ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Segmentation of Geometrical Shapes in Aerial Images

Master's Thesis

Zuoyue Li

April 2018

Supervisors: Prof. Dr. T. Hofmann, Dr. A. Lucchi, Dr. J. D. Wegner
Department of Computer Science, ETH Zürich

Abstract

Deep learning methods have recently demonstrated remarkable achievements on image segmentation. However, their architectures are still limited to pixel-wise labeling, which makes it hard to exploit high-level topology.

The goal of this project is to develop novel deep learning architectures for exploiting geometrical shapes of arbitrary structure. We want to depart from the standard paradigm that labels pixels but instead directly exploit and learn the geometry of the objects. In practical applications, we use dataset of aerial images, and aiming at extracting buildings' polygon shapes.

Two recent models, PolygonRNN and Mask R-CNN, draw our attention. Given the object bounding box, the former one can extract the geometrical shape of a single object within the box. The latter one integrates multi-target detection, classification and segmentation. Since our goal can be divided into two parts, objects detection and geometrical shapes segmentation, the basic idea of the proposed solution would utilize these two models in two steps.

Specifically, our proposed model, R-PolygonRNN (Region-based PolygonRNN) is the integration of FPN (Feature Pyramid Network) part in Mask R-CNN, and PolygonRNN. The model has three different versions, the two-step version, hybrid version and hybrid version with RoIAlign. The model can give multiple bounding boxes for each buildings within an satellite image, and for each bounding box, it outputs the polygon of the building.

Experiments show that (XXX).

Acknowledgment

Foremost, I would like to express my sincere gratitude to Dr. Aurelien Lucchi and Dr. Jan Dirk Wegner for supervising my Master's thesis project, supporting me continuously and contributing many useful ideas. Their guidance helped me in all the time of discussing project and writing thesis, and I have learned a lot in the field of object detection and geometrical shape segmentation.

I would also like to thank Prof. Thomas Hofmann for providing me with the opportunity of this interesting project, as well as his suggestions about beam search. I would say doing Master's thesis project at Data Analytics Lab is an unforgettable experience for me.

Besides my supervisors, I would like to thank Tianhao Wei, a junior to me at Zhejiang University, for giving me many suggestions for the implementation details about PolygonRNN.

My sincere thanks also goes to my friends, Jingxuan He, Xiaojuan Wang, Canxi Chen, Jie Huang, Junlin Yao, and Renfei Liu, for all their helps, supports and companionship.

Last but not the least, I would like to thank my parents Haiyan Dai and Fasheng Li, for their spiritual supports and understandings throughout my studying life in Switzerland.

Contents

Chapter 1

Introduction

This chapter mainly provides a brief introduction to the entire project. Section 1.1 presents the background and some fundamental concepts in order to give readers a basic understanding of this field. Section 1.2 defines the problems of this project to be solved. Section 1.3 gives a brief introduction to our contribution and our proposed new model. Section 1.4 illustrates the structure of this thesis for the convenience of readers.

1.1 Background

In this section, the background of this project is introduced. Subsection 1.1.1 focus on aerial image and its application. Subsection 1.1.2 presents the concept of image segmentation, and its commonly used methods. Subsection 1.1.3 introduces the idea of segmentation with geometry, which is the main point of our project.

1.1.1 Aerial Image

In our project, an aerial image generally refers to the "optical overhead imagery" [**?**] acquired by aircraft. This kind of image has an extremely wide range of applications in the field of geographical surveying and mapping, which can not only clearly depict the terrain, but can also show the structure and the layout of the city. Furthermore, it can also provide services such as land use status and remote sensing monitoring.

Especially in the metropolis, many buildings are constantly being updated with the expansion of the city, and many landscapes are changing with human's activities. Therefore, correspondingly, the city's electronic map needs to be updated accordingly with the change of city appearance as well. In this case, the acquisition of aerial images becomes necessary, as those images can provide significant detailed visual information from above of the city, which

is typically unaccessible from human perspective. Figure 1.1 shows two examples of aerial image.

(a) a tile of city Zurich (b) a tile of city Chicago

Figure 1.1: Example of two aerial images. This kind of images can provide people with important reference of the city layout from above.

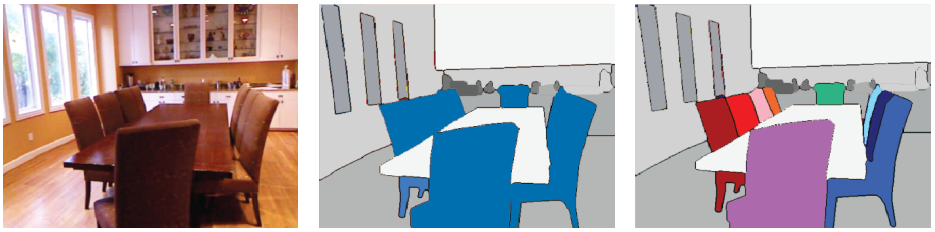
Nowadays, huge volumes of aerial images are captured with airborne or spaceborne platforms. The increasing volume makes manual interpretation prohibitive [**?**]. Hence, we should employ appropriate ideas and methods from the field of computer vision to utilize this kind of data. In fact, in order for the machine to better understand aerial images, we can perform image segmentation.

1.1.2 Image Segmentation

In the field of computer vision, image segmentation is the process of partitioning an image into multiple sets of pixels. The goal of segmentation is to simplify and change the representation of an image and make the image more meaningful and easier to analyze [**?**].

In our project, image segmentation mainly refers to the so-called semantic segmentation, which is a process of labelling each pixel of the image. Hence the segmentation is exactly a pixel-wise classification problem. Note that the labels between two adjacent pixels are not independent, but related to each other. Pixels with the same label are generally similar in the metric of certain visual characteristics, such as color, brightness or texture. Another kind of segmentation is the so-called instance segmentation. It not only does semantic segmentation, but also distinguishes between the object instances, even they have the same label. That is to say, object instances with same labels are required to have different IDs. Figure 1.2 shows the difference

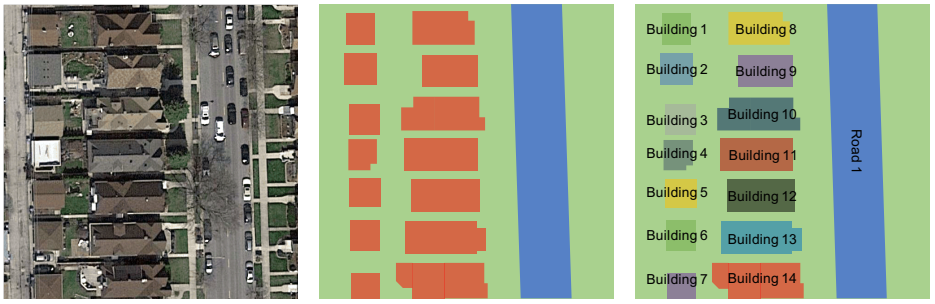
between semantic and instance segmentation.



(a) original image (b) semantic segmentation (c) instance segmentation

Figure 1.2: Examples of semantic segmentation and instance segmentation [**?**]. The original image (a) shows a room with several chairs. (b) is the semantic segmentation result of (a), where blue color denotes the chairs, and other colors such as white and gray denote irrelevant background. (c) is instance segmentation result of (a), where all chairs have the same label, but different colors which are used to indicate different instances.

Specifically, for semantic segmentation in aerial images, usually what we do is to label each pixel as buildings, roads or background, which can be seen in figure 1.3(b). Its instance segmentation result is shown in figure 1.3(c).



(a) original aerial image (b) semantic segmentation (c) instance segmentation

Figure 1.3: Examples of semantic segmentation and instance segmentation in aerial image. The original aerial image (a) is a tile of city Chicago. (b) is the semantic segmentation result of (a), where red color denotes the buildings, blue color denotes the roads, green color denotes the background. (c) is instance segmentation result of (a), where all buildings have the same label, but different "ID" numbers which can be used to indicate they are different instances.

As a matter of fact, for image where the objects (buildings) are separated like figure 1.3(a), we can easily obtain the instance segmentation result based on the pixel connectivity information from the semantic segmentation result. However, if objects overlap in the image like the chairs in figure 1.2(a), it is difficult to do such a thing. Thus, in this case, instance segmentation can show its advantages.

Traditional semantic segmentation methods include clustering method [**?**],

histogram-based method [**?**], compression-based methods [**?**], region-growing method [**?**] and so on. Recently, deep learning methods have demonstrated remarkable achievements in image segmentation tasks. For those methods, please refer to section 2.1 for more details.

1.1.3 Geometrical Shape

As introduced in subsection 1.1.2, the output format of either semantic or instance segmentation, is per-pixel mask. Although it is currently the mainstream choice of image segmentation, it is undeniable that the per-pixel mask has limitations on the representation of geometrical shapes. The geometrical shape here generally refers to a polygon represented by a series of ordered vectors or coordinates.

Indeed, we can undergo more processing steps to obtain geometrical shapes based on the instance segmentation result. However, we want to get rid of the pixel-wise labelling rigidity and directly describe geometrical shape of each object in an image. Specifically, in our project, deviating from the standard paradigm of labeling pixels and aiming to directly learn the geometry of the buildings in aerial images have following advantages:

- Polygon representation has much less redundancy and relatively less storage than pixel-wise labelling;
- Polygon representation is a kind of vector illustration, thus can be used at arbitrary levels or scales;
- Buildings with polygon representation can be modeled more naturally;
- Polygon representation can be directly marked in the electronic map, but per-pixel mask can not.

The polygon representation is therefore a more compact and useful representation in segmentation of buildings on aerial images.

As mentioned in subsection 1.1.2, deep learning methods have shown significant progress in tasks of image segmentation. However, the architectures used in these methods are still limited to conventional grid structure diagrams and their output is still pixel-wise. The rigidity of these networks makes it difficult to exploit high-level priors about the geometrical shapes of objects in the image. Modeling high-level, long-range object topology has proven to be difficult with standard architectures that model more local, per-pixel evidence.

Therefore, the purpose of this project is to develop novel deep learning methods for the geometrical shapes of arbitrary structures, which means that we want to introduce object geometrical shapes into deep learning techniques.

1.2 Problem Definition

Given an aerial image, our goal is to extract the polygon shape for each building in the image. The input is an aerial image, denoting

$$I = \{I_{ijk}\}_{i\in\{1,2,\ldots,h\},j\in\{1,2,\ldots,w\},k\in\{1,2,3\}}, \tag{1.1}$$

where I_{ijk} denotes the pixel value of i-th row, j-th column and k-th channel in the image, w and h denote the width and height of the input image. The output is polygons, denoting

$$P = \{P^{(n)}\}_{n\in\{1,2,\ldots,N\}}, \tag{1.2}$$

$$P^{(n)} = \{(i_t^{(n)}, j_t^{(n)})\}_{t\in\{1,2,\ldots,L_n\}}, \tag{1.3}$$

where N denotes the number of complete buildings (or polygons) in the image, $P^{(n)}$ and L_n denote the n-th polygon and its number of vertices, $(i_t^{(n)}, j_t^{(n)})$ denote the image coordinates of t-th vertex of the polygon $P^{(n)}$ in the original input image.

In short, our goal is to achieve instance segmentation of geometrical shapes for buildings in aerial images. Figure 1.4 shows an example of desired input and output.

(a) aerial image as input (b) expected polygons as output

Figure 1.4: Example of instance segmentation of geometrical shapes in an aerial image. (a) is the input aerial image containing 3 complete buildings. (b) is the visualized result for output polygons.

1.3 Focus of This Work

The problem defined in section 1.2 is challenging because it not only requires the correct detection (or localization) for all buildings in an aerial image, but

also needs to precisely segmenting each building in the representation of polygon rather than per-pixel mask. Therefore, it is exactly a combination of two tasks of computer vision. The first one is object detection, where the goal is to detect (or localize) each individual object in the image using a bounding box. The second one is the geometrical instance segmentation, where the goal is to extract polygon for a single instance.

