Automated classification of satellite data of informal urban settlements

Zequn Zhou

Master Thesis

Computer Science
August 26th, 2019

TU Delft

AUTOMATED CLASSIFICATION OF SATELLITE DATA OF INFORMAL URBAN SETTLEMENTS

by

Z.(Zequn) **Zhou**
Student number: 4714903

in partial fulfillment of the requirements for the degree of

Master of Science

in Computer Science (Data Science & Technology)

at the Delft University of Technology,

to be defended publicly on 2019-08-26.

The chair of the master thesis assessment committee:

Supervisor:

Dr. Neil Yorke-Smith	Delft University of Technology
Dr. ir. Árpád Rózsás	PNGK

Committee members:

Dr. Neil Yorke-Smith	Delft University of Technology
Dr. Odette Scharenborg	Delft University of Technology
Dr. ir. Árpád Rózsás	PNGK
Ivan Thung	UN-Habitat

PREFACE

Two and a half years ago, when I made the decision to go Delft University of Technology, I just feel like it it just two years of study. However, when I look back now, I would say this is one of my best choices in my life. Here, in TUDelft, I not only learned a lot on computer science, but also have a lots of feelings on life. This is an precious experience for me to feel the difference between East and West and learn to take care of myself. Sometimes I feel like I can't wait to start my career as I can apply my knowledege and achieve my ambitions, but sometimes I feel like being a student is great since I don't have to think about too many other things. As time goes by, I am almost finishing my thesis – my last work as a student. And before long, there will be another important decision waiting for me.

My thesis topic is interesting and challenging. Here I would like to express my greatest gratitude to those who assisted me to accomplish this graduate program.

First of all, my special thanks go out to Neil Yorke-Smith and Árpád Rózsás, my thesis supervisors, for guiding me along the way and giving me a lot of useful and great suggestions. Neil is super nice and always encourages me to explore new ways. Working with Arpad is amazing, as he help me a lot, from pointing out my mistakes, providing me lots useful documents and resources, to suggesting me to think further and more. Here I would like to mention Eric, who also help me a lot about the AWS.

I also want to thank the support of the UN-Habitat and Ivan, for providing me the satellite images and the corresponding labels. It is an amazing experience to work with the biggest organization all over the world and I also know more about their work.

No journey is complete without friends. I am super lucky to meet and know you that I can share my journey with. We almost meet at the same building and fight for the same goal everyday. I am so grateful to become friends with you. Best of luck to all of you.

Finally, I would like to thank my parents who are always supporting and caring about me. Also my girlfriend, who is always comforting me and making me happy.

Zequn Zhou
Delft, August 2019

ABSTRACT

Urban areas are rapidly expanding in developing countries. One of goals of the United Nations Human Settlement Programme (UN-Habitat) is to understand and guide urban development for some developing regions. Currently, the approaches that UN-Habitat is using cost plenty of workforce, material, and time. Therefore, UN-Habitat is interested in exploring new approaches on how to drive down costs and time, which would not only allow for faster responses but expanding their analysis. Since UN-Habit is already using satellite imagery for urban mapping, our research question is formulated as: Can we develop an automated system that provides valuable information about urban development for the UN-Habitat from satellite image data (e.g. building detection)?

After examining the satellite imagery provided by UN-Habitat and those available publicly (crowd AI and Inria Areial datasets), we define the main task as a building segmentation task. In this research, we study deep learning techniques for building segmentation on satellite image data. Duo to the fact that the number of images and the quality available for the region of interest (Middle East) for UN-Habitat are insufficient to solely rely on for training. Therefore, we use some public datasets (crowd AI and Inria Areial datasets) for training and evaluation, whose regions and construction practice are different. Starting with testing several classic segmentation algorithms (FCN8S, SegNet, Deep_Lab and U-Net), from the experiment results, we find that the performance can still be improved. Then, we propose two novel data reweighing methods, named border weight and inter-building distance weight, to improve the detection performance. By increasing the weights of the pixels outside but close to the border of the buildings, the model is encouraged to learn those information and thus performs better. Inspired by the idea of reweighing the non-building pixels, we investigate whether modifying building pixels can achieve further improvement. We propose a new label representation – multi-level boundary label that does help to improve the segmentation results. Based on the distance to the building boundary, we can divide building pixels into multiple classes, as their pixel values can be affected by some factors such as trees and shadows. From the experiment result, we can see that the performance is improved since the model captures more information about the buildings. Next, we propose a new neural network architecture that utilizes the two pixel weights, and the multi-level boundary label explained above. Our proposed model achieves state-of-the-art building segmentation performance compared with several classic segmentation methods. For example, the proposed model's mean intersection of union on the test dataset is 3% higher than that of FCN8S. Our model also uses fewer number of parameters (16 million in total) because we only use the first 13 layers of the VGG16 as the encoder and we do not use any convolutional layers in the decoder part.

The results using the publicly available datasets show that with enough good quality input the building segmentation is possible, hence should be possible in other regions as well. To see the performance of our proposed model on the UN-Habitat dataset, we train our model with public datasets (crowd AI and Inria Areial datasets) and then use transfer learning to fit the UN-Habitat dataset. The building detection performance is reduced still good results are obtained. For achieving comparable performance in the region of interest for UN-Habitat more labelled data is needed. Based on the results using the publicly available datasets, we are confident that a comparable performance is attainable.

Regarding the research question, our answer is definitely yes. We not only show that it is possible

to obtain information about urban development from satellite image but also propose a new model with great performance in our work.

Contents

List of Figures

List of Tables

1

INTRODUCTION

1.1. MOTIVATION

Urban areas are rapidly expanding which is well illustrated by the increasing number of city dwellers: about half million each week [16]. Figure 1.3 illustrates that this growth is uneven: it is the most pronounced in developing countries. Understanding the scale and speed of urban developments, as well as the type of dwellings associated with these developments is instrumental for addressing urban sprawl with evidence-based planning.

In many regions, such as in some parts of Africa and the Middle East, urban developments take place in an uncoordinated, unmonitored, and unrecorded manner. Among other goals the United Nations Human Settlement Programme (UN-Habitat) was established as a response to these challenges and it plays a crucial role in understanding and in a later stage guiding urban development in these regions. The prerequisite of understanding is a diagnosis, i.e., mapping the current urban conditions. This is a formidable task which involves recording the geographical location of each building (detection), identifying the building type, and in some cases also assessing the structural conditions, e.g., severely damaged, moderately damaged, or sound. In this thesis we focus solely on the challenge of building detection.

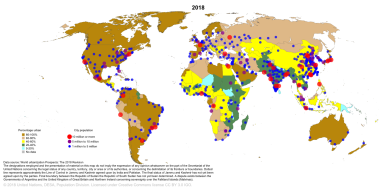

Figure 1.1: Year 2018.

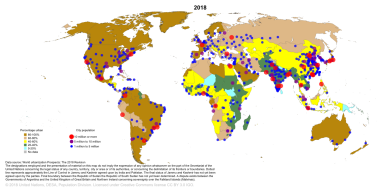

Figure 1.2: Year 2030.

Figure 1.3: Illustration of the growth of urban areas and urban agglomerations [1]

1.2. CURRENT UN-HABITAT APPROACH

The ultimate goal of UN-Habitat is to provide adequate shelter for all. Therefore it is vital to understand the speed and scale of urban developments, in regions where these developments take place without coordination, monitoring, and recording. In these situations, especially when the given data is satellite imagery, the number of buildings and their sizes can be useful data as with them we

can estimate the population number and the quality of people's life. In this thesis, we focus on one challenge, which is the building detection.

Currently, UN-Habitat uses two approaches and their combination to map urban areas in regions with uncoordinated and unmonitored urban development:

1. On the ground: visual inspection and mapping of buildings.
2. In the office: satellite imagery-based mapping by humans: visual inspection of images and manual identification.

To provide further insight, we compare the two approaches in respect of requirements, costs, and results. For the first approach, many people are required to check the buildings one by one. It usually costs a substantial amount of money and time. Inspectors need to travel to every building, inspect them, and then record what they see. For a city like Mosul whose area is 180 km$\hat{2}$, it may require two or three months to collect these data. The results of this approach are accurate and complete because the inspectors can clearly see the details of each building, and they can even talk with local people to confirm their findings.

Concerning the second approach: satellite images are required, and their quality such as resolution is very important since it directly affects the accuracy of the result. Compared to the first approach, it can save time and money as people can sit in front of their computer to finish the task. It takes about 10 to 15 seconds to annotate an outline of a building. Therefore, the task can be finished within a month with the same number of people as for the first approach. But the result may not be so good as the information of the satellite imagery is not complete (top-down view), especially for the damage assessment.

Both of these approaches require people to collect data manually, and they take some time to finish. There are some differences between the results: for example, for the second approach, we may underestimate the number of buildings when they are too close and can be mistaken as a single building. In this case, the total floor area of them can make more sense as the final goal is to estimate urban development. The working hypothesis of this study is that we can improve the current UN-Habitat approaches with an automated approach that does not require human labor.

1.3. PROBLEM STATEMENT

Both current UN-Habitat urban mapping approaches are so labor-intensive that it prohibits the mapping of large urban areas, let alone the continuous monitoring of their development in time. Hence the main challenge for UN-Habitat and in general in urban mapping is:

How to extend urban mapping in space and time in a formidable way, i.e., using the currently available capacities?

Current mapping approaches cost plenty of workforce, material, and time. Therefore, if we can build an automated system that can monitor the urban developments, we can save a lot and greatly benefit from it. In this case, satellite imagery can be used as a powerful source which contains rich and structured information. Compared to traditional images, satellite images started to attract attention recently from many researchers for map composition, population analysis, effective precision agriculture, and autonomous driving tasks [17]. From top-down view images we can estimate the number of buildings and their size in a city. From this we can gain multiple insights: (*i*) we can know the building density so that we can estimate the population density; (*ii*) we can also learn how cities sprawl and the number of informal dwellings; (*iii*) what is more, obtaining the building maps in

these areas can greatly improve the response of emergency preparedness actors when natural, and civil disasters occur.

This raises to the following main research question:

- **Can we develop an automated system that provides valuable information about urban development for the UN-Habitat from satellite image data (e.g. building detection)?**

To address this challenging question, we define the main task as a building segmentation task after examining the satellite imagery provided by UN-Habitat. With the development of computer vision and deep learning techniques, especially convolutional neural networks (CNN), the limits of what we can achieve have been pushed in recent years. In this thesis, we study deep learning algorithms for image semantic segmentation. It is an interesting topic to achieve a building detection system based on satellite images, during which we can see the impact of bridging modern computer vision with remote sensing data analysis on our understanding of environment such as urban growth.

Our main contributions of this thesis:

- Investigate whether a combination of different pixel weights could lead to better performance in the building segmentation task.
- Investigate whether a multi-class label is a feasible annotation type to improve the performance of building segmentation task.
- Propose a model[1] that achieves state-of-the-art building segmentation performance on publicly available datasets.
- Apply two post-processing steps on the segmentation results and demonstrate that we can obtain information about urban development from segmentation results.

1.4. STRUCTURE OF THE REPORT

The remainder of this thesis is organized as follows. Background and literature overview are presented in chapter 2. Chapter 3 describes the datasets that are used for the following experiments and the processing steps, as well as the evaluation metrics. We test some classic image segmentation models in chapter 4. The experiment settings are introduced and the results of the initial experiment are used as baseline for the comparisons in the following chapters. Next, motivated by the initial experiments, two methods to reweigh the data are proposed and tested in chapter 5. Chapter 6 introduces a multi-class label representation and investigates if the performance of the segmentation methods can be improved. Based on these (numerical) experiment results, in chapter 7, we propose a new model to utilize the two pixel weights and the multi-level boundary label explained in chapter 5 and 6. Chapter 8 introduces the post-processing steps to obtain urban development information from the output of the model. Finally, Section 9 summarizes our work and provides some discussion points and future work.

[1] In this thesis, the term 'model' refers to a deep learning model.

2

Background and Literature Overview

To have a clear understanding of the rest of the thesis, it is essential to know some basic concepts and prior research results. An overview of relevant theoretical information and terms that will be used in the rest of the thesis is presented. We also introduce some related work in this chapter.

2.1. Machine Learning

In computer science field, artificial intelligence (AI), sometimes called machine intelligence, is the intelligence that machines display, in other words, the simulation of human intelligence processes by machines, especially computer systems. As a subset of AI, the main research goal of machine learning is to achieve AI, especially to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly [18]. The name of *machine learning* was coined by Arthur Samuel, with the definition "a field of study that gives computers the ability to learn without being explicitly programmed [19]." In 1998, another well regarded machine learning researcher proposed a more precise definition "a computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E [20]."

Instead of designing a specific algorithm for a specific task or setting up strict rules to make a prediction, machine learning algorithms are a class of algorithms that automatically analyze and learn rules from data, then use these rules to make predictions on unseen (not used for training) data. Therefore, the quantity and quality of the input data is of paramount importance. Machine learning has been widely used in various fields, such as machine translation, recommendation systems, and data mining [21].

According to the type of input and output data, and the type of task or problem to solve, different types of machine learning algorithms can be distinguished as follows:

- Supervised learning: Models that can predict labels of unseen data based on labeled training data.
- Unsupervised learning: Models that can identify structures in unlabeled data.
- Semi-supervised learning: Models that often are used when labels of training data are incomplete. Semi-supervised learning can be regarded as a class of methods that falls between supervised learning and unsupervised learning.

- Reinforcement learning: Rewarded-based models that learn to maximize the reward by interacting with the environment.

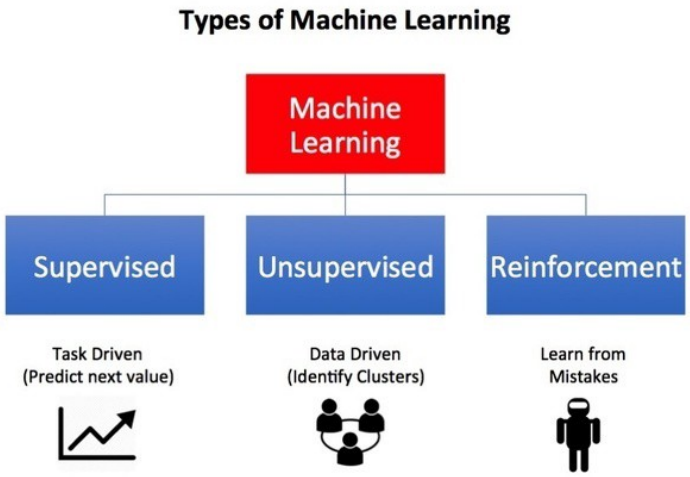

Figure 2.1: Types of machine learning [2]

2.1.1. SUPERVISED LEARNING

The majority of machine learning approaches are supervised learning algorithms. In a supervised learning setting, an algorithm is trained with a training dataset whose label or the desired outcome is known. The algorithm can learn and adjust to yield better results by minimizing an error between the output and the ground truth. When having sufficiently large training data and an appropriate learning algorithm, we can get a model with enough generalization capacity to function on real-world data. It is expected to perform good with excellent performance.

Typically, supervised learning problems are subdivided into classification tasks and regression tasks. In classification, the labels are discrete categories, while in regression, the labels are continuous quantities. Some popular examples of supervised machine learning algorithms are Linear Discriminant Analysis [22], Support Vector Machines [23], and Random Forest [24].

2.1.2. UNSUPERVISED LEARNING

Unsupervised learning involves modeling the features of a dataset without labels. It is also known as self-organization and allows modeling probability densities of given inputs [25]. Algorithms are left to their own to discover and present the interesting structures in the data. The goal of unsupervised learning is to learn more about the underlying processes and mechanism by modeling the underlying structure or distribution in the data.

Unsupervised learning problems can be further grouped into clustering tasks and dimensionality reduction tasks. Clustering algorithms identify distinct groups of data in a way that objects in the same group are more similar in some sense to each other than to those in other groups, while dimensionality reduction algorithms search for more succinct representations of the data which leads to only a small loss of accuracy. Several classic unsupervised machine learning algorithms are K-means Clustering algorithm [26], Principal Component Analysis [27], and Manifold Learning [28].

Given the data and the research goals, we will only consider supervised learning approaches further in this study.

2.2. DEEP LEARNING

Deep learning is a particular type of machine learning technique that uses neural networks, which learns data representations with multiple levels of abstraction. The relationship between AI, machine learning and deep learning is shown in figure 2.2 Inspired by the information processing and communication way of biological nervous systems, deep learning uses complex neural networks with many hidden layers. We discuss the details of the neural network in section 2.3.

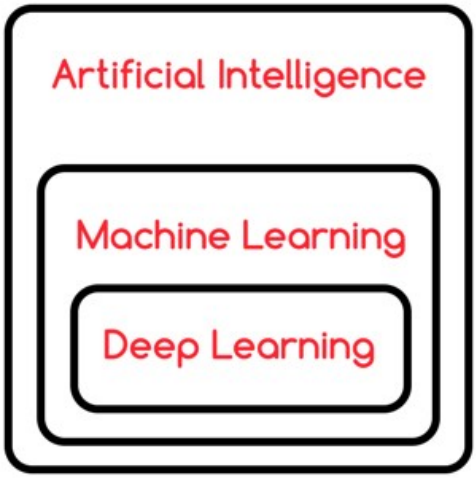

Figure 2.2: Relationship between AI, machine learning and deep learning [3]

Deep learning methods have dramatically improved the state-of-the-art performance in speech recognition, visual object recognition, object detection, and many other domains such as drug discovery and genomics [29].

