AUTOMATED BUILDING EXTRACTION FROM AERIAL IMAGERY WITH MASK R-CNN

by

Zilong Yang

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Civil Engineering

Lyles School of Civil Engineering

West Lafayette, Indiana

December 2020

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Jie Shan, Chair
Lyles School of Civil Engineering

Dr. James S. Bethel
Lyles School of Civil Engineering

Dr. Wen-wen Tung
Department of Earth, Atmospheric, and Planetary Sciences

Approved by:
Dr. Dulcy Abraham

ACKNOWLEDGMENTS

First and foremost, I express my sincere gratitude to my advisor, Prof. Jie Shan, for his continuous and enthusiastic support of my research as well as the immense knowledge he imparted to me during this process. I am also thankful to the other members of my advisory committee, Prof. James Bethel and Prof. Wen-wen Tung, for their kindness, encouragement, and insightful comments.

My sincere thanks also go to Dr. Fenggang Yang, who provided me a part-time research assistant position in his Center, which gave me the opportunity to apply the knowledge I learned in class to real practice. The experience I gained and the skills I developed from this opportunity will be invaluable as I go forward in my career.

I am extremely grateful to the following individuals for the knowledge they shared with me: Dr. Wenyuan Zhang, Dr. Xianjun Gao, and Zhixin Li, from whom I learned state-of-the-art methods in deep learning; Xiaoyi Peng and Ce Wang, from whom I learned how to analyze geospatial data; and Xiangxi Tian, from whom I learned image-based pavement macrotexture determination. I am also thankful to Indiana Map and OpenStreetMap for the open access I received to the data used in this work.

Last but not the least, I am eternally grateful to my parents for supporting me spiritually during my academic journey and for their unceasing encouragement and concern.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Average Distance Error	ADE
Convolutional Neural Network	CNN
Differential Morphological Profiles	DMP
Fully Connected Network	FCN
False Negative	FN
False Positive	FP
Feature Pyramid Network	FPN
Histogram of Gradient	HOG
Intersection over Union	IoU
Mean Pixel Accuracy	MPA
Regional Convolutional Neural Network	R-CNN
Rectified Linear Unit	ReLU
Residual Network	ResNet
Recurrent Neural Network	RNN
Region of Interest	RoI
Region Proposal Network	RPN
Single Shot multibox Detector	SSD
Scale Invariant Feature Transform	SIFT

Spatial Pyramid Pooling network	SPP-net
Support Vector Machine	SVM
True Negative	TN
True Positive	TP
You Only Look Once	YOLO

ABSTRACT

Buildings are one of the fundamental sources of geospatial information for urban planning, population estimation, and infrastructure management. Although building extraction research has gained considerable progress through neural network methods, the labeling of training data still requires manual operations which are time-consuming and labor-intensive. Aiming to improve this process, this thesis developed an automated building extraction method based on the boundary following technique and the Mask Regional Convolutional Neural Network (Mask R-CNN) model. First, assisted by known building footprints, a boundary following method was used to automatically best label the training image datasets. In the next step, the Mask R-CNN model was trained with the labeling results and then applied to building extraction. Experiments with datasets of urban areas of Bloomington and Indianapolis with 2016 high resolution aerial images verified the effectiveness of the proposed approach. With the help of existing building footprints, the automatic labeling process took only five seconds for a 500×500 pixel image without human interaction. A 0.951 intersection over union (IoU) between the labeled mask and the ground truth was achieved due to the high quality of the automatic labeling step. In the training process, the Resnet50 network and the feature pyramid network (FPN) were adopted for feature extraction. The region proposal network (RPN) then was trained end-to-end to create region proposals. The performance of the proposed approach was evaluated in terms of building detection and mask segmentation in the two datasets. The building detection results of 40 test tiles respectively in Bloomington and Indianapolis showed that the Mask R-CNN model achieved 0.951 and 0.968 F_1-scores. In addition, 84.2% of the newly built buildings in the Indianapolis dataset were successfully detected. According to the segmentation results on these two datasets, the Mask R-CNN model achieved the mean pixel accuracy (MPA) of 92% and 88%, respectively for Bloomington and Indianapolis. It was found that the performance of the mask segmentation and contour extraction became less satisfactory as the building shapes and roofs became more complex. It is expected that the method developed in this thesis can be adapted for large-scale use under varying urban setups.

1. INTRODUCTION

1.1 Background

Nowadays, building detection is one of the most important tasks for interpreting satellite remote sensing data. It plays a significant role in city planning, population estimation, and many other applications (Yang et al. 2019). To achieve satisfactory results, manually labeling buildings still is required, which is very time consuming, especially in high resolution imagery. Therefore, developing an automatic and effective building extraction method would be a tremendous achievement.

Research related to automatic building detection began in the late 1980s. Since the resolution of the images of that period was low, the developed algorithms were limited to low-level features such as edges, lines, and corners. These primitives were grouped to generate higher level features, such as rectangles and parallelograms (Lin and Nevatia 1998), which then were used in a verification process to determine the final building polygons. Until now, a set of building detection studies have continued to follow this simple approach: first detect the building with respect to its edges or contours, and then decide whether the object is a building based on user-defined geometric rules. Saeedi and Zwick (2008) and Guducu and Halici (2010) detected lines from an image, and then eliminated some of the lines according to geometric constraints dominated by user-defined parameters. In the verification process, the candidate buildings were verified by their rectangle and corner evidence.

The building height and the illumination angle of the sun make the shadow areas important clues about the building's location in optical remote sensing images. Liow and Pavlidis (1990) used shadows to determine the edges and corners of a building and then completed their boundary grouping process with shadow information. Peng et al. (2005) adopted a snake algorithm to detect the contours of candidate buildings and then refined the building candidates with their shapes and shadows. Izadi and Saeedi (2010) applied mean-shift segmentation and a set of shape-based rules to obtain candidate rooftops and then verified these candidates from shadow evidence.

Due to on-going advances in line feature extraction over time, a variety of methods based on Hough transform have been proposed to extract building from aerial images. Cha et al. (2006)

developed a probabilistic Hough transform algorithm to detect the lines in the image, and a rectangle was detected when the four peaks of the Hough image satisfied certain geometric conditions. Candidate rectangles then were able to be verified by shadow evidence (Jung and Schramm 2004). Hough transform was used to verify candidate buildings detected by mathematical morphology (Pakizeh and Palhang 2010). Differential morphological profiles (DMPs) are widely used in building detection since they can determine the morphological characteristics of connected components in an image through morphological reconstruction. DMPs are usually applied to detect initial building candidates, then Hough transform or shadow information are used for building verification (Jin and Davis 2005, Aytekin et al. 2009, Sportouche et al. 2009).

Although the above methods seem reasonable, some issues still exist. First, many building edges and lines cannot be correctly detected even in high spatial resolution images. Moreover, many false edges and lines can be extracted, generating incorrect hypotheses. The second issue is that the user-defined rules mentioned above are mostly threshold-dependent, which means they are not applicable for all types of situations. Therefore, building detection methods based on only low-level features are inadequate for detecting buildings with arbitrary shapes (Yuksel 2012).

Instead of using low-level features, the following studies applied supervised or unsupervised pattern recognition methods with spectral features. Lee et al. (2003) first segmented an image with ISODATA (Ball and Hall 1965) and then extracted a set of spectral features from the segments, followed by an multi-class process of classification using ECHO (Kettig and Landgrebe 1976). Sirmacek and Unsalan (2008) first used invariant color features to detect buildings with red rooftops in an image, from which they then detected missing buildings by using shadow information. In order to accurately determine building shapes, they also applied box fitting to refine the building edges.

In recent years, deep learning methods, such as the Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN), are gradually dominating the field of pattern recognition. Deep learning has enabled significant progress in image classification, image semantic segmentation, and object detection. Inspired by these recent achievements, increasingly more researchers in the remote sensing field are applying deep learning techniques to building extraction. Vakalopoulou et al. (2015) proposed an automated building detection method based

on deep convolutional neural networks. They employed a support vector machine (SVM) classifier with a large training dataset and then used a Markov random field (MRF) model to refine the pixel-level classification results. Makantasis et al. (2015) proposed a building detection framework from hyperspectral data based on deep learning classification. The spectral and spatial information of the pixels were effectively exploited by a CNN and then the classification task was conducted by a multi-layer perceptron (MLP). Xu et al. (2018) designed a neural network based on a deep residual network for image segmentation. They trained the network to extract the building at the pixel level and then employed a guide filter to optimize the classification results.

1.2 Related works

In an object extraction problem, a feature of an object is called an invariant of the object (Arica and Vural 2003) if it does not change among different images. For a robust object detection algorithm, it is essential to obtain an invariant of the object.

Before deep neural networks were widely used for object detection and segmentation tasks, classical object detection methods were mainly based on extracting feature descriptors, such as the histogram of orientated gradients (HOG) (Dalal and Triggs, 2005) or scale-invariant feature transform (SIFT) (Lowe 1999) with subsequent classification by a linear classifier, such as SVM. The basic idea of HOG is characterization of the shape of the structure by the distribution of local intensity gradient or edge detection. HOG first divides an image into a set of spatially-overlapped blocks. Then, each block is further divided into smaller n*n cells, where each cell contains a fixed number of gradient bins. Each pixel in the cell yields a score for the gradient orientation bin, which is proportional to the magnitude of the gradient at that pixel. The gradient orientations are quantized into several bins once the intensity gradients of each pixel are computed. In practice, HOG uses a “sliding window” (Sermanet et al. 2013) approach to extract and classify the regions at multiple scales and aspect ratios. Finally, the feature vector is processed by a linear classifier, such as SVM, to generate the final output. Although HOG characterizes the relationship between the pixels of the image well, it lacks robustness with respect to occlusion and deformation. HOG’s low detection speed and accuracy also preclude its use in many applications (Zafar 2018).

SIFT is an image descriptor that has proven to be very useful for image matching and object recognition under real-world conditions due to its invariance to rotations, translations, and scaling transformations during image processing (Lowe 1999). SIFT first extracts the key points from labeled grayscale images by creating a scale space that is obtained by constructing a set of Gaussian-blurred images. Then, a feature vector is computed by finding the histogram of the gradient directions in a local neighborhood around each key point, whereby the poor key points (e.g., low contrast regions) are eliminated; and calculations are carried out for the remaining key points to remove the effect of orientation. Finally, Hough transform is applied to identify the clusters that form a specific object and the probability of a feature representing an object is computed. A set of building detection methods based on SIFT also has been proposed. For example, Sirmacek and Unsalan (2009) proposed an approach that combined SIFT and graph theoretical tools for urban area building detection. Tao et al. (2011) detected airports from high resolution images using an improved SIFT matching method.

However, it is difficult to find an invariant since buildings vary in terms of shape, size, color and texture. Therefore, applying a deep learning method to extract the deep learning features is a more practical and effective approach. This will be the focus of this section.

1.3 Deep learning methods

In deep learning (LeCun et al. 2015), one of the deep neural networks is CNN, which is designed to solve difficult pattern recognition tasks (LeCun et al. 1998). As shown below in Figure 1.1, a typical CNN is consisting of three basic types of layers: 1) convolutional layers, 2) pooling layers, and 3) fully connected layers. The convolutional layer is the fundamental component of CNNs that relate to feature extraction of the input imagery. A convolutional layer is comprised of a set of convolution kernels of different sizes. These kernels are convolved with the input image to generate a 2D feature map. The pooling layers are applied to reduce the number of parameters and the model's computation time. During the pooling process, the feature of a certain location is replaced with a summary statistic of the neighbor features. There are many common pooling functions, such as the max pooling (Ranzato et al. 2007) and the average pooling (LeCun et al. 1998). In practice, it is recommended to use the max-pooling (Scherer et al. 2010). The max-pooling function takes a square region as its input and outputs the maximum value of the pixels in the region. The fully connected layer is the final layer of a CNN that

contains a number of neurons that are connected to each other between two adjacent layers. The fully connected layers are usually added to the end of CNNs to perform classification based on the features generated by the previous layers (Gu et al. 2018).

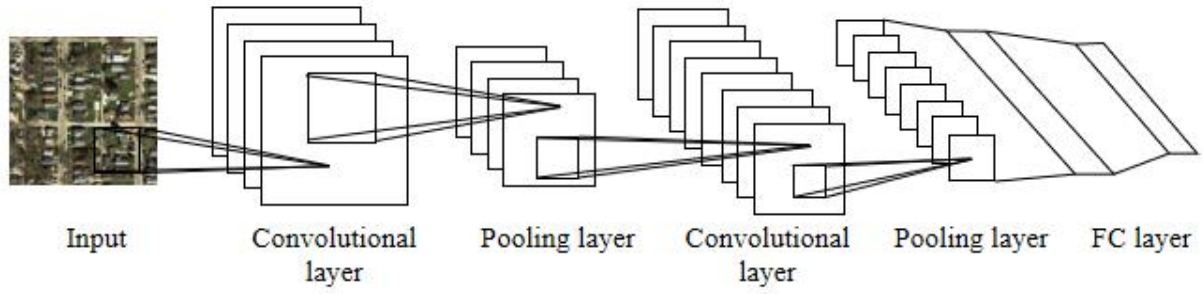


Figure 1.1. Illustration of a simple CNN for image classification (adapted from LeCun et al. 1998)

The increasing complexity of image classification problems necessarily requires higher performance CNNs. The most straightforward way to improve the performance of CNNs is by increasing their depth and width. Many well-known deep CNNs such as VGGNet (Simonyan and Zisserman, 2015) and AlexNet (Krizhevsky et al. 2017), are formed by simply stacking up many convolutional layers, pooling layers, and fully-connected layers. In those deep CNNs, back-propagation of the gradients is needed since the information flowing through the network passes through many stages of multiplication. This typically causes the gradients to either explode or vanish. The gradient exploding problem can be solved easily by applying gradient clipping. However, gradient varnishing is quite difficult to overcome due to problems such as the use of saturated activation functions such as the hyperbolic tangent or the logistic sigmoid (Xu et al., 2016). Therefore, using non-saturated activation functions, such as ReLU (Glorot et al., 2011) and Leaky ReLU (Goodfellow et al. 2016) as alternativesn is recommended. In practice, other layers, such as dropout (Srivastava et al. 2014), batch normalization (Ioffe and Szegedy, 2015), and group normalization (Wu and He, 2018) are often added to CNNs to improve performance and to avoid overfitting. Graphs of four different activation functions are shown in Figure 1.2. For the sigmoid function and hyperbolic tangent function, if the input value is very high or very low, the gradient is close to 0, which results in a gradient varnishing problem (Nwankpa et al. 2018). However, the gradient of ReLU (the gradient varnishing

problem of negative input can be removed by low learning rate) and Leaky ReLU is 1 when the input is positive, which can avoid the gradient varnishing problem (Xu et al. 2015).

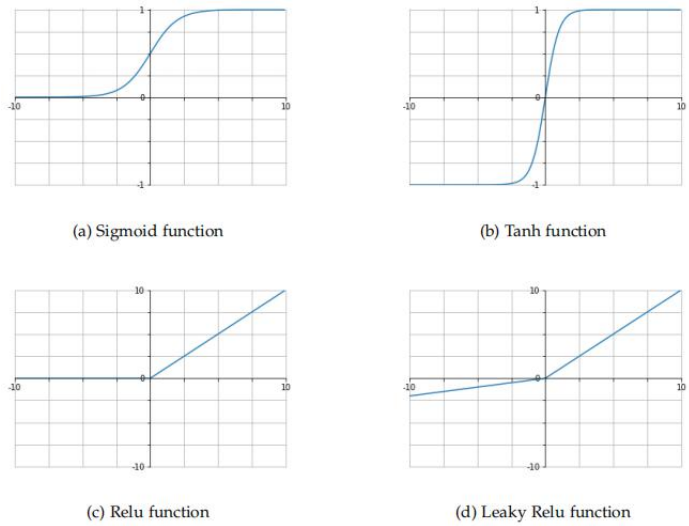


Figure 1.2. Comparison between four activation functions (adapted from Zafar 2018).

