東京大学大学院新領域創成科学研究科
社会文化環境学専攻

2017 年度
修　士　論　文

CNN を利用した非都市地域における建物の抽出手法に関する研究
Identification of Buildings in Rural Environment based on Convolutional Neural Networks

2017 年 7 月 18 日提出
指導教員　柴崎　亮介　教授

郭　直霊
Guo, Zhiling

THE UNIVERSITY OF TOKYO

MASTER'S THESIS

Identification of Buildings in Rural Environment based on Convolutional Neural Networks

Author:
Guo Zhiling

Supervisor:
Prof. Shibasaki Ryosuke

A thesis submitted in fulfilment of the requirements for the degree of Master of Philosophy

in the

Shibasaki Lab
Department of Socio-Cultural Environmental Studies

July 2017

Declaration of Authorship

I, Guo Zhiling, declare that this thesis titled, 'Identification of Buildings in Rural Environment based on Convolutional Neural Networks' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________________

Date: ______________________________________

THE UNIVERSITY OF TOKYO

Abstract

Graduate School of Frontier Sciences
Department of Socio-Cultural Environmental Studies

Master of Philosophy

Identification of Buildings in Rural Environment based on Convolutional Neural Networks

by Guo Zhiling

Since accurate building maps are often unavailable or outdated in undeveloped rural environment, building identification in such areas has gradually become a significant research field in remote sensing. Due to the high price-performance ratio, many recent corresponding studies are performed based on open high-resolution remote sensing (HRRS) data such as Google Earth (GE) images. However, most existing classification methods applied for identification of building in rural environment identification can only generate low- or middle-level image features with limited representation ability, which essentially prevents them from achieving good performance in various scenes. Meanwhile, in image classification field, the preponderance of Convolutional Neural Network (CNN) has been proved owing to its advantages such as efficiently generating high-dimensional abstract feature. In this dissertation, we present a specific CNN model which elaborately formulated based on state-of-the-art structures to identify buildings in rural environment from open HRRS images. First, the feasibility of proposed CNN based method is proven by comparing with other machine learning methods. Second, In order to optimize and mine CNN's capability for rural environment mapping and also be compatible with our classification targets, the basic model is carefully modified and adjusted based on a series of rigorous testing results. Third, the methods such as Transfer Learning, color balance and Data Augmentation are implemented to enhance the robustness of model. Finally, the generated model is applied in a pixel-level classification frame to generate high accuracy identification results. Experimental results of the test area at developing countries prove that the proposed CNN model significantly outperforms the previously best stated results, improving the overall accuracy from 96.30% to 99.26%, and Kappa from 0.56 to 0.86. For implementation on GPGPU and cuDNN, the required processing time accelerated approximately 30 times compared with CPU case.

Acknowledgements

I would like to thank all the people who contributed in some way to the work described in this dissertation. First and foremost, I thank my academic supervisors, Professor Shibasaki Ryosuke and Shao Xiaowei, for accepting me into Shibasaki Lab, engaging me in new ideas, helping and encouraging me when I face difficulties. Additionally, I would like to thank my vice supervisor Kobayashi Hiroki for this help in my work.

I would like to acknowledge Shao's group member: Ph.D. Xu Yongwei, Ph.D. Chen Qi and Wu Guangming. I greatly benefited from their keen scientific insight and attitude of life.

I am grateful for the project sources that allowed me to pursue my master studies: GRENE-ei (Green Network of Excellence, Environmental Information), the Ministry of Education, Culture, Sports, Science and Technology (MEXT) in Japan.

I would like to thank the various members with whom I had the opportunity to work and have not already mentioned: Miyazaki Hiroyuki, Ohira Wataru, Song Xuan, Sengoku Hiroaki, Hamada Tsuyoshi, Chen Quanjun, Fan Zipei, Yuan Wei, Saurav Ranjit, Jiang Renhe, Xia Tianqi, Huang dou, Huang Shuzhe, Hoshi Tomonori, Koga Yohei, Dou Jie. . .

I would like to acknowledge the University of Tokyo and the Department of Socio-Cultural Environmental Studies. My study experience here would greatly benifit my whole life.

Finally, I would like to acknowledge my dear friends, family and girlfriend Cloris who always supported me during my time here.

. . .

Contents

List of Figures

List of Tables

Abbreviations

GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
OSM	**O**pen**S**treet**M**ap
TM	**L**andsat **T**hematic mapper
ETM+	**E**nhanced **T**hematic **M**apper **Plus**
NOAA	**N**ational **O**ceanic and **A**tmospheric **A**dministration
AVHRR	**A**dvanced **V**ery **H**igh **R**esolution **R**adiometer
LiDAR	**Li**ght **D**ection **A**nd **R**anging
HRRS	**H**igh **R**esolution **R**emote **S**ensing
GE	**G**oogle **E**arth
RF	**R**andom **F**orest
Adaboost	**A**daptive **B**oosting
NN	**N**eural **N**etworks
SVM	**S**uper **V**ector **M**achine
NDBI	**N**ormalized **D**ifference **B**uild-up **I**ndex
CNN	**C**onvolutional **N**eural **N**etworks
NLP	**N**atural **L**anguage **P**rocessing
SGD	**S**tochastic **G**radient **D**escent
CUDA	**C**omput **U**nified **D**evice **A**rchitecture
cuDNN	NVIDIA **cu**DA **D**eep **N**eural **N**etwork library
UAV	**U**nmanned **A**erial **V**ehicles
GIS	**G**eographic **I**nformation **S**ystem

Dedicated to Peaceful World...

Chapter 1

Introduction

1.1 Research Background

With the rapid development of urbanization processes, maps used to illustrate buildings and their distribution are significant and required in a wide range of fields. Important applications include environmental monitoring, resource management, disaster response, and homeland security [1].

In urban areas, accurate maps are often available, but this is not always the case for rural environment [2]. Although the rural environment such as villages around the world have been significantly improved owning to the increasingly speed up development over the past decades, compared with relatively informationalized urban areas, the deficient and lagging behind information system would be a vital constraint, which prevents village from being developed into a higher lever. As an very important geoinformation, building maps are often insufficient, and there may be no digital version available.

The deficiency building map information in rural environment would bring the inconvenience and several negative consequences. First of all, building maps in catastrophe are quite indispensable. For instance, during catastrophic events such as the Wenchuan earthquake in 2008, disaster relief could not be conveniently provided due to inadequate rural information and because the location of residential buildings was uncertain, which resulted in serious loss of life and property [3]. A swift update of building information is very important and essential during catastrophes [4] because secondary disasters such as tsunamis, avalanches, and landslides may follow [5], causing swift changes to land conditions. Another important usage of building maps is their contribution to global navigation satellite system (GNSS). As a surface feature complement, building maps can provide accurate and convenient instruction, the popular GNSS such as Google Maps and OpenStreetMap(OSM) highly based on building maps [6]. Figure 1.1 shows the rural environment maps of the same area in Foxdale Britain by Google maps and OSM respectively, we can find out the building information in Google maps compared with OSM is quite insufficient, witch would bring the inconvenience. This is even the case in developed country, and there's no doubt that in developing countries the condition would be more severe. Rural environment building maps also play an important role in medicine and hygiene filed. For example, HIV has speared through the whole world in recent decades, as the most severe infected area, developing countries in Africa need to be focused on [7]. In order to investigate the disease, hygiene conditions and also avoid further spearing, residential settlements are usually be chosen as important inquire area. However, the residential settlements in Africa are quite scattered, and usually there is no map available which makes them hard to be find out even from the satellite images. Moreover, In rural environment area planing, which aims to benefit the inhabitants, public facilities also need to be developed on the basis of residential buildings' distribution information [8].

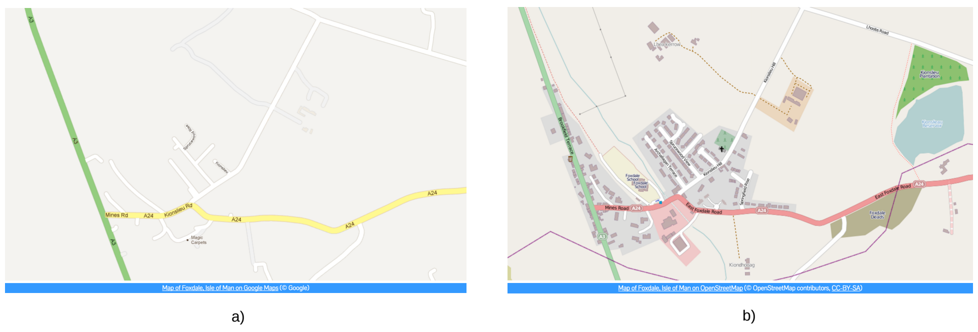


Figure 1.1: Map of Foxdale, Isle of Man on Google Maps(a) and OSM(b) respectively (2016.07).

In contrast to densely packed urban buildings, buildings in rural environment, as shown in Figure 1.2, own their characteristics such as sparsely scattered, arbitrarily change lack of regulation and architecture feature not distinct [9]. Also, corresponding buildings usually mixed with complex and diverse land features such as agriculture lands, mountains, rivers, etc. Hence, by considering both usage of building maps and such complexity of spatial and structural patterns makes rural environmental building identification a fairly challenging problem, and the tools used to identify buildings must provide rapid, accurate, efficient, and time-sequenced results.

Rather than fieldwork and ground investigation, identify buildings in rural environment depending on the remote sensing image would be more convenient and efficient. With

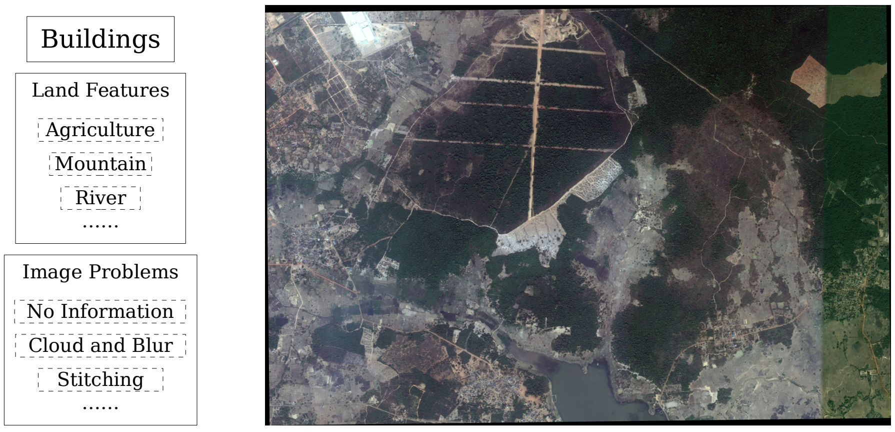


FIGURE 1.2: Buildings in rural environment.

the help of remote sensing satellite images [10–12], earth-observation activities on regional to global scales can be implemented owing to advantages such as wide spatial coverage and high temporal resolution [13, 14]. First, in terms of satellite categories, most previous studies that focus on rural environment mapping commonly use low- and medium-spatial resolution satellites such as Landsat Thematic Mapper (TM), The Enhanced Thematic Mapper Plus (ETM+), and National Oceanic and Atmospheric Administration (NOAA)/ Advanced Very High Resolution Radiometer (AVHRR) [15–17], whereas in recent years, high-resolution satellites such as QuickBird, Ikonos, and RapidEye facilitate high-accuracy identification. Unfortunately, considering the high cost of data acquisition, images provided by these high-resolution satellites are generally utilized in a specific small region and are rarely applied in large ones [18]. Second, different image data source categories such as widely used light detection and ranging(LiDAR), Hypersectral and Multispectral images, might provide high accuracy result in some specific conditions [19], but corresponding data source is usually not available in our condition.

A promising alternative solution is offered by 3 bands high-resolution remote sensing (HRRS) images such as Google Earth (GE) and Bing Maps, which provides open, highly spatially resolved images suitable for building identification in rural environment [20–23]. However, GE images and Bing Maps have rarely been used as the main data source in corresponding filed. Such images are limited to a three-band color code (R, G, and B), which is expected to lower the classification performance due to its poor spectral signature [24]. Actually, the potential for the classification of spatial characteristics by GE images and Bing Maps have been underestimated [25]. By analyzing the tone, texture, and geometric features in the image [23, 26], experts can recognize village buildings with high confidence. Consequently, we believe that GE images can provide a good data source for village mapping.

1.2 Related Works

In terms of classification technique, many methods have been studied by consulting published literatures. Note that the traditional visual interpretation of remote sensing images is a very complex and time-consuming process. Although with very high accuracy, it is not suitable to large-scale automation projects. The Figure 1.3, which implemented by a medical group [27] in Nagasaki University, shows the building identification result based on visual interpretation in Kenya. The manual visual interpretation work was taken several weeks and the related massive implementation seems impossible.

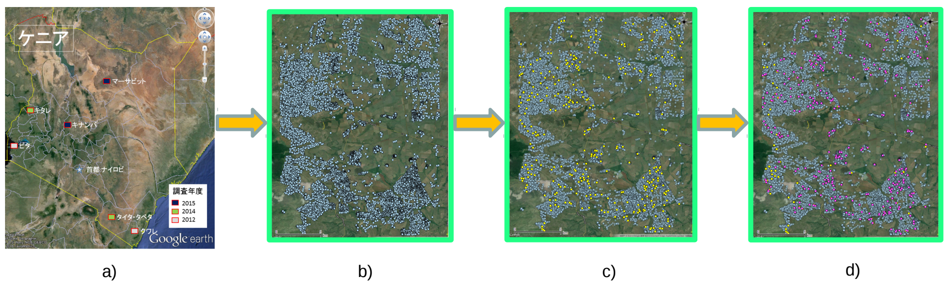


FIGURE 1.3: Identification of buildings in rural Environment based on Google Earth via visual interpretation.

In order to provide automatic and high accuracy identification result, with the help of image processing and feature extraction techniques, various machine learning algorithms [28] such as Random Forest (RF) [29], Adaptive Boosting (AdaBoost) [30, 31], Neural Networks (NN) [32] and Super Vector Machine (SVM) [33, 34] in remote sensing have been implemented. For instance, Zhang et al. [35] combine the K-means method with AdaBoost to classify buildings, and the overall accuracy is about 90%. Zongur et al. [36] utilize satellite images to detect an airport runway using AdaBoost with a circular-Mellin feature. Using an improved Normalized Difference Build-up Index (NDBI) and remote sensing images, Li et al. [37] dynamically extract urban land. Cetin et al. [38] use textural features such as the mean and standard deviation of image intensity and gradient for building detection. For the identification of forested landslides, Dou et al. [39] utilize a case-based reasoning approach and Li et al. [40] adopt two machine learning algorithms: RF and SVM. When dealing with classifying complex mountainous forests via remote sensing images, Attarchi et al. [41] verify the performances of three machine learning methods: SVM, NN, and RF. For mapping urban areas of DMSP/OLS nighttime light and MODIS data, Jing et al. [42] also utilize SVM.

As shown above, most existing classification methods applied for rural environmental building identification can only generate low- or middle-level image features with limited representation ability, which essentially prevents them from achieving good performance in various scenes. Meanwhile, in image classification field, the preponderance of convolutional neural networks (CNN) [43] has been proved in recent years owing to its advantages such as efficiently generating high-dimensional abstract feature. In the field of remote sensing detection, using the CNN method [44], Chen et al. [45] address vehicle detection, Li et al. [46] focus on building pattern classifiers, and Yue et al. [47] use both spectral and spatial features for hyperspectral image classification. To predict geoinformative attributes from large-scale images, Lee et al. [48] also choose CNN, and Sermanet et al. [49] utilize the CNN method to identify house numbers. Other important works such as Marmanis et al. [50] use pretrained CNN model and big dataset to classification land features while Ding et al. [51] add data Augmentation into CNN for SAT based building recognition. In high-resolution image processing, the innovated works conducted by Hu et al. [52], who use transfer learning to enhance obtained model in order to identify land features from HRRS and achieved an overall accuracy of approximately 98%, also Martin et al. [53] classify buildings using multiple CNN layers, pretrained model and K-meaning.

To the best of our knowledge, there is no existing research by combining HRRS images and CNN to identify buildings only focusing on rural environment. Considering the characteristics of GE images and Bing Maps, we herein explore the feasibility of rural environmental building identification using CNN. In order to prove the effectiveness of the proposed CNN based method, we compare CNN result with other supervised machine learning approaches, here we choose Adaboost as an example, while others methods delivered similar performance to Adaboost and required much longer computation time, therefore, we only introduce but exclude these methods in the present analysis. Both CNN and Adaboost adopt different feature extraction schemes, enabling full exploitation of the texture, spectral, geometry, and other characteristics in the images. The AdaBoost algorithm focuses on the color and textural information of the buildings and their surrounding areas; hence, it utilizes both color information and a large number of Haar-like features.

The performance of the AdaBoost method largely depends on the quality of feature selection, which is itself quite challenging. In contrast, the CNN method achieves more robust and stronger performance than AdaBoost because it mines the deeper representative information from low-level inputs [44]. With multilayer networks trained by a gradient descent algorithm, CNN can learn complex and nonlinear mapping from a high- to low-dimensional feature space.

In this dissertation, we first constructed a basic four layer CNN to describe the characteristics inside an 18 x 18-pixel window and applied it to the identification. After analyzing the classification result, we modify CNN based on some state-of-the-art models and vigorous principles, and explore their potential in building identification respectively. Considering the limitation of available datasets, we also use data augmentation [54] method to increase diversity and quantity of data. What's more, we utilize transfer learning [55] in order to enhance our obtained models' capability, and some other image processing methods are also implemented as post processing in identification results.

1.3 Outline of the Dissertation

This chapter introduced the rural environmental buildings identification problem, and described its key importance and difficulties. It then continued with a background introduction of building detection condition in machine learning, outlined our approach to the problem, and summarized our main contributions. The remaining chapters are organized as follows:

- Chapter 2: Describes the study area and the experimental dataset.
- Chapter 3: Introduces the principle of different machine learning methods, describes basic and state-of-the-art transformed CNN structures, and also proposes transfer learning, data augmentation, color balance, etc. The experiment platform is also briefly introduced.
- Chapter 4: Compares the experimental results of each algorithms, discusses the capability of proposed CNN structures, also shows the result enhancement after transfer learning, color balance, etc.
- Chapter 5: Shows the practical application of our proposed method by testing several images
- Chapter 6: Concludes our work and presents some proposals for future works

List of Figures – lists all the figures for the thesis.

List of Tables – contains all the tables' name and related location.

Chapter 2

Dataset

In contrast to densely packed urban buildings, buildings in rural environment tend to be more sparsely scattered. In this study, we define rural environmental buildings as any settlement with size less than 2 km.

2.1 Study Area

In terms of study area, to test the feasibility of proposed method in different regions, we choose rural environment in some developing countries such as Laos, Kenya and Thailand. One of study area (Figure 2.1), is located at the Savannakhet province (Figure 2.1c) in Laos. The remote top-view RGB image of Kaysone has a size of 3600 $\times$ 4500 pixels with a resolution of 1m, which was captured from Google's satellite map in February 2015. Its longitude and latitude range from E104°47′22" to E104°49′54" and from N16°34′28" to N16°36′26", respectively, showing an area of approximately 19.44 km^2, the study area is a complex and rural region with many different types of landscape, including abundant natural components such as mountains, rivers, and vegetation cover as well as artificial areas such as villages, roads, and cultivated land, which are typical in rural areas. As another study area, Kwale is a small town in the capital of Kwale County, Kenya. It is located at around S4°10′28" and E39°27′37", 30 km southwest of Mombasa and 15 km inland. The data source is obtained from Bing Maps with resolution 1.2m in May 2015, the corresponding area mainly covered by forest and other desolated landscapes, where the buildings are quite scattered. To compare the influence of resolution on the experiment, we also own dataset with resolution up to 0.6m captured in January 2016 Kenya, where the buildings and other land features could show more details. Besides, Ratchaburi is one of the western provinces of Thailand, 80 kilometers west of Bangkok, the HRRS images in this area also generated from Bing Maps with

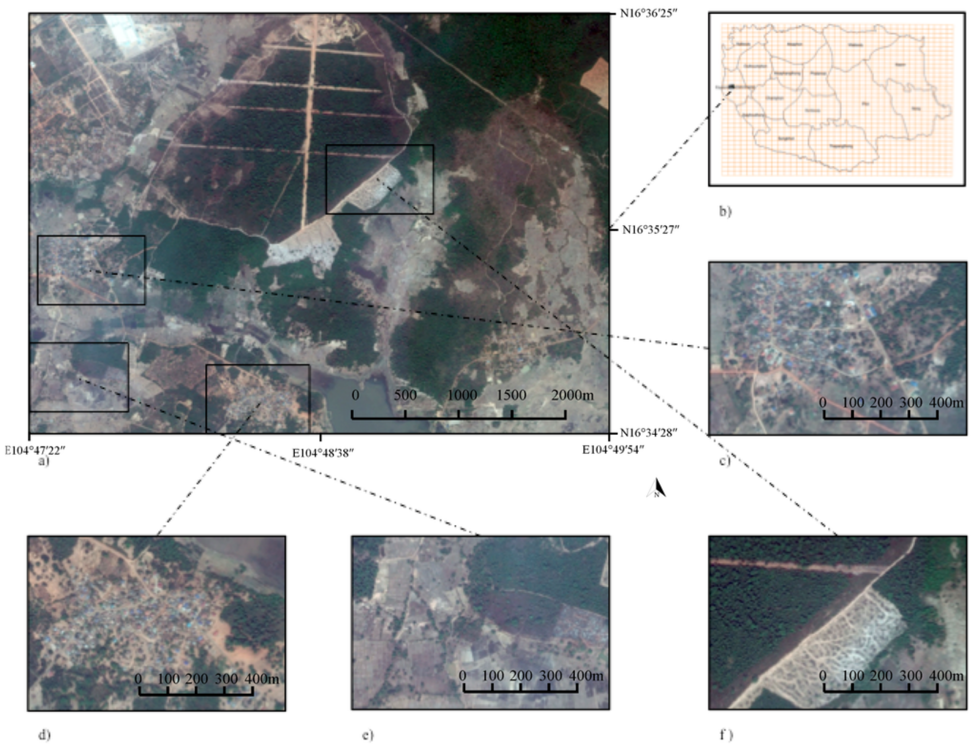


FIGURE 2.1: Study area example (a); located in Savannakhet Province, Laos (b). Panels (c) and (d) are enlarged views of typical village areas; and (e) and (f) show mountain and vegetation areas with similar tones to buildings. The resolution of all images (except (b)) is 1 m.

resolution 1.2m in May 2015. The projection of above datasets are in the UTM Zone 48N system and Datum WGS 84.

