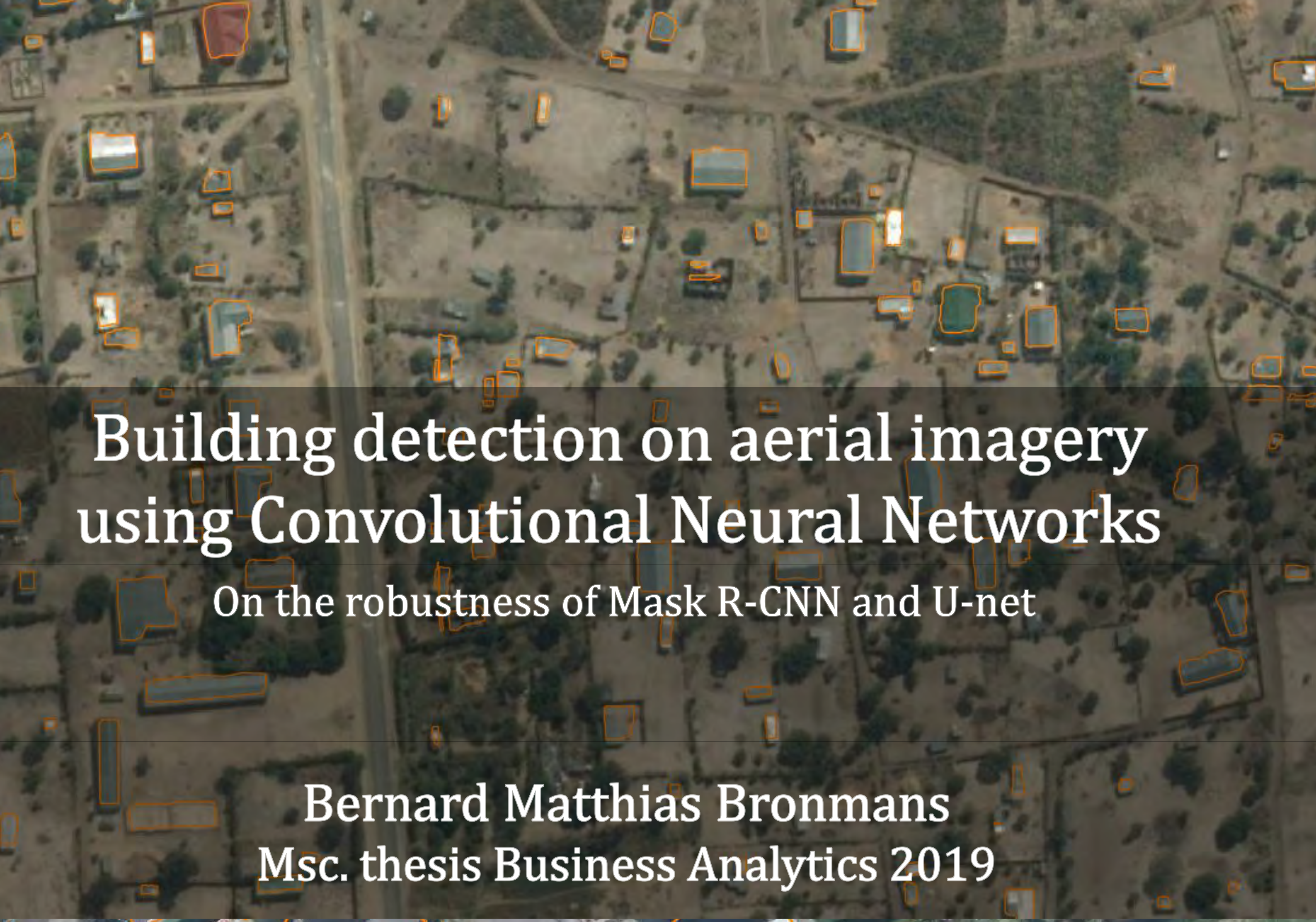


Building detection on aerial imagery using Convolutional Neural Networks

On the robustness of Mask R-CNN and U-net

Bernard Matthias Bronmans

Msc. thesis Business Analytics 2019

READAR real estate radar

VU VRIJE UNIVERSITEIT AMSTERDAM

510 AN INITIATIVE OF THE NETHERLANDS RED CROSS

Cover illustrations
Top: Satellite imagery of Mongu, Zambia with building outlines as predicted by a Mask R-CNN model.

Bottom: Aerial imagery of Heerlen, the Netherlands with building outlines as predicted by a (Res)U-net model.

BUILDING DETECTION ON AERIAL IMAGERY USING CONVOLUTIONAL NEURAL NETWORKS

ON THE ROBUSTNESS OF MASK R-CNN AND U-NET

A thesis submitted to the Vrije Universiteit Amsterdam in partial fulfillment of the requirements for the degree of

Master of Science in Business Analytics

by

Bernard Matthias Bronmans

April 2019

Bernard Matthias Bronmans: *Building detection on aerial imagery* (2019)

Supervisors:	Dr. Peter Bloem
	Msc. Rein van 't Veer
Co-reader:	Dr. Ger Koole
Company supervisors:	ir. Sven Briels (Readar)
	Msc. Arjen Crince (510)

Vrije Universiteit Amsterdam
Faculty of Science
De Boelelaan 1081a
1081 HV Amsterdam

Readar
Maliestraat 3
3581 SH Utrecht

510 - An initiative of
The Netherlands Red Cross
Anna van Saksenlaan 50
2593 HT Den Haag

SUMMARY

Mapping buildings on aerial imagery is a task often performed manually. This is currently the case for Readar, a company which extracts building and rooftop related information from aerial imagery, and the Missing Maps initiative and the 510 data team of the Netherlands Red Cross Society (NRCS). The goal of Missing Maps and the 510 data team is to map uncharted areas using satellite imagery to assist humanitarian aid projects.

In this report two perspectives are taken on the building localization task: First, we assume it is an object detection task and implement an object detection Convolutional Neural Network (CNN) called Mask R-CNN. Secondly, we regard the building localization task as an image segmentation task and implement a CNN called U-net. The robustness of both CNNs is tested with regards to geographical location, image resolution and building geometry. Data used includes imagery from four continents in 10, 30 or 60 cm ground resolution.

The Mask R-CNN model has shown to be surprisingly geographically robust. A Mask R-CNN model trained on a dataset located in North America managed to obtain a 0.98 precision and 0.60 recall score (building-wise) on a dataset containing 29,000 buildings over 67 km^2 of imagery of the Zambian city of Mongu in 48 minutes without any retraining or adjustments. Furthermore the model showed promising results on test sets located in Malawi, Peru and the Netherlands. However, the Mask R-CNN model proved to be sensitive to the combination of building size and image resolution, making the model inherently sensitive to image resolution. Mask R-CNN also has a limited range of expected building aspect ratios. These are sufficient for the vast majority of detached buildings but reduce performance on connected buildings with similar rooftops which are common in densely built areas. For the use case of the Red Cross, the limitations of this Mask R-CNN model are compensated for by its geographical robustness.

The U-net model showed a moderate geographical robustness, with a model trained on the same North American dataset scoring 0.63 on precision and 0.62 on recall (pixel-wise) on parts of the Zambian city of Lusaka. However, contrary to Mask R-CNN, the U-net architecture is more robust to image resolution and does not have any limitations with regards to building geometry. This is allows the U-net to perform well in densely built areas, resulting in a 0.81 score on precision and 0.86 on recall (pixel-wise) on the Dutch city of Heerlen, translating to a 0.81 score on precision and 0.92 on recall (building-wise). The efficiency of the U-net model and its robustness with respect to image resolution and building geometry make it suitable for the use case of Readar.

Another conclusion is that performance of both CNNs is heavily influenced by the quality of the annotated ground truth labels, image quality and the alignment between the ground truth and the imagery. For a given area of interest lacking accurate ground truth labelling, it might be recommended to train a CNN on a dataset of higher quality from a different area. This internship introduces transfer validation sets which monitor the transferability of a model to a different dataset during the training process. Other main recommendations include the addition of height data as model input for Readar and for the Missing Maps project to improve registration of the active imagery source used during the manual mapping process.

PREFACE

You now have the thesis in front of you that is my final project of the Master's programme Business Analytics at the Vrije Universiteit Amsterdam. The decision to study Business Analytics, or as I prefer the legacy name "Bedrijfswiskunde en Informatica" (roughly "Applied Mathematics and Computer Sciences"), has proven to be the right one for me.

At Business Analytics, the graduation thesis is part of an internship where the obtained knowledge and skills are put into practice to have a real, positive impact on a business case. This internship was a collaboration between Readar and the 510 data team of the Netherlands Red Cross Society. The mutual desire between Readar and 510 was to automatically detect buildings on aerial imagery. Both parties were interested in exploring the potential of Machine Learning techniques for this objective. The past few months have been a fun, inspiring and enriching learning experience for me and allowed me to meet some great people.

I would like to express my gratitude for everyone supporting me during my internship. Special thanks go to my supervisors at the Vrije Universiteit, Peter Bloem and Rein van 't Veer for providing me with weekly guidance and valuable feedback. I would also like to thank Readar for offering this opportunity to me and for providing an impressive amount of resources in terms of compute power, availability of high-quality data and a great daily lunch. In particular, I'd like to thank Sven Briels for his technical supervision and Jean-Michel Renders for sharing his Machine Learning expertise with me. The 510 data team of The Netherlands Red Cross Society provided me with an inspiring environment and a humbling amount of interest in the progress of my internship. From 510, I would specifically like to thank Arjen Crince and Jurg Wilbrink for their much appreciated knowledge and assistance on all geo-information related challenges on my path. Special thanks also go to Koos Krijnders, for providing me with the opportunity to present my work at the International Institute for Geo-Information Science and Earth Observation and his warm hospitality in general. I also gratefully look back on valuable discussions and knowledge sharing sessions on the subject of automated building detection with the DeepVGI team within 510.

Bernard Bronmans

Utrecht, March 2019

CONTENTS

ACRONYMS

AP	Average Precision	47
AR	Average Recall	47
BAG	Basis Administratie Gebouwen	2
BCE	Binary Cross Entropy	28
BGT	Basisregistratie Grootschalige Topografie	2
CNN	Convolutional Neural Network	9
COCO	Common Objects in Context	47
CPU	Central Processing Unit	9
FCN	Fully Convolutional Network	11
GPU	Graphics Processing Unit	9
ILSVRC	ImageNet Large Scale Visual Recognition Challenge	9
IoU	Intersection over Union	28
Mask R-CNN	Mask Region-based Convolutional Neural Network	11
MLP	Multilayer perceptron	13
NMS	non-maximum suppression	35
NLRC	Netherlands Red Cross Society	1
OSM	OpenStreetMap	39
ReLU	Rectified Linear Unit	17
ResU-net	Residual U-net	67
RGB	Red Green Blue	19
RPN	Region Proposal Network	35
SGD	Stochastic Gradient Descent	25
TTA	Test Time Augmentation	31

1 INTRODUCTION

1.1 BACKGROUND

Missing Maps
In August 2014 a new commercial satellite was launched, called WorldView-3 by DigitalGlobe. This satellite was the first in its kind to provide global coverage for imagery with a 31 cm ground resolution. This high-resolution imagery is detailed enough to accurately recognize individual buildings.

A few months later, in November 2014 the Missing Maps project was started. Missing Maps is an initiative in which volunteers can map areas in the developing world that have not been mapped yet. The mapping is done based on satellite imagery, where the WorldView-3 satellite is one of the main imagery providers. The aim of Missing Maps is to assist humanitarian aid projects by improving the geographic information available for active and future project areas. The results of the mapping are published in OpenStreetMap, a platform for open map data. The objective of Missing Maps is stated as follows:

> "To map the most vulnerable places in the developing world, in order that international and local NGOs and individuals can use the maps and data to better respond to crises affecting the areas."

510
Currently, Missing Maps is being maintained by the 510 data team, associated with the Netherlands Red Cross Society (NLRC). The 510 data team aims to improve the quality and effectiveness of humanitarian aid by gathering, processing and analyzing data. The name 510 refers to the total surface area of the earth in million square kilometres. The mission statement from 510:

> "Shape the future of humanitarian aid by converting data into understanding, and put it in the hands of humanitarian relief workers, decision makers and people affected, so that they can better prepare for and cope with disasters and crises."

The 510 team has been following the progress being made in the area of Machine Learning, especially in the field of Computer Vision. Given the fact that mapping individual buildings on satellite imagery requires a lot of manual effort, the idea persists within the 510 team that Computer Vision models could at one point drastically increase the speed of large-scale mapping projects.

Readar
That same idea is also being held by Readar. Readar is a company founded in 2015 which extracts building and rooftop data from aerial imagery. Readar offers services including 3-dimensional volume models for buildings, asbestos detection on building rooftops, solar panel detection and solar panel potential assessment. These services require prior knowledge about the location of buildings. At the moment, the locations of buildings in the Netherlands is extracted from databases from the Netherlands' cadastre, land registry and mapping agency, known as Kadaster: The Basis Administratie Gebouwen (BAG) and Basisregistratie Grootschalige Topografie (BGT) databases are maintained by Kadaster and together contain information about all registered buildings in the Netherlands.

1.2 USE CASES

1.2.1 Use case - Readar

Manual mapping is accurate, but slow. For a commercial party such as Readar, this manual labor comprises a significant part of the costs in the services it offers. It is also preferable to be independent of external data sources for building locations. Currently, the BAG and BGT are the main external data sources for building locations in the Netherlands. When looking abroad, the data quality of the counterpart of the Dutch Kadaster is generally worse and sometimes not openly available.

In the Netherlands, each municipality has the responsibility of keeping the BAG and BGT databases up to date within their administrative border. This decentralized approach results in varying data quality and an inconsistent appliance of rules and processes for updating the databases. Therefore, one service of Readar is change detection or 'bijkaarteren' in Dutch. For this service, Readar will report to their client all manually detected buildings on the aerial imagery that are not listed in one of Kadasters databases and will report all buildings that are intrinsically different to their listing. If Readar could automatically and reliably detect buildings on aerial imagery, Readar:

- Would be less dependent on the data maintained by Kadaster,
- Would have a commercial advantage by reducing costs for the change detection service,
- Could expand operations internationally by generating their own competitive building dataset.

1.2.2 Use case - 510 & Missing Maps

For the Missing Maps initiative which runs on volunteers, the advantages of having an automated building detection model would be twofold:

1. In the case of a (natural) disaster affecting a large, insufficiently mapped area, within hours a quick assessment could be made on the location and number of people affected for the initial disaster aid response.
2. Preselecting the inhabited parts of raw satellite imagery which should be mapped is another task that is currently performed manually but could be potentially automated.

If Missing Maps would have an automated building detection algorithm for satellite imagery, it would be possible to:

- Provide a population density assessment of the affected area in case of a (natural) disaster within hours,
- Automate the currently manual preselection of raw satellite imagery,
- Potentially reduce the workload of volunteers by automatically mapping first and let the volunteers only correct mistakes made by the automated model.

1.3 OBJECTIVE

Even though the use cases for Readar and the Missing Maps initiative are significantly different, their objective is the same. The Missing Maps initiative is mainly interested in building detection in the developing world whereas Readar interests lie in the developed world. Therefore, a robust model is required that is able to detect buildings worldwide. This sets the scope of this internship:

Robust building detection on aerial imagery using Deep Learning

In practical terms, it is preferred that as few adjustments to the building detection model are necessary for the model to perform at a satisfactory level on any location or aerial imagery. In this internship, satellite imagery is included in the umbrella term of aerial imagery. The main component of this internship will be to gain knowledge about the capabilities of Deep Learning and Computer Vision on the subject of the detection of buildings on aerial imagery. In order to provide this knowledge, Computer Vision models will be applied to data provided by the 510 team of the NLRC and Readar. This provides a relevant in-house case study for both organizations to give an indication to what extent Deep Learning can aid them in their activities. The aim is that the knowledge provided by this internship will guide both organisations in developing a well-fitting, realistic strategy for potential further implementation of Computer Vision techniques in related activities.

Both Readar and the 510 team of the NLRC have been working with Machine Learning for other projects, but no automated building detection model performed at satisfactory levels at the time this internship commenced. In order to obtain the required insights about robust building detection using Deep Learning, the practical objective of this internship is stated as follows:

The development of a robust Deep Learning model capable of worldwide building detection on aerial imagery.

Since this is an explorative study, well-written documentation of any Deep Learning model is more important than its performance. The internship can be considered a success when both Readar and the 510 team of the NLRC have enough knowledge to make well-informed decisions on the topic of Deep Learning and its applicability for their respective organizational objectives.

1.4 RESEARCH QUESTIONS

The robustness of a Deep Learning model will be explored with respect to three characteristics of the input data: *Geographical location*, *building geometry* and *image resolution*. This leads to the following three research questions:

1. What is the robustness of a Deep Learning model with respect to the geographical location of the input data?
2. What is the robustness of a Deep Learning model with respect to the resolution of aerial imagery?
3. What is the robustness of a Deep Learning model with respect to building geometry?

The generic term "a Deep Learning model" is used on purpose at this point because a wide range of Deep Learning models available. In order to limit the scope of this report, a model selection will be conducted. Section 1.5 contains an exploration on suitable computer vision task types and the corresponding model types. The current state of the art architecture for each task type is investigated in the second chapter. Promising architectures are selected for implementation at the end of chapter two, allowing us to refine the research questions in Section 2.5.

1.4.1 Definitions

A few key phrases that are used widely throughout the report are clarified here.

Building: The smallest structural unit that is roofed and functions independently. Under this definition a row of connected residences or terraced homes consists of multiple buildings instead of one large building.

Robustness: A term measuring the impact on the performance of the model given a change in one aspect of the input data. The larger the change in a certain aspect and the smaller the impact on performance, the more robust a model is considered to be regarding that aspect.

Geographical location: The location of the area covered by the dataset. Location is an embodiment of important visual features such as presence of potentially obstructing vegetation, presence of objects visually similar to buildings, prevalent architectural styles used and overall image scenery in the given location.

Geographical robustness: The geographical robustness of a Deep Learning model is determined by the difference in its performance on datasets with a different geographical location. A test dataset geographically similar to the training dataset is used to obtain a baseline and this is compared with performance on test datasets geographically different to the training dataset.

Image resolution: The ground surface distance covered per side of a pixel in the image.

Image resolution robustness: The image resolution robustness of a Deep Learning model is determined by the difference in its performance on a dataset at various resolutions. A test dataset using the same image resolution as the training dataset is used to obtain a baseline and this is compared with performance on the same test dataset with an artificially altered image resolution.

Building geometry: A selection of geometric characteristics of a building. In this internship the shape, size and orientation of a building will be covered by the term "building geometry".

Building geometry robustness: The image resolution robustness of a Deep Learning model is determined by the difference in its performance on buildings with varying sizes, shapes and orientations. The procedures of evaluating the robustness of a model to those three aspects are explained further on in this report.

1.5 PROBLEM ANALYSIS

1.5.1 Information requirements

A thorough understanding is required of the type of information that should be extracted by a building detection model. This information dictates the task to be solved and determines the appropriate performance metric. For example, if the number of buildings in a given image needs to be predicted, the model output could consist of a single integer and the performance metric could simply be the absolute deviation from the correct number of buildings.

Given the aim of this internship, the main information requirement is the location of individual buildings. The method of embedding location information is equally important: Many options are available, e.g. using the coordinate of the centroid of the building, using a bounding box surrounding the building outline or using a random coordinate within the building outline. The first two examples have a distinct preference over the last one. Using building centroids provides consistent location information, in contrast to using random points within the building outlines. Bounding boxes contain information about the expected size and aspect ratio of the building.

The secondary information requirements are the size and outline of individual buildings. Note that a building outline contains the information about building size but that a building size does not infer any usable information about the building outline. Tasks and performance metrics are non-trivial if information about location and size are desired, as is the case in this internship. Therefore, it is critical to have a clear understanding of the various task types used in the field of computer vision. Most models perform one specific type of task, limiting the options during model selection based on the task that need to be solved.

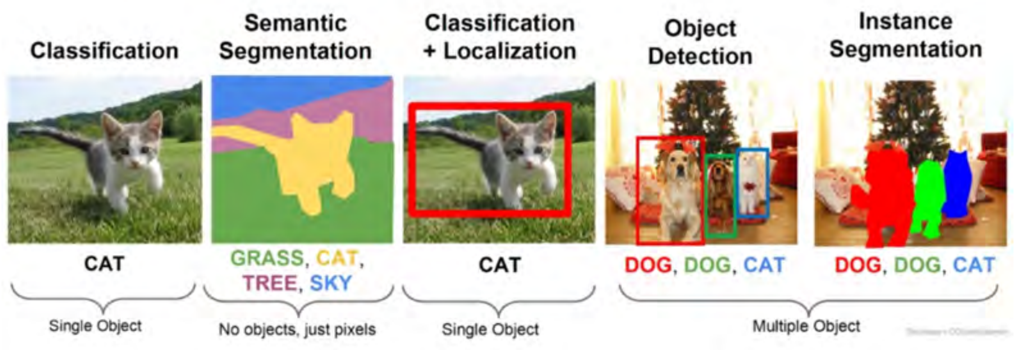


Figure 1.1: Object recognition task types. Source: [30]

1.5.2 Task types

First the distinction needs to be made between single-class and multi-class models. In multi-class models, the model can predict one of several classes of objects, e.g. "cat", "dog", "bird" and "other". Single-class models counter-intuitively require two classes: The class of interest and its opposite counterpart, e.g. "building" and "no building". Single-class models are for that reason also known as binary models. Note that from a technical point of view the multi-class and single-class models are equivalent, with the only difference being that the number of classes is two for single-class models and above two for multi-class models. For the task of building detection, we are looking for a binary model. Figure 1.1 illustrates the main task types for image recognition, which are explained in more detail below.

In *image classification* the entire image belongs to one class. Therefore, a prerequisite for image classification is that each image contains only one class, so e.g. either "cat" or "dog" but not both. There can be multiple objects of the given class in the image, e.g. multiple cats. The model predicts for each class the probability of being shown.

In *semantic segmentation*, also known as *image segmentation*, a prediction is made for every single pixel of the image. Similarly to image classification, the model output consists of the class-wise probabilities for each pixel, resulting in a pixel mask for each class.

In *object localization* and object detection, a bounding box is generated around the outline of the object of the class that is predicted. The differences between the object detection and the localization tasks are minimal and poorly defined. The two terms are sometimes used interchangeably, but the term object localization often focuses on a single object from a single class (find the cat) whereas object detection is generally used to describe multi-class localization (find all cats ánd dogs). Since the scope of this internship is limited to one class ("building") but multiple objects, the object detection terminology will be used from here on.

In *instance segmentation*, the objective is to find the pixels belonging to each unique instance of the detected classes. For every instance, the model predicts a separate pixel mask.

Side-note: One entity of a class can either be referred to as an "object" or an "instance". For disambiguation, the term "instance" is used when the uniqueness and identifiability to other entities of the same class is relevant. The term "object" is used when the identifiability of an entity is not required.

Task type	Multi-class question	Model output
Image classification	Which class is shown?	C probabilities
Image segmentation	To which class belongs this pixel?	C pixel masks
Object localization	Where is the object roughly?	1 bounding box
Object detection	Where are the objects roughly?	N bounding boxes
Instance segmentation	To which instance belongs this pixel (if any)?	N pixel masks

Table 1.1: Generic task overview, with C the number of classes and N the number of detected objects or instances in the image.

Task type	Single-class question	Model output
Image classification	Are there any buildings?	1 probability
Image segmentation	Does this pixel belong to a building?	1 pixel mask
Object localization	Where is the building roughly?	1 bounding box
Object detection	Where are the buildings roughly?	N bounding boxes
Instance segmentation	To which building belongs this pixel (if any)?	N pixel masks

Table 1.2: Specific binary task overview, with N the number of detected buildings in the image.

Table 1.1 provides a generic image recognition task overview and Table 1.2 specifies this overview for building detection. Since the objective is to find the rough locations (bounding boxes) and outlines (pixel masks) of buildings on aerial imagery, the image classification task does not provide the desired kind of information except for pre-selecting images based on signs of inhabitation alluded to earlier on in Section 1.2.2. Instead, this report will focus on object detection, image segmentation and instance segmentation.

1.5.3 Task selection

Each task has its own implications and consequences. For example, object detection and image segmentation are vastly different in nature and therefore require differing approaches. *Object detection* requires locating the entire object at once. To detect the entire object, some global and high-level information about the contents of the image is required, while the more fine-grained information such as the exact object outline is less relevant to score well on the object detection task. In contrast, *image segmentation* predicts on pixel-level. This requires having an increased focus on the precise location of borders between classes, implicating a higher dependency on local information. With image segmentation there is no need for integral object detection, reducing the relative importance of understanding global characteristics of the image. The *instance segmentation* task is the hardest task of the three. It combines the challenges of the object detection and image segmentation tasks.

Given the aim of this internship, where localization is the primary objective, **the object detection, image segmentation and instance segmentation tasks are all considered to be suitable tasks**. The object detection task has the lowest information extraction demand and is likely to be the most achievable task of the three. The instance segmentation task has the highest information extraction demand and is likely to be the most ambitious of the three.

1.6 REPORT STRUCTURE

Chapter two contains a problem analysis in which the problem is clearly defined and a literature review on the application of Deep Learning models on image recognition in general and on building detection from aerial imagery specifically. Chapter three contains a technical overview of Convolutional Neural Networks and the ResNet, Mask R-CNN and U-net architectures. Chapter four provides more details on the available imagery and data sources. The Mask R-CNN architecture is implemented and tested on robustness in Chapter five. Chapter six contains an implementation of the U-net architecture and includes robustness experiments. The main findings are summarized in the concluding chapter seven. Chapter eight translates the main findings into practical implications and recommendations to Readar and the 510 team of the NLRC.

2 LITERATURE REVIEW

In this chapter, a literature review is held on the Computer Vision area in general. This is followed by an overview of the state of the art architectures for the object detection, image segmentation and instance segmentation tasks. After a literature review is conducted on the subject of building detection on aerial imagery using Computer Vision, the chapter is concluded by selecting the most promising architectures for implementation.

2.1 COMPUTER VISION IN GENERAL

Figure 2.1 shows the progress being made by Computer Vision models on the yearly ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [54]. In the ImageNet challenge, models have to predict for each image in the test set of 50,000 images which object class from the 1,000 possible object classes is shown. As this has been one of the most popular and prestigious challenges in terms of image recognition, an overview of the development of the Computer Vision field in is obtained by looking at the winners of the ImageNet classification challenge.

2.2 CONVOLUTIONAL NEURAL NETWORKS

All of the winners of the ImageNet ILSVRC challenge in the most recent years have used a Convolutional Neural Network (CNN). A CNN is a Neural Network which uses convolutional layers based on the mathematical convolution arithmetic operation instead of traditional fully-connected layers in regular Neural Networks.

The first paper with the idea of convolutional layers was published in 1980 by Fukushima [10]. The first proper implementation of a CNN by LeCun et al. in 1989 [37] is widely regarded as the predecessor of contemporary CNNs. Interest in the CNN architecture decreased over time as it had a high computational cost. However, in 2010 Ciresan [4] showed the advantages of running a CNN on a Graphics Processing Unit (GPU) instead of the standard Central Processing Unit (CPU) which made the model much more potent and interest in CNNs was renewed. In 2012, AlexNet [36] won the ImageNet challenge, beating all competitors and previous results by a wide margin as shown in Figure 2.1. AlexNet was trained using multiple GPUs and had a lot more layers than previous CNN models. Because of the large number of layers, AlexNet can be considered the first Deep Learning CNN network. CNNs have been the dominant model type in all kinds of Computer Vision challenges since 2012. Winners of the 2013 and 2014 edition, ZFNet [66] and GoogLeNet [57] respectively, made several efficiency improvements on the AlexNet architecture and added more layers and used more computational resources.

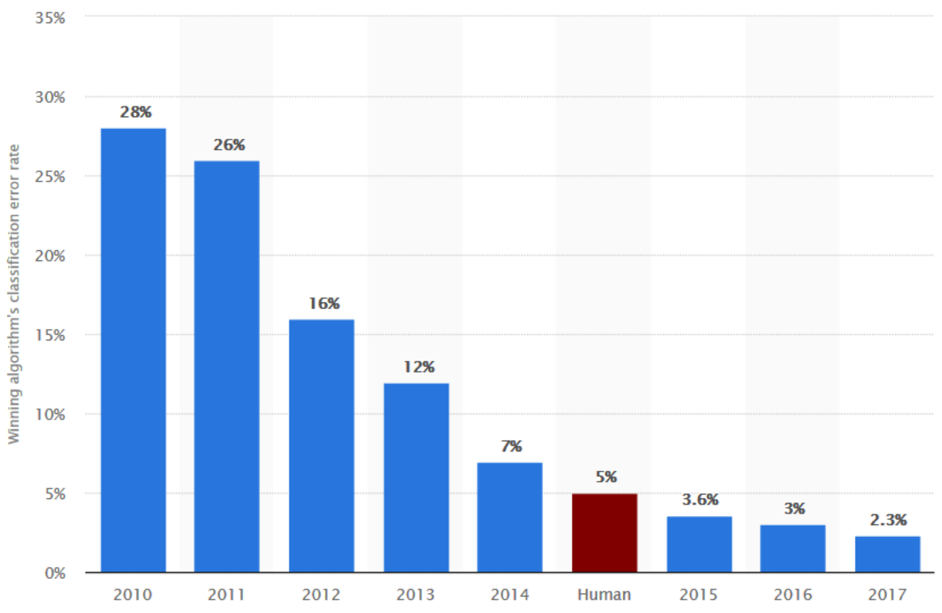


Figure 2.1: Error rate of the winning submission of the yearly ImageNet Large Scale Visual Recognition challenge. Source: [34]

The winner of the 2015 edition, Kaiming He, developed a novel model architecture and named it a Residual Neural Network or simply ResNet [21]. The ResNet model was the first model to defeat the human performance benchmark. Winners of the 2016 and 2017 challenges, the ResNeXt [60] and SE-ResNeXt [23] models adapted the successful ResNet architecture for even higher performance at the cost of increased model complexity and therefore higher computational costs.

Although the performance of the original ResNet architecture has been surpassed many times since 2015, these results have been predominantly achieved by adaptations of said ResNet. Therefore, the ResNet architecture can still be considered the state-of-the-art base architecture at the moment of writing this report. A technical overview of the ResNet architecture is given in Section 3.9.1. The last ImageNet ILSVRC was held in 2017. The reason for discontinuation, stated the challenge organizers, is that nearly all participants in 2017 outperformed the human benchmark. [1]

2.3 APPLICATIONS

Computer Vision models are being implemented for a wide range of applications. The combination of well performing Computer Vision models and the availability of high-resolution satellite imagery have allowed for new remote sensing applications. There is a broad range of applications in terms of size and scope. Examples include large-scale monitoring of deforestation [51], poverty mapping [59] and natural disaster damage assessment [35] to tasks requiring a higher level of detail such as building detection [68, 14, 3, 67, 50, 27, 63], road mapping [49] and solar panel detection [65].

1 https://www.newscientist.com/article/2127131-new-computer-vision-challenge-wants-to-teach-robots-to-see-in-3d/

2.4 COMPUTER VISION FOR BUILDING DETECTION ON AERIAL IMAGERY

Building detection on aerial imagery is a extensively researched topic within the Computer Vision area. With numerous Machine Learning challenges dedicated to this topic, interest has been relatively high. There have been three yearly editions of the SpaceNet challenge [56] and a fourth has been announced in October 2018. INRIA has been hosting a continuous challenge on building detection since 2017 [46]. Online Machine Learning platform Kaggle has hosted many image recognition challenges, among which the Dstl Satellite Imagery Feature Detection challenge in spring 2017. In this challenge satellite imagery had to be segmented into ten possible classes, one of which was buildings [8]. Facebook has hosted the DeepGlobe challenge, on the subjects of road extraction, building detection and land classification on satellite imagery in spring 2018 [9]. Online Machine Learning platform CrowdAI has hosted a challenge over the summer of 2018 on building detection, the Mapping Challenge [25].

These challenges, often offering a financial reward for the top performing participants, attracted many participants. Analyzing the models or model descriptions published post-challenge by the top performing participants provides insight into the state of the art building detection models. For the building detection challenge of the DeepGlobe contest, the three highest scoring participants [68, 14, 27] used three different model architectures. Ohleyer [50] applied several different models on the data of the INRIA challenge for a direct comparison. Besides the public data challenges, there are other sources which contribute to the exploration of the topic of building detection on satellite imagery. Interest from both commercial and scientific institutions leads to an increasing volume of publications on this topic [3, 67, 63, 64].

Looking at the types of models that are used for these building detection challenges and related image recognition challenges, a trend can be distilled in terms of network architecture. Over the past few years, most models used by top classifying participants fall into the category of Fully Convolutional Network (FCN) [44], which excel in the image segmentation task. The most popular architecture within the category of Fully Convolutional Networks is the U-net [53]. The U-net was introduced in 2015 but still is used by winning solutions for recent image recognition challenges [26, 17, 55]. Adaptations from the U-net architecture have been created, often integrating the residual skip connections introduced by the ResNet model into the U-net. One example is the Linknet model [2] which yields comparable results with the U-net model but requires less computational resources [14]. Since the Linknet model is an adaptation from the U-net architecture, the U-net model can still be considered the state of the art model for image segmentation.

The Mask Region-based Convolutional Neural Network (Mask R-CNN) model [19] was introduced in December 2016 and performs the object detection and instance segmentation task simultaneously. It does not fit into the Fully Convolutional Network model category, but is part of the family of (two-stage) region-based convolutional networks. The Mask R-CNN model has proven to be competitive with the U-net model and other Fully Convolutional Neural networks in challenges held in 2017 and 2018 [55, 61]. In terms of performance on the object detection task, the Mask R-CNN model competes with a family of models called Single Shot Detectors (SSD) including models such as RetinaNet [40] and the third generation of the YOLO single shot detector [52]. For the instance segmentation task however, the Mask R-CNN model currently is the undisputed state of the art model.

2.5 MODEL SELECTION AND RESEARCH QUESTION REFINEMENT

The U-net and Mask R-CNN architectures have been identified as top performing models for the three relevant image recognition task types for this internship. Both models are implemented and compared to each other in Chapter 7. The technical aspects of these two models are looked at in the next chapter. The selection of the two architectures allows us to transform the three broadly defined research questions from Section 1.4 into two sets of specific research questions:

1. What is the robustness of a Mask R-CNN model with respect to the geographical location of the input data?
2. What is the robustness of a Mask R-CNN model with respect to the resolution of aerial imagery?
3. What is the robustness of a Mask R-CNN model with respect to building geometry?

4. What is the robustness of a U-net model with respect to the geographical location of the input data?
5. What is the robustness of a U-net model with respect to the resolution of aerial imagery?
6. What is the robustness of a U-net model with respect to building geometry?

The experiments to provide answers to these research questions are described in Chapters 5 and 6.

3 MACHINE LEARNING

3.1 CHAPTER OVERVIEW

Section 3.2 of this chapter provides an overview of the main technical aspects of Artificial Neural Networks. The sub-class of Artificial Neural Networks used for image recognition tasks, known as Convolutional Neural Networks (CNNs) is covered in Section 3.3. From Section 3.4, the relevant aspects of a Neural Network from a functional perspective are covered: The process of fitting a Neural Network to a dataset in practice is explained in Sections 3.4 to 3.8. The final section elaborates on the network architectures of three CNN models that were identified as state-of-the-art in the literature review of chapter two. The design rationale behind the architectures of the ResNet model, the U-net model and the Mask R-CNN model is explained in Section 3.9.

3.2 ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks are often simply called Neural Networks for brevity. The first topics in this section look at Neural Networks from a structural point of view, explaining the fundamental concepts.

3.2.1 Neural Network Structure

The hierarchical structure of a neural network is relatively comprehensible: A neural network consists of multiple layers and each layer consists of one or more artificial neurons, as shown in Figure 3.1. The artificial neuron, or simply neuron, is the elementary unit of a neural network.

If each layer in a neural network receives its input from preceding layers only, and thus the network does not contain any feedback loops, the network can be categorized as a feedforward neural network. In a traditional neural network architecture, each layer receives its input from the directly preceding layer and sends the output to the directly succeeding layer as is the case in Figure 3.1. This classic feedforward neural network architecture is sometimes referred to as a Multilayer perceptron (MLP) for historic reasons, but this term is technically incorrect. The original MLP network consisted of neurons known as perceptrons which apply a linear, binary activation function, hence its name. Contemporary neural networks often use non-linear, continuous activation functions instead [16].

The number of neurons of each layer and the total number of layers in the network are important hyperparameters in the design of a neural network. Section 3.2.3 shows that the number of neurons in the input layer is determined by the input data and the number of neurons in the output layer is determined by the prediction type.

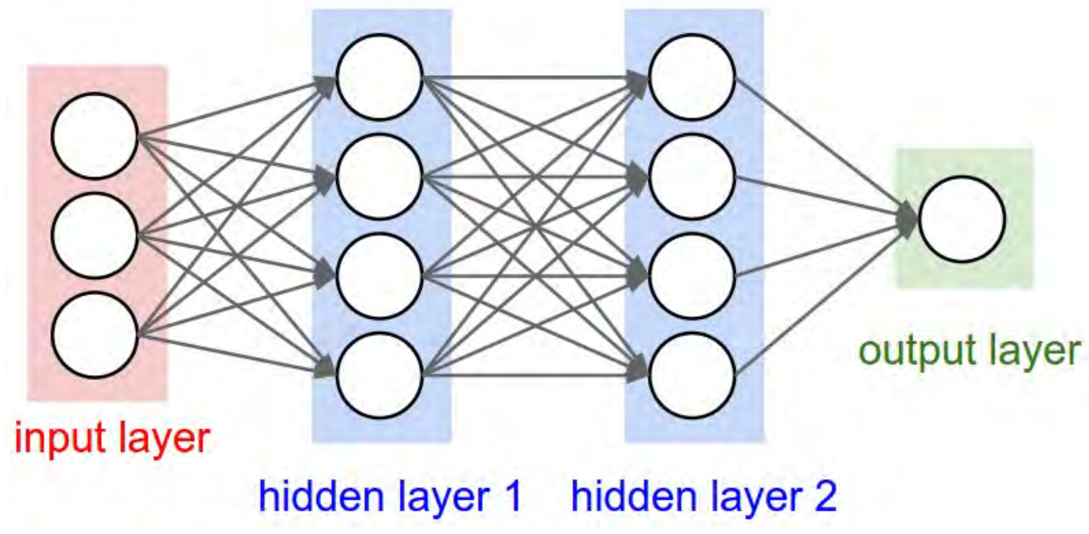


Figure 3.1: A schematic overview of a Neural Network, where circles depict artificial neurons, columns depict layers and arrows represent connections. Source: [7]

The number of hidden layers and the number of neurons in those layers are not restricted and these hyperparameters have to be carefully set. The combination of these hyperparameters determines the number of weights in the network which is a strong indicator for the complexity of the model.

3.2.2 Artificial Neurons

The artificial neuron was first introduced in 1943 by Warren McCulloch and Walter Pitts [47]. A schematic overview is illustrated in Figure 3.2. The neuron transforms the input by taking a weighted sum over the input signal, followed by applying a non-linear activation function. From a mathematical perspective: For any neuron, given m variable inputs with values $x_1, ..., x_m$ there are m corresponding weights $w_1, ..., w_m$ indicating the influence of that input on the output. Each input value is multiplied with its corresponding weight and to this result an additional weight b is added which is known as the bias. The resulting value v is used as input to the activation function φ, yielding:

$$y = \varphi(\sum_{j=1}^{m} w_j x_j + b)$$

or equivalently, in vector notation where x is the column vector of input values and w the row vector of weights:

$$y = \varphi(w \cdot x + b)$$

3.2.3 Layers

The number of layers in a network is known as the depth of the network. A network normally consists of an input layer, a number of hidden layers and an output layer. When each neuron in a layer receives input from all neurons in the preceding layer, the layer is said to be fully connected.

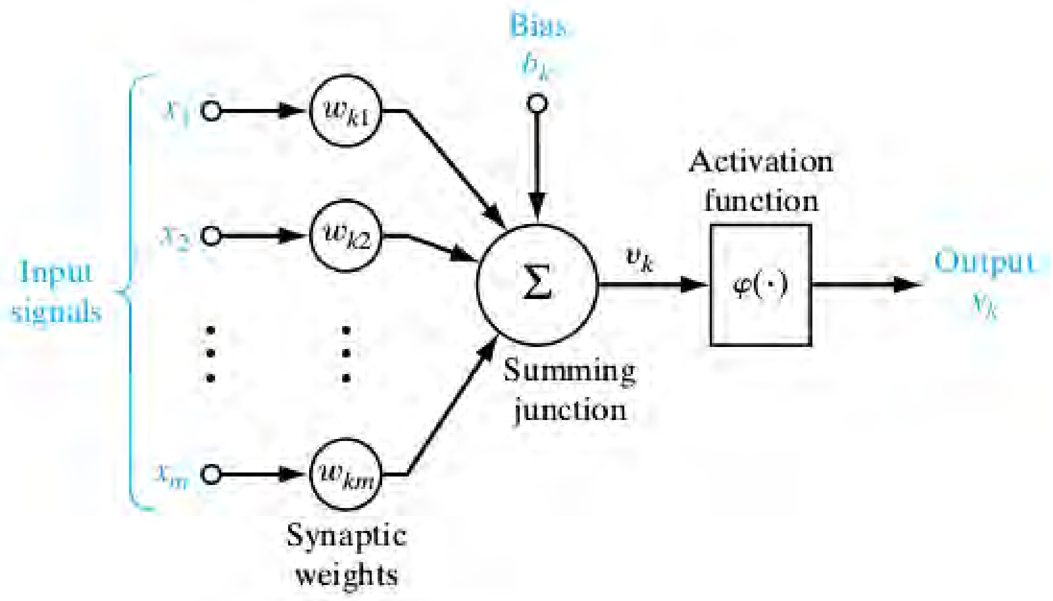


Figure 3.2: Schematic overview of a neuron. Source: [7]

Input layer
The first layer of a network is known as the input layer, all intermediate layers are called hidden layers and the final layer is called the output layer. The neurons in the input layer deviate from the default behaviour as they directly send the data they receive unaltered towards the next layer. For each neuron in the input layer, we simply note $y = x$.