One of the advantages of deep learning is that it can capture intricate structures of high-dimensional data which are hard to be discovered. Therefore, deep learning has a wide range of applications, such as image classification, speech recognition, and natural language understanding. However, deep learning comes at a high cost: it is hard to explain why and how the prediction is made. Deep learning methods usually have high accuracy while are not being interpretable.

2.3. NEURAL NETWORK

A neural network or artificial neural network(ANN) is based on a collection of connected units called "neurons". The units are combined and form a network. A unit is shown in figure 2.3 and formulated as follows:

To explain how it works, let us say N is the input dimension and N can be different for different neurons. Each unit has an N dimensional input vector $A = [a_1 a_2 ... a_N]^T$ and corresponding N dimensional learnable weight vector $W = [w_1 w_2 ... w_N]$ and bias value b. Firstly, weighed sum $z = \sum_{i=1}^{N} a_i w_i + b$ is computed then z is passed through an activation function f to get the final output a_{out}. Mostly, the activation function is a nonlinear function to add the non-linearity and allow for learning complicated mappings between inputs and outputs, such as Linear Rectified Units(ReLu) [30] and Sigmoid function. The type of activation function is important as it can be used to determine whether and to what extent that signal should progress further through the network to affect the final outcome.

Units are aggregated into layers, and layers are aggregated into a neural network. Different types of operations and transformations are performed by different layers. The output of each layer is the

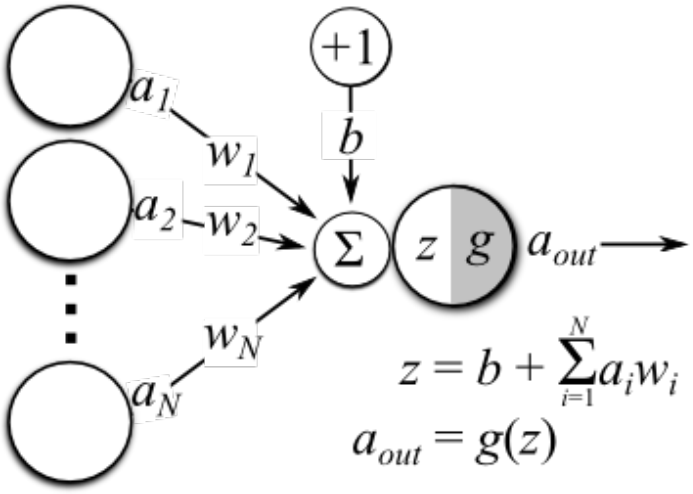

Figure 2.3: A single unit [4]

input of the subsequent layer, starting from an input layer receiving the input data. Figure 2.4 shows a network with two hidden layers.

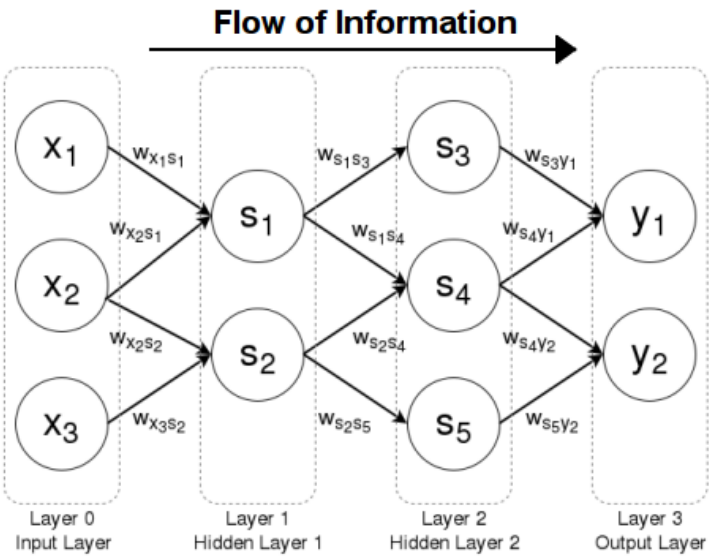

Figure 2.4: A neural network with 2 hidden layers [5]]

2.3.1. CONVOLUTIONAL NEURAL NETWORK

As the provided data are satellite imagery, we decide to use a class of networks that are most commonly applied to analyzing imagery data. Convolutional neural networks(CNNs) are a specialized kind of neural network for processing data that has a known, grid-like topology, inspired by the organization of the animal visual cortex. CNNs are neural networks that use convolution operation instead of general matrix multiplication [31]. The term convolution is a mathematical operation derived from two given functions by integration to express how the shape of one is modified by the other. In case of CNN, convolution is performed on the input data with the use of a filter or kernel to produce a feature map. Convolution is executed by sliding the filter over the input. At every location, a matrix multiplication is performed on the input and filter and sums the result onto the

feature map. A CNN consists of one or more convolutional layers, followed by activation functions and pooling layers. Figure 2.5 shows a typical CNN architecture.

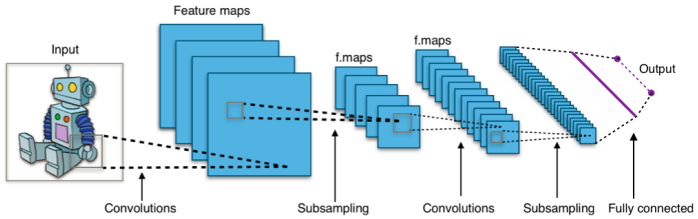

Figure 2.5: A CNN architecture [5]

Each convolutional layer has a group of convolution kernels or can be called learned filters whose receptive fields are small and spatial such as 3×3 or 5×5. The output of these kernels is collected by convolving each kernel on the input features, whose number is analogous to the number of kernels. Then non-linearity is added in an element-wise way by the activation function, and the pooling layer is applied whose function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control overfitting. [cite] The weight values of each kernel are learnable during the training steps. Normally, a deep CNN contains more than one convolution layer to capture different levels of feature representation step by step. For example, a simple feature can be an edge or orientation. Within an image, each pixel value has a higher correlation with near neighbors' than far neighbors'. Therefore, using a convolution operation makes sense because convolution can make good use of the local connectivity of each pixel. Moreover, the number of the parameters can be dramatically reduced compared to fully connected layer because the weights of each kernel are shared by all pixels of the input. The way a convolutional layer works is visualized in figure 2.6.

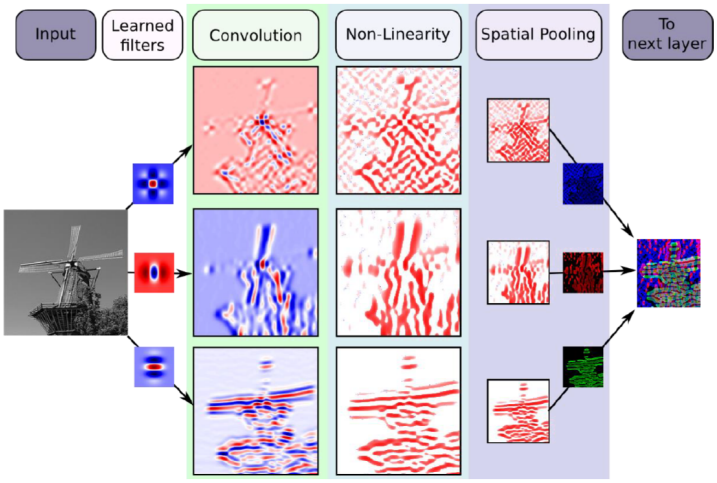

Figure 2.6: Visualization of how a convolutional layer works [cite]

2.4. SEMANTIC SEGMENTATION

Image classification refers to a classic problem in the computer vision area: classifying an image based on its visual content where the output to an image is a single class label. It is a natural step in the progression from coarse to fine inference, to make dense predictions inferring labels for every pixel. To recognize and understand what is in the image on a pixel level, image semantic segmentation is the process of assigning a label to every pixel in an image such that pixels with the same

label belong to the same object or region. The goal of segmentation is to simplify or change an image into a representation that is easier to understand and analyze. Specifically, we can obtain the objects' locations and boundaries in an image through semantic segmentation. So far, semantic segmentation is a widely used application in many fields, such as autonomous driving, object detection, face recognition and so on. Figure 2.7 shows an example of semantic segmentation.

Figure 2.7: Image semantic segmentation

In the following sections, several popular methods in deep learning techniques are described.

2.4.1. FULLY CONVOLUTIONAL NETWORKS

Fully Convolutional Networks (FCNs) [32] were proposed by Jonathan et al. in 2015. Their paper is a pioneering work that applies the CNN structure to the field of image semantic segmentation and achieves outstanding results. They successfully transformed existing and well-known classification models into fully convolution ones to output spatial maps rather than classification scores.

FCN is an end-to-end, pixel-to-pixel network, in other words, the input is the original raw data, and the output is the final result.

Instead of using fully connected layers as the last several layers, FCN uses convolution layers in the end, which enables a classification net to output a heatmap [32], which is more suitable for image segmentation. With the help of *deconvolution*, such that convolution with factor f and a fractional input stride of $1/f$, the image can be upsampled to the original size. In this case, a pixel-wise loss can be easily computed and can be learned through backpropagation. Figure 2.8 shows an example structure of FCN.

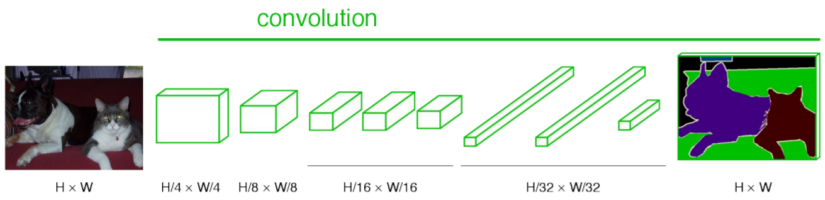

Figure 2.8: An example of fully convolutional network [CITE]

FCN allows input of arbitrary size rather than a fixed dimension and produces correspondingly sized output, and it achieves a better performance than prior work [32].

2.4.2. U-NET

Olaf et al. modified and extended the FCN architecture such that it works with fewer training images and performs better [6]. An important modification is that in the upsampling part, there are lots

of feature channels that allow the network to propagate context information to higher resolution layers. As a result, the architecture is symmetric and has a u-shape.

The architecture of U-Net [6] is showed in figure 2.9. The left part is the contracting path, consisting of a usual convolutional network where convolution layers are alternated with max-pooling layers to downsample the input. The size of the input is decreased, but the number of feature channels is doubled at each downsampling step. The right part is the upsampling path where deconvolution is applied to upsample the feature maps, which halves the number of feature channels in every step. To enhance the high-resolution, features from the contracting path are copied and concatenated with upsampled features. Based on the concatenation of coarse and fine features, a successive convolution layer can learn to assemble a more precise output.

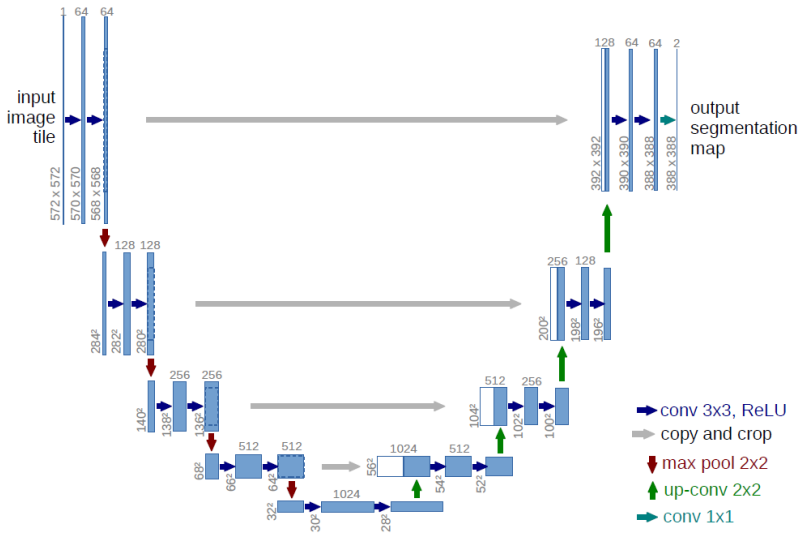

Figure 2.9: The architecture of U-Net [6]

U-Net is proposed for different biomedical segmentation applications [6]. To address the challenge in many cell segmentation tasks which is the separation of touching objects of the same class, the authors also proposed a weighted loss map, where the separating background labels between touching cells obtain a large weight in the loss function.

One weight map for each ground truth segmentation is computed to compensate for the different frequency of pixels from a certain class. Then to force the network to learn the small separation borders, another weight map related to the distances to the nearest cell is introduced. The final weight map is the sum of these two weight map, which is computed as:

$$w(x) = w_c(x) + w_0 * exp(-\frac{(d_1(x) + d_2(x))^2}{2\sigma^2}) \tag{2.1}$$

Where x is the pixel value of the training sample, $w_c(x)$ is the weight map to balance the class frequencies, $d_1(x)$ and $d_2(x)$ denote the Euclidean distances to the border of the nearest and second nearest, respectively. Also, w_0 and σ are two hyperparameters.

From figure 2.10, we can see the value of the final weight map. If we find out some characteristics of a particular dataset, we can also come up with a way to reweigh each pixel to improve the performance.

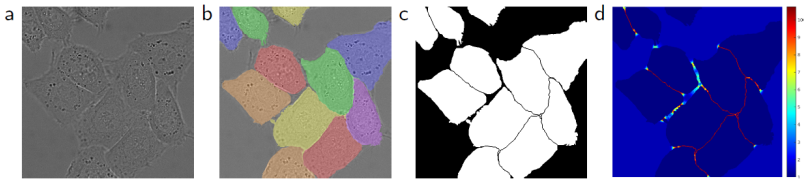

Figure 2.10: HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (a) raw image. (b) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (c) generated segmentation mask (white: foreground, black: background). (d) map with a pixel-wise loss weight to force the network to learn the border pixels. [6]

2.4.3. SEGNET

FCN-based architectures are popular and successful, and many segmentation architectures share the same encoder network as FCN, which produces low-resolution feature maps. The form of their decoder that maps the low-resolution feature maps to full resolution feature maps varies. SegNet [7] is a clear example of this divergence – the architecture of its decoder is different.

Instead of using deconvolution to upsample the features, SegNet upsamples its feature maps using the memorized max-pooling indices from their corresponding feature maps in the encoder phase. The decoder of SegNet is composed of a set of upsampling and trainable convolution layers to produce dense feature maps. Figure 2.11 shows the difference of upsampling between SegNet and FCN.

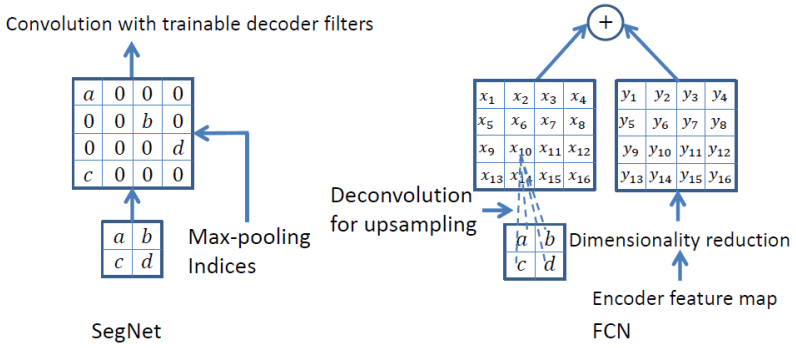

Figure 2.11: Comparison of SegNet (left) and FCN (right) decoders [7]

FCN does not reuse the max-pooling indices but instead transfer the entire feature map to the corresponding decoders. Reusing the max-pooling indices in the decoding process reduces the number of parameters enabling end-to-end training. The number of trainable parameters is also reduced compared with using deconvolution to upsample the feature maps. The architecture of SegNet is shown in figure 2.12

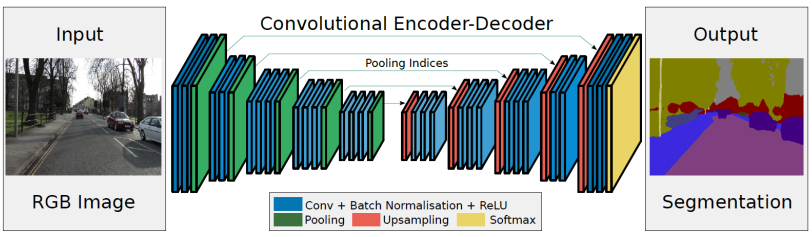

Figure 2.12: The architecture of SegNet [7]

2.4.4. DEEP_LAB

It is important to integrate local and global information for semantic segmentation. Local information is crucial to achieving good pixel-level accuracy. To resolve local ambiguities, the integration of information from the global context of the image is also necessary. Motivated by this idea, Deep_Lab [9] was proposed. We explain some concepts that are applied in Deep_Lab.

Dilated convolution [8] is a generalization of Kronecker-factored convolutional filters [33] that supports expanding receptive fields without losing resolution.

l is the dilated rate and illustrated in figure 2.13, we can see the relationship between the size of the receptive field and dilated rate. Imagine that when we build up a network out of multiple layers of dilated convolutions, the size of the receptive field grows exponentially with layer depth while the number of the parameter only keeps a linear growth.

