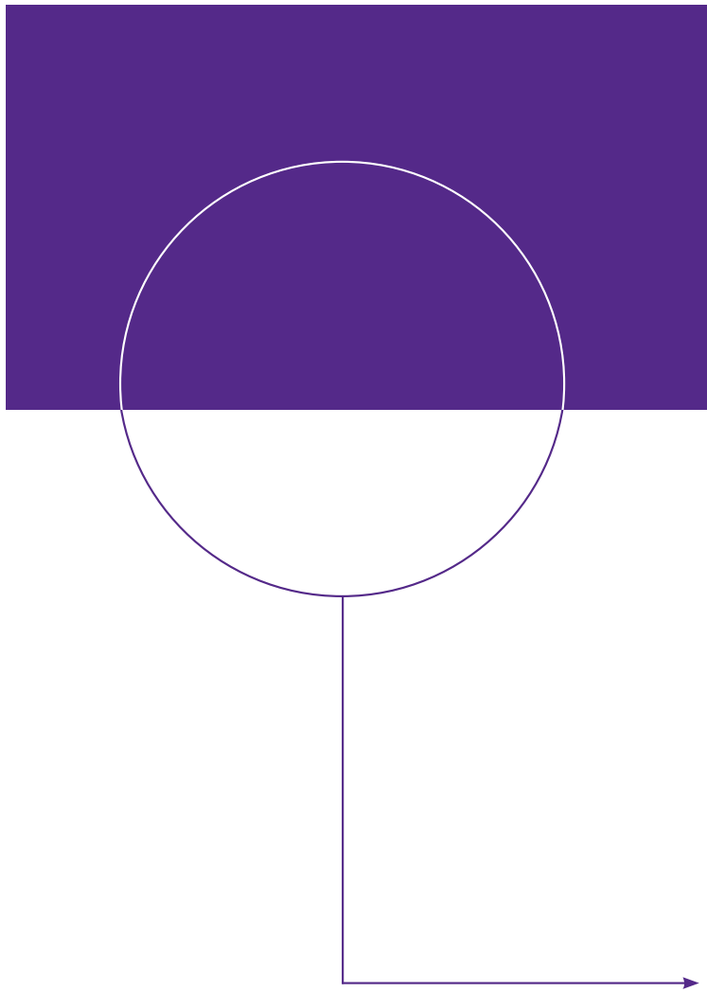


Master's thesis

2022

Katrine Nguyen, Martin Wangen-Eriksen

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering

Katrine Nguyen
Martin Wangen-Eriksen

A Deep Learning Based Approach for Detecting Change of Buildings from Aerial Images

June 2022

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

A Deep Learning Based Approach for Detecting Change of Buildings from Aerial Images

Katrine Nguyen
Martin Wangen-Eriksen

Engineering and ICT
Submission date: June 2022
Supervisor: Hongchao Fan

Norwegian University of Science and Technology
Department of Civil and Environmental Engineering

Preface

This master thesis completes our civil engineering degree in Engineering and ICT within the field of Geomatics. The master thesis is written for the Department of Civil and Environmental Engineering at the Norwegian University of Science and Technology in Trondheim, Norway, in Spring 2022.

We would like to thank our supervisor Hongchao Fan at NTNU, for his motivation during the project and invaluable encouragement. Our thesis have greatly benefited from your help, guidance, insight, and engagement. We will also thank Ph.D. candidate Gefei Kong for the support and guidance in the field of Deep Learning and Neural Networks.

We are also grateful to Trondheim municipality for providing us with Aerial images of Trondheim and the motivation of contextualizing our task by emphasizing its importance in cartography.

Our fellow students and family also deserve thanks for incredible encouragement, support, and help throughout our studies, especially this last semester. A special thanks go to Dr. Minh Nguyen, Katrine Nguyen's father, for proofreading expertise and improving our academic writing skills for this thesis.

Abstract

The rise of deep learning approaches for building and change detection have increased in the recent years. The accessibility and quality of high resolution satellite images have opened a wide variety of possibilities for practical applications and business problems. Temporal change detection using high resolution satellite images, have played an important role in mapping different changes on Earth's surface. Due to increasing interest in building and change detection, a range of different approaches in this field have become readily available. This thesis introduces a novel neural network, U-PSP-Net, specifically designed for building detection. The network is used to detect changes of buildings in Trondheim Municipality. The thesis also covers the creation of a novel building data set covering 2.3 square kilometers and 3784 buildings.

U-PSP-Net is a convolution neural network made by combining parts from U-Net and PSPNet, two other image segmentation networks. An ablation study was performed to optimize the network, exploiting the advantages of each network while also minimizing their weaknesses. The resulting network is four layers deep with a pyramid pooling module, consisting of the two smallest kernels, connecting the encoder and decoder structures. Reducing the depth makes U-PSP-Net a more efficient network, using only half as many weights as U-Net. The network achieved almost 98% recall and precision and a IoU of almost 97 %.

As part of the thesis, an experimental study in Trondheim was conducted. The goal was to use U-PSP-Net to, as accurately as possible, extract buildings from aerial images. The network was used to extract buildings from two data sets. Despite the high accuracy of the results, some misclassifications were made on the data sets. The classified images were later used for change detection analysis. The analysis encounters issues caused by differences in the data sets due to variation in the image collection phase. In conclusion, the U-PSP-Net proved to be efficient at building detection. However, the training data set could be expanded, and the images should be exposed to more augmentation. The change detection method was also proven to work, although, for automatic detection, true orthophotos are required.

The complete source code developed for this thesis can be found at this GitHub repository

Sammendrag

Fremveksten i bruken av dyplæringsmetoder for bygning- og endringsdeteksjon har økt jevnt de siste årene. Tilgjengeligheten og kvaliteten til høyoppløselige satellittbilder har tilrettelagt for en rekke muligheter innenfor praktiske bruksområder og forretningsproblemer. Deteksjon av endringer ved hjelp av satellittbilder med høy oppløsning har hatt en viktig rolle i å kartlegge endringer på jordas overflate. På grunn av økende interesse for bygning og endringsdeteksjon, har en rekke forskjellige tilnærminger på dette feltet blitt lett tilgjengelige. Denne oppgaven introduserer et nytt nevralt nettverk, U-PSP-Net, spesielt designet for bygningsdeteksjon. Nettverket brukes til å oppdage endringer av bygninger i Trondheim kommune. Oppgaven dekker også etableringen av et nytt bygningsdatasett som dekker 2,3 kvadratkilometer og 3784 bygninger.

U-PSP-Net er et konvolusjonalt nevralt nettverk laget ved å kombinere deler fra U-Net og PSPNet, to andre bildesegmenteringsnettverk. En ablasjonsstudie ble utført for å optimere nettverket, og å utnytte fordelene ved hvert nettverk samtidig som de minimerte deres svakheter. Det resulterende nettverket er fire lag dypt med en pyramidepoolingsmodul, bestående av de to minste kjernene, som forbinder koder- og dekoderstrukturene. Redusering av dybden gjør U-PSP-Net til et mer effektivt nettverk, som bruker bare halvparten så mange vekter sammenlignet med U-Net. Nettverket oppnådde nesten 98% dekning og presisjon og en IoU på nesten 97%.

Som en del av oppgaven, ble det gjennomført en eksperimentell studie i Trondheim. Målet var å bruke U-PSP-Net til å så nøyaktig som mulig kunne detektere bygninger fra flybilder. Nettverket ble brukt til å detektere bygninger fra to ulike datasett. Resultatene var svært nøyaktige, men med noen feilklassifiseringer på det ene datasettet. De klassifiserte bildene ble senere brukt til analyse av endring. Under analysen, oppstod det problemer forårsaket av forskjeller i datasettene på grunn av variasjon i bildeinnsamlingsfasen. For å konkludere, viste U-PSP-Net seg å være effektivt ved bygningsdeteksjon, men treningsdatasettet burde utvides og bildene bør eksponeres for mer behandling. Metoden for endringsdeteksjon har også vist seg å fungere, selv om det kreves sanne ortofoto for en automatisk deteksjon.

Den komplette kildekoden utviklet for denne oppgaven ligger på GitHub.

Table of Contents

List of Figures viii

List of Tables x

1 Introduction 1

1.1 Change Detection . . . 2

1.1.1 Classifying Building Changes . . . 2

1.1.2 Object-Based Detection . . . 3

1.1.3 Pixel Based Detection . . . 4

1.1.4 Change Detection Conclusion . . . 5

1.2 Our Main Contribution . . . 6

1.3 Motivation of Creating a new Data Set . . . 6

1.4 Structure of the Thesis . . . 8

2 Background 9

2.1 Building Detection . . . 10

2.1.1 Unsupervised . . . 12

2.1.2 Supervised . . . 12

2.1.3 Region-Based Segmentation . . . 13

2.1.4 Deep Learning . . . 14

2.1.5 Conclusion of Literature Study . . . 14

2.2 Convolution Neural Network . . . 15

2.2.1 Elements of Convolutional Neural Networks . . . 15

2.2.2 Hyperparameters . . . 19
2.2.3 Image Augmentation . . . 22
2.3 U-Net . . . 22
2.3.1 Architecture . . . 22
2.3.2 Advantages and Disadvantages with U-Net . . . 24
2.3.3 Visualization . . . 25
2.4 PSPNet . . . 26
2.4.1 Architecture . . . 26
2.4.2 Advantages and Disadvantages with PSPNet . . . 27
2.4.3 Visualization . . . 27
2.4.4 Ablation Study . . . 28
2.5 Conclusion . . . 29

3 Method **30**

3.1 Model . . . 31
3.1.1 Network Structure . . . 32
3.1.2 Encoder and Decoder Module . . . 32
3.1.3 Pyramid Pooling Module . . . 33
3.1.4 U-PSP-Net . . . 33
3.2 Method for Change Detection . . . 34
3.2.1 Creating a Network . . . 34
3.2.2 Training, Optimizing and Applying the Network . . . 34
3.2.3 Change Analysis . . . 35
3.3 Evaluation . . . 37
3.3.1 Precision and Recall . . . 37
3.3.2 Intersection over Union . . . 37
3.3.3 Training and Validation Loss . . . 38
3.3.4 Time and Space . . . 38

4 Data Set over Trondheim Municipality **39**

4.1 General information . . . 40

4.1.1 Camera and Flight . . . 40

4.1.2 Processing . . . 41

4.1.3 Orthorectification . . . 41

4.1.4 Radiometric processing . . . 41

4.2 Preprocessing . . . 42

4.2.1 Area of Interest . . . 42

4.2.2 Preprocessing and Data set . . . 43

4.2.3 Cropping . . . 43

4.2.4 Software . . . 43

4.2.5 Annotating . . . 44

4.2.6 Creating masks . . . 46

4.2.7 Image augmentation . . . 46

4.2.8 Patching . . . 47

5 Experimental Study in Trondheim **49**

5.1 Ablation Study . . . 50

5.1.1 Evaluation during the Ablation Study . . . 50

5.1.2 Results of Ablation Study . . . 51

5.2 Hyperparameter Optimization . . . 52

5.2.1 Batch Size . . . 52

5.2.2 Learning Rate and Reduction of the Learning Rate . . . 52

5.2.3 Loss Function and Optimizer Function . . . 52

5.2.4 Epochs and Early Stopping . . . 53

5.2.5 Visualization and Measurement tool . . . 53

5.3 Our Results . . . 53

5.3.1 U-Net . . . 54

5.3.2 U-Net + Pyramid Pooling Module . . . 55

5.3.3 U-PSP-Net . . . 56
5.3.4 Comparison of the models . . . 56
5.3.5 Parameters . . . 57
5.3.6 Building Detection . . . 58
5.3.7 Change Detection . . . 60
5.3.8 Data Set . . . 63

6 Conclusion **66**

6.1 Conclusion . . . 66
6.2 Further Work . . . 67

Bibliography **69**

Appendix . . . 76
A Kartblad Trondheim . . . 76
B Results . . . 77
C Building Detection Results . . . 79
D Building Detection, Irregular shapes . . . 81
E Difference in Color Properties . . . 82
F Change Detection Results . . . 83
G Area . . . 84

List of Figures

2.1 Buildings of various types: (a) Single buildings, (b) Building blocks, (c) Buildings of different sizes, (d) Buildings of different elevations. [29] 10

2.2 Building extraction using deep learning semantic segmentation. [30] 11

2.3 Multiple layers corresponding to a different filter, but the same spatial reference [54] 16

2.4 Common types of non-linearity [54] 18

2.5 Fully Connected Neural Network vs. Convolutional Neural Network 19

2.6 Fully connected layer, each node corresponds to a feature. 19

2.7 The 18 most important hyper-parameter subsets of Cohn-Kanade data set [63]. . . 20

2.8 The graphs above show examples of learning curves. The blue graph indicates training error, and the orange graph indicates validation error. From left to right, a model that requires further training, an overfit model, and a perfectly fitted model. [66] 21

2.9 U-Net architecture [73]. 23

2.10 Max-pooling of an 4 x 4 image with a window size 2 x 2 and stride 2 [75]. 23

2.11 U-Net predictions on aerial imagery over Dubai's landscape. The prediction is compared to ground truth and original image. [78] 25

2.12 PSPNet architecture [80]. 26

2.13 PSPNet predictions compared to ground truth and original image. [84] 28

3.1 Our model combined by U-Net and PSPNet, now called U-PSP-Net. 32

3.2 Flowchart describing our proposed method for change detection. 34

4.1 Illustration showing how the data was collected [90]. 40

4.2 The eight most common roof types in Europe [93]. 44

4.3 Overview over the data set distribution. 45

4.4 Roof type annotation using Labelbox. Different colors represent different roof types, where orange represent *Hipped* and pink *Shed*. 48

5.1 Training and validation loss graph for U-Net. 54

5.2 Training and validation loss graph for U-Net with Pyramid Pooling Layers. 55

5.3 Training and validation loss graph for U-PSP-Net. 56

5.4 Struggling with detecting image in the edge of an image. 58

5.5 Small anomalies in the prediction. 59

5.6 Overall image of change detected of a larger area. 60

5.7 Relief displacement of images from 2019 and 2020. 61

5.8 Change detected. Red shows parts that are removed, and green are parts added. . 62

5.9 Aerial images from 2019 and 2020, and the change detected. 63

1 Trondheim 76

2 Buildings detected from different areas in Trondheim 79

3 Buildings detected from different areas in Trondheim 80

List of Tables

4.1 Camera constants of the camera used for the aerial images 40

4.2 Numbers of labeled roofs in the different roof type categories 44

4.3 Numbers of labeled roofs in the different roof type categories. 45

5.1 Results of U-PSP-Net during ablation study were different Pyramid Pooling layers from PSPNet are included. 51

5.2 Results of U-PSP-Net with different depths. 51

5.3 Result of U-Net to 27^{th} epoch. 54

5.4 Result of U-Net with Pyramid Pooling Module from PSPNet to 27^{th} epoch. . . . 55

5.5 Result of U-PSP-Net to 27^{th} epoch. 56

5.6 Comparison of U-Net, U-Net with pyramid pooling modules and our U-PSP-Net after 27 epochs. 57

5.7 Parameter comparison of U-PSP-Net and U-Net. 57

Acronyms

AI Artificial Intelligence. 28

ANN Artificial Neural Network. 15

CNN Convolution Neural Network. 8, 9, 15, 22, 34

CVA Change Vector Analysis. 4

FCN Fully Convolutional Network. 27

GAN Generative Adversarial Network. 64

GIS Geographic Information System. 5

IoU Intersection over Union. 37

KT Kauth–Thomas transformation. 5

ML Machine Learning. 28

OA Overall Accuracy. 4

OBCD Object Based Change Detection. 3

PCA Principal Component Analysis. 5, 12

PPL Pyramid Pooling Layer. 51

PPM Pyramid Pooling Module. 8, 26, 27, 30–33, 49, 56, 57, 66

ReLU Rectified Linear Units. 17, 23

VHR RS Very High-Resolution Remote Sensing. 2

Chapter 1

Introduction

Problem Definition

There are many reasons detecting building changes is imperative, such as for urban management, disaster assessment, illegal building identification, and updating geographic information databases [1]. The ability to detect buildings is a crucial component of cartography, and it has been used extensively in many applications. Economic development and analysis of urban expansion, for instance, can supply urban planners with relevant and crucial information for emergency management. Building changes can also be detected and used to update maps when buildings are expanded or demolished.

The fact that high-resolution aerial imagery is available makes efficient building detection achievable. Currently, remote sensing information is primarily extracted manually through visual inspection, which is time-consuming, tedious, and extremely dependent on becoming automated. Automating and semi-automating building change detection is becoming increasingly popular because of the potential productivity increase. [1]

1.1 Change Detection

Change detection is by Singh [2] described as *"the process of identifying differences in the state of an object or phenomenon by observing it at different times"*. Deforestation, damage assessment, disaster monitoring, urban expansion, planning, and land management are just a few of the various applications of change information. Research is being conducted to develop methods for detecting changes in remote sensing. Change detection relies on remote sensing data because changes in the object of interest alter the spectral behavior (local texture or reflectance value) independent of other factors [3]. Many factors affect change detection in remote sensing data, including spatial, spectral, thematic, and temporal constraints, radiometric resolution, atmospheric conditions, and soil moisture conditions [4].

Change detection procedures are intended to detect and, if possible, interpret changes in objects or phenomena over different acquisition times $t_1, t_2, t_3, ...t_n$. To determine the degree of change, remote sensing image data can be compared with the value of an image pixel or object time t_1 and t_2 based on multi-temporal remote sensing image data. Various digital image processing methods, depending on the image's resolution, spectral properties, and temporal characteristics, have been developed throughout the last 30 years. [5]

Determining the most suitable algorithm or approach for change detection is a significant challenge in practice [6]. Researchers have expended great effort to develop different change detection methodologies that cover both traditional pixel-based and object-based methods [3]. The traditional pixel-based change detection methods are not considered appropriate for Very High-Resolution Remote Sensing (VHR RS) data, which object-based approaches may be more effective. In addition to the pixel- and object-based approaches, various algorithms have been developed to provide a wider range of choices. [7]

Change can be detected through a bi-temporal building map or by using two images in a change detection network. Bi-temporal building map is used by Ji et al. [8] as input into the change detection network, where the building extraction networks are used to produce binary maps of buildings and backgrounds. Change detection is based on concatenating maps of different periods, then incorporating them into the change detection network to filter out noise caused by the imperfect building extraction [8]. Due to complex combinations of factors such as structural changes and varying lighting conditions from different vantage points, it is hard to determine changes in buildings from bi-temporal images [9].

1.1.1 Classifying Building Changes

When detecting building changes, the changes can be organized. Within our task, four different building changes can be classified. New buildings, expanded buildings, demolished buildings, and partially demolished buildings. A new building is a separate building, while an expanded building is connected to an already exciting one. Demolishment of a building can either be total or partial.

1.1.2 Object-Based Detection

Increasing advances in computing hardware and algorithms have enabled accurate segmentation and classification of very high-resolution satellite images. The increasing advances have enabled a new method for detecting changes, object-based change detection. [10]

In essence, Object Based Change Detection (OBCD) is based on object-based image detection. OBCD is a development from object-based image analysis, which combines segmentation, spatial, spectral, and geographic data with the competence and experience of the analyst to create models of geographic entities from image files [11] [12]. To detect objects in images, pixels are grouped to form groups of pixels. Object detection is a whole field in its own right. In OBCD, objects are identified in two or more images, and changes in the same area are detected [10]. According to Mäkelä and Pekkarinen [13], image-object segmentation is less sensitive to misregistration errors compared to the traditional pixel-based methods.

Visual Inspection

Depending on the resolution of the images in question, detecting change can be seen as both pixel-based and object-based change detection. According to Lu et al. [6], visual analysis is helpful for a task where size, shape, texture, and patterns are vital elements. It is beneficial with a visual inspection in situations where the resolution is coarse. For example, when detecting a change in forest cover over larger areas or the development of weather phenomena, a visual inspection might be considered more effective than any other method. To this date, the municipality of Trondheim still uses visual inspection for change detection. One example of this is the same task this master's thesis wants to solve. The municipality of Trondheim uses manual labor, outsourced to other countries, to detect changes in buildings. This process is very accurate but, on the other hand, immensely time-consuming. An inaccurate map serves no purpose. Hence the municipality of Trondheim still uses manual labor for change detection. It might be some time before this process can become entirely automatic. Nevertheless, more and more, the process will take advantage of machine learning to save time and money.

Classification

Methods that use classified images to detect change are often known as classification methods. This category includes methods like post-classification comparison, unsupervised change detection, expected maximization algorithm, and artificial neural networks. A big advantage of these methods is that they provide a change detection matrix. The methods can also minimize the effect of atmospheric and environmental differences between the images. As with any method, there are also drawbacks. These methods heavily rely on training data. An accurate data set, sufficient in size, is required to achieve adequate change detection. Which is both time-consuming and difficult to create. [6]

Direct change detection

Although not a widespread method, it is possible to detect change directly. Deep learning networks can be trained to detect virtually anything as long as there is training data. By feeding a network sets of bi-temporal images with annotated changes, the network can learn to detect change. Wang et al. [14] proposes a method using a novel network, named GETNET, to detect change directly. The network is tested and used to identify changes in land cover and geographical objects like rivers or lakes. The proposed network achieved well above 90% Overall Accuracy (OA). Although detecting change directly is not the most common method for change detection right now, this might change in the future as networks and training data sets improve [14].

1.1.3 Pixel Based Detection

Pixels have been fundamental to image analysis since the beginning of remote sensing. The spectral properties of pixels are exploited in these techniques, allowing measurements and detecting changes independent of their spatial arrangement. In general, remote sensing images can be categorized into simple binary changes or detailed "from-to" changes. Several change analysis studies require detailed information on the "from-to" change, but a binary change versus no change will suffice. [3]

According to studies performed by Cleve et al. [15], Corcoran and Winstanley [16] and Hájek [17], object-based methods produce more accurate and robust classifications using high-resolution imagery than using pixel-based methods. However, for certain land cover categories, pixel-based land cover classification techniques have shown to be the most effective classification technique [18]. It may be beneficial to combine the best classification results from both methods. An analysis by Wang et al. [19] applied very high-resolution (VHR) IKONOS imagery to the classification of a mangrove ecosystem showed that combinations of pixel-based and object-based classification improved the classification accuracy.

Various accuracy assessments are commonly used in pixel-based change detection, including overall accuracy, producer accuracy, user accuracy, and kappa coefficient [6]. Using pixels as the basic unit for assessment makes it reasonable to evaluate the change results based on pixels-level truth data. [10]

Algebra Methods

Change Vector Analysis (CVA), image differencing, background subtraction, vegetation index differencing, image rationing, and image regression are some of the methods classified as algebra methods. All these methods share the same approach, in which they apply mathematical operations on each pixel to detect change [20]. A drawback of these methods is only allowing processing of one band of data at a time and the task of choosing a suitable threshold. Results heavily depend on the threshold; therefore, using an appropriate threshold is crucial [6] [2]. Of

the methods already mentioned, image differencing is the most commonly used.

Transformation Methods

The use of transforms on image pixels to detect changes in an image falls under the category of change detection. The most common of these transform methods are Gramm-Schmidt and Chi-squared transform, Principal Component Analysis (PCA), and Kauth–Thomas transformation (KT). The essential advantage of employing a transform-based change detection method minimizes redundant information between the bands. On the other hand, a disadvantage of using this method is the lack of a detailed change matrix. Similar to the algebra methods, it requires a threshold, which can be challenging to set. Since the images are transformed, identifying and labeling the change can be complicated.

Although these methods have some flaws, they are simple and easy to use. Before the turn of the millennium, transformation methods were popularized and used to detect change. Due to the fact that they are not complex and give satisfactory results, PCA and KT were the most popular [6].