In order to solve this problem, we propose a new model, R-PolygonRNN (Region-based PolygonRNN), which is a combination of the FPN (Feature Pyramid Network) [**?**] and PolygonRNN [**?**]. In our new model, FPN is used to localize buildings, i.e. to detect the RoIs (Regions of Interest) in the image, and PolygonRNN is used to find geometrical shape for a single object. The new proposed model can successfully find geometrical shapes for multiple buildings in an aerial image, thus it tackles the shortcoming of PolygonRNN, which can only segment single object. Furthermore, the thought of beam search is introduced to PolygonRNN when predicting polygons, which addresses the false vertex problem and significantly improves the prediction result.

1.4 Thesis Organization

This thesis is organized as follows. Chapter 2 reviews related work, focusing on the deep learning methods for image segmentation, previous work related to segmentation in aerial images, as well as the two recent new models, Mask R-CNN [**?**] and PolygonRNN. In chapter 3, the architectures of PolygonRNN and FPN are presented, and the structure of our proposed model is also explained in detail. Chapter 4 describes the ground truth dataset we use and gives the experiment configurations and results. Finally, chapter 5 makes conclusions, points out the problems which exist in our project and gives future direction.

In addition, the terminologies we used and its corresponding detailed explanations are shown in table 1.1.

Table 1.1: Common terminologies used in our project with explanations.

Terminology	**Explanation**
Object Detection	Detection via bounding boxes instead of masks
Semantic Segmentation	Pixel-wise classification without differentiating instances
Segmentation of Geometrical Shape (Geometrical Segmentation)	Polygon extraction for single object
Instance Segmentation	Both semantic and a form of detection
Instance Segmentation of Geometrical Shapes (Geometrical Instance Segmentation)	Geometrical segmentation and a form of detection

Chapter 2

Related Work

In this chapter, some related work are illustrated in detail. Section 2.2 presents two previous theses and points out their deficiencies with regard to our problem. Section 2.3 introduces two recent models, PolygonRNN and Mask R-CNN. Section 2.4 gives the summary of these models and further discusses the feasibility of applying these models to our problem.

2.1 Deep Learning Methods for Image Segmentation

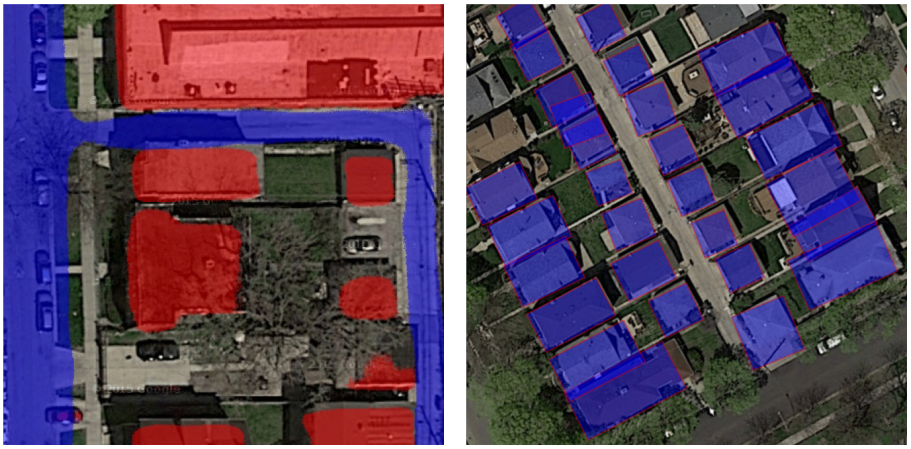
2.2 Previous Theses

There are two previous theses related to our problem, which are shown as follows.

Image Segmentation in Pixel Level In this thesis, our problem is treated as a pixel-wise image segmentation problem. The student employs a kind of deep neural network, U-Net, and make modification to classify each pixel in the aerial image as different labels, i.e. buildings, roads or background. The output of the model is a labelled mask. An example is shown in figure 2.1(a). We can see that the model can well distinguish between buildings and roads. However, there are also some limitations in this method: (1) we do not know which pixel belongs to which building, meaning that there is no instance segmentation here; (2) the buildings cannot be detected unless we do further pixel connectivity detection; (3) there are no geometrical shapes of the objects. Therefore, this model cannot be fully used to solve our problems.

Rotated Bounding Box Another thesis utilizes the RPN (Region Proposal Network) part in Faster R-CNN and adds a branch for orientation degree. It predicts the rotated bounding boxes to 'capture' each building instance in an aerial image. Thus, our problem here becomes a parameters regression

problem, which aims at finding out the center coordinates, width, height, and the rotation degrees of the bounding box. The model used here is the modified RPN. The role of RPN part in original Faster R-CNN is to find RoIs (Regions of Interest), which are generally represented as bounding boxes or rectangles. This kind of representation can be very suitable for horizontal or vertical buildings. But we know that buildings are not always regular, many of them are inclined in the image. Therefore, in this thesis project, the RPN is modified and another branch for rotating the bounding boxes is added. This can make the boundary of the bounding box closer to the building. Figure 2.1(b) shows an example output. We can see from the figure that each building instance can correspond to a rotated bounding box, and can be also limited tightly to the box. This methods can detect buildings very well, but the geometrical shapes found are limited to rectangles and cannot be more precise.



(a) pixel-wise labelled mask (b) rotated bounding boxes

Figure 2.1: Example outputs of two previous theses. In (a), the pixels with red color refer to buildings, while the pixels with blue color refer to roads. In (b), each bounding box relates to one instance.

In short, the analysis and discussion of the two theses above show that although they have their own advantages and the effects are still good, they cannot fully reflect the geometrical shapes.

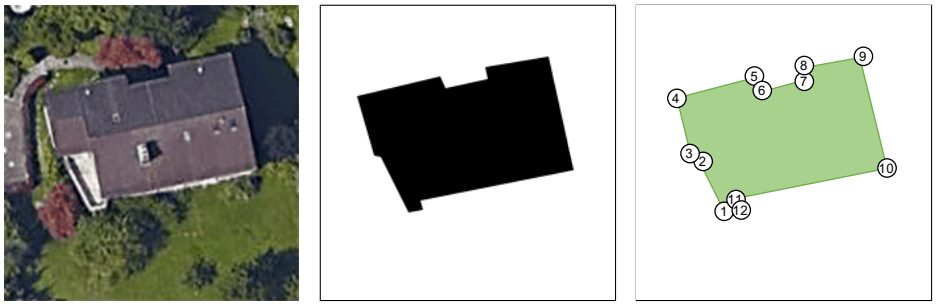
2.3 Recent Models

In this section, two recent models, PolygonRNN and Mask R-CNN are illustrated (see subsection 2.3.1 and 2.3.2 respectively). PolygonRNN focuses on the segmentation of geometrical shape and the FPN (Feature Pyramid Network) in Mask R-CNN can be used for the detection of buildings.

2.3.1 PolygonRNN

PolygonRNN aims at finding geometrical shape for an object instance, given the bounding box. The model is originally proposed for speeding up labeling the ground truth since it can achieve semi-automatic annotation of object instances, but the model can be used for single instance segmentation as well.

Most current methods regard the instance segmentation problem as a pixel-wise classification problem, generally labeling each pixel as object or background (see figure 2.2(b) for example). Different from this traditional way, the paper treats the segmentation task as a polygon prediction problem. In particular, PolygonRNN takes the image of instance as input and sequentially produces vertices of the polygon outlining the object (see figure 2.2(c) for example).



(a) an example building (b) pixel-wise mask (c) polygon prediction

Figure 2.2: Comparison of pixel-wise mask and polygon. (a) is the original image containing a building. (b) is the target mask of traditional pixel-wise instance segmentation, (c) is the desired prediction of polygon, of which the vertices are numbered.

We know that predicting a polygon is equivalent to predicting each of its vertices. Thus, the paper regards polygon as a series of vertices, and uses RNN as the model to make coherent prediction. RNN is very powerful when data is related to time series as it can carry complex information about the history. In our case, the prediction of each vertex is dependent on the position of its two previous vertices. The paper also mentioned that another advantage over traditional methods is that RNN can capture the shape of the object even in ambiguous cases like shadows and saturation.

In short, we hope that PolygonRNN can be used to solve our problems, extracting geometric shapes of buildings in aerial images, since the buildings in such images can be also regarded as polygons. For more details of the architecture, please refer to section 3.1.

2.3.2 Mask R-CNN

Mask R-CNN is a general framework for object instance segmentation. It is the latest model in the R-CNN family, which includes R-CNN, Fast R-CNN, Faster R-CNN as well. Mask R-CNN can efficiently detect objects in an image and generate a high-quality segmentation mask for each instance simultaneously. Figure 2.3 shows example results of Mask R-CNN. We can see from the figure that this model achieves excellent results. Figure 2.4 shows the simplified structure of Mask R-CNN. Each part of the structure are illustrated as follows.

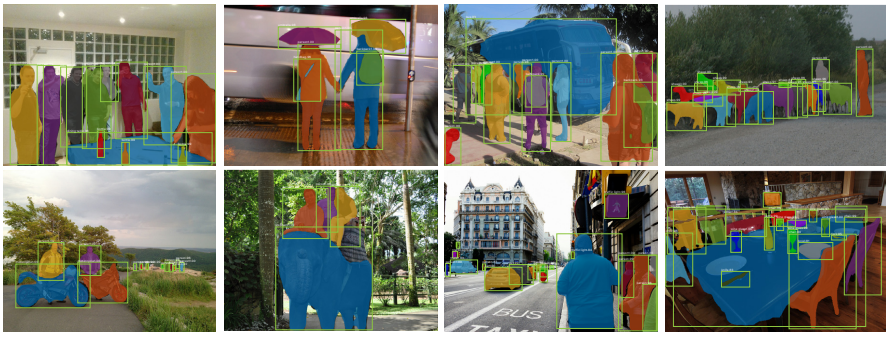


Figure 2.3: Example results of Mask R-CNN.

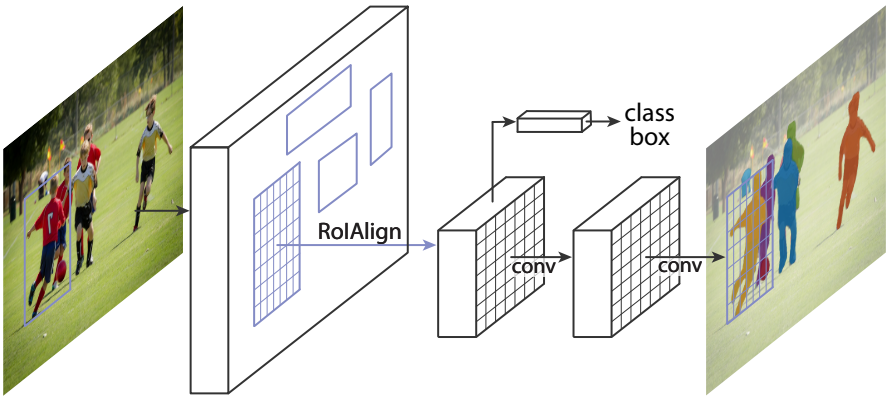


Figure 2.4: Simplified model structure of Mask R-CNN. Feature Pyramid Network locates at the first layer in the figure.

Feature Pyramid Network The original image is first passed through the FPN (Feature Pyramid Network), which locates at the first layer in figure 2.4. Just like the RPN mentioned in section 2.2, FPN is also a kind of network for finding RoIs. Actually, FPN is an upgraded version of RPN used in both Faster R-CNN and the previous thesis. Compared with RPN, FPN solves the

problem of multi-scale detection and introduces feature pyramid, and thus improves the accuracy of detection, especially for small objects in the image.

Classification Branch The classification branch exists from the earliest R-CNN to the present Mask R-CNN. It classifies each object in RoI found by FPN as different classes and gives probability, using fully connected layers. However, this branch is not relevant to our problem, since currently we only interested in buildings rather than kinds of different objects in an aerial image.

Mask Branch Mask R-CNN extends Faster R-CNN by adding the mask branch for predicting an object mask on each RoI in parallel with the existing classification branch. This branch uses small FCN (Fully Convolutional Network) for the pixel-wise semantic segmentation. The mask branch should be relevant to our problem, but the mask it predicts is still in the pixel level instead of polygon.