To further improve the performance of CNNs in training at very deep networks, residual network (ResNet) has been proposed (He et al., 2016). ResNet adds "shortcut" connections to standard CNN layers to enable the gradient signal to travel back from later layers to earlier layers. With the help of the "shortcut" connections, ResNet can successfully train very deep CNNs with 50, 101, and even 152 layers. Inspired by the success of "shortcut" connections in CNNs, Huang et al. (2017) proposed a novel network called dense convolutional network (DenseNet). The core idea of DenseNet is the use of multiple densely connected blocks in which all the layers are directly connected with each other. In that way, a layer in a dense block can use the feature maps of all the preceding layers in the block as inputs. Therefore, DenseNet achieves state-of-the-art performance with fewer parameters and less computation compared to other networks.

1.3.1 One-stage detector based method

Inspired by the success of CNNs in image classification, an increasing number of researchers have applied CNNs to solve the more challenging task of object extraction. The main purpose of object extraction is to localize and classify existing objects in images. In the past few years, many CNNs based on object extraction methods have been proposed, which can

be divided into two general categories: 1) models based on regression with one-stage detectors, such as You Only Look Once (YOLO) (Redmon et al. 2016), and 2) models based on region proposal with two-stage detectors like Mask R-CNN (He et al. 2017).

One-stage detectors take an image as the input and learn the class probabilities and coordinates of a bounding box by treating the above tasks as a simple regression problem. YOLO is a real-time object detection model that predicts class probabilities and bounding boxes in a single evaluation. The architecture of YOLO is shown in Figure 1.3. YOLO is extremely fast since it unifies region proposal and region classification into one single neural network. However, there are two major drawbacks that reduce the accuracy of YOLO. First, there is a considerable number of localization errors and low recall compared to the region proposal-based methods. To address the above drawbacks of YOLO, YOLOv2 (Redmon and Farhadi 2017) and YOLOv3 (Redmon and Farhadi 2018) were proposed. In order to regularize the model and improve its convergence, YOLOv2 adopted batch normalization on all the convolution layers. Instead of directly predicting the coordinates of the bounding boxes with the fully-connected layers, YOLOv2 predicts the location of the anchor boxes with the convolutional layers. In addition, YOLOv2 employs a higher resolution classifier to generate fine-grained features. With the above enhancements, YOLOv2 achieved 4% improvement in the mean average precision (mAP). YOLOv3 employs a new base model called Darknet-53 which has residual blocks. To improve the performance in small object detection, YOLOv3 applies a shortcut connect which allows the model to obtain finer grained information from the earlier feature map. Some methods based on YOLO have been employed in building detection. Xie et al. (2020) proposed a Locally Constrained (LOCO) YOLO model to detect small densely distributed building footprints. By employing LOCO as an invariant of YOLO, the model is able to improve the robustness of building size predictions. Ma et al. (2020) used an improved YOLOv3 model to detect collapsed buildings in post-earthquake remote sensing images and achieved a target precision of 90.89%.

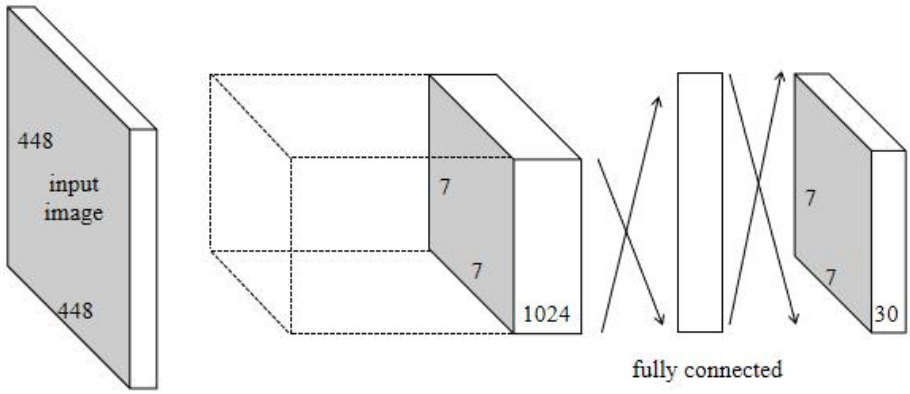


Figure 1.3. Illustration of YOLO architecture. (adapted from Redmon et al. 2016)

Single shot multibox detector (SDD) is another typical one-stage object detector (Liu et al., 2016). It solves the drawbacks of YOLO by a series of modifications: 1) prediction is performed at multiple feature maps from the later stage of a network to allow detection at multiple scales; 2) small convolutional filters are employed for predicting the object class and offsets in the bounding box locations; and 3) separate filters are utilized to predict objects at different aspect ratios. A VGG-16 model pre-trained on the ImageNet dataset is employed as the base network for feature extraction. Unlike YOLO, which operates on a single scale feature map, SDD adds an extra feature layer to the end of the base network to enable detection at multiple scales. Instead of employing a fully-connected layer for producing predictions like YOLO, SDD attaches a set of small convolutional filters to each added feature layer and uses them to predict object classes and offsets in bounding box locations. Default boxes with different scales and aspect ratios are also applied to several feature maps of different resolution. An example of building detection at multiple scales and aspect ratios based on SDD is shown in Figure 1.4. These modifications allow SSD to effectively detect objects at multiple scales and aspect ratios and make SSD both faster and more accurate than YOLO.

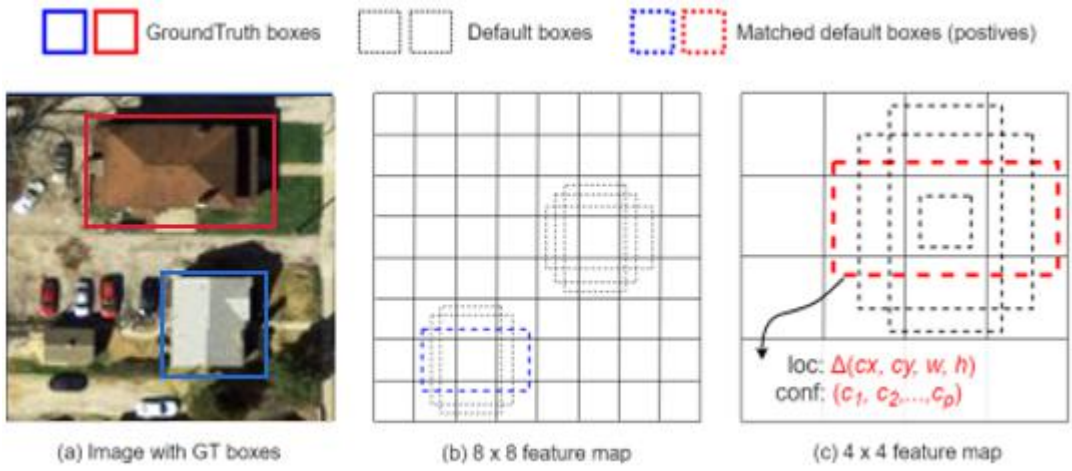


Figure 1.4. Illustration of SDD. SDD employs feature maps with different scales (8*8 in (b) and 4*4 in (c)) and default boxes of different aspect ratios to enable detection of buildings (a) at multiple scales and aspect ratios. (adapted from Liu et al. 2016)

1.3.2 Two-stage detector based method

Two-stage detectors such as Mask R-CNN use region proposal networks (RPNs) to generate regions of interest (RoI) at the first stage, which are then sent to the second stage for object classification and bounding box regression. In fact, Mask R-CNN was developed from a series of CNN models, such as R-CNN (Girshick et al. 2014), Fast R-CNN (Girshick 2015), and Faster R-CNN (Ren et al. 2015). Girshick et al. proposed a Region-based convolution neural network (R-CNN), which combines region proposals with CNNs to detect objects in an image via bounding boxes. R-CNN model consists of three steps. First, the selective search is used to generate region proposals by scanning the input image for possible object. The selective search algorithm is the most commonly used region proposal generation method that generates nearly 2, 000 region proposals per image. These region proposals are then warped into a CNN to extract a fixed-length feature vector from each region in an image. Finally, feature vectors obtained from the CNN are classified by a set of class specific linear SVMs. In addition, a linear regression model is employed for refining the bounding boxes. The three steps of an R-CNN are shown in Figure 1.5, Although, the R-CNN is intuitive, it still has many notable drawbacks. One of the major drawbacks is long training time and large storing space of 2,000 region proposals. Another major drawback is the fixed input size of region proposals that could lead to the generation of bad candidate region proposals.

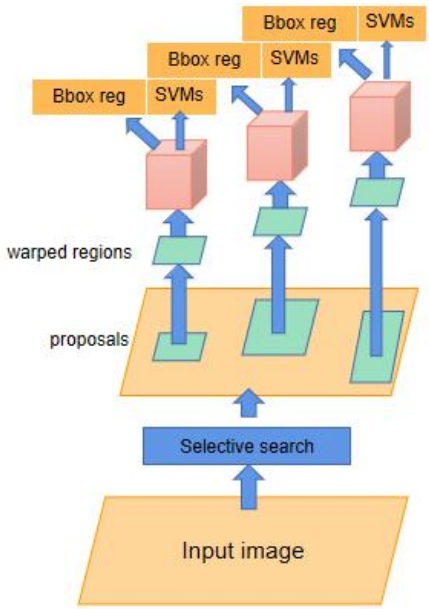


Figure 1.5. An overview of R-CNN (adapted from Girshick et al. 2014).

To solve the drawbacks of R-CNN, Spatial Pyramid Pooling networks (SPP-nets) (He et al. 2015) were proposed. The structure of SPP-net is shown in Figure 1.6. SPP-nets first compute the convolutional feature for the entire input image only once. Then, spatial pyramid pooling is employed to generate a fixed-length representation on the shared feature maps. Finally, after a set of linear SVMs are applied for proposal classification, a bounding box regressor is utilized for bounding box refinement. By employing computation-sharing via shared feature maps, SPP-nets can run faster than R-CNN; however, it has unresolved drawbacks, such as lengthy training time and large feature storage space requirements.

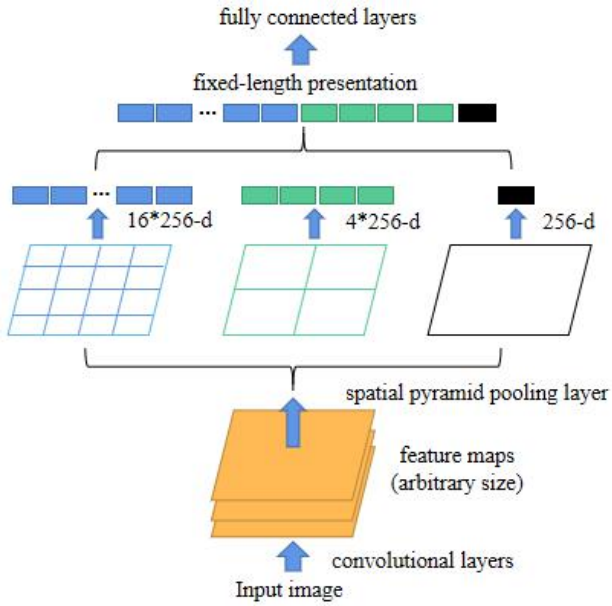


Figure 1.6. Structure of SPP-net (adapted from He et al. 2015).

To address the disadvantages of R-CNN and SPP-nets, Girshick et al. (2015) developed a faster object detection algorithm, Fast R-CNN. Instead of feeding 2,000 region proposals to the CNN every time, the Fast R-CNN executes the convolution operation only once per image, from which a feature map is generated. Also, an RoI Pooling layer is employed for extracting a fixed length feature vector for each proposal candidate. Their results indicated that the fast R-CNN was 10 times faster than R-CNN in training and 20 times faster in testing. Both R-CNN and Fast R-CNN use selective search to find region proposals. However, its selective search was still a slow and time-consuming process, which affected the performance of the network. Thereafter, Ren et al. (2015) proposed a new object detection algorithm that eliminated the selective search step and let the network itself detect the region proposals. The new algorithm is called Faster R-CNN and its structure is shown in Figure 1.7. They built a separate network to predict the region proposals and reshaped them with an RoI pooling layer, which was used to classify the region proposals and predict the offset values of the bounding boxes. Development of the above algorithms improved the speed and accuracy of object detection; but due to the drawbacks of RoI pooling and the lack of pixel level segmentation, accurately extracting the location and boundary of the object was still a problem.

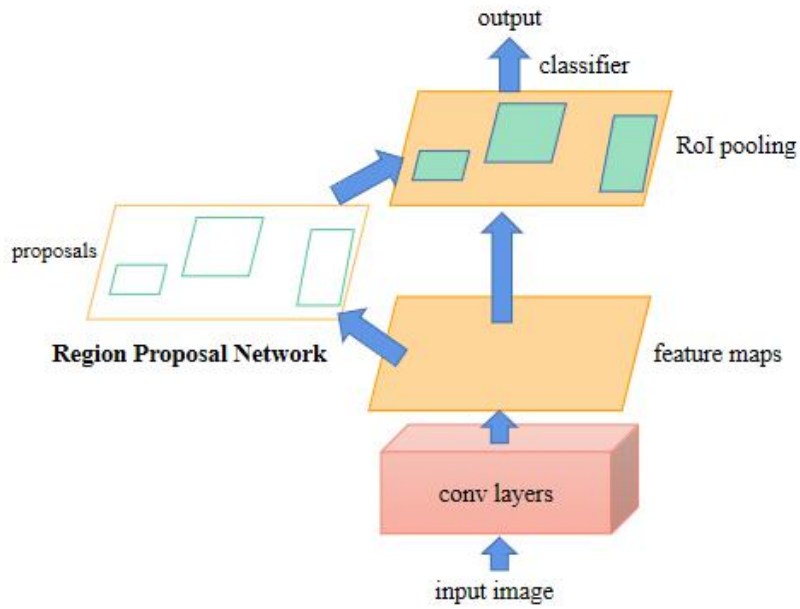


Figure 1.7. Structure of Faster R-CNN (adapted from Ren et al. 2015).

Mask R-CNN (He et al., 2017) extends Faster R-CNN to simultaneously detect objects in an image and generates a high-quality segmentation mask for each object. Mask R-CNN adds a mask branch to the detection network of Faster R-CNN to generate a binary mask for each RoI. In order to fix the misalignment between the network input and output of Faster R-CNN, RoI Align was proposed and utilized by the authors to preserve exact spatial location. To train the network from end-to-end, a multi-task loss was proposed, which is a combination of a classification loss, a bounding box loss, and a mask loss. The classification loss and the bounding box loss are identical to those employed in Faster R-CNN, while the mask loss is an average binary cross-entropy loss applied on a binary mask, each of which corresponding to a class.

In most of the deep learning tasks, a large amount of labeled data is needed to train the model to learn the features as fully as possible. Although researchers have developed many manual labeling tools, such as VGG image annotator (VIA) (Dutta and Zisserman 2019) and LabelMe (Torralba et al. 2010), it is still a time consuming and laborious task. For example, it takes five minutes to label an image with 30 objects. To address this problem, some annotation tools have been developed to achieve auto-labeling. They allow one load of pre-annotated data and apply artificial intelligence (AI) to annotate or label the dataset. In one image annotation experiment, auto-labeling combined with human-powered review and improvements was 10% faster than the 100% manual labeling process; however, the results also showed that it had

more than a 5-pixel margin of error for objects and missed objects far from the camera (Heffelfinger 2020). Auto-labeling tools can save a lot of time in the labeling process, but they also can result in unsatisfactory quality of annotation. Therefore, quick and accurate data labeling is still a common problem.

This thesis proposes a new automatic building detection method to effectively label a dataset and to generate accurate building bounding boxes and masks. A Mask R-CNN model is utilized for extracting building and evaluating the performance in terms of building detection and mask segmentation.

1.4 Structure of the thesis

The remainder of this thesis proceeds as follows. Chapter 2 introduces the proposed new automatic image labeling method and building extraction model. Chapter 3 describes the orthoimages and building footprint data used in this thesis. Chapter 4 presents the experiments and the corresponding results and analyses. Finally, Chapter 5 discusses the conclusions, limitations, and recommendations for future research.

2. METHODOLOGIES

As demonstrated in Figure 2.1, the proposed approach consists of two main parts: 1) automatic image labeling based on boundary following and 2) building detection based on the Mask R-CNN model. First, the building footprint is used to extract the candidate building area. Then, the Sobel operator and a boundary following method are applied to extract building contours and complete image labeling. Moreover, several data augmentation methods are employed to increase the size of the training data. Finally, the Mask R-CNN model is used to extract the buildings from the input images.