Hidden layers
The intermediate layers are called hidden layers because their output is immediately propagated to the next layer and not directly accessible or visible to the user of the network. In most situations, the bulk of the connections in a network are found in the hidden layers. The hidden layers have the objective to find meaningful patterns and interactions between the input values required for making accurate predictions. Assume a fully connected hidden layer l contains n neurons, we know that for the k^{th} neuron in vector notation:

$$y_k = \varphi(w_k \cdot x + b_k)$$

For layer l, combining weight vectors $w_1, ...w_n$ yields weight matrix $w^{(l)}$ of size k by m where element w_{km} is the weight belonging to the m^{th} input of neuron k. Similarly, writing the bias scalars $b_1, ..., b_n$ as column vector $b^{(l)}$ and writing the output scalars $y_1, ..., y_n$ as column vector $y^{(l)}$, allows all computations in the entire layer to be expressed as:

$$y^{(l)} = \varphi(w^{(l)}x + b^{(l)})$$

Remember that the output of layer $l - 1$ is the input of layer l. Assuming that each hidden layer uses the same activation function φ, we can rewrite the previous equation to:

$$y^{(l)} = \varphi(w^{(l)}y^{(l-1)} + b^{(l)})$$

Note that for brevity, the bias of a neuron can be included in its weight vector by introducing a superficial constant input $x_0 = 1$ and incorporating the bias as a new weight vector w_0. In the weight matrix $w_{00} = b$ and all other elements of the newly introduced row w_{0j} and column w_{i0} are set to zero. This shortens the notation of neuron k to $y_k = \varphi(w_k \cdot x)$, and the notation of layer l to:

$$y^{(l)} = \varphi(w^{(l)} y^{(l-1)})$$

which will be used from here on.

Output layer
The predictions of a neural network are generated in the output layer. As a consequence, the desired number of predictions determines the number of neurons in this layer. For example, in the case of the image classification task with c classes, there must be exactly c neurons in the output layer as each neuron will represent a class. The desired format of the predictions influences the activation function in the output layer.

3.2.4 Activation functions

A neural network that exclusively applies linear activation functions is only capable of learning linear combinations of the input values. Consider a Neural Network that uses the identity function $(x) = x$ as activation function, which is equivalent with not using any activation function. Each layer performs $y^{(l)} = \varphi(w^{(l)} y^{(l-1)}) = w^{(l)} y^{(l-1)}$ and a neural network of n layers can be described by:

$$y = w^{(n)} w^{(n-1)}, ..., w^{(1)} x$$

or

$$y = (\prod_{l=1}^{n} w^{(l)}) x$$

If we combine this with linear algebra theory that any valid multiplication of two matrices yields a matrix, all weight matrices could be multiplied with each other in the correct order and the result would be a single weight matrix. The entire network could thus be expressed a $y = wx$ where w is a weight matrix of size $k\,by\,m$. This is equivalent to a neural network without hidden layers, capable of performing nothing more than a single linear transformation on the input. Obviously, this model can only create a simplistic representation of the interactions between the input and therefore its predictive power will be poor for non-linear prediction tasks.

If we extend the thought experiment to a model where the activation function of each layer is any linear function, the model can be expressed as a sequence of matrix multiplications and activation functions. A network with three layers excluding the input layer could be expressed as:

$$y = \varphi(w^{(3)} \varphi(w^{(2)} \varphi(w^{(1)} x)))$$

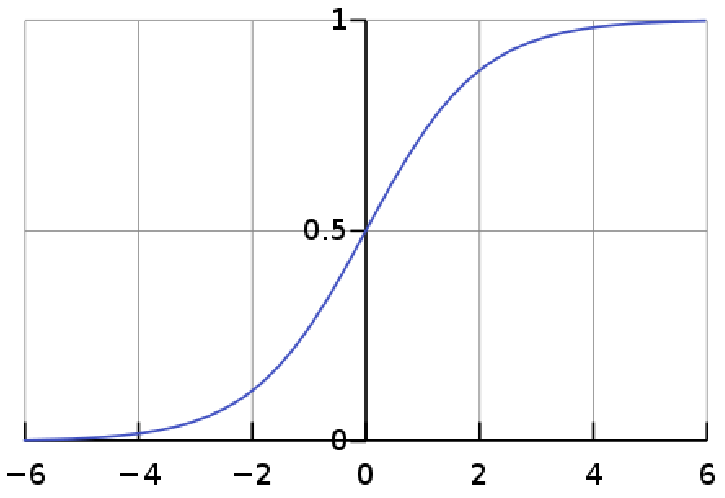


Figure 3.3: The sigmoid function with x on the x-axis and f(x) on the y-axis. Source: [58]

This is a sequence of linear transformations. Any linear transformation can be written as a matrix multiplication, as is proven in Chapter 2.2 of Goodfellow et al. [15]. Rewriting the activation functions as matrix multiplications will again yield a sequence of matrix multiplications and analogous to the previous experiment we end up with a neural network that can be simplified to a single linear transformation.

To allow the network to represent non-linear interactions between the input values, the model needs to be able to perform non-linear transformations. The activation function is included as an essential ingredient to the network architecture to provide this non-linearity. Many different activation functions exist. The activation of the output layer depends on the task to be performed. In the case of a binary classification task, the output layer of the neural net commonly uses the logistic sigmoid activation function. The logistic sigmoid function, or simply sigmoid function, is defined as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

One of the appealing properties of the sigmoid is that it squeezes any real, continuous value to the $(0, 1)$ domain as shown in Figure 3.3. By scaling the output between 0 and 1, the sigmoid activation function allows us to interpret the output of the layer as probabilities.

Historically, the sigmoid function was also used as activation function for the hidden layers. The sigmoid activation lost popularity as activation function for the hidden layers once networks became deeper. Undesirable properties of the sigmoid activation function cause difficulties with training a deep network, which is known as the vanishing gradient problem [13]. The hyperbolic tangent function $f(x) = \tanh(x)$ also is a popular activation function but holds the same undesirable properties for deep networks [13]. Nowadays, the most commonly used activation function is the Rectified Linear Unit (ReLU) [16]. The ReLU function is defined as:

$$f(x) = max(x)$$

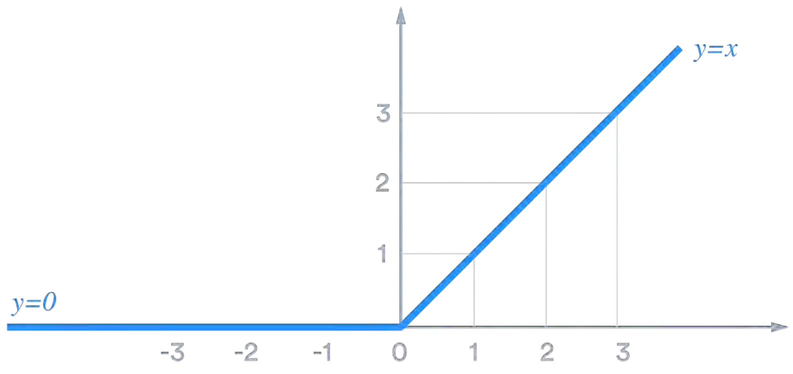


Figure 3.4: The ReLU activation function. Source: [42]

It is popular for its simplicity as it just sets all negative values to zero as shown in Figure 3.4. This allows for effectively training a deep neural network with ReLU activations as it does not suffer as badly from the vanishing gradient problem as networks using the sigmoid and hyperbolic tangent [13]. Other increasingly popular activation functions include the PReLU (Parametric ReLU) [20], ELU (Exponential Linear Unit) [5] and the LeakyReLU [45].

3.2.5 Model evaluation

The combination of the network architecture and its weights forms a model. In order to measure the quality of the predictions of a model, evaluation functions are required. Evaluation comes from two perspectives: The first viewpoint is based on the available data, model and objective function; the second viewpoint is based on the intended use case. The loss function is a measurement for the first perspective, and performance metrics are measurements for the second viewpoint.

The loss function is designed to describe the difference between the predictions and desired output; the lower the loss value, the better the model performs. A general form of the loss function L is $L(y, \hat{y}) = |y - \hat{y}|$, where y is the correct output belonging to a sample known as the ground truth and y is the prediction of the model. During training, the model is optimized by updating the weights of the model to minimize the loss function. It is important to note that the loss function does not only evaluate the quality of the predictions, but also dictates how the model should be changed to improve during the training process. A proper loss function should meet several criteria: It should properly describe the residual between the prediction and desired output, must be differentiable and should be evaluated in a computationally efficient way. These requirements are a direct result of the way a neural network is trained, which is described in Section 3.4.2. Because of these requirements, the loss function is often a proxy of the desired performance objective. However, this is no problem as the desired performance objective can be described by performance metrics.

In Machine Learning, a performance metric is a more direct measurement of the predictive qualities of the model with regards to its intended use case. A performance metric is only used to quantify the quality of the predictions, without influencing the training process in any way. As an example, for this internship the percentage of correctly detected buildings is an insightful metric. Image segmentation techniques provide pixel-wise instead of building-wise output. It is possible to aggregate pixel-wise output into building-wise statistics but this requires discrete, non-continuous functions which are non-differentiable.

This indicates that the loss function should likely be a pixel-wise evaluation function. Building-wise statistics such as the number of correctly detected buildings cannot be expressed in a continuous, differentiable function and are therefore ineligible to be used as a loss function. However, these statistics can be evaluated as a performance metric.

3.3 CONVOLUTIONAL NEURAL NETWORKS

Traditional Neural networks such as the MLP are extremely inefficient in image recognition tasks as there is no predefined spatial context within or between fully connected layers. It is theoretically possible for MLPs to perform this image recognition tasks, but due to aforementioned efficiency issues they are not used in practice. The convolutional layer was designed specifically to tackle image recognition tasks, utilizing the spatial relation between a pixel and its surrounding neighbours. In contrast to a fully connected layer, a convolutional layer has an inductive bias that the ordering of the inputs in a spatial layout contains meaningful relations. A network where the hidden layers consist of only convolutional and pooling layers (explained in Section 3.3.2) is known as a convolutional neural network or CNN. If the output layer is also a convolutional or pooling layer, the network is known as a fully convolutional network, or FCN.

3.3.1 Convolutional Layer

Assume that the dimensions of the input for a convolutional layers are $H_{in}\,x\,W_{in}\,x\,C_{in}$, where H_{in}, W_{in} and C_{in} denote the height, width and number of channels of the input. Images generally consist of three channels: Each channel contains the intensity values for either red, green or blue, also known as the Red Green Blue (RGB) colour values.

A convolutional layer consists of a number of filters. A filter, sometimes also referred to as a kernel, is a set of spatially ordered weights. Each filter contains H_{filter} x W_{filter} x C_{in} weights representing height, width and channel depth respectively. Any odd filter size is applicable in CNNs for the height and width dimensions but the channel dimension always equals the number of input channels. Odd filters sizes are used for the height and width because they are symmetric around the center pixel, known as source pixel. Even filter sizes also yield valid results, but it would lead to asymmetrical results. The $3\,x\,3\,x\,C_{in}$ configuration for the filter dimensions is commonly used and is abbreviated to a $3\,x\,3$ conv. A filter of size $3\,x\,3$ has a receptive field size of 3 on the input for that layer. Each filter slides over the input as a sliding window. At every position, the convolution is calculated between the receptive field and the filter. The output of a convolution is called a feature, and all features from a filter combined are called a feature map as shown in Figure 3.5. Given that each filter applied to the input of the layer yields one feature map, the output dimensions for the entire convolutional layer are $H_{out}\,x\,W_{out}\,x\,C_{out}$ where C_{out} equals the number of filters in the layer. The values of H_{out} and W_{out} depend on the way the filter slides over the input. This is influenced by two parameters: the stride S and padding P of the convolutional layer.

The stride S determines the “step-size” of the sliding window of the filter. The default value is $S = 1$, where the filter is shifted by one pixel every time it is moved to its new position. If $S = 2$, the window is moved by two pixels in the same direction, reducing the size of the feature map compared to $S = 1$.

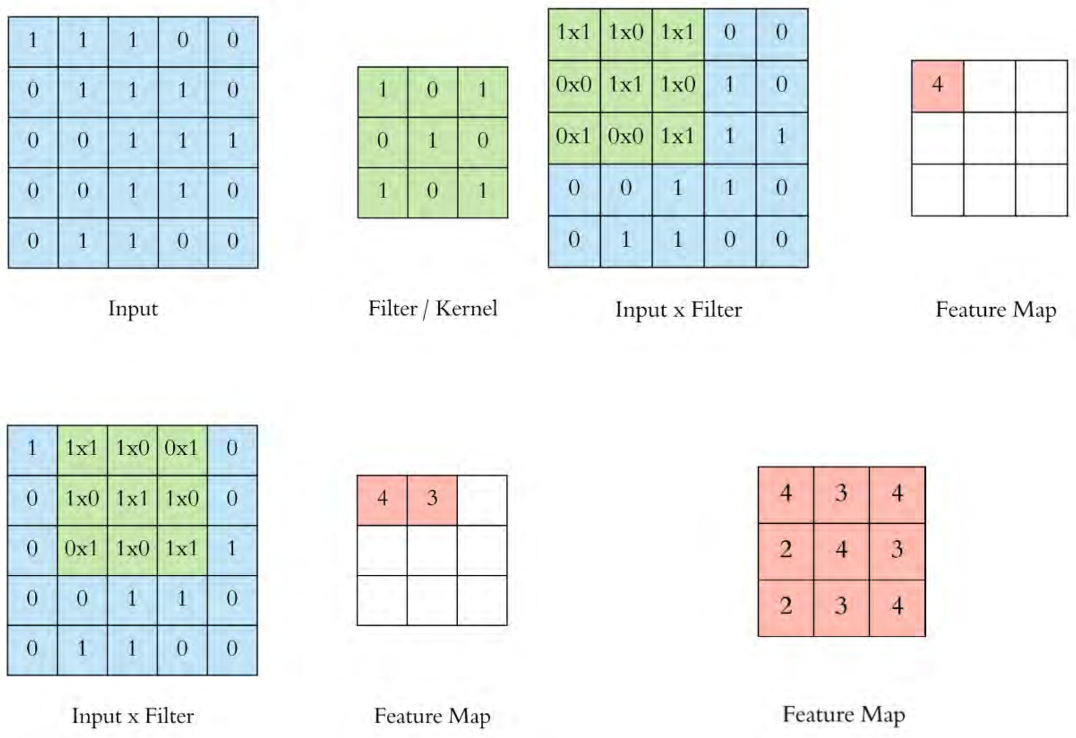


Figure 3.5: Top left: Example input of $5\,x\,5\,x\,1$, and a filter of $3\,x\,3\,x\,1$. Top right: The filter applied to the first position and the convolved feature. Bottom left: The filter applied to the second position and the first two convolved features. Bottom right: The complete resulting feature map. Source: [6]

The example in Figure 3.5 uses $S = 1$, yielding a $3\,x\,3$ feature map. If it used $S = 2$ the resulting feature map would be $2\,x\,2$ in size. Without padding, thus $P = 0$, for any $3\,x\,3$ or larger convolutional layer $H_{in} > H_{out}$ and $W_{in} > W_{out}$ respectively. Sometimes, it is desirable to retain the image dimensions, or $H_{in} = H_{out}$ and $W_{in} = W_{out}$. This can be achieved by artificially adding pixels around the borders of the input, which is known as padding as illustrated in Figure 3.6. The most common form of padding is zero-padding where the added pixels have a value of zero for all channels but other options are to repeat the values of the pixel at the border or to mirror pixels near the border. Combining all this, the feature map size of a convolutional layer can be calculated by the following formulae:

$$H_{out} = \frac{H_{in} - H_{filter} + 2P}{S} + 1$$

$$W_{out} = \frac{W_{in} - W_{filter} + 2P}{S} + 1$$

By reusing the same filter (set of weights) on multiple positions of the input, the number of learnable weights in the network stays relatively low, a concept known as parameter sharing. One convolutional layer generally consists of multiple filters allowing it to recognize multiple different patterns. For example, the illustration of Figure 3.7 contains ten filters.

A filter does not use any information from the input image outside the current input window, or in other words, a filter is location agnostic. Reusing a location agnostic filter in a sliding window approach has the beneficial consequence that the filter provides the exact same output to any specific input window regardless of the window's location in the input image.

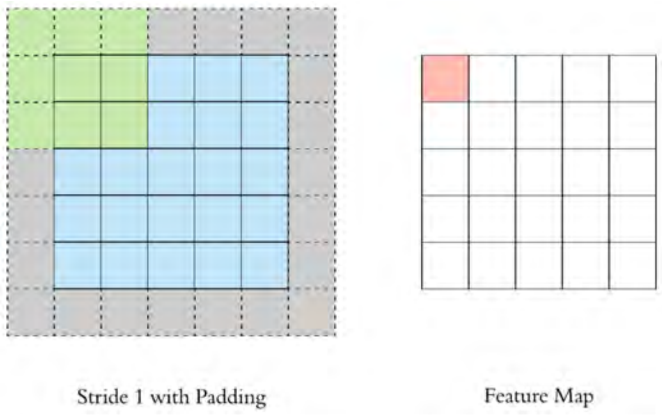


Figure 3.6: A $3\,x\,3$ filter with $S = 1$, $P = 1$. Source: [6]

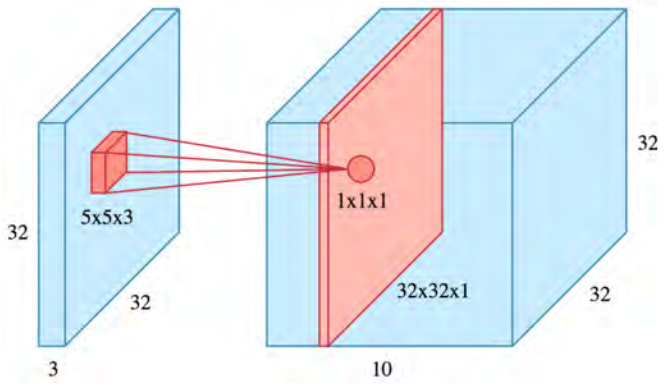


Figure 3.7: An example highlighting one of ten $5\,x\,5$ filters of a convolutional layer applied to a $32\,x\,32\,x\,3$ input with same-padding. Source: [6]

This property combined with the spatial information contained in the resulting feature map makes a convolutional layer spatially equivariant: A function is spatially equivariant when a spatial translation of the input yields an equivalent translation in the output. For example, if a building is shifted a bit to the right in an input image, the prediction for that building would be unaltered itself but its location proportionally shifted to the right in the output map.

3.3.2 Pooling Layer

The feature representations of the input data tend to become more complex as the signal progresses through the layers allowing for a more diverse set of features. To effectively utilize this effect, the number of features per layer can be progressively increased in the deeper layers. This comes at an increased computational cost, which can be compensated for by progressively reducing the number of pixels per layer. The functionality of a pooling layer is to reduce the spatial dimensionality of its input, also known as downsampling. The main motivation for reducing the dimensionality is to reduce memory footprint and computational costs for any subsequent layers.

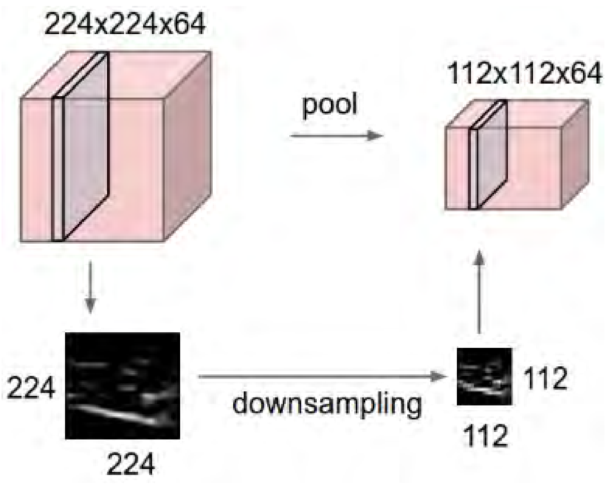


Figure 3.8: Pooling reduces the dimensionality of the input volume. In this case, by a factor of 4. Source: [31]

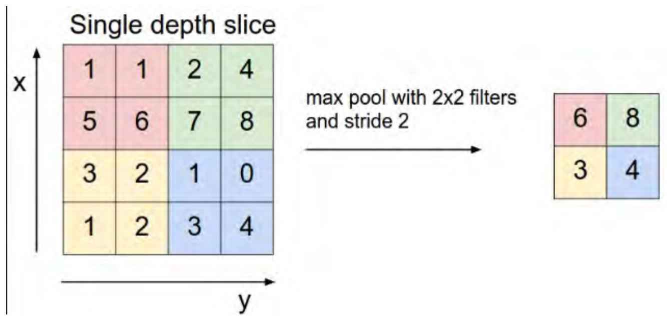


Figure 3.9: An example of max pooling. Source: [31]

There are several different types of pooling layer, of which max pooling and average pooling are the most common. Similarly to convolutional layers, they consist of a filter which slides over the input and use the stride S and padding P parameters. The max pooling layer returns the maximum of all values in the input window. The average pooling layer returns the average of all values in the input window. Pooling layers do not have a source pixel and therefore there is no benefit of using odd filter sizes over even filter sizes. The most common form of pooling is a $2x2x1$ filter applied with stride $S = 2$ and padding $P = 0$, reducing both the width and height by a factor of two whilst maintaining the same number of features, as shown in Figure 3.8.

By introducing max pooling layers to the network architecture, a bit of translation invariance is being introduced to the model. Translation invariance is when any translation of the input of a function is not translated to its output. For example, when a building is moved a bit to the right in an input image, an object detection model that is translation equivariant would shift the prediction for the building proportionally to the right. In the same situation, a translation invariant model merely predicting the presence of buildings would provide the same, unaltered output.

3.4 MODEL TRAINING

There are multiple methods to train a model. The most widely used method, backpropagation in the form of stochastic gradient descent, will be explained in the sections below.

3.4.1 Initialization

The weights of an untrained neural network are generally initialized randomly as no information about the data is known to the model yet at this point. However, the network structure is known at this point. So although the initialization of the weights is performed randomly, there are ways to intelligently initialize the weights based on the architecture of the model. In order to tackle the vanishing gradient problem, an initialization distribution was proposed by Xavier Glorot and Yoshua Bengio [12], known as Xavier or Glorot initialization:

$$w^{(j)} \sim Uniform[-\frac{\sqrt{6}}{\sqrt{n_{j-1}+n_j}}, \frac{\sqrt{6}}{\sqrt{n_{j-1}+n_j}}]$$

where n_j represents the number of neurons in layer j. For deep networks using the ReLU activation function in the hidden layers, He [20] showed that the following initialization scheme is more effective:

$$w^{(j)} \sim Gaussian(mean = 0, st.dev. = \sqrt{\frac{2}{n_{j-1}}})$$

3.4.2 Gradient Descent

There are multiple ways to train a model, from which the gradient descent is currently the most common optimization method for neural networks and will therefore be used in this report. Most of this section and the notation used is based on Goodfellow et al. [15]. In gradient descent, a model is trained by repeatedly performing two steps iteratively: forward propagation and backward propagation.

In forward propagation, input samples, denoted as x, are fed to the neural network and their signal is propagated through each layer of the network. Denote the values of all trainable parameters of the network as θ. Denote the model itself as a function $f(x, \theta)$. The outcome of $f(x, \theta)$, say $\hat{y}$, is compared with the desired output y to calculate the loss provided by the loss function L following the general formula:

$$L(y, \hat{y}) = |y - \hat{y}|$$

substituting the model function $f(x, \theta)$ results in:

$$L(y, f(x, \theta)) = |y - f(x, \theta)|$$

Note that x and y are fixed values describing the data so optimization of L is performed by changing in the backward propagation step. The objective of gradient descent is to find such that the average expected value of L is minimized over all samples in the dataset. Ideally, the objective would be to minimize the function L over the true underlying data distribution of the data. Since this data distribution often is unknown, the empirical distribution described by all data samples is used to optimize on instead.

Denote the expected loss value of L for samples coming from the empirical distribution of the dataset as $J(\theta)$. The aim of the gradient descent algorithm is to find the lowest possible value for $J(\theta)$. In an ideal scenario gradient descent would find the global minimum of $J(\theta)$ of zero, but the algorithm cannot provide any guarantees on convergence to the global minimum. This is because most loss functions L are non-convex; i.e. the second derivative of L is not equal or greater than zero in the entire domain. Non-convex functions can contain multiple local minima and saddle points, which are points where the first derivative of L is zero.

The idea behind backward propagation, or backpropagation, is to determine the gradient of L with respect to θ. This gradient describes an approximation of the change in L given a change in the weights. Once this gradient is known, the weights can be updated in such a way that L decreases. The gradient of L with respect to θ is denoted as:

$$\nabla_\theta L = \frac{\delta L}{\delta \theta}$$

Calculating this partial derivative of the loss function with respect to every single weight in the neural network can be very computationally demanding. In practice, it is performed in an efficient way by applying the chain rule on the Jacobian matrix of each layer of the network. The chain rule states, that for $z = f(y)$ and $y = g(x)$:

$$\frac{\delta z}{\delta x} = \frac{\delta z}{\delta y} \cdot \frac{\delta y}{\delta x}$$

where $\frac{\delta y}{\delta x}$ is the Jacobian of $g(x)$ and $\frac{\delta z}{\delta y}$ is the Jacobian of $y(x)$. The same principle holds for calculating the gradients of a neural network. When the gradient of layer j is known, the gradient for layer $j-1$ can be calculated using the chain rule. The gradient of the weights $w^{(j)}$ for hidden layer j can then be calculated recursively as follows:

$$\frac{\delta L}{\delta w^{(j)}} = \frac{\delta L}{\delta y^{(j)}} \cdot \frac{\delta y^{(j)}}{\delta x^{(j)}} \cdot \frac{\delta x^{(j)}}{\delta w^{(j)}}$$

This can be expanded by expressing $\frac{\delta L}{\delta y^{(j)}}$ as a function of $\frac{\delta L}{\delta y^{(j+1)}}$:

$$\frac{\delta L}{\delta w^{(j)}} = (\frac{\delta L}{\delta y^{(j+1)}} \cdot \frac{\delta y^{(j+1)}}{\delta x^{(j+1)}} \cdot \frac{\delta x^{(j+1)}}{\delta y^{(j)}}) \cdot \frac{\delta y^{(j)}}{\delta x^{(j)}} \cdot \frac{\delta x^{(j)}}{\delta w^{(j)}}$$

By expanding $\frac{\delta L}{\delta y^{(j+1)}}$ in the same way we obtain:

$$\frac{\delta L}{\delta w^{(j)}} = ((\frac{\delta L}{\delta y^{(j+2)}} \cdot \frac{\delta y^{(j+2)}}{\delta x^{(j+2)}} \cdot \frac{\delta y^{(j+2)}}{\delta x^{(j+1)}}) \cdot \frac{\delta y^{(j+1)}}{\delta x^{(j+1)}} \cdot \frac{\delta x^{(j+1)}}{\delta y^{(j)}}) \cdot \frac{\delta y^{(j)}}{\delta x^{(j)}} \cdot \frac{\delta x^{(j)}}{\delta w^{(j)}}$$

Repeating this step allows us to express the gradient of the weights of any hidden layer as a function of $\frac{\delta L}{\delta \hat{y}}$ and the Jacobian matrices of any intermediate layers. Gradients following the form of $\frac{\delta x^{(j+1)}}{\delta y^{(j)}}$ are trivial to calculate as the output of layer j is the input of layer $j+1$. From $y^{(j)} = x^{(j+1)}$ it follows that $\frac{\delta x^{(j+1)}}{\delta y^{(j)}} = 1$ which allows us to discard terms following this pattern.

This way, we can express the gradients of L with respect to the weights of layer j as:

$$\frac{\delta L}{\delta w^{(l)}} = \frac{\delta L}{\delta \hat{y}} \cdot \left(\frac{\delta y^{(l)}}{\delta x^{(l)}} \cdot \frac{\delta y^{(l-1)}}{\delta x^{(l-1)}} \cdot \ldots \cdot \frac{\delta y^{(j+1)}}{\delta x^{(j+1)}} \cdot \frac{\delta y^{(j)}}{\delta x^{(j)}}\right) \cdot \frac{\delta x^{(j)}}{\delta w^{(j)}}$$

For the output layer of a neural network, the gradient is simply the gradient of the loss function with respect to $\hat{y}$: $\frac{\delta L}{\delta w^{(l)}}$. This explains the requirement for a differentiable loss function if a training algorithm based on gradient descent is used. By temporarily storing the Jacobian matrix for each layer once calculated, each Jacobian has to be only determined once per iteration of the gradient descent algorithm which is computationally efficient.

The aim is to minimize L thus the weights of the network are updated by moving in the reverse direction of the gradient. The second step of the backpropagation phase is to perform the following update:

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta)$$

In this update rule, η is a hyperparameter known as the learning rate, deciding the "step size" of the update. A single subsequent application of the feedforward propagation and backward propagation is defined as an iteration. The process of feedforward propagation and backpropagation is applied iteratively until a stopping criterion is met. Stopping criteria can for example be based on reaching a low enough threshold value of J, lacking improvements of J over multiple iterations or reaching a predefined maximum number of iterations.

3.4.3 Stochastic gradient descent

As we noticed earlier, L is a function depending on x, y and θ. The calculated gradient $\nabla_\theta L(\theta)$ is based on the data samples (x, y) used to calculate L. Calculating the gradient of the loss function based on only one single sample per iteration results in inaccurate approximations of the gradient of the entire underlying data distribution. This yields suboptimal performance and large differences between the gradients of different iterations causing unstable convergence behaviour. On the other hand, it is computationally expensive to calculate the average $\nabla_\theta L(\theta)$ for all samples in the dataset at each iteration of the gradient descent algorithm when the dataset is large. Instead, the gradient is approximated by using a subset of samples from the training dataset. This is known as Stochastic Gradient Descent (SGD). In stochastic gradient descent, the entire dataset is split in subsets of a predefined size. Each subset of data samples used to calculate the gradient for one iteration is referred to as a batch and the predefined batch size is a hyperparameter of the model. One epoch is defined as SGD looping through all unique batches in the dataset exactly once.

3.4.4 Momentum

One improvement to the SGD algorithm is the addition of momentum. The idea of momentum is to include an exponentially decaying moving average of previous gradients into the update of the weights. This stabilizes the updates and allows the SGD algorithm to escape local minima more easily.

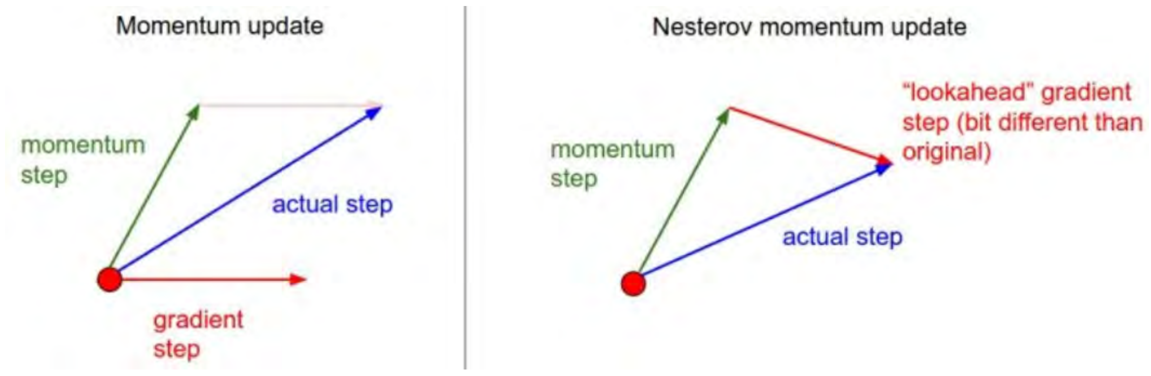


Figure 3.10: Difference between classic momentum and Nesterov momentum. Source: [32]

Let γ be a hyperparameter in the range of $[0, 1]$ denoting the relative weight of the previous updates. If the update in iteration t is called v_t, the new update rule including momentum is given by:

$$v_t \leftarrow \gamma v_{t-1} - \eta \nabla_\theta L(\theta)$$
$$\theta \leftarrow \theta + v_t$$

Nesterov momentum
Recently, Nesterov momentum has been gaining in popularity. The intuition behind the Nesterov momentum is relatively straightforward: Momentum will update the current position from θ to $\theta + \gamma v_{t-1}$ so it makes more sense to calculate the gradient of L with respect to the current batch at that position instead of the current position. As visualized in Figure 3.10, the gradient is not calculated using the current position $L(\theta)$, but at the position $L(\theta + \gamma v_{t-1})$, yielding the new update rule:

$$v_t \leftarrow \gamma v_{t-1} - \eta \nabla_\theta L(\theta + \gamma v_{t-1})$$
$$\theta \leftarrow \theta + v_t$$

3.5 DATASETS

A traditional Machine Learning experiment requires a dataset to be split in three subsets: a training set for training the model, a validation set for evaluating performance during training and optimizing hyperparameters between training sessions on unseen data, and a test set to evaluate the best performing version of the model on new unseen data which the model has not been indirectly or implicitly optimized for.

For a model to be able to detect models worldwide, it requires the model to have a generic representation of buildings. Buildings come in all sizes and shapes, and have different architectural styles in different parts of the world. From a Machine Learning perspective, this means that the "building" class has many possible representations. In other words, the data domain is large. For a model to correctly predict all buildings, a generic representation of the "building" class is required.

There are two ways to achieve this: The first, ideal approach is to provide training samples from the "building" class representing the entire domain. This requires that all types of buildings must be included in the training set. This is theoretically possible, but in practice it is impossible to create such a dataset. It would be extremely difficult to conclusively define all types of buildings in every relevant respect, and it would be extremely hard to create a database containing multiple samples for every entry of this inclusive list of building types. The second option is to use a subset of the domain as training set, and prevent the model from learning the specific characteristics of the subset. The second option is explored in this report.

For traditional Machine Learning models, the objective is to obtain the best possible fit on the underlying distribution of the training set. In this case however, the objective is to obtain the best possible fit on a wider data distribution. The assumption is made that a perfect fit on the distribution of the training data, which is a subset of the wider distribution, is too specific and therefore a sub-optimal fit for this objective. A method to measure the fit of the model on the wider distribution during the training process is proposed in Section 3.5.1. These measurements provide insights on the generalizability on the wider distribution of all model checkpoints. This allows for an informed decision on selecting a model with the best fit to the generic "building" class.

3.5.1 Transfer validation sets

A standard Machine Learning experiment requires three datasets: A train set for training the model, a validation set for evaluating performance during training on unseen data and a test set to evaluate the best performing version of the model on new unseen data which the model has not been indirectly or implicitly optimized for. For the following experiment, the *transfer validation set* is introduced as a new type of dataset.

The purpose of an transfer validation set is to evaluate performance on another data distribution during training at the end of each epoch, similar to the standard validation set. Design decisions such as hyperparameter selection and decisions made during the training procedure such as early stopping are still made based on the performance on the standard validation set. The purpose of introducing these transfer validation sets is to measure the generalizability of the model on instances of another distribution during training. A test set only provides information about the generalizability after training has finished, and often only for the checkpoint where the fit on the standard validation set is optimal. In contrast, a transfer validation set provides this generalizability information during the training process. The transfer validation set allows to select the most appropriate model checkpoint in terms of generalizability to another data distribution. Following the argumentation for the necessity of a test set because of implicit optimization on the standard validation set, each transfer validation set also requires a test set from the same data distribution.

3.6 LOSS FUNCTIONS

The importance of using the right loss function has been briefly highlighted in Section 3.2.5. A few popular loss functions for segmentation tasks are described in this section. These loss functions are defined for a single prediction. In image segmentation tasks, each pixel in the input image equals one prediction. Since one image is considered to be one sample, the average loss of all pixels in an image is reported as the sample-wise loss for brevity in segmentation tasks.

3.6.1 Cross-entropy loss

Cross-entropy loss is a popular loss function. Given that there are C classes to distinguish from each other, with ground truth label y and prediction $\hat{y}$ the cross-entropy loss function CE can be defined as:

$$CE = -\sum_{c=1}^{C} y_c \log(\hat{y}_c)$$

Recall that for detecting buildings, we are interested in one type of object and thus the problem space contains two classes: "building" and "no building". In the specific case of two classes to predict, the cross-entropy loss function is also known as Binary Cross Entropy (BCE). For binary ground truth value y and continuous prediction value $\hat{y}$ the BCE is defined as:

$$BCE = -y\log(\hat{y}) + (1-y)\log(1-\hat{y})$$

Note that the BCE is only a valid loss for values y in range $(0,1]$. When using the BCE loss function, it is therefore common to use the sigmoid activation function in the output layer of the neural network to map the network output to the desired $(0, 1)$ range.

3.6.2 Jaccard loss

Given two sets A and B the Jaccard index, which is also known as the Intersection over Union (IoU) is defined as:

$$Jaccard\,index = \frac{Intersection}{Union} = \frac{|A \bigcap B|}{|A \bigcup B|}$$

For binary ground truth vector y and binary prediction vector $\hat{y}$ the Jaccard index is defined as:

$$Jaccard\,index = \frac{\sum y_i \hat{y}_i}{\sum y_i + \sum \hat{y}_i + \sum y_i \hat{y}_i + \epsilon}$$

Here ϵ is a small value ensuring that the function is well-defined around zero when there are no positives in the ground truth vector or binary prediction vector. The Jaccard index is a performance metric. The corresponding Jaccard loss function is defined as:

$$Jaccard\,loss = 1 - Jaccard\,index$$

This definition cannot be directly used as a loss function for a neural network however as $\hat{y}$ must be differentiable and therefore cannot be binary. If the softmax activation function is used in the output layer of the neural network, we know that $\hat{y}$ is differentiable and will be in the range of $(0, 1)$. The Jaccard index can be approximated using the continuous probability vector $\hat{y}$ instead of a binarized $\hat{y}$. An advantage of this approximation over binarizing the continuous output based on a classification threshold is that the certainty of the prediction influences the outcome of the approximated Jaccard index. An advantage of the Jaccard loss function over the binary cross-entropy loss is that the Jaccard loss function is insensitive to class imbalance whereas the cross-entropy loss is sensitive to class imbalance issues [40].

3.6.3 Dice loss

The Dice coefficient is formally known as Sørensen-Dice or Dice similarity coefficient. The Dice coefficient is defined as:

$$Dice\,index = 2 \cdot \frac{\sum y_i \hat{y}_i}{\sum y_i + \sum \hat{y}_i + \varepsilon}$$

Similarly to the Jaccard index, the Dice index can be approximated by using the output of the softmax activation function as continuous probability vector $\hat{y}$ instead of a binarized version of $\hat{y}$. The Dice index is a performance metric. The corresponding Dice loss function is defined as:

$$Dice\,loss = 1 - Dice\,index$$

The relation between the Jaccard index and the Dice index can be expressed in the following way:

$$Jaccard\,index = \frac{Dice\,index}{2 - Dice\,Index}$$

3.6.4 L2 loss

The L2 loss function is used as a means of regularization and is used as a secondary loss function in conjunction with another loss function. The L2 loss function simply sums the squared weights of the network, as shown in its definition:

$$L2(W) = \lambda \sum_{w \in W} w^2$$

with W the weight matrix of the neural network and λ a hyperparameter. In a neural network minimizing loss function $L_{loss}(y, \hat{y})$, adding the L2 loss yields a new composite loss function L_{total} to be optimized instead:

$$L_{total} = L_{loss}(y, \hat{y}) + L_{L2}(W)$$

The L2 loss penalizes large weights, discouraging the optimization method to let weights to grow explosively large. The largest weights have the largest relative contributions to the L2 loss, because their relative magnitude is even further augmented by the quadratic penalty function. As stated in Section 3.2.2 the weights resemble the relative importance of that weight in the network. By discouraging the use of relatively large weights, the L2 loss tries to prevent that only a few weights have a dominant influence on the predictions of the neural network, rendering all other weights practically irrelevant.

3.7 PERFORMANCE METRICS

Similarly to the loss function, performance metrics are indicators of the model performance to be evaluated during training and testing the model. Performance metrics are often introduced because the loss function is limited in its ability to report the relevant performance measurements.

	Ground truth: building	Ground truth: no building
Prediction: building	True positive (TP)	False positive (FP)
Prediction: no building	False negative (FN)	True negative (TN)

Table 3.1: A confusion matrix classifying predictions given a binary task

3.7.1 Precision and recall

Predictions of a model performing the classification task should be accurate in two distinct ways. In binary classification there are positives (e.g. "building") and negatives (e.g. "no building"). The binarized predictions can be categorized in a so-called confusion matrix as shown in Table Table 3.1. When predictions are real valued instead of binary, a classification threshold can be applied on these values to obtain binarized predictions. To measure the quality of the predictions of a model, the *precision* performance metric is often used, which is defined as:

$$Precision = \frac{True\ positives}{True\ positives\ + False\ positives}$$

The *precision* measures which part of the predicted buildings are in fact a building. A model that only predicts buildings when it has a high confidence that its predictions are correct, will yield a high precision score. To measure the sensitivity, the *recall* performance metric is also often used, which is defined as:

$$Recall = \frac{True\ positives}{True\ positives\ + False\ negatives}$$

Recall measures which part of all buildings are correctly detected as being a building. Models generating many positive predictions will miss relatively few buildings yielding a high recall score.

