Using Deep Neural Networks for Automatic Building Extraction with Boundary Regularization from Satellite Images

Kang Zhao

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN EARTH AND SPACE SCIENCE

YORK UNIVERSITY
TORONTO, ONTARIO

August 2019

Abstract

The building footprints from satellite images play a significant role in massive applications and many demand footprints with regularized boundaries, which are challenging to acquire. Recently, deep learning has made remarkable accomplishments in the remote sensing community. In this study, we formulate the major problems into spatial learning, semantic learning and geometric learning and propose a deep learning based framework to accomplish the building footprint extraction with boundary regularization. Our first two models, Post-Shape and Binary Space Partitioning Pooling Network (BSPPN) integrate polygon shape-prior into neural networks. The other one, Region-based Polygon GCN (R-PolyGCN) exploits graph convolutional networks to learn geometric polygon features. Extensive experiments show that our models can properly achieve object localization, recognition, semantic labeling and geometric shape extraction simultaneously. The model performances are competitive with the state-of-the-art baseline model, Mask R-CNN. Especially our R-PolyGCN, consistently outperforms others in all aspects.

Acknowledgements

Firstly, I would like to extend my sincere gratitude to Dr. Gunho Sohn as my supervisor throughout my two-year Master program and one-year undergraduate study. Being exposed to the research world for the first time in my life, I have always been provided with immersive guidance and endless encouragement from Dr. Sohn. As a freshman for academic world, I was seldom misled thanks to the continuous profound advice from him. Appreciation is also granted to Dr. Sohn for his mental and physical care for me, a kid being abroad and away from home. Specially thank Dr. Sohn for bringing me to the Conference on Computer Vision and Pattern Recognition (CVPR) 2018 at Salt Lake City, US. The great trip impressed me with academic passions and craziness and encouraged me to be dedicated to be a better researcher. In addition, I would like to thank Dr. Costas Armenakis as my committee member for his critical suggestions on my research work.

Apart from the supervisors, I would like to thank all the collegues of Sohn's lab. Special thanks are given to Dr. Junwon Kang for his constant and strong help with the implementation of deep learning models, Dr. Jaewook Jung for his generous help with the codes of MDL optimization, Dr. Andreas Wichmann for his patient help with the BSP codes. Also, thanks for the supports from Dr. Connie Ko, Dr. Pio Claudio, Dr. Yujia Zhang, Ali Baligh, Kivanc Babacan, Maryam Jameela, Zahra Arjmandi, Muhammad Kamran, Afnan Ahmad, Kunwoo Park, Sun Park, David Recchia, Phillip Robbins and other members of the lab. In addition, I appreciate that Yar Rouf from the Department of Electrical Engineering and Computer Science of York University to provide crucial writing suggestions.

I would like to thank my friends, Bo Chen, Leihan Chan, Kunwoo Park, Chifeng Shen, Enzhong Li, Shihang Yang, Mahmoud Rouby, Joey Bose and so on. Your motivation and companionship light my days at Toronto.

I wish to appreciate the finacial supports from Natural Sciences and Engineering Research Council of Canada (NSERC), Intelligent Systems for Sustainable Urban Mobility (ISSUM) project and York University.

Last but not least, I would like to thank my parents Xirong Guo and Haiping Zhao, for their long-term unselfish love and spiritual supports and understandings during my time at Canada.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Acquiring information about the structure on the surface of the earth without making physical contact is generally achieved by the remote sensing techniques [1]. Satellite images, as one of the most important remote sensing data sources, are widely utilized in applications like digital mapping, land use analysis, disaster monitoring, climate modeling and so on, which benefit from the spatial, spectral, temporal and radiometric resolutions and the large-scale coverage of the satellite images. Among them, the satellite images has played an indispensable role in the generation of the digital maps for the Geographic Information System (GIS), for instance the google map[1] or the Bing map[2]. On a digital map, the building footprint information is essential for many tasks, such as urban planning, smart city construction and so on, thus making the building footprint extraction a continuous heated topic in the community. Moreover, the building footprints with regularized boundaries can provide polygons of vector representation,

[1]`https://www.google.ca/maps`
[2]`https://www.bing.com/maps`

which possess stronger transferability over multiple GIS platforms thus having a wider range of applications. For example, the regularized building polygons can produce 3D building models with higher accuracy [2, 3]. However, with dramatically growing availability and accessibility of the satellite images, the extraction of the building footprints has demanded higher quality yet has not been fully exploited due to the following challenges.

(1) The building footprints on the GIS maps require the manual or semi-automatic procedure to reach the high precision, which is quite time-consuming and labour-intensive. For example, the OpenStreetMap (OSM)[3] can provide building footprints with fine-grained qualities, which are utilized by many open datasets to improve the precision of their building annotations. Nevertheless, the OSM itself relies on massive amount of contributions from the online users to manually correct and enhance the building footprints. Note that the amount of the satellite images is always huge, which take plenty of time to process.

(2) The enormous diversity of the outlooks of the building roofs creates barriers for large-scale building footprint extraction from satellite images. Many models can well detect the footprints within small areas where the building roof outlooks are alike but fail to generalize to other areas with buildings of different styles.

(3) The recognition and localization trade-off is challenging to balance. The high-level features of the images present abstract **semantic** information to recognize objects but lack the **spatial** information for the localization and it's vice versa for the low-level features. The sizes of satellite images are usually quite large and contains buildings of various scales, thus making it more difficult to extract

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

high level features to achieve correct recognition while preserving the spatial resolution to produce accurate localization.

(4) The geometric potential of the satellite images has not been fulfilled. Due to the pixel-wise and grid-based representation of the images, it's fairly demanding to learn the **geometric** information of polygon shapes. The geometric features are vital for the problem of building boundary regularization, which can help to improve the quality of the building footprints and produce vector-based outputs for a wider range of applications. Previous works like [2, 3] adopt additional Light Detection and Ranging (LiDAR) point cloud data rather than images solely to provide a solution since the point clouds have more evident geometric attributes. However, it's not realistic to create large-scale maps with LiDAR point clouds due to the fact that they are much more expensive than images.

In recent years, deep learning has brought about a revolution in Artificial Intelligence (AI) and become prevailed in the fields of computer vision, speech recognition, natural language processing, etc and it can produce superior performance compared to many traditional techniques [4]. In the remote sensing community, deep learning is also gaining more and more attention and popularity. We investigate the application status of deep learning models to the topics of remote sensing and building footprint extraction by respectively counting the total numbers of the annual research papers on related subjects. The statistic data are collected from the web of science website[4]. The results are shown in the figures 1.1 and 1.2. It's obvious that the research into deep learning for remote sensing and building footprint extraction has sharply increased since 2015 and is showing a rapidly growing trend at present. The deep learning models, Deep Neural Networks (DNN) are able to automatically learn rich and abstract features from large

[4]`https://apps.webofknowledge.com/`

amount of training data and hold considerable ability of generalization to the testing (new) data. Besides, they have been proven to achieve the state-of-the-art outcomes in the area of computer vision, especially for the tasks of object detection and image segmentation.

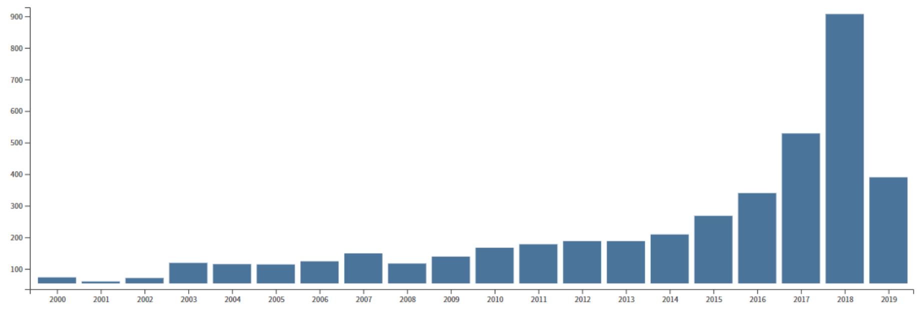


Figure 1.1: The number of annual research papers related to remote sensing and deep learning.

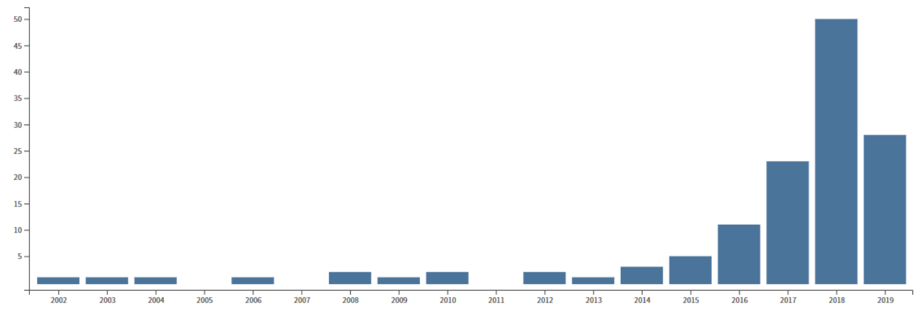


Figure 1.2: The number of annual research papers related to building footprint extraction and deep learning.

Above all, motivated by the massive usage of the building footprints from satellite images and the challenges of building footprint extraction plus building boundary regularization and to take advantage of the powers of deep learning, in this study, we will propose models based on deep neural networks with fully exploration and utilization of the spatial, semantic and geometric information to perform automatic building footprint extraction and tackle the problem of building boundary regularization.

1.2 Problem Domain

To build the deep learning frameworks to automatically extract building footprints with boundary regularization from the satellite images, we formulate the problems into three major parts: spatial learning, semantic learning and geometric learning.

1.2.1 Spatial Learning

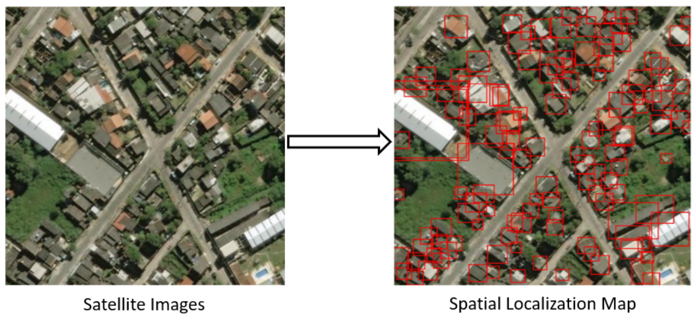


Figure 1.3: Spatial learning: object detection. Red boxes are the bounding boxes of buildings, which describe the building locations.

The spatial learning is targeted at the object localization problem, which in detail is to detect the locations of the bounding boxes for building objects (figure 1.3). To localize the objects in the image, the deep neural networks are designed to extract spatial features of the objects. We first wrap the possible regions containing the objects with features and then normalize the wrapped features into features of fixed size. In this way, the spatial features are extracted and they are also referred as the Region of Interest (RoI) features. Besides, the classification of the objects in the bounding boxes are usually performed with the object localization at the same time. So here the object localization is expanded to object detection.

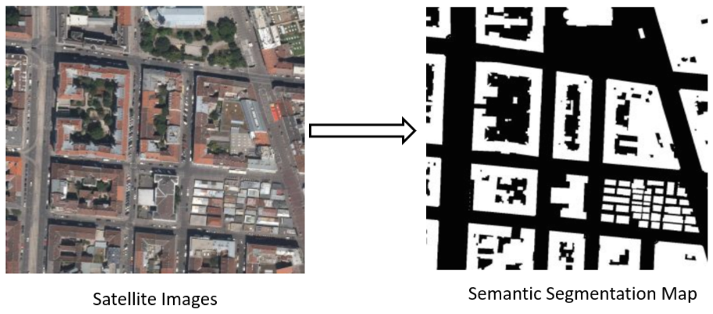


Figure 1.4: Semantic learning: image segmentation. In the binary map at right side, the white regions stand for the buildings while the black ones are the background.

1.2.2 Semantic Learning

The semantic learning is to classify the pixels of the image, which belongs to the image segmentation task. In our study, since there are two classes, the buildings and the background, the input image will be segmented into a binary map (figure 1.4). Deep learning models are able to obtain semantic features at all levels, from low-level edges to high-level object abstractions. High-level semantics are key to recognize the categories of the whole object while low-level and middle-level semantics play a significant role in the classification at pixel-level. Therefore, the ways to combine the semantic information of various levels from the networks are critical to the segmentation performance.

1.2.3 Geometric Learning

Geometric learning in our study is to predict the shapes of polygons. As shown in the figure 1.5, the footprints with irregular boundaries have no geometric properties while those with regularized boundaries have the geometric shapes of polygons. We argue that learning geometric information of polygons provides the solution to the boundary regularization problem. In geometry, polygon shapes can be delineated as polygon vertices or polylines (edges), which have non-grid structured representations. The geometric features extracted from the deep neural networks can be grid-based features with shape constrains like shape-prior. Non-grid features contain more geometric information and can be learned through deep learning models as well. By geometric learning, the optimal polygon shapes can be produced, thus generating building footprints with regularized boundaries.

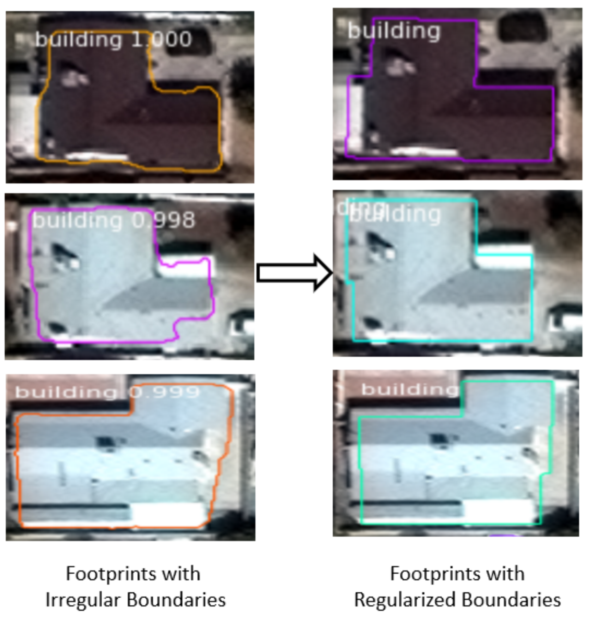

building 1.000
building
building 0.998
building
building 0.999
building
Footprints with Irregular Boundaries
Footprints with Regularized Boundaries

Figure 1.5: Geometric learning: polygon shape prediction.

1.3 Research Objectives

The major research objectives of this study are to establish deep learning models to provide solutions to the problems of automatic building footprints extraction with boundary regularization from satellite images. The spatial learning, semantic learning and geometric learning are supposed to be combined in our framework, which will be introduced in the following section.

1.3.1 General Research Framework

Our general framework employs the typical supervised learning mechanism illustrated in the figure 1.6. From the perspective of computer vision, our framework can be viewed as the instance segmentation model (in the subsection 2.1.3) combined with geometric learning, which can simultaneously recognize and localize multiple objects, assign semantic labels at pixel-level and predict polygons of geometric shapes.

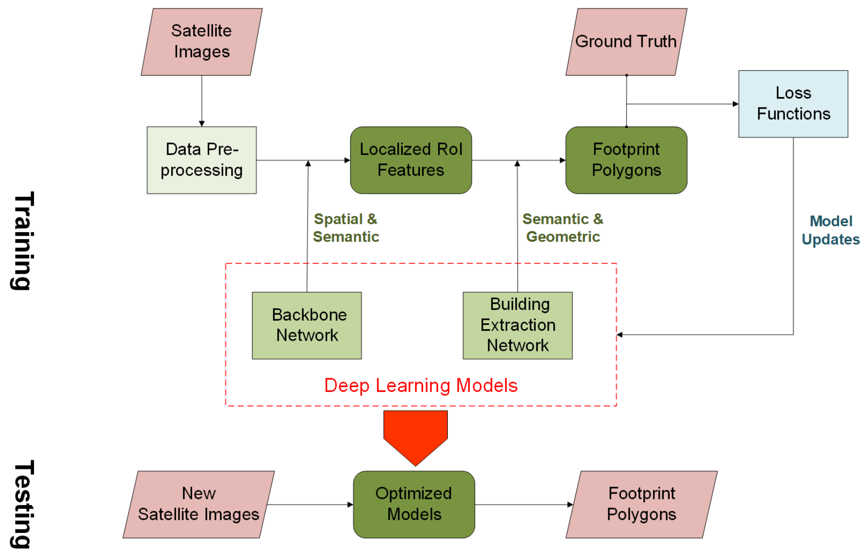


Figure 1.6: The general framework of our study.

The input data are satellite images and ground truth annotations of the building footprints, which are referred as training data. The satellite images are first pre-processed to inputs of appropriate formats for the neural networks. Our deep learning models are composed of the backbone network and the building extraction network. The back-bone network is designed as the feature encoder and object detector, which are able to learn the spatial information and semantics at object level to localize and classify the building objects. In addition, the backbone network extracts well-localized RoI features, which are shared and utilized by the following networks. On the top of the

RoI features, the building extraction network is built for semantic and geometric learning to predict polygons of the building footprints with regularized boundaries. Two types of architectures of the building extraction networks are created. One is to integrate polygon shape-prior into the image segmentation network and the integration is executed in two ways, as post-processing (the Post-Shape model) or by injecting within the network (the BSPPN model). The other, the R-PolyGCN is to utilize the graph model as representations of the the non-grid structured polygons and employ the Graph Convolutional Network (GCN) to learn the geometric shapes. Then the output footprint polygons are compared with the ground truth through the specially designed loss functions. The losses are computed and flow back to update the deep learning models for the network optimization. The whole procedure is a regular training phase of the supervised learning. The optimized models can be applied to the new dataset (testing data) for new building footprint extraction tasks.

1.3.2 Contributions

As mentioned before, we identify the key problems and provide the general deep learning framework for the tasks of the building footprint extraction with boundary regularization. In summary, the contributions of this study are as follows:

- we develop an unified deep learning framework to discover the spatial, semantic and geometric information from the satellite images to specifically extract building footprints with regularized boundaries. The models are combinations of the instance segmentation and geometric learning, which are able to simultaneously recognize and localize multiple building objects, assign pixel-wise semantic labels and predict polygons of geometric shapes.
- The polygon shape-prior is exploited by being integrated into the deep neural net-

works as the guidance to produce building footprints with more geometric properties, thus preserving more regularized boundaries. Two integration methods are proposed. The Post-shape model integrate shape-prior at the post-processing stage while the BSPPN model inject the shape-prior within the networks by a specially designed polygon-region pooling layer.

- The conventional grid-based pixel representation is replaced with the non-grid graph model to more geometrically represent the building polygons, which benefits the regularization of building boundaries. To utilize the graph features, the graph convolutional network is employed and combined with the instance segmentation model to form our R-PolyGCN model.

- The BSPPN and R-PolyGCN models are both able to be trained end-to-end for multi-task losses thanks to our specially designed training strategies.

- Extensive experiments and analyses are done to evaluate model performances.

1.4 Thesis Organization

This thesis is organized in 7 chapters. Chapter 1 introduces the background and motivation of our study and clarifies the problem domain and the research objectives. Chapter 2 reviews relevant research work as well as open datasets for building footprint extraction. In chapter 3, the first part of our framework, the backbone network is presented. Chapter 4 and chapter 5 give detailed illustrations of our two types of building extraction networks, the shape-prior integration networks and R-PolyGCN. Chapter 6 describes the adopted dataset, the experiments and the evaluation results. Finally, in chapter 7, we draw the conclusion of this study and specify the future works.