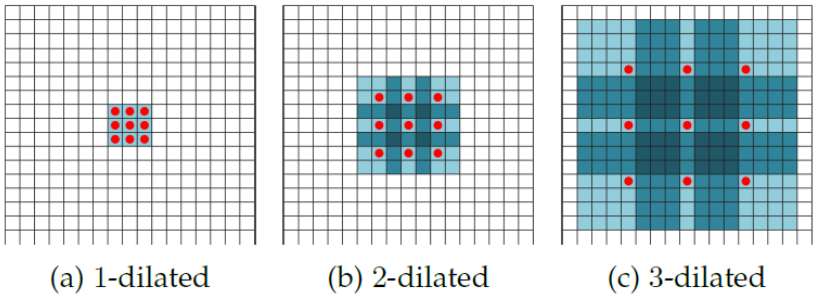

Figure 2.13: Dilated convolution filters with various dilation rates: (a) 1-dilated convolutions in which each unit has a 3 × 3 receptive fields, (b) 2-dilated ones with 7 × 7 receptive fields, and (c) 3-dilated convolutions with 15 × 15 receptive fields. [8]

To capture multi-scale features, multiple parallel filters are exploited, the author also proposed Atrous Spatial Pyramid Pooling (ASPP) _Lab[_Lab**?**] based on the idea of spatial pyramid pooling [34] [35]. Figure 2.14 shows an example of ASPP. When using dilated convolution with different kernel size and different dilated rates, we can obtain information from various scale. Then this local and global information can be integrated to produce better outputs.

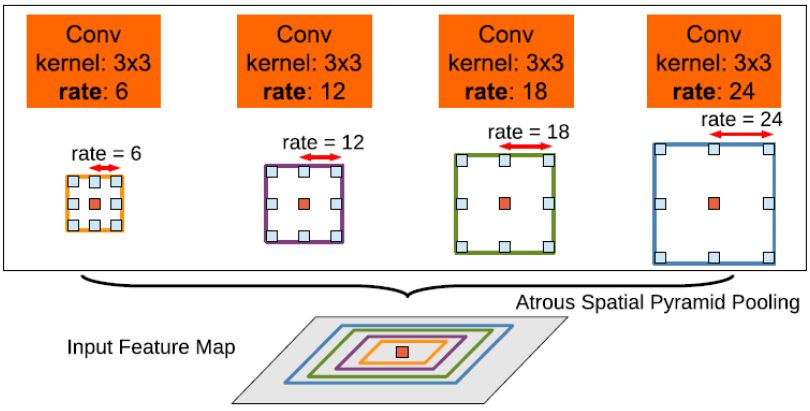

Figure 2.14: An example of Atrous Spatial Pyramid Pooling (ASPP) [9]

Deep_Lab [9] uses dilated convolution with upsampled filters for dense feature extraction. ASPP is applied to encode objects as well as image context at multiple scales.

2.5. RELATED WORK

In the field of remote sensing applications, the extraction of building footprints has been extensively studied over the past decade. This problem has been addressed by traditional computer vision techniques and was solved by traditional machine learning classifiers combined with hand-crafted features such as texture, color features [36] and vegetation indices [37] [38]. With the development of convolutional neural networks, semantic segmentation becomes one of the core challenges in computer vision, as the breakthroughs of deep learning in image classification tasks can be easily transferred to the semantic segmentation tasks. Researchers have made efforts to apply deep learning techniques for satellite images to detect buildings. Also, over the past few years, several challenges and competitions are focusing on building segmentation tasks, such as SpaceNet Off-Nadir Building Detection Challenge [39], DeepGlobe[17], and crowdAI Mapping Challenge [40].

Although the provided datasets of these contests are different from the dataset used in this project, the top solutions of these contests are good references from which we can learn.

A multi-layer perceptron approach is proposed in the paper [10] that can balance the trade-off between localization and classification for building labeling with the help of skip networks. Multiple intermediate features at different resolutions are extracted. In such a scheme, the high-resolution features have a small receptive field, while the low-resolution ones have a wider receptive field. Figure 2.15 shows the architecture.

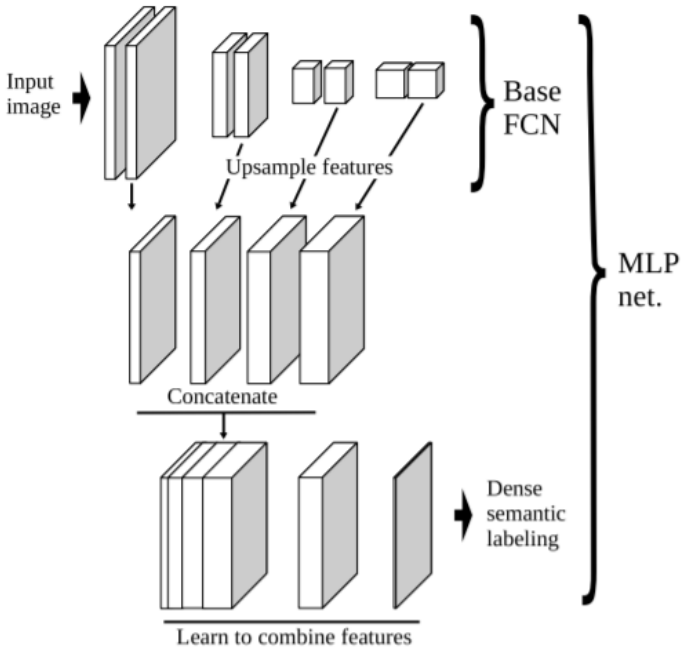

Figure 2.15: MLP network: intermediate CNN features are concatenated, to create a pool of features. Another network learns how to combine them to produce the final classification [10]

The segmentation of building footprints can be improved with a multi-task network. Specifically, semantic segmentation boundaries in high-resolution satellite images can be preserved by using a cascaded multi-task loss [41]. The classification loss function for the prediction of the distance-classes and the loss function for the segmentation mask were combined to get the total loss. Another multi-class land segmentation algorithm using feature pyramid network [42] was proposed by Selim et al. [43]. A new cascaded multi-task loss was introduced to preserve semantic segmentation

boundaries in high-resolution satellite imagery. Jaccard index and a discrete object classification loss function were added together with two hyperparameters to obtain the generalized loss function.

Building size can be considered as research target since prior works treat all buildings as a single class. Buildings with different sizes may have different features. A multi-task model [11] is proposed based on the specific size of buildings. Figure 2.16 shows the architecture. The model is based on U-Net [6] and consists of a shared feature extractor F and successive detectors. At lower levels, the detectors can share general features, and they can learn high-level features for the specific size of the building, as the size of each building has been categorized in the ground truth. It is interesting to consider the building size, and the experiment results show that this model improves building detection accuracy.

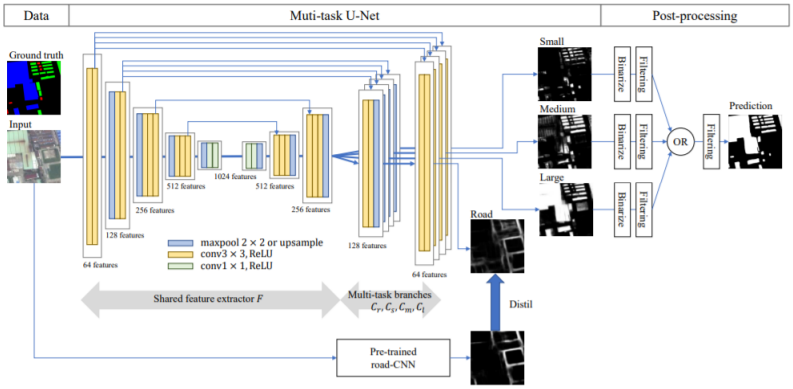

Figure 2.16: Architecture of the multi-task U-Net (centre) model. [11]

Based on a segmentation algorithm Mask R-CNN [44], building polygons can be obtained but have irregular shapes that are far different from real building footprint boundaries. A method combining Mask R-CNN with building boundary regularization is proposed by Kang et al. [45]. Polygons generated by Mask R-CNN can be converted into the regularized polygons with almost equivalent performance in terms of accuracy and completeness. Regularized polygons are directly applicable to numerous cartographic and engineering applications.

Besides the algorithms, some reweighing methods are proposed to address the properties of satellite imagery for semantic segmentation. The reweighing method of U-Net [6] has mentioned in section 2.4.2.

Another method proposed from [] modifies the binary annotation into three categories: border, pixels inside a building, and the background. The modified annotation is shown in figure 2.17 Then the weight maps are computed based on the relative frequency of each category.

Distance between objects can also be considered. In a solution, distances to the two closest objects are calculated, creating the distance map that is used for weighing. Figure 2.18 shows the distance weights. Pixels between buildings have high values, and the darker the color, the higher the value.

The representation of the ground truth label can also be modified. Figure 2.17 shows a way that the binary label is modified by adding the border as the third class [12]. In paper [13], a novel representation of object segments that is robust to errors in the bounding box proposals was proposed to go beyond the limitation of the bounding box. It is displayed in figure 2.19.

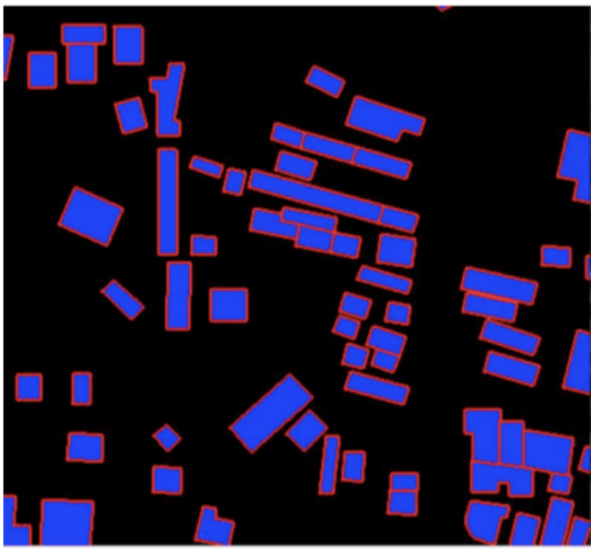

Figure 2.17: Modified label in 3 categories [12]

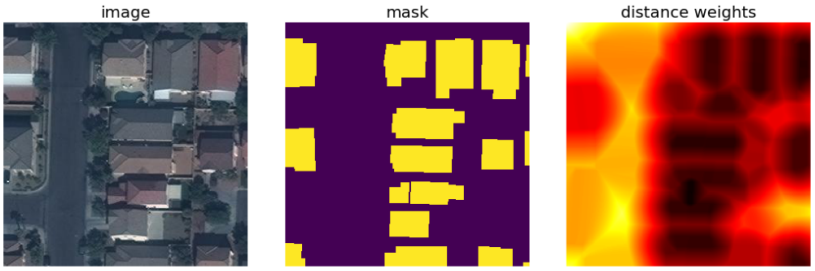

Figure 2.18: Visualization of distance weight

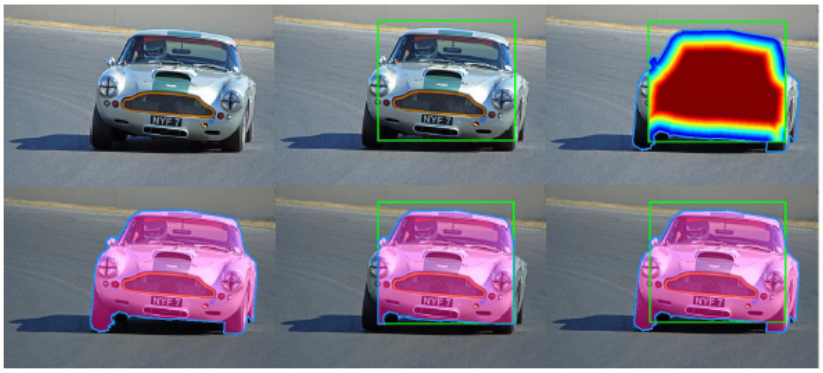

Figure 2.19: Multi-valued map label [13]

3

DATA UNDER STUDY

Ideally, we would train and evaluate a model based on the data relevant to UN-Habitat. However, during the completion of the thesis it became clear that the number of images and the quality of the annotations provided by UN-Habitat are insufficient to solely rely on for training. Therefore, we use some public datasets for training and evaluation. In this chapter, all the datasets that will be used for the following experiments are presented. We also explain the processing steps to transform the dataset into a unified format that can be directly applied to train the model, as we use the default format of the dataset we will use is a bit different. In the end, the evaluation metrics for our model are described, too.

3.1. DATASETS

3.1.1. UN-HABITAT DATASET

UN-Habitat dataset is the satellite images provided by UN-Habitat. Specifically, a set of high-resolution satellite images of Basra, Iraq. These images are owned by the United Nations. Therefore, corresponding annotations are not equipped. We are grateful for the support provided by UN-Habitat that they provide some labels that were extracted manually. But unfortunately, these labels are too rough to use. Figure 3.1 shows some examples of UN-Habitat dataset.

3.1.2. CROWDAI BUILDING DATASET

The CrowdAI building dataset is a dataset provided by Crowd AI for a building mapping challenge [40]. The data are individual tiles of satellite images in the type of RGB images, and their corresponding annotations that show the positions of buildings in a pixel-wise way. Specifically, the size of each image is in a resolution of 300*300, and the annotations are in MS-COCO format [46], including the bounding boxes and polygons for all objects. The training set and validation set consist of 280741 tiles and 60317 satellite images, respectively.

Figure 3.2 shows some examples of CrowdAI building dataset after preprocessing.

3.1.3. INRIA AERIAL IMAGE LABELING DATASET

The Inria Aerial Image Labeling Dataset [10] is created as a benchmark database to evaluate classification techniques and their generalization capabilities. The dataset consists of labeled imagery that

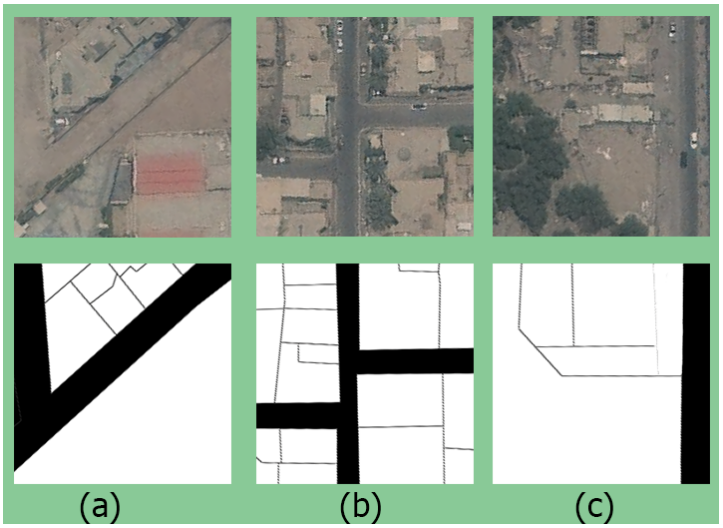

Figure 3.1: UN-Habitat dataset examples

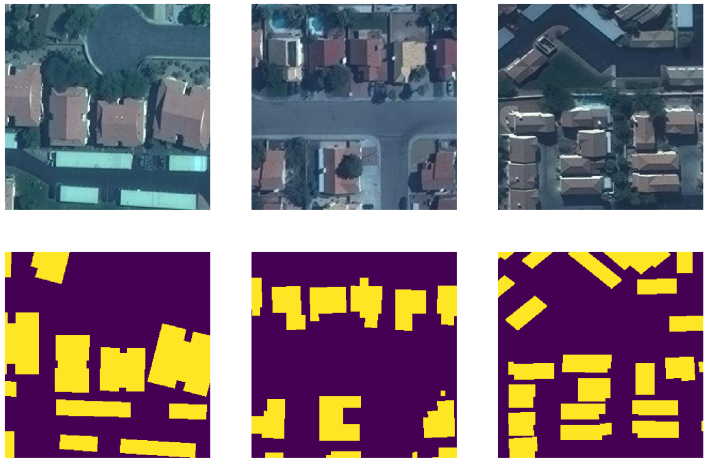

Figure 3.2: CrowdAI building dataset examples

covers varied urban landscapes, such as metropolitan financial districts and alpine resorts. The annotations are binary image masks assigning each pixel in an image into building and non-building classes. Specifically, the training set contains 180 colored image tiles of size 5000×5000 in GeoTiff format, covering a surface of 1500 m×1500 m each.

Figure 3.3 shows some examples of CrowdAI building dataset after preprocessing.

3.2. DATA PREPROCESSING

As we use multiple different datasets – listed in section 3.1 – whose formats are a bit different, we need to do some preprocessing to obtain the training data and ground truth labels that can be directly used to train and evaluate models and methods. All the preprocessing steps are introduced in this section.