2.2 Data

In general, dataset in supervised machine learning methods [56] can be divided into training dataset and testing dataset. training dataset is a set of data used to discover potentially predictive relationships, here refers to original RGB HRRS images and the corresponding ground truth information. Testing dataset is a set of data used to assess the strength and utility of a predictive relationship, here refers to remote sensing images which need to be tested.

In details, As the training dataset in Laos, we selected four typical village/non-village areas (see Figure 2.1) from the original image in Laos. Each image was sized 600 ×

900 pixels. Figure 2.1c,d in rich in rural environment building information, showing features such as buildings, roads, rivers, and cultivated lands. Figure 2.1f contains forests and mountains, whereas Figure 2.1e features crop and vegetation cover. The test data are contained within the entire area of Figure 2.1a. Within this area, the ground truth map of the village buildings was manually drawn beforehand using a polygon-based interaction tool. This ground truth map contains accurate information of the land categories and is chiefly used in sampling and result detection. Similar with Laos, training dataset in Kenya and Thailand also selected by considering the characteristic and diversity of landscape.

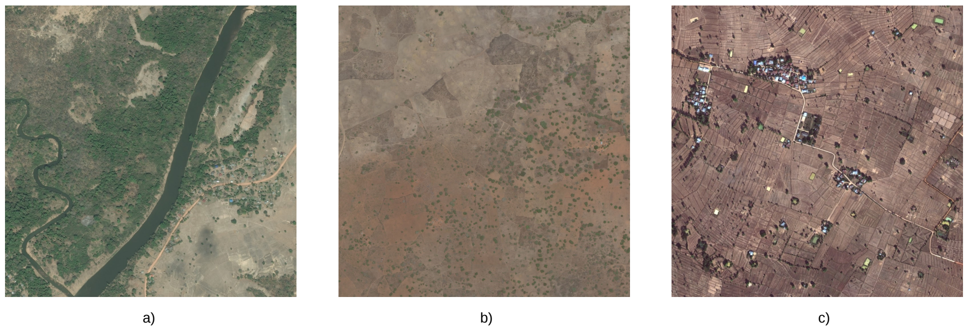


Figure 2.2: Testing area examples. (a), (b) and (c) locate in Laos, Kenya and Thailand respectively.

Testing dataset contains several different types of landscapes, and the land feature in different countries and areas shows the distinctive characteristics. As shown in Figure 2.2, lands features in Laos(a) and Thailand(c) is more abundant than Kenya(b). The diversity and complexity of image also bring the difficulty in identification task, which makes classification model need to adopt all these conditions.

Chapter 3

Methodology

To conduct CNN, which is a supervised machine learning algorithm, preparing training samples and their relevant labels is the first step; then, rigorously elaborate CNN architecture in order to implement the training procedure; After training, the CNN classifier is generated; in order to verify the feasibility and stability of model, cross validation is implemented in the next step; then, the testing result and the corresponding figure will be generated by testing the provided dataset via obtained classifier. Finally, post processing is needed to improve result accuracy.

3.1 Workflow

In details, an elaborate workflow is formulated which mainly contains the flowing 6 parts:

- Step 1: Training dataset preparation

  In general, a training dataset is a set of data used to train classification model and discover potentially predictive relationships. As mentioned in Chapter 2, the training dataset contains two parts in our experiment: 3 bands RGB HRRS images and the corresponding ground truth labels. What's important, the training dataset need to be prepared by considering the complexity and characteristics of identification target, the diversity is quite important. Moreover, color balance method is conducted when image facing severe unbalance problem in spectrum, while data augmentation method is proposed to prevent over-fitting [57].

- Step 2: Model Creation

  There are different choices of machine learning models for classification and regression problem. Although each of methods can be viewed as a blackbox, each model

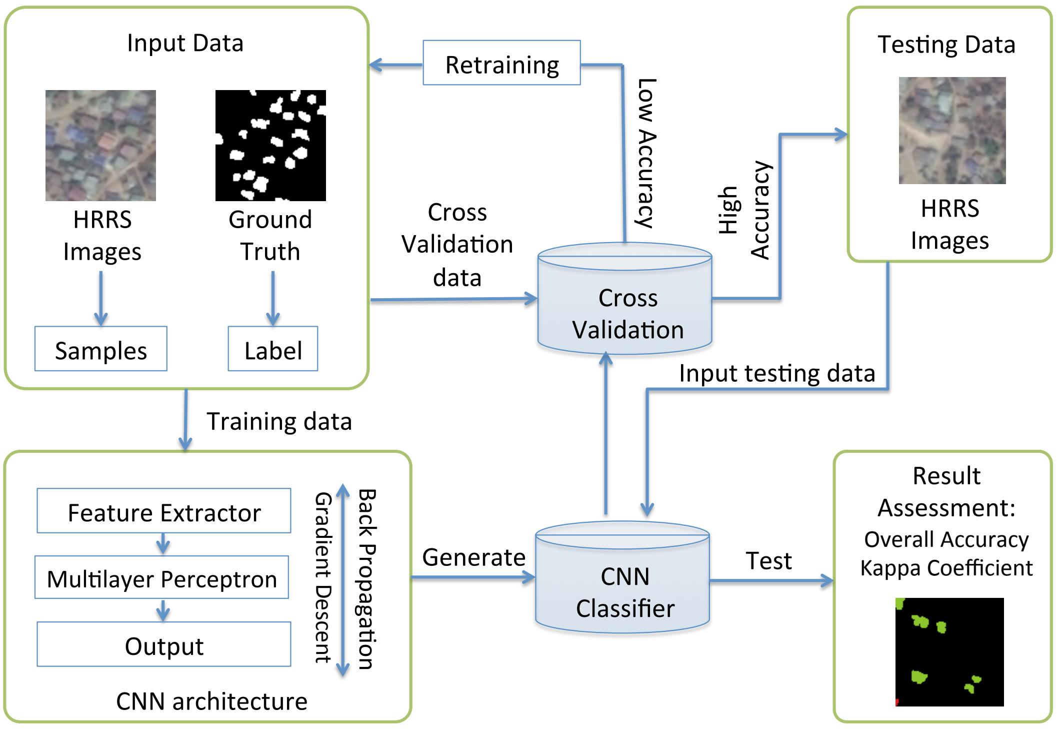


FIGURE 3.1: workflow.

comes from a different algorithm approaches and will perform different under different datasets. In this study, we present CNN model to identify village buildings from open HRRS images. In order to optimize and mine CNN's capability for rural environmental building identification and also be compatible with our classification targets, some state-of-the-art CNN structures are carefully modified and adjusted based on a series of rigorous testing results. We also create models which generated by other machine learning methods to be the comparison. What's important, transfer learning model can be proposed to train new model with limited dataset.

- Step 3: Model Training

  The machine learning algorithm will learn from the training dataset patterns that map the variables to the target, and it will output a trained model that captures these relationships. In this experiment, depending on propagation and gradient decent, a CNN model can be generated by training, which own the capability to identify buildings in rural environment.

- Step 4: Cross Validation

Cross validation is the best way to verify the feasibility and determine which model perform best on the test dataset. In order to avoid over-fitting and other problems, when any classification parameter needs to be adjusted, it is necessary to have a validation dataset in addition to the training and testing datasets. If good result can be generated in cross validation processing, the corresponding model can be used in practical, otherwise, the model need to be retrained. Here, To evaluate the accuracy and reliability of result, confusion matrix, Kappa Coefficient and Overall Accuracy are utilized in our experiment.

- Step 5: Testing

  Utilizing generated CNN model, buildings in rural environment can be identified via prepared testing HRRS dataset.

- Step 6: Post Processing and others

  The basic workhorse of morphological noise removal method such as close, can get rid of noise in the result, and make result more smoothly.

3.2 Machine Learning Approaches

3.2.1 Introduction

As a subfield of computer science, machine learning gives computers the ability to learn without being explicitly programmed. It evolved from the study of pattern recognition and computational learning theory in artificial intelligence and explores the study and construction of algorithms that can learn from and make predictions on data [58]. The important application fields including image processing, trajectory analysis [59] and natural language processing (NLP) [60], etc.

Machine learning approaches can be divided into two broad categories: supervised and unsupervised. In general, unsupervised learning methods infer to function to describe hidden structure from unlabeled dataset, also known as a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled response [61]. The most common unsupervised learning method is cluster analysis. In our experiment, the unsupervised learning methods are rather hard to learn the characteristics of GE images to a satisfactory level because of their limited distinguish ability and lack of prior knowledge. In contract with unsupervised learning, supported by training data, supervised learning methods usually deliver better classification results.

In this study, CNN is employed as classifier and the feature extraction step is designed accordingly. In order to analyze the feasibility and efficiency of using CNN, we also employed other powerful supervised machine learning methods to be the comparison, such as RF, Adaboost, NN and SVM.

Considering the limitation and characteristic of our training dataset, we utilize transfer learning, which focus on storing knowledge gained while solving one problem and applying it to a different but related problem.

Moreover, color balance and data augmentation methods also implemented to enhance our model. Some other image processing methods can used to get rid of paper-and-salt in result.

3.2.2 Random Forest

Evolved from Decision Tree [62], RF [63] is a popular method for various machine learning tasks, and the basic principle can be easily understood by its lively name. In terms of 'Random', it can be clarified in two aspects. First, rather than putting the whole samples into decision tree, only a part of them are randomly selected. Second, the amount of features is randomly chosen, which makes every decision tree owns its specific characteristics. 'Forest' means the method generated by combining several decision trees, and each Decision Tree can be seem as a weak classifier. For a input sample, every Decision Tree can generate a corresponding output respectively. The final result is decided by voting, which means choose the most prevalent output to be the final classification result.

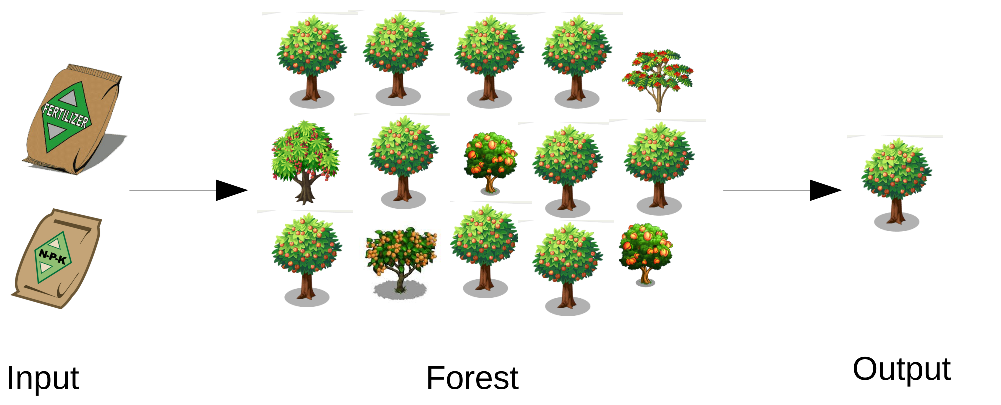


Figure 3.2: Random Forest.

As shown is Figure 3.2, the tree and grown fruit here refers to Decision Tree and the output result respectively. while input fertilizer here means input sample, and exterior appearance of tree means the selected feature. The name, which means final output, of a specific forest is decided by the most prevalent species, for instance, the peach forest means the forest which mainly grown peaches although some other species mixed inside.

The principle of RF makes it owns many advantages. Owning to the result decided by several weak classifiers, RF model usually has very high stability and accuracy. Also, due to only part of samples are input in each decision tree, big dataset can be involved without using excessive memory. What's more, the algorithm can also evaluate the importance of each feature without descending dimension.

3.2.3 Adaboost

The AdaBoost classification method [31, 64] has gained great popularity due to its high accuracy, low complexity, and ability to recognize salient features. Based on the sample data, this algorithm iteratively adjusts the weights of many weak classifiers and builds these weak classifiers into a strong classifier. The samples are defined as:

$$(x_1, y_1), ..., (x_m, y_m) \tag{3.1}$$

where m refers to the number of samples, and $x \in X, y_i \in Y = \{-1, +1\}$ .Note the inclusion of both positive and negative input samples. Initially, all samples are equally weighted as follows:

$$W_1(x_i) = 1/m \tag{3.2}$$

The weight is adjusted after checking the performance of the weak classifier. If a sample cannot be correctly classified, its weight is increased. The details of the AdaBoost algorithm are shown below:

For $t = 1, ..., T$:

Train weak learner using distribution W_t.

- Get weak hypothesis $h_t \rightarrow: X\{-1, +1\}$, with error :

$$\varepsilon_t = Pr[h_t(x_i) \neq y_i] \tag{3.3}$$

- Choose:

$$\alpha_t = \frac{1}{2} ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right) \tag{3.4}$$

- Update:

$$
\begin{aligned}
W_{t+1}(x_i) &= \frac{W_t(x_i)}{Z_t} \times \begin{cases} e^{-\alpha t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\
&= \frac{W_t(x_i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\end{aligned} \tag{3.5}
$$

  Where T is the total number of weak classifiers and Z_t is a normalization factor, which is chosen such that W_{t+1} is a distribution.

Output the final hypothesis:

$$
H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{3.6}
$$

where α_t indicates the weight of a weak classifier h_t. If α_t is large, the corresponding weak classifier plays an important role in the final combination, indicating a relatively the important feature.

Manually extracting the relevant features of village buildings is a challenging task. To improve the classification accuracy, the AdaBoost algorithm instead constructs the weak classifiers from a large number of simple features. Meanwhile, although the input dimension is quite high, the AdaBoost algorithm is robust to over-fitting problems [65]. In our experiment, we automatically generated 500 weak classifiers by applying the sample data to a three-layer decision tree.

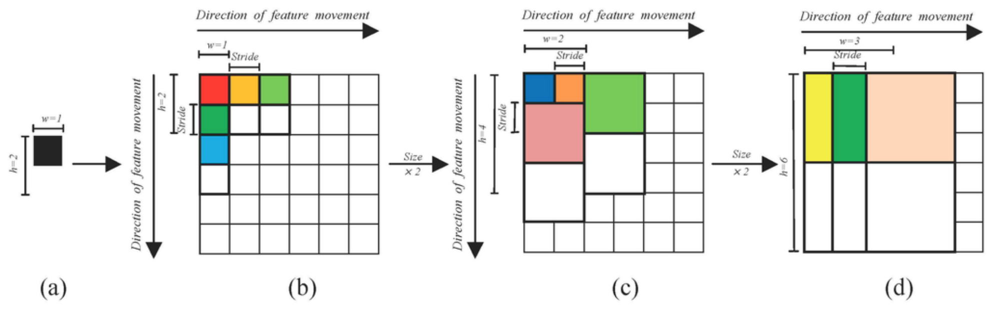


FIGURE 3.3: Haar-like feature's size change and movement: (a) Original 2 × 1 feature; (b) movement of original feature; (c) 4 × 2 feature; and (d) 6 × 3 feature.

Color Feature – To optimize the classifier for detecting the characteristics of color-coded GE images, we utilized four kinds of input samples from the images in Figure 2.1, varying the window size from 1 × 1 to 7 × 7 pixels to show the color information of specific blocks. Given an m × n sized color image patch, AdaBoost organizes the color feature by reshaping the color of each pixel in the patch into an m × n × 3 vector.

Haar-like Feature – Haar-like features are proven to be efficient tools for detecting textural and structural information of buildings [31]. In this study, a large number of Haar-like features are generated for classification. By considering not only the target pixel but also the pixels inside its neighborhood window, the texture and structural characteristics of buildings can be well recognized. The Haar-like features were generated by adopting several basic Haar filters and shifting and scaling them inside the neighborhood window. Figure 3.3 demonstrates the application of a 2 × 1 Haar filter. Here, w and h denote the width and height of the filter, respectively. The original filter is presented in Figure 3.3a. Shifting this filter inside the window, we obtain the various features shown in Figure 3.3b. Similarly, some results of shifting and scaling operations are presented in Figure 3.3c,d. In these panels, the original filter was enlarged to 4 × 2 and 6 × 3, respectively.

3.2.4 Neural Networks

Inspired by the biological neural networks that constitute by the biological neural networks, artificial Neural Networks offers an alternative way to perform machine learning when we have complex hypotheses with many features [66].

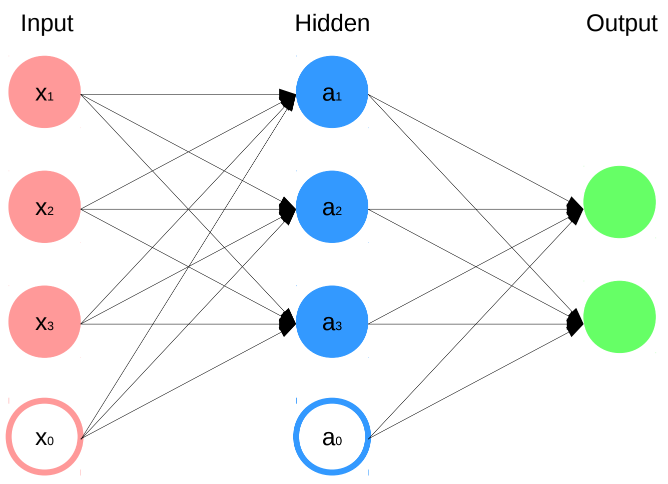


FIGURE 3.4: Neural Networks.

Intuitively, the structure of basic neural networks as shown in the Figure 3.4. based on a collection of connected units called artificial neurons, like neurons in brain, each connection between neurons can transmit a signal to another neuron [67]. After receiving signal, the neuron can process the signals and conduct further propagation. All of neurons are organized in layers, and there is no connection within a same layer. Different layers may perform different kinds of function such as extracting different features, the signals propagate from the first input layer to the last output layer.

We use sigmoid in equation 3.7 as an example to be the activation function to implement classification [68], and θs parameters are also called weights, intermediated layers of neurons between input and output layers are called hidden layers. We label these hidden layer nodes as $a_0^l, ..., a_n^l$ and call them activation units.

$$g(x) = \frac{1}{1 + exp^{-\theta^T x}} \tag{3.7}$$

Final hypothesis use $h_\theta(x)$. Only consider input and output, a simplistic representation looks like:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^2 \\ a_2^2 \\ a_3^2 \end{bmatrix} \rightarrow h_\theta(x) \tag{3.8}$$

a_i^l here refers to the activation of neuron i in layer l, and Θ means matrix of weights controlling function mapping from layer l to layer $l+1$. The values for each activation neuron is obtained as follows:

$$\begin{aligned} a_1^{(2)} &= g\big(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3\big) \\ a_2^{(2)} &= g\big(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3\big) \\ a_3^{(2)} &= g\big(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3\big) \end{aligned} \tag{3.9}$$

Final output $h_\Theta(x)$ generated by:

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}) \tag{3.10}$$

In our experiment, we train the model by inputing HRRS images and relative ground truth labels. Hypotheses result obtained by signals forward propagate from the first layer to the end layer. Generated by comparing hypotheses with ground truth labels, residual error used to implement the back propagation and adjust the weight between each layer. The corresponding process will be iterated several times until the model converge into a promising status. Like other machine learning methods, test procedure can be easily implemented by putting HRRS images, feature in which can be extracted, after propagating to final, the classified result can be generated.

Except for using in computer vision field, Neural networks also have been used in fields including speech recognition, video games and many other domains.

3.2.5 Super Vector Machine

The machine learning methods such as support vector machine (SVM) [34], which maximizes the margin in high-dimensional feature spaces using kernel methods for the samples, are introduced for classification. In general, an SVM training algorithm makes a plain called hyperplain in the space by given a set of training examples which marked as belonging to one or the other of two categories. As shown in Figure 3.5, two categories: green square and blue triangle, are divided by a clear gap that is as wide as possible. When new examples come, they also mapped into the same space separated by red hyperplain and predicted to belong to a category based on which side of the gap they fall.

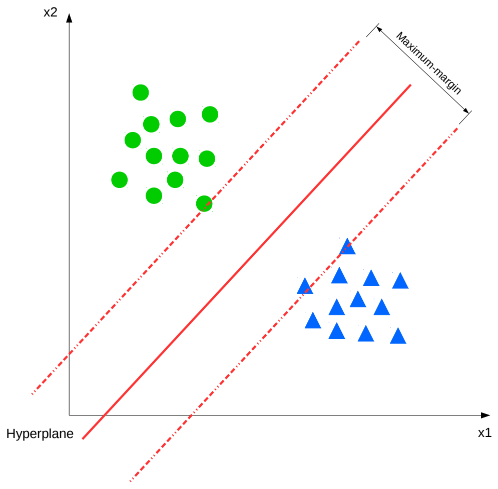


FIGURE 3.5: Super Vector Machine.

Here take linear SVM as example, a training dataset of n points of the form:

$$(\vec{x_1}, y_1), ..., (\vec{x_n}, y_n) \tag{3.11}$$

are given, where y_i are category labels 1 or -1, indicate the class to which the point $\vec{x_i}$ belongs. Target of SVM is to find the "maximum-margin hyperplane" that can separate all the $\vec{x_n}$ which belongs to label 1 from the rested dataset. The hyperplane can be written as the set of points $\vec{x_n}$ satisfying:

$$\vec{\theta} \cdot \vec{x} - b = 0 \tag{3.12}$$

where $\vec{\theta}$ is the normal vector to the hyperplane. In case the data are not linearly separable, the loss function can be formulated as:

$$\left[\frac{1}{n}\sum_{i=1}^{n} max(0, 1 - y_i(\vec{\theta} \cdot \vec{x_i} - b))\right] + \lambda\|\vec{\theta}\|^2 \tag{3.13}$$

where λ is the regularization part used to avoid over-fitting.

In our experiment, first, features of building and non-building objects are extracted, and then, SVM makes a hyperplain to separate all these features depending on ground truth labels. Buildings in testing dataset can be identified in the same way.

3.3 Convolutional Neural Networks

3.3.1 Introduction

CNN was inspired by biological entities. When multilayer networks are trained with gradient descent algorithms, they can learn complex nonlinear mappings from a high- to a low-dimensional feature space, where classification is evident [69]. Importantly, the simplified low-dimensional space can largely restore the information of high-dimensional features, and it mines deeper information of the input high-dimensional features. CNN vertically concatenates an m × n RGB image patch into a 3m × n matrix. Here, we briefly introduce the framework of CNN; the details are provided in [44]. When implementing CNN, we utilized the DeepLearnToolbox and Keras library developed by [70] and [71] respectively.

To simplify the high-dimensional feature, the approach utilizes a multilayer system, including convolution and subsampling layers, as shown in Figure 3.7. Convolution can enhance the raw signal while decreasing the noise signal; the subsampling layer can utilize the correlation of the contiguous pixels, pooling the feature into lower dimensions without losing meaningful information (Figure 3.6). With the increase of layers, the dimension decreases but the number of features increases.