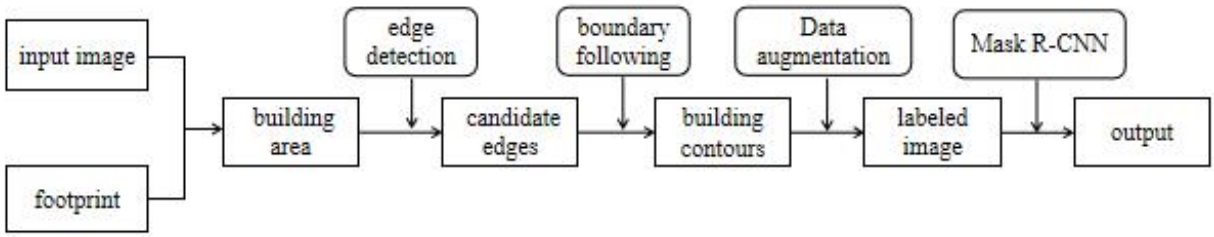


Figure 2.1. Flowchart of the proposed approach for automated building extraction.

2.1 Building boundary detection

This section introduces the proposed techniques for automatic image labeling in this thesis. In deep learning tasks, the quality of the training dataset labeling is directly related to the performance of the training model. However, many misalignments between building footprints and ground truth can occur, as shown in Figure 2.2.

Figure 2.2. Examples of misalignments (yellow polygon) between footprint (red polygon) and the ortho image.

These misalignments are often due to the fact that the image used to digitize the footprint does not correspond to the image being used for analysis (Vargas-Muñoz et al. 2018) (e.g., they are from independent sources or the images are not perfectly orthorectified). Therefore, before labeling the dataset begins, more accurate building contours must be determined. First, the building footprint is applied to extract the candidate buildings from the input image. Then, a Sobel operator is employed to locate the candidate edges of buildings. Next, the image of the edge detection results is transferred to gray and binarized by a threshold. Finally, the boundary following method is used to extract the building contours and to complete the image labeling.

2.1.1 Edge detection

Edge detection aims to detect and locate the edges of an image by the intensity changes. Therefore, the key to accurate edge detection is to find the approximate absolute gradient magnitude at each point of an input grayscale image (Vincent and Folorunso 2009). There are many useful methods of edge detection, such as the Roberts operator, Prewitt operator, Sobel operator, and Canny operator. The Sobel operator is simple and feasible and has a good performance in edge detection on images with a gradual change of grayscale. Although the Canny operator can produce a better result than the Sobel operator, it is time-consuming and difficult to reach a real-time response. Therefore, the Sobel operator was chosen to detect the edges in this thesis.

The Sobel operator is a discrete differential operator that combines Gaussian smoothing and differential derivation. It introduces the concept of weight, which assumes that the distance between adjacent pixels has different effects on the current pixel. The closer a pixel corresponds to the current pixel, the greater the impact, thus achieving image sharpening and highlighting the edge contour. The Sobel operator uses a 3×3 convolution mask in the horizontal and vertical directions. After convolution with the image, the approximate values of the brightness differences in the horizontal and vertical directions are obtained. The two-direction convolution is shown in Figure 2.3.

Gx			*Gy*		
-1	0	+1	-1	-2	-1
-2	0	+2	0	0	0
-1	0	+1	+1	+2	+1

Figure 2.3. Horizontal filter *Gx* and vertical filter *Gy* in the Sobel operator.

The gradient and the direction of the gradient for every pixel is calculated by Equation 2.1 and Equation 2.2.

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.1}$$

$$\theta = arctan\left(\frac{G_y}{G_x}\right) \tag{2.2}$$

An example of a Sobel operation is shown in Figure 2.4. It can be seen that one direction filter is not enough to detect all candidate edges. After combining two direction filters, more accurate contours of buildings are successfully extracted.

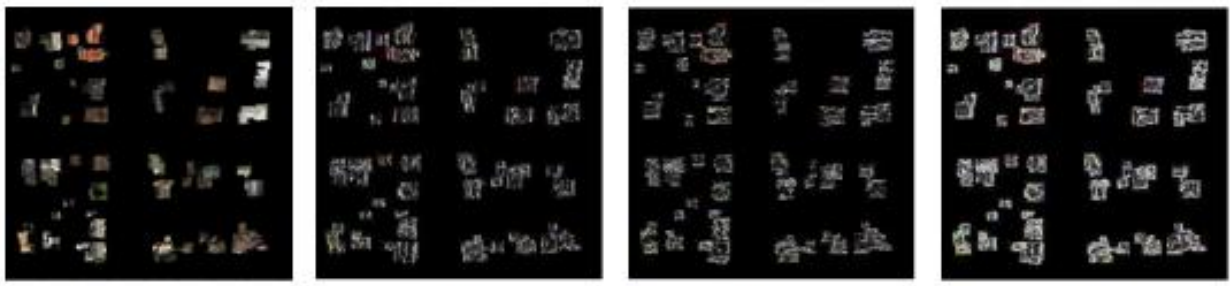


Figure 2.4. Sample results after applying Sobel operator. (a) input image; (b) result after horizontal convolution; (c) result after vertical convolution and (c) final result combined the two direction convolutions.

2.1.2 RGB to gray conversion

Before the binarized image boundary following method is used to extract the building contours, the input color image is transferred to a grayscale image. In image processing, most of the processing methods need to convert the color image to grayscale in advance to perform related calculations and recognition. In an RGB color image, color is formed by mixing the three

primary colors of red (R), green (G), and blue (B) in proportion. The basic unit of an image is a pixel, and a pixel needs three blocks to represent R, G, B. If 8 bits are used to represent a color, then 0-255 can distinguish a certain primary color with different brightness. Grayscale images use black with different saturation to represent each image point, such as 8-bit 0-255 numbers, to indicate the degree of "gray," and each pixel only needs one grayscale value.

The conversion between the RGB values and grayscale is actually the conversion of the human eye's perception of color to brightness. This is a psychological problem which can be represented by the following formula

$$Gray = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{2.3}$$

According to this formula, read the R, G, and B values of each pixel in turn, calculate the gray value (converted to an integer), assign the gray value to the corresponding position of the new image, and all the pixels are transformed once the conversion is complete. Its computational performance is optimized by converting floating-point operations into integer operations and integer operations into bit operations.

2.1.3 Binary image boundary following

After edge detection, the building boundary is extracted by implementing the binary image boundary following method. Suzuki (1985) presented a boundary following algorithm to convert a binary picture into the boundary representation. Below are some basic concepts and definitions related to the algorithm.

1. The 4- (8-) connected case. For two pixels p and q, if q is in the 4- (8-) neighborhood, then these two pixels are 4- (8-) connected.
2. Boundary point. In the 4- (8-) connected case, if a 1-pixel (i, j) has a 0-pixel (p, q) in its 8- (4-) neighborhood, it is called the boundary point.
3. Outer boundary and hole boundary. As shown in Figure 2.5, there are 0-component S_1, S_3 and 1-component S_2. S_1 directly surrounds S_2. Then, the border between S_2 and S_1 is called the outer border. S_2 directly surrounds S_3, and the boundary between S_2 and S_3 is called the hole border.
4. Parent boundary. As shown in Figure 2.5, the boundary between S_1 and S_2 is B_1, the boundary between S_2 and S_3 is B_2, then B_2 is the parent boundary of B_1.

5. NBD and LNBD. In boundary following processing, a unique number is assigned to each new boundary. NBD represents the number of the current boundary, and LNBD represents the number of the last boundary.

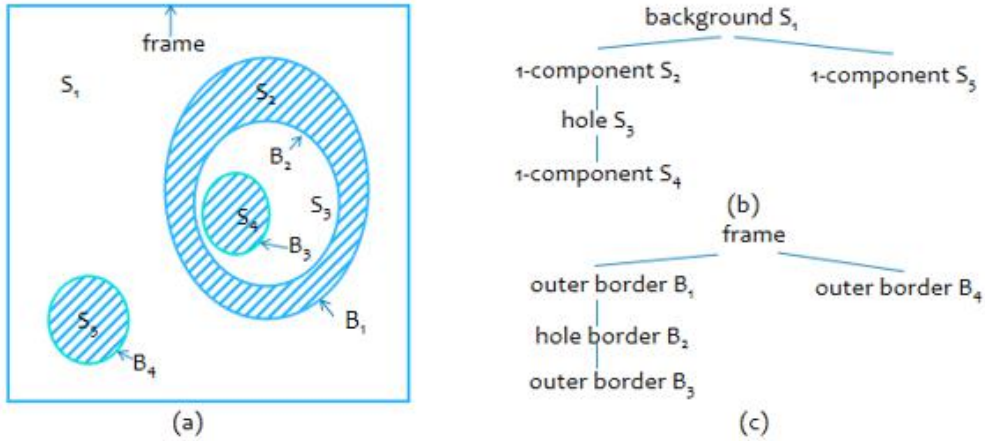


Figure 2.5. Boundary following algorithm. Surroundness figure (a) and relation among connected components (b) and among borders (c).

Table 2.1. Decision rule for the parent boundary of the newly found boundary B

	Outer boundary	Hole boundary
Outer boundary	The parent boundary of the boundary B	The boundary B'
Hole boundary	The boundary B'	The parent boundary of the boundary B'

Assuming that the input image is $F = f(i,j)$, the initial NBD (number of the current boundary) is set to 1 (taking the frame of F as the first boundary). Using a raster scan method to scan an image F: when the gray value at a certain pixel (i,j) satisfies $f_{ij} \neq 0$, it then performs the following steps. Every time the beginning of a new line of the picture is scanned, the LNBD is reset to 1. Below are the steps of the boundary following algorithm:

(1) When $f_{ij} = 1$, if $f_{i,j-1} = 0$, then point (i, j) is the starting point of the outer boundary, NBD+=1$(i_2,j_2) \leftarrow (i,j-1)$; if $f_{ij} \geq 1$, and $f_{i,j+1} = 0$, then point (i,j) is the starting point of the hole boundary, NBD+=1, $(i_2,j_2) \leftarrow (i,j+1)$.

(2) According to the previous boundary B' and the type of boundary B currently encountered, the parent boundary of the current boundary B from Table 2.1 (the first

row represents the type of boundary B' with LNBD and the first column represents the type of boundary B).

(3) Set point (i,j) as the center and (i_2,j_2) as the starting point, the four neighborhoods of the point are searched for the first non-zero pixels. If this point has not been checked in previous steps, the new point is set as the center and (i,j) as the starting point, and this step is repeated until the whole boundary has been detected. Then, raster scanning continues from point $(i,j+1)$. This step ends when the scan reaches the bottom right vertex of the image.

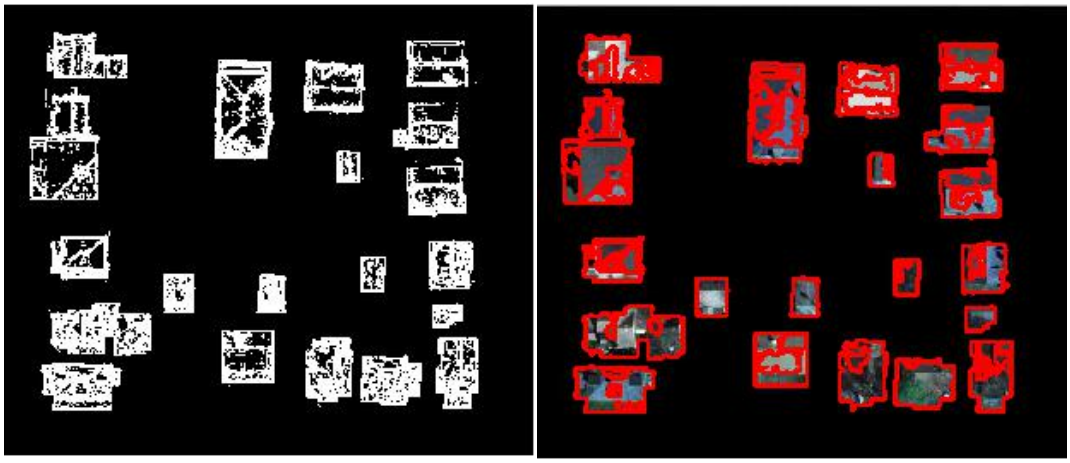


Figure 2.6. An example result of the boundary following algorithm. The left part is the binarized image and the right part is the boundary following results.

An example of the boundary following is shown in Figure 2.6. The left is a binarized image, and the right is the contours extracted by the boundary following algorithm. It is obvious that both the outer boundary and the hole boundary were accurately extracted.

2.2 Data augmentation

In supervised classification, the performance of classification highly depends on the statistical correlation between the training dataset and the test dataset. If the size of the training dataset is small, it is difficult to estimate the distribution of the entire dataset, which may cause overfitting. In order to avoid overfitting due to a small amount of data, data augmentation methods can be applied to enlarge the size of the dataset. Data augmentation techniques are used to enhance the size and quality of training datasets. In deep learning, the larger the scale and the

higher the quality of the data, the better the generalization ability of the model (Shorten and Khoshgoftaar, 2019).

Geometric transformation and color transformation are two major data augmentation approaches. Geometric transformation methods are a set of operations, such as flipping, rotation, and scaling, to perform a geometric transformation of the image. Geometric transformation can be calculated by applying an affine transform to the image. Suppose the coordinates of the original image and the augmented image are (x, y) and $(x^{'},y^{'})$ respectively. Then flipping, rotating, and scaling can be calculated by Equations 2.4 through 2.7, respectively.

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} w \\ h \end{bmatrix} \tag{2.4}$$

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.5}$$

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \tag{2.6}$$

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.7}$$

where w is the width of the image, h is the height of the image; θ is the rotation angle; b_0 and b_1 are the shift of two directions; S_x and S_y are the scale parameter of two directions.

(1) Original image (left), horizontally flipped (middle) and vertically flipped (right).

(2) Original image (left), 15° rotated (middle) and -15° rotated (right).

(3) Original image (left), horizontally shifted (middle) and vertically shifted (right).

(4) Original image (left), 1.3x scaled (middle) and 1.5 scaled (right).

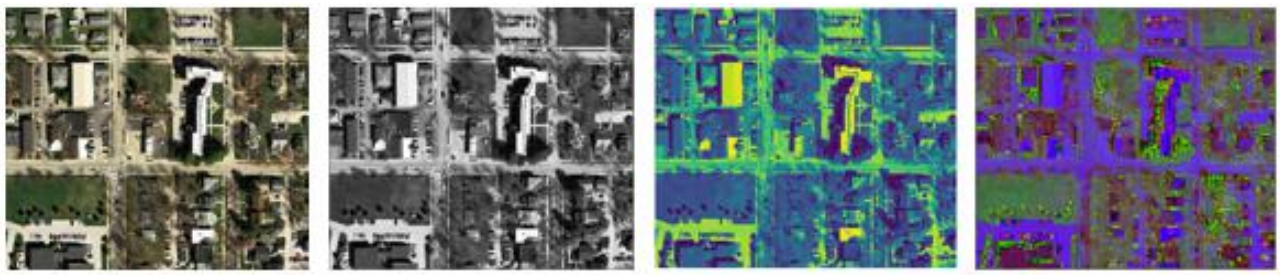
Figure 2.7. Original image and augmentation results after (1) horizontal and vertical flipping; (2) rotating; (3) horizontal and vertical shift and (4) scaling.

