Building Detection from Very High Resolution Remotely Sensed Imagery Using Deep Neural Networks

by

Mengge Chen

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Science
in
Geography

Waterloo, Ontario, Canada, 2019

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The past decades have witnessed a significant change in human societies with a fast pace and rapid urbanization. The boom of urbanization is contributed by the influx of people to the urban area and comes with building construction and deconstruction. The estimation of both residential and industrial buildings is important to reveal and demonstrate the human activities of the regions. As a result, it is essential to effectively and accurately detect the buildings in urban areas for urban planning and population monitoring. The automatic building detection method in remote sensing has always been a challenging task, because small targets cannot be identified in images with low resolution, as well as the complexity in the various scales, structure, and colours of urban buildings. However, the development of techniques improves the performance of the building detection task, by taking advantage of the accessibility of very high-resolution (VHR) remotely sensed images and the innovation of object detection methods.

The purpose of this study is to develop a framework for the automatic detection of urban buildings from the VHR remotely sensed imagery at a large scale by using the state-of-art deep learning network. The thesis addresses the research gaps and difficulties as well as the achievements in building detection. The conventional hand-crafted methods, machine learning methods, and deep learning methods are reviewed and discussed. The proposed method employs a deep convolutional neural network (CNN) for building detection. Two input datasets with different spatial resolutions were used to train and validate the CNN model, and a testing dataset was used to evaluate the performance of the proposed building detection method. The experiment result indicates that the proposed method performs well at both building detection and outline segmentation task with a total precision of 0.92, a recall of 0.866, an F1-score of 0.891. In conclusion, this study proves the feasibility of CNN on solving building detection challenges using VHR remotely sensed imagery.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Dr. Jonathan Li for the continuous support of my Master study and research, for his aspiring guidance, priceless constructive criticism, and generous sharing of his immense knowledge. I appreciate his help through all the time of research and writing of the thesis. Without his assistance and dedicated involvement, I would never accomplish this thesis.

I would also like to sincerely thank my thesis committee numbers, Dr. Michael Chapman, Professor at the Department of Civil Engineering, Ryerson University, Dr. Dongpu Cao, Associate Professor at the Department of Mechanical and Mechatronics Engineering, University of Waterloo, and Dr. Linlin Xu, Research Assistant Professor at the Department of System Design Engineering, University of Waterloo, for serving as my thesis examine committee, and for their insightful comments and encouragement.

Furthermore, I would like to thank Zilong Zhong, Lingfei Ma, Ying Li, Ming Liu, Gaoxiang Zhou, Zheng Gong, Dedong Zhang, Zhuo Chen, Weiya Ye and Yue Gu, all the members in the Mobile Sensing and Geodata Analytics Lab, for their academic perspectives and stimulating discussions during the group meeting. Also, sincere thanks go to the staff and colleagues at WatXtract.ai where I spent two terms during internship and had access to the advanced computing facility. Appreciation also goes to all the staff in the Department of Geography and Environmental Management, especially Alan Anthony and Susie Castela, for their assistance. Special thanks go to Dr. Ke Yang, for providing me support on programming and deep learning.

Last but not least, I would like to express my deepest gratitude to my parents for their endless love, understanding and moral support throughout my graduate study and my life in general. Thanks to my friends: Ansel Zhao, Changyuan Xiang, Jerry Liang, Liang Zhu, Sen Li, Wanxue Meng, Yifei Song, and Yunzhe Li, for their unconditional understanding and encouragement, for accepting nothing less than excellence from me, and for all the sleepless nights we have been through together.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AP	Average precision
AR	Averaged recall
CIS	Channel-wise inhibited softmax
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DBN	Deep Belief Network
DEM	Deep Energy Model
DSM	Digital surface model
DTM	Digital Terrain Model
FCN	Fully Convolutional Network
FN	False negative
FP	False positive
FPN	Feature Pyramid Network
GSD	Ground sample distance
GLGCM	Gray level-gradient co-occurrence matrix
HOG	Histogram of oriented gradients
IoU	Intersection-over-Union
LINZ	Land Information of New Zealand
mAP	mean Average Precision
mAR	mean Average Recall
MRF	Markov Random Field
MRS	Multi-resolution segmentation

MSI	Morphological shadow index
NAIP	National Agriculture Imagery Program
NZTM	New Zealand Transverse Mercator
OBIA	Object-based image analysis
RBF	Radial Basis Function
RBM	Restricted Boltzmann Machines
ResNet	Residual Network
ROI	Region of Interest
RPN	Region proposal network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TN	True negative
TP	True positive
VHR	Very high-resolution

Chapter 1 Introduction

1.1 Motivation

During the past few decades, the significant development of human societies leads to expanded urbanization. Due to the modernization and industrialization, the global urbanization signifies the rapid movement of the population from rural to urban areas (Konstantinidis, 2017). According to the Population Division report from the United Nations (2015), there is currently more than half of the world's population living in urban areas. And by the year 2050. the number will increase to 66%, which means almost two-thirds of the world's population is projected to be urban residents (UN, 2015). Large changes in settlement always follow the massive urbanization; therefore, the estimation of buildings, especially houses for inhabitants, can be regarded as a major statistic for urban understanding. The construction and demolition of buildings can reflect the distribution and impact of human activities in a region; thus, the urban building detection and mapping are essential for urban management. Accordingly, the aim of building maps is to understand the urban dynamics, including population estimation, urban planning promotion, and other applications in the field of socio-economic and environmental studies (Jensen & Cowen, 1999).

Object detection, one of the key issues in computer vision, can be represented as the procedure of detecting semantic objects from a certain class (i.e., building) through digital images or videos (Voulodimos, Doulamis, Doulamis, & Protopapadalkis, 2018). Building detection in the field of remote sensing is to identify and extract the building regions from aerial or satellite images by employing image processing and computer vision technologies (Cheng & Han, 2016; Konstantinidis, 2017). Currently, the extraction of building footprints using remote sensing imagery attracts people's attention, which gives rise to competitions such as the CrowdAI Mapping Challenge (CrowdAI, n.d.), DeepGlobe (Demir et al., 2018) and SpaceNet challenges (van Etten, Lindenbaum, and Bacastow, 2018). It is worth mentioning that the commercial very high-resolution (VHR) multispectral imagery, which can be easily obtained nowadays, provides huge opportunities for the identification of buildings. In recent years, the tremendous development of remote sensing technologies provides a great possibility for automatic object detection using the large numbers of VHR remote sensing images. The definition of VHR imagery is an image with a

ground sample distance (GSD) on the order of 10 cm (Marmanis et al., 2016). Using these high-resolution images, the detailed information in the urban environment can be comprehensively captured, making the tasks of the identification and classification on residential buildings more accurate and effective than low-resolution imagery (Konstantinidis, 2017; Turker & Koc-San, 2015). Additionally, although LiDAR data can offer 3D points in high accuracy, it misses the breakline information which can represent a distinct surface feature. VHR imagery, on the other hand, provides highly accurate breakline information (Shu, 2014).

However, challenges remain in the automated man-made object extraction and building detection using VHR remote sensing images for a lot of urban planning applications (Vakalopoulou, Karantzalos, Komodakis, & Paragios, 2015). The main challenges can be summarized as follows:

First challenge is the across and in-class diversity. The diversity of objects in the urban environment makes the spectral information unpredictable and challenges the conventional pixel- and spectral-based building detection methods, because the under-utilization of intensity information alone is insufficient for differentiate among impervious urban surface with similar spectral information (e.g., buildings, roads, and bare soil as shown in Figure 1.1(a)) (Huang, Zhu, Zhang, & Tang, 2014). Especially for VHR imagery, the increased resolution also results in more complex structural and contextual information, which makes it more difficult to find an accurate and robust building detection method. In addition, the complicated spatial and spectral varieties (e.g., size, colour, shapes and texture as shown in Figure 1.1(b)) within the characteristics of urban buildings bring to the impossibility of a universal template to define the major feature of buildings. And there is a need for context-based methods to utilize the spatial information of VHR images to accurately detect building objects (Huang et al., 2014).

The second challenge relates to the occlusions and shadows. On the one hand, buildings might be occluded by other urban objects such as high-rise buildings or tall trees (Figure 1.1(c)). In this case, it is challenging to assume the exact size or shape of the buildings for segmentation (Konstantinidis, 2017). On the other hand, the shadows brought by the high objects will influence the visibility of buildings, which requires additional shadow removal tasks. Also, the shadow cast by the building itself (Figure 1.1(d)) can be mistakenly identified as building candidates.

Furthermore, the detection and elimination of shadows are pretty difficult due to the complex generation of shadows as a result of radiance and reflectance, demanding more information about the features of occurred shadows.

(a) Across-class diversity (b) In-class diversity

(c) Occlusion (d) Shadow

Figure 1.1 Typical examples of challenges in building detection

The third challenge is the processing of large-scale dataset. The VHR imagery provides detailed building information for segmentation at the cost of a huge volume of data to be processed. Taking the dataset used in this study as an example, the aerial image covering of a small city has a size of 30 GB. As a consequence, the processing time for such large dataset is tremendous, and it is important to implement an effective method for building detection from VHR imagery.

Deep learning, the latest developed method in solving artificial intelligence problems, has a strong capability in object detection and semantic segmentation. Due to the predicting accuracy and processing efficiency, deep learning proves its competitiveness in the field of remote sensing (Zhu et al., 2017). Furthermore, deep learning shows its ability to learn hierarchies of features of various buildings from VHR imagery (Zhang, Zhang, & Du, 2016). Deep learning is also able to obtain higher accuracy and efficiency on the building detection task compared to the convention building detection methods (Saito & Aoki, 2015). Considering the advantages of both VHR imagery and deep learning on the building detection task, this thesis proposes an approach to detect and extract buildings from VHR remotely sensed imagery using deep learning models.

1.2 Objectives of the Thesis

The objective of this thesis is to develop an efficient and accurate framework for the automated detection and segmentation of urban buildings using deep learning networks on VHR remote sensing images. The specific objectives of the thesis are summarized as follows:

(1) To implement methods for the preprocessing of VHR images and digitized building footprints to make them consistent with the data structure used in deep learning networks;
(2) To propose and develop the deep learning networks for building detection and extraction from VHR images;
(3) To conduct an accuracy assessment to evaluate the performance of the proposed building detection method and compare the performance of the proposed method with other building detection methods.

1.3 Structure of the Thesis

The thesis consists of six chapters.

Chapter 1 describes the background, motivations, challenges of the study, followed by the research objectives as well as the structure of the study.

Chapter 2 provides a thorough review of the related research work in the field of automated detection of buildings in VHR images. It first reviews the traditional building detection methods, including template matching-based methods, knowledge-based methods, object-based methods,

and machine learning-based methods. The state-of-the-art deep learning models are then presented and discussed, including the structure of deep learning networks, deep learning algorithms and their applications in building detection.

Chapter 3 describes the study area and the datasets used in this study.

Chapter 4 presents the detail of methodology including pre-processing of data, the structure of proposed network, the implementation of proposed model, accuracy assessment and the implementation environment.

Chapter 5 shows the experimental results, including the selection of hyper-parameters, the qualitative and quantitative results of the proposed model on building detection. Additionally, the accuracy assessment and comparison with other models are also presented.

Chapter 6 concludes the key findings and contribution of the thesis, discusses limitations and recommendations for future research.

Chapter 2 Background and Related Studies

In this chapter, a literature review is presented to discuss the related research work performed in the topics of automated building detection in the field of remote sensing. Section 2.1 will discuss conventional building detection methods for remotely sensed images. The following Section 2.2 will introduce the current trends for object detection tasks, which mainly focus on the state-of-the-art deep learning methodology and its application in building detection for remotely sensed imagery.

2.1 Traditional Building Detection Methods

It is important to note that the appearances of buildings are in diverse shapes and sizes, and might be affected by the atmospheric condition, human actions, and light intensity, especially for aerial imagery. Therefore, the automatic detection and extraction of buildings is a significant yet challenging task, and the establishment of an effective method that can be utilized in a great range of remotely sensed imagery is extremely difficult.

Most of the researches involved in the building detection topic can be classified into two main categories in terms of the dimensionality and handling method of the data source: the methodology using 3D data and using 2D data (Konstantinidis, 2017). The first category, which processes 3D data for building extraction, mostly uses LiDAR point clouds and digital surface model (DSM) that are collected from laser scanners and maintains the height information of the terrain surface. With the representation of 3D data, two common solutions to building detection are the usage of appropriate height threshold (Baltsavias, Mason, & Stallmann, 1995; Brunn & Weidner, 1997) and the 2D feature extraction with a mapping onto 3D space (Hu, You, Neumann, & Park, 2004; Wang & Hsu, 2007; Huang, Zhuo, Tao, Shi, & Liu, 2017). Besides, the 3D building template that utilizes the building constructions from prepared urban knowledge to detect buildings is also used in researches (Forlani, Nardinocchi, Scaioni, & Zingaretti, 2006; Verma, Kumar, & Hsu, 2006; Karantzalos & Paragios, 2010; Hammoudi & Dornaika, 2011). What cannot be denied is the significant role of building height that is different from other objects shown on ground, with the exception of some tall trees and other man-made objects, that can be mistakenly identified as buildings under the height information (Huang et al., 2016). At the same time, there are still some

limitations to 3D data methodology: the high dependency on data resolution influenced by the sensor and the collecting procedure, the discrete height observation in the near neighbour pixels due to unexpected changes of surface height, and the heavy expenses in the acquisition of 3D point clouds (Konstantinidis, 2017). For those reasons, most building detection tasks fail to benefit from 3D data, which brings us to the second category as well as the focus for the rest of the literature review, the building detection utilizing 2D data.

Generally speaking, the building detection methods using 2D data can be divided into four main categories: the template matching-based, knowledge-based, object-based and machine learning-based methods, respectively. These four categories are not absolutely independent and have cross-cutting areas (Cheng & Han, 2016).

2.1.1 Template Matching-based Methods

Template matching-based method is the earliest and simplest technique for object detection. A template is first generated based on hand-crafting or training set, then used to match one image and to find the most possible object location (Cheng & Han, 2016). The most popular method in building detection is the active contour model also known as the snake model. An energy minimizing spline solution named 'snake' was first introduced by Kass, Witkin, and Terzopoulous (1988), in order to extract the edges and lines of an object of interest based on the guided constraints and image forces. Inspired by the research about 'snake', one research proposed the geodesic active contours to detect object boundaries based on the relationship between active contours and curve evolution (Caselles, Kimmel, & Sapiro, 1997). Therefore, the research about energy minimization and boundary segmentation methods can be employed in the building detection area.

The energy function can be produced relying on the texture, colour and shape information to characterize the attributes of buildings, and by minimizing the energy function, the specific type of building can be extracted (Konstantinidis, 2017). Theng (2006) proposed an active initialized contour model for automatic building extraction, where she used the enhanced snake energy function, combine with the circular casting algorithm to identify both structured and unstructured buildings from satellite imagery. Kovacs and Sziranyi (2012) extracted building contours with any

shape using the Harris corner detector for contour points, and the Chan-Vese active contour method was used to output the final boundary result of the building. Karantzalos and Argialas (2009) implemented a segmentation method in the use of a region-based energy function without manual localization of the contour area. The initialization of the proposed energy function shows that the boundaries can be automatically detected without the gradient and edge-map like other previous active contours (Karantzalos & Argialas, 2009). However, due to the diversity of colour and shape of the buildings in the image, it is difficult to find a universal energy function to identify the characteristic for each building and extract them (Konstantinidis, 2017).

2.1.2 Knowledge-based Methods

One type of commonly-used building detection methods is the knowledge-based building detection approach in remote sensing, which interprets the building detection tasks as hypotheses problem based on manifold knowledge and constraints. The most important part is the establishment of knowledge on targets of interest, and the most important knowledge is the object geometric knowledge that describes the prior knowledge using particular or general parametric shape models (Cheng & Han, 2016). A generic shape model was proposed by Huertas and Nevatia (1988) who defined buildings as rectangular or consisting of rectangular features (e.g., “box”, “T”, “L”, and “E” shapes) and used the model for building detection. Compared to the work of Huertas and Nevatia (1988), which ignored a more complex building structure, McGlone and Shufelt (1994) applied the geometric constraints and metric calculation on the building extraction. The line orientation to detect the corner and edge of buildings was used to hypothesize the building structure followed by considering shadow information to derive the final building information.

As a matter of fact, the shadow information is one important clue for another category of knowledge-based building detection, that is, the context knowledge. The context knowledge represents the spatial constraints or relations between the target of interest and the background, or the interaction between the object and its neighboring pixels. As a typical instance of context knowledge, shadow evidence was exploited by Peng and Liu (2005) to establish a shadow-context model to extract buildings from dense urban areas where the sunshine part and shadow part has a sharp contrast. In their research, the shadow information combined with context was first used to

estimate illuminating direction, then a partial snake function was applied on the building output to refine the boundaries, and finally, the intensity information was used to separate the roof and the self-shadow, based on the generic intensity profile (Peng & Liu, 2005). Ok, Senaras, and Yuksel (2013) also emphasized the importance of shadow information in building detection, where the spatial relationship between buildings and self-shadow shading direction was created using a novel fuzzy landscape generation method and GrabCut partitioning algorithm. Subsequently, Ok (2013) extended the building detection approach to a higher level of accuracy by using a new method based on solar angles to detect shadow areas and combining it with a graph theory framework to extract buildings from the shadow (Ok, 2013; Ghaffarian, 2014). One challenge is that the implicit prior knowledge about objects of interest requires an effective transformation to explicit constraints and models (Cheng & Han, 2016). The trade-off between the strict rules resulting in missing target objects and the loose rules producing the false positive demands more researches in order to provide a generic model for building detection.

2.1.3 Object-based Methods

With regard to the processing method, there are two types of methods: the pixel-based building detection and the object-based building detection. In general, the pixel-based method processes individual pixel, and assigns the label of one class on each pixel in the image or groups the pixels based on the spectral information alone (Shu, 2014). In contrast, object-based image analysis (OBIA) firstly groups relatively homogeneous pixels into the same region, also known as image segmentation; then develops a classifier to label these regions based on their characteristics. The unique process of OBIA can incorporate the spatial context of targeted objects during the classification, therefore offering a framework to exceed the limitations of traditional pixel-based object detection methods (Cheng & Han, 2016).

One of the most popular methods to sketch comparatively identical objects within the remotely sensed image is the multi-resolution segmentation (MRS), marked by the arrival of the first commercial object-oriented image analysis software eCognition (Baatz & Schape, 2000). The method mainly uses three parameters such as shape, compactness, and scale to segment one image into objects (Benz, Hofmann, Willhauck, Lingenfelter, & Heynen, 2004). The shape parameter

determines the similarity of objects based on a balance between the shape (e.g,. length or the number of edges) and colour (Benz et al., 2004). The compactness can be seen as a sub-parameter of shape, which is used to calculate the smoothness and compactness. The scale parameter has the ability to control the average size of the image object and is influenced by the spatial resolution of the matching image as well as the features in the study area (Kavzoglu & Yildiz, 2014). Based on the MRS and eCognition software, there are some attempts to study the automatic extraction of building from a remotely sensed image. For instance, Myint et al. (2011) used the three parameters to label the complex features in urban land-cover including buildings, vegetation, and other impervious surfaces. In the classification of building regions, the use of MRS, ratio PCA and NDVI showed the integration of pixel-based and object-based classifier. Additionally, studies also demonstrate the significance of the scale parameter that determines the scope and size of the extracted objects; therefore, directly impacts the following classification (Ma et al., 2017). However, the accuracy assessment in OBIA method is problematic in that a definition of one single and universal measure of the building segmentation accuracy is impossible; and due to the complex characteristics of buildings, it is challenging to select a suitable sample size and the use of a specific accuracy measurement to the applied algorithms (Cheng & Han, 2016).

2.1.4 Machine Learning-based Method

A recent cutting-edge development in object detection is the machine learning techniques, which regards the detection task as a typical classification problem and achieves desired performance. In machine learning, a classifier is learned from the training data in a supervised, semi-supervised or weakly supervised framework, and the classifier can conduct object detection using the variation among different class appearances (Cheng & Han, 2016).

The first step is the feature extraction. A sliding window or object proposal is applied on the raw image pixels, and the corresponding representation of features is extracted in a higher dimensional space (Khalid, Khalid, & Nasreen, 2014). As one of the most widely used features, Histogram of oriented gradients (HOG) was first introduced by Dalal and Triggs (2005) to calculate the distribution of intensity gradients, and the gradient orientation is regarded as the object feature for differentiation, which has a wide application in the edge or local contour

detection of objects. In one study, Ilsever and Unsalan (2013) applied HOG on the remotely sensed imagery to extract tall building regions. The HOG descriptor was first calculated using a sub-window to slide through the image, then a Support Vector Machine (SVM) was trained using the HOG descriptor and used to classify the building shape. An additional shadow shape detection was added to enhance the performance of the proposed building detection approach. Moreover, to strengthen the representation capability of the HOG descriptor, the concatenation of other features is proposed for high-performance building detection. As a continuation of previous work (Konstantinidis, Stathaki, Argyriou, & Grammalidis, 2015), Konstantinidis et al. (2017) combined HOG and LBP features as the descriptor, and the cosine-based distance between the two features was then put into the SVM classifier for training. After the derivation of candidate building regions, a region refinement method was proposed to obtain the final building extraction result. The feature fusion approach is robust to the variations among building shapes, and the parameters in the HOG-LBP features are precise for the specific task, thus can achieve acceptable performance for images with different spatial or spectral resolutions (Konstantinidis et al., 2017).

After the feature extraction, a classifier is trained from object regions with the corresponding feature representations, aiming at minimizing the error in the training set brought by misclassification (Cheng & Han, 2016). In practice, one commonly-used leaning approach will be explained in the following section, the SVM classifier. The current standard of SVM was first proposed by Cortes and Vapnik (1995) and becomes one of the most well-known and powerful machine learning algorithms in object classification. SVM is a supervised non-probabilistic learning model, thus the distribution of the dataset remains unknown before the training; additionally, SVM is trained iteratively until a relative optimal boundary is extracted to separate the training set (Mountrakis, Im, & Ogole, 2011). In the simplest version, SVM is a linear binary classifier that assigns one out of two class labels to a test sample which is an individual pixel in the case of remote sensing (Cheng & Han, 2016). The non-linear form of SVM, however, requires a kernel to project the data onto a higher dimensional feature space before the separating hyperplane is extracted, thus the accuracy of the detection result is influenced by the selection of the kernel function as well as proper parameters (Petropoulos, Kalaitzidis, & Vadrevu, 2012; Cheng & Han, 2016). In one research, San and Turker (2010) used SVM as the first step of building

extraction methodologies to extract the building patches, where they combined the satellite image with two additional bands (DSM and NDVI) into the SVM classifier. The Radial Basis Function (RBF) was selected as the kernel function that can manage linear and non-separable problems (San & Turker, 2010). However, SVM can only identify the building regions with irregular boundaries, thus the other segmentation methods are needed to improve the result, such as Hough Transform for building edge detection (Turker & Koc-San, 2015), template matching technique for height estimation (Turlapaty, Gokaraju, Du, Younan, & Anastoos, 2012), or morphological shadow index (MSI) for shadow detection (Huang & Zhang, 2012).

2.2 Deep Learning

As a subfield of machine learning, deep learning attempts to learn and discover high-level distributed representations in data by implementing hierarchical architectures (Guo et al., 2016). Since the defining paper brought by Krizhevsky et al. (2012) as a reconstruction of earlier work of Fukushima (1980) and LeCun et al. (1989), deep learning dramatically arises and becomes the most advanced technology in solving artificial intelligence problems (Marmanis et al., 2016). The popularity of deep learning nowadays results from the extremely enhanced chip processing abilities, the dramatically decreased cost of computing hardware, as well as the theoretical progress in current leading researches in machine learning and image processing (Deng, 2014). The foundation of modern deep learning is a neural network that is a computational simulation of the biological neural network. In biology, a neuron is a specialized cell that passes electrochemical signals to other neurons or parts of the nervous system; and the receiving neuron will process signals and then pass signals to the downstream connected neurons (Reagen, Adolf, Whatmough, Wei, & Brooks, 2017). As artificial neural networks, however, most of the mathematical and computational models are in either of two research directions. The first attempts to simulate the biological neurons to understand or illustrate behaviors, and the second utilizes models inspired by neurons to solve arbitrary problems (Reagen et al., 2017). In terms of the research interest, the second direction that focuses on bio-inspired models to solve real-world problems will be the main topic and discussed comprehensively in this thesis.

2.2.1 Structure of Deep Learning Models

Figure 2.1 shows the architecture of a simple neural network, where each neuron is expressed as a node, and each weight is a line that connects neurons. Specifically, the left neurons are identified as inputs to the right neurons, and the value of each neuron will be passed through to the right as well. The passed value, however, does not flow to the right directly, but is multiplied by the pre-defined weights and combined with other weighted neurons before it is passed forward to the next neuron. Briefly speaking, the information of each neuron will be transmitted to multiple neurons after the weighting computation, and each neuron will receive weighted information from multiple neurons. All the neurons that contain the input value are considered as an input layer; then the input layer distributes the input signals to the neurons in the hidden layers. Each neuron in the hidden layer operates the activation functions based on the given weights, and these activations will be transferred to the output layer which calculates the probability of each output and finally yield the result. It is worth noting that Figure 2.1 only represents the simplest example, as there might be more layers in the hidden layer. The number of layers is called depth, and the number of neurons in one layer is the width of that layer.