To restore the information of high-dimensional features to the greatest extent, the key point is back propagation [72]. Similar to neural networks, to connect layers i and j, one important parameter is the propagation weight W_{ij}, and another is the bias b_i. At the beginning, W_{ij} and b_i are randomly decided as 0. Then, we perform step-forward propagation

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)} \tag{3.14}$$

$$a^{(l+1)} = f\left(z^{(l+1)}\right) \tag{3.15}$$

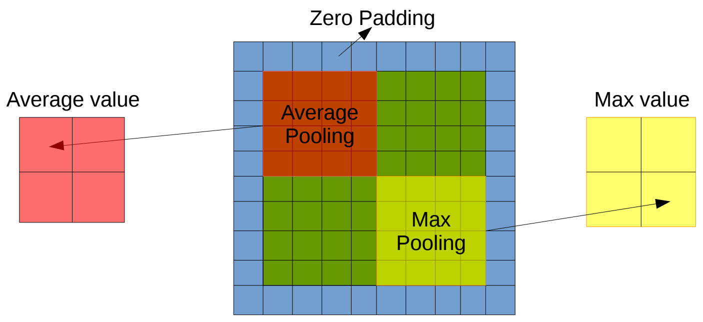


FIGURE 3.6: Pooling and Padding.

where $z^{(l)}$ denotes the total weighted sum of inputs to all the units in layer l, $f(X)$ is the activation function, such as a sigmoid, $a^{(l)}$ denotes the activation (meaning output value) of all the units in layer l. After the iteration, output $h_{W,b}(X)$ is produced. Due to the primitive definition of W_{ij} and b_i, there must be some error between the result and the true value. Two kinds of error are important, which can measure the accuracy of the network.

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} - \delta_j^{(l+1)} \right) f'\left(z_i^{(l)}\right) \tag{3.16}$$

$$\begin{aligned} \delta_i^{(nl)} &= \frac{\partial}{\partial z_i^{(n_t)}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 \\ &= -\left(y_i - a_i^{(n_l)}\right) f'\left(z_i^{(n_t)}\right) \end{aligned} \tag{3.17}$$

where δ^l refers to the error in each output unit i in each layer l; it measures how far the corresponding node is responsible for any error in the output. δ^{n_l} denotes the error for each output unit i in the output layer n_l. In addition to the error to get the optimum solution of W_ij and b_i, we utilize an efficient iteration algorithm called a gradient descent,

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) \tag{3.18}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b) \tag{3.19}$$

where α is the learning rate and the definition of $J(W, b)$ is an average sum-of-squares error term. We can efficiently compute the solution of the partial derivatives using back propagation. Incorporating $\delta_i^{(n_l)}$ and $\delta_i^{(l)}$, we can get

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) = a_j^{(l)} \delta_i^{(l+1)} \tag{3.20}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W,b) = \delta_i^{(l+1)} \tag{3.21}$$

After the iteration, W_ij and b_i are updated into the optimum solution; then we can calculate the optimum classification $h_{W,b}(X)$.

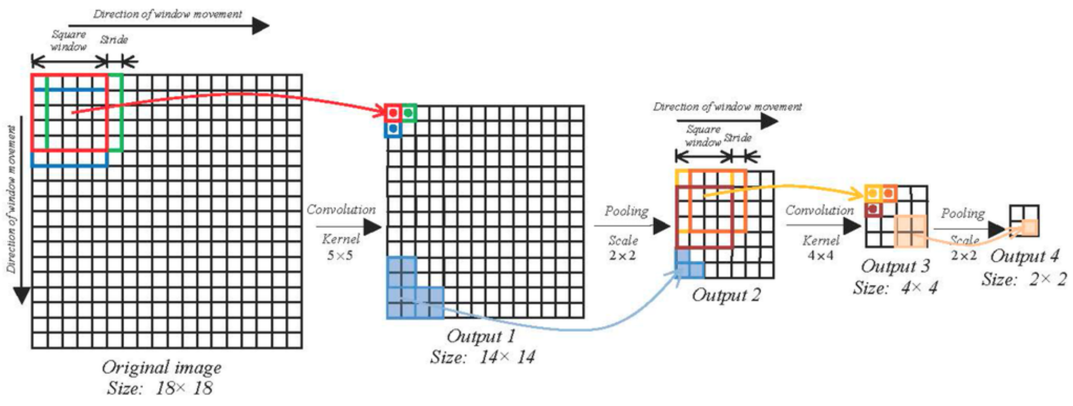


FIGURE 3.7: Convolution and Pooling process.

3.3.2 Basic Structure

Based on the multilayer networks, the parameters of convolution subsampling and other related cores are very significant. In our experiment, we formulated a simple architecture to compare the CNN with other machine learning methods.

In this basic architecture, the sequence of middle layers contains four layers $Conv_1$, $Pool_1$, $Conv_2$, and $Pool_2$. The data of each layer and some other information set is shown in Table 3.1, and the concrete process is shown in Figure 3.7.

Layer	Output Shape	Kernel Size	Scale	Param
Input	(18,18,3)	-	-	-
Conv 1	(14,14,6)	(5,5)	-	456
Pooling 1	(7,7,6)	-	2,2	0
Conv 2	(4,4,12)	(4,4)	-	8118
Pooling 2	(2,2,12)	-	2,2	0
Flatten	(48)	-	-	0
Dense	(1)	-	-	49
Activation	(1)	-	-	0
Total params: 1,669				

TABLE 3.1: Basic CNN structure settings

Both the kernel size and pooling scale can turn the sample from a high- to low-dimensional feature space. As set initially, the size of sample window is 18 × 18 in 3D; after going through the convolution for the first time, the size changes to 14 × 14, which becomes 7 × 7 after the first pooling; in the following procedure, convolution 2 and subsampling 2 are similar to the previous ones, whereby the size becomes 4 × 4 and 2 × 2, respectively. As observed from the result, size can be scaled to low dimensions after the corresponding process. Output maps decide the number of kernel core, influencing the number and category of the training data.

Although the preceding basic structure might own good feasibility in exploring feature and classification, actually, it is far away from mining the potential capability of CNN. In this experiment we plan to propose other architectures to achieve better result.

ImageNet [73], which organized according to the WordNet hierarchy is a large visual image dataset designed for use in visual object recognition research. As the most attractive point, characteristic of very large-scale would be tremendously helpful to researchers. Category here means any object possibly described by multiple words or word phrases, and ImageNet aims to provide on average 1000 images to illustrate each category, what's more, images are quality-controlled and human-annotated. This existing large-scale image database provides researchers convenience and help to tackle problems such as data collection and labeling.

Based on ImageNet, From 2010, An annual competition called ImageNet Large Scale Visual Recognition Challenge(ILSVRC) [74] held where research teams submit programs that classify and detect objects and scenes. It's important to note that in 2012, AlexNet [75] achieved the error rate from previous best stated 25% to 16%, and in the next couple of years, higher accuracy pattern recognition result achieved by famous models such as GoogLeNet [76], VGGNet [77], SqueezeNet [78] and ResNet[79].

To make the most of these mentioned state-of-the-art models, we carefully modified the their architectures by considering the characteristics of input HRRS image and our identification targets. Here, we propose self-designed structures called AlexNet-like, GoogLeNet-like, VGGNet-like and SqueezeNet-like, and utilize them to implement the identification.

3.3.3 Architecture Modification

In general, the principle of modifying CNN architecture is highly based on analyzing learning curve of both training and cross validation results. Except accuracy, there are two important index need to be pointed out: Bias and Variance. The bias is error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs. The variance is error from sensitivity to small fluctuations in the training set. High variance can cause over-fitting: modeling the random noise in the training data, rather than the intended outputs.

Intuitively, here we take target practice as an example. As shown in the Figure 3.8, in case a shooter cannot shoot central target, the shooting result far away from the center and clustered in other place is called bias; while the shooter can shoot target, but results are quite scattered around the center is called variance.

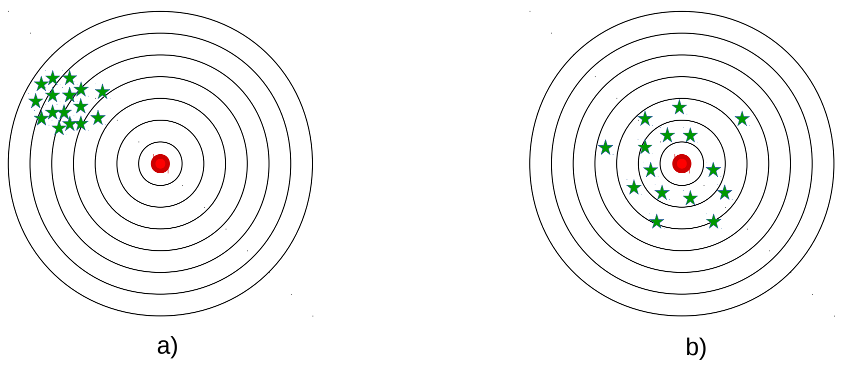


FIGURE 3.8: Bias and Variance.

In this experiment, both Bias and Variance will course severe problems. There's no meaning of adding training samples when the model suffers from high bias, and the buildings can not be identified correctly both in training and cross validation. Here are some ways to solve it:

- Modify the input training data accuracy. It means the training HRRS images and corresponding labels both in buildings and other land features must be as accurate as possible.
- Decrease the regularization coefficient [80] λ, which can solve under-fitting problems.
- Add feature amount, just like use more complex CNN structures.

When we face the problem of Variance, the training result is rather good while the cross validation curve is bad or unstable. In details:

- Add more training samples would be helpful. The data augmentation such as adding more training HRRS images dataset considering the diversity.
- Increase the regularization coefficient λ, which can solve over-fitting problems.
- Decrease the feature amount, can use the method such as Dropout [81].

Here, we modify our model depends on the preceding principles. We take VGGNet-like(will be introduced in later section) structure as an example to explore how to configure the CNN architecture based on the characteristics of VGGNet. The final promising structure is generated by gradually enhance a sample initial structure. Considering the experimental requirement, the four parameters to be evaluated in our experiment are iteration amount, number of filters, the depth of architecture, and sample window size. These parameters are connected in a way that determines the total number of units and the weight values in the whole structure.

The original architecture is based on the basic CNN structure 3.1, in which, the size of sample window is 18×18, 2 convolutional layers followed by average pooling are implemented with 6 and 12 filters respectively. To enhance the previous architecture, the iteration amount is chosen amount $t = [50\ 150\ 300\ 600\ 1000]$; the window size s is the surrounding area around the pixel to be classified and is chosen from 14 to 30 with interval $= 2$; number of filters is configured by multiplying the original filter amount, $f = [3\ 9\ 25\ 50\ 100]$ is the multiple; the added convolutional layer amount y from 2 to 12 with interval $= 2$. We evaluate the effect of each parameter depends on both the result and computation time, and then integrate all the optimal settings to obtain the promising VGGNet-like architecture finally.

3.3.3.1 Iteration

First, we evaluate the influence of iteration on the accuracy, and the CNN structure is based on the original one. Table 3.2 shows the relationship between iteration and result, while Figure 3.9 shows the learning curve depends on Accuracy and Loss error.

	Training				Testing		
Iteration	Acc%	Kappa	Epoch(s)	Total(s)	Acc%	Kappa	Total(s)
50	93.02	0.79	1.49	74.29	94.58	0.42	1.74
300	95.31	0.86	1.41	424.20	97.29	0.60	1.59
600	97.14	0.92	1.37	819.72	97.05	0.60	1.77
1000	97.06	0.91	1.38	1376.66	96.61	0.57	1.67

Table 3.2: Relationship between iteration and accuracy

From the result, it infers that after about 400 iterations accuracy dose not increase anymore in cross validation result. There's no over-fitting and the result even unstable more than 1000 iterations. According to the principle, we need to increase feature amount by adding layer and filter quantity.

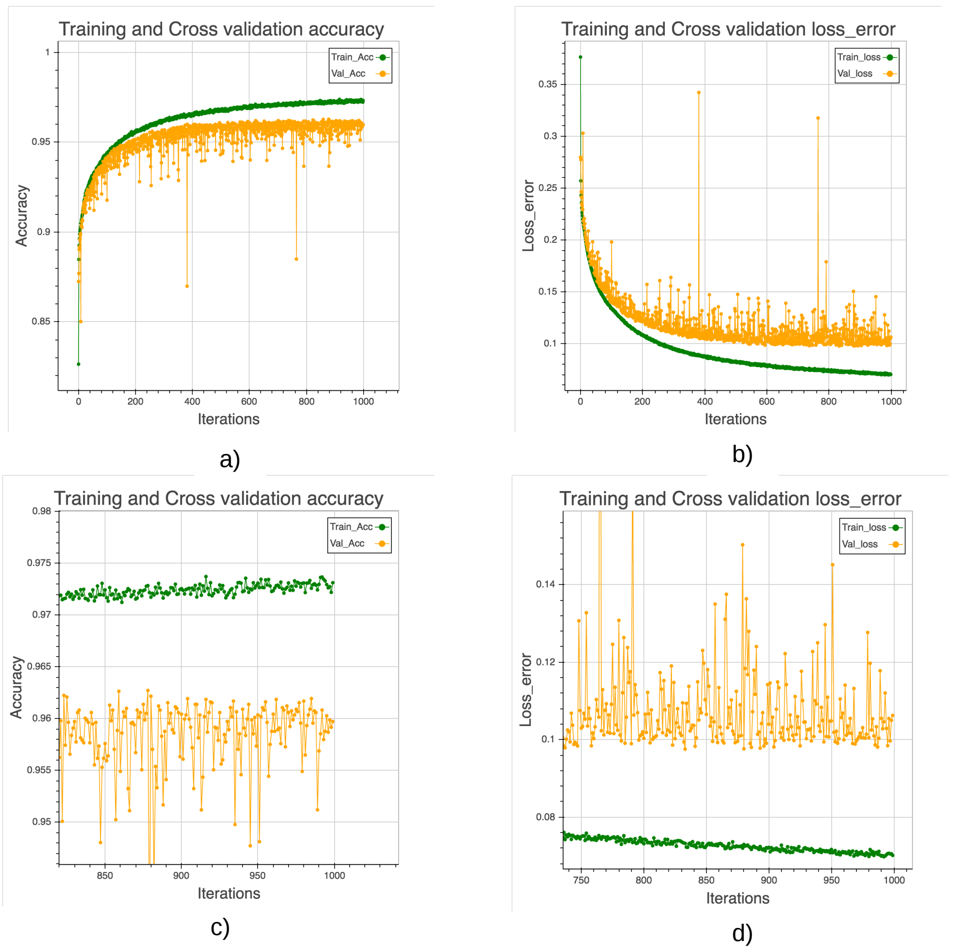


FIGURE 3.9: Influence of iteration.

3.3.3.2 Filter

In general, the more filters, the more features can be extracted. Here we gradually add feature amount from the original to f = [3, 9, 25, 100, 200] times. As shown in the Table 3.3, when the filter amount reach 25 times, the best training and testing result can be generated, and the model can achieve 98.98% and 0.83 in testing accuracy and Kappa respectively. Also, from the learning curve (Figure 3.10), until 200 and 300 times the model can still do not face over-fitting problems, it infers that the feature amount has not reached saturated yet. When adding mode filters, the model attend to converge faster, but considering both accuracy and computation time, the filter amount which can obtain enough good result would be suitable.

		Training				Testing		
Str	Para	Acc(%)	Kap	Epoch(s)	Total(min)	Acc(%)	Kap	Total(s)
Ori	1669	95.31	0.86	1.41	7.06	97.29	0.60	1.59
× 3	11917	98.94	0.97	1.57	7.85	98.15	0.72	1.93
× 9	97957	99.19	0.98	2.29	11.46	98.22	0.73	3.31
× 25	0.73M	99.73	0.99	7.09	35.44	98.98	0.83	10.16
× 100	11.57M	99.69	0.99	83.09	415.46	98.79	0.80	88.00
× 200	46.18M	97.54	0.92	299.06	1459.27	98.14	0.70	5.67

TABLE 3.3: Relationship between filter amount and accuracy.

Although the high accuracy could be achieved, the model still suffer from unstable convergence, and it can not be stable even add 200 times filter amount. In the next part, we will exploring the influence of architecture depth.

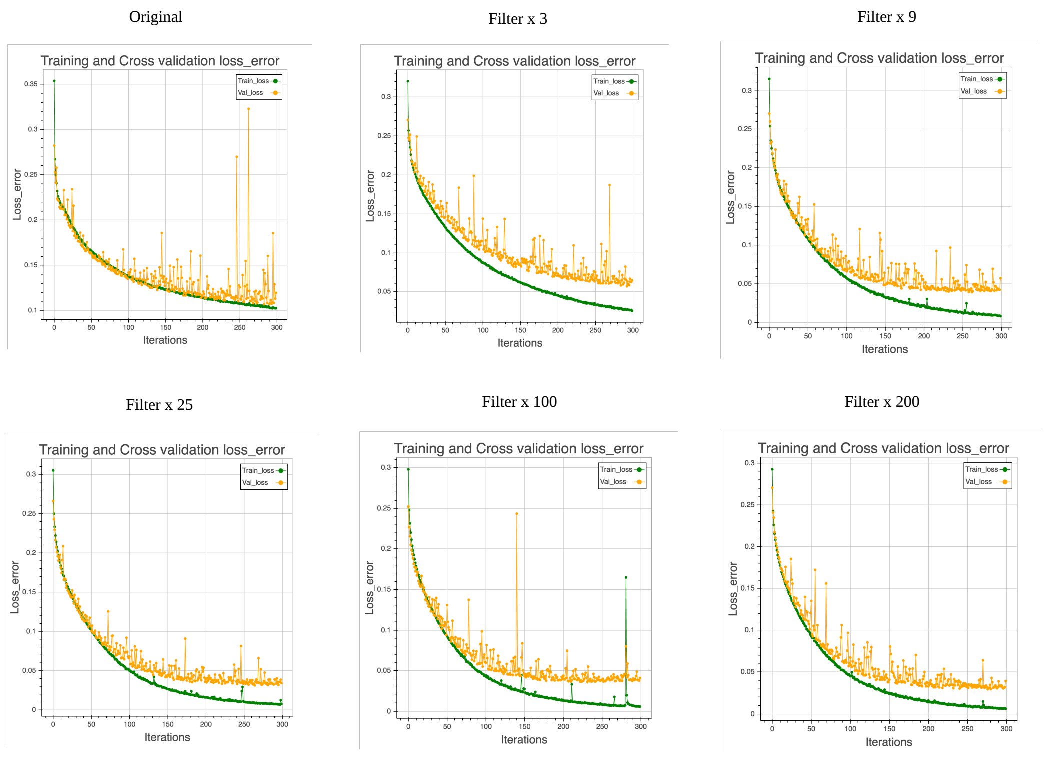


FIGURE 3.10: Influence of filter amount.

3.3.3.3 Depth

Convolutional Neural Networks, as a very important branch of Deep Learning, its preponderance highly based on the depth of architecture. by mining deeper and abstracter features and information from identification target, usually network in high depth could achieve better accuracy. Resent years, owning to the improvement of computation capability and hardware, the possibility of construct and compute very deep network has come possible. The recent state-of-the art architecture such as VGGNet [77] and ResNet [79] make the most of this principle.

		Training				Testing		
Structure	Para	Acc(%)	Kap	Epoch(s)	Total(min)	Acc(%)	Kap	Total(s)
Ori	1669	93.02	0.79	1.49	74.29	94.58	0.42	1.73
+ 2 Con	4891	96.30	0.89	2.06	103.16	96.84	0.58	2.60
+ 4 Con	8133	96.21	0.88	2.73	136.93	98.06	0.68	3.55
+ 6 Con	11335	97.45	0.92	3.44	172.09	98.18	0.70	4.65
+ 8 Con	14557	97.54	0.92	4.16	208.19	98.14	0.70	5.67
+ 10 Con	17779	97.80	0.93	4.94	247.03	97.67	0.65	6.73
+ 12 Con	21001	78.97	0.00	6.18	308.87	97.53	0.00	7.81

TABLE 3.4: Relationship between depth and accuracy.

In this experiment, we explore the effect of depth on CNN model of identification buildings in rural environment. We add convolutional layers from original 2 layers to 14 layers with interval = 2 . Both training testing settings and results as shown in the Table 3.4.

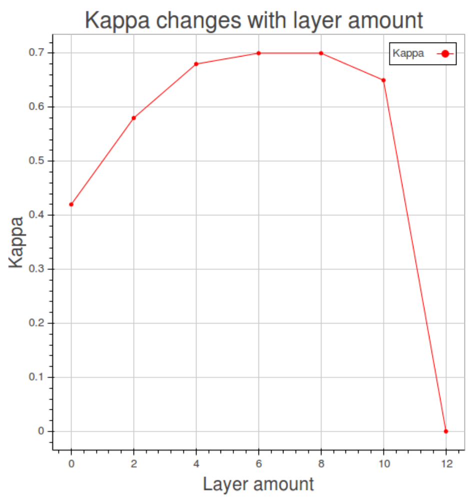


FIGURE 3.11: Gradient vanishing.

At beginning, model accuracy become higher and higher with the increase of depth. However, when convolutional layer amount bigger than 12, the network becomes stocked and even lose the identification capability anymore (Figure 3.11). After rigorous analysis, such problem would be caused by gradient vanishing [82].

Caused by different factors, gradient vanishing is an usual problem appears in CNN. As we know, CNN is based on gradient decent and back propagation, when implementing gradient decent, activation function is also be derivative, signal will be activated into saturated or diverged region. After that, with propagation processing, such phenomenon will be propagated in the whole model, and will cause the corresponding gradient vanishing and exploring.

The gradient vanishing in this experiment is also cased by the preceding reasons. There are several ways of solving it. For instance, use unsaturated activation such as Relu can relief the problem to a certain degree; Also, we can use batch normalization [83] which conducting feature scaling after convolution and could void the model fall into vanishing and exploding region; In this experiment we choose the simplest way by adding depth to the most suitable degree, which would get promising result while avoid suffering from gradient vanishing problem. Considering the efficiency and accuracy, here we add 4 convolutional layers into original structure and achieve the testing accuracy and Kappa with 98.18% and 0.70 respectively.