The most common approach to color transformation is changing the image's color pattern. Images are usually represented in RGB format, but there are many other color spaces and conversion methods that can be used to transform one color space to another. Each color space differs from the other in terms of the way they present a color. For instance, color in RGB space is represented by a percentage of red, green, and blue hues where HSV represents a combination of hue, saturation, and value. Hue is a general summary of the overall color; saturation is the brilliance and intensity of the color; and the value is the brightness of a color. The most widely used color transformations are RGB to Gray, RGB to BGR, and RGB to HSV. RGB to Gray transform can be computed by Equation 2.3, However, RGB to HSV is slightly more complicated than RGB to Gray. Three components of HSV can be calculated by Equations 2.8 through 2.10.

$$h = \begin{cases} undefined, & if\ max = min \\ 60^{\circ} \times \frac{G-B}{max-min} + 0^{\circ}, & if\ max = R, G \geq B \\ 60^{\circ} \times \frac{G-B}{max-min} + 360^{\circ}, & if\ max = R, G < B \\ 60^{\circ} \times \frac{B-R}{max-min} + 120^{\circ}, & if\ max = G \\ 60^{\circ} \times \frac{R-G}{max-min} + 240^{\circ}, & if\ max = B \end{cases} \tag{2.8}$$

$$f(x) = \begin{cases} 0, & if\ \max = 0 \\ 1 - \frac{min}{max}, & otherwise \end{cases} \tag{2.9}$$

$$v = max(R,G,B) \tag{2.10}$$



(a) (b) (c) (d)

Figure 2.8. Original image and its color augmentations. Original image (a) and augmentation results after (b) RGB to Gray; (c) RGB to BGR and (d) RGB to HSV.

By applying the above data augmentation methods, 30 new images were generated from each labeled image. Samples of the data augmentation results are shown in Figure 2.7 and Figure 2.8.

2.3 Mask R-CNN based building extraction

This section introduces the Mask R-CNN model used for building extraction. The Mask R-CNN framework shown in Fig. 2.9 consists of three parts. First, the backbone network extracts feature from the input image. Second, the output feature maps are sent to the RPN to generate RoIs. After the RoIs are aligned, they are mapped to extract the corresponding object features in the feature maps. Third, the fixed feature maps of the RoIs are sent to the FCN and the fully connected layers for instance segmentation and object classification, respectively.

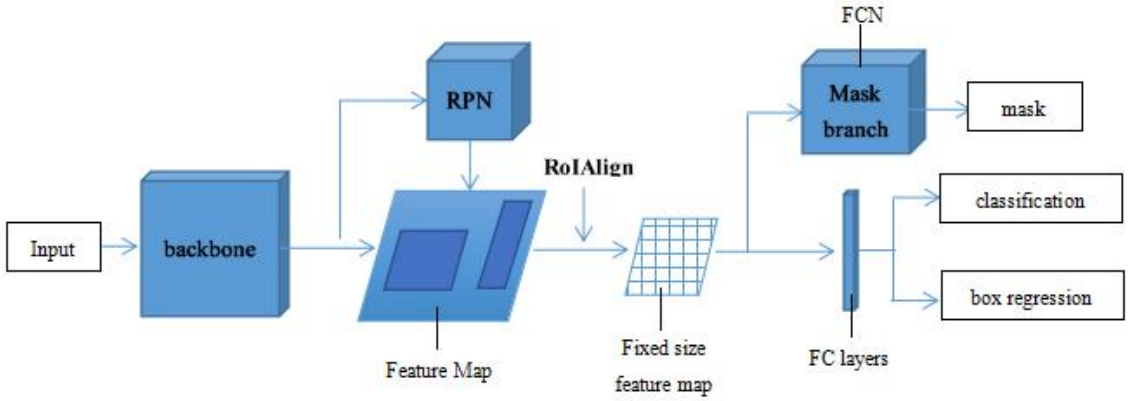


Figure 2.9. Workflow of the Mask R-CNN.

2.3.1 Backbone network

The backbone network is a standard deep convolutional neural network for feature extraction. The early layers extract the low-level features, such as edges and corners; and the later layers detect the high-level features that describe the target categories. Although deeper networks can achieve higher accuracy, the speed of training and detection will be reduced. The residual structure of ResNet does not generate additional parameters in backpropagation. However, it can effectively solve the problem of gradient disappearance (Targ et al. 2016). To a large extent, the residual network makes it easier to optimize deeper models. Therefore, ResNet is used as the backbone network in this thesis. Structure of a residual block is shown in Figure 2.10.

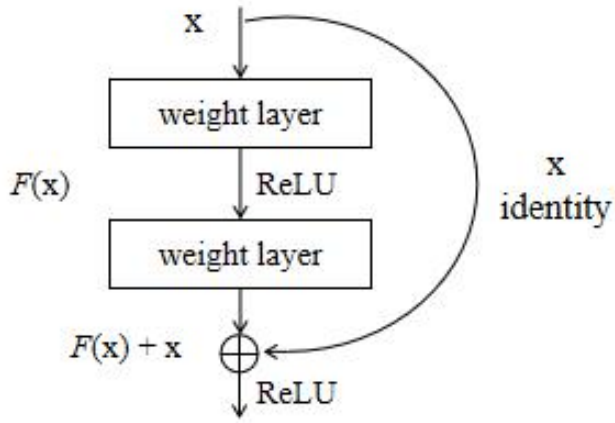


Figure 2.10. Structure of a residual block (adapted from He et al. 2016).

A residual block can be represented by Equation 2.12 and Equation 2.13.

$$y_l = h(x_l) + F(x_l, W_l) \tag{2.11}$$

$$x_{l+1} = f(y_l) \tag{2.12}$$

Where x_l and x_{l+1} represent the input and the output of layer l, F is the residual function, h is direct mapping function, f is ReLU activation function, W_l is the weight of layer l. Then we can obtain Equation 2.14

$$x_{l+1} = x_l + F(x_l, W_l) \tag{2.13}$$

For a deeper layer L, its relationship with layer l can be expressed as Equation 2.15.

$$x_L = x_l + \sum_{i=1}^{L-1} F(x_i, W_i) \tag{2.14}$$

According to the chain rule of backpropagation, the gradient of the loss function with respect to x_l can be expressed as Equation 2.15. During the training process, $\frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i, W_i)$ cannot always be -1. Therefore, there is no gradient disappearance in ResNet.

$$\frac{\partial loss}{\partial x_l} = \frac{\partial loss}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial loss}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i, W_i)\right) \tag{2.15}$$

In order to better present the building target on multi-scales, especially for small buildings, an FPN is used to extend the backbone network. The FPN structure includes three pathways: 1) bottom-up, 2) top-down, and 3) horizontal connection. As shown in Figure 2.11, this structure can merge the features of each level so that it has both strong semantic information and strong spatial information.

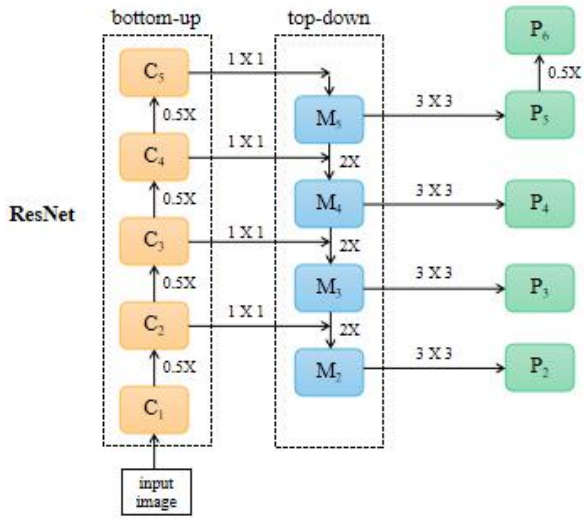


Figure 2.11. Structure of FPN. The left part is the output of bottom-up path, the middle part is the output of top-down path and the right part is the output of horizontal connection path.

The bottom-up pathway is a simple feature extraction process. In general, ResNet is divided into five stages according to the size of the feature map. The output of the last four stages is defined as [C_1, C_2, C_3, C_4, C_5]. The top-down pathway is an up-sampling process that starts from the highest layer. Instead of using deconvolution, the nearest neighbor up-sampling is used in this thesis since it is simple and can reduce the number of training parameters. A horizontal connection is used to fuse the results of the up-sampling with the feature map of the same size generated from the bottom-up pathway. A 1*1 convolution is applied for each layer in [C_1, C_2, C_3, C_4, C_5] and add it with [M_5, M_4, M_3, M_2, M_1]. In order to eliminate the aliasing effect of up-sampling, a 3*3 convolution kernel is used to process the fused feature map, and the final outputted feature map is [P_1, P_2, P_3, P_4, P_5]. Notably, P_1 is not used in our thesis due to the computational expense to calculate the feature map corresponding to C_1. Instead, P_6 is obtained and used to replace P_1 by down-sampling P_5. The specific correspondence of the feature map is shown in Equation 2.16 (Qiao et al. 2019).

$$\begin{cases} P_2 = conv(sum(upsample(P_3, conv(C_2)))) \\ P_3 = conv(sum(upsample(P_4, conv(C_3)))) \\ P_4 = conv(sum(upsample(P_5, conv(C_4)))) \\ P_5 = conv(conv(C_5)) \\ P_6 = downsample(P_5) \end{cases} \tag{2.16}$$

By doing so, the FPN can take advantage of both high-resolution feature maps and high-level semantic information for accurate localization. An example of output feature maps is shown in Figure 2.12.

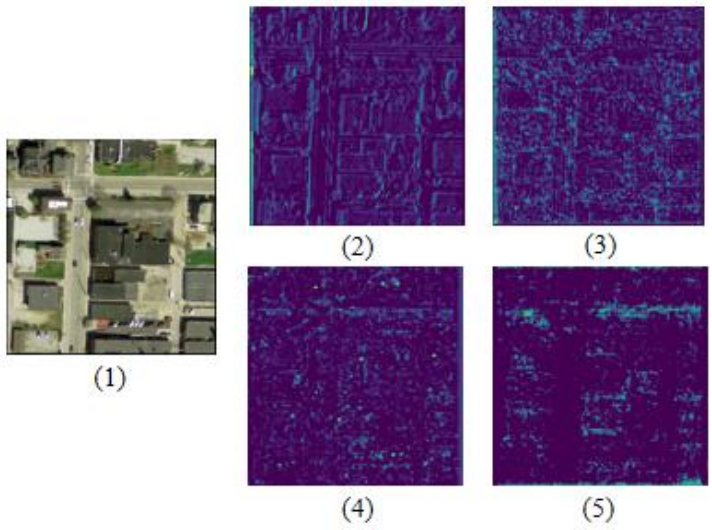


Figure 2.12. Sample result of feature maps generated from FPN. Input image (1) and output feature map of [P_2, P_3, P_4, P_5] (2-5).

2.3.2 RPN

The feature maps generated from the backbone network are sent to Regional Proposal Network (RPN) to generate the region of interests (RoIs). An RPN is a fully convolutional network that simultaneously predicts object bounding boxes. The RPN scans the feature maps and generates three kinds of areas {64, 128, 256} × {64, 128, 256} and three kinds of shapes {1:1, 1:2, 2:1} of rectangular boxes, which are here called anchors or proposals in every pixel position. Then, the IoU of the anchors with the ground truth objects by Equation 2.17.

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{2.17}$$

where B_p represents the bounding box of the anchor and B_{gt} represents the bounding box of the ground truth. Positive anchors are those that have an IoU >= 0.7 with the ground truth object, and negative anchors are those that do not cover the object by more than 0.3 IoU.

However, a positive anchor might not be centered perfectly over the object. So, the RPN estimates a delta (% change in x, y, width, height) to refine the anchor box to fit the object better. This process is called bounding box refinement (BBR) (Pinheiro et al. 2016). After BBR,

the optimal bounding box of the candidate object is found. In the process of target detection, a large number of candidate anchors are generated at the same target position. These candidate anchors may overlap each other. In this case, non-maximum suppression (NMS) (Hosang et al. 2017) is used to find the best target bounding box and to eliminate redundancy. Each object will only have one RoI after BBR and NMS. The steps of NMS are as follows:

(1) Sort all the anchors according to their confidence score within the target bounding box.

(2) Select the bounding box with the highest confidence score to add to the final output list and remove it from the bounding box list.

(3) Calculate the IoUs of the highest confidence bounding box with other candidate frames, and delete the bounding box whose IoU is greater than the threshold.

(4) Repeat the above process until the bounding box list is empty.

After these BRR and NMS, RoIs are obtained. Figure 2.13 illustrates a few sample RoIs generated from the RPN.

Figure 2.13. Sample results of final RoIs. Bounding box before (dashed polygons) and after BBR and NMS (solid polygons).

As mentioned above, FPN produces a feature pyramid [P_2, P_3, P_4, P_5, P_6] instead of one single feature map. To select the feature map of the most suitable scale to cut the RoIs, Equation 2.18 is applied.

$$k = \lfloor k_0 + log_2(\sqrt{wh}/500) \rfloor \tag{2.18}$$

500 is the size of the input image, k is the feature level of the RoI with the size $w \times h$, and k_0 represents the feature level of RoI with the size 500×500. $k_0 = 4$, which means that the feature

map of an RoI with the size $w \times h$ are cut from P_4. Suppose the size of RoI is 250×250, $k = k_0 - 1 = 4 - 1 = 3$, which means generating this RoI from a higher resolution feature map P_3. In addition, k will be rounded in case it is not an integer. Therefore, the large RoI should be cut from the low-resolution feature map, and the small RoI should be cut from the high-resolution feature map.

2.3.3 RoI Align

RoI Align is a regional feature aggregation method used to solve the problem of misalignment caused by the RoI pooling operation. In the commonly used RoI pooling operation, there are two quantification processes, from an image coordinate to a feature map coordinate and from a feature map coordinate to a fixed RoI feature coordinate. Since the position of the preselected box is usually obtained by regression of the model, it is generally a floating-point number, and the pooled feature map requires a fixed size. Therefore, after the above two quantifications, the candidate frame at this time has a certain deviation from the position that is initially returned, and this deviation affects the accuracy of detection or segmentation, which is called misalignment (Jaderberg et al. 2015). In order to solve the above shortcomings of RoI pooling, He et al. (2017) proposed an improved method of RoI Align. The idea of RoI Align is shown in Figure 2.14. It cancels the quantization operation and uses a bilinear interpolation (Mastylo 2004) to obtain the image values at the pixels. The procedure consists of the following steps:

(1) Traverse each candidate region, keep the floating-point boundaries unquantized.

(2) Divide the candidate region into $k \times k$ units, keep the boundaries of each unit unquantized.

(3) Calculate four fixed coordinate positions in each unit and calculate the values of these four positions by bilinear interpolation, and then perform the maximum pooling operation.

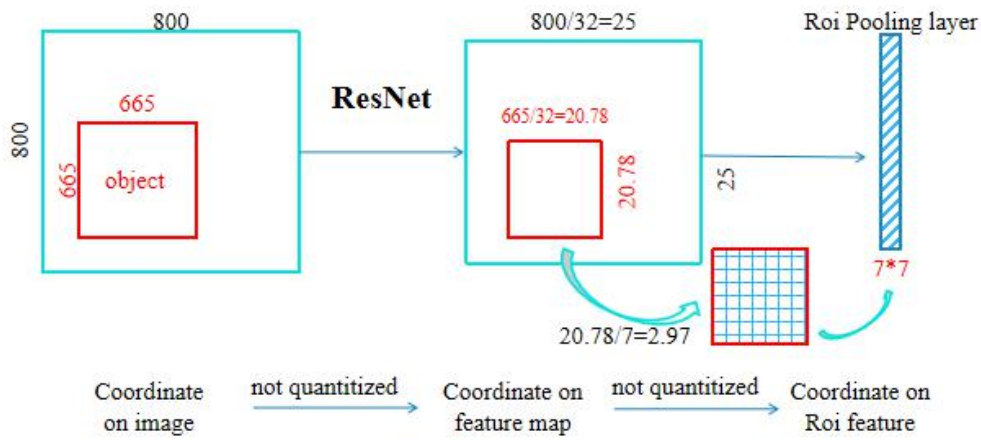


Figure 2.14. Operations in RoI Align.

An example of RoI Align is shown in Figure 2.14. Suppose the size of the input image is 800×800, and there is 665×665 RoI on the image. Since the stride of the backbone network is 32, the side length of the image and RoI are both 1/32 of the input. Instead of quantifying the side length of RoI as 20 in RoI Pooling, the decimal part is kept. Next, the features are pooled to a 7×7 layer, so the above RoI is divided into 7×7 rectangular areas evenly. Each rectangular bin is divided it into four equal parts and the bilinear interpolation method is used to calculate the pixel values at the center point of each part. An example of bilinear interpolation in RoI Align is shown in Figure 2.15. Finally, the maximum of the four-pixel values is taken as the pixel value of this small rectangular area; and this step is repeated in the entire layer until the final feature map of RoI is completed.