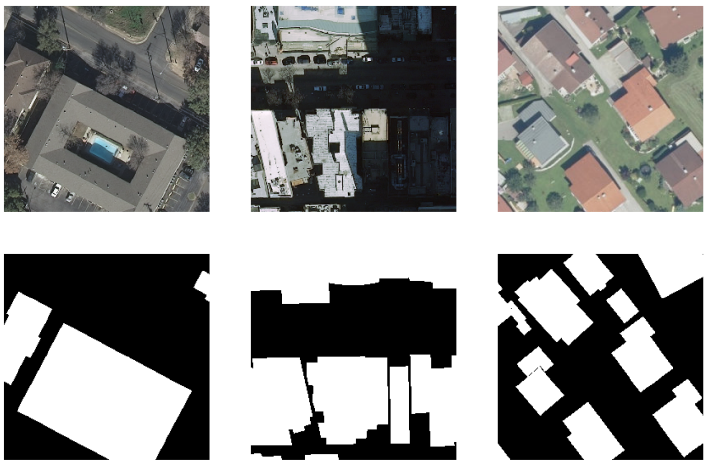

Figure 3.3: Inria Aerial Image Labeling Dataset examples

3.2.1. DATA FORMAT TRANSFORMATION

First, we need to obtain the ground truth annotations, as we have mentioned in section 3.1, the formats of the annotations in the dataset we use are different. What we will use for the training and evaluation of models or methods are RGB images and the corresponding binary labels, which are shown in figures 3.2 and 3.3. Fortunately, Inria Aerial Image Labeling Dataset provides binary labels such that we do not need to obtain the labels by ourselves. But for the CrowdAI Building Dataset, its labels are in MS-COCO format [46] which mark the positions of the bounding boxes and polygons for all objects. To obtain the annotations, we first need to read the coordinates of all objects and then use them to create binary images such that the pixel values of the background are 0 and the pixel values of the buildings are 1. After the transformation, we have both the RGB images and their binary label images with the same size. During the following experiments, we would chip images into disjoint smaller images when necessary to fit the appropriate size.

3.2.2. TFRECORD FILE TRANSFORMATION

Due to the large amount of data that we will use, we need to find a better way to deal with them such that we can make better use of memory and move, read and store these data quickly and conveniently. We decided to use TensorFlow [47] as the main framework which allows to transform the data into TFRecord File format. TFRecord File is a binary file format optimized for use with TensorFlow in multiple ways. Binary data takes up less space on disk, takes less time to copy and can be read much more efficiently. A TFRecord file contains a sequence of records, and the file can only be read sequentially. One of the main advantages is that we are able to specify the structure of our data before writing it to the file, which means that we can store the image data and corresponding binary labels in a unified manner, especially for the new ideas of modifying the binary labels in the following chapters. Specifically: each image can be encoded as a byte list, and its size can be stored as an integer list. When loading the data, we can obtain the image size directly and then decode the image back to the corresponding size.

3.2.3. DATA AUGMENTATION

Recent advances in deep learning models have been largely attributed to the quantity and diversity of data gathered in recent years [48]. Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collect-

ing new data. We provide an option of data augmentation during the training time such that the training data can be augmented if this option is selected. Specifically, we do the data augmentation in the following way: for each training sample, there is an assigned probability that is used to decide whether the sample is augmented. If so, there are four different operations: flipping left, and right(horizontally), flipping up and down(vertically), rotating 90 degrees and rescaling, which are illustrated in figure 3.4. One to four operations would be randomly applied to generate a new sample. Figure 3.5 shows some possible generated examples from data augmentations.

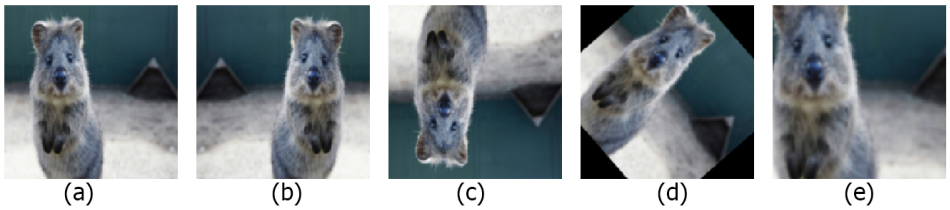

Figure 3.4: Data augmentation operations. (a) original image; (b) flip left and right; (c) flip up and down; (d) rotation(90 degrees); (e) rescale

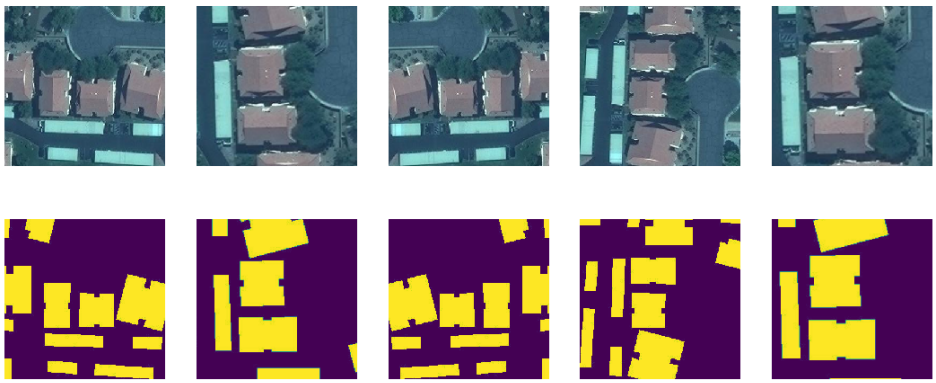

Figure 3.5: Several data augmentation examples. (a) original image; (b) (c) (d) (e) possible augmented samples

3.3. EVALUATION METRICS

To evaluate and compare the performance of different methods or models, standard metrics must be used to ensure fair comparisons. Here we pick up several well-known evaluation metrics that are commonly used for semantic segmentation tasks.

3.3.1. PIXEL ACCURACY

In terms of the evaluation metrics that are frequently used for semantic segmentation, **pixel accuracy(PA)** and **mean pixel accuracy(MPA)** can directly measure the pixel-wise accuracy. Before presenting the corresponding formulae, we need to introduce a notation:

Assume that there are a total of k classes and p_{ij} represents the number of pixels of class i inferred to belong to class j. When $i = j$, p_{ii} is the number of pixels that is correctly inferred to belong to class i, in other words, the number of true positives. When $i \neq j$, p_{ij} and p_{ji} are the number of false positives and false negatives, respectively. With the help of this notation, we can easily formulate the evaluation metrics.

As the most straightforward metric, pixel accuracy is the ratio of the total number of true positives

to the total number of pixels:

$$PA = \frac{\sum_{i=1}^{k} p_{ii}}{\sum_{i=1}^{k} \sum_{j=1}^{k} p_{ij}}. \tag{3.1}$$

3.3.2. MEAN PIXEL ACCURACY

In some cases, the data are unbalanced such that the numbers of pixels belonging to different classes are significantly different. To deal with this, an improved version of pixel accuracy, mean pixel accuracy was proposed by considering the accuracy in a per-class basis. The sum of the pixel accuracy of each class is computed, then it is averaged over the total number of classes.

$$MPA = \frac{1}{k} \sum_{i=1}^{k} \frac{p_{ii}}{\sum_{j=1}^{k} p_{ij}} \tag{3.2}$$

3.3.3. MEAN INTERSECTION OF UNION

Slightly different from the pixel accuracy described above, which straightforwardly measures the ratio of correctly classified pixels, the mean intersection of union, a.k.a MIoU, is another popular criterion for segmentation. In a segmentation task, we are more concerned about whether the target instances are correctly segmented. Intersection over union can be interpreted as the ratio of the intersection area of the ground truth and inferred segmentation to their union area.

[Add a figure here for better explaining]

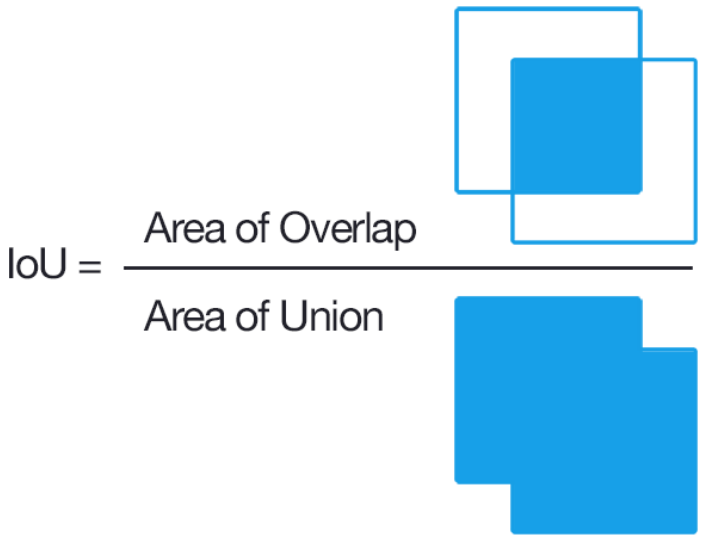

Figure 3.6: A nice visualization to explain MIoU [14]

We can compute MIoU similar to MPA, such that computing the IoU on a per-class basis then averaging the sum. Specifically, computing the MIoU is as simple as dividing the area of overlap between the bounding boxes by the area of the union, which is explained in figure 3.6. Based on the notations, MIoU can be formulated as follows:

$$MIoU = \frac{1}{k}\sum_{i=1}^{k}\frac{p_{ii}}{\sum_{j=1}^{k}p_{ij}+\sum_{j=1}^{k}p_{ji}-p_{ii}} \tag{3.3}$$

For semantic segmentation, MIoU is more important than PA and MPA. Because PA and MPA treat each pixel as an independent unit. But for MiOU, region is the main factor that is considered.

4

INITIAL EXPERIMENT

Before delving into the building segmentation of satellite imagery, we would like to see the performance of the classic semantic segmentation algorithms on the datasets we choose, as these methods are proposed based on other scenarios (e.g. biomedical segmentation, road and vehicle segmentation, indoor object segmentation). We call this step as the initial experiment. In this chapter, we present the process of how the initial experiment is completed and the performance of the selected methods.

4.1. CLASSIC METHODS TO TEST

From previous works [15] [49] , we know that currently there are several ways in which semantic segmentation algorithms are designed, such as transforming from a classification network into a segmentation network and integrating local and global context information. In our case, as our goal is to obtain building segmentation on satellite imagery, firstly, we select several methods of different types to see how they perform on the selected dataset. Specifically, as shown below in figure 4.1, we choose these methods of different types.

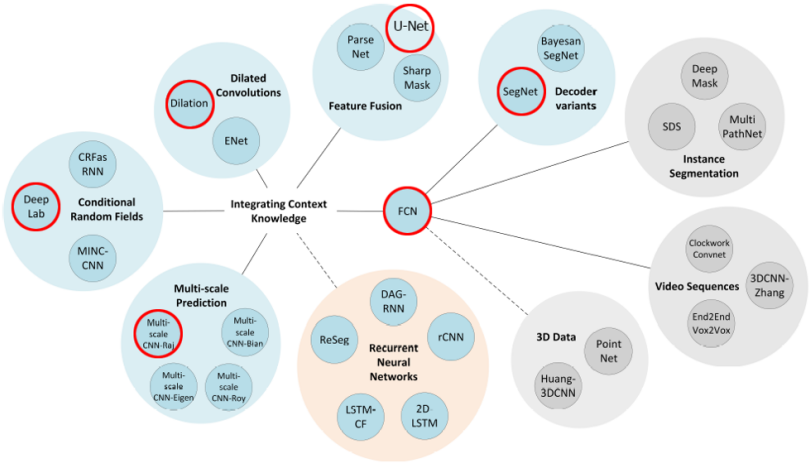

Figure 4.1: Selected segmentation methods used to test. Figure extracted and modified from paper [15]

As some source codes are available from their authors [50] and using some open source repositories [51] [52], we modify or even implement the selected methods withing a unified framework to make

Table 4.1: Result of the initial experiment

Method	Pre-train weights from	CrowdAI dataset			Inria Aerial dataset		
		PA	MPA	MIoU	PA	MPA	MIoU
FCN8S	VGG16	0.9414	0.9128	0.8479	0.9018	0.8735	0.7818
Deep_ab	ResNet-50	0.9266	0.8866	0.8241	0.8883	0.8565	0.7616
SegNet	VGG16	0.9356	0.9096	0.8481	0.8999	0.8676	0.7830
Multi-scale_aj	-	0.9158	0.8956	0.8142	0.8912	0.8601	0.7711
U-Net	-	0.9204	0.9111	0.8206	0.8978	0.8651	0.7819

sure that their building setup and training setup are identical and the only difference is their network architectures.

4.2. EXPERIMENT SETTING

To obtain fair results, we use the same training set and test set which consist 3000 images and 300 images, respectively. The size of all images are 256 × 256 pixels and the number of each training batch is 25. All methods are trained with Adam [53] using a fixed learning rate and the loss is computed by pixel-wise softmax with cross entropy. We use two settings to train the different methods. For methods using pre-trained weights, we use the fixed learning rate of 0.0001 and weight decay of 0.0002 and train them with 1000 epochs. During training, we allow all weights to be updated. For methods training from scratch, we first use the fixed learning rate of 0.05 and weight decay of 0.0005 to train them for 100 epochs and then reduce the learning rate to 0.005 and train them for 1000 epochs. We also apply our data augmentation method during the whole training process with probability 0.5.

4.3. EXPERIMENT RESULT

The following table 4.1 shows the performance of different methods on the test set.

The following observations can be made from the table: Overall, the performances are acceptable, as the numerical values are good. We can notice that models using pre-trained weights have better performances than those trained from scratch. This is easy to understand, as the pre-trained weights already have involved lots of information. But after checking the output segmentation results (showed in figures 5.8, 6.3, 7.2), we realize that there are still some problems existing and the results can still be improved. For example, the boundaries of buildings are failed to be captured.

We will use these results as the baseline results in the following chapters.

5

REWEIGHING METHODS

As can be seen from the results of the initial experiment (4.1), there is still room for further enhancement on accurate segmentation. In this chapter, two methods to reweigh the data are investigated and used to test if they can improve the performance of the building segmentation. First, we present the motivation of our proposal in section 5.1. Then we describe our proposed weight maps and how to compute them in section 5.2 and section 5.3. Finally, the experiments and the result are described in section 5.4.

5.1. MOTIVATION

From the initial experiment, we can see that the performance of classic methods on building segmentation tasks on satellite imagery is acceptable. However, when we check the segmentation results, we observe some problems. We not only hope to achieve excellent segmentation performance but also wish to obtain information on urban development. Therefore, we need to do some further research based on our data and task. Using different pixel weights can force or encourage the deep neural network model to learn accordingly, as the weights adjust the training losses. More specifically: from the training imagery, we can see that the pixels outside but near the building border could be affected or be 'covered' by other factors such as trees and shadows. However, the value of each pixel of label mask is merely binary, either building or non-building, which adds a lot of difficulty to the segmentation. Figure 5.1 shows some examples.

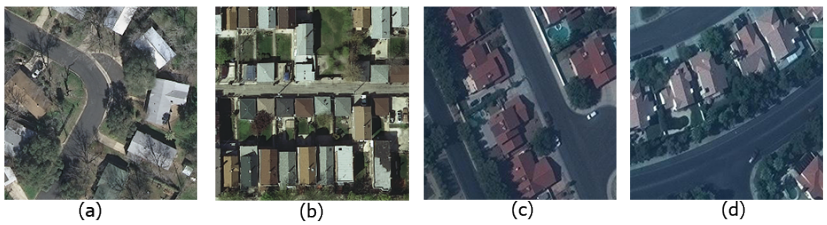

Figure 5.1: Some training image examples

Inspired by the related works that has been mentioned in the previous section 2.5, we propose two types of pixel weights and test them with different methods and datasets. We explain the details in the following sections.

5.2. BORDER WEIGHT

To encourage the model to learn more about the pixels outside but close to the border of the buildings, a straightforward way is to increase the weights of these pixels. The annotation labels represent pixels within the buildings. However, no matter how far from the building, pixels outside the buildings are represented as the same type, which is called non-building pixels. Taking the distance to the buildings into consideration, we can construct a pixel-wise weight map by relying on the transformation of distance. In other words, the weight of every pixel belonging to the background is computed according to the Euclidean distance to the building boundary. According to [13], we use the set of pixels on the object boundary and outside the object as the reference pixels. We called it "border weight," as it depends only on the distance to the border. Specifically, the boundary weight value is greatest at the boundary of the building, moving outward from the boundary of the building and decreasing with distance. We can compute the border weight of each pixel p outside the buildings as

$$B_w(p) = w_0 * \exp^{\frac{-(d_1+d_2)}{2\sigma_0}} \tag{5.1}$$

where d_1, d_2 denote the nearest and second nearest Euclidean distances to the reference pixels. w_0 and σ_0 are two parameters that control the value of the border weight. Specifically, the value of w_0 decides the maximum value of the border weight. The value of σ_0 controls the decrease of the equation. Figure 5.2 shows an example surface of the border weight value equation when w_0=10 and σ_0=0.5.

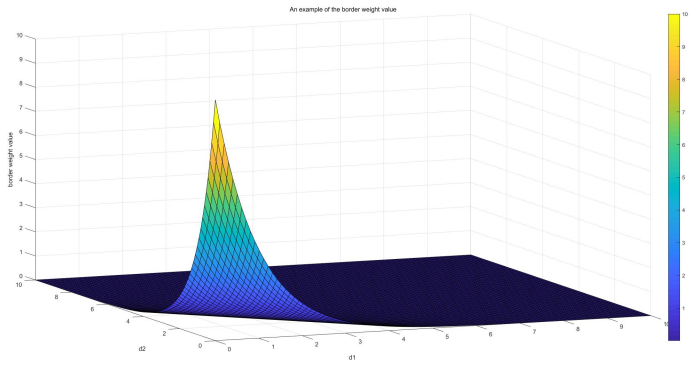

Figure 5.2: A example of the border weight value when w_0=10 and σ_0=0.5

Using an image as an example, figure 5.3 shows the steps to obtain the border weight. From the binary annotation label, we can receive the reference labels by comparing the value of each pixel with adjacent pixels. For each non-building pixel, first, we compute and obtain the nearest and second nearest Euclidean distances to the reference pixels. Then we can calculate its border weight with equation 5.1.