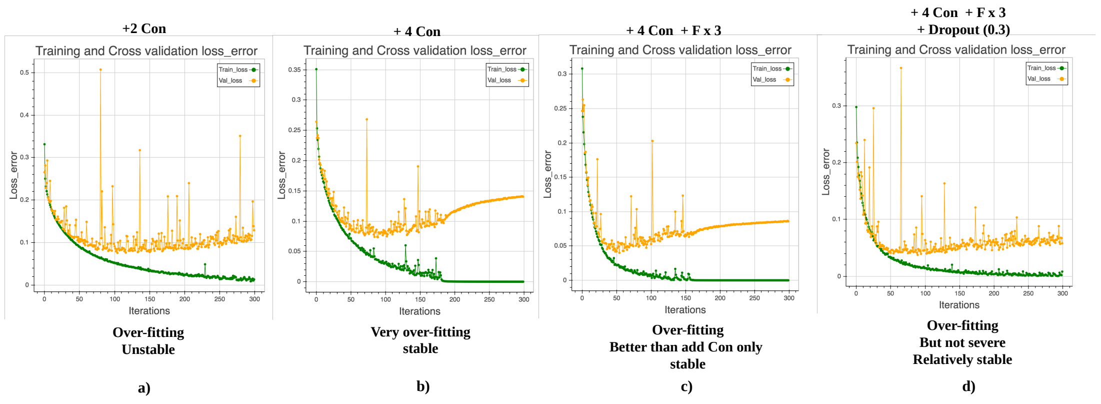


FIGURE 3.12: Influence of filter, depth and Dropout.

Summarizing and comparing the previous factors, here we choose the model with 6 convolutional layers, two pooling layers and 3 times filter amount comparing with the original model. However, by analyzing the learning curves (Figure 3.12c) we find that the model severely suffered from over-fitting problem. Here we add Dropout processing (Figure 3.12d) which git rid of 30% feature amount after the final flatten layer, which relief over-fitting to a certain degree and achieve overall accuracy and Kappa with 99.26% and 0.86 respectively as shown in Table 3.5.

		Training				Testing		
Structure	Para	Acc(%)	Kap	Epoch(s)	Total(min)	Acc(%)	Kap	Total(s)
+2 Conv	4891	98.31	0.95	2.25	11.12	98.87	0.80	2.53
+ 4 Conv	8133	99.60	0.99	2.78	13.91	98.52	0.76	3.60
+ 4 Conv 3×Filt	33283	99.76	0.99	4.54	22.72	99.07	0.84	6.53
+ 4 Conv 3×Filt Dropout	33283	99.56	0.99	4.62	23.12	99.26	0.86	6.57

TABLE 3.5: Relationship between depth, filter, Dropout and accuracy.

3.3.3.4 Window Size

In window slide based identification, size of input sample is a very significant factor. Considering the image resolution and characteristics of rural environmental buildings, the ideal window size need to be a little bit bigger than the ordinary buildings, meanwhile, the information of building's surroundings also need to be included.

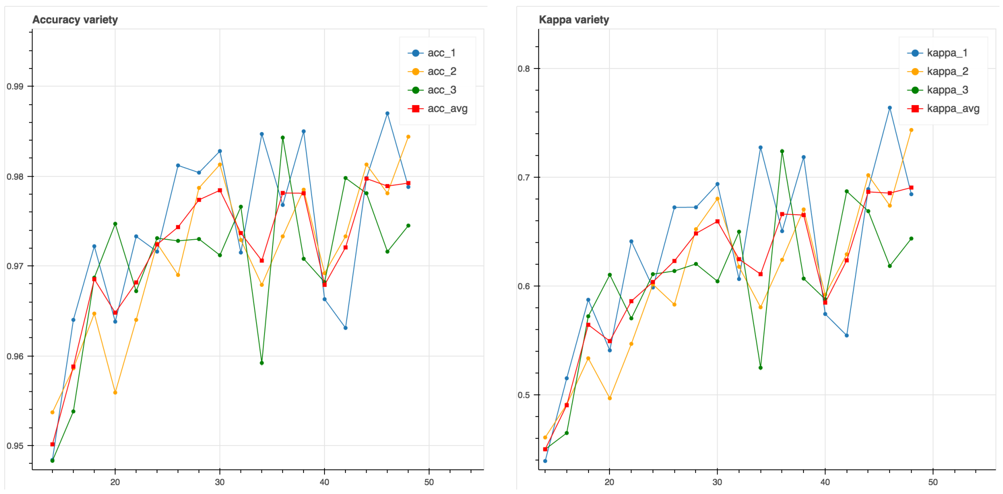


FIGURE 3.13: Window size: scenario 1.

The input sample window size of our original basic architecture is 18 × 18. According to the principle of window slide, here we present three scenarios for exploring window size. First, change size from 14 to 50, while use the original structure. Second, change size from 14 to 50, use more complex structure based on the previous principles such as filter and depth adding. Finally, we focus on some typical window size such as multiple relation. The best sample size is get by train different models based on different window size. Here we provide 10,000 positive and 40,000 negative samples to conduct training respectively.

Window size in the first scenario ranges from 14 to 50 with interval equals to 2, and the parameter amount increases along with window size increasing. Due to the unstable condition of model (samples are randomly chosen), we did the experiment for 3 times and get the average result in red line, as shown in the Fiture 3.13. although the results are different between each other, there's some information can be picked up: result is unstable, and the model fluctuates after size equals to 30, however, the overall trend is accuracy increasing. It means original 18 × 18 window size maybe not the best choice.

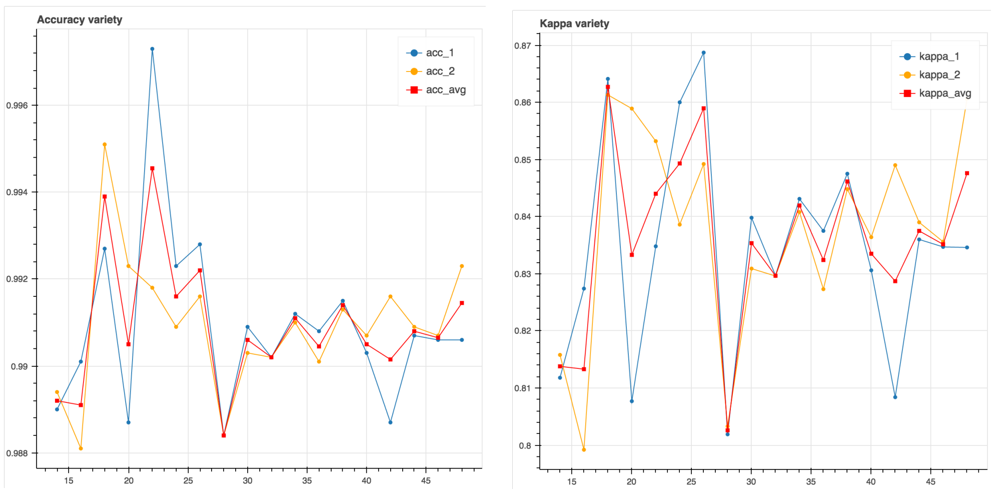


FIGURE 3.14: Window size: scenario 2.

Then, we conduct the second scenario, and explore influence of window size by implementing complex model, it means more features and information can be used. As shown in the Figure 3.14, like in first scenario, model is unstable, always fluctuate, and the best result come from size equals to 26. But whether size in 26 × 26 is the best choice or not still need to be explored.

Here, we propose scenario 3 by comparing the effect of window size in multiple relation, such as size 14 with 28, 16 and 32, etc. As shown in Figure 3.15, focus on result of multiple relation sizes with the same structure: double size will contain the information of small one, and the blue part is their common part.

From the testing result (Figure 3.16), by implementing simple structure, double size window could get better result, because it not merely contains the information of small size, but also owns other abundant information. However, if we implement complex structure, although double window size contains more information, can not get better result any more (Figure 3.17).

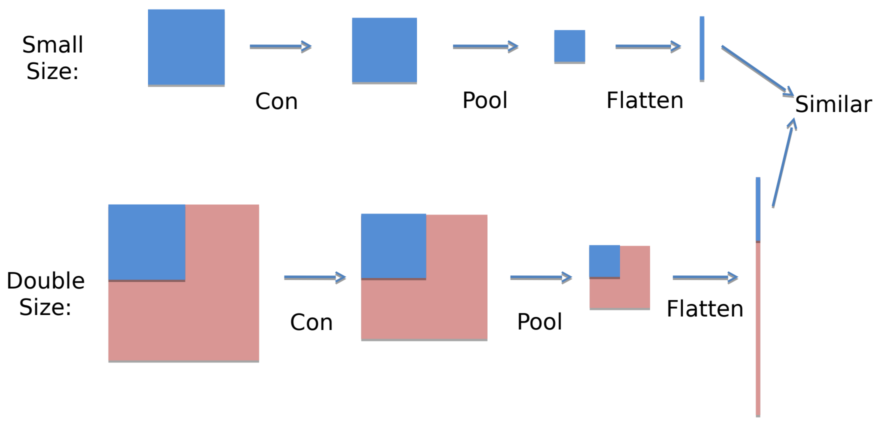


Figure 3.15: Window size: scenario 3.

In conclusion, with a same CNN structure, bigger window size can obtain more features and parameters, but other methods such as adding filters and depth can also increase feature amount. In scenario 1, accuracy gradually increases due to the insufficient feature; in scenario 2, accuracy becomes fluctuate, and almost have no difference between

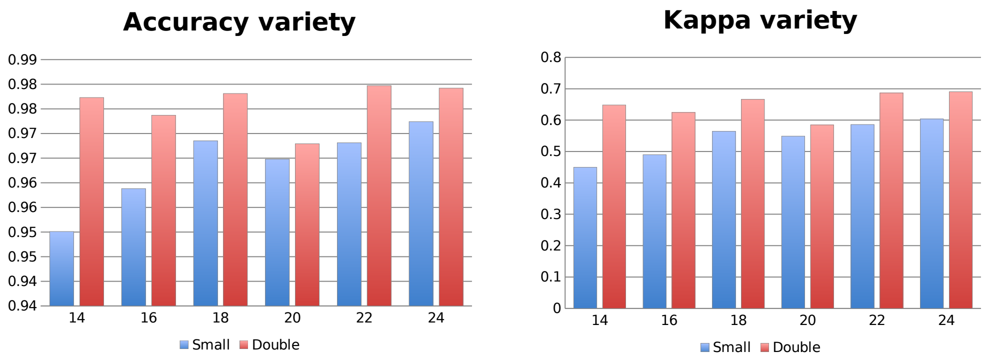


Figure 3.16: Window size: scenario 3 with simple structure.

each window size; the scenario 3 can infer that when the feature is insufficient, double size samples include all the information of small ones, which would lead good result, however, when we use complex structure which contains sufficient feature, bigger window size can no longer lead good result. Therefore, too big window size might get redundant and useless feature, which lead bad results. In this experiment, Choose a window size half bigger than the ordinary buildings, with enough feature and depth is important.

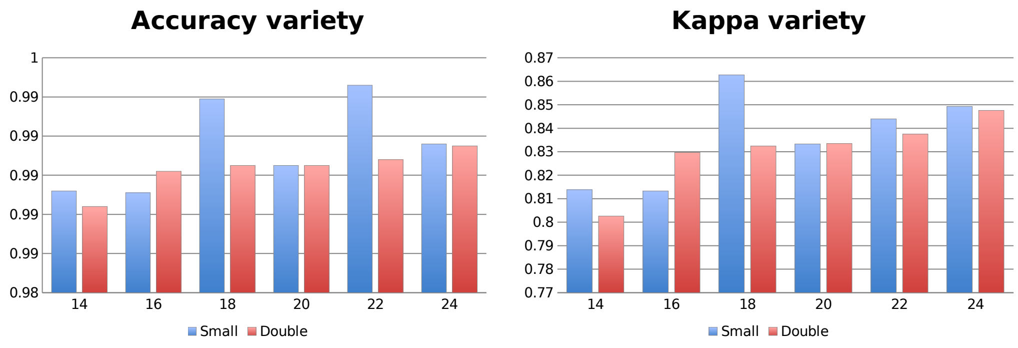


FIGURE 3.17: Window size: scenario 3 with complex structure.

3.3.4 AlexNet-like

AlexNet [75] is a revolution architecture of CNN. The parallel and merged structure makes it able to extract two sets of features while sharing information between each other. The Deep Convolutional Neural Networks is elaborately formulated with very high accuracy, in case modify the structure even in a little will influence its capability. Moreover, by running on GPUs implemented in CUDA, training CNN model depending on large-scale dataset become feasible.

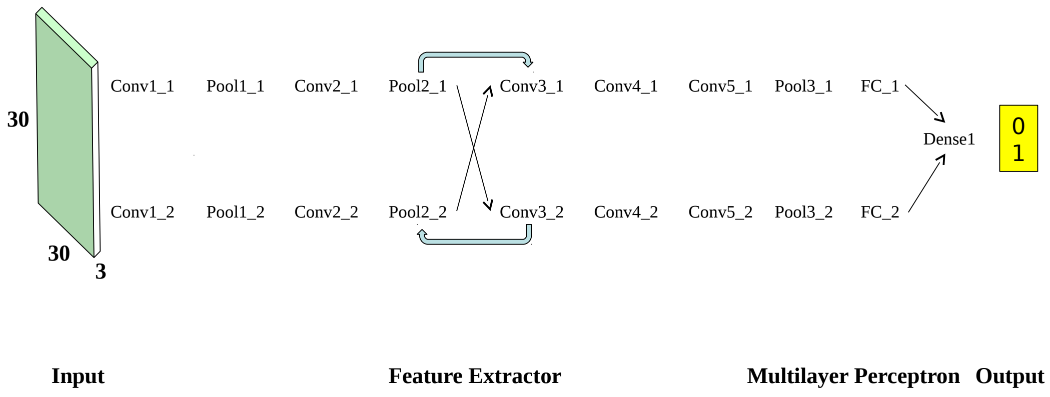


FIGURE 3.18: AlexNet-like architecture.

There are some tricks of AlexNet in both structure and processing. First of all, Just by conducting subsampling and feature scaling [84], the preprocessing procedure is completed. Then, rather than saturated activation method, AlexNet implements Relu:

$$f(x) = max(0, x) \tag{3.22}$$

which is very efficient and 6 times faster than tanh [85], can avoid gradient vanishing and exploding to a certain degree. Third, with the help of parallel structure, AlexNet can be efficiently trained on multiple GPUs, and every GPU shares half kernels. To reduce over-fitting, AlexNet uses tricks such as data augmentation, Dropout and overlapping pooling structure. Finally, Stochastic gradient descent (SGD) method [86] is used with configuration such as weight decay, momentum and learning rate gradually reduce.

In this experiment, we rigorously modify the AlexNet into AlexNet-like architecture (Figure 3.18) according to the settings in Table 3.6. To match our target, we reduce the input size to 30 × 30, and modify some intern settings such as quantity of filter and kernel size based on the modification principle, which make the total parameter reduces from about original 60 million to 67,665.

Layer	Output Shape	Kernel Size	Scale	Param	Connect to
Input	(30,30,3)	-	-	0	-
Padding 1	(32,32,3)	-	1	0	Input
Conv 1_1	(28,28,9)	(5,5)	-	684	Padding 1
Conv 2_1	(28,28,9)	(5,5)	-	684	Padding 1
Pooling 1_2	(13,13,9)	-	3,2	0	Conv 1_1
Pooling 2_2	(13,13,9)	-	3,2	0	Conv 2_1
Conv 1_3	(13,13,9)	(5,5)	-	2034	Padding 1_2
Conv 2_3	(13,13,9)	(5,5)	-	2034	Padding 2_2
Pooling 1_4	(6,6,9)	-	3,2	0	Conv 1_3
Pooling 2_4	(6,6,9)	-	3,2	0	Conv 2_3
Merge 1_1	(6,6,18)	-	-	0	Pooling 1_4
					Pooling 2_4
Merge 1_2	(6,6,18)	-	-	0	Pooling 1_4
					Pooling 2_4
Conv 1_5	(6,6,18)	(4,4)	-	5202	Merge 1_1
Conv 2_5	(6,6,18)	(4,4)	-	5202	Merge 1_2
Conv 1_6	(6,6,18)	(4,4)	-	5202	Conv 1_5
Conv 2_6	(6,6,18)	(4,4)	-	5202	Conv 2_5
Conv 1_7	(6,6,18)	(4,4)	-	5202	Conv 1_6
Conv 2_7	(6,6,18)	(4,4)	-	5202	Conv 2_6
Pooling 1_8	(2,2,18)	-	3,2	0	Conv 1_7
Pooling 2_8	(2,2,18)	-	3,2	0	Conv 2_7
Flatten 1_9	(72)	-	-	0	Conv 1_8
Flatten 2_9	(72)	-	-	0	Conv 2_8
Merge 2	(144)	-	-	0	Flatten 1_9
					Flatten 1_9
Flatten 3	(100)	-	-	14500	Merge 2
Dense	(1)	-	-	101	Flatten 3
Activation	(1)	-	-	0	Dense
Total params: 51,249					

TABLE 3.6: AlexNet-like architecture

3.3.5 VGGNet-like

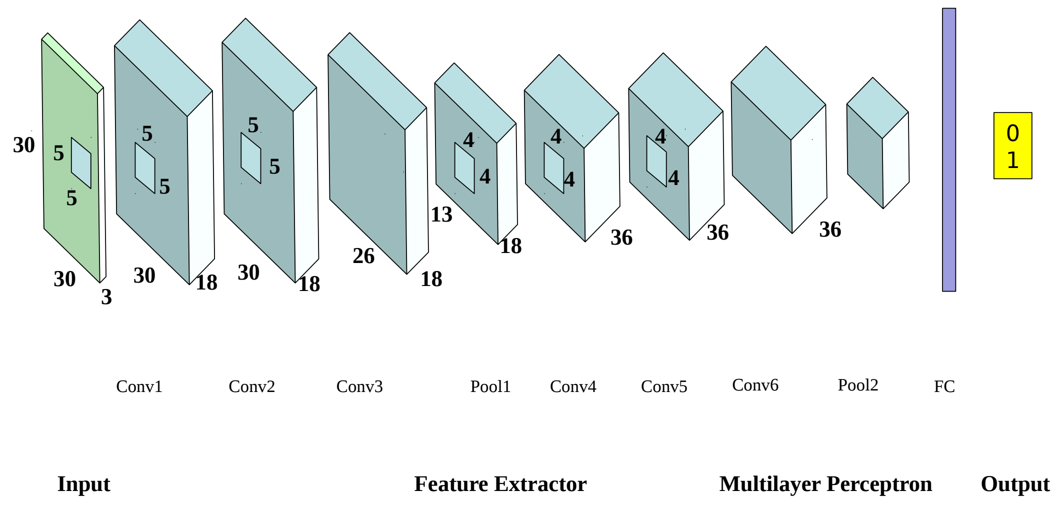


FIGURE 3.19: VGGNet-like architecture.

VGGNet [77] is the abbreviation of Very Deep Convolutional Networks. As its name mentioned, VGGNet addresses the important aspect of Convolutional Networks architecture design: its depth, which makes it be able to mine very deep and abstract features. The architecture steadily increase the depth of networks by adding more convolutional layers, and filter quantity also gradually increase from begging to the end. In terms of filter, very small convolutional filters with size 3 × 3 are used in all the layers, and the used 1 × 1 filter can be seen as a linear transformation of the input channels. Other layers such as Zeroppading (Figure 3.6), Maxpooling, Flatten, Dense and dropout are also increase its feasibility.

There are two kinds of VGGNets: 16 and 19 layers, which different in layer amount. The two types own parameter quantity in 138M and 144M respectively, while the flatten layer contains about 100M parameters. To avoid over-fitting, the method here is get rid of redundant features via Dropout.

In this experiment, we propose a VGGNet-like architecture (Figure 3.19), which owns the capability to identify buildings in rural environment based on HRRS images. the main idea of modification is revising input data shape, decreasing layer and filter amount while considering the characteristic of VGGNet. After modification, the parameter amount decreases from 140M to 70,453 (Table 3.7), which make the model easy to be trained.

Layer	Output Shape	Kernel Size	Scale	Param
Input	(30,30,3)	-	-	-
Conv 1	(30,30,18)	(5,5)	-	1368
Conv 2	(30,30,18)	(5,5)	-	8118
Conv 3	(26,26,18)	(5,5)	-	8118
Pooling 1	(13,13,18)	-	2,2	0
Conv 4	(13,13,36)	(4,4)	-	10404
Conv 5	(13,13,36)	(4,4)	-	20772
Conv 6	(10,10,36)	(4,4)	-	20772
Pooling 2	(5,5,36)	-	2,2	0
Flatten	(900)	-	-	0
Dropout	(900)	-	-	0
Dense	(1)	-	-	901
Activation	(1)	-	-	0
Total params: 70,453				

TABLE 3.7: VGGNet-like architecture

3.3.6 GoogleNet-like

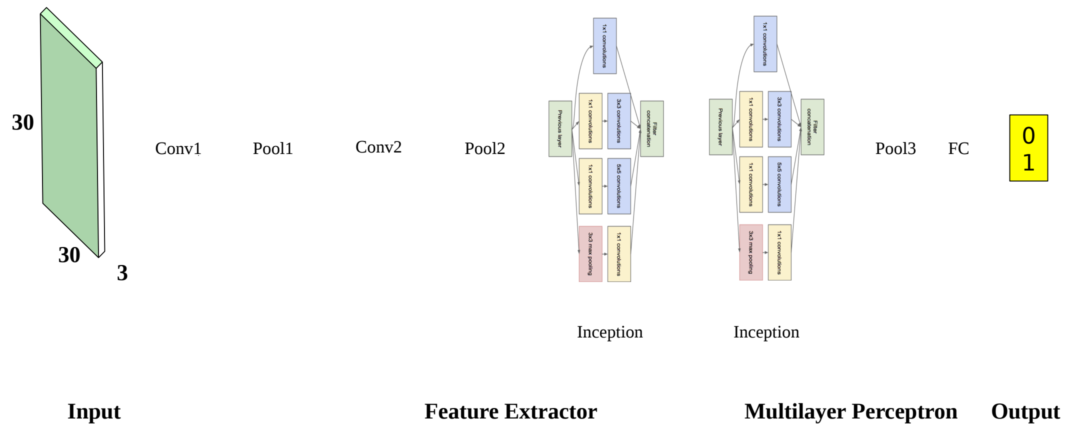


FIGURE 3.20: GoogleNet-like architecture.

The main innovation of GoogleNet [76] is using the architecture called Inception. In general, Inception is a network in network structure, and the optimal local sparse structure of a vision network spatially repeat from the beginning to the end. Three Inception structures which used in different circumstances are introduced, in common, 1 × 1 convolutions are used in Inception to compute reductions before the expensive 3 × 3 and 5 × 5 convolutions.

GoogleNet provides us an inspiration of how to building high capability architecture. Most of the progress is not only rely on more powerful hardware, large datasets and bigger models, but mainly a consequence of new ideas, algorithms and improved network architectures.

Learning from GoogleNet, in this experiment we build our GoogleNet-like structure (Figure 3.20). The Inception architecture is remained, while modify the layer and filter amount and sequence. Detailed settings in Table 3.8.