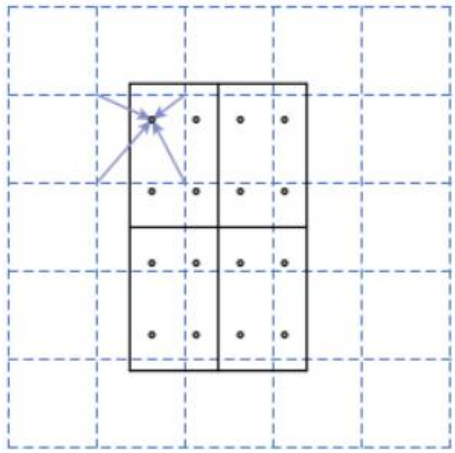


Figure 2.15. A feature map with 5×5 bin is mapped to an RoI of 2×2 bins, the dots represent the 4 sampling points in each bin (adapted from He et al. 2017).

The bilinear interpolation shown in Figure 2.16 is computed as follows. Suppose there are four points $Q_{11} = f(x_1,y_1)$, $Q_{12} = f(x_1,y_2)$, $Q_{21} = f(x_2,y_1)$, $Q_{22} = f(x_2,y_2)$, we want to obtain the value $f(P)$ at the point $P = (x,y)$. First, perform linear interpolation on the y-axis to obtain R_1 and R_2 by Equation 2.19 and Equation 2.20.

$$f(R_1) = f(x,y_1) = \frac{x_2-x}{x_2-x_1} f(Q_{11}) + \frac{x-x_1}{x_2-x_1} f(Q_{21}) \tag{2.19}$$

$$f(R_2) = f(x,y_2) = \frac{x_2-x}{x_2-x_1} f(Q_{12}) + \frac{x-x_1}{x_2-x_1} f(Q_{22}) \tag{2.20}$$

Then perform linear interpolation on the x-axis according to R_1 and R_2 and obtain the value $f(P)$ by Equation 2.21.

$$f(P) = \frac{y_2-y}{y_2-y_1} f(x,y_1) + \frac{y-y_1}{y_2-y_1} f(x,y_2) \tag{2.21}$$

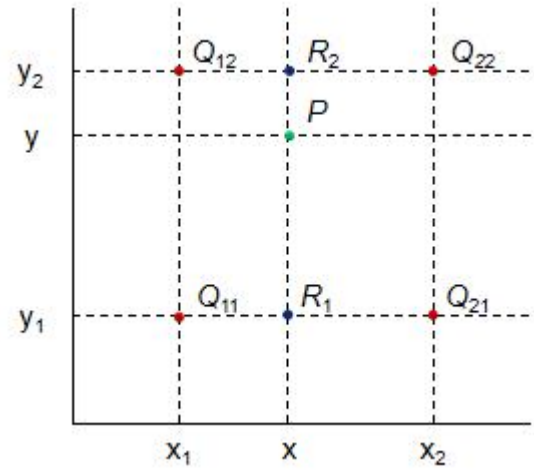


Figure 2.16. Illustration of bilinear interpolation in RoI align.

2.3.4 FC/FCN and loss function

The fixed size feature map output from RoI Align is sent to the fully connected network (FCN) and the FC layer. The FC layer contains two branches, one for classification prediction and the other for bounding box regression. The FCN is used for instance segmentation to generate the target mask.

During the network training, the loss function indicates the difference between the predicted values and the ground truth. For the Mask R-CNN-based building detection network in this

thesis, a multi-task loss function is used to train bounding box regression, classification, and mask segmentation. The regression loss and the classification loss are identical to those in Faster R-CNN, and the mask loss is an average binary cross-entropy loss. The loss function used is as follows:

$$L = L_{bbox} + L_{cls} + L_{mask} \tag{2.23}$$

where L_{bbox} is the bounding box regression loss; L_{cls} is the classification loss; L_{mask} is the mask loss.

The equation for bounding box regression loss L_{bbox} is computed by:

$$L_{bbox} = \frac{1}{N_{reg}} \sum_i p_i^* R\left(t_i, t_i^*\right) \tag{2.24}$$

where N_{reg} is the pixel count in the future map, t_i is the translation scaling parameters from the positive sample RoIs to the predicted region; t_i^* is the translation scaling parameter from positive sample RoIs to the real label; and $R(\cdot)$ is a smooth function.

The equation for classification loss L_{cls} is computed by:

$$L_{cls} = \sum_i - log\left[p_i^* p_i + (1 - p_i^*)(1 - p_i)\right] \tag{2.25}$$

where p_i is the probability that the *i*-th RoIs are predicted as positive samples. When the RoIs are positive, p_i^* = 1; otherwise, p_i^* = 0.

The equation for mask segmentation loss L_{mask} is computed by:

$$L_{mask} = \frac{1}{m^2} \sum_{1 \le i,j \le m} \left[y_{ij}^* - (1 - y_{ij})\, log\,(1 - y_{ij}^2)\right] \tag{2.26}$$

where y_{ij} is the label value of the coordinate point (i, j) in the $m \times m$ region and y_{ij}^* is the predicted value at that point.

2.3.5 Parameters of the Mask R-CNN

In this chapter, we will introduce the important parameters used in the different modules of the Mask R-CNN.

A backbone network aims to extract features of the input image. Since FPN takes advantage of both high-level semantic information and high-resolution feature maps for accurate localization, an FPN based on a ResNet50 network is applied in this thesis to achieve gains both in accuracy and speed. In order to generate feature maps with different resolutions, FPN uses

different strides in each layer, and then those feature maps are fed into an RPN to generate RoIs. The important parameters in the backbone network are shown in Table 2.2.

Table 2.2. Parameters for backbone network

Parameter	Parameter description	Value
Backbone	Backbone network architecture Supported resnet50, resnet101	Resnet50
Backbone strides	The strides of each layer of the FPN pyramid	[4, 8, 16, 32, 64]
FC layer size	Size of the fully-connected layers in the classification graph	1024
Top-down pyramid size	Size of the top-down layers used to build the feature pyramid	256
Num classes	Number of classification classes (including background)	2
Image channel size	Number of color channels per image RGB=3, Grayscale=1, RGB-D=4	3
Image resize mode	Resize mode for training and predicting	square
Image min dim	Minimal image size after resizing	800
Image max dim	Maximum image size after resizing	1024

The feature maps generated from the backbone network are used as input for the RPN network. Anchors with three different scales and three different aspect ratios are employed to generate positive anchors. Then BBR and NMS are applied to refine the bounding boxes of the anchors and to remove the redundant anchors. These anchors are called RoIs and are sent to the RoI Align layer. The parameters of the RPN are shown in Table 2.3.

Table 2.3. Parameters for Region Proposal Network (RPN)

Parameter	Parameter description	Value
RPN anchor scales	Length of square anchor side in pixels	[32, 64, 128, 256, 512]
RPN anchor ratios	Ratios of anchors at each cell (width/height)	[0.5, 1, 2]
RPN anchor stride	Anchor stride	1
Max GT instances	Maximum number of ground truth instances to use per image	100
RPN box std dev	Bounding box refinement standard deviation for RPN	[0.1, 0.1, 0.2, 0.2]
RPN NMS threshold	Non-maximum suppression to filter RPN proposals	0.7
RPN anchors per image	Number of anchors to use per image for RPN	256
RPN NMS limit	RoIs kept before non-maximum suppression	6000
Post NMS RoIs training	RoIs kept after non-maximum suppression (training)	2000
Post NMS RoIs inference	RoIs kept after non-maximum suppression (inference)	1000
Use mini mask	Resize instance masks to a smaller size to reduce memory load.	True
Mini mask shape	(height, width) of mini mask	(56, 56)

After the pooling operation, the RoIs are fed into fully-connected layers for building classification, box regression, and mask segmentation. Although the generated masks with 28*28 pixels size are low resolution, they are represented by floating numbers, which hold more details than binary masks. The small size of the mask also helps keep the mask branch light. The ground truth masks are downsampled to 28*28 to compute the loss in the training process and the predicted masks are upsampled to the size of the RoI to generate the final masks in the mask

segmentation process. The parameters for detection and mask segmentation are shown in Table 2.4.

Table 2.4. Parameters for fully connected network

Parameter	Parameter description	Value
Train RoIs per image	Number of RoIs per image to fed to classifier/mask head	200
RoI positive ratio	Percent of positive RoIs used to train classifier/mask head	0.33
Pool size	Size of pooled RoIs	7
Mask pool size	Size of pooled RoIs fed to mask head	14
Mask shape	Shape of output mask	[28, 28]
Bbox std dev	Bounding box standard deviation for final detection	[01, 0.1, 0.2, 0.2]
Detection max instances	Max number of final detections	100
Detection NMS threshold	Minimum probability value to accept a detected instance	0.7

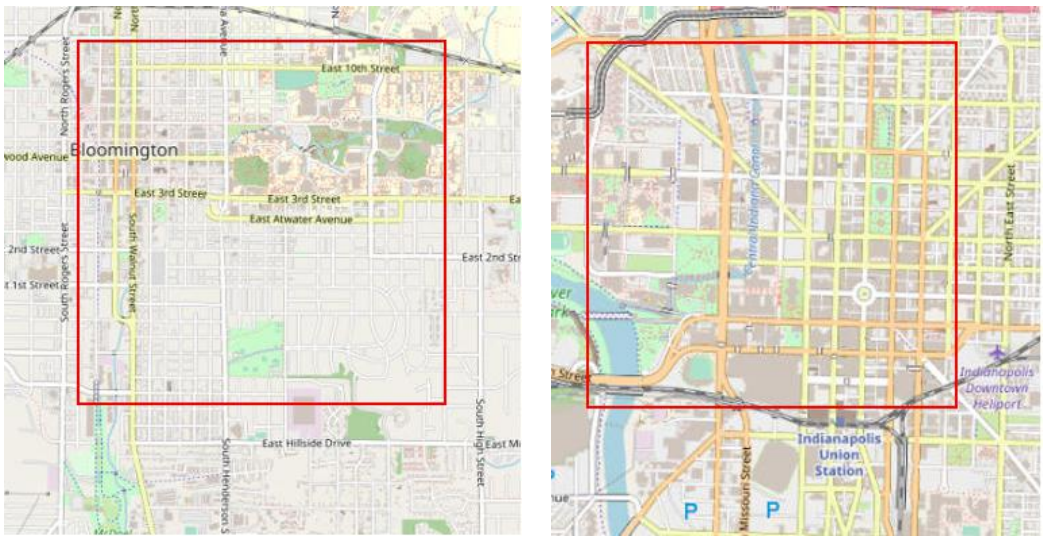
During the training process, the total training epochs are set to 50 and each epoch contains 100 iterations. The learning rate is 0.00, which is adjusted per 10 epochs with an adjustment factor of 0.9. During the detection process, detection will be skipped if its confidence is less than 90%.

Table 2.5. Parameters for model training

Parameter	Parameter description	Value
GPU count	Number of GPUs to use	16
Images per GPU	Number of images to train on each GPU	2
Num epochs	Number of training epochs	50
Steps per epoch	Number of training steps (iterations) per epoch	100
Validation steps	Number of validation steps to run at the end of every training epoch	50
Learning rate	Learning rate	0.001
Learning momentum	Learning momentum	0.9
Weight decay	Weight decay regularization	0.0001
Detection min confidence	Skip detection with < 90% confidence	0.9
Gradient clip norm	Gradient normal clipping	5

3. EXPERIMENT DATA

The proposed building extraction method requires high-resolution aerial images and corresponding building footprints. The Indiana Geographic Information Office provides high-resolution aerial images of the whole state of Indiana, and OpenStreetMap provides building footprint data for many cities in the U.S. In this thesis, the study areas are the urban area of Bloomington and Indianapolis, which are shown in Figure 3.1. The size of each study area is $3000m \times 3000m$.



(a) (b)

Figure 3.1. Map of the study areas (red polygon) in Bloomington (left) and Indianapolis (right) with scale 1: 80,000.

3.1 Orthopohoto data

In 2016, the Indiana Geographic Information Office began a three-year (2016-2018) update project to refresh the orthophotography (RGBI) data for the entire state of Indiana. Two of the experimental images in this thesis are shown in the left part of Figure 3.2 and Figure 3.3, which show urban areas of Bloomington and Indianapolis, respectively. The orthophoto was based on 2016 acquisition with a 0.4m resolution. Each dataset contains 9 orthoimages, and the size of each orthoimage is $1000m \times 1000m$.

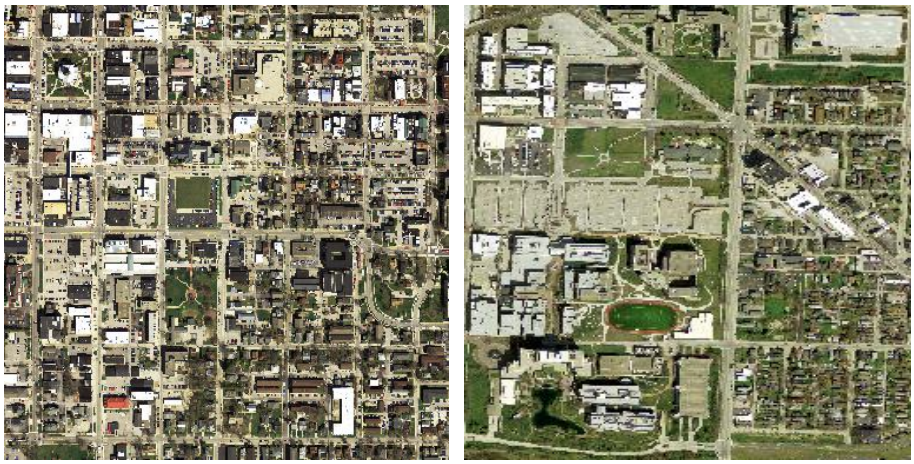


Figure 3.2. One sample orthoimage in Bloomington (left) and Indianapolis (right).

To produce small tiles as experiment data, fishnet is used to divide the original orthoimages into 144 (each original image is evenly divided into 16 tiles, size of each tile is $250m \times 250m$) tiles of images, which are saved in JPG format. Some tiles that contain zero or few buildings are removed, the number of rest tiles is 120. These 120 tiles are used as follows: 80 as training data, 20 as validation data, and 20 as test data. The image size of the training tile is 500×500. The study area in Bloomington has more buildings, but the average building size is smaller than those in Indianapolis. The number of buildings and the average area of the buildings were calculated for the two datasets based on the building footprints. The number of buildings and average pixel size per building in the two datasets are shown in Table 3.1.

3.2 Building footprint data

The building footprint GIS data shapefile for Bloomington is downloaded at http://www.data.gov. The shapefile was last updated on October 31, 2016 and covers the entire Monroe County area in Indiana. These building footprint data are in Esri shapefile format and are obtained in a zip file. Similar to how the orthophotograph data are handled, the footprint is divided into small pieces and every effort is made to assure they are consistent with the size and location of the orthophotograph on each image. The whole footprint image of Bloomington is shown on the left side of Figure 3.3. The building footprint of Indianapolis was downloaded from Microsoft building footprint data that was last updated in March 2017; however, it was digitized from images captured in 2014 and 2015. Since there was a gap in the production dates

between the orthoimage and the building footprints, some missing buildings can be detected in the footprint map by the human eye. The whole footprint image of Bloomington is shown on the right side of Figure 3.3. The building footprint of one sample image in the two datasets is shown in Figure 3.4.