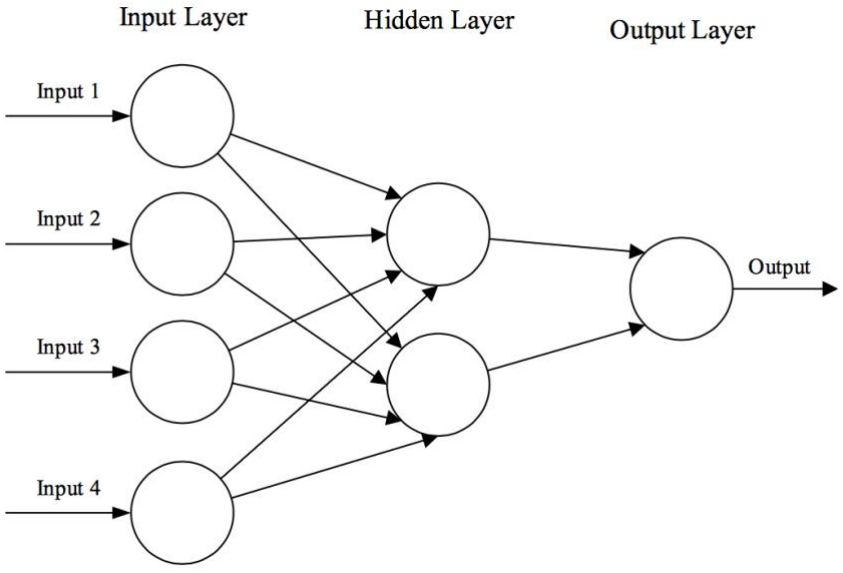


Figure 2.1 Structure of a simple neuron network

(Source: O'Shea & Nash, 2015)

2.2.2 Deep Learning Algorithms

Generally, methods using deep learning can be divided into four different categories: Restricted Boltzmann Machines (RBMs), Autoencoder, Sparse Coding and Convolutional Neural Networks (CNNs). Figure 2.2 shows the category along with the representative works. The following section will briefly introduce the concept of each deep learning method and its representative algorithms.

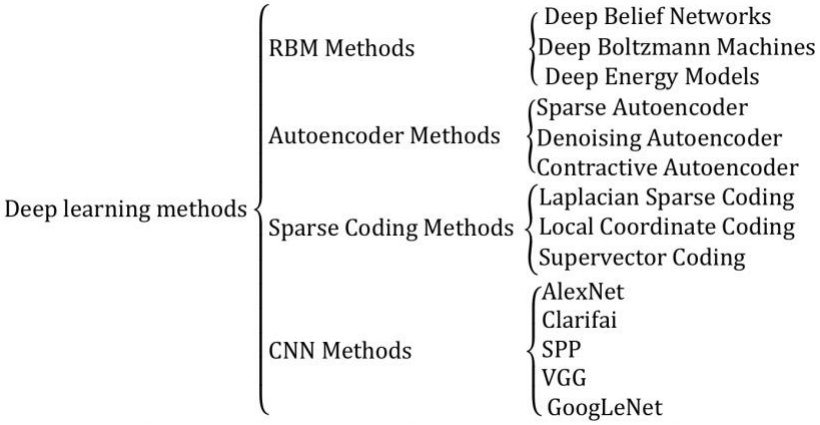


Figure 2.2 Categorization of deep learning methods and representative algorithms

(Source: Guo et al., 2016)

The RBM is a generative binary stochastic neural network that was proposed by Hinton and Sejnowski in 1986. As an alternative to the Boltzmann Machine, the RBM model is a bipartite graph using independent visible layers and hidden layers (Guo et al., 2016). In order to train the RBM model, a widely employed development is the rectified linear units, which allows each layer to preserve more information by using the same learning weight and learning bias (Nair & Hinton, 2010). By applying RBM as the learning model, there are three well-known deep models: Deep Belief Networks (DBNs), Deep Boltzmann Machines (DBMs), and Deep Energy Models (DEMs). DBN is a probabilistic generative model, in which the top two layers are the undirected graphical model while the lower layers are directed, generative models. Different from DBN, DBM is another deep learning approach where connections are undirected across all the layers in the network. DEM is a more recent model whose lower layers use deterministic hidden units, and stochastic hidden units for one single top hidden layer, compared to DBN and DBM with multiple stochastic layers (Ngiam, Chen, Koh, & Ng, 2011).

Autoencoder is a different type of neural network (unsupervised) that is specialized for efficient encoding learning, instead of predicting features given input by training the network (Liou, Cheng, Liou, & Liou, 2014). The general pipeline of an autoencoder is presented in Figure 2.3. By minimizing the error during the reconstruction process, the autoencoder first reconstructs the inputs into learned features (code), then decodes the features to obtain the output vectors that have the same dimensionality of input (Zhou, Arpit, Nwogu, & Venu, 2016). However, a single layer encoder is inadequate due to the complexity of raw data, thus the deep autoencoder is proposed by Hinton & Salakhutdinov (2006) to fully explain the particular and characteristic features. The deep autoencoder is formed by multiple levels of descriptions, composed of one visible layer, various hidden layers and an output layer to build the deeply stacked architectures. For each layer, the input is the previous layer's output, and the output with higher level features is wired to the successive layer (Dong, Liao, Liu, & Kuang, 2018). After the forward encoding process, a back-propagation is applied to the model to fine-tune the parameters within all the layers. Although the deep autoencoder outperforms the single layer autoencoder, the model requires a pre-trained weight to start the training, which can sometimes result in inaccurate predictions if errors occur at the beginning of training (Guo et al., 2016).

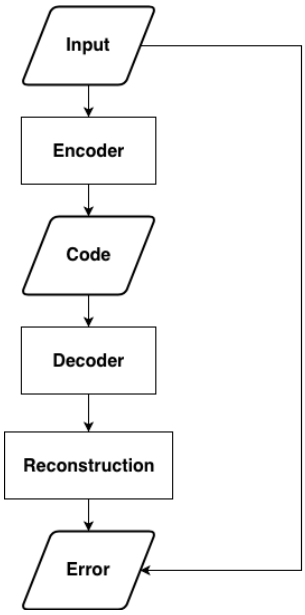


Figure 2.3 Architecture of an autoencoder

Sparse coding, another unsupervised learning algorithm, mainly focuses on studying a dictionary based on various natural images to explain the underlying structure in images; and after using the learned dictionary to represent an image, these representation coefficients are considered as features to interpret the characteristics of the image (Zhang, Yao, Sun, & Lu, 2013). As a widely-used dictionary learning model for deriving sparse representation, sparse coding only uses a small number of learning coefficients, thus making the encoding easier to interpret and minimizing the computational cost (Feng, Wu, & Zhou, 2017). The optimization of sparse coding has two main procedures during the encoding process: the weight update and activation inference. The projected gradient algorithm is one of the most commonly used algorithms in the weight update, which renormalizes the weight matrix after each epoch of the traditional Gradient descent algorithm. After the generation of weights, inferring the feature to activation is significant, thus the Iterative Shrinkage-thresholding Algorithm (ISTA) has been proposed for sparse coding activation inference. This algorithm optimizes the reconstruction object by using a gradient step and then uses a closing shrinkage operation for the sparsity term (Chambolle, DeVore, Lee, & Lucier, 1998).

CNN is the most commonly used and well-developed deep learning method in the field of image processing and analysis problems, aiming at handling image and video data (LeCun, Bengio, & Hinton, 2015). Basically, CNN is composed of three types of layers: the convolutional layer, the pooling layer, and the fully-connected layer. The architecture of a general CNN model is shown in Figure 2.4. The original image is first processed through multiple convolutional, non-linear and pooling layers to derive the feature maps, followed by fully-connected layers to obtain the 1D feature vector.

The convolutional layer in CNN is the core building part that convolves the large input image or the intermediate feature maps by a set of small, trainable kernels (weight matrix) to generate various feature maps (Guo et al., 2016). Therefore, each unit in the feature maps is connected to local patches of the feature maps from the previous layer, and the result of the weighted sum is passed through an activation function, usually non-linear such as sigmoids or rectified linear units (ReLU), to define the output of a layer (LeCun, Bengio, & Hinton, 2015). The convolution operation benefits the CNN noticeably in that: the same feature map sharing the same filter limits the number of parameters, and neighboring pixels with highly correlated values form unique motifs

can be easily detected; while the local statistics of processing image remain invariant to location (Zeiler, 2013). The pooling layer is always executed after a convolutional layer to decrease the dimensional property of feature maps; or summarily, it compacts the output of multiple neurons from the previous layer (Scherer, Müller, & Behnke, 2010). The average-pooling and the max-pooling are the most commonly used pooling methods that conduct the same operation, average or max, on a local spatial region to provide invariance to similar features (Reagen et al., 2017). Compared to average pooling, the max pooling is more preferable, since it considers the negative elements and avoids blurring of the gradients and activations (Zeiler, 2013). In the max pooling layer, the maximum value over a non-overlapping region is extracted as the output that down-samples the input image (Ciresan, Meier, Masci, Maria, & Schmidhuber, 2011).

After the final pooling layer, the fully-connected layer with weight matrix W and biases b follows to transfer the feature map in 2D to feature vector in 1D. The vector in the fully-connected layer can either be the various categories in an image classification task, or feature vector for the following process (Guo et al., 2016). However, there is one main disadvantage of the fully-connected layer. The model involves many parameters and demands massive computational resource in the training. The common solution is to decrease the connections or eliminate these layers. For instance, GoogLeNet replaced the fully-connected layers by sparsely connected architectures while maintaining the computational budget in the design of deep and wide neural networks (Szegedy et al., 2015).

The training phase of CNN mainly consists of two steps: the forward and backward steps. The forward step simply generates the feature maps for each layer based on the weights and biases, and after the generation of feature maps, the prediction for each class label is extracted to obtain the final output as well as the calculation of loss cost. The loss function essentially aims to qualify the scoring function which is used to classify the input data; but specifically, the function calculates the difference between the estimated label and the ground truth under some degrees of regularization. The backward step is then utilized to optimize the weights and biases by applying the gradient descent to each parameter. The computing gradient is used to adjust the weights and biases, attempting to arrive at a local/global minimum of the loss, and the gradient descent algorithm operates iteratively to update the model layer by layer. The repetition process stops until

the model reaches a predefined threshold of the model's loss cost or the limit of the number of iterations for both forward and backward steps.

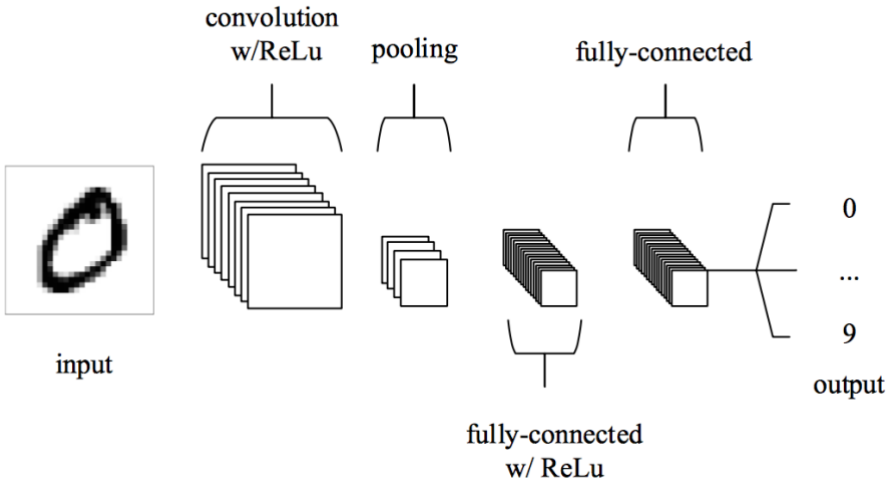


Figure 2.4 Architecture of a typical CNN model

(Source: O'Shea & Nash, 2015)

Some of the superiority and limitations of the above-mentioned deep learning model has been discussed in the previous section. Guo et al. (2016) summarized the comparison among four categories of the deep learning algorithms as shown in Table 2.1. Nine properties are listed in the summary. 'Generalization' interprets the overall performance of a model in different media and applications, such as visual recognition, text recognition, etc. The 'yes' mark shows that all four deep learning models can be effective in processing diverse inputs. 'Unsupervised learning' represents the ability to train a model without access to ground truth. CNNs is the only supervised method that requires labeled training data in the beginning, whereas the other three types of methods do not have such restrictions and can work in unsupervised mode. 'Biological understanding' and 'Theoretical justification' refers to the biological groundwork or theoretical base involved in the approaches. The sparse coding can be more effective in biological research due to its significant biological basis. 'Invariance', as one of the most important benefits in CNNs, shows the robustness of methods to transformation, including rotation, scale, and translation. Especially in computer vision tasks, the CNNs can abstract the input's features from the relative position or orientation of cameras and the object; therefore, the network can effectively identify

the object from the image where the pixel values are significantly different (Voulodimos et al., 2018). In general, the CNN outperforms other approaches in terms of feature learning, while considering non-visual input, RBMs performs better but fails in predicting joint probabilities and limiting the computational cost. On the other hand, the auto-encoder and sparse coding are more suitable for small training sets.

Table 2.1 Comparison among four categories of deep learning

(Source: Guo et al., 2016)

Properties	CNNs	RBMs	AutoEncoder	Sparse coding
Generalization	Yes	Yes	Yes	Yes
Unsupervised learning	No	Yes	Yes	Yes
Feature learning	Yes	Yes*	Yes*	No
Real-time training	No	No	Yes	Yes
Real-time prediction	Yes	Yes	Yes	Yes
Biological understanding	No	No	No	Yes
Theoretical justification	Yes*	Yes	Yes	Yes
Invariance	Yes*	No	No	Yes
Small training set	Yes*	Yes*	Yes	Yes

Note: 'Yes' indicates that the category does well in the property; otherwise, they will be marked by 'No'. The 'Yes*' refers to a preliminary or weak ability.

2.2.3 Building Detection with Deep Learning

Due to the outstanding capabilities in object detection, visual recognition, and semantic segmentation, deep learning has been introduced to solve problems in remote sensing. From the traditional research topics of image processing, pixel-level classification, and object recognition, to the latest challenging missions of high-level semantic extraction and remote sensing scene understanding, the state-of-the-art deep learning methods occur everywhere in the remote sensing analysis (Zhang, Zhang, & Du, 2016). Compared to the hand-crafted methods with a focus on texture and spatial context as mentioned in Section 2.1, the separation of feature extraction, pixel-based classification and context modeling becomes extremely insignificant with the application of deep learning (Marmanis et al., 2016). Due to the notable ability of feature understanding in high-

level scale, deep learning shows its efficiency in localizing specific ground objects, including the task of building detection (Zhu et al., 2017). Therefore, this section provides a methodical review and summarizes the representative models and their diverse characteristics in the application of the building detection domain.

During the early researches, patch-based CNN methods simply label a pixel using deep CNN to extract features from a local window. The general understanding is that patch-based methods are appropriate for texture feature derivation at the local scale, but not suitable for deriving geometric and texture features at a long range (Shu, 2014). However, the patch-based methods show effective solutions to object extraction with the sophisticated CNN implementation. The reason as mentioned in Mnih (2013) is that the former patch-based method lacks a strong feature extractor and a larger window that can include the investigated spatial context. Therefore, the development of CNN in the patch-based method can lead to a breakthrough in object extraction. In his experiment, Mnih (2013) used a patch-based labeling and learning framework with CNN for the automatic object extraction from aligned aerial images. In the post-processing step, the researcher extended the deep neural network to a deep Conditional Random Fields (CRFs) with a smoothness term among outputs. Mnih (2013) also introduced new loss functions in the deep learning training process, in order to achieve robustness to incomplete target maps. The results showed that the method of Mnih (2013) performs well on individual building extraction but works less well on continuous or larger buildings. Inspired by the patch-based method of Mnih (2013), Saito, Yamashita and Aoki (2016) employed a similar structure to train the CNN with multi-labeled patches, but he extended the algorithm to multispectral imagery including visible and infrared bands. Besides, a new output function named channel-wise inhibited softmax (CIS) was proposed in the training process, and the experiment showed an improvement on the object extraction result using the multi-channel mechanism with CIS (Saito et al., 2016). Vakalopoulou et al. (2015) proposed a supervised methodology in building extraction based on the AlexNet network. A pre-trained network using ImageNet dataset was utilized and trained on the VHR multispectral satellite imagery. After the classification process, the Markov Random Field (MRF) model was used to extract the building object, and the detection result was promising with quantitative validation. Nonetheless, the patch-based sliding method employed in the training and

testing procedure might be time-consuming (Huang et al., 2016). More specifically, the patch-based approach with overlapping brings about extreme computational load, and the averaging process might disregard valuable edge details (Fu, Liu, Zhou, Sun, & Zhang, 2017).

Using the standard CNN as a framework, the Fully Convolutional Network (FCN) was first proposed by Long, Shelhamer, and Darrell (2015). The fully-connected layers in the standard CNN were replaced with convolutional layers to train the classifier for pixel's likelihood score. Each pixel is then assigned to one object label according to the score; and this is the process of the CNN-based image segmentation (Fu et al., 2017). In remote sensing, there are several papers utilizing the FCN-based methods for building detection and segmentation. Sherrah (2016) labeled the aerial image in the use of FCN. By investigating the up-sampling and down-sampling architecture in CNN, a no-down-sampling approach was applied to the standard FCN to replace the down-sampling mechanism with the deconvolution to preserve the output resolution. Marmanis et al. (2016) used FCN and the succeeding aggressive deconvolution architecture as well to classify the aerial image. The extracted output proved the feasibility and effectiveness of FCN in aerial image segmentation, although the pixel-based classification was still limited by the receptive field of the classifier (Marmanis, et al., 2016). In the research presented in Maggiori et al. (2016), the patch-based CNN classification was compared with the FCN that was implemented using only the convolution and deconvolution operations, and the result showed that FCN performs better than the patch-based CNN in terms of classification accuracy and processing time. There are two advantages of FCN compared to the patch-based method. The first one has been mentioned in the previous paragraph about memory efficiency. Since the patch-based approach enters the overlapping patches into a minibatch, the same pixel will occur in the minibatch multiple times; while in FCN, each pixel is processed in minibatch only once (Sherrah, 2016). The second advantage is the training accuracy brought by FCN training since all the patches are processed in the training; whereas, in the patch-based method, only a random subset of all patches is selected for training. Therefore, more data are trained effectively in FCN and should result in higher accuracy (Sherrah, 2016).

Yang et al. (2018) compared four state-of-the-art CNN models at a very large scale across the entire United States continent using aerial images from the National Agriculture Imagery Program

(NAIP). The four deep learning architectures are namely: the branch-out CNN, FCN, conditional random filed as a recurrent neural network (CRFasRNN), and SegNet (Yang et al., 2018). The building extraction result showed that the SegNet model significantly outperforms other models on both the F1-score and Intersection-over-Union (IoU) at the instance level. Additionally, Yang et al. (2018) also considered the CRFs in the building extraction task for building boundaries refinement, and the experiments implied that the initial extractions from FCNs influenced the accuracy of the CRF module.

However, the FCN model and other identical convolutional encoder-decoder models, for example, the SegNet and DeconvNet, only apply a part of layers in the generation of final output but ignored edge accuracy (Wu et al., 2018). Therefore, a more advanced fully convolutional model, U-Net was proposed by Ronneberger, Fischer, and Brox developed for the biomedical image in 2015. The U-Net model utilizes up-sampling operators rather than pooling operations to concatenate features with higher resolution from the usual encoder part to better locate and learn the representations for the following convolution (Ronneberger, Fischer, & Brox, 2015).

However, there are two main limitations in U-Net as well. The first one is that during the backpropagation iteration, the parameter updating of the end layers on both sides of 'U' is before the intermediate layers, which makes intermediate layers less significant in terms of semantic representation (Wu et al., 2018). The other limitation is the sparse constraint applied in the intermediate features, which can be replaced by more explicit constraints for better performance and generalization of the model (Lin et al., 2017). Specifically, the constraints discussed here can be identified as the optimization between the predicted target and the related ground truth in a certain layer. Wu et al. (2018) utilized the multi-constraints based on the basic U-Net of skip connections to improve the usage of the feature's representation ability in the hidden layers. By updating the parameters using multi-constraints, the bias occurred in a single constraint solution was eliminated in each iteration (Wu et al., 2018). The building extraction result from the aerial image showed better performance compared to the classic U-Net in terms of the evaluation metrics. Additionally, Xu et al. (2018) integrated the framework of U-Net and the deep residual network (ResNet) to segment buildings using VHR aerial images. After the training of deep neural network Res-U-Net, a guided filter, which is an edge-preserving smoothing method, was proposed to fine-

tune the classification performance and remove the salt-and-pepper class noise (Xu et al., 2018). Different from Wu et al. (2018), the researcher in this project not only used infrared data but also the DSM for the purpose of accuracy improvement. However, Xu et al. (2018) used a more basic U-Net architecture without the application of multi-constraints as introduced in Wu et al. (2018). Table 2.2 summarizes the several building detection methods using different deep learning model. The strengths and limitations are discussed and compared as well.

Table 2.2 Summary of deep learning method on building detection

Model	Author	Implementation	Strengths	Limitation
Patch-based CNN	Mnih et al., 2013	• Using a patch-based CNN with extension of CRF	• Simple to implement • Appropriate for texture feature derivation at local scale	• Not suitable for texture feature derivation at large scale • Time consuming with extreme computing load; Averaging process disregards edge details
	Saito et al., 2016	• Using multispectral imagery including visible and infrared bands • Using CIS output function		
	Vakalopoulou et al., 2015	• Based on the AlexNet network • Using MRF model		
FCN	Sherrah, 2016	• Downsampling mechanism was replaced with no-down-sampling approach	• Memory efficiency • Training accuracy	• The missing of layers might reduce model feasibility • Ignoring edge accuracy
	Marmanis et al., 2016	• Using the succeeding aggressive deconvolution architecture		
	Maggiori et al., 2016	• Comparing between the patch-based CNN and FCN • Result shows that FCN outperforms patch-based CNN		
U-Net	Wu et al., 2018	• Using multi-constraints based on the basic U-Net	• Extraction accuracy • Concatenating features with higher resolution	• Intermediate layers are less significant • Sparse constraint
	Xu et al., 2018	• Integrating U-Net and ResNet		

On the other hand, the concept of feature fusion is considered in some researches, where multiple sources are used simultaneously in the neural network. One main component in feature fusion is data fusion, which concatenates multiple image sources into a single data cube to the following process (Zhu et al., 2017). In one study, the author first trained a VGG network using RGB data then combine the contribution of LiDAR that trained another VGG network using the DSM. After the training of CNN, the extracted factor was combined to train an SVM, and the final semantic class for each patch was then labeled (Lagrange et al., 2015). It is to be noted that the spectral information other than the standard RGB can improve the distinction abilities among the man-made objects such as buildings (Vakalopoulou et al., 2015). Different from a simple combination of RGB and DSM, Geng, Wang, Fan, and Ma (2017) extracted multiscale patch-based spatial features from SAR image and stacked them into a single sensor, then the output was used to train a supervised stacked autoencoder. Finally, a CRF was implemented after the stacked autoencoder to remove the influence of speckle noise in SAR images. The data fusion here is the consideration of different spatial information (e.g., edges, lines, and contours) extracted from three different spatial filters: gray level-gradient co-occurrence matrix (GLGCM), Gabor transform, and the histogram of oriented gradient (HOG) (Geng et al., 2017). Another aspect of feature fusion is to fuse the features extracted from various inputs, where two or more networks that are trained in parallel will be fused for the later stage (Zhu et al., 2017). Sherrah (2016) considered the feature fusion as well. He first trained the 3-bands data through a pre-trained VGG network to obtain colour features, and then utilized the elevation data such as DSM to learn an FCN from scratch. The object features from two different models were concatenated, and two randomly initialized FCNs were then trained from the concatenation. The visual context for feature fusion in the aerial image understanding was utilized in the paper of Marcu (2016), where a fully-connected layer performed the fusion between networks and was learned at various spatial scales. The performance showed that the two-pathway learning method can process the information complementarily and receive improved detection result.

2.3 Chapter Summary

This chapter reviews the related studies of building detection in the field of remote sensing, including the traditional building detection method and the advanced deep learning-based method. The template matching-based, knowledge-based, object-based and machine learning-based methods were introduced and discussed in the field of conventional building detection methods. As for the cutting-edge deep learning-based method, the patch-based CNN, FCN, U-Net and feature fusion were presented and compared in the review. It can be concluded that deep learning-based building detection method shows its capability to accurately and effectively detect and extract buildings from remotely sensed imagery.

Chapter 3 Methodology for Building Detection

This chapter gives a detailed introduction of the proposed method. Section 3.1 introduces the study area and the datasets, respectively. Section 3.2 introduces the workflow of the proposed method, with a detailed explanation of the processes at every stage. Section 3.3 describes the pre-processing for both aerial image and labelled data. Then, the architecture of CNN and the implementation of proposed model are described in Sections 3.4 and 3.5, respectively. Furthermore, the accuracy assessment and the implementation environment are introduced in Sections 3.6 and 3.7, respectively.

3.1 Study Area and Datasets

In order to examine the performance of the proposed building detection method, the city Christchurch in New Zealand, with more than 220,000 independent buildings covering 450 km^2, is selected in the thesis (see Figure 3.1). Christchurch is the largest city in the South Island, and the second-largest city in New Zealand with a population of 404,600 (Stats NZ, n.d.). Between September 2010 and February 2011, a series of earthquakes occurred in this region, which has resulted in many deaths and thousands of collapsed buildings across the city (Christchurch City Council, 2015). After that, the redevelopment project is undertaken involving the destruction of damaged buildings and the following construction of buildings with the same building footprints (Guo, Morgenroth, & Conway, 2018).

As shown in Figure 3.1, the selected study area contains the urban region of Christchurch including the Cathedral Square (the central city), the inner suburbs and its surroundings, which is covered by residential and industrial buildings with distributed grassland and trees. The majority of residential buildings in this study area are detached houses with a clear boundary, while most of the industrial buildings are large building clusters with different shapes. According to Figure 3.2, the urban area is mainly filled with various buildings; while the major vegetation region (Hagley Park) is located in the center of the study area surrounded by the commercial region, and a water region (Avon River) crosses the northeast of the study area.