In short, Mask R-CNN can surpass prior instance segmentation results, but it cannot give any geometrical information of the objects in an image. Therefore, FPN is the only part of Mask R-CNN, which can be utilized to solve our problems. We hope that FPN can make correct detection of all buildings and localizing each using a bounding box. For more details of the architecture of FPN, please refer to section 3.2.

2.4 Motivation

Table 2.1 makes a summary for all models mentioned in sections 2.2 and 2.3. From the table we can conclude that none of these models meets our requirements. Thus, combination of two or even more models becomes necessary.

Table 2.1: Summary of Related Models.

Model	Localization	Geometrical Shape	Classification
Modified U-Net	Limited[1]	No	**Yes**
Modified RPN	**Yes**	Limited[2]	No
PolygonRNN	No	**Yes**	No
Mask R-CNN	**Yes**	No	**Yes**
FPN	**Yes**	No	No
What We Want	**Yes**	**Yes**	N/A [3]

[1] As mentioned in section 2.2, localization requires further pixel connectivity detection.

[2] As mentioned in section 2.2, only rotated bounding boxes are found.

[3] As mentioned in section 1.2, segmentation of roads is currently not considered in our project, thus the cell here shows 'N/A'.

After observation, we propose a possible approach, simply replacing the mask branch in Mask R-CNN with PolygonRNN so that the modified model can predict polygon rather than pixel-wise mask for each RoI. This model takes advantages from both models, thus it can be applied to our problem. In practice, we just combine FPN and PolygonRNN and name it a new model R-PolygonRNN (Region-based PolygonRNN), of which the architecture is shown in detail in section 3.3.

Chapter 3

Model Architecture

In this chapter, the architectures of several models are presented. Section 3.1 gives a comprehensive explanation to the structure of PolygonRNN, while section 3.2 looks into the FPN part of Mask R-CNN. Considering our problem, we combine PolygonRNN and FPN part of Mask R-CNN together and come up with a new model, which is called R-PolygonRNN (Region-based PolygonRNN, see section 3.3). In theory, the proposed model can find out the bounding boxes of buildings within an aerial image and give geometrical shape for each building.

3.1 PolygonRNN

PolygonRNN is the core model for finding geometrical shapes in this project. Figure 3.1 shows the simplified structure of PolygonRNN. The CNN part (see subsection 3.1.1) can capture image features through multilayer convolutions and max pooling, which is then fed into the RNN part (see subsection 3.1.2), which can sequentially predict spatial location of the new vertex with the highest probability at each time step.

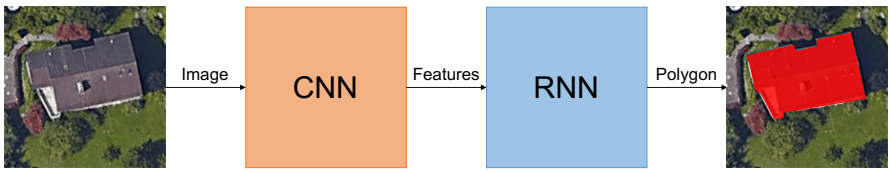


Figure 3.1: Simplified structure of PolygonRNN.

3.1.1 CNN Part

The CNN part of PolygonRNN uses VGG-16. Actually, VGG-16 is a form of VGGNet, which is very structured and focusing on deepening the neural net-

work without a large number of parameters. It generally believes that deeper networks have stronger expressive capabilities than shallow networks, and can accomplish more complex tasks. It also proven in practice that VGGNet has made great progress in performance compared to its previous network architecture (e.g. AlexNet).

The '16' in VGG-16 means that it is a VGGNet with 16 layers containing parameters (13 convolutional layers and 3 fully connected layers). It has around 138 million parameters in total. Figure 3.2 shows its detailed network structure. From the figure we can see that VGG-16 continuously does convolution with 3×3 small kernels and makes 2×2 max pooling. As the network deepens, the width and height of the image are reduced by half after each max pooling, and the number of channels is also doubly increasing after some convolution.

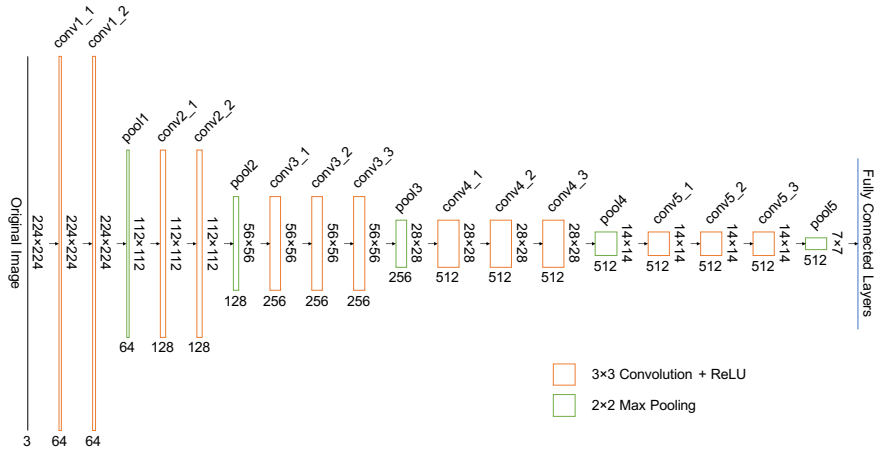


Figure 3.2: VGG-16 architecture. Only convolutional layers and max pooling layers are presented.

VGG-16 was mostly used for image classification before, so after the convolutional layers and max pooling layers there are fully connected layers and softmax layer for the class labels. However, these two kinds of layers are not required for VGG-16 used in PolygonRNN, because the CNN here is working as a feature extractor and mask predictor. Layer `pool5` is omitted as well because of the too low resolution.

Feature Extraction The modified VGG-16 provides RNN with useful features, which are taken from different convolutional and max pooling layers. Specifically, the features are extracted from layers `pool2`, `pool3`, `conv4_3` and `conv5_3`. Note that since the resolution of final features is fixed, when taking features from layer `pool2` and `conv5_3`, it requires another max pooling and upsampling respectively. All of these can be seen in figure 3.3.

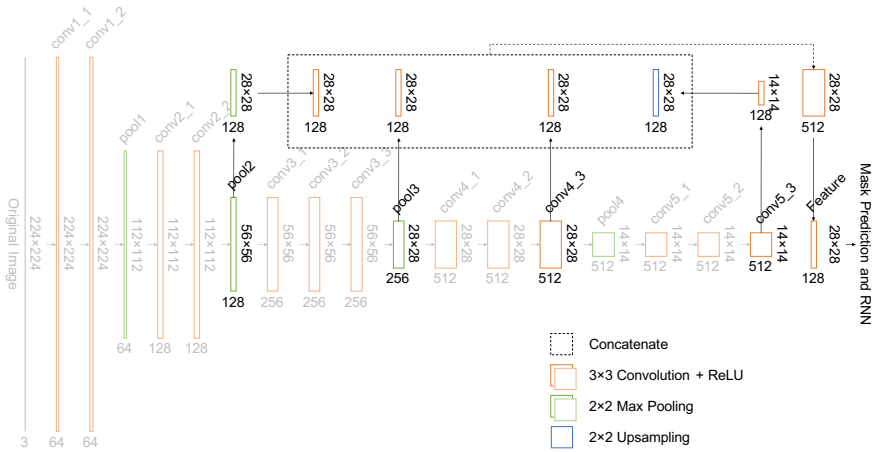


Figure 3.3: Modified VGG-16 architecture in PolygonRNN. In this figure, the highlight layers are used to extract features.

Mask Prediction Another function of the CNN part is to predict masks of boundary and vertices in a low resolution (one eighth of the original). Figure 3.4 shows the mask prediction phase. Different from the ReLU function used in the former convolutional layers, the activation function used here is the sigmoid function. Each entry of the boundary or the vertices mask indicates the probability that the pixel is located in the boundary or is a vertex respectively. The features and two masks are then sent into RNN together.

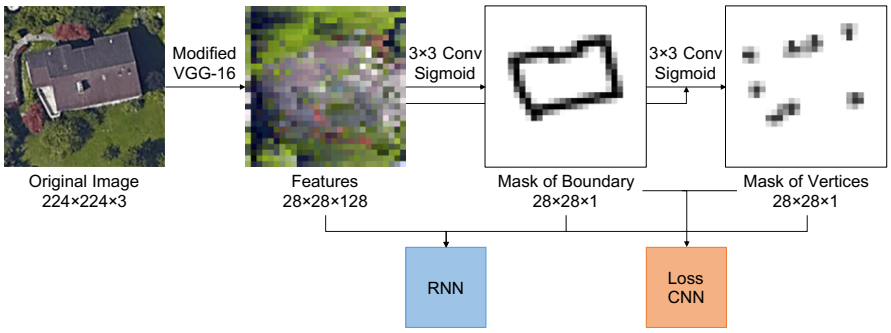


Figure 3.4: Mask prediction of VGG-16. Note that the mask of vertices is obtained by the convolution on the concatenation of the features and the mask of boundary.

3.1.2 RNN Part

The model used for predicting polygon vertices is RNN. We have already mentioned in subsection 2.3.1 that RNN is very powerful when data is related to time series, and in our case, we regards the polygon as a series of vertices.

We know that given two vertices on a polygon in an order (either clockwise or anticlockwise), the third vertex after the two points can be uniquely determined. What RNN here can do is to predict the probability distribution of the next vertex's position in a low resolution when given the history information about the two vertices before, as well as the image features and the position of the starting vertex. This process can be formulated as follows.

$$p(v_t \mid v_{t-1}, v_{t-2}) = F(v_{t-1}, v_{t-2}, v_0, f), \forall t \in \{2, 3, 4, \ldots, T\}, \tag{3.1}$$

where f denotes the extracted features by CNN (including the masks of boundary and vertices), $F(\cdot)$ denotes a function for computing the conditional probability, T denotes the number of vertices of the polygon, v_t denotes the vertex position at t-th prediction. Specifically, the vertex prediction problem can be formulated as a classification problem. The position of v_t can be then quantized to the resolution of output grid, and we can thus use one-hot encoding for v_t's representation.

End Signal Note that the positions for v_t are not only limited to the general output grid, the end signal for the closure detection of the polygon is also embedded in v_t, just like the 'end of sequence' token `<eos>` or `</s>` in the RNN language model. Thus, the number possible assignments of v_t equals to the resolution of the output grid plus one ($28 \times 28 + 1 = 785$ in our case). In order to correctly predict the end signal, the starting vertex v_0 is required for the conditional probability (equation 3.1) calculation, as it tells the model when to finish the prediction phase. If the current prediction is the same as, or very close to the starting vertex v_0, v_t will be forced to raise the end signal, indicating that the entire polygon is close, and the prediction phase is therefore complete. So generally, the end signal works when $t = T$.

Starting Vertex In equation 3.1, two special cases $p(v_0)$ and $p(v_1 \mid v_0)$ are not included yet. These two cases are different from the general case, and should be considered in addition. In particular, we can directly regard the mask of vertices (for example, the rightmost image in figure 3.4) predicted by the CNN as v_0's unnormalized probability distribution $\tilde{p}(v_0)$, and choose the position with the highest probability for v_0's assignment. As v_0 is known, there are typically two options for v_1, one is the next vertex on its left direction and another on right direction. To tackle this problem, we can simply specify the order of the polygon vertices to be fixed, so that v_1 can be uniquely determined. In our project, the order of polygon is set to be anticlockwise.