3.7.2 F1 score

The F1 score is a metric which balances the precision and recall performance metrics. The F1 score is the harmonic mean of the precision and recall and is defined as:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision\ + recall}$$

A model optimized for the F1 score will seek a balance between precision and recall.

3.7.3 Intersection over Union and Dice index

The IoU is also known as the Jaccard index as mentioned in Section 3.6.2. Similarly, the Dice index as described in Section 3.6.3 can also be used as a performance metric.

3.8 TRAINING OPTIMIZATION

There are many techniques which can aid the training process, i.e. by increasing the speed of convergence, reducing computational complexity or improving the quality of the model. A few of the most commonly used techniques are described in this section.

3.8.1 Batch normalization

The use of batches in stochastic gradient descent allows for another optimization technique called batch normalization. Batch normalization is often applied on the intermediate output of each hidden layer of a neural network. All output values $y^{(j)}$ for layer j are normalized by dividing by the standard deviation and subtracting the mean value of $y^{(j)}$ for all samples in the batch combined. This ensures that the magnitude of the values $y^{(j)}$ stay within a certain magnitude, which has a stabilizing effect on the gradients during the backpropagation phase. This stabilization allows for the usage of larger learning rates without experiencing gradients exploding in magnitude. Stable, regularized gradients result in stable, regularized weight updates, thus batch normalization also has a regularizing effect on the weights of the network [28].

3.8.2 Augmentations

The true data distribution is approximated by the empirical data distribution provided by the training samples. Using more training samples will increase the quality of this approximation, allowing the model to obtain a better fit to the true data distribution. The number of data samples available to train on is finite in the vast majority of cases, and often limited by data collection limitations. A popular technique is to artificially increase the number of training samples by creating a multitude of variations of the available training samples. For image datasets, these variations can be created by changing the orientation of the sample, cropping, resizing or blurring the image, or performing other affine transformations on the image. The gist of data augmentation is that the artificially augmented samples remain a representative sample of the data distribution the model should fit.

Another welcome effect of using data augmentation is that the variation introduced by the artificially augmented samples often increases the robustness of the model to these variations. As a result, the generalizability of the model often improves.

Augmentations can not only be used during the training phase, it can also be used when inferencing on new unseen data. This is known as Test Time Augmentation (TTA). Given a sample, normally it is fed to the model in inference mode resulting in predictions for that sample. With test-time augmentation, a number of augmented samples is created based on the unseen sample. These augmented samples and the original sample are all fed to the model in inference mode. The resulting predictions are merged, i.e. by averaging them, to create the final prediction for the original sample.

3.8.3 Pretraining

Another method to optimize the training process is by using a pretrained model. Training a CNN model to its fullest extent from random initialized weights is often computationally expensive without any guarantee that a global minimum will be eventually reached. No guarantees can be given on the optimization of neural networks due to the stochastic nature of weight initialization and stochastic gradient descent in combination with non-convex loss functions.

A pretrained model is a set of weights coming from a model that has been extensively trained on the same or a similar task. These weights already have meaningful values in relation to each other. Especially the weights from the first few layers in the network are valuable as they are already trained to describe general, conceptually simple patterns. The last few layers are more class or task specific and contain filters that won't be directly reusable, but even these have meaningful values in relation to filters in adjacent layers which is helpful when training the model to perform a different task [18]. As a result, models based on pretrained models often outperform randomly initialized models.

One particular popular source for pretrained models is the ILSVRC ImageNet Large Scale Visual Recognition challenge [54] which is described in section Section 2.1. The size and quality of the ImageNet dataset in combination with the variety in the many objects classes results in well-performing models of which the filters of the first few layers are relatively generic in nature. The weights of models trained on the ImageNet class are therefore relatively well-transferable to other image recognition tasks [18].

3.8.4 Overfitting and early stopping

Allowing a model to learn too many specific details about the training set is known as overfitting. An overfitting model performs well on samples from the training set, but fails to generalize on the underlying distribution of the data. Hence, an overfitting model will perform worse on validation or test data, even when the validation and test set come from the same distribution as the training set. To prevent overfitting it is crucial to either have training data of the same distribution as the data that the model will be used on, or to restrain the model in its learning capability.

A signature indication of overfitting is a steadily decreasing training loss while the loss on the validation set does not decrease for several subsequent epochs. As the name suggests, early stopping stops the training of a model when certain predetermined criteria are met. To prevent overfitting, early stopping can be used to stop the training when the validation loss does not decrease for a predefined number of subsequent epochs.

3.9 ARCHITECTURES: RESU-NET, U-NET AND MASK R-CNN

Three popular CNN architectures are described in this section. The ResNet architecture is a base architecture that many other architectures incorporate. The U-net architecture is a popular architecture for image segmentation and the Mask R-CNN architecture is popular for object detection and instance segmentation tasks.

3.9.1 ResNet

The ResNet classification models [21] are a popular base architecture of many convolutional neural networks. The power of a ResNet model is that it reformulates the mapping it attempts to learn. If $H(x)$ is the desired mapping to be learned by convolutional layers in a CNN, then a Residual Unit instead attempts to learn the residual function $F(x) = H(x) - x$.

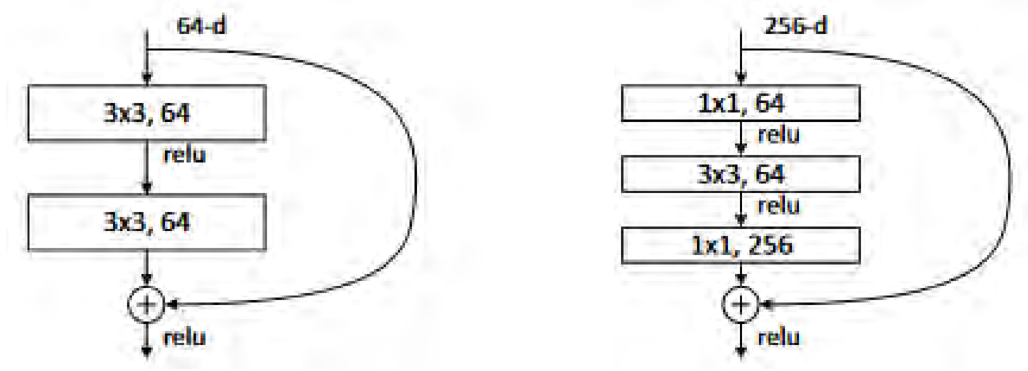
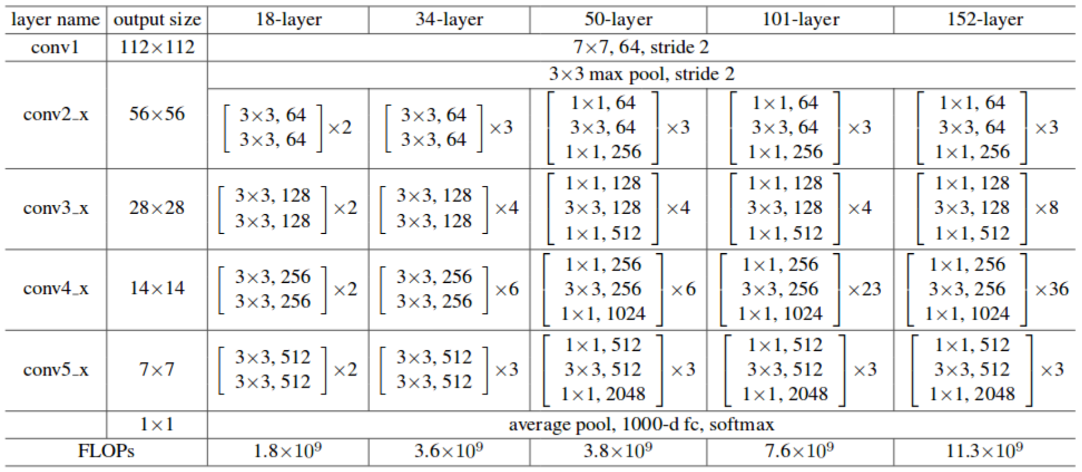


Figure 3.11: On the left a residual unit containing two $3\,x\,3$ conv layers with 64 filters learning the residual function $F(x) = H(x) - x$ and a skip connection performing the identity mapping. On the right: A bottleneck residual unit containing three conv layers where the 256 input filters are first reduced to 64 to lower the computational cost of the $3\,x\,3$ conv and restored to 256 filters afterwards. Source: [21]



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 3.12: Overview of architectures of ResNet models with 18, 34, 50, 101 and 152 layers respectively when using input images of $224\,x\,224$ pixels. Source: [21]

To provide the desired output, a skip connection which performs the identity mapping is added to the network, as shown in Figure 3.11. He [22] shows that it is easier to optimize a residual function $F(x)$ compared to the original function $H(x)$ and that the identity mappings aid the gradient flow throughout the layers during the training process. A ResNet model contains multiple residual units. ResNets up to 34 layers contain up to 17 traditional Residual Units consisting of two $3\,x\,3$ conv layers each, whereas deeper ResNets utilize so called Residual bottleneck blocks which consist of a $1\,x\,1$, a $3\,x\,3$ and a $1\,x\,1$ conv layer subsequently. In Figure 3.12 the architectures of multiple common ResNet models are displayed.

The ResNet architecture requires that the width and height of the input image are both multitude of $2^5 = 32$ pixels. The requirement of the input image to be a multiple of 32 is to ensure an integer number of pixels, even at the deepest levels of the network. In the ResNet architecture, the input image is downsampled five times by pooling layers reducing each image dimension by a factor of 2 as shown in Figure 3.12. If the dimensions of the input image are no multitude of 32, then the image can be padded first as explained in Section 3.3.1 to increase its dimensions to a multitude of 32.

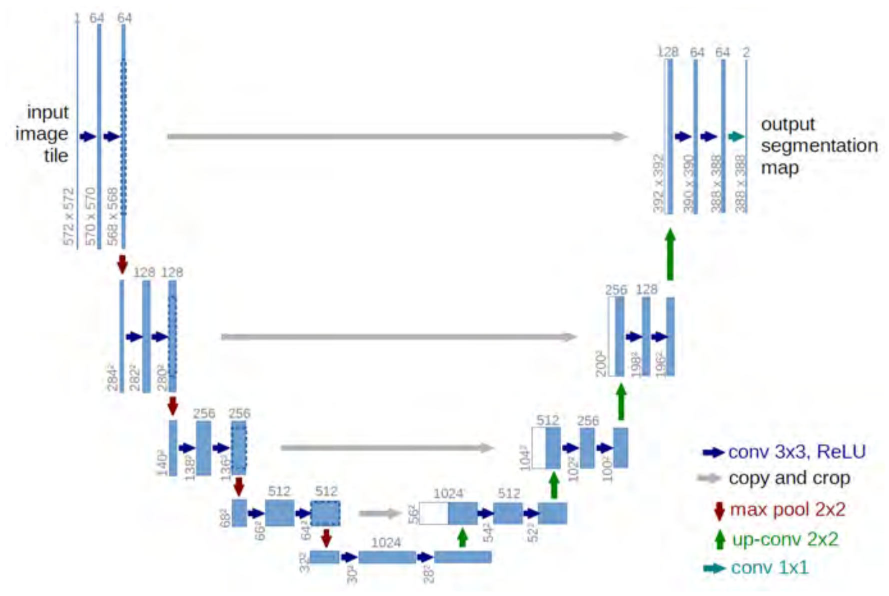


Figure 3.13: The U-net architecture. Source: [53]

3.9.2 U-net

The U-net architecture is shown in Figure 3.13. The U-net consists of two parts. The left side is called the encoder, in which information from the input image is being encoded using an increasing number of filters per layer at a decreasing spatial resolution. The purpose of the encoder is to obtain a high-level representation of the image. The encoder consists of several $3\,x\,3$ conv layers followed by ReLU activations and $2\,x\,2$ max-pooling operations for downsampling. At each downsampling step, the number of filters used by the conv layers is doubled. The right side of the U-net is called the decoder. The purpose of the decoder is to translate the features generated by the encoder into an accurate high-resolution feature map. The decoder consists of several $3\,x\,3$ conv layers followed by ReLU activations and strided convolutional layers which upsample the resolution by $2\,x\,2$. After each strided convolution, the number of features per conv layer is halved.

The grey horizontal arrows in Figure 3.13 represent so called skip connections, where the intermediate feature maps of the encoder are sent to symmetrically-equivalent layer in the decoder. In the decoder, these high resolution feature maps coming from the encoder are concatenated with the high-level representation from the deepest layer of the encoder. Combining spatially rich information with contextually rich information allows the decoder to create an accurate high-resolution feature map based on high-level features. There is no predefined loss function for a U-net. The most widely used image segmentation loss functions are mentioned in Section 3.6.

Sometimes, the encoder of the U-net is replaced by other popular encoding architectures such as models from the ResNet family because they are more powerful. When the encoder is replaced by a ResNet, the new hybrid architecture is called a ResU-net. Similar to the U-net, the outbound skip connections in the ResNet are inserted right before the resolution is downsampled. The final $1\,x\,1$ average pool layer with softmax activation of the ResNet is discarded. A ResNet reduces the resolution five times in total whereas the U-net reduces the dimensions only four times. This is solved in the first two layers of the ResNet where both layers reduce the resolution: only the second layer contains an outbound skip connection.

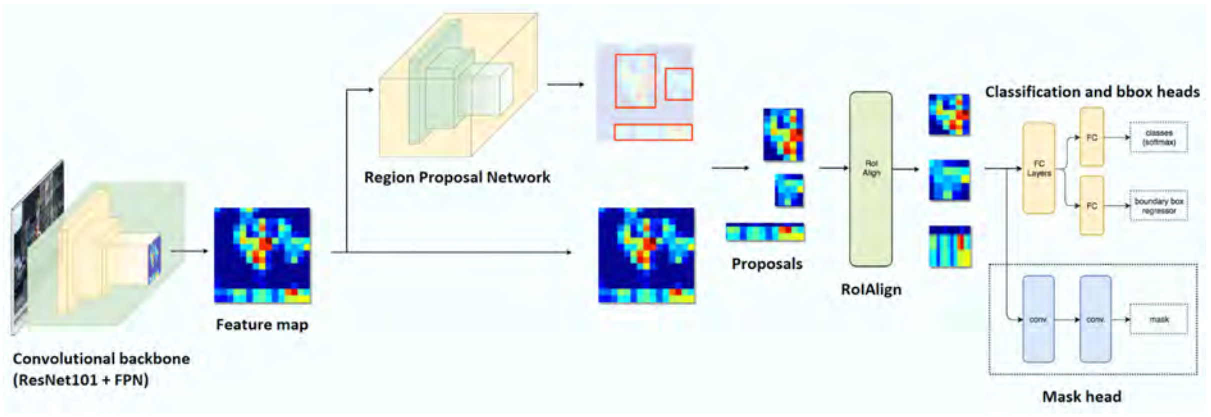


Figure 3.14: The Mask R-CNN architecture. Source: [24], adapted]

3.9.3 Mask R-CNN

Mask R-CNN is a two-stage object detection model. Its architecture is shown in Figure 3.14. The first part of the Mask R-CNN architecture is a ResNet backbone. By default the ResNet-101 model is used as backbone, optionally in conjunction with a Feature Pyramid Network [39]. The output of this backbone is a feature map. The next step is the Region Proposal Network, which uses the feature map to determine which regions are most likely to contain an object of interest.

The Region Proposal Network (RPN), generates a large number of anchor boxes. These anchor boxes are generated around positions in the image using predetermined intervals called anchors. The anchor boxes are generated by predetermined aspect ratios and scales. Any number of scales and up to three ratios can be used, with default values for the aspect ratios being 0.5 (horizontally oriented rectangle), 1 (square) and 2 (vertically oriented rectangle). Figure 3.15 illustrates a set of generated anchors generated by the Region Proposal Network using three scales and three ratios. In Mask R-CNN, for each anchor box, the average activation value of the feature map within the anchor box is calculated. By default the 2000 anchor boxes with the highest average activation values are stored, the others are discarded.

These 2000 anchor boxes with the relative highest probability of containing an object are called proposals (Figure 3.16). The proposals are all transformed to squares of the same dimension (a hyperparameter, usually 28x28 or 56x56 pixels) so they can be evaluated equivalently by the second part of the architecture. This transformation is performed by the RoIAlign layer.

The resized proposals are sent to the three network heads. The classification head predicts for each class the likelihood that the main object in the proposal is of the given class. The exact location and size of each proposal is refined using a box regression head, which predicts the required shift in location and window size to obtain the best possible fit to the object. To remove overlapping proposals, class-wise non-maximum suppression (NMS) is applied to discard all the lower-scoring overlapping proposals and retain only the highest scoring non-overlapping proposals as shown in Figure 3.17. The mask head predicts on a pixel-level which pixels belong to the object of interest and which are background pixels, resulting in a pixel mask of the object shown in Figure 3.18. The outcome of Mask R-CNN is shown in Figure 3.19.

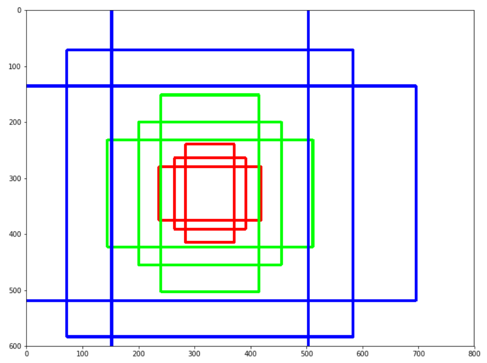


Figure 3.15: Figure 3.15: Nine anchor boxes generated centred on an anchor at (320, 320) using three anchor scales (160, 240 and 480) and three anchor ratios (0.5, 1 and 2). Source: [11]

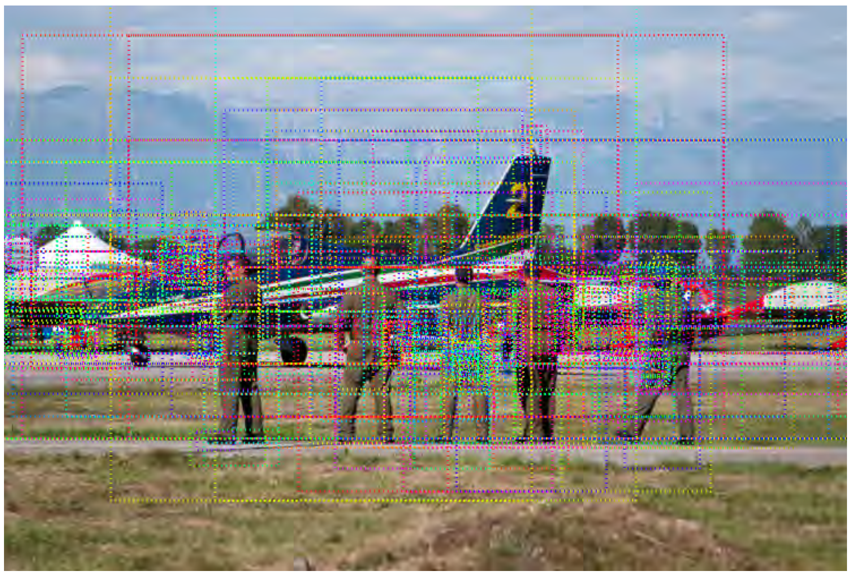


Figure 3.16: The proposals generated by the Region Proposal Network. Source: [1]

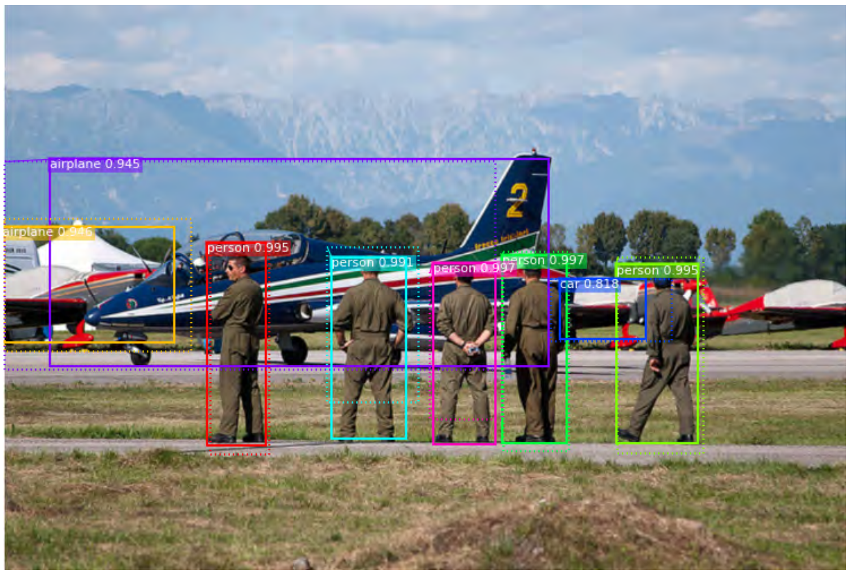


Figure 3.17: The results of the box and classification heads after applying non-maximum suppression to remove overlapping proposals. The original proposals are shown with dotted lines whereas refinements made by the box regression head are shown as solid boxes. The highest scoring class and the corresponding predicted certainty of the classification are also shown. Source: [1]

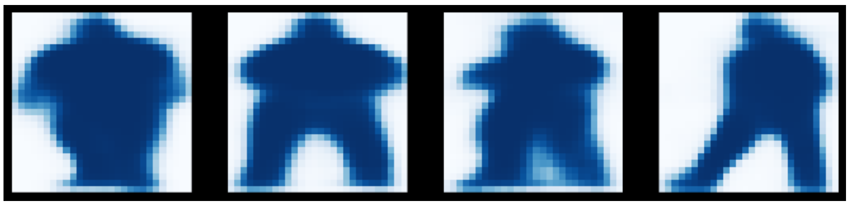


Figure 3.18: Pixel-wise predictions of the mask head for the refined proposals. Source: [1]

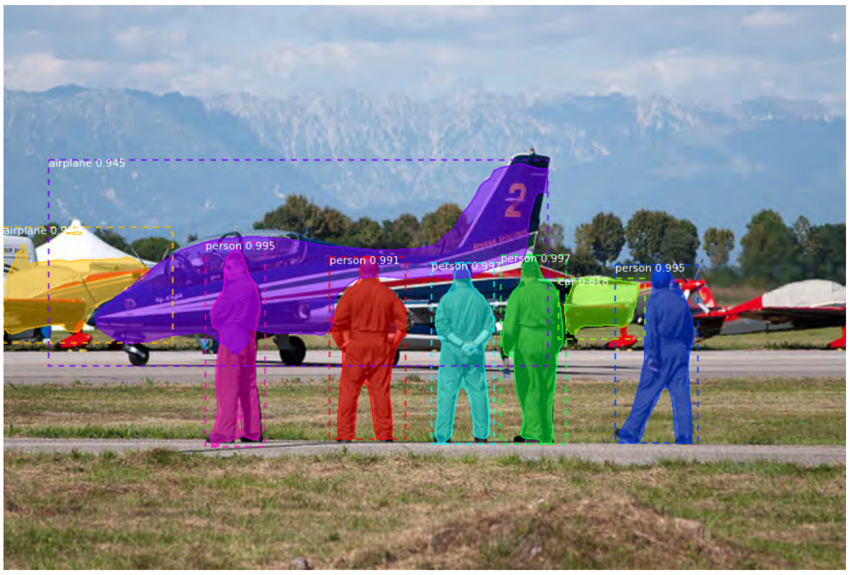


Figure 3.19: The final result of the Mask R-CNN algorithm. Source: [1]

4 DATA

4.1 CHAPTER OVERVIEW

For this internship, three data sources were used as shown in Table 4.1. The crowdAI dataset is a homogeneous, well-prepared dataset of a fixed size containing images from most likely the United States. The crowdAI dataset is selected because of its large size combined with a high data quality and consistency, especially with regards to the ground truth labelling and alignment. The data source consisting of Bing Maps and OpenStreetMap (OSM) is selected because it theoretically provides worldwide coverage, allowing for the creation of datasets of any custom size in any desired geographical area. The Readar data source is used because of its high resolution imagery and high quality annotations whilst being sufficiently large. The exact area definitions (geometries) can be found in Appendix A.1. A list of all datasets is provided in Table 4.2.

Source	Image size (pixels)	Resolution (cm)	Geographical coverage	Topography
crowdAI	300 x 300	30	Unknown (probably U.S.)	Mainly detached buildings
Bing Maps & OSM	256 x 256	30	Worldwide	Location dependent
Readar	1536 x 1536	10	The Netherlands	Mainly connected buildings

Table 4.1: Data sources overview.

Dataset	Data source	Images	Resolution	Ground truth issues
crowdAI train	crowdAI	280,741	30	None
crowdAI validation		60,317	30	None
crowdAI test		60,697	30	None
Malawi, Nsanje	Bing Maps & OSM	38,046	60	No ground truth available
Malawi, Nsanje		201	30	No ground truth available
Peru, Equitos		880	30	No ground truth available
Zambia, LusakaNorth		3360	30	Temporal gap
Zambia, LusakaWest		3168	30	Temporal gap
Zambia, Zambezi		3679	30	Temporal gap, misaligned
Zambia, Kalabo		749	30	No ground truth available
Zambia, Mongu		11,273	30	None
Amersfoort	Readar	3498	10	Minor temporal gap, parallax effect
Arnhem		4751	10	Minor temporal gap, parallax effect
Breda		3496	10	Minor temporal gap, parallax effect
Maastricht		1870	10	Minor temporal gap, parallax effect
Molenwaard		2628	10	Minor temporal gap, parallax effect
Heerlen		3819	10	Minor temporal gap, parallax effect

Table 4.2: Datasets overview.

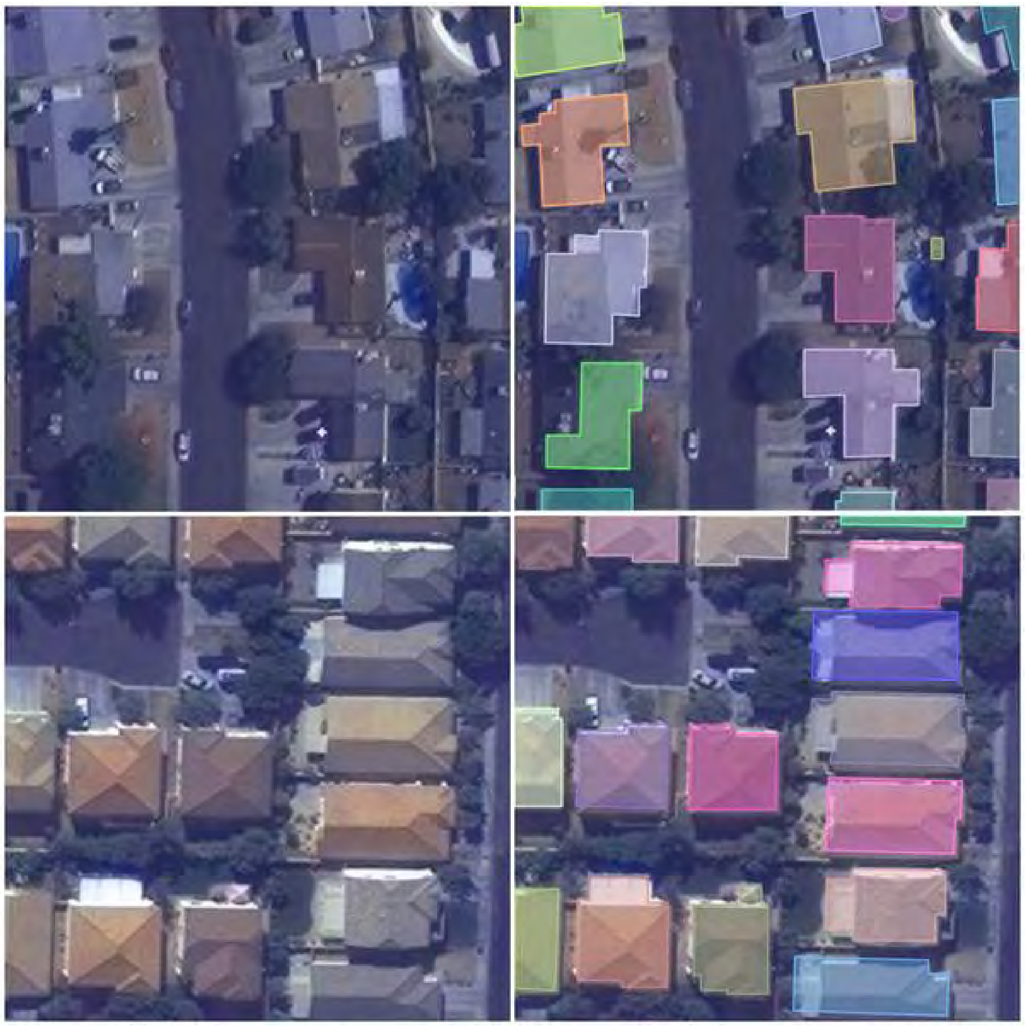


Figure 4.1: Left: Two sample images from the crowdAI dataset. Right: Sample images including ground truth annotation. Source: [25]

4.2 CROWDAI

CrowdAI is an online platform that aims to solve real-world problems by hosting Artificial Intelligence challenges. In the summer of 2018, crowdAI hosted the Mapping Challenge [25]. The goal of the Mapping Challenge is to detect buildings on satellite imagery. The imagery is provided as 300 by 300 pixel RGB images in jpeg format with a 30 centimeter ground resolution. A training set, validation set and test set were provided. The training set consists of 280,741 images with corresponding ground truth masks, the validation set consists of 60,317 images with corresponding ground truth masks and the test set contains 60,697 images without a ground truth. Every image contains at least one (part of a) building, with an average of 8.5 buildings per image. The average building size is 2344 pixels, thus on average buildings comprise around 20,000 of the 90,000 pixels per image, or roughly 22.22 percent. The ground truth is of excellent quality as the labels were manually created based on the imagery. The majority of the buildings in the data set are individual, freestanding buildings as shown in Figure 4.1. The dataset seems to be relatively homogeneous, meaning there is a low diversity in background scenery, building architecture and rooftop types.

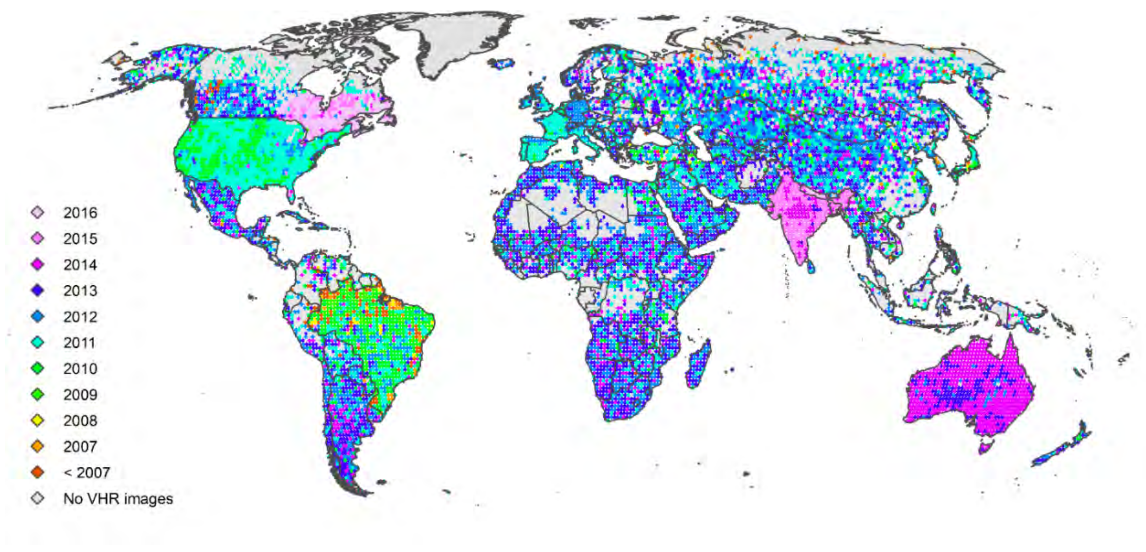


Figure 4.2: An overview of the vintage of Bing Maps imagery made in January 2017. Source: [38].

No geographical information was provided on the location of the imagery to prevent participants of the challenge to use external data sources of the same area to gain an unfair advantage. However, when looking at the characteristics of the imagery there is a strong suspicion that the imagery originates from the south-western part of the United States. Worth mentioning is that the Mapping Challenge was set up in collaboration with the Humanity & Inclusion organization to aid the Missing Maps project.

4.3 BING MAPS & OPENSTREETMAP

Datasets can also be created by combining publicly available data. The Missing Maps initiative uploads validated map data created by volunteers to the OpenStreetMap platform. Microsoft Bing also supports the OpenStreetMap platform and integrates the information in their Bing Maps service.

4.3.1 Bing Maps imagery

Bing Maps satellite imagery is available worldwide, but the quality and characteristics of the imagery differs between areas as the imagery provided by Bing Maps is a combination of various imagery sources. Although Bing Maps provides satellite imagery worldwide, only for most inhabited areas a 240 centimeter or better ground resolution is offered. For rural areas, the highest ground resolution available is often 60 cm. A ground resolution of 30 cm is almost always available for urban areas. Imagery is available as 256 by 256 pixel RGB images in jpeg format. For a handful of selected cities, special aerial imagery is available with a 15 cm ground resolution. Figure 4.2, published by Lesiv et al. [38], provides a global overview made in January 2017 of the availability of satellite imagery on Bing Maps with a ground resolution smaller than 5 meters.

4.3.2 OpenStreetMap ground truth

The OpenStreetMap data quality varies between locations, and often depends on the origin of the data. The origin of OpenStreetMap information could be some external dataset: often snapshots from cadastral databases or other official geospatial datasets are uploaded in bulk to OSM. For these data sources, we can assume that all data is georeferenced in a consistent way. In case of misalignment of the ground truth to Bing Maps imagery, a global shift on OSM data from this source will completely remove or significantly reduce the misalignment.

The origin of OSM data could also be manual mapping by volunteers. Local knowledge of volunteers affects the quality of the mapping. Sometimes volunteers map local areas they know well, often they map distant areas they have never visited in person and completely rely on satellite imagery. The imagery source used during the mapping is another important factor in the quality of manually mapped OSM data. Bing Maps and Digital Globe imagery is available worldwide with additional sources available depending on the area. From all available imagery sources available by default to volunteers, only the Bing Maps imagery can be used for this internship due to licencing issues. Therefore, the optimal situation for manually mapped OSM data is when Bing Maps is used as imagery source, or when the same underlying source imagery is used. In these cases, there is no misalignment between the ground truth and Bing Maps imagery. In all other cases, there could be misalignment issues as shown in Figure 4.3. The Missing Maps volunteers and other OpenStreetMap contributors have complete freedom in the selection of the parts they map. To further complicate matters, mappers can decide for themselves which imagery source they use and switch to another source whenever they like. There is no administration of the imagery source used during the mapping process, rendering the appliance of global shifts based on imagery source impossible. This results in an inconsistently georeferenced data set in OpenStreetMap, with varying degrees of misalignment.

For the Netherlands, snapshots from the BAG and BGT databases are regularly used to update OpenStreetMap, ensuring that the OpenStreetMap data is almost identical to the BAG and BGT databases from Kadaster with respect to buildings. In Section 1.2.1 potential issues with the data collection of the BAG and BGT databases are highlighted.

4.3.3 Temporal gap

An important issue when combining the Bing Maps imagery with OpenStreetMap data as ground truth is a temporal disparity between the data. The OpenStreetMap database is updated continuously and aims to reflect the current situation as accurately as possible, while Bing Maps imagery provides a snapshot from the situation in the past. As Figure 4.2 originating from Lesiv et al. [38] shows, for most parts in the world this results in a temporal gap of 5 to 7 years. All building construction and deconstruction during this period causes mismatches in the dataset. There is a history available for OpenStreetMap but historic versions often provide significantly lower coverage causing large gaps in the ground truth. The assumption is made that the inaccuracies introduced by the temporal mismatch have less impact on the dataset quality than using older datasets with lower coverage. Because of this assumption, the choice was made to use the latest version of the OpenStreetMap database.

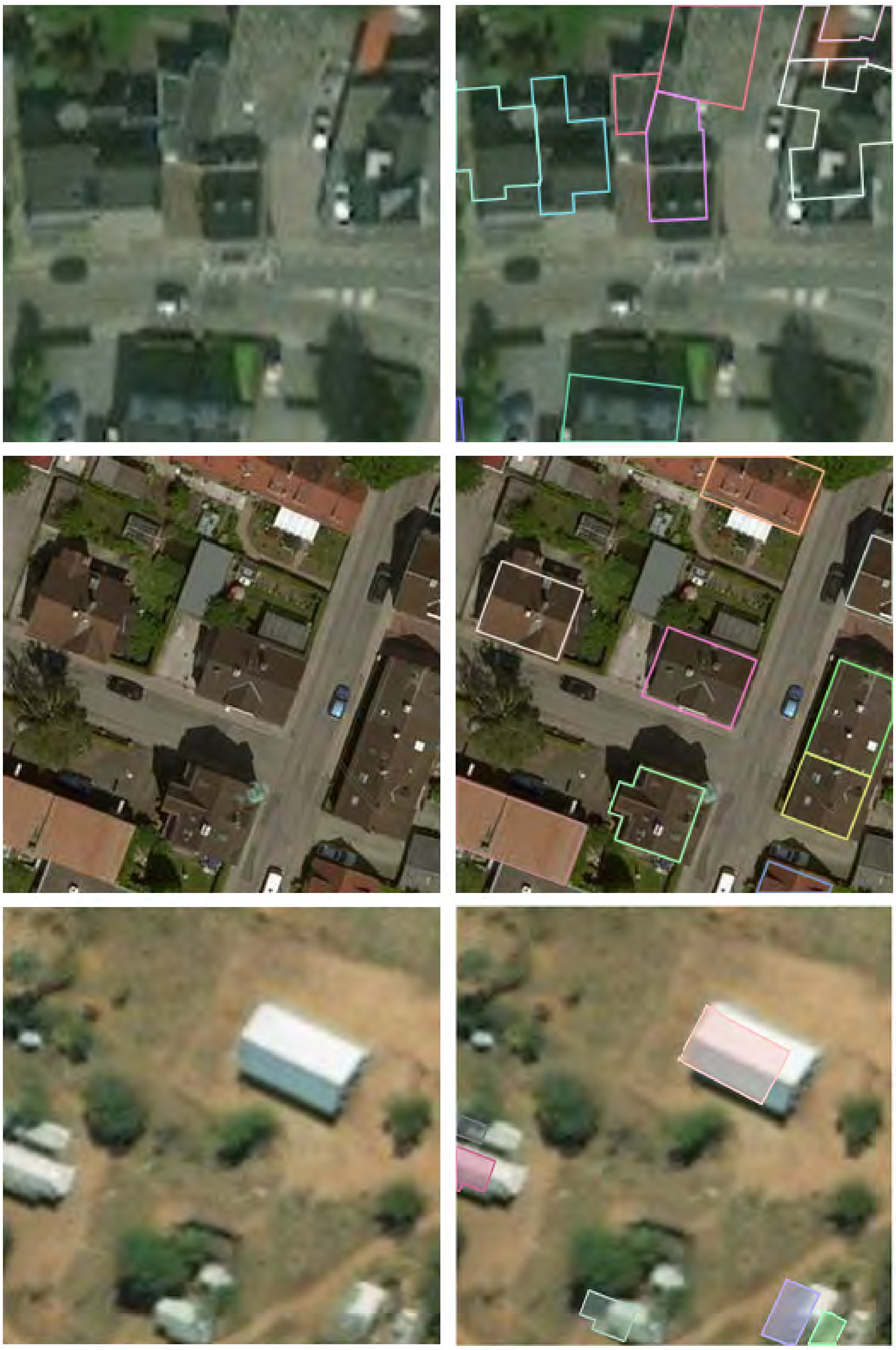


Figure 4.3: Left: Samples from Bing Maps imagery, right: OpenStreetMap building outlines plotted on Bing Maps imagery. Location in descending order: Hulsberg, Limburg, Netherlands, 30 cm resolution - Augsburg, Bayern, Germany, 15 cm resolution - Nsanje, Nsanje, Malawi, 30 cm resolution.

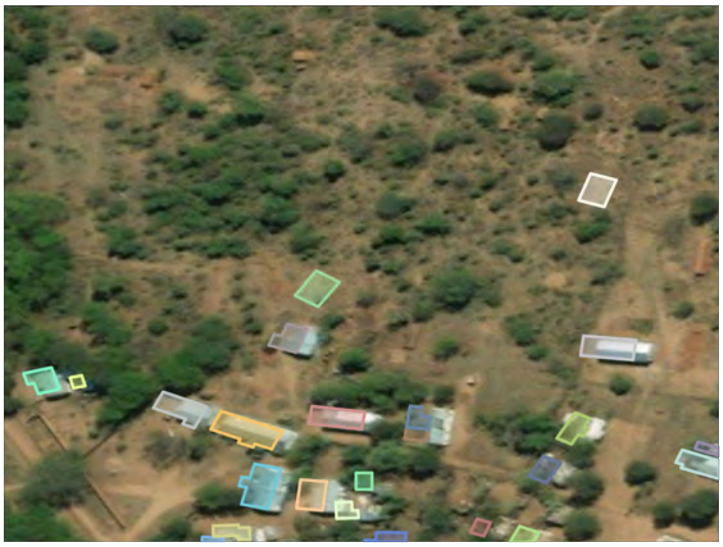


Figure 4.4: Sample from OSM ground truth labels overlaid on Bing Maps satellite imagery of Nsanje at 30 cm resolution. The imagery does not show any buildings belonging to the two most upper instances. There also is a consistent misalignment between the ground truth labels and the imagery.