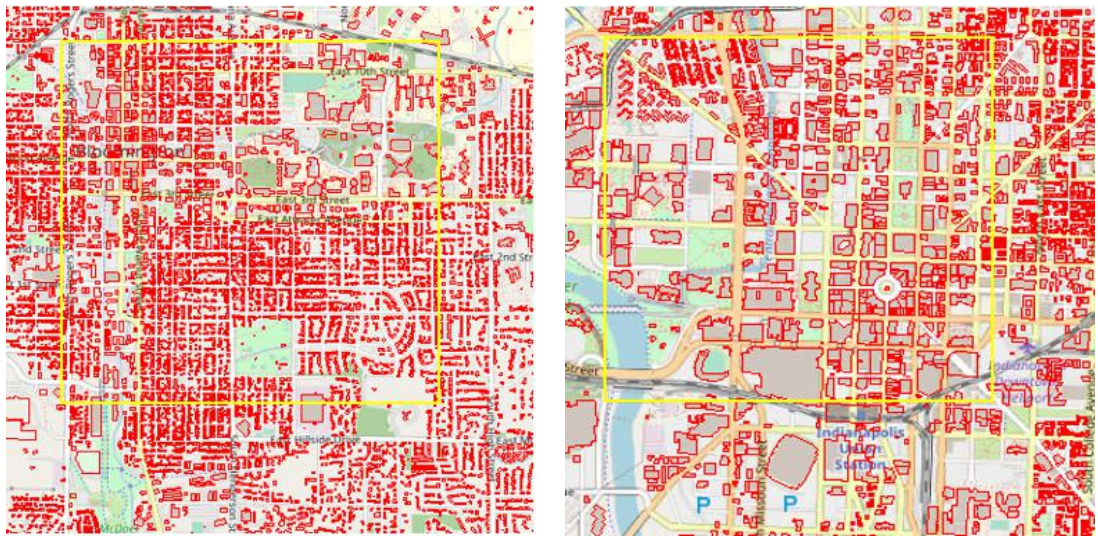


Figure 3.3. Building footprint of the study area (yellow polygon) in Bloomington (left) and Indianapolis (right) with scale 1: 80,000. The image is 250m by 250m on the ground.

Figure 3.4. Building footprint of sample image in Bloomington (left) and Indianapolis (right). New buildings that are not on the map are shown in the yellow polygon.

Table 3.1. Number of buildings and average area of the two datasets.

	Pixel size	# of buildings	Average pixel size per building
Bloomington	0.4m	4494	1029
Indianapolis	0.4m	1532	3015

3.3 Ground truth data

The ground truth data were obtained by manual labeling through the VGG Image Annotator (VIA) tool. Since manual labeling results depend on human visual interpretation, the shadows and occluded areas of the building were excluded. The number of ground truths in Bloomington and Indianapolis were 4,494 and 1762, respectively. An example of a manually-labeled result in two datasets is shown in Figure 3.5. These ground truths were subsequently used for the evaluation of automatic image labeling results and building extraction results.

Figure 3.5. One sample of manually labeled result in Bloomington (left) and Indianapolis (right).

4. RESULTS AND EVALUATION

The labeling results of the proposed automatic image labeling method are presented in this chapter utilizing the two experiment datasets in Chapter 3. The quality of the automatically-labeled mask was evaluated by an IoU between the automatically-labeled mask and the manually labeled ground truth. The building extraction results of the Mask R-CNN model are presented in this chapter as well. The performance of building detection was evaluated by precision, recall, and F_1 score, while the mean pixel accuracy (MPA) and the average distance error (ADE) were used to evaluate the performance of mask segmentation.

4.1 Automatic image labeling results

Compared with manual labeling, automatic labeling can be quite complicated and difficult; but with the help of a building footprint, automatic labeling can be relatively simple and fast. The key to automatic labeling is to accurately extract the building outlines from imagery. Edge detection and boundary following can extract the contours of all the objects in the imagery and is not limited to buildings. Thus, building footprints are utilized for the initial extraction of the building areas.

Figure 4.1. Original image (left), building footprint (middle) and selected candidate buildings (right) extracted by building footprint.

Similar to the image progression shown in Figure 4.1, the building footprint is used as the mask to extract the building area from the original image. After the candidate area of a building is obtained, the RGB image is transferred to a grayscale image and binarized before the

boundary following. Due to the different colors and shadow areas in the building roofs, the brightness of the rooftops also indicates obvious differences between different buildings. As can be seen in Figure 4.2, most of the buildings are relatively dark and only a few of buildings are bright. The distribution of the gray values of building pixels shown in Figure 4.3 also indicates the variety of gray values of the buildings.

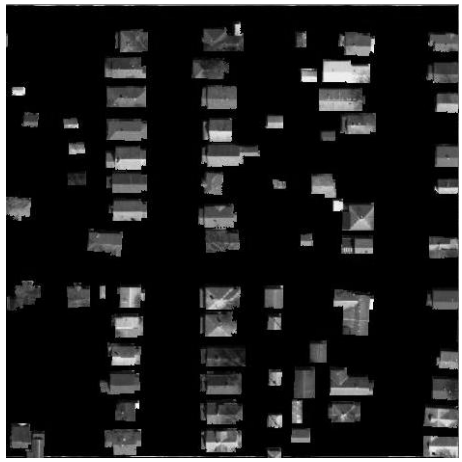


Figure 4.2. One sample grayscale image of initial candidate buildings.

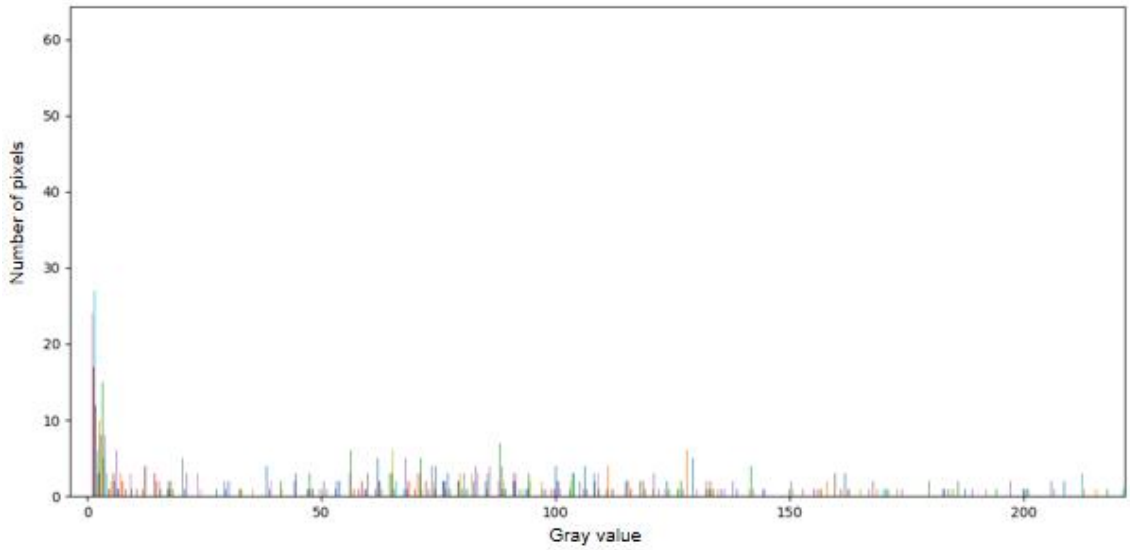


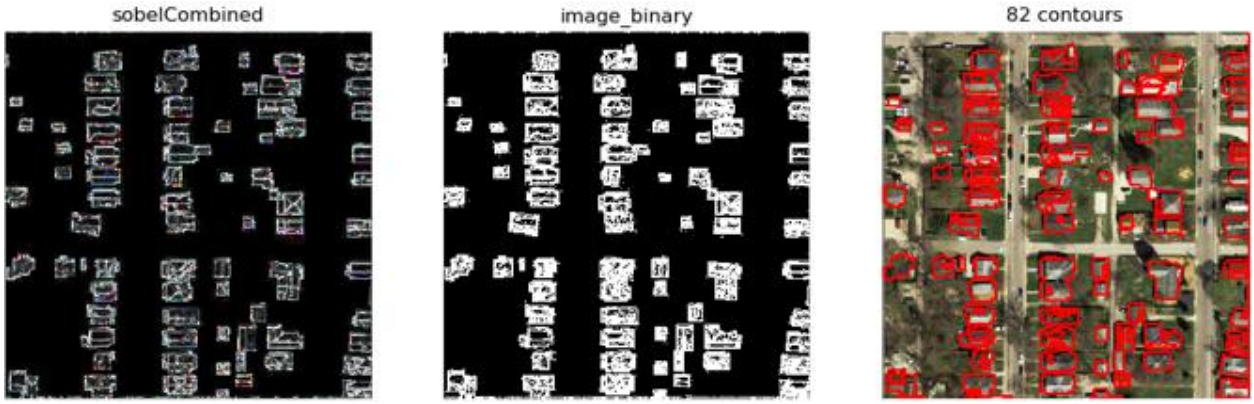
Figure 4.3. Histogram of grayscale value of initial building candidate areas.

Since Figure 4.3 does not show an obvious difference in gray values between the buildings and background, it was impossible to correctly binarize the grayscale image with a threshold. Therefore, an edge detection method is necessary to locate the building edges. Due to its good

performance in edge detection on images with a gradual change of grayscale, the Sobel operator is used to detect the candidate edges of the input image.

Figure 4.4. Edge detection results with Sobel operator.

It can be seen in Figure 4.4 that the candidate edges are successfully detected after convolution in two directions. The purpose of this process is to remove the shadow area of the misalignment. Therefore, a high threshold is used to binarize the image. Then, a boundary following algorithm is applied to determine the contours of the buildings. The extracted contours are shown in Figure 4.5 (c).



(a) (b) (c)

Figure 4.5. Boundary following results after edge detection. (a) edge detection result of Sobel operator (b) binary result of edge candidates (c) building contours of boundary following method.

Figure 4.6 shows that some of the holes within the rooftops also are extracted, which are redundant for building labeling. According to the relationship between boundaries, we can

remove holes and achieve the final building contours. During boundary following, both the label mask of the buildings and the coordinates of the contour points are obtained. The building label masks are used for data augmentation while the coordinates information of the contours is transferred to the annotation JSON file for training the Mask R-CNN model.

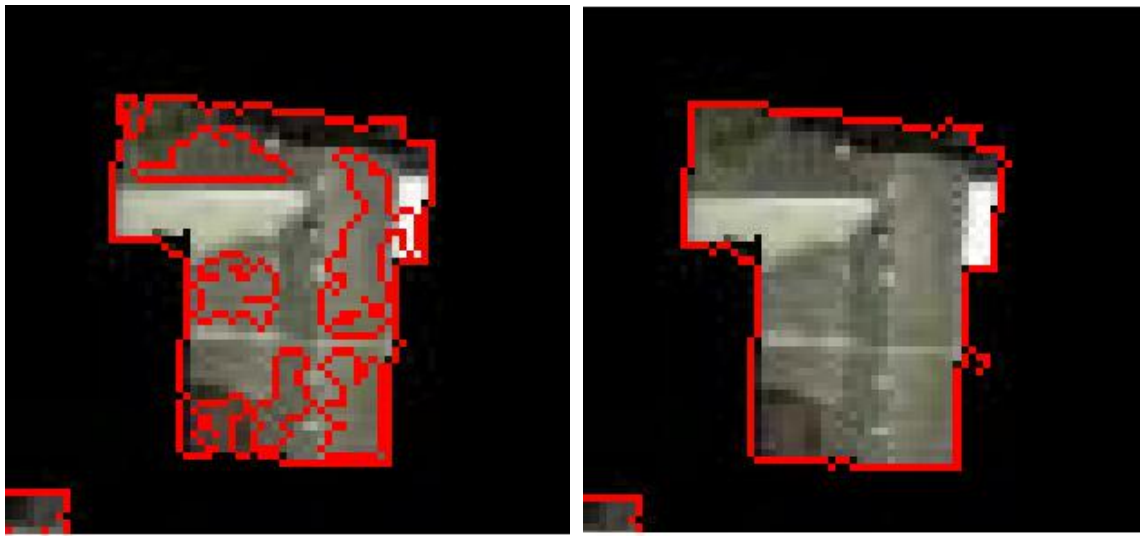


Figure 4.6. Sample result of boundary following algorithm. Building hole boundaries (left) and its contours (right) after cleaning.

The speed of manual labeling generally is about four to five minutes per image. The speed of automatic labeling in this thesis is an amazing five seconds per image.

An IoU between the automatically-labeled mask and the manually-labeled ground truth is computed to evaluate the quality of the labeled mask. A total of 100 tiles from the Bloomington dataset were manually labeled by the VIA tool. The Indianapolis dataset was not used in the IoU because some newly constructed buildings are not shown in its building footprint dataset. The final IoU of Bloomington dataset is 0.95, which indicates a high quality automatically-labeled mask. It can be seen that automatic labeling greatly reduces the data preprocessing time and improved the labeling efficiency, which is particularly important when the experimental data set is large. Examples of final automatic labeling results are shown in Figure 4.7.

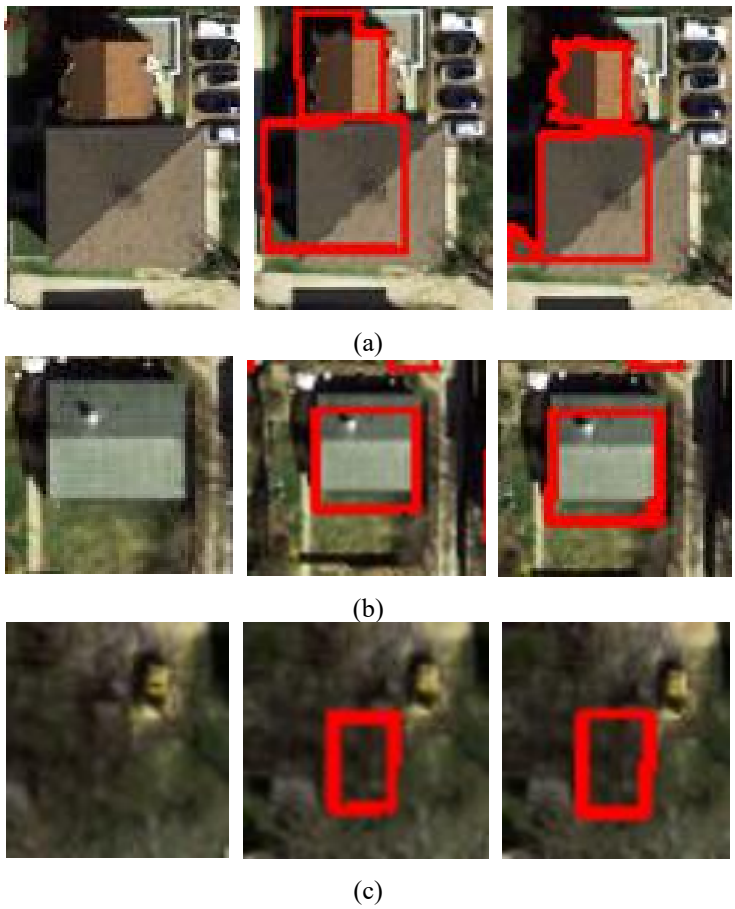
(a)

(b)

(c)

Figure 4.7. Sample results of automatic image labeling. Original image (left) and final labeled mask (right) in Bloomington (a)&(b) and Indianapolis (c).

Some comparison results between the footprint and the labeled mask are shown in Figure 4.8. The three buildings in the figure are shown under different situations: building (a) has a misalignment with a shadow, building (b) has a misalignment with bright vegetation, and building (c) is occluded by trees, which cannot be distinguished by the human eye. The labeled masks indicate that the proposed automatic labeling approach successfully removed the misalignment with shadow. However, for the misalignment with bright vegetation, the Sobel operator could not correctly detect the edge between the buildings and bright vegetation due to the small difference in gradient. It can be seen that the occluded building also shows little difference with its neighborhood. After extracting the candidate building areas with the footprint, the background area was removed and replaced by "0" pixels. Therefore, the gradient between the occluded building and "0" pixels was large enough for the Sobel operator to detect.



(a)

(b)

(c)

Figure 4.8 Comparison between the footprint and the labeled mask. Left part is the original image, middle part is the building contours in building footprint and right part is the labeled mask.