Geographic Information System (GIS)

Integrated GIS and remote sensing methods is another way to detect change using GIS. Contrary to other methods, you can incorporate different sourced data when using a GIS. A problem that must be avoided with additional sourced data is the difference in accuracy and format which will affect the result if it is not taken into consideration [6].

1.1.4 Change Detection Conclusion

Change detection is historically more used for land-use and land-cover changes. This is a field where extreme accuracy is not essential to get satisfactory results, and the difference is possible to detect even without actual high resolution. The change detection area of objects like buildings is less available than building detection. Change detection of things like buildings is mainly done by comparing bi-temporal images that are already classified — a so-called post-classification change analysis. Change detection is therefore done pixel-based by comparing masks containing said objects. Understanding this method clarifies that the change detection accuracy is directly related to the building detection. The comparison method can not affect the results or the accuracy. It is, however, only a method to be able to visualize and contextualize the change. Hence the actual task of building change detection is to classify the buildings as accurately as possible. This thesis will primarily focus on building detection methods and how to get the best possible results from building detection. Furthermore, the experiment optimizes existing building detection methods to improve their results.

1.2 Our Main Contribution

Our task in this master thesis has been to develop our deep learning network based on two state-of-the-art approaches for image segmentation. The report consists of a literature review of the excising methods for building detection, annotating methods for creating training data, as well as an ablation study for constructing our model. Apart from explaining some of the latest and advanced algorithms within this field, the report evaluates the results from our model and compares it to already existing image segmentation approaches.

Performing a literature study and an experiment in our Specialized project with U-Net and SegNet in advance of our master thesis gave us a fundamental insight into how the various state-of-the-art approaches within this field behaved. U-Net gave the best results in the experiment, which is why we chose U-Net as the backbone of our model. Precisely investigated theory in our Specialized project provided us the opportunity to experiment with different settings and parameters, which was a huge advantage when starting our master thesis. The process of developing our own model was simplified since the preparatory work had already been executed in advance.

The problem of change detection is often a problem of detecting building objects from aerial images at different time stamps and comparing the results with each other. Consequently, buildings in two images need to be precisely detected to find the correct change. Considering our master thesis focuses on change detection on buildings from high-resolution aerial images in Trondheim, the process of extracting buildings is paramount to assure satisfying results. Thus, accurate building extraction is requisite. Pixel-based change detection is then executed to find changes that have occurred. Pixel-based change detection is highly dependent on building extraction accuracy, so our primary emphasis is to implement and create an accurate building detection method.

A throughout assessment of the results of our model was done based on selected criteria. The intersection of Union, precision, recall and visual comparison were considered when evaluating our model results. As we called our model, U-PSP-Net got an IoU, precision and recall of 98%.

1.3 Motivation of Creating a new Data Set

The world of machine learning is constantly developing and is gaining popularity and acknowledgment by the day. Guides, forums, communities, and data sets are just something of what the internet has to offer when it comes to machine learning. In the last ten years, there has been a massive jump in the search term "Machine Learning", and the same is true for "Machine Learning Data Set" [21]. Machine learning would be impossible without data sets, in the same way, the task in question would not be possible. As mentioned, many publicly available data sets exist for all types of problems, including building segmentation. One of them is a data set over Massachusetts. The before mentioned data set was used in earlier work for preliminary testing.

However, this data set is not up to the standard required for this task. The lack of adequate data set has been the limiting factor for scientific progress and improvements in machine learning [22] [23]. The most significant breakthroughs in the field have their foundation in large benchmark data sets like ImageNet [24], and GLUE [25]. Sun et al. [22] emphasizes that the size of the data set is directly linked with the model's performance. There is a logarithmic relationship between the performance and the volume of the data set. Jain et al. [26] also underlines the importance of data quality and how the results are upper bound by its quality.

Another contribution of this master thesis is establishing a training data set for the deep learning based approach. Large amounts of open-source data sets are available with labeled rooftops. The reasons for our choice of labeling our data set are many. First of all, the results are required to be highly accurate. Creating a new data set guaranteed high resolution and precise annotations. One purpose of identifying change is to update maps. Accordingly, an accurate data set over Trondheim is required. Precise training data over Trondheim equals better building detection, which leads to more accurate change detection. As of now, there exist no viable building data sets over the municipality of Trondheim. Data layers with buildings exists, however they are not attached to images. Fusing the data layer and images would not result in anything usable either.

Data sets with accurate annotations and high-resolution images already exist and are available, nevertheless, this is not the only reason a new data set is required. A good deal of other traits with a data set will affect the results. The machine learning model acquires knowledge by identifying distinct features for the given object. These features are easier to identify in a prediction setting if the prediction data set is as similar as possible to the training data set. This includes the buildings' shape, type, and texture, the surrounding nature and environment, and lighting and contrast conditions.

By creating a new data set over the study area, the best possible conditions are given for the model to perform. The machine learning model will learn what buildings in the local area look like and not anywhere else. This, of course, makes the model worse at segmenting buildings that are not similar to those in Trondheim. However, this is not an issue because the model only needs to perform over the given area. It is possible to train a network to be able to be accurate on any building. However, creating a general model will require an immense amount of training data. Buildings in Trondheim are pretty homogeneous. A consequence of this is that creating a data set based on the local area will benefit the results as the network will benefit from a specific data set for the local buildings.

It was crucial to label buildings in a proper annotation tool to achieve good results. Annotating buildings is not complicated, but the process was time-consuming and required accuracy and precision. The effect of a precise and large amount of training data was remarkable better results. Approximately 4000 roof planes were annotated and labeled, which only covered 25% all the images included in the data set. Since the annotation was time-consuming, a decision was made to stop the process and would instead continue if more training data was required.

1.4 Structure of the Thesis

The report starts with a chapter including a literature review of building detection and associated methods. The chapter will go through different methods for building detection and will be divided into unsupervised and supervised approaches. Furthermore, related works and their results will be presented in short. The advantages and disadvantages of different techniques will be discussed, and brief reasoning for choosing the detection method for this thesis. The following sections of this chapter will present knowledge essential for understanding, using, and optimizing a deep learning network. The general workings and concepts of a Convolution Neural Network (CNN) will be explained. An explanation will also be given about parameters in the network that are important for tuning and optimization later on. As well as an introduction of ablation study and its importance. Lastly, two specific neural networks, architecture and properties will be presented.

In the third chapter, our method is presented in general terms, followed by the development and an architecture description of our U-PSP-Net model. The reason for combining U-Net and PSPNet and their advantages and disadvantages are considered in this chapter. The models' different parts are described and discussed in detail, from the encoder and decoder to the Pyramid Pooling Module (PPM). Further on, the process of change detection is thoroughly explained. Diverse settings like batch size and learning rate must be configured when training a CNN or a deep learning model. The last step of our method is the change analysis. Evaluation of the building detection results is also included in the chapter. This part mentions the different parameters on which the results will be evaluated. Additionally, a brief argumentation of why each parameter is included in the evaluation will be given.

The data utilized is presented in Chapter 4, where information regarding cameras and flight is explained in detail. The chapter starts with general information about the image collection and expands on the process of creating a novel building data set. The preprocessing process is also explained, including the area of interest and the different preprocessing techniques such as cropping, annotation, creation of masks, image augmentation, and patching.

Chapter 5, Experimental study in Trondheim, starts with the results from the ablation study to develop our model and the belonging results. Further on, the hyperparameter optimization is addressed, followed by the outcome. Our model's result is then compared to the original U-Net and U-Net with PPM based on binary IoU, precision, recall, and training and validation loss graph. The result shown is evaluated and discussed, and how different factors may have affected our final result. Further on, the process of building detection is accounted for, and a presentation of some good and worse detected buildings in Trondheim using our model is then presented and discussed the possible reasons for the result. The change detection result is then presented, followed by a discussion of the change detection result. Lastly, a discussion regarding the data set used and the process of making our training data set.

Chapter 2

Background

Background knowledge is essential in every field of research. To be able to use and understand advanced methods, you have to know the fundamentals that underlie the processes. Therefore, it is vital to research the underlying topics that this thesis is built upon. Hence this chapter will present the most outstanding issues, methods, and theories to conduct an accurate building detection.

Fundamental knowledge is crucial when developing a deep learning network. Recent research has focused on the automatic and semiautomatic extraction of cartographic features from aerial and spatial images. Among the many cartographic features to be spotted, buildings are perhaps the most noteworthy. Building detection approaches are accounted for, followed by CNN theory. The elements of CNN's and different hyperparameters are defined, and how to execute an ablation study is explained. Further, a deeper analysis of the Convolution Neural Networks, U-Net, and PSPNet is conducted. The architecture, as well as advantages and disadvantages, are accounted for in this chapter.

2.1 Building Detection

Detecting cartographic features from aerial and spatial images has been a major research focus. Buildings are among the most visible and complex cartographic features. [27]

For various reasons, automating the extraction of buildings is a challenging process. Spectral and geometric characteristics of each region can differ, building structures can include intricate architectural details, and disturbing objects surround the building. Roofs and ground may have similar spectral characteristics and can be difficult to differentiate [28] [27]. Various forms of single structures and/or building blocks can be found in dense urban areas. A single building is a stand-alone structure with a fixed height, shown in Figure 2.1 (a). Figure 2.1 (b) depicts a building block, which is a collection of single buildings with varying roof materials and heights. In densely populated places, the fundamental problem is the low contrast between buildings and non-buildings. Courtyards, parking lots, bare soil, roads, pavements, and cars are examples of non-buildings. Buildings range in size, colors, and forms and can be hard to distinguish. The building height can also vary from single-floor to towers, as illustrated in Figure 2.1 (d). [29]

Figure 2.1: Buildings of various types: (a) Single buildings, (b) Building blocks, (c) Buildings of different sizes, (d) Buildings of different elevations. [29]

Figure 2.2: Building extraction using deep learning semantic segmentation. [30]

Deep learning and semantic segmentation methods that enable autonomous object extraction from high-resolution satellite images have grown in prominence in recent years. It is critical to extract buildings from high-resolution aerial imagery for geospatial applications, including urban planning, telecommunications, disaster monitoring, navigation, updating geographic databases, and dynamic urban monitoring. Figure 2.2 depicts the buildings that were retrieved using deep learning semantic segmentation. [28]

In land cover classification, pixel-based classification is widely used and is divided into Unsupervised and Supervised. The unsupervised classification technique, also called clustering, is used for classifying imagery without pre-defining classes. In contrast, supervised classification is designed to classify pixels after a user defines the number of classes and characteristics of each category. A more thorough explanation of unsupervised and supervised are executed in the following sections. [31]

2.1.1 Unsupervised

The most common method for clustering pixels in a data set is a technique called unsupervised classification that relies on statistics based on no user-defined training classes [31]. Neither prior knowledge of the area under investigation nor previous statistics of the previously defined classes are required to apply the classification technique [32]. The output consists of unclassified clusters and labeling change trajectories, which is a significant drawback with unsupervised classification [6]. After clustering, each cluster is manually assigned a class label. Another drawback is the selection of the number of clusters. The outcome is influenced dependent on the choice of an appropriate number of clusters [33]. The two most frequently used unsupervised classification algorithms are ISODATA and K-means [31].

ISODATA is an iterative method and utilizes minimum distance as a measure of similarity to cluster data elements into distinct classes [34]. In order to achieve this, the mean value is recalculated, and all of the pixels are then reclassified until all of them fit into the threshold input [31]. K-means is also an iterative method using minimum distance to cluster pixels into different categories. K-means requires several classes to be defined before the calculation, unlike ISODATA. Distinguishing between obvious categories like water, snow, and clouds and detecting differences between two images taken with different time stamps are K-means areas of application. [35] [31]

In a paper written by Aytekin et al. [36] a novel automatic and unsupervised method to extract buildings is presented. The technique uses mean shift segmentation and PCA to filter out shadows, vegetation, and roads in an urban area. The method can extract buildings, however, not without including a lot of false positives. Wei et al. [27] proposes another method using unsupervised clustering and edge detection for building detection. The method first clusters the image before detecting shadows. Building candidates are then picked based on the detected shadows before refined with edge detection and Hugh transform. The method is reported as successful, although it over-classifies to some degree.

2.1.2 Supervised

Supervised classification is a commonly used pixel-based classification method [34]. Maximum Likelihood, Mahalanobis distance, and Spectral Information Divergence are some classifiers that were in the early years of remote sensing data processing software widely used [37] [38] [39]. Machine learning-based classifiers like Classification and Regression Tree, Support Vector Machine, and Random Forest have recently been proven to perform better, which made them frequently used in land cover classification [31].

In a paper from 2014, Ghaffarian and Ghaffarian [40] proposes a novel supervised method for building detection which exploits the fact that 3D building objects cast a shadow. With this approach Ghaffarian and Ghaffarian [40] achieved an F1-score of 84.8%. Senaras and Vural [41] are using another supervised method to detect buildings. Their self-supervised decision fusion

method achieved promising results with an F1 score of 81.9%. Grigillo and Fras [42] presents a novel method for building detection which combines supervised and unsupervised detection. Initially, buildings are detected by applying a supervised method. Later an unsupervised method is applied to enhance the initially detected building, using region growing and unsupervised classification. The study does not calculate any pixel-level metrics, solely object by object. Given this, the detection percentage for the buildings came out to 85%.

2.1.3 Region-Based Segmentation

Region-based segmentation is a set of approaches for segmenting various areas based on their commonalities. Homogeneous regions are formed, and their outlines are retrieved. Looking at the intensity of each pixel is a frequent strategy. On the other hand, noisy segmentation often occurs even though this method is widely used. This phenomenon is accentuated by high-definition satellite imagery. Newer segmentation algorithms utilize features like texture and context to improve segmentation. The image is divided into different segments in the segmentation process based on their features. This process separates the components based on similarities, but no classes are formed, and each segment is treated as unique. A classification step is therefore executed to extract different categories from the image. On that account, the following approaches presented combine segmentation and classification methods to extract useful information.

A region-based segmentation approach is a fuzzy segmentation. This method assigns each pixel with a probability of belonging to a segment, in contrast to crisp segmentation, which gives each pixel to a single segment [43]. The method presented by Lizarazo and Elsner [43] consists of three steps. The three steps include assigning pixels with a degree of membership to each land cover class, feature analysis, and assigning regions to land cover class based on their contextual indices and membership value. The overall accuracy of the complete segmentation was 82%. However, the method had almost 90% accuracy when only considering buildings and background.

One thresholding technique, Otsu's Method, is presented by Yohannes and Utaminingrum [44] is used to separate pixels into two classes. In order to achieve this, both the in-between class variance and the class variance between pixels need to be maximized. In essence, the algorithm maximizes differences between classes and minimizes differences within each class to determine the best threshold. The method considered the most humble of the region-based approaches is region growing. During the initialization phase, the system includes seed pixels, making the method pixel-based [44]. The fundamental principle of the algorithm is to take into account the similarity of adjacent pixels before including or excluding them from the segmented area. Yohannes and Utaminingrum [44] only conducted a small experiment to detect buildings, and the accuracy achieved was 70%.

2.1.4 Deep Learning

Deep learning techniques are gaining popularity due to their remarkable outcomes. Not just in geomatics but also biomedicine, finance, and various other fields. Deep learning is most commonly employed in geomatics for autonomous segmentation of aerial and satellite images.

Over the last few years, deep-learning-based semantic segmentation classification as a pixel-based classifier has performed exceptional results. According to the state-of-the-art systems, semantic segmentation networks could be divided into two categories: encoder-decoder and multi-scale pyramid pooling [45]. Within the two categories, U-Net, SegNet, and PSPNet are the most commonly used and comparable architectures [31].

Deep learning models can be used for a variety of tasks. Li et al. [46] presented a novel deep learning framework that enables automated extraction of building footprint polygons from very high-resolution aerial imagery. From very high-resolution remote sensing images, U-Net, Cascade R-CNN, and Cascade CNN deep learning models were used to obtain building segmentation maps, building bounding boxes, and building corners, respectively. Delauney triangulation was used to construct building footprint polygons. Building corners were triangulated using building boundary boxes and segmentation maps as constraints. [46]

By using Res-U-Net combined with guided filters Xu et al. [47] detected buildings using the Potsdam [48] and Vaihingen [49] data set. They were able to achieve 96% accuracy on their building detection and stated that their method could achieve better results than other methods. Zhao et al. [50] combined R-CNN with building boundary regularization to extract buildings. The method achieved an average F1 score over four different areas of 71%. Although the relatively low score is affected by the different characteristics of the four areas, it is still a good score compared to other methods. An advantage of the method is using building boundary regularization, which creates regularized polygons that greatly use cartography.

2.1.5 Conclusion of Literature Study

High-resolution satellite imagery can be difficult to classify because of the high information content and detail level. When pixels within an object area have different texture features, an effect called *salt and pepper* occurs. The result is a reduction in the accuracy of pixel-based classification. To address these concerns, an object-based solution was proposed, where corresponding pixels are first aggregated into an object through segmentation to avoid the salt and pepper effect [31]. Different and more traditional image classifiers applied for building detection have proven to be quite effective. However, they are not able to achieve the same results as the state-of-the-art deep learning methods. Since the introduction of AlexNet, CNNs have achieved better classification results than any other method [51]. The salt and pepper effect is one of the main concerns with traditional per-pixel classifiers used for image classification [52] [53]. As well as not being advanced enough to consider as many features as CNN's, the less advanced model falls short of deep learning methods.

Consequently, the focus further on will therefore be on deep learning approaches, where CNN, U-Net, and PSPNet will be accounted for. Deep learning has shown remarkable outcomes throughout the years compared to other traditional approaches. To be able to achieve accurate enough results that could be used further on in a change detection analysis, it is, therefore, necessary to use a deep learning approach.

2.2 Convolution Neural Network

A CNN is a class of Artificial Neural Network (ANN). Most commonly, CNNs are used for image analysis, but they can also be used for voice recognition and natural language processing. In contrast to ANNs, CNNs require significantly fewer parameters, allowing for the construction of larger models that are capable of solving more complex tasks [54]. Additionally, the amount of pre-processing required is significantly lower in comparison to other classification algorithms. This is a consequence of the network learning to optimize its filters through learning, rather than the traditional algorithm of hand-engineered filters.

CNN's history began in 1988 when Fukushima released the first paper introducing CNN's [55]. It was not frequently utilized at the time due to the limited computational power available. CNN's were refined throughout the following few decades, and researchers were able to accomplish state-of-the-art results with them. Nevertheless, it was not until 2012, with the release of AlexNet, that CNN achieved wide recognition. AlexNet participated in the ImageNet Large Scale Visual Recognition Challenge and outperformed all other traditional machine learning, and computer vision approaches [51]. The network outperformed the runner-up by more than 10.8% points, achieving top-1 and top-5 test set error rates of 37.5% and 17.0%, respectively. Following this, the advancements and evolution of CNNs did not seem to slow down, and there are now a plethora of networks capable of almost perfect classifications. [56] [57]

A CNN consists of several elements and layers to execute the image segmentation. The different elements play an essential role for the network to work properly. The subsequent sections will go more thoroughly through the various aspects of a CNN. Further on, different CNNs like U-Net and PSPNet will be accounted for.

2.2.1 Elements of Convolutional Neural Networks

Convolution Layer

Convolution is a procedure in a CNN that is used to reduce the number of parameters. Throughout most situations, CNNs get input in the form of images, which are typically three layers deep. Depth is determined by the number of distinct bands employed in the collection of images. RGB images have a depth of three containing red, green, and blue bands. Considering this, multi-spectral imaging can be used with layers anywhere between 3 to 15. Combine the

depth, height, and number of pixels to get the dimensions of the CNN's input.

Matrix multiplication links each input and output unit in traditional neural networks. In other words, each output unit needs to interact with each input unit, resulting in a vast number of connections. CNN's exploit *sparse interactions*, which indicates that not every unit is connected and fewer parameters have to be saved [58]. This is achieved by applying a kernel and performing a convolution. Kernels are matrices that can be slid over an image to blur or detect edges based on their size. The method is known as convolution and produces a feature map.

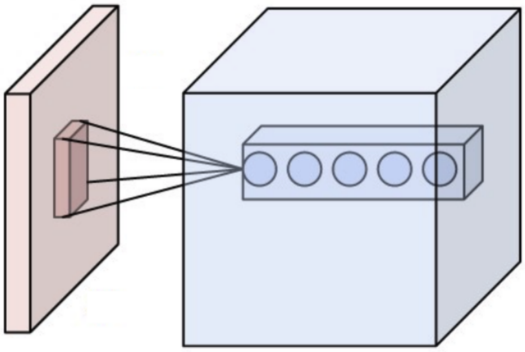


Figure 2.3: Multiple layers corresponding to a different filter, but the same spatial reference [54]

If a 100 x 100-pixel RGB image is processed in a fully connected feedforward neural network, each neuron on the second layer must include 10 000 weights. In contrast, regardless of picture size, a CNN with a 5 $\times$ 5 kernel and equal weights only requires learning just 25 x 3 weights. Rather than treating each pixel in the image as dependent on every other pixel, all pixels in the kernel become dependent on each other. Additionally, each convolution can result in features that can be used in the classification problem, making them an effective way to limit parameters. Figure 2.5 depicts a simplified illustration of this approach.

Convolution is done repeatedly with various kernels where each kernel generates a unique feature map. When various features are discovered, the network learns which filters to activate. Using this strategy, the network will recognize which kernels to utilize and which to avoid in specific problems after the learning phase [59]. Through the convolution process, the network uses the same size kernels, and the location of the output is maintained regardless of which kernel is used. This enables stacking of the output images, as seen in Figure 2.3. The depth dimension is used to stack the output images, which forms the complete output from the convolutions. As a result of arranging the outputs in this manner, another interpretation of the results can be made. Each output can now be viewed as an output from a single neuron that focuses a small portion of the input, all with the same parameters as the kernel's neurons. [54]

Stride

Stride is another method for reducing network parameters. When the kernel goes from pixel to pixel during convolution, there is a lot of overlap between nearby pixels. The overlap may be

adjusted by changing the stride. Stride is the number of pixels the kernel moves for each iteration. Albawi et al. [54] provided an example applying a 3 x 3 kernel on a 7 x 7 image. Increasing the stride value from 1 to 2 reduced the overlap and limited the output from 5 x 5 to 3 x 3.

Padding

There is a disadvantage in the loss of information at the edges of the input during convolution. A certain amount of information will always be lost at the edges depending on the kernel size. An input image of 7 x 7 will result in an output image of 5 x 5 by using a 3 x 3 kernel. Padding can be used to prevent information from being lost. This simple method of placing zeros around the edge will ensure that no data is lost. This procedure will not only keep all of the information around the edges, but it will also keep the output dimension. [59]

Convolutional Formula

In Equation 2.1, the convolution of a specified pixel in a subsequent layer is calculated.

$$S_{ij} = (I * K)_{ij} = \sum_{a=\lfloor -\frac{m}{2} \rfloor}^{\lfloor \frac{m}{2} \rfloor} \sum_{b=\lfloor -\frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} I_{i-a,j-b}\, K_{\frac{m}{2}+a,\frac{n}{2}+b} \tag{2.1}$$

With a and b being the rows and columns of the matrix of weights, K represents the matrix of weights. Input pixel values are represented by matrix I, where S_{ij} is the output pixel.