Chapter 2

Related Work

In this chapter, the research work related to our study will be introduced. We will divide the related work into four sections including the general deep learning models for object detection and segmentation (section 2.1), solutions to the problems of building footprint extraction (section 2.2) and building boundary regularization (section 2.3) and the introduction to the relevant open datasets (section 2.4).

2.1 Deep Learning for Object Detection and Segmentation

Deep learning is the major techniques we will apply to build our models to solve the computer vision problems. Hence, the basic concepts of deep learning for computer vision and the classic and current deep learning models for the object detection and segmentation are briefly reviewed in this section.

2.1.1 Basic Deep Learning for Computer Vision

Basic deep learning models adopt a multi-layer structure to learn the data representation or features with multiple levels of abstraction [4]. The structure is referred as deep neural networks. Compared to the traditional techniques that use hand-crafted features, deep neural networks utilize a complicated combination of linear and non-linear operations to form a layer-by-layer connected architecture to automatically encode deep features from input data. On the top of the features, different frameworks can be employed to produce outcomes for specific tasks. The procedure from input data to outcomes is referred as the feed forward of the networks. At the layer n of the network, the feed forward can be defined as:

$$\begin{aligned} z^n &= w^n a^{n-1} + b^n \\ a^n &= \sigma(z^n) \end{aligned} \tag{2.1}$$

where a^{n-1} and z^n denote the input and output of the current layer n; w^n and b^n denote the weight and bias parameters of the layer, which apply a linear operation of the input; σ denotes the activation function, which usually has nonlinear attributes. In the supervised learning scenario, the network outcomes are compared with the ground truth through a objective function, which is typically referred as the loss function and calculate the distance between the network outcomes and the ground truth. In order to make the outcomes closer to the ground truth, the networks are then optimized to minimize the loss function. In detail, the gradients of the loss over the network parameters at the last layer are first computed and then propagated back through the whole networks based on the chain rules. The network parameters are adjusted by their gradients and gradually reach the optimal in a training loop. The optimization approach is referred as gradient descent and the whole procedure is called back-propagation [5]. The gradient calculation and the parameter update of the layer n at the training step i can be formulated as:

$$\begin{aligned} \nabla L_i(\theta_i^n) &= \frac{\delta L_i}{\delta \theta_i^n} = \frac{\delta L_i}{\delta z_i^n}\frac{\delta z_i^n}{\delta \theta_i^n} \\ \theta_{i+1}^n &= \theta_i^n - \eta \nabla L_i(\theta_i^n) \end{aligned} \tag{2.2}$$

where L_i is the loss at step i and θ_i^n is the network parameters of the layer n; $\nabla L_i(\theta_i^n)$ denotes the gradient of the loss with regard to the parameters; the parameters θ_{i+1}^n at the next step are updated with a learning rate η.

For computer vision tasks, a special form of neural networks, the Convolutional Neural Networks (CNN) are commonly employed, whose major components are the convolution layers and pooling layers. A classic CNN architecture is LeNet-5 [6] displayed in the figure 2.1 and we use it as an example for the structure of CNN. LeNet-5

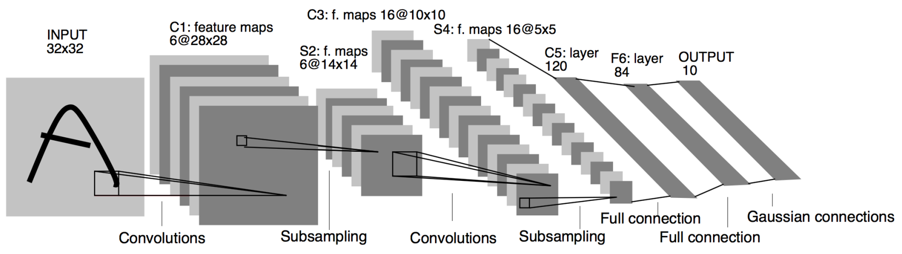


Figure 2.1: LeNet-5: a typical CNN architecture.

was proposed to recognize and classify the images of the hand-writing characters. It has 7 layers in a hierarchy structure including 3 convolution layers (C1, C3 and C5), 2 sub-sampling (pooling) layers (S2 and S4) and 1 fully connected layer. The convolution layers adopt 5×5 convolution filters with attributes of local connection and weight sharing. The outputs of each layer are generally referred as features or feature maps. At each convolution operation, the convolution filters are applied only at fixed-size of connected regions because of the strong correlation of the local values of the image or the feature map. The same convolution filters are operated throughout the whole image or feature map, which allows the different parts share the same weights and detect the same visual patterns. The pooling layers merge the 4×4 regions and take the averages, which can combine the locally similar semantics to generate more abstract high-level features. The fully connected layer is applied to generate predictions from the feature maps.

2.1.2 Models for Object Detection

Object detection is a heated topic in deep learning and computer vision community and it is also our major concern in this study. The deep learning models for object detection can be categorized into two groups: two-stage and one-stage models.

Two-stage models: these models generally utilize a two-stage detection pipeline: the region proposal generation and the refined localization and classification, which can be considered as a coarse-to-fine scheme. The Region-based Convolutional Neural Network (R-CNN) is the meta-model for the two-stage models. The R-CNN model first came up in the paper [7], which adopts a search algorithm to produce about 2000 region proposals from the image and fed these regions into CNN to extract features. Then the support vector machine (SVM) [8] is used to classify the regions and predict bounding boxes based on the extracted features. Fast R-CNN [9] inputs the image into CNN first to extract a feature map and crop the region proposals with the feature map to generate the region of interest (RoI) features. For localization and classification, Fast R-CNN employs the fully connected layer and the softmax function [10]. Compared to R-CNN, Fast R-CNN applies CNN over the whole image once thus getting rid of feeding 2000 regions into CNN every time. However, both of them need a pre-processing step to generate the region proposals, which is a selective search algorithm and costs too much time. To address the problem, Faster R-CNN [11] abandons the search algorithm and designs a Region Proposal Network (RPN) to generate proposals from pre-defined anchor boxes. The RPN utilizes the power of the convolution neural network to let the region proposals directly be learned from the networks, thus speeding up the whole process and improving the detection accuracy.

One-stage models: These models skip the stage of the region proposal generation and directly apply the one-shot detection over densely sampling possible locations of the input image. The advantage of the one-stage models over the two stage ones is that they are more efficient because of their simplified and unified network design. YOLO [12] splits the image into fixed-size of grids, on each of which the CNN is applied to predict the bounding boxes and class probabilities. SSD [13] additionally employs a series of convolution layers with decreasing sizes to extract the pyramid of multi-scale features, on which objects of different sizes can be detected. Recent works like [14, 15] accomplish the object detection by directly using CNN to detect representative key points of objects like corner points or center points, from which the bounding box predictions can be produced.

2.1.3 Models for Image Segmentation

Image segmentation is the core issue in our study and huge amount of efforts have also been invested in this area by the deep learning researchers. Generally there are two types of segmentation, semantic segmentation and instance segmentation, whose differences are shown in the figure 2.2. The semantic segmentation is to assign the class labels to every pixel of the image (in the middle part of the figure, image pixels are labeled according to their categories) while the instance segmentation can also produce

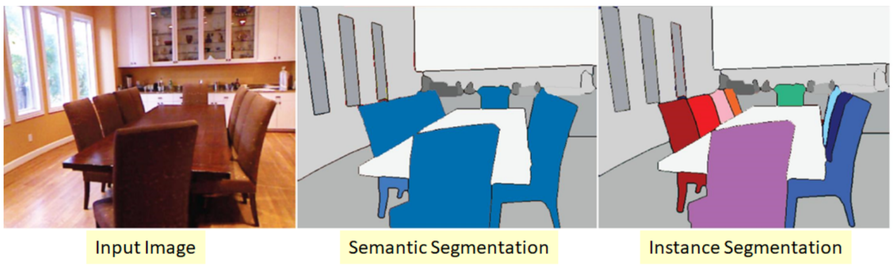


Figure 2.2: Semantic segmentation and instance segmentation[1].

semantic pixel-wise labels but additionally predict instance-aware labels, that is, distinguish the individual objects (different chairs are separated using labels of different colors in the right part of the figure).

Semantic segmentation models: Fully Convolutional Network (FCN) [16] is a typical deep learning model for the semantic segmentation, which is illustrated in the figure 2.3. The input image is fed into the common convolution layers and pooling

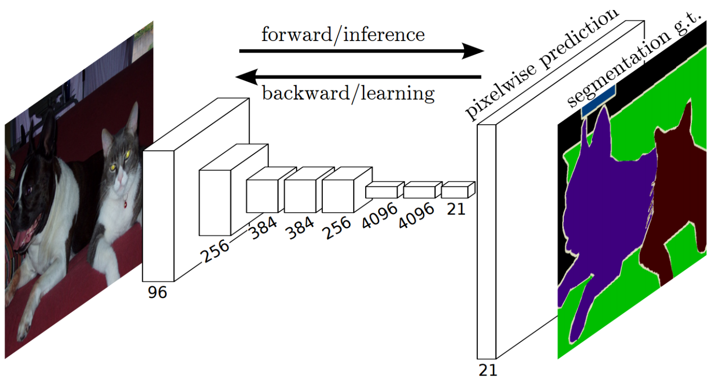


Figure 2.3: Architecture of FCN for semantic segmentation. The figure copyright is owned by [16].

layers, at the end of which the down-sampled features are obtained. Here FCN learns a deconvolution layers to up-sample the feature map to the resolution of the image size. On the up-sampled feature map, pixel-wise prediction is applied to produce semantic segmentation results. On the basis of FCN architecture, SegNet [17] and U-Net [18]

[1]https://towardsdatascience.com/review-deepmask-instance-segmentation-30327a072339

design a encoder-decoder structure where the decoder part is utilized to more precisely learn the up-sampling of the feature map. To do so, SegNet passes the pooling indexes from the encoder to the the decoder. U-Net transfers the features from the encoder to those of the decoder to form a combination of features from different scales, allowing it to be more capable of detecting small objects and segmenting images with dense number of objects.

Instance segmentation models: Two approaches are generally adopted to achieve instance segmentation. One is to first perform semantic segmentation over the image and then apply instantiation by grouping connected pixels to identify individual objects. This pipeline is utilized by DeepMask [19] and SharpMask [20]. The other approach is put forward by Mask R-CNN [21], whose architecture is presented in the figure 2.4. In short, Mask R-CNN first performs instantiation and then segmentation. The object

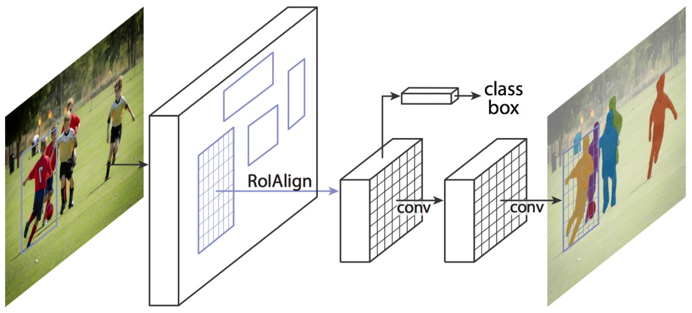


Figure 2.4: Architecture of Mask R-CNN for instance segmentation. The figure copyright is owned by [21].

detection network (almost same with Faster R-CNN) is employed to distinguish and localize objects. The detection part can also generate well-localized RoI features, over which the semantic segmentation model FCN is applied to obtain object masks. The whole network has an end-to-end unified design. The segmentation accuracy of Mask R-CNN surpasses the models adopting the first approach on most of the benchmarks.

2.2 Building Footprint Extraction

Deep learning has been aggressively utilized for the building footprint extraction and has outperformed many traditional methods. Most of the research works simply apply the existing CNN models or their variants, for instance, the segmentation models introduced in the subsection 2.1.3. Hence, we divide the related works into semantic segmentation models and instance segmentation models.

Semantic segmentation models: Early works [22] trained a basic CNN for building labeling, which only contains three layers including one convolution layer, one pooling layer and one fully connected layer. It shows competitive results compared to other complicated traditional algorithms but the simple CNN is quite sensitive to the hype-parameter setting. More recent works employ more complex CNN models. [23] designs a multi-layer perceptron (MLP) structure, which has a skip connection similar to the U-Net to combine features of different scales. The SegNet is directly used by [24] to train an additional loss representing the distance to the building boundary apart from the pixel-wise classification loss. [25] utilizes the U-Net as the basic model with multiple constraints, which restrict the outputs from feature maps of different scales to be compared with the ground truth images of corresponding scales. Other works adopt the data-fusion idea to boost the segmentation performance and still use the semantic segmentation models to deal with data of multi-sources. LiDAR point clouds and images are combined in [26] through a U-Net model. [27,28] employ the U-Net architecture to utilize the satellite images and GIS maps like OpenStreetMap, Google Map and so on to take advantage of the more precise vectorized maps. Digital Surface Model (DSM) serves as auxiliary data to the image data in [29] and a FCN model is adopted. Besides, the generative models, Generative Adversarial Networks (GAN) are gaining more and more interests and are recently applied to the semantic segmentation for the building footprint extraction. [30] designs a matching-GAN architecture, which modifies the basic GAN model by using a matching network as the alternative of the discriminator and successfully applies the matching GAN to semantic segmentation tasks. [31] employs the U-Net enhanced by the GAN model with spatial and channel attention mechanisms to produce more discriminative prediction maps and tackle segmentation ambiguities.

Instance segmentation models: The application of the Mask R-CNN is explored in [32] for the building extraction problem and achieves a satisfying instance segmentation performance. [33] further improves the Mask R-CNN model by introducing the rotational bounding boxes to enhance detection quality and stacking the receptive field blocks to handle scale viability issues.

We find that the semantic segmentation models are fairly popular for the research of building footprint extraction, especially the U-Net, which are able to recognize small buildings. By contrast, the instance segmentation models are not fully explored and still have huge potential to provide better solutions because the buildings in the large-scale images are usually closely situated or connected (especially in the urban areas) and the instance segmentation models are able to properly separate them.

2.3 Building Boundary Regularization and Geometric Learning

The building boundary regularization is one of the center problems of our study, which is typically associated with geometric learning of polygon shapes. Therefore, in this section we will introduce several different solutions to the problem and many of them adopt the geometric learning.

Traditional methods: Before the deep learning era, the building footprint extraction relies more on the processing of the LiDAR point clouds than the images because the point clouds hold the spatial locations of the points, which are more geometrically meaningful. [2, 3] mainly use point cloud data and adopt a Binary Space Partitioning (BSP) process and a Minimum Description Length (MDL) based algorithm to generate and optimize the building polygon shapes for building footprint detection and boundary regularization. The papers also accomplish the 3D building roof reconstruction. [34] also employs similar shape optimization algorithms to extract regularized building polygons from point clouds.

Image segmentation methods: Many segmentation models are specially designed to attach attentions to the building boundaries or to learn more geometric information. To fully exploit the boundary information, [35] feeds the fusion of the images and Digital Elevation Model (DEM) to the SegNet model combined with extra edge and boundary predictions produced from the FCN model and adopt a multi-task learning (ensemble learning in the paper) strategy. [24,36,37] also utilize the multi-task learning scheme to train additional boundary loss on FCN or U-Net models, among which [36, 37] claim that they can produce building boundaries with regularities. Moreover, some conventional polygonal models such as the active contour (ACM) or snake model [38] are recycled in the modern CNN architectures. The deep structured active contours (DSAC) [39] and the deep active ray network (DARNet) [40] both integrate the ACM model into their segmentation networks to learn richer geometric information to better predict the polygon contours and delineate the building boundaries. Nevertheless, the segmentation models still produce pixel-wise building polygons, which are unlikely to output building polygons with perfectly smooth and regularized boundaries.

Polygon learning methods: There are also deep learning models trying to directly generate polygons instead of pixel-wise segmentation maps. Many of them do so by producing the optimal locations of the polygon vertices and linking the predicted vertices with straight lines, which will intuitively produce polygons with regularized boundaries. Simple works like [41] uses a fully connected layer to predict the coordinates of the vertices, which falls short of the involvement of any geometric information. PolyRNN [42] and PolyRNN++ [43] employ the recurrent neural networks (RNN) to predict the locations of polygon vertices in sequence, that is, the current vertex prediction is influenced by the previous predictions. These two models are applied for semi-automatic annotation with bounding boxes provided, thus failing to produce object detection results in their frameworks. [44] borrows the ideas of PolyRNN and Mask R-CNN to build a unified pipeline to accomplish object detection and sequential polygon vertex prediction and applies the framework on large-scale image datasets to extract building footprints and road lines. CurveGCN [45] explores the usage of the graph convolutional networks (GCN) to produce polygons as a graph representation, which is much more efficient and utilize more geometric features than RNN models. However, like PolyRNN and PolyRNN++, CurveGCN is also used for annotation tasks and is unable to perform object detection.

Shape-prior or shape-primitive learning methods: Shape-prior or shape-primitive are essential to learn geometric shapes like polygons. Here we investigate some state-of-the-art deep learning models integrated with them, even though many of the models are not aimed to predict polygon shapes or extract building footprints but they are rather promising to be applied to produce polygons with better qualities and improve the building boundary regularization. [46, 47] both design a super-pixel pooling network (SPPN) to combine the contexts of super-pixels into their models. Shape priors with convolutional neural networks (SP-CNN) and tunable SP-CNN (TSP-CNN) [48] create the shape-prior templates and design a learnable shape-prior layer to guide the networks to learn shape-prior information more appropriately to detect cell nucleus with diverse shapes. ShapeMask [49] also creates a collection of shape-prior templates by clustering the masks in the ground truth and the shape-prior is utilized to refine the coarse mask prediction. The whole pipeline is incorporated into an unified instance segmentation model. 3D shape-primitive RNN (3D-PRNN) [50] builds a generative recurrent neural network to learn 3D primitive representations of objects. DeepPrimitive [51] proposes a novel framework to learn the 2D shape-primitives and predict the sequences and relations of the shape-primitives to represent a complete object.

2.4 Open Datasets for Building Footprint Extraction

To evaluate the models for the building footprint extraction, various open datasets are publicly available. Generally the datasets comprise aerial or satellite images and building footprints annotated with pixel-wise labels or object-wised labels, which refers to the coordinates of building polygons at object-level. The details of some state-of-the-art datasets are presented here.

- **Massachusetts datasets** [22] cover about 340 km^2 of the City of Boston with 151 RGB aerial images. The image size is 1500×1500 pixels for an area of 2.25 km^2 and the image resolution is 100 cm, which is relatively lower than other datasets. The annotations were made from OpenStreetMap[2]and rasterized into binary images with building footprints as the foreground, thus indicating that they are pixel-wise labels.

- **ISPRS benchmark on urban object classification and 3D reconstruction** [52] provides aerial imagery for two cities, Vaihingen and Potsdam with 38 and 33

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

image patches respectively. The Vaihingen images have a size of 6000×6000 pixels and a high resolution of $5cm$ while the Potsdam images are of 2500×2500 pixels and $9cm$ resolution. All the images are 4-band of IRGB (near infrared, red, green and blue) and are labeled at pixel-level for the landcover classification task with six classes including impervious surfaces, building, low vegetation, tree, car and clutter. The benchmark also contains corresponding digital elevation model (DEM) data and the Point Cloud (PC) data of Vaihingen and Toronto for the 3D building roof reconstruction task.

- **Inria aerial image labeling benchmark** [53] has satellite images for 10 cities in Austria and USA. Each region has a area of about $81km^2$ covered by 36 image tiles with the size of 1500×1500 pixels, the resolution of $30cm$ and RGB bands. The images are pixel-wise labeled into building and non-building classes.