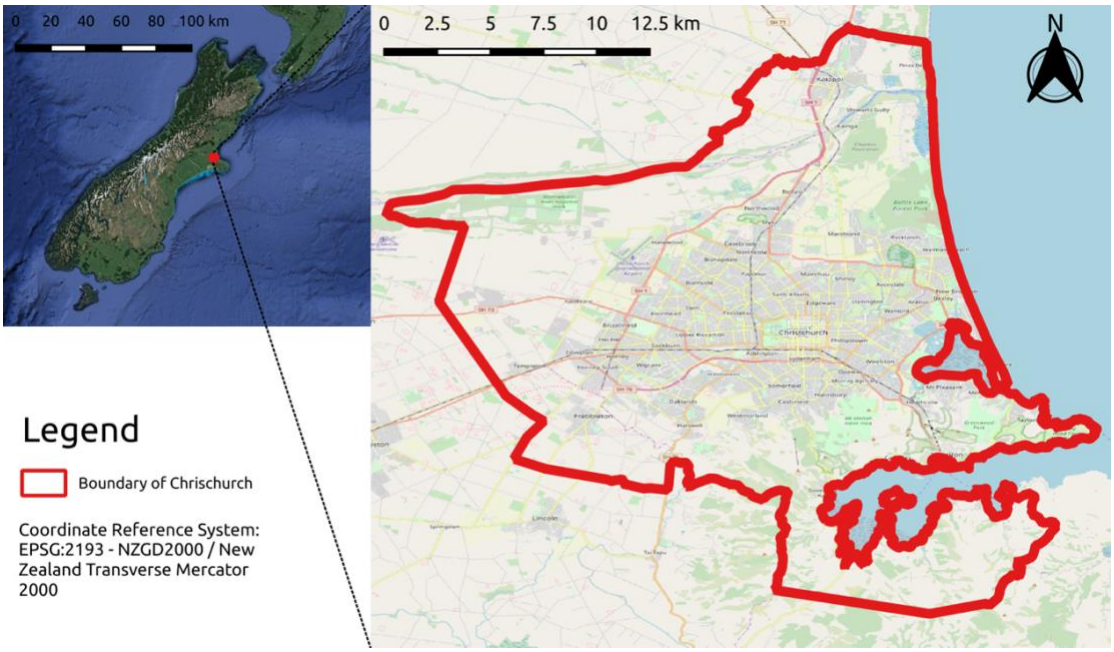


Figure 3.1 Location of Christchurch, New Zealand

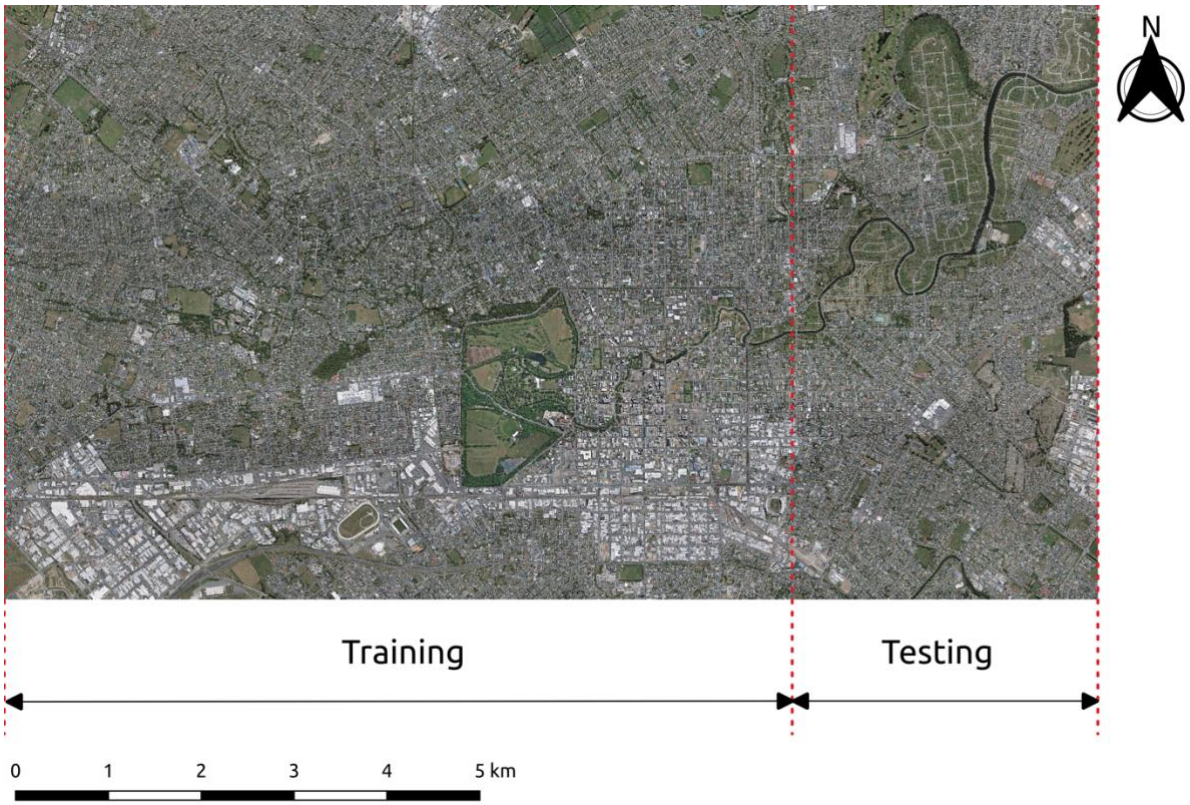


Figure 3.2 Aerial imagery of the study area

There are two datasets used in the study: the aerial image and the building vector data.

The ortho-rectified aerial images were collected from the Land Information of New Zealand (LINZ) on February 24, 2011, in response to the 22 February 2011 earthquake in Canterbury (LINZ Data Service, 2014). All images depict areas of the city Christchurch in a relatively cloud-free condition using the Vexcel UCXp camera at an elevation of approximately 1,600m. The ortho-rectified aerial images are in New Zealand Transverse Mercator (NZTM) projection with a spatial resolution of 7.5cm in bands R, G and B. The dataset contains 1785 tiles and each tile has 4,800*7,200 pixels. These tiles are then merged into a single mosaic. The compressed mosaic has a total size of around 25G, in the format of GeoTIFF. Figure 3.2 presents that the study area is divided into two areas: one for training (70%) and one for testing (30%). Table 3.1 summarizes the detailed specifications of main parameters in the aerial image data.

Table 3.1 Specifications of aerial image data

Parameter	Specifications
Data Source	LINZ
Captured Date	February 24, 2011
Camera	Vexcel UCXp
Flight Height	1,600m
Channels	R, G and B
Resolution	7.5cm
Projection	NZTM
Format	GeoTIFF

The building vector data, which is also the ground truth data, is manually edited by the laboratory from Wuhan University, based on Christchurch's building vector data (Ji, Wei, & Lu, 2018). The building vector data presents all the digitized building masks as shown on the aerial image. The vector data is divided through labeling building maps into shapefile with two classes: the building class and non-building class, respectively. Therefore, these images can be used for the supervised model training on both detection and segmentation tasks by employing the ground truth building masks.

After pre-processing, the training Dataset 1 has a total of 7,058 images with a pixel size of 1024*1024, combined with the corresponding building mask data in the same size. The validation dataset separated from training dataset to fine-tune the model has the same image size as the training Dataset 1. The training Dataset 2 has a total of 27,582 images with a pixel size of 512*512, combined with the corresponding building mask data in the same size. The corresponding validation dataset has the same image size as the training Dataset 2. The performance of building

detection model is then evaluated in the testing dataset with a total of 15,810 images and corresponding building mask data as a comparison. The details of the pre-processing process will be discussed in the following section.

3.2 Workflow of Proposed Methodology

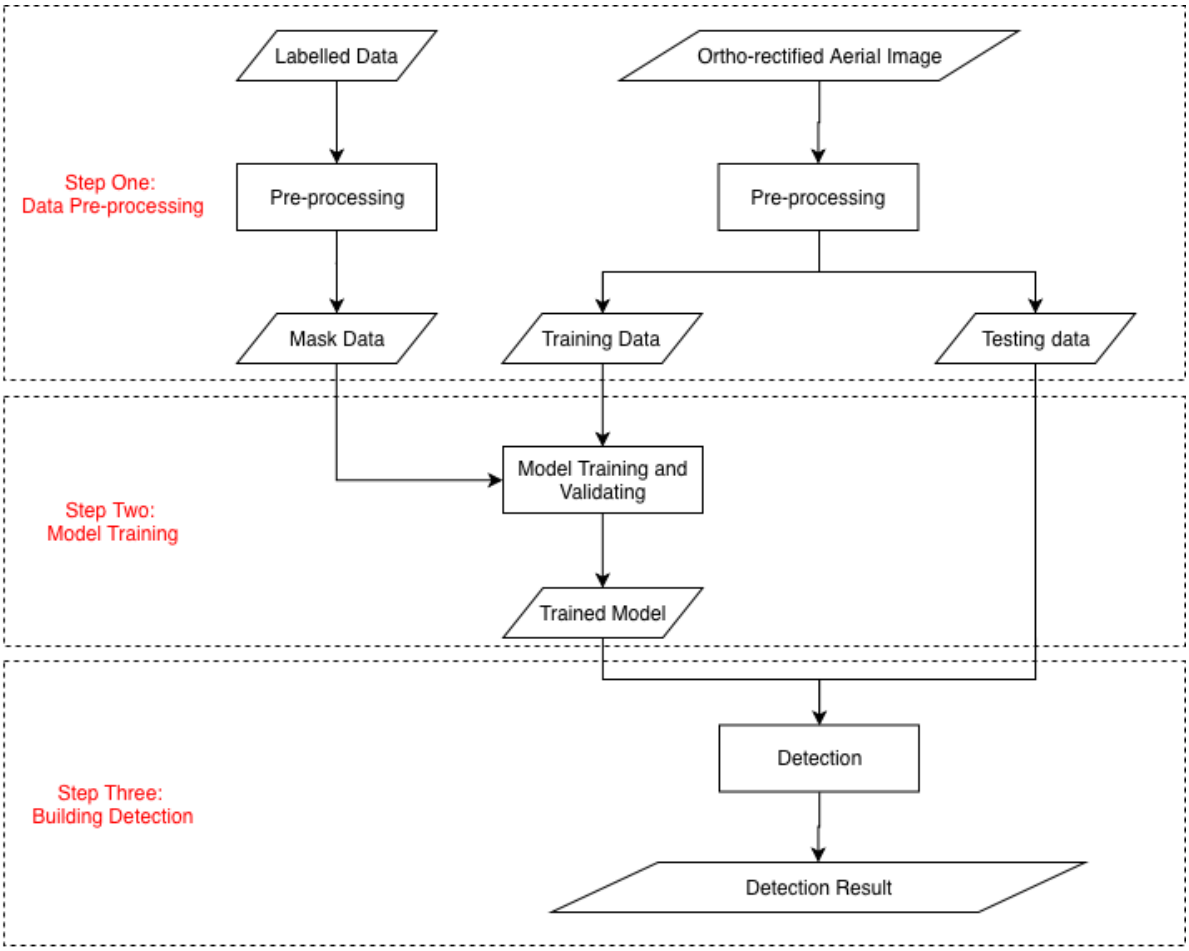


Figure 3.3 Workflow of proposed methodology

Figure 3.3 shows the overall workflow of the methodology that is implemented in this study, which can be divided into three main steps: data pre-processing, model training, and building detection. In the first step, both the aerial image and the labeled ground truth data were pre-processed to fit into the CNN model. After the preprocessing, these aerial image slices were divided into the training dataset and the testing dataset, combined with the corresponding labeled dataset. During the training process, the deep learning network was established, and the training dataset was used to train and validate the network. Through training and validation, the hyper-

parameters, the number of iterations, the spatial resolution of images and other parameters were optimized and determined to find the optimal detection model. Furthermore, the corresponding labeled dataset was employed to compare the correctness between the predicted result and the ground truth to train the model. In this step, a well-trained model with defined parameters was developed for building detection. Finally, the trained CNN model was applied on the testing dataset to detect the building objects and segment buildings from the aerial image, and the detection and segmentation result was extracted for accuracy assessment and future application.

3.3 Data Pre-processing

The aim of the pre-processing is to establish the appropriate structure of input data to be used in the CNN model, which can be divided into two sections: the pre-processing of the aerial image, and the pre-processing of the labeled data.

3.3.1 VHR Aerial Image Pre-processing

Before the pre-processing, the aerial image was ortho-rectified by the provider (LINZ Data Service, 2014). During the orthophoto generation, the original image was processed using the NZGD2000 reference system and the Digital Terrain Models (DTM) from the LINZ source. The ortho-rectified process means that the distortion as a result of sensor and earth's topography has been geometrically removed. Therefore, the ortho-rectified aerial image can be used directly in the study for a more accurate measurement of buildings.

It cannot be denied that the high resolution of aerial imagery will also lead to shadow visibility. As discussed in Section 1, the shadow information can be mistakenly classified as building due to the similar outline, which requires an additional shadow removal algorithm. However, in the use of deep learning, the shadows cannot be mistaken due to the pixel intensity consideration during the feature extraction process. Therefore, there is no need for shadow removal algorithm in the data pre-processing for building detection.

The aerial image from the study area was primarily divided into training and testing regions as shown in Figure 3.2. The aerial images from both regions were then scanned and sliced into patches based on a pre-defined sliding window. For training Dataset 1, the sliding window cropped the image into slices of 1024*1024 pixels at a stride of 500 pixels. The setting of the sliding window is important. A large slice size will result in the increase of computation and the complexity of retrieve feature during the training process; while a small slice size might not be enough to capture the spatial context of one building. The stride length is almost half of the image, the decision of which will be explained in Section 3.3.2. For training Dataset 2, the original training image slices were down-sampled during the cropping process by performing a 2*2 averaging window. Therefore, the image in training Dataset 1 with 1024*1024 pixels was down-sampled to 512*512

pixels to increase computational efficiency. Through further experiments, whether the down-sampled process will influence the accuracy of the building detection result are tested and discussed.

Considering the overfitting issue, two solutions were employed in the preparation of data: the bias data removing and data augmentation. Since biased data will contribute to overfitting, a threshold was applied on the extracted slices to filter out images with no building object (Goodfellow, Bengio, & Courville, 2016). In addition, the band shuffling was applied to 30% of the random testing data for data augmentation. The red, green and blue channels in the selected images were shuffled to present the image with a different combination of bands, which will decrease overfitting effect and increase the model performance (Bei et al., 2018).

3.3.2 Labelled Data Pre-processing

To be consistent with the aerial image, the labeled data was sliced into the same size as the image patch with 512*512 pixels. Since the CNN model only accepts labeled data in image format, the labeled data was transferred into a binary mask map in which 0 represents background class, and 1 represents building class. In the processing of image cropping, every independent building was detected to output a mask image. Thus, for each aerial image patch, multiple mask images would be extracted and processed in the training process, if there were more than one building in that aerial image. However, the buildings that cross the boundary of each patch will be cropped into two or more incomplete parts using the sliding window, and the incomplete building mask will influence the prediction accuracy. Therefore, two solutions were used to avoid this situation. The first one was to detect and remove buildings across the boundary of the image patches. The other solution was to use the stride during the window sliding, which can make neighboring images overlap and include buildings that were removed in the previous step. The setting of stride is important. Although a small stride length can enhance the extraction accuracy, it will also increase the overall computation load. In this study, the stride was set to 500 pixels, which is close to the half of the image size, to cover most of the independent buildings.

Finally, after data pre-processing, the training Dataset 1 contained 7,058 image samples with the corresponding mask image, and the training Dataset 2 contained 27,582 image samples with the corresponding mask image. The testing dataset contained 15,810 image samples with the labeled vector map.

3.4 Proposed Model

In Section 2.2.2, the typical architecture of a normal CNN was introduced and consists of the convolutional layers, pooling layers, and fully-connected layers. However, during the implementation of this study, the architecture of the proposed model is different from normal

CNN, and the detailed model will be illustrated in the next section. For this section, the key components of the proposed CNN model including main layers, training, and testing are discussed to provide a better overview of the implemented method.

3.4.1 Convolutional Layer

The most important layer and main calculation operation in the CNN should be the convolutional layer, where kernels are used to convolve over input 2D image and extract feature map using the following equation:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n K(i-m, j-n) I(m,n) \tag{3-1}$$

where *i, j* denotes the row and the column number of the image, and *m, n* denote the serial number in the kernel. *I* is the input image, and *K* is the 2D kernel. *S* is the output image after convolution. Taking Figure 3.4 as an example, it shows the convolution process of a 5*5 3-channel image with 2 kernels. The blue matrix represents the intensity value of the three colour channels in the input image with a size of 5*5, and the image is extended to 7*7 to fit the final output size using zero-padding on the edges. The right matrices indicate the two kernels (or filters) with a size of 3*3; since the input has three channels, each filter has three matrices as well. With a stride of 2, the kernel slides through each channel of the input image for convolution. At each location of the kernel, an element-wise multiplication between the input arrays and the kernel is executed, and the values of these multiplications are summed and added to a bias to get a single value in the output of the corresponding location. The green matrix is the output which is also the feature map, and the dark green square shows the corresponding location of the dot product. In this case, two separate kernels convolve the image input horizontally and vertically, until the image input is convolved entirely, to produce the two feature maps shown on the right (Santos, n.d.). Moreover, the relationship between the input size and the output size can be calculated using the following equation:

$$W = \frac{W - F + 2P}{S + 1} \tag{3-2}$$

$$H = \frac{H - F + 2P}{S + 1} \tag{3-3}$$

$$D = K \tag{3-4}$$

where [*H, W*] is the input size padded by *P*; *F* is the size of a square kernel; and *S* is the stride value. And the dimension of output (*D*) equals to the number of kernels (*K*).

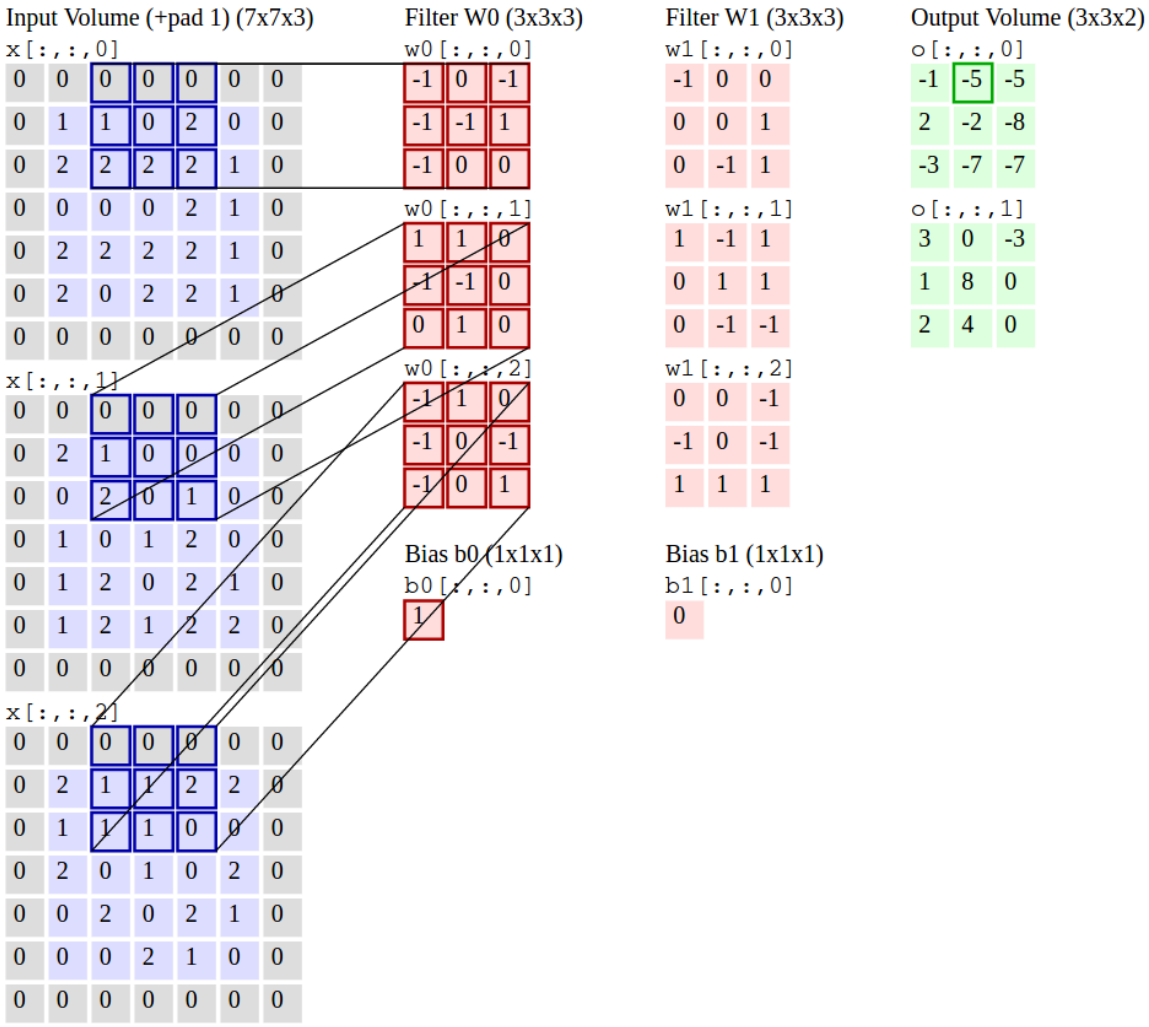


Figure 3.4 Convolution operation

(Source: Karpathy, 2016)

3.4.2 Pooling Layer

Unlike normal CNN where the convolutional layer alternates with pooling layer, the implemented CNN does not involve pooling operation much. In ResNet, there is only one max pooling layer and one global average pooling layer at the start and the end of the network, respectively. As discussed before, the max pooling layer is used to add hierarchy and reduce dimensionality. Instead of multiplying with each array and producing the multiplied result, this special filter only extracts the maximum value outside of the processed region. Figure 3.5 shows an example of a max pooling layer with a size of 2*2 with a stride of 2. For every 2*2 block with

different colours in the input, the largest value is selected to be the output value, and this operation iterates by taking a step or two pixels, until the whole image input has been processed. The max pooling layer used in ResNet has a size of 3*3 with a stride of 2. Average pooling, on the other side, is to get the average value of the selected block. The aim of using an average pooling layer at the end of the network in ResNet is to avoid a fully-connected layer. However, in this study, since ResNet is combined with FPN, there is no need for an average pooling layer. The extracted feature maps at different stages of ResNet were combined directly using the Region of Interest (ROI) pooling.

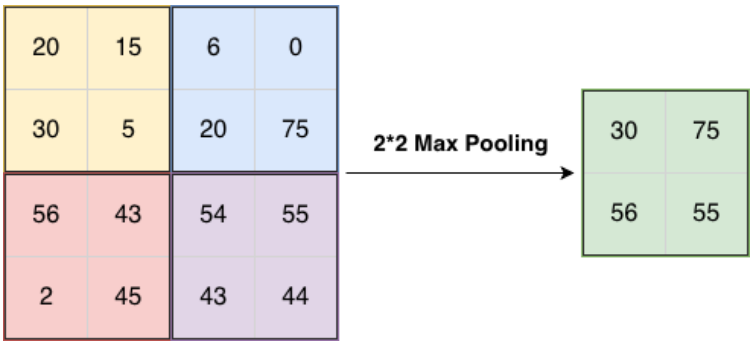


Figure 3.5 Representation of max pooling

3.4.3 Batch Normalization

As one of the commonly used normalization methods, batch normalization aims to normalize the output of the convolutional layer and keep the mean value close to 0 and standard deviation close to 1. The value of normalized x is computed using the following algorithm:

$$\hat{x}_I = \frac{x_i - \mu_\beta}{\sqrt{\delta_\beta^2 + \varepsilon}} \tag{3-5}$$

where μ_β is the mean value over the mini-batch β, and σ_β is the standard deviation over the mini-batch β:

$$\mu_\beta = \frac{1}{M}\sum_{i=1}^{m} x_i \tag{3-6}$$

$$\sigma_{\beta}^{2} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\beta})^2 \tag{3-7}$$

where *m* is the total inputs of the mini-batch. The e is set to a small positive value such as 1e-6 to avoid zero denominators.

Batch normalization has been proved to effectively increase the speed of convergence, allowing the use of a higher learning rate, reduce overfitting and shorten the training time. This results in a better performance of the implemented model (Ioffe & Szegedy, 2015).

3.4.4 Activation

After the input is convolved and normalized, an activation layer is applied to the output to determine whether a node is accepted or denied for the following process. Since there is no weight and bias learned inside activations, activations are not technically 'layers' in the architecture of deep learning. In practice, the activation function employed is the ReLU function and Sigmoid function. ReLU is an activation function that calculates max(0, x) where x denotes the output matrix after convolution. After the process of ReLU activation, outputs with a negative value are assigned to zero, and positive values and zero values are left unchanged. Sigmoid activation is used in the ROI pooling layer as the last layer of FCN. Sigmoid activation uses the following equations:

$$t = \sum_{i=1}^{n} w_i x_i \tag{3-8}$$

$$s(t) = 1/(1 + e^{-t)} \tag{3-9}$$

where *t* represents the weighted sum, *s* is the sigmoid function, *w* is the weight, and *x* is the input image value with a total number of *n*.

Both ReLU and Sigmoid activations are non-linear functions and produce the non-binary result. After applying activation functions, an activation map is produced for each kernel; in other words, the dimension of the activation map is the same as the number of kernels.

3.4.5 Fully-connected Layer

The fully-connected layer is applied at the end of CNN, where all the activations in the previous layer are connected. The fully-connected layer used in the proposed model is also called a dense

layer. A linear operation is conducted in which every input/node is connected to every output by weight using the following formula:

$$Output = f(\sum_{i=1}^{n} w_i x_i + b) \tag{3-10}$$

where *w* is the weight, *x* is the input image value with a total number of *n*, *b* is the bias value, and *f* is the activation function. The activation used in the proposed model for bounding box refinement branch is the linear activation, and no activation is processed in the classifier branch.