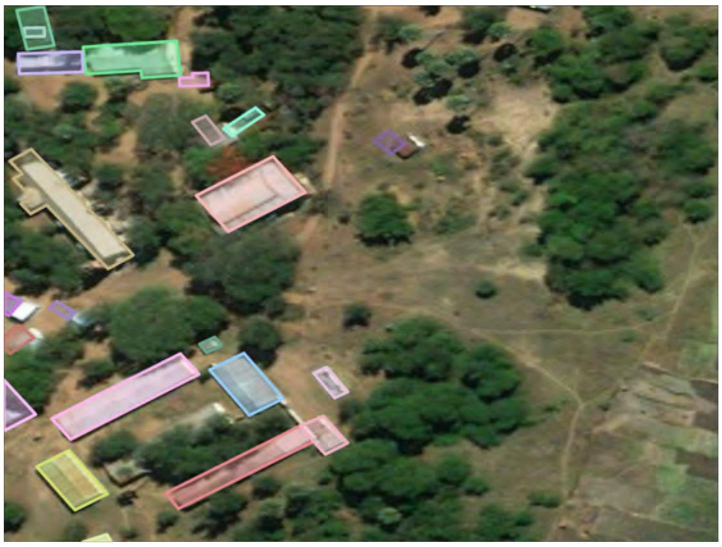


Figure 4.5: Sample from OSM ground truth labels overlaid on Bing Maps satellite imagery of Nsanje at 30 cm resolution. Some labels are properly aligned with the imagery, whereas some others are misaligned. One large building in the bottom left corner of the imagery is not labelled in the ground truth and in the top left corner the ground truth contains an incorrect label.

4.3.4 Dataset: Nsanje, Malawi

Two datasets for Nsanje in Malawi are available. The largest dataset contains 38,046 images and has a 60 cm resolution with no ground truth available. The vast majority of the images in this dataset do not contain buildings. The smaller dataset contains 201 images at a 30 cm resolution and OpenStreetMap labels as ground truth. Roughly half of all images in this set contain at least one building. However, Figure 4.4 and Figure 4.5 highlight some of the issues with the alignment between Bing Maps imagery and OSM ground truth labels.

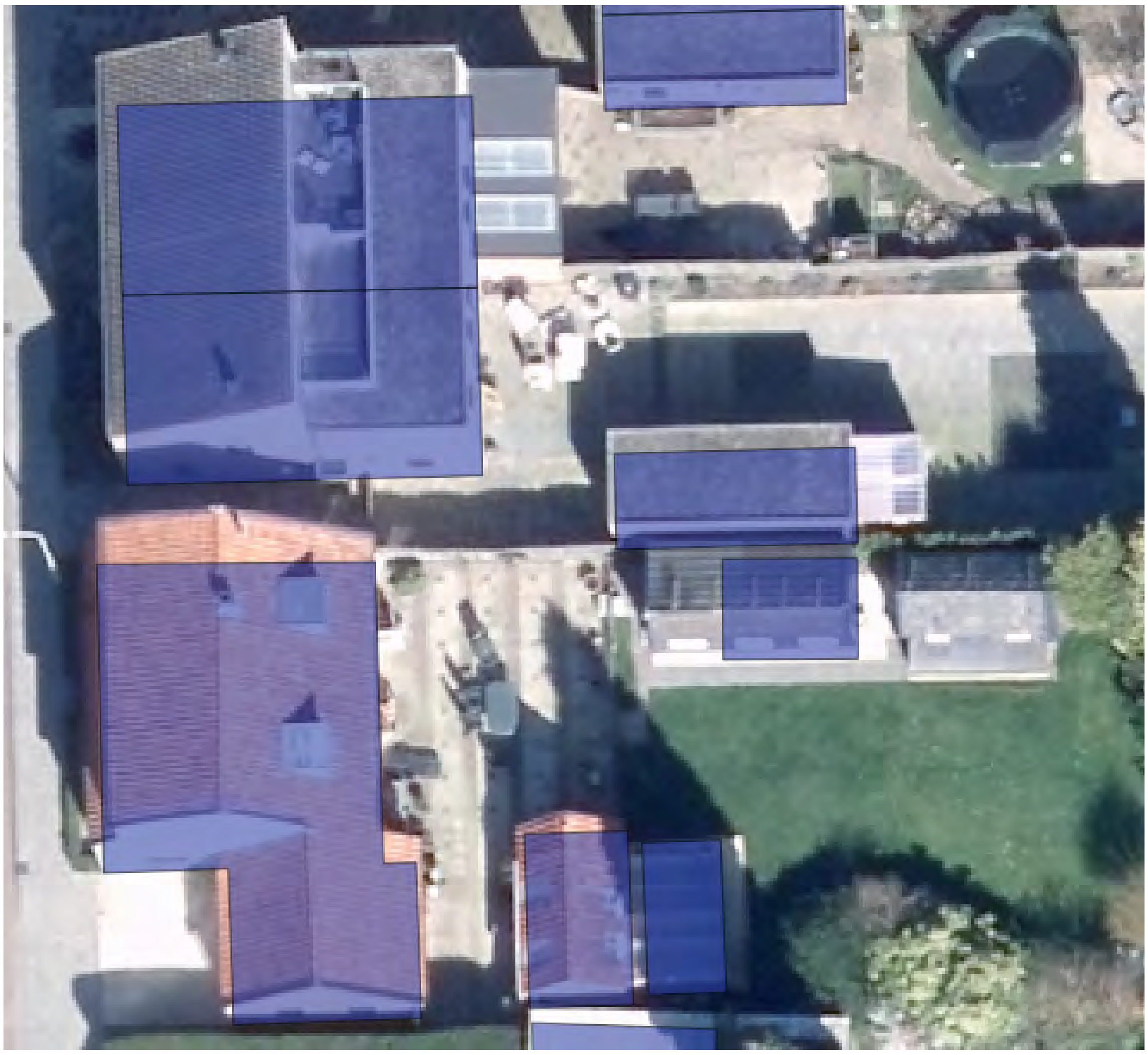


Figure 4.6: This sample image illustrates the quality of Readar's aerial imagery and ground truth labels. This also illustrates the parallax effect, revealing the building facades.

4.4 READAR

Readar provides a private, in-house dataset using aerial imagery of the Netherlands. The ground resolution of the imagery is 10 cm and any custom tile size can be selected to crop RGB images in TIFF format from the imagery. The aerial imagery has been created by flying over the Netherlands on several different days in 2017. In this case, a tile size of 1536 by 1536 pixels was used. Figure 4.6 shows a sample of this dataset.

4.4.1 Temporal gap

Readar also has its own database of buildings, consisting of the BAG and BGT databases from Kadaster and buildings manually detected on aerial imagery. The database of Readar is updated several times per year. The database covers the entire Netherlands, resulting in a substantial portion of connected, terraced housing which is common in the Netherlands. Similarly to the Bing imagery and OpenStreetMap data, there is a temporal gap between the imagery and the building database. In this case the temporal gap is between one and two years, which is significantly lower compared to the five to seven years of the other dataset.

Similarly to the Bing Maps data source, the choice is made to use the latest version of the building database. In this case, it is caused by the addition of the manually mapped buildings. The manually mapped buildings have been added for a number of municipalities in the weeks or months after the aerial imagery was created. Reducing the temporal gap by selecting a historical version of the Readar database close to the creation date of the aerial imagery would exclude these manually mapped buildings. The assumption is made that including the manually mapped buildings is more beneficial to the overall datasource quality than reducing the temporal mismatch, leading to the choice of using the latest version of the Readar building database.

4.4.2 Parallax effect

The Readar imagery is non-orthorectified aerial imagery obtained from cameras attached to aeroplanes. Non-orthorectified imagery indicates that no post processing has been applied on the imagery to correct for optical distortions caused by the perspective and view angle of the camera. One of these distortions is known as the parallax effect. Figure 4.6 shows the parallax effect on a sample image from the Readar imagery. The parallax effect causes a misalignment issue between the optical building outline and the ground truth label. The ground truth label accurately describes the location of the base of the building. The displacement of the roof and the optical size of the facade scale linearly to the height of the building.

4.5 CONCLUSION

There are three data sources each used for their own merits: The crowdAI datasource is valuable because of its size and excellent image and label quality. The combination of Bing Maps imagery and OpenStreetMap labels is theoretically available worldwide, allowing us to create datasets for any region in the world. In practice, the availability and quality of both the imagery and the OpenStreetMap labels very between regions. The Readar datasource is large although limited in geographical scope. Its aerial imagery has the highest resolution and the ground truth is generally of good quality.

5 MASK R-CNN

5.1 CHAPTER OVERVIEW

The Mask R-CNN architecture has been identified as one of the most promising architectures for the object detection task and has the added benefit that it simultaneously performs the more ambitious instance segmentation task. This chapter contains details on the implementation of the Mask R-CNN architecture for this internship. A transfer learning experiment without retraining is set up where a model is trained on crowdAI's Mapping Challenge dataset and tested on different locations. These tests are used to infer the robustness of Mask R-CNN with respect to geographical location, imagery resolution and building geometry. Details about the technical environment can be found in Appendix A.2.

5.2 IMPLEMENTATION

The open-source implementation of the Mask R-CNN architecture published by Matterport is used [1]. The model is trained on the crowdAI train set. Initially, the model was run with the default hyperparameter settings.

5.2.1 Preprocessing

The model is trained on the crowdAI dataset, using the crowdAI train set to train on and the crowdAI validation set to validate the model on. The images are 300 by 300 pixels, but the Mask R-CNN architecture includes a ResNet backbone which requires that the input image is a multitude of $2^5 = 32$ pixels as explained in Section 3.9.1. Therefore reflection-padding is applied on the images to increase the dimensions from 300 by 300 to 320 by 320 pixels.

5.2.2 Initial results on crowdAI Mapping Challenge dataset

After training the model, the resulting model was inferenced with default hyperparameter settings on the crowdAI validation set as no ground truth labels were provided for the test set of the crowdAI dataset. A visual sample of the predictions is provided in Figure 5.1, the quantitative results are shown in Table 5.1, following the widely-used Common Objects in Context (COCO) evaluation format [41]. According to the format of COCO, objects smaller than 32^2 pixels are considered small, objects between 32^2 pixels and 96^2 pixels fall into the medium sized category and objects above 96^2 pixels are considered to be large. The Average Precision (AP) and Average Recall (AR) are calculated at certain Intersection over Union thresholds to determine the amount of overlap required between the prediction and the ground truth label to classify the prediction as a successful positive detection.

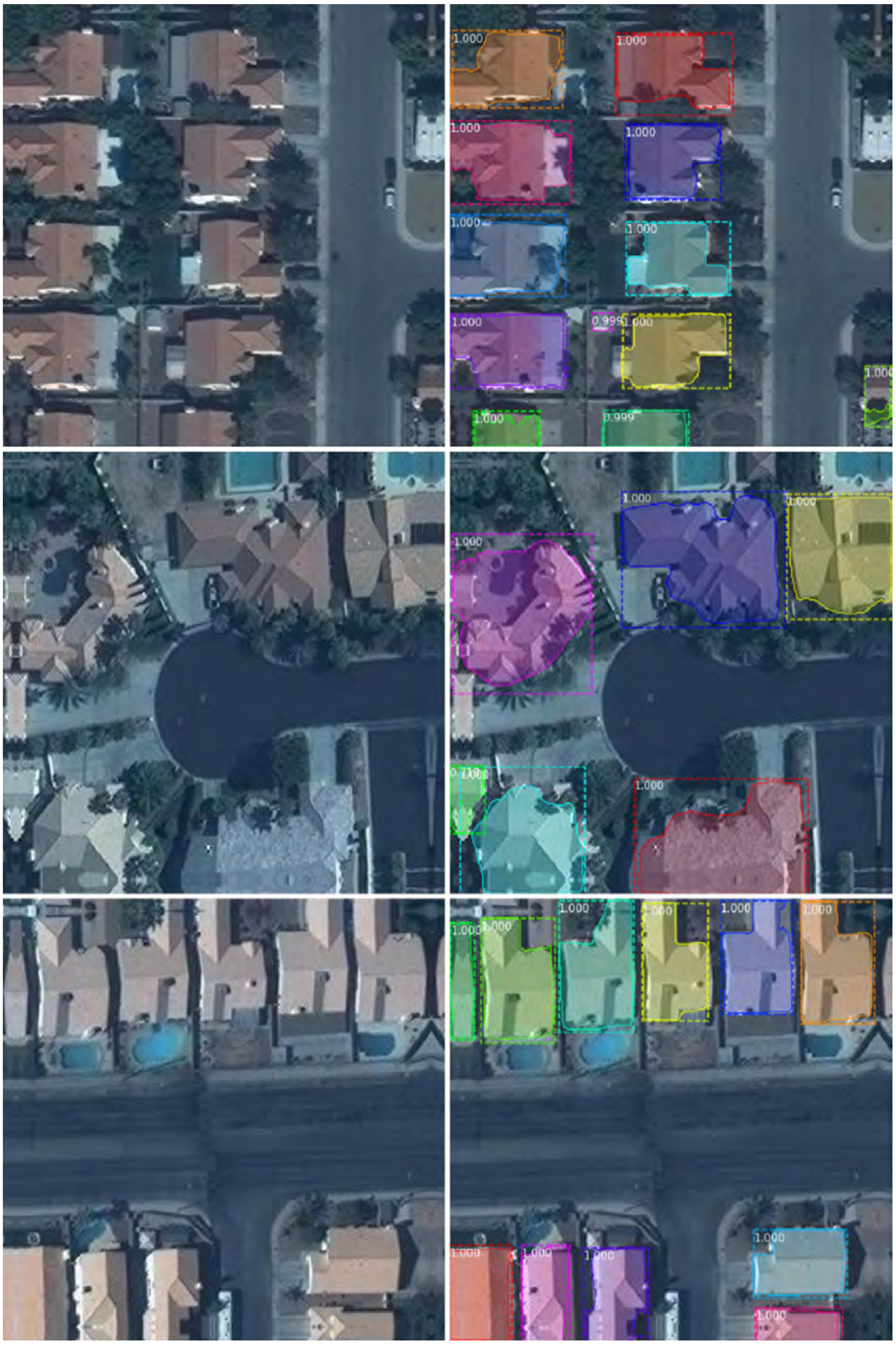


Figure 5.1: Left: Sample images from the crowdAI test set, with reflection-padding applied. Right: Predictions of Mask R-CNN using default parameters. The dotted line visualizes the bounding box, the area within the solid line is the predicted pixel mask and the number represents the certainty of the prediction.

Initial results on the crowdAI validation set

Performance metric	Area	Score
AP @IoU=0.50:0.95	all	0.395
AP @IoU=0.50	all	0.680
AP @IoU=0.75	all	0.432
AP @IoU=0.50:0.95	small	0.135
AP @IoU=0.50:0.95	medium	0.551
AP @IoU=0.50:0.95	large	0.508
AR @IoU=0.50:0.95	all	0.458
AR @IoU=0.50:0.95	small	0.185
AR @IoU=0.50:0.95	medium	0.629
AR @IoU=0.50:0.95	large	0.653

Table 5.1: Evaluation scores of Mask R-CNN model with default parameters applied on the crowdAI Mapping Challenge validation set.

An average AP or average AR is calculated by averaging scores obtained at multiple thresholds for the Intersection over Union. COCO reports AP and AR @IoU=0.50:0.95, which is the average AP or AR measured starting at IoU = 0.5 increasing with steps of 0.05 to 0.95.

5.2.3 Hyperparameter optimization

The complete list of hyperparameter settings can be found in Appendix A.3. For some of the hyperparameters a basic grid search was performed, not to just find the optimal values for fitting this dataset, but also to obtain an indication of the influence of these hyperparameters on the model's performance. Most hyperparameters use the default settings, but those that deviate from the default values are described here. The RPN anchor scales hyperparameter is used to describe the expected range of building surface sizes in pixels whilst the RPN anchor ratios hyperparameter describe the expected aspect ratios.

RPN anchor scales

This appeared to be the most influential hyperparameter as changing this hyperparameter yielded the largest performance increase. This hyperparameter determines the sizes of the anchor boxes generated by the Region Proposal Network. The default values were 32, 64, 128, 256 and 512 indicating that the smallest square anchor boxes are 32 by 32 pixels and the largest square anchor boxes are 512 by 512 pixels large. With the default settings, the smallest anchor boxes contain an area of $32^2 = 1024$ pixels or $9.6^2 = 92.16$ square meters given imagery with a 30 cm resolution.

The bounding boxes obtain a score based on the average activation value of all pixels within the bounding box and only the highest scoring anchor boxes are kept. With a sigmoid activation function in the final output layer, the activation value of a pixel directly corresponds which the probability of being part of a building. This means that anchor boxes which fit tightly around a predicted building generally have the highest scores. Anchor boxes which only partly overlap a predicted building and anchor boxes with a wide fit around a building receive lower scores due to having a lower average activation value.

The average building size in the crowdAI dataset is 2344 pixels with over 20 percent of all buildings being smaller than 1024 pixels. Buildings substantially smaller than 1024 pixels are overlaid by wide fitting anchor boxes, resulting in relatively low scores for these anchor boxes. This is problematic because all but the highest scoring anchor boxes are discarded. Another issue with the default RPN anchor scales is that the largest anchor boxes are 512 by 512 pixels. The entire image is only 320 by 320 pixels large, rendering this anchor scale useless. For those two reasons the RPN anchor scales were adjusted to 8, 16, 32, 64, 128 and 256. Of all hyperparameter adjustments, this change had the largest performance on the main performance metric. The performance metric AP@IoU0.5 improved by ~ 0.10 from ~ 0.68 to ~ 0.78. This roughly equates an increase in correct predictions by 10 percentage points.

RPN anchor ratios
The RPN anchor ratios are meant to describe the aspect ratios of the objects to detect. The anchor ratio is defined as the length of the anchor box divided by its height. A value of one results in a square anchor box; values above 1 yield a rectangle with a vertical orientation whereas values below 1 yield rectangles with a horizontal orientation. At each anchor point and for each anchor scale, three anchor boxes are generated following the anchor ratios of this hyperparameter. Evidently buildings in general do not have any predefined shape or aspect ratio which makes it challenging to find suitable values are applicable worldwide. The default values of the RPN anchor ratios are $[0.5, 1, 2]$. The options $[0.6, 1, 1.67]$, $[0.4, 1, 2.5]$, $[0.33, 1, 3]$ and $[0.25, 1, 4]$ have been applied on the crowdAI's validation set as well but did not yield improvements over the default values. For the tested options the performance in terms of AP@IoU0.5 decreased by 0.012, 0.003, 0.014 and 0.037 respectively.

Batch size
A batch size of 5 is used for training. Out-of-memory errors occurred at larger batch sizes. No batch normalization was used due to the small batch size.

Learning rate
A constant, relatively small learning rate η of 10e-6 is used to train. The default training schedule starts with a learning rate of 0.02, which is reduced by a factor 10 after 40 and 120 epochs. Training is slower compared to the default training schedule but when using a small batch size the training process is more stable with lower learning rates.

L2 regularization / Weight decay
In this case, using Stochastic Gradient Descent in combination with momentum, Weight decay is the exact same thing as L2 regularization described in Section 3.6.4. L2 regularization is used to regularize the weights of the model and prevent overfitting. The default value of weight decay parameter λ was increased from 0.0001 to 0.1 with the intention to improve the generalizability of the model and increasing its base performance on other datasets. The value of 0.1 was chosen because for that value the L2 norm constitutes roughly 7 to 8 percent of the total loss, hopefully giving it a significant but not dominant impact on the training process. Changing this hyperparameter did not seem to affect the performance on the crowdAI dataset during training, obtaining similar loss values (excluding the L2 loss) compared to using the default value of 0.0001.

Mean pixel value
Machine Learning algorithms tend to perform better when the input channels are normalized and zero-centered [33]. Subtracting the channel-wise mean pixel value is one way of zero-centering the input data. The mean value of each colour channel of the training set is calculated to obtain the most accurate approximation of the underlying data without leaking information from the validation or test set into the training process. The exact effect on performance of the model caused by zero-centering is unknown in this case as it was applied from the very first run onwards. Investigation into the exact magnitude of this effect falls outside the scope of this internship.

5.2.4 Filling failed masks

As seen in the upper right image in Figure 5.1, sometimes the mask head of the Mask R-CNN model fails to generate an accurate pixel mask within the bounding box. In the upper image a building is detected in the lower right corner of the image. This building is predicted to be a building with 1.000 certainty and the predicted bounding box seems to be reasonable accurate. However, in the pixel mask a tree that is overlapping with the rooftop is predicted to be the building. For a small shed that is detected to be a building with a 0.995 certainty relatively central in the image, the mask head failed to produce a pixel mask at all.

To address this issue, a post-processing step dealing with this issue was included in the pipeline. Since the bounding boxes are generally well fitting, they are used as a reference to find pixel masks which are smaller than expected. If a pixel mask is 10 times as small as the area covered by the corresponding bounding box, it is considered to be a failed mask. Failed masks are replaced by the area of the bounding box. This is a guaranteed overestimation of the correct pixel mask of the building and therefore the area is shrunk by five percent. In the crowdAI test dataset, 2.7 percent off all pixel masks were replaced. In datasets with Bing Maps as imagery source, this percentage was significantly lower, generally being in the range of 0.2 to 0.5 percent. The cause for this disparity is unknown but falls outside the scope of this internship.

5.2.5 Test Time Augmentation

As an enhancement to the Mask R-CNN model, TTA was implemented. The predictions of eight differing image orientations are combined to obtain the TTA results. The eight unique orientations are obtained by horizontally flipping the image and performing rotations on both the original image and the flipped image. Both images are rotated by 0, 90, 180 and 270 degrees.

The predictions for these eight unique orientations all slightly differ from each other. The intention is that the combining these predictions yield an improved prediction compared to a single prediction based on the original image. When overlapping detections are combined, they are merged. A new certainty score of this merged prediction is determined by summing the certainty scores of each overlapping detection multiplied by their respective size in pixels and dividing this by the number of pixels in the merged prediction. The new certainty score is not bounded in the range of [0, 1] anymore but can exceed 1.

Optimized results on the crowdAI validation set

Performance metric	Area	Score	Improvement
AP @IoU=0.50:0.95	all	0.478	0.083 (+21.0%)
AP @IoU=0.50	all	0.795	0.115 (+16.9%)
AP @IoU=0.75	all	0.545	0.113 (+26.1%)
AP @IoU=0.50:0.95	small	0.236	0.101 (+74.8%)
AP @IoU=0.50:0.95	medium	0.615	0.064 (+11.6%)
AP @IoU=0.50:0.95	large	0.591	0.083 (+16.3%)
AR @IoU=0.50:0.95	all	0.554	0.096 (+21.0%)
AR @IoU=0.50:0.95	small	0.338	0.153 (+82.7%)
AR @IoU=0.50:0.95	medium	0.691	0.062 (+9.9%)
AR @IoU=0.50:0.95	large	0.683	0.030 (+4.6%)

Table 5.2: Evaluation scores of Mask R-CNN model with adjusted hyperparameters applied on the crowdAI Mapping Challenge validation set, compared with the initial results

Results on the crowdAI test set

Metric	Area	Score
AP @IoU=0.50	all	**0.802**
AR @IoU=0.50	all	**0.850**

Table 5.3: Evaluation scores of Mask R-CNN model applied on the crowdAI test set.

5.2.6 Optimized results on crowdAI Mapping Challenge dataset

With the optimized parameters, the model has again been evaluated on the crowdAI validation set. The results are shown and compared with the initial results in Table 5.2. To participate in the crowdAI Mapping Challenge competition, the predictions on the test set had be uploaded to crowdAI's servers where they would be evaluated on Average Precision @IoU=0.50 and Average Recall @IoU=0.50 as performance metrics.

As shown in Table 5.3, this Mask R-CNN model scored 0.802 at Average Precision@IoU=0.50 and 0.850 at Average Recall@IoU=0.50, resulting in a 15th place out of 54 competitors. Worth mentioning is that the model that obtained second place also used Matterports Mask R-CNN implementation [1] scoring 0.937 at the Average Precision@IoU=0.50 metric and 0.959 at the Average Recall@IoU=0.50 metric [61]. This difference in performance can be explained by the underlying incentives: For this internship there is no intention of winning the Mapping Challenge competition but a submission is made merely to obtain the performance of the model on the test set. Secondly, the broader objective of this internship is to create a model that is reusable in other geographical areas. Therefore, it is not of interest to extensively optimize the hyperparameters or make other adaptations to the model in order to obtain a perfect fit to this specific dataset.

5.3 GEOGRAPHICAL ROBUSTNESS

One way to obtain a method for worldwide building detection is the appliance of transfer learning. The basis of transfer learning is an existing model that is trained on an extensive, high-quality dataset for a similar, related task. This model is retrained on a (potentially small) dataset specific to its new task in order to adapt the model to perform well at this new task. In the case of this internship, the crowdAI dataset forms an excellent base due to its size, quality and similarity in task to create a base model for building detection using transfer learning.

To obtain a baseline for the performance of the unaltered model, is is first applied to predict buildings on imagery of different geographical areas around the world without any transfer learning or other adjustments. These locations are:

- **Malawi**: Nsanje
- **Zambia**: Zambezi, Kalabo, Zambia, Mongu
- **Peru**: Equitos
- **The Netherlands**: Amersfoort

The imagery source used is Bing Maps for all locations except Amersfoort in the Netherlands. These locations are selected based on the intended use cases. Amersfoort is selected as a test area for the use case of Readar which has a geographical focus on Western Europe. Both Missing Maps and the 510 team of the NLRC have a worldwide scope. Composing an exhaustive selection of areas representing the global scope is not an option due to practical limitations in terms of available resources and time. Instead, areas of currently active projects within Missing Maps and the 510 data team are selected instead as an approximation of the intended global scope. Areas in Zambia and Peru are selected based on active mapping projects of Missing Maps. Imagery of Malawi is already available within the 510 team originating from a similar Deep Learning project.

5.4 MALAWI

The 510 team of the NLRC has been running projects in the Nsanje region in the south of Malawi and already has imagery from Bing Maps available for this region at a 60 cm resolution, making it a suitable region to test the Mask R-CNN model on. For the Nsanje region there is no proper ground truth due to misalignment issues between the OpenStreetMap data and the Bing Maps imagery. The lack of aligned ground truth labels mean that only a visual inspection can be made to assess the prediction quality.

Since the predictions are used for visual inspection only, a subset of 433 random images of the original 38,046 images is used. To obtain a performance baseline, no changes are made to the model at the inference stage. Only one adjustment has been made: the dimension of the input images has been changed from 320 to 256 pixels, the native dimension of the images from the Bing Maps imagery. Visual inspection of the predictions of the Mask R-CNN model provides a mixed view: There seem to be false positives as a number of detections appear to be shrubbery or trees instead of a building. On the other hand, in images containing buildings such as the third sample of Figure 5.2, most of the detections seem to be buildings.

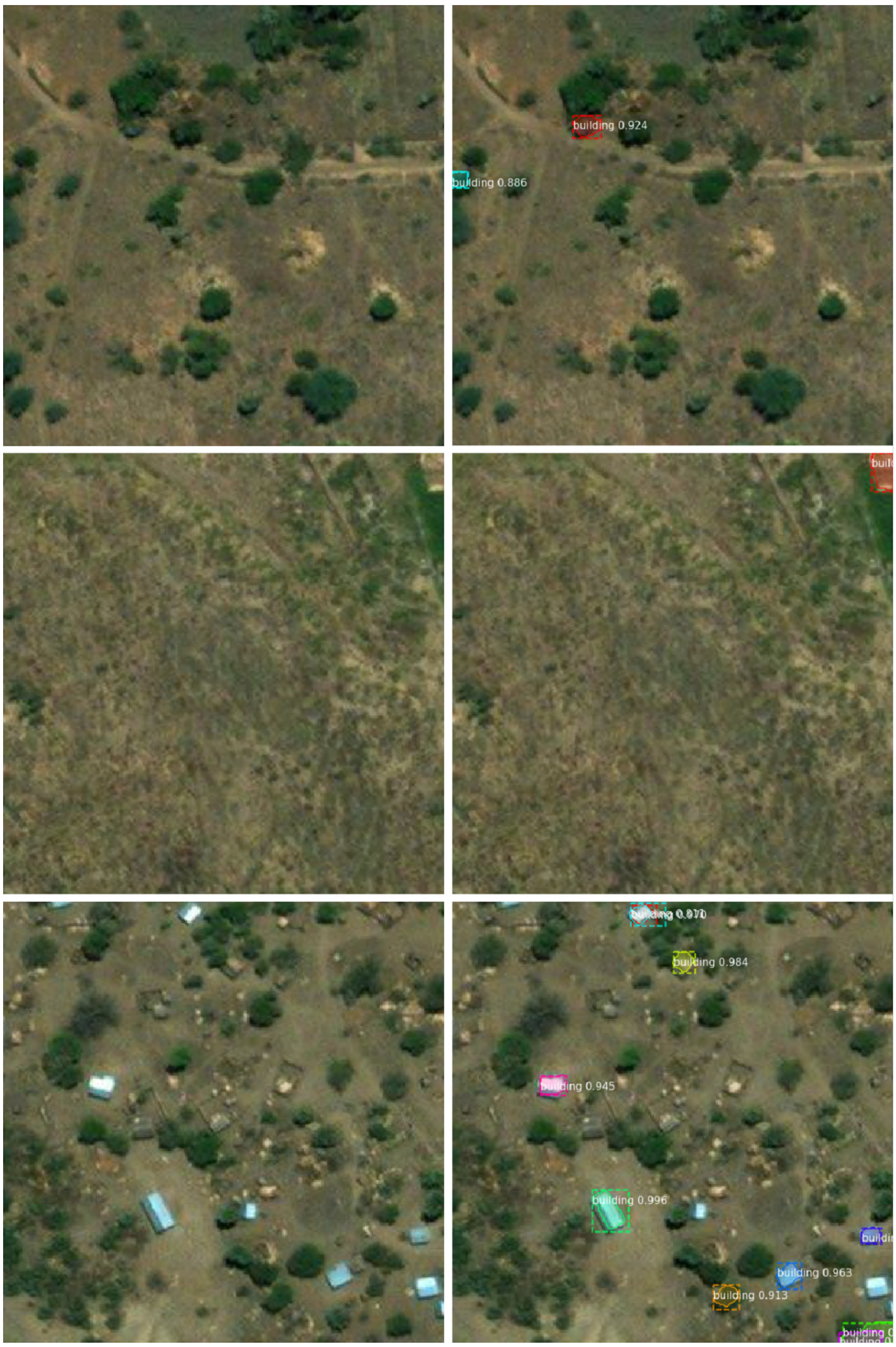


Figure 5.2: Left: Three sample images of 153.6 by 153.6 meters from the 60 cm resolution Bing Maps imagery of the Nsanje region in Malawi. Right: Mask R-CNN predictions on those images. Only predictions with a certainty above 0.5 have been plotted.

One conclusion of the visual inspection on the initial results of 433 images is that larger buildings are more likely to be detected than smaller buildings. Results on the crowdAI dataset showed the importance of having anchor scales for the region proposal network that fit the size distribution of the objects to be detected. With this in mind, the average building size of the misaligned, but properly scaled ground truth labels was calculated to be 47 pixels at 60 cm resolution and 187 pixels at 30 cm resolution. This is a large deviation from the distribution of building sizes the model was trained on, which averaged 2344 pixels.

5.4.1 Image resolution robustness experiment

To solve the discrepancy between the crowdAI building size distribution and Nsanje building size distribution, either the region proposal anchor sizes or the building size distribution itself has to be adjusted. Changing the region proposal anchors would require retraining on data with a similar building size distribution which is unpractical. The more robust solution is to artificially upsample the Bing Maps imagery to ensure the building sizes are similar to that of the crowdAI dataset. The hypothesis is that when the number of pixels is increased by a factor of $\frac{\textit{average crowdAI building size}}{\textit{average Nsanje building size}} = \frac{2344}{187} \approx 12.5$, performance of the model will be optimal for the 30 cm resolution imagery. This means both dimensions should be increased by a factor of $\sqrt{12} \approx 3.5$ from 256 to $256 \cdot 3.5 \approx 900$ pixels. The dimensions of the 60 cm resolution imagery should be increased by a factor of ~ 7 for a minimal discrepancy between the train and test building sizes.

For completeness, predictions are made at various upsample scales. Both dimensions of the satellite imagery are upsampled by a factor of 2, 3, 4, 5, 7 and 10 using bilinear interpolation resulting in an increase in pixels of 2^2, 3^2, 4^2, 5^2, 7^2 and 10^2 respectively.

5.4.2 Image resolution robustness results

The predictions of the Mask R-CNN model on the upsampled imagery are shown in Figure 5.3. A clear trend is shown: A larger upsampling factor results in a larger number of detections. The increase in detections mainly stem from smaller buildings. Imagery that was upsampled by a factor 10 has the most predictions, but also the smallest average predicted building size. One observation of the imagery is that there are many buildings with an area below 4 square meters, which might serve as chicken pens or tool sheds. These buildings are likely uninhabited and for that reason not of interest to the Red Cross. The average building size obviously increases when those small buildings are excluded, requiring a lower upsampling ratio. Visual inspection of the predictions indicates that an upsampling factor of 5 yields the best results on imagery of 60 cm resolution. An upsampling factor of 2.5 is used for imagery on 30 cm resolution. Some results are shown in Figure 5.4, additional visualizations of predictions can be found in Appendix B.2.

No independent, unbiased assessment could be made on the robustness of the model with respect to image resolution due to the inevitable influence of building sizes and anchor scales hyperparameter. This experiment mainly exposed the interaction between resolution and the region proposal network of Mask R-CNN. Particularly the influence of the anchor box scales hyperparameter is highlighted.

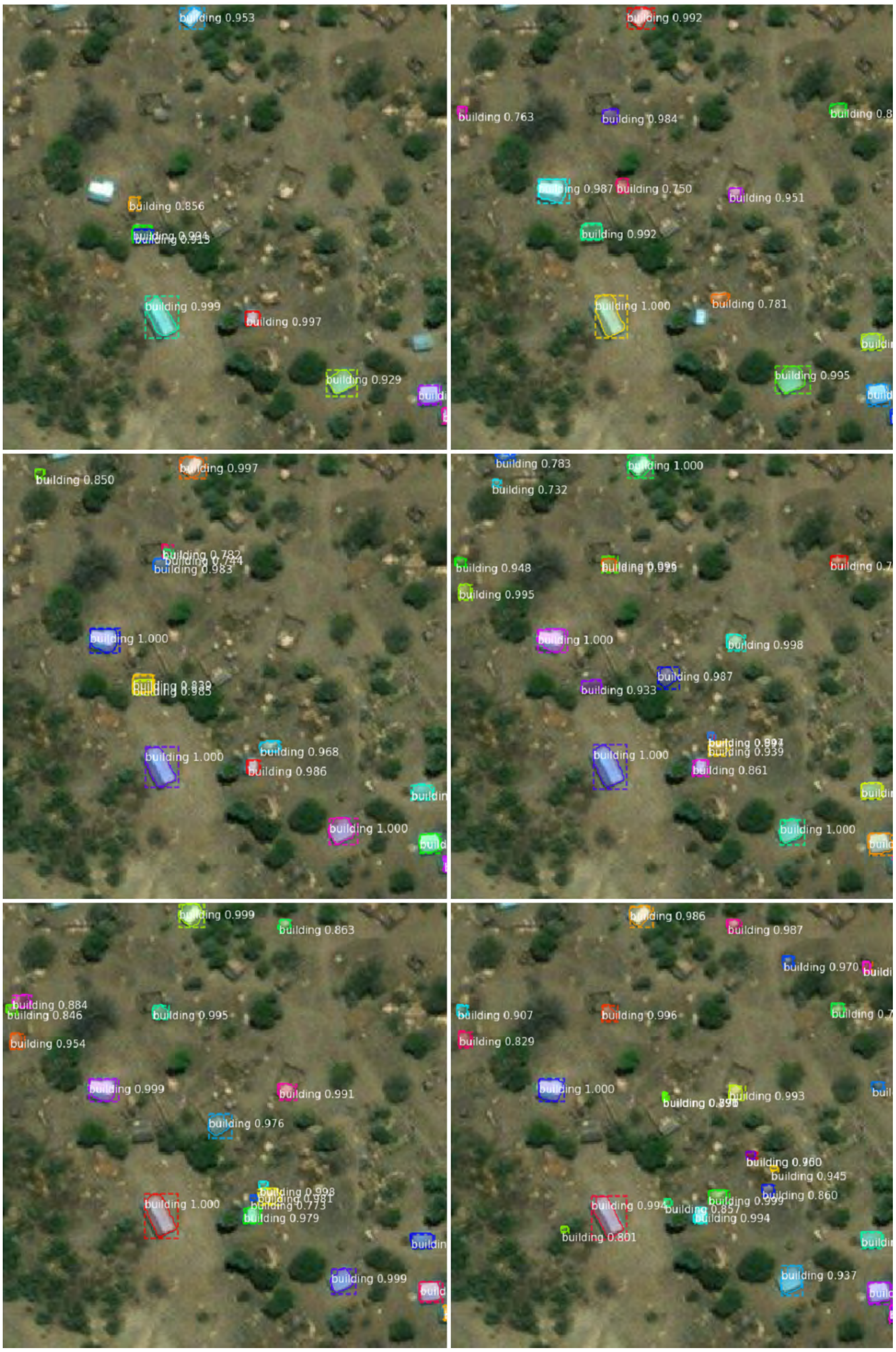


Figure 5.3: Mask R-CNN predictions on different upsampling scales. From left to right and top to bottom the upsampling ratios are 2, 3, 4, 5, 7 and 10 respectively.

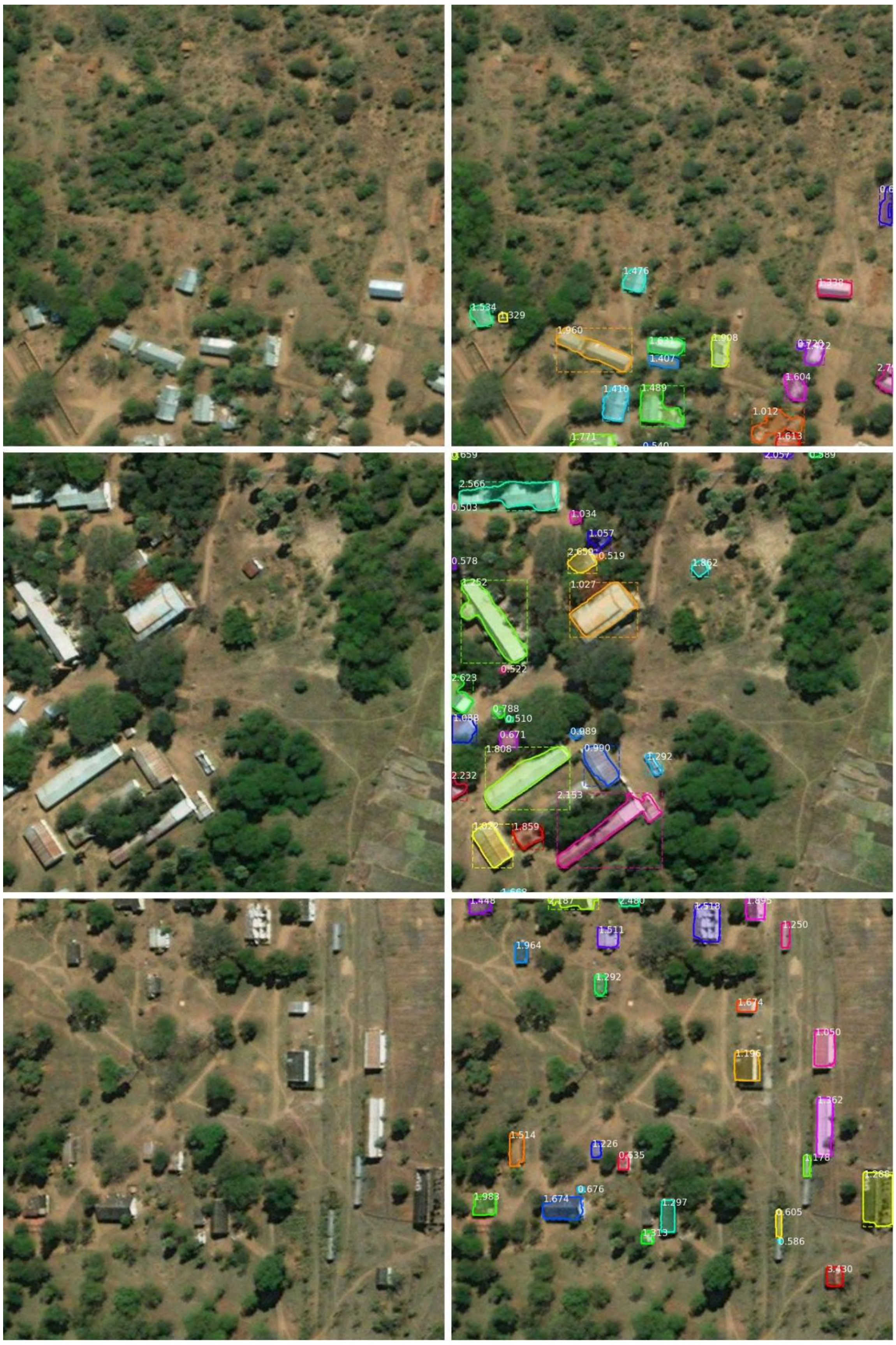


Figure 5.4: Left: Samples of 230.4 by 230.4 meters from imagery of Nsanje at 30 cm resolution. Right: TTA predictions of Mask R-CNN on imagery upsampled by a factor of 2.5.

The first conclusion of this showcase of Mask R-CNN on Malawi is that the model seems to handle the geographical differences between crowdAI and Malawi relatively well. The second conclusion is that the Mask R-CNN architecture itself is not robust with respect to resolution, or more accurately, its interaction with object size. However, by resizing the image based on the expected object size, a better fit can be achieved at inference time. Artificially increasing the resolution slows down the inference speed but does not require retraining. Issues arise when the resolution of the test imagery should be downsampled to fit the expected object size distribution as downsampling decreases the quality of the imagery.