- **Toronto city benchmark** [54] covers about $712.5km^2$, a large region of the greater Toronto area (GTA) with aerial images of 10 cm resolution. The annotations consist of around 400,000 building footprints and 8,439km of road. The building annotations include both pixel-wise labels and object-wised vectorized polygons. Corresponding point cloud data are also provided.

- **AIRS (Aerial Imagery for Roof Segmentation) dataset** [55] covers about $457km^2$ of land of the city of Christchurch, New Zealand with aerial images of $7.5cm$ resolution and RGB bands. The annotations consist of over 220,000 buildings with both pixel-wise and object-wised labels. In addition, the building roof outlines are specially refined and aligned.

- **SpaceNet building dataset** [56] contains 24,586 satellite images covering a total area of around $3011km^2$ for four cities, Las Vegas, Paris, Shanghai, and Khartoum which are from four different continents with the coverage of both rural and urban regions to bring about fairly high diversity of building roof styles. Various types of images are provided, including single-band panchromatic, RGB images and 8-band multi-spectral images with pan-sharpened and unpan-sharpened versions. The RGB images have the size of 650×650 pixels and resolution of 30*cm* and each image covers $200m \times 200m$ area. 302,701 building footprints are annotated at object-level.

The Statistics of the datasets are summarized in the table 2.1. For the building footprint extraction task, all the datasets mainly provide aerial images of RGB or multi-spectral and some contain REM or point cloud as auxiliary data. The image resolution ranges from several centimeters to one meter. The building footprints have two types of annotations, including pixel-wise labels which refer to binary images with the pixels

Table 2.1: Statistics of state-of-the-art datasets for building footprint extraction

Dataset	**Massachusetts**	**ISPRS**	**Inria**	**Toronto**	**AIRS**	**SpaceNet**
Location	Boston	Vaihingen/ Potsdam	10 cities in Austria/ US	GTA	Christchurch	Las Vegas, Paris, Shanghai, Khartoum
Coverage(km^2)	340	1.4/3.4	810	712.5	457	3011
Data Type	RGB	IRGB+DEM +PC	RGB	RGB+PC	RGB	single-band+ RGB+8-band
Image Amount	-	38/33	360	-	-	24586
Image Resolution (cm/pixel)	100	5/9	30	10	7.5	30
Image Size (pixels)	1500×1500	6000×6000/ 1500×1500	1500×1500	-	-	650×650
Annotation Amount (Building Polygons)	-	-	-	>400,000	>220,000	302,701
Pixel-wise Labels	yes	yes	yes	yes	yes	yes
Object-wised Labels	no	no	no	yes	yes	yes

of buildings categorized as foreground and object-wised labels, which are individual building polygons with the coordinates of the polygon vertices. The pixel-wise images are equipped with only semantic information while the object-wised polygons are able to provide additional geometric attributes, which are essential to learn the polygon shapes of the buildings. Furthermore, note that the object-wised labels can be conveniently converted into the pixel-wise ones and the conversion process is irreversible, which suggests that the object-wised labels are of potential for much wider range of applications. Therefore, only the dataset with object-wised labels are considered, including Toronto city benchmark, AIRS dataset and SpaceNet building dataset. In view of the data diversity, we select SpaceNet dataset for our experiments because it covers four cities with various building roof outlooks.

2.5 Summary

In this chapter, we present the related work. First, the basic concepts of deep learning and CNN is introduced. Then we illustrate the object detection and image segmentation models, which will be frequently adopted in our study. Next, the semantic segmentation and instance segmentation models for the building footprint extraction are demonstrated, where we observe that instance segmentation models can be further utilized for our applications. Lastly, solutions to the building boundary regularization are reviewed. Most of the works employ geometric learning to predict building polygons with regularized boundaries. The models integrated with shape-prior or shape-primitive are introduced. All of the research provide inspirations to our novel models.

Chapter 3

Backbone Network

In this chapter, we will introduce the backbone network. In deep learning models, the backbone network like VGG-16 [57] is usually used to extract features from input data. Our backbone network is also designed as a feature encoder. Besides, it also has the function of object detection and localization with spatial and semantic learning. Eventually, the backbone network will output well-localized region of interest features, which will play a significant role in our following models in next chapters.

3.1 Overview

The backbone network is designed for feature encoding and building object detection and localization. We utilize a combination of Residual Network (ResNet) [58] and Feature Pyramid Network (FPN) [59] to extract deep features at multiple scales. To detect and localize building objects, a two-stage object detection model is employed including the Region Proposal Network (RPN) [11] and a localization layer, including bounding box regression and classification layers. Besides, to obtain well-localized

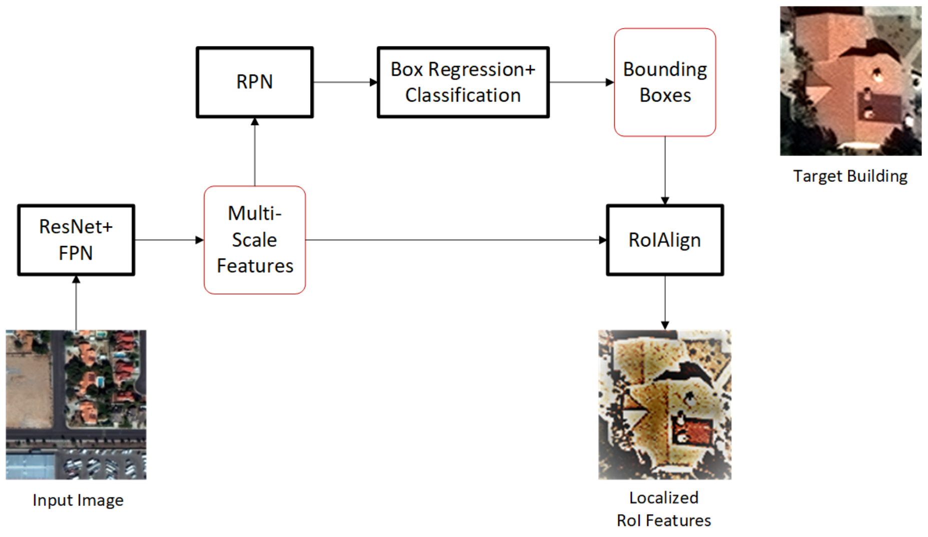


Figure 3.1: Backbone Network

RoI features for each target building object, we also apply a RoIAlign layer [21] to precisely crop the bounding boxes with the feature map. The localized RoI features will be essential in the future tasks like pixel-wise segmentation or geometric shape learning. The whole structure of the backbone network is illustrated as figure 3.1.

3.2 Feature Encoding

Figure 3.2 shows the details of the feature encoding network that we are using. It's divided into the bottom-up and the top-bottom pathways. At the bottom-up stage, the image is input into a five-stage (C1 - C5) ResNet, where each stage of ResNet consists of several convolution layers and applies 2×2 pooling at the last layer to downsample the feature map to half size. The residual blocks are adopted in the convolution modules of the bottom-up part to make the networks deeper and the features better. The top-down part of the network integrates features from different scales generated from the bottom-up part. It first applies a 1×1 convolution kernel to the current feature

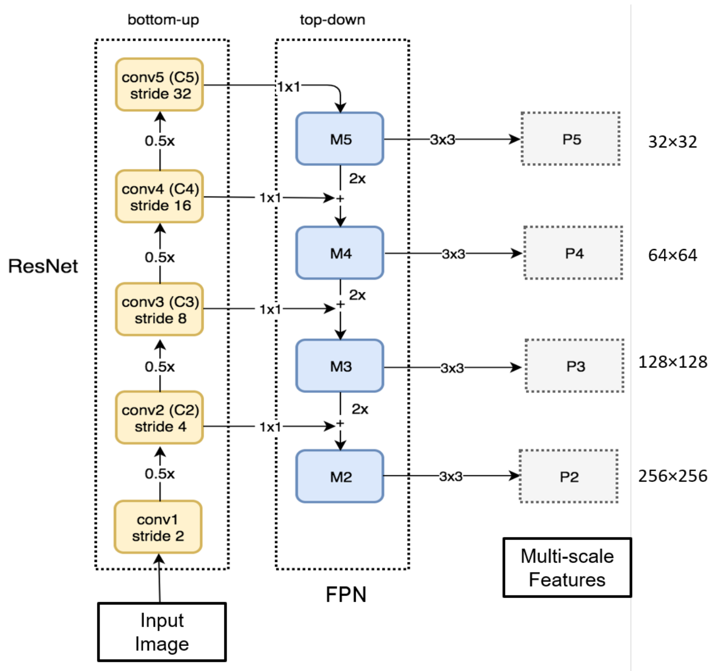


Figure 3.2: Multi-Scale Deep Feature Encoding[1]

map and add it to its upsampled previous feature map element-wise. To reduce the distortion effect of upsampling, it also learns a 3×3 convolution kernel to output the feature maps, which have different scales: P5 (32×32), P4 (64×64), P3 (128×128), P2 (256×256). The numbers 1-5 here represent the levels of scales of the feature maps. Detecting objects at different scales is an essential task since our input satellite images cover lager area of lands and contain many building objects with various sizes. The multi-scale feature maps obtained from the feature encoding network are capable of recognizing building objects from different scales compared to those using feature maps of only one scale.

On the top of the multi-scale feature maps, we utilize a two-stage object detection network (figure 3.3) including a RPN layer, a bounding box regression and classification layer and a RoIAlign layer, which will be introduced in the following subsections.

[1]`https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn`

These three components are connected as our localization network to produce precisely localized RoI features.

3.3 Localization

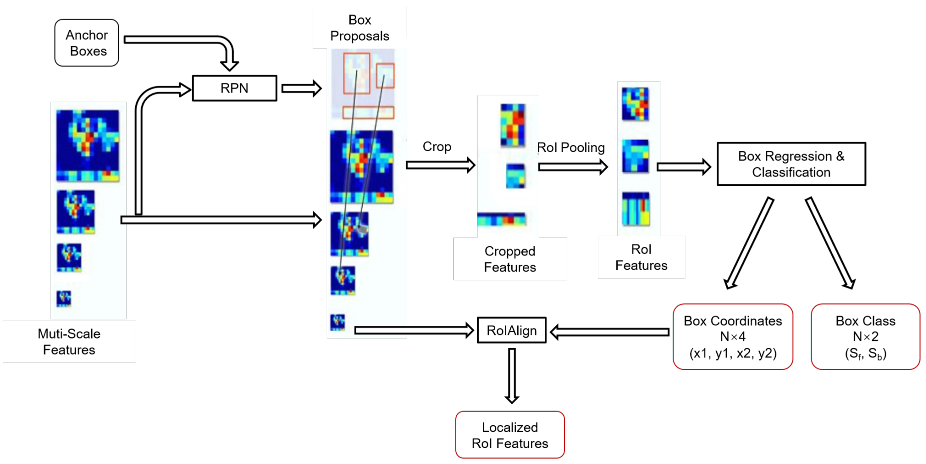


Figure 3.3: Localization Network[2]. In the end of the flowchart, N means the number of the bounding boxes detected; the box coordinates (x1, y1, x2, y2) refer to the coordinates of the top-left and bottom-right corner points of the bounding boxes; the box class (S_f, S_b) denotes the class scores for the labels of foreground (building) and background (non-building).

The RPN first takes the features and pre-defined anchor boxes to generate the initial proposed bounding boxes, which are used to crop with the feature maps to get the cropped features. The RoI pooling is then operated on the cropped features to obtain RoI features, which are fed into the box regression and classification layer to produce the coordinates and class scores of the refined bounding boxes. Lastly, the multi-scale feature maps and the final bounding boxes are input into the RoIAlign layer to generate precisely localized RoI features.

[2]`https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn`

3.3.1 Region Proposal Network

Figure 3.4 shows the structure of the region proposal network (RPN). The inputs of RPN are the multi-scale feature maps and pre-defined anchor boxes. For different strides on the input image, we generate anchor boxes with different sizes and width-to-height ratios. In our network, the stride values are assigned as $\{4, 8, 16, 32, 64\}$

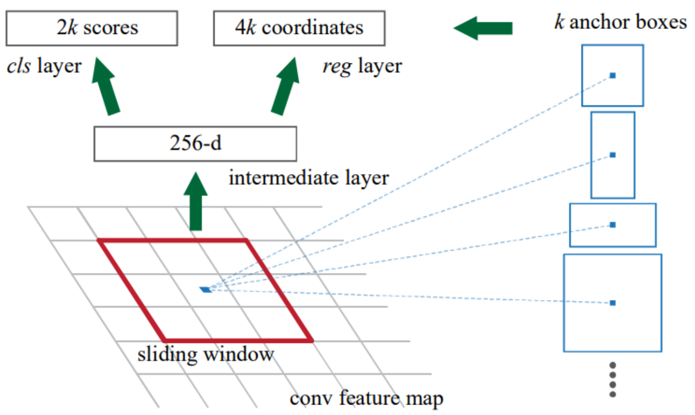


Figure 3.4: Region Proposal Network. The image copyright is owned by [11].

and anchor box sizes and shapes are set as $\{32, 64, 128, 256, 512\}$ and $\{0.5, 1, 2\}$. The anchor shape here refer to the ratio of the width to the height of the box. In this way, we can obtain 261888 anchor boxes and each anchor box corresponds to an unique entry of the multi-scale feature maps. In the region proposal network, on each entry of the feature maps, it will predict bounding box proposals for the corresponding anchor boxes through a box regression layer and class scores through a classification layer. The classification layer here is designed to distinguish positive and negative boxes, which means the boxes contain objects or not. So the class scores of RPN are objectness score. The bounding box proposals are then filtered by a Non-Maximum Suppression (NMS) based on a threshold on the objectness score to reduce its total numbers and to maintain the ratio of the positive and negative proposed boxes (usually 3:1). The box proposals are more close to object locations in the image compared to the input anchor boxes but still need further improvement, which will be done in subsection 3.3.2.

3.3.2 Localization Layer

The multi-scale feature maps are cropped using the box proposals from RPN. In the cropping process, we pick the feature maps to crop based on the size of the box proposals according to the equation 3.1, following [59].

$$k = [k_0 + log_2(\sqrt{wh}/224)] \tag{3.1}$$

where w, h are the wdith and height of the box proposal; $k_0 = 4$ and k is the level of scale (defined in 3.2) we select as the feature map to crop. Since the cropped features have various sizes, we feed them into a RoI pooling layer to normalize them into RoI features with the fixed size (14×14). The cropping operation here involves a feature scale selection, thus allowing the feature scales to match the size of the detected objects, which means feature maps with larger resolution correspond to smaller objects and those with smaller resolution correspond to bigger objects. Finding these correspondence takes advantage of the multi-scale feature maps and can exploit richer and more accurate semantic information from different scales. Then like RPN, in the localization layer, the box regression and classification layer is applied to the RoI features to predict the coordinates and class scores of the bounding boxes. This second-time localization can further refine the bounding box proposals generated from RPN. Note that the class scores here are estimated probabilities for each class (building and non-building).

3.3.3 RoIAlign

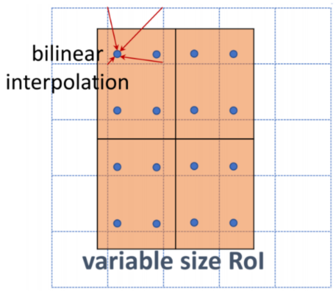


Figure 3.5: RoIAlign. The image copyright is owned by [21].

The coordinates of the RoI features are usually floating numbers produced from the box regression layer while the cropping and RoI pooling (in subsection 3.3.2) will simply convert them into integer numbers in a quantization process, thus causing rounding errors and misalignment. Mask R-CNN [21] investigates that simple cropping and RoI pooling are not sufficient to get precisely localized feature maps for pixel-wise segmentation, which requires pixel-level accuracy. Therefore, it designs a RoIAlign layer (in figure 3.5) to tackle the misalignment problem and improve the accuracy of the RoI features. Instead of directly taking the integer of the floating RoI coordinates with roundoff errors in the cropping and RoI pooling, RoIAlign reserve the floating coordinates and use the differentiable bilinear interpolation to get the values of floating points and the final localized RoI features.

3.4 Loss Design

For the whole backbone network, we need to calculate losses for the object detection of two stages, RPN and box regression and classification. The loss functions for both stages deal with two types of losses, box regression loss and classification score loss, thus forming a multi-task training scenario.

Following Faster R-CNN [11], the box deltas are calculated as inputs of the box regression loss function rather than box coordinates. The deltas are defined as following:

$$
\begin{aligned}
&t_x = (x - x_a)/w_a, t_y = (y - y_a)/h_a, t_w = log(w/w_a), t_h = log(h/h_a), \\
&t_x^* = (x^* - x_a)/w_a, t_y^* = (y^* - y_a)/h_a, t_w^* = log(w^*/w_a), t_h^* = log(h^*/h_a)
\end{aligned}
\tag{3.2}
$$

In the equation of box regression, x, y, w and h represent the coordinates of the center point of the bounding box and its width and height. Respectively, x, x_a, x^* are for the predicted box, anchor box and ground truth box (likewise for y, w, h). Our backbone network will predict the box deltas (t_x, t_y, t_w, t_h), which are equivalent to the regression from an pre-defined anchor box to a predicted box. To compare the predicted box deltas with the ground truth ones $(t_x^*, t_y^*, t_w^*, t_h^*)$, which represent the regression values from an anchor box to its closest ground truth box, we adopt the smooth L_1 loss:

$$
L_{reg}(t, t^*) = \sum_{i \in x,y,w,h} smooth_{L_1}(t_i - t_i^*) \tag{3.3}
$$

where a $smooth_{L_1}$ loss [60] is used here:

$$
smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if} |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{3.4}
$$

As for the classification score loss, we compute a binary cross entropy loss since it's a binary classification problem:

$$
L_{cls}(p(y)) = -(ylog(p(y)) + (1 - y)log(1 - p(y))) \tag{3.5}
$$

in which y is the class label predicted (0 or 1) and $p(y)$ is the probability score for the class label.

When training RPN, the anchor boxes are assigned class labels as positive or negative, which we refer as objectness scores. The box proposals are also generated as a coarse localization results. Therefore, based on equations 3.3 and 3.5, we define RPN loss as:

$$L_{rpn}(p^{obj}, t^{rpn}) = \frac{1}{N_{cls}} \sum_{i=1}^{N_{cls}} L_{cls}(p_i^{obj}(y)) + \frac{1}{N_{box}} \sum_{i=1}^{N_{box}} L_{reg}(t_i^{rpn}, t_i^*) \tag{3.6}$$

where p^{obj} and t^{rpn} are the objectness scores and proposed box deltas by RPN and N_{cls} and N_{box} are the number of all the boxes (both positive and negative) after NMS and the number of the positive boxes. Only box deltas of positive ones are used for box regression since negative boxes containing no objects have no contributions here. Sometimes we need to balance the classification loss and box regression loss with weight coefficients. However, according to [11], different coefficients make little difference to the performance. So here we simply add the two losses without using any coefficients.

The localization layer classifies the boxes into two classes, building and non-building and produces final location predictions of the bounding boxes. For these two tasks, we compute a localization loss and a similar loss function design is adopted:

$$L_{loc}(p^{class}, t^{loc}) = \frac{1}{N_{cls}} \sum_{i=1}^{N_{cls}} L_{cls}(p_i^{class}(y)) + \frac{1}{N_{box}} \sum_{i=1}^{N_{box}} L_{reg}(t_i^{loc}, t_i^*) \tag{3.7}$$

where p^{class} and t^{loc} are the class scores and the predicted bounding box deltas and N_{cls} and N_{box} are the number of all the boxes (both positive and negative) from RPN and the number of the positive boxes. Since these losses will be trained with others in the following chapters, the training strategies will be discussed later.