ConvLSTM The RNN uses ConvLSTM (Convolutional LSTM) cells as the polygon decoder to sequentially predict vertices. For simplicity, we can regard ConvLSTM as the function $F(\cdot)$ in equation 3.1. In fact, the structure of a ConvLSTM cell is almost the same as that of an ordinary LSTM cell, except

that it employs convolution in a 2D image with multiple channels instead of matrices or vectors multiplication. The introduction of ConvLSTM can significantly reduce the the number of parameters when it is compared with a fully connected RNN. The following equations define the computation process within a ConvLSTM cell, which is also visualized in figure 3.5.

$$\begin{bmatrix} f_t \\ i_t \\ g_t \\ o_t \end{bmatrix} = \begin{bmatrix} W_{hf} \\ W_{hi} \\ W_{hg} \\ W_{ho} \end{bmatrix} * h_{t-1} + \begin{bmatrix} W_{xf} \\ W_{xi} \\ W_{xg} \\ W_{xo} \end{bmatrix} * x_t + \begin{bmatrix} b_f \\ b_i \\ b_g \\ b_o \end{bmatrix} = W_h * h_{t-1} + W_x * x_t + b, \tag{3.2}$$

$$c_t = \sigma(f_t) \circ c_{t-1} + \sigma(i_t) \circ \tanh(g_t), \tag{3.3}$$

$$h_t = \sigma(o_t) \circ \tanh(c_t), \tag{3.4}$$

where x_t, h_t, c_t denote the input, the hidden state (or cell output), and the cell state of the cell at time step t respectively, i_t, o_t, f_t denote the states of input, output, and forget gate at time step t respectively, g_t denotes an intermediate variable, $\sigma(\cdot)$, $*$, $\circ$ denote the sigmoid function, convolution, and Hadamard (element-wise) product respectively, W_x and W_h denotes two convolution kernels for x_t and h_t respectively, W_{xf}, W_{xi}, W_{xg}, W_{xo} denote the four components of W_x in the dimension of channels and W_{hf}, W_{hi}, W_{hg}, W_{ho} denote the four components of W_h in the dimension of channels.

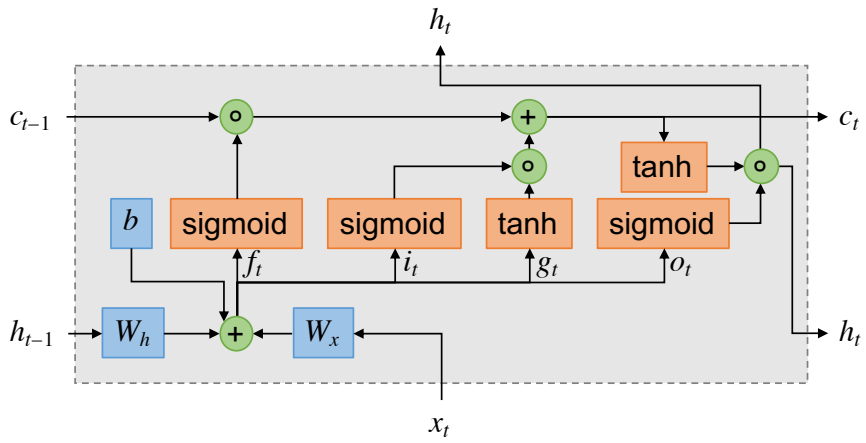


Figure 3.5: Visualization for LSTM cell.

Multilayer RNN PolygonRNN uses multilayer RNN with ConvLSTM cells. The connection between two neighboring layers can be described as follows, which means that the cell input of this layer comes from the cell output of the previous layer.

$$x_t^{(k)} = h_t^{(k-1)}, \forall k \in \{2, 3, \ldots, n\}, \tag{3.5}$$

where n is the number of RNN layers or ConvLSTM cells, k in the superscript denotes the k-th layer. Recall that the initial cell input (i.e. the cell input of the first layer) at time step t consists of the spatial information f from CNN (including the masks of boundary and vertices), the two previous vertices v_{t-1} and v_{t-2}, and starting vertex v_0, here it is formulated as follows.

$$x_t = x_t^{(1)} = v_{t-1} \oplus v_{t-2} \oplus v_0 \oplus f, \tag{3.6}$$

where $\oplus$ denotes the concatenation operation in the dimension of image channel. Thus, the equation 3.1 can be written as follows.

$$y_t = p(v_t \mid v_{t-1}, v_{t-2}) = \text{softmax}(W_y \text{vec}(h_t^{(n)})), \tag{3.7}$$

where W_y denotes the weights of the final fully connected layer, $\text{vec}(\cdot)$ denotes the vectorization function for a matrix. Figure 3.6 shows the entire structure of the RNN part and highlights the process at time step t, using the same notation as the equations 3.5, 3.6 and 3.7.

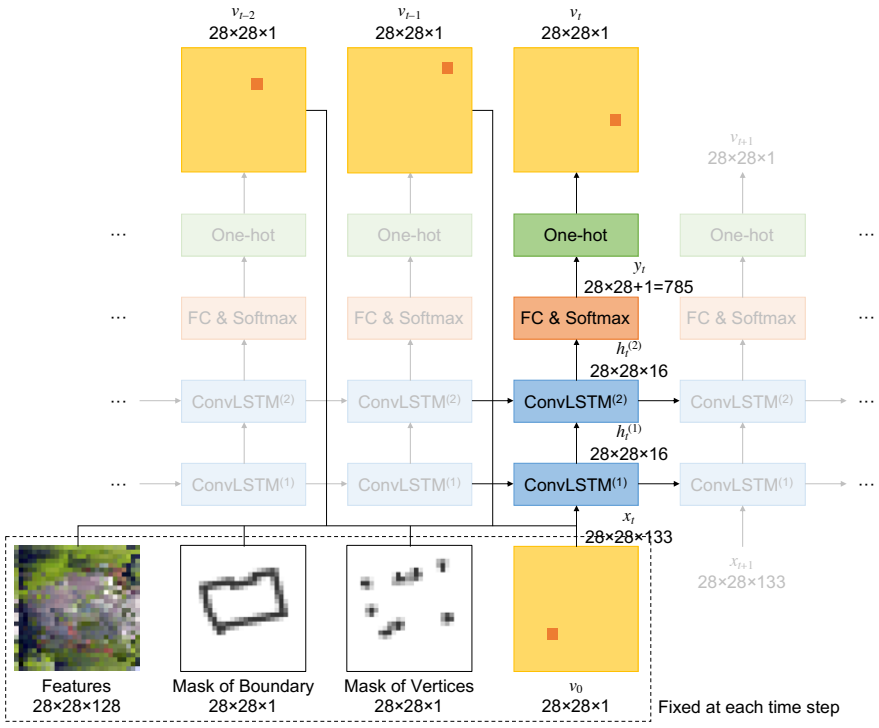


Figure 3.6: Visualization for the time step of the RNN decoder. In this figure, the outputs of the end signal are omitted, and the configuration of the ConvLSTM cell is the same as what is used in the original PolygonRNN paper. Specifically, the paper sets the number of RNN layers to 2 and uses ConvLSTM cells with 3×3 kernel size and 16 channels.

Note that at each time step, the features, the masks of boundary and vertices as well as the one-hot encoding of starting vertex are fixed. Only v_{t-1} and v_{t-2} change over time.

3.1.3 Loss Function

The loss of PolygonRNN consists of two parts, loss of CNN and loss of RNN. The loss function of CNN part uses log loss, since the mask prediction is equivalent to binary classification for each pixel. We also notice that the non-boundary pixels or the non-vertex pixels occupy the majority, the loss function should be further weighted. As for the loss of RNN, the loss function used is cross-entropy loss, because the polygon prediction is equivalent to multiclass classification at every time step.

3.2 Feature Pyramid Network

FPN is the core model for object detection in this project. Object detection problem is exactly multiple bounding boxes regression problem, which finds out multiple RoIs within an image. Subsection 3.2.1 first illustrates problem of single bounding box regression. Subsection 3.2.2 introduces some basic concepts related to anchor, and subsection 3.2.3 further illustrates multiple bounding boxes regression. Finally, subsection 3.2.4 looks into the backbone of FPN, presents how feature pyramid is generated and how FPN can tackle the multi-scale detection problem.

3.2.1 Single Bounding Box Regression

Single bounding box regression can be used for such a scenario, where the target image has only one RoI. Generally, a bounding box can be uniquely determined by four parameters, either box center and box size, denoting (x, y) and (w, h) or the coordinates of the upper left and lower right corners of the box, denoting (l, u) and (r, d). They have following relationships.

$$\begin{bmatrix} x \\ y \\ w \\ h \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -2 & 0 & 2 & 0 \\ 0 & -2 & 0 & 2 \end{bmatrix} \begin{bmatrix} l \\ u \\ r \\ d \end{bmatrix}, \tag{3.8}$$

$$\begin{bmatrix} l \\ u \\ r \\ d \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \\ h \end{bmatrix}. \tag{3.9}$$

Therefore, we can regard the problem of finding out single bounding box of RoI as a parameter regression problem. With the image as input, any set of

the four parameters, (x, y, w, h) or (l, u, r, d), can be selected as the target of the regression. Figure 3.7 shows the architecture of the network for the single bounding box regression, using box center and size as outputs.

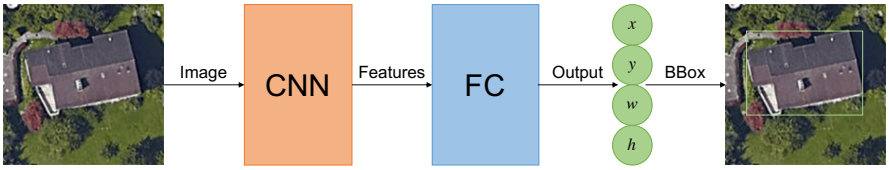


Figure 3.7: Simplified structure of the network for single bounding box regression. In this figure, the output of FC layer refers to the center and size of the bounding box.

In practice, usually the normalized parameters of box center and size rather than the coordinates of two corners are used as training targets. As for the normalization functions, see subsection 3.2.3 for more details.

3.2.2 Anchor and its Properties

Anchor is a basic concept in multiple bounding boxes regression. The following paragraphs introduce the idea of anchor and its properties in detail.

Anchor An anchor refers to an artificially specified bounding box in the original image, regardless of the number of RoIs or where these RoIs locate. An arbitrary rectangular area in the image can be an anchor. For an image with width w and height h, the total number of possible anchors n can be calculated as

$$n = \frac{1}{4} w(w+1)h(h+1). \tag{3.10}$$

Thus, even for a very small image patch with size 224×224, n is already very large, reaching 635.04 million. Figure 3.8 illustrates some examples of anchors in an image with multiple RoIs.

IoU Score Usually IoU (Intersection over Union) score is used to evaluate the similarity between an anchor and a ground truth bounding box. The following equation shows its computation.

$$s = \frac{|A \cap T|}{|A \cup T|} \in [0, 1], \tag{3.11}$$

where s denotes the IoU score, $|\cdot|$ denotes the size of the set, A and T refer to the sets of pixels covered by the anchor and the ground truth bounding box respectively. $s = 1$ means that the anchor and the ground truth bounding box overlap perfectly. Figure 3.9 shows the coverage level under different IoU scores.

(a) Positive (b) Natural (c) Natural (d) Negative

Figure 3.8: Example anchors in image with multiple RoIs. (a)–(d) show four possible anchors in an image with with multiple RoIs, with each showing one anchor. The red and light green rectangles refer to anchors and ground truth bounding boxes respectively. The polarities of these anchors are also shown, using the typical threshold.

(a) IoU = 0 (b) IoU = 0.3 (c) IoU = 0.7 (d) IoU = 1

Figure 3.9: Coverage level of anchor and ground truth bounding box under different IoU scores.

Anchor Polarity Generally, there are several buildings in an aerial image, thus multiple RoIs exist. Based on the IoU scores of an anchor with these ground truth bounding boxes, we can make a judgment on the polarity of this anchor, which is defined as follows.

$$d_i = \underset{j}{\operatorname{argmax}} \operatorname{IoU}(a_i, g_j), \tag{3.12}$$

$$s_i = \operatorname{IoU}(a_i, g_{d_i}), \tag{3.13}$$

$$p_i = \begin{cases} 1, & s_i > t_h, \\ 0, & t_l \leqslant s_i \leqslant t_h, \\ -1, & s_i < t_l, \end{cases} \tag{3.14}$$

where a_i and g_j denote the i-th anchor and j-th ground truth bounding box, s_i, d_i, and p_i denote the highest IoU score that a_i can get, the index of the 'closest' ground truth box of a_i, and the polarity of a_i, t_l and t_h denote two threshold for the polarity judgment, and typically, $t_h = 0.7$, $t_l = 0.3$. An anchor is labeled as positive, natural or negative in case of $p_i = 1$, $p_i = 0$ or $p_i = -1$ respectively. Figure 3.8 also shows anchors with different polarities.