5.5 ZAMBIA

The North-Western province of Zambia is being mapped for a project within the 510 team. In order to provide an impression of the expected quality of automated building detection, the city of Zambezi has been selected as a test set for the Mask R-CNN model. Bing Maps imagery is available at a 30 cm ground resolution for the city of Zambezi. As a second African location, it is used to show the generalizability of the model and to further evaluate the predictive potential of this model on imagery of the African continent.

The assumption is made that the distribution of the building sizes in Zambezi in Zambia is similar to that of Nsanje in Malawi. Hence, the imagery of Zambezi is upsampled by the same ratios as used for Nsanje. Three sample predictions made by the Mask R-CNN model on imagery from Zambezi are shown in Figure 5.5. Although no quantification of the performance could be made due to a lacking ground truth, Missing Maps project managers evaluated the performance on Zambezi as "promising" for providing an initial impression on building density.

The same routine has been followed for two other cities in Zambia. 30 cm resolution imagery is also available on Bing Maps for the cities of Kalabo and Lusaka to analyze the consistency of the model. Kalabo is relatively similar to Zambezi but Lusaka is a larger and more densely built city. Additional sample imagery and corresponding predictions of Kalabo and Lusaka can be found in Appendix B.2, confirming the consistency of the model.

5.5.1 Geographical robustness experiment

To accurately quantify the performance of the model, a ground truth of decent quality is required. For the previous locations, OpenStreetMap data did not meet those requirements, mainly due to misalignment issues. The Zambian city of Mongu was selected for a comparison between the Mask R-CNN model and human mappers. Mongu was recently manually mapped and extensively checked for errors by experienced mappers to ensure a high quality standard. Not all buildings were mapped based on Bing Maps imagery causing a temporal gap in the dataset as described in Chapter 4. However, all mapped buildings were consistently aligned with the Bing Maps imagery during the quality checks. This resulted in a usable ground truth for Mongu to quantify the quality of the predictions.

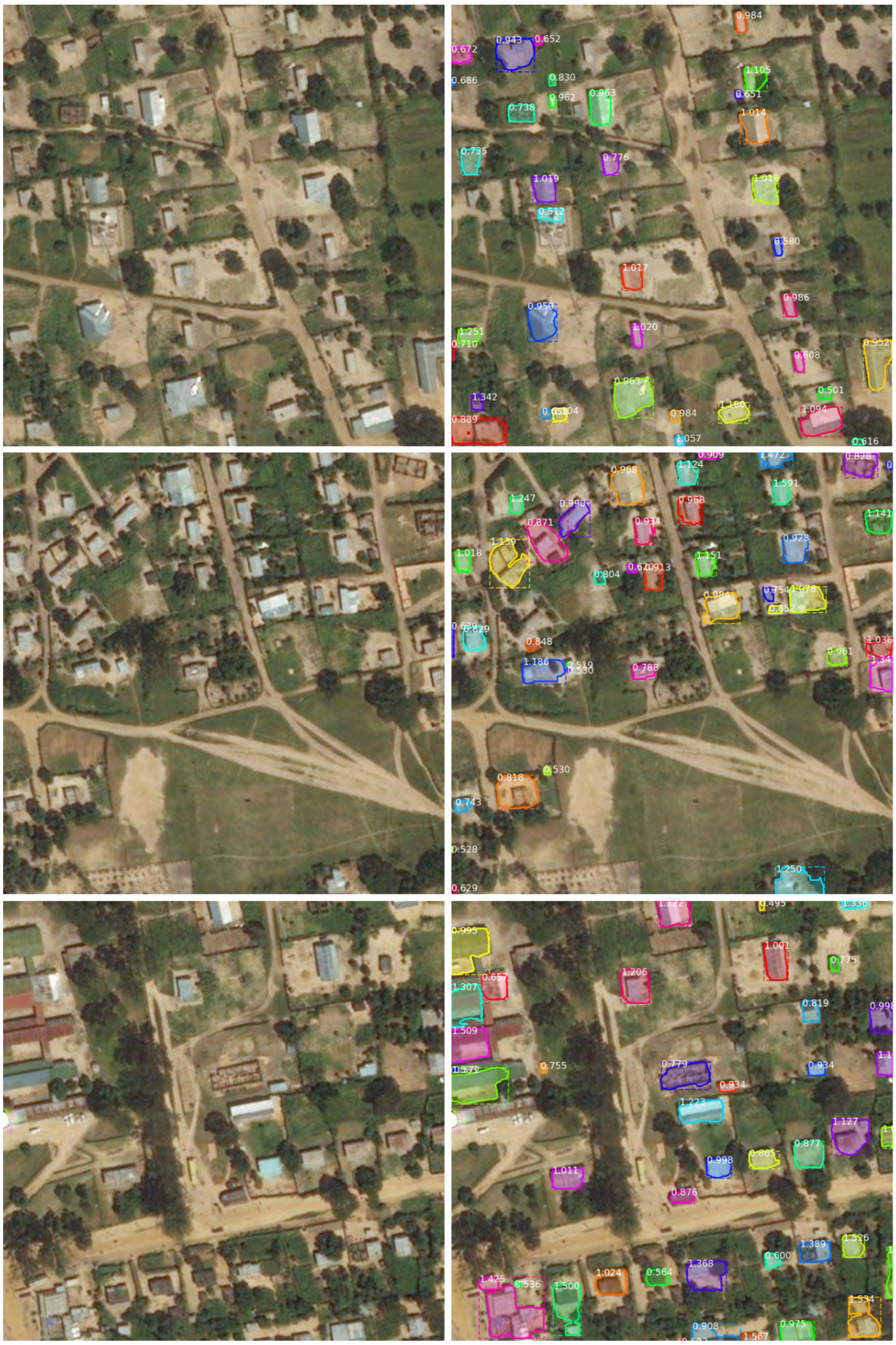


Figure 5.5: Left: Samples of 230.4 by 230.4 meters from imagery of Zambezi at 30 cm resolution. Right: TTA predictions of Mask R-CNN on imagery upsampled by a factor of 2.5.

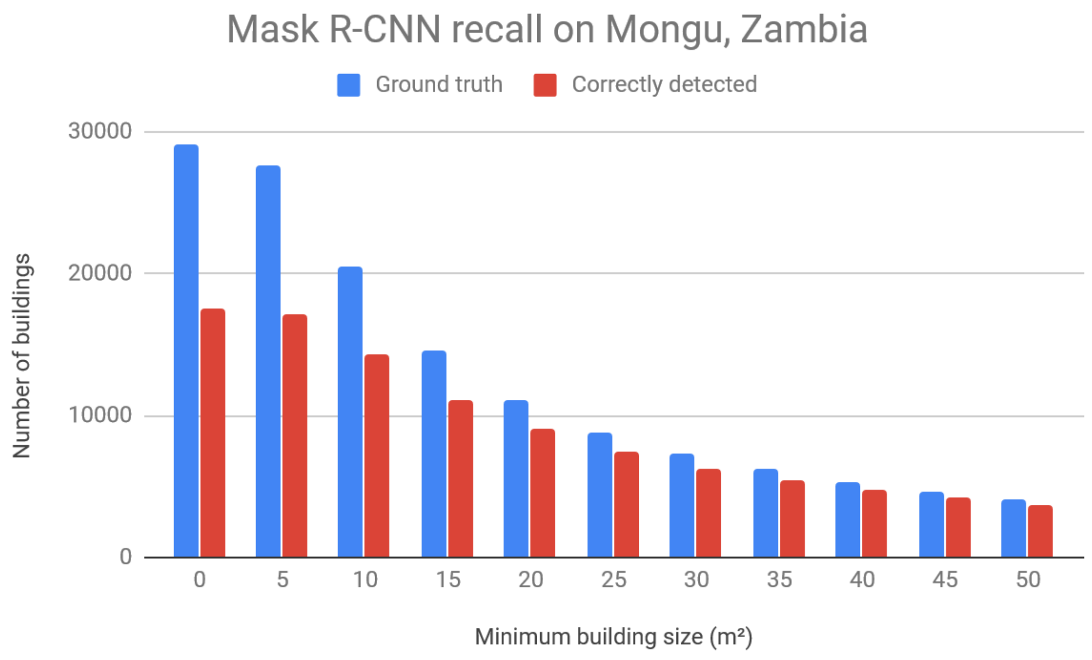


Figure 5.6: Breakdown of Mask R-CNN recall at 0.9806 precision based on minimum building size for predictions on Mongu, Zambia.

5.5.2 Geographical robustness results

Running the model through 11,273 images combined spanning approximately 67 km^2 took 48 minutes. The main results over the entire dataset of 29,061 buildings in the ground truth and 17,863 predictions are presented in Table 5.4 and Figure 5.6. The model obtained a precision of ~ 0.98 and a recall of ~ 0.60. Visual samples of the satellite imagery, OpenStreetMap ground truth and predictions can be found in Appendix B.3. A breakdown of the recall at various minimum ground truth building sizes in Table 5.4 shows that building size is an important factor for the chance of detection.

These quantitative results on the city of Mongu combined by empirical tests on different locations in Zambia affirm the previously cautious conclusion that the model is able to predict buildings in African locations based on a training set located in North-America.

5.6 PERU

After testing the model on two African countries, a new continent was selected. Managers of the Missing Maps project mentioned that manually mapping Equitos in Peru proved to be challenging. Wondering if the Mask R-CNN model could potentially aid the mapping process, Equitos was selected to see how the model would cope with the dense building style common in Peru. Because the area was being actively mapped by the Missing Maps project, no validated ground truth was available in OpenStreetMap at the time the model was tested on Equitos. Bing Maps satellite imagery was available at a 30 cm resolution for the city of Equitos. The dataset contains 880 tiles of 256 by 256 pixels.

Precision and recall on Mongu, Zambia

Performance metric	Min. building size	Score
Precision	**$0\,m^2$**	**0.9806**
Recall	**$0\,m^2$**	**0.6027**
Recall	$5\,m^2$	0.6208
Recall	$10\,m^2$	0.6962
Recall	$15\,m^2$	0.7581
Recall	$20\,m^2$	0.8165
Recall	$25\,m^2$	0.8458
Recall	$30\,m^2$	0.8616
Recall	$35\,m^2$	0.8770
Recall	$40\,m^2$	0.8945
Recall	$45\,m^2$	0.9034
Recall	$50\,m^2$	0.9171

Table 5.4: Performance metrics of the Mask R-CNN model on Mongu, Zambia.

Although the Mask R-CNN model attempts to perform the instance segmentation task mentioned in Section 1.5.2, building rooftops apparently often lack distinguishability. In these cases it causes the model to fail at the instance segmentation task. Instead, it will combine multiple connected buildings into one prediction. Looking how buildings are densely grouped together in Equitos using the same rooftop materials, this leads to the assumption that the expected average building size is large. As a result, no upsampling is performed on the imagery of Equitos and the native resolution is used instead. A visualization of the predictions seems to indicate a high precision but relatively low recall. Three sample predictions made by the Mask R-CNN model on imagery from Equitos are shown in Figure 5.7. Suspicions are that the low recall is a result of an aggressive removal of overlapping bounding boxes by the NMS. The potential issue of aggressive removal of overlapping bounding boxes seems to be especially prevalent in the images where the streets and buildings have a diagonal orientation.

5.6.1 Building geometry robustness experiment

For buildings with a diagonal orientation, a tight-fitting bounding box has to be relatively large compared to the building itself. This bounding box might overlap with bounding boxes of buildings in close proximity. When the overlap between bounding boxes measured by the Intersection over Union exceeds a predefined threshold, the NMS algorithm discards all predictions except the one with the maximum certainty score with the intention of removing duplicate predictions.

In Mask R-CNN, a default value for the non-maximum suppression threshold of 0.7 is used. By increasing this threshold, a larger overlap between bounding boxes is allowed, theoretically resulting in a larger number of partly-overlapping detections. Adjusting this threshold does not require retraining of the model as it only influences the prediction phase. To investigate whether the default value of this threshold is limiting the performance of the model, the model is tested again with values of 0.3, 0.5 and 0.8 for the non-maximum suppression threshold.

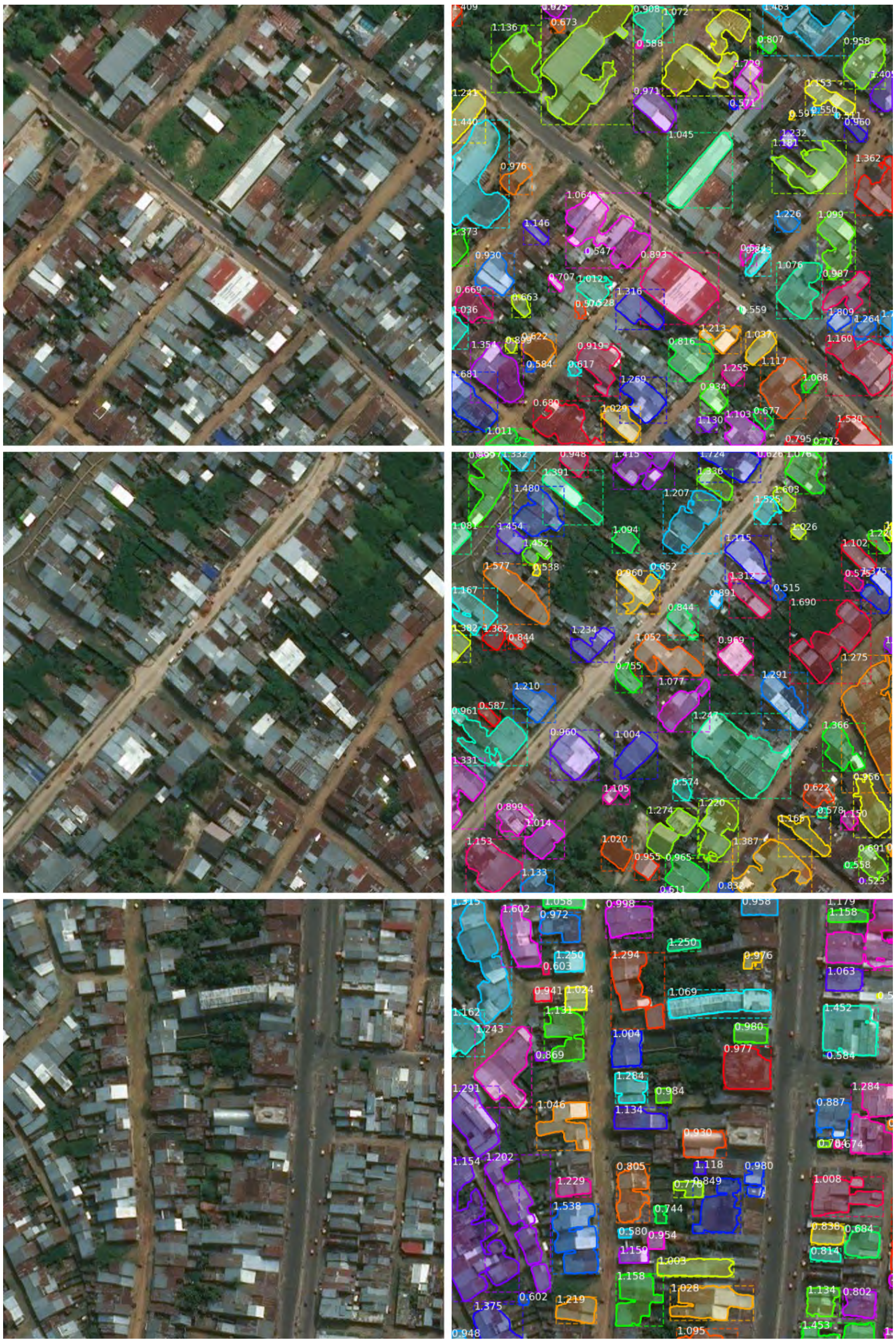


Figure 5.7: Left: Samples of 230.4 by 230.4 meters from imagery of Equitos at 30 cm resolution. Right: TTA predictions of Mask R-CNN on the original imagery.

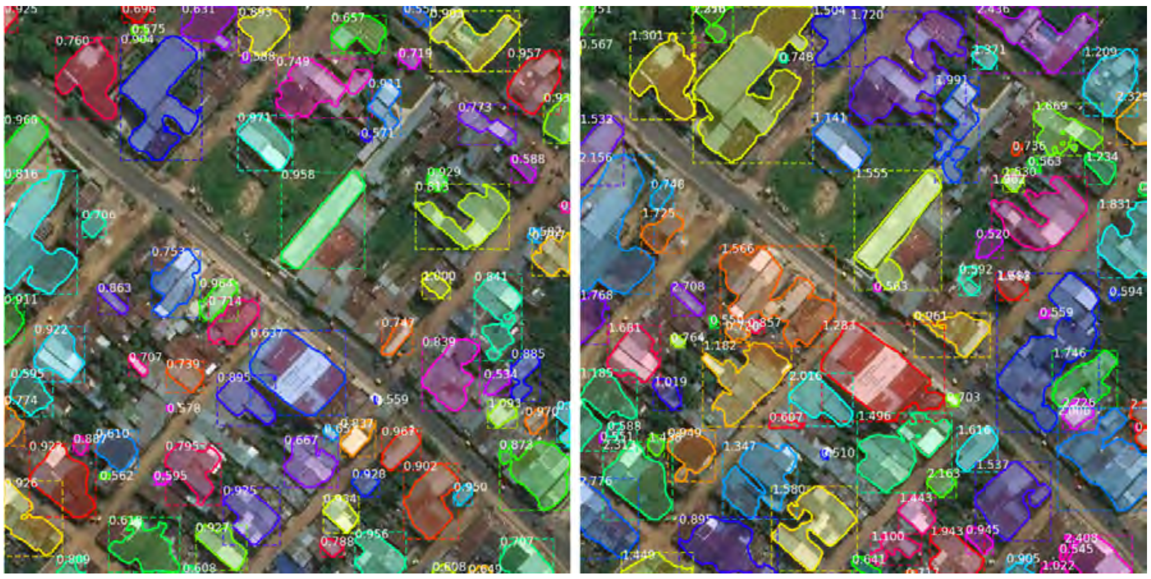


Figure 5.8: Left: TTA predictions with a 0.3 NMS threshold. Right: TTA predictions with a 0.8 NMS threshold.

5.6.2 Building geometry robustness results

A test with a NMS threshold value of 0.9 was aborted prematurely due to excessive prediction file sizes (prediction file sizes exceeding 1 GB per image). One sample comparing predictions using a 0.3 and a 0.8 NMS threshold is shown in Figure 5.8. Predictions made with the adjusted NMS thresholds appear to demonstrate that for densely populated environments the Mask R-CNN model is sensitive to the NMS and minimum detection thresholds. Visual inspection on a subset of 90 images of 768 by 768 pixels seems to indicate an improved recall with similar precision with a higher NMS threshold. The conclusion is that the in-built non-maximum suppression module of Mask R-CNN does not only remove duplicate predictions of a single building, but also suppresses detections of unique buildings in densely built areas due to overlapping bounding boxes. Visual inspection implies that this occurs more frequently in images where buildings and streets have a diagonal orientation.

5.7 THE NETHERLANDS

Amersfoort is a densely populated city in the Netherlands with many connected buildings. Following the behaviour of the model observed on imagery of Equitos, the non-maximum suppression threshold is set at 0.8. The imagery of Amersfoort has a 10 cm resolution and a unsubstantiated assumption is made that the building size distribution of Amersfoort is similar to that of the crowdAI dataset. Therefore, a scaling factor of 0.33 is used to downsample the imagery of Amersfoort to a 30 cm resolution equal to the crowdAI image resolution using bilinear interpolation. Samples of predictions on imagery of the original 10 cm resolution can be found in Appendix B.4, a sample from predictions on the imagery downsampled to 30 cm is shown in Figure 5.9.

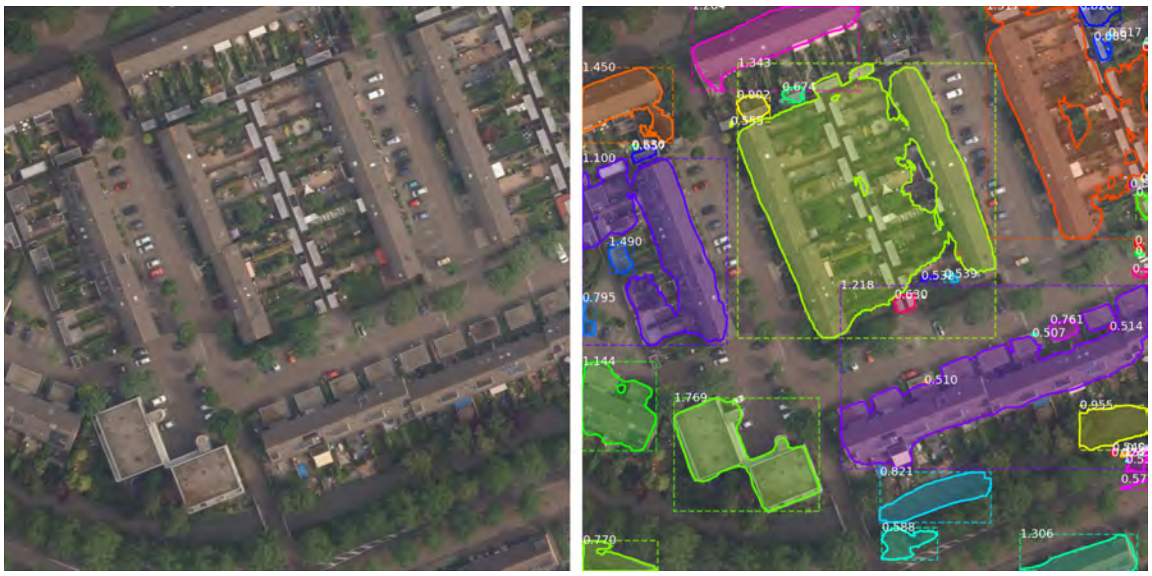


Figure 5.9: Left: Sample of 153.6 by 153.6 meters from imagery of Amersfoort at 10 cm resolution. Right: TTA predictions of Mask R-CNN on the sample image downsampled to 30 cm, plotted on the original 10 cm imagery.

The first observation is that the precision of the model for Amersfoort is lower than for any other previous test set. Not only are more false positives observed, i.e. water or roads classified as building, but the pixel masks of predictions containing a building seem to include a larger percentage of non-building pixels. The nature of the Dutch building style with many buildings having identical rooftops connected to each other might have an undesired effect on the predictions: the larger anchor box sizes obtain relatively high certainty scores in the RPN. The subsequent RoIAlign layer resizes all proposals of the RPN to 56 by 56 pixels (the "mini-mask"). For large proposals, this is a significant reduction in resolution. This might explain the relatively poor pixel-wise performance by the mask head of Mask R-CNN.

5.8 REVIEW OF TRANSFER LEARNING INTENTIONS

The original intention was to determine a performance baseline measuring the geographical robustness of the model without retraining. The underlying thought was that retraining would be required to achieve satisfactory performance. Previous sections of this chapter showcase a model that already performs at a satisfying level without any retraining. Due to varying data quality and misalignment issues discussed in Chapter 4, retraining the model on Bing maps imagery and OpenStreetMap labels could possibly reduce the performance of a model trained on the large and accurate crowdAI's Mapping Challenge dataset.

There is also a more general argument against retraining the model: From a process-oriented perspective the argument is that retraining a model requires a new dataset resembling the area of interest. Finding or creating such a dataset costs time and effort. The effort and time required to obtain a dataset of good quality and sufficient size would significantly delay deployment of the retrained model. This delay negates the benefits of a rapid automated mapping method.

5.9 CONCLUSION

Data alignment issues prevented proper quantification of the results, except for the crowdAI dataset and a manually curated dataset of Mongu in Zambia. Visual inspection of predictions in different scenarios generated enough insights to provide a cautious empirical verdict.

5.9.1 Robustness evaluation

The first impression is that the Mask R-CNN model performs remarkably well without retraining when testing on different geographical locations. This chapter also demonstrates the influence of the region proposal network of the Mask R-CNN architecture. In particular the influence of the anchor scales used to generate anchor boxes is highlighted. In certain cases, the performance of the model suffers when there is a mismatch between the anchor boxes and building geometry. This is mainly noticeable in densely populated areas. For all locations the interaction between the anchor scales setting, imagery resolution and building size distribution seems to be a dominating factor in the performance of the Mask R-CNN architecture.

Empirical verdict on robustness:

- **Mask R-CNN is robust with respect to geographical location**, as a model trained on an North American dataset can successfully detect buildings on imagery of locations in Africa, South America and Europe. On 29,061 buildings in Mongu, Zambia, the model obtained a building-wise precision of ~ 0.98 and a recall of $\sim$ 0.60.
- **Mask R-CNN is not robust with respect to image resolution**, but only because the model is sensitive to the interaction between the anchor scales hyperparameter, image resolution and building size distribution. Due to the influence of this interaction, no unbiased assessment could be made on the image resolution robustness of Mask R-CNN.
- **Mask R-CNN is not robust with respect to building geometry**. Not only is the model sensitive to the building size distribution, but dealing with a large variety in building shapes and orientations also appears to be challenging when the model is limited to three anchor ratios.

5.9.2 Task evaluation

Especially the sensitivity to building geometry is an issue caused by the region proposal network. These sensitivity issues can only be solved by removing the region proposal layer. Removing the region proposal layer has the consequence that no bounding boxes are generated and that the object detection task cannot be performed anymore. Therefore, the conclusion of this chapter is that the object detection task is not the most suitable task to perform in this case.

The image segmentation task does not require bounding boxes but works on a pixel-based level. Performing only the image segmentation task will hopefully yield a model that is more robust with respect to both resolution and building geometry. The U-net has been identified in Section 2.4 as the state-of-the-art architecture for performing image segmentation. In the next chapter, the U-net architecture is implemented to review its performance.

6 RESU-NET

6.1 CHAPTER OVERVIEW

In this chapter an adaptation of the popular U-net architecture is implemented called Residual U-net (ResU-net). The ResU-net architecture is a less complex architecture performing the image segmentation task and contains fewer hyperparameters compared to the Mask R-CNN architecture. The ResU-net architecture is implemented with the intention of mitigating the sensitivity issues of Mask R-CNN with respect to building geometry. These sensitivity issues arise mainly in densely built locations such as The Netherlands. This, combined with the availability with high quality data, is the reason why the performance of the ResU-net will be mainly evaluated on the Netherlands. The idea is that a less complex architecture contains fewer limitations resulting in a more robust model, especially with regards to building size and aspect ratio.

Besides details on the implementation of the ResU-net architecture, this chapter contains experiments to quantify the robustness of the ResU-net architecture with respect to geographical location, image resolution and building geometry.

6.2 IMPLEMENTATION

A ResU-net architecture is a traditional U-net structure in which the encoder of U-net is replaced by a ResNet model excluding the final fully-connected classification layer. For the implementation of the ResU-net architecture the Python library *segmentation_models* [62] is used. The *segmentation_models* library contains multiple architectures and for each architecture a model with weights pretrained on ImageNet as explained in section Section 3.8.3. The initial ResU-net model is trained on Readar's in-house data. In this case, Readar's data is preferred over crowdAI's Mapping Challenge dataset as it allows for a manual selection of suitable datasets and due to a larger presence of densely built environments. Readar's imagery is downsampled from 10 to 30 cm for fair benchmarking purposes. Details about the technical environment can be found in Appendix A.2.

6.2.1 Readar data

Five areas each roughly spanning one Dutch municipality are selected to together form a dataset. These areas are Amersfoort, Arnhem, Breda, Maastricht and Molenwaard which combined contain 16,243 images of 1536 by 1536 pixels at a 10 cm resolution, spanning 383 km^2. The validation set consists of 20 percent of each area and for the test set 10 percent of each area is held out. A dataset of the city of Heerlen consisting of 3436 images is used as an external dataset to test the geographical robustness of the model.

6.2.2 Preprocessing

No preprocessing is used except for subtracting the mean pixel values of each training set to zero-center the input data.

6.2.3 Hyperparameters

Appendix A.4 contains an overview of all hyperparameters used. Most hyperparameters of the ResU-net model are related to its training schedule. There is one architectural hyperparameter determining the number of layers in the ResNet. In this case, experiments are conducted with a ResNet50 containing 50 layers (detailed architecture in Section 3.9.1). The certainty threshold for binarizing the output of the sigmoid activation function of the final layer is also a hyperparameter, set at 0.5. The ResU-net architecture has no predefined cost function so the cost function can also be regarded as a hyperparameter. In this case, the Jaccard loss function is used as described in Section 3.6.2. Stochastic gradient descent with Nesterov momentum is used as optimizer. An initial learning rate η of 0.1 is used with a Nesterov momentum γ of 0.8. If the validation loss does not improve over three subsequent epochs, the learning rate is multiplied by 0.3 and training is stopped when there is no improvement over 10 subsequent epochs.

6.2.4 Train augmentations

Random crops of size 448 by 448 pixels are taken out of the training samples of 1536 by 1536 pixels. The images are cropped to allow a larger batch size. The larger batch size increases the diversity in scenery within one batch, making the batch more representative for the entire data domain which increases performance. Larger batch sizes also increase the effectiveness of batch normalization. Another advantage is that random crops prevents overfitting as the number of unique image samples provided to the model during training greatly increases.

Each cropped image is then horizontally flipped with a 50 percent chance and rotated by 0, 90, 180 or 270 degrees with a 25 percent probability for each option. This orientation augmentation increases the number of unique training samples by a factor of eight and hopefully increases the robustness of the model with regards to object orientation.

6.2.5 Test-time augmentations

The predictions of eight unique image orientations are combined to obtain the TTA results. The eight unique orientations are obtained by horizontally flipping the image and performing rotations on both the original image and the flipped image. Both images are rotated by 0, 90, 180 and 270 degrees. After the predictions of the individual images are binarized, the predictions are rotated and flipped back to the original orientation. The eight resulting pixel masks are combined by averaging them pixel-wise.

6.2.6 Initial results on Readar dataset

The results of the initial test performed at a downsampled resolution of 30 cm are shown in Table 6.1, including test results on an external dataset, the Dutch city of Heerlen. TTA predictions of a sample from the test set are visualized in Figure 6.1.

Results on the Readar dataset at 30 cm

Dataset	IoU	Precision	Recall
Train set	0.795	0.876	0.893
Validation set	0.746	0.849	0.858
Test set	0.752	0.850	0.866
Heerlen	0.664	0.755	0.834

Table 6.1: Pixel-wise evaluation scores of ResU-net model trained and tested on Readar's data with the imagery downsampled to a 30 cm resolution.

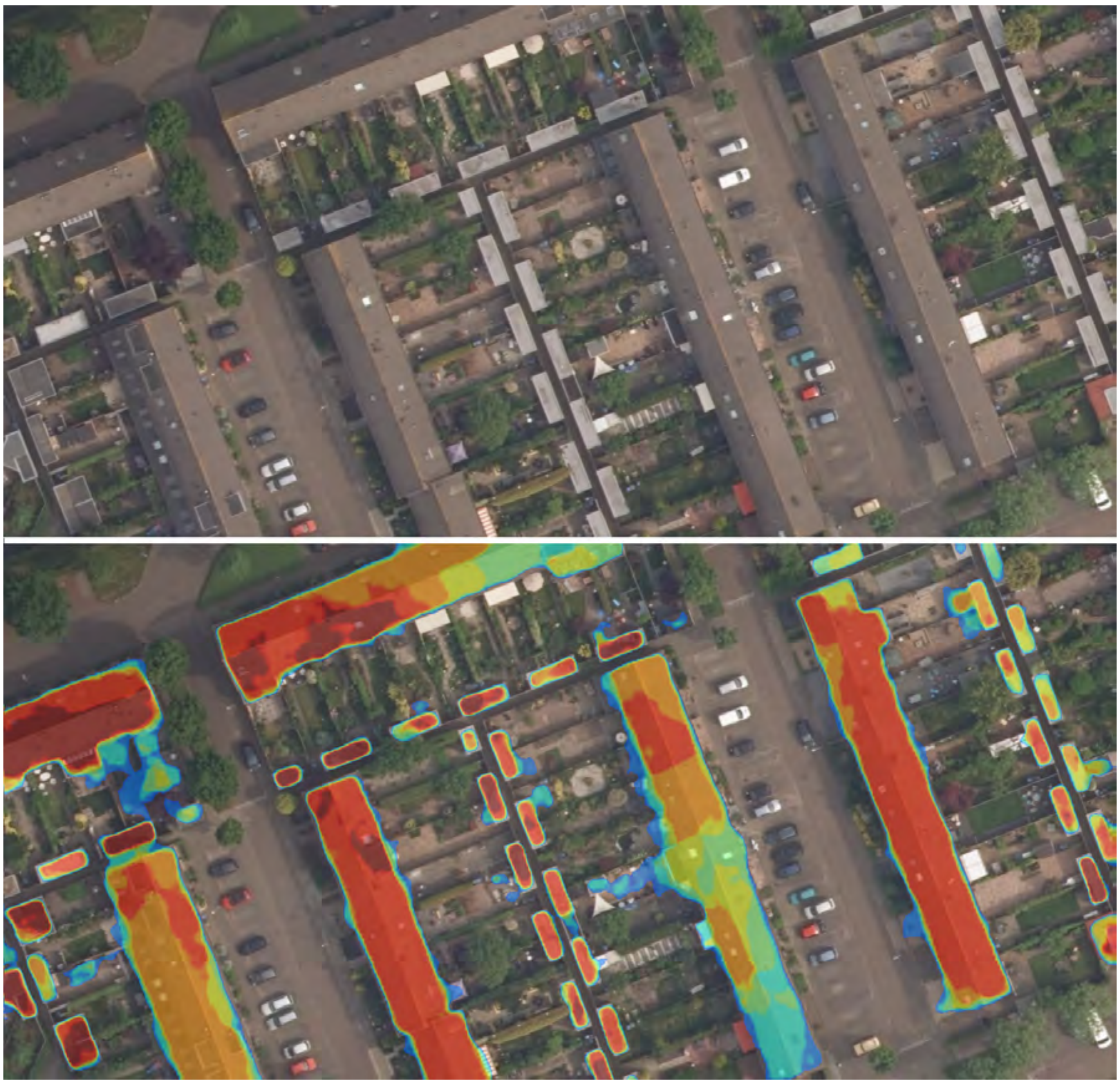


Figure 6.1: Top: Sample image of 153.6 by 72.2 meters from the test set showing the Amersfoort municipality at 10 cm resolution. Bottom: TTA ResU-net predictions inferenced on imagery downsampled to 30 cm resolution but plotted over the original 10 cm imagery. A darker red color indicates a higher predicted probability of the underlying pixel being a building. Only predictions above a 0.2 probability are plotted.

6.3 GEOGRAPHICAL ROBUSTNESS

In this Section, two experiments will be conducted. The first experiment will closely resemble the geographical robustness experiment conducted with Mask R-CNN. The aim of the second experiment is to investigate the added value of inferring a model's transfer learning capabilities during training.

6.3.1 Geographical robustness experiment setup

In order to obtain an indication of the robustness of the ResU-net architecture, it will be trained on the crowdAI training set which is located in the United States and tested on the dataset of the Zambian city of Mongu. In order to obtain a fair comparison between the performance of the Mask R-CNN architecture and the ResU-net architecture, the setup of the geographical robustness experiment described in Section 5.5.1 will be as accurately reproduced but with the ResU-net architecture replacing the Mask R-CNN architecture.

6.3.2 Geographical robustness results

Running the ResU-net model trained on the crowdAI train set through the 11,273 images of the Mongu dataset which span approximately 67 km^2 took 27 minutes. The main results over the entire dataset of 29,061 buildings in the ground truth and 108,119 predictions larger than 32 pixels are presented in Table 6.2 and Figure 6.2. With a precision of 0.2688 and a recall of 0.295 on the Mongu dataset, the quality of the predictions of this model seem to be insufficient for direct application of the model on other data distributions on a satisfactory level.

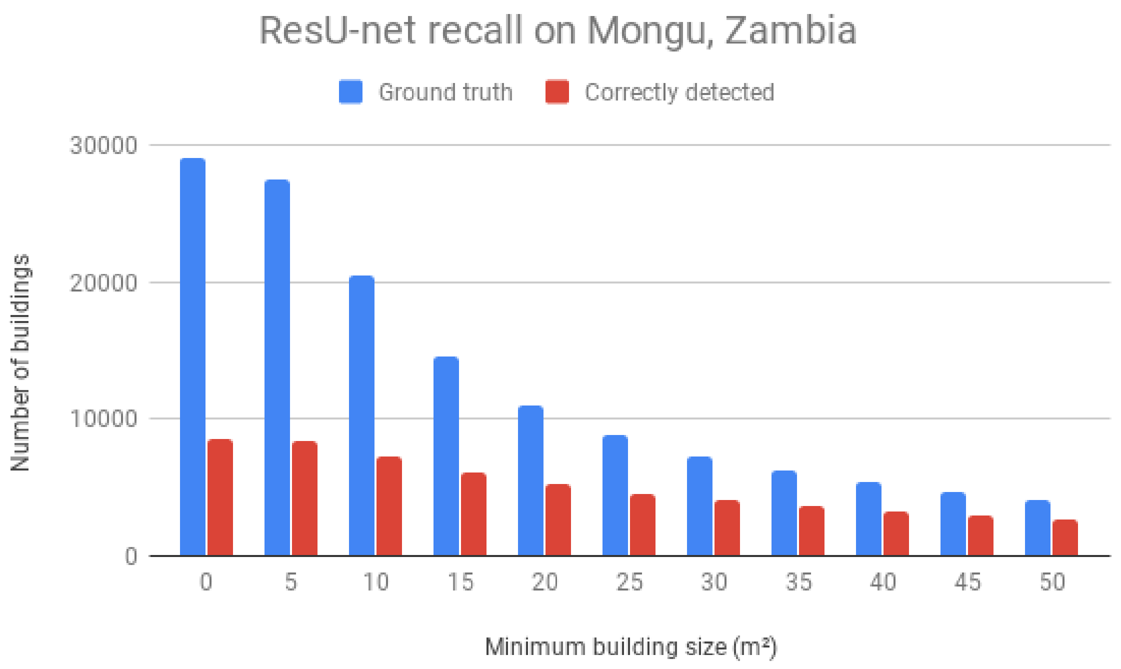


Figure 6.2: Breakdown of ResU-net recall at 0.2688 precision based on minimum building size for predictions on Mongu, Zambia.

Precision and recall on Mongu, Zambia

Performance metric	Min. building size	Score
Precision	**$0\,m^2$**	**0.2688**
Recall	**$0\,m^2$**	**0.2958**
Recall	$5\,m^2$	0.3045
Recall	$10\,m^2$	0.3551
Recall	$15\,m^2$	0.4217
Recall	$20\,m^2$	0.4752
Recall	$25\,m^2$	0.5212
Recall	$30\,m^2$	0.5590
Recall	$35\,m^2$	0.5891
Recall	$40\,m^2$	0.6157
Recall	$45\,m^2$	0.6381
Recall	$50\,m^2$	0.6639

Table 6.2: Performance metrics of the ResU-net model on Mongu, Zambia.

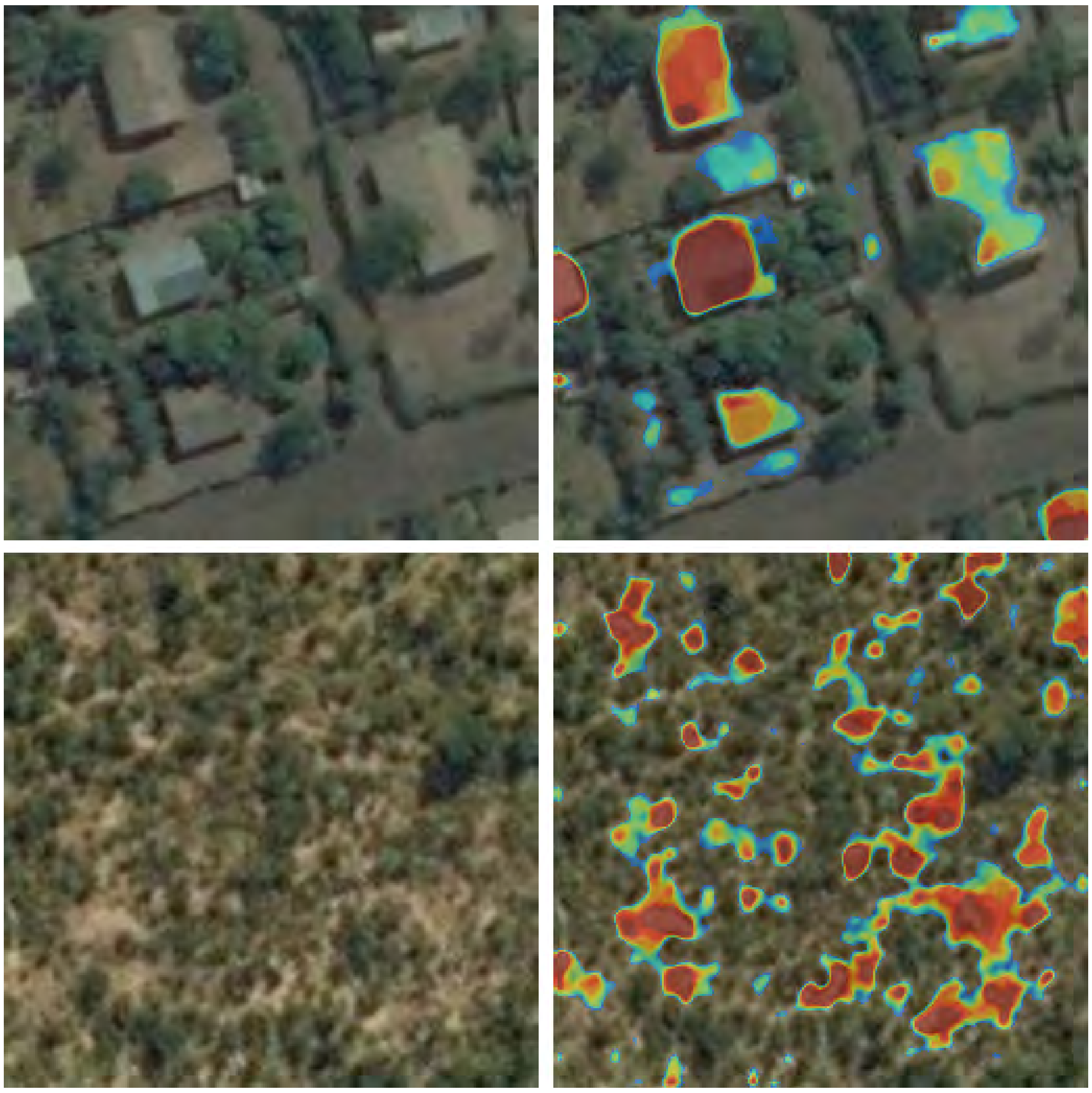


Figure 6.3: Left: Sample imagery of 72.8 by 72.8 meters from Mongu, Zambia. Right: ResU-net TTA predictions.