3.5 Summary

In this chapter, the backbone network is introduced, which is utilized to extract multi-scale features and classify and localize the target objects (buildings) from input images. It is also designed to obtain well-localized RoI features, which are the fundamentals of the framework in the next chapters. Overall, our backbone network is a typical two-stage object detection network, which employs a coarse-to-fine pipeline. To train the backbone network, a two-stage and multi-task loss function is presented.

Chapter 4

Shape-prior Integration Network

In this chapter, we will propose our first building extraction network following the backbone network to address the problems of building extraction and boundary regularization, an integration model of polygon shape-prior and deep neural networks. Two ways of integration will be introduced.

4.1 Overview

On the top of the backbone network from chapter 3, we integrate shape-priors into the model as the guidance to discover and learn polygon geometric information, specifically aiming to obtain more regularized building boundaries. The shape-prior integration is done in separately two ways: one is to apply a Minimum Description Length (MDL) based model to iteratively optimize the shape of the building polygons at the post-processing stage (outside of the neural network); the other is to inject the shape-prior within the network through a polygon region pooling layer, thus making network trainable end-to-end. Both models are instance segmentation networks springing from the meta framework for object instance segmentation, Mask R-CNN [21].

4.2 Post-Shape

The first way to integrate shape-prior, referred as post shape, is to utilize a MDL based optimization as a post-processing step, which is a polygon shape optimization algorithm and can specially exploit the distinct building geometric information. Before that we combine the backbone network and a mask prediction network to design an instance segmentation network to generate pixel-wise segmentation masks of building regions.

4.2.1 Mask Generation

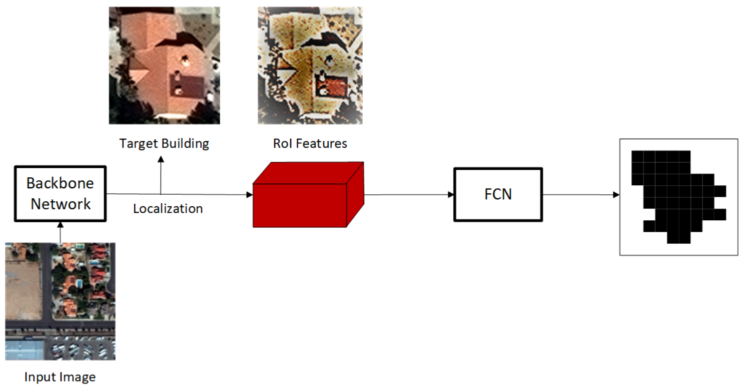


Figure 4.1: Mask Prediction. A combination of the backbone network and a FCN branch as the mask prediction network. The FCN is applied on the RoI features for each target building.

As shown in figure 4.1, we predict the pixel-wise mask of the target building from the localized RoI features extracted from the backbone network. The combination of the backbone network and the mask prediction network is a typical instance segmentation model. To complete the segmentation task for each target building, a FCN (Fully

Convolutional Network) branch (figure 4.2) is adopted on each RoI feature to predict the class probability of every pixel, that is to estimate whether it belongs to building class or non-building. The FCN branch consists of several convolution, batch normalization (Bn) and ReLU layers to learn the mask features and one deconvolution layer to increase the resolution. Lastly, a sigmoid activation layer is added to produce the pixel-wise mask logits, which can be used to compute the segmentation loss. The mask logits have 2 channels, representing the probabilities for two classes, background (non-building) and foreground (building). At the inference stage, a threshold 0.5 is used to obtain a binary image from the foreground mask logits so that we can get the pixel-wise segmentation mask for building and non-building regions.

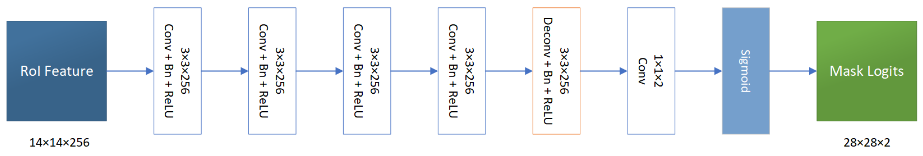


Figure 4.2: FCN Branch. The input RoI feature ($14 \times 14 \times 256$) is first fed into several 3×3 convolution layers, batch normalization and ReLU layers. The convolution layers here will not change the size or the depth of the input feature. Then a deconvolution layer is used to double the feature size to $28 \times 28 \times 256$. Two 1×1 convolution kernels are then applied to get a $28 \times 28 \times 2$ feature map, which is fed into the sigmoid activation layer to produce the mask logits.

4.2.2 Minimum Description Length based Post-processing

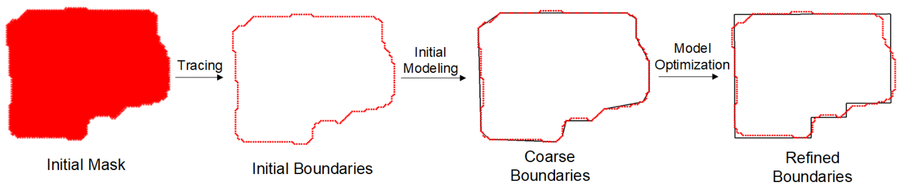


Figure 4.3: Post-processing for building boundary regularization. The initial mask representing the building region (the red region) is the output of the mask prediction network. By tracing the region border, the initial boundaries (red points) are obtained. Then we take a coarse-to-fine step to get the regular boundaries (black lines).

By tracing the border of a building regions, we can get the initial polygons for the buildings. These polygon vertices correspond to building boundaries and have normally an irregular shape. To create the building polygons with more regular boundaries, we utilize a MDL based optimization algorithm at the post-processing stage to convert the outcome of the network into regularized polygons of buildings. Inspired by our previous work [61], the boundary regularization process takes the following coarse-to-fine steps: initial modeling to get coarse boundaries and model optimization to refine the boundaries. The optimization consists of two steps: hypotheses generation and MDL optimization. The whole procedure is briefly illustrated in figure 4.3.

Initial modeling: The initial polygon points are first converted into simplified shaped polygons, by the Douglas-Peucker (DP) [62] algorithm . A set of representative line slopes are estimated based on the results of DP, with which the initial polygon is adjusted by applying weighted least-square adjustment method.

Hypothesis Generation: A triplet of vertices are selected (non-selective to the selection order) from the initial polygon, as described in figure 4.4. We label the triplet points as Anchor Point (AP), Floating Point (FP) and Guiding Point (GP) in a sequential order. Then, we generate two basis lines: Floating Line (FL), which is a set of AP and FP, and Guiding Line (GL), which is a set of GP and FP. A group of local hypothetical models are generated by moving FP along GL following the representative line directions estimated. We also allow the elimination of FP for hypothetical model generation. In this case, new FP and GP are selected by shifting the previously selected point triplet in a sequential order. Both clock-wise and counter-clockwise are selected to generate local model hypotheses for each point triplet.

MDL Optimization: MDL framework is selected for determining an optimal model hypothesis among the generated candidate models. The Description Length (DL) of a

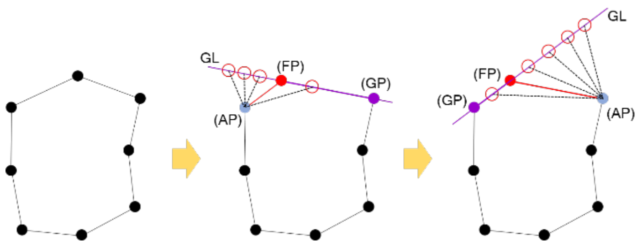


Figure 4.4: Hypotheses generation. The black nodes in the first figure are initial points as input and second figure is processing the first anchor points and the third one is processing the next anchor point. The red points represent the selection of hypotheses.

model in MDL framework is decomposed into two parts: (i) model closeness favoring low residuals between boundary points extracted by boundary tracing algorithm and hypothesized model; (ii) model complexity favoring simpler model with respect to the number of vertices, the number of representative line slopes and closeness to orthogonal angles. The detail of the MDL encoder adopted in this study is described in [2, 3]. The MDL optimization process is applied for determining the best model hypothesis locally over point triplet selected. Then, a globally optimized hypothesis is chosen by selecting a model to produce the minimum DL among all local optimum solutions. The same process is sequentially applied to all point triplets. The optimization step is able to produce building polygons which have the least localization errors as well

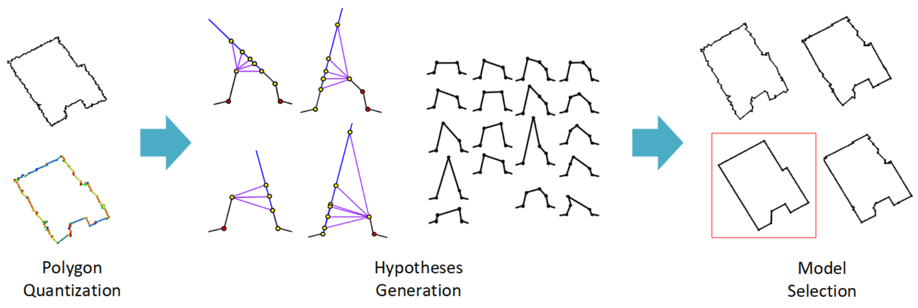


Figure 4.5: MDL based polygon optimization. The initial polygon is firstly quantified into vertex coordinates; in the figure we generate four different hypotheses and get four models to describe the polygon; then a global optimization based on the model description length is applied to select the model with the highest closeness and the lowest complexity. The polygon in the red square is selected here as the optimal outcome.

as the most simple shapes ans most regularized boundaries. The overall optimization procedure including the hypotheses generation is displayed in 4.5.

4.3 BSP Pooling Network

In addition to integrating the shape-prior outside of the network, we design the BSP Pooling Network (BSPPN) to inject the polygon shape-prior within the common instance segmentation network and train the network in an end-to-end fashion. We argue that integrating the shape-prior within the network has three major benefits: with additional polygon shape cues, the segmentation network can produce more polygon-like results and the boundaries can be more regularized; thanks to the deep neural network, the polygon geometric information can be automatically and implicitly learned and its expression complication and abstraction can be enhanced compared to those from hand-crafted algorithm; the MDL based post-processing is quite time-consuming and an end-to-end unified network has higher efficiency. We adopt a Binary Space Partitioning (BSP) process to extract the polygon shape-prior and design a polygon region pooling layer to integrate the shape-prior into the network. The incorporation of these components and the backbone network (in chapter 3) is our unified BSP pooling network (BSPPN).

4.3.1 Binary Space Partitioning Process

A typical BSP algorithm [63] can recursively subdivide the image space into hypothesis polygons with straight line segments. For our application, we borrow the idea of classical BSP and modify it to generate polygon regions. In detail, we firstly detect the edges of the input image and extract straight line segments from edges then extend the line to divide the image space into several polygon spaces. The whole process is illustrated as below:

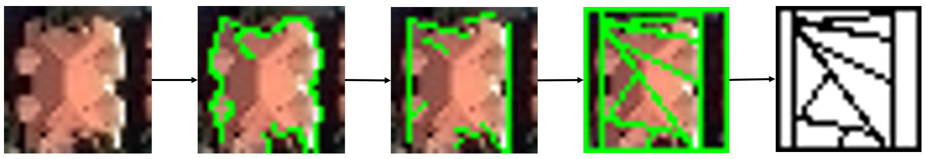


Figure 4.6: Our BSP process. From left to right are the input image, image edges, straight line segments, polygon regions and BSP map.

We argue that the polygon regions of BSP map represent certain shape prior and geometric features of the building polygon. The straight lines of the polygon regions can also help to smooth and regularize the building boundaries.

4.3.2 BSP Pooling Layer

The pooling layer in deep neural network typically operates the pooling function over $n \times n$ image grid regions. To integrate and utilize the BSP map as shape-prior in the network, a BSP pooling layer, inspired by a super-pixel pooling network [46], is specially designed to perform pooling operation over each polygon region in the BSP map rather than regular grid regions. As shown in figure 4.7, with the feature map and BSP map as inputs, the BSP pooling layer, a polygon-region pooling layer is able to generate polygon-region constrained feature maps. Given that the height and width of the input feature map is X, Y and its depth is C and the number of the polygons in BSP map is K so that we have a input feature map (C, X, Y) and a BSP map (K, X, Y), in the BSP pooling layer the pixel values from the feature map is averaged out if they belong to the same polygon region in the BSP map. In this way, a (K, C) pooled feature map can be generated. Then BSP un-pooling is used to restore the (K, C) pooled feature map back to (X, Y) BSP feature map, which simply aggregates the features from C channels and fill them into corresponding polygon regions in the feature map.

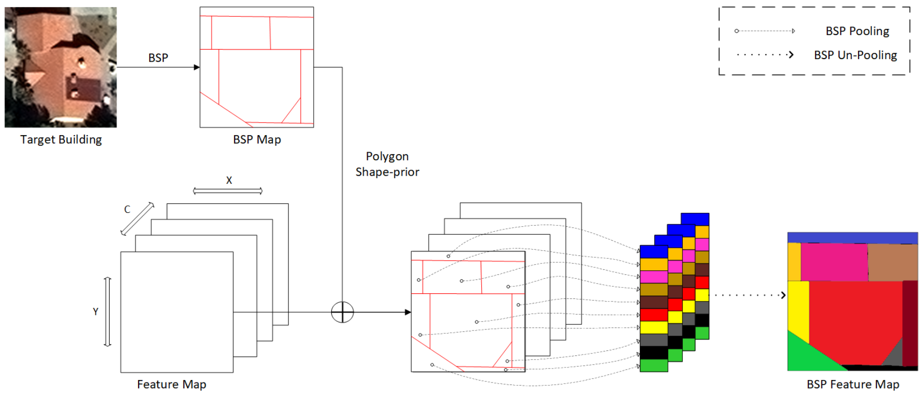


Figure 4.7: BSP Pooling Layer:

More computation details of BSP pooling layer are presented here. Considering that each element of the input feature map is I_{dij} $(d = 1,...,C; i = 1,...,X; j = 1,...,Y)$ and each polygon region of BSP map is P_t $(t = 1,...,K)$, the BSP pooling outcome for each polygon region B_{dt} is calculated as:

$$B_{dt} = \frac{1}{N_t} \sum_{(i,j)\in P_t} I_{dij} \tag{4.1}$$

in which $(i, j) \in P_t$ means the elements in the feature map that belong to t^{th} polygon region and N_t denotes the total number of the elements in t^{th} region.

For the back-propagation during the network training, the gradient update of BSP pooling layer is defined as:

$$\frac{\delta B_{dt}}{\delta I_{dij}} = \begin{cases} \frac{1}{N_t} & \text{if}(i,j) \in P_t \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

In BSP pooling layer, the BSP map provides critical polygon shape information to guide the pooling operation, which is constrained to collect features only from the same polygon region in the input feature map to represent one homogeneous feature for the corresponding polygon region. In this way, we argue that the building polygon shape-prior information is integrated into our network. Besides, integrating polygons with more smooth boundaries from BSP process into the neural network can benefit the boundary regularization task.

4.3.3 Architecture of BSP Pooling Network

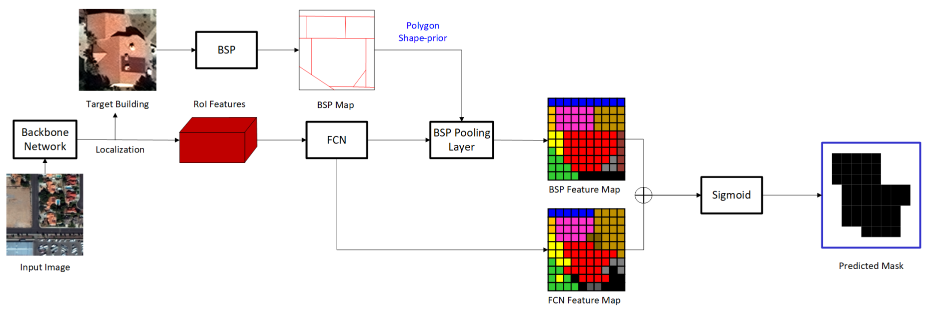


Figure 4.8: BSPPN Architecture.

As illustrated in figure 4.8, our BSPPN combines the backbone network, BSP process, BSP pooling layer and FCN to predict its segmentation masks for target buildings. We first input the image into the backbone network to get the target buildings and corresponding RoI features. The images of the target buildings are input into the BSP process to generate the BSP maps, which possess shape prior of polygon regions. Note that BSP process happens outside of the network, thus not participating the network training and not optimized by gradient descent. The RoI features are fed into a FCN layer to output pixel-wise FCN feature maps. The FCN layer shares the same structure of that in figure 4.2 except that we don't apply sigmoid in this layer. Then our BSP pooling layer is utilized to integrate the FCN feature maps and shape-prior from BSP maps to obtain BSP feature maps. Finally, BSP feature maps and FCN feature maps are combined element-wise and a sigmoid is applied to get segmentation masks.

Compared to the building extraction framework in section 4.2, BSPPN utilizes two types of feature maps, FCN and BSP feature maps for the mask prediction and boundary regularization tasks. Both frameworks share the same pixel-wise FCN feature maps, which have different features among each element of the feature maps to delineate mask predictions at pixel level. In contrast, BSP feature maps will predict segmentation masks at polygon-region level because the features from one polygon region are the same in BSP feature maps. By combining both FCN and BSP feature maps, our BSPPN can additionally learn polygon geometric information and produce more polygon-like mask predictions rather than purely pixel-wise segmentation masks. The polygon shape-prior can also bring better regularized boundaries.

4.4 Loss Design

To accomplish building object detection and segmentation, three losses are calculated including RPN loss and localization loss 3.4 from the backbone network and mask prediction loss.

Because the backbone network is used in our shape-prior integration frameworks, RPN loss and localization loss from the backbone network need to be calculated. For the mask prediction, since it's targeted to assign binary semantic class labels of building and non-building to each pixel of the input image, the binary cross entropy loss is employed:

$$L_{mask}(p(y)) = -\frac{1}{N}\sum_{i=1}^{N}(ylog(p(y)) + (1-y)log(1-p(y))) \tag{4.3}$$

where N denotes the total number of the pixels in the input image and $p(y_i)$ denotes the probability that the $i^t h$ pixel belongs to class label y (0 and 1). Here $p(y)$ is the output of our networks.

Overall, the loss function for our shape-prior integration networks are:

$$L_{Shape-prior} = L_{rpn} + L_{loc} + L_{mask} \tag{4.4}$$

Since all the losses are either smooth L_1 loss or entropy loss and are regularized to certain scale, we can train them in parallel.

4.5 Summary

In this chapter, we present two novel architectures to integrate building polygon shape-prior into the existing instance segmentation model. The shape-prior integration is aimed for polygon geometric information extraction to enhance building boundary regularization. Inspired by Mask R-CNN, we combine the backbone network, a object detection model and a semantic segmentation model, FCN applied on each RoI feature to accomplish instance segmentation. Based on this, a polygon shape optimization algorithm is applied to post process the output of the instance segmentation network. In addition, shape-prior is injected in the middle of the network to form BSP pooling network, which can obtain polygon based BSP features to guide the original networks to learn more geometric information to better regularize the building boundaries.

Chapter 5

Region-based Polygon GCN

In this chapter, we will present another building footprint extraction network, Region-based Polygon GCN (R-PolyGCN), which is based on the object detection network and the graph neural network [64], to achieve the building extraction and solve boundary regularization problems via geometric learning.