Layer	Output Shape	Kernel Size	Scale	Param	Connect to
Input	(30,30,3)	-	-	0	-
Padding 1	(32,32,3)	-	1	0	Input
Conv 1	(28,28,18)	(5,5)	-	1368	Padding 1
Padding 2	(30,30,18)	-	1	0	Conv 1
Pooling 1	(14,14,18)	-	3,2	0	Padding 2
Conv 2	(14,14,18)	(1,1)	-	342	Pooling 1
Conv 3	(14,14,18)	(5,5)	-	8118	Conv 2
Padding 3	(16,16,18)	-	1	0	Conv3
Pooling 2	(7,7,18)	-	3,2	0	Padding 3
Conv 4_1_1	(7,7,20)	(1,1)	-	380	Pooling 2
Conv 4_1_2	(7,7,20)	(3,3)	-	3620	Conv 4_1_1
Conv 4_2_1	(7,7,20)	(1,1)	-	380	Pooling 2
Conv 4_2_2	(7,7,20)	(5,5)	-	10020	Conv 4_2_1
Pooling 4_3_1	(7,7,18)	-	3,1	0	Padding 2
Conv 4_3_2	(7,7,20)	(1,1)	-	380	Pooling 4_3_1
Conv 4_4	(7,7,20)	(1,1)	-	380	Pooling 2
Merge 1	(7,7,80)	-	-	0	Conv 4_1_2 Conv 4_2_2 Conv 4_3_2 Conv 4_4
Conv 5_1_1	(7,7,15)	(1,1)	-	1215	Merge 1
Conv 5_1_2	(7,7,15)	(3,3)	-	2040	Conv 5_1_1
Conv 5_2_1	(7,7,15)	(1,1)	-	1215	Merge 1
Conv 5_2_2	(7,7,15)	(5,5)	-	5640	Conv 5_2_1
Pooling 5_3_1	(7,7,80)	-	3,1	0	Merge 1
Conv 5_3_2	(7,7,15)	(1,1)	-	1215	Pooling 5_3_1
Conv 5_4	(7,7,15)	(1,1)	-	1215	Merge 1
Merge 2	(7,7,60)	-	-	0	Conv 5_1_2 Conv 5_2_2 Conv 5_3_2 Conv 5_4
Pooling 6	(1,1,60)	-	7,1	0	Merge 2
Flatten	(60)	-	-	0	Pooling 6
Dropout	(60)	-	-	0	Flatten
Dense	(1)	-	-	61	Dropout
Activation	(1)	-	-	0	Dense
Total params: 37,589					

Table 3.8: GoogleNet-like architecture

3.3.7 SqueezeNet-like

Compared with other architectures, SqueezeNet [78] owns very little parameters while remaining the similar high accuracy. It's amazingly achieve AlexNet-level accuracy with 50 times fewer parameters and <0.5MB model size, identify patterns with very few parameters while preserving accuracy.

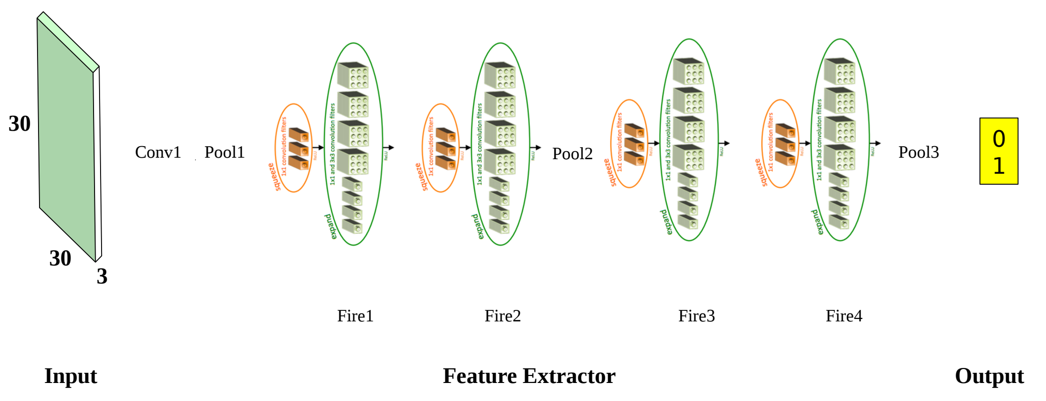


FIGURE 3.21: SqueezeNet-like architecture.

There are some tricks in its structure. First of all, is the structure called fire which looks like fire blazing through match. Rather than 3 × 3 convolutional core used in GoogLeNet, SqueezeNet replace some 3 × 3 filters with 1 × 1 filters in some layers, since 1 × 1 filters has 9 times fewer parameters than a 3 × 3. The fire module is comprised by a squeeze convolution layer(which has only 1 × 1 filters), and the aforementioned layer fed into an expand layer that has a mix of 1 × 1 and 3 × 3 convolutional filters. Then, to decrease parameters, decreasing input channel quantity is a solution. Third, downsample lately in the network so that convolutional layers can have larger activation maps, which lead to higher classification accuracy. Finally, In stead of fully-connected layer, the output is generated by Pooling layer directly, this structure can decrease filter amount in a large-scale. For instance, final convolutional layer obtains the features in size 13 × 13 × 1000, after that, Pooling layer subsampling the feature into size 1 × 1 × 1000, and make a 1000 possibilities prediction.

In this experiment, we designed a SqueezeNet-like architecture (Figure 3.21) also begin with a standalone convolutional layer, then, 4 fire modules followed. Emulating original SqueezeNet structure, we also gradually increase the number of filters per fire module from the beginning to end. Maxpooling (overlapping pooling)with stride is implemented

Layer	Output Shape	Kernel Size	Scale	Param	Connect to
Input	(30,30,3)	-	-	0	-
Conv 1	(30,30,18)	(5,5)	-	1368	Input
Pooling 1	(14,14,18)	-	3,2	0	Conv 1
Conv 1_1	(14,14,16)	(1,1)	-	304	Pooling 1
Conv 1_1_1	(14,14,64)	(1,1)	-	1088	Conv 1_1
Conv 1_1_2	(14,14,64)	(3,3)	-	9280	Conv 1_1
Merge 1	(14,14,128)	-	-	0	Conv 1_1_1 Conv 1_1_2
Conv 2_1	(14,14,16)	(1,1)	-	2064	Merge 1
Conv 2_1_1	(14,14,64)	(1,1)	-	1088	Conv 2_1
Conv 2_1_2	(14,14,64)	(3,3)	-	9280	Conv 2_1
Merge 2	(14,14,128)	-	-	0	Conv 2_1_1 Conv 2_1_2
Pooling 2	(6,6,128)	-	3,2	0	Merge 2
Conv 3_1	(6,6,32)	(1,1)	-	4128	Pooling 2
Conv 3_1_1	(6,6,20)	(1,1)	-	660	Conv 3_1
Conv 3_1_2	(6,6,20)	(3,3)	-	5780	Conv 3_1
Merge 3	(6,6,40)	-	-	0	Conv 3_1_1 Conv 3_1_2
Conv 4_1	(6,6,20)	(1,1)	-	820	Merge 3
Conv 4_1_1	(6,6,20)	(1,1)	-	420	Conv 4_1
Conv 4_1_2	(6,6,20)	(3,3)	-	3620	Conv 4_1
Merge 4	(6,6,40)	-	-	0	Conv 4_1_1 Conv 4_1_2
Dropout	(6,6,40)	-	-	0	Merge 4
Conv 5	(6,6,1)	(1,1)	-	41	Dropout
Pooling 3	(1,1,1)	-	6,6	0	Conv 5
Flatten	(1)	-	-	0	Pooling 3
Dense	(1)	-	-	0	Flatten
Activation	(1)	-	-	0	Dense
Total params: 39,941					

Table 3.9: SqueezeNet-like architecture

after $Conv1$ and $Merge2$, the final average pooling layer make the output categories into two types building and non-building (Table 3.9).

The original networks is developed based on Caffe [87] framework, however, Caffe can not support a convolutional layer that contains multiple filter resolutions(e.g. 1 × 1, 3 × 3), to make it work, we can use the zero padding to 1 × 1 filter. Instead, another Deep Learning framework called Keras [71] can satisfy all our needs.

3.4 Transfer Learning based CNN

Transfer Learning [55] has become a very important method in machine learning. Generally speaking, transfer learning makes the classifier own the ability of identifying new things. As an example shown in Figure 3.22, if we train a cat identification classifier by providing brown cats samples only, when the black cat come, the classifier is hard to identify the cat correctly. Obviously, the bad behavior is caused by lacking corresponding training samples of black cat. However, there must be many common characteristics of cats in different colors, and how to transfer the original model into a new one which could identify both two color cats is important.

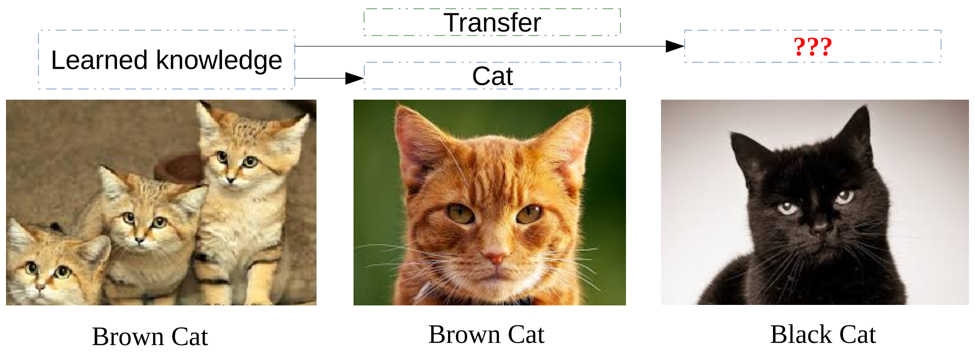


FIGURE 3.22: Transfer Learning.

There are some things need to be pointed out. First of all, collecting well-rounded images is impossible, which means we can not get all the cat images around the world. Second, retraining the original model every time when some new training dataset provided would be very time consuming. Finally, the intelligent creatures such as human beings own the capability to identify new color cat with no doubt, while the model cannot do so. Transfer Learning would be a method of solving these problems.

Here we explore the possibility of implementing transfer learning method in CNN based on simple handwritten digits dataset called MNIST [88]. Our target is create an original model which could only identify 0∼4, then make it own the capability of identifying 5∼9 while do not forget 0∼4.

As shown in Figure 3.23, first of all, we train a model which could identify 0∼4. Analyzing the common characteristics of 0∼4 and 5∼9, there mast be many common features in these two datasets. Therefore in the second step, we frozen the feature extractor part of the original model, just make multilayer perceptron part trainable and mutable. Then we input new dataset 5∼9 into the original model and retrain a new one.

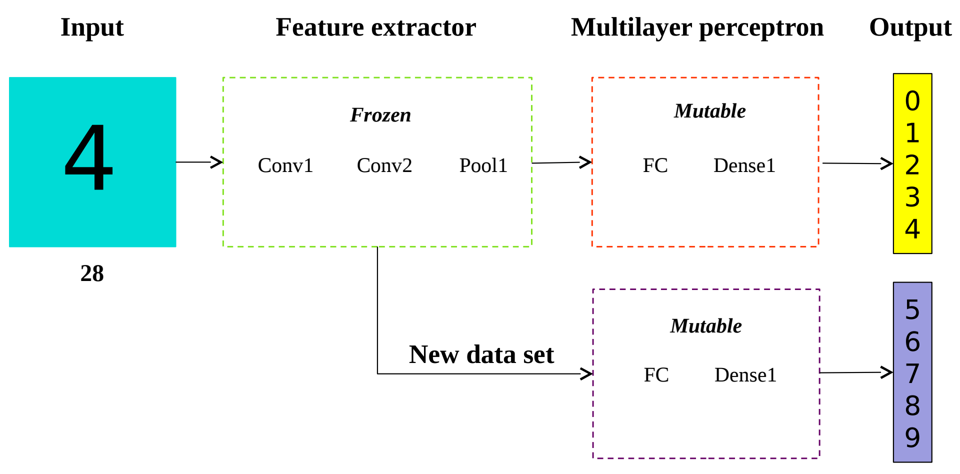


FIGURE 3.23: Transfer Learning trial.

By the preceding processing, the new knowledge and model are learned based on the old knowledge and model respectively, while using the same feature extractor and extracting similar features. In many conditions, new data (new knowledge) is very limited, and small new data set can also generate a new classifier. Whats more, since the new classifier get influenced by both old and new datasets, maybe own the ability of identifying all the digits.

Here (Table 3.10) is our hypothesis in case the model owns transfer learning capability. The original model can only identify 0∼4, while the transfered model can handle all the digits.

	Original Data Set 0∼4	New Data Set 5∼9	All Data Set 0∼9
Original Model	Good	Bad	Bad
Transfered Model	Relatively Good	Good	Good

TABLE 3.10: Transfer Learning hypothesis.

As a trial, we provide new training dataset 5∼9 which only contains 1/60 of original amount to train a new model based on the old one. Result shows that training time only costs 1/4 comparing with original one and with accuracy up to 88%, which means, the new model transfered from 0∼4 classifier can identify handwritten digits from 5 to 9 in high accuracy. In order to validate our hypothesis, here we utilize our original and transfered model to test all the dataset respectively. The result in Table 3.11 match the hypothesis (Table 3.10) quite well.

Testing result must be highly based on the new training dataset 5∼9 amount. here we choose new dataset with amount ratio: 1/1, 3/1, 1/6, 1/10, 1/20, 1/40, 1/60, 1/80,

	Original Data Set 0∼4	New Data Set 5∼9	All Data Set 0∼9
Original Model	99.71%	34.71%	67.89%
Transfered Model	95.16%	89.97%	92.62%

TABLE 3.11: Transfer Learning result in MNIST.

1/100, 1/200 compared the original training dataset 0∼4. The result as shown in Figure 3.24, and we can infer that:

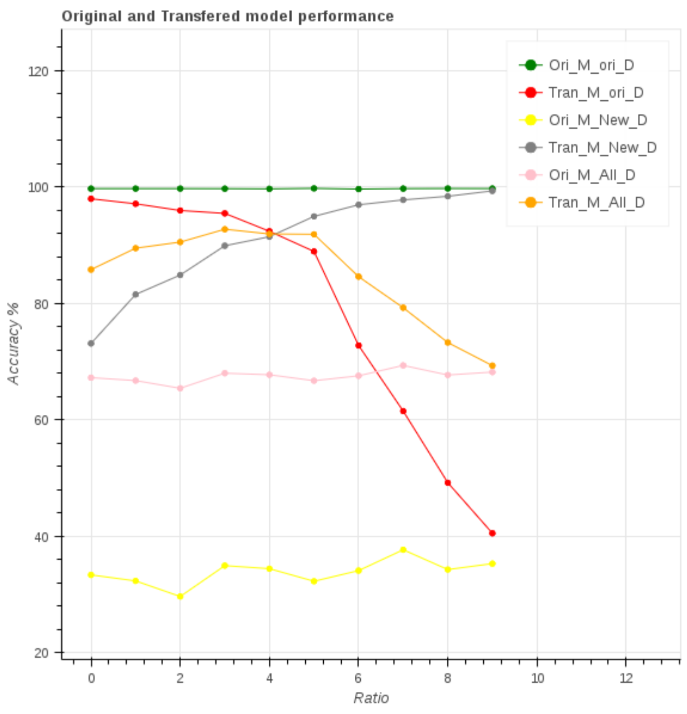


FIGURE 3.24: Transfer Learning "memory curve".

- Green: original model test original data. Good result, and very stable.
- Yellow: original model test new data. Bad result, because it do not have such knowledge.
- Pink: original model test all data. Normal performance, it can not deal with new data set.
- Red: transferred model test original data. When new training data amount increasing, transferred model tend to forget old knowledge.
- Gray: transferred model test new data. Bigger new data amount can get more knowledge

- Orange: transferred model test all data. Performance like a convex, when new data not enough, can not understand new knowledge well, when new data too much, tend to forget old knowledge

Table 3.12 is the detailed result:

	1/200	1/100	1/80	1/60	1/40	1/20	1/10	1/6	1/3	1/1
OM_OD	99.72	99.72	99.72	99.71	99.68	99.76	99.64	99.72	99.74	99.74
TM_OD	97.97	97.11	95.96	95.46	92.39	88.92	72.76	61.49	49.18	40.50
OM_ND	33.34	32.31	29.63	34.93	34.41	32.25	34.08	37.66	34.26	35.28
TM_ND	73.11	81.53	84.87	89.90	91.45	94.95	96.94	97.78	98.40	99.33
OM_AD	67.23	66.73	65.41	68.00	67.73	66.71	67.55	69.34	67.69	68.19
TM_AD	85.80	89.48	90.53	92.74	91.93	91.87	84.59	79.25	73.27	69.30

TABLE 3.12: Transfer Learning "memory curve" result.
O=old, M=model, T=transferred, N=new, A=all, D=Data

According to the previous analysis, the CNN based transfer learning would be useful to solve the problem in case training a new model without enough new training dataset. With relatively short training time, the transfered model owns the capability to identify in both old and new data set with high accuracy.

When identification buildings in rural environment, usually the model is trained by a specific dataset, and building's characteristics similar with training dataset can be correctly identified. Also, new training dataset is hard to prepare which lead lacking training data. What's more, training a brand new model would be really time consuming. Transfer learning would be able to provide (Figure 3.25) us a possibility to solve these problems.

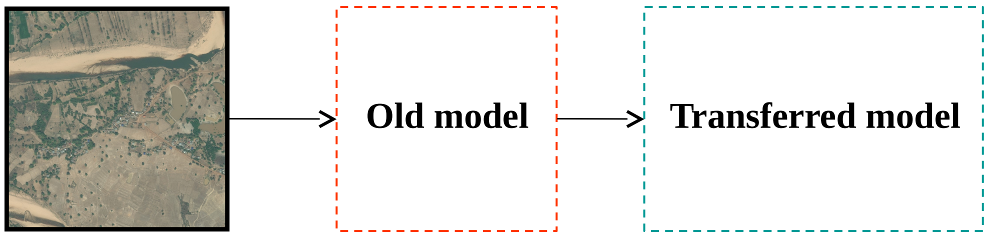


FIGURE 3.25: Transfer Learning in building identification.
Put limited new training dataset into old model, retrain it and generate transfered one

3.5 Other Processing

Except for creating machine learning model, we usually face many other different problems in machine learning identification tasks, As shown in Figure 3.27. In general, when lacking training data in both diversity and amount, we can consider using Data Augmentation method, which can also prevent over-fitting. As mentioned in the previous section, transfer learning can help us obtain new model with good capability efficiently. When the testing and training dataset quite differ in color spectrum, we can consider color balance method to make some modification. After generating result, the post processing such as close can be conducted to get rid of salt-and-pepper noise.

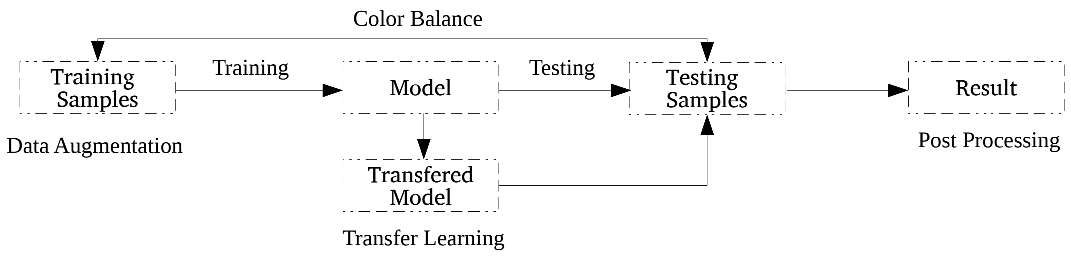


FIGURE 3.26: Other processing.
Data Augmentation, Color Balance, Post Processing

3.5.1 Data Augmentation

Artificially enlarge the dataset using label-preserving transformations is the most common method to reduce over-fitting. Here we employ two forms of data augmentation [89]. First of all, considering the diversity of building, we enrich the category by rotating or stretching the training samples, in that case, many new buildings in different structure will be generated with accurate ground truth. Then, by changing the RGB bands sequence, building in same exterior appearance with different color will also be obtained. The preceding Data Augmentation is randomly implemented when collecting training samples, and 10% samples are generated by the proposed method.

3.5.2 Color Balance

As an image filter, Wallis filter [90] scans the image and makes every pixel in the output image have a specified mean and standard deviation. In this experiment we choose it to be the color balance tool, and the target image will be converted into a new one based on the provided template. As shown in the following equation:

$$g(x,y) = [f(x,y) - m_c]\frac{CV_s}{CV_s + (1-C)V_s} + bm_s + (1-b)m_c \tag{3.23}$$

Where $g(x,y)$ refers to output result, $f(x,y)$ indicates input, V_c and V_s refer to input and template variance respectively, while m_s and m_c means template and output mean values. When utilizing Wallis filter in our experiment, template HRRS image need to be prepared, in order to calculate its variance and mean value.

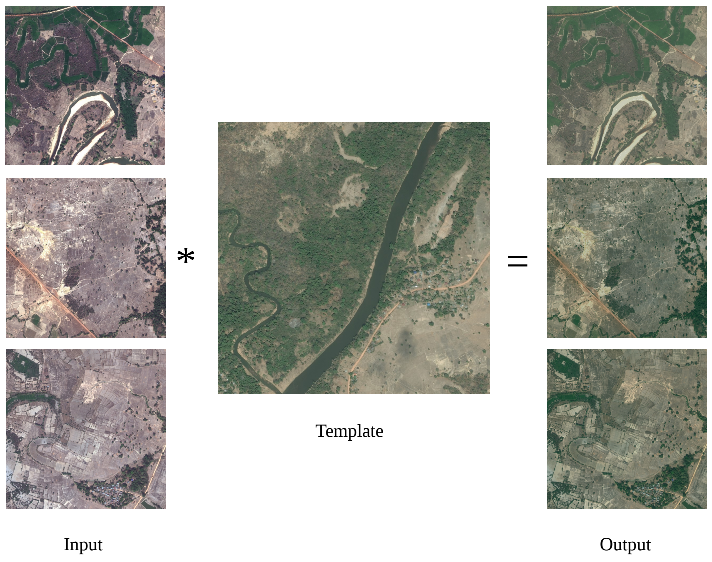


FIGURE 3.27: Color Balance.

The result as shown in Figure 3.27, the left input images are converted in a new color space which could be balanced with the template image. We will conduct this method in case we need.

3.6 Experimental Platform

We conduct our experiment on two sets of platforms, As shown in Table 3.13. The first experimental platform is developed among Window 10 operation system with programming language Matlab [91]. The Deep Learning libraries such as Caffe [87] and DeepLearnToolbox [70] can support our experiment on the this platform. The computation core we use is CPU, which take very long time to training the model and test image.