3.4.6 Loss Function

The loss function is one of measurements to estimate the error between the network prediction and the ground truth. The smaller the loss, the better the predictor is at predicting the correct class type of the input. In this study, there are five losses considered to produce the final loss function: the RPN anchor classifier loss, the RPN bounding box loss, the loss for the classifier of Mask R-CNN, the loss for Mask R-CNN bounding box refinement, and the loss for mask segmentation. In practice, these losses all have a weight of 1 in the final loss calculation, which means they are considered evenly:

$$Total\ loss = rpn\_class\_loss + rpn\_bbox\_loss + mrcnn\_class\_loss + mrcnn\_bbox\_loss + mrcnn\_mask\_loss \tag{3-11}$$

For the RPN anchor classifier loss, both positive (matched and true) and negative (matched but false) anchors contribute to the loss while neutral anchors (not matched) do not, and the cross-entropy loss was used to measure the difference between the predicted anchor type and the ground truth. For the Mask R-CNN classifier loss, both positive (predicted and true) and negative (predicted but false) class labels contribute to the loss, and the cross-entropy loss was used to measure the difference between the predicted class type and the ground truth. The cross-entropy loss calculates loss by:

$$CE = -\sum_{i}^{C} t_i log(s_i) \tag{3-12}$$

where *CE* is the cross-entropy loss, t_i is the ground truth, S_i is the standard scoring function form for each class I in C:

$$s = f(x_i, W) \tag{3-13}$$

where x_i is the input minibatch, and *W* is the weight matrix. f is the activation function.

For the RPN bounding box class loss, only positive anchors contribute to the loss, negative and neutral anchors do not. The smooth L_1 loss was used to measure the bounding box loss for bounding box refinement. For the Mask R-CNN bounding box loss, only positive ROIs (predicted and true) contribute to the loss, and it uses smooth L_1 loss as well. The smooth L_1 loss is a combination of L_1loss and L_2 loss, where it uses L_1 loss when the difference between the predicted value and true value is less than 1and L_2 loss otherwise:

$$Smooth\ L_1\ loss\ = \begin{cases} 0.5|x|^2, & |x| < 1 \\ |x| - 0.5, & otherwise \end{cases} \quad (3\text{-}14)$$

where $|x|$ denotes the L_1 loss, which calculates the difference between the predicted value and ground truth:

$$L_1 \coloneqq (a, b, N) \longrightarrow \sum_{i=1}^{N} |a[i] - b[i]| \quad (3\text{-}15)$$

where *a* denotes the predicted value, and *b* denotes the ground truth. *N* denotes the number of classes.

For the loss of mask segmentation, only the positive ROIs (predicted and true) contribute to the loss, and a binary cross-entropy is used to calculate the loss:

$$CE = -\sum_{i=1}^{C'=2} t_i\ log\ (s_i)\ = -t_1\ log\ (s_1)\ -(1 - t_1) log(1 - s_1) \quad (3\text{-}16)$$

where two classes C_1 and C_2 are considered. t_1 and s_1 are the ground truth and the score for class C_1 , and t_2 = 1-$t_1$and s_1 =1-s_1 are the ground truth and the score for C_2.

3.4.7 Backpropagation

Backpropagation is the main process in CNN to train the model. The backpropagation calculates the gradient of the loss function in the final layer of the network and uses the gradient to iteratively update the weights in the network. Figure 3.6 is diagram for backpropagation. In the forward pass, the input is passed through the network and the output predictions are acquired. The loss function is executed to calculate the difference (error E) between the actual output and the desired output (ground truth). In the backward pass, the derivative of loss function or the gradient is computed to update the weights by using the chain rule in the computational process. And by the end of the process, the effect of each parameter in the kernels on the final loss of the network can be identified. The parameter of the weight update depends on the choice of the optimizer. The forward

and backward steps iterate through the network, until the model converges, or the number of iterations is met.

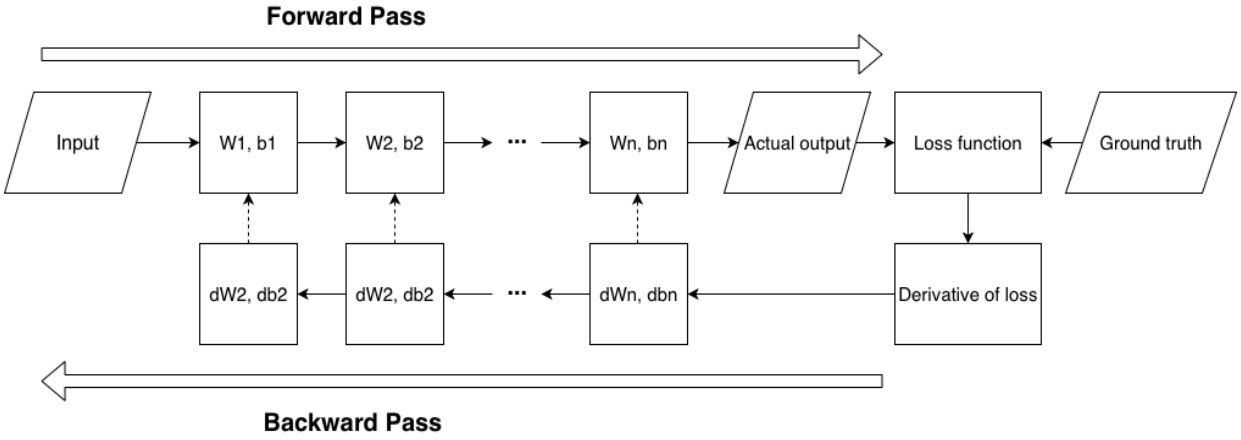


Figure 3.6 Diagram of backpropagation

3.4.8 Optimization Method

Optimization method decides the parameter of the weight update as stated in the last paragraph. One widely used optimizer in deep learning is the Stochastic Gradient Descent (SGD) (LeCun, Bottou, Orr, & Muller, 2012). SGD calculates the gradient and updates the weight on a small batch of input data, and the use of small patch means that an update occurs for every batch, rather than an update per epoch (Rosebrock, 2017). The formula of SGD is shown in the following algorithm:

$$V_t = \beta V_{t-1} + (1-\beta)\nabla_w L(W, X, y) \quad (3\text{-}17)$$

$$W_t = W_{t-1} - \alpha V_t \quad (3\text{-}18)$$

where W_{t-1} is the original weight, and W_t is the updated weight. α is the learning rate to control the speed of weight update which will be tested and decided in Section 4.1.3. β is the momentum to the accumulated gradient of the past for faster convergence. Specifically, the aim of momentum is to increase the speed of updates for gradients in the same direction while decreasing the effect of gradients in the opposite direction. The default value of momentum is 0.9, which is proved to be the optimal value supported by papers and tutorials (Mitliagkas, Zhang, Hadjis and Ré, 2016). *L(W, X, y)* is the loss function using weigh *W*, input *X*, and ground truth *y*.

3.4.9 Fine-tuning

These networks already contain rich and distinguishable filters which can be used on other datasets and classes. However, the network cannot be directly applied to the dataset. A solution called ‘fine-tuning’ was used to modify the architecture and re-train the networks. Figure 3.7

shows the typical architecture of a CNN network (VGG16). The blue box represents different convolutional layers; the white box represents a different pooling layer while the green box represents the final fully-connected layers. The "head" is the final set of layers including the three fully-connected layers and the softmax classifier. During the fine-tuning process, the original head in the architecture was replaced by the new fully-connected head; therefore, the new head with random initialization can be fine-tuned to the specific dataset used in this study (shown in Figure 3.8). Basically, in the forward propagation, the training data passed through the network normally as discussed before. However, in the backpropagation, the rest layers before the head were 'frozen', which means the parameters in these layers remained unchanged, and only the fully-connected layer was trained from the highly discriminative convolutional layer. The formal layers were unfrozen and started to be trained until the new fully-connected layer learned patterns and achieve a certain accuracy. Although the process of fine-tuning requires more complex work and relies heavily on the choice of new heads parameters, it can take the full advantage of the pre-existing and discriminative network architectures that have been trained on the ImageNet dataset, and increase the final model accuracy with less effort, compared to training the network from the beginning.

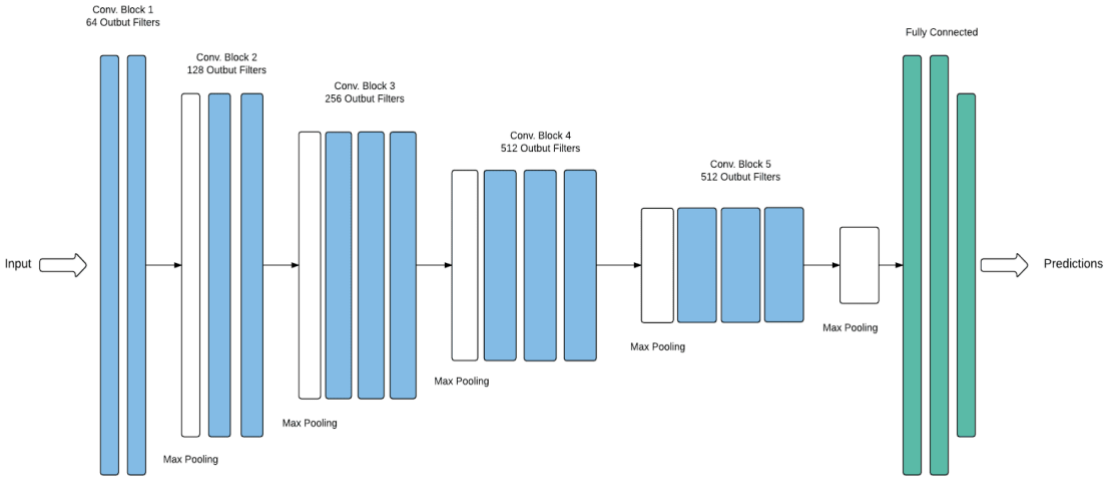


Figure 3.7 Architecture of VGG16 network

(Source: Amaratunga, 2019)

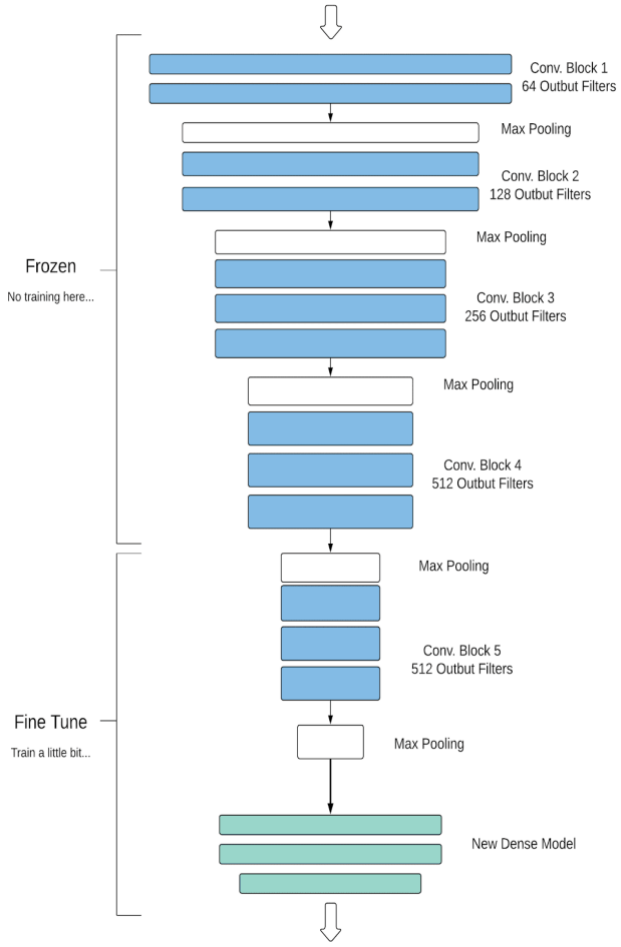


Figure 3.8 The fine-tuning of model

(Source: Amaratunga, 2019)

3.5 Proposed Network Implementation

The technical details of the employed CNNs are described in the following sections. The literature references that introduce the proposed network are provided for additional information.

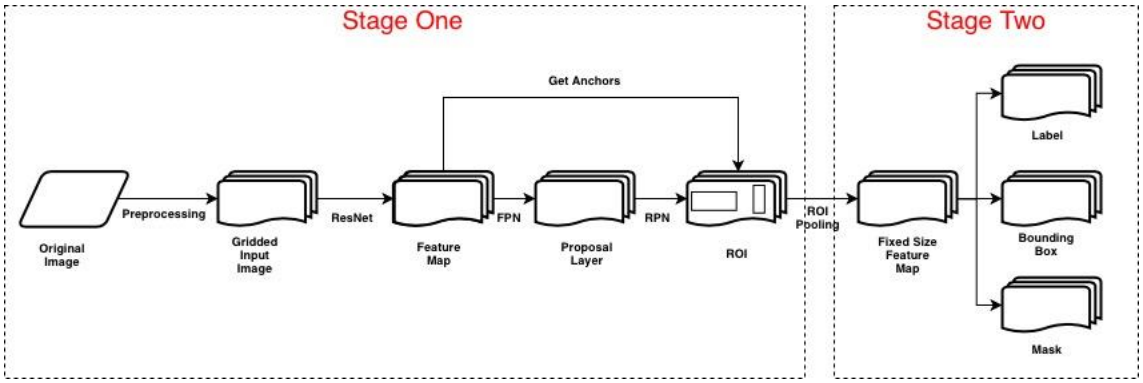


Figure 3.9 Architecture of proposed Mask R-CNN

Proposed by He, Gkioxari, Dollár, and Girshick on 2017, Mask R-CNN is based on Faster R-CNN (Ren, He, Girshick, & Sun, 2015), in which the author added a third branch of mask segmentation to the Faster R-CNN architecture. Figure 3.9 shows the overall architecture of Mask R-CNN used in this study, which was built based on the Matterport Mask R-CNN implementation (Abdullat, 2018).

Generally, there are two stages in Mask R-CNN (see Figure 3.9): first, the feature map was extracted from input images to generate the region proposal layer using the region proposal network (RPN); second, the proposed regions were squashed into a feature map to produce the final mask, bounding box and label after processing. Therefore, during the forward process, the input of Mask R-CNN was the original image slices, and the output was the bounding box of the building class and the mask vector map, both in the format of shapefile.

3.5.1 ResNet and Feature Pyramid Network

In the first stage, the original image was first gridded into image slices based on the grid size during the pre-processing. In the Mask R-CNN implementation, the image can be resized to any shapes (i.e., rectangular and square) with a scaling parameter to fit into the CNN model. However, since the gridded images already had the appropriate size and resolution, this step was skipped in this study. The image slices were then passed through a deep CNN to extract the feature map. The deep CNN used here is also called backbone, in which the standard CNN was used to transfer a raw image into a feature map. The backbone feature extractor used in the study is ResNet and Feature Pyramid Network (FPN), both of which were pre-trained using the ImageNet classification dataset (Russakovsky et al., 2015).

The ResNet introduces the concept of residual learning where the subtraction of the input, known as residual, is learned for the next layer. The residual can be simplified as the following residual function:

$$F(x) = H(x) - x \quad (3\text{-}19)$$

where *H(x)* is the output, and *x* is the input to the first of the layers, both of which is in the same dimension. And F(x) is the residual. Therefore, the original *H(x)* function has become the following function

$$H(x) = F(x) + x \quad (3\text{-}20)$$

for subsequent operations.

Rather than learned directly from the former layers, a right branch as shown in Figure 3.10 was added to the output, which is called "shortcut connections". The input of the nth layer was directly connected with the additional operation to the *(n+x)*th layer (*x* represents the number of layers in between) before the ReLU activations, while the left branch still follows the standard CNN construction in which a set of convolutions, batch normalizations and activations were applied on the input. Additionally, the ResNet implementation with more than 50 layers utilized the 'bottleneck' design with three layers in each identity block to sum residuals in each stage, which is a simple and efficient solution for deeper neural network implementation (Li et al., 2018).

The early layers in ResNet detected low-level features such as edges and corners, and later layers could successively detect high-level features such as rooftops and buildings. By passing through the backbone network, one image was transformed from 512*512*3(RGB channels) to a feature map of shape 1*1*2048, which became the input for the following stages.

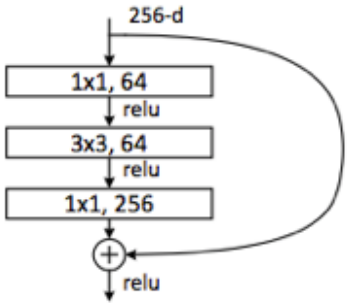


Figure 3.10 Architecture of a residual block

(Source: He, Zhang, Ren, & Sun, 2016)

Additionally, in order to better describe features of objects at multiple scales, the FPN was implemented as well in Mask R-CNN. By building up multiple levels of representations at different scales in the forward process, the FPN collected the feature activations output from different stages of ResNet as shown in Figure 3.11, which was also called "bottom-up pathway" (Lin et al., 2017).

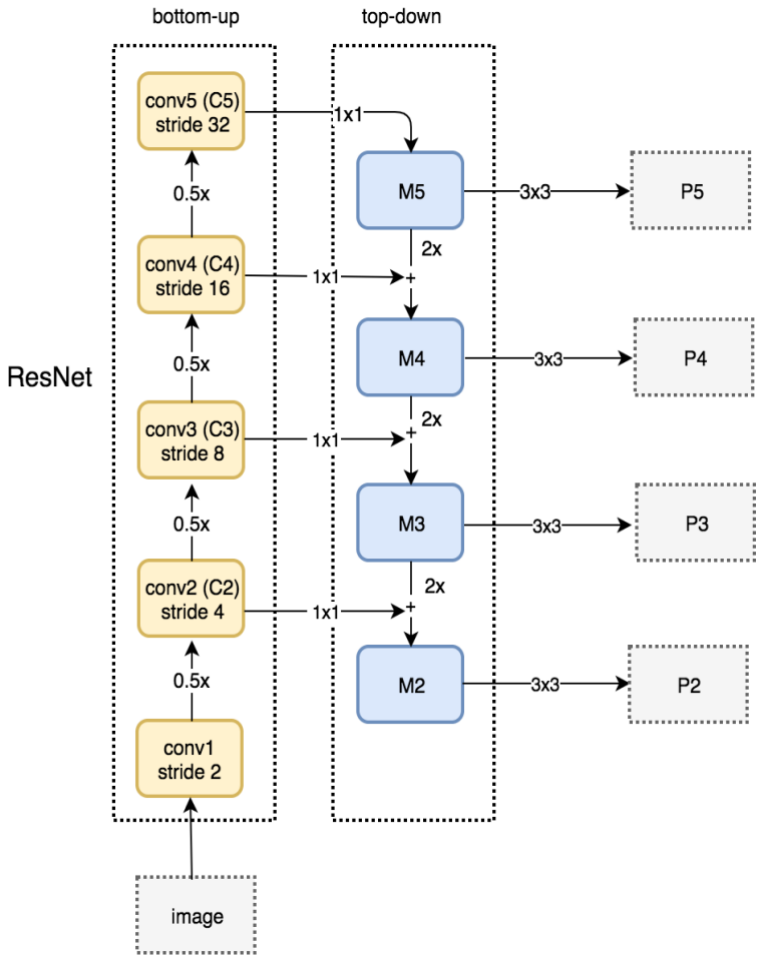


Figure 3.11 Diagram of FPN on the bottom-up and the top-down pathway

(Source: Hui, 2018)

Although the map resolution decreased as the pyramid layer increased, the semantic value of each layer was increasing with higher-level structures detected. Then, in the top-down pathway, higher resolution features were constructed from higher pyramid levels to lower pyramid levels using up-sampling by a factor of 2. Furthermore, the reconstructed up-sampling map (M5, M4, M3 and M2 in Figure 3.11) was merged with the corresponding feature maps to better detect object's location by element-wise, during which a 1*1 convolutional layer was implemented to reduce channel dimensions (Lin et al., 2017). And finally, to reduce the aliasing effect brought by up-sampling, a 3*3 convolutional layer was applied on each merged layer and the final feature maps (P5, P4, P3 and P2 in Figure 3.11) were then produced.

3.5.2 Region Proposal Network

After multiple feature maps were extracted by FPN, the feature map layer at the most proper scale based on the size of the object was selected to extract feature patches in the next important process: Region Proposal Network (RPN).

RPN is a neural network for object detection that uses a sliding window to scan the image within a region and identifies regions containing objects. The region in RPN is called “anchors”. A set of locations were distributed evenly and uniformly on the extracted feature map as show in Figure 3.12. For each location, a total of 15 anchors with different shapes and scales were created to pick up the ROIs. In this study, the shape and size of anchors were determined by two parameters: RPN_ANCHOR_SCALES and RPN_ANCHOR_RATIOS, respectively. The RPN_ANCHOR_SCALES was set to (32, 64, 128, 256, 512), where each number represents the length of square anchor side in pixels. The RPN_ANCHOR_RATIOS was set to [0.5, 1, 2] where 1 represents the square anchor, and 0.5 and 2 represent rectangular anchor. The ratio value here equals to the ratio of width to height. Then, the FPN applied a 3*3 convolutional layer on the feature map with two 1*1 convolutional layers for a class label and a boundary box refinement separately. The three convolutional layers are called the RPN head. The class label simply defined whether there is a possibility of an object in the anchor with two classes: the foreground or the background. The bounding box refinement was applied to estimate the offset of the foreground anchor and refine the boundary bounding box (anchor) to better fit the object. With multiple anchors on the same object, a Non-maximal suppression was applied on these duplicate detections, and the anchor with the highest probability score was selected as the ROI.

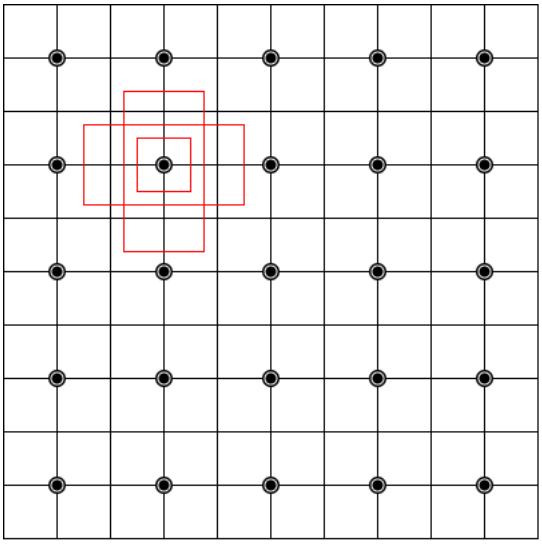


Figure 3.12 Visualization of anchors

Since the ROI had been extracted, there were two outputs in one ROI that was similar to RPN: the class and bounding box. However, the class label was no longer to differentiate the foreground and background but to classify the specific classes (e.g., cats, dogs and human); while in this study, there is only one class: building. The bounding box was operated with another refinement to further localize the object and refine the boundary and size of the bounding box to encapsulate the object. Moreover, since the ROIs were in different sizes and shapes after the bounding box refinement process, while the classifier requires a fixed input size, the ROI pooling process was needed before classification. In this study, ROI pooling refers to the simplified process where a part of the featured map containing the object was cropped and resized to a fixed size (7*7 in setting).

3.5.3 Mask Generation

Finally, the extracted positive regions from ROI was fed into another CNN to generate masks covering the contour of an object at pixel-scale. The CNN employed here is a small FCN, which was introduced in Section 2.2.3. During the implementation, four convolutional blocks including a convolutional layer, a batch normalization layer and a ReLU activation, followed by a deconvolutional layer and sigmoid activation was processed to extract a mask represented by float numbers. The size of the mask was set by MINI_MASK_SHAPE, and the setting in this study was set to (128, 128), representing the height and width of the mini-mask in pixels. A smaller mask size can save the memory load, while the mask resolution becomes lower compared to the ROI bounding box. Despite that, the float numbers contain more details than the final binary mask. Hence, after the mask was extracted in the prediction process, the mask was scaled up to the size of the bounding box to produce the final binary mask for the object. Consequently, by applying Mask R-CNN, the model processed the input image and produced three outputs: the bounding box around the building object, the class label, and the mask of the object.

During the implementation, for the first two training stages, the ResNet backbones layers were frozen, and only the RPN, classifier and mask heads were trained for 10 and 40 epochs with a learning rate of 0.001 and 0.0001, respectively. Next, the ResNet layers in stage 3 and up were unfrozen and trained for 120 epochs to fine-tune layers with a learning rate of 0.0001. Finally, the rest of the layers were added in the training to finetune layers for 200 epochs with a learning rate of 0.00001. Furthermore, the pre-trained weights from COCO dataset were used as the model initial weight parameters to achieve faster convergence and higher accuracy.

3.6 Accuracy Assessment

With various parameters in the implementation of models, it is difficult to decide the optimal model for the particular detection problem in this study. In order to verify the performance and

reliability of the proposed model, accuracy assessment was conducted on the building detection result using the following metrics.

3.6.1 Intersection-over-Union

Before the discussion of accuracy assessment, an important concept needed to be noted is the IOU. IOU is an evaluation metric to measure the accuracy of an object detection result over a specific dataset.

$$IOU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{3-21}$$

where in the numerator, the area of overlap between the predicted object region and the ground truth region is calculated. In the denominator, the area of union where the predicted object region and the ground truth region intersects is calculated. Dividing the area of overlap by the area of union yields the final output-- the IOU score. The higher the IOU score, the better the prediction is. For example, if the IOU is set to 0.5, then once the overlap between the predicted object region the ground truth region is greater than or equal to 50%, the prediction is identified as a correct prediction.

3.6.2 Confusion Matrix

A confusion matrix, also known as an error matrix, is a table to visualize the performance of a method, especially for a supervised learning model. Table 3.2 gives an example of a confusion matrix for binary classification, which will be used in this study as well. This table shows the correct predictions (the diagonal) and the incorrect predictions on what classes are assigned. In detail, true positive (TP) denotes that the prediction is a correct detection, which means the IOU between the prediction and ground truth is greater than or equal to the threshold. False positive (FP) denotes that the prediction is a wrong detection, which means the IOU between the prediction and ground truth is smaller than the threshold. False negative (FN) denotes that a ground truth region fails to be detected by the detector, and true negative (TN) denotes that the detector does not detect a region and it is correct. However, since there are only two classes-- building and background in this case-- the ‘background object’ makes no sense, thus TN is not considered for accuracy assessment.

