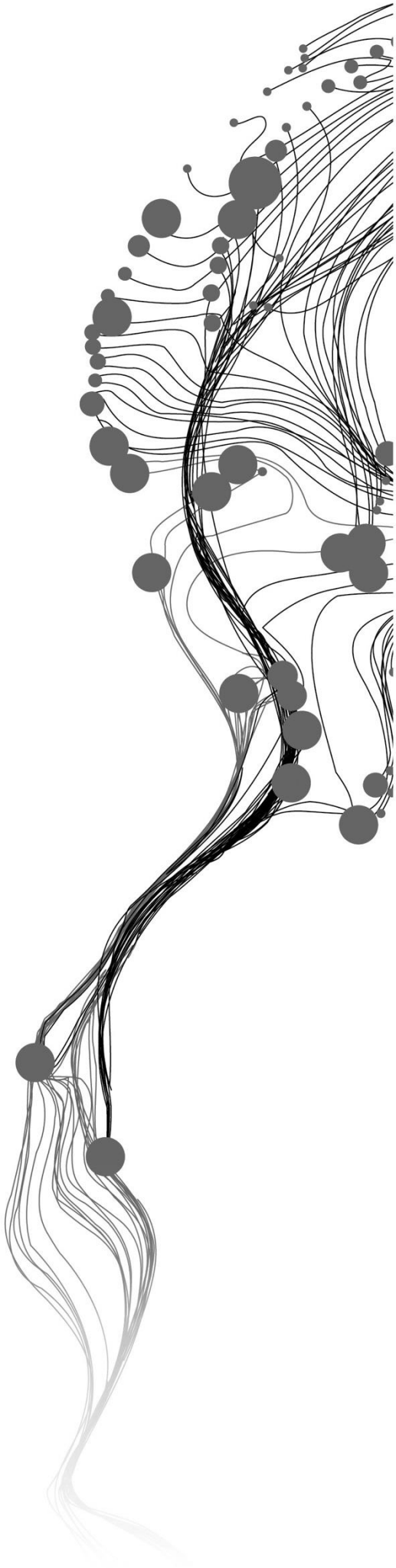
BUILDING SEGMENTATION IN OBLIQUE AERIAL IMAGERY

SHAN HUANG
February, 2019

SUPERVISORS:
Dr, F.C. Nex
Dr, M. Y. Yang



BUILDING SEGMENTATION IN OBLIQUE AERIAL IMAGERY

SHAN HUANG
Enschede, The Netherlands, February, 2019

Thesis submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: Geoinformatics

SUPERVISORS:
Dr, F.C. Nex
Dr, M. Y. Yang

THESIS ASSESSMENT BOARD:
Prof.dr.ir. M.G. Vosselman (Chair)
Dr. R.C. Lindenbergh; Delft University of Technology, Optical and Laser Remote Sensing

ABSTRACT

With the explosion of urbanization, the demand of city planning has been increased. These new challenges have to be faced in regard to the planning and environmental sustainability or urban areas. To tackle these problems, the use of more detailed and complete geographic information is necessary. "Smart Cities" aim at delivering smart and complete information thanks to digital technologies. And the building is a sub-problem and it is a key component to the reconstructing of LoD3 city modeling. In the past, the data to generate a 3D building model were almost based on terrestrial views. However, with the development of image matching technique, the airborne systems have been applied in many tasks to acquire airborne multi-view data. Compared to terrestrial views, airborne datasets can cover larger areas in the urban areas and it also been found more convenient and economic. In my study, the oblique aerial images acquired from oblique airborne systems are used as a data source for building segmentation.

With the popularity of Deep Learning, tasks in the field of computer vision can be solved in easier and effective ways. Fully convolutional network is an end-to-end and pixel-based neural network, it shows a good performance in semantic tasks to get a dense prediction result. In this study, we propose a method to apply deep neural networks to building segmentation. In particular, the FC-DenseNet and the DeepLabV3+ networks are used to segment the building from aerial images and get semantic information such as wall, roof, balcony and opening area (window and door). Due to the limited computation resource, the patch-wise segmentation is used in the training and testing process to get information at pixel level. To address the problem of imbalanced classes, the weighted loss function is used in the experiment instead of the common loss function. Softmax function is used as to reconstruct from patches to original images. Different typologies of input have been considered: beside the conventional 2D information (i.e. RGB image), we combined 2D information with 3D features extracted from dense image matching point clouds to improve the performance of the segmentation.

Experiment results show that FC-DenseNet trained with 2D and 3D features achieves the best result, IoU up to 64.41%, it increases 5.13% compared to the result of the same model trained without 3D features (59.28%). The overall accuracy is increased from 89.08% to 91.30%. Results on roof in FC-DenseNet and DeepLabV3+ using 2D combined with 3D features are better than these two models only trained with 2D information: the class accuracy increased from 91.76% to 94.61% and 92.25% to 95.78% respectively.

In conclusion, 3D features give benefit on the performance of segmentation. It can improve the performance of specific classes, for this thesis, the third component of the normal vector provides extra information to distinguish if the pixel on the same plane.

Keywords

Building segmentation, DeepLabV3+, FC-DenseNet, patch-wise, 3D features

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere thanks to my first supervisors Dr. F.C. Nex, who gives me the patient guidance and valuable suggestions. Without his support and encouragement, my thesis would have been impossible. Besides these, he has also given me much idea and methods of my thesis.

I also would like to thank my second supervisors Dr, M. Y. Yang, for critical comments to my experiment and creative ideas. And he also pushed me a lot to help me finish this thesis.

Besides, I express my sincere gratitude to all the teachers in ITC. Their lectures have broadened my scope of vision, and which is helpful to my future academic life and work.

Furthermore, I would like to thank my friends, Yiwen, Yaping. They have helped me and encouraged me a lot when I met difficulties during my thesis time. And I am also grateful to all my friends, they have given me warm encouragement and support.

Last but not least, I would like to thank my family, whose love and support are with me when I felt depressed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation and problem statement

Due to the explosion of urbanization and the increase in population in recent years, new challenges have to be faced in regard to the planning and environmental sustainability or urban areas. To tackle these problems, the use of more detailed and complete geographic information is mandatory. "Smart Cities" aim at delivering smart and complete information thanks to digital technologies. In this regard, the realization of 3D city modeling allows to interoperate and share many data in an efficient way. Different levels of city models can be then generated. City Geography Markup Language (CityGML) is considered the standard for 3D city modeling. In CityGML, building parts and accessories can be classified into four levels-of-detail from LoD1 to LoD4 (Gröger & Plümer, 2012). In LoD1, buildings are modeled in a generalized way, like parallelepipeds. In LoD2, the roof shape of the building is represented. LoD3 is a more detailed level, openings (window, door) and detailed rood structures (chimney) are added for buildings on the facades, and in LoD4, the interior (room) are represented too. Currently, the low level (LoD1 and LoD2) can be generated (almost) automatically, but this process is not feasible for LoD3. Many details such as the building components cannot be reliably extracted in an automated way and therefore, they cannot be automatically inserted into a 3D model.

The semantic segmentation of a building can be therefore considered a sub-problem of the automatic generation of virtual cities with LoD3 models. The task of the building façade segmentation is to assign each pixel of human-made structures to a semantic label such as window, balcony, and door. However, manual delineation over large urban areas is time-consuming. An automatic way for semantic segmentation of building is the unique choice from a practical point of view.

Early methods for building façade segmentation were based on an appropriate shape grammar (Gadde et al., 2018) following the predefined architectural constraints (e.g. windows are of the same size on the façade and not placed randomly; doors can be found on the first floor at street -level; the roof is above the top floor; all balconies have the same dimensions, etc.). These rules can reduce the errors of the segmentation result, but they heavily rely on prior knowledge.

Machine learning is an efficient and automated method to parse building. There are a few classifiers that can be applied to tackle this task, for example, Support vector machine (SVM), RANSAC (Boulaassal et al., 2007), randomized decision forest (Yang et al., 2012). However, these algorithms typically return noisy pixels in their segmentation results, due to the lack of neighboring information (Rahmani et al., 2017). Conditional Random Field (CRF) (Lafferty et al., 2001) is also a popular method to refine the output of the classifier to improve the accuracy of the result.

Recently, deep learning outperformed the traditional method (SVM, RF) in terms of accuracy and robustness. Convolutional Neural Networks (CNNs) have shown a good performance and high efficiency in image recognition, object detection, semantic segmentation. (Long, et al., 2015) proposed an end-to-end network using fully convolutional architecture-FCN outperforming previous algorithms in the task of semantic segmentation. Compared to the classical convolutional neural networks, FCN replaces the final fully connected layer with a convolutional layer and outputs a pixel-wise labelled image instead of a classification score. FCN accepts arbitrarily sized images as the input and recovers shrunken images after a

series of convolutional layers thanks to the deconvolutional layer (Garcia-Garcia, et al., 2017). However, training models from the beginning is time-consuming work and cannot produce good results with random initialization. Thus a common trend in the segmentation is to apply transfer learning (Yosinski et al., 2014) fine-tuning the pre-trained classification networks, where pretrained models can be used as the starting points to speed up the training process.

Building data can be captured from multiple platforms. Compared to the terrestrial data, the airborne oblique imagery is more productive in the urban area as it can cover larger areas and it can acquire the same object from different images. Compared to aerial nadir images many more details can be then acquired and used to further improve the generation of 3D models (Xiao et al., 2012). The cost of oblique images is also lower than other terrestrial methods.

In this study, the use of FCN for façade segmentation is investigated. In particular, two Deep Neural Networks, namely FC-DenseNet and DeepLabV3+ are adopted to parse buildings from oblique images captured by airborne systems.

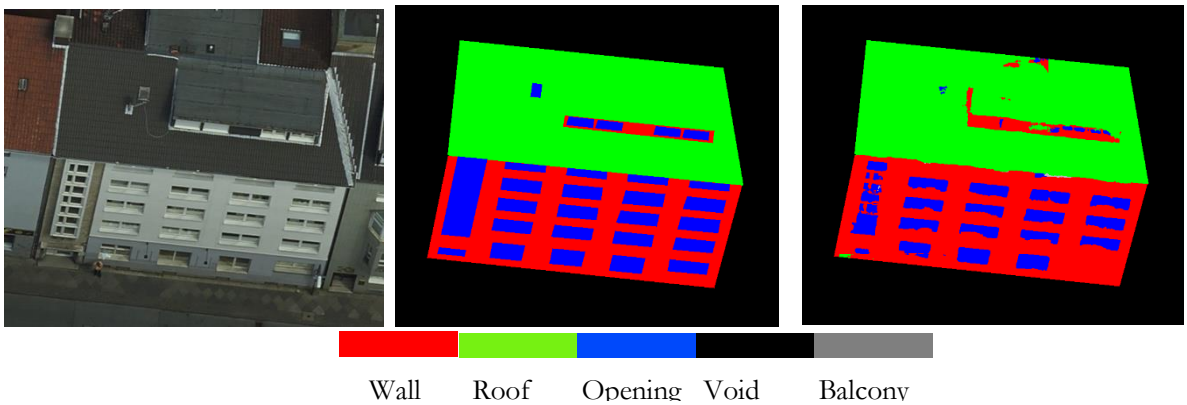


Figure 1: Examples of our task, from left to right are Original image, Ground truth, Result from FC-DenseNet trained with 2D and 3D feature.

The contribution of the study is that the input of the network includes not only 2D image information (RGB), but also point clouds to provide extra 3D information (the third component of the normal vector) for improving accuracy. For the training process, the weighted loss function is used to solve the problem of imbalanced classes. We also use patch-wise segmentation (splitting original images into small patches for training) in our task to keep the original image sizes and we choose the maximum probability in the score map instead of their direct combination to delineate the negative effect when reconstructing from small patches to original images.

1.2. Research identification

1.2.1. Research objectives

At the moment there is no way to automatically segment building façade so LoD3 is not feasible from a practical perspective. The main aim of this study is to classify the building to provide the information allowing to generate a LoD3 model. The oblique images are captured from IGI Pentacam system. It's installed on an airborne system. The method is based on convolutional neural network and can get dense prediction results. There are two models will be used in the experiment: Fully convolutional DenseNet and DeepLabV3+. And patch-wise segmentation will be implemented, the original image will be split into small patches for training. The objective can be divided into the following sub-objectives:

1. Adjust the neural networks to this specific task of building classification and exploit both 2D and 3D information.
2. Compare different architecture and define the most relevant elements in their architecture to perform a high-quality facade.
3. Test the existing ones and add a 3D feature to improve performance for this task.
4. Assess the accuracy of the achieved results and find the most proper parameters.

1.2.2. Research questions

Adapt the neural networks to the building classification specific task

1. What is the code already available for the building classification?
2. Which kind of 3D information can be used in the network?
3. How to use the extra 3D information for this task to get improvements on segmentation results?

Compare different architecture and define the most relevant elements in their architecture to perform a high-quality facade.

1. What are the existing architectures that can be used for this task?
2. What are the parameters that seem to influence more the quality of the results?

Test the existing ones and add a 3D feature to improve performance for this task.

1. Which part can be modified to improve the accuracy of the results?
2. How to choose the parameters in the networks for 2D combined with 3D?

Assess the accuracy of the achieved results and find the most proper parameters.

1. Which is the best metric can be used to evaluate the result?
2. Which network has a better performance compared to others?
3. Does 3D give any benefit for results?

1.2.3. Innovation aimed at

This study aims to solve an open problem like the building segmentation. The innovation of the presented in this work is given by the following aspects:

- Different networks are compared in this task, FC-DenseNet and DeepLabV3+, based on only 2D information and combining 2D and 3D in the task of building segmentation.
- Furthermore, we use patch-wise segmentation instead of image resizing to avoid distortions. The original images with different resolutions will be split into small patches with the same resolution as the input of neural networks.
- In the process of reconstruction from small patches to original images, the common way is just to combine two images together, and the result of border regions has a poor performance with gaps or confused pixels. Instead of that, overlapping splitting and Softmax function have been implemented in this process to improve the performance in border regions.
- Instead of a commonly used loss function, cross-entropy, the weighted cross-entropy loss function is used for each class to deal with the imbalance data problem.

1.3. Thesis structure

Chapter 1 gives an overall introduction to this thesis. It explains the motivation and current problem waited to be addressed in the study. In chapter2, the related work will be introduced firstly, it gives a brief view of past works. And a basic background of Neural Networks reviewed in this chapter, including basic concepts and some promising networks. In chapter 3, the background of the neural network is introduced to make easier to understand this thesis. Chapter 4 mainly explains the methodology used in this thesis. In chapter 5, the experiment details will be explored, and the results of the study are shown in this chapter, different networks are compared and there is a short discussion of results. In chapter 6, the conclusion and a short recommendation for further work are given, research questions are answered in this chapter.

2. LITERATURE REVIEW

In this chapter, it briefly reviews the approaches for the semantic segmentation researches related to this study. Firstly, traditional methods are given in section 2.1, and followed by an introduction to deep learning in section 2.2. The existing promising neural networks for semantic segmentation will be reviewed.

2.1. Traditional methods

Many kinds of research have been working on the façade detection and classification. These works have aimed to estimate the position and size of various structural (e.g., window, door, roof) and non-structural elements (e.g., sky, road, building) exploiting their shape or their appearance on the given images (Fröhlich et al., 2010). The previous works can be classified into different categories according to the data source: image-based (2D) and laser-based (3D) algorithms. These can be then subdivided into the airborne and the terrestrial according to the used platforms.

2.1.1. 2D information

(Cohen et al., 2014) presented a method using a dynamic programming algorithm to praise the façade of the building and applying the hard-architectural constraints. (Gadde et al., 2015) have used the learning split grammars from annotated images to perform the pixel-wise classification. In (Delmerico et al., 2011) a method has been proposed using three main steps: discriminative modeling, candidate plane detection through PCA and RANSAC, and energy minimization of MRF potentials refining the result with the plane fitting. (Martinović et al., 2012) shows a three-layer architecture where the semantic segmentation gives the low-level information, middle-level is based on a pairwise multi-label Markov Random Field (MRF) solved by a graph-cut algorithm about objects in the facade, and top-level is according to the architectural knowledge. Randomized decision forest (RDF) is also an excellent classifier to classify the building façades. (M. Y. Yang & Wolfgang, 2011) demonstrated an approach of region-wise classification by RDF and local features refining the result with the conditional random field (CRF). They trained an RDF on the labeled data and split them by a decision tree learning algorithm. (K. Rahmani et al., 2017) proposed a method using a Structure Random Forest for façade labeling and get a good performance result on the ECP and Graz façade datasets. Fully connected CRFs can model long-range spatial dependencies and make use of contextual information. (Li & Yang, 2016) used the fully connected CRF (all nodes are connected in pairs) as the basic framework for the façade parsing task. They chose the trained Textonboost as the unary classifier and obtained maximum posterior marginal (MPM) results by filter-based mean-field approximation inference. The use of oblique images is a way to capture multi-views of building facades. In this regard, (Tu et al., 2017) extract the feature following local symmetrical and using a sliding window to determine the location of the local symmetry feature point.