5.1 Overview

Even integrated with polygon shape-prior, the models in the last chapter are no more than traditional segmentation methods, which are intended to label every pixel of the the images and are not capable of directly exploiting the geometric shape information because the pixel-wise representation of the polygon shape has much less geometric meaning than using vertices and edges to delineate polygons. Nevertheless, graph model is exactly a representation of data structure with vertices and edges, which can be employed to depict our building polygons with much richer geometric property. The graph neural network can also be designed for convolution operations to allow the feature information exchange among vertices for geometric learning [45, 65]. Hence, to leverage the geometric nature of the graph model and the graph neural network, we combine the object detection backbone network in chapter 3 and a graph convolutional network to propose our R-PolyGCN, a regional based GCN to specially detect buildings from satellite images and directly predict the locations of polygon vertices by implicitly learning the geometric polygon shapes. Then simply joining the vertices in order with straight lines will result in building footprint extractions with much more regularized boundaries.

5.2 Network Details

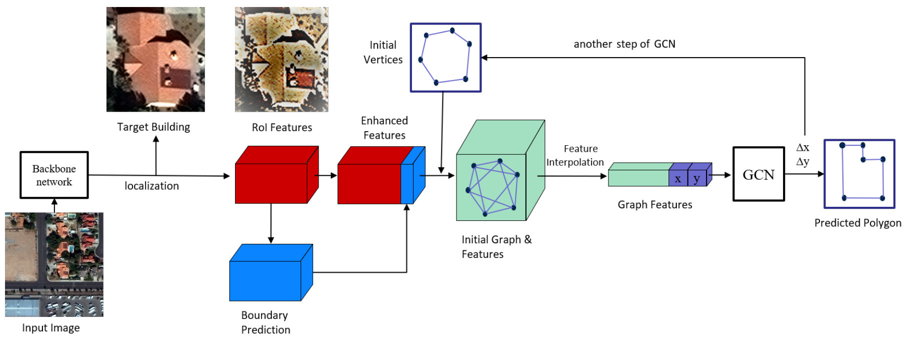


Figure 5.1: R-PolyGCN Architecture.

Inspired by [45], the architecture of R-PolyGCN is demonstrated in the figure 5.1. Firstly, the backbone network is still utilized to detect the target buildings and provide well-localized RoI features. Next the geometric shapes of building polygons are learned through GCN on these well-localized regions. From the RoI features we additionally predict boundary masks and concatenate them as boundary features onto the RoI feature map to obtain enhanced features. Since the major goal of our GCN is to move the initial polygon vertices to the boundary of the target building polygons, we first generate initial vertices, which follow a pre-defined order. Then initial graph are generated and the graph features for each vertex are interpolated from the enhanced features. Next, graph features are fed into a multi-step and multi-layer graph convolutional networks, which can predict the vertex offsets. By adding the vertex offsets to the initial vertex coordinates and connecting the vertices we can acquire the final polygon prediction. More details are introduced in the following sections.

5.2.1 Feature Enhancement

On the top of the RoI feature maps extracted from the backbone network, we specially train two fully connected layers to predict polygon boundary masks including edge masks and vertex mask of the target building. The boundary prediction represents the pixel-wise probabilities of edges and vertices of the building polygon. The edge logits and vertex logits of the predicted boundary are then concatenated with RoI features to create an enhanced feature map, which is denoted as F_{en} in this chapter. The enhanced features can outperform plain RoI features in terms of recognizing building boundaries because of their confidence of polygon boundary existence.

5.2.2 Graph Initialization

The polygon vertices of the target building are initialized using N points, which are allocated as the vertices of a regular polygon. Linking the initial vertices with straight lines generates the initial polygon. Note that the number of the vertices per polygon is unified and kept the same with the ground truth data and the vertices are in clockwise order, which makes the sequence of vertices well defined and the topology of the polygon well reserved. Then the initial vertices are put at the central part of the enhanced feature map F_{en}. Let $v_i = [x_i, y_i]$ denote the location of the i^{th} vertex and

$V = (v_1, v_2, \ldots, v_N)$ be the set of all polygon vertices, which serve as the initial nodes of our graph model. The edges of the graph E are produced by connecting each node in V with its four neighboring nodes. Linking nodes of the graph in this way allows five neighboring nodes to exchange information and effect each other in GCN, which means longer range geometric coherence. Lastly, we define the initial graph as $G = (V, E)$.

For one node v_i, based on its location in F_{en} the bilinear interpolation is adopted to obtain its node features $F_{en}(x_i, y_i)$ from the enhanced features. Then we concatenate the node's current location (x_i, y_i) and its node features in the following way:

$$f_i = concat\,\{F_{en}(x_i, y_i), x_i, y_i\} \tag{5.1}$$

where f_i is the graph feature for the node v_i and will be input into GCN. Therefore, the input graph features for each vertex are a combination of the enhanced features and the vertex location information.

5.2.3 Graph Convolutional Network

We employ a multi-step architecture here to achieve a coarse-to-fine polygon prediction. At the first step, the initial graph features are fed into a GCN to get first-round initial offsets of the polygon vertices. Then we adjust the locations of the vertices by the predicted offsets and obtain new graph features interpolated from the enhanced features again following the subsection 5.2.2. Then feed them into another GCN and produce another vertex offsets in the second step. The procedure will be iterated in the following steps so that we can get more and more accurate vertex locations as well as polygon prediction. In this work, we adopt a three-step GCN.

Within each step, a multi-layer GCN is adopted. Assuming that $N(v_i)$ denotes the neighboring nodes connected to v_i in the graph and w_0^l, w_1^l are the network weight parameters at the layer l, the basic graph convolution calculation for the node v_i at this layer is defined as:

$$f_i^{l+1} = w_0^l f_i^l + \sum_{v_j \in N(v_i)} w_1^l f_j^l \tag{5.2}$$

where f_i^l is the graph feature for node v_i and f_i^{l+1} is the output of the convolution.

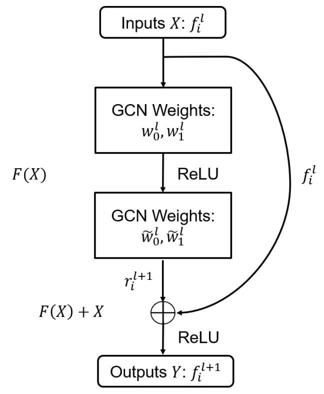


Figure 5.2: GCN: Residual Block. A skip connection directly passes the identity of the input X to the output while the GCN convolutions $F(X)$ (equation 5.3 and 5.4) are designed to learn residual information. The final output Y is a summation of the residual and the identity (equation 5.5).

Instead of the basic convolution operations, we utilize a residual block from ResNet [58] for our GCN inspired by [45, 66], which is displayed in figure 5.2. The computations of the residual block are formulated as:

$$r_i^l = ReLU(w_0^l f_i^l + \sum_{v_j \in N(v_i)} w_1^l f_j^l) \tag{5.3}$$

$$r_i^{l+1} = \tilde{w}_0^l r_i^l + \sum_{v_j \in N(v_i)} \tilde{w}_1^l r_j^l \tag{5.4}$$

$$f_i^{l+1} = ReLU(r_i^{l+1} + f_i^l) \tag{5.5}$$

in the equations above and in the figure 5.2, the first two equations are two-layer graph convolutions same with equation 5.2, but are aimed to output the residual r_i^{l+1}. The convolution weights are $w_0^l, w_1^l, \tilde{w}_0^l$ and $\tilde{w}_1^l$. Then we add the residual r_i^{l+1} and the identity of the input f_i^l. After a *ReLU* activation layer, the final output f_i^{l+1} is obtained.

5.3 Loss Design

Apart from the object detection losses from the backbone network, we design the loss functions for the boundary prediction and the GCN vertex prediction. The strategies of training these losses together are also provided.

5.3.1 Loss Functions

To accomplish objection detection for well-localized regions of interest, we still need to train the RPN loss and localization loss described in section 3.4. The rest of losses are for the boundary prediction and polygon vertex prediction.

Boundary prediction loss: in the subsection 5.2.1, the polygon boundary, including vertex masks and edge masks are produced to enhance the features. Both masks are binary. So the binary cross entropy loss function is applied to compute the vertex mask loss L_{v_mask} and edge mask loss L_{e_mask}:

$$L_{v\_mask}(p^v(y)) = -\frac{1}{N}\sum_{i=1}^{N}(ylog(p_i^v(y)) + (1-y)log(1-p_i^v(y))) \tag{5.6}$$

$$L_{e\_mask}(p^e(y)) = -\frac{1}{N}\sum_{i=1}^{N}(ylog(p_i^e(y)) + (1-y)log(1-p_i^e(y))) \tag{5.7}$$

where $p^v(y)$ and $p^e(y)$ are the pixel-wise probability of vertex mask and edge mask; y

is the binary class label 0 or 1; N is the total number of pixels. Therefore, we can have the boundary prediction loss L_{boun}:

$$L_{boun} = L_{e\_mask} + L_{v\_mask} \tag{5.8}$$

Polygon localization loss: A polygon vertex location is denoted by its coordinates $v(x,y)$ and a polygon location is represented by its N vertices: $p = \{v_i \mid i = 1,2,...N\}$. Assume our model has extracted K polygons, which are $P = \{p_k \mid k = 1,2,...K\}$. In the subsection 5.2.2, the polygon vertices are defined in clock-wise order and our ground-truth vertices are also in this order. Both point sets have the same amount of vertices per polygon as well. Therefore, we can compute the polygon location difference or the polygon distance between the GCN predicted polygon p^{pre} and the ground-truth p^{gt} by using the geometric L_1 distance, which is defined as:

$$L_1(p^{pre}, p^{gt}) = \sum_{i=1}^{N} (|x_i^{pre} - x_i^{gt}| + |y_i^{pre} - y_i^{gt}|) \tag{5.9}$$

However, the vertex correspondences aren't matched between these two point sets since the starting vertices are unknown. To find such correspondences, we fix the starting vertex of ground truth point sets and adopt an exhaust search for the optimal corresponding starting vertex of the predicted sets, which means the predicted point sets will be expanded by using every vertex of as the starting one. For one polygon, assume that the number of the vertices per polygon is N. Then N different predicted point sets are generated from original point set for the polygon. These point sets have the same clock-wise order but N different starting vertices. The L_1 distances will be calculated between the ground truth point sets and all of N expanded predicted point sets, thus resulting in N polygon distances. Among these distances, the smallest one will be selected as the polygon localization loss and the optimal correspondence of vertices can also be found. Taking all K extracted polygons into account, we have K ground truth polygons and the number of predicted polygons will be expanded to $K \times N$. Then the loss function for all the K polygons can be formulated as:

$$L_{poly}(P^{pre}, P^{gt}) = \frac{1}{K} \sum_{k=1}^{K} \min_{j \in (0,1,\ldots,N-1)} (L_1(P^{pre}_{(k+j)}, P^{gt}_k)) \tag{5.10}$$

where L_{poly} denotes our polygon localization loss, which is an average polygon distance with vertex correspondences.

Overall, the total loss function for our R-PolyGCN is:

$$L_{R-PolyGCN} = L_{rpn} + L_{loc} + L_{boun} + L_{poly} \tag{5.11}$$

5.3.2 Training Strategy

Training all the losses of R-PolyGCN is challenging and we provide some training strategies. Most of our losses here are either entropy loss or smooth L_1 loss, which can be well trained in parallel. However, the reality of the polygon localization loss is geometric point distance, which leads to obstacles when training with other losses for several reasons. Firstly, the geometric point distance has various scales and is not normalized. On the other hand, feasible and stable polygon localization loss can be only obtained until the target building regions are stabilized, which happens when the RPN loss and localization loss are small and stable. Before that, due to the region localization is not fully optimized in the backbone network, incomplete building polygons with less geometric meaning will be generated. Because our GCN model relies on the polygon geometric features, the polygon localization loss will become unreasonable causing the GCN not well trained or even wrongly trained. To tackle the training obstacles, the following strategies are utilized.

Polygon localization loss normalization: we first restrict the coordinates of the polygon vertices to the range of 0 to 1. The polygon distance is then divided by the vertex number of the polygon. A weight coefficient λ is added to the polygon localization loss as well. Then our new loss function is:

$$L_{R-PolyGCN} = L_{rpn} + L_{loc} + L_{boun} + \lambda \frac{1}{N} L_{poly} \tag{5.12}$$

where N is the number of vertices per polygon. λ will be considered as a hype-parameter and referred as polygon localization weight. The parameter is set to belong to $\{0.5, 0.75, 1.0, 1.25\}$ and will be fine-tuned during training. In this way, the polygon localization loss is regularized to same scale as other losses, which allows balanced losses to be trained.

Multi-stage training: The training is divided into majorly three stages. At the early stage of the training, we only train the RPN and localization layer in the backbone network and the boundary prediction part and ”freeze” the GCN polygon vertex prediction part. To do so, the gradient update of GCN will be shut down during the back propagation. It’s intended to obtain stably localized regions for the target buildings by only training the backbone network. After certain epochs of training, the GCN part begins to be trained to optimize the polygon prediction while keeping the backbone network frozen. Finally, the GCN part and the backbone network are trained together to fine tune the whole network. By adopting the multi-stage training pipeline, the negative effects that the backbone network can possibly bring to the GCN polygon prediction can be avoided.

5.4 Summary

In this chapter, our novel R-PolyGCN, a combination framework of the object detection network (the backbone network) and the graph neural network is demonstrated. Rather than adopt a pixel-wise representation for polygons in typical image segmentation models, we take advantage of the graph model, which uses vertices and edges to represent polygons and naturally possesses the attributes of geometric shapes. The graph convolution network is then employed to implicitly learn the geometric information of polygon shapes to predict polygon vertex locations. Specifically, a multi-step GCN is utilized to gradually adjust the polygon locations in a coarse-to-fine scheme. The predicted polygon vertices are in the pre-defined order. Therefore, the building boundary regularization can be accomplished by simply connecting the predicted vertices with straight lines. Moreover, the loss function to learn polygon locations is designed and the multi-stage training strategy is applied.

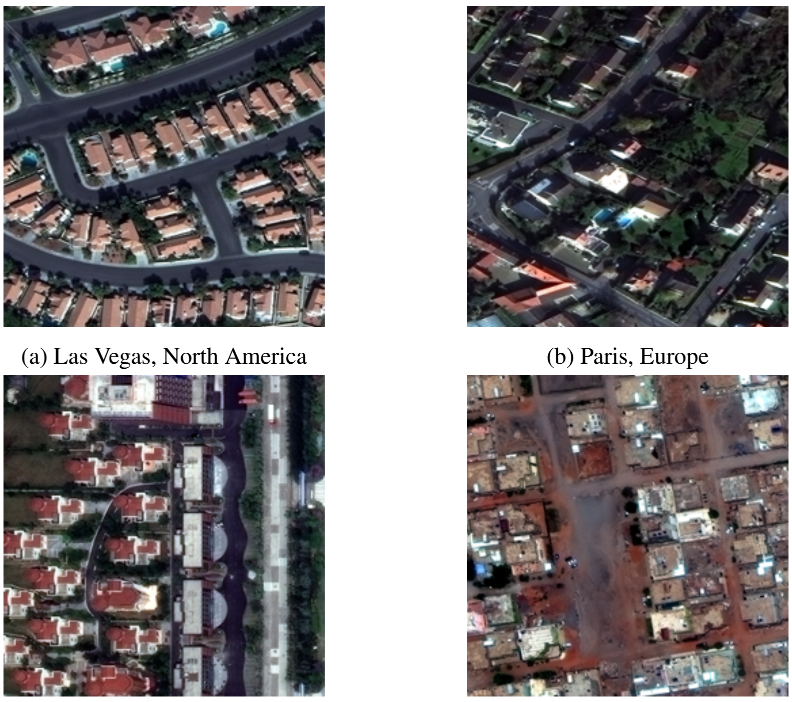
Chapter 6

Experiments and Results

In this chapter, we will introduce the characteristics of the data (in the section 6.1) and the entire experiments (in the section 6.2), from data acquisition and pre-processing, implementation of the network models to the training and testing process. The experimental results will be presented (in the section 6.3), including the accuracy and efficiency of building extraction and the performance of boundary regularization of our three novel deep learning models and the baseline model. Furthermore, some variants of our models will be compared and their limitations and problems will be discussed (in the section 6.4).

6.1 Data Characteristics

We utilize the open dataset provided by the building extraction challenge of DeepGlobe workshop [56] at Conference On Computer Vision and Pattern Recognition (CVPR) 2018. The data contain high-resolution satellite images and ground truth for the building footprints. The workshop adopts the SpaceNet building dataset introduced in section 2.4. The study area of this dataset consists of four cities (Las Vegas, Paris, Shanghai, and Khartoum) and covers both urban and suburban regions. The four cities are situated at four different continents, thus assuring the high diversity of the outlooks of the building roofs. Sample images are shown in the figure 6.1. The images


(a) Las Vegas, North America

(b) Paris, Europe

(c) Shanghai, Asia

(d) Khartoum, Africa

Figure 6.1: Sample satellite images of the four cities located at four different continents.

are captured by the DigitalGlobe Worldview-3 Satellite with GeoTiff data format. The image size is 650×650 and the resolution is $30cm$, which allows the image to cover regions of $200m \times 200m$ area. The entire dataset have 24,586 labeled satellite images with 302,701 building footprint polygons fully annotated in the whole study area. The annotations are object-wised and in the GeoJson format. However, DeepGlobe workshop only allowed 10,593 images with labeled files for public use. For the other image scenes, the labeled files were not publicly provided and the prediction results could only be evaluated on the competition website[1]. The online evaluation system is still alive after the workshop. Therefore, we selected the 10,593 public labeled images as the dataset for our study and the dataset information is displayed in the table below.

Table 6.1: Information of the dataset for our study.

City	Area (km^2)	Building Annotations	Number of Images	Data Amount (GB)
Vegas	216	108,328	3851	23
Paris	1030	16,207	1148	5.3
Shanghai	1000	67,906	4582	23.4
Khartoum	765	25,046	1012	4.7
Total	3011	217,487	10,593	56.4

6.2 Implementation Details

6.2.1 Data Pre-processing

The experimental data were acquired from Amazon Web Service (AWS) with licence permitted by the DeepGlobe workshop. The dataset provides several types of satellite images, from which we select the Pan-sharpened RGB images for our experiments. Before inputting the images into our deep learning models, the data were pre-processed in following steps.

(1) The raw RGB images are in 48 bits so we first transferred the 48-bit images into 24-bit RGB so that they can be displayed by most common image viewers.