When w_0 and σ_0 hold different values, the results of border weight are different. Figure 5.4 shows different border weights of one image with different values of parameters.

5.3. INTER-BUILDING DISTANCE WEIGHT

Border weight encourages the model to learn more about the non-building pixels near buildings. However, from the predicted images of the initial experiment, another problem occurs that when

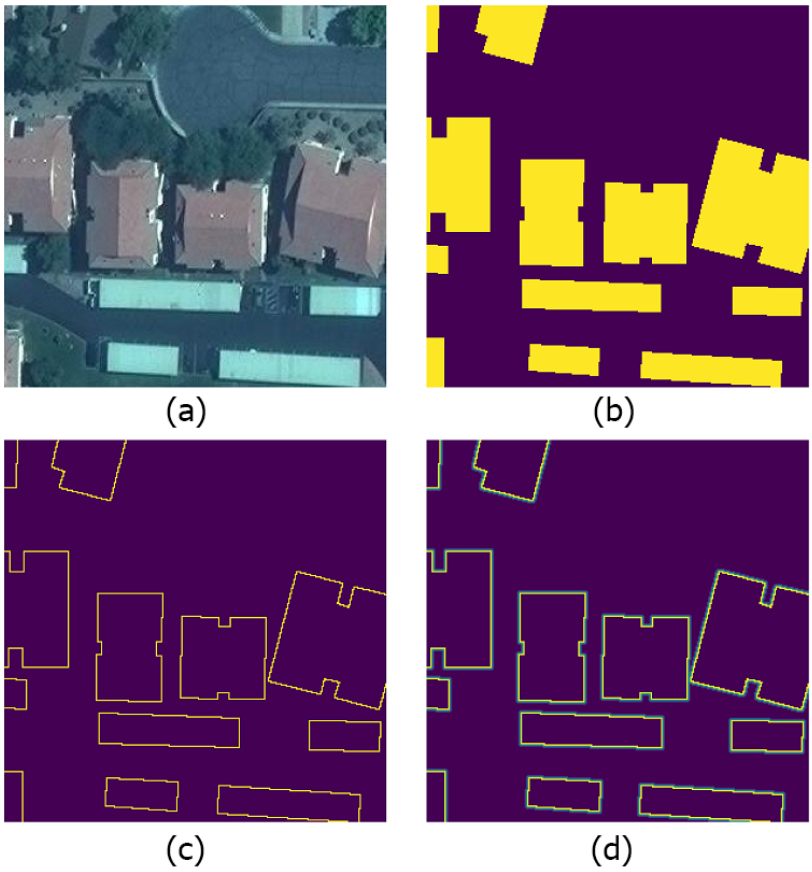

Figure 5.3: Steps to obtain the border weight map (a)original image, (b) binary annotation label, (c) reference pixels (building boundary), (d) pixel-wise border weight.

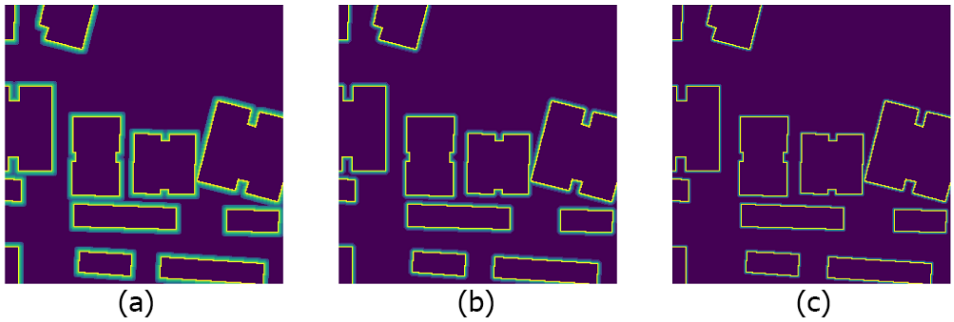

Figure 5.4: Border weight maps when using fixed $w_0 = 10$ and different σ_0 values: (a) $\sigma_0 = 0.2$, (b) $\sigma_0 = 0.5$, (c) $\sigma_0 = 0.8$

.

two buildings are too close, the non-building pixels are often inferred as building pixels. As the building count is one of the information that we hope to obtain, border weight cannot address this issue. To improve the performance, we introduce the inter-building distance weight.

The reweighing method used in U-Net [6] inspires us, which considers the distances to the nearest cell. From figure 2.10, we can see that mostly the interval between two cells is too small, which is different from our building segmentation task.

In our scenario, the interval sizes between buildings are various. To emphasize the pixels laid in between the buildings, we can reweigh them based on the distances to the nearest and second near-

est buildings. We call it "inter-building distance weight" to reflect this idea, and the weight value should highly depend on the distance to the nearest building than the second nearest building. We can compute the inter-building distance weight of each pixel p outside the buildings as

$$I_w(p) = w_1 * \exp^{\frac{-(d_1^2+d_2)}{2\sigma_1^2}} \tag{5.2}$$

where d_1, d_2 denote the Euclidean distances to the nearest and second nearest buildings. To emphasize the importance of the distance to the nearest building, here we use d_1^2 and only use d_2 as a difference. w_1 and σ_1 are two parameters that control the value of the inter-building distance weight. Specifically, the value of w_1 decides the maximum value of the inter-building distance weight. The value of σ_1 controls the decrease of the equation. Figure 5.5 shows an example surface of the inter-building distance weight value equation when w_1=10 and σ_1=5.

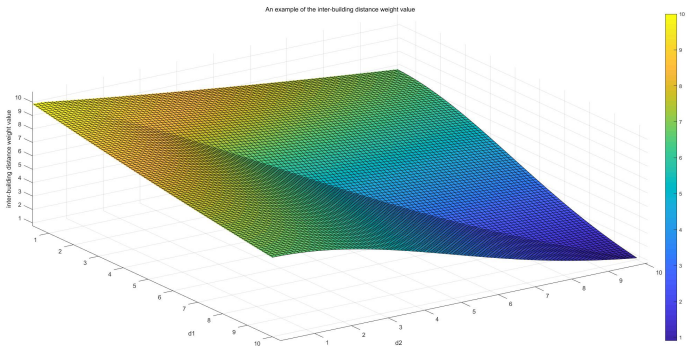

Figure 5.5: A example of the inter-building distance weight value when w_1=10 and σ_1=5

Figure 5.6 shows an example of the steps to obtain the inter-building distance weight. From the binary annotation label, we can get the coordinates of the pixels belonging to the buildings. Then we aggregate pixels belonging to the same building into a group. For each non-building pixel, first, we compute and obtain the Euclidean distances to the nearest and second nearest buildings. Then we can calculate its inter-building distance weight with equation 5.2.

When w_1 and σ_1 hold different values, the results of inter-building distance weight are different. Figure 5.7 shows different border weight of one image with different values of parameters.

5.4. EXPERIMENT

To test if the border weight and inter-building distance weight are helpful to improve the building performance and see how their parameters affect the performance, we do the following experiment.

According to the experiment result table 4.1 of the initial experiment, here we choose three segmentation methods FCN8S [32], SegNet [7] and Deep_Lab [**?**] to test the performance. The baseline is the performance when training the data without border weight or inter-building distance weight. To evaluate the border weight, we set up different values for the parameters w_0 and σ_0, such that $w_0 \in [2,5,10,20]$ and $\sigma_0 \in [0.2,0.5,0.8]$. To assess the inter-building distance weight, we set up different values for the parameters w_1 and σ_1, such that $w_1 \in [2,5,10,20]$ and $\sigma_1 \in [2,5,10]$. Therefore, for each weight, we train the models with 12 different settings.

We use a similar experiment setting as the initial experiment. At this time, pre-trained weights are used for all methods, and we use the fixed learning rate of 0.0001 and weight decay of 0.0002 and

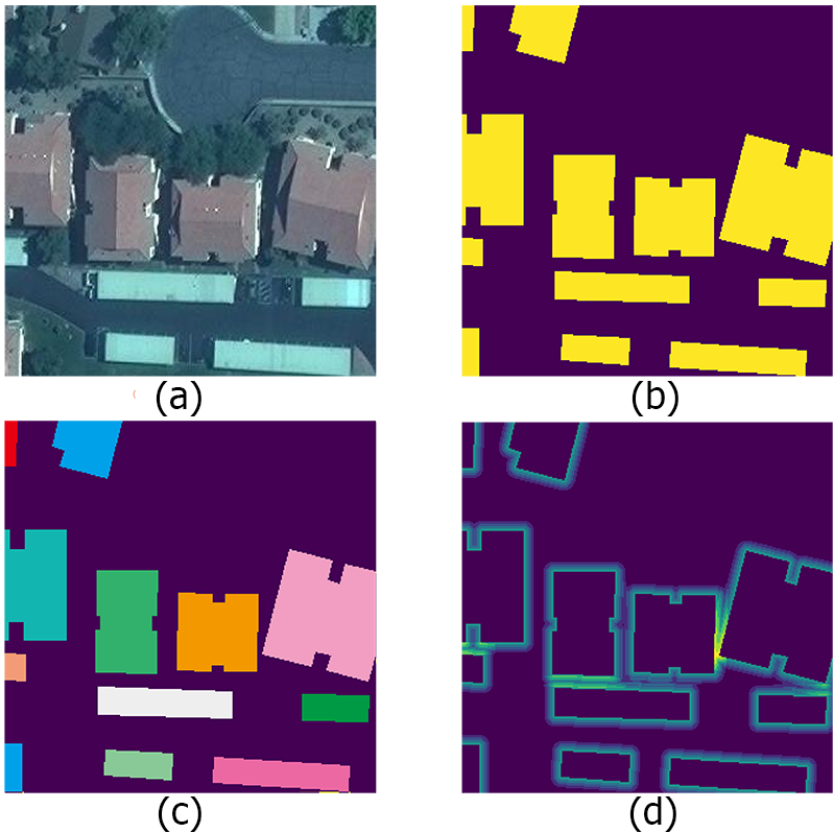

Figure 5.6: An example of different representation of an image. (a)original image, (b) binary annotation label, (c) different building groups, (d) pixel-wise inter-building distance weight.

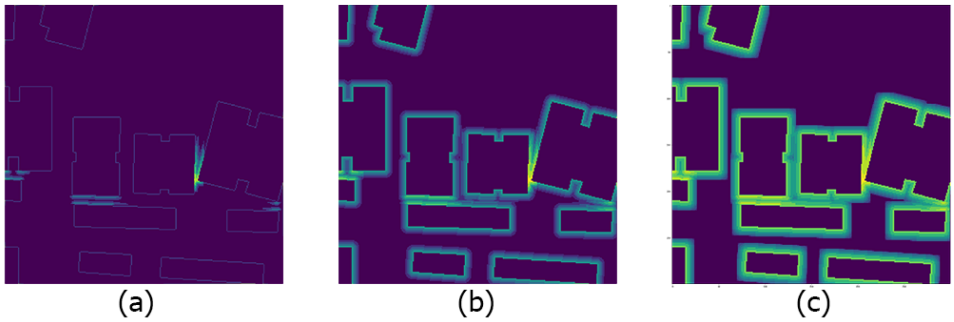

Figure 5.7: Border weight maps when using a fixed $w1$ and different σ_1 values: (a) $\sigma_1 = 3$, (b) $\sigma_1 = 5$, (c) $\sigma_1 = 8$

train them with 1000 epochs. During training, we allow all weights to be updated. Data augmentation is applied during the whole training process with probability 0.5.

The following tables 5.1 and 5.2 show the performance of the baseline and the best performance of the border weight.

The full results are presented in the appendix 9.2

From the results, we can notice that with the help of border weight, the performances can be improved on the selected methods. When setting the parameters w_0 and σ_0 properly, We can see a small performance improvement. Here we only apply a simple grid search on the parameters to see how the methods perform when using border weight. For different datasets, the optimal parameter values can be different. It is quite hard to obtain a group of value that can be applied to every

Table 5.1: Border weight results on Crowded AI dataset

Methods	Baseline			Best Performance			Best parameter values
	PA	MPA	MIoU	PA	MPA	MIoU	
FCN8s	0.9414	0.9128	0.8479	0.9515	0.9228	0.8609	$w_0 = 20, \sigma_0 = 0.2$
SegNet	0.9356	0.9096	0.8481	0.9467	0.9178	0.8600	$w_0 = 2, \sigma_0 = 0.2$
Deep_Lab	0.9266	0.8866	0.8241	0.9370	0.9032	0.8370	$w_0 = 2, \sigma_0 = 0.2$

Table 5.2: Border weight results on Inria Aerial dataset

Methods	Baseline			Best Performance			Best parameter values
	PA	MPA	MIoU	PA	MPA	MIoU	
FCN8s	0.9018	0.8735	0.7818	0.9122	0.8841	0.7988	$w_0 = 10, \sigma_0 = 0.5$
SegNet	0.8999	0.8676	0.7830	0.9156	0.8948	0.8076	$w_0 = 5, \sigma_0 = 0.8$
Deep_ab	0.8883	0.8565	0.7616	0.9009	0.8804	0.7803	$w_0 = 2, \sigma_0 = 0.2$

dataset. But we can notice that for Crowded AI dataset, the best parameter values for all methods are close, especially the values of σ_0 are the same. Also, there is no significant increase in running time due to the use of border weight. Another advantage is that we do not need to modify the existed method but only need to compute the border weight map for the training data. Calculating the border weight maps is pretty quick, and as we mentioned in section 3.2.2, we can compute and save the weight maps during the data preprocessing step.

The following tables 5.3 and 5.4 show the performance of the baseline and the best performance of the border weight.

The full results are presented in the appendix 9.2

From the results, we can notice that the performances can be improved when using inter-building distance weight.

Similarly, the enhancement is dependent on the values of the parameters w_1 and σ_1, and obtaining a group of value that can be applied to every dataset is infeasible. The running time is almost unchanged, and we can just run the existed methods without modifying. However, calculating the inter-building distance weight maps is not that quick, and the data preprocessing step takes some time.

From the evaluation metrics, we notice that the improvement is slight, and it is hard to explain the differences between these two proposed reweighing methods. Let us look at the output images. Figure 5.8 shows two examples of the results. Obviously, the result looks better when using reweighing methods. The border of buildings can be classified more correctly than the baseline. However, we also can notice some errors in the corner of the result examples. Listing the difference between border weight and inter-building distance weight is not accessible from the results. One possible reason is that the border pixels of the building are also highlighted to some extent in the inter-building dis-

Table 5.3: Inter-building distance weight results on Crowded AI dataset

Methods	Baseline			Best Performance			Best parameter values
	PA	MPA	MIoU	PA	MPA	MIoU	
FCN8s	0.9414	0.9128	0.8479	0.9448	0.9154	0.8556	$w_1 = 2, \sigma_1 = 2$
SegNet	0.9356	0.9096	0.8481	0.9477	0.9218	0.8629	$w_1 = 5, \sigma_1 = 5$
Deep_ab	0.9266	0.8866	0.8241	0.9353	0.9021	0.8332	$w_1 = 2, \sigma_1 = 10$

Table 5.4: Inter-building distance weight results on Inria Aerial dataset

Methods	Baseline			Best Performance			Best parameter values
	PA	MPA	MIoU	PA	MPA	MIoU	
FCN8s	0.9018	0.8735	0.7818	0.9061	0.8856	0.7898	$w_1 = 5, \sigma_1 = 5$
SegNet	0.8999	0.8676	0.7830	0.9153	0.8787	0.8021	$w_1 = 10, \sigma_1 = 5$
Deep_ab	0.8883	0.8565	0.7616	0.9015	0.8701	0.7769	$w_1 = 20, \sigma_1 = 5$

tance weight map. Therefore their performances are similar.

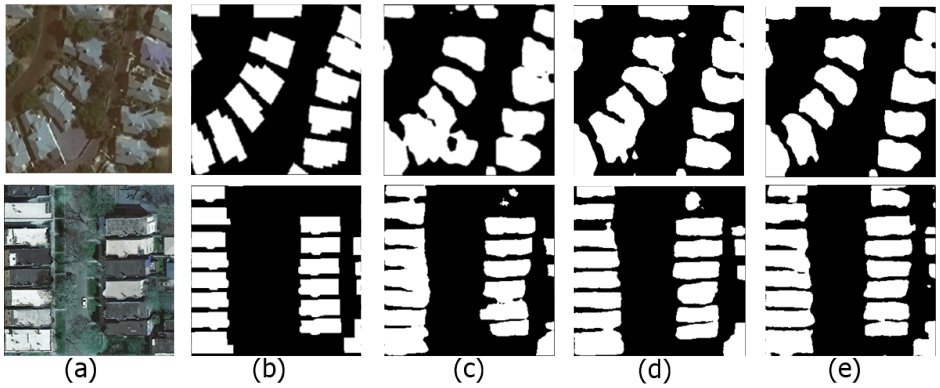

Figure 5.8: Two image examples: (a)original image, (b) ground truth, (c) baseline result, (d)border weight result, (e)inter-building distance weight result

6

MULTI-LEVEL BOUNDARY LABEL

From chapter 5, we learned that we can reweigh the non-building pixels to improve the segmentation performance. The conclusion inspires us: can we come up with an idea using the pixels belonging to the buildings? In this chapter, we propose a new label representation based on binary labels. First, the motivation of our proposal is presented in section 6.1. Then we describe our proposal and how to compute it in section 6.2. Last, we evaluate the performance when applying our proposal in section 6.3.

