and established deep learning method used for object detection, YOLOv3. The tests included data combinations used previously in experiments 6 and 7. The choice of experiment 7 was clear as it showed the best results among all the Mask R-CNN based approaches. Experiment 6 did not deliver the best results, but was chosen because it had the highest detection rate of front door candidates. This suggests that data types and location allowed for the best detection score, but failed with the large number of false positives. Comparing these two models with the one obtained from training YOLOv3 should reveal how the different techniques using the same data performed in external door detection.

Figure 39 illustrates the results of training the YOLOv3 architecture, while Figure 40 illustrates examples of the front door detection on the Abingdon-Oxfrod test dataset. In experiment 9, the model was trained on a dataset including data from both locations (used in experiment 7), whereas the data used previously in experiment 6 were used to train YOLOv3 in experiment 10. Figure 39 shows that YOLOv3 trained on the full set of data from Nottingham showed marginally better results compared with the model trained on the full dataset (from both locations).

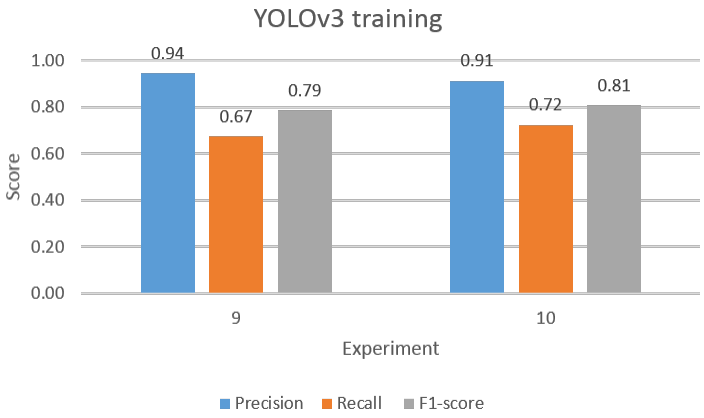


Figure 39: Evaluation metrics for the YOLOv3 experiments.



Figure 40: Example of the detection results for experiments 9 (top row) and 10 (bottom row), left and middle column: Abingdon © Crown copyright 2020 OS, Right column: Oxford Copyright © 2016 Oxford Robotics Institute

Figure 41 presents a comparison of the results for the models trained with YOLOv3 and their Mask R-CNN counterparts. It is clear that the overall detection rate of the YOLOv3 trained models produced higher overall results for front door detection. F1-score for the YOLOv3 trained models were 0.79 and 0.81 for experiments 9 and 10 respectively, whereas the best model trained on the Mask R-CNN architecture produced F1-score equal to 0.70. This presents superiority of the YOLOv3 over the Mask R-CNN for front door detection from MMS images. However, the rate of correct detections of door instances delivered by Mask R-CNN was higher than YOLOv3, suggesting that using an additional method offering a reduction in the number of false negatives obtained by the Mask R-CNN models could minimize the difference or even cause the Mask R-CNN to outperform YOLOv3.

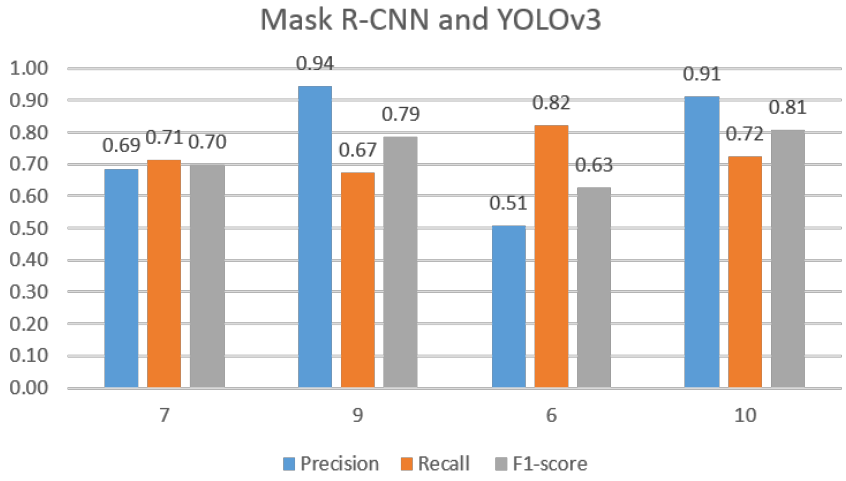


Figure 41: A comparison between Mask R-CNN and YOLOv3.

4.4 Sub-class analysis

The results presented in this work have focused on the ability of a trained model to detect front doors from a set of images, without distinguishing types of objects within the class. Table 10 presents results for all the experiments with the division of door and background classes.

Table 10: Detection results for each of the experiments with division into type of doors detected and type of objects from the background class which were incorrectly classified as doors.