2.1.2. 3D information

Also, a laser system can generate the point cloud to provide extra information on three-dimensions. (Boulaassal et al., 2007) applied RANSAC algorithm on TLS data to automatic segmentation. After two years, (Boulaassal et al., 2009) proposed a new adaptive RANSAC algorithm to extract the planar and achieve the extraction of contour points composing the boundary of each plane to be applied in the further work on the 3D modeling. (B. Yang et al., 2013) proposed a coarse-to-fine method to parse building facades from mobile LiDAR point clouds. The method first covert the point cloud into images,

and regard it as an image-based problem. (Brostow et al., 2008) first used classification combined with the spare 3D point cloud from Structure from Motion. In (Martinovic et al., 2015), they did not use 3D information as the reference for the 2D classifier, they designed a 3D pipeline and proposed a weak 3D architectural principle for façade parsing. (Gadde et al., 2018) also applied not only 2D information (images) but also 3D information (point cloud) into the façade task, the result shows the combination can be improved the IoU performance. The features they utilized extract from the 3D point cloud, such as the mean RGB color values, LAB values, the estimated normal at the 3D point, the distance between the point and an estimated facade plane. (Tutzauer & Haala, 2015) proposed a radiometric segmentation method that using point clouds from dese image matching with imagery. (Fritsch et al., 2013) is focused on using point clouds from dense image matching to model facade structure by formal grammars.

2.2. Neural networks for semantic segmentation

Many remote-sensing applications can be also achieved by using deep learning, such as hyperspectral image analysis, interpretation of SAR images, interpretation of high-resolution satellite images, multimodal data fusion, and 3D reconstruction (Zhu et al., 2017). (Kujtim Rahmani & Mayer, 2018) mainly introduced the Region Proposal Network (RPN) based on a Convolutional neural network to generate the prior information for the building elements, such as window, door, balcony with their probability, and then put it into the Structured Random Forest as the input.

2.2.1. AlexNet

AlexNet was one of the first deep neural networks architecture to solve the classification task. (Krizhevsky et al., 2012) proposed it and won the ILSVRC-2012 (ImageNet Large-Scale Visual Recognition Challenge) with a top-5 error of 15.3%. The architecture includes five convolutional layers, rectified linear units as nonlinearity functions (ReLU), a max-pooling layer, three fully-connected layers, and dropout layers. And the model can be trained on GPUs, it reduces the time of training and makes available to solve the task on a large data set.

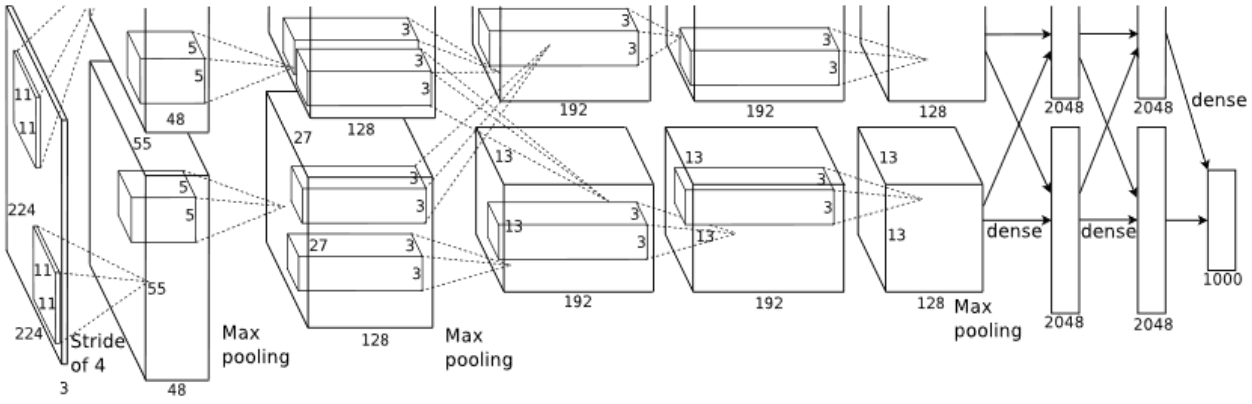


Figure 2: The structure of AlexNet from (Krizhevsky et al., 2012).

2.2.2. VGG

VGG network is one of the most influential networks that demonstrates the importance of depth in the classification task. It proposed by the Visual Geometry Group from the University of Oxford in (Simonyan & Zisserman, 2015). And one of them that VGG-16 achieved the accuracy of 92.7% on the

ILSVRC-2013. It proves that the depth of the networks can improve performance. From Figure 3, we can see that it uses 3 × 3 filters on top of each layer with stride 1 instead of larger ones in AlexNet, and 2 × 2 max-pooling layers. The parameters of the model are less than before, making the model easier to be trained. The visualization of the network structure is shown in Figure 4.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 3: The configuration of network from (Simonyan & Zisserman, 2015).

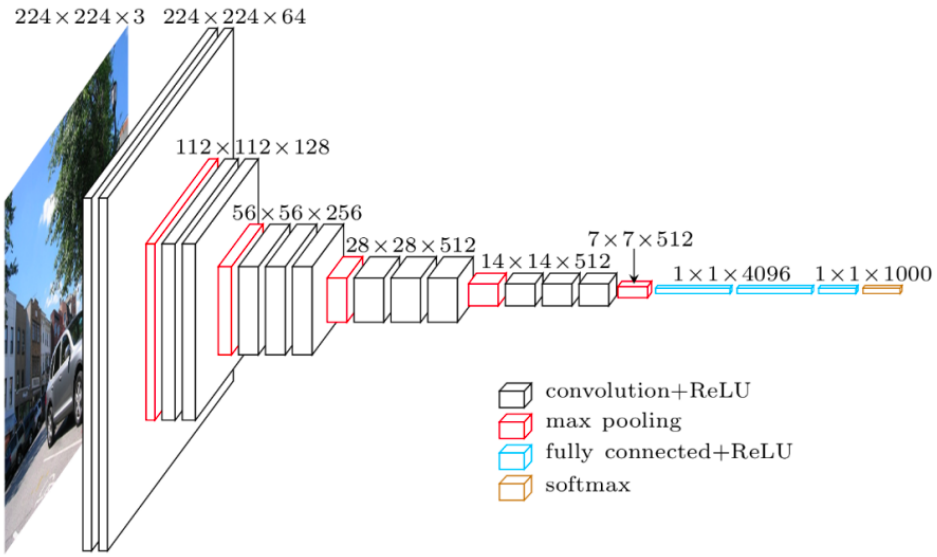


Figure 4: An illustration of the structure VGG-16 from (Jordan, 2018).

2.2.3. Resnet

Microsoft proposed Reset (He et al., 2016) on the ILSVRC-2016, and it won the challenge with the accuracy of 96.4% in classification. And the network, 152 layers, is much deeper than previous (AlexNet 8 layers, VGG 19 layers, GoogLeNet 22 layers). The residual blocks are also first to be introduced by adding shortcut connections to improve efficiency when training a deep model. The structure of the residual block is shown in Figure 5. Batch normalization is heavily used in the network. The architecture also removes fully connected layers at the end of the network. The comparison of VGG and ResNet is shown in Figure 6.

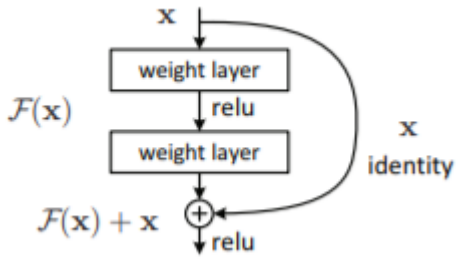


Figure 5: Residual block from (He et al., 2016).

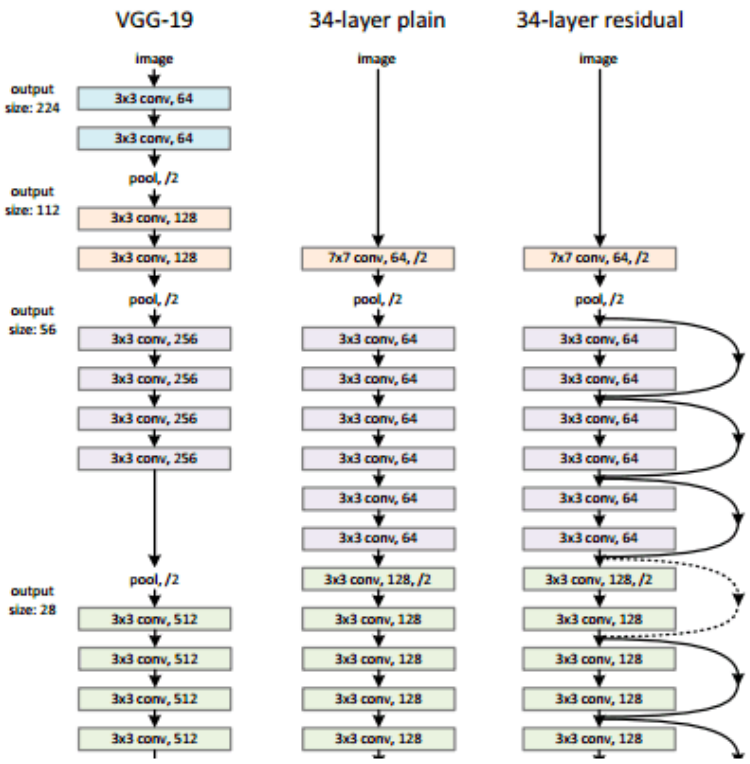


Figure 6: The structure of ResNet networks (He et al., 2016).

2.2.4. Fully convolutional network

(Long et al., 2015) proposed the first Fully Convolutional Networks (FCN) that is an end-to-end deep neural network for semantic segmentation. It makes dense predictions for semantic segmentation using the arbitrary size of the input by adding up-sampling layers to restore the spatial resolution of the input. A skip connection is also added to the networks. A CNN can be converted into FCN in few steps: First, replacing all fully connected layers with a 1 x 1 size of the convolutional layer, the process is depicted in Figure 7. Second, adding a deconvolutional layer to recover the spatial information which has been down-sampling by the pooling layers. Therefore, the existing models of CNNs can also be used in FCN. (Liu et al., 2017) applied FCN into the 2D façade paring problem, they proposed a symmetric regularization term and to train the neural network with a novel loss function and boosting the performance with the post-processing based on object detection. (L. C. Chen et al., 2018) proposed an idea combined with the deep convolutional neural networks based on ResNet and fully-connected conditional random fields.

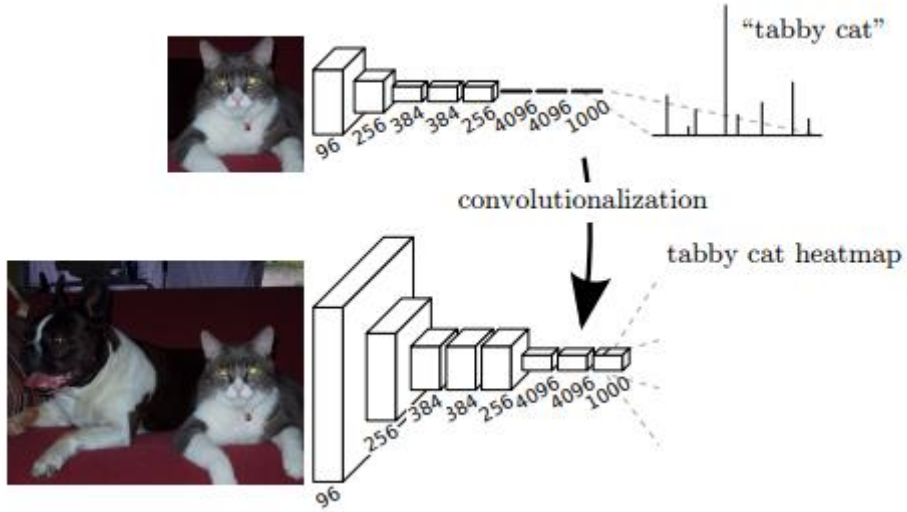


Figure 7: An illustration of transforming fully connected layers to convolutional layers (Long et al., 2015).

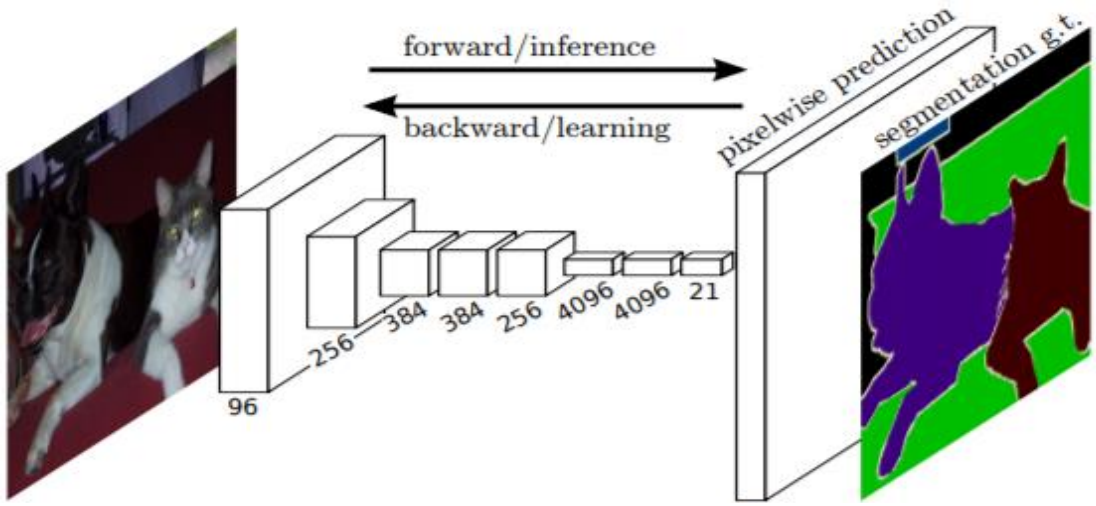


Figure 8: The structure of FCN (Long et al., 2015).

2.2.5. DenseNet

Densely Connected Convolutional Networks (Huang, et al., 2017), continued to increase the depth of neural networks. Figure 9 shows an example of DenseNet structure with dense blocks. The advantage of DenseNet is that it has fewer parameters than traditional convolutional networks. It makes the neural network easier to be trained. With the depth going deeper, the problem of gradient vanished is arisen. Because the path from the input layer to the output is too far. To solve this problem, the solution is that each layer simply connects with each other layers in DenseNets, it can be seen in Figure 10.

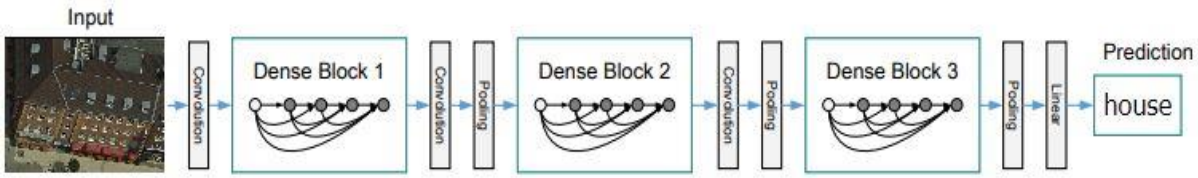


Figure 9: An example of DenseNet with three dense blocks.

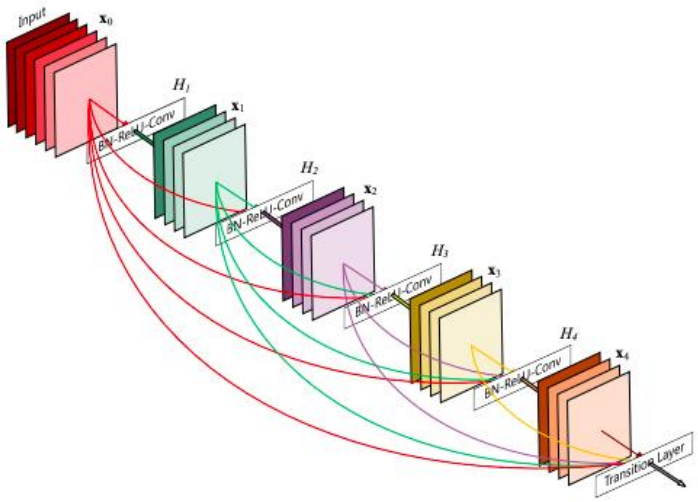


Figure 10: The structure of DenseNets from (Huang, et al., 2017).

2.2.6. DeepLab

Semantic segmentation is an end-to-end task, it needs to achieve a pixel-level result. Pooling layers will decrease the size of the input, but it will lead to a loss of spatial information. DeepLab networks by (Chen et al., 2018) based on VGG networks, and replace the final fully connected layer with a convolutional layer. To keep the spatial information, the last two pooling layers are removed in DeepLab networks. Instead of pooling layers, dilated convolution (also called atrous convolution) is implemented in DeepLab. It has the same effect on increasing receptive field by changing atrous rate (sampling rate). Figure 11 shows that an illustration of dilated convolution, Where the kernel size is 3 by 3, the atrous rate is 2, and the receptive field increases from 3 to 5.