6.1. MOTIVATION

In the previous chapter we devoted some time to further improve the initial experiment, and we showed that the the performance can improve without modifying the network architectures by applying reweighing methods to the pixels belonging to the background. This raises a new question: can we improve the performance by considering the building pixels? We can notice that the building pixels near the building border can also be affected by trees and shadows, showed in figure 5.1.

The analyses presented in this chapter is inspired by the paper [13], in which a new object segment representation was proposed. This representation was robust to errors in the bounding box proposals. In our case, we do not have the bounding boxes in our dataset. We can present the boundary of buildings better without the limitation of the bounding box. Next, we explain the details of our proposal.

6.2. MULTI-LEVEL BOUNDARY LABEL

In chapter 5, we reweighed each non-building pixel near to building boundaries to encourage the model to learn more about them. Here we focus on the building pixels. The probability that a building pixel near the boundary is misclassified is relatively high. To address this issue, we introduce the multi-level boundary label to highlight the building pixel according to its distance to the boundary. We modify the binary label into multi-level boundary label in the following way:

A set of building boundary B is the set of closest non-building pixels around the buildings. Then we can compute the distance from every pixel p representing the building to the building boundary as:

$$D(p) = \min_{\forall b \in B} \lceil d(p,b) \rceil \tag{6.1}$$

where $\lceil x \rceil$ is the ceiling function that maps x to the least integer greater than or equal to x, and $d(p,b)$ is the Euclidean distance between pixel p and b. To focus on the pixels near the boundary, a threshold T is used to represent the largest distance. Therefore, the truncated distance pixel-wise map can be computed as:

$$D(p) = \min(\min_{\forall b \in B} \lceil d(p,b) \rceil, T) \tag{6.2}$$

From figure 6.1 we can see that the values of the truncated distance map are continuous and the number of pixels with a truncated distance between 0 and threshold T is small. To further facilitate the truncated distance map, we quantize the distance values into L levels such that we cluster the pixels into L classes according to their distance values. We call this quantized map as multi-level boundary label. Compared to the binary mask, We can see that the multi-level boundary label captures not only the buildings but also their shapes and boundaries.

Figure 6.1 shows the steps to get the multi-level boundary label. The threshold T and the number of level L are two parameters that can be set and figure 6.2 shows three different settings.

Figure 6.1: Steps to obtain the multi-level boundary label. (a) Satellite image, (b) Binary label, (c) Building boundaries, (d) Truncated distance map, (e) Multi-level boundary label.

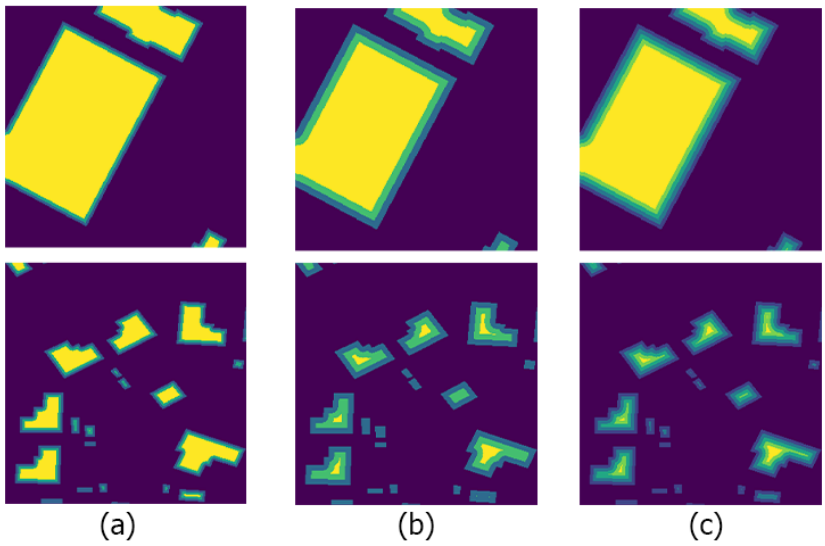

Figure 6.2: Multi-level boundary labels with different parameter values: (a) T=10 and L=3, (b) T=20 and L=3, (c) T=20 and L=5.

In case of using multi-level boundary labels as training labels, we are facing a multi-class segmentation task. Since the number of pixels of each class is imbalanced, we use median frequency weights[54] to reweigh each class in the loss function. Specifically, the weight of each class c is

computed as:

$$w_c = median\_freq / freq(c) \tag{6.3}$$

where $freq(c)$ is the relative frequency of class c which is the number of pixels of class c divided by the total number of pixels and $median\_freq$ is the median frequency of all classes.

6.3. EXPERIMENT

To test if the multi-level boundary label is useful to improve the performance of the segmentation methods and see how its parameters affect the performance, we do the following experiment.

Similarly, we we choose FCN8S [32], SegNet [7] and Deep_Lab [**?**] as the methods for comparison. The baseline is the performance of the initial experiment. For the experiment, we first use the same training images with the multi-level boundary label to train the model. Here we use a simple post-processing step such that if the pixel is inferred as belonging to the non-background class, we will regard it as a pixel representing the building. After that, we obtain binary segmentation results, and we can use the evaluation metrics to assess the performances.

Before experimenting, we have used grid search to check the reasonable range of the parameters on our datasets. As it takes a long time to obtain the multi-level boundary label on a big dataset, here we present the results of the following three groups of parameter values: (*i*) $T = 10$, $L = 3$; (*ii*) $T = 20$, $L = 3$, (*iii*) $T = 20$, $L = 5$.

At this time, pre-trained weights are used for all methods, and we use the fixed learning rate of 0.0001 and weight decay of 0.0002 and train them with 1000 epochs. Data augmentation is applied during the whole training process with probability 0.5.

The following tables 6.1 and 6.2 show the results on both datasets. Compared with the baseline, we can see that the performance is dependent on the parameter values. However, when using inappropriate parameter values, methods could perform even worse. If we can find the appropriate parameter values and use them, the performances can be improved when applying multi-level boundary label. The running time is almost equal, although we transform the binary segmentation task into a multi-class segmentation task. But compared with the reweighing methods in chapter 5, obtaining the multi-level boundary label takes much longer time. It is a bit annoying but worthwhile.

Table 6.1: Multi-level boundary label results on CrowdAI dataset

Method	Baseline			T=10, L=3		
	PA	MPA	MIoU	PA	MPA	MIoU
FCN8S	0.9414	0.9128	0.8479	0.9360	0.9057	0.8377
Deep_Lab	**0.9266**	**0.8866**	**0.8241**	0.9218	0.8824	0.8049
SegNet	0.9356	0.9096	0.8481	0.9402	0.9023	0.8449
	T=20, L=3			T=20, L=5		
FCN8S	**0.9431**	**0.9141**	**0.8537**	0.9423	0.9017	0.8489
Deep_Lab	0.9223	0.8790	0.8047	0.9192	0.8745	0.7980
SegNet	**0.9442**	**0.9115**	**0.8553**	0.9422	0.9080	0.8504

Let us have a look at the inferred images. Figure 6.3 shows three examples of the results. The multi-class segmentation results are good because the boundaries or the outlines of the buildings can be captured correctly, which means that the model does learn the multi-level feature during the training steps. But we can still notice that the shape of the building influences the result. The segmentation result is good when the building shape is regular and is relatively bad for an irregular building shape.

Table 6.2: Multi-level boundary label results on Inria Aerial dataset

Method	Baseline			T=10, L=3		
	PA	MPA	MIoU	PA	MPA	MIoU
FCN8S	0.9018	0.8735	0.7818	0.9238	0.8564	0.7875
Deep_Lab	0.8883	0.8565	0.7616	**0.9071**	0.8481	0.7556
SegNet	0.8999	0.8676	0.7830	0.9281	0.8584	0.7961
	T=20, L=3			T=20, L=5		
FCN8S	0.9195	0.8591	0.7807	**0.9282**	**0.8748**	**0.8022**
Deep_Lab	**0.9071**	**0.8651**	**0.7621**	0.9084	0.8535	0.7598
SegNet	**0.9283**	**0.8784**	**0.8036**	0.9263	0.8730	0.7980

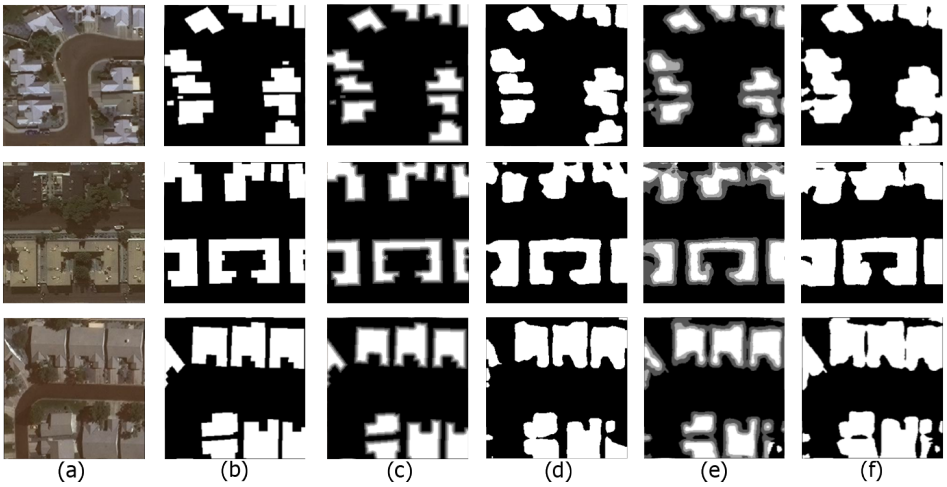

Figure 6.3: Three image examples: (a) original image, (b) binary label (ground truth), (c) multi-level boundary label, (d) baseline, (e) multi-class segmentation result (f) post-processing result

The reason why the evaluation metrics do not improve as much as expected could be the post-processing step we are using. Here we simply transform the multi-class result into a binary segmentation result. Compared with the ground truth, we can see that some non-building pixels around the building borders are misclassified as building pixels. When applying other post-processing methods, for example, a pixel is classified as building pixel only if its probability higher than a threshold. The segmentation results look better. Figure 6.4 shows the difference between these two different post-processing steps. Therefore, a great post-processing step is helpful to the segmentation task in this case. We will not further dig into the post-processing step here, as we aim to focus on the building segmentation.

In conclusion, when using multi-level boundary label, the model can learn more information about the buildings. However, appropriate parameter values are not that easy to find, and it takes some time to obtain the multi-level boundary label.

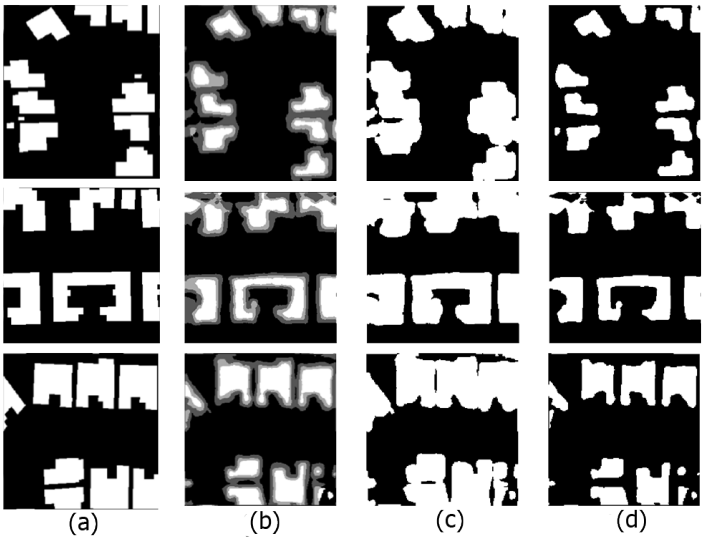

Figure 6.4: Three image examples when using different post-processing steps: (a) ground truth, (b) multi-class segmentation result, (c) simply transform the multi-class result into a binary segmentation result, (d) Using a threshold to do the transformation

7

PROPOSED MODEL

According to the experimental results of the previous chapters, we can see that the multi-level boundary label, border weight, and inter-building distance weight do help to improve the performance. Based on the previous experiment results, we propose a model that can make good use of these properties. In this chapter, we will explain the details of our proposed model and compare its performance with other segmentation methods.

7.1. DETAILS OF OUR PROPOSED MODEL

We have verified that a model can learn more information from the multi-level boundary label. But pixels near the building borders are misclassified with a higher probability. Therefore, we also use border weight and inter-building distance weight to encourage the model to perform better. In our approach, we use two branches to make our model learn from both multi-level boundary label and binary label. The idea of using two branches is to encourage the model to learn from both: the ground truth binary label and also the multi-level boundary label that captures the detailed information about the buildings.

The architecture of our proposed model is shown in figure 7.1.

To extract abstract feature representations from the inputs, the decoder consists of a series of CNN layers. In practice, we use the first 13 convolutional layers of the VGG16 [55] architecture trained on ImageNet [56] as the encoder to utilize their pre-trained weights. Following the shared encoder, the decoder consists of two branches. We upsample the feature maps as follows: we use deconvolution to upsample the feature maps from the previous layer, then we concatenate the upsampled feature maps with the features in the corresponding size, either from the encoder or the branch using multi-level boundary label. The first branch (marked as *'Branch1'* in figure 7.1) can learn some high-level features representing the buildings, with the help of the multi-level boundary label and some low-level features from the decoder. We can use these high-level features for the second branch (marked as *'Branch2'* in figure 7.1) to obtain the final binary segmentation output.

The output of the first branch is a multi-class segmentation result, and a binary segmentation result is the output of the second branch. We can compute the losses with the corresponding type of label.

We use the pixel-wise cross-entropy as the loss function for each branch. For the first branch, we apply the median frequency weights [54] to reweigh each level of the label, and for the second branch, we use the border weight and inter-building distance weight to reweigh the non-building pixels. We

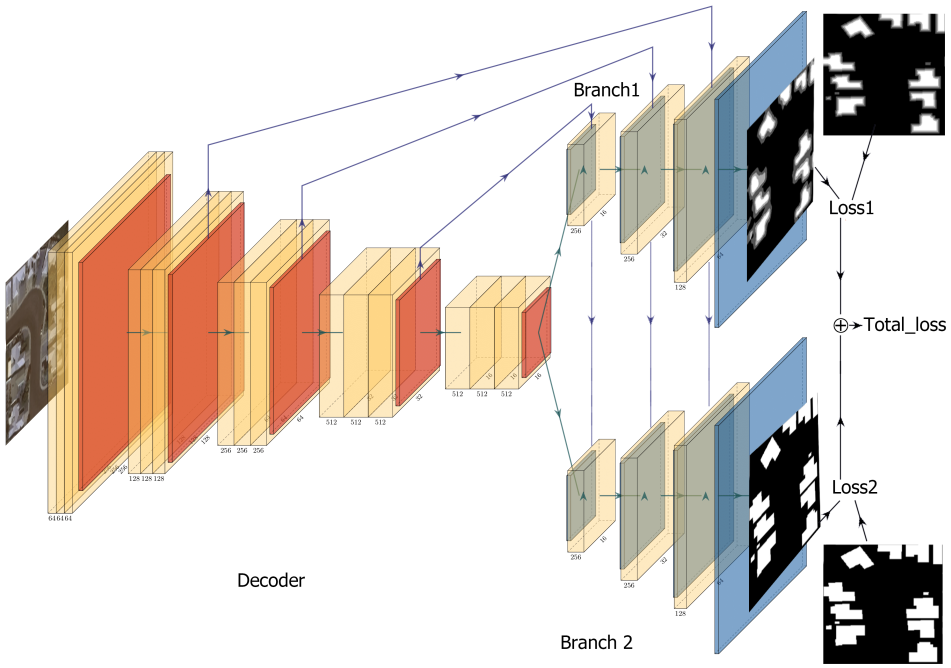

Figure 7.1: The architecture of our proposed model.

7

use a joint loss function which combines the losses from two branches.

7.2. EXPERIMENT

We do the following experiments to see how our model performs and compare it with other methods.

First, we compare the performance of our model with the initial experiment. We use the same training set and test set that are used for the initial experiment. According to the results in previous chapters, we choose the values of parameters that produce an excellent performance for obtaining the border weight, inter-building distance weight, and multi-level boundary label. As our proposed model uses pre-trained weights, we use weight decay of 0.002 and train them with 1000 epochs. For the learning rate, we choose the initial learning rate of 0.0001 and reduce the learning rate to half for every 1000 iterations. We also apply our data augmentation method during the whole training process with probability 0.5. To analyze the effect brought by border weight or inter-building distance weight, we also test the performance of our model when not using both of the proposed reweighing methods.

To analyze the effect casued by border weight or inter-building distance weight, we test the performance of our model when not using both of the proposed reweighing methods.

The following tables 7.1 shows the performances on both datasets.