4.2 Mask R-CNN training results

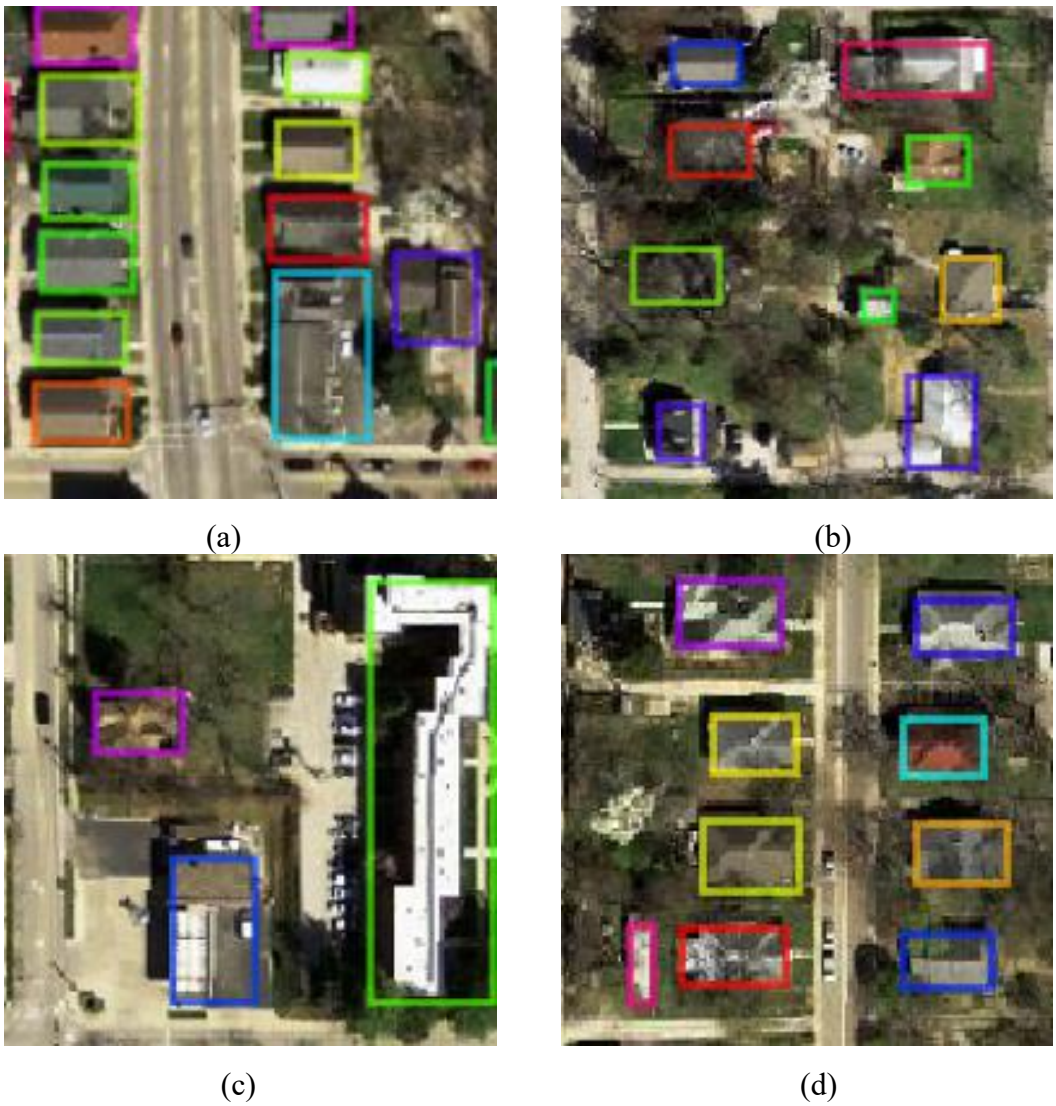
Experiments were conducted on the two datasets collected in Bloomington and Indianapolis. After data augmentation, 6,000 tiles were used for training (80% training dataset and 20% validation dataset). During the experiments, Tensorflow (Abadi et al. 2016) with GPU was used to construct the building extraction model. In terms of the Mask R-CNN model, ResNet50 and FPN were used as the backbone. In addition, the pre-trained weights of the COCO (Lin et al. 2014) dataset also were adopted for the training process. Although the COCO dataset did not contain buildings, as a dataset with 81 classes and more than one million images, part of its weights was still useful for accelerating the training process. The learning rate was 0.001 and was adjusted per 10 epochs (each epoch had 100 iterations) with an adjustment factor of 0.9. Training loss and validation loss are calculated by a combination of classification loss, bounding box loss and mask loss. It took approximately eight hours for 50 epochs of training, at which time the loss function reached a convergence state.

Figure 4.9. Loss function of 6000 tiles in two datasets for the Mask R-CNN training.

From Figure 4.9, it can be seen that the loss function showed a downward trend during training, which indicates that the joint loss decreased gradually by updating the parameters during the optimization process. The loss function value of both the training set and the validation set decreased to less than 0.4 and tended to be stable when the number of epochs was more than 40, which indicates that the training process ran well and the loss function achieved a state of convergence.

4.3 Building detection results

In order to verify the reliability and stability of the trained model, 40 test tiles (20 Bloomington and 20 Indianapolis) are selected for the model evaluation. The masks of the test tiles are manually labeled by the VIA tool. The detection performance of Bloomington and Indianapolis are shown in Figure 4.10 and Figure 4.11, respectively; and all the detection results for the two datasets are listed in Table 4.1.



(a) (b)

(c) (d)

Figure 4.10. Some building detection results of the test data in Bloomington. (a) adherent buildings; (b) occluded buildings; (c) a large building and (d) small buildings.

(a)

(b)

(c)

Figure 4.11. Building detection results in Indianapolis datasets. New built buildings (yellow polygon) in footprint (left) are detected in detection results (right).

6Table 4.1. Building detection results of 40 test tiles

	#Predicted Buildings	#Missing Detection	#False Detection	# Detected New Buildings	# New Buildings
Bloomington	1124	82	33	0	0
Indianapolis	442	15	11	34	41

It can be seen from Figure 4.10 that the detection model performed well in both datasets. Small buildings (Figure 4.10 (d)) and large ones (Figure 4.10 (c)) in different shapes and colors were successfully detected. Some adherent buildings (Figure 4.10 (a)) and occluded ones (Figure 4.10 (b)) were also accurately detected. The detection results shown in Figure 4.11 confirm that the detection model can successfully detect a building not in the building footprint.

For quantitative evaluation, we compared the automatic detection results with the manually-labeled ground truth of buildings. The precision (*P*), recall (*R*), and F_1 score were calculated to evaluate the building detection performance by Equation (4.1).

$$P = \frac{TP}{TP+FP},\ R = \frac{TP}{TP+FN},\ F_1 = \frac{2PR}{P+R} \tag{4.1}$$

where *TP* is the number of cases that are building and detected as building, *FP* is the number of cases that are background but detected as building, and *FN* is the number of cases that are building but detected as background (Yang et al. 2018). In other words, TP represents the correct detection, FP represents the false detection and FN represents the missing detection. The detection results of 40 test tiles showed that the F_1 score of Bloomington and Indianapolis were 0.951 and 0.964, respectively. The detailed results for precision and recall are shown in Table 4.2.

Table 4.2. Precision, recall, and F1 score of 40 test tiles.

	Precision	Recall	F_1
Bloomington, 20 tiles	0.932	0.971	0.951
Indianapolis, 20 tiles	0.966	0.977	0.968

From Table 4.1 and Table 4.2, it can be seen that the detection results for the Indianapolis dataset were better than for the Bloomington dataset. The main reason for this performance difference between the two datasets is that there are more missing detection cases in the

Bloomington dataset than the Indianapolis dataset. Missing detection usually occurs in the case of small buildings. For example, in Figure 4.12 (b) there is a small building that is not detected. This building is only 20% of the size of the other buildings, which is not enough to yield reliable results (Yu et al. 2019). In some cases, missing detection also happens when a building is partially occluded by trees. Since the feature map of the occluded building is a mixture of buildings and trees, this building was more likely to be detected as background. Missing detection also can happen when a slender-shaped building is located at the edge of an image.

Figure 4.12. False detection (a, c) and missing detection (b, d, e, f) cases.

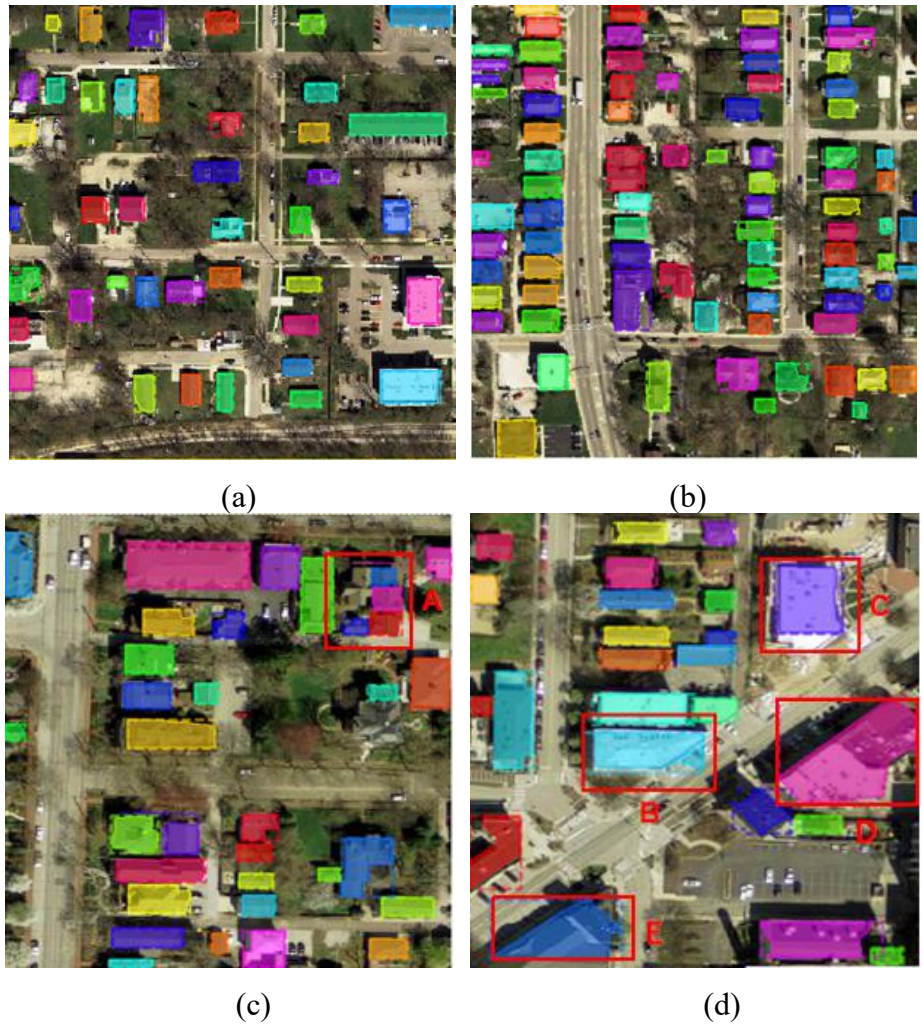
False detection also affects the performance of building detection. Some background objects having similar features with buildings can be detected as buildings. Distinguishing those objects among buildings is difficult, even for the human eye. For example, part of the parking lot in Figure 4.12 (a) and the bare ground in Figure 4.12 (c) are detected as buildings. Fortunately, the number of false detection cases is much less than for the case of missing detection.

4.4 Mask segmentation results

After classification and bounding box regression in the FC layer, the positive RoIs are sent to the FCN to generate a target mask. Each segmentation image is transformed into a binary matrix, where the forehead (buildings) is regarded as "1" and background pixels are regarded as "0". Then, the building contour is extracted since the building contour line is the "1" pixels surrounded by the background pixels. Some of the segmentation results are shown in Figure 4.12, in which (a)&(b) are from the Bloomington dataset and (c)&(d) are from the Indianapolis dataset.

The Mask R-CNN model performed well on mask segmentation for building roofs of different sizes and shapes. Most of the buildings in Figure 4.13 are successfully segmented, especially the buildings with regular shapes and simple textures (Figure 4.13 (a)&(b)). Using FPN, the spatial relationships between multi-level features are obtained, thus adherent buildings are successfully segmented into different objects (Figure 4.13 (c)&(d)). In addition, the tilted buildings are also successfully segmented (e.g., the building D and E in Figure 4.13 (d)). However, there are some misalignments between the segmentation results and the ground truth. For example, Building A in Figure 4.13 (c) is over-segmented to several small segments, and Building C in Figure 4.13 (d) has a smaller segmentation mask than its ground truth. In some cases, the segmentation mask of a building will exceed the contour of its ground truth if it is smaller than the ground truth. For example, the segmentation mask of Building B in Figure 4.13 (d) is larger than the size of its ground truth. More detailed mask segmentation examples of the Mask R-CNN are shown in Figure 4.14. It can be seen that the accuracy of segmentation declined when the complexity of the shape and texture increase. The main reason is that the number of small and regular buildings is greater than the number of large and complicated

buildings, which leads to the feature learning of small and regular buildings is better than large and complicated ones.



(a) (b)

(c) (d)

Figure 4.13. Sample results of building mask segmentation in two datasets. (a)&(b) are results in Bloomington and (c)&(d) are results in Indianapolis.

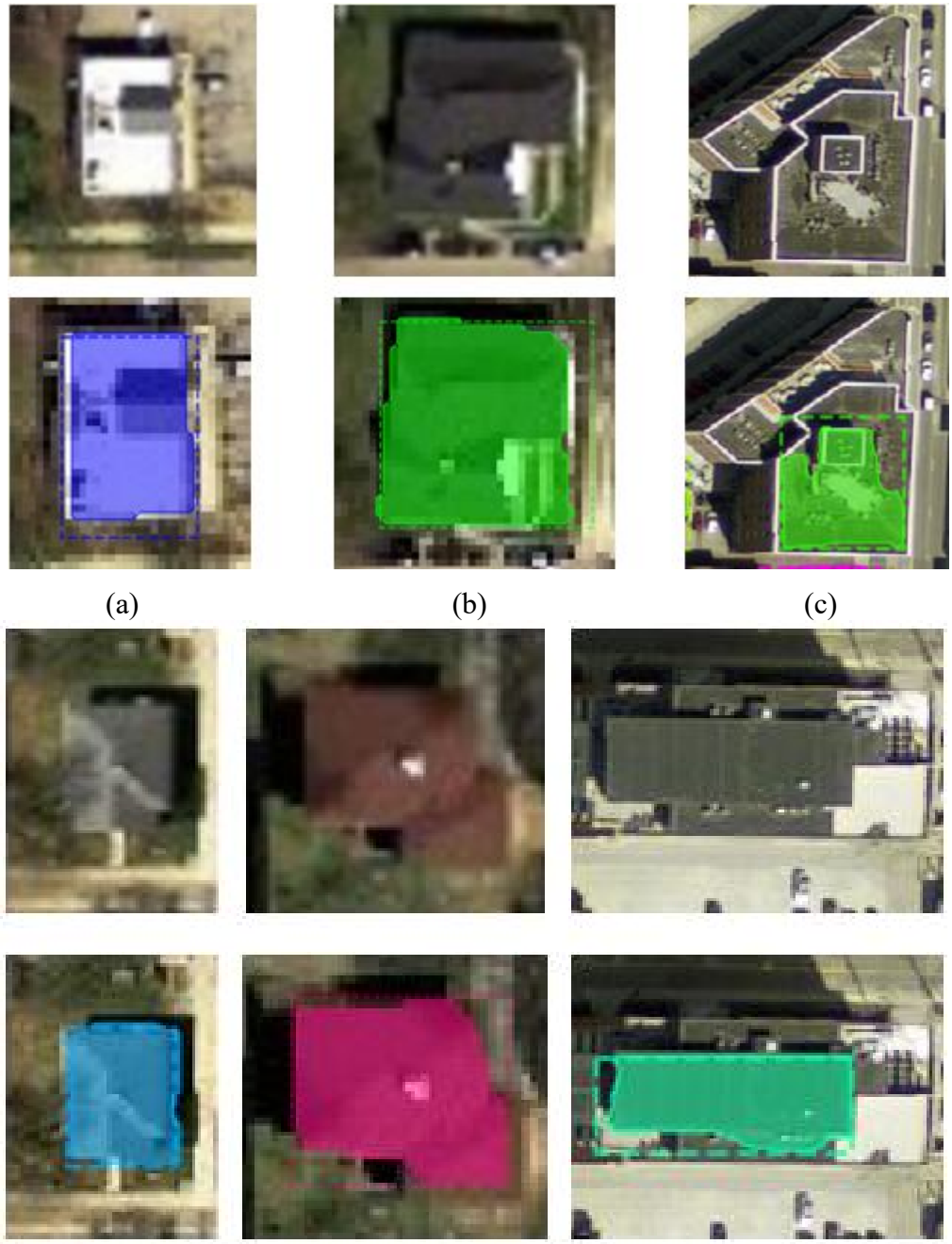


Figure 4.14. Mask segmentation of building in different size and shape.