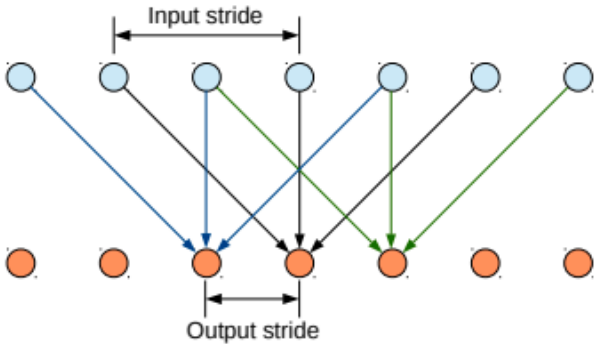


Figure 11: An illustration of hole algorithm, kernel size =3, input stride=2 and output stride=1 (Chen et al., 2018).

Another contribution in DeepLab V1 is that Conditional Random Forest is applied as the post-process to refine the noisy segmentation results. The whole process is shown in Figure 12. However, in DeepLab V3 (Chen et al., 2017) discards the use of post-processing CRF to refine the result, and adds the image level features to the ASPP structure. The results of segmentation are better than DeepLabV2 with CRF.

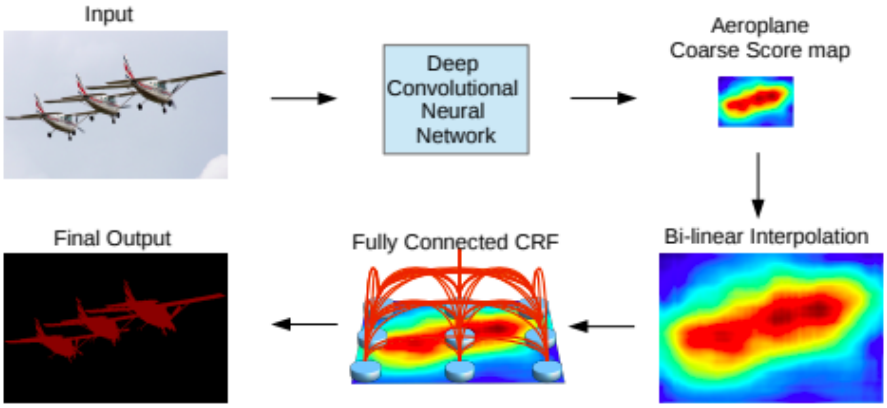


Figure 12: Model illustration (Chen et al., 2018).

3. BACKGROUND OF NEURAL NETWORKS

Some basic knowledge of Neural Networks will be introduced in this chapter; it will help to understand the study. A brief introduction to convolutional neural networks is given; it includes the architecture, the components of neural networks and operations.

3.1. Introduction to convolutional neural networks

Recent years, deep learning becomes a popular method in the field of computer vision. It has proven to have a good performance for tasks like object detection, classification, and segmentation. A concept of Convolutional Neural Networks will be given, and basic structures will be introduced as backgrounds.

3.1.1. The architecture of a CNN

The usual architecture of Neural Network is composed by three parts (see Figure 13): input layers, the hidden layers, which values are not visible, and output layers, which only has one node. Convolutional Neural Networks have 3 dimensions: width, height and depth. For a color image, the three channels contain information for the red, green and blue values. Figure 14 shows an illustration of ConvNet (Stanford University et al., 2016). The input of CNNs are images instead of one-dimensional vector of inputs. From this figure, we can see, the red input layers refer to an image, the width and the heights would be the dimension of the input image. The depth means channels including red, blue and green channels. At the end of the CNN, there would be a $1 \times 1 \times C$ vector, which refers to the class score, C refers to the number of classes.

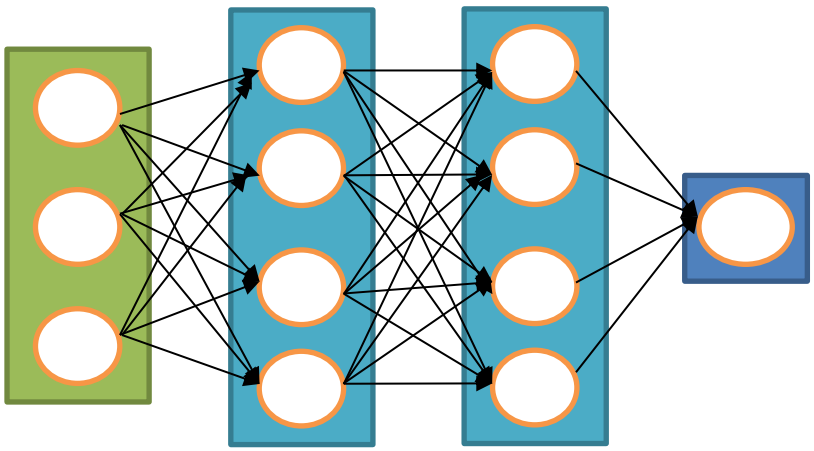


Figure 13: An illustration of the architecture of regular Neural Network. The left one is the input layer, the middle two are hidden layers, the right one is the output layer.

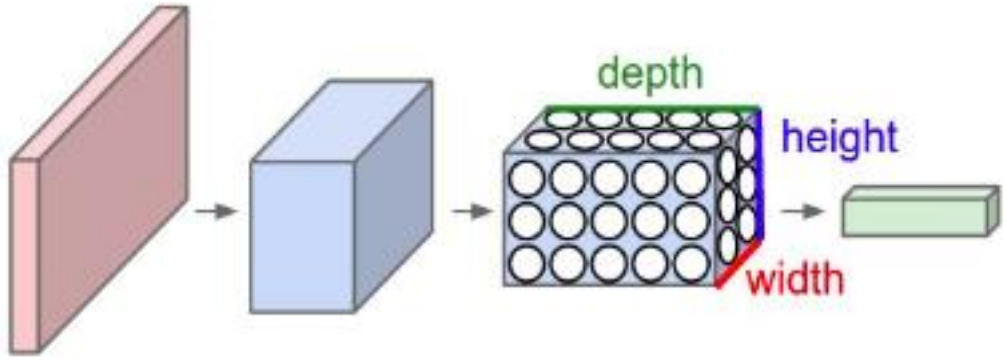


Figure 14: An illustration of ConvNet from (Stanford University et al., 2016).

In Figure 15, it shows an example of one neuron with four inputs. x refer to the input, w is the weight and $\sum$ is the weighted sum of inputs. As is shown in Equation 3-1. After that, the output is activated by an activation function.

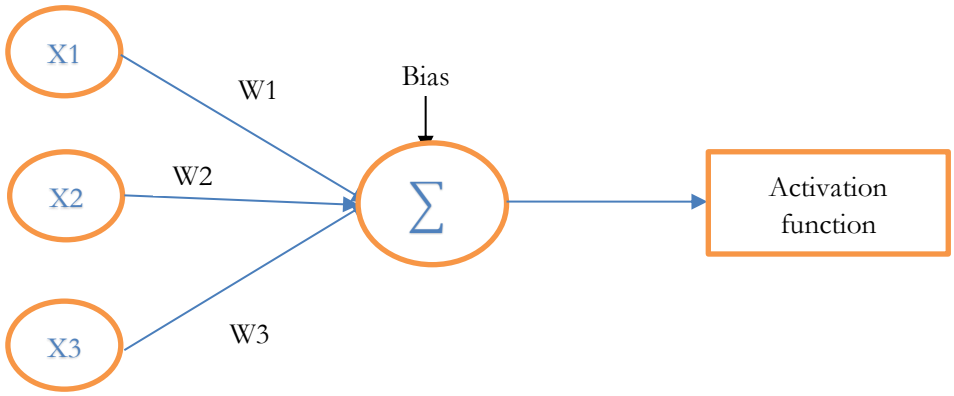


Figure 15: An example of one neuron with four inputs, X refers to the input, W refers to the weight.

Sigmoids Activation function was commonly used in the classification task, shown in Equation 3-2. But it will lead a vanishing gradient problem that when training a deep network, the gradient tends to vanish ("0"), and the network cannot continue to learn (Bengio et al., 1994). To avoid and rectify this problem, another activation function was introduced, it defines as a rectified linear unit (ReLU). It defined as the positive part of the function, as in the following Equation 3-3.

$$f(x) = w_1x_1 + w_2x_2 + w_3x_4 + b = W^Tx \quad \text{Equation 3-1}$$

$$f(x) = \frac{1}{1+e^{-x}} \quad \text{Equation 3-2}$$

$$f(x) = \max(0, x) \quad \text{Equation 3-3}$$

3.1.2. Convolutional layer

The convolutional layer is the core part of a neural network to extract features from images. In Figure 16, the input is an image with $32 \times 32 \times 3$ (32 width, 32 height, and 3 color channels) (Stanford University et al., 2016). There is a sliding filter across the whole image to do the dot product computation. Each filter across all positions on the images and generate a 2-dimension feature map. For the size of each filer (also called receptive field) is predefined. The width and height are generally small than original inputs. The stride is defined as the distance of each step filter shift. When the stride is "1", it means the filters move 1 pixel during each step. If the stride is "2", it means that the filters will move 2 pixels to the convolution operation. It can be increased depends on demand. Sometimes, the original resolution of images cannot fit for the filter crossing the whole image. To make sure that filters can extract features from borders, zero-padding (adding zeros to the border of the image) is applied to the original images.

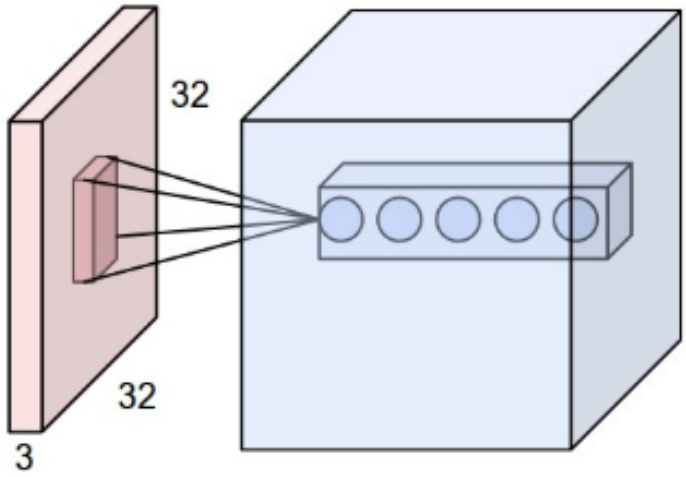


Figure 16: An example volume of input and an example volume of neurons from (Stanford University et al., 2016).

The final output is stacked by these feature maps along the depth dimension. The following Equation 3-3 can compute the size of the output, W is the size of the input, F is the size of the filter, P refers to the size of zero-padding and S means the stride of the shift.

$$(W - F + 2P)/S + 1 \qquad \text{Equation 3-3}$$

3.1.3. Pooling layer

To reduce parameters in the neural networks, pooling layers is commonly applied to insert following the convolutional layer. After passing through the pooling layer, the spatial size of feature maps also will be reduced. Therefore, the operation of pooling layers in neural networks is called sub-sampling. Figure 17 depicts a commonly used pooling way, max-pooling. Firstly, 2 by 2 filters selected as a window to do the sliding operation over the feature map, and stride 2. In each sliding window, only the maximum pixel will be kept. For example, in the red window, only "6" kept in the output feature map. After four times the same operation, the output is a 2 by 2 feature map, the depth is unchanged, pixels kept "6", "8", "3" and "4".

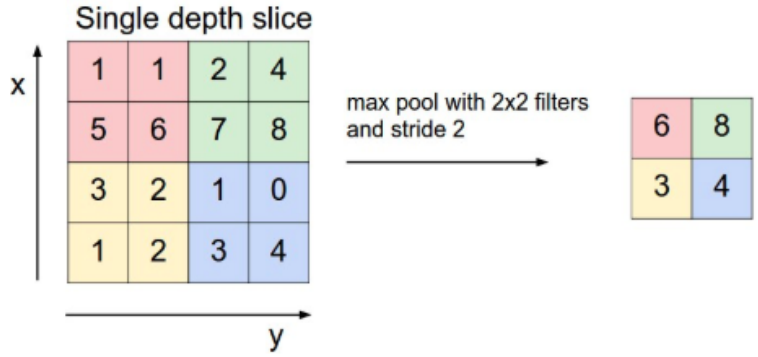


Figure 17: An example of max pooling from (Stanford University et al., 2016).

Similar to zero-padding, there is an equation to compute the size of the output:

$$W_2 = \frac{W_1 - F}{S} + 1 \qquad \text{Equation 3-4}$$

$$H_2 = \frac{H_1 - F}{S} + 1 \qquad \text{Equation 3-5}$$

Where W means the width, H means the height, F means the filter, and S is stride.

3.2. Optimization and Regularization

3.2.1. Loss function

The loss function is a tool to measure the performance of a classification model. The common loss function widely used in classification tasks is the cross-entropy function (or called log loss function). The output is a probability distributed between 0 and 1. When a value of log loss equals to 0, it means that the model has a perfect performance. The equation of a binary cross-entropy loss function is shown in Equation 3-6. N refers to the total amount of pixels, y means the ground truth ("0" stands for false, "1" refers to true) and p refers to the probability of prediction.

$$Loss = -\sum_{i=1}^{N} [y_i \, log \, p_i + (1 - y_i) \, log(1 - p_i)] \qquad \text{Equation 3-6}$$

For our task, it is a multiclass task instead of binary classification. For a multiclass classification, the equation is shown in Equation 3-7. Where N is a total amount of pixels, y refers to the ground truth in One-Hot format (as explained in 3.2.2), p refers to the prediction and K is the number of class. $p_{i,k}$ means the probability of i^{th} pixel belongs to the k^{th} class.

$$Loss = -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{K-1} y_{i,k} \, log \, p_{i,k} \qquad \text{Equation 3-7}$$

But the main disadvantage of the common cross-entropy loss function is that it cannot deal with the imbalance problem of classes. To this end, the weighted loss function was chosen for our task to ensure that we can get a better result. And it will be introduced in Section 3.

3.2.2. Softmax function

The Softmax function is usually added as the final layer to achieve multi-class classification. It is a form of logistic regression that converts a number of score values to values following probability distribution between 0 and 1, whose total sums up to 1. Where, e^{y_i} is the scores of the input in the form of one-hot encode, i with the length equal to the number of classes J . In this task J equals to 4, i=0, 1, 2, 3.

$$S_i = \frac{e^{y_i}}{\sum_j e^{y_j}} \qquad \text{Equation 3-8}$$

3.2.3. One-Hot Encoding

In this study, RGB values correspond to the predefined classes (0,1,2,3,4). Table 1 is an illustration of One-Hot encoding. If a sample belongs to a class, we will mark it as "1", if not, mark it as "0".

In some of machine learning tasks, categorical data prepared for the experiment instead of numeric data. Categorical data is also called nominal. For example, a value named "animal" includes "tiger" and "lion", "color" includes "pink", "black" and "green". "place" includes "first", "second" and "third". The problem of categorical data is that some of the algorithms cannot work with categorical data directly and there are some nature relationships in the categorical data, such as a nature ordering, it will have an impact on the result. To solve this problem, a common way is to convert categorical data to numerical data by one-hot encoding (Brownlee, 2018).

	Class 0	Class 1	Class 2	Class3
(0,0,0)	0	0	1	0
(255,255,255)	0	1	0	0
(0,255,0)	1	0	0	0

Table 1: An illustration of One-Hot encoding.

3.2.4. Overfitting

Overfitting is a common problem in machine learning. Overfitting is that the model includes more terms than are necessary or the approach we used is more complicated than needed (Douglas M. Hawkins*, 2004). It means that a model is trained too well, and it can perform well on training data, but bad on test data. Overfitting happens when the model chooses the best solution to fit for the specific situation, not for the overall. It will lead that the model cannot fit for the new data.

There are some ways can improve the ability of the generalization for models to avoid overfitting:

1. Add more data for training
2. K-fold Cross-validation. K refers to the number of the group that datasets will be split into randomly. Each time, a fold is selected for validation and K-1 folds for training (Kohavi, 1995).
3. Lower the learning ability of the model.

For the third way that lower the learning ability of the model, there are a few methods that can be implemented in the neural network. Dropout is a simple way to prevent overfitting. The term "dropout" means that units in hidden and visible are dropped in a neural network (Srivastava, N. et al., 2014). We can set a fixed probability p independent of units, where p can be set between 0 to 1. The probability normally is chosen closer to 1 than to 0.5. An illustration of dropout is shown in Figure 18.