We can see that our proposed model has a better performance than the baseline, even when not using both of the reweighing methods. Our model can perform better when getting the optimal parameter values of the reweighing methods and multi-level boundary label. Regarding the impact of the reweighing methods on the performance, we can notice that our proposed model performs better when using both reweighing methods. This is another evidence showing that our proposed reweighing methods can improve segmentation performance.

Table 7.1: Performances of segmentation methods on both datasets

Method	CrowdAI dataset			Inria Aerial dataset		
	PA	MPA	MIoU	PA	MPA	MIoU
FCN8S	0.9414	0.9128	0.8479	0.9018	0.8735	0.7818
Deep_Lab	0.9266	0.8866	0.8241	0.8883	0.8565	0.7616
SegNet	0.9356	0.9096	0.8481	0.8999	0.8676	0.7830
Multi-scale_Raj	0.9158	0.8956	0.8142	0.8912	0.8601	0.7711
U-Net	0.9204	0.9111	0.8206	0.8978	0.8651	0.7819
P_model_NoB [a]	0.9454	0.9137	0.8523	0.9236	0.8887	0.8133
P_model_NoI [b]	0.9469	0.9128	0.8488	0.9221	0.8909	0.8069
P_model_NoBoI [c]	0.9361	0.9108	0.8433	0.9180	0.8831	0.8043
P_model [d]	**0.9543**	**0.9273**	**0.8743**	**0.9419**	**0.9054**	**0.8434**

[a] Performance of our proposed model without using border weight
[b] Performance of our proposed model without using inter-building distance weight
[c] Performance of our proposed model without using inter-building distance weight or border weight
[d] Performance of our proposed model

Let us have a look at some examples of the segmentation results. In figure 7.2, we can see differences between the results of our proposed model and the baseline.

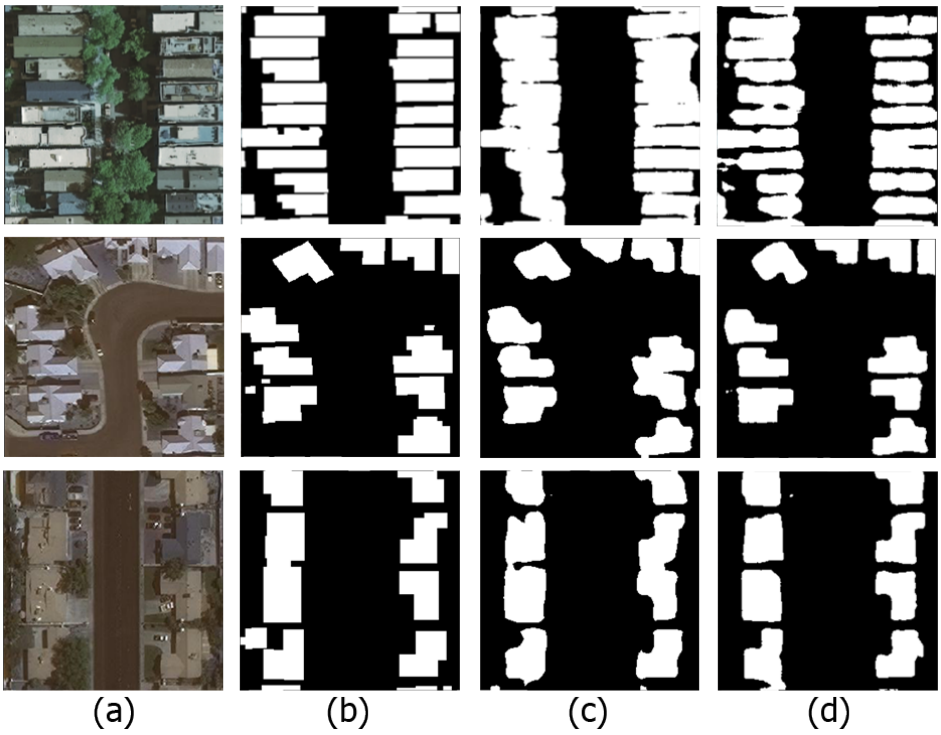

Figure 7.2: Selected examples of the segmentation result: (a) original images, (b) ground gruth, (c) baseline result, (d) proposed model result.

For building pixels, most of them can be detected. However, in some situations such as the building size is too small, our model fails to detect them. Non-building pixels, particularly the pixels laid in between the buildings, are classified more correctly. The boundaries of the buildings are more

clearly reflected, although some misclassifications still exist. As a result, the building segmentation result looks better such as the similarity of the building shapes between the result and the ground truth.

Then let us compare the number of parameter of our proposed model with other segmentation methods, which represents the complexity of the model and memory usage during training to some extents.

The details are listed in table 7.2. Although it is hard to interpret how the prediction is made from a deep learning model, we can explain why our proposed model uses fewer number of parameter than the other models.

Table 7.2: The number of parameter of different models

Method	Number of parameter
FCN8S	133M
SegNet	29M
Deep_Lab	27M
U-Net	7M
P_model	16M

Our testing Deep_Lab [**?**] is built up based on the backbone ResNet-50 [57] and its number of parameter is heavily influenced by ResNet-50. FCN8S [32], SegNet [7] and our model all use the pre-train weights from VGG16[55]. However, FCN8S uses all pre-trained weights of VGG16, as it transforms the fully connected layers into convolution layers. SegNet and our model only use the pre-trained weights from the first 13 convolution layers. As we know, normally the parameter number of fully connected layer is much larger than the convolutional layer. That's why FCN8S has such a large number of parameter. In the decoder part, SegNet uses the max-pooling indices to upsample the feature maps, which reduces the number of parameters compared with using deconvolution. However, the upsampled feature maps are sparse, illustrated in figure 2.11. To produce dense feature maps, a set of convolutional layers are added following every upsampling layer. For our proposed model, we do not need to use any convolutional layer in both decoder branches.

Even though our model looks complex as we have two branches and use a joint loss function, the number of parameters is not that large.

We also need to know the additional requirement of our model: (*i*) border weight maps; (*ii*) inter-building distance weight maps, (*iii*) multi-level boundary label. The computation time and extra space to store them should be considered, especially for large dataset.

7.3. CONCLUSION

In conclusion, we proposed a model that achieves state-of-the-art building segmentation performance. We apply the multi-level boundary label and two reweighing methods on our model and build up the model with a less number of parameter, compared with several classic segmentation methods.

However, the performance of our model depends on the hyperparameters of the multi-class label and the reweighing functions. Currently, we do not have a great way to obtain the optimal value of them. Also, the time of computing the multi-class label and the weight maps and the extra space to store them should be noticed.

8

URBAN INFORMATION

The goal of UN-Habitat is to learn about urban development from satellite images, specifically, building counts and built-in areas of regions like the Middle East. This means that some post-processing steps are needed to obtain the information from the segmentation results. In this chapter, we will explain the post-processing steps for obtaining the building counts and building areas from the building segmentation results. We hope to see whether our proposed model works well on the satellite images provided by the UN-Habitat. We will use transfer learning to see the performance of our proposed model on a new dataset.

8.1. POST-PROCESSING

We can see that the outputs of our proposed model are binary images representing the building and non-building information of the input images. Before describing the way to obtain the building counts and building areas, let us have a look at their definitions. Building count: the number of the building within an image. A single building is formed by connecting adjacent building pixels horizontally and vertically. Building size: the number of pixels forming a building.

It is straightforward to obtain the building counts and building sizes. We collect all coordinates of pixels representing buildings and aggregate pixels belonging to the same building according to their connectivities. Then we can know the size of each building, which is the number of pixel of each group, and also the total number of buildings, which is the number of aggregated groups.

As an example, from figure 8.1, we list the extracted information in table 8.1.

Table 8.1: Urban development information from a example segmentation result

Buidling number	Building size (pixels)
1	754
2	6355
3	5567
4	649
5	4565
6	5162
7	1935

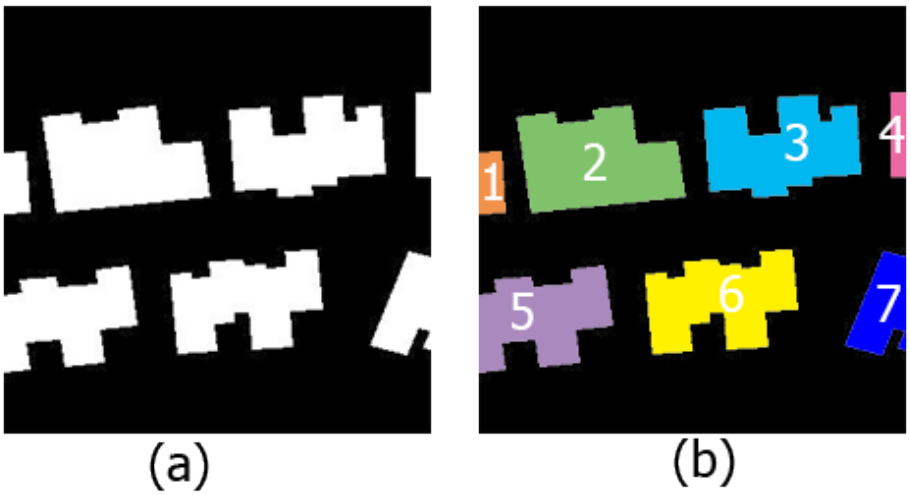

Figure 8.1: An example to get the information on urban development: (a)segmentation result, (b) post-processing result (building numbers and colors are added manually for a better understanding)

8.2. TRANSFER LEARNING EXPERIMENT

To see the performance of our proposed model on the UN-Habitat dataset, we train our model with public datasets and then use transfer learning to fit the UN-Habitat dataset. At this time, we combine the CrowdAI building dataset and Inria Aerial image labeling dataset together to train our proposed model. Specifically, we use the 6000 images (3000 for each dataset) as the training data to train our proposed model using the same settings as the previously experiments. We use the values of parameters that produce an excellent performance based on the results in previous chapters, for obtaining the border weight, inter-building distance weight, and multi-level boundary label. As we only have 170 images with rough label, here we use 150 images provided by UN-Habitat to fine-tune the model, using a learning rate of 0.0001. We allow all weights to be updated during training, and we run the model for 100 epochs. The resting 20 images are used as a test set to check the results.

Some example results are shown in figure 8.2.

We can clearly see the differences between the labels and the segmentation results. However, it does not mean that the segmentation results are really bad. If we look at these results carefully, we can notice that the quality of the labels is very low and the segmentation results are much better. In this case, we can say that our proposed model performs well on a new dataset and we can imagine if we have a sufficient amount of satellite images with good quality labels, the segmentation results will be much better.

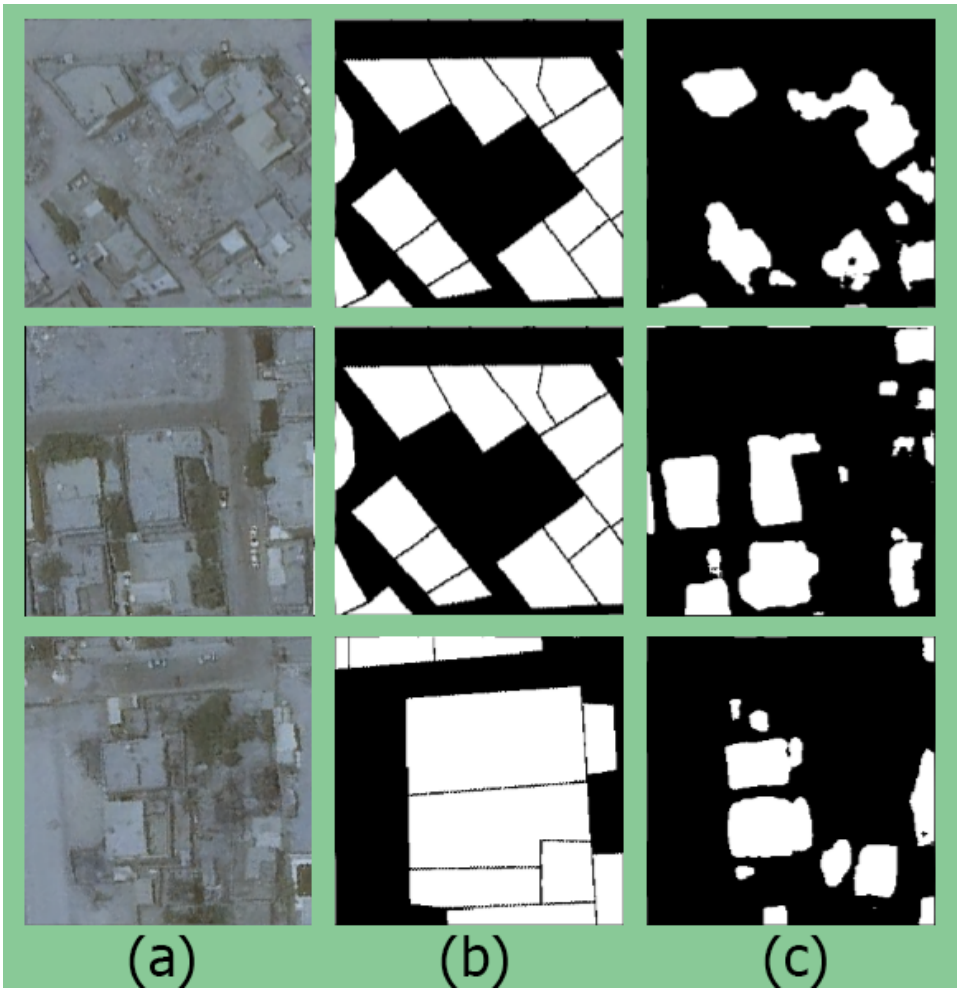

Figure 8.2: Three image examples: (a)original image, (b)provided label from UN-Habatit, (c) segmentation result

9

Conclusions and Discussion

In this research, we study deep learning techniques for building segmentation on satellite image data. Starting with testing several classic segmentation algorithms, from the experiment results, we find out that the performance can still be improved. Then, we propose two data reweighing methods, named border weight and inter-building distance weight, to improve the performance. By increasing the weights of the pixels outside but close to the border of the buildings, the model is encouraged to learn those information and thus performs better. Inspired by the idea of reweighing the non-building pixels, we investigate whether modifying building pixels can achieve further improvement. We propose a new label representation – multi-level boundary label that does help to improve the segmentation results. Based on the distance to the building boundary, we can divide building pixels into multiple classes, as their pixel values can be affected by some factors such as trees and shadows. From the experiment result, we can see the performance is improved since the model captures more information about the buildings. Next, we propose a new model to utilize the two pixel weights, and the multi-level boundary label explained above. Our proposed model achieves state-of-the-art building segmentation performance compared with several classic segmentation methods. Our model also uses fewer number of parameter because we only use the first 13 layers of the VGG16 as the encoder and we do not use any convolutional layers in the decoder part.

To get the information about urban development (building counts and building size), we apply two post-processing steps on the segmentation results and show that we can obtain information about urban development from segmentation results. Specifically, we collect all coordinates of pixels representing buildings and aggregate pixels belonging to the same building according to their connectivities. The size of each building is the number of pixel of each group, and the total number of buildings is the number of aggregated groups. Also, to see whether our proposed model works well on the satellite images provided by UN-Habitat, we train our model with public datasets and use transfer learning to fine-tune the model. From the result we can see that our proposed model performs well on a new dataset. Based on these conclusion, we demonstrate that we can obtain information about urban development from segmentation results.

Regarding the main research question:

- **Can we develop an automated system that provides valuable information about urban development for the UN-Habitat from satellite image data (e.g. building detection)?**

Our answer is a definitive **YES!**. We not only show that it is possible to obtain information about urban development from satellite image but also propose a new model with great performance in

our work.

9.1. DISCUSSION

In this thesis, we address some problems from the building segmentation results testing on classic methods, such as the pixels close to the building borders could be affected by trees and shadows. Instead of focusing on building a new model or modifying some well-known architecture, we examine our task and the datasets we need to use. Then we find out that we can improve the performance by modifying the data rather than modifying the model. If you do not have a great understanding of the model, it is so hard to modify it correctly and get the result as you expect. Focusing on the data can be a good idea because in many cases, we can find out some characteristics of the data. As I have mentioned above, an excellent outcome is obtained by both a good method and high-quality data. For our work, we cannot promise that our reweighing methods and multi-class label representation work well on tasks other than building segmentation. But we are confident that they do improve the building segmentation results.

Regarding our proposed model, one novel point about the architecture is that we transform the original label into another representation and then use two branches to let the model learn from different information. The data label may include more information than we expect. It is good to pay attention to data.

From the UN-Habitat's perspective, an important question they hope to know the answer is whether satellite images can be used in combination with some automated processes to obtain information about urban development. According to our work, we can say the answer is yes, and we can also tell how to achieve it. Obtaining the building count and building size can be defined as a building segmentation task. The performance and results are shown in the experiment results in this thesis, they are deemed to be good enough. However, as different datasets are so different, to produce great results, we need to have high-quality labels to let the model learn the data. If UN-Habitat is interested in the approach explored in this thesis, we suggest UN-Habitat to generate some high-quality labels and then try our model or related methods.

9.2. FUTURE WORK

Hyper-parameters Tuning

In our research, we propose two methods to reweigh the data, named border weight and inter-building distance weight, and a new label representation, named multi-level boundary label. Their values are dependent on their parameters' values and the optimal parameters' values are dependent on the algorithm and the dataset that would be used. In our thesis, we use grid search to find the reasonable ranges. It will be compelling if we can develop a better matching strategy to obtain the values of these parameters, especially for new datasets.