Nonlinearity

Non-linearity is the second stage of the convolutional network. In this stage, the output can be adjusted or cut off [54]. To saturate or limit the generated output, each linear activation is sent through the non-linear activation function. In non-linear transform stage of convolution neural networks, Sigmoid and Tanh functions are typically used as saturated activation functions [60]. The gradient signal fades as the neural network architecture becomes more complicated, a phenomenon known as *vanishing gradient*. As illustrated in Figure 2.4, the gradient of those functions is nearly always close to zero except for those in the center. On the other hand, the activation function named Rectified Linear Units (ReLU) has a constant gradient for positive inputs. Despite the fact that the function is not differentiable, it is often neglected in practice. A common type of non-linearity is shown in Figure 2.4, with ReLU and Softplus being unsaturated activation functions and constant gradients for positive inputs. Deep convolutional neural networks using ReLUs train multiple times faster than with Tanh units [51]. The concept of ReLU and its gradient are shown in Equation 2.2 and 2.3. [54]

$$ReLU(x) = max(0, x) \tag{2.2}$$

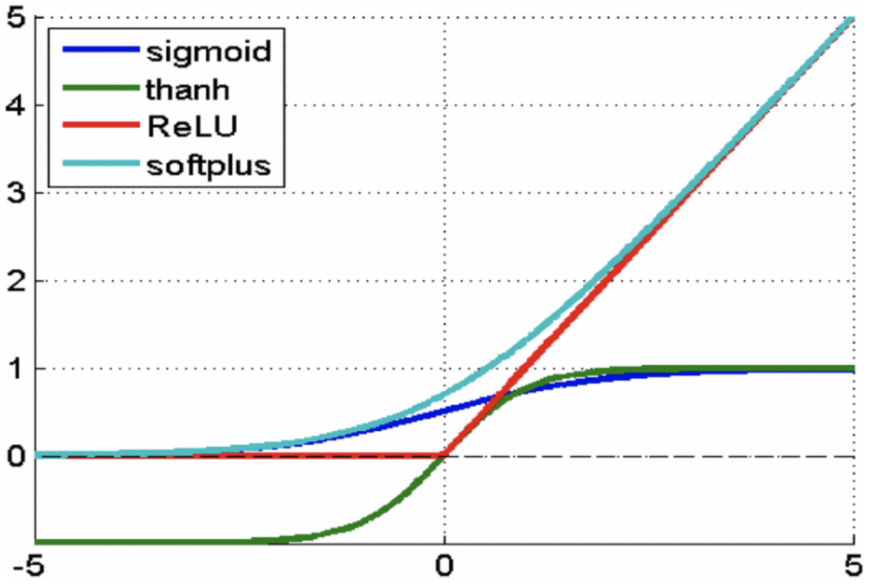


Figure 2.4: Common types of non-linearity [54]

$$\frac{d}{dx}(x) = 1 \quad if\ x > 0\ ;\ 0 \quad otherwise \tag{2.3}$$

Pooling

The pooling function is the third stage, where the output of the layer is modified, and the image output is substituted at a given location with a summary of adjacent outputs [58].
Down-sampling is a fundamental principle behind pooling that reduces complexity and yields typical features for further layers. Maximum and average pooling are two types of pooling. The most prevalent type of pooling is max pooling, which outputs the maximum value from the area of the image covered by the kernel. On the other hand, average pooling returns the average of all the values from the kernel-covered portion of the image. [54]

Fully-Connected layer

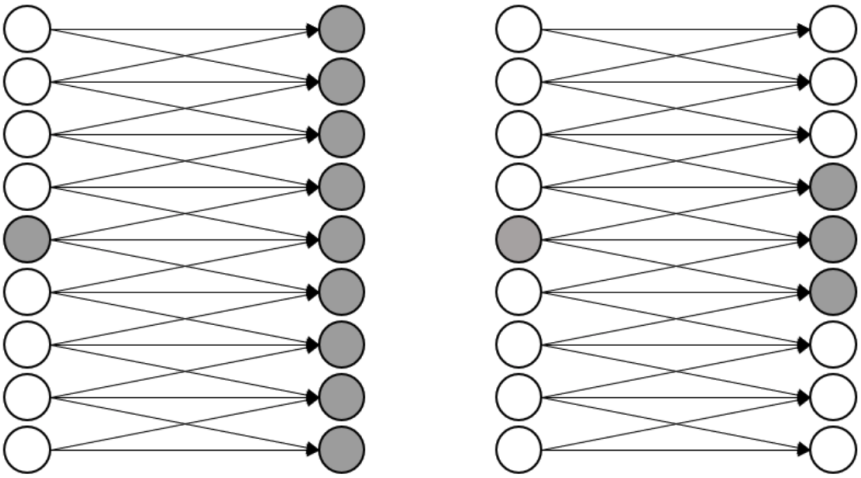


Figure 2.5: Fully Connected Neural Network vs. Convolutional Neural Network

In the final segmentation, the layers are fully connected. Traditionally, fully connected layers are found in neural networks [54]. Nodes are connected to each other in the next and previous layers, as the name implies. Figure 2.6 illustrates the connection. Because of the large number of parameters in these layers, training them is the most expensive aspect of CNN. There can be numerous layers in a network, depending on how complex it is. AlexNet, for example, has three levels fully connected layers. [51]

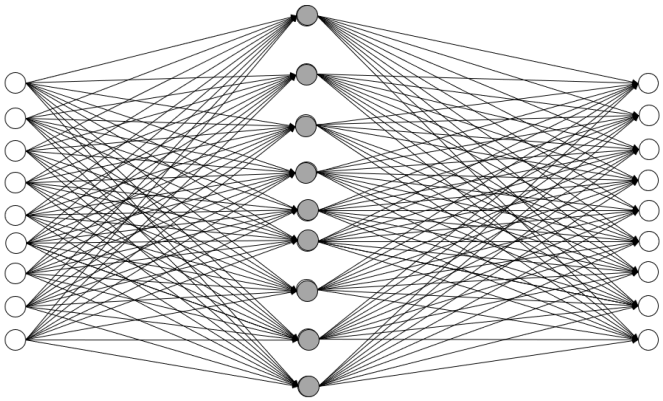


Figure 2.6: Fully connected layer, each node corresponds to a feature.

2.2.2 Hyperparameters

The structure of a CNN is extremely complex, making it difficult to build an efficient network configuration which is a challenging but necessary task. Decisions such as how many convolution kernels to employ in the convolutions layer and how large the kernel should be? Which type of activation function is the most beneficial? How can you modify the CNN's learning rate in a way

that it learns faster? In order to define these parameters, the hyper-parameters of the neural network must be set [61]. The CNN structure is almost always fixed, and then hyper-parameters are set based on the network designers' previous experiences [62]. The network designers modify the parameters through extensive trial-and-error experiments before determining how the final model would be employed. [61]

Deeper models with a greater number of filters on each layer might require a long time to evaluate given hyper-parameter values for CNNs. When optimizing for comparable data sets, hyper-parameters that are suited for a given data set may be a good place to start. Although the same hyper-parameters from various data sets cannot be utilized in the same way but must be evaluated to determine proper value ranges. [63]

The number of layers and nodes, batch size, learning, and dropout rates is all examples of hyper-parameters [64]. According to Smith [65], there is no simple and quick method to adjust the hyper-parameters, specifically learning rate, batch size, momentum, and weight decay. The relevance of various hyper-parameters can be assessed using Analysis of Variance (ANOVA). The relevance of a subset of hyper-parameters is defined as the amount of variance that a subset of hyper-parameters provides to the validation error. A selection of hyper-parameters that describe the validation error with the greatest variance will have greater significance. According to Hinz et al. [63] studies using a Cohn-Kanade data set, the importance of the different hyper-parameters is more or less consistent across the different input sizes. Figure 2.7 illustrates how the 18 most significant hyper-parameter subsets of the Cohn-Kanade data set affect validation error. [63]

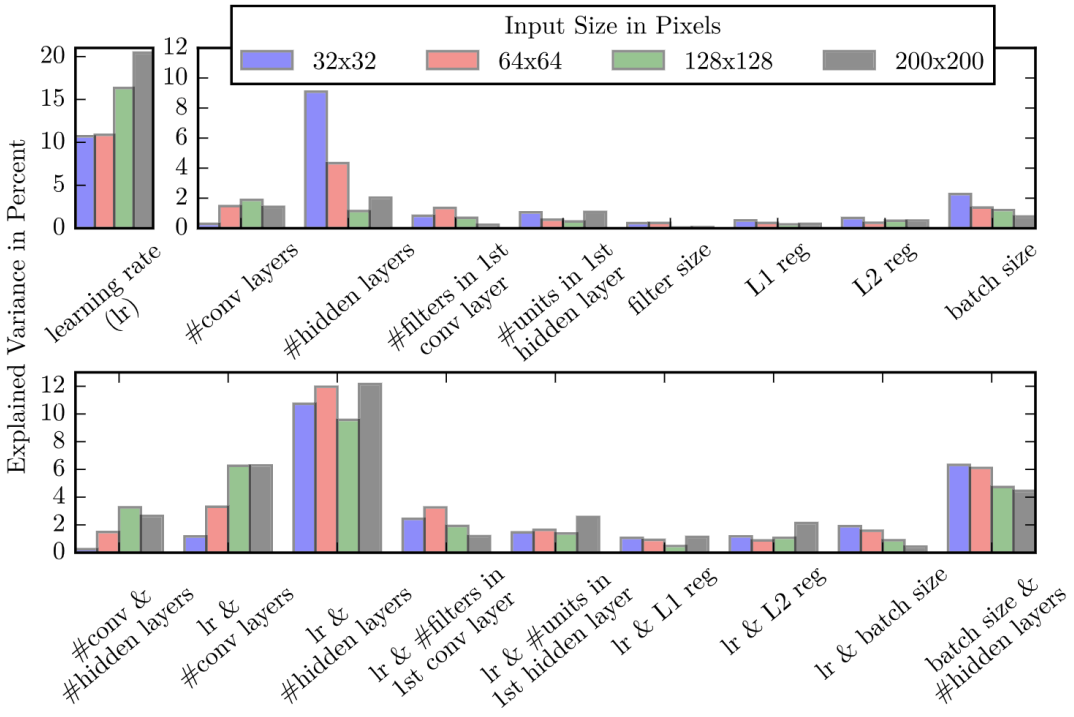


Figure 2.7: The 18 most important hyper-parameter subsets of Cohn-Kanade data set [63].

Learning rate

Learning rate is one hyper-parameter that is broadly utilized. Optimization algorithms use learning rates as tuning parameters to determine the step size at each iteration as they work toward a minimum loss function. In other words, it's an indicator of how fast a machine learning model learns since it reflects the amount of newly acquired information that overwrites previously learned information. When the model slows down its improvements, learning rate decay is used to reduce the learning rate. If the validation loss does not change for a set number of epochs, the learning amount will be reduced by a fixed percentage. Given that the weights the network has learned during the training are an average of multiple steps, it is favorable to reduce the learning rate to prevent learning from anomalies.

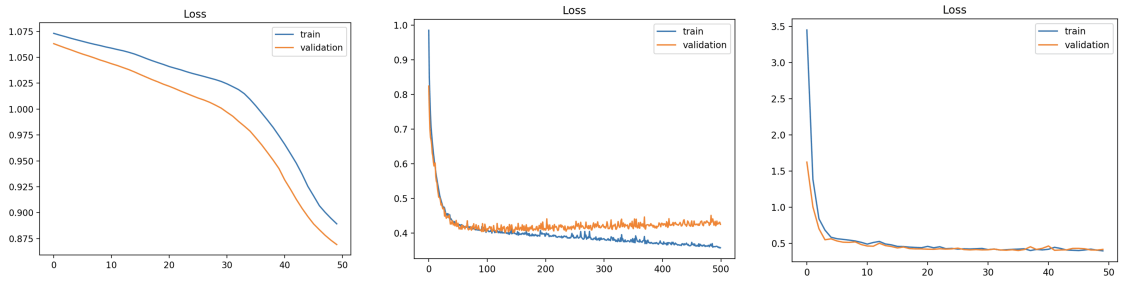


Figure 2.8: The graphs above show examples of learning curves. The blue graph indicates training error, and the orange graph indicates validation error. From left to right, a model that requires further training, an overfit model, and a perfectly fitted model. [66]

Batch Size

The batch size is also a hyper-parameter and controls the number of samples to process before updating the internal parameters. In essence, a batch is a loop that iterates over multiple samples to make predictions. An error is calculated by comparing the predicted output variables to the predicted output variables at the end of the batch. In order to improve the model, the algorithm also uses this error to move down the error gradient.

Epochs

Epochs are the number of times the learning algorithm traverses the complete training data set. Epochs are typically large, enabling the learning procedure to continue until the model's error has been reduced to a reasonable level. Learning curves are line charts where epochs are assigned to the x-axis as time, and the model's error or skill is displayed on the y-axis. A plot like this can be used to determine whether a model is overlearned or under learned or whether it is suitably fitted to the training data set. Figure 2.8 illustrates these effects in detail.

Input Shape

Another hyper-parameter is the input shape. CNN's need to be fed a 4D array as input, so the shape of the input is (batch size, height, width, depth). The image's dimensions are represented by its height, breadth, and depth. The color channels of the picture are represented by depth. For example, RGB images have a depth of 3, while greyscale images have a depth of 1.

2.2.3 Image Augmentation

Deep Learning models that are trained on big data sets are generally considered to be more accurate [23]. It can be very challenging to assemble enormous data sets due to the manual effort involved with collecting and labeling data [67]. Therefore, image augmentation is a technique for expanding the data set and improving learning with minimal work.

Simple transformations such as horizontal flipping, color space augmentations, and cropping are often used to illustrate the effectiveness of Data Augmentation. Horizontal axis flipping is far more common than vertical axis flipping. This is one of the simplest augmentations to build and has proven beneficial on data sets such as CIFAR-10 and ImageNet. Rotation, translation and noise injection, and color space transformation are other image augmentation methods for expanding the data set. Besides the simple augmentation methods, there are many other methods ranging in complexity.

2.3 U-Net

U-Net is a CNN designed for biomedical image segmentation developed by researchers at the University of Freiburg [68]. The model's architecture consists of skip connections connecting the encoder and decoder, including convolution and pooling layers. Despite a lack of labeled training data, the network has shown excellent results in segmenting medical images and has become the de-facto standard for segmenting medical images. [69] Although U-Net is mostly used for medical images, its versatility allows for other uses as well. The way it is constructed makes it adaptable to the training set it is given, allowing it to be able to segment everything from buildings to urban scenes to plant leaf disease [70] [71] [72].

2.3.1 Architecture

Encoder

A symmetric representation of U-Net's architecture is illustrated in Figure 2.9. The model includes an encoder and decoder. The spatial features are extracted from the image during the encoding phase by performing two 3 x 3 convolution operations followed by max-pooling. The max-pooling stage gets out the most typical features by using a max filter, as shown in Figure

2.10. For each region covered by the filter, the max of that region is used to create a new output matrix. The output matrix represents the max of a region in the original input. Using a pooling window size of 2 x 2 and a stride of 2 avoids overlapping. This sequence is repeated a number of times, and the filters in the convolutional layers are doubled with each down-sampling. Then the progression of two 3 x 3 convolution operations connects the encoder to the decoder. [69]

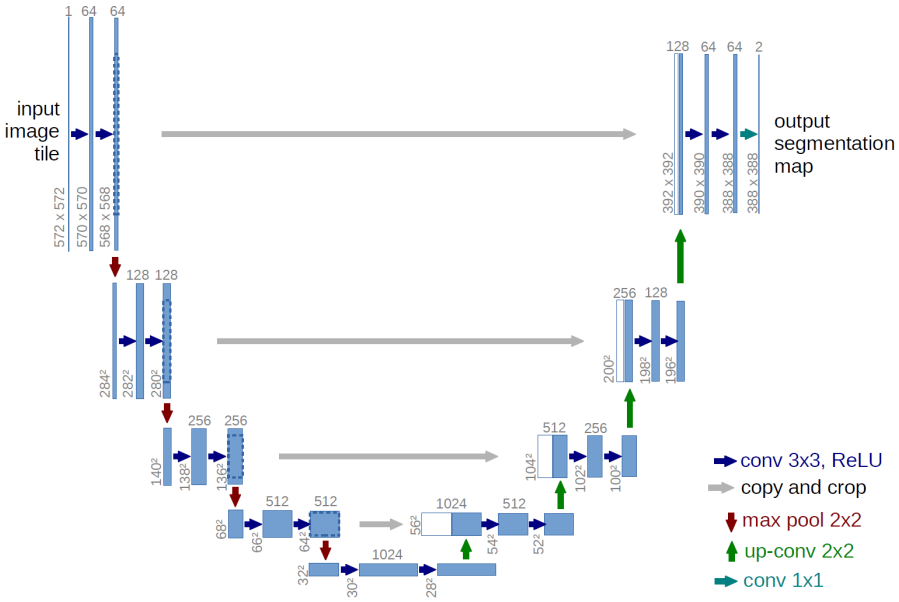


Figure 2.9: U-Net architecture [73].

Decoder

Using a 2 x 2 transposed convolution operation, the decoder constructs the segmentation map from the encoded features [74]. As a result, the features are cut in half, and another sequence of two 3 x 3 convolution operations is performed. Like the encoder, the decoder repeats up-sampling and convolution multiple times, halving the number of filters at each stage. The final step to get the final segmented map is a 1 x 1 convolution operation executed. The U-Net architecture uses the ReLU activation function for every layer with convolutional, except for the final layer, which uses the Sigmoid activation function [69].

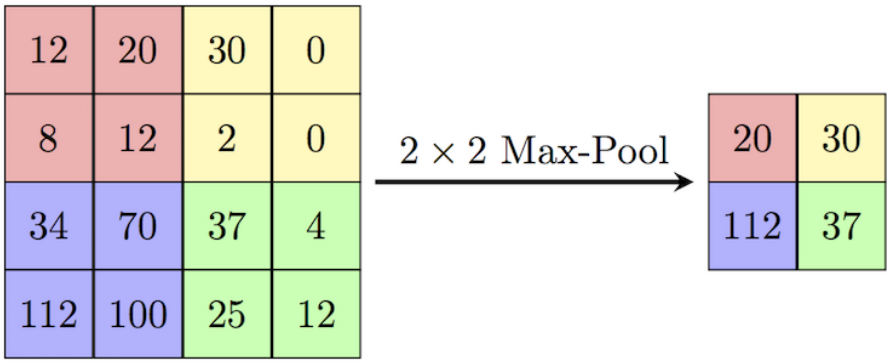


Figure 2.10: Max-pooling of an 4 x 4 image with a window size 2 x 2 and stride 2 [75].

2.3.2 Advantages and Disadvantages with U-Net

One of the most ingenious aspects of the U-Net architecture is its implementation of skip connections. The output of the convolutional layer is transferred to the decoder in all four layers before the pooling operation of the encoder. Feature maps are concatenated with up-sampled output, and the resultant map is transmitted to the subsequent layers. The network can regain details and spatial information lost during pooling operations when down-sampling through skip connections. The data is sent to the encoding phase while up-sampling is taken place so that all spatial information is retained [69] [76]. Another way the information is retained during the up-sampling is by using a large number of feature channels. This allows for context information to be kept when propagating upwards in the layers.

An immediate disadvantage with U-Net is the number of trainable weights. Compared to other CNNs, U-Net, with its U-shape and symmetry, is a lot bigger than other networks. This is something that will affect the training time and, to some degree, the prediction time. This will also affect the GPU memory footprint. For larger images, this will limit the batch size unless one has unlimited GPU memory. According to Zhou et al. [77] despite U-Net's success, it has two main limitations. Firstly the optimal depth for a specific problem is not known prior to test it. Finding the optimal depth requires either researching similar segmentation problems or performing extensive architecture testing. Secondly, U-Net's skip connections do not allow for the fusion of features that are not of the same scale.

Figure 2.9 shows the architecture consisting of many layers, where each layer executes max pooling. For each max-pooling, the image resolution decreases, as depicted in Figure 2.10. Considering the number of layers, the model takes a significant amount of time to train.

2.3.3 Visualization

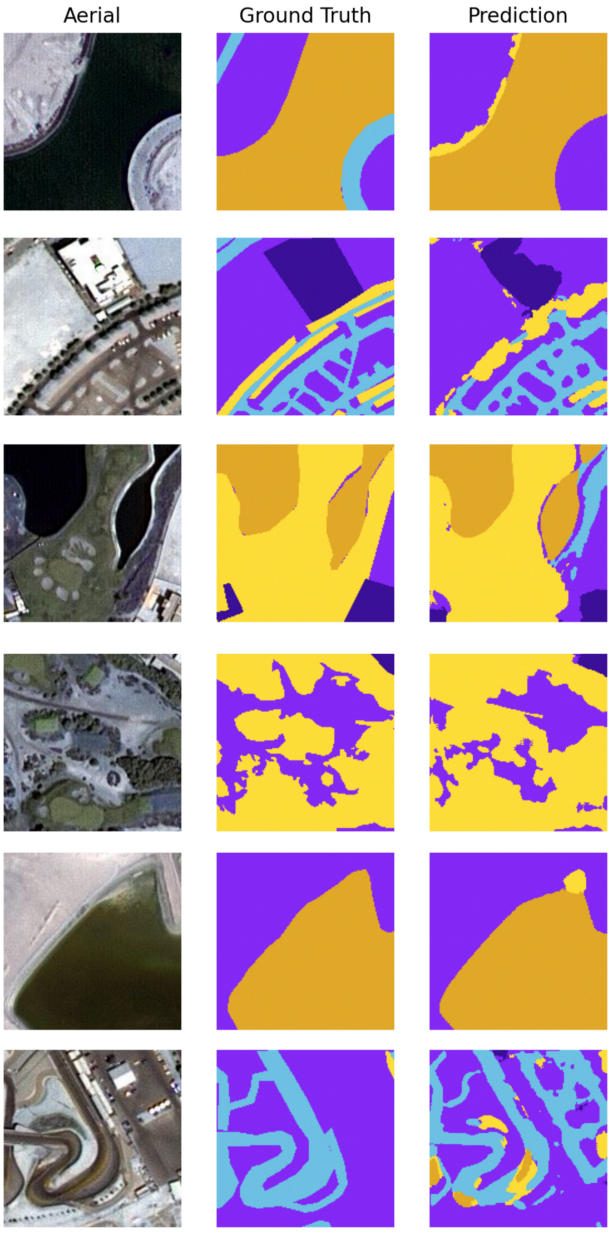


Figure 2.11: U-Net predictions on aerial imagery over Dubai's landscape. The prediction is compared to ground truth and original image. [78]

Figure 2.11 illustrates an example of a U-Net segmentation prediction of an aerial image over Dubai, United Arab Emirates. The columns show the aerial photo compared to the ground truth and the prediction from the U-Net model defined in TensorFlow. The different colors mark the different areas segmented from the aerial image. As the Figure depicts, the model can predict each area quite well. The more significant areas seemed uncomplicated, and the model distinguishes between them well. As the U-Net model utilizes max-pooling in several layers, the most essential and typical values are extracted. This results in the most significant areas being detected easier than the slightly smaller areas, which is shown in Figure 2.11.

2.4 PSPNet

Pyramid Scene Parsing Network (PSPNet) is a semantic segmentation model that utilizes a pyramid parsing module that exploits global context information. PSPNet provides an effective global contextual prior for pixel-level scene parsing. With a PPM, you can collect levels of information that are more representative than a global pooling module. PSPNet's encoder consists of different CNNs, which are used as the backbone with dilated convolutions, as well as the PPM. [79] [80]

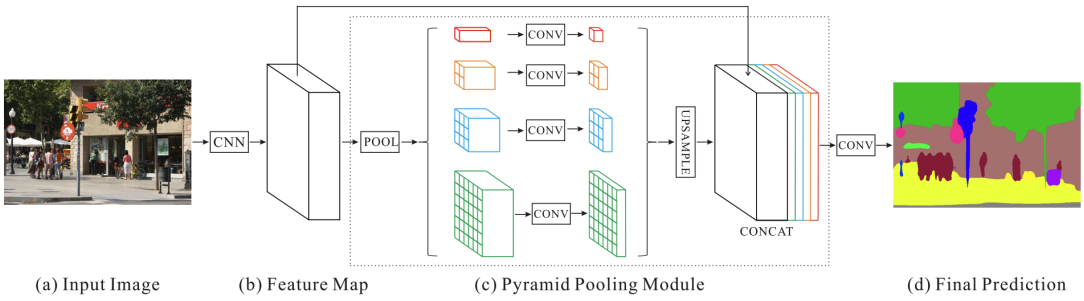


Figure 2.12: PSPNet architecture [80].