3.2.3 Multiple Bounding Boxes Regression

Multiple bounding boxes regression can be very different from the single one. Since the number of RoIs is unknown, if we still directly use the parameters regression as what we do in case of the single bounding box, then the number of nodes of the FC output layer will not be fixed. Besides, the RoIs in the image usually reflects the local information, usually we cannot directly use all global features obtained by CNN, either.

Anchor Assignment The basic idea for multiple bounding boxes regression is to regress them on positive anchors. Specifically, each RoI is regressed on the anchors which are assigned to it. There are two rules for the assignment: (1) each ground truth bounding box should have at least one anchor (either positive, natural or negative) assigned to it; (2) each positive anchor should be assigned to its 'closest' ground truth bounding box. For those non-positive anchors that have assignments, we also treat them as positive anchors. For details of the matching algorithm, please refer to section A.4.

Classification and Regression on Anchors As a matter of fact, the single bounding box regression can be regarded as regression on the anchor, which is exactly the whole image. Similar to this, we can simply increase the number of nodes in the FC output layer (see figure 3.7) according to the number of anchors. Generally, each anchor requires 6 output nodes, 2 for binary classification and 4 for parameters regression. Specifically, 2 classification nodes output the logits, denoting l_p and l_n, indicating how close the anchor is to an RoI. Thus the predicted probability of an anchor referring to an RoI p^* can be described as

$$p^* = \frac{e^{l_p}}{e^{l_p} + e^{l_n}}, 1 - p^* = \frac{e^{l_n}}{e^{l_p} + e^{l_n}}. \tag{3.15}$$

The rest 4 regression nodes output the refined box information, for recovering the corresponding RoI. Suppose we have a fixed positive anchor, which is assigned to a ground truth bounding box. The refined (or normalized) box parameters are used for regression target, which can be described as follows.

$$r_x = \frac{x_g - x_a}{w_a}, r_y = \frac{y_g - y_a}{h_a}, r_w = \log \frac{w_g}{w_a}, r_h = \log \frac{h_g}{h_a}, \tag{3.16}$$

where (x_a, y_a, w_a, h_a) and (x_g, y_g, w_g, h_g) denotes the four parameters of the anchor and the ground truth bounding box respectively, (r_x, r_y, r_w, r_h) denotes the refined parameters for regression target. As for the recovery phase, we have

$$x_g^* = x_a + r_x^* w_a, y_g^* = h_a + r_y^* h_a, w_g^* = w_a e^{r_w^*}, h_g^* = h_a e^{r_h^*}, \tag{3.17}$$

where $(r_x^*, r_y^*, r_w^*, r_h^*)$ and $(x_g^*, y_g^*, w_g^*, h_g^*)$ denote the predicted refinement parameters and the parameters of the final predicted bounding box respectively.

The reason why the refined parameters are used for training, instead of original box, is that the former one eliminates the location information thus more reasonable.

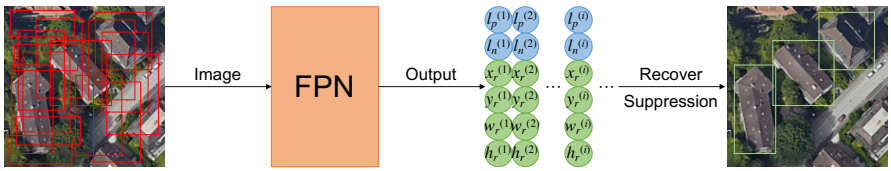


Figure 3.10: Simplified structure of multiple bounding box regression.

Loss As for training, only positive and negative anchors are used, where classification uses both and regression only uses positive anchors. The loss used for classification is the log loss, described as follows.

$$L_c = \mathbb{1}[p = 1] \log \frac{1}{p^*} + \mathbb{1}[p = -1] \log \frac{1}{1 - p^*}, \tag{3.18}$$

where L_c is the classification loss, p is the polarity of the anchor, p' is the predicted probability, $\mathbb{1}[\cdot]$ is the indicator function. The loss used for regression is the smooth L_1 loss, described as follows.

$$f(x) = \begin{cases} \frac{1}{2}x^2, & |x| < 1, \\ |x| - \frac{1}{2}, & |x| \geqslant 1, \end{cases} \tag{3.19}$$

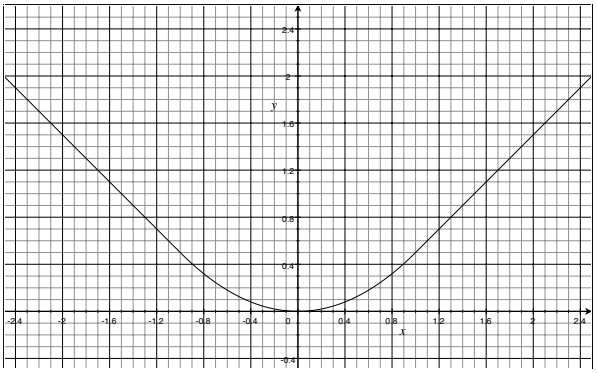


Figure 3.11: Smooth L_1 loss function.

$$L_r = f(x_g^* - x_g) + f(y_g^* - y_g) + f(w_g^* - w_g) + f(h_g^* - h_g), \tag{3.20}$$

where L_r is the regression loss. The total loss is defined as

$$L = \sum_i L_c^{(i)} + \lambda \sum_i \mathbb{1}[p_i = 1] L_r^{(i)}, \tag{3.21}$$

where λ is a self-defined parameter.

3.2.4 Feature Pyramid and Backbone

From equation 3.10 we know that even a small image can have a huge number of possible anchors. However, it is impossible to involve all anchors in the calculation, since the output layer would have too many nodes. Thus, how to select the proper anchors in the original image remains a problem. The selected anchors should cover the ground truth bounding boxes as much as possible. Under this guideline, selected anchors should have different kinds of shapes, in different scales and evenly distributed in the image.

In fact, in RPN from Faster R-CNN, there is a simple way to select anchors. For each entry of the convoluted feature map (or each pixel of the original image with corresponding stride), we generate k different shapes of anchors also in different scales, which is shown in figure 3.12. For example, if we select anchors from image with size 320×320, stride 8×8, 3 different shapes $(1:1, \sqrt{2}/2:\sqrt{2}, \sqrt{2}:\sqrt{2}/2)$ and 4 scales (16, 32, 64, 128), we will have $(320/8)^2 \times 3 \times 4 = 19200$ anchors.

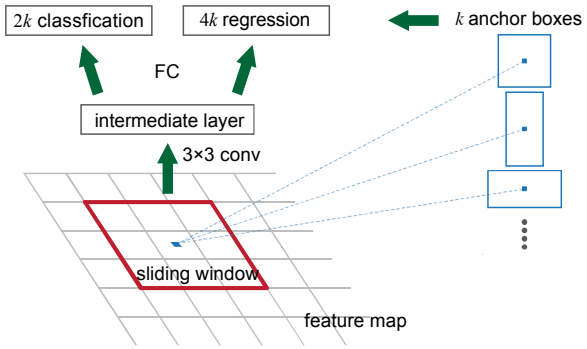


Figure 3.12: Anchor selection.

But this method produces two problems: (1) large anchors are too close to each other, resulting in redundancy; (2) the stride may be too large for small anchors, thus a typically small ground truth bounding box which are sandwiched between two small anchors is very likely to be missed. Both problems are tackled with the introduction of feature pyramid.

Feature pyramid is a basic component in recognition systems for detecting objects. It actually refers to semantic feature maps at different scales. FPN exploits the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids. The network uses a top-down architecture with lateral connections is developed for building an in-network feature pyramid from a single-scale input. Figure 3.13 shows its architecture.

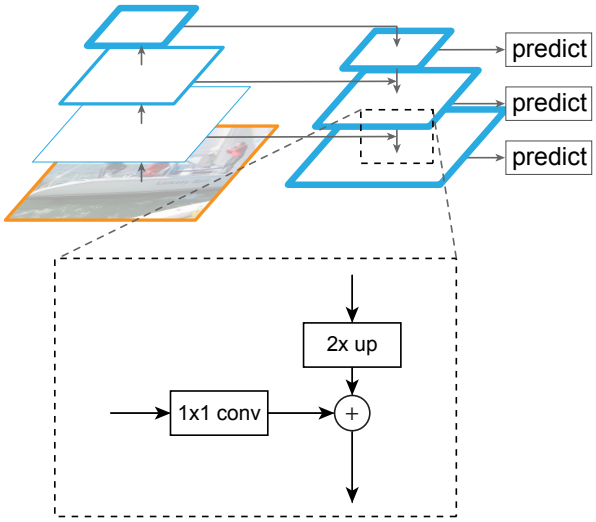


Figure 3.13: Structure of Feature Pyramid Network. The upper left part is the general convolutional process. The upper right part shows the feature pyramid. The lower part shows the lateral connections in detail.

The relationship between the feature pyramid and anchors is that each level of the feature pyramid corresponds to single scale of anchors, which can be seen in figure 3.14. In general, we have

$$S_i = 2^{i-1} S_1, i \in \{1, 2, \ldots\}, \tag{3.22}$$

where S_i denotes the smallest anchor size, S_i denotes the anchor size corresponding to the feature map with 2^{-i} resolution. For example, for an image with size 320×320, if the smallest anchor size is set to be 16, the anchor size for the feature map with resolution 40×40 would be 128.

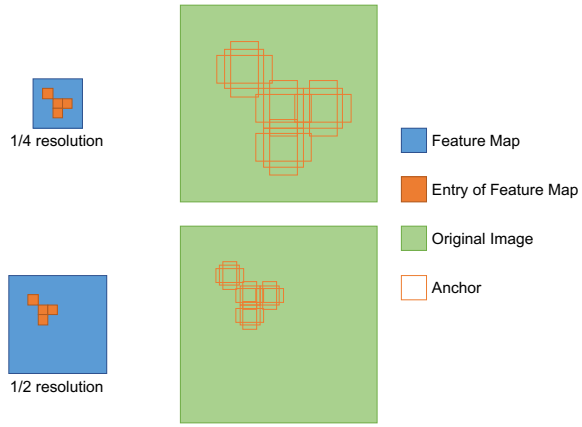


Figure 3.14: Generation of anchors from different layers of feature pyramid.

Therefore, in FPN, the feature map with larger resolution corresponds to

smaller anchor size (i.e. smaller stride with regard to the original image). That is the core reason why the problems mentioned above can be addressed. On the other hand, the total number of anchors n_l has an upper bound $\Theta(wh)$, which can be derived as

$$n_l = \sum_{i=1}^{l} \frac{w}{2^i}\frac{h}{2^i}k = whk\sum_{i=1}^{l} 4^{-i} = \frac{1}{3}whk(1-4^{-l}) \leqslant \frac{1}{3}whk, \tag{3.23}$$

where the subscript l denotes the number of feature maps, i.e. the depth of feature pyramid, w and h denotes the image width and height, k denotes the number of designed anchor shapes. Compared with equation 3.10, the total number of anchors is reduced by two orders of magnitude.

The upper left part of figure 3.13 is so-called backbone of FPN, which is exactly a general CNN. In Mask R-CNN, the backbone used for feature extraction is ResNet-101, which can give excellent gains in both accuracy and speed. However, in our project, VGG-16 is chosen as the backbone of FPN, because we want FPN and PolygonRNN share the same VGG-16. For more details about the shared VGG-16, please refer to subsection 3.3.2.

Figure 3.15 shows the architecture of FPN with VGG-16 backbone used in our project for feature pyramid extraction. Actually, the lateral connections are not fixed. It means that each layer of VGG-16 can be lead out in a reasonable situation. Here we use 4-layer feature pyramid, and features are taken from layers `conv3_3`, `conv4_3` and `conv5_3`. For the network to make further prediction is already shown in figure 3.12.