		Door class					**Background class**			
		Door	**Glass**	**Other**	**Open**	**∑**	**Window**	**Bus stop**	**Other**	**∑**
Total	Abingdon	42	30	3	11	**86**	-	-	-	**-**
	Oxford	15	-	-	-	**15**	-	-	-	**-**
Expt. 1	Abingdon	21	10	3	2	**36**	25	-	-	**25**
	Oxford	1	-	-	-	**1**	4	-	-	**4**
Expt. 2	Abingdon	39	18	3	8	**68**	40	-	15	**55**
	Oxford	9	-	-	-	**9**	2	-	2	**4**
Expt. 3	Abingdon	20	15	3	7	**45**	41	1	6	**48**
	Oxford	3	-	-	-	**3**	2	-	2	**4**
Expt. 4	Abingdon	37	17	3	6	**63**	35	1	5	**41**
	Oxford	7	-	-	-	**7**	1	-	-	**1**
Expt. 5	Abingdon	30	23	3	11	**67**	56	2	6	**64**
	Oxford	10	-	-	-	**10**	2	-	4	**6**
Expt. 6	Abingdon	40	22	3	9	**74**	71	-	7	**78**
	Oxford	9	-	-	-	**9**	1	-	2	**3**
Expt. 7	Abingdon	38	16	3	9	**66**	28	-	4	**32**
	Oxford	6	-	-	-	**6**	1	-	-	**1**
Expt. 8	Abingdon	36	20	3	6	**65**	42	-	6	**48**
	Oxford	7	-	-	-	**7**	2	-	-	**2**
Expt. 9	Abingdon	34	18	2	11	**65**	1	-	3	**4**
	Oxford	3	-	-	-	**3**	-	-	-	**-**
Expt. 10	Abingdon	38	23	3	4	**68**	3	-	3	**6**
	Oxford	5	-	-	-	**5**	-	-	1	**1**

Sub-classes which can be distinguished within the two main classes are:

- Door class: doors (building front doors of private properties), glass doors (French doors and shop entrance doors as they are largely composed of glass), open doors and others (e.g. lattice doors).
- Background class: windows (the most common object in this class incorrectly classified as doors), bus stops (images with this street furniture were chosen as their material, similarly to windows, can lead to incorrect classification as doors) and others (e.g. trash bins).

From the information presented in Table 10, it can be seen that models trained on images from Lublin delivered lower detection results for the *door* sub-class compared with Nottingham or the datasets with mixed images (including test data from both Abingdon and Oxford). The average number of front doors detected for this sub-class for models trained on Lublin datasets was 28, which was 9 and 6 detected front door instances lower than the average for Nottingham and mixed datasets respectively. Furthermore, a similar trend also appeared when analysing the model performance for detecting *glass* doors. Although the number of correctly classified glass doors was highest for experiment 5 (Lublin dataset), this number was only one detection higher than the corresponding dataset from Nottingham (experiment 6) and equal to that achieved by training model with YOLOv3 architecture (experiment 10). On average, models trained on Lublin datasets detected 16 *glass* doors, whereas this number increased by 3 and 2 detection for Nottingham and mixed datasets respectively. In the case of *other* doors, all the identified instances (3 candidates in this sub-class) were detected, except for experiment 9 (2 candidates detected). However the sample was too small to allow a firm conclusion on the ability of trained models in detecting this particular type of door. Door detection on the Oxford dataset was satisfying for the experiments 2, 5 and 6 where the detection rate was above 60%. Based on the information contained in Table 10, it is apparent that Mask R-CNN performed better at door detection on the Oxford dataset. Although the detection rate for all the experiments where the model was trained on Mask R-CNN, (except experiments 1 and 3) achieved door detection on Abingdon dataset on the level between 73%-86%, the method suffers with the false positive cases. For experiments 5 and 6, the number of incorrectly classified objects belonging to the background class (70 for experiment 5 and 81 for experiment 6) was almost as high as the number of correctly classified door instances (77 for experiment 5 and 83 for experiment 6), resulting in a decreased overall performance. When analyzing the results, it is evident that YOLOv3, even though its door class detection may be lower compared to Mask R-CNN (e.g. 72 for experiment 7 and 68 for experiment 9), outperformed latter with the number of false positives, resulting in a superior overall performance.

4.5 Enhanced door detection using Mask R-CNN

The greatest disadvantage of using the Mask R-CNN in this project was the high number of false positives, which means that the background class was frequently misclassified as doors. The majority of these incorrectly identified objects were windows. This section introduces the post processing analysis conducted to investigate if there is any relationship between the width and height of bounding boxes around both the correctly and incorrectly detected doors, in order to reduce the number of false positives. The dataset used for this analysis was the one used in experiment 7 (data of all types and both locations,) which gave the best overall detection score among all the Mask R-CNN based experiments.

After testing the model on the test data, the coordinates of the bounding boxes created

around the objects classified as doors (both positive and negative) were used to calculate the width and height of these boxes. Next, the ratio between these two measures was determined. The analysis revealed that the typical ratio for doors does not exceed 0.8, therefore it allowed for the rejection of bounding boxes of a width-height ratio greater than 0.8. Table 11 presents the changes in evaluation metrics after applying the width-height ratio. It reveals improvement in all F1-score, precision and recall metrics. Increase between these parameters for enhanced experiment 7 compared to initial experiment 7 was from 0.70 to 0.74 for F1-score and 0.69 to 0.83 for precision. Applying the width-height ratio to experiment 7 reduced recall by 0.4 for the enhanced experiment 7 (reduction from 0.71 to 0.67). Figure 42 illustrates F1-score, precision and recall for experiment 7, enhanced 7 and 9 and therefore experiments using the combine dataset of Nottingham-Lublin training dataset.

Table 11: Comparison of the evaluation metrics before and after implementing the width-height ratio screening (Experiment 7 enhanced).