To quantitatively evaluate the performance of mask segmentation , the MPA and the ADE (Qiao et al. 2006) are applied, respectively. MPA is an important indicator to evaluate image segmentation. It is derived from the correctly segmented pixel with Equation (4.2).

$$MPA = \frac{1}{k}\sum_{i=0}^{k}\frac{p_{ii}}{\sum_{j=0}^{k}p_{ij}} \quad (4.2)$$

where k is the total number of output classes including the background (k = 2 in this experiment), p_{ii} represents the number of pixels correctly classified and p_{ij} represents the number of pixels that belong to class i but misjudged as class j.

For evaluation of the extract contour line, the ADE was computed by Equation (4.3).

$$ADE = \frac{A_{union} - A_{intersection}}{T_{contour}} \quad (4.3)$$

where A_{union} is the union area of the predicted mask and the ground truth, $A_{\text{int}ersection}$ is the intersection area of the predicted mask and the ground truth, and $T_{contours}$ is the pixel number of the ground truth contour line.

Table 4.3. Results of mask segmentation and contour extraction

Study area	# Buildings	MPA	ADE
Bloomington (20 tiles)	1206	0.92	13.17
Indianapolis (20 tiles)	467	0.88	22.64

The results of the MPA and ADE of the two datasets are listed in Table 4.2. It can be seen that the achieved MPA value for the Bloomington dataset is approximate 4% higher than the MPA for the Indianapolis dataset, which illustrates that instance segmentation performs better on buildings that are small in size and have regular shapes. These results also show that the model achieves 13.17 ADE of the extracted contours in Bloomington datasets, which is better than the 22.64 ADE for the Indianapolis dataset, indicating again that the mask segmentation of Mask R-CNN performed better on buildings with regular shapes.

5. CONCLUSION

This thesis proposed a novel automated building extraction method for aerial imagery based on Mask R-CNN. More than 200 tiles of Bloomington and Indianapolis urban areas were collected from the Indiana Map website for this study. An automatic image labeling method first was introduced and evaluated. Using known building footprint, the speed of automatic labeling was reduced to five seconds for a 500×500 pixel image without human interaction. A 0.951 IoU between the labeled mask and the ground truth confirmed the high quality of the labeled mask.

The performance of the proposed approach was discussed in terms of both building detection and mask segmentation. In general, the detection approach performed better on large buildings while small buildings and occluded ones were more likely to be undetected or mislabeled. False detection occurred when the features of the background objects were similar to the features of the buildings. In addition, 84.2% of the newly built buildings in the Indianapolis image dataset were successfully detected, which indicated that the proposed method can be used in building change detection and building footprint updating.

According to the segmentation results on the two datasets, the proposed Mask R-CNN model achieved an MPA of 0.92 and 0.88 respectively for the Bloomington dataset and the Indianapolis dataset. Based on the Mask R-CNN segmentation, the achieved ADE of the extracted building contours on the Bloomington dataset was 13.17 while the achieved ADE for Indianapolis was 22.64. The performance of the mask segmentation and contour extraction processes declined as the complexity of the building shapes and rooftops increased. In some cases, a complex building with inconsistent features was over-segmented into several pieces.

There are several areas where the proposed method could be improved. First, the computation speed of the model was a little slow due to the long computation time required in the deep neural network. Future research may consider applying a lightweight neural network for feature extraction to accelerate the training process. Second, the ratio of rare features did not change when data augmentation was applied to the whole dataset. Therefore, more augmentation of rare features could balance the distribution of different features and improve the performance of building extraction. Third, the threshold of positive region proposals was slightly higher than

needed. Although a high threshold may have guaranteed high quality RoIs, it also filtered some small objects. Future research could test a lower threshold, such as 0.6, to generate a higher capture rate of small objects. Lastly, the IoU of the bounding box in the box regression branch was computed, but no IoU was computed for the mask in the mask branch, which could have resulted in high performance on detection but low performance on segmentation. To avoid this, a mask IoU could be added in the mask branch, such as Mask Score R-CNN. The experimental results in this thesis for this Mask Score R-CNN demonstrated the improved mask segmentation.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. Tensorflflow: a system for large-scale machine learning. *OSDI*, Vol. 16. pp. 265–283, 2016.

Arica, N. and Vural, FT. BAS: a perceptual shape descriptor based on the beam angel statistics. *Pattern Recognition Letters*, Vol. 24, Issues 9-10, pp. 1627-1639, 2003.

Aytekin, O., Ulusoy, I., Abacioglu, E.Z., and Gokcay, E. Building detection in high resolution remotely sensed images based on morphological operators. *Recent Advances in Space Technologies RAST 4th International Conference on*, pp. 376-379, 2009.

Ball, G.H. and Hall, D.J. Isodata, a novel method of data analysis and pattern classification. *Technical Report, DTIC Document*, 1965.

Cha, J., Cofer, R.H., and Kozaitis, S.P. Extended ough transform for linear feature detection. *Pattern Recognition*, Vol. 39, No. 6, pp. 1034-1043, 2006.

Dalal, N. and Triggs, B. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, CVPR 2005, IEEE, Computer Society Conference on*, Vol. 1, pp. 575-588, 2005.

Dutta, A. and Zisserman, A. The VIA annotation software for images, audio, and video. *Proceedings of 27th ACM International Conference on Multimedia*, pp. 2276-2279, 2019.

Girshick, R., Donahue, J., Darrell, T. and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of IEEE International Conference on Computer Vision (ICVV), pp. 580-587, 2014.

Girshick, R. Fast r-cnn. *Proceedings of IEEE International Conference on Computer Vision (ICVV)*, pp. 1440-1448, 2015.

Glorot, X., Bordes, A. and Bengio, Y. Deep sparse rectifier neural networks. In *Proceedings of Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pp. 315-323, 2011.

Goodfellow, I., Bengio, Y. and Courville, A. Deep learning. MIT Press, 2016, http://www.deepleringbook.org.

Gu, J. Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X. and Cai, J. Recent advances in convolutional neural networks. *Pattern Recognition*, Vol. 77, pp. 354-377, 2018.

Guducu, V. and Halici, U. Hypothesis based detection of building with rectilinear projection in satellite images using shade and color information, *Signal Processing and Communications Applications Conference (SIU), 2010 IEEE 18th*, pp. 680-683, 2010.

He, K., Zhang, X., Ren, S. and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *IEEE Transition on Pattern Analysis and Machine Intelligence*, Vol. 37, no. 9, pp. 1904-1916, 2015.

He, K., Zhang, X., Ren, S. and Sun, J. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pp. 770-778, 2016.

He, K., Gkioxari, G., Dollarand, P., and Girshick, R. Mask R-CNN, *Proceedings of IEEE International Conference on Computer Vision (ICVV)*, pp. 2961-2969, 2017.

Heffelfinger, B. Cloud factory blog. https://blog.cloudfactory.com/top-benefits-and-limitations-of-auto- labeling. 2020.

Hosang, J., Benenson, R., and Schiel, B. Learning non-maximum suppression. P*roceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4507-4515, 2017.

Huang, G., Liu, Z., Maaten, L. and Weinberger, K. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2261-2269, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 448-456, 2015.

Izadi, M. and Saeedi, P. Automatic building detection in aerial images using a hierarchical feature-based image segmentation. *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 472-475, 2010.

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. Spatial transformer networks. *Proceedings of 28th International Conference on Neural Information Processing System (NIPS)*, Vol.2, pp. 2017-2025, 2015.

Jin, X. and Davis, C.H. Automatic building extraction from high-resolution satellite imagery in urban area using structural, contextual, and spectral information. *EURASIP, Journal on Applied Signal Processing*, Vol. 14, pp. 2196-2206, 2005.

Jung, C.R. and Schramm, R. Rectangle detection based on windowed Hough transform. *Proceedings of 17th Brazilian Symposium on Computer Graphic and Image Processing, Curitiba, Brazil*, pp. 113-120, 2004.

Kettig, R.L. and Landgrebe, D.A. Classification of multispectral image data by extraction and classification of homogeneous objects. Geosciences Electronics, IEEE Transaction on, 14(1): 19-26, 1976.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. Imagenet classification with deep convolutional neural networks. Commun, ACM, 60(6): 84-90, 2017.

LeCun, Y., Botton, L., Bengio, Y., Haffner. Gradient-based learning applied to document recognition. *Proceedings of IEEE*, 86(11): 2278-2324, 1998.

LeCun, Y., Botton, L. and Hinton, G. Deep learning. *Nature*, 521(7553): 436-444, 2015.

Lee, D.S., Shan, J., and Bethel, J.S. Class-guided building extraction from Ikonos imagery. *Photogrammetric Engineering and Remote Sensing*, Vol. 69, No. 2, pp. 143-150, 2003.

Lin, C. and Nevatia, R. Building detection and description from a single intensity image. *Computer Vision and Image Understanding: CVIU*, 72(2), pp. 101-121, 1998.

Lin, T., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. and Dollar, P. Microsoft COCO: Common objects in context. *European Conference on Computer Vision (ECCV)*, pp. 740-755, 2014.

Liow, Y.T. and Pavlidis, T. Use of shadows for extracting buildings in aerial images. *Computer Vision, Graphics, and Image Processing*, Vol. 49, No. 2, pp. 242-277, 1990.

Liu, W., Anguelov, D.. Erhan, D., Szegedy, C., Reed, S.E., Fu, C. and Berg, A.C. SSD: single shot multibox detector. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, pp. 21-37, 2016.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Lowe, D. Object recognition from local scale-invariant features. In *Proceedings of seventh IEEE International Conference on*, Vol. 2, pp. 1150-1157, 1999.

Ma, H., Liu, Y., Ren, Y., and Yu, J. Detection of collapsed buildings in post-earthquake remote sensing image based on the improved YOLOv3. *Journal of Remote Sensing*, 12(1), 2020.

Makantasis, K., Karantzalos, K., Doulamis, A., and Loupos, K. Deep learning-based man-made object detection from hyperspectral data. *Advances in Visual Computing. ISVC*, 2015.

Mastylo, M. On interpolation of bilinear operators. Journal of Functional Analysis, Vol. 214, pp. 260-283, 2004.

Nwankpa, CE., Ijomah, W., Gachagan, A. and Marshall, S. Activation functions: comparison of trends in practice and research for deep learning. arXiv:1811.03378v1, 2018.

Pakizeh, E. and Palhang, M. Building detection from aerial images using Hough transform and intensity information. *Electrical Engineering (ICEE), 2010 18th Iranian Conference o*n, pp. 532-537, 2010.

Peng, J., Zhang, D., and Liu, Y. An improved snake model for building detection from urban aerial image. *Pattern Recognition Letters*, Vol. 26, No. 5, pp. 587-595, 2005.

Pinheiro, P., Lin, T., Collobert, R., and Dollar, P. Learning to refine object segments. *European Conference on Computer Vision (ECCV)*, pp. 75-91, 2016.

Qiao, Y., Truman, M. and Sukkarieh, S. Cattle segmentation and contour extraction based on Mask R-CNN for precision livestock farming. *Computers and Electronics in Agriculture*, Vol. 165, 2019.

Ranzato, M, Boureau, Y. and LeCun, Y. Spare feature learning for deep belief networks. In *NIPS*, 2007.

Redmon, J. and Farhadi, A. YOLO9000: better, faster, stronger. *Computer Vision and Pattern Recognition, CVPR, 2017 IEEE Conference on, Honolulu, HI, USA, July 21-26, 2017*, pp. 6517-6525, 2017.

Redmon, J. and Farhadi, A. YOLOv3: An incremental improvement, *GoRR*, abs/1804.02767, 2018.

Redmon, J., Divvala, S.K., Girshick, R.B., and Farhadi, A. You only look once: Unified, real-time object detection. *2016 Computer Vision and Pattern Recognition, CVPR, IEEE Conference on, Las Vegas, NV, USA, June 27-30, 2016*, pp. 779-788, 2016.

Ren, S., He, K., Girshick, R. and Sun, J. Faster R-CNN: Towards real-time object detection with region proposal network. *Proceedings of 28th International Conference on Neural Information Processing System (NIPS)*, 2015.

Saeedi, P. and Zwick, H. Automatic building detection in aerial and satellite images. *Control, Automation, Robotics and Vision, 2008. ICARCV 2008, 10th International Conference on*, pp. 623-629, 2008.

Scherer, D., Muller, A. and Behnke, S. Evaluation of pooling operations in convolutional architectures for object recognition. *Lecture Notes in Computer Science*, Vol. 6534, 2010.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. and LeCu, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229, 2013.

Shorten, C. and Khoshgoftaar, M.T. A survey on image data augmentation for deep learning. *Journal of Big Data* 6, No. 60, 2019.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Sirmacek, B. and Unsalan, C. Building detection from aerial images using invariant color features and shadow information. *Computer and Information Science, 2008 ISCIS'08. 23rd International Symposium on*, pp. 1-5, 2008.

Sirmacek, B. and Unsalan, C. Uran-area and building detection using SIFT key points and graph

theory. *Geoscience and Remote Sensing, IEEE Transactions on*, Vol. 47, no. 4, pp. 1156-1167, 2009.

Sportouche, H., Tupin, F., and Denise, L. Building detection by fusion of optical and SAR features in metric resolution data. Geoscience and Remote Sensing Symposium, IEEE International, IGARSS, pp. 769-772, 2009.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929-1958, 2014.

Suzuki, S. and Be, K. Topological structural analysis of digitized binary images by boundary following. *Computer Vision, Graphics and Image Processing*, 30(1), pp.32–46 1985.

Tao, C., Tan, Y., Cai, H. and Tian, J. Airport detection from large Ikonos images using clustered SIFT key points and region information. In IEEE Geosicence and Remote Sensing Letters, Vol. 8, no. 1, pp. 128-132, 2011.

Targ, S., Almeida, D. and Lyman, K. Resnet in resnet: Generalizing residual architectures. arXiv:1603.08029v1, 2016.

Torralba, A., Russell, B.C., and Yuen, J. Labelme: Online image annotation and application. *Proceedings of the IEEE*, Vol. 98, No. 8, pp. 1467-1484, 2010.

Vargas-Muñoz, J.E., Marcos, D., Lobry, S., Santos, J.A.. Falcão, A.X. and Tuia, D. Correcting misaligned rural building annotations in Open Street Map using convolutional neural networks evidence. *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, Valencia, pp. 1284-1287, 2018.

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

Vincent, O.R. and Folorunso, O. A descriptive algorithm for Sobel image edge detection. *Proceedings of Informing Science & IT Education Conference (InSITE)*, 2009.

Wu, Y. and He, K. Group normalization. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, pp. 3-19, 2018.

Xie, Y., Cai, J., Bhojwani, R., Shekhar, S. and Knight, J. A locally-constrained YOLO framework for detecting small and densely-distributed building footprints, *International Journal of Geographical Information Science*, 34(4): 777-801, 2020.

Xu, B., Wang, N., Chen, T. and Li, M. Empirical evaluation of rectified activations in convolution networks. arXiv:1505.00853v2, 2015.

Xu, B., Huang, R. and Li, M. Revise saturated activation functions. *CoRR*, abs/1602.05980, 2016.

Xu, Y., Wu, L., Xie, Z., and Chen, Z. Building extraction in very high resolution remote sensing imagery using deep learning and guided filters. *Remote Sensing*, Vol. 10, 2018.

Yang, J., Ji, L., Ge, X., Yang, X., and Zhao, X. Building detection in high spatial resolution remote sensing imagery with the U-Rotation Detection Network. *International Journal of Remote Sensing*, Vol.40, 2019.

Yang, Q., Xiao, D., Lin, S. Feeding behavior recognition for group-housed pigs with the Faster R-CNN. *Computers and Electronics in Agriculture,* 155, 453–460, 2018.

Yu, Y., Zhang, K., Yang, L. and Zhang, D. Fruit detection for strawberry harvesting robot in no-structural environment based on Mask R-CNN. *Computers and Electronics in Agriculture*, Vol. 163 (2019).

Yuksel, B. Automated building detection from satellite images by using shadow information as an object invariant. 2012.

Zafar, W. Object detection and segmentation using Region-based deep learning architectures. 2018.