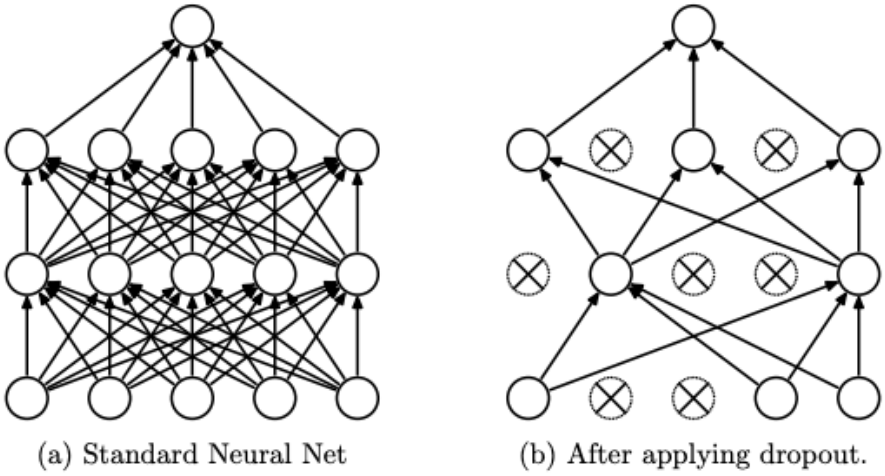


Figure 18: Left: Standard Neural Network. Right: Appling dropout to the neural network (Srivastava, N., et al., 2014).

3.2.5. Batch Normalization

With the depth of networks deeper, it is more difficult for neural networks to be trained. In order to tackle this problem, we have to do some pre-processing to the input data. Due to the parameters change after passing through each layer, the input of each layer usually is not able in the same range. To make it easier to train the network, normalization is a method to resemble a normal distribution. However, there is a certain drawback that internal covariate shift. It happens in the internal layers due to the change during the training process. This will increase the time of the training process because it needs to take a longer time for each layer to adapt to the new distribution.

(Ioffe & Szegedy, 2015) proposed a normalization method named Batch Normalization to normalize the inputs of each layer by mini-batch in a neural network. It computes the mean and variance of the layers input. The equations that Batch mean and Batch variance are shown in the following:

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} x_i \qquad \text{Equation 3-9}$$

$$\sigma^2{}_B = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 \qquad \text{Equation 3-10}$$

Then, mean and variance are used to normalize the inputs, where ϵ is a small number, epsilon to prevent divide by zero:

$$\overline{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma^2{}_B + \epsilon}} \quad \text{Equation 3-11}$$

The output is obtained by scaling and shifting the previous normalized inputs, where γ and β are learned during training with the weight parameters:

$$y_i = \gamma \overline{x_i} + \beta \quad \text{Equation 3-12}$$

4. METHODOLOGY

In this chapter, the method used in this experiment will be introduced. There are four predefined classes, including wall, opening, balcony, and roof. More details and examples are shown in chapter 5. The developed methodology can be divided into a sequence of steps described in the following sub-sections. In section 4.1, how to do the patch-wise segmentation. In section 4.2, Softmax function is defined. In section 4.3, the input in the neural network are explained, 2D information, 3D feature and the combination of 2D information with the 3D feature. In section 4.4, how to solve the imbalance class problem. In section 4.5, there are two main experiments implemented in this study. The first one is based only on 2D information. The second one is based on 2D information combined with 3D feature extracted from point clouds. There are two promising networks implemented in this experiment, FC-DenseNet, and DeepLabV3+. Figure 19 demonstrates an overview workflow of the whole experiment:

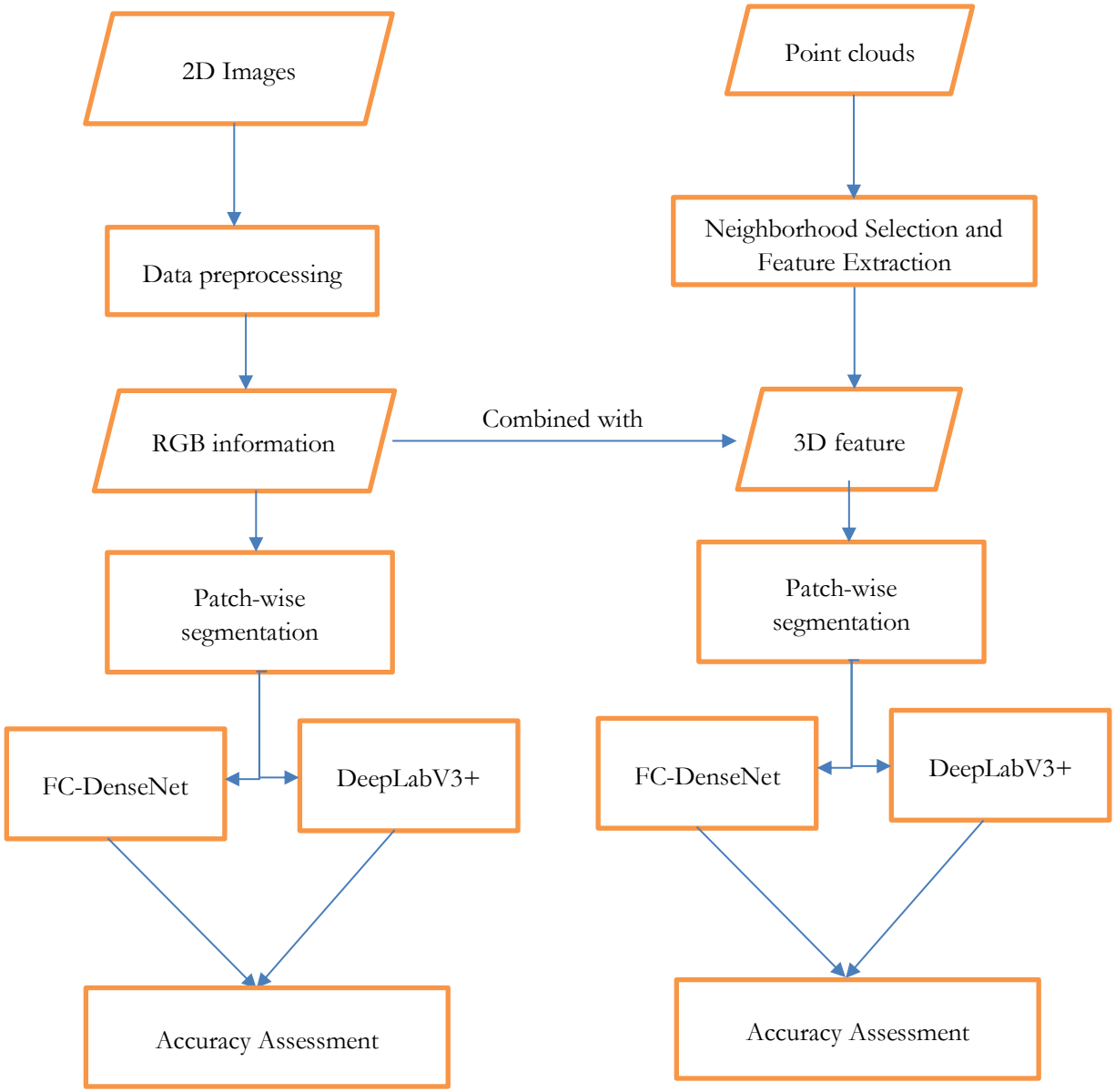


Figure 19: The workflow of the method

4.1. Patch-wise segmentation

In this task, the data have different resolutions. Due to the limited memory source of the GPU and efficiency the patch-wise strategy has been adopted to train our neural network. Compared to image resizing, patch-wise segmentation can keep the contextual information and keep the original shapes of images, without any distortion. First, in the training process, the images will be split y small patches (320×320). To get a better performance of the border, we take 50% size of each patch as the overlapping region to deal with the gap between adjacent images. If the image is not divisible by 320, it will be padded with zero first. Figure 20 shows an example of splitting strategy.

Figure 20: An illustration of patch-wise segmentation.

In the testing stage, the original images are then reconstructed from small patches using a fusion strategy. The neural networks give a proper probability distribution for each pixel by $softmax$ function.

In the overlapping region, the probability of each pixel chooses the maximum Softmax score between two regions, s_1 and s_2.

$$P(i) = argmax(s_{i1}, s_{i2}) \quad \text{Equation 4-1}$$

4.2. The input of Neural Networks

The different inputs of the neural network are introduced in this section. 2D information, 3D feature, and 2D information combined with the 3D feature.

4.2.1. 2D information

For 2D information, the RGB value is considered as the color information in this experiment. There are three channels in each image, red, green and blue. The original images are splitting into small patches, 320 by 320 pixels, for training.

4.2.2. 3D feature extraction

The third component of the normal vector extracted from the local neighborhood is the 3D feature involved in convolutional neural networks. The 3D feature provides extra information to distinguish confused pixels. This can tell whether surfaces are horizontal, vertical or slanted. The normal vector is derived from a cluster of neighboring points which can be selected by different searching strategies and searching ranges. This experiment uses 'K-nearest neighbors' as the searching strategy (Weinmann et al., 2014) and pick 100 neighboring points to calculate the normal vector for each point, where K value is defined as the optimal value that makes sure the range large enough and the noise in the data is minimized. The value is defined after testing several times (20,100,500). Here, geometric Feature Extraction (Weinmann et al., 2015) as the tool for extracting low-level geometric features. In this study, the vertical component of the normal vector is used in the experiment. Figure 21 shows an illustration of the effect of this 3D feature in point clouds. The point of the roof almost in dark blue and wall almost in red. They have an apparent difference in angle between normal vector and z-axis.

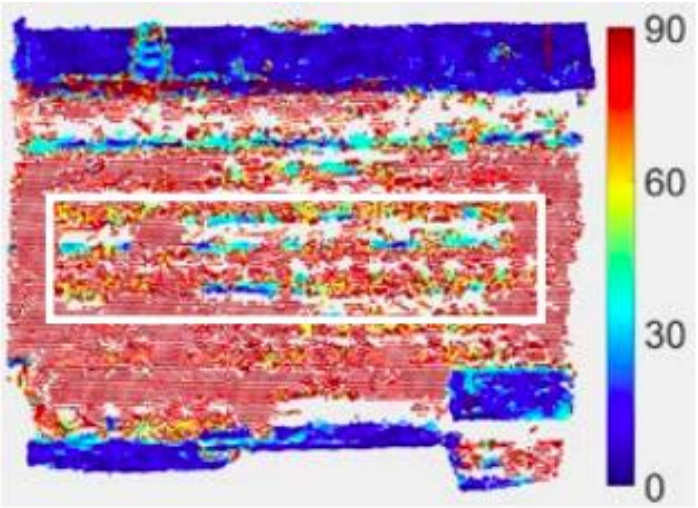


Figure 21: An illustration of the effect of normal vector in point clouds from (Lin et al., 2018)

4.2.3. Feature combination

Our networks are based on 2D CNN architectures: the 3D feature is therefore projected into image space and taken as the fourth channel of the network input. The projection to oblique airborne images is based on P-matrices which are obtained after dense matching point cloud generation in the Pix4D software. During the projection, one point can be associated with image patches of different sizes: pixels within the same patch share the same 3D feature.

The equations are shown in the following: Where (x, y, z) are pixels in 3D world coordinates, and (u, v) are pixels in 2D images. Pmatrix is the output of each image, which generated by Pix4D mapper software. It is a 3 by 4 matrix that consists of interior parameters (e.g. focal length, principal point related to the image coordinates) and exterior parameters (e.g. rotation and translation related to the real world and camera coordinate systems).

$$(x, y, z)^t = Pmatrix * (x, y, z, 1)^t \quad \text{Equation 4-3}$$

$$\mathrm{u} = \frac{x}{z};\ v = \frac{y}{z}; \quad \text{Equation 4-4}$$

When multiple points are projected to the same patch, the averaged feature value is assigned to the patch. In real experiments, small patches leave voids in image space, while large patches reduce the void percentage but, at the same time, lead to coarse features that are insufficient to provide detailed information. To avoid void space in the image, and keep detailed information, the optimal patch size is set as 4 pixels by 4 pixels (Lin, 2018). Figure 21 shows an image with different patch size when project back to 2D.

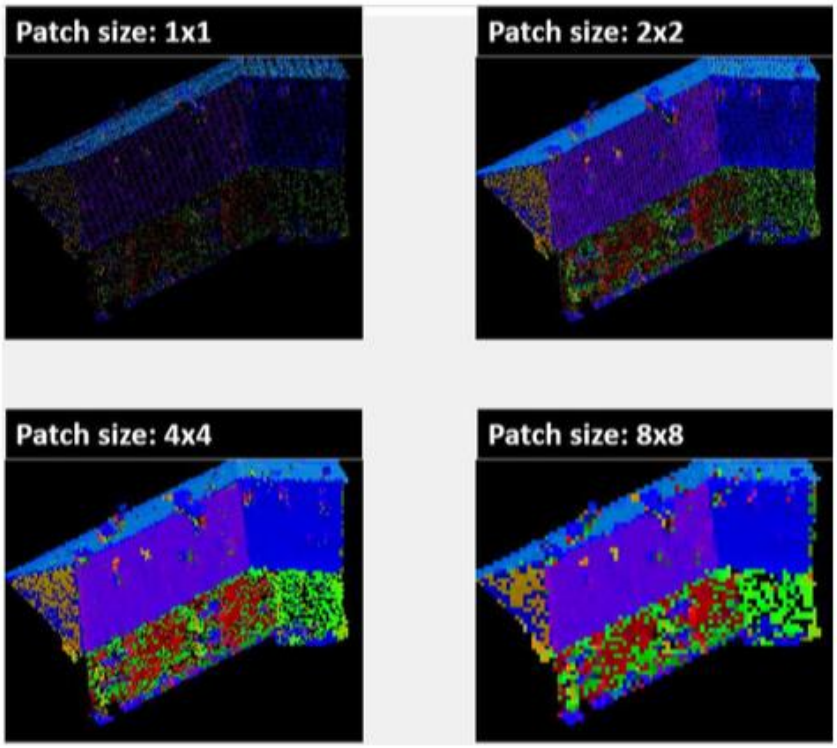


Figure 22: An illustration of the projected images with different patch size from (Lin, 2018).

4.3. Class imbalance

Classes with fewer pixels are likely to cause the imbalance problem during the training. In this study, for some classes, such as balcony, the number of pixels is much less than other classes., which can be seen from Figure 23.

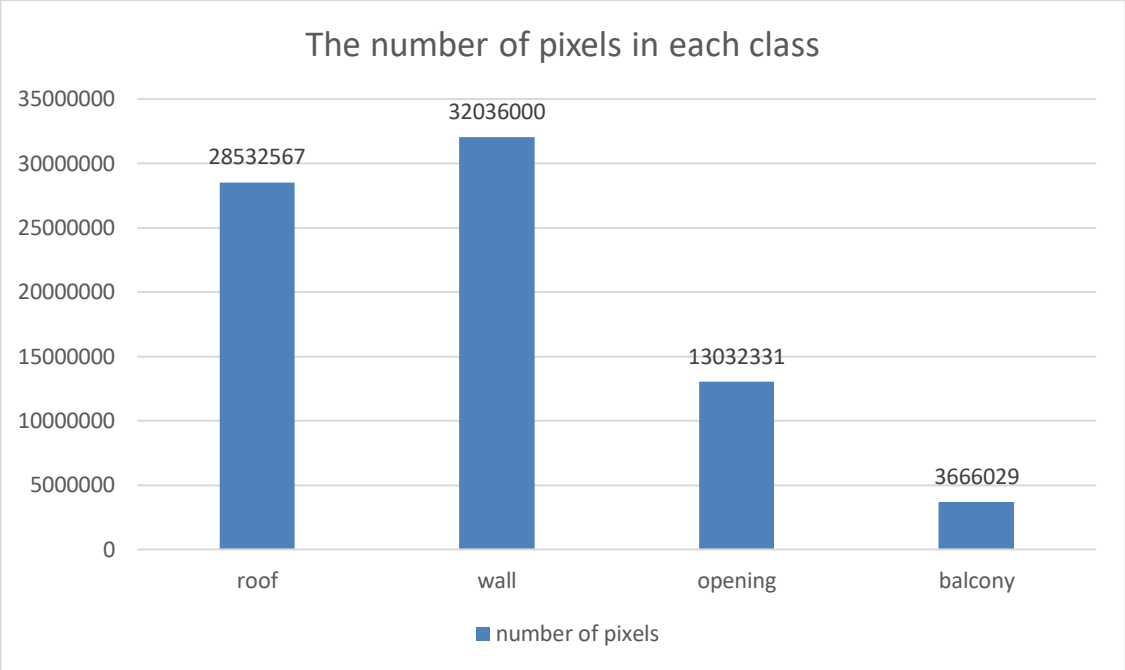


Figure 23: The number of pixels in each class for training.