(2) The building footprint annotations are in following format: {*ImageId, BuildingId, PolygonWKT-Pix, PolygonWKT-Geo*}, where *ImageId* and *BuildingId* specify the unique identity of the images and building instances; *PolygonWKT-Pix* and *PolygonWKT-Geo* denote the coordinates of building polygon vertices in image

[1] `https://competitions.codalab.org/competitions/18544`

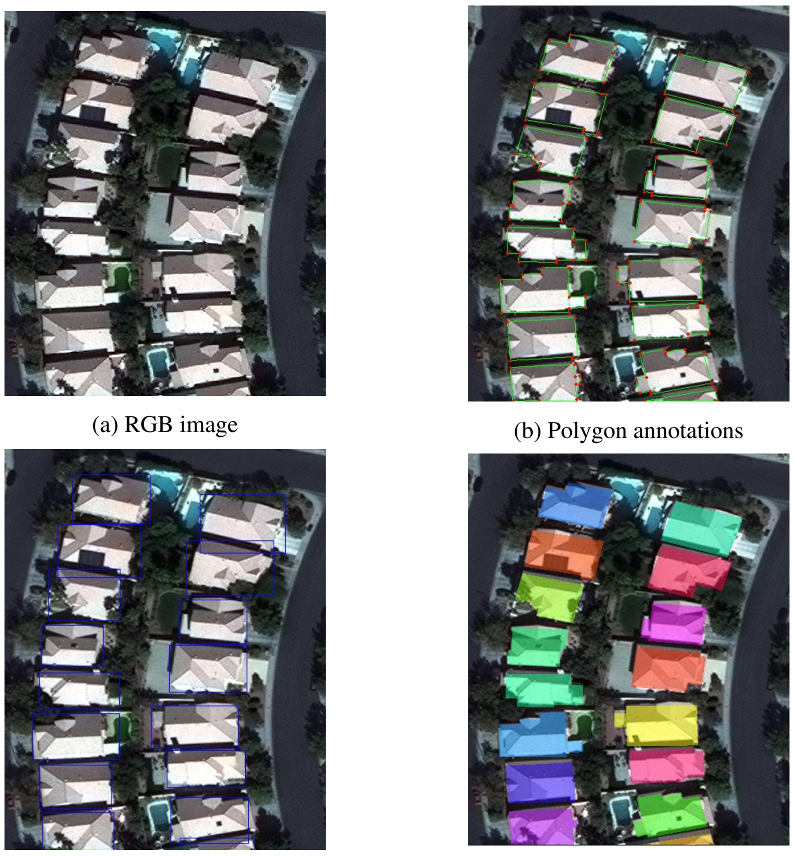
space (x,y) and in geographical space $(latitude, longitude)$. Both coordinates are in Well Known Text (WKT)[2] format. We deleted the geographic coordinates that had no usage in our experiments and converted the original geojson files to normal json format, which allowed us to load the data more easily. Besides, we made sure that the polygon vertices were in clock-wise order.

(3) Next, extra ground truth data including the bounding boxes and the pixel-wise labels were made from the building footprint annotations. For each annotated polygon, the minimum and maximum coordinates $(x_{min}, y_{min}, x_{max}, y_{max})$ were found and the top-left and bottom-right points of the corresponding bounding box were (x_{min}, y_{min}) and (x_{max}, y_{max}). To obtain the pixel-wise labels, we utilized the OpenCV[3], an open source computer vision library to generate pixel-wise polygon masks from the annotated polygon vertices. All types of annotations of the ground truth are displayed in the figure 6.2.

(4) Based on the vertex coordinates, bounding boxes and polygon masks, we discarded the blank images without any building footprints or those images with building polygons of too small areas. The threshold for the polygon area is 50 pixels.

(5) To fit original images into desired neural networks, we reshaped them from 650×650 to 1024×1024. Specifically, the images were firstly upsampled to 800×800 and a 112 padding was added. Accordingly, the sizes of annotations including the polygon vertex coordinates, bounding boxes and polygon masks were changed as well. To normalize the images, the pixel intensities were subtracted by the mean RGB values $(103.9, 116.8, 123.7)$ and were centered around 0.

[2]`https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry`
[3]`https://opencv.org/`


(a) RGB image

(b) Polygon annotations

(c) Bounding box annotations

(d) pixel-wise mask annotations

Figure 6.2: Different types of annotations of the ground truth data.

(6) The pre-processed images of each city were further divided randomly into 70% training samples, 15% validation samples and 15% testing samples along with their ground truth data.

6.2.2 Network Implementations and Configurations

Our neural network models and training and inference codes were implemented with Python 3.6 on Pytorch[4] 0.4.0, an open source deep learning platform. All the codes are publicly released at our github site[5]. Next, the hype-parameter configurations of our

[4]`https://pytorch.org/`
[5]`https://github.com/Miracle2333/`

networks and some model variants will be presented.

Backbone Network: the configurations of the backbone network is displayed in the table 6.2. The Res-101 means that a 101-layer ResNet architecture is used and the max box number denotes the maximum number of box proposals produced from RPN layer. All the other parameters have already been introduced in chapter 3.

Table 6.2: Configurations of the backbone network

Items		**Configurations**
Feature Encoding	Input Image Size	(1024, 1024)
	ResNet Layers	Res-101
	FPN Feature Sizes	(32, 32), (64, 64), (128, 128), (256, 256)
RPN	Anchor Stride	(4, 8, 16, 32, 64)
	Anchor Shape	(0.5, 1, 2)
	Anchor Scale	(32, 64, 128, 256, 512)
	NMS Threshold	0.5
	Max Box Number	256
	Positive/Negative Ratio	1:3
Localization Layer	RoI Size	(28, 28)

The shape-prior integration models: our Post-Shape and BSPPN share similar network configurations. The Post-Shape and BSPPN apply an additional FCN on the RoI feature maps and the configurations of the FCN have been presented in the section 4.2. The MDL based post-processing algorithm in Post-Shape and the BSP process in BSPPN were both implemented using C++ codes, which were complied on Qt platform[6]. For BSPPN, we explored to use different types of inputs for the BSP process to generate polygon shape-priors. In the original version, the images containing the target buildings were directly input into the BSP process. Here the binary masks of the buildings generated from the FCN layer were utilized as the alternative input so that

[6]`https://www.qt.io`

we have a variant of our BSPPN model. The performances of these two variants of BSPPN will be compared in later sections.

R-PolyGCN: The key parameters for our R-PolyGCN model are summarized in the table 6.3, in which 1+4 means that one node is connected with four neighboring nodes in the graph. For R-PolyGCN, we experimented a variant model, which doesn't include the boundary prediction as the feature enhancement.

Table 6.3: Configurations of our R-PolyGCN model

Items	**Configurations**
Number of Vertices per Polygon	16
Vertex Order	Clock-wise
Connected Nodes in the Graph	1+4
GCN steps	3
Weight λ of Polygon Loss	0.75

6.2.3 Training and Testing Details

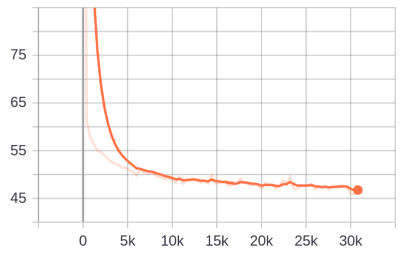
To accelerate the network training and inference, a powerful graphic processing unit (GPU), NVIDIA Geoforce 1080 with 8GB memory has been utilized. For the training, we adopt the pre-trained weights from ImageNet [67] to initialize our backbone network, on the basis of which our three novel network models are trained. To train the networks, all of the models utilize a epoch-by-epoch training scheme and each epoch has 1000 steps of iterations. The batch size is set as 1 (we couldn't increase it due to the limited GPU memory) and we employ the Adam optimizer [68]. During the training of Post-Shape and BSPPN, the initial learning rate is set as 10^{-4}. We first train the middle and end of the network, mainly the localization and FCN parts for 20 epochs and then change the learning rate to 10^{-5} and train the FPN part for 5 epochs. Finally the learning rate is decreased to 10^{-6} and the whole network is fine-tuned for another 5 epochs. Following the training strategies in the subsection 5.3.2 for our R-PolyGCN model, we initialize the learning rate as 10^{-4} and first train the backbone network for 10 epochs with the weight decay of 5×10^{-7} per 100 steps. Then the boundary prediction and GCN parts are trained for 15 epochs with the learning rate 3^{-5}, which has a weight decay of 10^{-8} per 10 steps. Finally, the whole network is fine-tuned together for 5 epochs. After the training, the models are evaluated on the testing dataset. The state-of-the-art instance segmentation model, Mask R-CNN is also trained and evaluated on the same dataset as the baseline. The training of Mask R-CNN follows the same strategies of our BSPPN.

6.3 Experimental Results

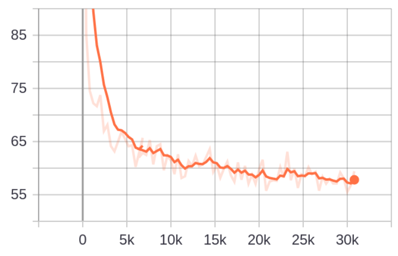
A total of four deep learning models including our three novel networks and one baseline were trained and evaluated, thus producing four sets of results. We first presented the training results of the models and gave an qualitative overview of the building extraction results and then compared the quantitative evaluation results primarily from three perspectives, the building extraction accuracy, the computation efficiency and the performance of the building boundary regularization.

6.3.1 Network Training Results

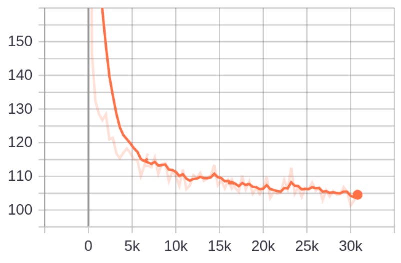
As mentioned in the subsection 6.2.3, the training has 30 epochs with 1000 steps per epoch, thus 30,000 steps in total. In the figure 6.3, we plot the sum of all kinds of losses and the average Intersection over Union (IoU) as the training results. The loss value here is multiplied by 100 for visualization. As shown in the figure, the total losses of all of the models rapidly decrease in the first 10 epochs and then reach to a stable convergence in the following epochs, which verifies the feasibility of our training process. Since the neural network part of our Post-Shape is the same with Mask R-CNN, they share one training result. The figure of the average IoU indicates that the building extraction accuracy is increasing while the deep learning models are gradually optimized during the training and proves our training works well.



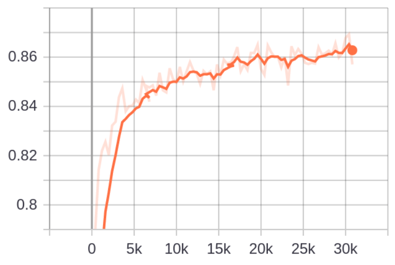
(a) Post-Shape & Mask R-CNN



(b) BSPPN



(c) R-PolyGCN



(d) Average IoU

Figure 6.3: The training results of different models. For the figure (a), (b) and (c), the horizontal axe is the training steps (30,000 in total) and the vertical axe is the sum of all the losses. Note that the total loss is multiplied by 100 here. In the figure (d), the change of the average IoU during training is also presented.

6.3.2 Overview of the Building Extraction Results

The building footprints extracted from the baseline model and our models are displayed in the figure 6.4, where input RGB images are selected from all four cities. The results are qualitatively analyzed to demonstrate the properties of our models and to prove that they can conquer the challenges of building extraction mentioned in the chapter 1.

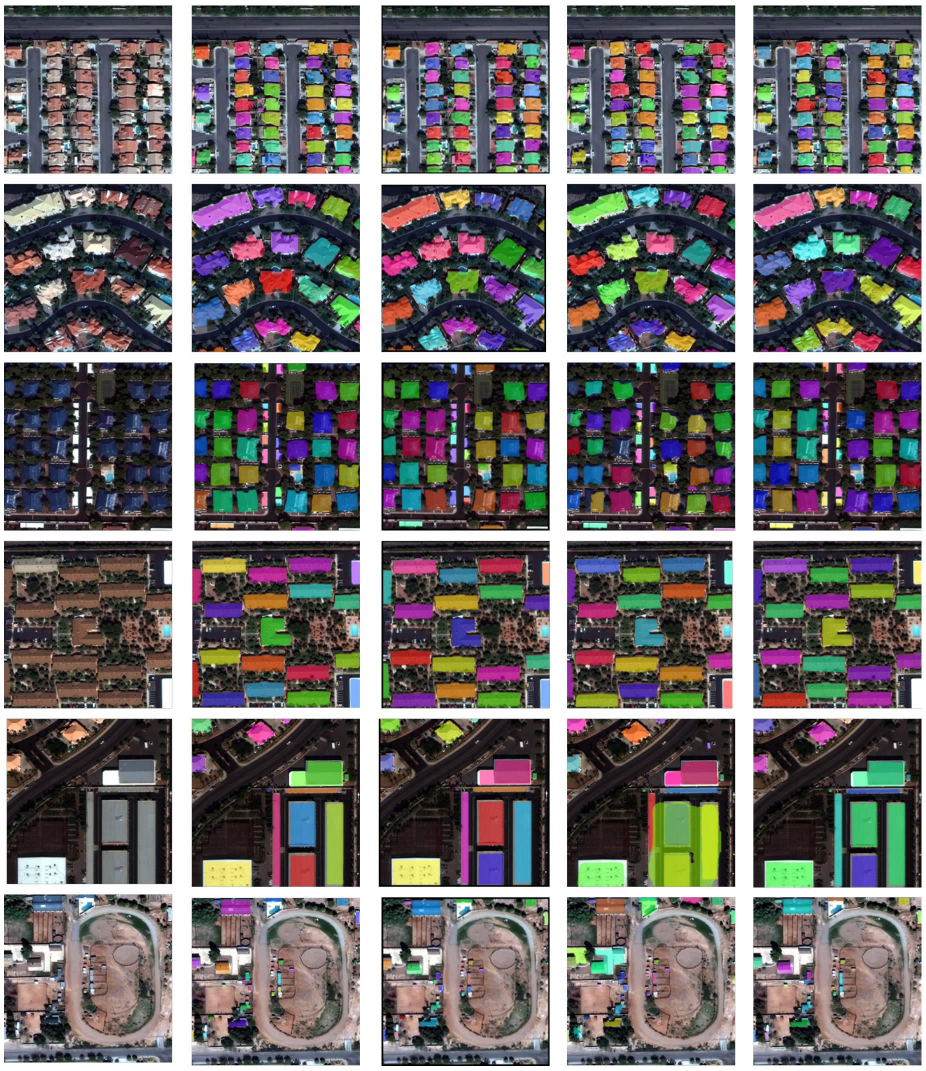


Figure 6.4: Overview of results of building footprint extraction. From left to right are RGB images and outputs of Mask R-CNN, our Post-Shape, BSPPN and R-PolyGCN. Masks of different colors represent individual building footprints the models extract.

Automatic extraction procedure: our models are completely automatic workflows. At the inference stage, the input images are directly fed into the models and processed to produce outcomes without the manual intervention.

Handling the diversity of building roof outlooks: As shown in the figure 6.4, the building roofs of different colors, textures, orientations and shapes can be properly detected by our models.

Balance between recognition and localization: the models are capable of well recognizing and localizing building objects at the same time. The recognition of building footprints are distinctive from other object categories like cars, parking lots, sports courts, trees and so on, as observed from the figure 6.4. The extracted masks are also precisely located at the correct places.

Distinguishing closely located buildings: Our models can not only detect individual buildings but also well distinguish the buildings closely situated with each other. The effects are significant in the images with densely distributed buildings, which is shown in the figure 6.5.

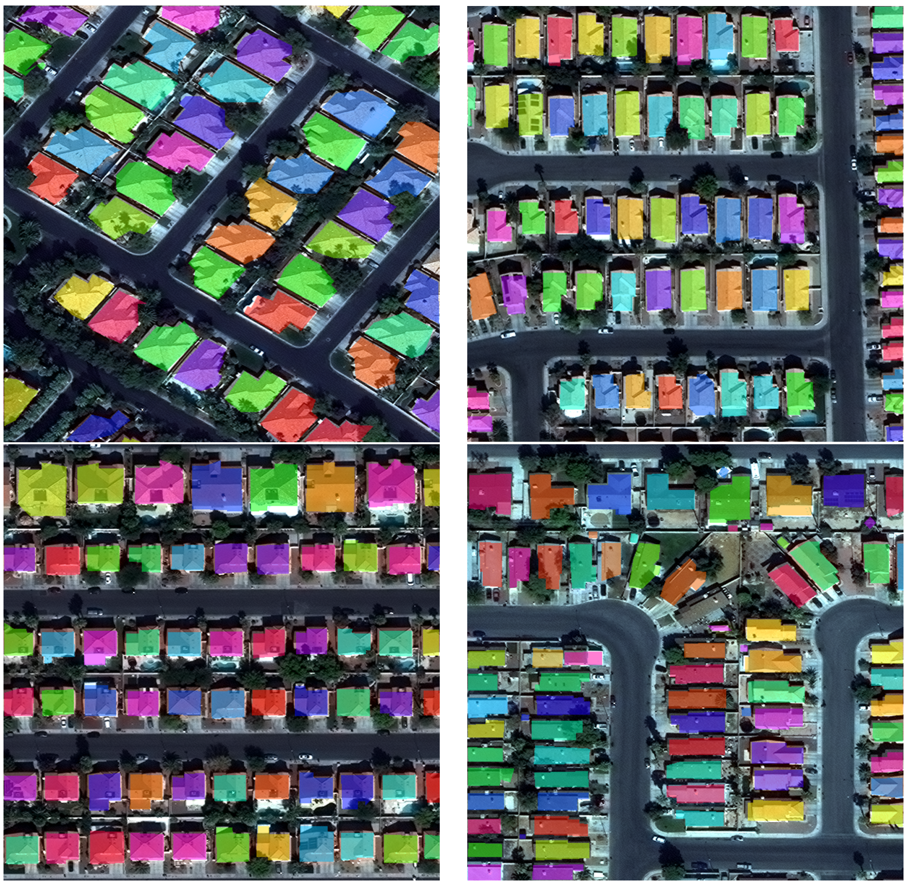


Figure 6.5: Buildings located closely are distinctively detected.

Detection of buildings of various sizes: As shown in the figure 6.6, buildings of large, middle and small sizes can be detected together, especially for the small buildings, which can be easily ignored in other detectors.

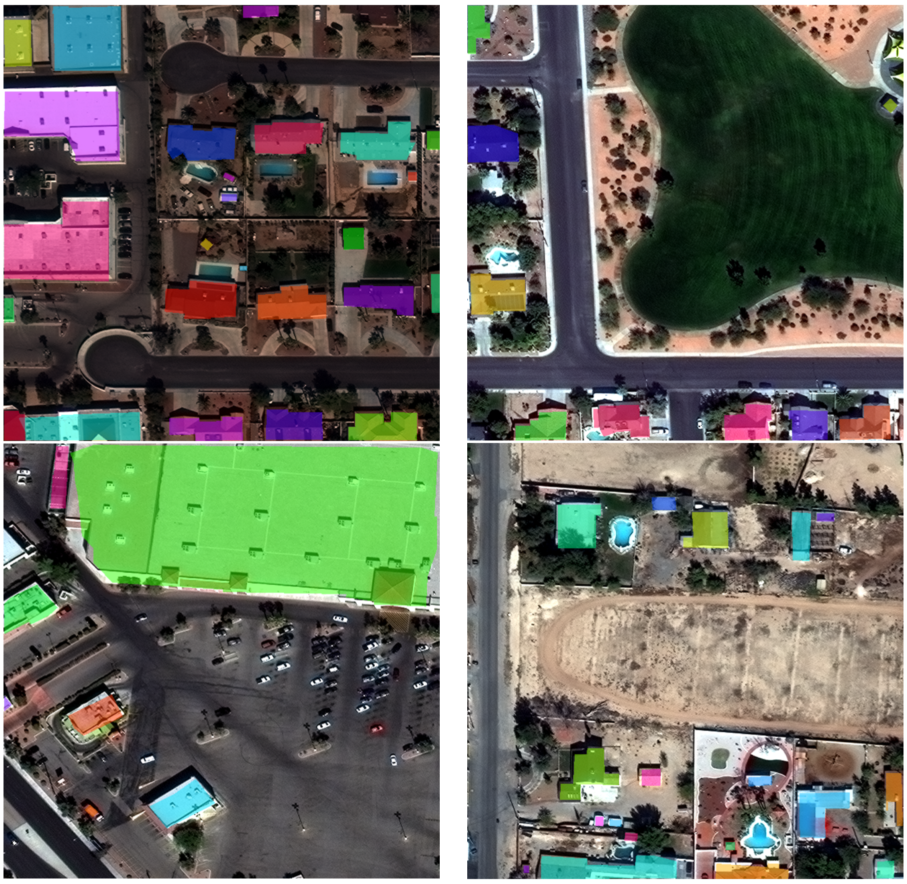


Figure 6.6: Buildings with various sizes are detected. The figures at left show the buildings of all sizes can be well detected and figures at right show that the small buildings can be spotted and localized.