Generalization Ability of our Proposed Model

In this thesis, the architecture of our model is proposed to follow and match our task – binary building segmentation and the datasets we use – satellite images with a top-down view. From the experiment results, we know that our model performs well on building segmentation task. However, does our model still has a good performance on other segmentation tasks (e.g. road and vehicle segmentation for automated driving) ? Also, for other cases, will our multi-level boundary label, our border weight and our inter-building distance weight still help our model to achieve a improvement? Further experiments are needed to the get the answers.

APPENDIX - FULL EXPERIMENT RESULTS OF REWEIGHING METHODS

Table 9.1: Full experiment results of border weight of method FCN8S

FCN8S	Crowd AI dataset			Inria Aerial dataset		
	PA	MPA	MIoU	PA	MPA	MIoU
Baseline	0.9414	0.9128	0.8479	0.9018	0.8735	0.7818
w0=2,σ_0=0.2	0.9455	0.9140	0.8567	0.9080	0.8776	0.7902
w0=2,σ_0=0.5	0.9417	0.9140	0.8491	0.9074	0.8793	0.7898
w0=2,σ_0=0.8	0.9440	0.9230	0.8566	0.9068	0.8835	0.7902
w0=5,σ_0=0.2	0.9415	0.9116	0.8480	0.9034	0.8650	0.7886
w0=5,σ_0=0.5	0.9445	0.9192	0.8562	0.9076	0.8806	0.7905
w0=5,σ_0=0.8	0.9421	0.9128	0.8483	0.9031	0.8731	0.7811
w0=10,σ_0=0.2	0.9449	0.9167	0.8561	0.9025	0.8610	0.7857
w0=10,σ_0=0.5	0.9419	0.9133	0.8481	0.9122	0.8841	0.7988
w0=10,σ_0=0.8	0.9439	0.9129	0.8531	0.9071	0.8805	0.7897
w0=20,σ_0=0.2	0.9515	0.9228	0.8609	0.9050	0.8679	0.7821
w0=20,σ_0=0.5	0.9416	0.9217	0.8508	0.9041	0.8649	0.7796
w0=20,σ_0=0.8	0.9425	0.9115	0.8499	0.9067	0.8808	0.7891

Table 9.2: Full experiment results of border weight of method SegNet

SegNet	Crowd AI dataset			Inria Aerial dataset		
	PA	MPA	MIoU	PA	MPA	MIoU
Baseline	0.9356	0.9096	0.8481	0.8999	0.8676	0.7830
w0=2,σ_0=0.2	0.9456	0.9154	0.8571	0.9156	0.8948	0.8076
w0=2,σ_0=0.5	0.9408	0.9199	0.8487	0.9082	0.8894	0.7943
w0=2,σ_0=0.8	0.9413	0.9172	0.8499	0.9087	0.8760	0.7906
w0=5,σ_0=0.2	0.9439	0.9249	0.8560	0.9065	0.8796	0.7885
w0=5,σ_0=0.5	0.9445	0.9132	0.8543	0.9099	0.8809	0.7942
w0=5,σ_0=0.8	0.9467	0.9178	0.8600	0.9044	0.8830	0.7864
w0=10,σ_0=0.2	0.9418	0.9062	0.8470	0.9084	0.8903	0.7948
w0=10,σ_0=0.5	0.9458	0.9177	0.8582	0.9120	0.8815	0.7977
w0=10,σ_0=0.8	0.9434	0.9139	0.8522	0.9063	0.8722	0.7856
w0=20,σ_0=0.2	0.9457	0.9180	0.8579	0.9096	0.8722	0.7907
w0=20,σ_0=0.5	0.9471	0.9169	0.8605	0.9062	0.8766	0.7869
w0=20,σ_0=0.8	0.9430	0.9177	0.8524	0.9108	0.8866	0.7974

Table 9.3: Full experiment results of border weight of method Deep_Lab

Deeplab	Crowd AI dataset			Inria Aerial dataset		
	PA	MPA	MIoU	PA	MPA	MIoU
Baseline	0.9266	0.8866	0.8241	0.8883	0.8565	0.7616
w0=2,σ_0=0.2	0.9354	0.9060	0.8347	0.9009	0.8804	0.7803
w0=2,σ_0=0.5	0.9351	0.8929	0.8302	0.8995	0.8557	0.7693
w0=2,σ_0=0.8	0.9339	0.9022	0.8307	0.8986	0.8693	0.7730
w0=5,σ_0=0.2	0.9355	0.9014	0.8334	0.8984	0.8707	0.7731
w0=5,σ_0=0.5	0.9333	0.9028	0.8298	0.8581	0.8629	0.7158
w0=5,σ_0=0.8	0.9352	0.8977	0.8319	0.8986	0.8678	0.7725
w0=10,σ_0=0.2	0.9364	0.9031	0.8356	0.8901	0.8576	0.7663
w0=10,σ_0=0.5	0.9344	0.9001	0.8310	0.8977	0.8546	0.7662
w0=10,σ_0=0.8	0.9357	0.9022	0.8341	0.8994	0.8772	0.7770
w0=20,σ_0=0.2	0.9370	0.9032	0.8370	0.8994	0.8632	0.7720
w0=20,σ_0=0.5	0.9352	0.8977	0.8319	0.8923	0.8533	0.7579
w0=20,σ_0=0.8	0.9370	0.9019	0.8366	0.9014	0.8700	0.7774

Table 9.4: Full experiment results of inter-building distance weight of method FCN8S

FCN8S	Crowd AI dataset			Inria Aerial dataset		
	PA	MPA	MIoU	PA	MPA	MIoU
Baseline	0.9340	0.9059	0.8440	0.8966	0.8647	0.7770
w0=2,σ_0=2	0.9448	0.9154	0.8556	0.9047	0.8758	0.7844
w0=2,σ_0=5	0.9422	0.9103	0.8489	0.9064	0.8796	0.7884
w0=2,σ_0=10	0.9416	0.9095	0.8476	0.9006	0.8764	0.7784
w0=5,σ_0=2	0.9421	0.9179	0.8507	0.9067	0.8778	0.7882
w0=5,σ_0=5	0.9447	0.9108	0.8540	0.9061	0.8856	0.7898
w0=5,σ_0=10	0.9321	0.8986	0.8262	0.9097	0.8842	0.7950
w0=10,σ_0=2	0.9448	0.9127	0.8548	0.9091	0.8777	0.7919
w0=10,σ_0=5	0.9411	0.9062	0.8457	0.9052	0.8703	0.7833
w0=10,σ_0=10	0.9409	0.9046	0.8448	0.9025	0.8766	0.7813
w0=20,σ_0=2	0.9443	0.9099	0.8530	0.9076	0.8750	0.7886
w0=20,σ_0=5	0.9417	0.9136	0.8489	0.9109	0.8799	0.7955
w0=20,σ_0=10	0.9408	0.9059	0.8451	0.9083	0.8781	0.7907

Table 9.5: Full experiment results of inter-building distance weight of method SegNet

SegNet	Crowd AI dataset			Inria Aerial dataset		
	PA	MPA	MIoU	PA	MPA	MIoU
Baseline	0.9323	0.8981	0.8385	0.8992	0.8767	0.7850
w0=2,σ_0=2	0.9434	0.9143	0.8525	0.9088	0.8844	0.7936
w0=2,σ_0=5	0.9415	0.9083	0.8470	0.9090	0.8792	0.7922
w0=2,σ_0=10	0.9479	0.9213	0.8632	0.9052	0.8817	0.7871
w0=5,σ_0=2	0.9444	0.9113	0.8536	0.9082	0.8778	0.7905
w0=5,σ_0=5	0.9477	0.9218	0.8629	0.9112	0.8944	0.8005
w0=5,σ_0=10	0.9443	0.9135	0.8539	0.9071	0.8775	0.7887
w0=10,σ_0=2	0.9424	0.9103	0.8495	0.9096	0.8898	0.7966
w0=10,σ_0=5	0.9402	0.9119	0.8455	0.9153	0.8787	0.8021
w0=10,σ_0=10	0.9436	0.9158	0.8531	0.9096	0.8930	0.7976
w0=20,σ_0=2	0.9454	0.9142	0.8564	0.9108	0.8850	0.7969
w0=20,σ_0=5	0.9451	0.9109	0.8549	0.9108	0.8864	0.7974
w0=20,σ_0=10	0.9442	0.9148	0.8541	0.9089	0.8777	0.7915

Table 9.6: Full experiment results of inter-building distance weight of method Deep_Lab

Deeplab	Crowd AI dataset			Inria Aerial dataset		
	PA	MPA	MIoU	PA	MPA	MIoU
Baseline	0.9254	0.8871	0.8221	0.8841	0.8598	0.7566
w0=2,σ_0=2	0.9348	0.8976	0.8311	0.9017	0.8649	0.7760
w0=2,σ_0=5	0.9348	0.8989	0.8314	0.8986	0.8688	0.7728
w0=2,σ_0=10	0.9353	0.9021	0.8332	0.8880	0.8644	0.7560
w0=5,σ_0=2	0.9353	0.8972	0.8320	0.8978	0.8722	0.7727
w0=5,σ_0=5	0.9312	0.8921	0.8263	0.8957	0.8662	0.7676
w0=5,σ_0=10	0.9348	0.8986	0.8314	0.8979	0.8665	0.7710
w0=10,σ_0=2	0.9344	0.9034	0.8320	0.8985	0.8611	0.7699
w0=10,σ_0=5	0.9290	0.9047	0.8225	0.8989	0.8663	0.7724
w0=10,σ_0=10	0.9368	0.9012	0.8359	0.8983	0.87837	0.7742
w0=20,σ_0=2	0.9346	0.9037	0.8325	0.8956	0.8681	0.7682
w0=20,σ_0=5	0.9331	0.8974	0.8346	0.9015	0.8701	0.7769
w0=20,σ_0=10	0.9357	0.8987	0.8331	0.8982	0.8613	0.7695

BIBLIOGRAPHY

[1] *World urbanization prospects 2018,* https://population.un.org/wup/Maps/ (2018).

[2] H. Heidenreich, *What are the types of machine learning?* https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f (2018).

[3] E. Yates, *What is the difference between ai, machine learning, and deep learning?* https://towardsdatascience.com/clarity-around-ai-language-2dc16fdb6e82 (2019).

[4] M. labs, *Everything you need to know about neural networks,* https://hackernoon.com/everything$-$you-need-to-know-about-neural-networks-8988c3ee4491 (2017), accessed: 2019-1-3.

[5] *Feedforward neural networks,* https://brilliant.org/wiki/feedforward-neural-networks/ (2018).

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

[7] V. Badrinarayanan, A. Kendall, and R. Cipolla, *Segnet: A deep convolutional encoder-decoder architecture for image segmentation,* IEEE transactions on pattern analysis and machine intelligence **39**, 2481 (2017).

[8] F. Yu and V. Koltun, *Multi-scale context aggregation by dilated convolutions,* arXiv preprint arXiv:1511.07122 (2015).

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,* IEEE transactions on pattern analysis and machine intelligence **40**, 834 (2017).

[10] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, *High-resolution aerial image labeling with convolutional neural networks,* IEEE Transactions on Geoscience and Remote Sensing **55**, 7092 (2017).

[11] R. Hamaguchi and S. Hikosaka, *Building detection from satellite imagery using ensemble of size-specific detectors,* in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2018).

[12] *wleite's implementation,* https://github.com/SpaceNetChallenge/BuildingDetectors/tree/master/wleite (2018).

[13] Z. Hayder, X. He, and M. Salzmann, *Boundary-aware instance segmentation,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017) pp. 5696–5704.

[14] *Intersection over union (iou) for object detection,* https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (2016).

[15] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, *A review on deep learning techniques applied to semantic segmentation,* arXiv preprint arXiv:1704.06857 (2017).

[16] *68% of the world population projected to live in urban areas by 2050, says un,* https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html (2018).

[17] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar, *Deepglobe 2018: A challenge to parse the earth through satellite images,* in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2018).

[18] *What is machine learning? a definition,* https://www.expertsystem.com/machine-learning-definition/ (2018).

[19] A. L. Samuel, *Some studies in machine learning using the game of checkers. ii—recent progress,* in *Computer Games I* (Springer, 1988) pp. 366–400.

[20] T. Mitchell, *Machine Learning* (McGraw-Hill, New York, 1997).

[21] Wikipedia, *Machine learning — Wikipedia, the free encyclopedia,* http://en.wikipedia.org/w/index.php?title=Machine%20learning&oldid=903232737 (2019), [Online; accessed 01-January-2019].

[22] R. A. Fisher, *The use of multiple measurements in taxonomic problems,* Annals of eugenics **7**, 179 (1936).

[23] B. Schölkopf, A. J. Smola, F. Bach, *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond* (MIT press, 2002).

[24] L. Breiman, *Random forests,* Machine learning **45**, 5 (2001).

[25] G. E. Hinton, T. J. Sejnowski, and T. A. Poggio, *Unsupervised learning: foundations of neural computation* (MIT press, 1999).

[26] J. MacQueen *et al.*, *Some methods for classification and analysis of multivariate observations,* in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1 (Oakland, CA, USA, 1967) pp. 281–297.

[27] K. Pearson, *Liii. on lines and planes of closest fit to systems of points in space,* The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2**, 559 (1901).

[28] J. H. Ham, D. D. Lee, and L. K. Saul, *Learning high dimensional correspondences from low dimensional manifolds,* (2003).

[29] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning,* nature **521**, 436 (2015).

[30] A. L. Maas, A. Y. Hannun, and A. Y. Ng, *Rectifier nonlinearities improve neural network acoustic models,* in *Proc. icml*, Vol. 30 (2013) p. 3.

[31] Y. B. Ian Goodfellow and A. Courville, *Deep learning,* (2016), book in preparation for MIT Press.

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

[33] S. Zhou, J.-N. Wu, Y. Wu, and X. Zhou, *Exploiting local structures with the kronecker layer in convolutional networks,* arXiv preprint arXiv:1512.09194 (2015).

[34] S. Lazebnik, C. Schmid, and J. Ponce, *Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,* in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2 (IEEE, 2006) pp. 2169–2178.

[35] K. He, X. Zhang, S. Ren, and J. Sun, *Spatial pyramid pooling in deep convolutional networks for visual recognition,* IEEE transactions on pattern analysis and machine intelligence **37**, 1904 (2015).

[36] S. Jabari, Y. Zhang, and A. Suliman, *Stereo-based building detection in very high resolution satellite imagery using ihs color system,* in *2014 IEEE Geoscience and Remote Sensing Symposium* (IEEE, 2014) pp. 2301–2304.

[37] X. Sun, X. Lin, S. Shen, and Z. Hu, *High-resolution remote sensing data classification over urban areas using random forest ensemble and fully connected conditional random field,* ISPRS International Journal of Geo-Information **6**, 245 (2017).

[38] Y. Wei, W. Yao, J. Wu, M. Schmitt, and U. Stilla, *Adaboost-based feature relevance assessment in fusing lidar and image data for classification of trees and vehicles in urban scenes,* ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences **1**, 323 (2012).

[39] *Spacenet off-nadir building detection challenge,* https://spacenetchallenge.github.io/Challenges/challengesSummary.html (2018).

[40] *Crowdai mapping challenge,* https://www.crowdai.org/challenges/mapping-challenge (2018).

[41] B. Bischke, P. Helber, J. Folz, D. Borth, and A. Dengel, *Multi-task learning for segmentation of building footprints with deep neural networks,* arXiv preprint arXiv:1709.05932 (2017).

[42] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, *Feature pyramid networks for object detection,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017) pp. 2117–2125.

[43] S. Seferbekov, V. Iglovikov, A. Buslaev, and A. Shvets, *Feature pyramid network for multi-class land segmentation,* in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2018).

[44] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn,* in *Computer Vision (ICCV), 2017 IEEE International Conference on* (IEEE, 2017) pp. 2980–2988.

[45] K. Zhao, J. Kang, J. Jung, and G. Sohn, *Building extraction from satellite images using mask r-cnn with building boundary regularization,* in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2018).

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

[47] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster,

J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems,* (2015), software available from tensorflow.org.

[48] D. Ho, E. Liang, I. Stoica, P. Abbeel, and X. Chen, *Population based augmentation: Efficient learning of augmentation policy schedules,* arXiv preprint arXiv:1905.05393 (2019).

[49] M. Thoma, *A survey of semantic segmentation,* arXiv preprint arXiv:1602.06541 (2016).

[50] *deeplab tensorflow implementation,* https://github.com/tensorflow/models/tree/master/research/deeplab (2019).

[51] *fcn tensorflow implementation,* https://github.com/MarvinTeichmann/tensorflow-fcn (2016).

[52] *segnet tensorflow implementation,* https://github.com/toimcio/SegNet-tensorflow (2017).

[53] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization,* arXiv preprint arXiv:1412.6980 (2014).

[54] D. Eigen and R. Fergus, *Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,* in *Proceedings of the IEEE international conference on computer vision* (2015) pp. 2650–2658.

[55] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition,* arXiv preprint arXiv:1409.1556 (2014).

[56] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *ImageNet: A Large-Scale Hierarchical Image Database,* in *CVPR09* (2009).

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