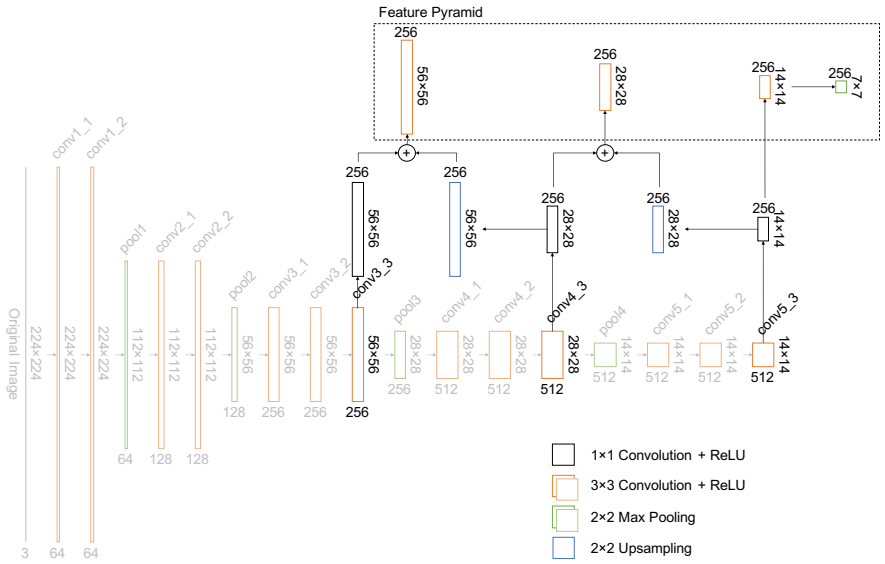


Figure 3.15: FPN with VGG-16 backbone.

3.3 R-PolygonRNN

In this section, the final model, R-PolygonRNN (Region-based PolygonRNN), is derived and introduced in detail. Here we propose 3 possible versions, denoting two-step version, hybrid version and hybrid version with RoIAlign, which are described in subsection 3.3.1, 3.3.2 and 3.3.3, respectively.

3.3.1 Two-step Version

The two-step version is very natural and easy to think of. In the first step, RPN is used to get the RoIs from the aerial image and each RoI contains a building. In the second step, PolygonRNN is used to get the geometrical shape. Thus, our problem is solved after these two steps. Figure 3.16 shows its pipeline.

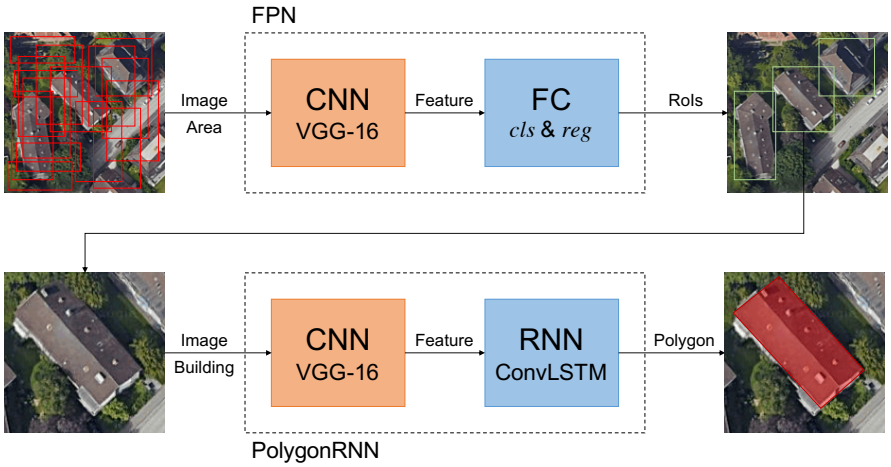


Figure 3.16: R-PolygonRNN , two-step version.

From the figure 3.16 we can see that RPN and PolygonRNN are completely separate. Therefore, during the training phase, we can train the two models separately and integrate them when making prediction.

3.3.2 Hybrid Version

Note that in figure 3.16, either when the area image is inputted to FPN or when the building image is inputted to PolygonRNN, they both pass CNN first, and their CNNs both adopt VGG-16. In fact, regardless of the whole model of FPN or PolygonRNN, the feature extraction in their CNN parts are both from the lateral connection on the basis of VGG-16. Although they extract features from different levels, the backbone of the original VGG-16 is

exactly the same for both of them. Therefore, they can share the same VGG-16 skeleton, just like what figure 3.17 shows.

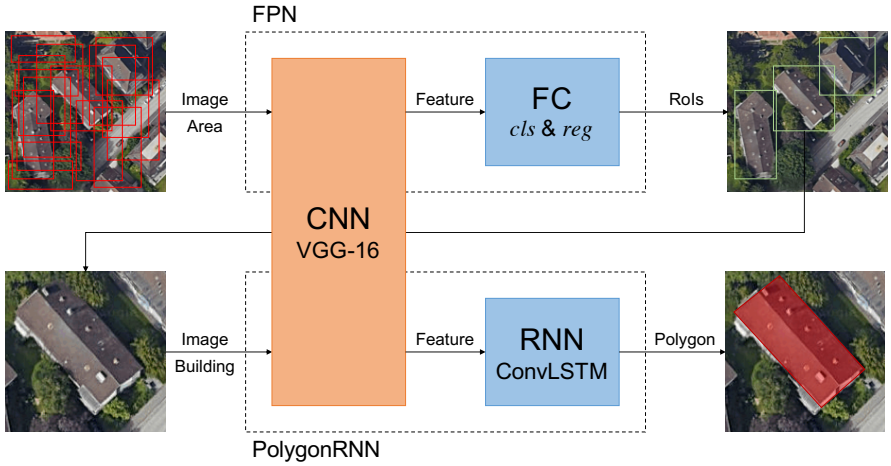


Figure 3.17: R-PolygonRNN , hybrid version.

As a result, the training phase of the whole model, R-PolygonRNN, cannot be separated and multi-task training is needed. In general, multi-tasking training can make the training phase better, and can therefore improve the the model's prediction effect and accuracy.

3.3.3 Hybrid Version with RoIAlign

Note that both two-step version and hybrid version do convolution for the area image and the building image respectively. However, we know that the building image is taken from the area image, so there is redundancy in the twice convolutions. Considering that we already have the feature map of the area image, it is unnecessary to do a second convolution since we can take the building feature from the corresponding position in the feature map of the area image, when given the RoI.

In Mask R-CNN, RoIAlign is proposed to solve this kind of problem. Actually its previous version, RoIPooling is already used in Faster R-CNN. The role of RoIPooling is to pool the corresponding area in the feature map into a fixed-size tensor, so as to perform subsequent classification phase. The positions of the RoIs are usually obtained by regression of the model, which are generally floating numbers. RoIPooling has twice process of rounding these floating numbers: (1) box boundaries are rounded to integers; (2) the region of box is divided into several cells, boundaries of each cell are rounded to integers. After the twice rounding process, each cell has a certain deviation from its initial position, which will affect the accuracy of subsequent classi-

fication or segmentation (for Mask R-CNN). This problem is summarized as 'misalignment' in Mask R-CNN paper.

In order to tackle the shortcomings of RoIPooling, Mask R-CNN introduces RoIAlign, using bilinear interpolation instead of rounding numbers. It can be used to obtain the value at the pixel whose coordinates are floating numbers, thereby transforming the pooling process into a continuous operation. Figure **??** shows an example of RoIAlign.

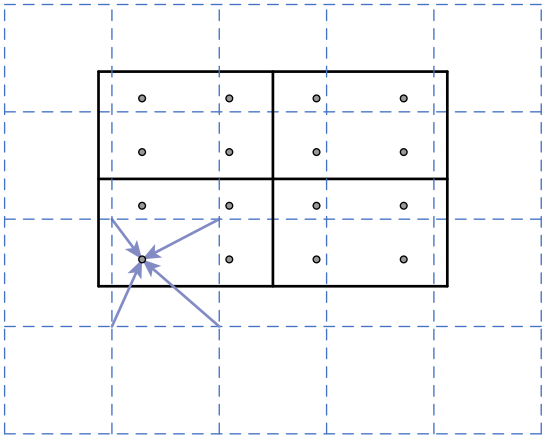


Figure 3.18: Example of RoIAlign.

Therefore, we can simply apply the bounding boxes obtained from FPN calculation directly on the feature map by RoIAlign and send the pooled features to the RNN part of PolygonRNN. Figure 3.19 shows R-PolygonRNNmodel architecture with RoIAlign.

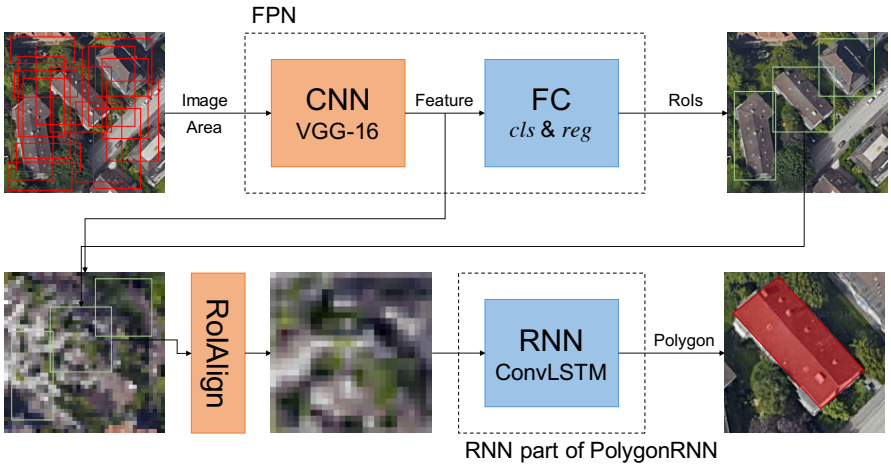


Figure 3.19: R-PolygonRNN , hybrid version with RoIAlign.

As a result, VGG-16 is used only once and therefore it can speed up the prediction in theory. Note that this version of R-PolygonRNNis not implemented yet since the resolution problem of the area image.

Chapter 4

Experiments and Results

In this chapter, the entire experimental process is presented, from the acquisition of the ground truth dataset (see section 4.1), to the implementation, training and prediction phases of the R-PolygonRNN (see section 4.2), and finally to the results evaluation and analysis (see section 4.3).

4.1 Ground Truth

Our ground truth dataset consists of satellite images and buildings' coordinates, which are used as inputs and labels respectively when training. In this project, all of the satellite images are collected from Google Static Maps API and all of the latitude and longitude coordinates of the polygon vertices of buildings are collected from OpenStreetMap. For details of the two APIs mentioned above, please refer to subsections 4.1.1 and 4.1.2.

As mentioned in section 3.3, the training phase of our model R-PolygonRNN requires two different kinds of ground truth dataset, areas with multiple bounding boxes for FPN and buildings with geometrical shapes for PolygonRNN, all of which are illustrated in subsection 4.1.3.

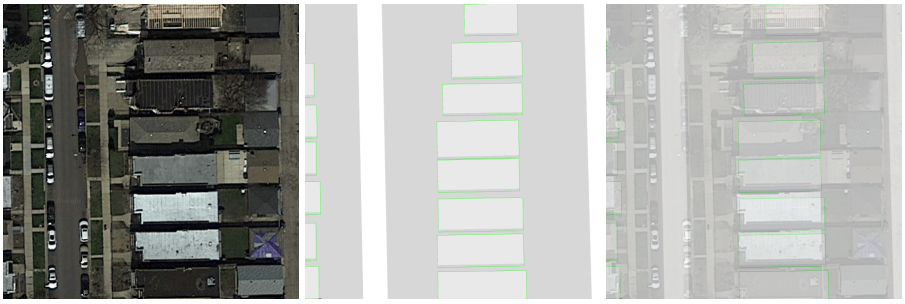
Since the whole dataset is collected from two different sources, the problem of inconsistency may exist. Subsection 4.1.4 describes details of the problems in the ground truth dataset, and subsection 4.1.5 proposes a solution to adjust the shift between buildings' images and polygons.