Table 3.2 An example of confusion matrix for binary classification

		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

3.6.3 Precision, Recall and F1-score

In order to fully evaluate the effectiveness of the proposed model, precision and recall are used to quantitatively illustrate the confusion matrix.

Precision aims to find what proportion of positive prediction is actually correct; in other words, precision defines how much the predictor can identify the object as buildings. The formula of precision is shown as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3-22}$$

where precision equals to the number of correct predictions divided by the total number of predictions. Average Precision (AP) is the averaged precision over all the classes:

$$AP = \frac{1}{M}\sum_{i}^{C} P_i \tag{3-23}$$

where P_i is the precision for each class i in C, and M is the number of classes. Since there is only one class (building) in this study, AP equals the precision.

Recall, however, attempts to find what proportion of ground truth are correctly reflected by predictions; more specifically, recall evaluates how the predictor can correctly identify the object as buildings. The formula of recall is shown as follows:

$$Recall = \frac{TP}{TP + FN} \tag{3-24}$$

where recall equals to the number of correct predictions divided by the total number of ground truth. Averaged Recall (AR) is the averaged recall over all the classes:

$$AR = \frac{1}{M}\sum_{i}^{C} R_i \tag{3-25}$$

where R_i is the precision for each class *i* in *R*, and M is the number of classes. Since there is only one class (building) in this study, AR equals the recall.

In addition, mean Average Precision (mAP) denotes the averaged AP over IOUs for the testing dataset, where IOU is set to 0.5, 0.6, 0.7, 0.8:

$$mAP = \frac{1}{4}\sum_{i}^{4} AP_i \tag{3-26}$$

where AP_i is the AP over IoU at 0.5, 0.6, 0.7, 0.8, respectively.

Mean Average Recall (mAR) denotes the averaged AR over IOUs for the testing dataset, where IOU is set to 0.5, 0.6, 0.7, 0.8:

$$mAR = \frac{1}{4}\sum_{i}^{4} AR_i \tag{3-27}$$

where AR_i is the AR over IoU at 0.5, 0.6, 0.7, 0.8, respectively.

According to the equations of precision and recall, it is easy to find that the two metrics are in tension, which means that an increase in precision leads to a decrease in recall and vice versa. Therefore, it is important to introduce a metric that balances precision against recall: the F1-score.

The F1-score is the harmonic mean of precision and recall using the following equation:

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{3-28}$$

The harmonic mean of the two metrics can effectively decrease the influence of extreme values. F1-score has a range between 0 and 1, where 0 denotes the worst value and 1 denotes the best score.

3.7 Implementation Environment

The implementation of deep learning method in this thesis uses Keras with Tensorflow backend. Keras is an open-source neural network library written in Python. It aims to establish a user-friendly, modular and extensible environment (Keras, n.d.). Keras not only supports convolutional networks, but also recurrent networks and the combination of both. The high-level wrappers of Tensorflow in Keras make it easier to combine standalone modules for the creation of a new deep earning model. These modules including the neural layers, the cost functions, the activation functions, optimizers, etc.

There are many backend frameworks in Keras, such as TensorFlow, CNTK or Theano. In this study, TensorFlow was selected as the backend library. TensorFlow is an open-source software library for tensor calculation with a neural network.

The proposed model is implemented on the computing environment with the following hardware specifications:

- One Intel® CPU i7-9700k @ 3.60GHZ (eight cores)
- 64GB RAM
- 1TB of local SSD storage
- One NVIDIA GeForce GTX 1080 8GB GPU

3.8 Chapter Summary

In this chapter, the methodology of building detection using the deep neural network from VHR imagery is presented in detail. The workflow of building detection method includes three steps: the data pre-processing, the model training and the building detection. In data pre-processing, both aerial image data and building vector data were processed to fit the CNN model. Then, the processed data was used to train and validate the CNN model. Finally, the optimal CNN model was used to conduct building detection on the testing data. The CNN model employed in the thesis is Mask R-CNN that has two stages. In the first stage, the input image was processed to extract the feature map, and the feature map was used to generate the region proposal layer using RPN. The proposed regions were then used to produce the bounding box, class label and mask. In addition, the metrics of accuracy assessment used in the thesis are IoU, confusion matrix, precision, recall, and F1-score. The implementation environment of the method is presented as well.

Chapter 4 Results and Discussion

This chapter presents and discusses the results acquired by the proposed methodology described in Chapter 4. Section 4.1 compares and evaluates the performance of models using different hyper-parameter values to find the optimal combination of five hyper-parameters. In Section 4.2, the quantitative and qualitative results of the training Dataset 1 and training Dataset 2 are presented and evaluated. Furthermore, the proposed model is compared with other representative deep learning models in Section 4.3.

4.1 Hyper-parameters Optimization

Hyper-parameter refers to parameters that cannot be optimized or learned from the training process, opposite to model parameters like weight and bias, thus the word "hyper-" represents higher-level properties of the model. Theses hyper-parameters are required to be fixed before the training process starts. In deep learning, hyper-parameters and their values vary from models to models, and hyper-parameters also depends on the training dataset. The selection of hyper-parameter value is always a challenging task in the model. In this study, optimization about five hyper-parameters that will influence the performance of the model was conducted in the implementation of the proposed neural network, while other hyper-parameters remained as a default based on former experiments and experiences. For consistency, each hyper-parameter was tested individually, while other hyper-parameters are constant. All the test works used the training and validation Dataset 2, thus the only variate was the tested hyper-parameter. For each test, there was a total of 40 epochs in the training process.

4.1.1 Dataset Division

Generally, the allocation of validation dataset from the training dataset is around 10% to 20%. The ratio of validation data extracted from training data can influence the model performance, and inappropriate amount of training data might lead to overfitting. In order to find the optimal division of the two datasets, five ratios of training and validation data were tested using the same testing dataset, which is 30%, 25%, 20%, 15%, and 10%, respectively. Figure 4.1 illustrates the overall model performance when different ratios of validation dataset are used. The horizontal axis

represents the ratios of validation dataset separated from the training dataset; and the vertical is the calculated value of metrics (mAP, mAR, and F1-score). According to Figure 4.1, both F1-score and mAP reaches the highest value at the model using 15% of validation data, with a steeply drop when the ratios of validation dataset increase. The mAR values are relatively close when using different ratios of validation dataset. Except for the ratio of 10%, the left four ratios have the same mAR value. With regard to the overfitting issue, the differences of overall loss between the training dataset and validation dataset during the training process are presented in Figure 4.2. Smaller loss difference indicates the similarity of the prediction between the training data and validation data. Therefore, according to Figure 4.2, the model using 15% of validation data has the lowest loss difference, which denotes that the overfitting is relatively avoided when 15% of validation data is separated from training data. As a result, the ratio of 15% validation data is the optimal dataset division ratio used for the proposed network.

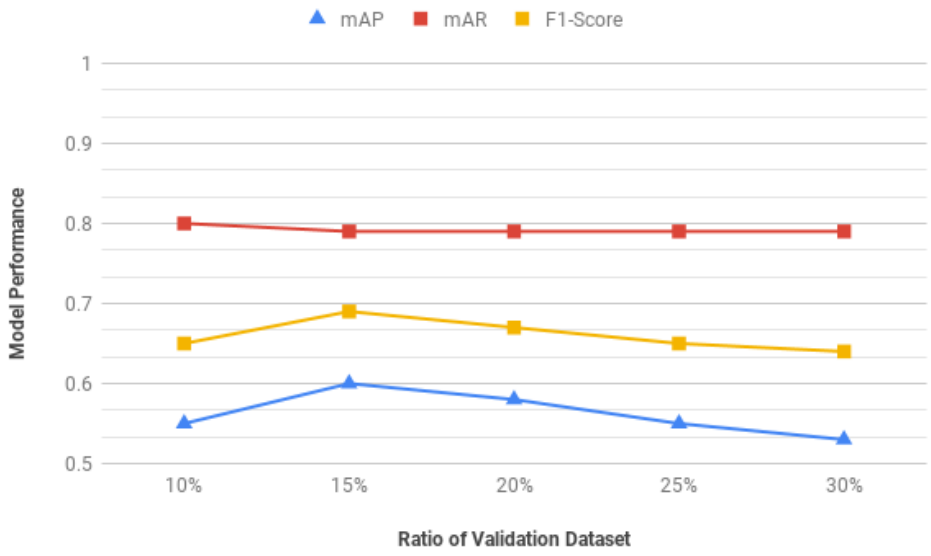


Figure 4.1 Overall model performance for different ratios of validation dataset

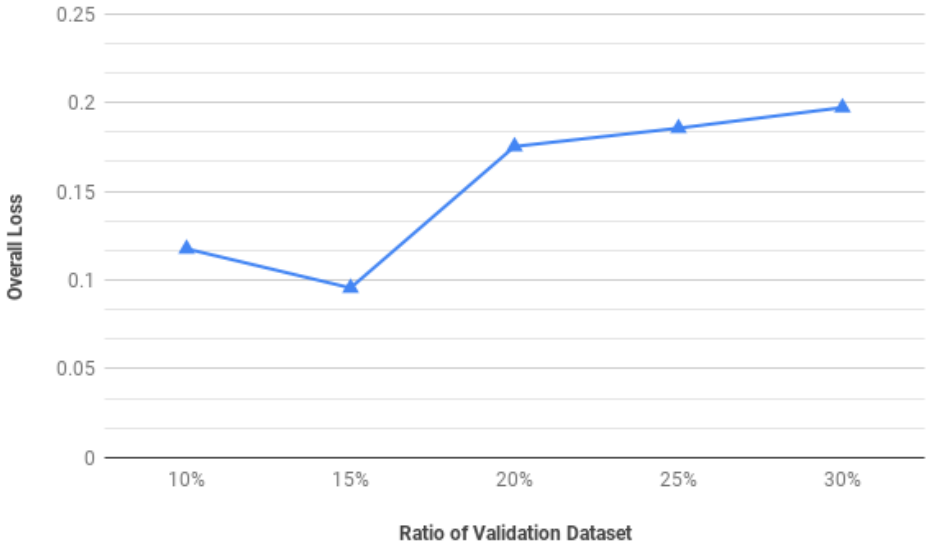


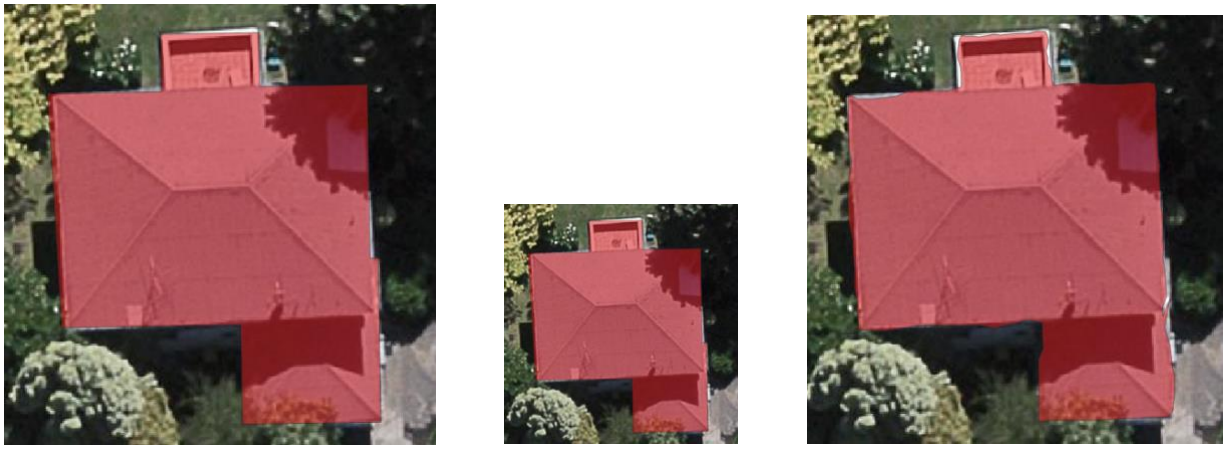
Figure 4.2 Difference of Loss between training dataset and validation dataset

4.1.2 Mini Mask

In order to reduce memory load, the mask can be rescaled to a size smaller than the image size. Although the downscaled process will decrease the resolution of the extracted mask; as using high-resolution images, the effect on the performance can be insignificant. Figure 4.3 (a) presents the original image and mask, with the downscaled image and mask in the middle (Figure 4.3 (b)). The re-projected image and mask are shown in Figure 4.3 (c), where the pixel loss is identified at the edge of the mask.

Thus, in order to find whether the resized mask will influence the model performance significantly, the mask size is set to unchanged, (128, 128), (256, 256), which represents the height and width in pixels. 'Unchanged' means that all the segmented masks remain as original during the training process, and the size of the mask depends on the size of building in the input image. (128, 128) and (256, 256) means that all the segmented masks are re-scaled to a fixed square size with height and width are both 128 or 256 in pixels. Three metrics, mAP, mAR and F1-score are used to represent the model performance using different mask sizes. As shown in Figure 4.4, the model using a mini mask with the size of (128, 128) outperforms other two models on all the three metrics, which means that by rescaling mask actually can predict the mask more accurate. The

model without rescaling mask has a better performance than the model using a mini mask with the size of (256, 256); in other words, the model using a mini mask with the size of (256, 256) has the lowest accuracy.



(a) (b) (c)

Figure 4.3 (a) Original image and mask, (b) downscaled image and mask (c) re-projected image and mask

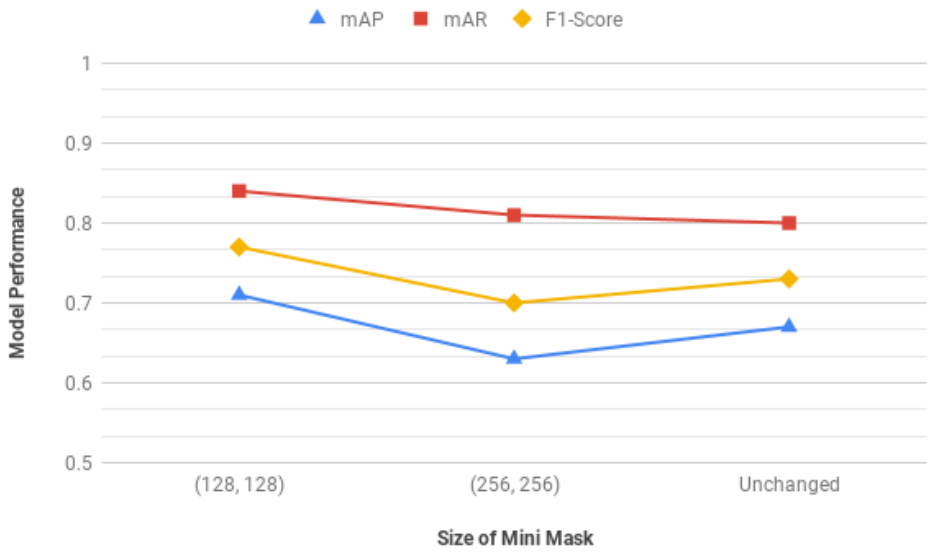


Figure 4.4 Overall model performance for different size of mini masks

Why rescaling mask increases the accuracy is complicated. However, one reason might relate to the size of buildings in the input image. Since most of the buildings in the input image are detached houses having a size of 100*100 in pixels, the upscaling of the mini mask gives more details in the boundary of the segmented objects. Looking into the performance of each model considering AP at different IoU threshold as shown in Figure 4.5, the model using the mini mask with the size of (128, 128) has the highest AP over different IoU thresholds; while the differences among AP over different IoU increases with the increase of IoU threshold value. Comparing the model using a mini mask of (128, 128) and the model without a mini mask, the AP at IoU of 0.5 is relatively similar, thus for detection task, both models perform almost the same. The accuracy gap between the two models increases with rising IoU threshold implies that the employment of the mini mask influences the result of the segmentation task. Therefore, considering the accuracy, the most appropriate model in terms of the mini mask use is the model by using a mini mask with the size of (128, 128).

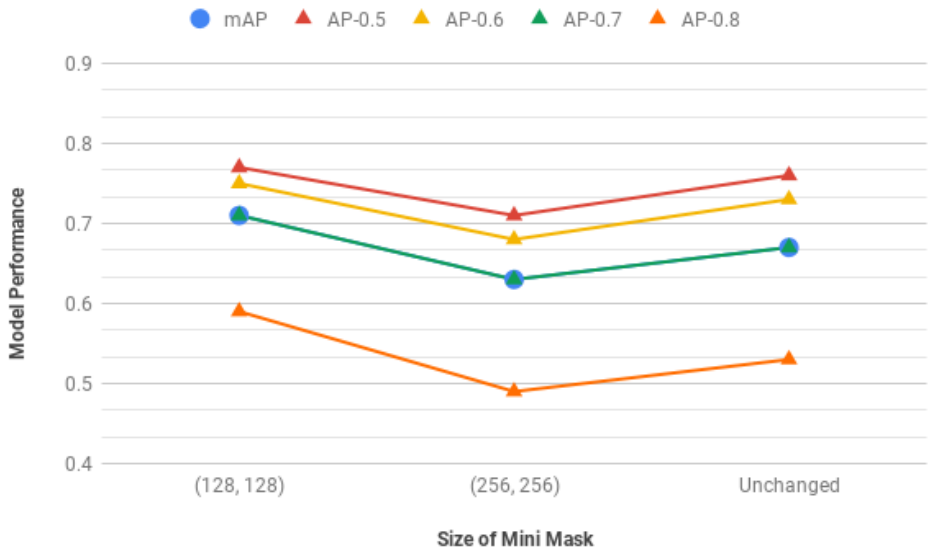


Figure 4.5 Model performance for different size of mini masks

4.1.3 Learning Rate

Since the backpropagation process updates the weight by using the gradient multiplied by a learning rate, the learning rate controls the amount of change to the model for each step of the

training process. If the learning rate is set to be large, although it can make the model to learn faster, too large learning rate will overlook the optimum model and results in divergence. On the contrary, small learning rate may allow the model to reach the local or global minimum of loss with tiny steps, but it will consume a large amount of time (training iterations) and fails to converge. In order to find the optimal learning rate used for the proposed model, four learning rates are selected and tested, which is 0.005, 0.001, 0.0005 and 0.0001, respectively. According to Figure 4.6, the obvious difference of mAP indicates the influence of learning rate on the output precision; while learning rate 0.005 and 0.001 have a similar mAP result. The difference between 0.005 and 0.001 becomes distinct at the mAR result, which leads to the variance of F1-score. In conclusion, the model reaches the best performance with the learning rate of 0.001, and the learning rate of 0.001 is selected for the final training process.

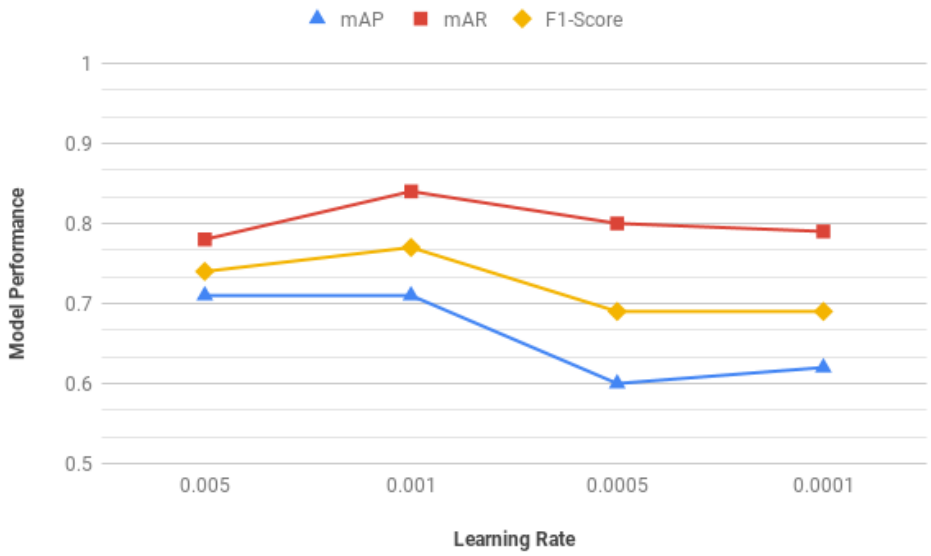


Figure 4.6 Overall model performance for different learning rates

4.1.4 Backbone

There are two ResNet backbones available in this study: the ResNet-50 and ResNet- 101. The main difference between these two backbones is the number of layers that ResNet-50 has 50 layers while ResNet-101 has 101 layers with considerably increased depth. The detailed architectures of ResNet-50 (ResNet-50, n.d.) and ResNet-101 (ResNet-101, n.d.) can be found from references.

Basically, ResNet-50 has one convolutional layer at the beginning with batch normalization and ReLU activation, 16 ResNet blocks, and one dense layer in the end. Each block has three identity blocks. For each identity block, there is one convolutional layer, one batch normalization layer and one ReLU activation; while the only convolutional layer is identified as a layer. Compared to ResNet-50, ResNet-101 has 17 more ResNet blocks, which is 33 ResNet blocks in total. Although the increased depth involves more parameters and a more complex network, it does not mean a more accurate model in terms of different datasets and experiments. Therefore, the dataset was tested by using ResNet50 and ResNet 101 architectures separately. According to Figure 4.7, a model using ResNet-101 backbone has a better performance on all the three metrics. Additionally, the training time of the model using ResNet-101 is only nine minutes longer than the model using ResNet-50. Therefore, considering both accuracy and efficiency, ResNet-101 is selected as the backbone of the proposed model. In this study, the variation inside the building class requires a deeper neural network to learn more features and predict a more accurate result.

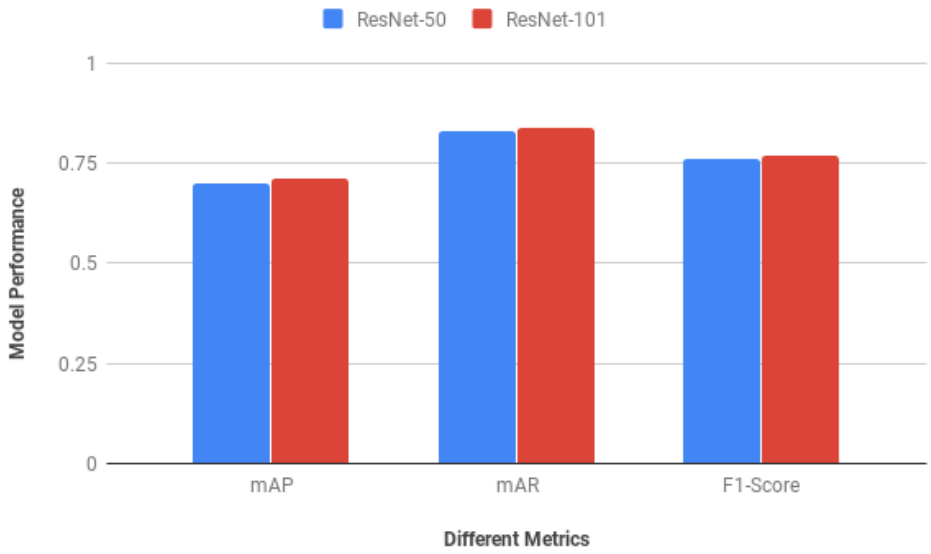


Figure 4.7 Overall model performance between ResNet-50 and ResNet-101

4.1.5 Model Initialization

Compared to training a model from scratch, a more advanced method is to employ an existing pre-trained model learned from other datasets as a starting point of the proposed model. The pre-

trained model, in this case, is the set of feature extractors to form the architecture of the model. Instead of initializing the model with random weights, using a pre-trained model can lead to faster convergence and increase the detection accuracy. In this study, two available pre-trained models are used and compared with the model with random initialization. One pre-trained model is the ResNet 50 that was trained on ImageNet dataset downloaded from Keras Application (Keras Documentation, n.d.). The ImageNet dataset contains more than 1.4 million images consisting of 20,000 classes (e.g., human, car, fish and building) (ImageNet, n.d.). Another pre-trained model is the ResNet 101 that was trained on COCO dataset. COCO dataset contains more than 330,000 images consisting of around 200 classes, specially designed for object detection and segmentation (COCO, n.d.). As shown in Figure 4.8, the model using pre-trained weights from COCO dataset achieves the best performance in terms of the three metrics, while the accuracy of the model using random initialization is significantly lower than the other two models. Therefore, it is essential to use a pre-trained model rather than randomly picking up weights. And in this case, the model using pre-trained weights from COCO dataset is selected as the base model for the following training process.

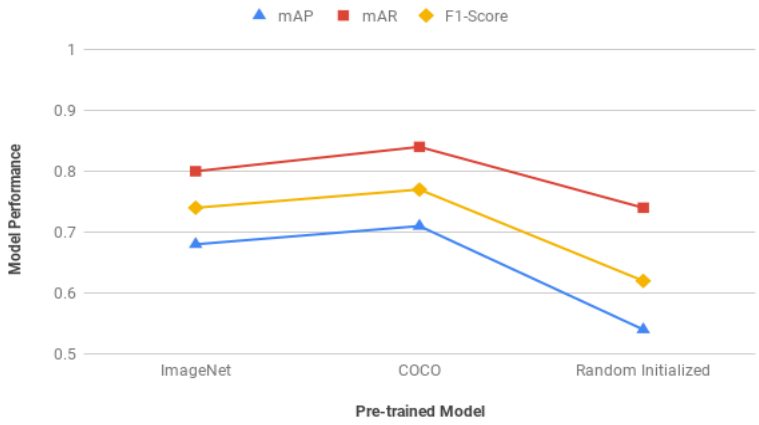


Figure 4.8 Overall model performance for different model initialization

4.1.6 Summary of Hyper-parameters

Generally, five hyper-parameters are tested and discussed above. Table 4.1 summarizes the selection of the five hyper-parameters and the optimal value (highlighted in bold) for each hyper-

parameter. In conclusion, the optimal hyper-parameters are: the 15% ratio of validation dataset, the mini mask with the size of (128, 128), the learning rate at 0.001, the ResNet-101 backbone and the model initialed with pre-trained weight from COCO dataset.