Version	Platform 1	Platform 2
System	Windows 10	Ubuntu 14.04
Language	Matlab	Python 3.5
Hardware	CPU	GPU + CPU
Library	Deep learning box	Theano + Keras
Accuracy	Similar	
Velocity / Epoch	123.9s	1.4s

TABLE 3.13: Experimental platform.

Then we change our platform into Linux Ubuntu 14.04 and Python [92] 3.5, the platform owns the advantages such as very convenient and supported by many deep learning libraries. Here we list the detailed information of some famous Deep Learning libraries and frameworks in Table depending our investigation and understanding.

Framework	Language	CUDA	Platform	Install	Difficulty	Effi	Popu
Caffe	C++ Python Mat	Yes	Win Ubun OS X	Hard	Middle	High	High
Theano	Python	Yes	Win Ubun OS X	Middle	Hard	High	Middle
TensorFlow	C++ Python	Yes	Win Ubun OS X	Middle	Hard	High	High
Chainer	Python	Yes	Win Ubun OS X	Easy	Easy	High	Low
Torch	C eLua	Ys	Win Ubun OS X	?	?	?	?

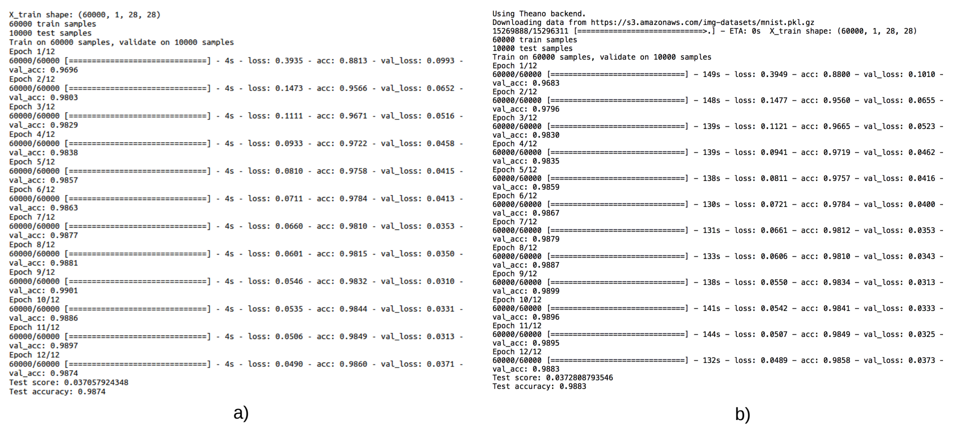
TABLE 3.14: Deep Learning libraries

In this experiment, we use the framework called Keras [71], which is a minimalist and highly modular neural networks library, written in Python and capable of running on top of either TensorFlow [93] or Theano [94]. Keras (Table 3.15) is very easy to construct CNN architecture also can efficiently use Compute Unified Device Architecture (CUDA) to accelerate computation.

Framework	Backend	CUDA	Language	Platform	Install	Difficulty
Keras	Theano TensorFlow	Yes	Python	Win, OS X Ubuntu	Middle	Easy

TABLE 3.15: Keras

The NVIDIA CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. By using cuDNN, GPU can be 30 times faster than CPU, the comparison as shown in Figure 3.28.



```
X_train shape: (60000, 1, 28, 28)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 4s - loss: 0.3935 - acc: 0.8813 - val_loss: 0.0993 -
val_acc: 0.9696
Epoch 2/12
60000/60000 [==============================] - 4s - loss: 0.1473 - acc: 0.9566 - val_loss: 0.0652 -
val_acc: 0.9803
Epoch 3/12
60000/60000 [==============================] - 4s - loss: 0.1111 - acc: 0.9671 - val_loss: 0.0516 -
val_acc: 0.9829
Epoch 4/12
60000/60000 [==============================] - 4s - loss: 0.0933 - acc: 0.9722 - val_loss: 0.0458 -
val_acc: 0.9838
Epoch 5/12
60000/60000 [==============================] - 4s - loss: 0.0810 - acc: 0.9758 - val_loss: 0.0415 -
val_acc: 0.9857
Epoch 6/12
60000/60000 [==============================] - 4s - loss: 0.0711 - acc: 0.9784 - val_loss: 0.0413 -
val_acc: 0.9863
Epoch 7/12
60000/60000 [==============================] - 4s - loss: 0.0660 - acc: 0.9810 - val_loss: 0.0353 -
val_acc: 0.9877
Epoch 8/12
60000/60000 [==============================] - 4s - loss: 0.0601 - acc: 0.9815 - val_loss: 0.0350 -
val_acc: 0.9881
Epoch 9/12
60000/60000 [==============================] - 4s - loss: 0.0546 - acc: 0.9832 - val_loss: 0.0310 -
val_acc: 0.9901
Epoch 10/12
60000/60000 [==============================] - 4s - loss: 0.0535 - acc: 0.9844 - val_loss: 0.0331 -
val_acc: 0.9886
Epoch 11/12
60000/60000 [==============================] - 4s - loss: 0.0506 - acc: 0.9849 - val_loss: 0.0313 -
val_acc: 0.9897
Epoch 12/12
60000/60000 [==============================] - 4s - loss: 0.0490 - acc: 0.9860 - val_loss: 0.0371 -
val_acc: 0.9874
Test score: 0.037057924348
Test accuracy: 0.9874
```

a)

```
Using Theano backend.
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.pkl.gz
15269888/15296311 [============================>.] - ETA: 0s  X_train shape: (60000, 1, 28, 28)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 149s - loss: 0.3949 - acc: 0.8800 - val_loss: 0.1010 -
val_acc: 0.9683
Epoch 2/12
60000/60000 [==============================] - 148s - loss: 0.1477 - acc: 0.9560 - val_loss: 0.0655 -
val_acc: 0.9796
Epoch 3/12
60000/60000 [==============================] - 139s - loss: 0.1121 - acc: 0.9665 - val_loss: 0.0523 -
val_acc: 0.9830
Epoch 4/12
60000/60000 [==============================] - 139s - loss: 0.0941 - acc: 0.9719 - val_loss: 0.0462 -
val_acc: 0.9835
Epoch 5/12
60000/60000 [==============================] - 138s - loss: 0.0811 - acc: 0.9757 - val_loss: 0.0416 -
val_acc: 0.9859
Epoch 6/12
60000/60000 [==============================] - 130s - loss: 0.0721 - acc: 0.9784 - val_loss: 0.0400 -
val_acc: 0.9867
Epoch 7/12
60000/60000 [==============================] - 131s - loss: 0.0661 - acc: 0.9812 - val_loss: 0.0353 -
val_acc: 0.9879
Epoch 8/12
60000/60000 [==============================] - 133s - loss: 0.0606 - acc: 0.9810 - val_loss: 0.0343 -
val_acc: 0.9887
Epoch 9/12
60000/60000 [==============================] - 138s - loss: 0.0550 - acc: 0.9834 - val_loss: 0.0313 -
val_acc: 0.9899
Epoch 10/12
60000/60000 [==============================] - 141s - loss: 0.0542 - acc: 0.9841 - val_loss: 0.0333 -
val_acc: 0.9896
Epoch 11/12
60000/60000 [==============================] - 144s - loss: 0.0507 - acc: 0.9849 - val_loss: 0.0325 -
val_acc: 0.9895
Epoch 12/12
60000/60000 [==============================] - 132s - loss: 0.0489 - acc: 0.9858 - val_loss: 0.0373 -
val_acc: 0.9883
Test score: 0.0372808793546
Test accuracy: 0.9883
```

b)

Figure 3.28: CPU and GPU comparison.

Chapter 4

Result and Discussion

During the experiment, we first trained the classification model using the sample data and then applied the model to the test dataset. The classification performance was evaluated by three widely used parameters.

- Confusion Matrix (see Table 4.1): This parameter comprehensively describes the performance of a binary classification result.

True Negatives (TN)	False Positives (FP)
False Negatives (FN)	True Positives (TP)

Table 4.1: Confusion Matrix.

  where True Positives: actual buildings that were correctly classified as buildings, False Positives: non-buildings were incorrectly labeled as buildings, False Negatives: buildings that were incorrectly marked as non-buildings, True Negatives: all the things correctly classified as non-buildings.

- Overall Accuracy: Measures the overall performance of the classifier.

$$OverallAccuracy = \frac{TP + TN}{n} \tag{4.1}$$

  where n is the total number of pixels.

- Kappa: This parameter comprehensively describes the classification accuracy by measuring the inter-rater agreement among qualitative items [95]. Kappa is defined as follows:

$$Kappa = \frac{2 \times (TP \times TN - FP \times FN)}{TP(2TN + FP + FN) + TN(FP + FN) + FP^2 + FN^2} \tag{4.2}$$

The performances of AdaBoost and basic CNN model are evaluated in Sections 4.1 and 4.2, respectively. All input training data are contained in the 900 × 600 pixel RGB image shown in Figure 2.1d. The same image is used for the test data. The test results of the two algorithms are compared and discussed in Section 4.3. In Section 4.4, we test the entire image (Figure 2.1f, with a size of 3600 × 4500 pixels) by basic CNN. To enhance the performance, we also alter the input training data. The accuracy of the experiment is evaluated by comparing the result with the ground truth, which contains the tree labels (building areas, non-building areas, and unknown areas such as cloud cover). In section 4.4, we enhance basic CNN into stronger models and compare their feasibility in building identification. To solve the problem of color unbalance, over-fitting, etc., we implement color balance and transfer learning in the rest sections.

4.1 Result of Adaboost

In the training section, the positive samples contain information of the recognized target, which herein refers to building information; conversely, negative samples are without information of a recognized target such as trees, roads, and unknown areas.

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x) - Terra\right) \tag{4.3}$$

The confidence obtained by AdaBoost is thresholded by a parameter called Terra. However, when Terra was set to 0 (as in traditional AdaBoost methods), the performance was greatly degraded by the large number of false positives in the classification results. After observing the results for different values of Terra, we selected the optimal Terra based on the Kappa criterion. The following result indicates the progress from using color features to add Haar-like features.

4.1.1 Color Feature

As previously mentioned, the classifier was optimized by trialing four kinds of input samples from Figure 2.1d. All the testing samples were also prepared from Figure 2.1d. There must be a one-to-one size correspondence between the testing and training samples, as shown in Table 4.2. Since the edges are ignored, the number of testing samples differs among the tests.

After the training procedure, four kinds of stronger classifier emerged: AdaBoost_A–D.

		Training Samples		
Classifier	Sample Size	Positive Samples	Negative Samples	Weak Classifiers
AdaBoost_A	1×1	48,591	124,803	500
AdaBoost_B	3×3	40,071	102,163	500
AdaBoost_C	5×5	32,079	81,787	500
AdaBoost_D	7×7	24,975	63,675	500

Table 4.2: Training data of color features.

To test the accuracy of classifiers generated in the training part, we utilize the classifiers obtained to detect the testing samples, respectively, and the results are as follows (Table 4.3).

Classifier	Testing Samples	Terra	Kappa	Overall Accuracy
AdaBoost_A	540,000	1.7	0.31	95.64
AdaBoost_B	537,004	3.5	0.31	96.04
AdaBoost_C	534,016	5.0	0.30	96.16
AdaBoost_D	534,016	8.5	0.29	96.65

Table 4.3: Comparison of accuracy based on color features.

As we can infer from the result table, classifier B has the optimum solution, with a Kappa of 0.31 and overall accuracy of 96.04%.

4.1.2 Haar-Like Features + Color Features

To enhance the accuracy of the classifier, we incorporate Haar-like features and color features to mine deeper texture information of the target. Based on the feature value theory, by calculating the entire feature value via an integral image algorithm, the AdaBoost method can be achieved.

The training samples are captured from Figure 1e, with 111 pieces of positive sample and 283 negative samples, all sized at 25 × 25 pixels in gray-scale. The positive and negative samples are the gray-scale images of buildings and non-buildings, respectively. The input training data is a matrix of Haar-like feature values. The size of the Haar-like features can be stretched or reduced, and they can move within the entire image. The dimension of a two-rectangle feature would reach 3,328; therefore, considering that the dimension of the input training data is extremely large, only four kinds of common Haar-like feature have been used. Based on the principle of Figure 3.3, the total dimension of the input features is $3328 \times 2 + 2600 + 2276 = 11532$. The testing data is also shown in Figure 2.1e. After combining with the color features, the result of the identification is enhanced to Kappa of 0.31 and the overall accuracy is enhanced to 96.22%. Compared with the result of using color features only, this result is somewhat improved.

4.2 Result of Basic CNN

In the CNN algorithm training process, placing the training samples and their relevant labels together is necessary. To test the influence of the input parameters on the final classifier, we utilize a control variate method as follows (Table 4.4).

Classifier	Sample Size	Positive Samples	Negative Samples	Iteration
CNN_A	18×18	13,300	50,000	50
CNN_B	18×18	13,300	50,000	300
CNN_C	18×18	26,150	50,000	300

Table 4.4: Training data of basic CNN.

After the training part of CNN, three different classifiers are generated from CNN_A to CNN_C. After utilizing the classifiers to test the given figure, the result and the corresponding figure will be produced (Table 4.5).

Classifier	Kappa	Overall Accuracy
CNN_A	0.41	93.96%
CNN_B	0.43	94.13%
CNN_C	0.56	96.30%

Table 4.5: Comparison of accuracy based on basic CNN.

Compared with the respective results, the obtained classifier CNN_C performs well in detecting the buildings from the given figure. Kappa is enhanced to 0.56 and the overall accuracy reaches 96.30%. In such cases, the confusion matrix and the outcome figures are as follows (Table 4.6).

True Negatives: 549,034	False Positives: 17,366
False Negatives: 799	True Positives: 12,540

Table 4.6: Confusion matrix based on basic CNN in pixel.

As we mentioned in the definition of the confusion matrix, in the method based on CNN, actual buildings that were correctly classified as buildings is 12,540, non-buildings incorrectly labeled as buildings is 17,366, buildings that were incorrectly marked as non-buildings is 799, and number of correctly classified non-buildings is 459,034. Although the number of non-buildings that were incorrectly labeled as buildings is a little bit high, impacting the performance of the result, other parts of the matrix perform well.

In the outcome figure of CNN_C, gray refers to the unknown part, green means the actual buildings that were correctly classified as buildings, blue indicates the non-buildings that were incorrectly labeled as buildings, red shows the buildings that were incorrectly marked as non-buildings, and black denotes the correctly classified non-buildings. As shown in the result, the CNN_C classifier has high performance in detecting buildings.

In Figure 4.1b, it can be seen that the classification error decreases stably as the iteration increases.

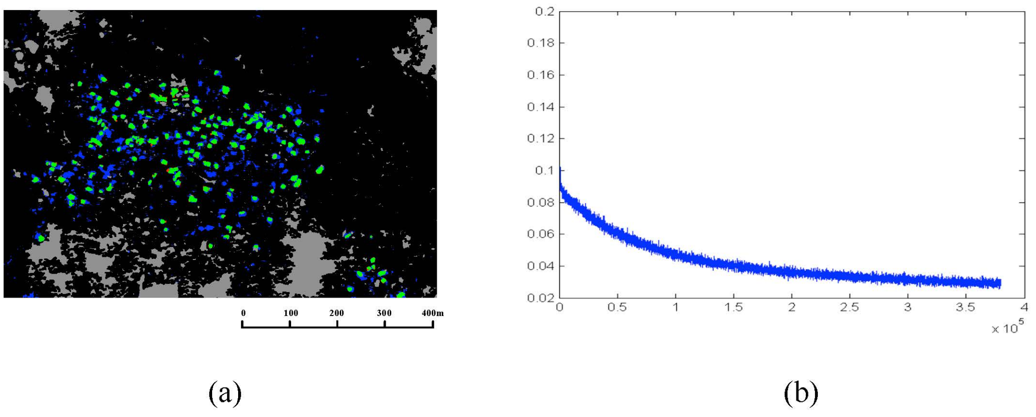


FIGURE 4.1: Basic CNN: CNN_C result.

4.3 Compare Adaboost and Basic CNN

We utilized machine learning methods including AdaBoost and CNN to identify buildings from remote sensing images and generated the relative testing results, as mentioned in Sections 4.1 and 4.2, respectively. Inferring from the comparisons in accuracy, the best result of Kappa and overall accuracy are 0.31 and 96.20% in the AdaBoost algorithm, respectively, whereas in CNN, the result is enhanced to 0.56 and 96.30%, respectively. According to the comparison, in the corresponding testing area, the Kappa of our CNN method is approximately 25% higher than that of AdaBoost. For overall accuracy, CNN also outperforms AdaBoost.

Note that the traditional visual interpretation of remote sensing images is a complex and time-consuming process. Although it has very high accuracy, it is not suited to large-scale automation projects. The effect of AdaBoost methods is highly dependent on the training feature. The color and Haar-like features chosen in our experiment cannot express all the helpful and useful features of the buildings, which impacts the accuracy of the result. In contrast, CNN can mine and extract deeper information on the input features of building, which can be helpful in identification.

4.4 Basic CNN Implementation

In this section, we demonstrate how the CNN method works via the entire image in Figure 1f, which is 30 times bigger than that in Figure 1d. We compare two kinds of input training data and separately obtain the identification results as follows (Table 4.7).

Training Type	Input Training Area	Positive Samples	Negative Samples
Train_A	Figure 4.1a,d	26,150	50,000
Train_B	Figure 4.1a,b,d,e	26,150	150,000

TABLE 4.7: Different training data using basic CNN.

The training data in type Train_B contains more diverse negative sample information than that in Train_A, with 150,000 samples including information of mountains and other types of land. The identification results are as follows (Table 4.8).

Training Type	TP	FP	FN	TN	Kappa	Overall Accuracy
Train_A	82,578	265,923	12,306	15,009,525	0.366	98.19%
Train_B	83,351	175,495	11,533	15,099,953	0.466	98.78%

TABLE 4.8: Comparison of accuracy based on basic CNN.

As observed from the testing result, the Kappa and overall accuracy increase when the negative training samples are expanded. In Figure 4.1, most of the village buildings can be identified, but the inaccuracy points are mainly distributed in the village boundary and some building-like areas. From Figure 4.2b, areas such as mountains can be identified as non-buildings areas, whereas they could not be detected in Figure 4.2a.

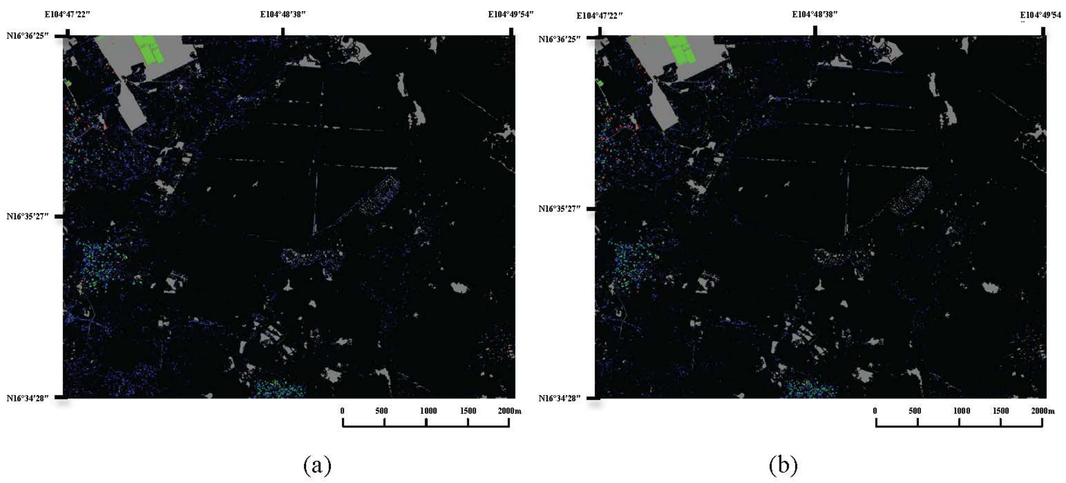


FIGURE 4.2: Classification results of basic CNN: (a) Train_A; and (b) Train_B.

To detect the details of the performance of Train_B, we split the image into 30 small images of size 900 × 600 pixels as follows (Figure 4.3).

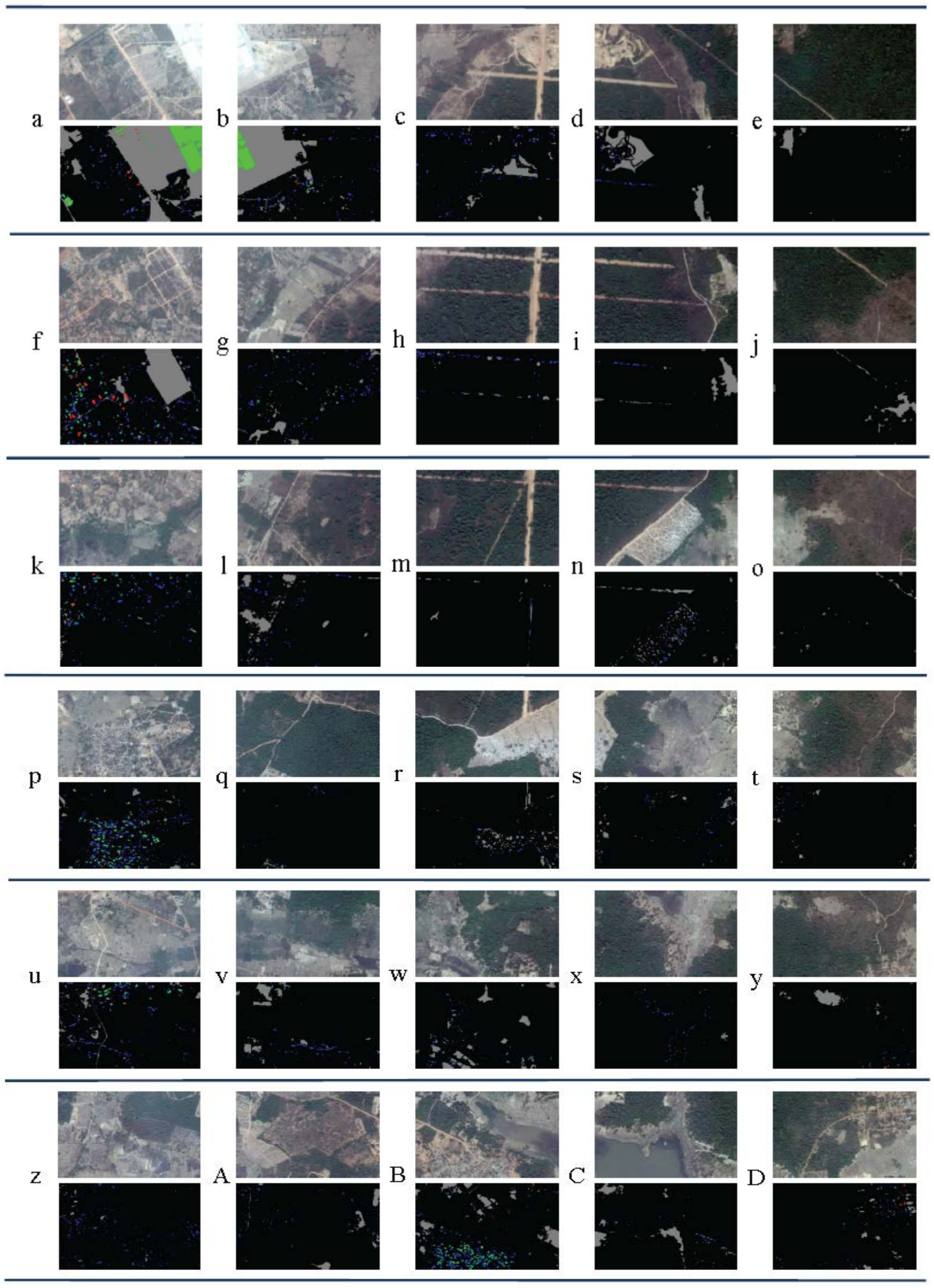


FIGURE 4.3: Classification results (a-D) of test areas using CNN_C.