To solve this problem, the weighted loss function has been used to address the imbalanced training. Cross-entropy loss is an ordinary loss function that can be used in segmentation tasks. Where y is the ground truth and $\hat{y}$ refers to the prediction generated by the last layer output.

$$\text{Loss} = -\text{y} \cdot \log(\hat{y}) \qquad \text{Equation 4-5}$$

The weighted loss function is shown in following, where W_c is the class weight computed by the number of pixels for each class in training images and L is cross-entropy loss.

$$L_{weighted} = L \cdot W_c \qquad \text{Equation 4-6}$$

4.4. Networks

There are two main structures of models used for semantic segmentation tasks in deep neural networks, namely spatial pyramid pooling and encoder-decoder structure. The major advantage of the first one is its capability to capture multi-scale information. The detailed information will be introduced in the following section. And the advantage of the Encoder-Decoder structure is that can recover the spatial information and obtain the sharp object boundary. In the following sub-sections, the network architectures that we used in our task are introduced.

4.4.1. FC-DenseNets

Fully Convolutional DenseNets was proposed by (Jegou et al., 2017). The whole process for semantic segmentation is shown in Figure 24. It is based on DenseNets (Huang et al., 2017) and extended to deal with the problem of semantic segmentation task by combining FCN with DenseNets. The goal is to not only to classify but also achieve the pixel-to-pixel segmentation and keep the original image resolution by adding the up-sampling path (the right part of "U" shape in Figure 24) to recovery from low resolution to high. This network contains fewer parameters and is not necessary to be pretrained on large datasets. Table 2 shows the configuration of FC-DenseNet103 used in this experiment.

FC-DenseNet Architecture
Input
3×3 Convolutional
DB (4 layers) + TD
DB (5 layers) + TD
DB (7 layers) + TD
DB (10 layers) + TD
DB (12layers) + TD
DB (15 layers)
TU+DB (12 layers)
TU+DB (10 layers)
TU+DB (7 layers)
TU+DB (5 layers)
TU+DB (4 layers)
1×1 Convolution
Softmax

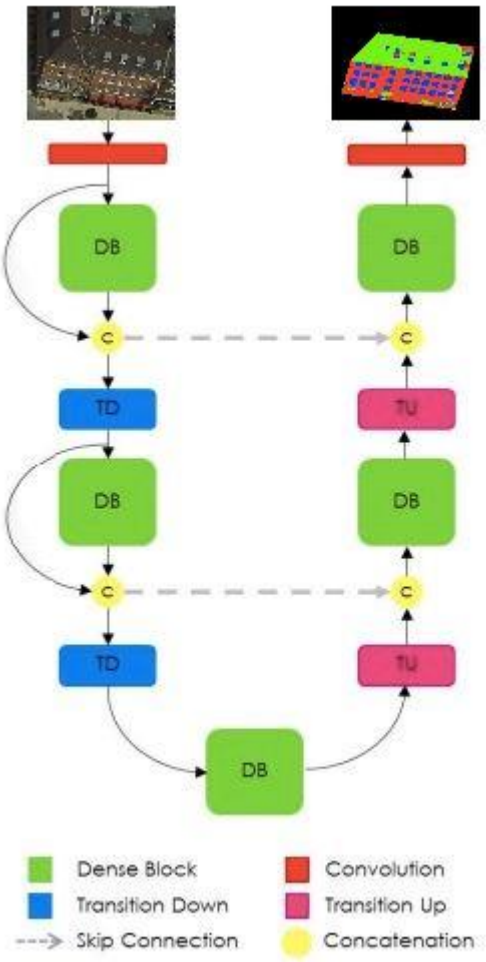


Figure 24 : The diagram of FC-DenseNet for segmentation

Table 2: The configuration of FC-DenseNet103 model.

The use of Dense Blocks gives the main feature of DenseNets. Figure 25 shows a Dense Block of 4 layers. Starting from an input x_0 with m feature maps, after going through the first layer, the output x_1 of dimension k, is generated by applying a non-linear transformation $H_1(x_0)$, where "1" indexes the layer. The input of the next layer is from stacked features by a concatenation $([x_0, x_1])$ (Jegou et al., 2017).

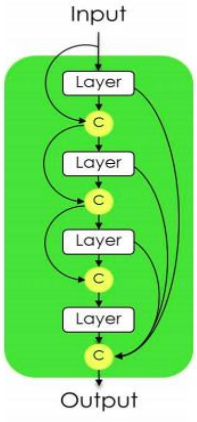


Figure 25: Dense Blocks (Jegou et al., 2017)

4.4.2. DeepLabv3 plus

The main advantage of DeepLabv3 is that it can capture the contextual information at multiple scales by applying a spatial pyramid pooling module (Chen et al., 2017). However, there is a certain drawback associated with the boundary of objects. Deeplabv3 plus, as shown in Figure 26, improved the performance based on DeepLabv3 (see the next section), by adding an encoder-decoder structure that is able to obtain sharp object boundaries (Chen et al., 2018). The X-ception model (Chollet, 2017) had provided promising images of classification results. In DeepLabv3 plus, the author modified the model and adapted it to semantic segmentation tasks, as the new backbone to extract features. In the performed tests, ResNet101 has been used as a backbone as used in DeepLabv3.

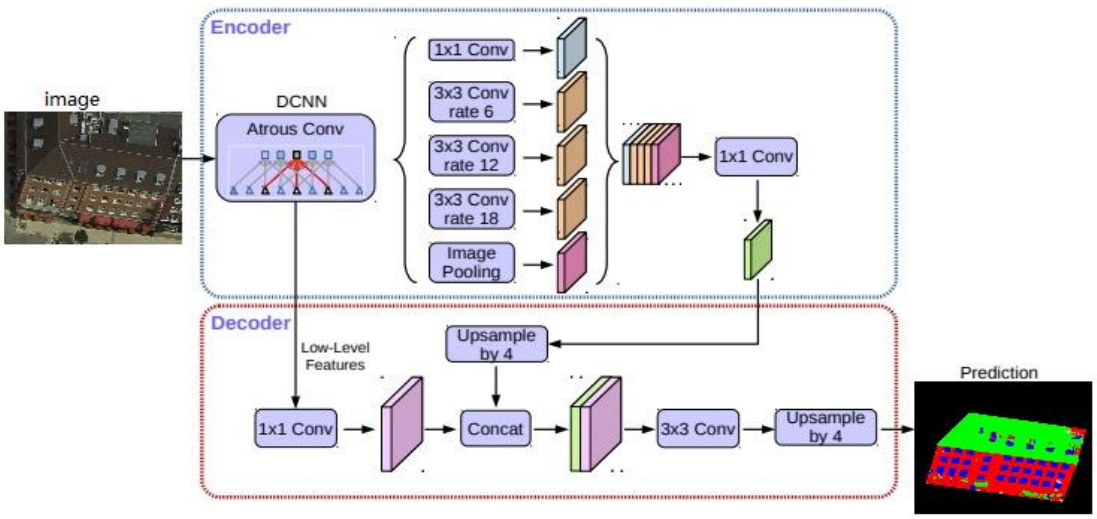


Figure 26: An illustration of DeepLabV3 + for semantic segmentation.

ASPP – Atrous Spatial Pyramid Pooling

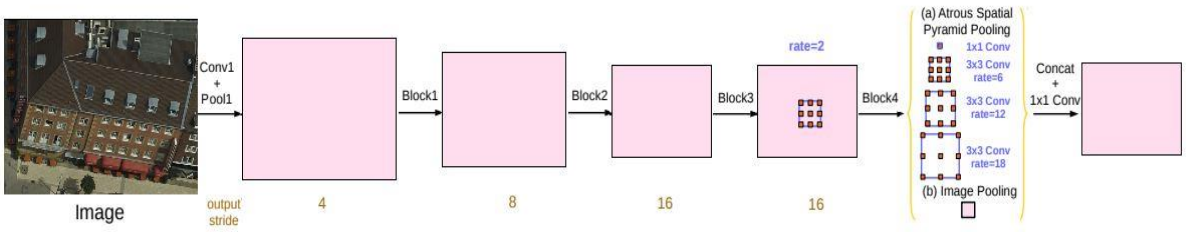


Figure 27： An illustration of ASPP structure.

Atrous Spatial Pyramid Pooling was applied in DeepLabv3 to capture contextual information in different scales with different rates, as shown in Figure 27. Spatial pyramid pooling is used to extract multi-scale feature by different sizes of spatial bins in each layer of the pyramid (He et al., 2015). The size of the spatial bin is related to the image size. Max-pooling is applied to each spatial bin, and the output of pooling is the fixed-length vector.

Spatial pyramid pooling can compensate for the loss of information by using pooling or convolution layers. ASPP includes multi-scale astrous convolution to extract the feature from the image, where astrous convolution has been explained in section 2.2.6. In this regard, the conventional stride pooling, although the main features are kept in the final feature map, it loses the small detailed information, like boundaries. Figure 28 shows the comparison of different structures.

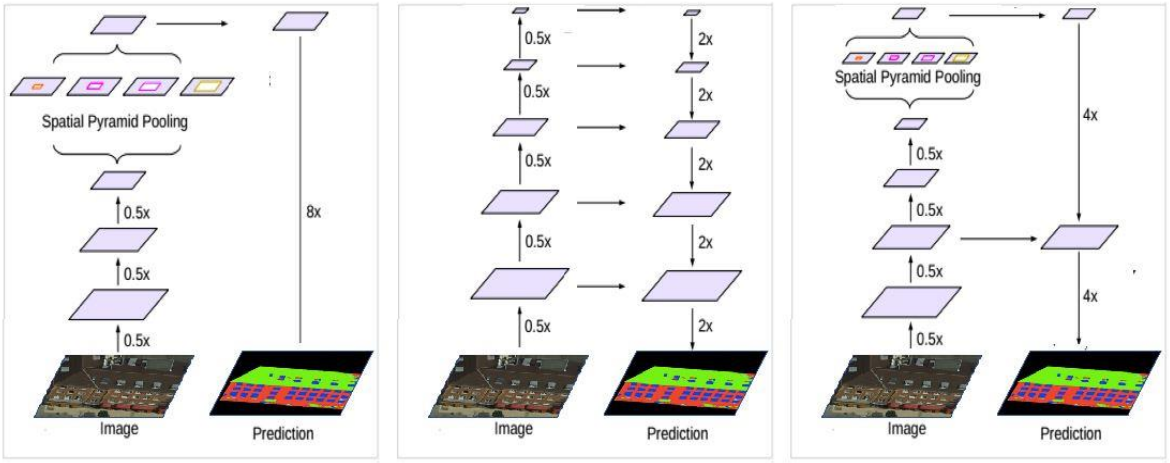


Figure 28: Left: Spatial Pyramid Pooling (DeepLab), Middle: Encoder-Decoder structure, Right: Spatial Pyramid Pooling with Encoder-Decoder structure (DeepLabV3+).

5. EXPERIMENTS AND RESULTS

5.1. Airborne datasets

The data is based on (Lin, et al. 2018) captured from Dortmund city center on July 7th, 2016, shown in Figure 29. The dataset consists of multiple views of images. The ground sampling distance is 4.5cm for the oblique image.

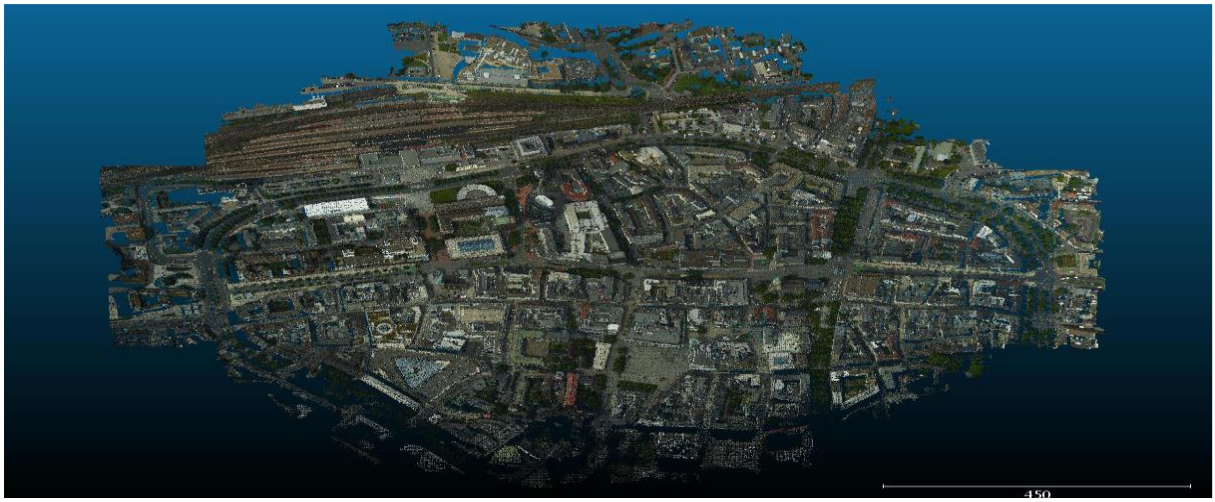


Figure 29: Dense matching point cloud of study area.

In this study, the input of the neural network is the oblique aerial image with different resolutions. And there are four classes are considered, roof, wall, balcony, and opening (window and door). Wall is the main component of the building, it is a vertical structure and takes a large ratio. The roof is a horizontal structure, which can be captured in this study, due to the aerial imagery. The Balcony is considered as a man-made structure for the building. Windows and doors can be regarded as the object named opening, because due to the reflection, it can not be distinguished. Matlab labeller has been used as the tool for the image annotation. Each pixel can be labeled in different classes with an index value (0,1,2,3,4). The example of the ground truth in this task is shown in Figure 30. In the ground truth, red means wall, green means roof, blue means blue, black means void, and grey means balcony.

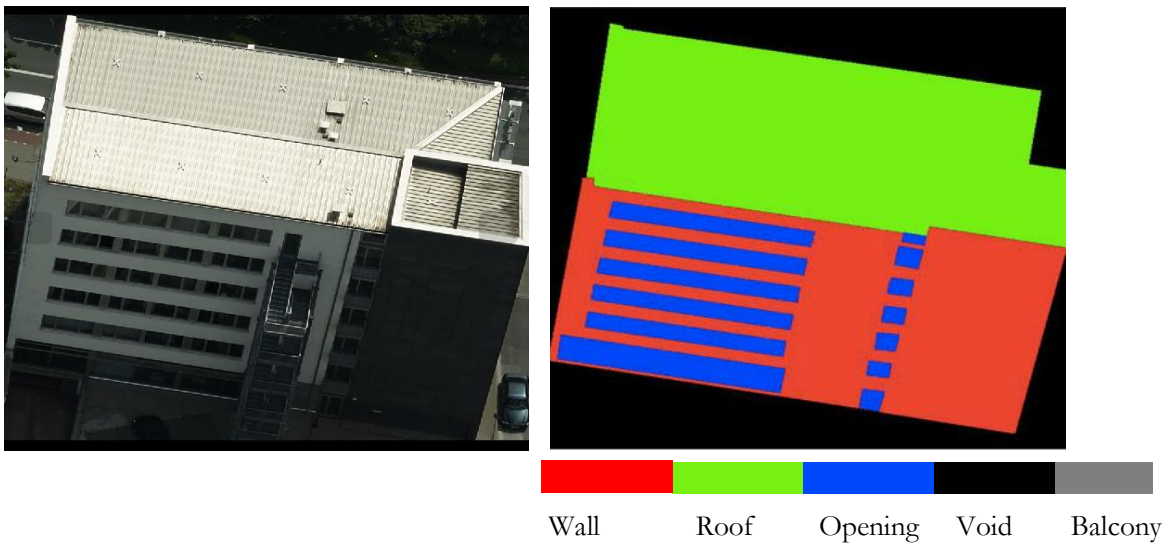


Figure 30: An example of annotation. Left: original image. Right: ground truth.

5.2. Region of Interest

The facades in each original image are in different scales. Then, the facades are not rectified into vertical perspectives, so the area surrounded the facade will influence the performance of results. To reduce the noise and train it in an efficient way, a cropped region of interest in correspondence of the façade is used as the input of networks before splitting it into patches. An example is shown in Figure 31.

Figure 31: Left: original image. Right: Region of Interest.

5.3. Experiment setup

Due to the limited dataset, the data augmentation is used in our task to increase the training data. The number of original training data is 160 images. After splitting into patches, there are 1132 images used for training. The framework of the networks is based on (Seif, 2018). As already mentioned, ResNet is used as the backbone to extract features, using the ImageNet pretrained model. Then FC-DenseNet and DeepLabV3+ are fine-tuned on our dataset.

5.3.1. Training strategy

Due to the limited memory source, patch-wise segmentation is used in the experiment. The input images are reduced to 320 by 320 for training. Besides, several tests have been performed to define the most proper configuration. In this regard, the data augmentation has been adopted too. The data augmentation is used to increase the number of training data to avoid overfitting. Table 3 shows that the data augmentation used in the experiment. First, horizontal flip has been used, reversing the images horizontally. Second, rotation values were set to 20 degrees, it means randomly changing this value for each patch in the range $[-20\ 20]$. And brightness value was set to 0.2: this value refers to randomly change the factor of brightness from 0 to 20%.