6.3.3 Transfer learning experiment setup

For this internship, the objective is to obtain a model that is capable of worldwide building detection. Therefore, the following experiment will be conducted with all imagery using a 30 cm resolution which requires downsampling the 10 cm resolution Readar imagery.

The rationale behind the experiment shown in Table 6.3 is that it will exhibit the generalization capabilities when training and validating on one data distribution (crowdAI) but also evaluating every epoch on two additional data distributions (the Netherlands and Zambia). In this experiment, the purpose of these transfer validation sets is to measure the generalizability of the model on instances of the broader distribution of the "building" class during training. Two transfer validation sets are introduced for this experiment. To infer the model performance during training on the Netherlands, the validation dataset from Readar is used. The model checkpoints belonging to the final epoch and the epoch with the lowest loss on the Readar validation set and will be used to test on the Readar test set. For Zambia a similar approach is taken: The Bing Maps LusakaNorth is used for evaluating the transferability of the model during the training process and the final model checkpoint and best performing model checkpoint on LusakaNorth are applied to the Bing Maps LusakaWest set as test set.

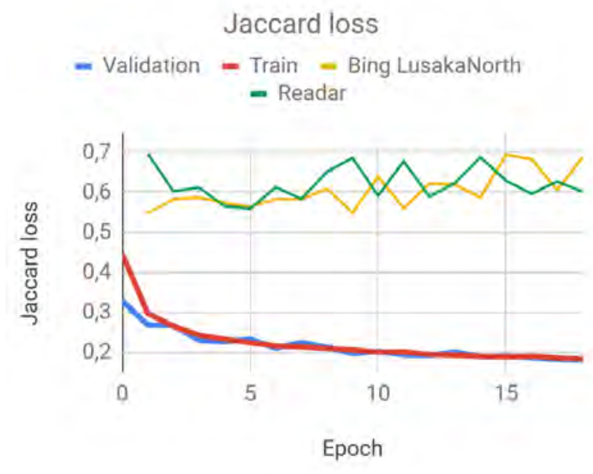
Experimental setup using transfer validation sets

Train set	Validation set	Transfer validation sets	Test sets
crowdAI train	crowdAI validation	Readar validation	Readar test
		LusakaNorth	LusakaWest

Table 6.3: Overview of geographical generalization experiment using ResU-net.

6.3.4 Transfer learning experiment results

The performance of the ResU-net model on the different datasets during training is plotted in Figure 6.4 and Figure 6.5. The first observation from these results is that performance on both transfer validation sets seems volatile. However, for the LusakaNorth transfer validation set a clear trend of increasing loss values and decreasing performance measures can be distilled. Thus of all model checkpoints, the model checkpoint of the first epoch will likely perform best on the LusakaWest test set. For the Readar transfer validation set, the Jaccard loss is the lowest at epoch 5 and the binary cross entropy loss the lowest at epoch 3. The model checkpoint of epoch 4 is selected to apply on the Readar test set because both the binary cross entropy and Jaccard loss values are close to their minima at that epoch. Table 6.4 contains the performance on the two test sets using their respective optimal model checkpoints.



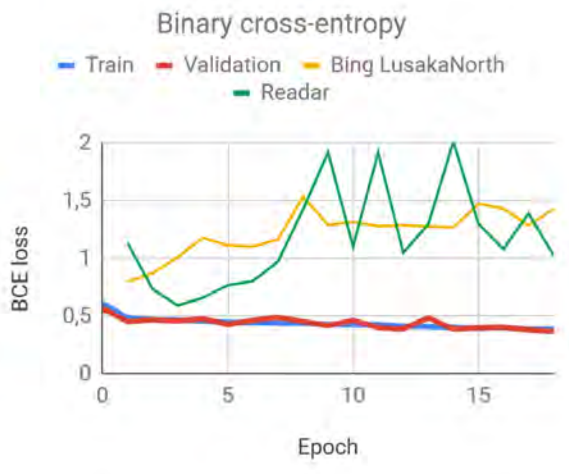


Figure 6.4: Jaccard loss function and binary cross-entropy of ResU-net training session on crowdAI with transfer validation sets from Lusaka and Readar.

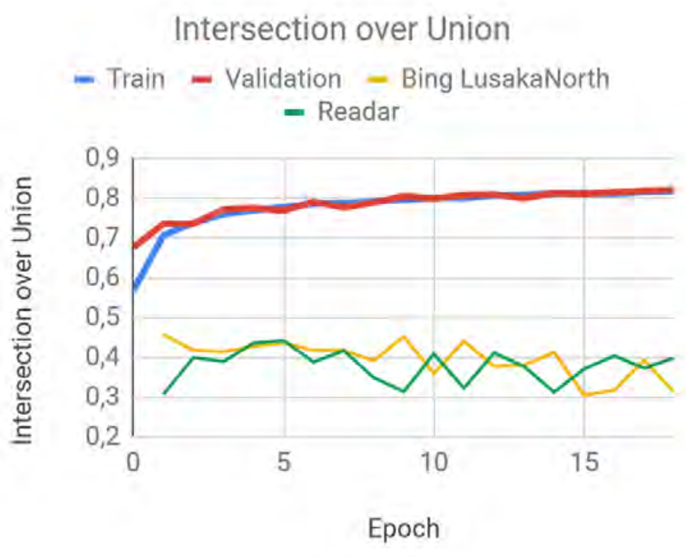


Figure 6.5: Intersection over Union of ResU-net training session on crowdAI with evaluation sets from Lusaka and Readar.

Figure 6.6 shows the predictions of the crowdAI trained model on a sample image of LusakaWest. This experiment indicates that ResU-net model does not seem to achieve the same level of geographical robustness as Mask R-CNN. One potential reason might be that the mask, box and classifier heads of Mask R-CNN receive a fixed amount of proposals to evaluate, even when activation values in the feature map generated by the convolutional backbone are low. This might possibly introduce a bias towards predicting buildings but additional research is required to investigate the true underlying cause.

In addition to testing the geographical robustness of the ResU-net model, this experiment shows that transfer validation sets can be effectively used to monitor the direct transferability of a model to other datasets during the training process. This information allows to make an informed model selection, resulting in a reduction of the Jaccard loss function of $0.676 - 0.558 = 0.118$ for the Lusaka test set and $0.606 - 0.568 = 0.038$ for the Readar test set.

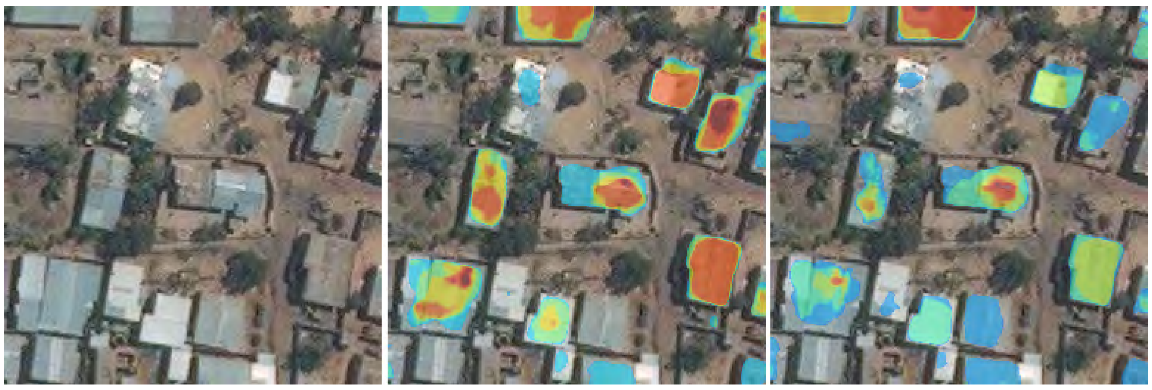


Figure 6.6: Left: Sample of 76.8 by 76.8 meters from Bing imagery of LusakaWest. Centre: TTA predictions of the first epoch of ResU-net trained on the crowdAI dataset. Right: TTA predictions of the last epoch of ResU-net trained on the the crowdAI dataset.

Transfer learning results on the Lusaka and Readar datasets

		Jaccard loss	BCE	IoU	Precision	Recall
Lusaka datasets						
Epoch 1	Transfer validation set	0.547	0.8	0.458	0.638	0.620
	Test set	0.558	0.821	0.440	0.603	0.619
Epoch 18	Transfer validation set	0.688	1.43	0.311	0.872	0.327
	Test set	0.676	1.406	0.331	0.875	0.347
Readar datasets						
Epoch 4	Transfer validation set	0.565	0.660	0.436	0.560	0.671
	Test set	0.568	0.678	0.432	0.551	0.667
Epoch 18	Transfer validation set	0.602	1.029	0.399	0.472	0.730
	Test set	0.606	1.041	0.393	0.458	0.736

Table 6.4: Evaluation scores of ResU-net trained and validated on crowdAI data and evaluated on Bing Maps + OSM data and Readar data.

6.4 RESOLUTION ROBUSTNESS

An experiment is set up to measure the influence of image resolution on the quality of performance of CNNs. Experiments on image resolution using Mask R-CNN are influenced by the Region Proposal Network module of Mask R-CNN. The ResU-net model or any other FCN is more suitable for this experiment as it only contains core elements of a CNN: convolutional and pooling layers.

The ResU-net model is more suitable for this type of experiment compared to Mask R-CNN as the latter uses anchor box sizes as a hyperparameter which inherently influence the results of the Mask R-CNN model. The ResU-net model does not contain any hyperparameter with respect to object size. Hence, the ResU-net model is preferred in this case to safeguard the experimental results from any unwanted influences inferred by the anchor box size hyperparameter of Mask R-CNN.

6.4.1 Image quality experiment

To obtain a baseline for the performance at different ground level resolutions, the following experiment is set up: the model is trained on the Readar dataset twice: Once resized to 30 cm ground resolution using bilinear interpolation and once maintaining the original 10 cm resolution. The experimental environment is kept identical between both experiments except for the resolution of the input data. The intention of this experiment is to measure the difference in performance between imagery of various resolutions.

Altering the resolution has two side effects that must be kept in mind: First, both training and testing the model will theoretically be nine times as slow with the 10 cm resolution experiment compared to the 30 cm resolution experiment. Secondly, the receptive field size of a CNN is measured in pixels. The 10 cm resolution experiment will have a higher level of detail available, but its effective receptive field size in meters will be three times lower in both dimensions, reducing the amount of spatial context available to extract information from.

6.4.2 Image quality results

In Table 6.5 a comparison is made between the predictions of the ResU-net model trained on 10 cm imagery compared to 30 cm imagery. Interestingly, the model trained on 10 cm imagery did not outperform the model trained on 30 cm imagery based on these performance metrics. The expectation was that the 10 cm imagery would yield more accurate predictions. A visualisation of the predictions provides clues to the underlying reasons for this lack of improvement. Figure 6.7 shows the comparison in performance on a close-up image containing some common residential buildings and small sheds. As expected, the 10 cm model shows a strongly improved recall for sheds and smaller buildings. The predictions of the 10 cm model also trace the building outlines more closely and seem to contain fewer false positives.

Results on the Readar dataset at 10 cm

Dataset	IoU	Precision	Recall
Train set (10cm)	0.803 (+1.0%)	0.876 (+0.0%)	0.904 (+1.2%)
Validation set (10cm)	0.728 (-2.4%)	0.830 (-2.2%)	0.850 (-0.9%)
Test set (10cm)	0.722 (-3.9%)	0.809 (-4.8%)	0.859 (-0.8%)
Heerlen (10cm)	0.662 (-0.3%)	0.755 (+0.0%)	0.830 (-0.5%)

Table 6.5: Evaluation scores of ResU-net model trained, validated and tested on Readar's data using the native 10 cm resolution imagery, compared to the results on imagery downsampled to 30 cm.

Figure 6.8 explains why the pixel-wise performance metrics do not show an improvement over the 30 cm model: The 10 cm performs worse on the larger buildings. The certainty of the predictions on the flats is slightly lower on average and it misses large parts of one large building in the lower right corner. Since pixel-wise performance metrics are used, the influence of a building on the performance metrics directly correspond with its size. Missing one large building can negate the positive effect of correctly detecting a multitude of small sheds, which is exactly what seems to be happening. In line with expectations, the conclusion is that the 10 cm model outperforms the 30 cm model on detecting smaller buildings whereas the 30cm model performs better at detecting the entirety of larger buildings.

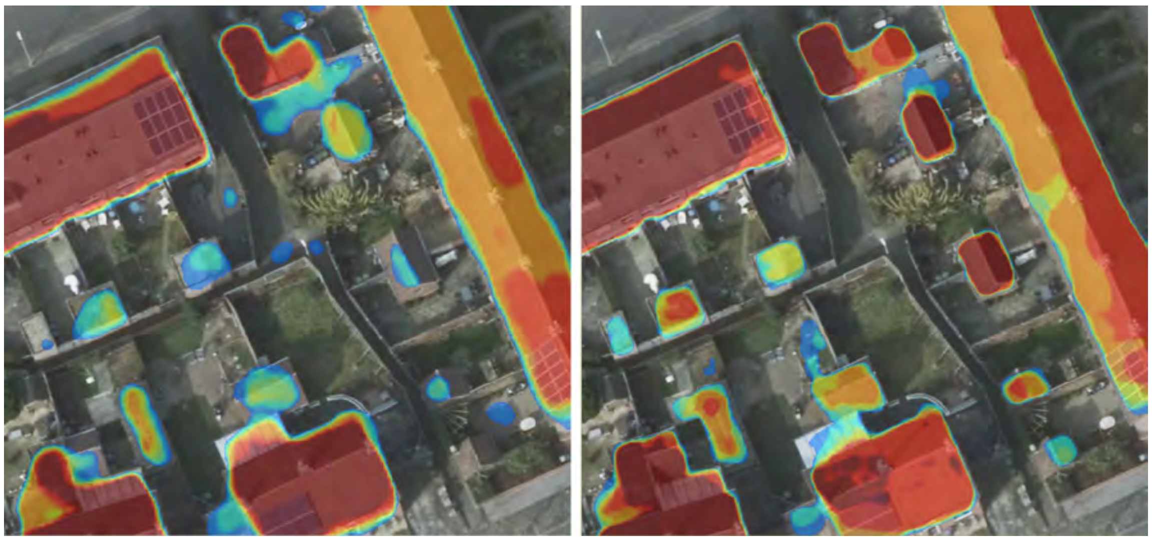


Figure 6.7: Left: A 88.6 by 88.6 meter 10 cm resolution image of Heerlen containing predictions of a model trained and tested on 30 cm imagery. Right: Predictions of a model trained and tested on 10 cm imagery.

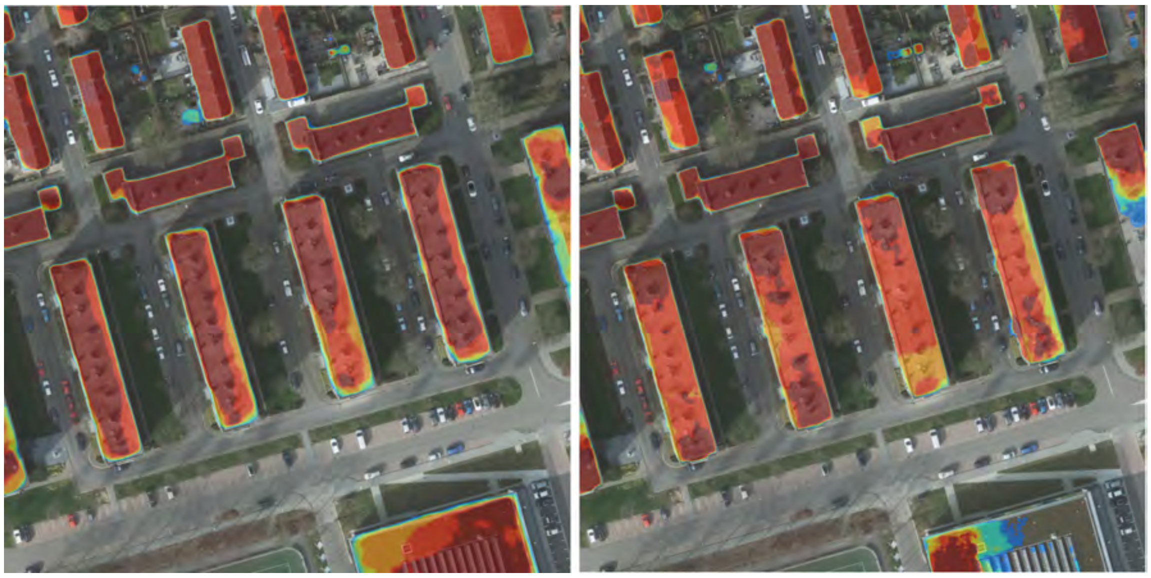


Figure 6.8: Left: A 153.6 by 153.6 meter 10 cm resolution image of Heerlen containing predictions of a model trained and tested on 30 cm imagery. Right: Predictions of a model trained and tested on 10 cm imagery.

For the use case of this model, building-wise statistics are more relevant. A breakdown based on size is especially of interest. Smaller buildings have a relatively low coverage in existing maps whereas most of the largest buildings are generally properly mapped. Section 6.6 provides some building-wise statistics.

6.4.3 Resolution robustness experiments

To measure the robustness of the ResU-net architecture with respect to image resolution, the model which was trained on the 10 cm resolution train set is tested on the 30 cm resolution test set and vice versa. The expectation is that the performance of both models suffers greatly when testing on a resolution deviating from the training resolution. The model trained on 30 cm and tested on 10 cm is expected to outperform the other model for two reasons: The first reason is that the average building size of the 10 cm resolution is nine times larger compared to the 30 cm resolution and larger buildings have shown to be easier to detect. The second reason is that it is easier to score well on a test set of 10 cm resolution because of the higher level of detail compared to the 30 cm test set the other model has to predict on.

6.4.4 Resolution robustness results

Table 6.6 combines the results of the resolution robustness experiments with the earlier experiments performed in this chapter. As expected, the experiments where the test set resolution differs from the train and validation set resolution show significantly reduced performance. Predictions for all four experiments are visualized for comparison in Appendix B.5.

Experiment	Train & validation resolution	Test resolution	IoU	Precision	Recall
1	30 cm	30 cm	0.752	0.850	0.866
2	10 cm	10 cm	0.722	0.809	0.859
3	30 cm	10 cm	0.483	0.633	0.671
4	10 cm	30 cm	0.201	0.704	0.220

Table 6.6: Overview of test scores of the ResU-net model trained and tested on Readar data.

6.5 BUILDING GEOMETRY ROBUSTNESS

Most CNN models are to a certain extent sensitive to object orientation. The sensitivity to orientation can be used as an advantage when the available dataset is limited in size. Popular data augmentation techniques include image rotation and image flipping to increase the number of unique images available to either train or test on. Artificially increasing the size of the training set this way can increase performance and should theoretically make the model more orientation robust. Combining predictions of the same image at different orientations is known as TTA. TTA acts as an ensembling method to increase the performance at test time. These augmentation techniques are commonly used, with the increase in overall model performance being the most widely reported quantifier on the effectiveness of these techniques.

6.5.1 Building geometry robustness experiment

In this experiment, not the model performance but the orientation robustness is quantified. The most direct way to quantify the orientation sensitivity of a model is to measure the pixel-wise differences between predictions for the same model on the same image at different orientations. The influence of augmentations during training and during testing is measured by the following setup. Two ResU-net models will be trained with the same default settings, except that one model uses the orientation augmentations described in Section 6.2.4 during training and the other model does not. When testing on the Heerlen dataset, test time augmentation will be applied using these eight orientations and the prediction for every unique orientation is stored individually. Training and testing for both models is conducted on a 30 cm resolution.

When evaluating differences between the eight unique orientations, the predicted probabilities for each unique orientation are binarized with a cut-off threshold at 0.5. A pixel is predicted unanimously if the predictions for all orientations are either all positive (8 positive predictions) or all negative (0 positive predictions). The mean percentage of unanimously predicted pixels will be the quantitative measure for the orientation robustness of the model.

6.5.2 Building orientation robustness results

Table 6.7 shows the comparison between the model trained with image orientation augmentations and the model trained without these augmentations. The model without train augmentations has slightly more consistent predictions between orientations. With this model, 93.49 percent of all pixels receive the same evaluation from all eight orientations against 93.00 percent of the model trained with orientation augmentations. The pixels unanimously predicted to be negative (no building) are the main contributors to this figure with 89.64 and 89.43 percent of all pixels respectively are unanimously predicted to be negative. Figure 6.9 shows a breakdown of the cases where at least one of the eight predictions is positive. The model with train augmentations seems to be more sensitive to object orientation as there is more disagreement in predictions between different orientations of the same image.

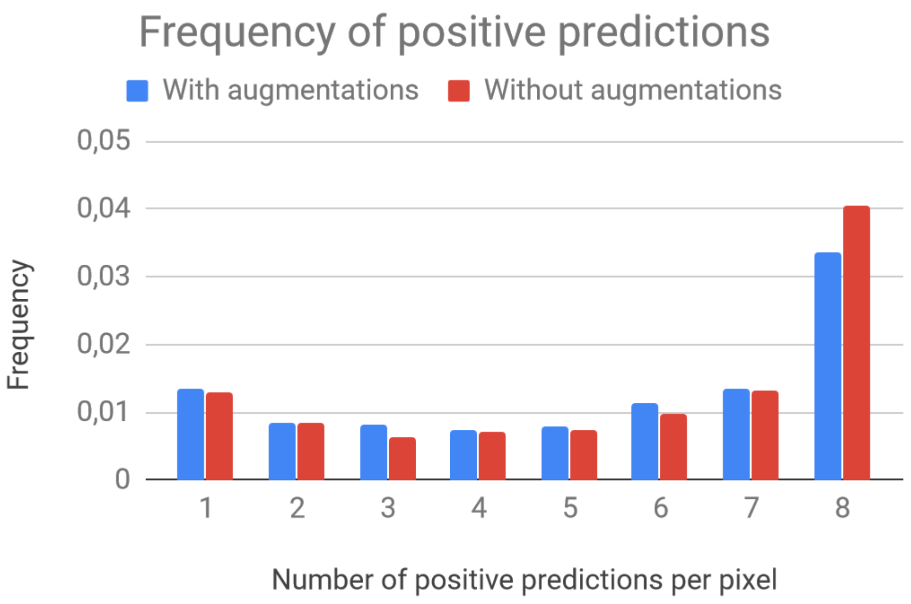


Figure 6.9: Visualization of the distribution of positive predictions per pixel on the Heerlen dataset of one ResU-net model trained with orientation augmentations and one model trained without.

Model	Train loss (Jaccard)	Val. loss (Jaccard)	Test loss (Jaccard)	All positive predictions (freq)	All negative predictions (freq)	Unanimous predictions (freq)
With augm.	0.2252	0.2542	0.336	0.0337	0.8964	0.9301
Without augm.	0.2266	0.2535	0.333	0.0406	0.8943	0.9349

Table 6.7: Comparison of predictions on Heerlen of two models showing the frequency of unanimous pixel-wise predictions.

6.6 BUILDING-WISE EVALUATION

As explained Section 6.4.2, pixel-wise evaluation metrics are biased with respect to building size. Therefore, the pixel-wise segmentation results from the ResU-net are vectorized. The pixel mask predictions are first binarized using a threshold of 0.5. When using TTA, the eight unique binary prediction masks per image are averaged pixel-wise resulting in the discrete set of possible probabilities of $\{0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1\}$. Here another cut-off threshold is used of 0.2, meaning that the pixel has to be classified as positive in at least two of the eight predictions. The resulting binary masks are vectorized into prediction vectors. These steps have been performed for two sets of predictions. The first set is the set of predictions on Heerlen by the 10 cm model discussed in Section 6.4. The second set is a set of predictions on Amersfoort by a model trained with the exact same settings but excluding all Amersfoort data from the training and validation set. This reduces the size of the training set from 374 km^2 with 80 km^2 to 294 km^2 and reduces the number of buildings in the ground truth by 100,350. Table 6.8 shows that the model trained without Amersfoort shows a significantly larger gap between the training loss and validation loss, a potential indication of overfitting.

Model	Train loss	Validation loss
Including Amersfoort	0.2699	0.3418
Excluding Amersfoort	0.1700	0.4076

Table 6.8: The 10 cm model as described in Section 6.4 which included Amersfoort in its dataset compared with a model which is trained identically but excludes Amersfoort.

The building-wise evaluation is performed based on an Intersection over Union between the ground truth and prediction vectors. A single prediction vector might contain multiple connected buildings in the ground truth. Consequently the decision has been made that for a prediction to be classified as a correct detection, the Intersection over Union with the ground truth has to be larger than zero, i.e. only a minimal overlap of one pixel is required. Any other IoU threshold would yield an unfair evaluation on prediction vectors spanning multiple buildings. Table 6.9 shows the evaluation of the predictions using this performance metric.

Initial results on Amersfoort and Heerlen

	Amersfoort	Heerlen
Total buildings in ground truth	100,350	82,773
Total predictions	82,231	57,611
Precision	0.524	0.579
Recall	0.911	0.939

Table 6.9: Initial building-wise precision and recall on Amersfoort and Heerlen.

Inspection of the results reveals that many predictions are smaller than 5 m^2: 45,301 predictions for Amersfoort are below 5 m^2 and 23,391 predictions for Heerlen. Figure 6.10 shows the precision of the predictions at various prediction sizes. The precision for predictions below 5 m^2 is 0.295 and 0.237 for Amersfoort and Heerlen respectively. Based on this information, all prediction vectors with an area smaller than 5 m^2 are discarded. The effects of introducing this minimum detection size threshold are shown in Table 6.10. A breakdown of recall given various building sizes is presented in Figure 6.13, showing that the ResU-net is not robust with regards to building size. Visual samples of the vectorized predictions are given in Figure 6.11 for Amersfoort and Figure 6.12 for Heerlen. The same samples including ground truth are provided in Appendices B.7 and B.8.

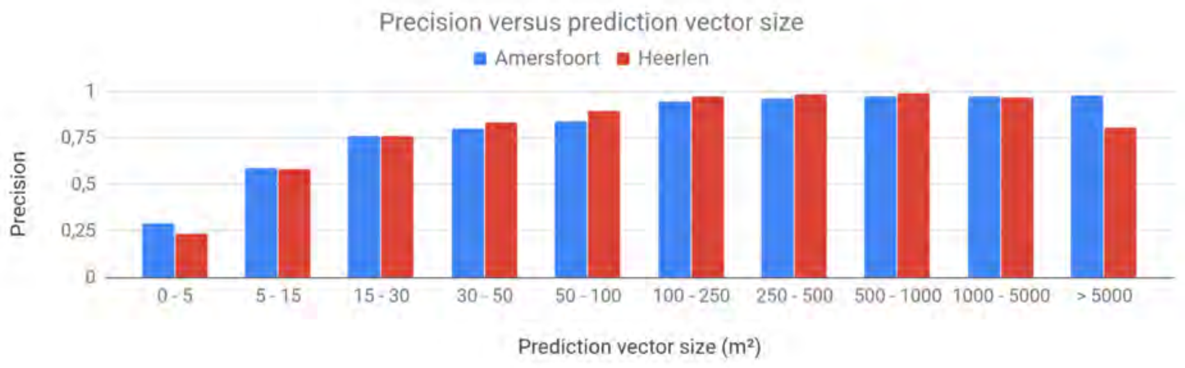


Figure 6.10: Precision of predictions on Amersfoort and Heerlen for various prediction vector area ranges.

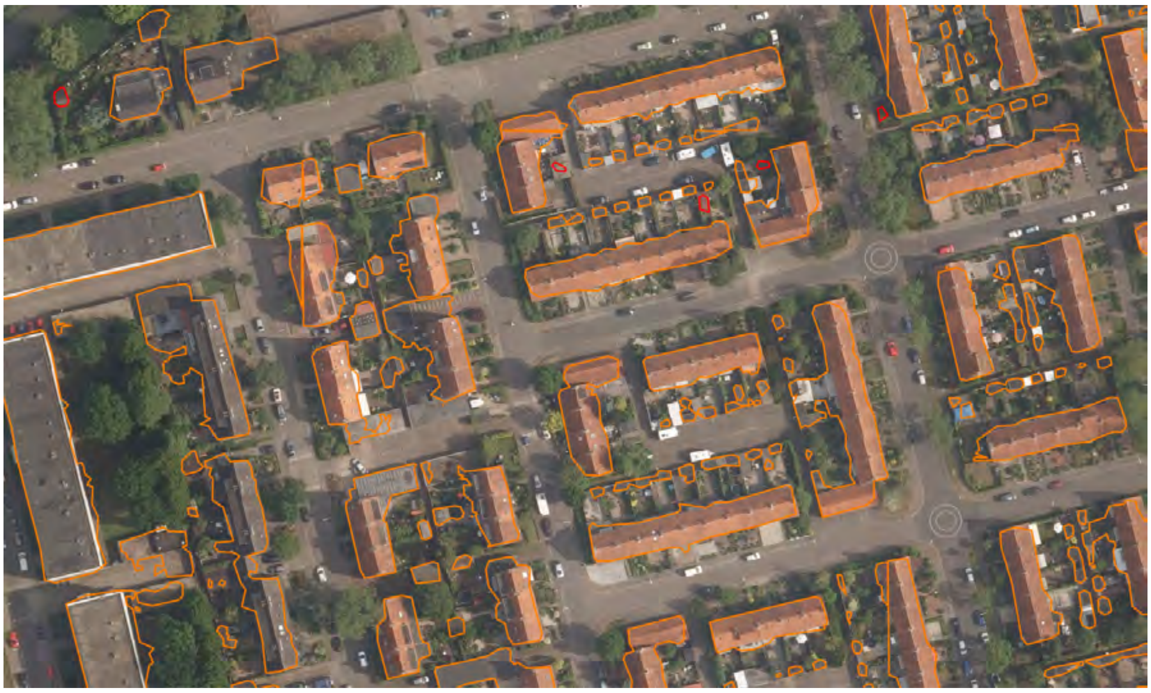


Figure 6.11: Vectorized predictions on Amersfoort. An orange outline indicates the prediction overlaps at least one building in the ground truth, red indicates no overlap.

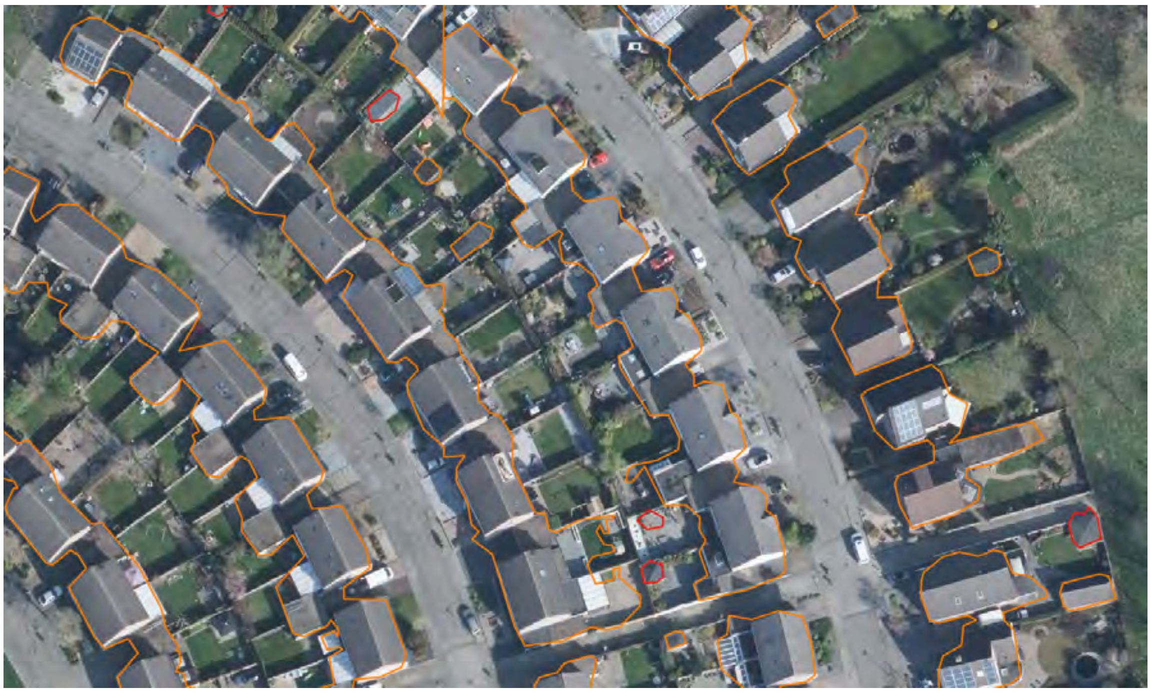


Figure 6.12: Vectorized predictions on Heerlen. An orange outline indicates the prediction overlaps at least one building in the ground truth, red indicates no overlap.

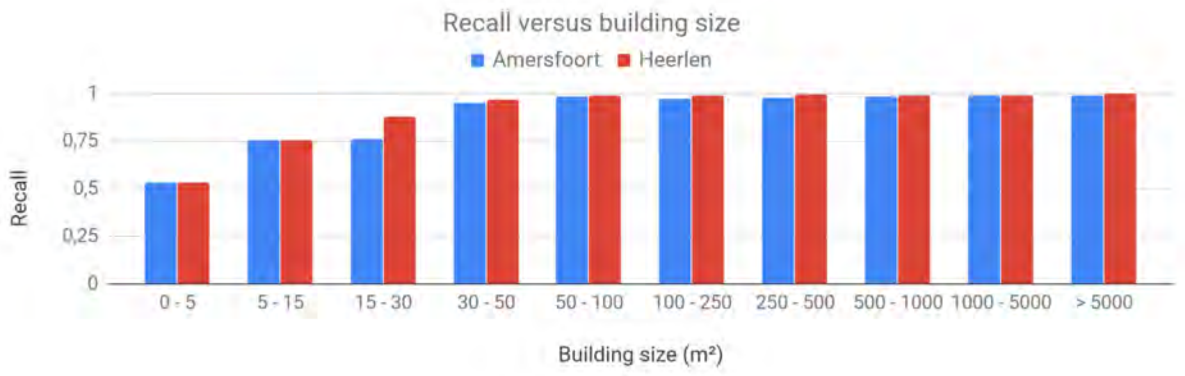


Figure 6.13: Recall of predictions on Amersfoort and Heerlen for various building area ranges.

Results on Amersfoort and Heerlen (prediction $> 5\,m^2$)

	Amersfoort	Heerlen
Total predictions	36,929 (-45,302)	32,3320 (-23,391)
Precision	0.806 (+0.2817)	0.816 (+0.2374)
Recall	0.884 (-0.0270)	0.922 (-0.0157)

Table 6.10: Building-wise precision and recall on Amersfoort and Heerlen using a minimum prediction size of 5 m². In brackets the comparison to the non-thresholded predictions.

6.6.1 Building-wise results per sub-datasource

As described in Chapter 4, the ground truth in these experiments is an aggregate of two main cadastral databases for buildings (BAG and BGT) and manually detected buildings. Readar offers a service where it reports all buildings visible on aerial imagery that are currently not listed in any cadastral database. This building detection on aerial imagery is currently performed manually. To provide more detailed insights for the comparison between manual mapping and the ResU-net model, the performance is broken down based on the sub-datasources of the ground truth. Figure 6.14 presents a detailed overview on recall for Heerlen per sub-datasource and Table 6.11 provides the overall figures for both Amersfoort and Heerlen.

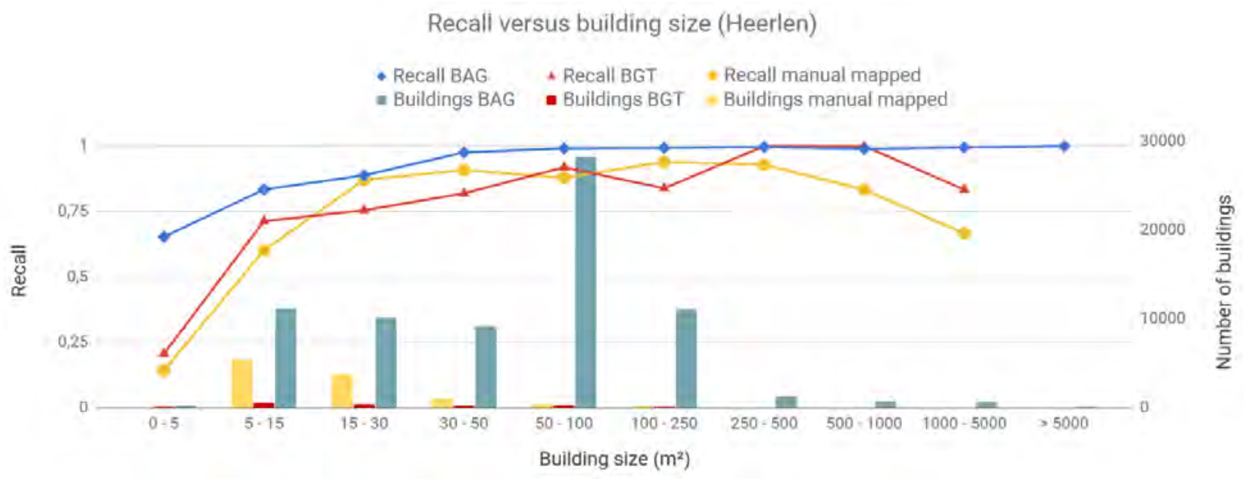


Figure 6.14: Recall of predictions larger than 5 m² on Heerlen per sub-datasource for various area ranges, with on the right axis the number of buildings in each sub-domain.

	Recall	
Data source	Amersfoort	Heerlen
Manual mapped	0.680	0.740
BGT	0.649	0.757
BAG	0.905	0.950
BGT + BAG	0.895	0.946
Manual Mapped + BGT + BAG	0.884	0.922

Table 6.11: Overall recall of predictions larger than 5 m^2 on Amersfoort and Heerlen per sub-datasource.

6.6.2 Evaluation of incorrect predictions

Experience tells that the aggregated ground truth is still not completely accurate. Therefore, a visual inspection is made of the predictions classified as "incorrect" based on the used ground truth. Figure 6.15 shows the results of a visual inspection on 283 of these false positives of Amersfoort and 300 false positives of Heerlen. Roughly 3 and 34 percent of all false positives respectively have been reclassified as buildings in the manual validation process. These buildings initially classified as false positives are the newly detected buildings currently not present in the ground truth. Given the use case of Readar, these newly detected buildings embody the added value of the model. The precision and recall have been recalculated by interpolating these results to all false positive predictions in Table 6.12. Figure 6.16 shows a breakdown of the interpolated classification results of the predictions.

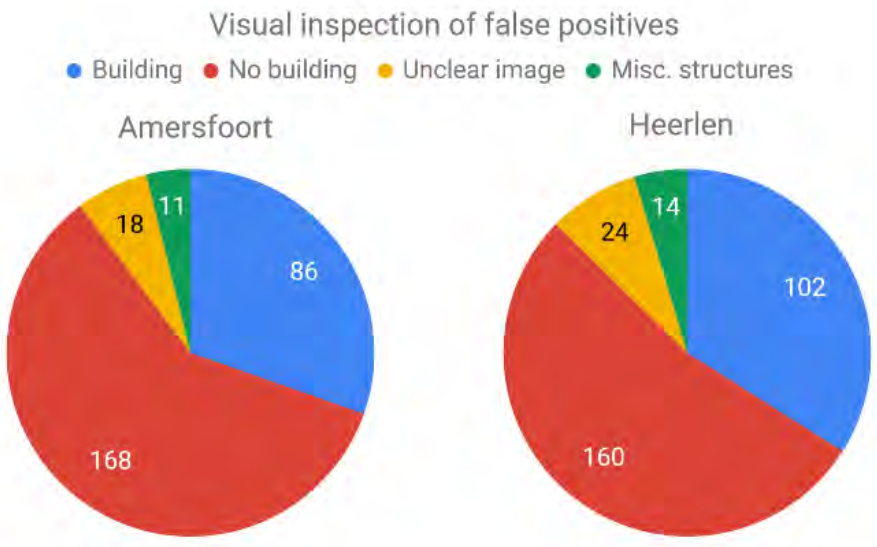


Figure 6.15: Manual reclassification of false positive predictions of Amersfoort and Heerlen. The category miscellaneous structures includes pylons, windmills, industrial structures, shelter at bus stops, etc.

Interpolated precision and recall based on manual reclassification results

	Amersfoort	Heerlen
False positives	7162	6277
Interpolated newly detected buildings	2387 (30% of 7162)	2134 (34% of 6277)
Interpolated precision	0.864 (+0.058)	0.879 (+0.063)
Interpolated recall	0.887 (+0.003)	0.924 (+0.002)

Table 6.12: Interpolated precision and recall of Amersfoort and Heerlen based on manual reclassification of false positives. In brackets the comparison to the results based on the used ground truth.