The performance of each part is different and the accuracy comparison is as follows.

For different study areas, the overall accuracy of the CNN method can be up to 99.00%. We only show part of the results in Table 4.9, especially those containing information of buildings (positive pixels) where Kappa can be up to 0.89, which indicates high performance. We can also infer that the overall accuracy of the proposed method is considerably stable in different areas. As Kappa is relatively low in some areas, identification accuracy can be improved by expanding the training data, as shown in previous processes.

Study Area	Positive Pixels	Negative Pixels	Kappa	Overall Accuracy
a	41,262	307,805	0.89	97.54%
b	14,392	398,102	0.76	98.00%
c	9075	495,639	0.47	96.47%

Table 4.9: Accuracy assessment of all the testing data in Figure 4.2 based on basic CNN.

In general, the experimental results indicate that our proposed method based on machine learning, especially on CNN, has relative high performance in remote sensing identification of buildings.

4.5 Different Structures Result

According to the result, identification capability of basic structure is still limited, which can not achieve very high Kappa in many testing areas. As shown in the previous Chapter, after analyzing the learning curve, we self-design 4 kinds of structure based on the state-of-the art networks. In this section, we compare the capability of corresponding structures by learning curve and testing result.

	Parameter		Training			
Structure	Former	New	Acc	Kappa	Iter(s)	Total(s)
AlexNet-like	60.97M	51,249	99.77	0.99	5.42	1626.95
VGGNet-like	143.67M	70,453	99.78	0.99	13.81	4142.61
GoogLeNet-like	7.00M	37,589	99.71	0.99	6.62	1986.80
SqueezeNet-like	1.25M	39,941	99.73	0.99	7.23	2171.02

Table 4.10: Training result by different structures.

Based on the principle and our former exploration about how to build a high accuracy architecture, here we set the experiment parameters as follows: iteration = 300, window size = 30 $\times$ 30, learning rate = 0.03, activation Relu and Sigmoid. In terms of dataset, 50,655 and 12,664 images are chosen as training and cross validation samples respectively, in which contains 13319 positive samples and 50,000 negative samples. As an example, the testing image here we also choose Laos image.

The filter amount and depth are different between each architecture. Detailed settings and training result in Table 4.10.

The results represented in overall accuracy, Kappa and confusion matrix are shown in Table 4.11.

	Testing			Confusion Matrix			
Structure	Acc	Kappa	Total(s)	TN	FP	FN	TP
Basic	96.30	0.56	180.70	549,034	17,366	799	12,540
AlexNet-like	98.92	0.81	9.88	520,878	5803	39	13,280
VGGNet-like	99.14	0.85	24.60	52,2081	4600	26	13,293
GoogLeNet-like	98.91	0.81	12.19	520,837	5844	63	13,256
SqueezeNet-like	98.89	0.81	17.93	520,713	5968	45	13,274

Table 4.11: Testing result by different structures.

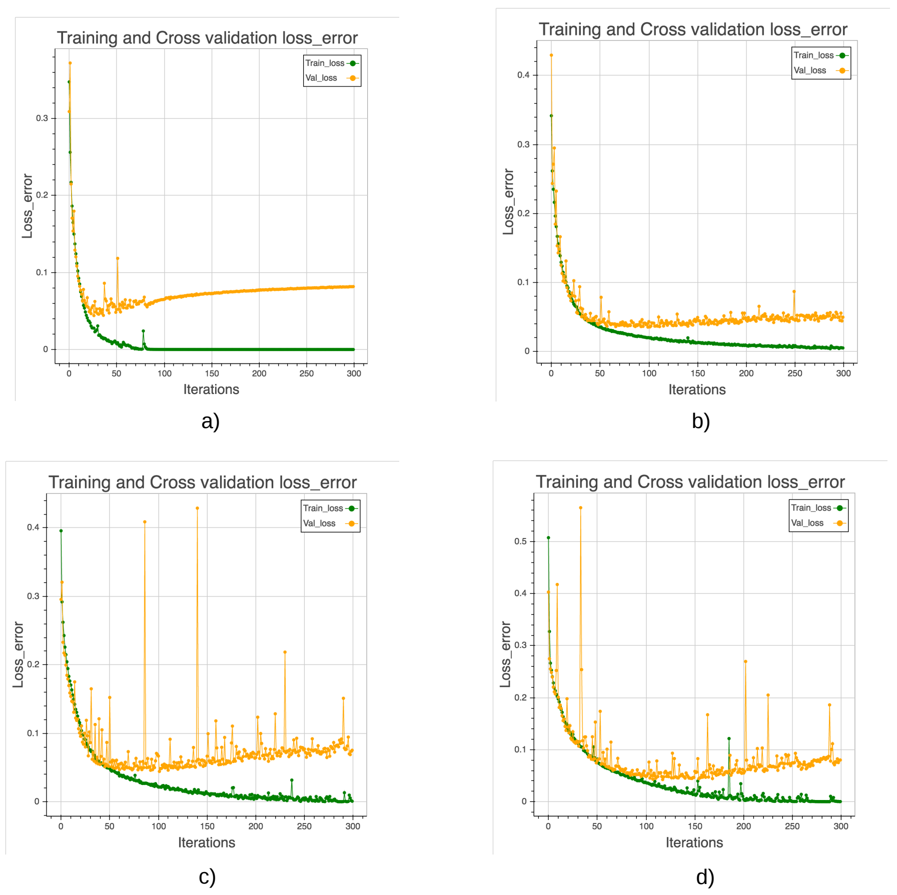


Figure 4.4: Learning curve of different structures.
(a)AlexNet-like (b)VGGNet-like (c)GoogLeNet-like (d)SqueezeNet-like

From the testing result, self-design structures outperform the basic structure and increase 3% of accuracy and 30% Kappa, also the confusion matrix shows that TP and TN increase a lot while FP and FN decreased. The testing result proves the feasibility of our modification method, and VGGNet-like architecture achieves the best.

Merely analyze model's ability based on result is not enough, here we illustrate learning curves (Figure 4.4) of these models in training part.

The curve shows that, although all of the architectures modified based on the current state-of-the-art architectures own the strong ability in both training and testing, the deliberately designed GoogleNet-like and SquezzNet-like are still relatively not stable and also suffer from over-fitting. in contrast, AlexNet-like and VGGNet-like are stable without over-fitting. The result comparison in visualization as shown in Figure 4.5.

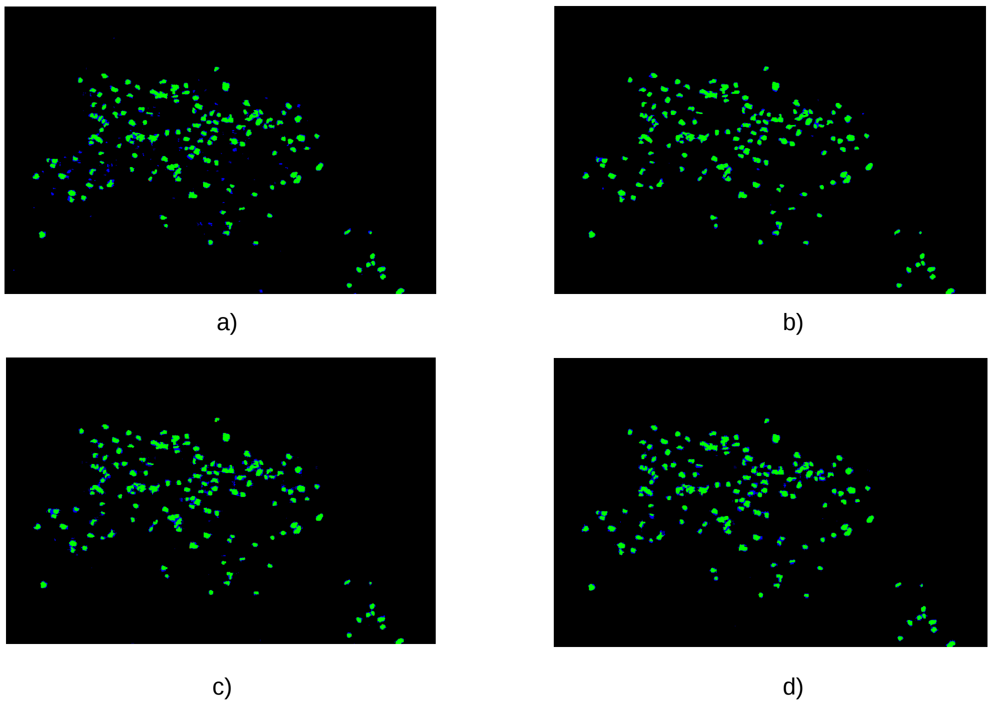


Figure 4.5: Different structure result comparison.
(a)AlexNet-like (b)VGGNet-like (c)GoogLeNet-like (d)SqueezeNet-like

Considering the accuracy, stability and efficiency when conducting our experiment, we choose VGGNet-like and utilize it to test more dataset. Here to compare the feasibility of VGGNet-like and basic CNN structure, we test image in Kenya as example (Figure 4.6). And concrete accuracy, Kappa and confusion matrix as Table 4.12, also shows the capability of our created model.

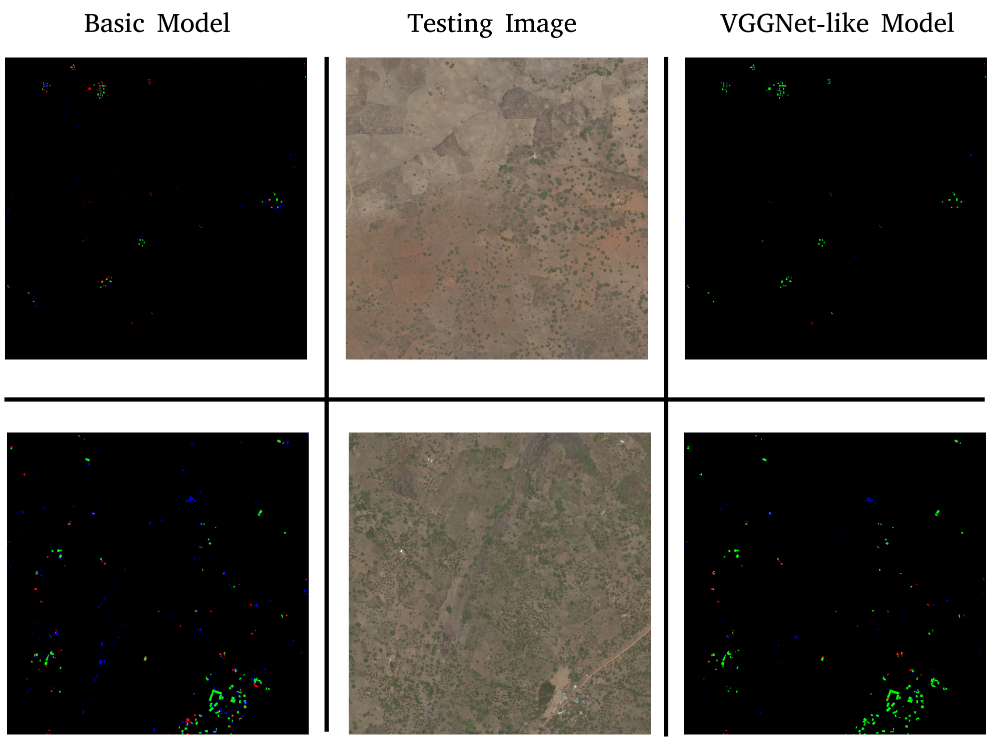


FIGURE 4.6: Kenya tesing result comparison.
Left: result of basic model. Middle: Testing image. Right: result of VGGNet-like

		Testing		Confusion Matrix			
Image	Structure	Acc	Kappa	TN	FP	FN	TP
1	Basic	99.83	0.44	4,184,127	4,221	3059	2897
1	VGGNet-like	99.96	0.87	4,187,087	1261	451	5505
2	Basic	99.33	0.57	4,147,663	20,176	7843	18,622
2	VGGNet-like	99.77	0.82	4,223,170	6173	3552	22,913

TABLE 4.12: Kenya testing result by basic and VGGNet-like structure.

In oder to test the robustness of our model, we conduct the experiment using training and testing sample in different country. What's more, spectrum and building's appearance are also different in two datasets (Table 4.13).

Area	Savanakhet(Laos)	Kwale(Kenya)
Date	2016.02	2016.11
Resolution(m)	1.2	0.6 and 1.2
Size	1024 × 1024	10240 × 6656 and 5376 × 3328
Amount	8	2(same area with different resolution)
Quality	Good	Good

TABLE 4.13: Two different datasets.

The training samples obtained from Laos with resolution 1.2m, the sample diversity also taken into consideration. While Kenya testing image contains two pieces in the same area with different resolution 0.6m and 1.2m respectively, as shown in Figure 4.7.

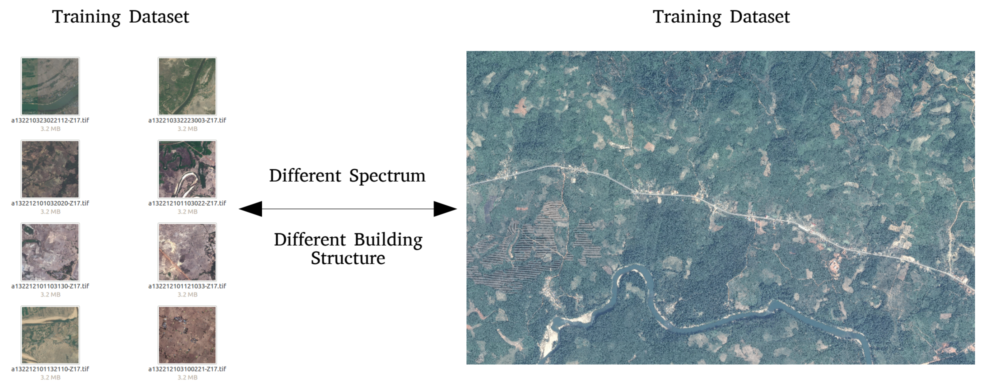


FIGURE 4.7: Two different datasets implemented in training and testing respectively.

After training, we utilize obtained model to test image. The learning curve indicates that although get unstable in cross validation, the model can still achieve high accuracy result in training part. However, relatively bad results are generated in testing (Figure 4.8), where black and white points means building and other land features respectively.

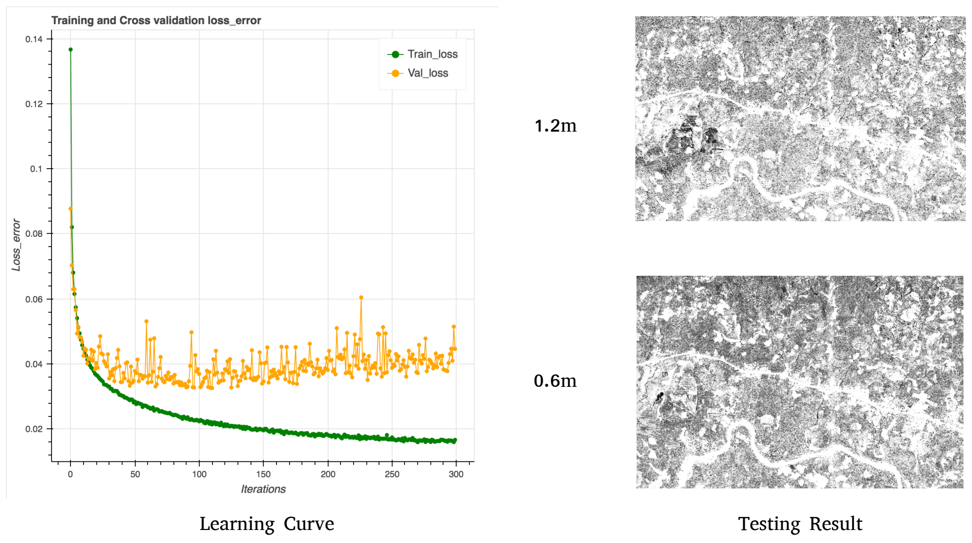


FIGURE 4.8: Kenya testing result.

The big difference between training and testing dataset causes such bad result, and providing training data which can match testing dataset can be obviously make sense. However, suitable training data cannot always be available, and making new samples would be very time consuming. Here we propose color balance and transfer learning respectively to solve such problem.

4.6 Color Balance

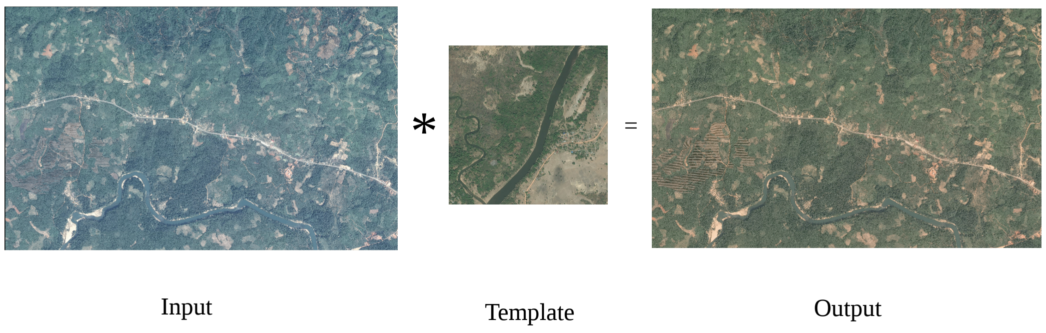


FIGURE 4.9: Balance image color.

We balance the training and testing dataset spectrum based on the same template by Wallis filter [90] (Figure 4.9) introduced in the previous chapter. After converting dataset into a new spectrum, we train the model and find that balanced training dataset can make model converge better (Table 4.14).

	Balanced	Original
Positive samples	57,860	57,860
Negative samples	320,105	320,105
Train Acc	99.51	99.41
Train Loss	0.01	0.17
Valid Acc	0.99	0.99
Valid Loss	0.04	0.05
Kappa	0.98	0.97

TABLE 4.14: Comparison between balanced model and original model.

Then we use original and balanced models to test original and balanced images respectively, the result as shown in Figure 4.10. While color unbalance is a significant problem, utilize proposed method can increase accuracy in a large degree, especially false positive pixels can be decreased.

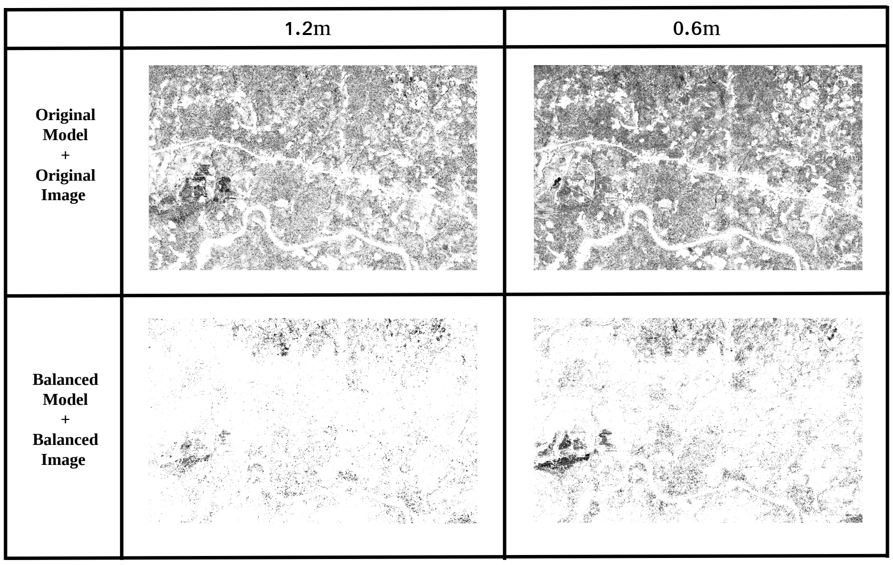

1.2m
0.6m
Original Model + Original Image
Balanced Model + Balanced Image

FIGURE 4.10: Result of transformed image.

4.7 Transfer Learning

Except for color unbalance, other problems such as texture difference still cause bad results and need to be solved. Therefore, in order to make model own the capability to identify new buildings, the new training dataset must be provided although might in a small quantity.

When a little bit new dataset come, how to learn new knowledge without training a brand new model is a challenge. Here we propose CNN based transfer learning method introduced in previous Chapter.

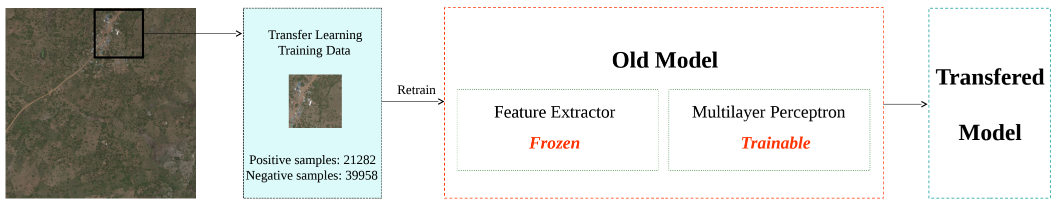
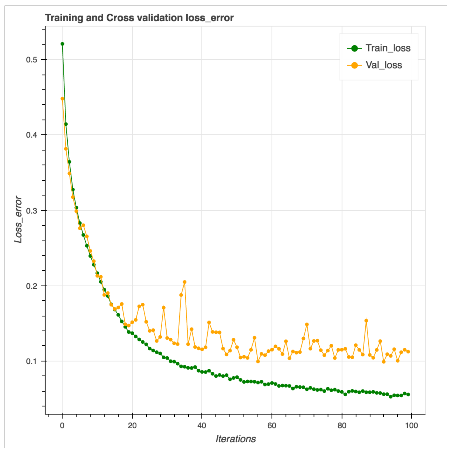
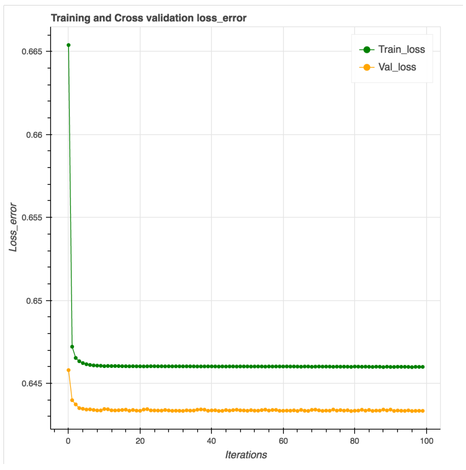


FIGURE 4.11: Transfer Learning illustration.