Table 4.1 Summary of hyper-parameter

Hyper-parameters	Value	Performance
Ratio of validation dataset	10%	Medium
	15%	**Best**
	20%	Medium
	25%	Medium
	30%	Worst
Mini Mask	(128, 128)	**Best**
	(256, 256)	Worst
	Unchanged	Medium
Learning Rate	0.005	Medium
	0.001	**Best**
	0.0005	Worst
	0.0001	Medium
Backbone	ResNet-50	Worst
	ResNet-101	**Best**
Model Initialization	ImageNet	Medium
	COCO	**Best**
	Random Initialized	Worst

4.2 Analysis of Building Detection Result

In the experiment, two training datasets with different spatial resolutions were learned using Mask R-CNN, and two models trained separately with the training datasets were applied on the testing dataset. In order to find the optimal building detector and avoid overfitting, each model was

trained from 10 epochs to 200 epochs. For each trained model, four model weights were selected from different epoch range (i.e., 0-10, 11-40, 41-120 and 121-200). Figure 4.9 presents the loss trend of training dataset and validation dataset within 200 epochs. The lower loss represents that the error in building detection is decreasing. The difference between the training dataset and validation dataset shows that the trained model is overfitting after 200 epochs. Therefore, 200 epochs for training is enough. Consequently, eight models with different spatial resolutions and training epochs are tested and compared in the following section, in terms of qualitative and quantitative assessment.

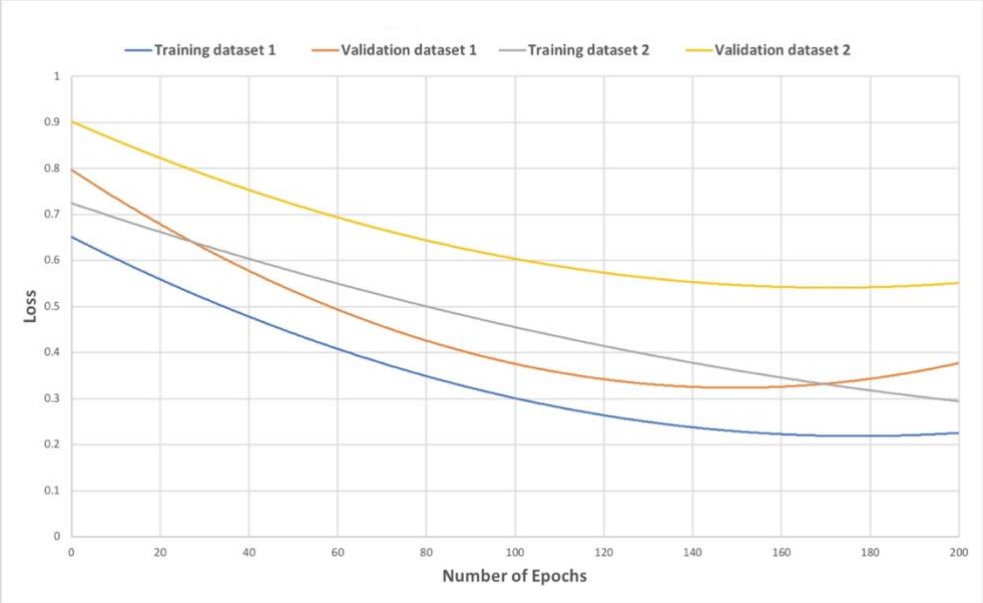


Figure 4.9 The loss of training dataset and validation dataset

4.2.1 Qualitative Result Comparison

4.2.1.1 Region-level Comparison

Figure 4.10 shows the detection result (bounding box) of the model using training Dataset 1 from Epoch 10 to Epoch 200, in comparison with the original image and ground truth. The image shows the detection result at the region level; thus, each image represents a square area of 576m * 576m. Five region-level samples are presented, including the top left, top right, center, bottom left and bottom right of the study areas. For a residential area such as top left, central and bottom left,

all models are capable of dense building detection compared to the ground truth. However, for large building cluster in the center of the top left image (the bright white building), both models at 10 epochs and 200 epochs failed to identify this building; while models at 40 epochs and 120 epochs recognize part of the building. In a rural region such as top right and bottom right, all the models can successfully detect the detached houses, even the smallest house in the right center of the top right image. Conclusively, all the models are able to detect individual houses, and the models at 40 epochs and 120 epochs can identify large buildings as well. Precisely, Figure 4.11 presents the segmentation result (mask) of the four models at the same five regions. In the residential region, all the models are prone to segment buildings correctly, although only the model at 40 epochs can segment the two large buildings in the top left image. In the non-residential region, all the models are capable of individual building segmentation with a clear boundary between each building. In conclusion, all the models can detect and segment the majority of buildings in both residential and rural regions, while improvements are needed for large building clusters' detection.

Figure 4.12 presents the detection result of the models using training Dataset 2 from Epoch 10 to Epoch 200, in comparison with the aerial image and the ground truth at the same region level. As shown in Figure 4.12, all the models are capable of building detection, especially performing better in large building detection compared to training Dataset 1. The large buildings in the top left image are successive to be detected in all four models, although the parking lot between the two large buildings (highlighted in red circle) is mistakenly identified as buildings by the model at 40 epochs. In the non-residential region, all the models can detect individual houses, which is similar to training Dataset 1. When looking into details, Figure 4.13 presents the segmentation result of models. As shown in Figure 4.13, all the detached buildings are successfully segmented. However, the models segment the large buildings in the center of the top left and bottom left images into many small pieces. Although the number of predictions is increased, the number of FP is rising as well. As a result, the models using training Dataset 2 is skilled at detection and segmentation on both small and large buildings, while the fragmental segmentation of large buildings requires merging to increase the precision.

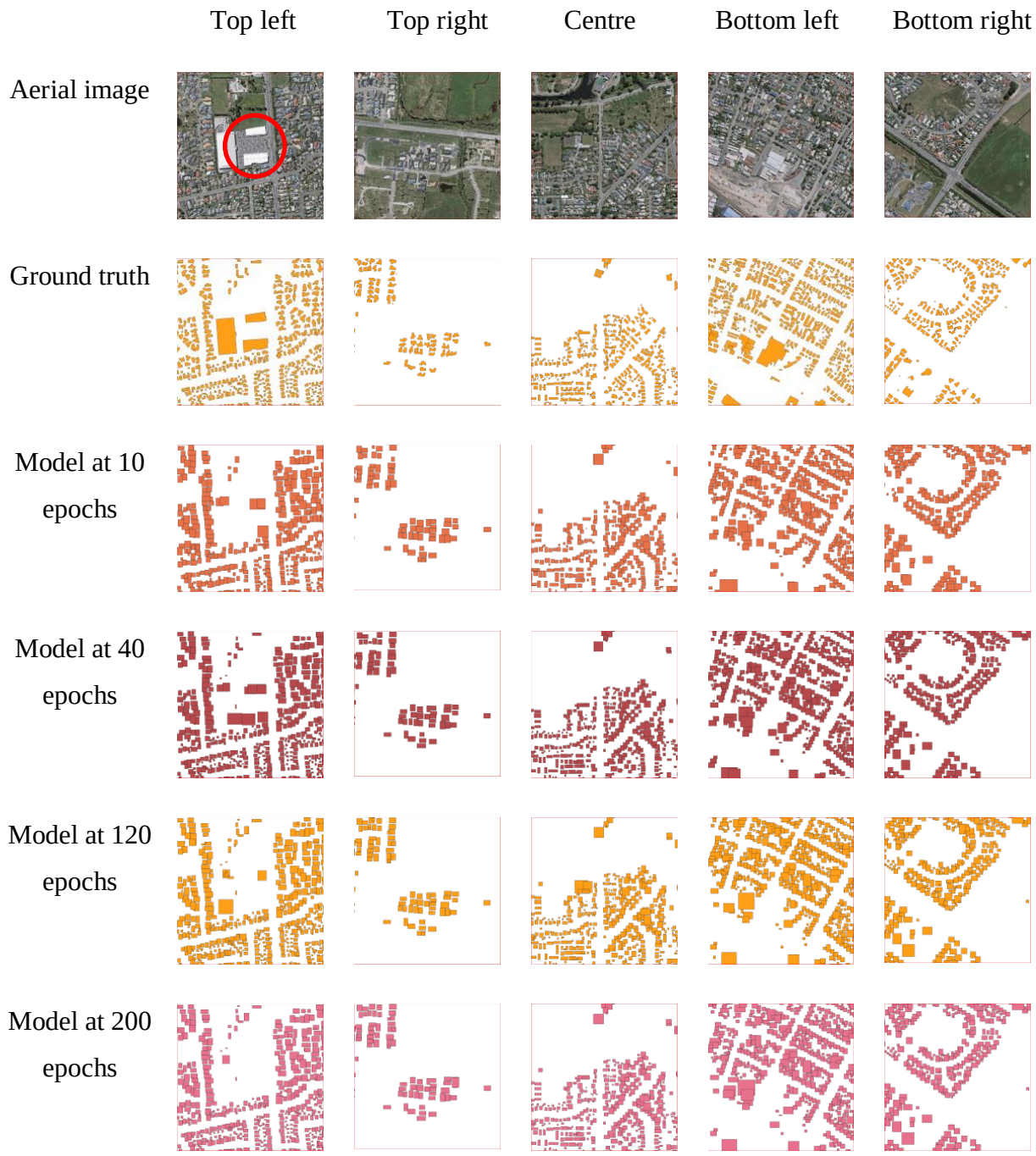


Figure 4.10 Detection result by models using training Dataset 1 at region level

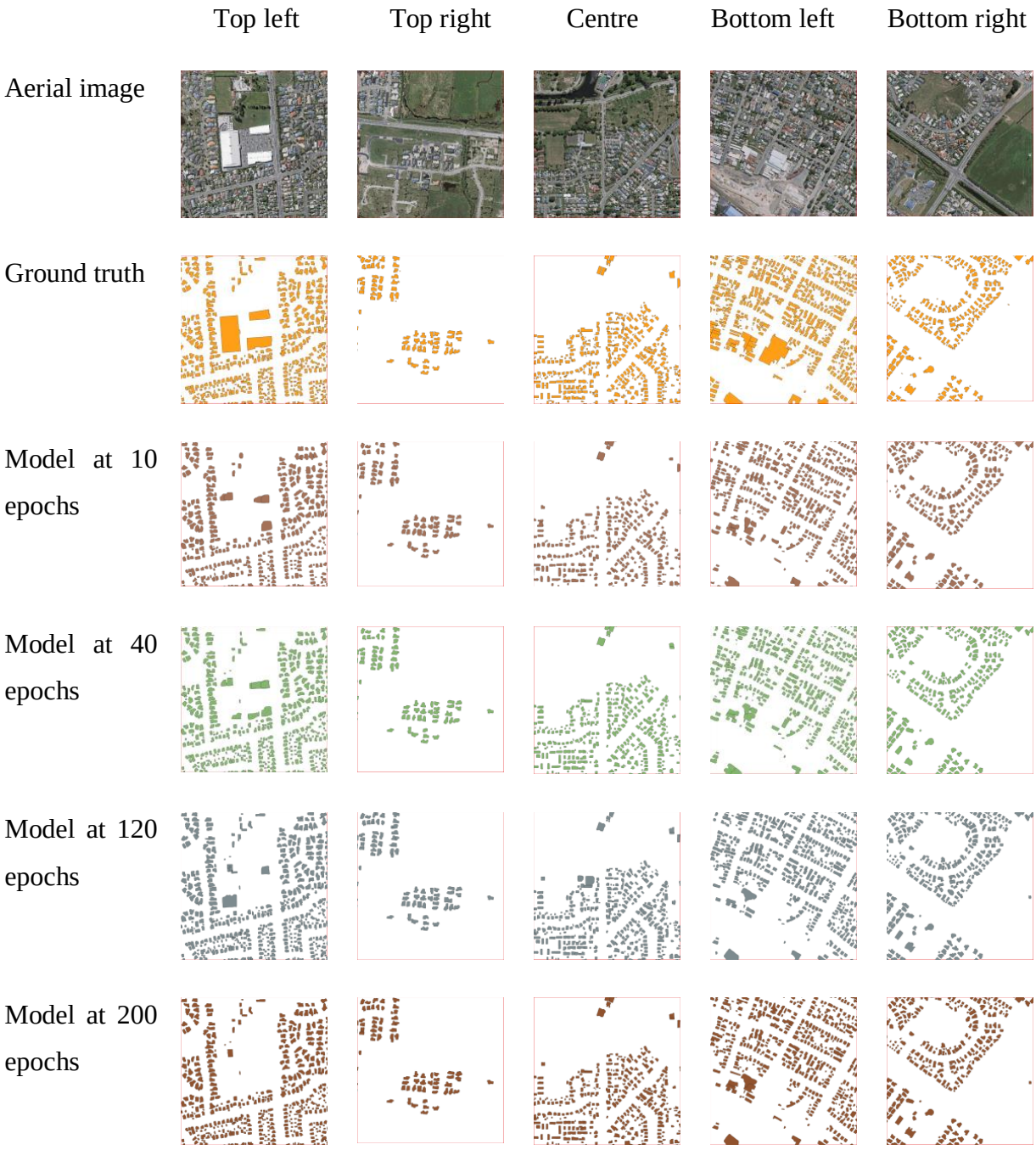


Figure 4.11 Segmentation result by models using training Dataset 1 at region level

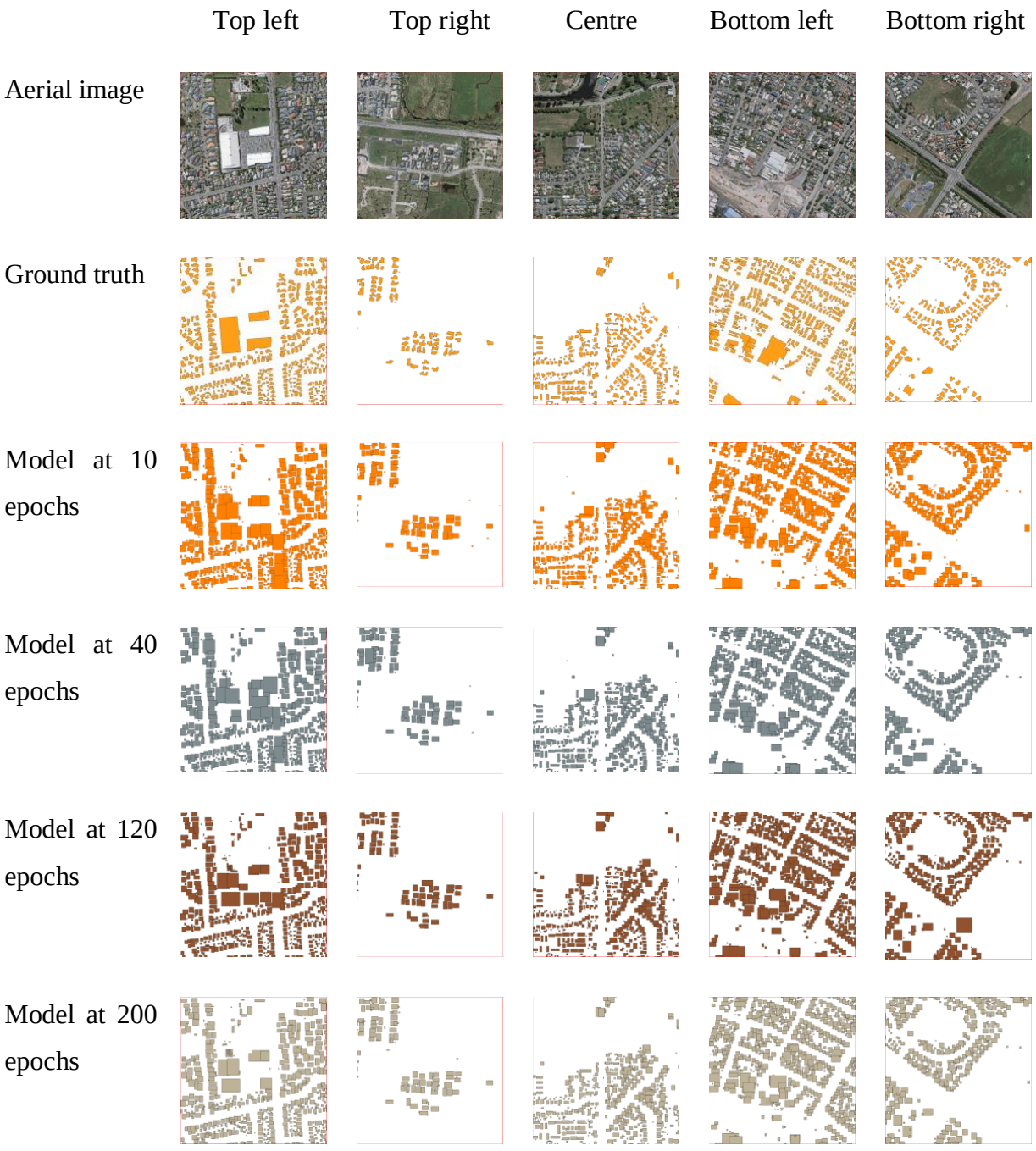


Figure 4.12 Detection result by models using training Dataset 2 at region level

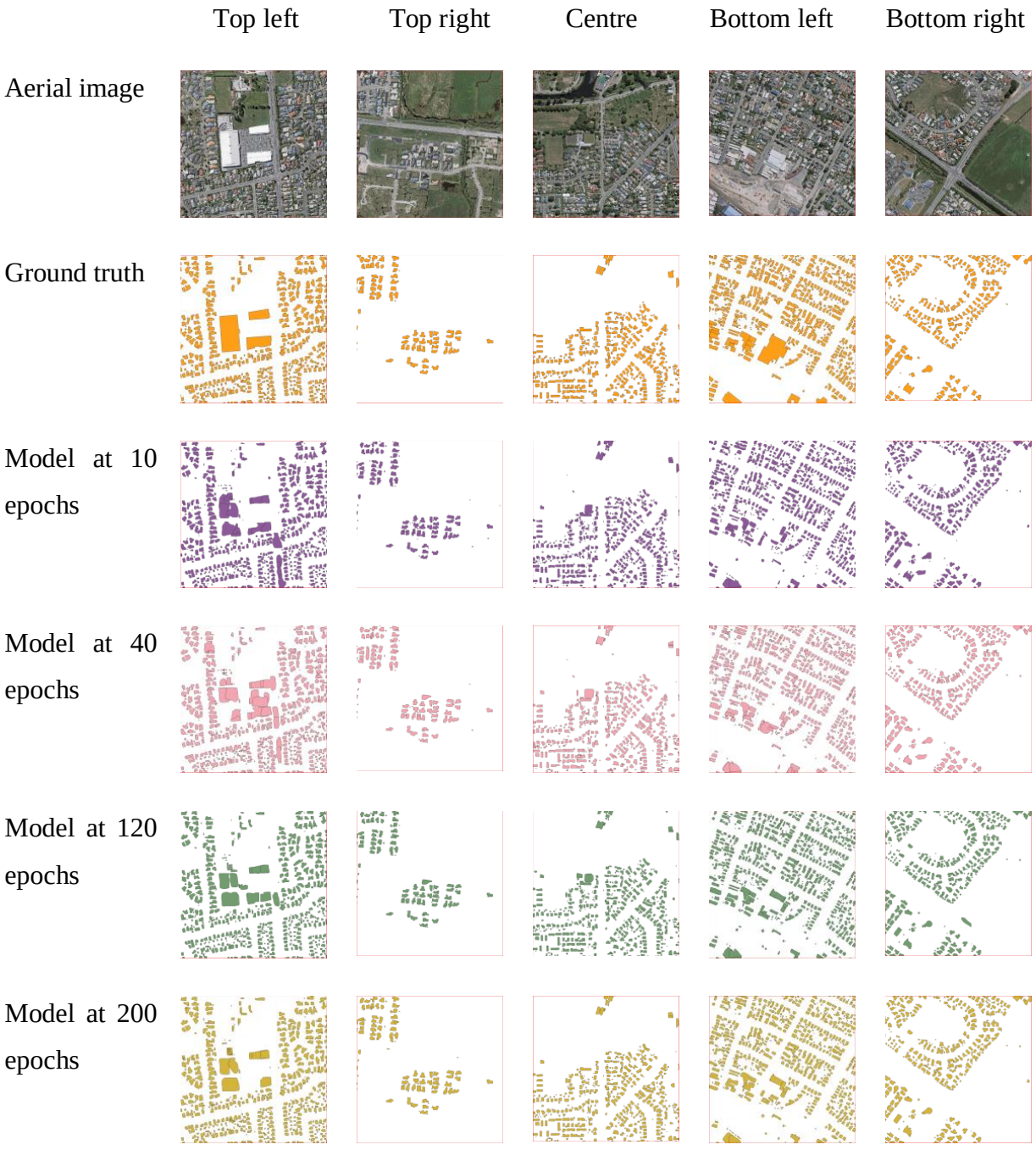


Figure 4.13 Segmentation result by models using training Dataset 2 at region level

4.2.1.2 Single-house-level Comparison

In response to the discussion of building detection challenges as introduced in Section 1.1, the detection result of typical samples at single-house level is presented from Figures 4.14 to 4.17.

As shown in Figures 4.14 and 4.15, five representative samples of the segmentation result by the proposed models at 10 epochs to 200 epochs, in comparison to the aerial image and ground truth. Generally, all the models can extract the main section of buildings in different directions, shapes and colours. In addition, all the models can segment the mask closing to the actual shapes of individual buildings, except that the models using training Dataset 1 at 10 and 40 epochs failed to extract corners of the building a and b. Furthermore, in response to the shadow and occlusion challenges, Figures 4.16 and 4.17 present two occlusion samples and two shadow samples, and the corresponding segmentation result. In Figure 4.16, the models using training Dataset 1 at 120 epochs and 200 epochs outperform the other models in the segmentation of buildings occluded partially by trees in the image a. The building part that is occluded by trees can be successfully extracted from the image by the two models, while all the models using training Dataset 2 and the other two models using training Dataset 1. Moreover, all the models can segment the building section covered by tree shadow in the image d, while the ground truth fails to identify the covered part. In conclusion, the proposed building detection method can solve the in-class diversity, shadow and occlusion challenges, superior to the conventional building detection method, with a clear and precise boundary segmentation.