2.4.1 Architecture

The model's architecture is depicted in Figure 2.12. A feature map is pooled from the CNN backbone at different sizes and then passed through a convolution layer. Thereafter, the different sizes are upsampled to the same size as the original feature map. A final step is to concatenate the upsampled maps with the original feature map before passing them to the decoder. Through this technique, features from different scales are combined to create an overall context. [80]

PSPNet is composed of a PPM of four levels, which is used to gather context information. By utilizing the 4-level pyramid, the pooling kernels cover the entire image, half of it, and small portions. All the layers are fused as the global prior, which is concatenated with the original feature map in the final part. Then a convolution layer is followed to generate the final prediction map. [80]

Four pyramid scales are combined in the pyramid pooling module for four different scales. The different scales, depicted in Figure 2.12, are 6, 3, 2, and 1 for the colors green, blue, orange, and red, respectively. The coarsest level, indicated by red, is global pooling for a single bin output and captures all the information in just 1 x 1 spatial location. The smaller the pyramid size is, the smaller area the model captures. A bigger pyramid size equals bigger features, and the model is able to capture higher resolution features. [80]

The following pyramid layer separates features into sub-regions, and pooled representations for different locations are derived. Each level of the PPM produces a feature map that contains varying sizes of features. To preserve the weight of the global feature, Zhao et al. [80] applies a

convolution layer of 1 x 1 after each pyramid level of a pyramid of size N in order to reduce the original dimension of context representation to 1/N of what it originally was. The low-dimension feature maps are directly up-sampled through bilinear interpolation to obtain the same size features as the original feature map. To complete the pyramid pooling global feature, different levels of features are concatenated. [80]

2.4.2 Advantages and Disadvantages with PSPNet

The main advantage of PSPNet is the model's ability to set up hierarchical global prior and take information with varying scales from different subregions into account. It is crucial for CNN to have a big receptive field to see the image in a global context. Increasing the receptive field without decreasing the final conversion is, on the other hand, a challenge. The PPM in PSPNet increases the receptive field without a momentous decrease in the output resolution or increase in parameter or layer count. Dilated convolution is also used in PSPNet in order to increase the receptive field and retain the resolution.

A drawback with PSPNet is its risk of losing small features compared to the ground truth. Shi et al. [81] reported after testing PSPNet that the model lost some details, especially those within other objects. Shi et al. [81] experiments with PSPNet, and concludes that the model is better than other semantic image segmentation models such as Deeplab [82] and Fully Convolutional Network (FCN) [83] in especially classes like walls, fences, poles, traffic lights, and so forth. Classes with massive areas like sky, road, and building are not giving remarkable results compared to other methods.

2.4.3 Visualization

Figure 2.13 illustrates an example of the Cityscape data set, which contains 5000 images collected from 50 cities in different seasons. Column (a) shows the original image, and (b) shows the ground truth, followed by (c), which shows the predictions done by using PSPNet. As the Figure shows, the PSPNet is able to detect small objects like humans, lampposts, cars far away, and traffic signs. The model also detects more significant objects like roads, sidewalks, cars, and trees. The reason for this is the different sized kernels in the PPM in the PSPNet. The different scaled kernels capture differently sized features, making the network detect more small and significant objects.

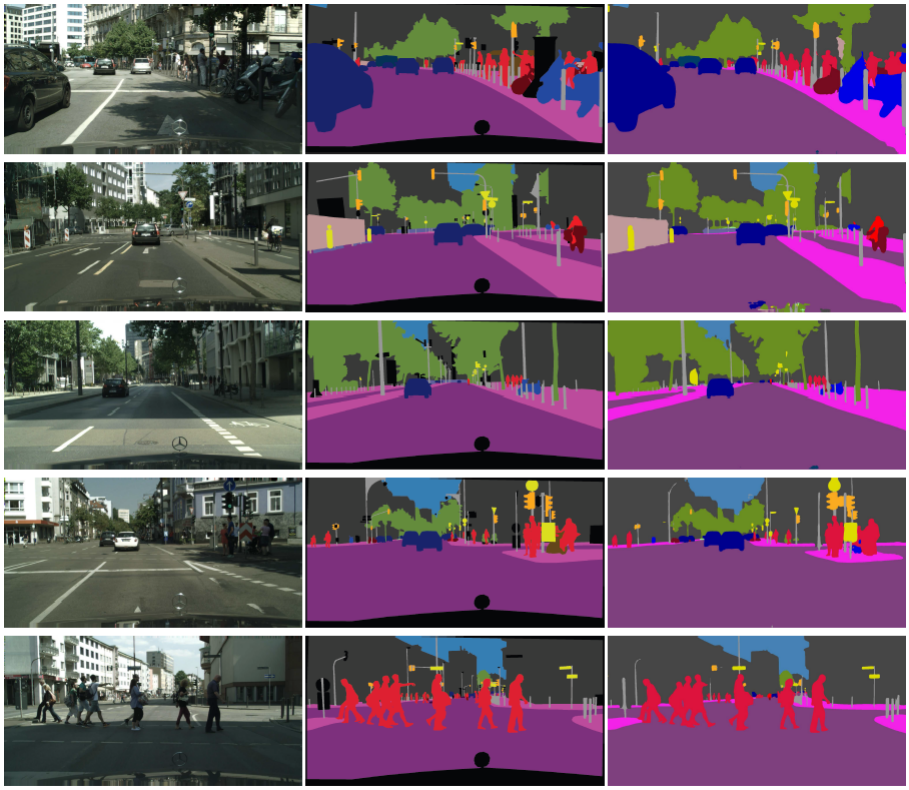


Figure 2.13: PSPNet predictions compared to ground truth and original image. [84]

2.4.4 Ablation Study

When building a machine learning model, you commonly add various concepts to improve the overall model performance. A component of a system can be investigated by breaking it and seeing what changes. This is called an ablation study. When conducting an ablation study, you remove sections of the network system to determine which elements that are significant to the result.

Machine Learning (ML) and Artificial Intelligence (AI) use ablation to remove a component of an artificial intelligence system. During an ablation study, specific elements of an AI system are removed from the system to determine how they contribute to the system's performance. It is an analogy to biology (the process of removing components of an organism) and is predominantly used in analyzing artificial neural nets by analogy with ablative brain surgery.

For ablative study to be successful, a system must display graceful degradation: it must be able to perform even when some of its components are missing or degraded [85]. Ablation techniques are commonly employed where the total system is known, but uncertainties exist about the relationship between various compositions and functioning. By damaging and/or removing certain components in a controlled setting, ablation studies can investigate the effectiveness of each action on the performance and capabilities of the system. [86]

Research on ablation is crucial for deep learning. To create trustworthy knowledge, which is any researcher's goal, one must first understand causality in your system. Ablation is, therefore, a low-effort method of investigating cause and effect.

2.5 Conclusion

From earlier reasearch and testing U-Net has proven to give great results in building detection. Although U-Net's impressive results, it is a deep network with a lot of weights. Therefore the choice was made to take U-Net further and develop a model in combination with PSPNet. The reason for our choice of combining two different convolution neural networks was to exploit the Pyramid Pooling Module to handle all the different scales in PSPNet as well as U-Net's ability to classify roof planes. The challenging task was how to integrate the models in an appropriate way that would give better results than the models individually. An ablation study was necessary when combining, and different numbers of layers in U-Net, as well as different numbers of pooling layers, were experimented with.

Chapter 3

Method

This chapter will explain the process of developing a new deep learning network and the reason behinds its architecture. The advantages and disadvantages of both U-Net and PSPNet are considered and discussed to be used to our benefit. The encoder, decoder, and the PPM of our new network are described in detail. Further in the chapter, the selected method for change detection is presented. The procedure for creating a new network is accounted for, including parts like training and optimizing the model, followed by the final change analysis. Lastly, the various evaluation methods for building detection are explained. Parameters like precision, recall, IoU, training, and validation loss followed by time and space are factors the detection are evaluated after.

3.1 Model

The main contribution of our work is the novel network presented in this thesis. The network is an improvement of state-of-the-art image classifiers, specifically developed for remote sensing classification, in our case, trained to detect buildings. Our network is created by combining U-Net and PSPNet, as depicted in Figure 3.1. The encoder-decoder part is similar to the original U-Net, but a PPM inspired by PSPNet is added between the encoder-decoder. Consequently, our model combines U-Net and PSPNet, which is now called U-PSPNet. Combining two quite different networks takes advantage of each network's superiority while mitigating their disadvantages. Both U-Net and PSPNet have widely used image segmentation networks, each with advantages and disadvantages.

U-Net includes layers of convolution and max-pooling. The max-pooling layer target the most typical and essential features by ignoring the lower pixel values. This efficient way of extracting the most distinct features eminent in classification. However, in a high-level network, the information becomes more and more abstract in the deeper layers, which means that important information may get lost. Nevertheless, U-Net, five layers deep network, has been proven to be one of the most efficient architectures for image segmentation [68].

PSPNet is used for handling objects of different scales and mainly for scene segmentation. Scene segmentation is associated with an environment with multiple classes, often with an uneven distribution. By utilizing PSPNet's PPM, the model is able to distinguish between minor and prominent features. Another consequence of the PPM module is that the model is more stable, meaning the model can learn and contain more global context information with the help of global information. The network will, for that reason, not make mistakes when classifying at a pixel level.

3.1.1 Network Structure

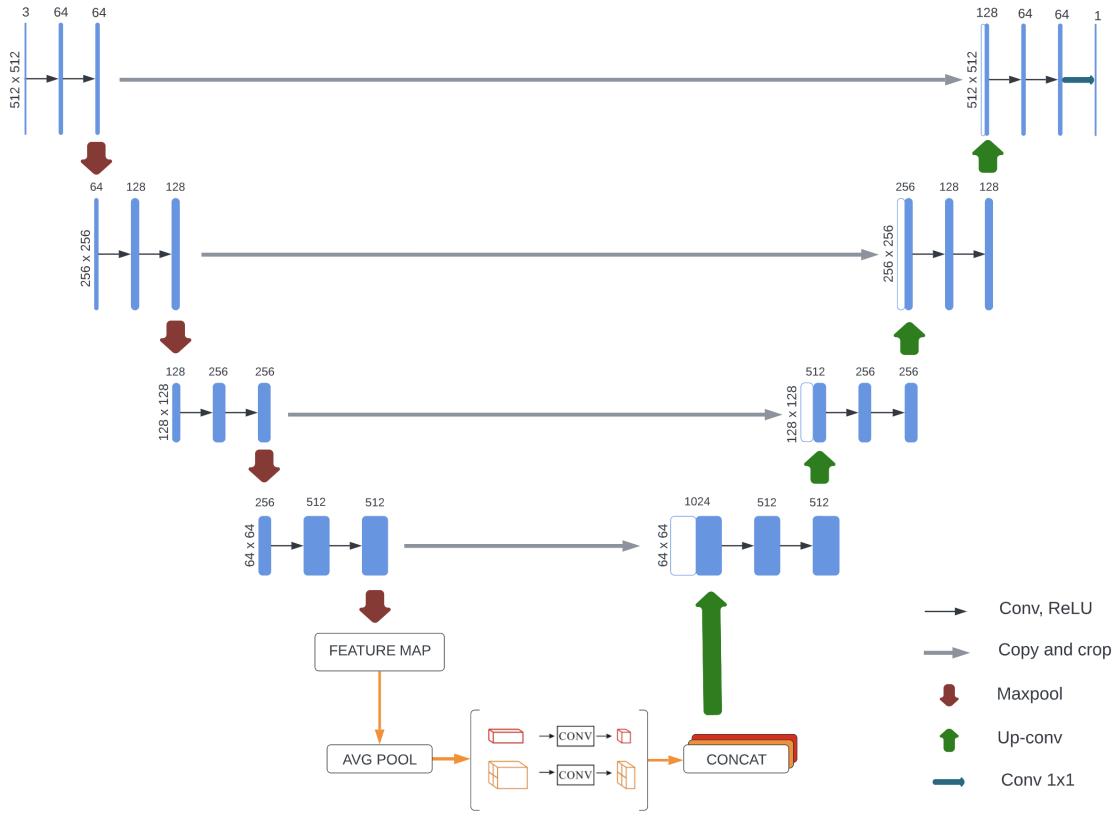


Figure 3.1: Our model combined by U-Net and PSPNet, now called U-PSP-Net.

Figure 3.1 illustrates the proposed network. Initially, the characteristically U-shaped architecture associated with U-Net comes into sight. The difference is adding a part of the PPM from PSPNet. The network can be divided into three parts, the encoder, the PPM, and the decoder. Combining two networks instead of using another established network is to exploit each network's advantages and mitigate its disadvantages.

3.1.2 Encoder and Decoder Module

The encoder part of the network consists of convolution and max-pooling layers, with ReLU as the activation function. The encoder is four layers deep, compared to U-Net's five layers of depth. The decoder is structured similarly to the encoder, except that it is inverted, which means that instead of the max-pooling layer, the decoder part uses an up-convolution layer. In addition, an additional convolution is added at the end to create the binary prediction. As mentioned, U-Net is proven to be an effective image classifier. However, this comes at a cost. U-Net's depth with multiple convolutions results in significant quantities of weights. This again affects the memory and time consumption when training and applying the model. Furthermore, it can also restrict particular possibilities when choosing hyperparameters.

3.1.3 Pyramid Pooling Module

PSPNet's PPM gathers contextual information by utilizing a 4-level pyramid. The different sized kernels extract information at different levels of context. In the original PSPNet, two larger kernels are in addition to the two kernels in our model. The larger kernels are used to extract more preeminent information in a global context. The two smaller kernels are used to extract more local information. Owing to the fact that the buildings are roughly the same size and cover an equally small area of an image, only the two smallest kernels of the PPM are needed. The bigger kernels might have been considered and contributed to a greater extent if the building roofs had covered a larger area of the images. This was also emphasized in the ablation study conducted in Section 5.1.2. In order to take full advantage of the pyramid pooling module, the convoluted images should not be too abstract. This is a delicate balance and is one of the reasons we have chosen to only include four depth layers in the encoder and decoder modules. Using an average pool for downsampling before the PPM, information about the general context of a block is retained and assists in capturing more context information.

3.1.4 U-PSP-Net

The network is developed for building segmentation from remotely sensed images. The whole development process is based on creating a network that outperforms generalized networks. Considering our goal in the first place was to detect buildings, which are a relatively small part of an image, the pyramid scales of a larger size, were assumed only to affect the results to a small degree. For that reason, the model was tested with all the pyramid scale resolutions and removed convolutions from green to yellow one by one. The optimal solution was using only the two smallest pyramid kernels and reducing the original depth of U-Net by one. The network has fewer weights by adding the reduced PPM and removing the deepest layer of U-Net before the decoder part. This mitigates the issue with U-Net and its large amount of weights caused by the deepness of the network. By reducing the network by one layer, only half as many weights make up the network, significantly affecting the time and memory used.

3.2 Method for Change Detection

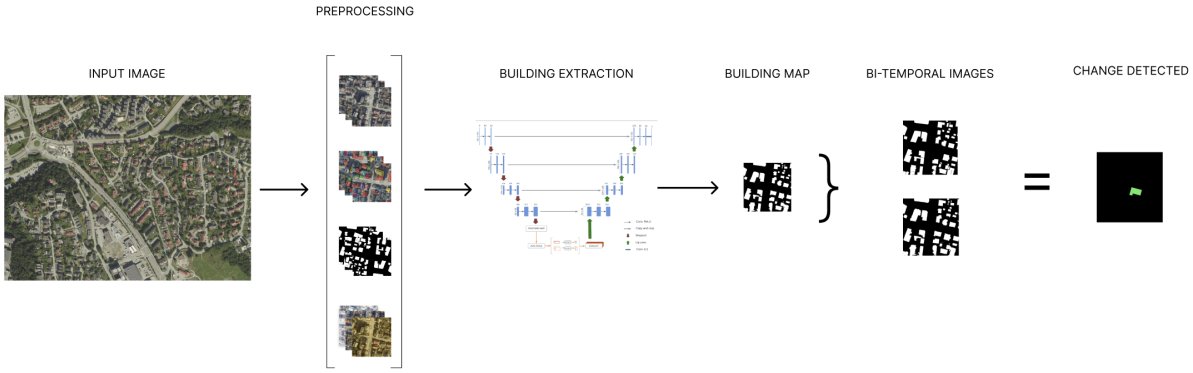


Figure 3.2: Flowchart describing our proposed method for change detection.

The method for change detection includes several steps to get the final segmented map with change. The flowchart in Figure 3.2 visualizes our proposed method, from the input image to the final change detection map. Preprocessing includes cropping, annotation, mask creation, image augmentation, and patching. Further on, buildings are extracted using our new network, U-PSP-Net. When the buildings in two time-stamps are detected, the change is caught in a change detection map.

3.2.1 Creating a Network

When using neural networks for change detection, it is essential to use a network that will benefit your task. This is not always an easy task. Over- and under-fitting are issues that you want to avoid. The characteristics of the problem itself should give clues to what type of model that is beneficial. However, an ablation study targeting to decide the best composition is recommended. Our project utilized two popular CNN in order to detect buildings. A novel network for detecting buildings has been created using an ablation study to optimize the architecture.

3.2.2 Training, Optimizing and Applying the Network

Diverse settings must be configured when training a convolution neural network or deep learning model. Following the last step, where the network architecture is created, the network needs to be trained. This can be accomplished by feeding the network image pairs corresponding to building images and binary masks.

Batch size

Larger batch size can reduce the training time but also has other advantages. One of those is that the network handles more samples from the data set simultaneously. This is beneficial for the reason that it prevents the network from learning potential anomalies and errors in the data set. Furthermore, the batch size is highly constrained by the computer's computational power, and a larger batch size requires more data to be processed in the RAM simultaneously.

Learning rate

There is no consensus that implementing a learning rate decay is the best way to optimize a network [87] [88]. According to You et al. [87] learning rate decay can help optimize and generalize the network. Smith et al. [88] claims that instead of decaying the learning rate, one should increase the batch size. As already mentioned, increasing the batch size is not an option, so learning rate decay was implemented.

Loss function and Optimization Function

When picking a loss function and optimizer function, one should have excellent knowledge of a neural network's different possibilities and inner workings. Consequently, for the network in question, the loss function and optimizer function are the standard for binary classification problems. Binary cross-entropy loss is used to calculate the loss. This function compares the predicted probabilities to the classes [0,1], calculates the distance, and subsequently scores the prediction based on the distance.

Applying the Model

During the training, the weights through the network will be continuously adjusted, enabling the network to detect buildings. After sufficient training and the network is saturated. Then the model needs to be verified with the test data set before the model can be used for detecting buildings. After applying the model, you are left with a segmented data set consisting of building masks.

3.2.3 Change Analysis

Following the last step of building detection, the images are now segmented. The next step in the process is to extract the change in buildings between years. A matching pair of bi-temporal images are then compared pixel to pixel, analyzing if any change has occurred.

Image subtraction

The comparison is made by subtracting each image from the other. This will produce two new images, each displaying a different type of change. Image subtraction removes each corresponding pixel, as seen in the example Listing 1. Since the images are building masks, they only consist of black (0,0,0) or white (255,255,255), indicating buildings. As a consequence, the resulting image will also be black and white.

```
for each pixel:
    new_pixel_value = image1_pixel_value - image2_pixel_value
```

Listing 1: Code for image subtraction

Mass will be left when subtracting the youngest image from the oldest new building. In the opposite example, demolished buildings will be left where the oldest image is removed from the latest image. The new change images can be added together to visualize the total change.

Visualization

To be able to differentiate between a positive and negative change in building mass, the changes will be marked in individual colors. Green indicates a positive change, an increase in building mass. In contrast, red indicates a negative change, a decrease in building mass. The images can be added together afterward to include both types of change in one singular image. Resulting in an image depicting both types of change visualized in a distinct color.

3.3 Evaluation

As with any machine learning model, obtaining a good fit is extremely important. The aim is to balance biases and variances, or in other words, to find a balance between under-fitting and over-fitting. A machine learning prediction is usually categorized in four ways: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).

Convolutional neural networks are used to solve a wide range of computer vision tasks, such as image classification, object detection, and semantic segmentation. A thorough evaluation is required to justify the need for another model by developing a new model for building detection. Different factors will be considered when evaluating the model.

3.3.1 Precision and Recall

There is often tension between precision and recall. The recall is usually reduced when precision is improved, and vice versa. In essence, precision is the ratio of True Positives to all Positives. Using recall, you can determine to what degree the model has covered the True Positives. The recall parameter gives the number of pixels from each building that have been correctly identified. Precision and recall by themself can give misleading numbers of the classification results. It is therefore important to view them in light of each other. This can both be used to understand the model's performance directly. However, if the values differ significantly, the values can be used to understand why the model is underperforming. A high recall and low precision indicate over-classification and vice versa. Equation 3.1 and 3.2 shows the formulas for calculating precision and recall respectively.

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP)\ +\ False\ Positives\ (FP)} \tag{3.1}$$

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP)\ +\ False\ Negatives\ (FN)} \tag{3.2}$$

3.3.2 Intersection over Union

By using the Intersection over Union (IoU) metric, or Jaccard index, we can calculate the degree of overlap between our prediction output and the target mask. In Equation 3.3, the area of overlap between predicted and ground truth building pixels is calculated. IoU is often used in classification problems and gives a good indication of the results in itself. It is therefore a good benchmark to compare different models.

$$IoU = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP)\ +\ False\ Negatives\ (FN)\ +\ False\ Positives\ (FP)} \tag{3.3}$$

3.3.3 Training and Validation Loss

Throughout the training, multiple metrics are calculated and monitored. Two of those values are training and validation loss. The development of these losses over time is widely used in machine learning. In the process of training the network, losses can be plotted to monitor the progress. The most important part of evaluating training and validation loss is their relationship. If the training loss is much greater than validation loss, this is a sign of underfitting. And the opposite case is a sign of overfitting. Both over- and underfitting are to be avoided as it indicates that the model is not optimized. Therefore the plots of training and validation loss will be looked at during the development of the network.

3.3.4 Time and Space

Another essential factor that should be taken into account when comparing models is the time of both the training and prediction. This is closely related to the number of trainable weights in the network. The more weights in the network, the longer it takes to update them after each epoch. Additionally, feeding images through the network is time-consuming, and this directly influences prediction time. In a small-scale experiment, the time factor is not that relevant. However, the time factor is important if the experiment is scaled up with larger data sets for training and prediction. By this account, it is essential to consider the number of parameters in the network.