Data Augmentation	Value
Horizontal Flip	True-horizontal
Rotation	0 to 20
Brightness	0.2

Table 3: Data augmentation

The dataset is divided into three parts: training, validation, and test: 65% of the dataset as training, and 18% of the dataset as validation, the others as testing. The number of images in each part is shown in Table 4. After splitting into small patches for patch-wise segmentation, the number of images is increased in each part, which is shown in Figure 32: The number of images after splitting into small patches.

data	Value
Training	160
Validation	44
Test	42

Table 4: The number of each stage before splitting into patches

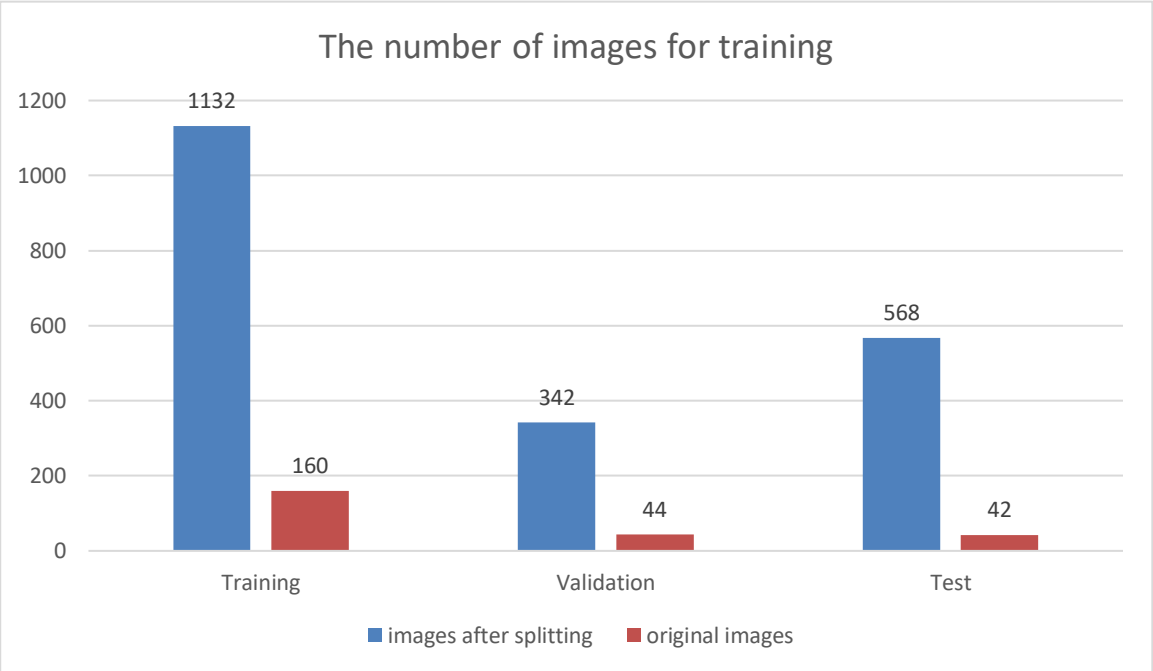


Figure 32: The number of images after splitting into small patches.

5.3.2. 2D segmentation

The input of the networks in 2D segmentation only contains three channels (RGB). The configuration and parameters shown in the following section are optimal after several testing.

FC-DenseNet

FC-DenseNet is pretrained on ImageNet and fine-tune on our task. The parameter is only shown the best performance after testing several times. For example: learning rate (0.01, 0.001, 0001, 0.00001), batch size between 1 and 8; Epochs between 50 and 150; depth 103 and 56. The optimal parameters can be seen in Table 5: The learning rate was set to 0.0001 in our task, and the decay rate set was 0.995 of original learning rate after each epoch to avoid overfitting.

Parameter	Value
Learning rate	0.0001
Decay rate	0.995
Batch size	4
Epochs	80
Depth	103

Table 5: parameters in the training process of FC-DenseNet.

DeepLabV3+

DeepLabV3+ is pretrained on ImageNet and fine-tune on our task. The most proper configuration is shown in Table 6 after several testing. According to (Chen et al., 2018), fine-tune batch normalization requires a large batch size, due to the limited source of memory, it did not fine-tune batch normalization in this experiment.

Parameter	Value
Learning rate	0.0001
Astrous rates	[6,12,18]
Decay rate	0.995
Batch size	8
Epochs	80
Batch-normalization	False

Table 6: parameters in the training process of DeepLab.

5.3.3. Combination of 2D and 3D features

The input of the networks in 2D was combined with the 3D feature in four channels, RGB and normal vector. The weights pretrained on ImageNet used the initial three channels (RGB), while the extra channel for normal vector was initialized with 0 values. The first layer was modified into $3 \times 3 \times 4 \times 64$ instead of $3 \times 3 \times 3 \times 64$ to fit the four channels input. Comparing to the 2D segmentation, the training strategy almost the same, but removes rotation from the data augmentation and set epochs to be 100.

5.3.4. Accuracy assessment

To evaluate the performance of the segmentation result, there are some metrics to be defined. Overall accuracy is the metric evaluated for the whole image, and the class accuracy is for the pixel accuracy in each class. IoU is a common way in dense prediction tasks. We take the mean over the IoU of each predefined class. In these equations, TP refers to true positives. While FP refers to the false positive, FN refers to false negatives and TN indicates the true negatives. The formulas of the used metrics are reported below.

$$Accuracy_{overall} = \frac{TP}{TP+FN} \quad \text{Equation 5-1}$$

$$Accuracy_{class} = \frac{TP}{TP+FN} \quad \text{Equation 5-2}$$

$$IoU = \frac{TP}{TP+FP+FN} \quad \text{Equation 5-3}$$

5.3.5. Convergence

The simple way to evaluate the model's performance during the training process is using a validation set. The performance on the validation set can be check if the model is fully trained. When the curve is flattened, the process of training is finished. The different learning rate and batch sizes can be affected by the rate of convergence. Figure 33 shows that the performance accuracy and IoU testing on a validation set, the curves are increasing until its convergences. And Figure 34 shows that the curve of loss testing on a validation set, which is decreasing until its convergences.

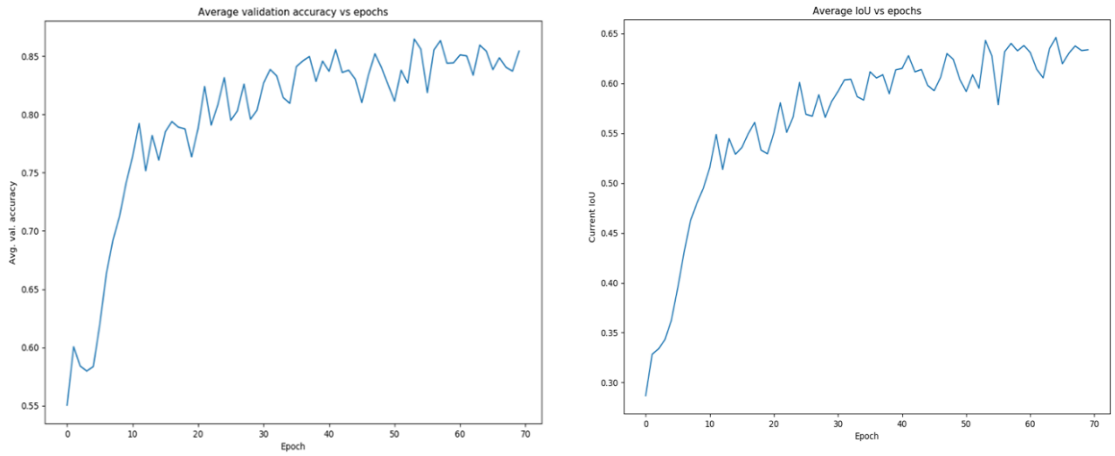


Figure 33: Left: The accuracy performance on the validation set. Right: The IoU performance on the validation set.

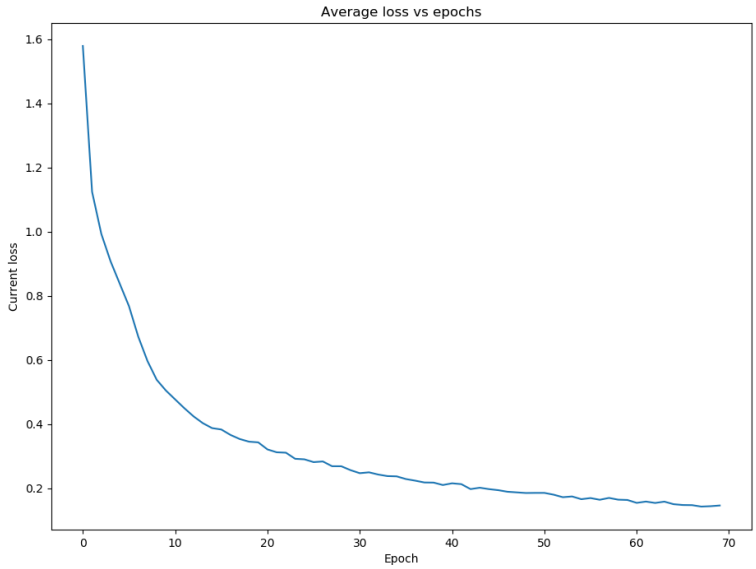


Figure 34: The curve of loss testing on a validation set.

5.4. Result and analysis

42 complete images were used for testing before splitting into patches. The visualizations of some results which have the best performance are shown in the following: Figure 35 shows that the first row is the original images, while the second gives the ground truth. In Figure 36, the first row shows the result from FC-DenseNet trained with only 2D information. The second row shows the result from DeepLabV3+ trained using only 2D information. Figure 37 shows that results generated from two models trained with 2D information and 3D feature.

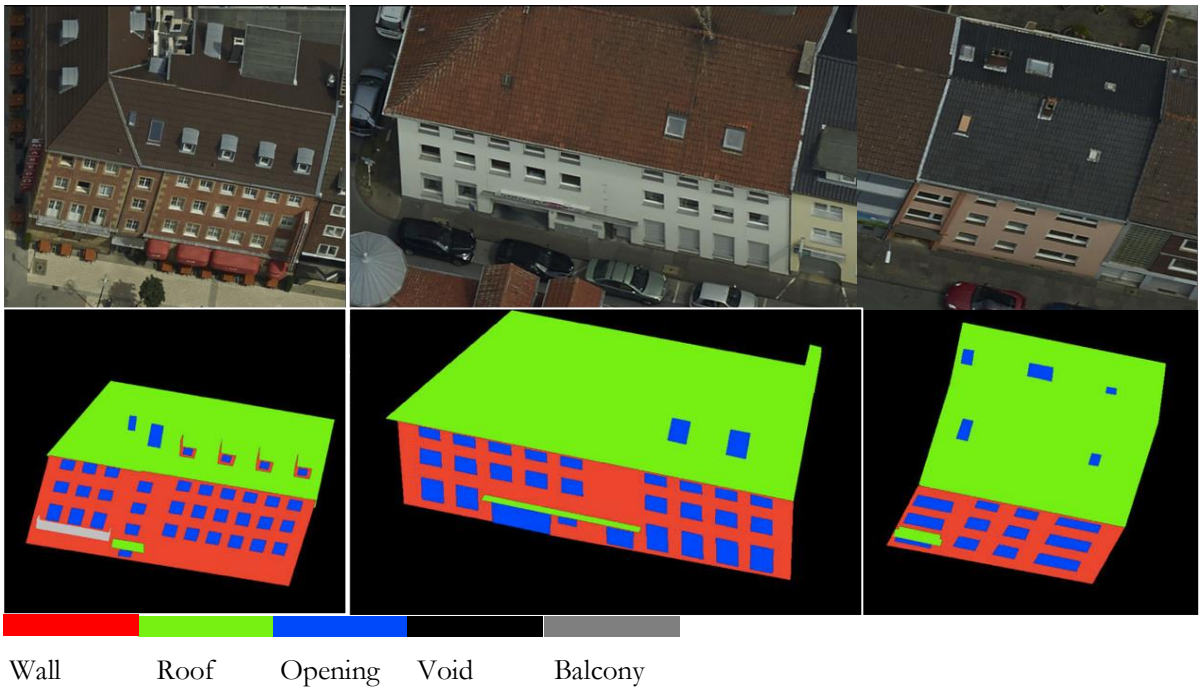


Figure 35: The first row is original images; the second row is the ground truth.

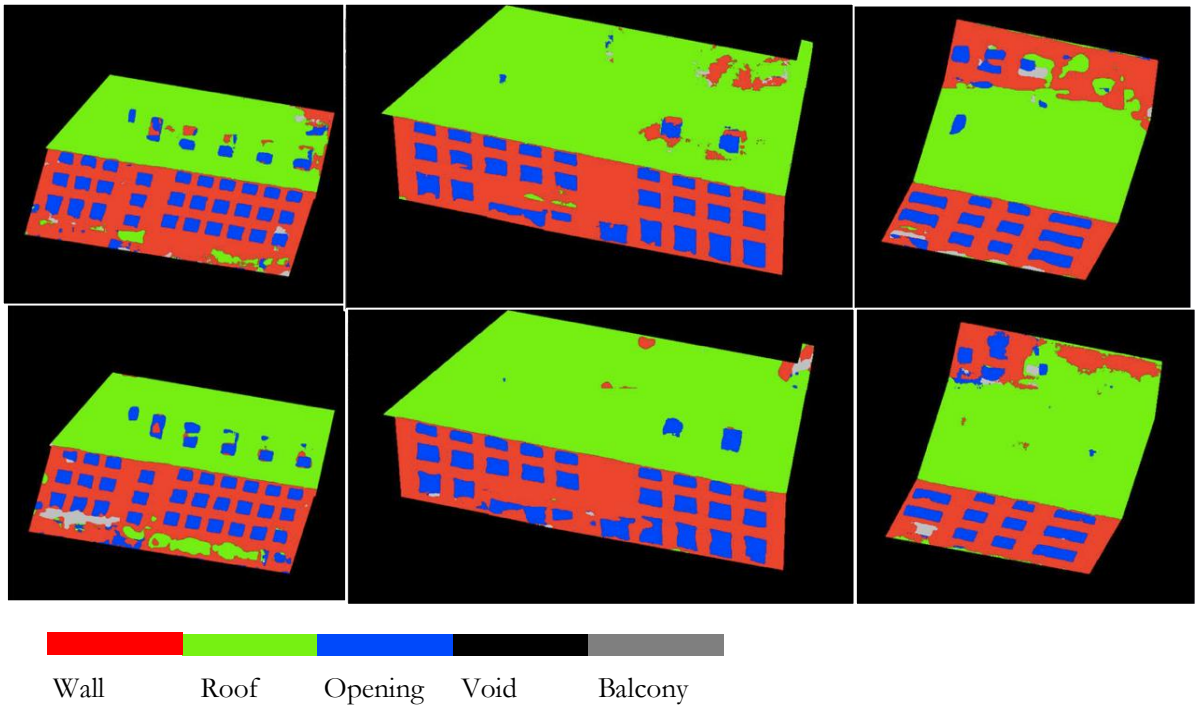


Figure 36:The visualization of results. First row: Results from FC-DenseNet trained with only 2D information; Second row: Results from DeepLab trained with only 2D information.

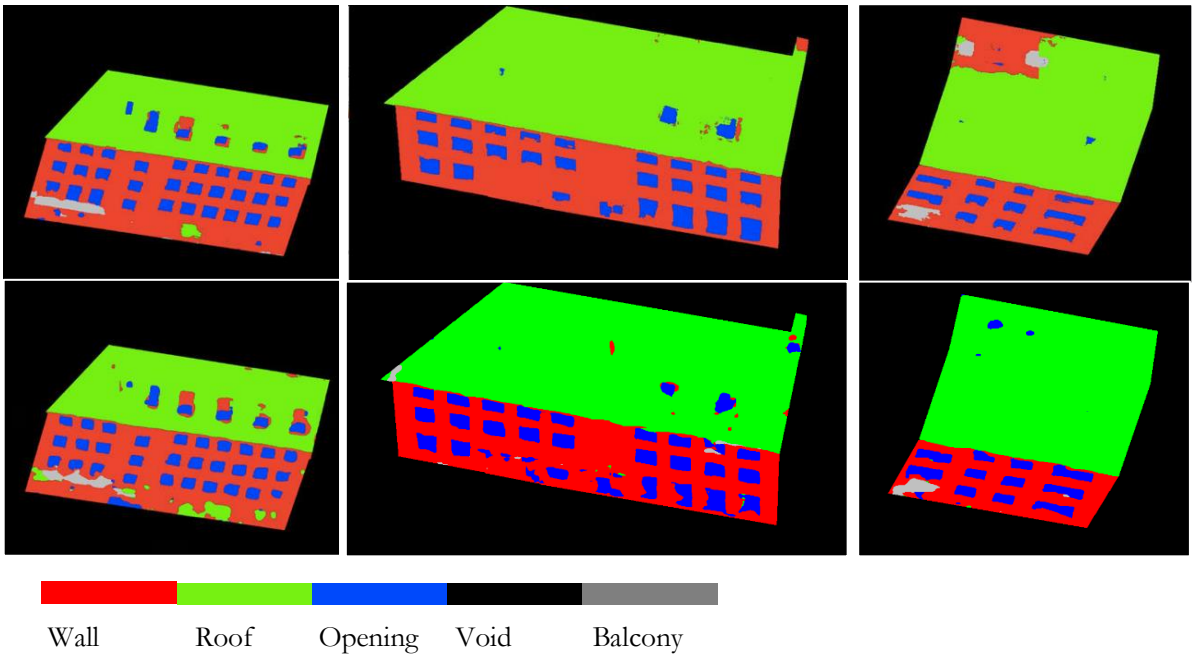


Figure 37: First row: Results from FC-DenseNet trained with 2D information and 3D feature; Second row: Results from DeepLab trained with 2D information and 3D feature.