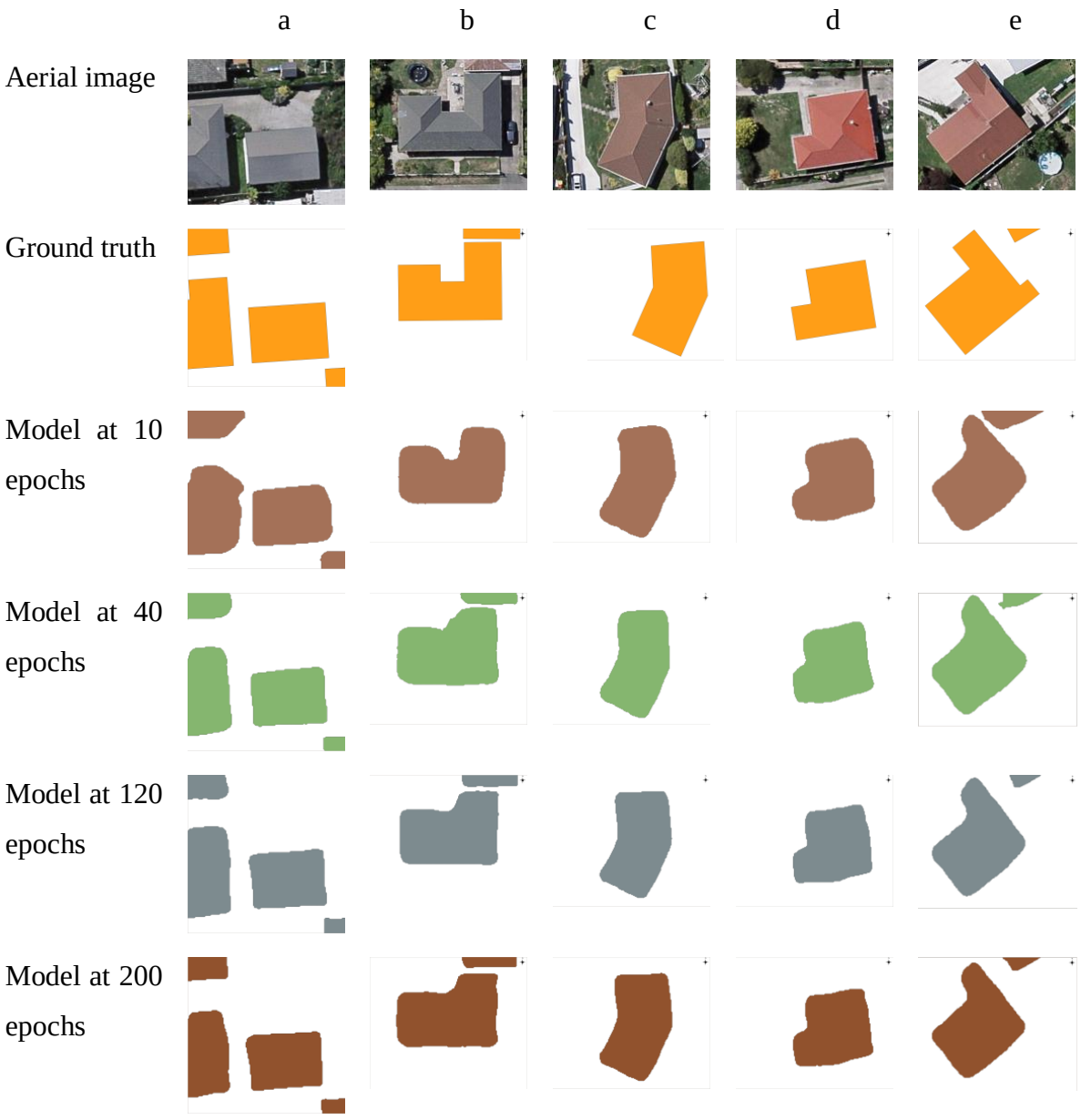


Figure 4.14 Representative samples of segmentation result by models using training Dataset 1 at single-house level

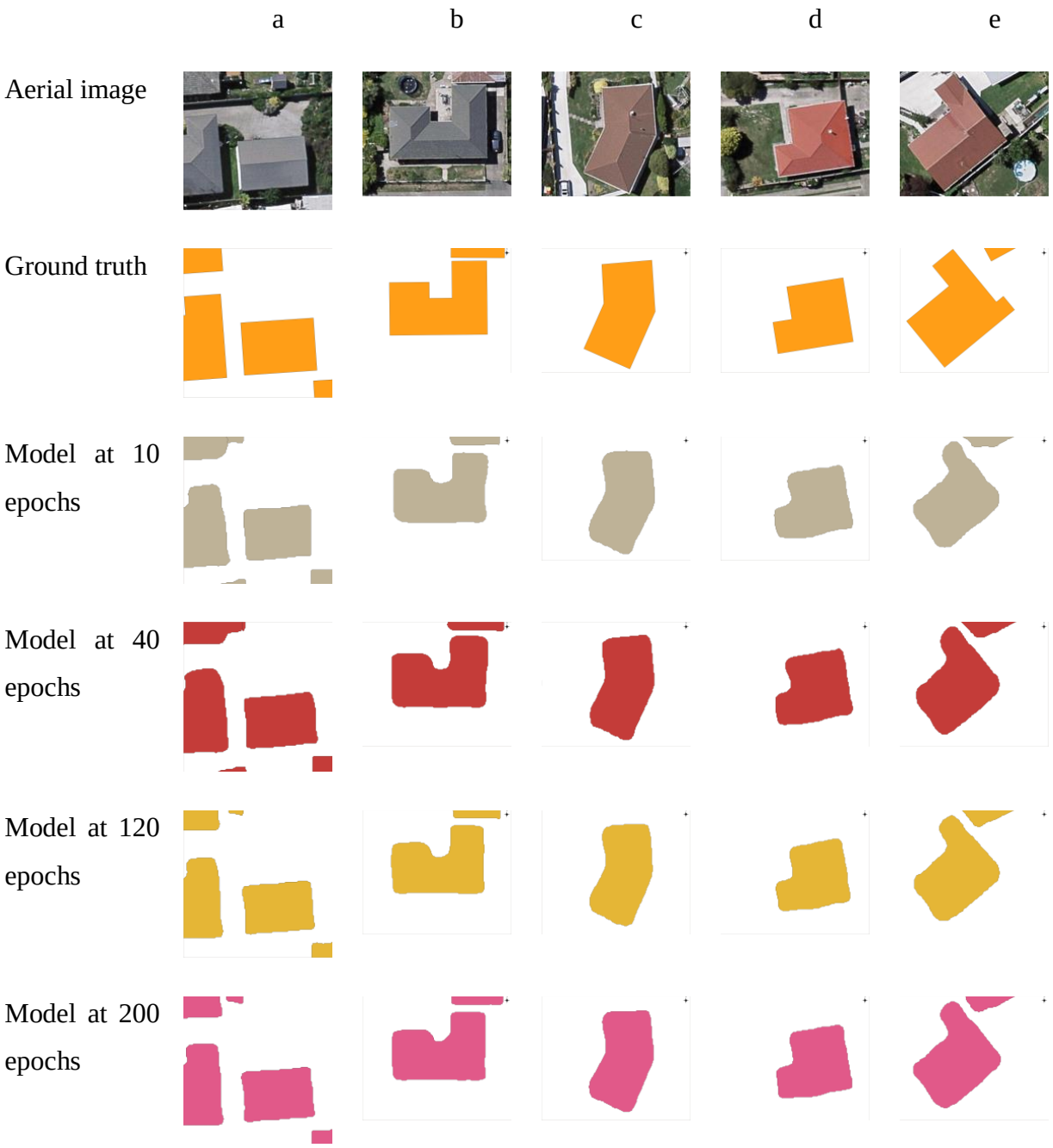


Figure 4.15 Representative samples of segmentation result by models using training Dataset 2 at single-house level

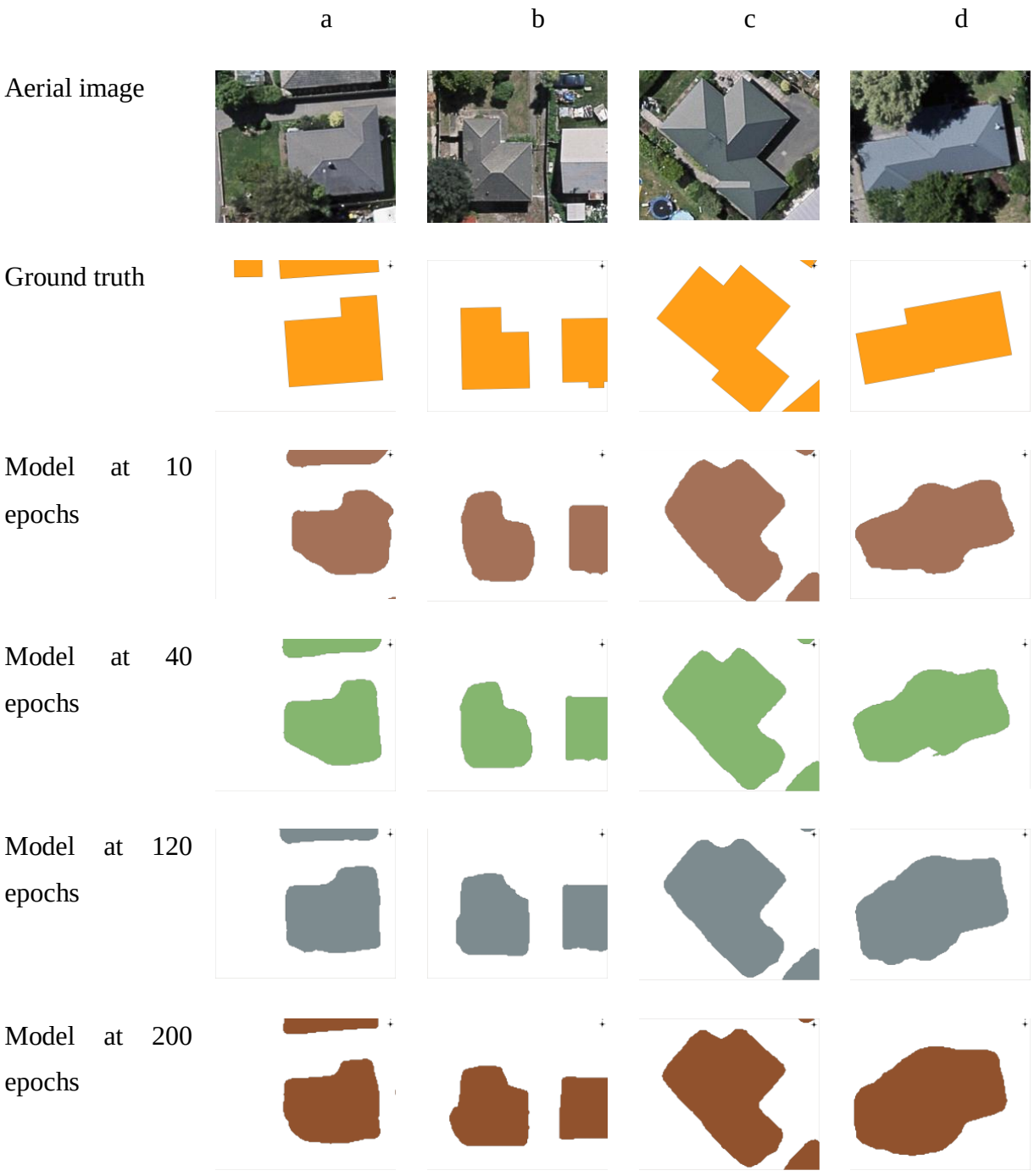


Figure 4.16 Representative tree occlusion and shadow samples of segmentation result by models using training Dataset 1 at single-house level

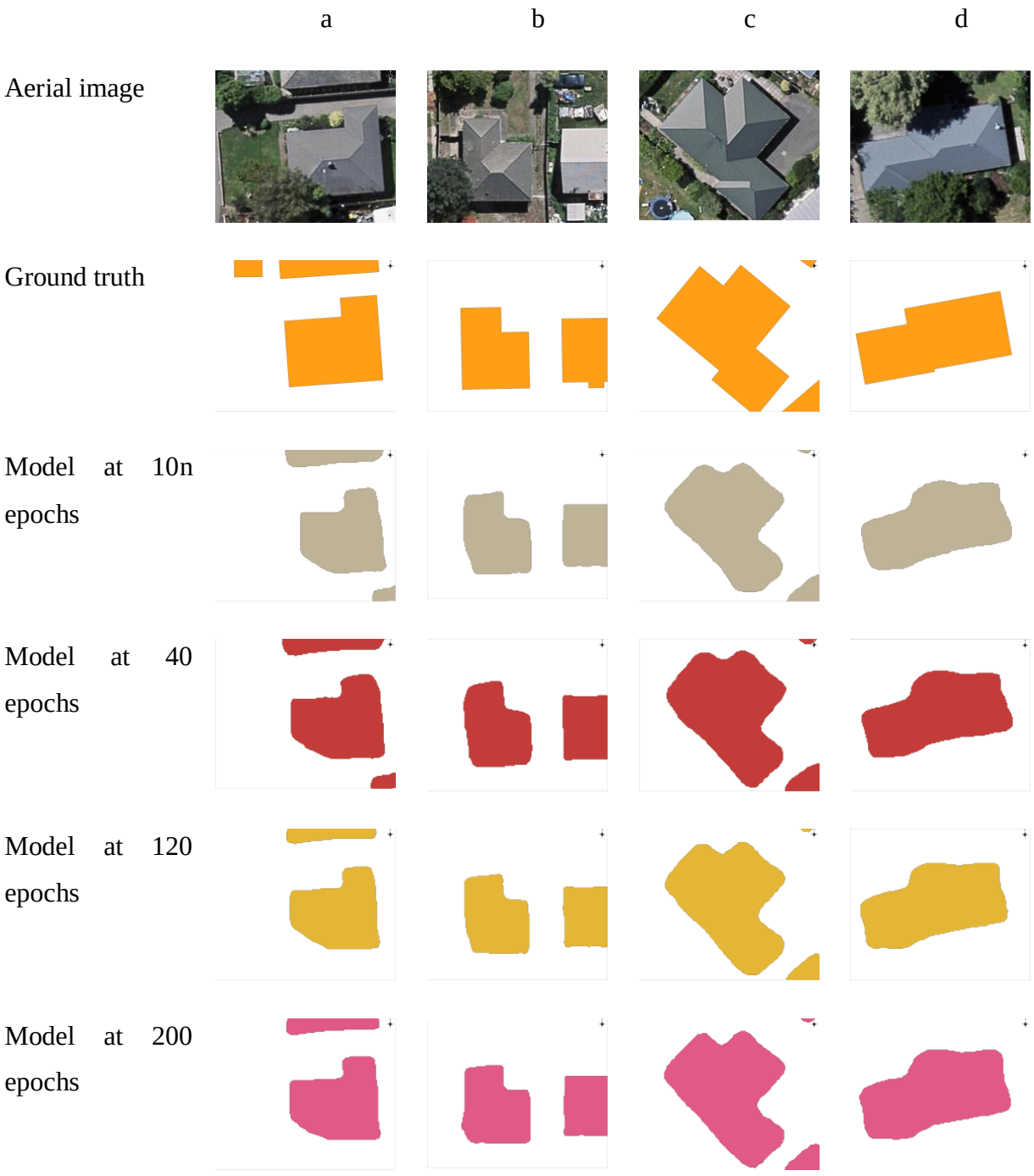


Figure 4.17 Representative tree occlusion and shadow samples of segmentation result by models using training Dataset 2 at single-house level

4.2.2 Quantitative Result Comparison

In order to quantitatively evaluate the performance of different models, the AP at different IoU, mAP, AR at different IoU, mAR, and corresponding F1-score is calculated as shown in Table 4.2 and Table 4.3 for training Dataset 1 and training Dataset 2, respectively. The highest values for each evaluation metric are highlighted in bold.

Table 4.2 Accuracy Assessment of model using training Dataset 1

Epochs	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	mAP	F1-score
10	0.892	0.853	0.79	0.611	0.786	0.67
40	0.895	0.865	0.814	0.677	0.813	0.774
120	0.92	0.884	0.831	0.685	0.83	**0.805**
200	**0.942**	**0.906**	**0.855**	**0.713**	**0.854**	0.804
	$AR_{0.5}$	$AR_{0.6}$	$AR_{0.7}$	$AR_{0.8}$	mAR	
10	0.661	0.638	0.586	0.453	0.583	
40	0.813	0.785	0.74	0.615	0.738	
120	**0.866**	**0.833**	**0.782**	**0.645**	**0.781**	
200	0.838	0.806	0.76	0.634	0.759	

Table 4.3 Accuracy Assessment of model using training Dataset 2

Epochs	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	mAP	F1-score
10	0.742	0.695	0.623	0.454	0.628	0.698
40	0.817	0.778	0.717	0.566	0.72	0.767
120	**0.852**	0.809	0.736	0.553	0.738	0.768
200	0.848	**0.816**	**0.762**	**0.627**	**0.763**	**0.801**
	$AR_{0.5}$	$AR_{0.6}$	$AR_{0.7}$	$AR_{0.8}$	mAR	
10	0.927	0.868	0.779	0.566	0.785	
40	0.932	0.888	0.818	0.646	0.821	
120	0.924	0.878	0.799	0.6	0.8	
200	**0.937**	**0.902**	**0.842**	**0.693**	**0.844**	

Generally, both training datasets have the highest AP and AR when the IoU is at 0.5, which means that both models perform well in the task of building detection. However, when it comes to building segmentation task where the IoU threshold is increased, the performance of both models becomes less positive. The highest overall F1-score for training Dataset 1 and training Dataset 2 is 0.805 at Epoch 120 and 0.801 at Epoch 200, respectively. Since higher F1-score represents a better balancing model, the value of F1-score in both models shows a good balance.

As shown in Table 4.2, the model using training Dataset 1 reaches the highest AP (0.942) when the epoch is at 200 and the IoU is at 0.5. The model reaches the highest AR (0.866) when the epoch is at 120 and the IoU is at 0.5. In general, the model at Epoch 200 outperforms the previous model in terms of AP and mAP (0.854), which means that the model at Epoch 200 is more precise when predicting building targets. However, considering AR and mAR, the model at Epoch 120 has the best performance where mAR equals to 0.781. The better performance of the model at Epoch 120 on AR indicates that the model is easier to assign an object to the class of building, although the precision is not good. Additionally, in terms of F1-score, the model at Epoch 120 has a relatively higher F1-score (0.805) than the model at Epoch 200 (0.804) demonstrating that the combination of precision and recall is more balanced after 120 epochs of training. For each IoU threshold and epoch, the value of precision is always higher than the value of recall, which indicates that, in general, the model learned from training Dataset 1 can misclassify some buildings into the class of background.

As for training Dataset 2, the precision, recall and F1-score result is shown in Table 4.3. According to Table 4.3, the model reaches the highest AP (0.852) when the model is at 120 epochs and the IoU is at 0.5. The model reaches the highest AR (0.937) when the model is at 200 epochs and the IoU is at 0.5. With regards to precision, the model at 120 epochs has the highest AP at IoU of 0.5, while with the increase of IoU threshold, the model at 200 epochs always has the highest AP. Therefore, the mAP (0.763) is the highest when the model is trained for 200 epochs. When it comes to recall, the model trained for 200 epochs outperforms other models over all the IoU thresholds. Both the F1-score (0.801) and mAR (0.844) has the highest at 200 epochs as well. Different from the model learned from training Dataset 1, in training Dataset 2, the value of recall

is always higher than the value of precision, which implies that the model trained using Dataset 2 tends to classify more objects to the class of buildings.

In order to evaluate the detection performance of different models in detail, the confusion matrix for each model with IoU at 0.5 is extracted as seen in Table 4.4 to Table 4.11. These tables obviously present the performance of the detection models in terms of the number of correct classification (TP) and the number of misclassification (FP and FN). In addition, the highest values for each evaluation metric are highlighted in bold. The total number of predictions (at the lower left) changes when different models are employed, while the total number of actual (at the upper right) is a constant (29958) that represents the number of ground truth within the testing area. By comparing these tables, it is obvious that the model at Epoch 200 predicts more buildings than other models (see Table 4.7 and Table 4.11). For training Dataset 1, FP is lower than FN, indicating that more buildings are misclassified to the background. For training Dataset 2, on the contrary, FN is lower than FP, which implies that the model fails to detect certain buildings. Additionally, the training Dataset 2 has a relatively higher predictive value than the training Dataset 1, which denotes that the model using the training Dataset 2 can detect more buildings than training Dataset 1, while the detection precision of training Dataset 2 cannot be guaranteed. In regard to the model at different epochs, for training Dataset 1, TP has the highest value, and FN has the lowest value at 120 epochs, which means that the model trained for 120 epochs can detect more buildings in the testing area. At 200 epochs, FP is the lowest among all the models, indicating that the model trained for 200 epochs is more precise in building detection task. The phenomenon also verifies the higher AR for 120 epochs, and higher AP for 200 epochs in Table 4.4. For training Dataset 2, the model has the highest TP and the lowest FN at Epoch 200, showing that the model trained for 200 epochs can detect more buildings than other models. The model trained for 120 epochs has the lowest FP value, which indicates that the model makes fewer mistakes in differentiating the buildings and background. The phenomenon also confirms the higher AR for 120 epochs and higher AP for 200 epochs.

Table 4.4 Confusion matrix for training Dataset 1 at 10 epochs

		Prediction		
		Positive	Negative	Total
Actual	Positive	27757	2201	29958
	Negative	9674		
	Total	**37431**		

Table 4.5 Confusion matrix for training Dataset 1 at 40 epochs

		Prediction		
		Positive	Negative	Total
Actual	Positive	27916	2042	29958
	Negative	6262		
	Total	34178		

Table 4.6 Confusion matrix for training Dataset 1 at 120 epochs

		Prediction		
		Positive	Negative	Total
Actual	Positive	27680	2278	29958
	Negative	**4805**		
	Total	32485		

Table 4.7 Confusion matrix for training Dataset 1 at 200 epochs

		Prediction		
		Positive	Negative	Total
Actual	Positive	**28068**	**1890**	29958
	Negative	5046		
	Total	33114		

Table 4.8 Confusion matrix for training Dataset 2 at 10 epochs

		Prediction		
		Positive	Negative	Total
Actual	Positive	19815	10143	29958
	Negative	2411		
	Total	22226		

Table 4.9 Confusion matrix for training Dataset 2 at 40 epochs

		Prediction		
		Positive	Negative	Total
Actual	Positive	24353	5605	29958
	Negative	2861		
	Total	27214		

Table 4.10 Confusion matrix for training Dataset 2 at 120 epochs

		Prediction		
		Positive	Negative	Total
Actual	Positive	**25919**	**4039**	29958
	Negative	2282		
	Total	**28201**		

Table 4.11 Confusion matrix for training Dataset 2 at 200 epochs

		Prediction		
		Positive	Negative	Total
Actual	Positive	25096	4862	29958
	Negative	**1546**		
	Total	26642		

After above comparing and discussing, the model using training Dataset 1 has the best precision performance at Epoch 120 and the best recall performance at Epoch 200, and the model using training Dataset 2 has the best precision performance at Epoch 200 and the best recall performance at Epoch 120. Table 4.12 summarizes the AP, AR and F1-score of training Datasets 1 and 2 with the best performance at IoU equaling to 0.5, where the highest values for each metric are highlighted in bold. According to 5.12, the model using training Dataset 1 at Epoch 200 has the highest precision (0.942), and the model using training Dataset 2 at Epoch 120 has the highest recall (0.937). Considering the combination of precision and recall, training Dataset 1 has the highest F1-score (0.891).

Table 4.12 Summary of optimal models

	Precision	Recall	F1-score
Training Dataset 1 at Epoch 120	0.92	0.866	**0.891**
Training Dataset 1 at Epoch 200	**0.942**	0.838	0.886
Training Dataset 2 at Epoch 120	0.852	0.924	0.887
Training Dataset 2 at Epoch 200	0.848	**0.937**	0.89

Additionally, the processing time of training models from Datasets 1 and 2 are presented in Figure 4.18. The overall processing time of the training model using training Dataset 1 is 256.2 mins, which is almost twice longer than the training model using training Dataset 2 (145.52 mins). In conclusion, the processing time increases significantly with the increase of spatial resolution. Considering both quantitative result and qualitative, the model using training Dataset 1 at Epoch 120 is selected as the optimal model for VHR imagery building detection task.

Figure 4.18 Training time of models

4.3 Comparison of Building Detection Methods

In order to further evaluate the performance of the proposed Mask R-CNN building detection method, a comparative study was constructed by implementing other state-of-the-art building detection methods (i.e., U-Net and baseline Mask R-CNN) on the VHR aerial imagery. Considering the better performance of training Dataset 1, as well as the optimal model as tested and discussed in Section 4.2, the proposed Mask R-CNN model using optimal hyper-parameters after 120 epochs training was performed and compared with other two models using the training Dataset 1.

As reviewed in Section 2.2.3, U-Net is a fully convolutional model that is used in the field of building detection (e.g., Wu et al., 2018, Xu et al., 2018). The detailed architecture of U-Net is shown in Figure 4.19. Each blue box represents a calculated feature map from the previous input image, and the number of channels for each feature map is presented on the top of the blue box. The gray arrow in the middle represents the skip connections that pass the feature map from down path to up path at different levels. In the encoder or down-path on the left side of the architecture, the convolutional blocks are followed by max pooling layer to encode the image into feature maps at multiple levels. In the decoder or up path on the right side of the architecture, the up-sampling

operation is followed by convolutions to expand the feature map dimensions to fit the size of feature maps from the left side. Therefore, the location information from the down-sampling path and the contextual information from the up-sampling path are combined to obtain the general information from both localization and context, and a precise segmentation result is produced using this architecture. Additionally, during the implementation of U-Net in the thesis, the optimal hyper-paramters in the U-Net model were selected and tuned. The optimizer is Adam with a learning rate of 0.0001, and the loss function is binary cross-entropy.

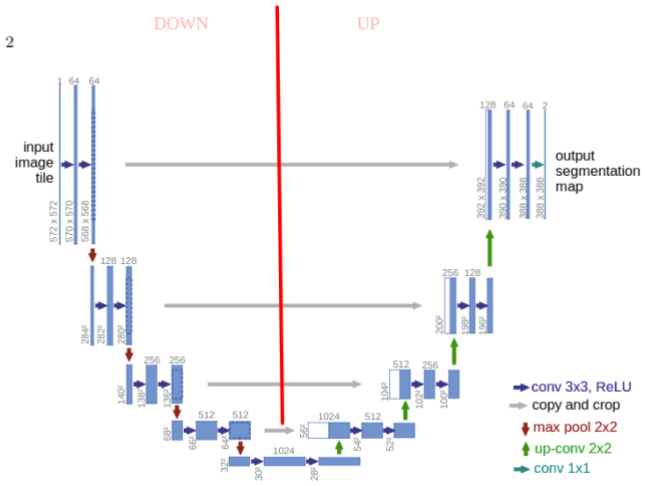


Figure 4.19 Architecture of U-Net

(Source: Ronneberger, Fischer & Brox, 2015)

Another model used for comparison is a baseline Mask R-CNN model acquired from the Mapping Challenge of CrowdAI (CrowdAI, n.d.). Figure 4.20 presents five representative samples of detection result using the three models, in comparison to the original image and ground truth. As shown in Figure 4.20, the majority of buildings are extracted from images by the three models. The proposed model can identify all the buildings in the images with a more rounded boundary; while U-Net performs better in the boundary segmentation. The baseline Mask R-CNN has worse performance on the building detection with more background are misclassified into buildings. Furthermore, although U-Net has a more precise boundary, it is sensitive to buildings located at

the image corner. For instance, the buildings (circled in red) at the bottom-left corner of image b is successfully detected and segmented by the proposed and baseline model, while U-Net only extracts a small portion of the building. However, the proposed model failed to identify one of the building, which might be a result of IoU threshold. According to the qualitative result, the proposed model has the best performance on building detection, while U-Net focuses on segmentation rather than detection, and the baseline model performs worse compared to the other two models.