Chapter 4

Data Set over Trondheim Municipality

Images can display information about objects frozen in a specific moment. There have existed cameras that can capture images since the early 19^{th} century. Still, it was only later in the start of the 20^{th} century that remote sensing became a viable method to collect data over the earth. During World War I, remote sensing became inevitable and shifted towards more aerial photography. Evidently, the images collected were of value to the military. Due to this, resources and recognition have been directed toward aerial photogrammetry specifically. The need for remote sensing only increased as the world went through World War II and the cold war. Yet again, the equipment kept improving, and researchers started using the same equipment for other uses. As early as the 1950s, aerial photography was used for cartographic purposes. In the 1960s, satellites were used for meteorological purposes. The 1970s saw the launch of Landsat 1, the first of many satellites whose sole purpose was to collect images of the earth systematically. Time progressed, as did the cameras, software, and collection methods. Everything is nearly perfect, from high-resolution images taken today with satellites and images collected with air crafts with extreme quality to images and laser scanned data from drones. This project utilizes images collected with an airplane with a 0.1 x 0.1 meters resolution. [89]

4.1 General information

4.1.1 Camera and Flight

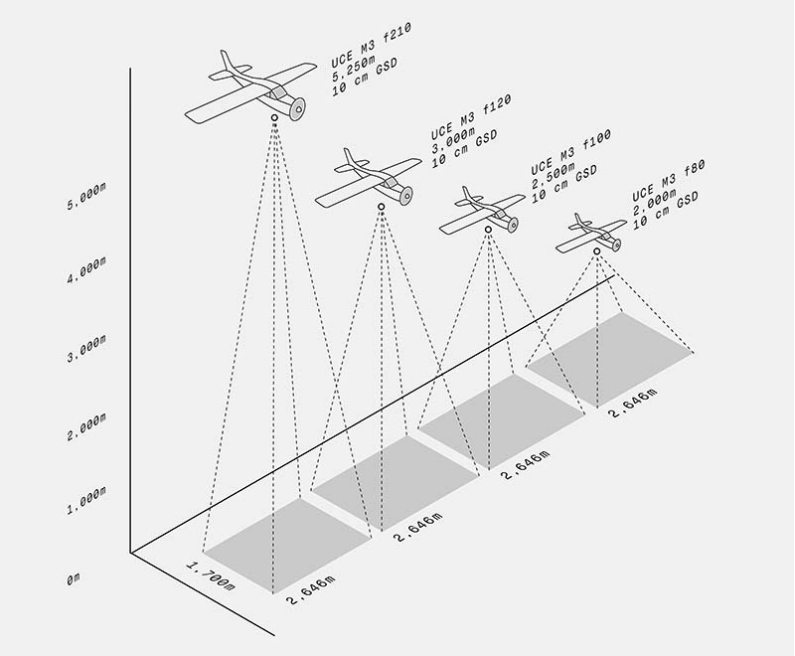


Figure 4.1: Illustration showing how the data was collected [90].

The images were collected using a Commander 690C airplane in April and June 2019. In total, 976 images were captured during 31 flight strips. The resolution of the collected images was 17.004 x 26.460 pixels. Five different bands were collected, red, green, blue, panchromatic, and infrared. The overlap of the images was, on average, 39% for the lateral overlap and 60% for the forward overlap. Camera constants are shown in Table 4.1, where type, focal length, calibration date, FMC and spectral bands are shown. The airplane was traveling at a height of approximately 2500 meters above sea level during the flight. The chosen camera for the job was a Vexel UltraCamEagle M3 mounted on a Leica PAV80 gyro mount. The camera has a footprint of 450 megapixels with four different interchangeable lenses. The camera calibration was done only a few months earlier in the same year. The way the images are collected can be seen in Figure 4.1, and this specific case corresponds to the airplane second furthest to the right.

Camera constants	
Type	Vexcel UltraCamEagle M3
Focal length (mm)	100, 500
Calibration date	11.02.2019
FMC	YES
Spectral bands	PAN, R, G, B, IR. 14 bit

Table 4.1: Camera constants of the camera used for the aerial images

4.1.2 Processing

After the collection of the images, the images were all processed using different softwares and methods. The collected images do not necessarily need to be processed. However, this is done to improve their quality and the ability to extract information from them [91]. Various algorithms and methods can be applied to enhance the images, and the methods will reflect the intended use at a later stage. In order to train the model, images were orthorectified, radiometrically processed, and contrast and color corrected for defects caused by rectification and mosaicing.

4.1.3 Orthorectification

Orthorectification is the process of geometrically correcting the images by projecting them onto the terrain model. This process makes the images more accurate as they consider the terrain's slope. The software used for rectification is *OrthoMaster* which is the most precise method. Instead of using high values in certain intervals, this method uses high values for each pixel. Using this method when projecting the image onto the terrain model ensured the most accurate representation. Although this process properly projects the images on the elevation model, it does not mitigate all relief displacement, which means that the images used are orthophotos and not true orthophotos.

4.1.4 Radiometric processing

Following the standard, it was emphasized to achieve the best possible contrast and color in each image. Additionally, keep the set of images as homogeneous as possible. Radiometric features from the camera were calculated during the calibration and used to correct the images. In this process, information from the overlapping area between sub-images is also utilized, thus obtaining the most homogeneous composite image possible. The software used for this process was *OrthoVista*. A function called *"Global tilting adjustment"* was used to observe differences in intensity, contrast, and color between overlapping images and flight strips and calculate relative corrections that compensate for the differences.

4.2 Preprocessing

The following section will describe the process of choosing an area and preprocessing the data distributed by Trondheim municipality. The preprocessing of the data set, including steps like cropping, annotation, mask creation, image augmentation, and patching, are considered.

4.2.1 Area of Interest

All images used are from the greater area around Trondheim, Norway. There exist publicly available imagery of most of Norway. However, our city, Trondheim, is an excellent representation of a town in Norway, so it was chosen for this project. Since the task relies on data from multiple years, selecting an area that meets these requirements is necessary. This is also true for Trondheim and emphasizes the choice of picking Trondheim. It should be noted that Trondheim is a large city compared to other Norwegian cities but becomes microscopic compared to large metropolitan areas like Tokyo or Los Angeles.

Trondheim municipality extends almost 350 square kilometers, and a large part is uninhabited or sparsely inhabited. Because of this, specific areas for the data set had to be manually selected, and those areas had to meet particular criteria in order to be included. Appendix A depicts the area of images distributed by Trondheim municipality divided into several map sheets.

- Densely populated
- Limit the amount of relief displacement
- Not include buildings with rare features

Choosing the most suitable map sheets was highly reliant on finding the areas with enough roofs of our interest. The criteria were chosen to aid the network's ability to learn. An image sparsely covered with buildings will have difficulty learning the relevant features. This is a consequence of the uneven distribution of the different classes, buildings, and backgrounds. An uneven distribution will lead to good artificial results early on and a hurdle for further learning. Relief displacement is avoided for the same reason. The facades will not be included in the annotation, but seeing as they are so closely related to the roofs of the buildings, this can influence the results, especially if the data set is not big enough. In this setting, buildings with rare features are defined as buildings that are uncommon in the area of research. Buildings with rare features include downtown areas with urban structures, the port area, and industrial areas. These are avoided because they are too rare for the network to learn their features reliably. Which again would result in misclassifications and worse results. This issue is a consequence of the research area and not a deficiency in the model.

4.2.2 Preprocessing and Data set

A training data set with high quality is a crucial prerequisite for a precise classification result. Multiple factors need to be considered, for example, the resolution, the accuracy of the annotations, and the area. The steps during the preprocessing are essential for the results, and multiple courses of action can enhance the outcome. Some frequently used preprocessing methods are image augmentation, cropping, and color correcting. The primary focus of this step in the procedure is to create the best foundation for the model to perform later stages in the process. Preprocessing will make the images easier to interpret and assists the network in the learning phase.

A data set is divided into a training data set used to train the model, a test data set to test the data set, and a validation data set. The validation set is used for validation during the training, and it calculates metrics like recall, precision, and intersect over union and can be monitored during and after the training. Likewise, training loss and validation loss are also valuable metrics to monitor the training.

4.2.3 Cropping

Images can be captured in various sizes, meaning the images needs to be processed to get the correct dimensions. The images must be cropped to establish a base size for all images fed into the network. Earlier research showed that the ideal area covered by an image was about 60 x 60m. According to this argument, there should be a balance between the classes.

Cropping was done directly in Python, and the image size was set to 1024 x 1024 pixels. Each cropped image covered an area of about 100 x 100m and would later be patched in even smaller images. The reason for dividing the images at two different steps in the prepossessing process was to save labor. Even though the general area the original image covered was densely inhabited, some areas might not be. To prevent the work of manually going through 4 times as many images to check if they all met the requirements. All images sized 1024 x 1024 pixels were checked to see if they contained a fair amount of buildings.

4.2.4 Software

An annotation tool is required when creating one's data set for a classification task. The possibilities are endless, and one can use everything from Microsoft Paint to V7. Paint only allows drawing on top of an image, whereas V7 is a fully integrated annotation tool that allows for model training, inference, and statistics. The most advanced annotation tools are either expensive or require a subscription. As a consequence, several open-source annotation tools exist as well. The annotation tool of choice became Labelbox, a state-of-the-art annotation tool [92].

Labelbox

For annotating the data, Labelbox was used. Labelbox gives modern artificial intelligence teams access to data-centric infrastructure. In addition to enabling AI teams to build and operate production-grade machine learning systems efficiently, Labelbox provides rapid training data creation in a unified platform. Four products constitute Labelbox's unified platform, where the former has been used; Annotate, Model Diagnostics, Catalog, and Boost (service layer that includes integrated data labeling service).

Data set	
Images	223
Buildings labeled	3784
Covered area	2.3 sqkm

Table 4.2: Numbers of labeled roofs in the different roof type categories

4.2.5 Annotating

The eight most common roof types in Europe depicted in Figure 4.2 were used for the different labels. In addition, a category termed *Other* was used for uncategorized roof types. Labelbox allowed you to mark the different roof planes through a simple click and drag method, shown in Figure 4.4. All annotations have been done with the greatest precision to ensure the highest possible quality. During annotating, it was prioritized to annotate images with houses over taller and bigger buildings. The area of the annotated buildings can be seen in Figure 4.3. The orange covers the area which is used. More detailed images of the area can be seen in Appendix G.

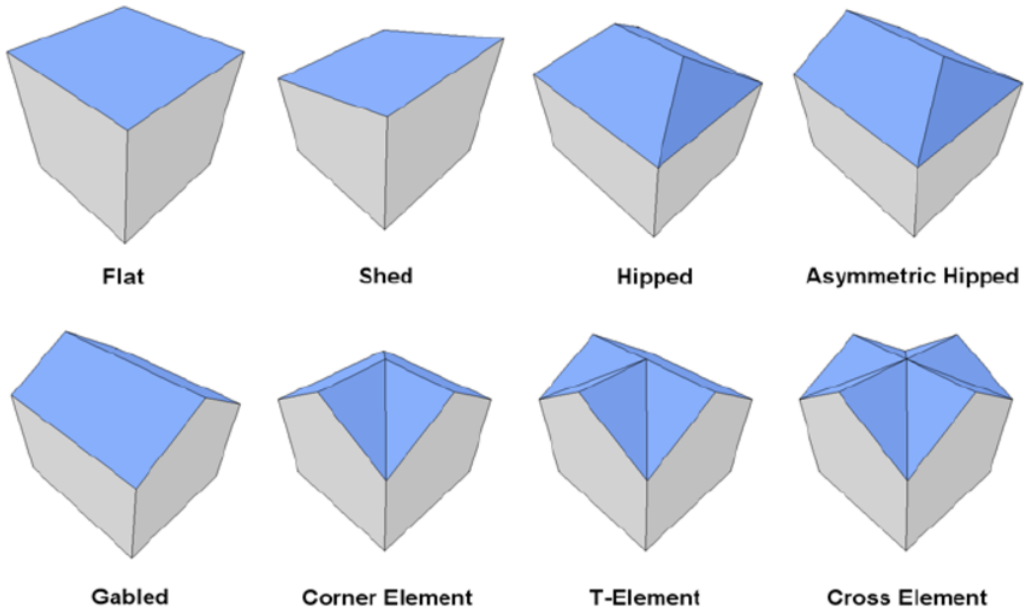


Figure 4.2: The eight most common roof types in Europe [93].

Table 4.3 shows the number of the various types of roofs that are labeled in our training data set, and Table 4.2 presents the total number of labeled roofs and the covered area. Several roof planes consisted of a combination of several types of roof types. All continuous buildings were classified as one building and therefore categorized as Other. An overview of which types of roofs excising in Trondheim municipality would give an impression of which roof types were a minority. Some categories lacked labeled buildings, resulting in incomplete training data. The consequence of insufficient and lacking training data in some special categories impacts the models' ability to detect that type of roof. The roof type categories were, for that reason, not used in training. Nevertheless, they could be used when inspecting the results in case some roof-type categories were misclassified.

Roof type	#
Gabled	1450
Other	778
Flat	624
Shed	414
Hipped	331
Corner element	122
Asymmetric hipped	43
T-element	24
X-element	17

Table 4.3: Numbers of labeled roofs in the different roof type categories.

Figure 4.3: Overview over the data set distribution.

Initially, the annotation process focused on different roof planes and how to identify them. Prior to starting the project, a decision was made to concentrate on roofs in the suburb and leave out tall and prominent buildings. Additionally, interconnected buildings, of which there are more

downtown, were also left out. The reason for this choice was the lack of training data on these kinds of buildings. As the tall buildings may have a lot of relief displacement, it will not be beneficial to include them as a part of the training data set. The overall result might improve by including structures with more relief displacement, but a significant amount of buildings with relief displacement would have been necessary. Trondheim does not have enough tall buildings, with considerable relief displacement for our model to thoroughly learn their features. The number of connected buildings in the city is also small, which results in a lack of training data and the unique features of the buildings, making them hard to categorize.

4.2.6 Creating masks

Building masks are created based on the assumption that building annotation data exists. The annotated data from LabelBox is exported as a nested list consisting of lists of label-coordinates corresponding to each image. The labels are then drawn on a blank black image, the same size as the original annotated image, and saved. The code to create masks can be seen in Listing 2.

```
for each annotated_image in list of annotations:
    Create a blank image of same size as the annotated_image
    for each label in annotated_image:
        Draw label on blank image
    Save mask
```

Listing 2: Code for creating masks.

Before the annotated data could be made into binary masks, they had to be exported from Labelbox. To export all labeled images, a modified version of Labelbox's own Jupyter notebook to visualize data was used. The notebook is linked in the Github repository, which is linked in the Abstract.

The script returns a nested list of all annotations and their class belonging to each image. To clarify, it was only the image coordinates of the labels and not the images that were exported. As the network does not read data in this format, it had to be further processed. All images had the exact dimensions, so no consideration had to be put into the sizes. Firstly a fully black image of the specific size was created. Then the points of the labels were projected onto the black image to create polygons of the buildings.

4.2.7 Image augmentation

To easily expand the data set, all images were geometrically augmented. Image augmentation is a technique applied to expand the data set and improve learning by increasing the amount of training data. There exist multiple ways of achieving this. Some of them are geometrical transformations, color space transformation, and random erasing [67]. Shifting is one of the

multiple techniques to modify an image, to expand the number of training data. When working with small data sets, it is necessary to create variations of the desired pattern to avoid overfitting [94]. The image augmentation script is shown in Listing 3.

```
for image in data set:
    flip_vertical = flip image vertically
    flip_horizontal = flip image horizontally
    flip_both = flip images both vertically and horizontally
    contrast = Randomly adjust the contrast
    brightness = Randomly adjust the brightness
    color_intensity = Randomly adjust the color intensity

    Add all new images to data set
```

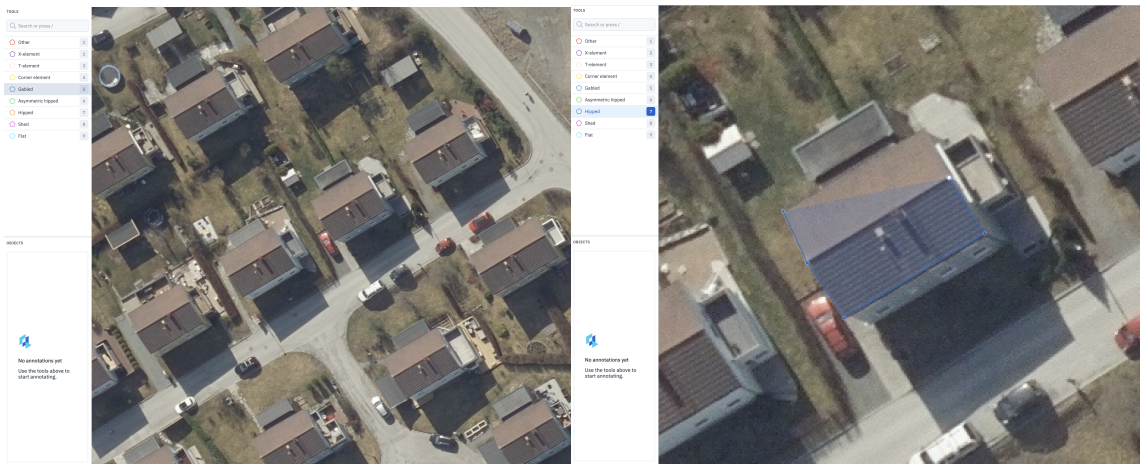
Listing 3: Code for augmenting the images and expanding the data set

The same augmentation was done to each image. The simplest of the augmentation methods are geometric transformations, some of which got used to expand the data set. And thus making the data set four times larger in a matter of seconds. The augmentation methods for expanding the data set also included a random change of contrast, brightness, and color intensity. The data set now included six copies of each original image, each with its augmentation.

4.2.8 Patching

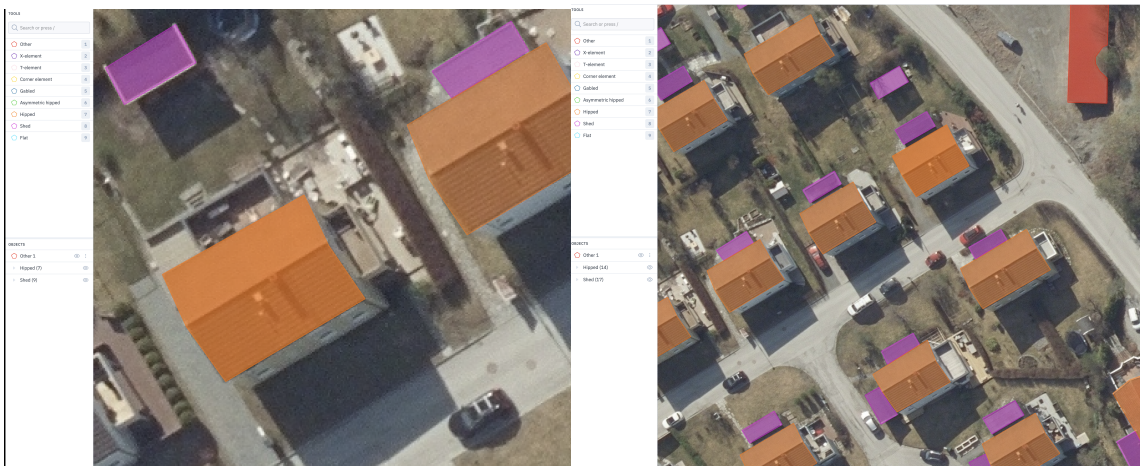
As previously mentioned, the images are yet to be the ideal size for the network, and all images still need to be cropped once more. However, in this step, it is done differently. Two benefits come with dividing an image into patches. Firstly, the CNN handles the patched image with a reasonable size instead of the whole image. A second advantage of dividing images into patches is the increased number of samples, which is generally ideal when training a CNN [95].

Patching was used to minimize the amount of data stored locally and keep as much of the process in the pipeline as possible. This is a feature in Tensorflow that lets you extract patches of the images. Four patches were extracted from each image, and all patches are stacked in the depth of the output tensor. At this point in the preprocessing, images are at a dimension of 512 x 512 pixels and cover an area of 51 x 51m.



(a) Overview of unlabeled image.

(b) Selecting roof type and belonging corners.



(c) Labeled roofs.

(d) Overview of labeled image.

Figure 4.4: Roof type annotation using Labelbox. Different colors represent different roof types, where orange represent *Hipped* and pink *Shed*.

Chapter 5

Experimental Study in Trondheim

In this chapter, the method described in Chapter 3 is adapted to the image data set over Trondheim municipality. The process of the experimental study, as well as the following results, will be presented in this chapter. The experimental study takes place in Trondheim, Norway, focusing on buildings in the suburb. Prominent, unusual buildings and buildings in urban areas were left out of our research because of the lack of that particular type of buildings in Trondheim.

The experimental study aimed to test the various approaches in practice and how the different models behaved over the same area. Firstly the ablation study and the belonging result are presented. Tuning and testing different model hyperparameters and the reasons for our choice are accounted for in this chapter. Considering our model uses U-Net as the backbone, we found it appropriate to compare the results from our U-PSP-Net model against the original U-Net and U-Net with PPM. This chapter will present graphs and tables with every 5^{th} epoch to give an impression of how the models behave. Lastly, the building and change detection results are presented, followed by a discussion of the outcome.

5.1 Ablation Study

An ablation study was executed step by step from the already working U-Net model. To achieve the best possible model, depth and convolution layers were added and removed before being compared against each other. The ablation study allowed us to begin with a functioning model and, step by step, remove parts and tweak it to a better version.

To perform the ablation study, components were gradually removed from the network. As part of the ablation study, different sized convolutions were removed from the Pyramid Pooling module to obtain the best possible network. Additionally, a deeper network than initially was set up when testing the depth, so layers gradually could be removed to find the ideal depth.

At the beginning of the ablation study, the network was five layers deep as the original U-Net and included all four convolution sizes from the PSPNet architecture. Considering earlier work, the network would most certainly overfit the data, and this might not be true for all circumstances.

The model used U-Net's architecture as a starting point, illustrated in Figure 2.9, with a depth of 5 layers. Different depths and numbers of convolutions were experimented with during the ablation study to get the best result possible. Table 5.1 and 5.2, shows the varying results of the different number of layers and convolutions. The optimal combination was four layers deep and the two smallest convolutions from PSPNet. Table 5.5 shows the ten first epochs and their respecting values.

5.1.1 Evaluation during the Ablation Study

As is for everything else, to guarantee a fair comparison between the different runs of the network, every factor was the same. The only aspect that differentiated between the runs was which components were included. Each configuration of the network was trained for at least ten epochs and was later compared at this stage. Multiple metrics were used to compare the results of the different compositions of the network. Intersect over union, precision, and recall was used, as well as a visual comparison of the training and validation loss graphs.

Gradually during the ablation study, the depth decreased until the result started getting worse. All test runs were executed under the same conditions, including the same training data and hyperparameters. When the optimal depth was found, the next step was checking the number of convolutions and sizes that would provide the best results. Since buildings and edges, in particular, are relatively small, it was assumed that the smallest convolutions would have the most significant impact. Therefore a decision was made to exclude a single convolution after each test, starting with the largest.

5.1.2 Results of Ablation Study

In determining U-PSP-Net's architecture, the number of Pyramid Pooling Layer (PPL) was considered. As mentioned, the original PSPNet has four convolutions for the various resolutions, respectively, in the colors red, orange, blue, and green. Table 5.1 shows the different results of various convolutions after ten epochs. There is no considerable difference between the different configurations, but the best result is with the two smallest kernels.

Layers	Binary IoU	Precision	Recall
ROBG	0.8798	0.9128	0.9073
ROB	0.8747	0.9094	0.9023
RO	0.8822	0.9147	0.9091
R	0.8810	0.9166	0.9046

Table 5.1: Results of U-PSP-Net during ablation study were different Pyramid Pooling layers from PSPNet are included.

Depth	Binary IoU	Precision	Recall
5 layers	0.8801	0.9102	0.9103
4 layers	0.8822	0.9147	0.9091
3 layers	0.8364	0.8851	0.8600

Table 5.2: Results of U-PSP-Net with different depths.

When testing the depth, the network started with five layers of depth. As for the number of Pyramid Pooling layers from PSPNet, the various configurations did not significantly differ. Both 5 and 4 layers gave a binary IoU of 88% and precision and recall of approximately 90-91%. However, a four layer deep model consisting of the two smallest sized kernels gave the overall best results. The outcome of our ablation study results in the model depicted in Figure 3.1.