Capture of geometric shapes of polygons: Equipped with geometric learning, our models can predict polygon shapes of the buildings. From the figure 6.7, buildings of various shapes, simple rectangles or complex polygon shapes, can be detected, even some subtle components of the polygon shapes, like small turning corners or short fluctuation of polylines can be captured. The boundaries of the building polygons are also regular.

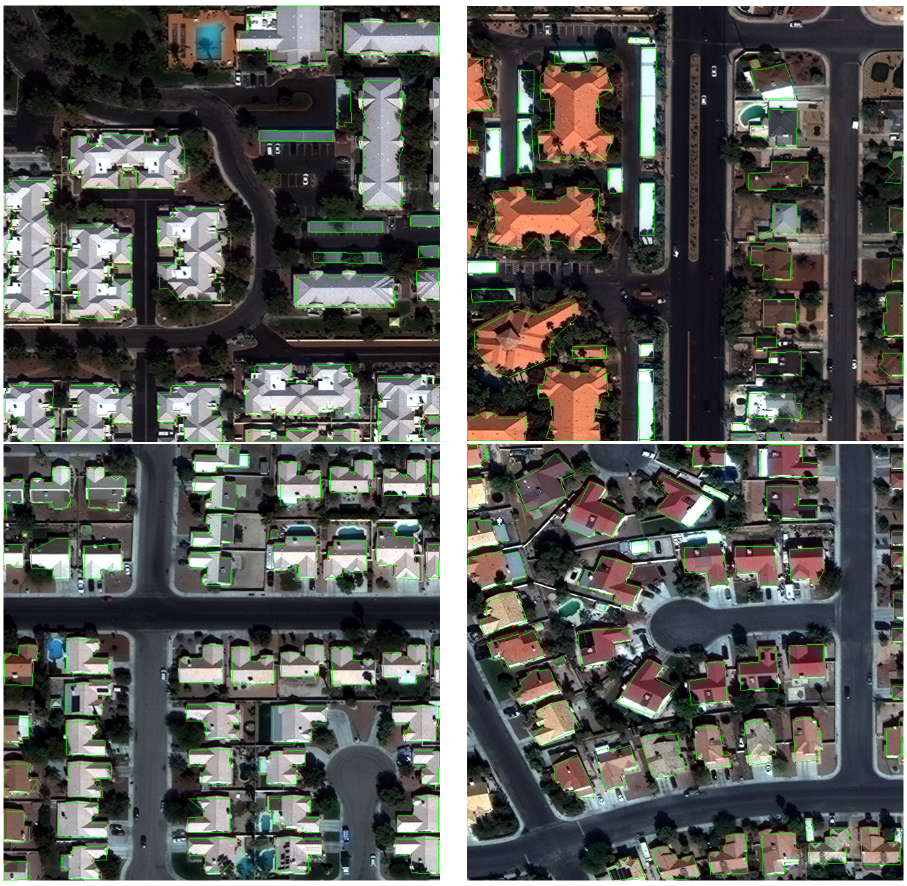


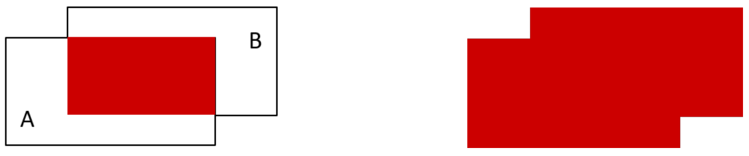
Figure 6.7: Polygon shapes of the buildings are detected. The outlines the extracted building footprints are used to represent the polygon shapes. The red points and green lines are the detected polygon vertices and polylines.

6.3.3 Building Extraction Accuracy

We took advantage of the accuracy metrics provided by the DeepGlobe workshop, which was computed by comparing the locations of the predicted building polygons and the ground truth ones. Firstly, we utilized the metric of Intersection over Union (IoU), which was calculated as:

$$IoU = \frac{Area(A \cap B)}{Area(A \cup B)} \tag{6.1}$$

where A and B denotes the predicted and the ground truth building polygons; IoU is equal to the intersection area of A and B divided by their union area, which can be illustrated in the following figure:



(a) Intersection area (b) Union area

Figure 6.8: Definition of IoU areas, referred as the red regions in the figures.

The predicted building polygon was counted as a true positive if it was the closest (measured by the IoU) proposal to a labeled polygon and the IoU between the prediction and the label was beyond the prescribed threshold of 0.5. Otherwise, the proposed polygon was a false positive. The labeled polygon that were not detected or missed in the predictions was denoted as false negative. After counting the number of true positive polygons (TP), the number of false positive polygons (FP) and the number of false negative polygons (FN), we employed the F1-score, which is a harmonic mean of precision and recall, combining the accuracy in the precision measure and the completeness in the recall measure. Suppose there are N polygon labels for the ground truth building footprints and M predicted polygons. The F1-score is calculated by the following steps:

$$Precision = \frac{TP}{TP+FP} = \frac{TP}{M} \tag{6.2}$$

$$Recall = \frac{TP}{TP+FN} = \frac{TP}{N} \tag{6.3}$$

$$\text{F1-score} = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{M+N} \tag{6.4}$$

The F-1 scores for all the four cities and a total score were computed from our three models and the baseline model. The results are showed in the table 6.4. Compared

Table 6.4: Building extraction accuracy: F1-scores evaluated on different models

Models		Las Vegas	Paris	Shanghai	Khartoum	Total
Baseline	Mask R-CNN	0.881	0.760	0.646	0.578	0.717
Ours	Post-Shape	0.878	0.754	0.642	0.571	0.714
	BSPPN	0.880	0.751	0.638	0.569	0.711
	R-PolyGCN	**0.892**	**0.786**	**0.682**	**0.612**	**0.744**

to the state-of-art instance segmentation model, Mask R-CNN, our Post-Shape and BSPPN, two shape-prior integration models show marginal accuracy decline mainly due to the fact they rely on the algorithms like the BSP process for the generation of polygon shape-prior, which are not optimized within the network and their parameters need to be pre-determined manually and are not robustly adaptive to various inputs, thus leading to unstable and unfavorable polygon shape-priors and the reduction of the extraction accuracy. Our third model, R-PolyGCN consistently has the highest detection accuracy over all other models and on the dataset of all cities. Note that the relatively low F1-scores for Shanghai and Khartoum result from the annotation of lower quality and much more buildings not orthogonal.

Because our dataset was acquired from the open challenge by the SpaceNet building dataset and DeepGlobe workshop, many participants had produced their results which were recorded and ranked on the public leaderboards[7,8]. Note that our mehtod, Post-Shpae also took part in the DeepGlobe challenge and was ranked at the fourth place. To compare with these participants, we also applied our models on the same testing data provided by the workshop, which only had raw images but no ground truth

[7]`https://community.topcoder.com/longcontest/stats/?module=ViewOverview&rd=16892`

[8]`http://deepglobe.org/leaderboard.html`

data. For evaluation, we can only submit the outcomes onto their online evaluation system to obtain the results. The online evaluation system only provided the total F1-score without the ones for individual cities. Hence, we compared the evaluation results of the total F1-scores of our models with other top participants in the table below.

Table 6.5: Building extraction accuracy: comparison to other participants

Models or	Others: SS + PP				Ours: IS + Shape		
Participants	Wleite	XD_XD	Lyft	Pasco	Post-Shape	BSPPN	R-PolyGCN
F1-score	0.643	0.693	0.736	0.739	0.713	0.710	**0.742**

Other participants include the top-2 winners of the SpaceNet competition, Wleite and XD_XD and top-2 ranked players of the challenge of the DeepGlobe workshop, Lyft and Pasco. All of these participates adopted the semantic segmentation models (SS) followed by post-processing algorithms (PP) and our approaches were the instance segmentation models (IS) with geometric learning for polygon shapes. From the table 6.5, our Post-Shape and BSPPN models produced almost equivalent accuracy to those of other participants while our R-PolyGCN outperformed all of them.

The results above reveal that the selection of the basic segmentation models and the incorporation of the geometry of polygon shapes contribute to our accuracy gain. The semantic segmentation models only produced pixel-wise semantic labels to classify buildings and the background and were unable to distinguish individual building objects. To address the problem, the post-processing algorithms were employed to separate the building regions and generate building polygons from pixel-by-pixel masks. Consequently, without post-processing, the SS models were not able to extract individual building objects while the combination of PP and SS were not capable of learning any shape information, thus failing to produce polygons with geometric properties. These strategies were what all the top participants adopted. On the contrast, our choice of the instance segmentation models can produce semantic labels as well as distinguish individual building objects, where no more post-processing was needed. Besides, our models exploited the polygon shape information as much as possible. Especially the R-PolyGCN utilized the graph models to implicitly learn the geometric attributes of polygons, which boosted the extraction accuracy.

6.3.4 Building Extraction Efficiency

Table 6.6: Training and inference times

Models		Training Time (h) for 30k steps	Inference Time (s) for one image
Baseline	Mask R-CNN	13	0.27
Ours	Post-Shape	13	250
	BSPPN	32	2.15
	R-PolyGCN	**9**	**0.18**

The training and inference time of the models can represent the building extraction efficiency. Therefore, we measured the training time for 30k steps with same training data and batch size and the inference time per image for the same testing data for the baseline and our models. The results in the table 6.6 indicate that our shape-prior integration approaches, Post-Shape and BSPPN are much more time-consuming than the baseline model, which primarily results from the MDL based shape optimization process for the Post-Shape and the BSP process for BSPPN. Running outside of the neural networks, these shape-prior generation processes are not able to utilize the GPU resources to speed up their computation and they also cost huge amount of time due to the iterations within them. Despite that Post-Shape and Mask R-CNN have almost the same network architecture and they cost equal training time for the network, the MDL based post-processing algorithm is much slower causing the inference of Post-Shape to be much less efficient. On the other hand, our R-PolyGCN shows its speed advantage by saving around 30% training time and about 60% time at inference stage than the baseline Mask R-CNN. For the high efficiency of R-PolyGCN, the credit is granted to its straightforward and unified network design without any extra procedures outside of the network compared to the shape-prior integration models. Given that Mask R-CNN and our R-PolyGCN adopt similar structure for the backbone network, we argue the phenomenon that the graph convolution network can directly output the outline of the building rather than the whole polygon region from Mask R-CNN makes difference to their efficiency performance.

6.3.5 Boundary Regularization Performance

The performance of the building boundary regularization was evaluated and the figure 6.9 displayed building boundaries extracted from the baseline model and our models and the ground truth. Since Mask R-CNN and BSPPN both outputed pixel-wise segmentation results without vertex or line prediction, we used a contour tracing algorithm in OpenCV to obtain the boundaries from the masks.

In terms of the regularization of building boundaries, we can observe that Mask R-CNN shows almost no evidence to regularize the boundaries because of its nature of grid-based pixel-by-pixel representation and lack of shape information; our BSPPN holds slightly more capacity of regularizing the boundaries benefiting from the fusion of shape-priors springing from the BSP process. However, due to the fact that the generation of shape-priors is not robust and stable and the model still uses a pixel-by-pixel representation to render polygons, BSSPN is unable to provide an ideal solution. Among all these, our Post-Shape and R-PolyGCN models can produce boundary lines closest to the ground truth with regularized characteristics. Post-Shape utilizes the

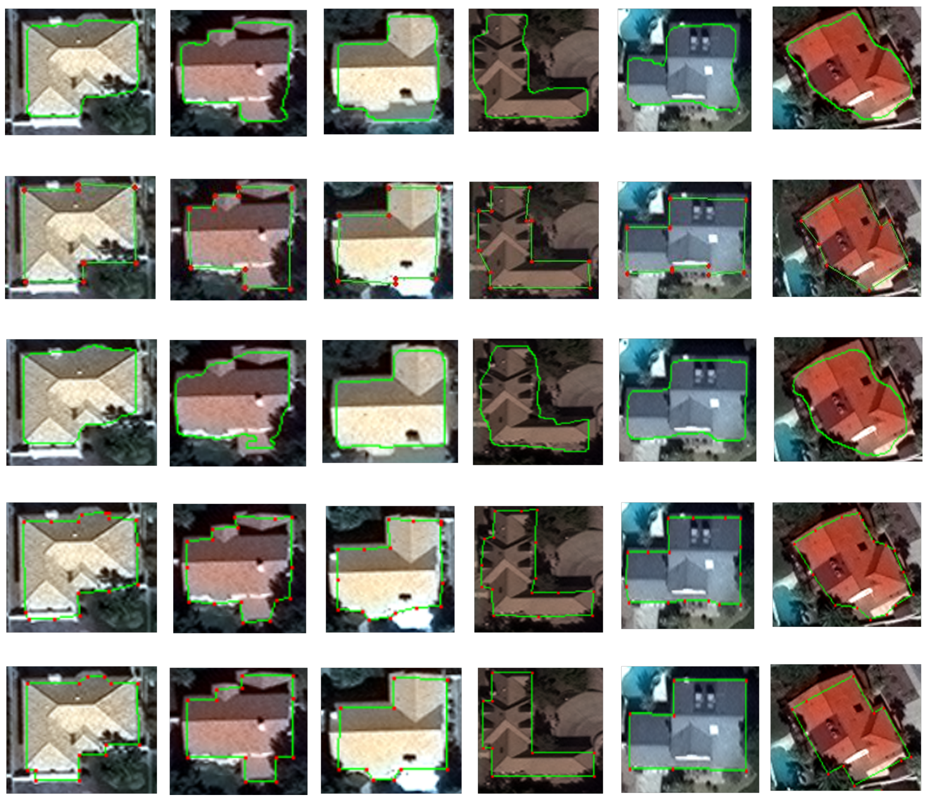


Figure 6.9: Comparisons of the performance of building boundary regularization of different models. Examples of building footprint extraction with focus on the boundaries (red points as vertices and green lines as polylines). From top to bottom are results from: Mask R-CNN, Post-Shape, BSPPN and R-PolyGCN. The images at the last row are the ground-truth.

polygon shape optimization algorithm as a post-processing step, which is able to learn the regularity of the polygons. However, Post-Shape cannot be trained end-to-end and its efficiency is too low. For R-PolyGCN, as a natural representation for the vertex, edge and polygon, the graph model employed can provide a straightforward polygon prediction based on their geometric features. Once the optimal polygon vertices are acquired, connecting them by straight lines in a pre-defined order can easily produce regularized boundaries.

6.4 Discussions

6.4.1 Variants of Models and Ablation Study.

We build variants of our models, which are trained, evaluated and compared with the original models. The variants include different hype-parameter settings, modifications of network structures and so on.

Variants of BSPPN: The variants of BSPPN are made by changing the inputs of the BSP process for the polygon shape-prior generation. Originally, the cropped images containing the target buildings are the inputs. We think that the noises of images can possibly disturb the BSP process. Especially the interior structure of the buildings can negatively affect the polygon partitioning results because the structure inside the buildings are useless in this study, thus leaving redundancy and noises. Therefore, the binary masks, which are produced from the FCN layer and have homogeneous interior structures, can be used as the alternative inputs of the BSP process. After the modification, experiments show that the total F1-score increases from 0.711 to 0.719. The BSP results produced from the two inputs are shown in the figure 6.10.

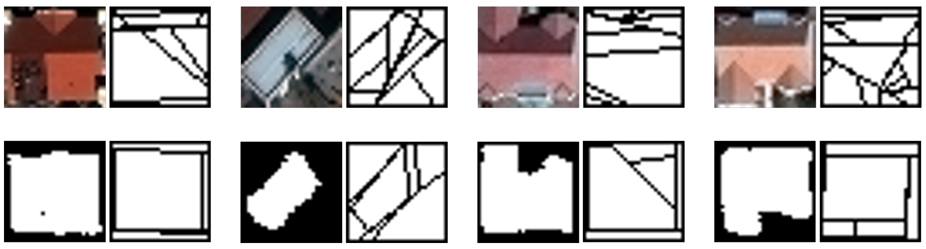


Figure 6.10: BSP maps generated from two different inputs: images and binary masks. The figures at top row are the cropped images and their outputs from BSP process; the figures at the bottom are the binary masks and their BSP maps.

Variants of R-PolyGCN: The R-PolyGCN has a boundary mask prediction within the networks to enhance the features. We experiment to remove the boundary mask prediction and feed the GCN with plain features. Meanwhile, the feature encoder of

Table 6.7: Results of ablation study of R-PolyGCN

Model Variants	Total F1-Scores
ResNet-50	0.718
ResNet-50 + Boundary	0.739
ResNet-101	0.722
ResNet-101 + Boundary	**0.744**

the backbone network is experimented with different choices including ResNet-101 and ResNet-50. The table 6.7 shows the results of the ablation study. From the table, the models with the boundary prediction has a considerable accuracy gain compared those without it. It indicates that the boundary masks provide necessary supplements to the plain features. ResNet-101 with deeper layers than ResNet-50, however, has limited contributions to the increase of the accuracy.

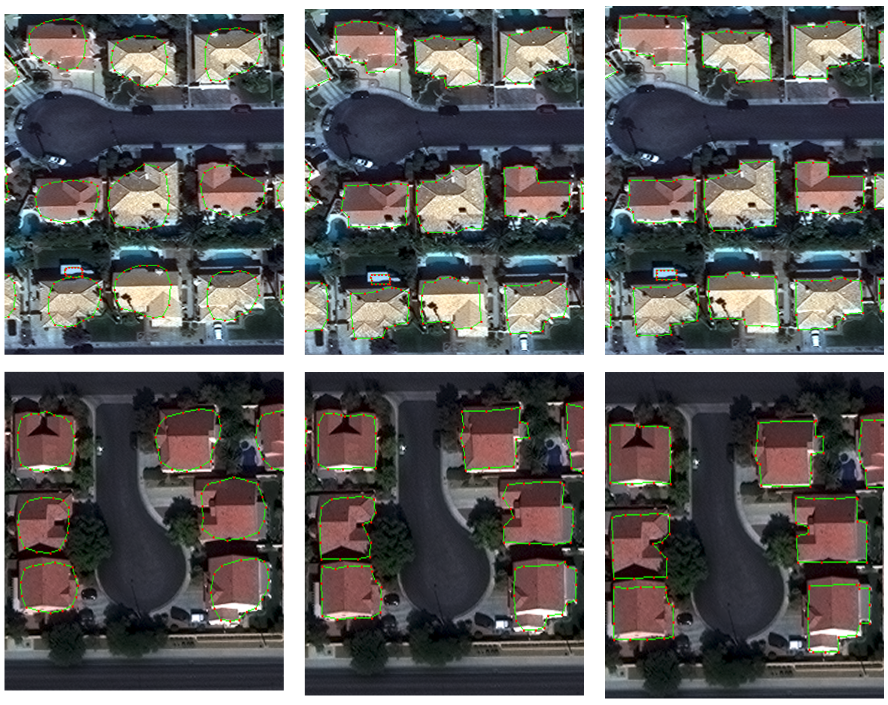

(a) 1-step (b) 2-step (c) 3-step

Figure 6.11: Results from GCN of different steps.