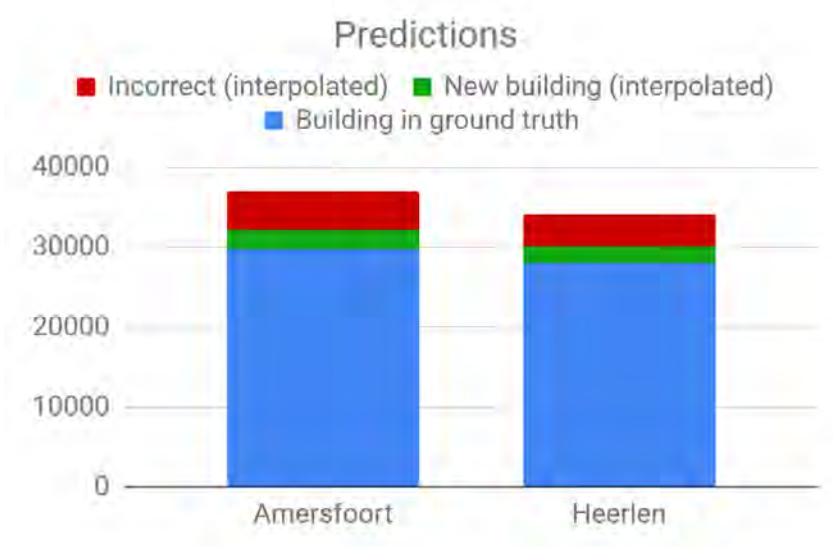


Figure 6.16: Interpolated classification results of the predictions on Amersfoort and Heerlen.

6.7 CONCLUSION

The ResU-net architecture was implemented with the intention of outperforming Mask R-CNN on the area of building geometry robustness, enabling the ResU-net to achieve satisfactory performance in densely populated areas. ResU-net obtained a 0.81 score on precision and 0.86 on recall (pixel-wise) on the Dutch city of Heerlen, translating to a 0.81 score on precision and 0.92 on recall (building-wise). Manual reclassification of a subset of false positives results in an interpolated 0.88 precision score (building-wise). These results combined with the robustness experiments give the impression that the ResU-net architecture is more robust than Mask R-CNN with respect to building geometry.

6.7.1 Robustness evaluation

The ResU-net architecture shows mediocre geographical robustness when training and testing on different geographical locations without retraining. The performance of a model on the transfer validation sets varies greatly between epochs during the training process, as shown by the introduction of transfer validation sets.

Empirical verdict on robustness:

- **ResU-net is not robust with respect to geographical location**. When trained on the crowdAI dataset assumedly located in North America, it obtains an Intersection over Union validation score of 0.867 and test scores of 0.440 on the city of Lusaka in Zambia and 0.432 on the Readar test set (various locations in the Netherlands). Trained on the crowdAI dataset, it also yields meagre scores of 0.269 precision and 0.296 recall (building-wise) on the Zambian city of Mongu.

- **ResU-net as an architecture is robust with respect to image resolution**. It has been trained twice: on 10 cm resolution imagery and the same imagery downsampled to 30 cm. The two models obtained Intersection over Union test scores of 0.722 and 0.752 respectively using identical hyperparameter settings. When the model trained on 30 cm is applied on 10 cm test imagery the test IoU drops to 0.483. The model trained on 10 cm applied to 30 cm yields a 0.201 IoU, indicating that a trained ResU-net model is not robust with respect to resolution.

- **ResU-net is partially robust with respect to building geometry**. Visually, it seems to be robust to varying building aspect ratios and shapes. It is not robust to building size as the likelihood of detection increases from 52% for buildings up to 5 m^2 to above 99% for buildings above 50 m^2. When binarizing the predictions using a threshold of 0.5, up to 6.99 percent of all pixels could obtain a different classification when a different orientation of the image is used. This indicates that the ResU-net is not robust with respect to orientation.

6.7.2 Task evaluation

The ResU-net architecture was implemented due to performance limitations of the object detection architecture Mask R-CNN. These performance limitations are caused by sensitivity of Mask R-CNN's object detection algorithm to three main components of building geometry: Orientation, aspect ratio and size. The pixel-wise segmentation approach of the ResU-net removes these limitations. This improves the robustness with respect to object orientation and aspect ratio as intended. An additional benefit of the segmentation approach of the ResU-net architecture is that its reduced complexity requires fewer hyperparameters to fine-tune. For example, the ResU-net also does not require any hyperparameter tuning to the expected building size distribution, image resolution or their interaction. Overall, the experiments with the ResU-net verify that the segmentation task suits the challenge of robust building detection on aerial imagery well.

7 CONCLUSION

In this chapter the objectives and research questions posed for this internship will briefly be reiterated. The performance and robustness of the Mask R-CNN and ResU-net models will be compared and evaluation in relation with the intended use cases of the 510 team of the NLRC and Readar. Finally, some open questions and topics for further research will be addressed.

7.1 OBJECTIVE AND RESEARCH QUESTIONS

The objective of this internship was to investigate the potential of Deep Learning for building detection on aerial imagery. The practical objective was stated as follows:

The development of a robust Deep Learning model capable of worldwide building detection on aerial imagery

As a result of a literature review, the Mask R-CNN model architecture has been implemented to perform the object detection and instance segmentation tasks and the ResU-net model architecture has been implemented to perform the image segmentation task. The robustness of Mask R-CNN and ResU-net has been evaluated with respect to three characteristics of the input data: Geographical location, image resolution and building geometry. The following research questions have been (partially) answered:

1. What is the robustness of a Mask R-CNN model with respect to the geographical location of the input data?

   Mask R-CNN is robust with respect to geographical location, as a model trained on an North American dataset can successfully detect buildings on imagery of locations in Africa, South America and Europe. On $\sim 29,000$ buildings in Mongu, Zambia, the model obtained a building-wise precision of $\sim$0.98 and a recall of $\sim$0.60.

2. What is the robustness of a Mask R-CNN model with respect to the resolution of aerial imagery?

   Mask R-CNN is not robust with respect to image resolution, but only because the model is sensitive to the interaction between the anchor scales hyperparameter, image resolution and building size distribution. Due to the influence of this interaction, no unbiased assessment could be made on the image resolution robustness of Mask R-CNN.

3. What is the robustness of a Mask R-CNN model with respect to building geometry?

   Mask R-CNN is not robust with respect to building geometry. Not only is the model sensitive to the building size distribution, but dealing with a large variety in building shapes and orientations also appears to be challenging when the model is limited to three anchor ratios.

4. What is the robustness of a U-net model with respect to the geographical location of the input data?

   ResU-net is not robust with respect to geographical location. When trained on the crowdAI dataset assumedly located in North America, it obtains an Intersection over Union validation score of 0.867 and test scores of 0.440 on the city of Lusaka in Zambia and 0.432 on the Readar test set (various locations in the Netherlands). Trained on the crowdAI dataset, it also yields meagre scores of 0.269 precision and 0.296 recall (building-wise) on the Zambian city of Mongu.

5. What is the robustness of a U-net model with respect to the resolution of aerial imagery?

   ResU-net as an architecture is robust with respect to image resolution. It has been trained twice: on 10 cm resolution imagery and the same imagery downsampled to 30 cm. The two models obtained Intersection over Union test scores of 0.722 and 0.752 respectively using identical hyperparameter settings. When the model trained on 30 cm is applied on 10 cm test imagery the test IoU score drops to 0.483. The model trained on 10 cm applied to 30 cm yields a 0.201 IoU, indicating that **a trained ResU-net model is not robust with respect to resolution.**

6. What is the robustness of a U-net model with respect to building geometry?

   ResU-net is partially robust with respect to building geometry. Visually, it seems to be robust to varying building aspect ratios and shapes. It is not robust to building size as the likelihood of detection increases from 52% for buildings up to 5 m^2 to above 99% for buildings above 50 m^2. When binarizing the predictions using a threshold of 0.5, up to 6.99 percent of all pixels could obtain a different classification when a different orientation of the image is used. This indicates that the ResU-net is not robust with respect to orientation.

Empirical robustness assessment of Mask R-CNN and ResU-net

Robustness aspect	Mask R-CNN	ResU-net
Geographical location	Yes	No
Image resolution	No	Yes (individual model: No)
Building geometry	No	Partial
Size	No	No
Shape	No	Yes
Orientation	No	Partial

Table 7.1: Overview of empirical robustness assessment of Mask R-CNN and ResU-net.

7.2 COMPARISON MASK R-CNN AND RESU-NET

A comparison of the robustness assessment of Mask R-CNN and ResU-net is shown in Table 7.1. The main conclusions are highlighted below.

The Mask R-CNN model seems to be more geographically robust than the ResU-net model but it is unclear why. Using a similar experimental setup, a Mask R-CNN model obtained a 0.98 precision and 0.60 recall (building-wise) against 0.27 precision and 0.30 recall of a ResU-net model. This difference could potentially be explained by the mechanisms of the RPN of Mask R-CNN but further research is required for improved understanding on the underlying cause. Transfer validation sets showed that the performance of the ResU-net on external datasets is relatively unstable during the training process.

No unbiased assessment could be made of the robustness of Mask R-CNN with respect to input image resolution. A strong correlation was found between model performance and the interaction between image resolution, building size distribution and the anchor size hyperparameter. The ResU-net also shows a correlation between model performance and the interaction between image resolution and building size distribution. However, the ResU-net model is hyperparameter-free with regards to image resolution or building size, allowing it to be trained on any resolution without requiring hyperparameter tuning.

An important conclusion is that the Region Proposal Network module of Mask R-CNN uses hyperparameters which indirectly describe the expected range of building sizes and building aspect ratios. The specificity imposed by these hyperparameters makes the Mask R-CNN model less robust with regards to building geometry as its performance suffers on buildings which deviate from the expected ranges. The ResU-net architecture does not contain these self-imposed limitations due to the difference in nature between the object detection task and the image segmentation task.

7.2.1 Use case 510

For the 510 data team of the NLRC the benefit of an automated building detection model is the greatest in poorly mapped areas in which the Red Cross is active. This mainly encompasses rural areas in the developing world. This means that a key aspect for the 510 team of the NLRC in the potential deployment of an automated building detection model is its geographical robustness. The Mask R-CNN model trained on the crowdAI dataset has shown to be geographically robust with promising results in areas of interest for the 510 team of the NLRC and Missing Maps. The Mask R-CNN model outperforms the ResU-net model which was trained on the same dataset. The geographical robustness makes the Mask R-CNN model the most suitable building detection model for the 510 team of the two implemented in this internship.

7.2.2 Use case Readar

For Readar, the key aspect is strong performance on imagery of Western Europe. This includes both rural and urban environments. To integrate an automated building detection model into their commercial mapping services, high precision and high recall are required. The ResU-net model is relatively robust to varying building sizes, shapes and orientations, in contrast to the Mask R-CNN model. This is a significant advantage over Mask R-CNN in densely populated areas. Results show that the performance of Mask R-CNN is unsatisfactory in Western European urban environments but that ResU-net copes well in Dutch cities. This robustness with respect to building geometry makes the ResU-net the most suitable building detection architecture for Readar of the two implemented in this internship.

7.2.3 Discussion

The main potentially solvable issues observed in this internship are related to the sensitivity of Mask R-CNN towards building geometry. The building orientation issue could possibly be solved by changing the Mask R-CNN architecture. The underlying reason of inefficient bounding boxes is not the orientation of the building itself, but the horizontal and vertical orientation of the bounding box. Changing the backbone of Mask R-CNN into a rotation-invariant CNN with rotating bounding boxes as proposed in [43] might solve this problem.

A more thorough approach which solves both the orientation and building size sensitivity issues of Mask R-CNN, would be to change the RoIAlign layer. The RoIAlign layer resizes each proposal to predefined square dimensions. Swapping this basic transformation for a more advanced, spatial transformation network similar to Spatial Transformer Networks [29] could potentially solve both the orientation and building size issues of Mask R-CNN. Adding a spatial transformation network to the ResU-net as an architectural enhancement is likely to improve its orientation robustness and potentially also its image resolution robustness.

One other discrepancy that remains unexplained in this internship is the difference in observed geographical robustness between Mask R-CNN and ResU-net. Potentially the RPN of Mask R-CNN introduces a bias towards making positive detections by generating a fixed number of proposals regardless of the activation values in the feature map generated by the backbone. Further research is required to further investigate this issue.

8 RECOMMENDATIONS

8.1 GENERAL RECOMMENDATIONS

One recurring theme in this internship was the quality of available data. Issues included geographic misalignment between imagery and ground truth labels and temporal gaps between the collection of imagery and ground truth labels. In some cases a varying quality of ground truth labels prevented accurate quantitative analysis on the performance of the Machine Learning models. A lack of meta-data on origin of the ground truth labels caused issues with solving alignment issues between imagery and the ground truth labels, mainly with the Bing Maps imagery and OpenStreetMap data.

8.2 RECOMMENDATIONS TO 510 AND MISSING MAPS

The main recommendation to the Missing Maps initiative is to improve the administration of the active background source that is being used to map buildings. As long as this meta-data is not well-maintained, it remains a challenge to properly align the mapping results of multiple volunteers onto one imagery source.

For the 510 data team, the availability of resources to experiment with Deep Learning is a key aspect to initiate successful development and deployment of such Deep Learning models. For this, a strategic investment in both substantial computing power and a reliable source of aerial or high-resolution satellite imagery is required. This would allow the 510 team to further explore the capabilities of Deep Learning for building detection and related remote sensing tasks. This internship merely showcased one potential use case of Deep Learning for remote sensing. Besides this showcase, it must be stressed that the potential benefit of Deep Learning methods for the 510 data team is not limited to only the building detection task. The potential benefits of Deep Learning methods are most likely attainable for many other remote sensing tasks as the Machine Learning field keeps rapidly evolving and expanding.

For the Red Cross in general, it will be important to maintain warm relations with influential and resourceful stakeholders. For example, the Microsoft Bing Maps team has created a model that was used to detect buildings in the entire United States of America and Canada [48]. The Microsoft Bing Maps team has a large capacity in human, computational and data resources allowing them to handle deploy projects on a large scale. Although there are obvious disadvantages when relying on external partners for data compared to in-house projects, the additional resources can significantly increase project scales and increase deployment speed.

8.3 RECOMMENDATIONS TO READAR

The main recommendation is to use the ResU-net model for building detection. The model in its current form still has room for improvement, so one of the main recommendations is to improve the model using the following steps:

- Increase the size of the training set.
  More training data is available, and using additional data to train the model would increase the quality of the predictions.

- Add the height data as a fourth input channel.
  The assumption is that using height data would improve the quality of the predictions. Creating a nDSM (normalized Digital Surface Model) using private in-house techniques based on the imagery that is used as input would provide even more accurate data which would likely increase the quality of the models predictions even further.

- Figure out how the 3-channel pretrained weights from ImageNet or previous well-performing models can be used as initialization for the new model with 4 input channels.
  Using the weights file that is pretrained on ImageNet improves the convergence speed of the training process. These pretrained weights are normally used for 3-channel input, whereas a 4-channel input is desired for the addition of height information.

- Change encoder backbone from ResNet-50 to ResNet-101.
  The ResNet-101 model has more layers which results in a deeper understanding of the image and therefore increased performance. The ResNet model with 101 layers also has a layer receptive field compared to the ResNet-50 model which should increase the model's accuracy on relatively large buildings. The downside of the ResNet-101 model is that the batch size during training has to be decreased because of increased memory requirements. This will have a negative impact on performance.

- Use more GPUs to increase the batch size during training.
  Using more GPUs will yield small performance increases in terms of speed. However, the main benefit of using multiple GPUs is that a larger batch size can be used during the training process. The larger batch size will lead to weight updates that closer represent the data distribution and it increases the benefits of batch normalization. This results in improved convergence and will thus yield a better performing model.

BIBLIOGRAPHY

Abdulla, W. (2017). Mask r-cnn for object detection and instance segmentation on keras and tensorflow. https://github.com/matterport/Mask_RCNN.

Chaurasia, A. and Culurciello, E. (2017). Linknet: Exploiting encoder representations for efficient semantic segmentation. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE.

Chen, Q., Wang, L., Wu, Y., Wu, G., Guo, Z., and Waslander, S. L. (2019). Aerial imagery for roof segmentation: A large-scale dataset towards automatic mapping of buildings. *ISPRS Journal of Photogrammetry and Remote Sensing*, 147:42–55.

Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Dertat, A. (2018). Applied deep learning - part 4: Convolutional neural networks. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. Accessed: 2018-09-19.

Dormehl, L. (2018). What is an artificial neural network. https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/. Accessed: 2018-09-23.

dstl (2017). Kaggle challenge: Dstl satellite imagery feature detection. https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection. Accessed: 2018-09-29.

Facebook (2018). Deepglobe - satellite image understanding challenge. https://www.deepglobe.org/. Accessed: 2018-09-29.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.

Gao, H. (2017). Faster r-cnn explained. https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8. Accessed: 2018-09-20.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

Golovanov, S., Neuromation, O., Kurbanov, R., Artamonov, A., Davydow, A., and Nikolenko, S. (2018). Building detection from satellite imagery using a composite loss function. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 219–2193. IEEE.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947.

Hamaguchi, R. and Hikosaka, S. (2018). Building detection from satellite imagery using ensemble of size-specific detectors. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 223–2234. IEEE.

He, K., Girshick, R., and Dollár, P. (2018). Rethinking imagenet pre-training. *arXiv preprint arXiv:1811.08883*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.

Hui, J. (2018). Image segmentation with mask r-cnn. https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272. Accessed: 2018-09-25.

Humanity and Inclusion (2018). Crowdai - mapping challenge. https://www.crowdai.org/challenges/mapping-challenge. Accessed: 2018-09-29.

Iglovikov, V., Mushinskiy, S., and Osin, V. (2017). Satellite imagery feature detection using deep convolutional neural network: A kaggle competition. *arXiv preprint arXiv:1706.06169*.

Iglovikov, V., Seferbekov, S., Buslaev, A., and Shvets, A. (2018). Ternausnetv2: Fully convolutional network for instance segmentation. *arXiv preprint arXiv:1806.00844*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025.

Karpathy, A. (2018a). Stanford university cs231n: convolutional neural networks for visual recognition. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf. Accessed: 2018-09-22.

Karpathy, A. (2018b). Stanford university cs231n: convolutional neural networks for visual recognition. http://cs231n.github.io/convolutional-networks/#pool. Accessed: 2018-09-21.

Karpathy, A. (2018c). Stanford university cs231n: convolutional neural networks for visual recognition. http://cs231n.github.io/neural-networks-3/#sgd. Accessed: 2018-09-21.

Karpathy, A. (2018d). Stanford university cs231n: convolutional neural networks for visual recognition. *URL: http://cs231n. stanford. edu/syllabus. html*.

Kashyapa, R. (2018). Historic error rates for the imagenet competition. http://qualitastech.com/artificial-intelligence-manufacturing. Accessed: 2018-09-22.

Kersbergen, D. (2018). Automated building damage classification using remotely sensed data: Case study: Hurricane damage on st. maarten.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Lesiv, M., See, L., Laso Bayas, J., Sturn, T., Schepaschenko, D., Karner, M., Moorthy, I., McCallum, I., and Fritz, S. (2018). Characterizing the spatial and temporal availability of very high resolution satellite imagery in google earth and microsoft bing maps as a source of reference data. *Land*, 7(4):118.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Liu, D. (2017). A practical guide to relu. https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7. Accessed: 2018-09-23.

Liu, L., Pan, Z., and Lei, B. (2017). Learning a rotation invariant detector with rotatable bounding box. *arXiv preprint arXiv:1711.09405*.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3.

Maggiori, E., Tarabalka, Y., Charpiat, G., and Alliez, P. (2017). Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Microsoft Bing Maps Team (2017). Microsoft releases 12 million canadian building footprints as open data. https://blogs.bing.com/maps/2019-03/microsoft-releases-12-million-canadian-building-footprints-as-open-data. Accessed: 2019-03-13.

Mnih, V. and Hinton, G. E. (2010). Learning to detect roads in high-resolution aerial images. In *European Conference on Computer Vision*, pages 210–223. Springer.

Ohleyer, S. (2018). Building segmentation on satellite images. *Web: https://project. inria. fr/aeri-alimagelabeling/files/2018/01/fp_ohleyer_c ompressed. pdf*.

Planet.com (2017). Kaggle challenge: Understanding the amazon from space. https://www.kaggle.com/c/planet-understanding-the-amazon-from-space. Accessed: 2018-09-30.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Seferbekov, S. (2019). Kaggle challenge: Data science bowl - [ods.ai] topcoders, 1st place solution. https://www.kaggle.com/c/data-science-bowl-2018/discussion/54741. Accessed: 2018-09-28.

SpaceNet (2019). Spacenet building footprint extraction challenge. https://spacenetchallenge.github.io. Accessed: 2018-09-27.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Wikimedia Commons (2018). Sigmoid function - wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Sigmoid_function. Accessed: 2018-09-22.

Xie, M., Jean, N., Burke, M., Lobell, D., and Ermon, S. (2016). Transfer learning from deep features for remote sensing and poverty mapping. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500.

Yak (2019). Crowdai - yak's mapping challenge summary, 2nd place solution. https://www.crowdai.org/topics/yak-s-mapping-challenge-summary/discussion. Accessed: 2018-09-29.

Yakubovskiy, P. (2017). Segmentation models. https://github.com/qubvel/segmentation_models.

Yang, H. L., Yuan, J., Lunga, D., Laverdiere, M., Rose, A., and Bhaduri, B. (2018). Building extraction at scale using convolutional neural network: Mapping of the united states. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(8):2600–2614.

Yuan, J. (2018). Learning building extraction in aerial scenes with convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(11):2793–2798.

Yuan, J., Yang, H.-H. L., Omitaomu, O. A., and Bhaduri, B. L. (2016). Large-scale solar panel mapping from aerial images using deep convolutional networks. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 2703–2708. IEEE.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Zhang, A., Liu, X., Gros, A., and Tiecke, T. (2017). Building detection from satellite images on a global scale. *arXiv preprint arXiv:1707.08952*.

Zhao, K., Kang, J., Jung, J., and Sohn, G. (2018). Building extraction from satellite images using mask r-cnn with building boundary regularization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 242–2424. IEEE.

A APPENDIX: TECHNICAL SETUP

A.1 DATASET LOCATIONS AND SIZES

In “Rijksdriehoekscoördinaten” (EPSG 28991), geometries defined using bounding boxes:

- Amersfoort, 80 km^2, [151000, 470000, 159000, 460000]
- Arnhem, 110 km^2, [186000, 446000, 197000, 436000]
- Breda, 81 km^2, [106400, 404000, 118000, 397000]
- Maastricht, 42 km^2, [173700, 320400, 182000, 315300]
- Molenwaard, 61 km^2, [111300, 436000, 122400, 430500]
- Heerlen, 88 km^2, [190738, 327572, 199400, 317400]

In “WSG84” (EPSG 4326), geometries defined using ESRI geometries:

- Malawi, Nsanje, 201 tiles, 1.2 km^2
  Point(-16.9203123, 35.2645404).buffer(0.005)
- Zambia, Zambezi, 3679 tiles, 20 km^2
  Point(-13.5436715, 23.1118652).buffer(0.023)
- Zambia, Kalabo, 749 tiles, 4.2 km^2
  Point(-14.994510, 22.684117).buffer(0.01)
- Zambia, Mongu, 11,273 tiles, 67 km^2, 29k buildings
  Point(-15.273368, 23.151050).buffer(0.04)
- Peru, Iquitos, 880 tiles, 5 km^2
  Polygon([(-3.732905, -73.264316),(-3.732905, -73.258759),(-3.735253, -73.258759),(-3.735253, -73.264316),(-3.732905, -73.264316)])
- Zambia, LusakaNorth, 3360 tiles, 21 km^2, 42k buildings
  Polygon([[-15.348487, 28.277491],[-15.348487, 28.334464],[-15.374740, 28.334464],[-15.374740, 28.277491],[-15.348487, 28.277491]])
- Zambia, LusakaWest, 3168 tiles, 14 km^2, 40k buildings
  Polygon([[-15.374740, 28.334464],[-15.374740, 28.224526],[-15.453267, 28.224526],[-15.453267, 28.334464],[-15.374740, 28.334464]])

A.2 TECHNICAL ENVIRONMENT

All computations are run on Amazon's EC2 cloud computing services. One Amazon EC2 p3.2xlarge instance was used, which has a NVIDIA Tesla V100 GPU with 16 GB GPU memory and eight virtual CPUs with 61 GB RAM available. An Amazon Machine Image (AMI) from NVIDIA was used as base image, the NVIDIA Volta Deep Learning AMI. Version 18.09.01 of the AMI was used for the experiments. The AMI contains Ubuntu 16.04 LTS as operating system, Docker, and Nvidia-docker. NVIDIA maintains several Docker images in the NVIDIA GPU Cloud with various Deep Learning software packages installed. Version 18.09 of the Docker image containing Tensorflow 1.12 and Python 3.5 was selected. The customized Docker image contained additional Python packages including Keras (a high-level Machine Learning framework on top of Tensorflow), PostGIS (database software for geospatial data), GDAL (geospatial data processing software), OpenCV (image processing software) amongst others.

A.3 HYPERPARAMETERS MASK R-CNN

```
class Config(object):
# NUMBER OF GPUs to use. For CPU training, use 1
    GPU_COUNT = 1
    #Default value

    IMAGES_PER_GPU = 5
    #Out-of-memory errors occurred at larger batch sizes

    # Backbone network architecture
    BACKBONE = "resnet101"
    #Default value

    # The strides of each layer of the FPN Pyramid. These values
    # are based on a Resnet101 backbone.
    BACKBONE_STRIDES = [4, 8, 16, 32, 64]
    #Default value

    # Number of classification classes (including background)
    NUM_CLASSES = 2
    #Technical amendment
    #Two classes: Building and background/no building

    # Length of square anchor side in pixels
    RPN_ANCHOR_SCALES = (8, 16, 32, 64, 128, 256)
    #Default was (32, 64, 128, 256, 512).
    #Optimization amendment.
    #AP@IoU0.5 increased by 0.101

    # Ratios of anchors at each cell (width/height)
    # A value of 1 represents a square anchor, and 0.5 is a wide anchor
    RPN_ANCHOR_RATIOS = [0.5, 1, 2]
    #Default value.
    #The options [0.6, 1, 1.67], [0.4, 1, 2.5], [0.33, 1, 3] and
    #[0.25, 1, 4] have been tested as well but did not yield improvements.
    # Anchor stride
    # If 1 then anchors are created for each cell in the backbone feature map.
    RPN_ANCHOR_STRIDE = 1
    #Default value

    # Non-max suppression threshold to filter RPN proposals.
    RPN_NMS_THRESHOLD = 0.7
    #Default value. Extensive testing has been performed for values in the
    #range of 0.1 to 0.9, where higher values perform better in dense
    #urban regions containing many connected buildings.

    # How many anchors per image to use for RPN training
```

```
RPN_TRAIN_ANCHORS_PER_IMAGE = 256
#Default value

# ROIs kept after non-maximum supression (training and inference)
POST_NMS_ROIS_TRAINING = 2000
#Default value

POST_NMS_ROIS_INFERENCE = 2500
#Default was 1000
#Optimization amendment
#AP@IoU0.5 increased by 0.091 from 0.690 to 0.781

# If enabled, resizes instance masks to a smaller size to reduce
# memory load. Recommended when using high-resolution images.
USE_MINI_MASK = True
#Default value

MINI_MASK_SHAPE = (56, 56)  # (height, width) of the mini-mask
#Default was (28, 28)
#Optimization amendment
#Unknown performance increase as this was changed before the first run.

# Input image resizing
# square: Resize and pad with zeros to get a square image
#         of size [max_dim, max_dim].
IMAGE_RESIZE_MODE = "square"
#Default value

IMAGE_MIN_DIM = 320
IMAGE_MAX_DIM = 320
#Technical amendment
#Values have changed to 320 as that was the first multiple of 32
#above the original 300 pixel image size

# Image mean (RGB)
MEAN_PIXEL = np.array([77.7, 89.6, 101.3])  #Mean of crowdAI train set
#Default was [123.7, 116.8, 103.9], the mean values of ImageNet
#Optimization amendment
#Unknown performance increase as this was changed before the first run.

# Number of ROIs per image to feed to classifier/mask heads
TRAIN_ROIS_PER_IMAGE = 200
#Default value

# Percent of positive ROIs used to train classifier/mask heads
ROI_POSITIVE_RATIO = 0.33
#Default value

# Pooled ROIs
```

```
POOL_SIZE = 7
#Default value

MASK_POOL_SIZE = 14
#Default value

# Shape of output mask
MASK_SHAPE = [28, 28]
#Default value

# Maximum number of ground truth instances to use in one image
MAX_GT_INSTANCES = 100
#Default value

# Bounding box refinement standard deviation for RPN and final detections.
RPN_BBOX_STD_DEV = np.array([0.1, 0.1, 0.25, 0.25])
#Default was [0.1, 0.1, 0.2, 0.2]
#Optimization amendment
#AP@IoU0.5 increased by 0.001

BBOX_STD_DEV = np.array([0.1, 0.1, 0.25, 0.25])
#Default was [0.1, 0.1, 0.2, 0.2]
#Optimization amendment
#AP@IoU0.5 increased by 0.001

# Max number of final detections
DETECTION_MAX_INSTANCES = 100
#Default value

# Minimum probability value to accept a detected instance
# ROIs below this threshold are skipped
DETECTION_MIN_CONFIDENCE = 0.7
#Default value

# Non-maximum suppression threshold for detection
DETECTION_NMS_THRESHOLD = 0.7
#Default value.
#In terms of AP@IoU0.5, it outperforms a value of 0.8 by 0.008,
#0.75 by 0.002 and 0.65 by 0.007

#########################################
###          TRAIN PARAMETERS
#########################################
# Number of training steps per epoch
STEPS_PER_EPOCH = 1000
#Default value

# Number of validation steps to run at the end of every training epoch.
VALIDATION_STEPS = 50
```

```
#Default value

LEARNING_RATE = 0.0001
#Default was 0.001
#Optimization amendment
#Training for an additional 5 epochs at a learning rate of 0.0001
#instead of the default value 0.001 increased the AP@IoU0.5 with 0.010

LEARNING_MOMENTUM = 0.9
#Default value

# Weight decay regularization
WEIGHT_DECAY = 0.0001
#Default value

# Use RPN ROIs or externally generated ROIs for training
USE_RPN_ROIS = True
#Default value

# Train or freeze batch normalization layers
#     None: Train BN layers. This is the normal mode
#     False: Freeze BN layers. Good when using a small batch size
#     True: (dont use). Set layer in training mode even when inferencing
TRAIN_BN = False  # Defaulting to False since batch size is often small
#Default value

# Gradient norm clipping
GRADIENT_CLIP_NORM = 5.0
#Default value
```

A.4 HYPERPARAMETERS RESU-NET

```
class Config(object):
    # NUMBER OF GPUs to use. For CPU training, use 1
    GPU_COUNT = 1

    CPU_COUNT = 8

    #One of resnet34, resnet50 or resnet101
    BACKBONE = 'resnet50'


    # Number of images to train with on each GPU.
    IMAGES_PER_GPU = 10

    #With False the default ImageNet pretrained weights are used instead
    USE_PRETRAINED_MODEL = True
    if USE_PRETRAINED_MODEL:
            MODEL_NAME = "10cm_18nov1"
    else:
            MODEL_NAME = None #ImageNet weights

    #Initial learning rate
    LEARNING_RATE = 0.1

    LEARNING_MOMENTUM = 0.8

    # Weight decay/L2 regularization
    WEIGHT_DECAY = 1.5

    STEPS_PER_EPOCH = 500

    VALIDATION_STEPS = 500

    GRADIENT_CLIP_NORM = 5.0

    #The number of epochs without improvement after which
    #the learning rate is lowered
    REDUCE_LR_ON_PLATEAU_PATIENCE = 3

    #The factor with which the learning rate is multiplied
    #once the model plateaus
    REDUCE_LR_ON_PLATEAU_FACTOR = 0.3

    #Learning rate won't reduced below this number
    MIN_LEARNING_RATE = 1e-12

    #Training will be stopped once no improvement has been
```

```
#seen after this number of subsequent epochs
EARLY_STOPPING_PATIENCE = 10

MAX_EPOCHS = 1000

USE_TRAIN_AUGMENTATIONS = True

USE_BATCH_NORM = IMAGES_PER_GPU > 5

#Select how many epochs the first stage of training
#should last before training on the full model should start
PRETRAIN_DECODER_EPOCHS = 0
```

B APPENDIX: VISUAL SAMPLES

B.1 CROWDAI IMAGERY

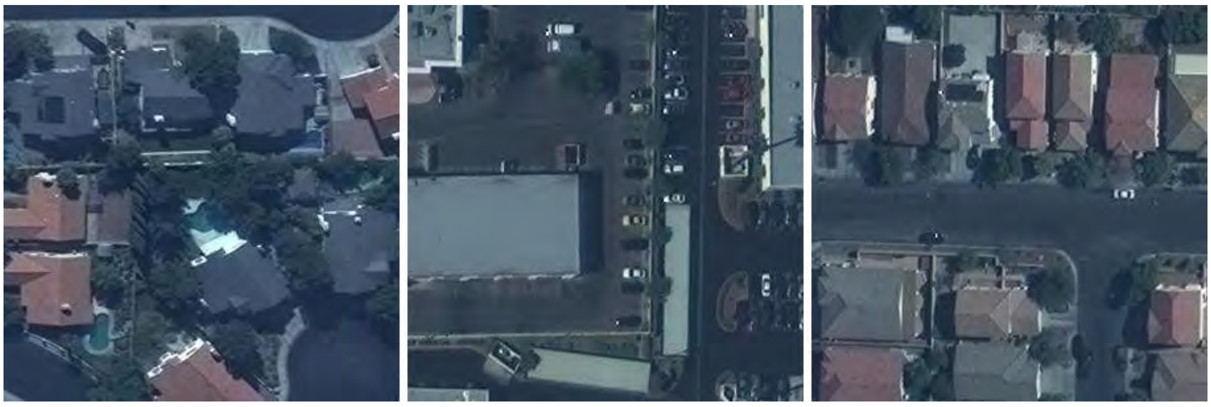


Figure B.1: Three sample images from the crowdAI dataset.

B.2 MASK R-CNN ON ZAMBIA

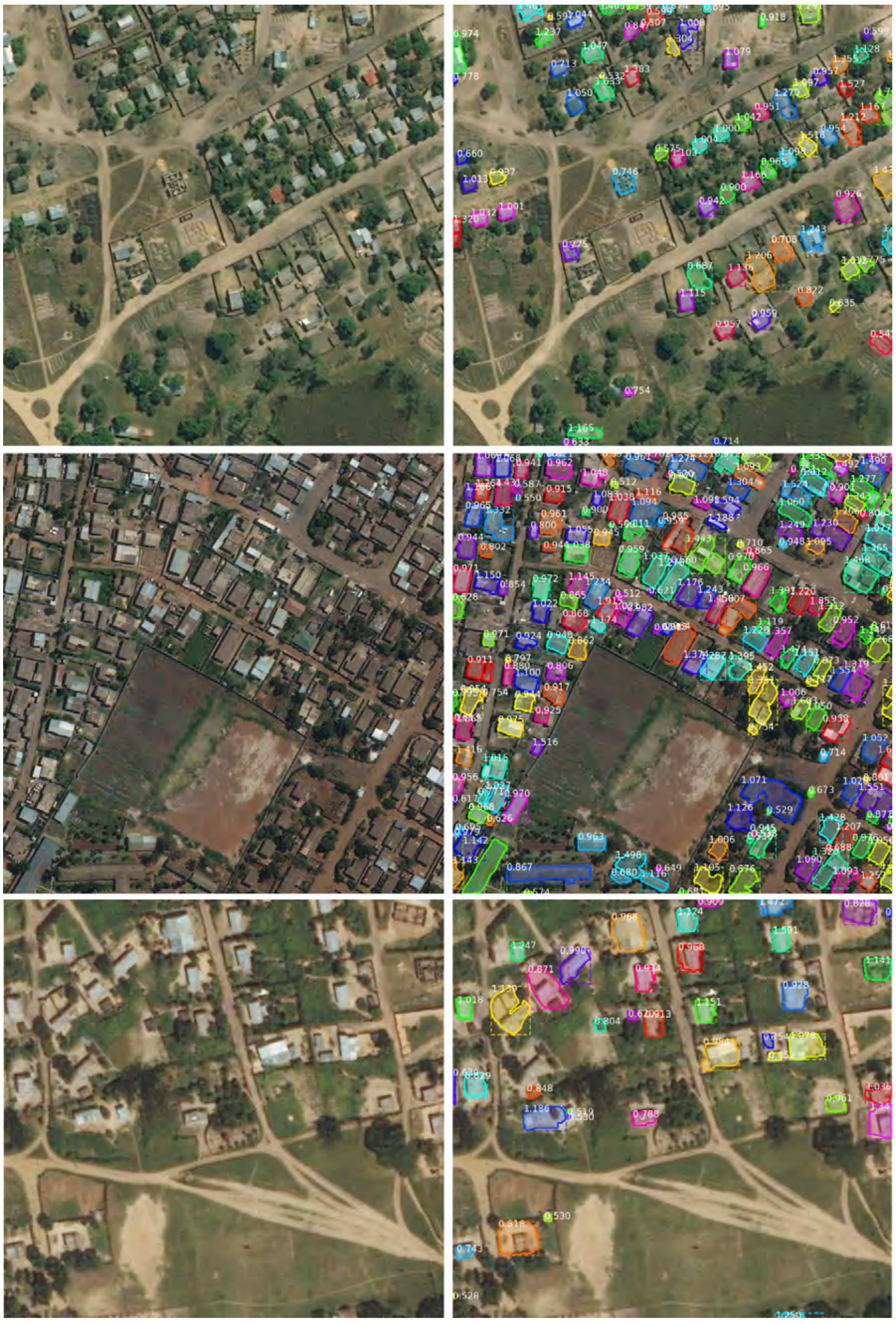


Figure B.2: Left: Samples from imagery of Zambia at 30 cm resolution. Right: TTA predictions of Mask R-CNN on imagery upsampled by a factor of 2.5. Top row: A 307.2 by 307.2 meter sample from Kalabo. Middle row: A 307.2 by 307.2 meter sample from Lusaka. Bottom row: A 230.4 by 230.4 meter sample from Zambezi.

B.3 OSM AND MASK R-CNN ON ZAMBIA, MONGU

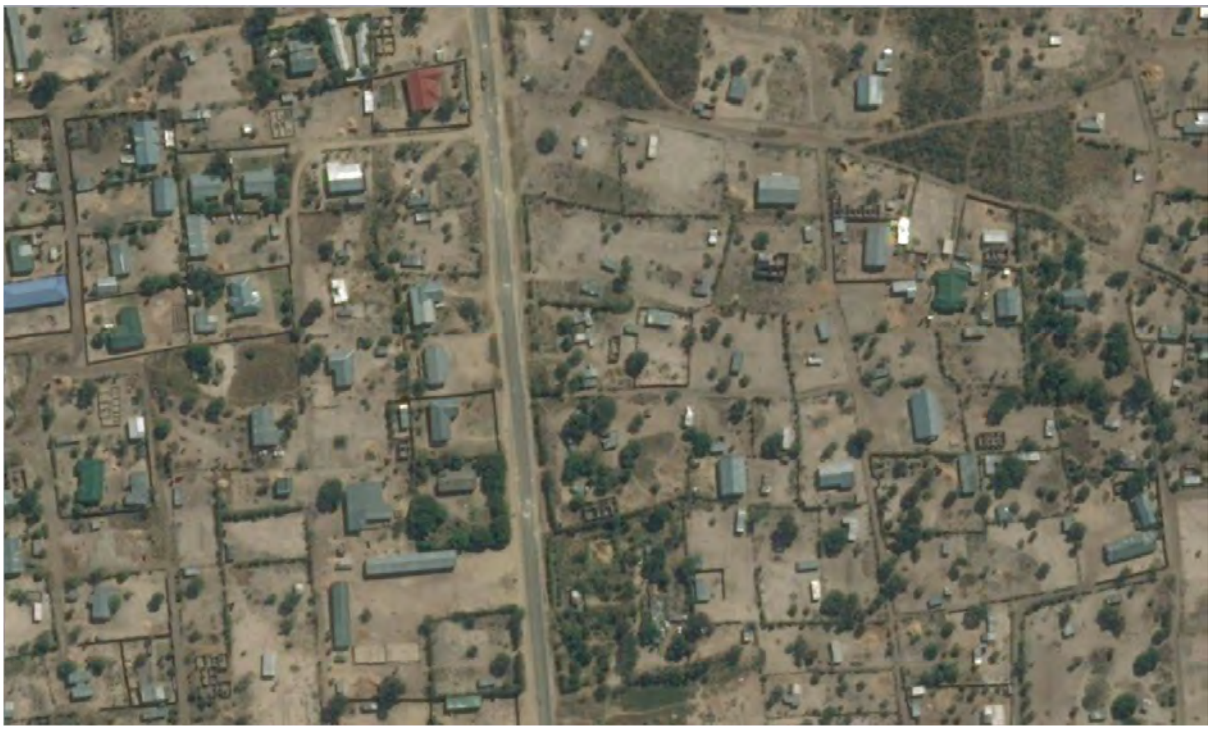


Figure B.3: Sample of Bing Maps satellite imagery at 30 cm resolution of Mongu, Zambia (QGIS schreenshot).