5.2 Hyperparameter Optimization

There are numerous ways of improving a network. In addition to refining or expanding the data set, contextualizing the data, changing the model architecture, cross-validating during training, and tuning hyperparameter, there are many other options. This section explains how the hyperparameters were tuned during the project, why it was done that way, and the limitations and obstacles encountered in the process. The environment used in this process was kept the same through all tests. Each hyperparameter was tested independently, a measure to quickly identify each change's effect. Preliminary tests have already been conducted in earlier work, so most of the testing is just done to confirm our assumptions.

5.2.1 Batch Size

There was a limitation to how much we could test the maximum batch size cause of the RAM size. Therefore, the batch size was 4, which was the maximum possible considering the RAM. Ideally, a larger batch size would have been preferred, be that as it may, this was impossible without causing an overflow in the RAM.

5.2.2 Learning Rate and Reduction of the Learning Rate

Based on research and earlier work, the learning rate was initially set to 0.0001. The value was both increased and decreased to measure any improvement. However, it was anticipated that the initial value gave the best results. One value that did change after experiments was the learning rate decay patience. To begin with, the patience was set to 10 epochs. Following experiments with different values, the decision fell to change it to 5 epochs. The consequence of this change was that the learning rate would be reduced by a factor of 0.1 after 5 epochs. The reduction is triggered after 5 consecutive epochs without a difference more significant than 0.0001 in the validation loss. This change gave better results and made sense observing that the network quite rapidly became saturated.

5.2.3 Loss Function and Optimizer Function

Adam optimizer was used as the optimizer function. The name Adam is derived from adaptive moment estimation, and the function is a stochastic gradient descent function. The optimizer function is used when updating the weights throughout the network and is an iterative function to find the best fit.

5.2.4 Epochs and Early Stopping

The number of epochs is highly dependent on the size of the data set and the speed at which the network acquires knowledge.

As mentioned earlier, an epoch is when the network has passed through the entire data set once. This project aimed to get the best possible results; thus, the network trained until it was saturated. In other words, disregarding the testing, there was no upper limit for how many epochs the network would train for. The network, however, only terminated if the early stopping function was triggered.

The patience of the function was set to 10 epochs, and the default value of 0.0001 was used as the threshold. It became evident early on that the network became saturated relatively fast, and the number of epochs for comparing later ended up at 27. This function ends the training after a given amount of epochs without a set amount of change in validation loss, similar to how learning rate decay works.

5.2.5 Visualization and Measurement tool

There exist numerous tools to track and measure the network during experimentation. Tools like Neptune, TensorBoard, Guild AI, and Sacred are all viable options, where TensorBoard, by far, is the most popular. The tool is of great help in visualizing the contribution of each hyperparameter, tracking the progress of the model, and the development in the training and validation loss. This tool was unfortunately not used in this task, and will be discussed later in the discussion.

5.3 Our Results

Our model was tested against U-Net and U-Net with the original pyramid pooling module to evaluate the characteristics of our model. Based on the comparison shown in Table 5.6, our model, U-PSP-Net appears to be the best choice both when it comes to binary IoU, precision, and recall regarding building detection. The evaluation of change detection highly depends on the assessment of the detection of buildings. The model is dependent on getting a precise building detection to get an accurate change detection. The U-PSP-Net's Binary IoU is 97%, which is excellent.

Binary IoU, training loss, learning rate, precision, and recall were essential factors in presenting the results from the different models. As a consequence of early stopping, all the models are compared at their 27^{th} epoch to make a fair comparison. A graph showing the training and validation loss is also a significant factor when evaluating other models. A big gap between the validation and training loss indicates an overfitted model.

5.3.1 U-Net

Table 5.3 shows every fifth epoch to 27, where you can see that the binary IoU, precision and recall slowly increases. The loss and learning rate is decreasing which indicates a slow down in knowledge acquisition. Graph in Figure 5.1 shows some outliers in the validation loss, and a small gap between the training and validation curve. The complete result of each epoch from the original U-Net are showed in Appendix B.

Epoch	Binary IoU	Loss	Learning Rate	Precision	Recall
5	0.8487	0.1437	0.0001	0.8901	0.8769
10	0.9010	0.0907	0.0001	0.9289	0.9262
15	0.9189	0.0731	0.0001	0.9434	0.9395
20	0.9610	0.0354	0.00001	0.9715	0.9739
27	0.9668	0.0300	0.000001	0.9756	0.9781

Table 5.3: Result of U-Net to 27$^{\text{th}}$ epoch.

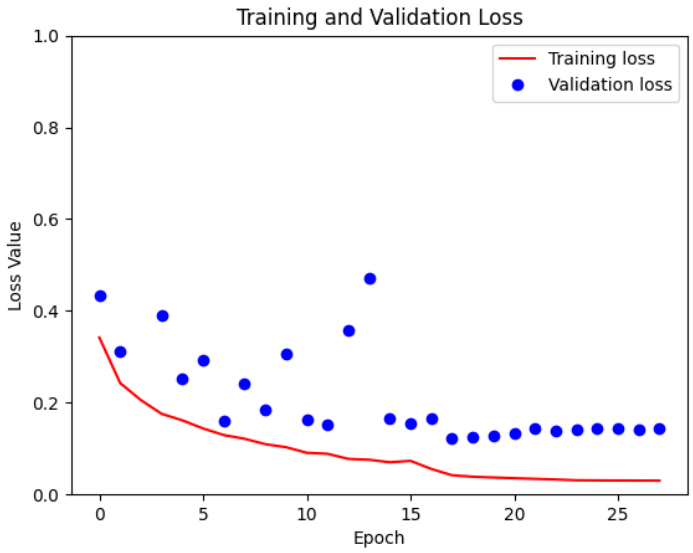


Figure 5.1: Training and validation loss graph for U-Net.

5.3.2 U-Net + Pyramid Pooling Module

Table 5.4 presents the results of U-Net with Pyramid Pooling Module from PSPNet added between the encoder and decoder. As in U-Net, the binary IoU, precision, and recall increase, and the loss and learning rate decrease. In Figure 5.2, the training and validation loss are presented. Compared to the U-Net graph in Figure 5.1, there are fewer outliers and an even smaller gap between the curves. Nevertheless, two quite significant outliers are still shown in the plot.

Epoch	Binary IoU	Loss	Learning Rate	Precision	Recall
5	0.8329	0.1591	0.0001	0.8795	0.8598
10	0.8870	0.1031	0.0001	0.9199	0.9126
15	0.9094	0.0814	0.0001	0.9362	0.9326
20	0.9395	0.0537	0.00001	0.9562	0.9577
27	0.9645	0.0317	0.000001	0.8742	0.9764

Table 5.4: Result of U-Net with Pyramid Pooling Module from PSPNet to 27$^{\text{th}}$ epoch.

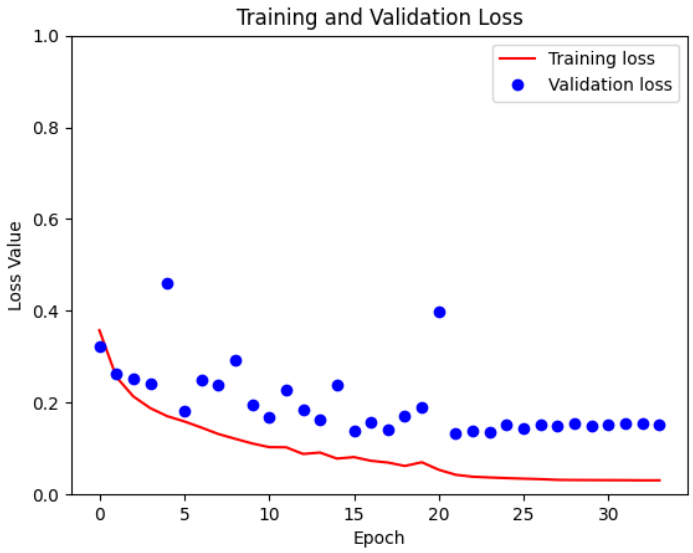


Figure 5.2: Training and validation loss graph for U-Net with Pyramid Pooling Layers.

5.3.3 U-PSP-Net

Our final models result, presented in Table 5.5, consists of U-Net with PPM from PSPNet with only the two smallest kernels. The final results end with a higher binary IoU, precision, and recall than the other models presented. Figure 5.3 shows the training and validation loss graph where the gap between the curves is approximately the same as in U-Net with PPM but without the significant outliers. The validation loss lies closer to the training loss. The complete result of U-PSP-Net, with every epoch, is shown in Appendix B.

Epoch	Binary IoU	Loss	Learning Rate	Precision	Recall
5	0.8344	0.1586	0.0001	0.8797	0.8618
10	0.8845	0.1068	0.0001	0.9164	0.9117
15	0.9222	0.0704	0.0001	0.9436	0.9439
20	0.9581	0.0377	0.00001	0.9691	0.9719
27	0.9675	0.0290	0.000001	0.9762	0.9786

Table 5.5: Result of U-PSP-Net to 27$^{\text{th}}$ epoch.

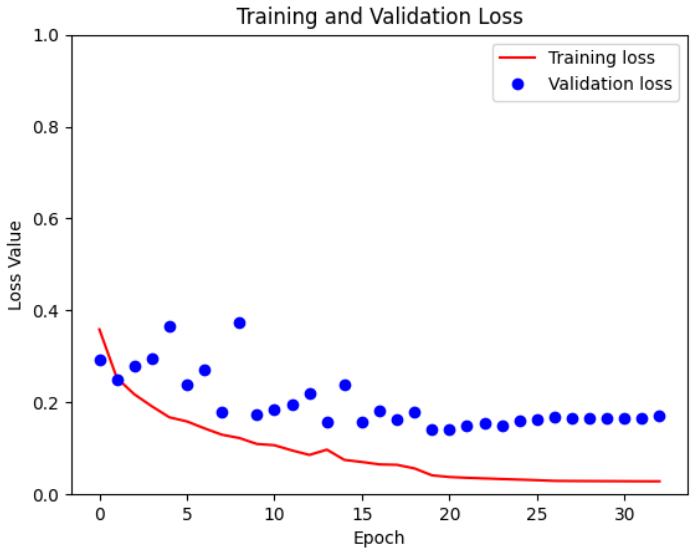


Figure 5.3: Training and validation loss graph for U-PSP-Net.

5.3.4 Comparison of the models

Table 5.6 shows the comparison of the above mentioned models after the 27$^{\text{th}}$ epoch. U-PSP-Net achieves the best binary IoU, precision, and recall. However, U-Net is not far behind. Appendix B shows the entire process of both U-Net and U-PSP-Net.

Model	Binary IoU	Loss	Learning Rate	Precision	Recall
U-Net	0.9668	0.0300	0.000001	0.9756	0.9781
U-Net + PPM	0.9645	0.0317	0.000001	0.9742	0.9764
U-PSP-Net	0.9675	0.0290	0.000001	0.9762	0.9786

Table 5.6: Comparison of U-Net, U-Net with pyramid pooling modules and our U-PSP-Net after 27 epochs.

5.3.5 Parameters

Table 5.6 illustrates our model's different results, versus the original U-Net and U-Net with PPM. The table shows the 27^{th} epoch with belonging values for binary IoU, loss, learning rate, precision, and recall. The different models' binary IoU on the 27^{th} epoch is respectively 96.8% and 96.7%. This is not an enormous difference, but the number of parameters is not considered. Table 5.7 illustrates that the number of total parameters in U-Net is doubled compared to U-PSP-Net. Precision and recall are pretty much the same, but U-PSP-Net is a fraction better.

	U-PSP-Net	U-Net
Total parameters	16 168 513	31 055 297
Trainable parameters	16 160 833	31 043 521
Non-trainable parameters	7 680	11 776

Table 5.7: Parameter comparison of U-PSP-Net and U-Net.

The final results do not show significant differences between the different models. However, as seen in the results in Table 5.6 and 5.7, the model proposed in this thesis outperformed U-Net by only using half as many trainable parameters. Any model will benefit by using a good training data set over a homogeneous area. Nevertheless, the results speak for themself when our model outperforms one of the most recognized CNNs available using far fewer parameters. The novel model proposed in this thesis might not be as versatile as U-Net, yet it can predict buildings more accurately than what it is up against, which is the model's sole purpose, to detect buildings. The building detection was immaculate when only considering building detection is done on the same data set as the model was trained on. A 100% perfect automatic building detection is almost impossible. Nevertheless, it is possible to get close. This is exactly what we achieved, near-perfect results. There were, of course, some issues with the building detection, which will be discussed.

5.3.6 Building Detection

Figure 2 in Appendix C, depicts the building detection results over U-PSP-Net from different areas in Trondheim. The predicted buildings are white with the background being the aerial image. Most of the buildings are fully covered, and the polygons have sharp edges that distinguish the different buildings.

Some of the areas of building detection were not as good as the rest. Figure 3 in Appendix C, shows examples of building detection with misclassifications. Areas with more unusual buildings and urban areas seem to give worse building detection results. The detected buildings are incomplete roof planes, include small holes and does not have sharp edges to distinguish each roof clearly. Other objects like roads, landscapes, and facades are also in some cases detected as a buildings.

Figure 2 and 3 in Appendix C, shows the results of the building detection. As the image depicts, the results are outstanding. The great results are caused by a combination of a well-made model, a homogeneous area, sufficient training data, and a high-resolution data set. Our model is thoroughly developed and tweaked to get the best possible outcome. The area from which the training data is from, is homogeneous and is similar to the general area where the predictions were made. This is an essential factor for our outcome. Additionally, the high-resolution data set is a beneficial factor in producing good results.

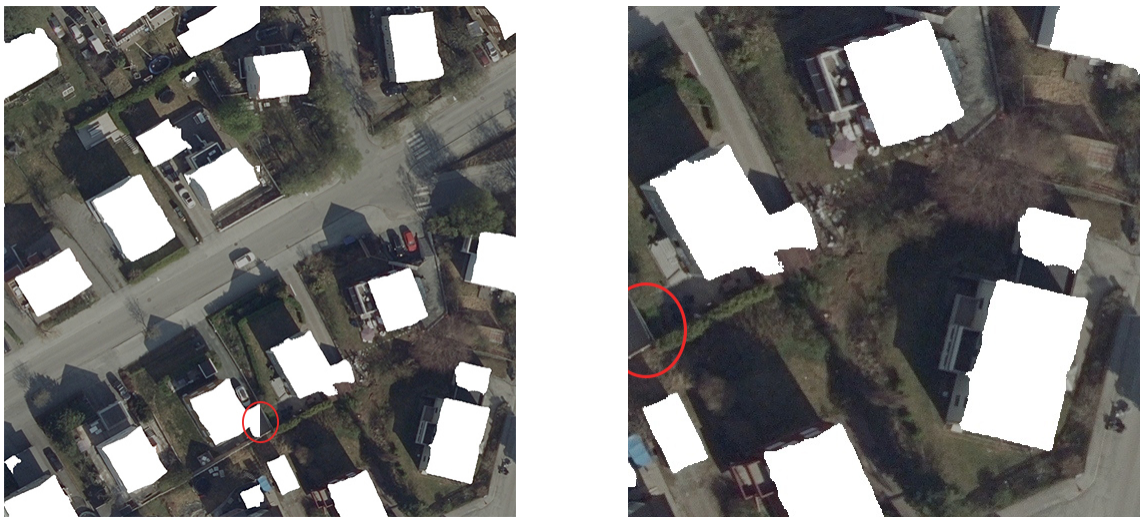


Figure 5.4: Struggling with detecting image in the edge of an image.

Nevertheless, such a positive result has not been without its problems. The prediction result had some misclassifications, and the worst results are shown in Figure 3 in Appendix C. Some roofs were left out, while some facades were classified as rooftops. Urban areas seemed hard to predicate, as the image shows roads and facades classified as buildings. Unusual buildings were also hard to distinguish, as the training data did not include enough distinctive buildings. Cause of insufficient training data in these particular areas, the building detection did not consist of filled polygons or sharp and distinct edges. Again, this is a problem with the area since such

unique structures and areas do not exist to the same extent. As Table 4.3 presented, the different roof types were not balanced when labeling. The unique structures were often classified as Other, indicating various unique roof types, making it hard for the network to learn their features.

The model also seemed to struggle to detect smaller roofs and roofs cut by the edge of the image, partly existing in both photos. Figure 5.4 illustrates a prediction that shows this instance. The left image is a fusion of four images, where you clearly can see that our model misses out on the roof in the bottom right image compared to the bottom left. It is a clear distinction between the cut marked in the red circle, where the images split, and the prediction does not cover the whole roof. The right image shows a closer look at the missed corner. The model recognizes roofs, but it's easier to identify the bigger and most significant ones. Smaller roofs such as garages and sheds were more problematic, as Figure 5.4 presents some examples. Predicting smaller surfaces or parts of a roof that disappear at the edges seems to be a weakness of our model.

After getting a prediction of rooftops, minor anomalies appeared in addition to the predicted rooftops. Our deep learning model learns what roofs look like and how they are shaped by going through all the labeled tops. The final result may look cleaner by removing the anomalies, but errors may occur. The anomalies are a part of the result of the training and can therefore affect the outcome as well as the IoU, precision, and recall if not removed. Three examples of anomalies in our prediction are depicted in Figure 5.5, where it is predicted roofs where it does not exist.

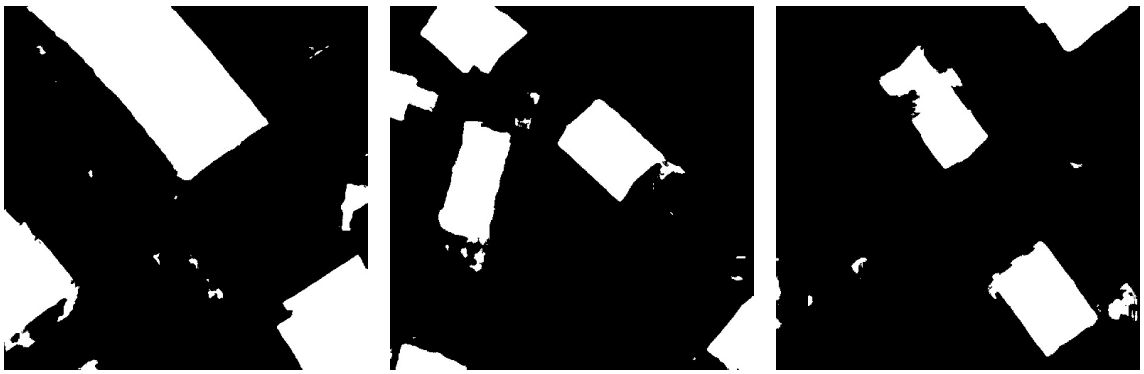


Figure 5.5: Small anomalies in the prediction.

Buildings are regularly square-shaped in some way, with sharp edges. The desired outcome of the building detection was complete polygons covering the roof planes with sharp edges. In some cases, the prediction had more irregular shapes, holes, and not equally straight edges. This is caused by, for example, trees covering the building, unusual roof shapes, or different lighting. Appendix D presents examples of irregular shapes from the result of the building detection. Some roofs were covered by trees when annotating roof planes for training data. Although we were only two persons annotating the buildings, these types of occurrences may have caused some inconsistency when labeling.

Using the model to predict buildings on the data set from 2019 went well. The buildings were consistently detected with only a few exceptions. The prediction did not go as trouble-free when

using the images from 2020. The first effort resulted in a failure. Almost no predictions were made at all, indicating that something was wrong. The different color properties of the data sets had been overlooked, which was now causing an issue. The model was far too specialized on the data set from 2019 to be able to predict any buildings in the images from 2020. The steps to minimize this problem will be further discussed later in Section 5.3.8 Data Set. Succeeding minimizing the problem, the building detection became more than acceptable. The detection results from 2019 are superior to 2020. However, both results were good enough to conduct a change detection analysis.

5.3.7 Change Detection

The final change detection result using our model, U-PSP-Net, is depicted in Appendix F where the left column shows aerial images from 2019, the column in the middle 2020, and the right column shows our predicted change. The green areas represent parts added from 2019 to 2020, and the red is removed.

The two data sets were images from 2019 and 2020 over the same areas in Trondheim. There were only a few changes during one year, making it more challenging to detect changes over such a large area. Additionally, the images were not true orthophotos, meaning that objects that are not flat will lean in the opposite direction from the nadir of the camera. Therefore, it was hard to observe the changes when putting images together since a change was detected in several places where only the buildings had moved slightly. Figure 5.8 shows what the predicted change detection images looked like on some buildings. All the building's outlines are classified as change, even though no change had occurred. This is because of the difference in camera position between the years. To find which areas change had happened, we had to create a method. Two corresponding images were subtracted from each other before they were added together. This method resulted in an image that mainly looked dark in areas without change, and areas with change had more color. The combined image is depicted in Figure 5.6, where the left image shows the overall image, and the marked area highlights an area with change. The right image presents a closer look at the significant area, where you can see that a change has occurred.

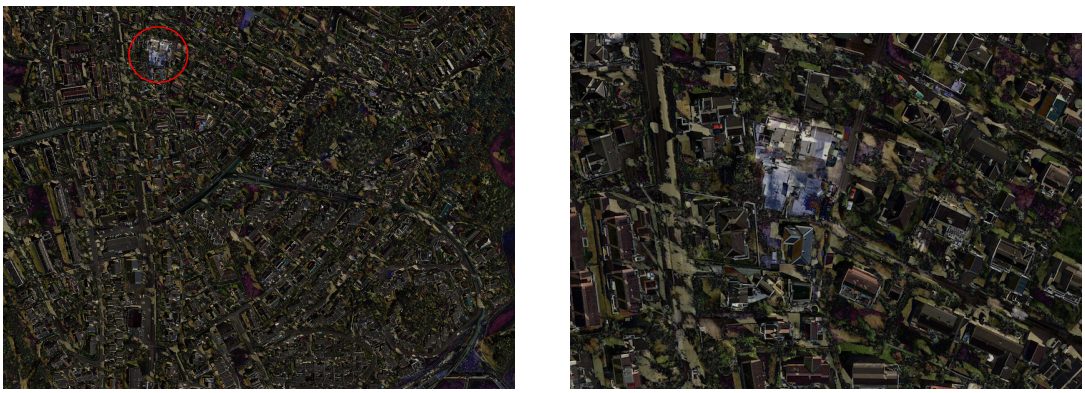


Figure 5.6: Overall image of change detected of a larger area.

A consequence of the images not being true orthophotos is that there will be relief displacement in the images. The images are created by mosaicing multiple images collected during a flight, leading to some unusual-looking images seen in Figure 5.7. As a consequence of the image, the tall buildings are seen leaning in different directions. The effect is exaggerated in taller buildings. However, the result can be observed in every object with a height above the ground. The effect is present in the complete data set, although varying between images. This can be seen in Figure 5.8 where all buildings are of similar type. Based on the thickness of the colored outline, we can tell how much variation there is between images. The thickness is directly related to the distance between the building and the nadir. The thin outline indicates that the nadir of the images belonging to both years is close to each other. As a result of this thin outline, the same building is oriented in the same direction and has the same relief displacement, if there is one.

Regarding the cases with a thicker outline, the nadir of the images was far apart. The image pair was most likely taken from the opposite side of the buildings. Seeing that almost every building's exact location varies between the years, the task of automatic change detection becomes a lot harder. An actual change will be harder to identify as the change caused by the dissimilarity in the data sets are everywhere. Dissimilarity in the data sets is an unfortunate consequence of using images that are not true orthophotos. True orthophotos require more images to be collected, more post-processing, and more labor in general. Therefore there are fewer data sets consisting of true orthophotos, and unfortunately, not many are available.