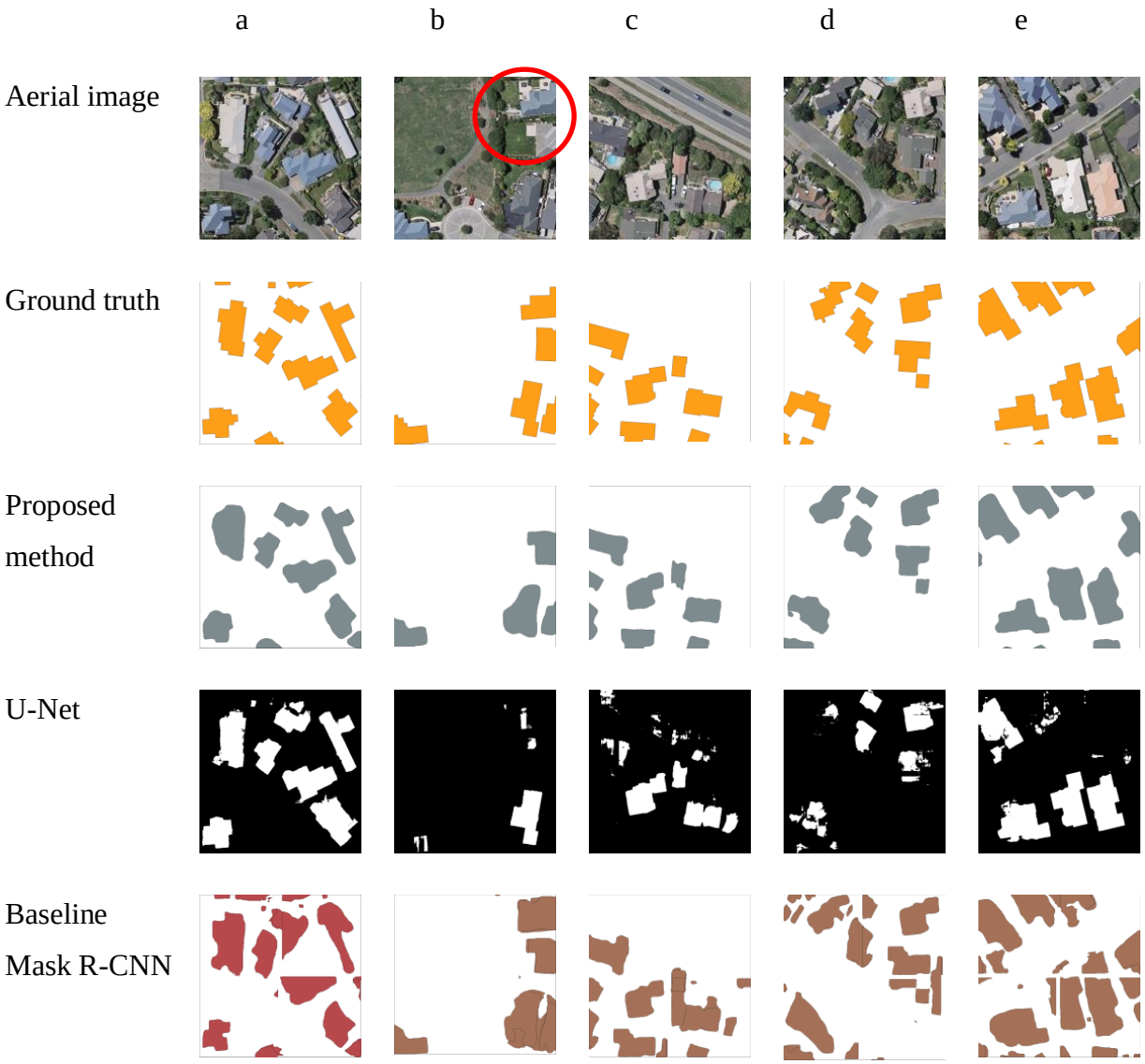


Figure 4.20 Representative samples of detection result by the proposed model, U-Net and baseline Mask R-CNN

The overall performance of the three models is quantitatively evaluated on the accuracy assessment (i.e., precision, recall and F1-score). According to Table 4.13, the U-Net method can achieve an average of 0.918 in precision, 0.771 in recall and 0.838 in F1-score, respectively. The baseline method can achieve an average of 0.3 in precision, 0.887 in recall and 0.447 in F1-score, respectively. Nevertheless, the proposed method can achieve an average of 0.92 in precision, 0.866 in recall and 0.891 in F1-score. According to the qualitative and quantitative assessments, the proposed method outperforms the other two models in terms of precision and recall. Additionally, it indicates that the proposed building detection method using Mask R-CNN model can accurately and effectively detect and segment buildings from VHR aerial imagery.

Table 4.13 Overall performance of models

Methods	Precision	Recall	F1-score
The proposed method	0.92	0.866	0.891
U-Net	0.918	0.771	0.838
Baseline Mask R-CNN	0.3	0.887	0.447

4.4 Chapter Summary

In this chapter, the qualitative and quantitative results of building detection were presented and discussed. First, the hyper-parameter values involved in the implementation of the CNN model were compared and tuned. The optimal hyper-parameters are: 15% of validation dataset; mini mask (128, 128); learning rate at 0.001; ResNet-101 backbone and COCO pre-trained weights. Then, the models using optimal hyper-parameters were trained and validated using training Dataset 1 and training Dataset 2 from 10 to 200 epochs, and the building detection results were obtained from the testing dataset. The optimal model was the model trained on training Dataset 1 at 120 epochs in terms of both qualitative and quantitative results. In the accuracy assessment, the optimal model has a precision of 0.920, a recall of 0.866 and an F1-score of 0.891, which also outperforms the state-of-the-art U-Net model and baseline Mask-RCNN.

Chapter 5 Conclusions and Recommendations

5.1 Conclusions

In this thesis, an automated building detection methodology using deep neural networks from VHR aerial imagery was proposed and implemented. Mask R-CNN was utilized as the framework of the proposed detection methodology. Two datasets with different spatial resolutions were used to verify the feasibility and practicability of the proposed model. Generally, the proposed model achieves 0.920 in precision, 0.866 in recall, and 0.891 in F1-score, respectively. In conclusion, the proposed Mask R-CNN model has the ability to operate automated building detection and segmentation from aerial imagery.

Five hyper-parameters were tested and evaluated to find the optimal model. Concerning the experiment results, the ratio of validation dataset at 15% has the highest accuracy and limits the problem of overfitting. The model using the mini mask with a size of (128, 128) outperforms the model without a mini mask and has a better performance on the segmentation task. The model using learning rate at 0.001 has the highest score of all three metrics (mAP, mAR and F1-score); thus, the optimal learning rate helps the model to achieve the highest accuracy and avoid divergence. The ResNet-101 backbone has a better performance than ResNet-50 backbone, although the training time of ResNet-101 is a little bit longer. However, the selection of ResNet-101 back used in the model is the result of a trade-off between accuracy and efficiency. The better performance of the model using ResNet-101 backbone in this study proves the necessity of a more complex and deeper neural network on the object detection task, considering the complexity and variation of the target category. Finally, the model initialized with pre-trained weights from COCO dataset was used as a starting point of the proposed model. The significant difference between the pre-trained weight and random initialized weight demonstrates that the lower-level features such as line and edges are similar among various objects, and the usage of a pre-trained model is necessary to start training a new model. Additionally, the experiment result evinces that the concept of transfer learning is fundamental during the establishment and development of deep learning (Pan & Yang, 2010).

Comparing with other building detection models, the baseline Mask R-CNN model has the worst performance, with a precision of 0.3, recall of 0.887 and F1-score of 0.447; while U-Net

is capable of achieving 0.918 in precision, 0.771 in recall and 0.838 in F1-score. The proposed Mask R-CNN method has the best performance, with a precision of 0.92, recall of 0.866 and F1-score of 0.891. The overall performance verifies the capability of deep learning method in the building detection task. Furthermore, the building detection model using deep convolutional neural networks proposed in this thesis can automatically and accurately detect and segment buildings in VHR aerial imagery.

5.2 Contributions

The proposed method in this thesis contributes to the researches and developments of building detection from VHR aerial imagery at large scale. The experimental results demonstrate the potential of the deep learning-based model to solve building detection problem. Consequently, the major contributions of this thesis can be summarized as follows:

First, an accurate and effective deep neural network using the framework of Mask R-CNN is proposed to automatically detect and extract buildings from VHR aerial imagery at large scale. Second, the proposed building detection method proves the feasibility of deep learning-based method on building detection problem. The proposed method also solves the challenges as mentioned in Chapter 1: the across and in-class diversity, the occlusion and shadow issues, and the processing of large-scale dataset.

Furthermore, the proposed building detection method can be utilized as a baseline method for future building detection task by using different format of the dataset and different locations of the study area. Also, the proposed building detection method can be utilized as the base step for the building-related task such as change detection, urban land classification, post-earthquake manipulation, etc.

5.3 Limitations and Recommendations

Generally, deep learning is empirical. There is no good theory on the actual working of or the decision made inside the large networks. The setting such as learning rate, the number of layers or other parameters can only be decided empirically rather than analytically. Therefore, future experiments are required to further enhance the performance of building detection

model. Moreover, post-processing such as edge detection, boundary regularization. corner detector can be made to improve the segmentation result. In addition, the majority of buildings in this study is detached house. In order to derive a more general models, more training data are needed to comprehensively present the features and variation of buildings.

In the meantime, the computational capability needs to be considered due to the effective demand in the experiment. Therefore, more computing resources and advanced computing devices are encouraged to train a deeper neural network and reduce time cost.

In addition, since the dataset collected on 24 February 2011, two days earlier to the occurred earthquake, the DTM for orthorectification might not be accurate and influence the position of objects. However, due to the objective of the proposed method is to detect buildings, this limitation shows less effect on the final result.

The further aim of deep learning is that a trained model can generalize other datasets without training or fine-tuning; therefore, a generalized model can be re-applied in many other cases. In order to achieve this goal, more training data are needed to comprehensively present the features and variation of buildings. For example, the images containing large building clusters, such as malls, warehouses and stadiums, are essential for the identification of urban buildings.

References

Abdullat, W. (2017). Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. Retrieved from https://github.com/matterport/Mask_RCNN. Last accessed on April 12, 2019.

Amaratunga, T. (2019). *Build Deeper: The Path to Deep Learning* (Second ed.).

Baatz, M., & Schäpe, A. (2000). Multiresolution segmentation: an optimization approach for high quality multi-scale image segmentation. *Angewandte Geographische Informationsverarbeitung, XII,* 12–23.

Baltsavias, E., Mason, S., & Stallmann, D. (1995). Use of DTMs/DSMs and orthoimages to support building extraction. *In Automatic Extraction of Man-made Objects from Aerial and Space Images*, 199-210.

Bei, Y., Damian, A., Hu, S., Menon, S., Ravi, N., & Rudin, C. (2018). New techniques for preserving global structure and denoising with low information loss in single-image super-resolution. *In 2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, 874-881.

Benz, U. C., Hofmann, P., Willhauck, G., Lingenfelder, I., & Heynen, M. (2004). Multi-resolution, object-oriented fuzzy analysis of remote sensing data for GIS-ready information. *ISPRS Journal of Photogrammetry and Remote Sensing, 58(3-4),* 239-258.

Brunn, A., & Weidner, U. (1997). Extracting buildings from digital surface models. *International Archives of Photogrammetry and Remote Sensing, 32(3),* 27-34.

Caselles, V., Kimmel, R., & Sapiro, G. (1997). Geodesic active contours. *International Journal of Computer Vision, 22(1),* 61-79.

Chambolle, A., DeVore, R. A., Lee, N. Y., & Lucier, B. J. (1998). Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage. *IEEE Transactions on Image Processing, 7(3),* 319-335.

Cheng, G., & Han, J. (2016). A survey on object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing, 117,* 11-28.

Cheng, G., & Han, J. (2016). A survey on object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing, 117,* 11-28.

Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L., & Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. *In 2011 International Joint Conference on Artificial Intelligence (IJCAI), 22(1),* 1237.

COCO. (n.d.). *Common Objects in Context*. Retrieved from http://cocodataset.org/#home. Last accessed on April 12, 2019.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning, 20(3),* 273-297.

Council, C. C. (2015). *Christchurch District Plan*. Retrieved from https://ccc.govt.nz/the-council/plans-strategies-policies-and-bylaws/plans/christchurch-district-plan/. Last accessed on April 22, 2019.

CrowdAI. (n.d.). *Mapping Challenge: Building Missing Maps with Machine Learning*. Retrieved from https://www.crowdai.org/challenges/mapping-challenge. Last accessed on April 22, 2019.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *In 2005 International Conference on Computer Vision and Pattern Recognition (CVPR), 1,* 886-893.

Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., ... & Raska, R. (2018). Deepglobe 2018: A challenge to parse the earth through satellite images. *In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),* 172-17209.

Deng, L. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing, 3,* e2.

Dong, G., Liao, G., Liu, H., & Kuang, G. (2018). A Review of the Autoencoder and Its Variants: A Comparative Perspective from Target Recognition in Synthetic-Aperture Radar Images. *IEEE Geoscience and Remote Sensing Magazine, 6(3),* 44-68.

Feng, X., Wu, S., & Zhou, W. (2017). Multi-hypergraph consistent sparse coding. *ACM Transactions on Intelligent Systems and Technology (TIST), 8(6),* 75.

Forlani, G., Nardinocchi, C., Scaioni, M., & Zingaretti, P. (2006). Complete classification of raw LIDAR data and 3D reconstruction of buildings. *Pattern Analysis and Applications, 8(4),* 357-374.

Fu, G., Liu, C., Zhou, R., Sun, T., & Zhang, Q. (2017). Classification for high resolution remote sensing imagery using a fully convolutional network. *Remote Sensing, 9(5),* 498.

Geng, J., Wang, H., Fan, J., & Ma, X. (2017). Deep supervised and contractive neural network for SAR image classification. *IEEE Transactions on Geoscience and Remote Sensing, 55(4),* 2442-2459.

Ghaffarian, S. (2014). Automatic building detection based on supervised classification using high resolution Google Earth images. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 40(3),* 101.

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

Guo, T., Morgenroth, J., & Conway, T. (2018). Redeveloping the urban forest: The effect of redevelopment and property-scale variables on tree removal and retention. *Urban Forestry & Urban Greening, 35,* 192-201.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing, 187,* 27-48.

Hammoudi, K., & Dornaika, F. (2011). A featureless approach to 3D polyhedral building modeling from aerial images. *Sensors, 11(1),* 228-259.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. *In 2017 IEEE International Conference on Computer Vision (ICCV)*, 2961-2969.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 770-778.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313(5786),* 504-507.

Hinton, G. E., & Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 1,* 282-317.

Hu, J., You, S., Neumann, U., & Park, K. K. (2004). Building modeling from LiDAR and aerial imagery. *In 2004 ASPRS, 4*, 23-28.

Huang, X., & Zhang, L. (2012). Morphological building/shadow index for building extraction from high-resolution imagery over urban areas. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 5(1),* 161-172.

Huang, X., Zhu, T., Zhang, L., & Tang, Y. (2014). A novel building change index for automatic building change detection from high-resolution remote sensing imagery. *Remote Sensing, 5(8),* 713-722.

Huang, Y., Zhuo, L., Tao, H., Shi, Q., & Liu, K. (2017). A novel building type classification scheme based on integrated LiDAR and high-resolution images. *Remote Sensing, 9(7),* 679.

Huang, Z., Cheng, G., Wang, H., Li, H., Shi, L., & Pan, C. (2016). Building extraction from multi-source remote sensing images via deep deconvolution neural networks. *In 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 1835-1838.

Huertas, A., & Nevatia, R. (1988). Detecting buildings in aerial images. *Computer Vision, Graphics, and Image Processing, 41(2),* 131-152.

Hui, J. (2018). *What do we learn from single shot object detectors (SSD, YOLOv3), FPN & Focal loss (RetinaNet)?* Retrieved from https://medium.com/@jonathan_hui/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d. Last accessed on April 22, 2019.

ImageNet. (n.d.). *IMAGENET*. Retrieved from http://image-net.org/index. Last accessed on April 22, 2019.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *In 2015 ICML*, 448-456.

Jensen, J. R., & Cowen, D. C. (1999). Remote sensing of urban/suburban infrastructure and socio-economic attributes. *Photogrammetric Engineering and Remote Sensing, 65,* 611-622.

Ji, S., Wei, S., & Lu, M. (2018). Fully convolutional networks for multisource building extraction from an open aerial and satellite imagery data set. *IEEE Transactions on Geoscience and Remote Sensing, 99,* 1-13.

Karantzalos, K., & Argialas, D. (2009). A region-based level set segmentation for automatic detection of man-made objects from aerial and satellite images. *Photogrammetric Engineering & Remote Sensing, 75(6),* 667-677.

Karantzalos, K., & Paragios, N. (2010). Large-scale building reconstruction through information fusion and 3-d priors. *IEEE Transactions on Geoscience and Remote Sensing*, 48(5), 2283-2296.

Karpathy, A. (2016). Cs231n convolutional neural networks for visual recognition. *Neural Networks, 1*.

Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision, 1(4),* 321-331.

Kavzoglu, T., & Yildiz, M. (2014). Parameter-based performance analysis of object-based image analysis using aerial and Quikbird-2 images. *ISPRS Annals, 2(7),* 31.

Keras Documentaiton. (n.d.). Applications. Retrieved from https://keras.io/applications/. Last accessed on April 24, 2019.

Keras: The Python Deep Learning library. (n.d.). Retrieved from https://keras.io/. Last accessed on April 24, 2019.

Khalid, S., Khalil, T., & Nasreen, S. (2014). A survey of feature selection and feature extraction techniques in machine learning. *In 2014 Science and Information Conference*, 372-378.

Konstantinidis, D. (2017). *Building Detection for Monitoring of Urban Changes.* (Doctoral dissertation, Imperial College London). Retrieved from https://spiral.imperial.ac.uk/handle/10044/1/57036. Last accessed on April 25, 2019.

Konstantinidis, D., Stathaki, T., Argyriou, V., & Grammalidis, N. (2015). A probabilistic feature fusion for building detection in satellite images. *In 2015 International Conference on Computer Vision Theory and Applications*, 205–212.

Konstantinidis, D., Stathaki, T., Argyriou, V., & Grammalidis, N. (2017). Building detection using enhanced HOG–LBP features and region refinement processes. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 10(3)*, 888-905.

Kovacs, A., & Sziranyi, T. (2012). Orientation based building outline extraction in aerial images. ISPRS Annals of Photogram. *ISPRS Annals, 1*, 141-146.

Lagrange, A., Le Saux, B., Beaupere, A., Boulch, A., Chan-Hon-Tong, A., Herbin, S., ... & Ferecatu, M. (2015). Benchmarking classification of earth-observation data: From learning explicit features to convolutional networks. *In 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 4173-4176.

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

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. (2012). Efficient backprop. *In Neural networks: Tricks of the Trade*, 9-48.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521(7553)*, 436.

Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., & Sun, J. (2018). Detnet: A backbone network for object detection. *arXiv preprint*, arXiv:1804.06215, 2018.

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

LINZ Data Service. (2014). Christchurch Post-Earthquake 0.1m Urban Aerial Photos (24 February 2011). *Data.linz.govt.nz*. Retrieved from https://data.linz.govt.nz/layer/51932-christchurch-post-earthquake-01m-urban-aerial-photos-24-february-2011/metadata/. Last accessed on April 22, 2019.

Liou, C. Y., Cheng, W. C., Liou, J. W., & Liou, D. R. (2014). Autoencoder for words. *Neurocomputing*, 139, 84-96.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431-3440.

Ma, L., Li, M., Ma, X., Cheng, L., Du, P., & Liu, Y. (2017). A review of supervised object-based land-cover image classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, *130*, 277-293.

Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2016). Fully convolutional neural networks for remote sensing image classification. *In 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 5071-5074.

Marcu, A., & Leordeanu, M. (2016). Dual local-global contextual pathways for recognition in aerial imagery. *arXiv preprint*, arXiv:1605.05462.

Marmanis, D., Wegner, J. D., Galliani, S., Schindler, K., Datcu, M., & Stilla, U. (2016). Semantic segmentation of aerial images with an ensemble of CNNs. *ISPRS Annals*, *3*, 473-480.

McGlone, J. C., & Shufelt, J. A. (1994). Projective and object space geometry for monocular building extraction. *In 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 54-61.

Mitliagkas, I., Zhang, C., Hadjis, S., & Ré, C. (2016). Asynchrony begets momentum, with an application to deep learning. *In 2016 Annual Allerton Conference on Communication, Control, and Computing*, 997-1004.

Mnih, V. (2013). *Machine Learning for Aerial Image Labeling*. (Doctoral dissertation, University of Toronto). Retrieved from https://www.cs.toronto.edu/~vmnih/docs/Mnih_Volodymyr_PhD_Thesis.pdf. Last accessed on April 25, 2019.

Mountrakis, G., Im, J., & Ogole, C. (2011). Support vector machines in remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing, 66(3),* 247-259.

Myint, S. W., Gober, P., Brazel, A., Grossman-Clarke, S., & Weng, Q. (2011). Per-pixel vs. object-based classification of urban land cover extraction using high spatial resolution imagery. *Remote Sensing of Environment, 115(5), 1,* 145-1161.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. *In 2010 International Conference on Machine Learning*, 807-814.

Ngiam, J., Chen, Z., Koh, P. W., & Ng, A. Y. (2011). Learning deep energy models. *In 2011 International Conference on Machine Learning*, 1105-1112.

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint,* arXiv:1511.08458.

Ok, A. O., Senaras, C., & Yuksel, B. (2013). Automated detection of arbitrarily shaped buildings in complex environments from monocular VHR optical satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing, 51(3),* 1701-1717.

Ok, A. O. (2013). Automated detection of buildings from single VHR multispectral images using shadow information and graph cuts. *ISPRS Journal of Photogrammetry and Remote Sensing, 86,* 21-40.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 22(10),* 1345-1359.

Peng, J., & Liu, Y. C. (2005). Model and context‐driven building extraction in dense urban aerial images. *International Journal of Remote Sensing, 26(7),* 1289-1307.

Petropoulos, G. P., Kalaitzidis, C., & Vadrevu, K. P. (2012). Support vector machines and object-based classification for obtaining land-use/cover cartography from Hyperion hyperspectral imagery. *Computers and Geosciences, 41*, 99-107.

Reagen, B., Adolf, R., Whatmough, P., Wei, G. Y., & Brooks, D. (2017). Deep learning for computer architects. *Synthesis Lectures on Computer Architecture, 12(4),* 1-123.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems,* 91-99.

ResNet-101. (n.d.). Retrieved from http://ethereon.github.io/netscope/#/gist/b21e2aae116dc1ac7b50. Last accessed on April 22, 2019.

ResNet-50. (n.d.). Retrieved from http://ethereon.github.io/netscope/#/gist/db945b393d40bfa26006. Last accessed on April 22, 2019.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-assisted Intervention*, 234-241.

Rosebrock, A. (2017). *Deep Learning for Computer Vision with Python: Starter Bundle.* Pyimagesearch.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Berg, A. C. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision, 115(3)*, 211-252.

Saito, S., & Aoki, Y. (2015). Building and road detection from large aerial imagery. *IS&T/SPIE Electronic Imaging*, 94050K– 94050K.

Saito, S., Yamashita, T., & Aoki, Y. (2016). Multiple object extraction from aerial imagery with convolutional neural networks. *Electronic Imaging, 2016(10),* 1-9.

San, D. K., & Turker, M. (2010). Building extraction from high resolution satellite images using Hough transform. *ISPRS Archives*, 38(Part 8), 1063-1068.

Santos, L. A. (n.d.). *Convolution*. Retrieved from https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolution.html. Last accessed on April 22, 2019.

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. *In 2010 International Conference on Artificial Neural Networks (ICANN)*, 92-101.

Sherrah, J. (2016). Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. *arXiv preprint* arXiv:1606.02585.

Shu, Y. (2014). *Deep Convolutional Neural Networks for Object Extraction from High Spatial Resolution Remotely Sensed Imagery*. (Doctoral dissertation, University of Waterloo). Retrieved from https://uwspace.uwaterloo.ca/bitstream/handle/10012/9140/Shu_Yuanming.pdf?sequence=5&isAllowed=y. Last accessed on April 22, 2019.

Stats NZ. (n.d.). *Subnational population estimates (UA, AU), by age and sex, at 30 June 1996, 2001, 2006-18 (2017 boundaries)*. Retrieved from http://nzdotstat.stats.govt.nz/wbos/Index.aspx?DataSetCode=TABLECODE7541. Last accessed on April 22, 2019.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. *In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1-9.

Theng, L. B. (2006). Automatic building extraction from satellite imagery. *Engineering Letters, 13(4)*, 255-259.

Turker, M., & Koc-San, D. (2015). Building extraction from high-resolution optical spaceborne images using the integration of support vector machine (SVM) classification,

Hough transformation and perceptual grouping. *International Journal of Applied Earth Observation and Geoinformation, 34*, 58-69.

Turlapaty, A., Gokaraju, B., Du, Q., Younan, N. H., & Aanstoos, J. V. (2012). A hybrid approach for building extraction from spaceborne multi-angular optical imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 5(1),* 89-100.

UN, D. (2015). World urbanization prospects: The 2014 revision. *United Nations Department of Economics and Social Affairs, Population Division: New York, NY, USA*.

Vakalopoulou, M., Karantzalos, K., Komodakis, N., & Paragios, N. (2015). Building detection in very high-resolution multispectral data with deep learning features. *In 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 1873-1876.

van Etten, A., Lindenbaum, D., & Bacastow, T. M. (2018). Spacenet: A remote sensing dataset and challenge series. *arXiv preprint*, arXiv:1807.01232.

Verma, V., Kumar, R., & Hsu, S. (2006). 3D building detection and modeling from aerial LIDAR data. *In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2*, 2213-2220.

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: a brief review. *Computational Intelligence and Neuroscience*, 1-13.

Wang, C., & Hsu, P. (2007). Building detection and structure line extraction from airborne lidar data. *Journal of Photogrammetry and Remote Sensing, 12(4),* 365-379.

Wu, G., Shao, X., Guo, Z., Chen, Q., Yuan, W., Shi, X., ... & Shibasaki, R. (2018). Automatic building segmentation of aerial imagery using multi-constraint fully convolutional networks. *Remote Sensing, 10(3)*, 407.

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

Zeiler, M. D. (2013). *Hierarchical Convolutional Deep Learning in Computer Vision.* (Doctoral dissertation, New York University). Retrieved from https://search.proquest.com/openview/62c046242f67ce115a76b9224e66a69c/1?pq-origsite=gscholar&cbl=18750&diss=y. Last accessed on April 22, 2019.

Zhang, L., Zhang, L., & Du, B. (2016). Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine, 4(2),* 22-40.

Zhang, S., Yao, H., Sun, X., & Lu, X. (2013). Sparse coding based visual tracking: Review and experimental comparison. *Pattern Recognition, 46(7),* 1772-1788.

Zhou, Y., Arpit, D., Nwogu, I., & Govindaraju, V. (2014). Is joint training better for deep auto-encoders? *arXiv preprint,* arXiv:1405.1380.

Zhu, X. X., Tuia, D., Mou, L., Xia, G. S., Zhang, L., Xu, F., & Fraundorfer, F. (2017). Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine, 5(4),* 8-36.