	Experiment 7	**Experiment 7 enhanced**
TP	72	68
FP	33	14
FN	29	33
Precision	0.69	0.83
Recall	0.71	0.67
F1-score	0.7	0.74

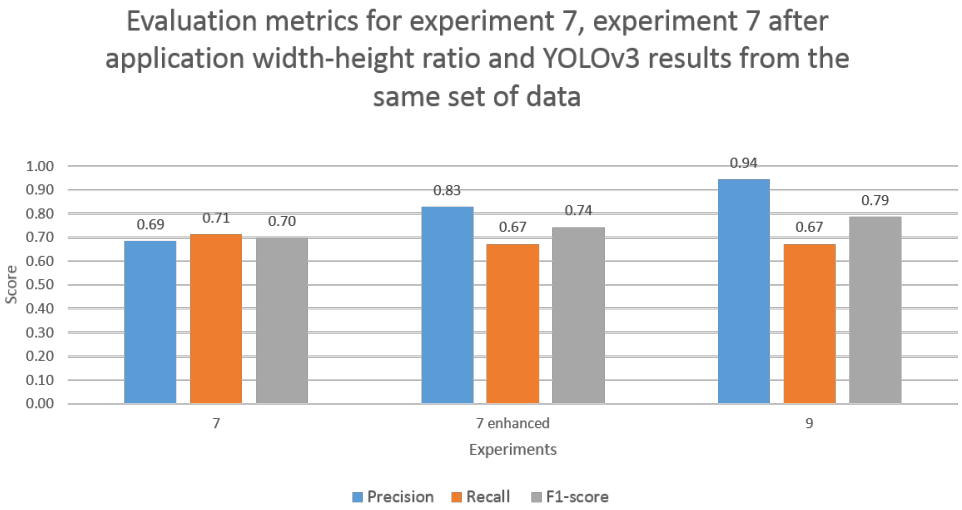


Figure 42: Evaluation metrics for experiment 7, experiment 7 after the application of the width-height ratio method and the YOLOv3 results from the same set of data.

Applying the width-height ratio to the data changed the quality in the final detection rate:

1. reduction of false positives from 33 to 14, which is 58%
2. reduction of true positives from 72 to 68, which is 6%

This operation made the Mask R-CNN results more comparable to those obtained from YOLOv3, indicating that this is an effective approach for detecting front doors from street level images.

4.6 Summary

This section presents the analysis performed on the results generated through the set of experiments where different data types, different location from where the training data were collected were used to train different DL architectures. From the results, it is evident that YOLOv3 gave superior overall results in the detection of front doors from street level images compared to Mask R-CNN. However, the addition of a single post-processing step based on screening according to the width-height ratio of the object, allowing an increase in the number of results delivered by Mask R-CNN, brought the performance of the models trained through the Mask R-CNN model closer to the results obtained by the YOLOv3 architecture. Moreover, the analysis shows that the location and the size of training dataset had a greater impact on the door detection capabilities than the data type used for training. Models trained on the data from Nottingham detected more front door instances. However the addition of data from different locations reduced the number of false positives.

5 DISCUSSION

This project explored the possibility of using instance segmentation for front door detection using Mobile Mapping Systems, which, it is believed, is the first study to investigate this problem using this particular method and data. The Mask R-CNN method implemented to address the main research question offered not only object detection (bounding box), but also included identification of the boundaries of objects at a pixel level, enriching traditional object detection methods with an additional *mask* element. Although such a solution offers a wider range of applications, it is also associated with more complex processing stages. This complexity, which relates not only to the instance segmentation problem, but more generally to all deep learning tasks within the computer vision field, often leads to deep learning being referred to as a *black box*. This term relates to the very complex and poorly understood architecture of the models. Hence, as the work of Rudin (2019) suggests, to trust the results provided by DL models (as they are heavily dependent on the training stage), it is necessary to trust the training data, its completeness and comprehensiveness. Therefore, the performance of a model based on a variety of training data is a focus of this study.

Earlier research on door detection presented in Section 2.1.1 focused on segmentation of building facade images, and door extraction was only one of many elements of the process (Schmitz and Mayer, 2016). The images used in these studies were acquired from street-level and often contained the entire facade of the building within their frame. Doors presented in this way were much smaller than those in this study and had much more contextual information, which may have contributed to better detection results. For example in study by Schmitz and Mayer (2016) the door detection accuracy was 95%, wheres in this study the detection of door instances for the best training data combination achieved an F1-sore equal to 0.74. In addition, the research was conducted on benchmark data, but not tested on external image sets. These studies had the advantage that the training and validation images were collected in same area and were carefully prepared. However, this does not answer the question of whether the implemented approach is capable of generalising when applied to other locations or whether the data used for training, collected in a different location to the test set, can still deliver satisfactory results. The only existing study that uses data from different locations to test the method performance is presented by Gadde et al. (2017). The results presented in this study showed that front doors, with the detection rate varied from 64% to 81%, were the most difficult to detect among all classes. In order to answer the question of whether it is still possible to obtain satisfactory results with data from different locations, as well as to assess whether the trained model can be generalized without regard to the origin of the training and testing data, various experiments were carried out in this project. The conducted research showed that the location of the data used to train the model had an impact on the final result. The analysis presented in Section 4.1.2 demonstrates the improved overall results for models trained on data from Nottingham and not Lublin. Therefore, it seems that if the images used in the throughout process pipeline originate from the same country, a higher detection score for front doors is expected, regardless of the amount of training data. It may be due to differences in architecture, as well as the appearance of the front door instances for different locations. Lublin's old town architecture is reminiscent of Nottingham's architecture, and the front doors in this area are similar to one of Nottingham's. In the Lublin training data set, the front doors from the area of the old town constituted about 25% of the whole set (40 out of 166 samples). Architecture in the rest of the city is characterized by a large number of apartment blocks with front doors of different appearance to those in the datasets from UK cities. Many of them are wider and most of them have a glass element. Additionally, front doors in apartment blocks often have a porch, which changes the context around them. Therefore these factors may be a reason for differences in detection score for models trained on training data originating from different locations (e.g. for a model trained on training dataset containing images from mobile

phone and glass shop door F1-score for Lublin was 0.47 and for Nottingham O.66). However, the model trained on the data set with the highest number of images from Lublin, was able to detect a similar number of front door instances (77 front doors detected) to the model trained on a corresponding combination of images from the Nottingham dataset (83 instances). This behaviour of the model may be related to the increased amount of training data or be because the samples from the GSV (front doors instances are more oblique) represented external doors with a height-width ratio closer to the one for front doors from Nottingham. These results suggest that it is possible to perform instance segmentation on street level images for front door detection with one of the following combinations:

- A smaller amount of training data from a location similar to that from which the images used for model testing were taken is sufficient for the detection of front door instances (e.g. different cities in the same country).
- A greater amount of training data from a completely different location is required for external door detection.