Class	2D-FC	2D-DL	FC-23	DL-23
Roof	91.76	92.25	94.61	**95.78**
Wall	83.84	82.12	**92.86**	82.08
Opening	78.10	**81.25**	64.34	65.42
Balcony	61.66	**70.73**	64.93	63.08
Mean accuracy	78.89	**81.58**	79.18	76.59
Overall accuracy	88.42	89.08	**91.30**	91.10
IoU	59.28	62.09	**64.41**	62.16

Table 6: Results from two models with different inputs (The best is marked in Bold).

From Table 6, it can be seen that the best result is obtained by FC-DenseNet trained with 2D and 3D features: it achieves 64.41% IoU and 91.30% accuracy. The second best is given by DeepLab V3+ trained with 2D and 3D features, it obtains 62.16% IoU and 91.10% accuracy.

Compared to the only use of 2D information, the overall accuracy and the IoU using 2D and 3D features increased for both DeepLabV3+ (from 89.08% to 91.10%) and for FC-DenseNet, (from 88.42% to 91.30%). In addition, the IoU of FC-DenseNet improves more than 5%, while DeepLabV3+ achieves a less extensive improvement (1% less improvement).
Results on roof in FC-DenseNet and DeepLabV3+ using all the features are better than these two models only trained with 2D information: the roof class accuracy increased from 91.76% to 94.61% and 92.25% to 95.78% respectively;
In correspondence of walls, the wall accuracy increased from 83.84% to 92.86% and 82.12% to 82.08 respectively in two models comparing to only used 2D. T
The classification of the openings using FC-DenseNet trained using 2D achieved the best performance, 81.25%. On the other hand, when adding the 3D feature, DeepLabV3+ gave opposite results, decreasing from 81.25% to 65.42%, FC-DenseNet decreases from 78.10% to 64.34%.
Results of balcony in FC-DenseNet get a little improvement from 78.89% to 79.18% by adding a 3D feature, while the same configuration in DeepLabV3+ decreases 7.65% than using the only 2D information. Mean accuracy of class in FC-DenseNet improves 0.29% while decreases 4.99% in DeepLabV3+.

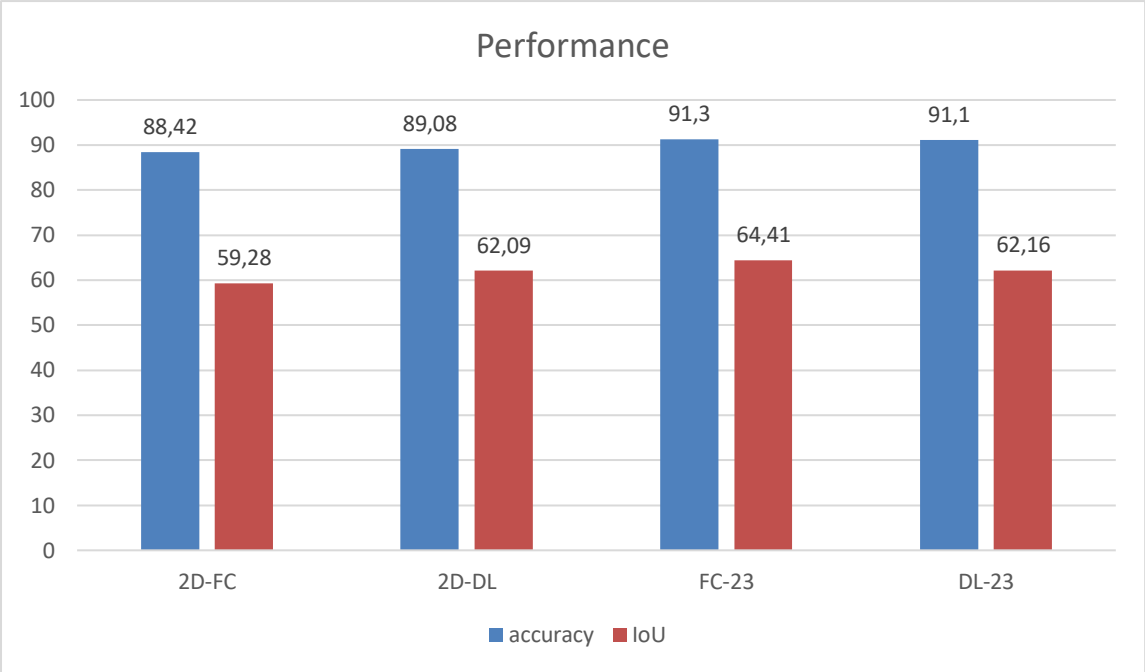


Figure 38: The Performance of two models with different inputs.

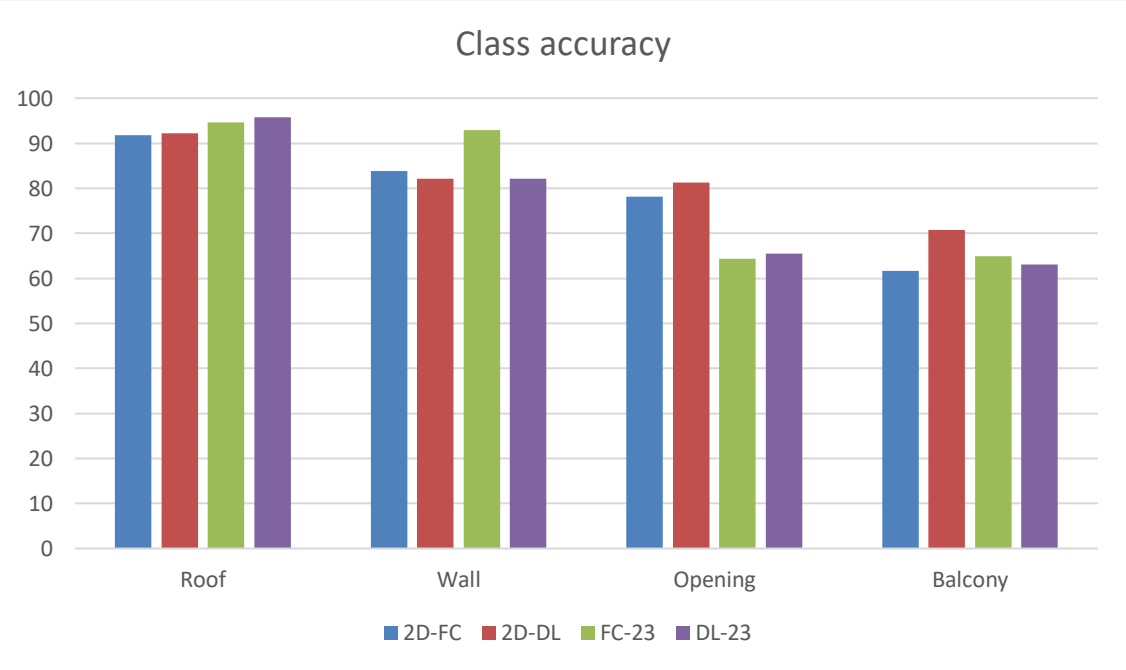


Figure 39: The performance of class accuracy.

5.5. Discussion

This section discusses the results presented in the previous section. In section 5.5.1, the confusion matrix is discussed as a tool to analyze the result. In section 5.5.2, the limitation of this experiment will be discussed.

5.5.1. Confusion Matrix

Confusion Matrix is an important metric in the classification task to evaluate the result. There are four tables of confusion matrix of different models with different inputs in the following, where the number of each class refers to the pixels belong to the specific class:

True / Pred	roof	wall	Opening	Balcony
roof	14015343	143413	54536	75569
wall	1023542	1976030	112238	189896
Opening	228889	193135	640844	94038
Balcony	140182	54596	7080	134750

Table 7: The confusion matrix of FC-DenseNet trained with only 2D information.

True / Pred	roof	wall	Opening	Balcony
roof	14106205	155145	41388	81755
wall	994362	1924928	102337	148332
Opening	159133	196898	653706	55772
Balcony	148256	90203	17267	208394

Table 8: The confusion matrix of DeepLab trained with only 2D information.

True / Pred	roof	wall	Opening	Balcony
roof	14469070	77123	45643	89937
wall	829015	2187051	234119	225248
Opening	56004	67833	525681	32607
Balcony	53867	35167	9255	146461

Table 9: The confusion matrix of FC-DenseNet trained with 2D and 3D information.

True / Pred	roof	wall	Opening	Balcony
roof	14692491	129326	85396	159949
wall	515680	1948335	155768	159406
Opening	94507	185577	550964	55050
Balcony	105278	103936	22570	119848

Table 10: The confusion matrix of DeepLab trained with 2D and 3D information.

From the above confusion matrix tables, it can be seen roof and wall have better performance comparing to other classes. Because they take the majority components of the building, and they have enough pixels

for training. In 2D experiments, the roof is mainly misclassified with the wall. Also, opening also has good performance. The misclassification of opening mainly between opening and wall. However, the balcony has a poor performance compared to others. There are some confusions with the wall. The reason is that most of the balcony on the outside of the walls.

In 3D experiments, the confusion between roof and wall is decreased in both models by adding the normal vector. However, the performance of opening is worse than the model only used 2D information. For the balcony class, FC-DenseNet gets a little benefit from adding a 3D feature, but in DeepLab, the performance is worse than only use 2D information.

Overall, the achieved results indicate that model predictions get benefits from 3D information and achieve the best performance (FC-DenseNet trained 2D with 3D). With RGB input, there are some confusions between roof and wall. Adding 3D features, as it can be seen from confusion matrixes, reduces the misclassified pixels: the normal vector can help the network to easily distinguish between these two classes and to solve the confusion by adding extra vertical spatial information. On the other hand, the normal vector has limited effects on classifying other classes where the geometric information provided by 3D data is less discriminative in the classification process: in this case, the performance in the classification of the opening can slightly decrease. These trends have been confirmed in both networks. The results provided by architectures provide similar results in terms of accuracy. Compared to these two models, DeepLabV3 plus has a better performance on the 2D experiment. Most of the class accuracy (except wall) are a bit higher than FC-DenseNet. Overall accuracy and IoU higher than FC-DenseNet as well. The confusion between wall and roof is less in DeepLab (Figure 42). Furthermore, DeepLab shows a strong power to extract the class which has less training data (balcony). When adding a 3D feature, in both models, the confusion between wall and roof is decreased. Because the normal vector provides vertical information, FC-DenseNet can get the biggest benefit from it and achieve the best performance. DeepLabV3+ is a robust network on classification task, but in this task, due to the limited dataset and hardware, it cannot achieve the best performance. The larger resolution images and more batch size (normally more than 12) will improve the performance of segmentation results. FC-DenseNet is kind for the small dataset, and the result also proved it.

5.5.2. Limitation

There are some limitations to this experiment. First, the data used in this experiment is captured by the airborne system. Comparing to the data used in (Liu et al., 2017), the oblique image is not rectified and asymmetry, there are some distortions on the shape of objects (such as window, door). It makes the neural network hard to recognize. The comparison can be seen in Figure 39.

Besides, different architecture styles also an important point for the result. The unregular shape of opening and balcony also make neural network hard to classify them. The boundary of opening in the segmentation results is blur from Figure 41. And some balconies are on the wall, and some are on the outside of the roof, as described in Figure 40.

Furthermore, the spatial information is also of importance for the segmentation result. The spatial information will be lost after several pooling layers. Due to the limited source of memory, the input has a small resolution. It cannot keep the contextual information. For a complex neural network, limited memory source is not sufficient to apply the optimal configuration.

Finally, the size of the dataset is a critical component of the classification task. Although data augmentation and patch-wise segmentation are implemented to increase the number of training data, the more data will be the benefit to the result. From the result, the weighted loss function has a limited effect on imbalance problem when using small dataset for training.

Figure 40: A comparison between our data and others

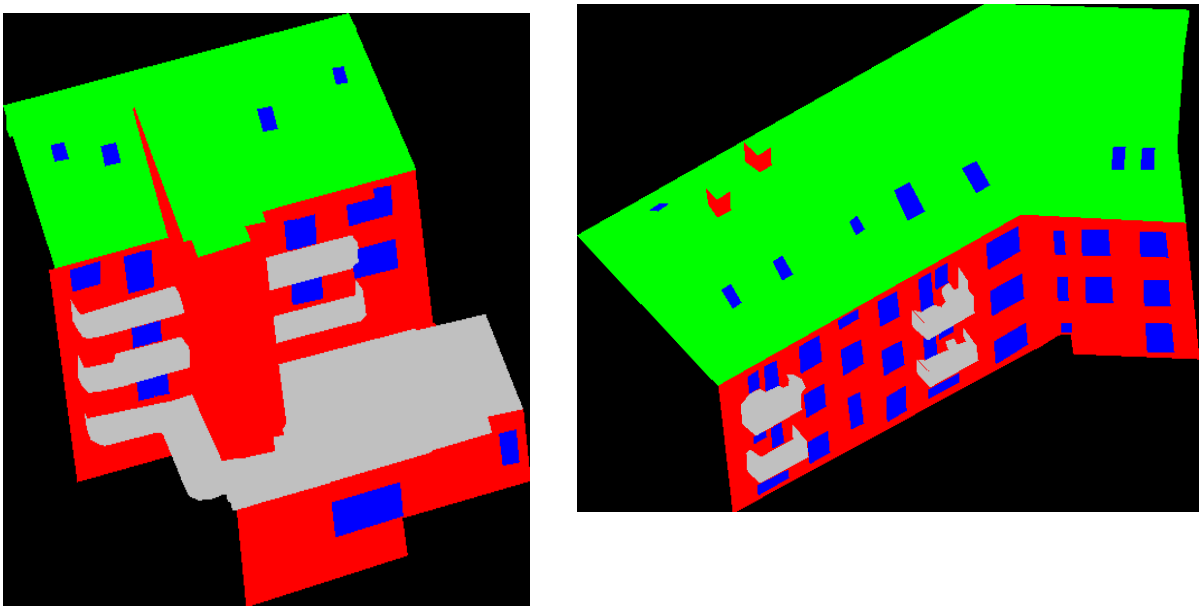


Figure 41: An example of different architectures of balcony and window.