4.1.1 Google Static Maps API

Google Static Maps API[1] provides an interface that implements maps as high-resolution images. Users can download customized map based on URL with different parameters, which is sent through a standard HTTPS request.

[1] `https://developers.google.com/maps/documentation/static-maps/`

The parameters in URL includes the map type (satellite, roadmap, etc.), latitude and longitude coordinates of the image center, the resolution of the image, the zoom level, and the scale. For the usage of the Google Static Maps API, please refer to section A.1 in appendices. Figure 4.1(a) and 4.1(b) shows two types of images can be obtained by Google Static Maps API.



(a) satellite image (b) roadmap image (c) hybrid image

Figure 4.1: Example images downloaded through Google Static Maps API. (a) and (b) are obtained directly by the API, with 41.9399708 degrees north latitude, 87.7380649 degrees west longitude of the center (located in Chicago), both width and height 600 pixels, zoom level 20 and scale 1, but different in map types. (c) is obtained by overlaying (a) with (b), with 75% opacity. It can be clearly seen that the two images cannot match very well.

We would have liked to obtain buildings' coordinates from Google Static Maps API as well, but we find that the API does not provide such information directly. Instead, it gives the styled roadmap images like figure 4.1(b), where querying coordinates requires further corner detection but the boundaries may be still inaccurate (see figure 4.1(c) for example). Thus, this thesis project, only satellite images from Google Static Maps API are used.

4.1.2 OpenStreetMap

OpenStreetMap[2] provides an interface that implements a map as a XML format `.osm` file. The file contains all the information which can be used to recover the map. When the map's bounding box is given, users can retrieve the latitude and longitude coordinates of the buildings' vertices and roads' central lines within the map. For the usage of OpenStreetMap and the format of the `.osm` file, please refer to section A.2 in appendices.

The coordinates obtained by the API is the original latitude and longitude coordinates of the buildings' vertices, which cannot be directly used for training. We need to convert them to relative pixel coordinates with regards to an given satellite image.

[2] `https://www.openstreetmap.org/`

As a matter of fact, every fixed point on the earth can correspond to a unique pixel on the map at a specific zoom level. This projection process can be regarded as a function. Since we have already known the relative pixel coordinates of the image center (half width and half height), the projection for an arbitrary point with given latitude and longitude coordinates can be therefore computed. This process can be described as following equations.

$$(c_x, c_y) = (\frac{w}{2}, \frac{h}{2}) = f(c_{lon}, c_{lat}, z) + (\Delta_x, \Delta_y), \tag{4.1}$$

$$(p_x, p_y) = f(p_{lon}, p_{lat}, z) + (\Delta_x, \Delta_y), \tag{4.2}$$

where f is the function that projects latitude and longitude coordinates to global pixel coordinates (see A.3 in appendices for more details), c is the image center, p is an arbitrary point, the subscripts x, y, lon, lat mean the relative pixel and longitude and latitude coordinates respectively, z is the zoom level, Δ is the translation constant from the global to the relative pixel coordinate system. Thus, we have

$$(p_x, p_y) = f(p_{lon}, p_{lat}, z) - f(c_{lon}, c_{lat}, z) + (\frac{w}{2}, \frac{h}{2}), \tag{4.3}$$

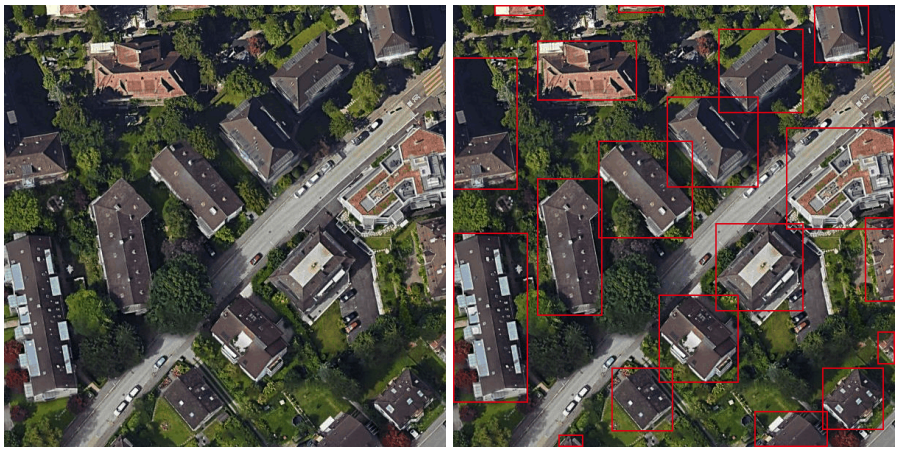
for any arbitrary point p.

4.1.3 Areas and Buildings

As mentioned in section 3.3, R-PolygonRNN is formed by FPN and PolygonRNN. However, the training phase of these two models requires two different kinds of ground truth dataset.

Areas for FPN In subsection 3.2 we have shown that FPN aims at finding rectangular regions of interests. Thus, training FPN generally requires a relatively large image with several objects in it. When considering our problem, an example of the ground truth can be an area of a city with several bounding boxes. Each box contains a single building, regardless of its tight polygon outline. Figure 4.2 gives an example area for training FPN and its visualized ground truth label.

Buildings for PolygonRNN In section 3.1 we have mentioned that PolygonRNN can only deal with images with single object, meaning that the bounding box of the object should be first given. Thus, training PolygonRNN generally requires a relatively small image with single object, as well as the information of its polygon outline. When it comes to our problem, an example of the ground truth can be a building with some padding and the pixel coordinates of building's vertices. Figure 4.3 gives example buildings for training PolygonRNN and its visualized ground truth label.



(a) example area (b) example area with bounding boxes

Figure 4.2: An example area in Zurich for FPN training. (a) is the original satellite image obtained by Google Static Maps API. (b) is (a) covered by multiple bounding boxes of buildings, which is visualized based on the ground truth.

(a) example buildings

(b) example buildings with polygon outlines

Figure 4.3: Example buildings in Zurich for PolygonRNN training. (a) is the original satellite images obtained by Google Static Maps API. (b) is (a) covered by the corresponding polygons' edges, which are visualized based on the ground truth.

4.1.4 Problems

Recall that the satellite images and polygons' coordinates are collected from two different sources, the problem of inconsistency may exist. That is to say, the polygon and the actual building we see in the image may not match well in some cases. This inconsistency is mainly reflected in the following aspects.

(a) general shift (b) polygon size and shape

(c) angle issue (d) adjacent buildings

Figure 4.4: Problems existed in the ground truth dataset.

General Shift Figure 4.4(a) gives two typical shift examples in the dataset. This shift is generally within the range of 30 pixels, either up and down, or left and right.

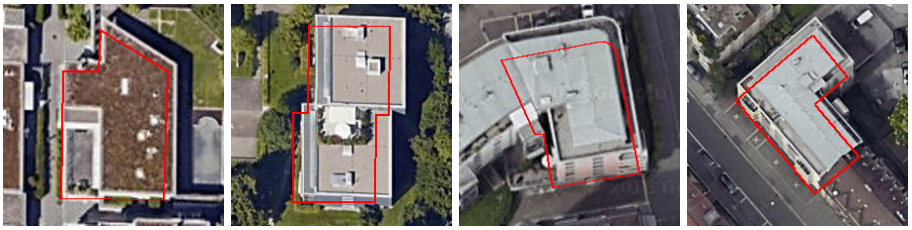
Polygon Size and Shape Sometimes the polygon size and shape is different from the actual building size as well. This is typically because OpenStreetMap measures the bottom of the building and the aerial image is taken from the top. The roof of the house we see from the top can be different from the floor area. Figure 4.4(b) shows a building with a smaller polygon and a building with a too rough outline.

Angle Issue Some tall buildings may suffer from the angle issue. This is because the satellite cannot always be vertically right above the building, so the side of these very high buildings will be photographed. Figure 4.4(c) gives two examples.

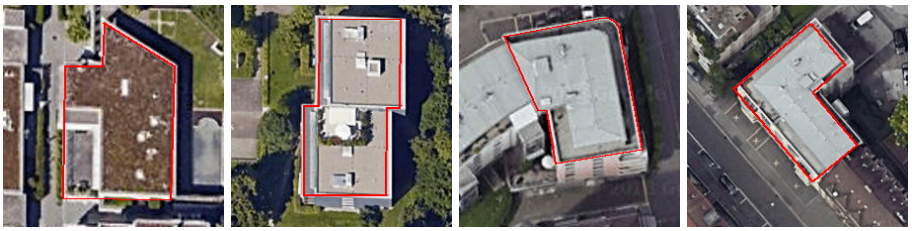
Adjacent Buildings Some adjacent buildings will be very close together, and some even share a common roof. Thus, from the satellite image, it is generally difficult to distinguish the boundaries between them. However, these buildings are separated in the dataset of OpenStreetMap. It means that where there are originally edges between buildings, it looks like there is no edge in the image. This kind of inaccuracy would have negative effect on the training phase. Figure 4.4(d) and gives two examples.

4.1.5 Adjustments

Problems such as too rough polygon shape, angle issue and adjacent buildings are unsolvable unless the data sources can correct themselves. What we can improve is to tackle the general shift and the polygon size problem and make the image-polygon pairs more matching. By observation, we can get to some conclusions for a matching image-polygon pair.



(a) before adjustment



(b) after adjustment

Figure 4.5: Adjustment examples. (a) shows the polygons before adjustment, while (b) shows the polygons after adjustment.

Color Variance The color variance of image pixels covered by polygon generally reaches the local minimum, because the roof of a building usually has only one color or several similar colors.

Edges The polygon edges generally lie within the output of the edge detection (e.g. Sobel, Canny) performed on the image.

Corners The polygon vertices can generally match the output of the corner detection (e.g. Harris, SIFT) performed on the image.

Based on these conclusions, we propose a brute force shift adjustment method. Briefly, we resize and translate the initial polygon within a certain range, and traverse all the cases to see where the minimum color variance, maximum edge and corner coverage are reached. Figure 4.5 illustrates some examples which compare the polygons before and after the adjustment. Results show that this algorithm can well address the shift problem to some extent.

4.2 Implementation Details

In this section, several implementation details are given in order to the replicate the model in the future if needed. Subsection gives information and some notes of the dataset. Subsection 4.2.2 gives the basic configuration of the model. Subsection 4.2.3 and 4.2.4 focus on the training and prediction phase respectively.

4.2.1 Dataset Information

The dataset consists of two cities, Zurich and Chicago, including all the buildings in the range shown in table 4.1.

Table 4.1: Sampling range of buildings in Zurich and Chicago.

City	Zurich	Chicago
East Boundary	E 8.4092916°	W 87.8039744°
South Boundary	N 47.2879518°	N 41.8799477°
West Boundary	E 8.6729490°	W 87.6721554°
North Boundary	N 47.4664863°	N 41.97799394°
Number of Buildings	96,573	228,074

Note that buildings with more than 20 or less than 4 vertices are excluded. The building images are all squares but in different sizes, and include paddings. All polygons are set to be clockwise, and in each polygon, three consecutive vertices cannot be collinear. If so, the second vertex is removed.

Aerial images are taken based on the information shown in table 4.2.

Table 4.2: Sampling range of areas in Zurich and Chicago.

City	Zurich	Chicago
Center Longitude	E 8.5468221°	W 87.7380649°
Center Latitude	N 47.3768587°	N 41.9289708°
Step Longitude	$4.023094° \times 10^{-4}$	$4.469772° \times 10^{-4}$
Step Latitude	$2.724221° \times 10^{-4}$	$3.324594° \times 10^{-4}$
Horizontal Step Range	(-144, 144)	(-144, 144)
Vertical Step Range	(-144, 144)	(-144, 144)
Zoom Level	19	20
Scale	1	1
Image Size	(600, 600)	(600, 600)
Number of Areas	57,741	77,716

Note that areas without any buildings are excluded. Besides, there is an overlap between neighboring images because each building is required to appear completely at least once within all area images. For example, in figure 4.2(a),

for the incomplete buildings locate near the image top edge, we can find their complete shapes in its northern neighboring area image.