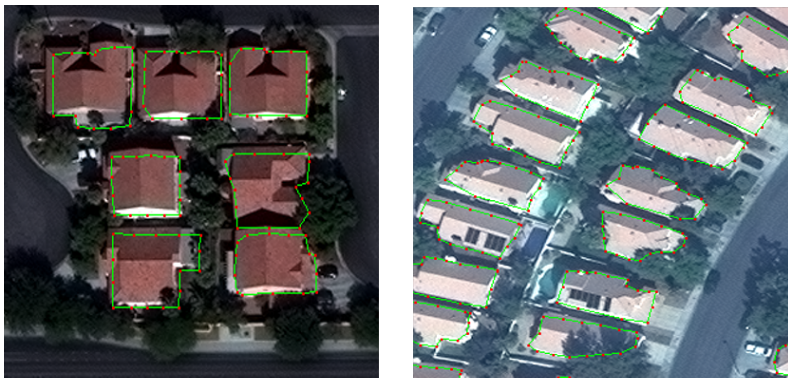
Different numbers of iteration steps of GCN are also made as variants. We experiment with 1-step, 2-step and original 3-step GCN structures, resulting in the total F1-scores of 0.711, 0.739 and 0.744 respectively. The building extraction results of the variants are displayed in the figure 6.11, which show that the 2-step GCN can produce rough polygon shapes and achieve substantial enhancement compared to the results of the 1-step GCN. The 3-step GCN shows marginal gain of the extraction accuracy but is able to refine the building polygon shapes. The results indicate that more steps of GCN might not increase the accuracy much and 3-step GCN can produce well-localized building footprints with refined polygon shapes.

6.4.2 Problems of Our Models

The existing problems of our models are discussed here.

Problems of Post-Shape: Post-Shape shares the same architecture with Mask R-CNN at the neural network parts. In addition to that, it employs a polygon shape optimization algorithm as post-processing, which converts the pixel-wise segmentation results from Mask R-CNN to geometric polygons, thus regularizing the building boundaries. The first drawback of Post-Shape is that it cannot be trained end-to-end, which means the shape optimization part is unable to be learned through the training of deep neural networks. The shape optimization can only polish the polygon boundaries and still heavily relies on the quality of the pixel-wise segmentation results. If some of the segmentation outcomes are inaccurate, the post-processing is unable to correct them. The second and the biggest problem of Post-Shape is that it is too time-consuming. It averagely takes over 4 minutes to process one image. The longest processing time for one image can be as much as 28 minutes.

Problems of BSPPN: As mentioned before, BSPPN integrates the polygon shape-prior within the networks in order to produce building polygons with more regularized boundaries. However, the improvement of the regularization performance is not as obvious as expected mainly due to the BSP process is not self-adaptive and not stable enough. Besides, the grid-based pixel-wise representation without much geometric properties, still has the limitations to render the polygon shapes.


(a) Simple buildings extracted with redundant vertices

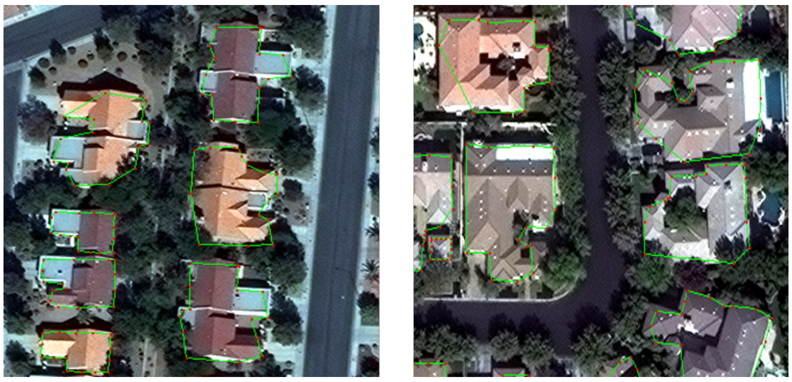

(b) Complex buildings extracted with insufficient vertices

Figure 6.12: Problems of R-PolyGCN results. The fixed number of polygon vertices causes unfavorable building footprint extractions.

Problems of R-PolyGCN: The shortcomings of R-PolyGCN are majorly the inflexibility of the initialization of the polygon graphs, where the polygons are initialized with fixed number of and pre-defined order of the vertices. As shown in the figure 6.12, simple buildings are predicted as polygons with redundant vertices while complex buildings have inadequate vertices to depict them, thus missing some polygon details or producing wrong polygons.

Another problem is that the geometric learning of GCN is implicit. Like most of the deep learning models with the characteristic of ”black box”, the mechanism inside GCN cannot be readily accessible, leading to poor interpretability of the models. In terms of the loss functions, learning for the geometric shapes of polygons are simply determined by the locations of the polygon vertices without taking other polygon geometric into account, for instance, the interior angles, the parallel of polylines, the whole polygon orientation and so on.

6.5 Summary

In this chapter, we introduce the adopted dataset and the data pre-processing. The implementation details of our models is also presented. We cover the whole procedure of the experiments and illustrate the qualitative and quantitative results, which reveal that our models can properly solve the problems of the building footprint extraction with boundary regularization. The comparison of the model performances shows that our models are competitive with the state-of-the-art instance segmentation model Mask R-CNN and our R-PolyGCN are consistently the most accurate, the most efficient and can produce building footprints with superior regularized boundaries. Discussions on the variants of the models are presented. Finally we discuss the problems of our models.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this study, we are aimed to develop a deep learning framework to automatically extract building footprints with boundary regularization from satellite images. Firstly, the massive applications of building footprints from satellite images and challenges of the building footprint extraction and boundary regularization are investigated, which are the major motivations of our study. We further formulate the main problems into the tasks of spatial learning, semantic learning and geometric learning and propose a general deep learning based framework, including the backbone network and the building extraction network, with the combination of spatial, semantic and geometric learning to provide a solution to the problem of the building footprint extraction with boundary regularization. Related research work is also reviewed.

Our methodology is deep learning models composed of the backbone network and the building extraction network. The backbone network has the functions of multi-scale feature encoding and object detection and can produce the well-localized RoI features, which will be essential in the following building extraction network. We explore two pipelines to design the building extraction network. One is to integrate the polygon shape-prior with the deep neural networks to take advantage the shape-prior information. Two types of integration models are proposed, integrating the shape-prior at the post-processing stage (Post-Shape) and injecting them within the network (BSPPN). The former employs a MDL based polygon shape optimization algorithm to process the segmentation masks produced from the networks; the latter specially designs a polygon region based pooling layer, BSP pooling layer to inject the polygon shape-prior produced from the BSP process into the networks. Our second building extraction network is R-PolyGCN, which exploits the graph representation for polygons and the graph convolutional networks for geometric learning. In total, three models, Post-Shape, BSPPN and R-PolyGCN are proposed.

Comprehensive experiments are conducted on an open dataset. Pre-processing is first applied to the raw satellite images and annotations. Then our three models and the baseline model, Mask R-CNN are trained and evaluated. The qualitative results show that our models can successfully perform the automatic extraction of building footprints. The building extraction accuracy results show that our models are competitive with the baseline model and the models from the top participants in the leaderboard of the open building extraction challenge. Particularly, our R-PolyGCN outperforms all the others in terms of extraction accuracy. The efficiency of the models are also analysed, which shows that the shape-prior integration models, Post-Shape and BSPPN are more time-consuming than the baseline model while R-PolyGCN is the most efficient one at both training and inference stages. We compare the performances of models on the building boundary regularization and find that our Post-Shape and R-PolyGCN demonstrate outstanding capacity to produce regular building boundaries. The experiments on the variants of the models and the ablation study are also carried out to provide deeper views of the deep neural network models. Finally, the drawbacks and limitations of our models are discussed.

7.2 Future Work

Based on the existing problems of our study and current development of the field, the future directions of our work are summarized as follows:

- **The backbone network can be further improved:** utilizing the ideas of the classic Faster R-CNN, our backbone network is a typical two-stage object detection network, which relies on the selection and localization of anchor boxes before the final localization layer. Recently, the one-stage networks are gaining more and more attentions, which can directly detect bounding boxes from image grids or key points instead of regression from the anchor boxes. The state-of-art one stage models like [13, 69, 70] are showing competitive detection accuracy with much higher efficiency and much easier training. Therefore, we can change the backbone network to one-stage structure to simplify our networks while preserving the performance.

- **The shape-prior generation can be learned through neural networks:** The MDL based polygon optimization in our Post-Shape model and the BSP in our BSPPN model are both generating polygon shape-prior outside of the neural networks, thus being unable to be optimized with the networks. The independence causes the shape-prior generation process hard to control. These days many conventional computer vision algorithms successfully become differentiable and learnable layers of the neural network, like the superpixel generation in [71].

Thus building the learnable networks for the polygon shape-prior generation is a feasible and valuable direction.

- **More geometric learning can be introduced into the GCN model:** Even though our R-PolyGCN is quite powerful to learn the geometric information, it still relies on simple geometric features and more can be certainly added to strengthen the GCN model, as mentioned in the subsection 6.4.2. One way is to pose more constrains at the loss functions. For example, instead of pre-defining a fixed number, we can take the number of polygon vertices as a learnable parameter and design a loss function to find the optimal number. Or the interior angles of the polygons can be taken into consideration and put into the loss functions given that many regularized polygons have the orthogonal angles. Moreover, the interpretability of GCN can be further studied for a fuller utilization of the graph models for geometry learning.

- **More quantitative analysis on the building boundary regularization can be conducted:** The regularity of the polygon shapes or shape similarity can be employed to measure the performance of the boundary regularization to provide a more quantitative analysis.

Bibliography

[1] J. Campbell and R. Wynne, *Introduction to Remote Sensing, Fifth Edition*. Guilford Publications, 2011.

[2] J. Jung, Y. Jwa, and G. Sohn, "Implicit regularization for reconstructing 3d building rooftop models using airborne lidar data," *Sensors*, vol. 17, no. 3, 2017.

[3] J. Jung and G. Sohn, "A line-based progressive refinement of 3d rooftop models using airborne lidar data with single view imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 149, pp. 157–175, 2019.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2014.

[8] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, 1995.

[9] R. Girshick, "Fast r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec 2015.

[10] J. S. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Advances in Neural Information Processing Systems 2*, 1990.

[11] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *The European Conference on Computer Vision (ECCV)*, 2016.

[14] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *The European Conference on Computer Vision (ECCV)*, 2018.

[15] X. Zhou, J. Zhuo, and P. Krahenbuhl, "Bottom-up object detection by grouping extreme and center points," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[16] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, Apr 2017.

[17] 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*, 2017.

[18] O. Ronneberger, P.Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, ser. LNCS, vol. 9351, 2015.

[19] P. O. Pinheiro, R. Collobert, and P. Dollar, "Learning to segment object candidates," in *Advances in Neural Information Processing Systems 28*, 2015.

[20] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," in *The European Conference on Computer Vision (ECCV)*, 2016.

[21] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[22] V. Mnih, "Machine learning for aerial image labeling," Ph.D. dissertation, University of Toronto, 2013.

[23] 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*, vol. 55, no. 12, pp. 7092–7103, Dec 2017.

[24] B. Bischke, P. Helber, J. Folz, D. Borth, and A. Dengel, "Multi-task learning for segmentation of building footprints with deep neural networks," *ArXiv*, vol. abs/1709.05932, 2017.

[25] G. Wu, X. Shao, Z. Guo, Q. Chen, W. Yuan, X. Shi, Y. Xu, and R. Shibasaki, "Automatic building segmentation of aerial imagery using multi-constraint fully convolutional networks," *Remote Sensing*, vol. 10, no. 3, 2018.

[26] Y. Liu, S. Piramanayagam, S. T. Monteiro, and E. Saber, "Dense semantic labeling of very-high-resolution aerial imagery and lidar with fully-convolutional neural networks and higher-order crfs," in *The IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jul 2017.

[27] N. Audebert, B. Le Saux, and S. Lefevre, "Joint learning from earth observation and openstreetmap data to get faster better semantic maps," in *The IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jul 2017.

[28] W. Li, C. He, J. Fang, J. Zheng, H. Fu, and L. Yu, “Semantic segmentation-based building footprint extraction using very high-resolution satellite images and multi-source gis data,” *Remote Sensing*, vol. 11, no. 4, 2019.

[29] W. Sun and R. Wang, “Fully convolutional networks for semantic segmentation of very high resolution remotely sensed images combined with dsm,” *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 3, pp. 474–478, Mar 2018.

[30] G. Máttyus and R. Urtasun, “Matching adversarial networks,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2018.

[31] X. Pan, F. Yang, L. Gao, Z. Chen, B. Zhang, H. Fan, and J. Ren, “Building extraction from high-resolution aerial imagery using a generative adversarial network with spatial and channel attention mechanisms,” *Remote Sensing*, vol. 11, no. 8, 2019.

[32] 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 Workshops (CVPRW)*, Jun 2018.

[33] Q. Wen, K. Jiang, W. Wang, Q. Liu, Q. Guo, L. Li, and P. Wang, “Automatic building extraction from google earth images under complex backgrounds based on deep instance segmentation network,” *Sensors*, vol. 19, no. 2, 2019.

[34] S. Xia and R. Wang, “Extraction of residential building instances in suburban areas from mobile lidar data,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 144, pp. 453–468, 2018.

[35] D. Marmanis, K. Schindler, J. Wegner, S. Galliani, M. Datcu, and U. Stilla, “Classification with an edge: Improving semantic image segmentation with boundary detection,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 135, pp. 158–172, 2018.

[36] M. Volpi and D. Tuia, “Deep multi-task learning for a geographically-regularized semantic segmentation of aerial images,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 144, pp. 48–60, 2018.

[37] G. Wu, Z. Guo, X. Shi, Q. Chen, Y. Xu, R. Shibasaki, and X. Shao, “A boundary regulated network for accurate roof segmentation and outline extraction,” *Remote Sensing*, vol. 10, no. 8, 2018.

[38] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, Jan 1988.

[39] D. Marcos, D. Tuia, B. Kellenberger, L. Zhang, M. Bai, R. Liao, and R. Urtasun, “Learning deep structured active contours end-to-end,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2018.

[40] D. Cheng, R. Liao, S. Fidler, and R. Urtasun, “Darnet: Deep active ray network for building segmentation,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[41] N. Girard and Y. Tarabalka, “End-to-end learning of polygons for remote sensing image classification,” in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Valencia, Spain, Jul 2018.

[42] L. Castrejón, K. Kundu, R. Urtasun, and S. Fidler, "Annotating object instances with a polygon-rnn," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[43] D. Acuna, H. Ling, A. Kar, and S. Fidler, "Efficient interactive annotation of segmentation datasets with polygon-rnn++," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2018.

[44] Z. Li, J. D. Wegner, and A. Lucchi, "Polymapper: Extracting city maps using polygons," *CoRR*, vol. abs/1812.01497, 2018.

[45] H. Ling, J. Gao, A. Kar, W. Chen, and S. Fidler, "Fast interactive object annotation with curve-gcn," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[46] S. Kwak, S. Hong, and B. Han, "Weakly supervised semantic segmentation using superpixel pooling network," in *AAAI*, 2017.

[47] M. Schuurmans, M. Berman, and M. B. Blaschko, "Efficient semantic image segmentation with superpixel pooling," *arXiv e-prints*, p. arXiv:1806.02705, Jun 2018.

[48] M. Tofighi, T. Guo, J. K. P. Vanamala, and V. Monga, "Prior information guided regularized deep learning for cell nucleus detection," *IEEE Transactions on Medical Imaging*, pp. 1–1, 2019.

[49] W. Kuo, A. Angelova, J. Malik, and T.-Y. Lin, "ShapeMask: Learning to segment novel objects by refining shape priors," *arXiv e-prints*, p. arXiv:1904.03239, Apr 2019.

[50] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem, "3d-prnn: Generating shape primitives with recurrent neural networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[51] J. Huang, J. Gao, V. Ganapathi-Subramanian, H. Su, Y. Liu, C. Tang, and L. J. Guibas, "Deepprimitive: Image decomposition by layered primitive detection," *Computational Visual Media*, vol. 4, no. 4, pp. 385–397, Dec 2018.

[52] F. Rottensteiner, G. Sohn, J. Jung, M. Gerke, C. Baillard, S. Benitez, and U. Breitkopf, "The isprs benchmark on urban object classification and 3d building reconstruction," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. I-3, pp. 293–298, 2012.

[53] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark," in *IEEE International Symposium on Geoscience and Remote Sensing (IGARSS)*, Fort Worth, United States, Jul 2017.

[54] S. Wang, M. Bai, G. Mattyus, H. Chu, W. Luo, B. Yang, J. Liang, J. Cheverie, S. Fidler, and R. Urtasun, "Torontocity: Seeing the world with a million eyes," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[55] Q. Chen, L. Wang, Y. Wu, G. Wu, Z. Guo, and S. L. Waslander, "Temporary removal: Aerial imagery for roof segmentation: A large-scale dataset towards automatic mapping of buildings," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 147, pp. 42–55, 2019.

[56] 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 Workshops (CVPRW)*, Jun 2018.

[57] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[59] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[60] P. J. Huber, "Robust estimation of a location parameter," *Ann. Math. Statist.*, vol. 35, no. 1, pp. 73–101, Mar 1964.

[61] J. Jung, Y. Jwa, and G. Sohn, "Implicit regularization for reconstructing 3d building rooftop models using airborne lidar data," *Sensors*, vol. 17, no. 3, 2017.

[62] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing*, vol. 1, no. 3, pp. 244–256, Nov 1972.

[63] B. Naylor, J. Amanatides, and W. Thibault, "Merging bsp trees yields polyhedral set operations," *SIGGRAPH Comput. Graph.*, vol. 24, no. 4, pp. 115–124, Sep 1990.

[64] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, Jan 2009.

[65] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, pp. 18–42, 2017.

[66] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *The European Conference on Computer Vision (ECCV)*, Sep 2018.

[67] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[68] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.

[69] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," *arXiv e-prints*, p. arXiv:1904.08189, Apr 2019.

[70] X. Zhou, J. Zhuo, and P. Krähenbühl, "Bottom-up object detection by grouping extreme and center points," *arXiv e-prints*, p. arXiv:1901.08043, Jan 2019.

[71] V. Jampani, D. Sun, M.-Y. Liu, M.-H. Yang, and J. Kautz, "Superpixel sampling networks," in *The European Conference on Computer Vision (ECCV)*, Sep 2018.

Glossaries

AI	Artificial Intelligence
AP	Anchor Point
BSP	Binary Space Partitioning
BSPPN	BSP Pooling Network
CNN	Convolutional Neural Networks
DEM	Digital Elevation Model
DL	Description Length
DNN	Deep Neural Networks
DSM	Digital Surface Model
FCN	Fully Convolutional Network
FL	Floating Line
FP	Floating Point
FPN	Feature Pyramid Network
GAN	Generative Adversarial Networks
GCN	Graph Convolutional Network
GIS	Geographic Information System
GL	Guiding Line
GP	Guiding Point
LiDAR	Light Detection and Ranging
MDL	Minimum Description Length
NMS	Non-Maximum Suppression
OSM	OpenStreetMap
PC	Point Cloud
R-CNN	Region-based Convolutional Neural Network

ResNet	Residual Network
RoI	Region of Interest
RPN	Region Proposal Network
R-PolyGCN	Region-based Polygon GCN