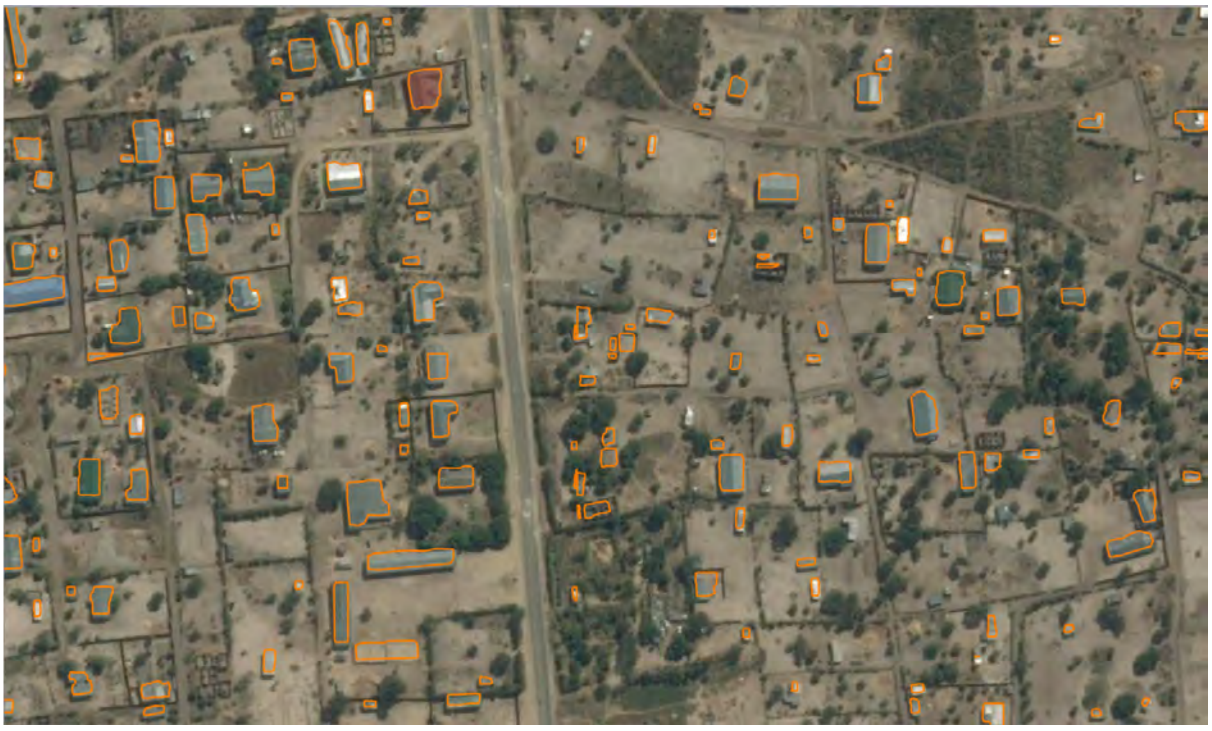


Figure B.4: Sample of vectorized TTA Mask R-CNN predictions on Mongu, Zambia (QGIS screenshot).

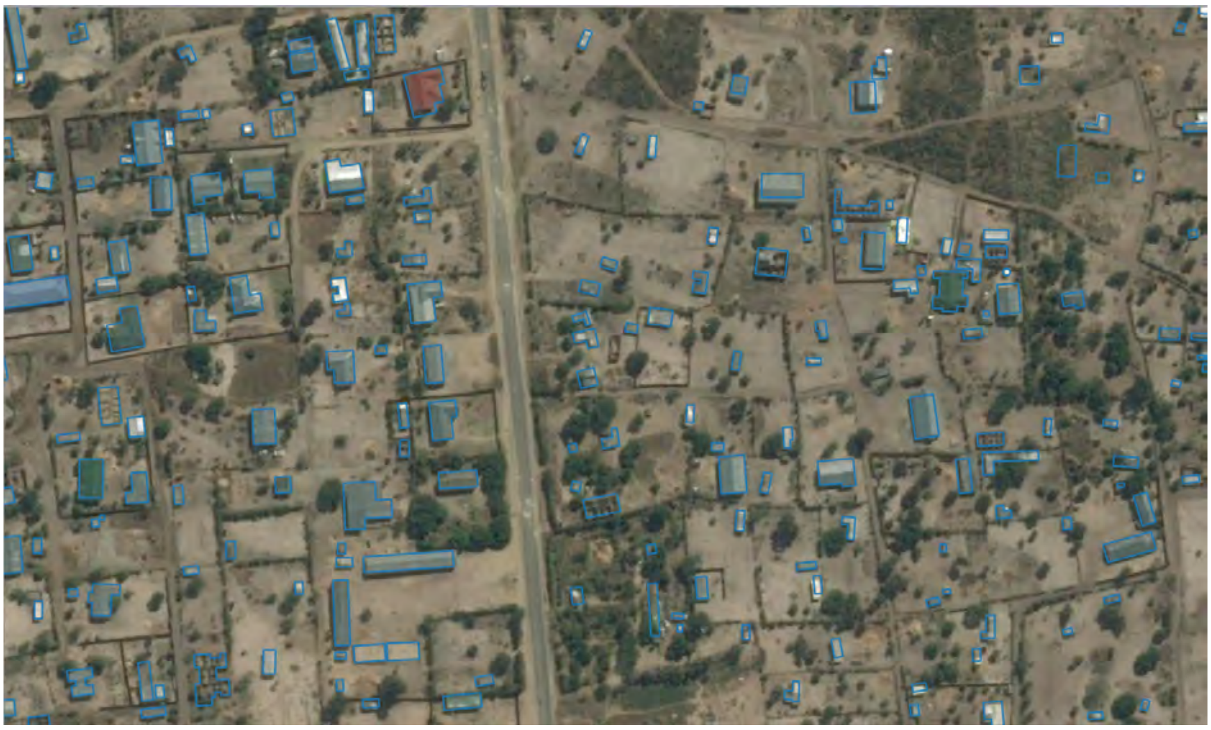


Figure B.5: Sample of OpenStreetMap labels on Mongu, Zambia (QGIS screenshot).

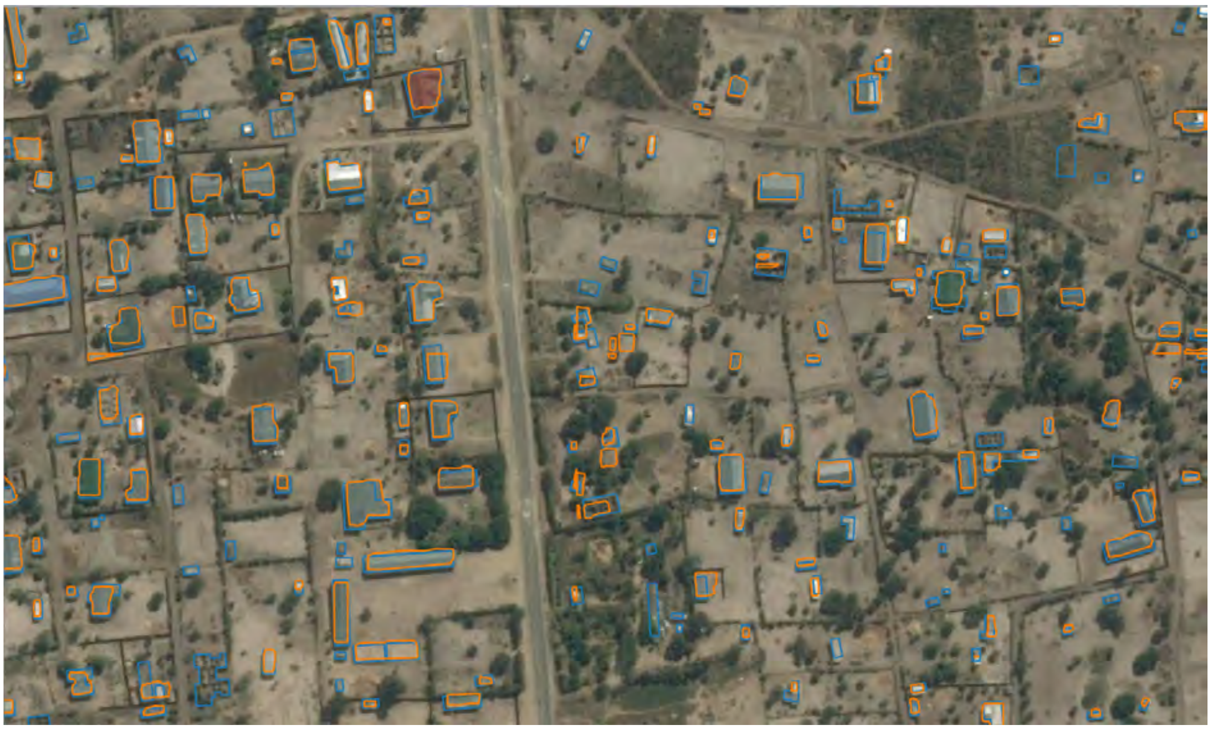


Figure B.6: Sample of vectorized TTA Mask R-CNN predictions (orange outlines) and OpenStreetMap labels (blue outlines) on Mongu, Zambia (QGIS screenshot).

B.4 MASK R-CNN ON THE NETHERLANDS, AMERSFOORT, 10CM

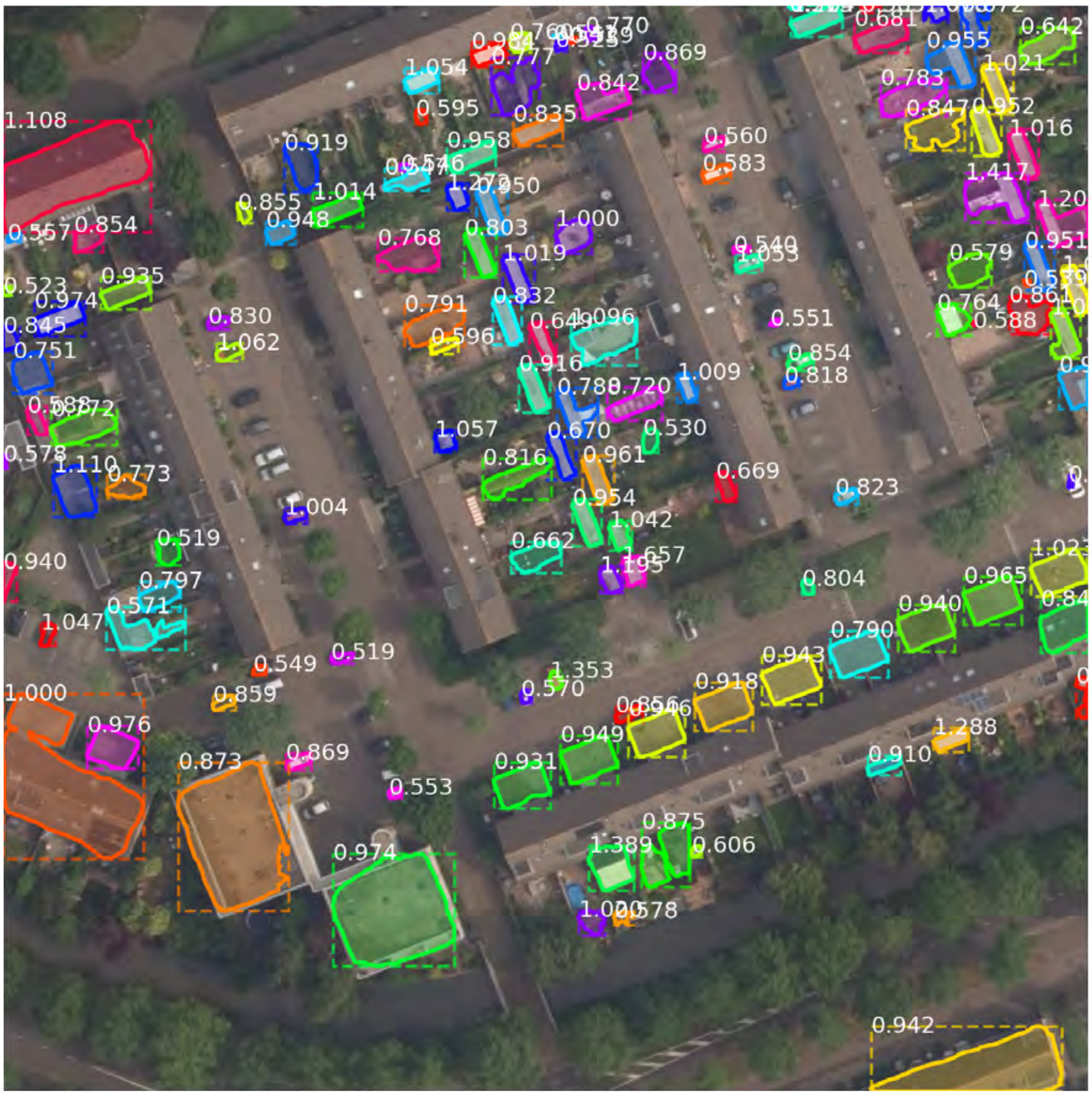


Figure B.7: TTA predictions of Mask R-CNN tested and plotted on native 10 cm resolution imagery of Amersfoort, The Netherlands.

B.5 RESU-NET ON THE NETHERLANDS, HEERLEN, 10 AND 30 CM

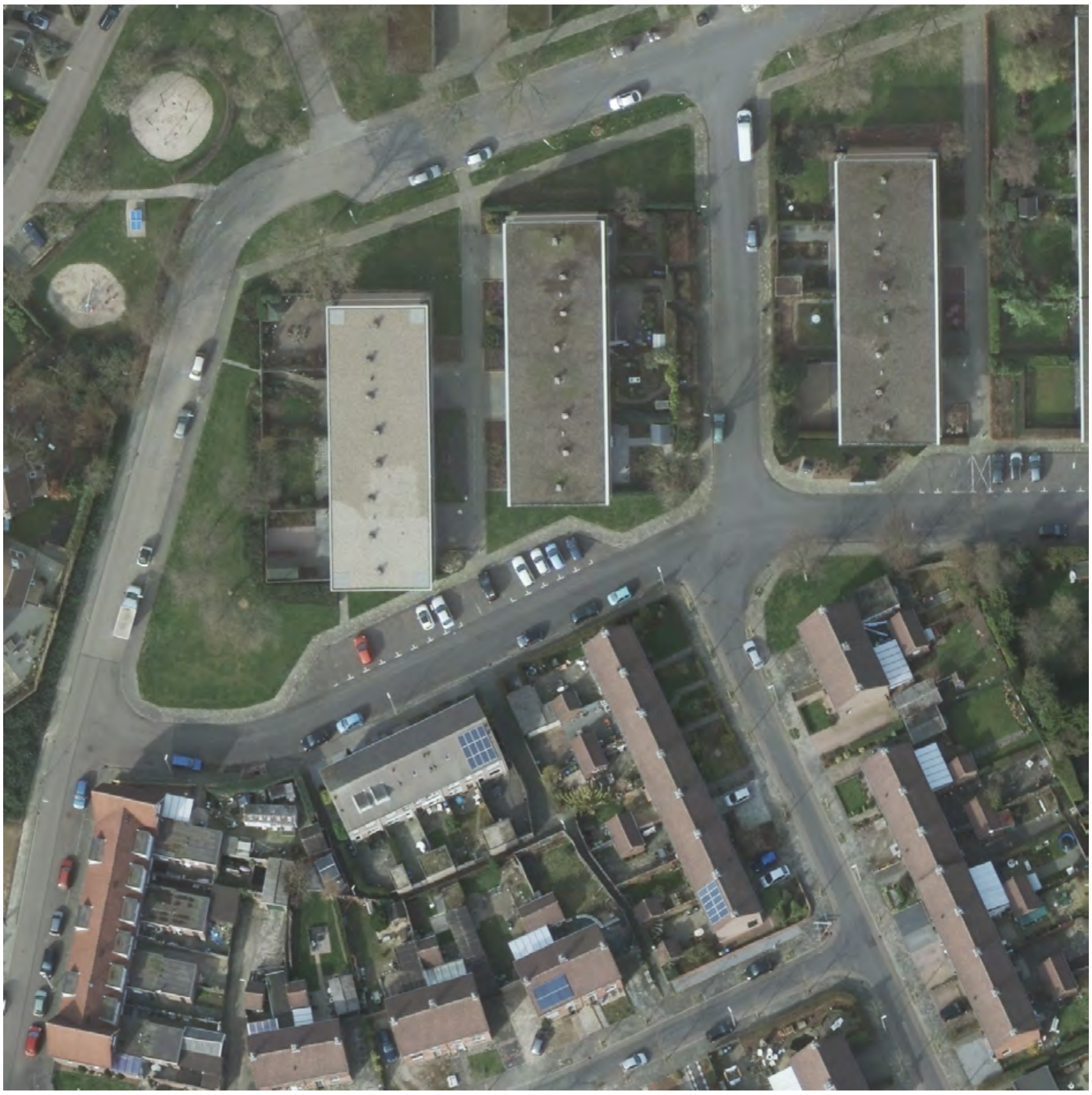


Figure B.8: Sample imagery at 10 cm resolution of Heerlen, the Netherlands.

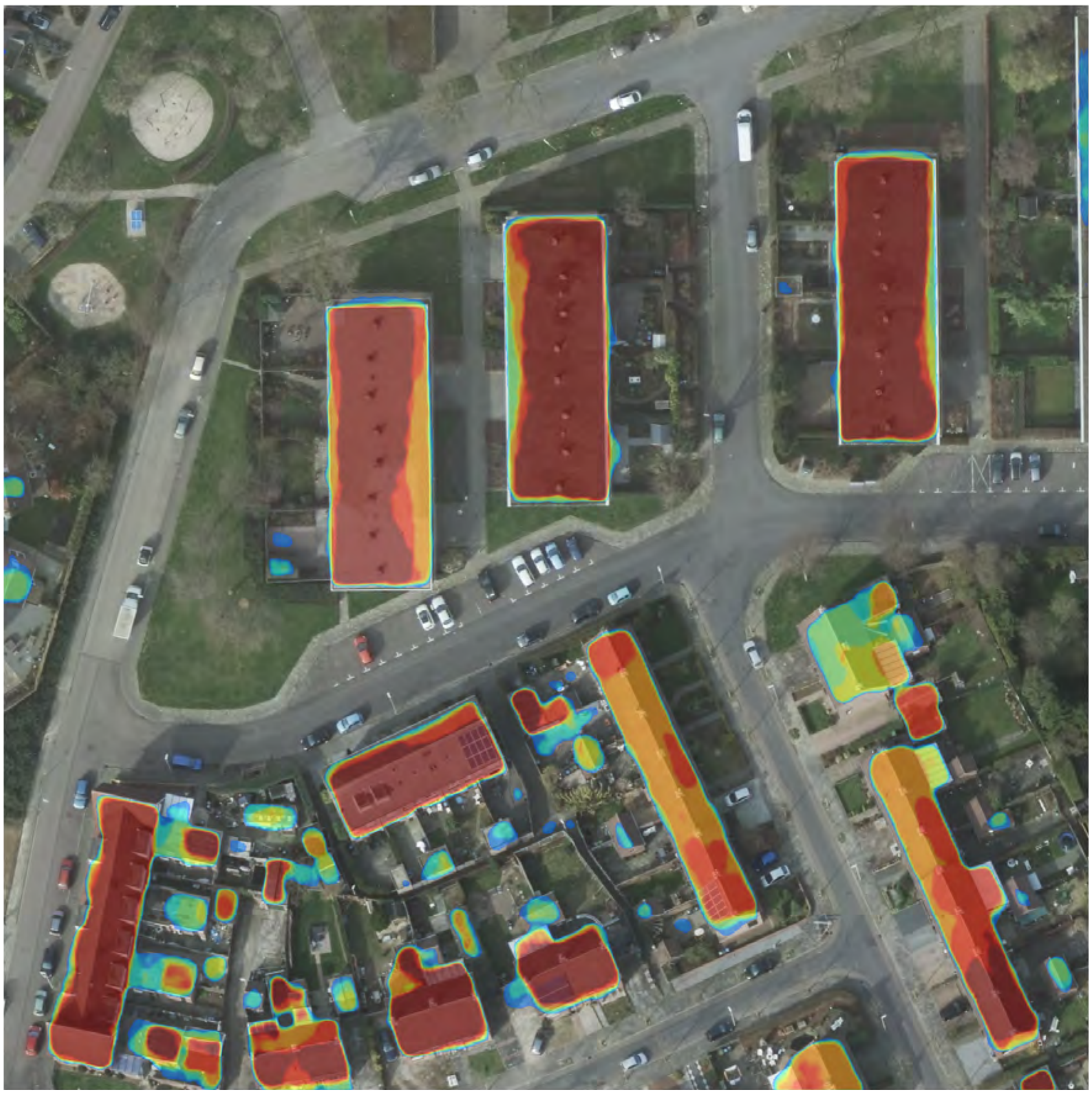


Figure B.9: TTA predictions of ResU-net resolution robustness experiment 1: Trained on 30 cm resolution imagery, tested on 30 cm resolution imagery of Heerlen, The Netherlands. Plotted on 10 cm imagery.

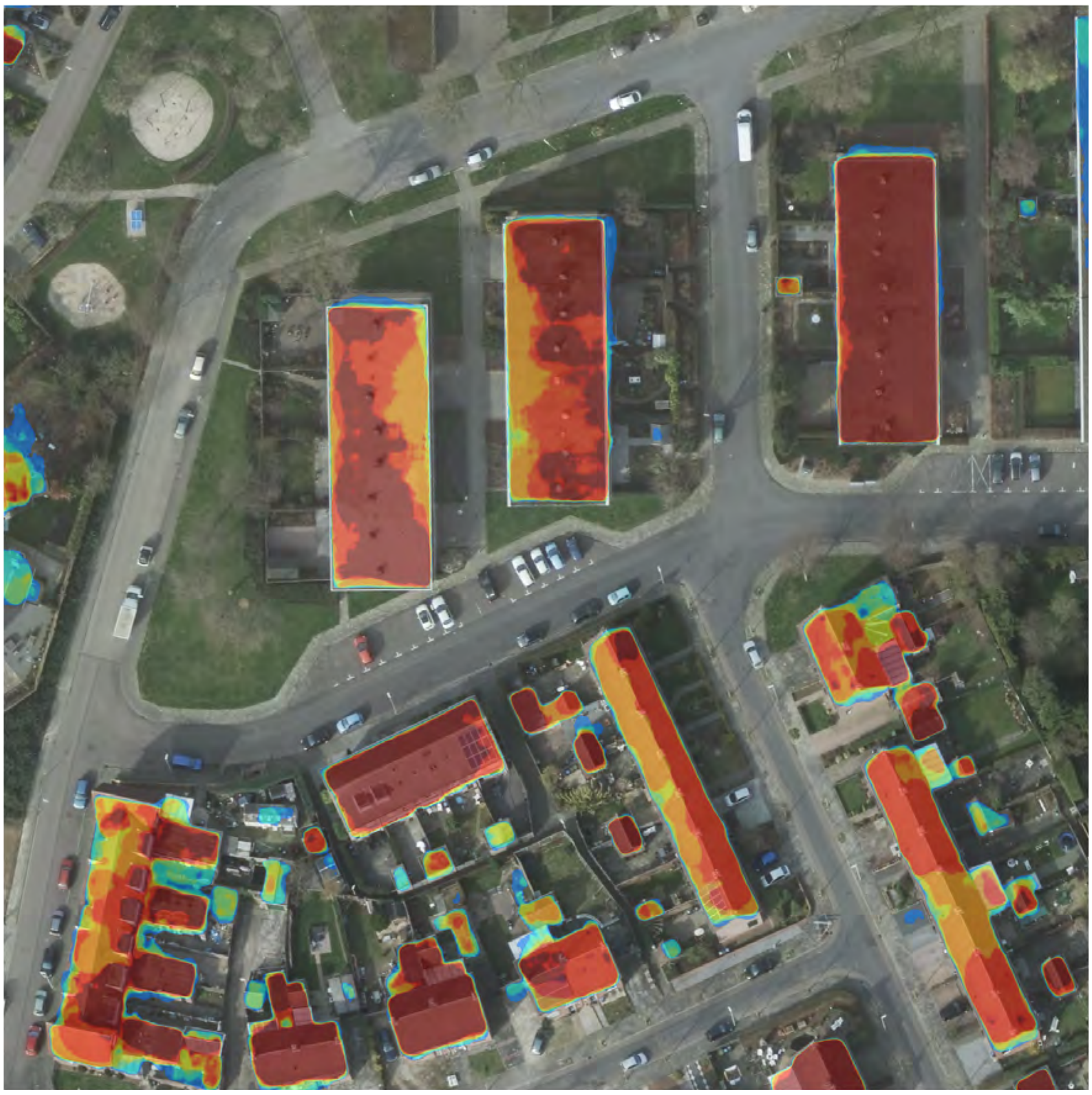


Figure B.10: TTA predictions of ResU-net resolution robustness experiment 2: Trained on 10 cm resolution imagery, tested on 10 cm resolution imagery of Heerlen, The Netherlands. Plotted on 10 cm imagery.

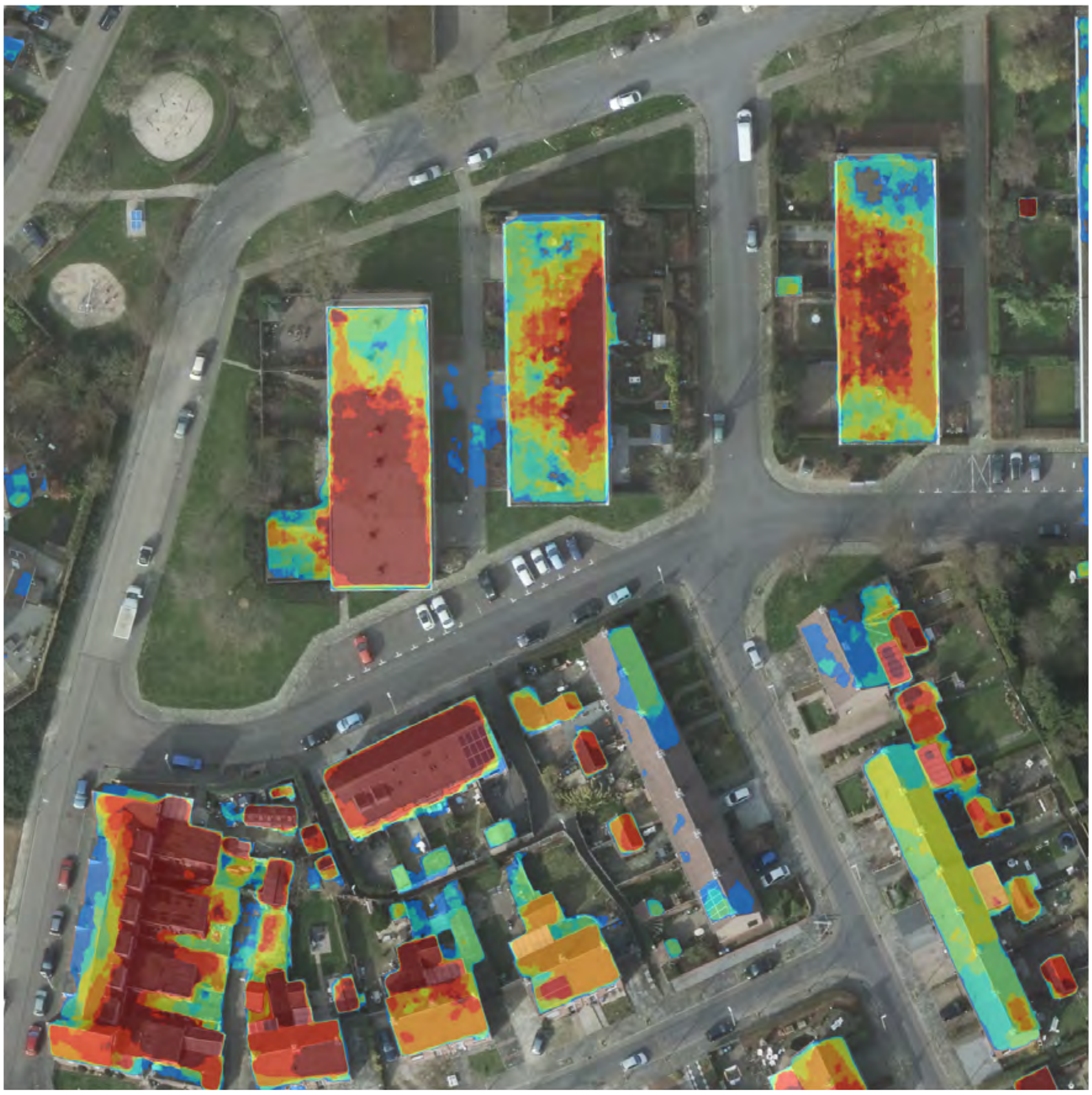


Figure B.11: TTA predictions of ResU-net resolution robustness experiment 3: Trained on 10 cm resolution imagery, tested on 10 cm resolution imagery of Heerlen, The Netherlands. Plotted on 10 cm imagery.

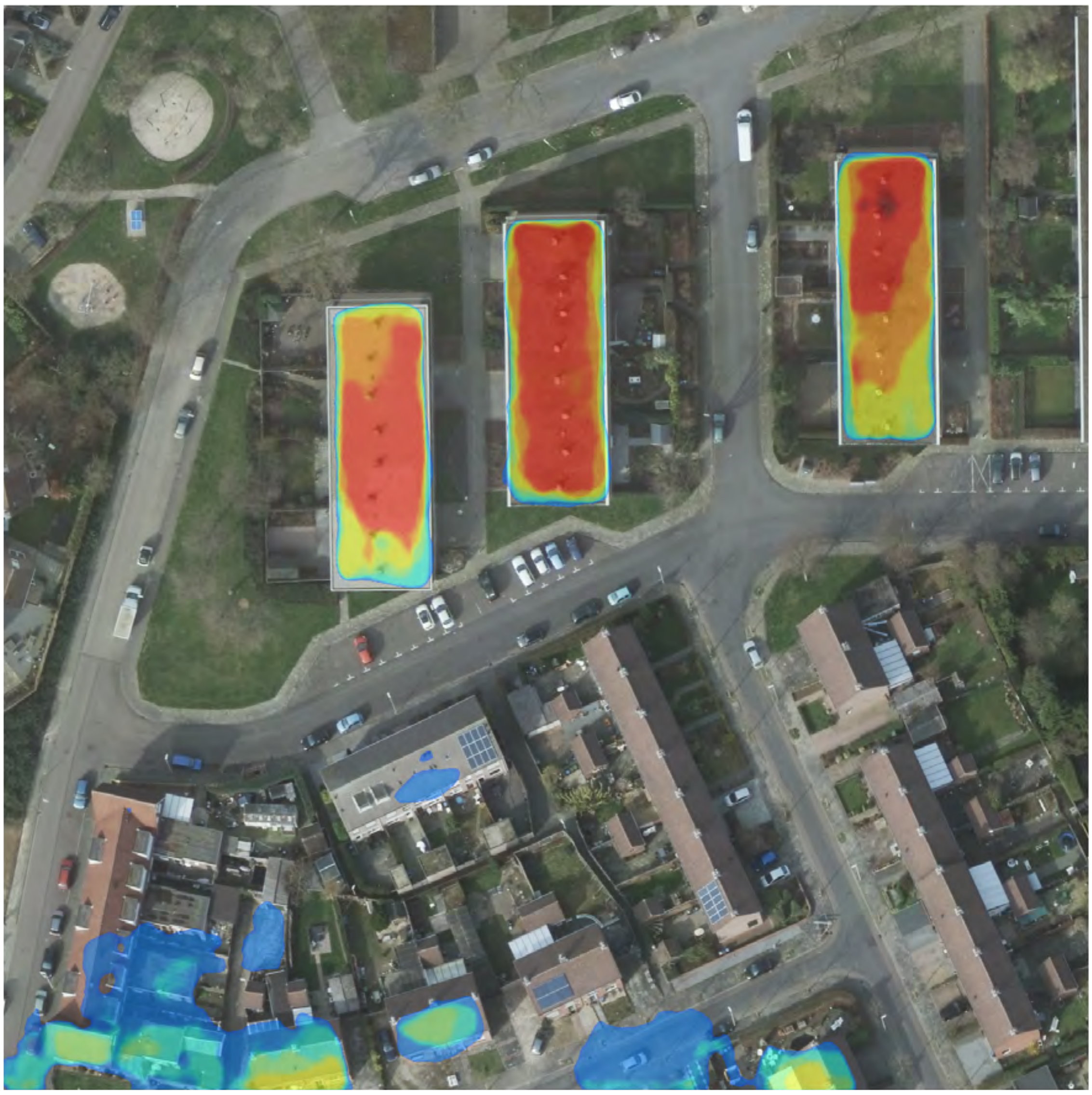


Figure B.12: TTA predictions of ResU-net resolution robustness experiment 3: Trained on 10 cm resolution imagery, tested on 30 cm resolution imagery of Heerlen, The Netherlands. Plotted on 10 cm imagery.

B.6 RESU-NET ON THE NETHERLANDS, HEERLEN, 30 CM

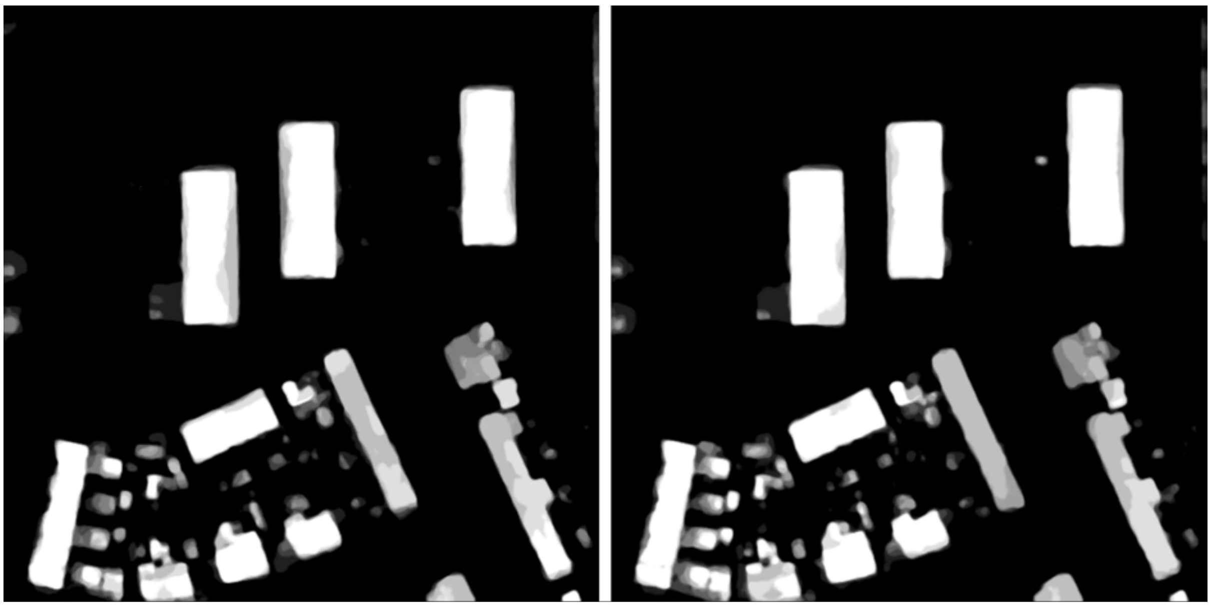


Figure B.13: Sample predictions of Heerlen by ResU-net. Black indicates all negative predictions, white indicates all positive predictions. Left: Predictions of a model trained with orientation augmentations. Right: Predictions of a model trained without orientation augmentations.

B.7 VECTORIZED RESU-NET ON THE NETHERLANDS, AMERSFOORT, 10 CM

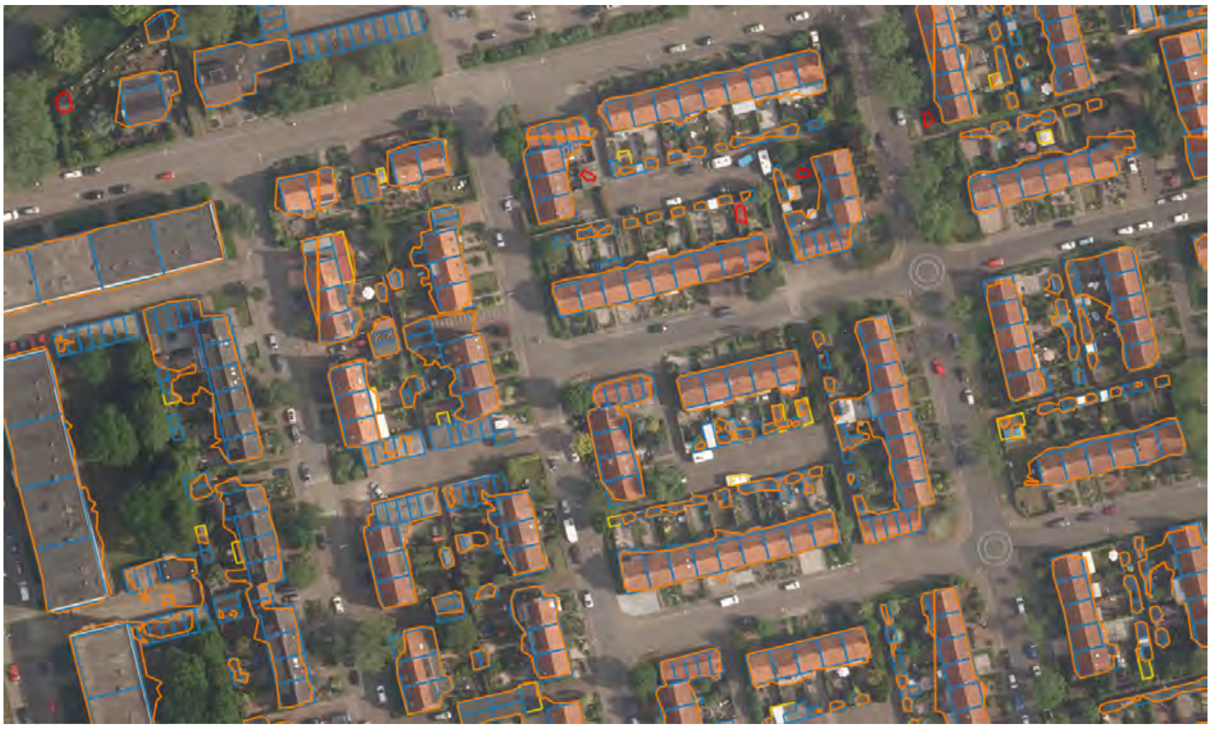


Figure B.14: Vectorized sample predictions on Amersfoort by ResU-net. An orange outline indicates the prediction overlaps at least one building in the ground truth, red indicates no overlap. Blue outlines are buildings in the BAG or BGT database, yellow outlines are manually mapped buildings (QGIS screenshot).

B.8 VECTORIZED RESU-NET ON THE NETHERLANDS, HEERLEN, 10 CM

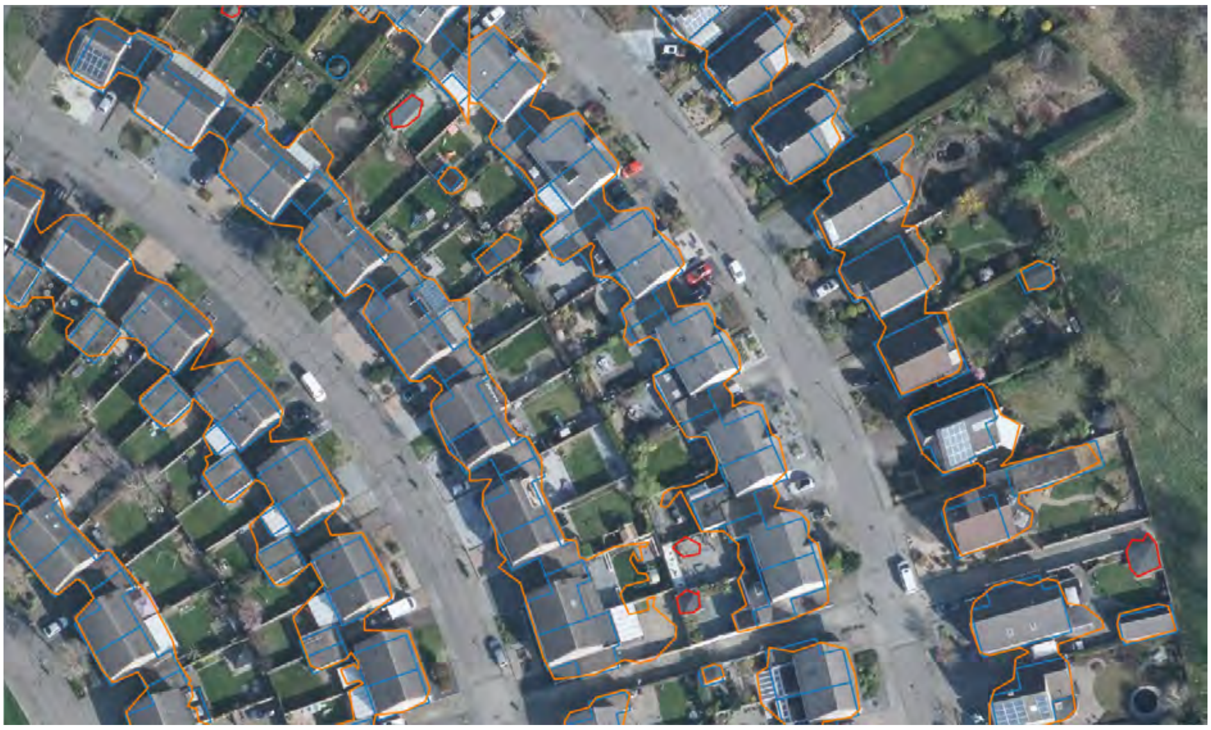


Figure B.15: Vectorized sample predictions on Heerlen by ResU-net. Vectorized sample predictions on Heerlen by ResU-net. An orange outline indicates the prediction overlaps at least one building in the ground truth, red indicates no overlap. Blue outlines are buildings in the BAG or BGT database (QGIS screenshot).

Illustration: Predictions of a (Res)U-net model on Amersfoort, the Netherlands