This findings suggest that with appropriate data selection in terms of location and the quantity of training images, it is possible to use images from any location and achieve good detection results. Along with the analysis on the impact of the location of the data, analysis on the impact of data type was conducted. Different types of images are related to the sensor from which they were acquired (mobile phone or Google Street View) or additional information included (glass doors). This was done in order to evaluate the effect of these variables on detection capabilities. The overall results from this analysis did not show an enhanced improvement in the detection ratio for models trained on images from Nottingham (the lowest F1 score was 0.63 and the highest 0.66) but a more significant improvement for the models trained on the images from Lublin (the lowest F1 score was 0.44 and the highest 0.62). The reason for the increase in detection acquired by models from Lublin datasets may be because glass doors were collected in Nottingham only, but also used in Lublin cases. This adds to Lublin dataset samples of front door instances from a country where test images originate. Moreover, the addition of images from the GSV introduced front door instances a the height-width ratio more akin to the front doors from Nottingham. However, as the additional type of data included to training datasets did not significantly enhance the overall detection rate for Nottingham cases, it is difficult to determine if the type of images or the increasing amount of data was responsible for this improvement. What is more, adding images of different types increased the number of false positives, which was opposite to the expected behaviour of the trained models. Among the background class, windows were the most significant source of the false positives. The preliminary results of this project highlighted this problem, which was a reason to add more images representing glass doors (in particular shop doors, as they are part of a large shop window). However, shop windows often include posters and advertisements, which show similarities in features with some front door instances (Figure 43). This may lead to confusion for a model during a training phase, as features for front doors and backgrounds (windows) are similar. This issue could be reduced in the case of entire image segmentation when additional classes are introduced to the model. Even though it is difficult to clearly assess the impact of the type of the image on the detection performance, variety in the type of images is not seen as the most essential factor for achieving the best model performance.

Schmitz and Mayer (2016) postulated that a small amount of training data (50 samples) is sufficient to obtain good segmentation results. However, unclear information about the recall and precision for these objects followed by the lack of statistical information on the number of doors that were used in the training phase, makes it difficult to relate their results to those presented in this work. The high accuracy achieved for door detection (95%) is highly possible,



Figure 43: Top row: examples of shop windows from test datasets (© Crown copyright 2020 OS) which features resemble doors from training datasets (bottom row).

when both training and test data sets originated from the same set of images. However, this does not agree with the results of this thesis. One of the objectives of this study was to determine if the amount of training data had an impact on the detection rate. Two experiments with the combined data from Nottingham and Lublin were conducted to answer this question. Based on the analysis of the results obtained from these experiments, it is evident that even though the number of detected doors was the same for both experiment 7 and experiment 8 (which differed in the number of samples: 525 and 410 respectively), the larger data set reduced the number of false positives, thus increasing the overall model performance. This may be because the algorithm had achieved the maximum capability for front door detection (regardless of the amount of training data), but increasing the training data allowed for a better understanding of what the background class represented, and therefore reduced the number of false positives. In other words, a model, instead of memorizing the examples, starts to learn how the training data was created. Nevertheless, in order to confirm this hypothesis, tests must be performed on an increased (e.g. by augmentation) number of images. The analyses carried out in this thesis indicated that a greater number of training samples as well as data from various locations has a positive impact on the detection capabilities. Nevertheless, when adding data to a training dataset, it is essential to control the behaviour of the model during the training phase, to detect and prevent model overfitting. (Kopelowitz and Engelhard, 2019).

Although the obtained results demonstrate the efficacy of the Mask R-CNN method for instance segmentation of front doors from images acquired by a Mobile Mapping System, the lack of relevant studies on the same topic does not allow for a more critical evaluation of the performance of the method. For this reason, it was decided to conduct experiments with the selected data sets used to train Mask R-CNN to also train a model commonly used for object detection, YOLOv3. The YOLOv3 method produced better overall results (F1 score for experiment 9: 0.79 and for experiment 10: 0.81) due to the significant reduction in false positives (4 and 7 for experiment 9 and experiment 10 respectively), as the number of detected doors was no greater than that delivered by the best model trained by the Mask R-CNN architecture. The lower number of false positives delivered by the YOLOv3 models may suggest that the model trains better on what is not the front door instance. This may be because the YOLOv3 task is to detect objects only, whereas the Mask R-CNN does both

object detection and images segmentation. It is in line with the previous implication that the larger size of the training dataset helps the Mask R-CNN models to learn better about the background class and thus, reduce the number of false positives. Moreover, as presented in work of Ruiz-Santaquiteria et al. (2020), the main drawback of Mask R-CNN is the limitation of the segmentation step due to the detection, which may also be a reason for the high number of false positives presented in this thesis.