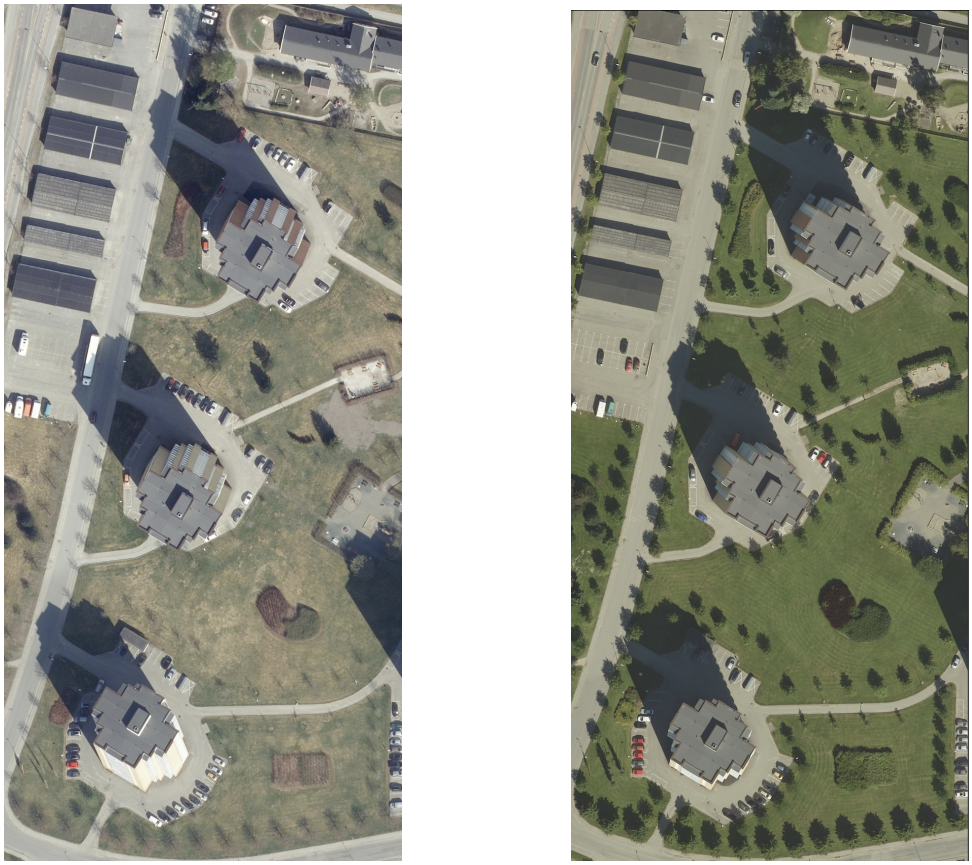


Figure 5.7: Relief displacement of images from 2019 and 2020.

Another complication with the data set is that they are collected in consecutive years. The data sets are collected within a period of 15 months. Considering the construction of buildings is a process that often takes more than a year, the number of changes in the available images was limited. In hindsight, a longer period of time between when the images were collected should be used. However, only the sample size of changes is hindered by this, not the method. Combine the lack of changes with the misclassification of changes, and a problem occurs. Automatic change detection is impossible due to the imbalance of actual change compared to change caused by the difference in the images. The changes then have to be observed manually. A task that also proved to be complicated. Therefore a method to help identify areas with change was developed. Subtracting and adding pairs of bi-temporal images created a new image where the change was visible.

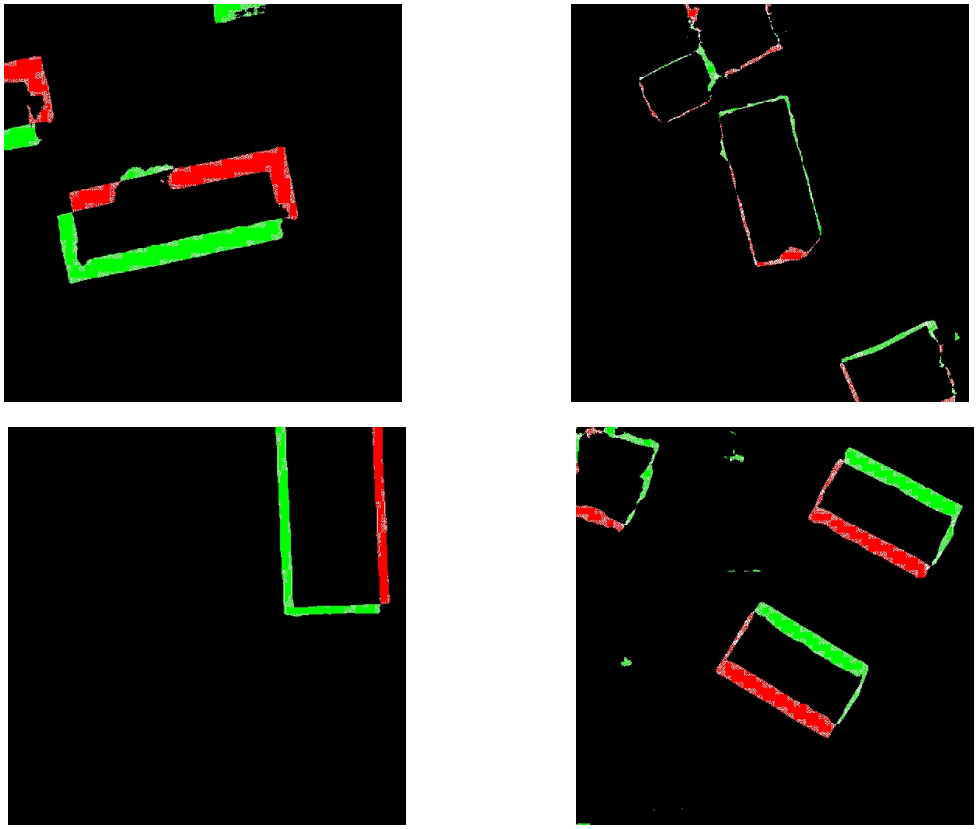


Figure 5.8: Change detected. Red shows parts that are removed, and green are parts added.

Not having true orthophotos makes automatic change detection more challenging to perform. When detecting change, the model does not distinguish between an already existing or new roof when they occur in the same place. An example of this is presented in Figure 5.9 from 2019, where a typical Gabled roof rebuilds into three smaller Gabled roof planes. The building's overlap is detected as no change and a mix of red and green parts from where the 2020 roofs have changed compared to 2019. For the visual aspect of the change detection, buildings detected that are unchanged from previous years should be marked to make it clear that the building still stands. More significant changes would be more noticeable by filling the buildings with bigger polygons.

Figure 5.9: Aerial images from 2019 and 2020, and the change detected.

The results from the change detection analysis do not include any accurate measurements. The simple reason for this is that there does not exist any ground truth data set over the given area. To correctly calculate any metrics, the results need to be compared to the ground truth, which was impossible. The ground truth data set also had to be made from true orthophotos. These, as mentioned, are unavailable images, neither for the building training data set nor for a possible change data set. Without a ground truth data set, this would not have been of any use to us either. Since the detected images are not true orthophotos, it would make no sense to compare them to a ground truth data set created with true orthophotos.

5.3.8 Data Set

Creating our data set has resulted in more accurate results, which is crucial for our problem. Several factors could have affected our outcome if we had avoided making our data set. Since our task involved change detection in Trondheim, training data from a different place would result in an unsatisfactory product. There was no data set with labeled rooftops in Trondheim. Building roofs are different and unique and have particular characteristics for each place you go. The most common roof types in Europe are not the same as in Asia, and some areas may contain more urban areas or taller buildings than in Trondheim. Different regions will lead to variance in roof types and, for example, nature, sunlight, shadow, contrast, and resolution. Training our model on a data set with a lower resolution than the data set distributed from Trondheim municipality would have influenced the change detection accuracy. Accordingly, training data from the right area is crucial.

Images collected at different times of the day, during different seasons, or during different weather will have divergent color properties. The color hue might be more intense, the sun might increase the brightness of the objects captured, or the season will shift the landscape's color. Accordingly, when creating a data set, images are color corrected relative to each other, which is not the case for different data sets. Meaning that data sets, although they cover the same area, will not have the same color properties. A network trained on a data set with specific color properties will perform worse when predicting on a data set with other color properties.

The training data for our model were images from May 2019. To detect the change, a comparison with another year was necessary. Trondheim municipality gave us images over the same area collected in August 2020. This color difference is shown in Appendix E, where the same area of 2019 and 2020 are depicted. We did not consider this when starting to detect buildings, but the result was surprisingly bad. The consequence of images taken from different years and times of the year was a variation of colors. The brightness, saturation, and contrast were all different. Our network had only trained on images from that specific data set. The model did not perform satisfactorily when applying the model to a new data set with different colors. By visual inspection, it was soon detected that the bi-temporal images had other color properties, causing the lack of performance. This was a surprising effect that we had not considered when creating the data set. Given that we made a data set over the area of interest, we anticipated that the color variance in the images would not affect the results to that degree.

Two measurements were taken to mitigate the issues stemming from the color variance between the data sets. Firstly the data set was further augmented. Three copies of each image were added to the data set where contrast, brightness, and color hue were randomly tuned in the copies. A common reason for data augmentation is to make the model invariant to any difference in the color properties when predicting. Expanding the training data set with images with random contrast, brightness, and color hue will increase the model's ability to identify buildings in images from different years. This follows the simple argument that the model is exposed to images that are not homogeneous and will learn to identify buildings in more conditions. The prediction results improved after training the network with the new training data. However, we were not yet satisfied with those predictions. The building detection was still superior on images from 2019 and not good enough on the images from 2020. The second measure we took to improve the building detection was to color correct the images from 2020 to be more similar to those from 2019. This process was done manually in an image editing software. During this operation, the images were tuned by adjusting the saturation, contrast, shadows, highlights, exposure, tint, and temperature. As this was done manually by comparing the image pairs while adjusting, it was not a perfect way of color-correcting the images. Nevertheless, this gave us good enough results to be able to detect buildings in both years reliably.

More measurements could have improved the model if we had sufficient time. Firstly, and most obvious method is to include images from multiple years in the training data. It could be as simple as to include data only from the years in question for the change analysis or to include every year with available data. This would make the model even more general. This is a simple method to mitigate the issue. Still, it is incredibly time-consuming and was therefore not a viable option. Secondly, it is possible to use Generative Adversarial Network (GAN) to transfer the image style and directly generate new training images to generalize the model. Similar to the first option, this method would be too time-consuming. A third option that could have generalized the model is to use black and white images. This would not have fully mitigated the differences between the bi-temporal images. Nevertheless, some of it would be removed. This method would only include changing all the images to greyscale. Yet this was not done, even

though it is not that time-consuming, considering that a benefit was not guaranteed.

Another consequence of using the data set we created was that the network saturated surprisingly quickly. From experience, we anticipated that training the network would use more epochs and, therefore, more time. We suspect this is because of the homogeneous nature of the data set. Since all the images were almost similar, the amount of knowledge the network needs to obtain is severely reduced.

Chapter 6

Conclusion

The final chapter of the thesis summarizes the work and experiment. It will cover an overall conclusion of the network's performance regarding building detection and a conclusion regarding conducting a change detection analysis based on the results from the building detection. Lastly, the chapter will cover recommendations for further work.

6.1 Conclusion

The main objective of this thesis was to create a method for automatic building detection that had to produce highly accurate detections. From a precise building detection, change detection analysis would make a change detection map over the given area. To test the proposed method, an experimental study was conducted. The study started with preprocessing the data set before the images were classified, and lastly, pairs of images were compared to detect change. The thesis also included the creation of a novel network and data set.

This thesis provides a novel deep learning network for building detection used for change detection of buildings. The network was created by combining parts from U-NET and PSPNet. By performing an ablation study, the optimal combination of the two networks was discovered, and U-PSP-Net was created. U-PSP-Net consists of four layers of U-Net's encoder/decoder module and the two smallest kernels from PSPNet's PPM. The architecture proved to be highly effective by exploiting the advantages of both U-NET and PSPNet while diminishing their downsides. U-PSP-Net provided above 97% IoU for building detection by only using half as many weights as U-NET.

The network was trained on a novel building data set created of images over Trondheim, Norway. The new data set was also created as a part of this thesis. By annotating accurate aerial images, the data set was made. The data set covers 2.3 square kilometers and includes over 3700 buildings, making it the first of its kind in Trondheim. The data set is suitable for training neural networks used for building detection. However, it only includes buildings over a local area in

Norway. The network showed excellent results when detecting buildings on images from the same data set as the novel data set was created. That said, the network did not prove to be as effective on other data sets. Building detection on another data set over the same area gave worse results than the novel data set. The network was only trained on the novel data set, a data set that was not general enough.

There are two main reasons why automatic change detection was impossible with the method and data set used in this thesis. Firstly the data set was not suitable for change detection. To perform change detection, images from at least two years are required. A minimum requirement we discovered for those data sets is that they are true orthophotos. As seen in the results, not using true orthophotos creates a lot of false positives due to the difference in nadir when collecting the images. This inhibits an algorithm from reliably detecting changes since most detected change is just a difference in the data sets. The second reason automatic change detection was not possible was the accuracy of the building detection. The building detection from both years has to be highly accurate. In our case, buildings detected on the data set on which the network was not trained were not accurate enough. This also leads to false positives in the change detection.

6.2 Further Work

Change Detection with True Orthophotos

Considering the issues caused by relief displacement, executing automatic change detection was not viable. There were too many misclassified changes caused by relief displacement to filter out the actual modifications in buildings properly. There must be considerably fewer false-positive detected changes to perform automatic change detection. To test the ability of the network and method, we recommend that the method should be tested with true orthophotos. With true orthophotos, the difference in relief displacement will be mitigated, and the change analysis will give more correct results. As long as the building detection is accurate enough, we see no reason why an automatic building detection would not be feasible using our network and method.

Test Method with Generalized Data Set

The network should be trained on a data set consisting of images from multiple years and areas to enhance the method further. This would make for a more reliable building detection, even considering the difference in color properties and the area of interest. Therefore, it is recommended to use the method with an expanded data set that allows for more accurate building detection in both bi-temporal images. This change in the method would most certainly result in a more precise outcome for building detection and, therefore, in the overall change detection.

Further Testing with U-PSP-Net

To test the network's full potential as a general network that can detect buildings in any condition or location, it needs to be tested on another data set from a different area. Our model is only tested in one specific area, which it is also trained on. Different data sets for training and testing should be used for further testing, development, and improvement of U-PSP-Net. The network has proven to be accurate and efficient for building detection. It would also be interesting to see the network applied in other contexts and for other uses would grant similar results.

Expanding and Improving the Data Set

This thesis covered the creation of a novel training data set for building detection. As mentioned in the discussion, the data set was not general enough and only included images from one year and over a homogeneous area. However, the data set is highly accurate both in the annotations and resolution and would benefit from an expansion. We recommend that the data set should be further expanded with more images. Firstly the data set should include more images from different periods, still covering the same area. If the data set is proven to be effective for both building and change detection, the data set could be further expanded. This expansion should include images covering other areas in Norway and internationally.

Bibliography

[1] Keisuke Nemoto et al. 'Building change detection via a combination of CNNs using only RGB aerial imageries'. In: *Remote Sensing Technologies and Applications in Urban Environments II*. Vol. 10431. International Society for Optics and Photonics. 2017, 104310J.

[2] Ashbindu Singh. 'Review article digital change detection techniques using remotely-sensed data'. In: *International journal of remote sensing* 10.6 (1989), pp. 989–1003.

[3] Masroor Hussain et al. 'Change detection from remotely sensed images: From pixel-based to object-based approaches'. In: *ISPRS Journal of photogrammetry and remote sensing* 80 (2013), pp. 91–106.

[4] John R Jensen et al. *Introductory digital image processing: a remote sensing perspective.* Ed. 2. Prentice-Hall Inc., 1996.

[5] Irmgard Niemeyer and Morton J Canty. 'Pixel-based and object-oriented change detection analysis using high-resolution imagery'. In: *Proceedings 25th Symposium on Safeguards and Nuclear Material Management*. 2003, pp. 2133–2136.

[6] Dengsheng Lu et al. 'Change detection techniques'. In: *International journal of remote sensing* 25.12 (2004), pp. 2365–2401.

[7] Dengsheng Lu et al. 'Land-use and land-cover change detection'. In: *Advances in Environmental Remote Sensing Sensors, Algorithms, and Applications. CRC Press Taylor & Francis Group, New York* (2011), pp. 273–290.

[8] Shunping Ji et al. 'Building instance change detection from large-scale aerial images using convolutional neural networks and simulated samples'. In: *Remote Sensing* 11.11 (2019), p. 1343.

[9] H Gökhan Akçay and Selim Aksoy. 'Building detection using directional spatial constraints'. In: *2010 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2010, pp. 1932–1935.

[10] Gang Chen et al. 'Object-based change detection'. In: *International Journal of Remote Sensing* 33.14 (2012), pp. 4434–4457.

[11] Thomas Blaschke and Geoffrey J Hay. 'Object-oriented image analysis and scale-space: theory and methods for modeling and evaluating multiscale landscape structure'. In: *International Archives of Photogrammetry and Remote Sensing* 34.4 (2001), pp. 22–29.

[12] Thomas Blaschke, Stefan Lang and Geoffrey Hay. *Object-based image analysis: spatial concepts for knowledge-driven remote sensing applications*. Springer Science & Business Media, 2008.

[13] Helena Mäkelä and Anssi Pekkarinen. 'Estimation of timber volume at the sample plot level by means of image segmentation and Landsat TM imagery'. In: *Remote Sensing of Environment* 77.1 (2001), pp. 66–75.

[14] Qi Wang et al. 'GETNET: A general end-to-end 2-D CNN framework for hyperspectral image change detection'. In: *IEEE Transactions on Geoscience and Remote Sensing* 57.1 (2018), pp. 3–13.

[15] Casey Cleve et al. 'Classification of the wildland–urban interface: A comparison of pixel-and object-based classifications using high-resolution aerial photography'. In: *Computers, Environment and Urban Systems* 32.4 (2008), pp. 317–326.

[16] Padraig Corcoran and A Winstanley. 'Using texture to tackle the problem of scale in land-cover classification'. In: *Object-based image analysis*. Springer, 2008, pp. 113–132.

[17] Filip Hájek. 'Process-based approach to automated classification of forest structures using medium format digital aerial photos and ancillary GIS information'. In: *European Journal of Forest Research* 127.2 (2008), pp. 115–124.

[18] David Flanders, Mryka Hall-Beyer and Joan Pereverzoff. 'Preliminary evaluation of eCognition object-based software for cut block delineation and feature extraction'. In: *Canadian Journal of Remote Sensing* 29.4 (2003), pp. 441–452.

[19] L Wang, WP Sousa and P Gong. 'Integration of object-based and pixel-based classification for mapping mangroves with IKONOS imagery'. In: *International Journal of Remote Sensing* 25.24 (2004), pp. 5655–5668.

[20] Daniel J Hayes and Steven A Sader. 'Comparison of change-detection techniques for monitoring tropical forest clearing and vegetation regrowth in a time series'. In: *Photogrammetric engineering and remote sensing* 67.9 (2001), pp. 1067–1075.

[21] Google Trends. *Machine Learning - Utforsk - Google Søketrender*. https://trends.google.com/trends/explore?date=all&q=Machine%20Learning. (Accessed on 05/16/2022).

[22] Chen Sun et al. 'Revisiting unreasonable effectiveness of data in deep learning era'. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 843–852.

[23] Alon Halevy, Peter Norvig and Fernando Pereira. 'The unreasonable effectiveness of data'. In: *IEEE intelligent systems* 24.2 (2009), pp. 8–12.

[24] Jia Deng et al. 'Imagenet: A large-scale hierarchical image database'. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[25] Alex Wang et al. 'GLUE: A multi-task benchmark and analysis platform for natural language understanding'. In: *arXiv preprint arXiv:1804.07461* (2018).

[26] Abhinav Jain et al. 'Overview and importance of data quality for machine learning tasks'. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 3561–3562.

[27] Yanfeng Wei, Zhongming Zhao and Jianghong Song. 'Urban building extraction from high-resolution satellite panchromatic image using clustering and edge detection'. In: *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*. Vol. 3. Ieee. 2004, pp. 2008–2010.

[28] Fırat Erdem and Uğur Avdan. 'Comparison of different U-net models for building extraction from high-resolution aerial imagery'. In: *International Journal of Environment and Geoinformatics* 7.3 (2020), pp. 221–227.

[29] Mohsen Ghanea, Payman Moallem and Mehdi Momeni. 'Building extraction from high-resolution satellite images in urban areas: recent methods and strategies against significant challenges'. In: *International journal of remote sensing* 37.21 (2016), pp. 5234–5248.

[30] 2016. URL: http://jiangyeyuan.com/bldgExt.html.

[31] Xin Zhang et al. 'How well do deep learning-based methods for land cover classification and object detection perform on high resolution remote sensing imagery?' In: *Remote Sensing* 12.3 (2020), p. 417.

[32] Yakoub Bazi, Farid Melgani and Hamed D Al-Sharari. 'Unsupervised change detection in multispectral remotely sensed imagery with level set methods'. In: *IEEE Transactions on Geoscience and Remote Sensing* 48.8 (2010), pp. 3178–3187.

[33] John Alan Richards and JA Richards. *Remote sensing digital image analysis*. Vol. 3. Springer, 1999.

[34] Sotiris B Kotsiantis, I Zaharakis, P Pintelas et al. 'Supervised machine learning: A review of classification techniques'. In: *Emerging artificial intelligence applications in computer engineering* 160.1 (2007), pp. 3–24.

[35] Turgay Celik. 'Unsupervised change detection in satellite images using principal component analysis and k-means clustering'. In: *IEEE Geoscience and Remote Sensing Letters* 6.4 (2009), pp. 772–776.

[36] O Aytekin et al. 'Automatic and unsupervised building extraction in complex urban environments from multi spectral satellite imagery'. In: *2009 4th international conference on recent advances in space technologies*. IEEE. 2009, pp. 287–291.

[37] Howard D Bondell. 'Minimum distance estimation for the logistic regression model'. In: *Biometrika* 92.3 (2005), pp. 724–731.

[38] AG Wacker and DA Landgrebe. 'Minimum distance classification in remote sensing'. In: *LARS Technical Reports* (1972), p. 25.

[39] Shiming Xiang, Feiping Nie and Changshui Zhang. 'Learning a Mahalanobis distance metric for data clustering and classification'. In: *Pattern recognition* 41.12 (2008), pp. 3600–3612.

[40] Salar Ghaffarian and Saman Ghaffarian. 'Automatic building detection based on supervised classification using high resolution Google Earth images'. In: (2014).

[41] Caglar Senaras and Fatoş T Yarman Vural. 'A self-supervised decision fusion framework for building detection'. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 9.5 (2015), pp. 1780–1791.

[42] Dejan Grigillo and Mojca Kosmatin Fras. 'Classification based building detection from GeoEye-1 images'. In: *2011 Joint Urban Remote Sensing Event*. IEEE. 2011, pp. 381–384.

[43] Ivan Lizarazo and Paul Elsner. 'Fuzzy segmentation for object-based image classification'. In: *International Journal of Remote Sensing* 30.6 (2009), pp. 1643–1649.

[44] Ervin Yohannes and Fitri Utaminingrum. 'Building segmentation of satellite image based on area and perimeter using region growing'. In: *Indonesian Journal of Electrical Engineering and Computer Science* 3.3 (2016), pp. 579–585.

[45] Chenxi Liu et al. 'Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 82–92.

[46] Ziming Li et al. 'A Deep Learning-Based Framework for Automated Extraction of Building Footprint Polygons from Very High-Resolution Aerial Imagery'. In: *Remote Sensing* 13.18 (2021), p. 3630.

[47] Yongyang Xu et al. 'Building extraction in very high resolution remote sensing imagery using deep learning and guided filters'. In: *Remote Sensing* 10.1 (2018), p. 144.