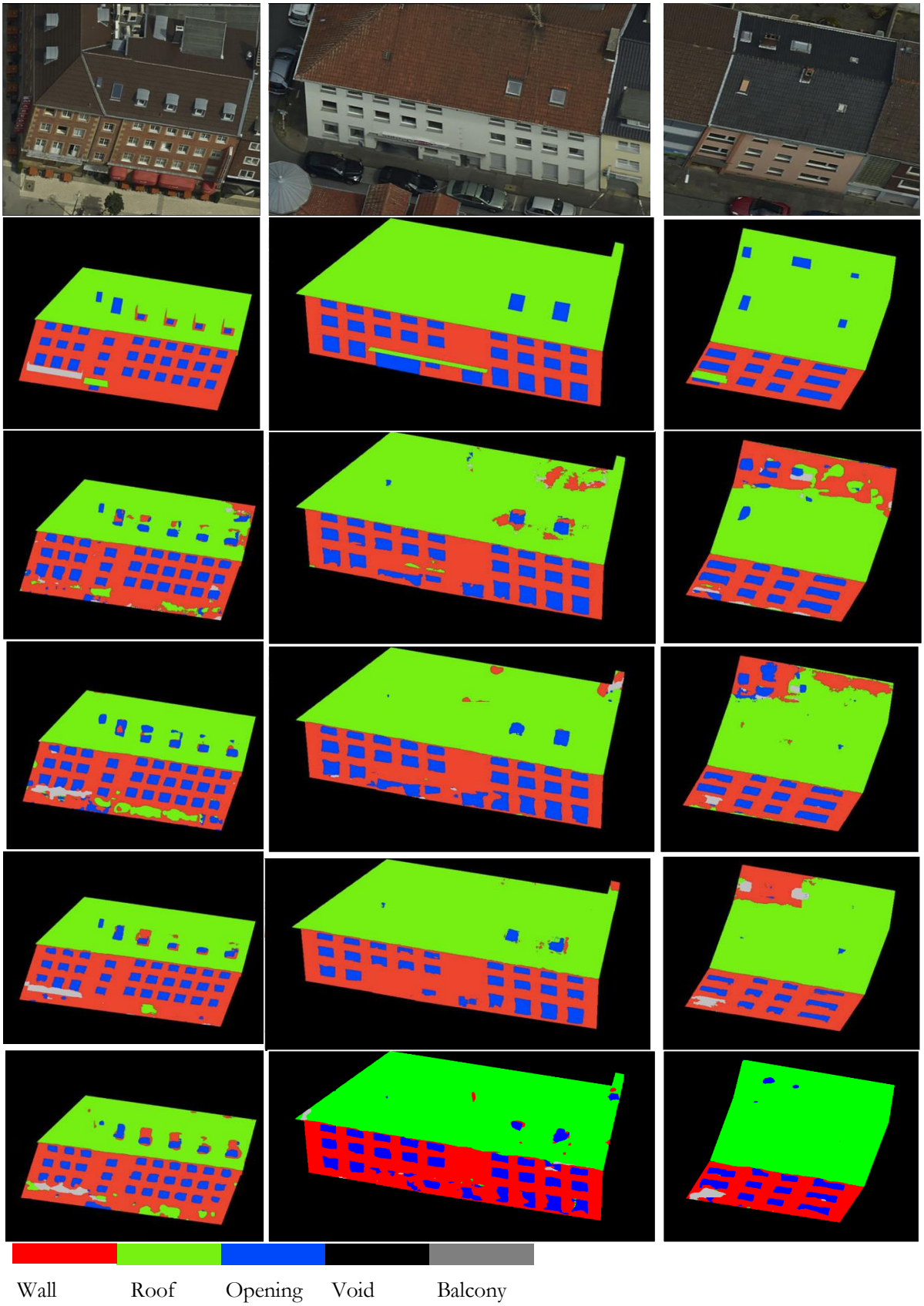


Figure 42: First row: Original images. Second row: Ground truth. Third row: Results generated by FC-DenseNet. Fourth row: Results generated by DeepLab. Fifth row: Results generated by FC-DenseNet 2D with 3D features. Sixth row: Results generated by DeepLab 2D with 3D features.

6. CONCLUSION AND RECOMMENDATIONS

6.1. Conclusion

The motivation of this study is to the reconstruct of LoD3 city modeling. Building segmentation is a sub-question of it. The detailed information of the building will be considered. The task is to parse each pixel of human-made structure to a semantic label automatically. Instead of traditional terrestrial data, airborne data cover a larger area to be investigated such as roofs that are difficult to be covered in terrestrial datasets. In this task, there are four classes considered in the task: roof, wall, opening (including window and door) and balcony. The input is not only used 2D image information but also used 2D image information with a 3D feature (the third component of the normal vector) in the neural networks.

In this study, two neural networks are applied in the experiment, FC-DenseNet and DeepLabV3+, to segment buildings captured from the airborne oblique camera system. The experiment is based on patch-wise semantic segmentation. The original images are split into small patches before putting into neural networks and Softmax function is used to reduce the effect of the combination when reconstruction from small patches to original images. Besides, the weighted cross-entropy loss function is chosen to solve the problem of imbalanced class.

The results indicate that 3D features can correct misclassified pixels by providing extra spatial information, such as confusions between wall and roof, which can be seen from confusion matrixes. The classification of balcony also gets an improvement in FC-DenseNet trained with 2D information combined with the 3D feature. The performance of FC-DenseNet and DeepLabV3+ all improves by adding a 3D feature. In FC-DenseNet, IoU increases from 59.28% to 64.41%; In DeepLab, IoU increases from 62.09% to 62.16%. Overall accuracy also all increase in two models: from 88.42% to 91.30% in FC-DenseNet and from 89.08% to 91.10% in DeepLab respectively. FC-DenseNet trained with 2D combined 3D feature gets the best result with 91.30% accuracy and 64.41% IoU. It proves that 3D feature can improve the performance of segmentation, the confusion between roof and wall reduced in this task. Compared these two models, DeepLabV3 plus has a better performance on the 2D experiment, and FC-DenseNet achieves the best performance with 91.30% accuracy and 64.41% IoU when adding a 3D feature.

There are also some limitations in this study. The experiment is only based on one dataset, few training samples can be used in the neural networks, although the data augmentation is used for increasing the training data. Limited memory source cannot make the network achieve the best performance in this task. The larger resolution of images can make the spatial information keep as much as possible after passing pooling layers.

6.2. Answers to research questions

1. What is the code already available for this study?
   The framework used in this thesis is based on TensorFlow framework. And the 3D feature extracted from point clouds based on MATLAB code.

2. which kind of 3D information can be used in the network?
   The third component of the normal vector is as the 3D feature applied in convolutional neural networks. It can provide extra information to distinguish surfaces are horizontal, vertical or slanted.

The normal vector is derived from a cluster of neighbouring points which can be selected by different searching strategies and searching ranges.

3. How to use the extra 3D information for this task to get improvements on segmentation results?
   The 3D feature is extracted from point clouds and projected into image space and taken as the fourth band of the image.

4. What are the existing architectures that can be used for this task?
   Fully Convolutional DenseNet and DeepLabV3 plus. Because FC-DenseNet has a good performance on small data and less parameters in the networks; DeepLabV3 plus is one of the most promising networks for segmentation task

5. Which part can be modified to improve the accuracy of the results?
   In 3D experiments, the first layer of neural networks was modified into $3 \times 3 \times 4 \times 64$ instead of $3 \times 3 \times 3 \times 64$ to adapt the 4 channels input.

6. How to choose the parameters in the networks?
   I have tried different combinations of parameters and configurations. And the optimal will be chosen as the final one to put into the neural network after testing several times and checking the curve of testing on a validation set.

7. Which is the best metric can be used to evaluate the result?
   Overall accuracy is the metric evaluated for the whole image, and the class accuracy is for the pixel accuracy in each class. IoU is a common way in dense prediction tasks. We take the mean over the IoU of each predefined class. Confusion matrix also shows the performance of each class in the segmentation result. In these equations, TP refers to true positives. while FP refers to the false positive, FN refers to false negatives and TN indicates the true negatives.

8. Which network has a better performance compared to others?
   Results show that FC-DenseNet trained with 2D and 3D features achieves the best result, IoU ups to 64.41% and accuracy ups to 91.30%.

9. Does 3D give any benefit for results?
   FC-DenseNet trained with 2D and 3D features increases 5.13% compared to the result of the same model trained without 3D features. Also, it can reduce the confusion between roof and wall in both models (FC-DenseNet and DeepLab). The classification of balcony gets a little bit benefit from the 3D feature in FC-DenseNet.

6.3. Recommendations

- In further work, more 3D features can be involved in the training process to provide more spatial information to improve the performance of other classes, such as planarity, linearity etc.
- CRF or other methodology can be applied as the post-processing to refine the result. In this task, there is also some noise in the result, and the boundary of small objects is blur. The post-processing can consider removing the noise and extracting the border of small objects for better performance.

- Furthermore, the limitation of this work is the limited memory of resource, if increase the memory of resource, more advanced neural networks can be applied in the experiment and testing large resolution of images on the semantic segmentation of buildings.
- The size of a dataset is critical for the classification task. Enlarging the size of the dataset can improve the performance of segmentation

- Different models can be tested in different steams. At the final stage, it can make a fusion of different results by different models to improve the performance compared to the individual one.

- Rectification can be done in the pre-processing to increase the accuracy of classification and reduce the confusions. In this task, the distortion of objects makes the neural network hard to classify the pixels into the correct class.
- Image classification can be combined with point clouds segmentation, divided into two streams and combinate in the final stage. For example, one stream can be a neural network for image segmentation, and the other one is PointNet to deal with point clouds.

LIST OF REFERENCES

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. https://doi.org/10.1109/72.279181

Boulaassal, H., Landes, P., Grussenmeyer, F., & Tarsha-Kurdi, F. (2007). Automatic Segmentation of Building Facades using Terrestrial Laser Data. *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, *XXXVI*, 65–70. https://doi.org/10.1007/978-3-540-72135-2_9

Boulaassal, H., Landes, T., & Grussenmeyer, P. (2009). Automatic Extraction of Planar Clusters and Their Contours on Building Façades Recorded by Terrestrial Laser Scanner. *International Journal of Architectural Computing*, *7*(1), 1–20. https://doi.org/10.1260/147807709788549411

Brostow, G. J., Shotton, J., Fauqueur, J., & Cipolla, R. (2008). Segmentation and Recognition using Structure from Motion Point Clouds. *Eccv*, *5302*, 1–14. https://doi.org/10.1007/978-3-540-88682-2_5

Brownlee, J. (2018). Why One-Hot Encode Data in Machine Learning? Retrieved February 8, 2019, from https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. https://doi.org/10.1159/000018039

Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 834–848. https://doi.org/10.1109/TPAMI.2017.2699184

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* (Vol. 2017–Janua, pp. 1800–1807). https://doi.org/10.1109/CVPR.2017.195

Cohen, A., Schwing, A. G., & Pollefeys, M. (2014). *Efficient structured parsing of facades using dynamic programming. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2014.410

Delmerico, J. a., David, P., & Corso, J. J. (2011). Building facade detection, segmentation, and parameter estimation for mobile robot localization and guidance. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1632–1639. https://doi.org/10.1109/IROS.2011.6094778

Douglas M. Hawkins*. (2004). The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences*, *44*(19), 1–12. https://doi.org/10.1021/ci0342472

Fritsch, D., Becker, S., & Rothermel, M. (2013). Modeling Façade Structures Using Point Clouds From Dense Image Matching. *Proceedings of the Intl. Conf. on Advances in Civil, Structural and Mechanical Engineering*, (Author 2), 57–64. https://doi.org/10.3850/978-981-07-7227-7_22

Fröhlich, B., Rodner, E., & Denzler, J. (2010). A fast approach for pixelwise labeling of facade images. *Proceedings - International Conference on Pattern Recognition*, 3029–3032. https://doi.org/10.1109/ICPR.2010.742

Gadde, R., Jampani, V., Marlet, R., & Gehler, P. V. (2018). Efficient 2D and 3D Facade Segmentation Using Auto-Context. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(5), 1273–1280. https://doi.org/10.1109/TPAMI.2017.2696526

Gadde, R., Marlet, R., & Paragios, N. (2015). Learning Grammars for Architecture-Specific Facade Parsing. *International Journal of Computer Vision*, *117*(3), 290. https://doi.org/10.1007/s11263-016-0887-4>

Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., & Garcia-Rodriguez, J. (2017). A Review on Deep Learning Techniques Applied to Semantic Segmentation. In *arXiv preprint* (pp. 1–23). https://doi.org/10.1007/978-1-4471-4640-7

Gröger, G., & Plümer, L. (2012). CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, *71*, 12–33. https://doi.org/10.1016/j.isprsjprs.2012.04.004

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904–1916. https://doi.org/10.1109/TPAMI.2015.2389824

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). https://doi.org/10.1109/CVPR.2016.90

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* (Vol. 2017–Janua, pp. 2261–2269). https://doi.org/10.1109/CVPR.2017.243

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Retrieved from http://arxiv.org/abs/1502.03167

Jegou, S., Drozdzal, M., Vazquez, D., Romero, A., & Bengio, Y. (2017). The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2017–July*, 1175–1183. https://doi.org/10.1109/CVPRW.2017.156

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, *2*, 1137–1143. https://doi.org/10.1067/mod.2000.109031

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, 1–9. https://doi.org/http://dx.doi.org/10.1016/j.protcy.2014.09.007

Lafferty, J., McCallum, A., & Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning*, *8*(June), 282–289. https://doi.org/10.1038/nprot.2006.61

Li, W., & Yang, M. Y. (2016). Efficient Semantic Segmentation of Man-Made Scenes Using Fully-Connected Conditional Random Field. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLI-B3*(July), 633–640. https://doi.org/10.5194/isprsarchives-XLI-B3-633-2016

Lin, Y. (2018). *SEMANTIC BUILDING FAÇADE SEGMENTATION FROM AIRBORNE OBLIQUE IMAGES*. Retrieved from https://library.itc.utwente.nl/papers_2018/msc/gfm/lin.pdf

Lin, Y., Nex, F., & Yang, M. Y. (2018). Semantic Building Façade Segmentation from Airborne Oblique Images. *ISPRS Technical Commission II Symposium 2018*.

Liu, H., Zhang, J., Zhu, J., & Hoi, S. C. H. (2017). Deepfacade: A deep learning approach to facade parsing. *IJCAI International Joint Conference on Artificial Intelligence*, 2301–2307.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Vol. 07–12–June, pp. 3431–3440). https://doi.org/10.1109/CVPR.2015.7298965

Martinovic, A., Knopp, J., Riemenschneider, H., & Van~Gool, L. (2015). 3d all the way: Semantic segmentation of urban scenes from start to end in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4456-4465).

Martinović, A., Mathias, M., Weissenberg, J., & Van, L. (2012). *A Three-Layered Approach to Facade Parsing - Supplementary Material*. Retrieved from www.springerlink.com.

Rahmani, K., Huang, H., & Mayer, H. (2017). FACADE SEGMENTATION with A STRUCTURED RANDOM FOREST. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *4*(1W1), 175–181. https://doi.org/10.5194/isprs-annals-IV-1-W1-175-2017

Rahmani, K., & Mayer, H. (2018). HIGH QUALITY FACADE SEGMENTATION BASED on STRUCTURED RANDOM FOREST, REGION PROPOSAL NETWORK and RECTANGULAR FITTING. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (Vol. 4, pp. 223–230). https://doi.org/10.5194/isprs-annals-IV-2-223-2018

Seif, G. (2018). Semantic-Segmentation-Suite. Retrieved from https://github.com/GeorgeSeif/Semantic-Segmentation-Suite

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICRL)*, 1–14. https://doi.org/10.1016/j.infsof.2008.09.005

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, *15*, 1929–1958. https://doi.org/10.1109/ICAEES.2016.7888100

Stanford University, Simonyan, K., Zisserman, A., Sebastian Ruder, Pan, S. J., Yang, Q., … Hassabis. (2016). CS231n Convolutional Neural Networks for Visual Recognition. *ArXiv Preprint ArXiv:1511.07289*, *22*(10), 1345–1359. https://doi.org/10.1109/IJCNN.2016.7727519

Tu, J., Sui, H., Feng, W., Sun, K., Xu, C., & Han, Q. (2017). Detecting building façade damage from oblique aerial images using local symmetry feature and the gini index. *Remote Sensing Letters*, *8*(7), 676–685. https://doi.org/10.1080/2150704X.2017.1312027

Tutzauer, P., & Haala, N. (2015). Façade reconstruction using geometric and radiometric point cloud information. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *40*(3), 247.

Weinmann, M., Jutzi, B., & Mallet, C. (2014). Semantic 3D scene interpretation: A framework combining optimal neighborhood size selection with relevant features. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *II-3*, 181–188. https://doi.org/10.5194/isprsannals-II-3-181-2014

Weinmann, M., Urban, S., Hinz, S., Jutzi, B., & Mallet, C. (2015). Distinctive 2D and 3D features for automated large-scale scene analysis in urban areas. *Computers & Graphics*, *49*, 47–57. https://doi.org/10.1016/j.cag.2015.01.006

Xiao, J., Gerke, M., & Vosselman, G. (2012). Building extraction from oblique airborne imagery based on robust façade detection. *ISPRS Journal of Photogrammetry and Remote Sensing*, *68*, 56–68. https://doi.org/10.1016/J.ISPRSJPRS.2011.12.006

Yang, B., Wei, Z., Li, Q., & Li, J. (2013). Semiautomated building facade footprint extraction from mobile LiDAR point clouds. *IEEE Geoscience and Remote Sensing Letters*, *10*(4), 766–770. https://doi.org/10.1109/LGRS.2012.2222342

Yang, M. Y., Förstner, W., & Chai, D. (2012). Feature evaluation for building facade images-an empirical study. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences: [XXII ISPRS Congress, Technical Commission I] 39 (2012), Nr. B3* (Vol. 39, No. B3, pp. 513-518). Göttingen: Copernicus GmbH.

Yang, M. Y., & Wolfgang, F. (2011). Regionwise classification of building facade images. In *ISPRS Conference on Photogrammetric Image Analysis* (pp. 209-220). Springer, Berlin, Heidelberg.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Proceeding NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 Pages 3320-3328*, (November). https://doi.org/10.1109/IJCNN.2016.7727519

Zhu, X. X., Tuia, D., Mou, L., Xia, G. S., Zhang, L., Xu, F., & Fraundorfer, F. (2017). Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. *IEEE Geoscience and Remote Sensing Magazine*, *5*(4), 8–36. https://doi.org/10.1109/MGRS.2017.2762307