A detailed analysis of the results indicates problems with the high numbers of false positives produced by Mask R-CNN models. The vast majority of these false positives were associated with the model facing challenges with distinguishing between doors and windows. During the visual inspection of the results, it was noted that the bounding box around windows incorrectly identified as doors often had a different height-width ratio compared with front door instances. Implementation of so called *shape grammar rules* is commonly used in building reconstruction (Mathias et al., 2011; Teboul et al., 2011), and was the inspiration for investigating whether the relationship between the height and width of the detected objects related to the positive and negative samples. As a result of the analysis of the width-height ratio, it was found that, this ratio could not be greater than 0.8 for doors. Consequently, a post-processing step was implemented for the Mask R-CNN approach to filter detected objects that exceeded this ratio, which resulted in a decrease in false positives by 58% with a simultaneous decrease of true positives by 6%. Evaluation of the results with the applied height-width ratio for experiment 7, (the one that gave the best results for the R-CNN mask approach), presents this improvement for all the evaluation metrics. Thus the F1-score increased from 0.7 to 0.74, precision increased from 0.69 to 0.83 and recall decreased from 0.71 to 0.67. This proves that application of the height-width ratio enhanced the front door detection capabilities with Mask R-CNN approach.

6 CONCLUSION

The motivation for this research was to assess the detection capabilities of the Mask R-CNN approach for extraction of front door instances. Front door detection can help creating building models of higher LoD, or improve management and logistics for delivery services and waste management companies. Using a CNN in conjunction with MMS images can make door detection more automated and efficient, regardless of where the front door is located, and thus introducing robustness and flexibility to the front door detection pipeline. For this reason, the the aim of this research was to determine whether deep-learning based instance segmentation has potential for exterior front door extraction from images acquired using a Mobile Mapping System. The main research question was To what extend is Mask R-CNN suitable for instance segmentation of doors from street-level images? The five research objectives were formulated to assist with answering this question.

To determine the capabilities of Mask R-CNN trained models on front door detection, several experiments with different configurations of training data were conducted. The datasets examined in this research varied based on their location and the sensors used to collect them. These differences facilitated the implementation of experiments to determine how different combinations of these datasets might impact the final results and to assess which combination offered the best outcome for future work.

The presented experiments and the subsequent analysis show that the Mask R-CNN has potential for performing external door detection from images collected by Mobile Mapping Systems. The conducted experiments demonstrated that, although the use of training images from only one location allowed the correct identification of objects, it could also produce a large number of false positives. This problem was reduced by using data from both sites (Nottingham and Lublin) to train the model. The results obtained from this test indicate that a combination of data from different locations improves the overall model performance by reducing the number of objects of the background class that are incorrectly classified as doors, assuming the amount of input data is sufficient. This demonstrates the robustness of the Mask R-CNN method in detecting front door instances regardless of the location where training and test datasets were collected and, thus, its potential to be used in various locations.

Several experiments in this research were also conducted to assess how different types of data influence instance segmentation. The results showed that there was no notable impact on the final results, which offers flexibility when it comes to collecting training data. The only important consideration is to avoid introducing training samples with a highly distorted height-width ratio. This is related to the fact that the analysis carried out on enhancing the performance of the Mask R-CNN method using this ratio was found to be effective in improving final detection score.

Even though the presented study shows the potential of using Mask R-CNN for front door segmentation and extraction, there is still scope for further investigation and improvement. For instance, instance segmentation is often used to segment an entire image. For this reason, it would be valuable to assess its usability for segmenting entire images used in this research. Furthermore, part of the analysis in this work showed detection of the door class for four of its sub-classes. One of the future experiments could focus on determining if such detailed segmentation would give satisfactory results. Mobile Mapping Systems, often in addition to camera data, collect data from a scanner. An investigation of how segmentation of data from these two sensors can complement the results would be a challenging but exciting option for future work. Moreover, the analysis presented in this study suggests that increasing the amount of training data improves the model detection capabilities. For this reason, further studies on the changes in detection rate based on the size of training dataset could be con-

ducted in more detail. What is more, data used for the testing phase originated from the UK; hence it would be valuable to continue the analysis of detection and segmentation capabilities of Mask R-CNN with extension to other test locations. Deep learning is a fast-evolving field with the constant increase in the development of the algorithms. Mask R-CNN proves its usability and robustness for front door detection, however, it would be an undeniable advantage to conduct studies on different deep learning architectures to find the one that is most suitable for front door detection.

References

Abdulla, Waleed (2017). *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. https://github.com/matterport/Mask_RCNN.

Arachchige, N Hetti and Sanka Perera (2014). 'Automatic modelling of building façade objects via primitive shapes'. In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 40.3, p. 115.

Bacharidis, Konstantinos et al. (2020). '3D Building Façade Reconstruction Using Deep Learning'. In: *ISPRS International Journal of Geo-Information* 9.5, p. 322.

Becker, Susanne and Norbert Haala (2007). 'Refinement of building fassades by integrated processing of LIDAR and image data'. In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Science* 36.2007, pp. 7–12.

Borra, Surekha et al. (2019). *Satellite image analysis: Clustering and classification*. Springer.

Boureau, Y-Lan et al. (2010). 'A theoretical analysis of feature pooling in visual recognition'. In: pp. 111–118.

Bradski, Gary and Adrian Kaehler (2008). *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc."

Cai, Yiming et al. (2020). 'Geo-Location Algorithm for Building Targets in Oblique Remote Sensing Images Based on Deep Learning and Height Estimation'. In: *Remote Sensing* 12.15, p. 2427.

Chen, Meixu et al. (2020). 'Quantifying the Characteristics of the Local Urban Environment through Geotagged Flickr Photographs and Image Recognition'. In: *ISPRS International Journal of Geo-Information* 9.4, p. 264.

Chen, Wei et al. (2014). 'Door recognition and deep learning algorithm for visual based robot navigation'. In: pp. 1793–1798.

Cordts, Marius et al. (2016). 'The cityscapes dataset for semantic urban scene understanding'. In: pp. 3213–3223.

Díaz-Vilariño, L et al. (2017). 'Indoor modelling from SLAM-based laser scanner: Door detection to envelope reconstruction'. In: *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci* 42, pp. 345–352.