[48] *2D Semantic Labeling Contest - Potsdam*. https://www.isprs.org/education/benchmarks/UrbanSemLab/2d-sem-label-potsdam.aspx. (Accessed on 06/02/2022).

[49] *2D Semantic Label. - Vaihingen*. https://www.isprs.org/education/benchmarks/UrbanSemLab/2d-sem-label-vaihingen.aspx. (Accessed on 06/02/2022).

[50] Kang Zhao et al. 'Building extraction from satellite images using mask R-CNN with building boundary regularization'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 247–251.

[51] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. 'Imagenet classification with deep convolutional neural networks'. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[52] Yang Shao, Gregory N Taff and Stephen J Walsh. 'Shadow detection and building-height estimation using IKONOS data'. In: *International journal of remote sensing* 32.22 (2011), pp. 6929–6944.

[53] Jan Stuckens, PR Coppin and ME Bauer. 'Integrating contextual information with per-pixel classification for improved land cover classification'. In: *Remote sensing of environment* 71.3 (2000), pp. 282–296.

[54] Saad Albawi, Tareq Abed Mohammed and Saad Al-Zawi. 'Understanding of a convolutional neural network'. In: *2017 International Conference on Engineering and Technology (ICET)*. Ieee. 2017, pp. 1–6.

[55] Kunihiko Fukushima. 'Neocognitron: A hierarchical neural network capable of visual pattern recognition'. In: *Neural networks* 1.2 (1988), pp. 119–130.

[56] Leonid Ivanovsky et al. 'Building detection on aerial images using U-NET neural networks'. In: *2019 24th Conference of Open Innovations Association (FRUCT)*. IEEE. 2019, pp. 116–122.

[57] Heechul Jung et al. 'ResNet-based vehicle classification and localization in traffic surveillance systems'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 61–67.

[58] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[59] Md Zahangir Alom et al. 'The history began from alexnet: A comprehensive survey on deep learning approaches'. In: *arXiv preprint arXiv:1803.01164* (2018).

[60] Jinjiang Wang et al. 'A multi-scale convolution neural network for featureless fault diagnosis'. In: *2016 International Symposium on Flexible Automation (ISFA)*. IEEE. 2016, pp. 65–70.

[61] Yu Guo, Jian-Yu Li and Zhi-Hui Zhan. 'Efficient hyperparameter optimization for convolution neural networks in deep learning: A distributed particle swarm optimization approach'. In: *Cybernetics and Systems* 52.1 (2020), pp. 36–57.

[62] Kaiming He et al. 'Deep residual learning for image recognition'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[63] Tobias Hinz et al. 'Speeding up the hyperparameter optimization of deep convolutional neural networks'. In: *International Journal of Computational Intelligence and Applications* 17.02 (2018), p. 1850008.

[64] Gonzalo I Diaz et al. 'An effective algorithm for hyperparameter optimization of neural networks'. In: *IBM Journal of Research and Development* 61.4/5 (2017), pp. 9–1.

[65] Leslie N Smith. 'A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay'. In: *arXiv preprint arXiv:1803.09820* (2018).

[66] https://www.facebook.com/MachineLearningMastery. *How to use Learning Curves to Diagnose Machine Learning Model Performance*. 2019. URL: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/.

[67] Connor Shorten and Taghi M Khoshgoftaar. 'A survey on image data augmentation for deep learning'. In: *Journal of big data* 6.1 (2019), pp. 1–48.

[68] Olaf Ronneberger, Philipp Fischer and Thomas Brox. 'U-net: Convolutional networks for biomedical image segmentation'. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[69] Nabil Ibtehaz and M Sohel Rahman. 'MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation'. In: *Neural Networks* 121 (2020), pp. 74–87.

[70] Nanjun He, Leyuan Fang and Antonio Plaza. 'Hybrid first and second order attention Unet for building segmentation in remote sensing images'. In: *Science China Information Sciences* 63.4 (2020), pp. 1–12.

[71] Bhakti Baheti et al. 'Eff-unet: A novel architecture for semantic segmentation in unstructured environment'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 358–359.

[72] Mohit Agarwal, Suneet Kr Gupta and KK Biswas. 'Plant Leaf Disease Segmentation Using Compressed UNet Architecture'. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2021, pp. 9–14.

[73] 2015. URL: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/.

[74] Matthew D Zeiler et al. 'Deconvolutional networks'. In: *2010 IEEE Computer Society Conference on computer vision and pattern recognition*. IEEE. 2010, pp. 2528–2535.

[75] 2021. URL: https://paperswithcode.com/method/max-pooling.

[76] Michal Drozdzal et al. 'The importance of skip connections in biomedical image segmentation'. In: *Deep learning and data labeling for medical applications*. Springer, 2016, pp. 179–187.

[77] Zongwei Zhou et al. 'Unet++: Redesigning skip connections to exploit multiscale features in image segmentation'. In: *IEEE transactions on medical imaging* 39.6 (2019), pp. 1856–1867.

[78] Andrew Joseph Davies. *Semantic Segmentation of Aerial Imagery Using U-Net in Python*. 2022. URL: https://towardsdatascience.com/semantic-segmentation-of-aerial-imagery-using-u-net-in-python-552705238514.

[79] Wei Liu, Andrew Rabinovich and Alexander C Berg. 'Parsenet: Looking wider to see better'. In: *arXiv preprint arXiv:1506.04579* (2015).

[80] Hengshuang Zhao et al. 'Pyramid scene parsing network'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2881–2890.

[81] Ge Shi, Shuying Guan and Xian Yang. *Semantic Image Segmentation with PSPNet and Dense CRF*. URL: https://geshijoker.github.io/files/research-3.pdf.

[82] Liang-Chieh Chen et al. 'DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2018), pp. 834–848. DOI: 10.1109/tpami.2017.2699184. URL: https://arxiv.org/pdf/1606.00915.pdf.

[83] Jonathan Long, Evan Shelhamer and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2015. URL: https://arxiv.org/pdf/1411.4038.pdf.

[84] 2016. URL: https://hszhao.github.io/projects/pspnet/.

[85] Richard Meyes et al. 'Ablation studies in artificial neural networks'. In: *arXiv preprint arXiv:1901.08644* (2019).

[86] Dabbala Rajagopal Reddy. *Speech recognition: invited papers presented at the 1974 IEEE symposium*. Elsevier, 1975.

[87] Kaichao You et al. 'How does learning rate decay help modern neural networks?' In: *arXiv preprint arXiv:1908.01878* (2019).

[88] Samuel L Smith et al. 'Don't decay the learning rate, increase the batch size'. In: *arXiv preprint arXiv:1711.00489* (2017).

[89] James B Campbell and Randolph H Wynne. *Introduction to remote sensing*. Guilford Press, 2011.

[90] *UltraCam Eagle Mark 3 Large Format Camera • Vexcel Imaging*. https://www.vexcel-imaging.com/ultracam-eagle/. (Accessed on 06/02/2022).

[91] Bl Basavaprasad and M Ravi. 'A study on the importance of image processing and its applications'. In: *IJRET: International Journal of Research in Engineering and Technology* 3 (2014), p. 1.

[92] 2022. URL: https://labelbox.com.

[93] Martin Kada. *Scale-Dependent Simplification of 3D Building Models Based on Cell Decomposition and Primitive Instancing*. URL: http://www.geosensor.net/cositprivate/32.pdf.

[94] Dominik Müller and Frank Kramer. 'MIScnn: a framework for medical image segmentation with convolutional neural networks and deep learning'. In: *BMC Medical Imaging* 21.1 (2021). DOI: 10.1186/s12880-020-00543-7. URL: https://bmcmedimaging.biomedcentral.com/articles/10.1186/s12880-020-00543-7.

[95] Le Kang et al. 'A deep learning approach to document image quality assessment'. In: *2014 IEEE International Conference on Image Processing (ICIP)* (2014). DOI: 10.1109/icip.2014.7025520. URL: https://ieeexplore.ieee.org/abstract/document/7025520.

Appendix

A Kartblad Trondheim

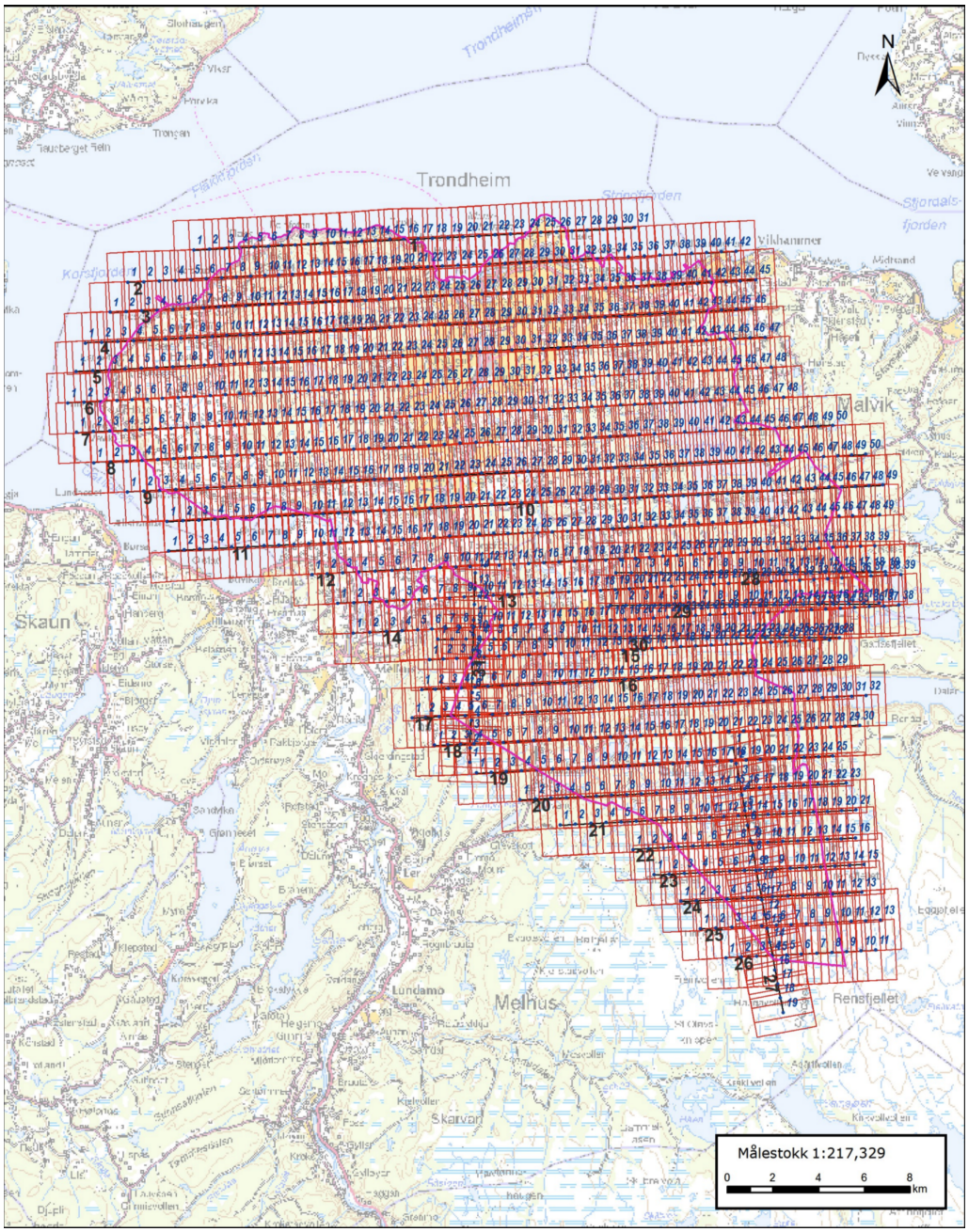


Figure 1: Trondheim

B Results

U-PSP-Net

Epoch	Binary IoU	Loss	Learning Rate	Precision	Recall
0	0.6299	0.3589	0.0001	0.7333	0.6063
1	0.7521	0.2530	0.0001	0.8273	0.7499
2	0.7818	0.2175	0.0001	0.8421	0.7984
3	0.8060	0.1915	0.0001	0.8588	0.8301
4	0.8272	0.1676	0.0001	0.8734	0.8545
5	0.8344	0.1586	0.0001	0.8797	0.8618
6	0.8492	0.1436	0.0001	0.8896	0.8778
7	0.8617	0.1297	0.0001	0.8982	0.8915
8	0.8687	0.1225	0.0001	0.9036	0.8977
9	0.8822	0.1095	0.0001	0.9147	0.9091
10	0.8845	0.1068	0.0001	0.9164	0.9117
11	0.8960	0.0956	0.0001	0.9249	0.9214
12	0.9059	0.0857	0.0001	0.9312	0.9309
13	0.8940	0.0972	0.0001	0.9246	0.9191
14	0.9173	0.0749	0.0001	0.9391	0.9407
15	0.9222	0.0704	0.0001	0.9436	0.9439
16	0.9281	0.0651	0.0001	0.9475	0.9492
17	0.9288	0.0642	0.0001	0.9485	0.9487
18	0.9369	0.0565	0.0001	0.9542	0.9559
19	0.9541	0.0413	0.00001	0.9666	0.9685
20	0.9581	0.0377	0.00001	0.9691	0.9719
21	0.9600	0.0360	0.00001	0.9705	0.9733
22	0.9614	0.0346	0.00001	0.9715	0.9743
23	0.9629	0.0332	0.00001	0.9728	0.9752
24	0.9641	0.0321	0.00001	0.9735	0.9762
25	0.9655	0.0308	0.00001	0.9747	0.9771
26	0.9670	0.0293	0.000001	0.9758	0.9786
27	0.9675	0.0290	0.000001	0.9762	0.9786
28	0.9677	0.0288	0.000001	0.9763	0.9788
29	0.9679	0.0287	0.000001	0.9766	0.9787
30	0.9680	0.0285	0.000001	0.9766	0.9789
31	0.9682	0.0283	0.0000001	0.9766	0.9792
32	0.9682	0.0283	0.0000001	0.9767	0.9792

U-Net

Epoch	Binary IoU	Loss	Learning Rate	Precision	Recall
0	0.6549	0.3419	0.0001	0.7711	0.6140
1	0.7590	0.2429	0.0001	0.8384	0.7537
2	0.7919	0.2056	0.0001	0.8508	0.8093
3	0.8203	0.1757	0.0001	0.8691	0.8471
4	0.8332	0.1614	0.0001	0.8777	0.8608
5	0.8487	0.1437	0.0001	0.8901	0.8769
6	0.8637	0.1294	0.0001	0.9015	0.8911
7	0.8696	0.1213	0.0001	0.9071	0.8956
8	0.8818	0.1095	0.0001	0.9150	0.9086
9	0.8887	0.1029	0.0001	0.9206	0.9136
10	0.9010	0.0907	0.0001	0.9289	0.9262
11	0.9031	0.0886	0.0001	0.9305	0.9270
12	0.9148	0.0774	0.0001	0.9393	0.9369
13	0.9165	0.0756	0.0001	0.9410	0.9382
14	0.9222	0.0701	0.0001	0.9446	0.9436
15	0.9189	0.0731	0.0001	0.9434	0.9395
16	0.9384	0.0556	0.0001	0.9556	0.9565
17	0.9537	0.0419	0.0001	0.9663	0.9684
18	0.9574	0.0386	0.00001	0.9689	0.9713
19	0.9594	0.0369	0.00001	0.9703	0.9728
20	0.9610	0.0354	0.00001	0.9715	0.9739
21	0.9625	0.0340	0.00001	0.9725	0.9750
22	0.9639	0.0326	0.00001	0.9736	0.9760
23	0.9658	0.0308	0.000001	0.9749	0.9776
24	0.9662	0.0306	0.000001	0.9752	0.9777
25	0.9665	0.0304	0.000001	0.9754	0.9779
26	0.9666	0.0302	0.000001	0.9755	0.9780
27	0.9668	0.0300	0.000001	0.9756	0.9781

C Building Detection Results

Figure 2: Buildings detected from different areas in Trondheim

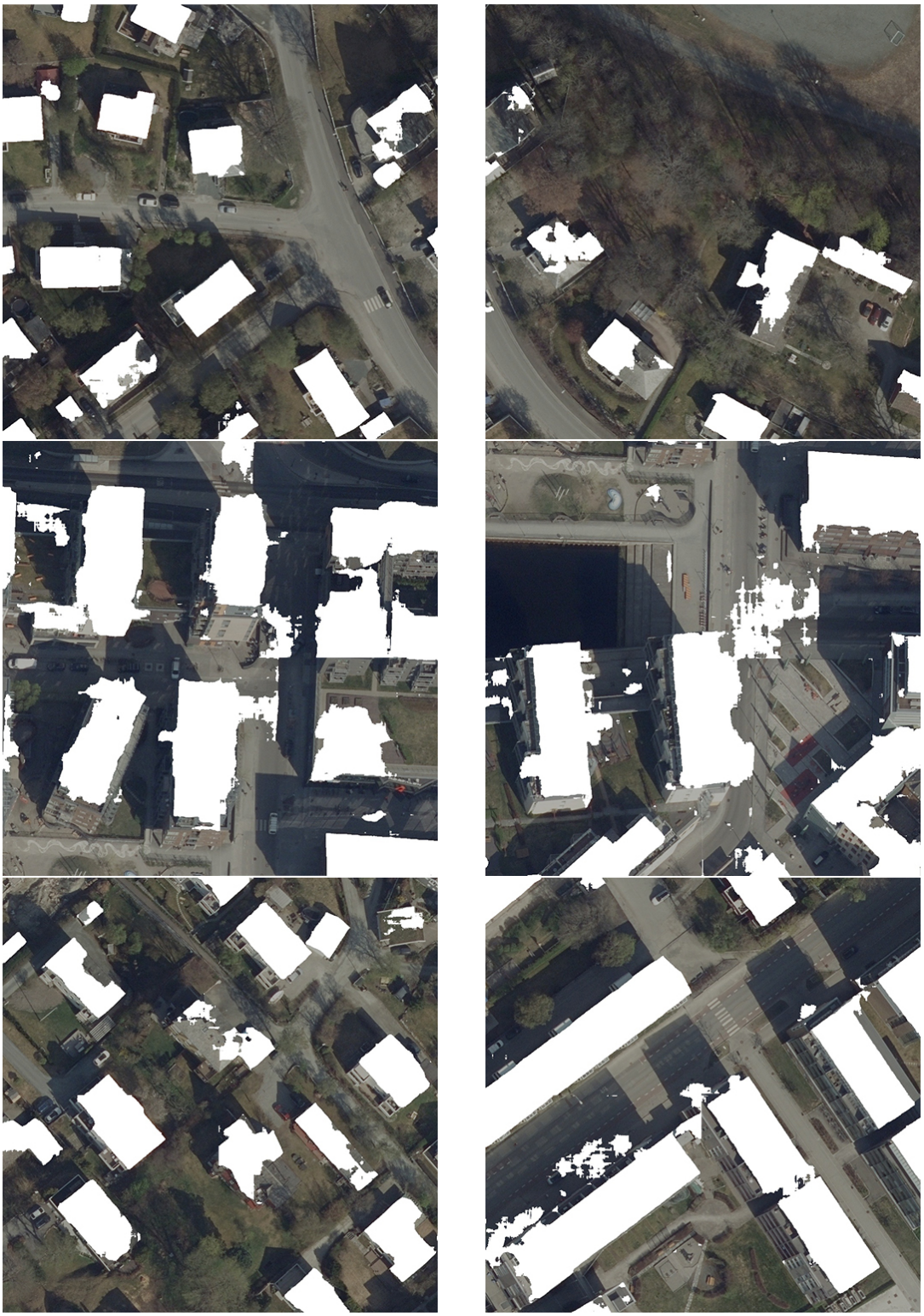


Figure 3: Buildings detected from different areas in Trondheim

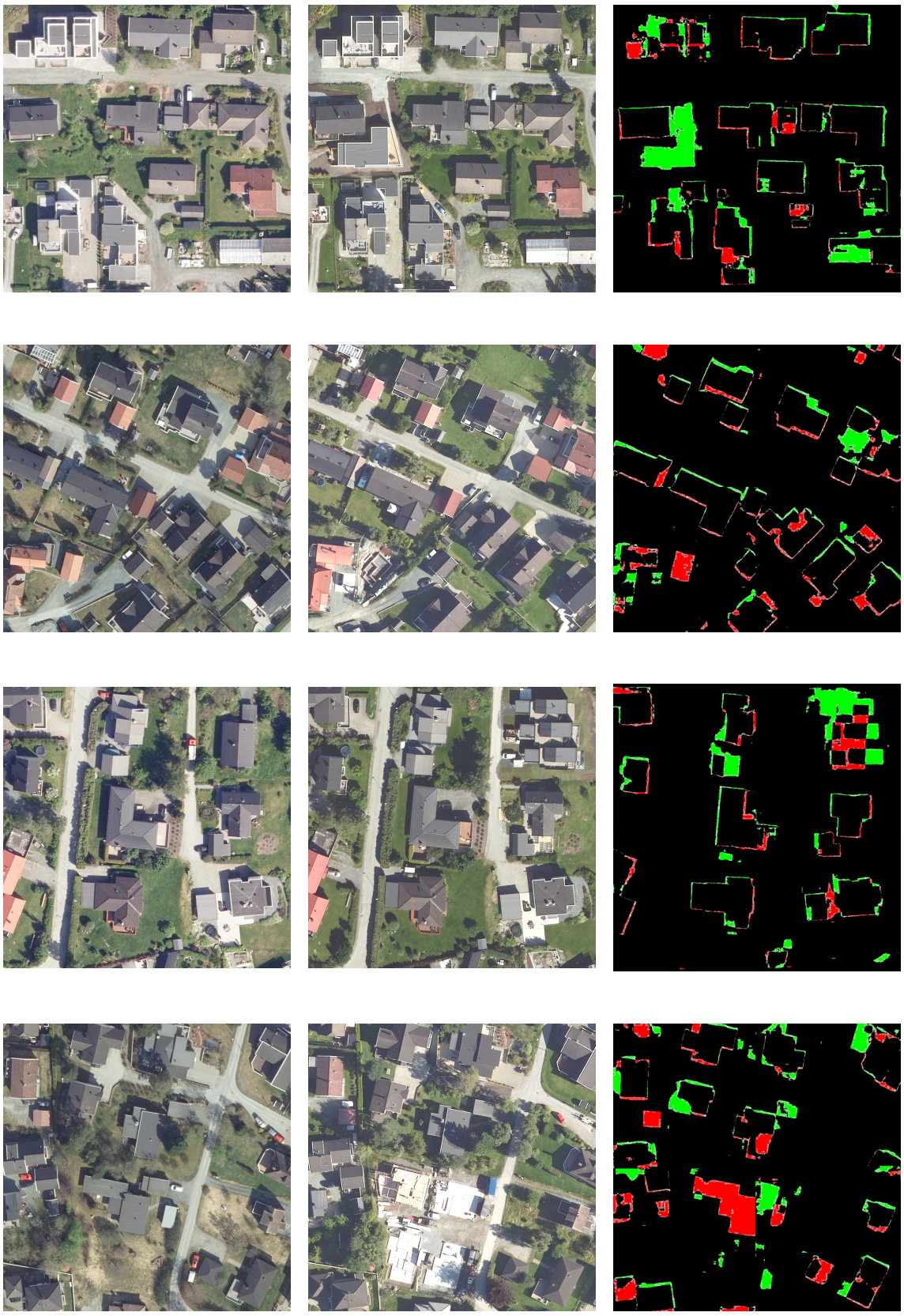
D Building Detection, Irregular shapes

E Difference in Color Properties

Aerial Image from 2019

Aerial Image from 2020

F Change Detection Results



G Area


© 2022 Google
© 2022 Google