4.2.2 Model Configuration

Table 4.3 shows the parameters used in our model.

Table 4.3: Configuration parameters

Item	Value
Area Image Resize	(256, 256)
Building Image Resize	(224, 224)
Resolution of Output	(28, 28)
RNN Max Sequence Length	20
RNN Number of Layers	3
LSTM Number of Output Channels	(32, 16, 8)
Number of Layers of Feature Pyramid	4
Anchor Scale	(16, 32, 64, 128)
Anchor Shapes	(1:4, 1:2, 1:1, 2:1, 4:1)
Feature Shapes	(64×64, 32×32, 16×16, 8×8)
Feature Stride	(4, 8, 16, 32)

4.2.3 Training Phase

Table 4.4 shows the configuration during the training phase of the model.

Table 4.4: Configuration during training phase

Item	Value
Optimizer	Adam
Learning Rate	10^{-4}
Area Batch Size for FPN	4
Building Batch Size for PolygonRNN	12

Furthermore, buildings near the edges of area images are typically ignored when training since they are more likely to be incomplete. In this case, the polygon would be clipped by the edges, resulting in new vertices, which can be very hard to compute. Therefore, in this project, two different kinds of ground truth dataset are collected separately, which means that in a specific round of training, the buildings come from the dataset instead of patches from area image. Training in this way can keep the batch size of buildings unchanged.

4.2.4 Prediction Phase

In the prediction phase, the beam search algorithm is introduced. It is a heuristic graph search algorithm, which is usually used when the solution space of a graph is relatively large. In order to reduce the space and time occupied by the search, at each step of depth expansion, some poor quality knots are cut off, and higher quality nodes are kept. This process can significantly reduce space consumption and improve time efficiency, but the disadvantage is that there may be potentially optimal solutions discarded. Therefore, the search result is not necessarily the optimal solution. In beam search, the beam width refers to the number of nodes to be kept and in our project, the beam width is set to be 6. Figure 4.6 shows a single step of the beam search algorithm with beam width 3.

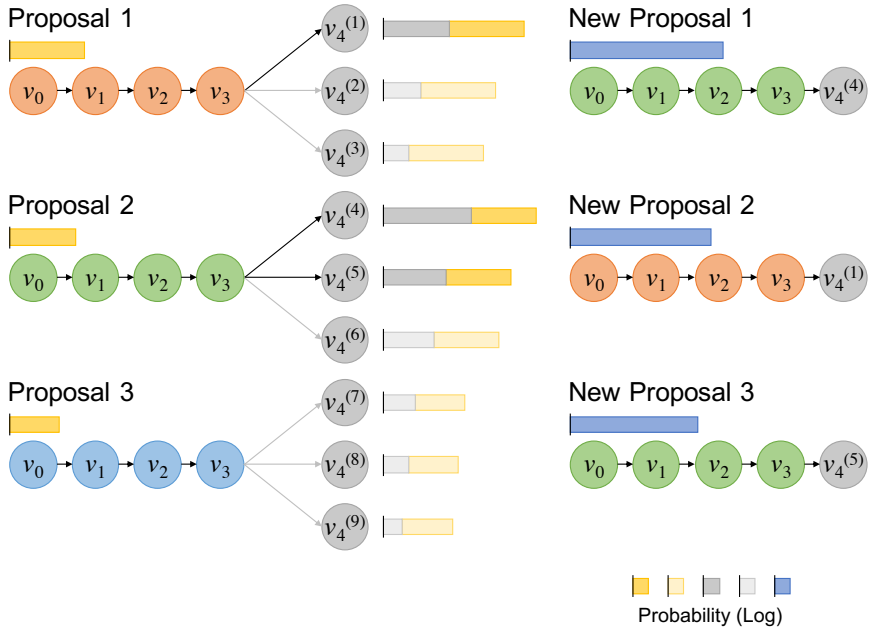


Figure 4.6: Example of single step of beam search algorithm.

4.3 Experiment Results

4.3.1 Single Building Segmentation

4.3.2 Buildings Localization

4.3.3 R-PolygonRNN

Chapter 5

Problems and Future Work

In this chapter, some problems in the prediction results are presented in section 5.1, and the future directions are illustrated in section 5.2.

5.1 Problems

This section mainly shows the two existed problems in the prediction result, and proposes corresponding possible solutions. Subsection 5.1.1 shows the resolution problem causing the inaccurate prediction and subsection 5.1.2 shows the false vertex problem.

5.1.1 Output Resolution

We know that the output grid for a single vertex has the resolution 28×28. The input resolution of PolygonRNN is 224×224. Thus, a single pixel in the output grid corresponds to an area of 8×8 in the original image. This will result in inaccurate output vertices.

(Example)

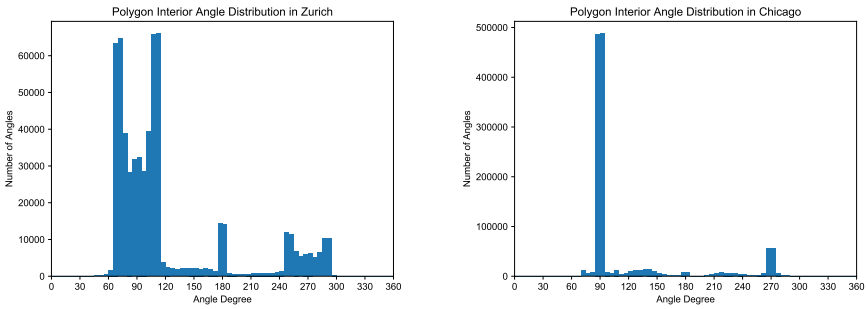
The possible solution is to extract features from relatively shallow levels. However, these layers may not have much semantic information, which may further have negative effect on the prediction accuracy of RNN part. In addition, because of the presence of FC layer at the end of the output part of the RNN, the number of its weights would increase at the fourth power. For example, currently the FC layer has $(28 \times 28)^2$ weights, if we increase the resolution to 56×56, the number of weights would be $(56 \times 56)^2$, 16 times the original. Therefore, the conclusion is that if the resolution is increased, then the network structure would certainly need to make a number of changes.

5.1.2 False Vertex

Actually, after the introduction of beam search algorithm, the false vertex problem has been solved a lot and the prediction is greatly improved, but there are still some in the results.

(Example)

The possible solution comes from the point of view of geometry, penalizing the angle degree of the polygon. Figure 5.1 shows the polygon interior angle degree distribution in the dataset of Zurich and Chicago.



(a) angle distribution in Zurich (b) angle distribution in Chicago

Figure 5.1: Polygon interior angle distribution.

From figure 5.1(b) we can see that in Chicago, most angles are at 90° or 270°, indicating that most of them are right angles. In Zurich's figure 5.1(a), although most angles are around 90° or 270°, they have certain deviations about ±20°. Also, there is a peak near 180° in figure 5.1(a), which means that there are many three-point collinearity in the dataset of Zurich.

When doing the probability computation of the next vertex, we can use the distribution of the angle as a priori to guide the prediction.

5.2 Future Work

In this section, several future directions are presented, such as possible direction of road segmentation, the improvement of the dataset, the methods that are helpful for training and the possible extensions of the model.

Road Segmentation At present, our project does not include road segmentation. This is because roads cannot usually be described in terms of polygons. We hope to add this in the future so that the geometrical segmentation in aerial images can become complete. The idea is to convert roads into centerlines and intersections. Actually, in Mask R-CNN, the part of human

posture detection is doing this, where a person is structurized as several keypoints and edges that connect them. Figure 5.2 shows the human posture detection result of Mask R-CNN. However, it is difficult to apply this method

Figure 5.2: Human posture detection in Mask R-CNN.

to our project, because the road is different from the human body, the number of keypoints of roads in each image is not fixed. In addition, paper (XXX) uses incremental graph construction process, which may be possible to be used in our project.

Ground Truth Correction Subsection 4.1.5 has already introduced a method to correct ground truth. However, there are still some image-polygon pairs inaccurate, especially for those adjacent buildings (see figure 4.4(d)). We hope that dataset can be collected from single source, so that the polygons and the buildings are more likely to match each other. As for the adjacent buildings, it is difficult for us to merge their polygons into a single one. Thus, we hope polygons of buildings instead of houses can be provided.

RoIAlign Implementation Subsection 3.3.3 has already introduced the hybrid version with RoIAlign of R-PolygonRNN. We hope that in the future this kind of version can be implemented.

Training Method Our model does not use pre-trained VGG-16 during training, neither as an initializer nor as fixed weights. As for training, in the future we hope the alternating training method can be introduced, just like what shows in the Fast R-CNN paper.

Model Generalization In fact, our model can be applied not only to the geometrical shape segmentation in the aerial images, but also to the general multi-object segmentation. We hope that our model R-PolygonRNNcan be generalized to other fields and we can compare the performance under different dataset.

Appendix A

Appendix

A.1 Example URL of Google Static Maps API

`https://maps.googleapis.com/maps/api/staticmap?maptype=satellite&center=4`

`&style=feature:all|element:labels|visibility:off&style=feature:landscape.`

A.2 OpenStreetMap

According to the Google Maps JavaScript API [1],

A.3 Projection

A.4 Anchor Assignment

$$A_{ij} = \begin{cases} 1, & \text{if } p_i = 1 \text{ and } j = d_i, \text{ or} \\ & j \notin \{d_i \mid i = 1, 2, \ldots\} \text{ and } i = \underset{k}{\operatorname{argmax}} \operatorname{IoU}(a_k, g_j) \\ 0, & \text{otherwise,} \end{cases} \tag{A.1}$$

$$A_{ij} = \begin{cases} 1, & \text{if } a_i \text{ is assigned to } g_j, \\ 0, & \text{otherwise,} \end{cases} \tag{A.2}$$

where A is the assignment matrix.

A.5 Non-max Suppression

dummy text

[1] https://developers.google.com/maps/documentation/javascript/coordinates

A.6 Beam Search

$$p(v_0 v_1 \dots v_n) = p(v_0)p(v_1 \mid v_0)p(v_2 \mid v_0 v_1) \dots p(v_n \mid v_0 v_1 \dots v_{n-1}), \quad \text{(A.3)}$$

$p(v_0^{(1)}), \dots, p(v_0^{(b)})$

$p(v_1^{(1)} \mid v_0^{(i)}), \dots, p(v_1^{(b)} \mid v_0^{(i)})$

[?, ?, ?, ?, ?, ?, ?]

List of Figures

List of Tables

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

ETH Zurich
ETH-Bibliothek
ETH E-Collection
Rämistrasse 101
8092 Zurich
Tel. +41 44 632 78 64
Fax +41 44 632 13 57
www.library.ethz.ch

Declaration of consent

For the publication of a diploma or master thesis on the ETH E-Collection institutional repository of ETH Zurich

ETH Zurich's institutional repository allows users to access diploma or master theses written at ETH Zurich. This offers academics the opportunity to present their work worldwide. The published theses fulfil the specified formal quality criteria.

Declaration of the author

I hereby declare that I consent to the ETH-Bibliothek making my diploma or master thesis, which I shall submit to the ETH-Bibliothek as a pdf file, available to the public on the institutional repository. The rights of third parties are not infringed by this publication. I consent to any subsequent necessary conversions into other data formats being made.

Author:
Title of the publication:

Department:
Publication year:
E-mail/Telephone number:
Private Address:

Zurich, on Signature _____

Recommendation of the responsible ETH professor or research supervisor

I have supervised this diploma or master thesis at ETH Zurich and recommend its publication. In particular, I hereby declare that the publication of this thesis does not infringe the rights of third parties and that any rights to confidentiality are protected.

Zurich, on Signature _____

Declaration of the ETH-Bibliothek

The ETH-Bibliothek guarantees the long-term availability of the thesis as well as the integrity of its content and its authenticity and will make the document accessible via the Internet. The intellectual property rights remain with the author.