Dutta, Abhishek and Andrew Zisserman (2019). 'The VIA Annotation Software for Images, Audio and Video'. In: *Proceedings of the 27th ACM International Conference on Multimedia*. MM '19. Nice, France: ACM. ISBN: 978-1-4503-6889-6/19/10. DOI: 10.1145/3343031.3350535. URL: https://doi.org/10.1145/3343031.3350535.

Fathalla, Radwa and George Vogiatzis (2017). 'A deep learning pipeline for semantic facade segmentation'. In:

Feng, Di et al. (2020). 'Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges'. In: *IEEE Transactions on Intelligent Transportation Systems*.

Gadde, Raghudeep et al. (2017). 'Efficient 2D and 3D facade segmentation using auto-context'. In: *IEEE transactions on pattern analysis and machine intelligence* 40.5, pp. 1273–1280.

Gählert, Nils et al. (2020). 'Cityscapes 3D: Dataset and Benchmark for 9 DoF Vehicle Detection'. In: *arXiv preprint arXiv:2006.07864*.

Girshick, Ross (2015). 'Fast r-cnn'. In: pp. 1440–1448.

Goodfellow, Ian et al. (2016). *Deep learning*. MIT press.

Ha, Q. et al. (2017). 'MFNet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes'. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5108–5115.

Haque, Akkas et al. (2020). 'UAV Autonomous Localization Using Macro-Features Matching with a CAD Model'. In: *Sensors* 20.3, p. 743.

Hartling, Sean et al. (2019). 'Urban tree species classification using a WorldView-2/3 and LiDAR data fusion approach and deep learning'. In: *Sensors* 19.6, p. 1284.

He, Kaiming et al. (2017). 'Mask r-cnn'. In: pp. 2961–2969.
He, Linbo (2019). *Improving 3D Point Cloud Segmentation Using Multimodal Fusion of Projected 2D Imagery Data: Improving 3D Point Cloud Segmentation Using Multimodal Fusion of Projected 2D Imagery Data*.
Hebbalaguppe, Ramya et al. (2017). 'Telecom Inventory management via object recognition and localisation on Google Street View Images'. In: pp. 725–733.
Hofmarcher, Markus et al. (2019). 'Visual scene understanding for autonomous driving using semantic segmentation'. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, pp. 285–296.
Huang, Yen-Kai and Vivian Yang (2017). 'Street View Segmentation using FCN models'. In: *s.l.: Stanford University*.
Hubel, David H and Torsten N Wiesel (1959). 'Receptive fields of single neurones in the cat's striate cortex'. In: *The Journal of physiology* 148.3, p. 574.
Ibrahim, Mohamed R et al. (2019). 'WeatherNet: Recognising weather and visual conditions from street-level images using deep residual learning'. In: *ISPRS International Journal of Geo-Information* 8.12, p. 549.
Juenemann, Christopher et al. (2010). 'Robust door detection'. In: *Final Project, Course EE368, Stanford Electrical Engineering Department, Stanford, California*.
Kathuria, Ayoosh (2018). *What's new in YOLO v3?* URL: `https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b`. (accessed: 04.08.2020).
Keiser, A. (2017). *Hayo:What is computer vision?* URL: `https://hayo.io/computer-vision/`.
Kirillov, Alexander et al. (2019). 'Panoptic segmentation'. In: pp. 9404–9413.
Kopelowitz, Evi and Guy Engelhard (2019). 'Lung Nodules Detection and Segmentation Using 3D Mask-RCNN'. In: *arXiv preprint arXiv:1907.07676*.
Korc, Filip and Wolfgang Förstner (2009). 'eTRIMS Image Database for interpreting images of man-made scenes'. In: *Dept. of Photogrammetry, University of Bonn, Tech. Rep. TR-IGG-P-2009-01*.
Krizhevsky, Alex et al. (2012). 'Imagenet classification with deep convolutional neural networks'. In: pp. 1097–1105.
Krylov, Vladimir A et al. (2018). 'Automatic discovery and geotagging of objects from street view imagery'. In: *Remote Sensing* 10.5, p. 661.
Kyrkou, Christos and Theocharis Theocharides (2019). 'Deep-Learning-Based Aerial Image Classification for Emergency Response Applications Using Unmanned Aerial Vehicles.' In: pp. 517–525.
Laina, Iro et al. (2016). 'Deeper depth prediction with fully convolutional residual networks'. In: pp. 239–248.
Laroca, Rayson et al. (2018). 'A robust real-time automatic license plate recognition based on the YOLO detector'. In: pp. 1–10.
Law, Stephen et al. (2017). 'An application of convolutional neural network in street image classification: The case study of London'. In: pp. 5–9.
Le, Quoc V (2013). 'Building high-level features using large scale unsupervised learning'. In: pp. 8595–8598.
LeCun, Yann et al. (2015). 'Deep learning'. In: *nature* 521.7553, pp. 436–444.
Lin, Chih-Yang et al. (2020). 'Global-and-Local Context Network for Semantic Segmentation of Street View Images'. In: *Sensors* 20.10, p. 2907.
Lin, Tsung-Yi et al. (2017a). 'Feature pyramid networks for object detection'. In: pp. 2117–2125.
Lin, Tsung-Yi et al. (2017b). 'Focal loss for dense object detection'. In: pp. 2980–2988.
Liu, Hantang et al. (2017). 'Deepfacade: A deep learning approach to facade parsing'. In:
Liu, Li et al. (2020). 'Deep learning for generic object detection: A survey'. In: *International journal of computer vision* 128.2, pp. 261–318.
Liu, Shu et al. (2018). 'Path Aggregation Network for Instance Segmentation'. In:

Llopart, Adrian et al. (2017). 'Door and cabinet recognition using convolutional neural nets and real-time method fusion for handle detection and grasping'. In: pp. 144–149.
Lu, Dengsheng and Qihao Weng (2007). 'A survey of image classification methods and techniques for improving classification performance'. In: *International journal of Remote sensing* 28.5, pp. 823–870.
Maddern, Will et al. (2017). '1 year, 1000 km: The Oxford RobotCar dataset'. In: *The International Journal of Robotics Research* 36.1, pp. 3–15.
Mathias, Markus et al. (2011). 'Procedural 3D building reconstruction using shape grammars and detectors'. In: *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*. IEEE, pp. 304–311.
Mohan, Rohit and Abhinav Valada (2020). 'Efficientps: Efficient panoptic segmentation'. In: *arXiv preprint arXiv:2004.02307*.
Muehlemann, Anton (2019). *TrainYourOwnYOLO: Building a Custom Object Detector from Scratch*. URL: https://github.com/AntonMu/TrainYourOwnYOLO.
O'Mahony, Niall et al. (2019). 'Deep learning vs. traditional computer vision'. In: pp. 128–144.
Quintana, Blanca et al. (2018). 'Door detection in 3D coloured point clouds of indoor environments'. In: *Automation in Construction* 85, pp. 146–166.
Redmon, Joseph and Ali Farhadi (2018). 'Yolov3: An incremental improvement'. In: *arXiv preprint arXiv:1804.02767*.
Redmon, Joseph et al. (2015). 'You Only Look Once: Unified, Real-Time Object Detection'. In: *CoRR* abs/1506.02640. arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.
Redmon, Joseph et al. (2016). 'You only look once: Unified, real-time object detection'. In: pp. 779–788.
Ren, Shaoqing et al. (2015). 'Faster r-cnn: Towards real-time object detection with region proposal networks'. In: pp. 91–99.
Rudin, Cynthia (2019). 'Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead'. In: *Nature Machine Intelligence* 1.5, pp. 206–215.
Ruiz-Santaquiteria, Jesus et al. (2020). 'Semantic versus instance segmentation in microscopic algae detection'. In: *Engineering Applications of Artificial Intelligence* 87, p. 103271.
Schmitz, Matthias and Helmut Mayer (2016). 'A convolutional network for semantic facade segmentation and interpretation.' In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 41.
Sirirattanapol, Chairath et al. (2019). 'Bangkok CCTV image through a road environment extraction system using multi-label convolutional neural network classification'. In: *ISPRS International Journal of Geo-Information* 8.3, p. 128.
Sokolova, Marina et al. (2006). 'Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation'. In: *Australasian joint conference on artificial intelligence*. Springer, pp. 1015–1021.
Springenberg, Jost Tobias et al. (2014). 'Striving for simplicity: The all convolutional net'. In: *arXiv preprint arXiv:1412.6806*.
Su, Tengfei et al. (2020). 'Multi-Spectral Image Classification Based on an Object-Based Active Learning Approach'. In: *Remote Sensing* 12.3, p. 504.
Sun, Pei et al. (June 2020). 'Scalability in Perception for Autonomous Driving: Waymo Open Dataset'. In:
Szeliski, Richard (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.
Teboul, Olivier et al. (2011). 'Shape grammar parsing via reinforcement learning'. In: *CVPR 2011*. IEEE, pp. 2273–2280.
Teuwen, Jonas and Nikita Moriakov (2020). 'Convolutional neural networks'. In: *Handbook of Medical Image Computing and Computer Assisted Intervention*. Elsevier, pp. 481–501.

Tian, YingLi et al. (2013). 'Toward a computer vision-based wayfinding aid for blind persons to access unfamiliar indoor environments'. In: *Machine vision and applications* 24.3, pp. 521–535.

Tian, Yunong et al. (2019). 'Apple detection during different growth stages in orchards using the improved YOLO-V3 model'. In: *Computers and electronics in agriculture* 157, pp. 417–426.

Tu, Bing et al. (2019). 'Hyperspectral image classification with multi-scale feature extraction'. In: *Remote Sensing* 11.5, p. 534.

Waleed, A. (2018). *Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow*. URL: https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46.

Xiong, Yuwen et al. (2019). 'Upsnet: A unified panoptic segmentation network'. In: pp. 8818–8826.

Xu, Bing et al. (2015). 'Empirical evaluation of rectified activations in convolutional network'. In: *arXiv preprint arXiv:1505.00853*.

Yamashita, Rikiya et al. (2018). 'Convolutional neural networks: an overview and application in radiology'. In: *Insights into imaging* 9.4, pp. 611–629.

Yang, Tao et al. (2019). 'Semantic segmentation via highly fused convolutional network with multiple soft cost functions'. In: *Cognitive Systems Research* 53, pp. 20–30.

Yang, Xiaodong and Yingli Tian (2010). *Robust door detection in unfamiliar environments by combining edge and corner features*, pp. 57–64.

Zeiler, Matthew D and Rob Fergus (2013). 'Stochastic pooling for regularization of deep convolutional neural networks'. In: *arXiv preprint arXiv:1301.3557*.

Zhang, Ce et al. (2018). 'A hybrid MLP-CNN classifier for very fine resolution remotely sensed image classification'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 140, pp. 133–144.

Zhang, Chi et al. (2019). 'Detecting large-scale urban land cover changes from very high resolution remote sensing images using cnn-based classification'. In: *ISPRS International Journal of Geo-Information* 8.4, p. 189.

Zhang, Tianwen and Xiaoling Zhang (2019). 'High-speed ship detection in SAR images based on a grid convolutional neural network'. In: *Remote Sensing* 11.10, p. 1206.

Zhang, Weixing et al. (2018). 'Using deep learning to identify utility poles with crossarms and estimate their locations from google street view images'. In: *Sensors* 18.8, p. 2484.

Zhang, Ziyu et al. (2016). 'Instance-level segmentation for autonomous driving with deep densely connected mrfs'. In: pp. 669–677.

Zhao, Xin et al. (2019). 'Use of unmanned aerial vehicle imagery and deep learning unet to extract rice lodging'. In: *Sensors* 19.18, p. 3859.

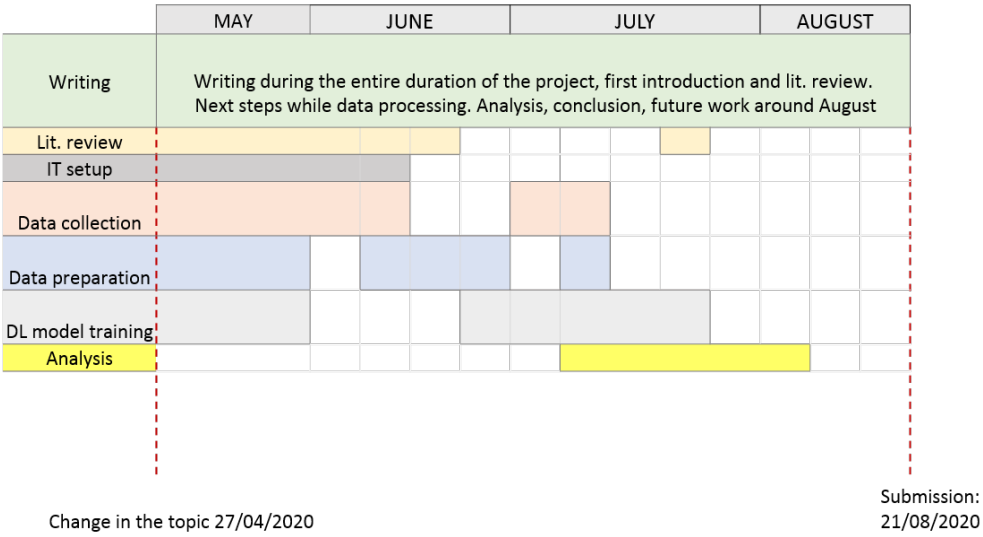
Zhao, Zhong-Qiu et al. (2019). 'Object detection with deep learning: A review'. In: *IEEE transactions on neural networks and learning systems* 30.11, pp. 3212–3232.

Zhou, Bolei et al. (2017). 'Places: A 10 million image database for scene recognition'. In: *IEEE transactions on pattern analysis and machine intelligence* 40.6, pp. 1452–1464.

Zhou, Shibo et al. (2020). 'Deep SCNN-based Real-time Object Detection for Self-driving Vehicles Using LiDAR Temporal Data'. In: *IEEE Access* 8, pp. 76903–76912.

A Project Management Plan

Gantt chart below presents the project management plan. The initial MRes project involved point cloud segmentation. Therefore literature review, as well as investigation of commercial fools and programming libraries, was conducted. Unfortunately, due to challenges associated with the COVID-19 pandemic, project scope needed adjustment. A new topic on image segmentation was agreed in late April, however, delay in receiving the data from the industrial partner allowed to fully start working from late May.



B Data Management Plan

1. Provide the title and briefly describe the aim and objectives of MRes project
   Project title Instance Segmentation of FrontDoors in Mobile Mapping System Images
   Aim Determine to what extent is Mask R-CNN suitable for instance segmentation of doors from street-level images
   - To determine the usefulness of deep learning based instance segmentation in extracting door instances from street-level images
   - To determine the influence of the amount of training samples on the ability of a trained model to extract door instances
   - To assess the effects of different types of training data on front door detection
   - To evaluate the robustness of a trained model in detecting front doors in independent locations
   - To compare the performance of instance segmentation and object detection for front door detection from street-level images
2. What data will be produced? (Data types, format, standards, scale and method)
   - Input data used to conduct project:
     - 2D images (in digital form) in formats: .png, .jpg
     - files with annotation in formats: .txt, .json
   - Output data produced during the project
     - binary masks and bounding box coordinates from Mask R-CNN and YOLOv3 training in formats: .txt, .json, .csv, .jpg
     - weights for trained models in format: h5
     - tables with metrics used to ealluate
3. What metadata standards will you use? (Metadata content and format)
   - .csv and .txt with model training details, files stored in separate folder
   - .txt file with list of images used in the project with information when and where they were collected
4. How will your data be structured and stored? (Project storage)
   - Estimate size of data used and produced will be around 200GB.
   - Data will be stored in multiple folders on the laptop hard drive as well as Google Drive. This was due to a processing which was conducted using Google Colab platform. The easiest way to work with this platform is to store all the necessary data (images, codes) on Google Drive
   - Things which need to be kept in mind:
     - Name folders properly, so the file system will be easier to navigate
     - Decide on a consistent way of naming folders, so it is easy to find the necessary information
     - Hierarchical structure of folders with less folders for broader topics
     - Work separation. As the number of files and folders will increase in time, it will be useful to separate completed work from the ongoing
5. Outline the approach to data selection and long-term preservation?
   - Data used in project in majority will originate from industrial sponsor – OS. Thus it will need to be discussed on what terms data will be available.

6. Who has responsibility for implementing the DMP and are resources required?

    - Responsibility for data management plan is on the students's side, which may change with the end of the project due to the use of the industrial partner data
    - Responsibility on data access need to be discussed in more details with the industrial partner
    - At the moment there is no need for additional storage costs, nor hardware/software support.