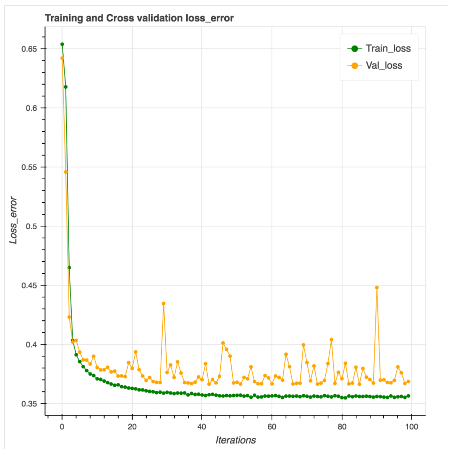
According to the principle, we input the new training dataset into the old model (Figure 4.11), frozen the feature extractor part and make multilayer perceptron part trainable only. In trainable part, usually add new dense layer will increase accuracy [71]. There-



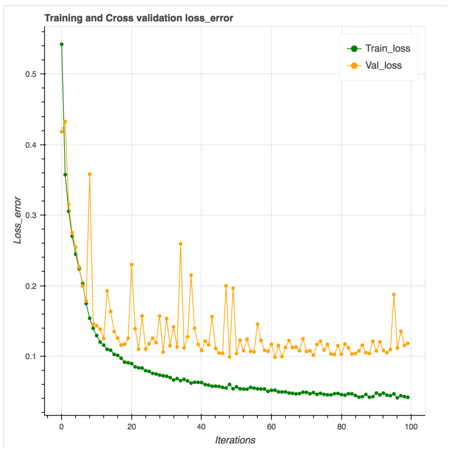
a)



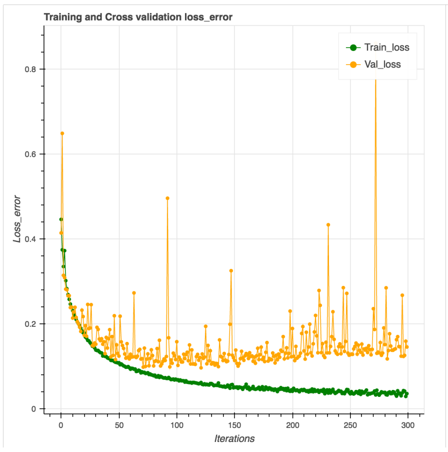
b)



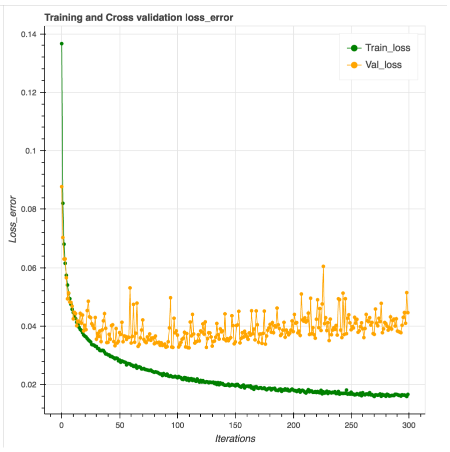
c)



d)



e)



f)

FIGURE 4.12: Transfer Learning learning curve.
(a)Non-frozen + Retrain, (b)Frozen + Retrain, (c)Non-frozen + Add_Dense + Retrain, (d) Frozen + Add_Dense + Retrain, (e)Retrain, (f)Original

fore, in this experiment, we compare the feasibility of transfer learning in 6 cases (Figure 4.12):

Based on the learning curve and result (Table 4.15), frozen the feature extractor and add dense layer in multilayer perceptron can obtain the best identification ability. The corresponding model could contain more adequate feature information and also trained efficiently.

	Non-frozen Retrain	Frozen Retrain	Non-frozen Add_Dense Retrain	Frozen Add_Dense Retrain	Retrain	Original Model
Train Acc	97.97	65.32	87.26	98.15	98.50	99.39
Train Kappa	0.96	00.00	0.72	0.96	0.97	0.98
Test Acc	95.88	98.39	95.32	97.11	95.09	98.47
Test Kappa	0.34	00.00	0.30	0.40	0.29	0.21
Iteration(s)	9.47	2.71	9.52	2.76	9.24	55.81

TABLE 4.15: Transfer Learning result. Comparison between transfer models.

Transfer learning not only has remarkable effect on handwritten digits introduced in previous Chapter, but also on identifying buildings in rural environment. Considering the diversity, building itself is much more complex than digits, and frozen feature extractor part can highly reduce training time. however, when former model don't contain enough feature information, transfer former knowledge well is too hard. Adding additional layers in multilayer perceptron part will increase the accuracy, for it can extract more new information. What's more, transferred model have ability to test both former and new building images.

Chapter 5

Practical Application

In this Chapter, we illustrate the practical applications of identification buildings in rural environment based on CNN models. The testing dataset which in different countries provided without ground truth, result in Figure 5.1 and Figure 5.2, while the part owning ground truth as shown in Figure 5.3 and Figure 5.4.

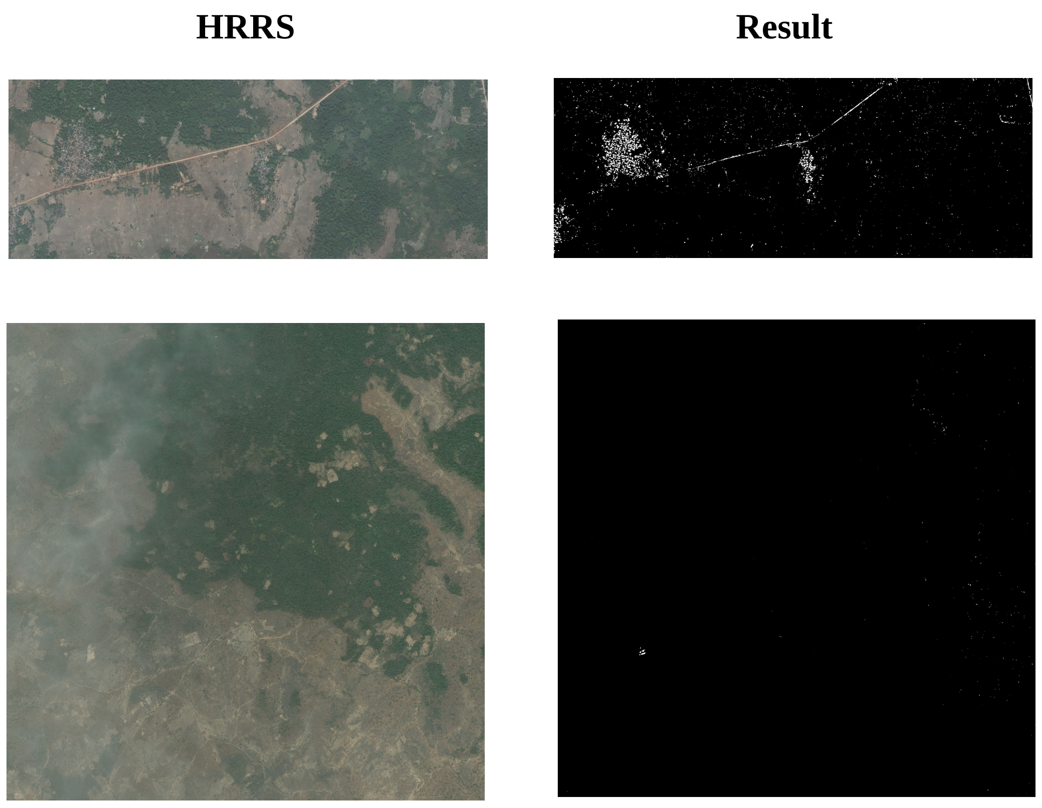


FIGURE 5.1: Result illustration 1. White: buildings, black: Non-buildings

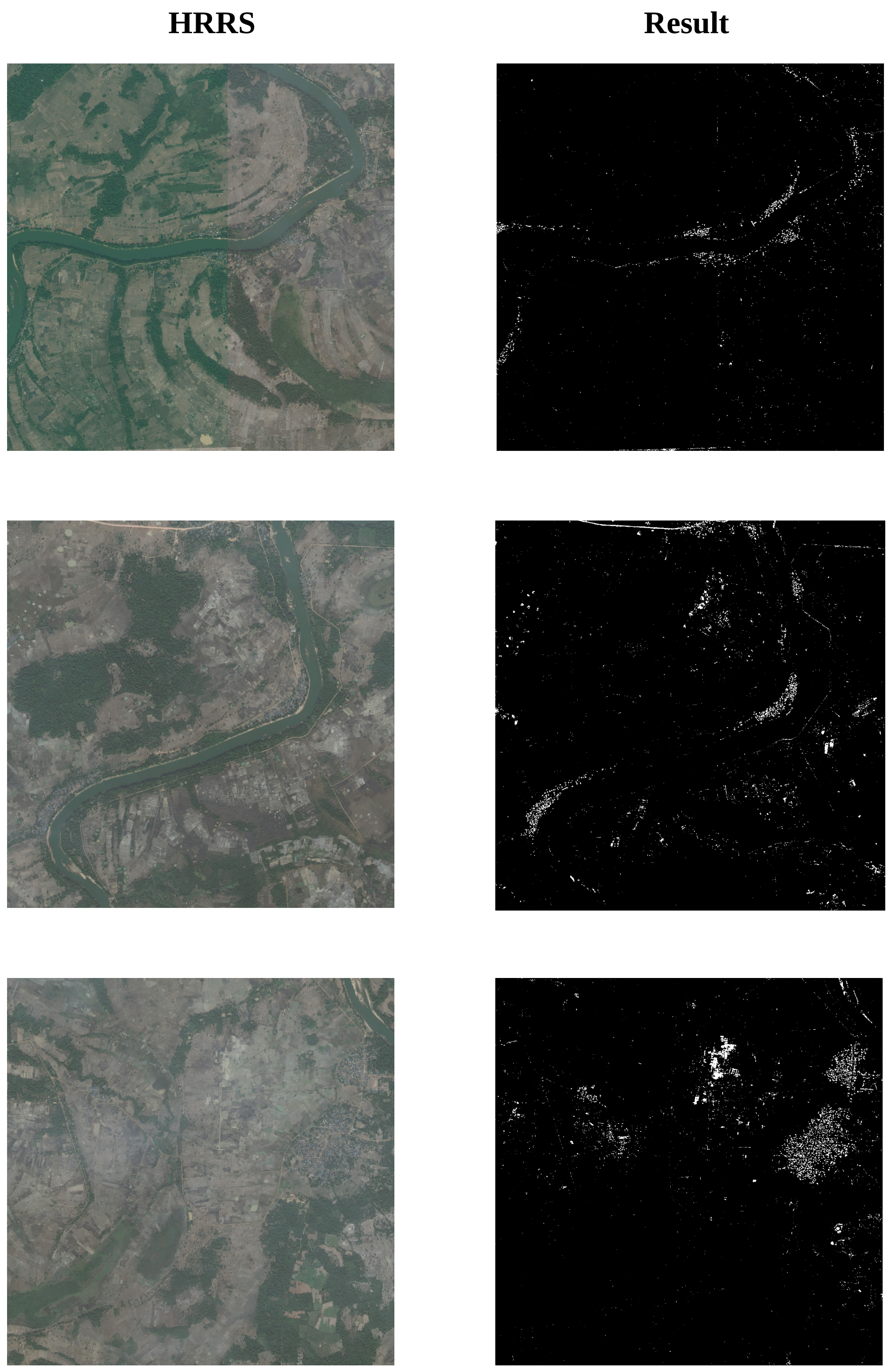


FIGURE 5.2: Result illustration 2. White: buildings, black: Non-buildings

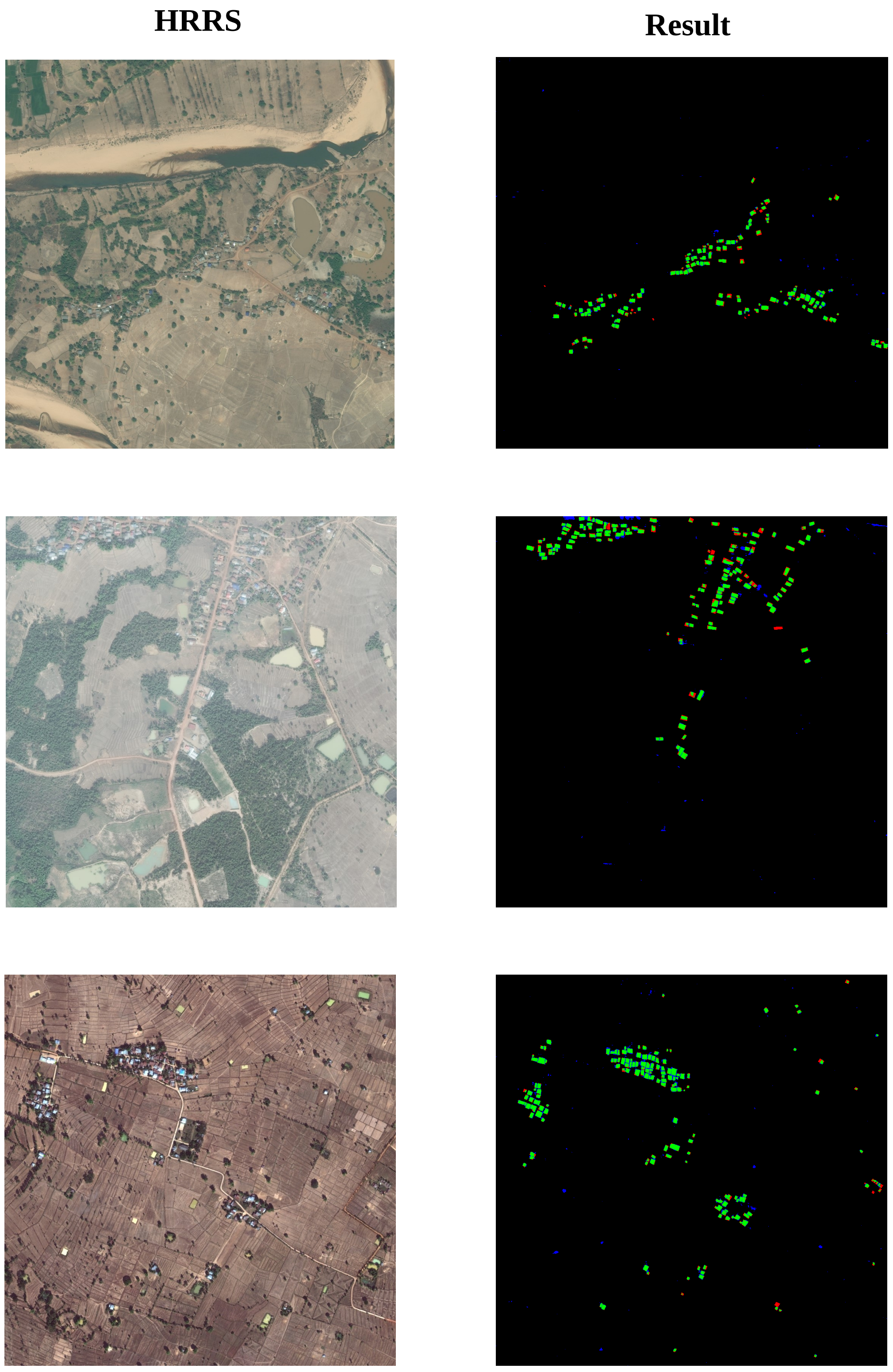


FIGURE 5.3: Result illustration 3. Green: actual buildings that were correctly classified as buildings, blue: the non-buildings that were incorrectly labeled as buildings, red: the buildings that were incorrectly marked as non-buildings, black: the correctly classified non-buildings.

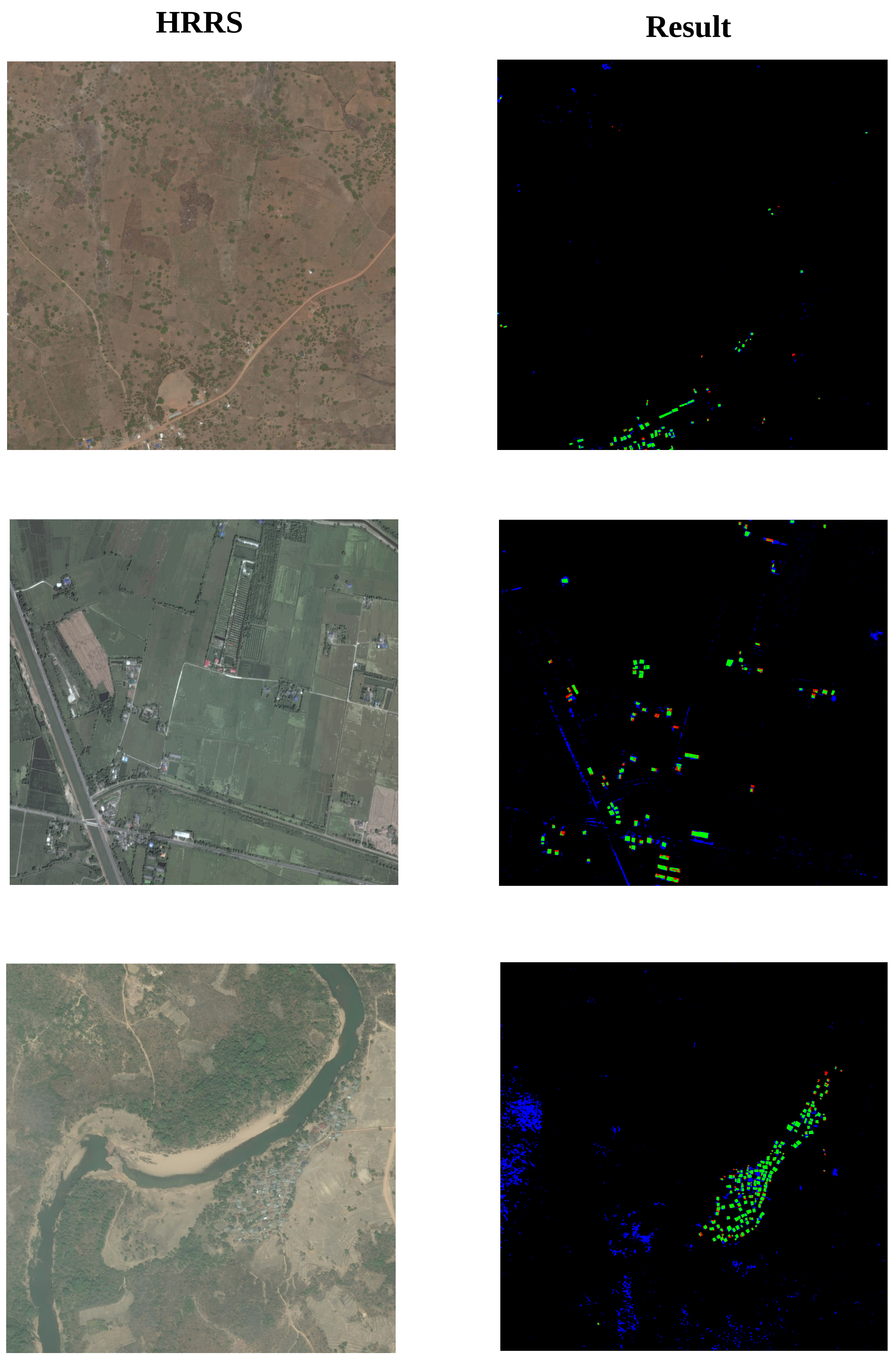


FIGURE 5.4: Result illustration 4. Green: actual buildings that were correctly classified as buildings, blue: the non-buildings that were incorrectly labeled as buildings, red: the buildings that were incorrectly marked as non-buildings, black: the correctly classified non-buildings.

In general, the experimental results indicate that our proposed method based on CNN, has high performance in remote sensing identification of buildings in rural environment. Meanwhile, the results also infer the land features such as roads and rivers are very easy to be misclassified, because the exterior structure of which are quite similar with building, solving such problems would become very significant to enhance performance in the future.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

In this dissertation, we propose a supervised machine learning method based on Convolutional Neural Networks for building identification in rural environment via high resolution remote sensing images. The corresponding machine learning methods help us obtain building identification classifiers by training the designed samples. Applying the obtained CNN classifiers to automatically extract information of buildings from the remote sensing images generates the identification maps of buildings.

The proposed method shows the ability of CNN in building detection, which is experimentally demonstrated by including several kinds of areas in developing countries such as Laos and Kenya. The obtained classifier can be used for all cases and no manual interaction is needed. Our method of CNN achieves a very high overall accuracy and Kappa, which outperforms other methods. What's more, we reconstruct CNN model based on state-of-the-art structures and also provide methods such as Transfer Learning, color balance and data augmentation to enhance the robustness of building identification classifier.

Furthermore, the proposed method can be efficiently utilized in remote sensing recognition of not merely buildings. By training the prior knowledge of the corresponding identification samples, the proposed method can generate a classifier that has the ability to classify relative targets and leads to promising classification results. Therefore, based on the result of this experiment, our proposed method is a promising approach that might be applied to many potential applications in the near future.

6.2 Future Works

Although this study indicates the proposed method could be efficiently used in building identification, further and more detailed exploration on the method is required in the future. First, to test the method's stability, more extended and sophisticated areas need to be tested. Second, to alleviate the labor-intensive task of training the data, we will apply the learned model of one area to other areas with similar landscapes. Unfortunately, the spectral characteristics of remote sensing images respond to the varying conditions of image capture. To improve the performance in such cases, we will collect more training dataset and consider a better transfer learning technique. Third, we will apply the proposed method to other feature identifications in high-resolution remote sensing images, such as roads and agricultural land. We are also interested in extending this method to classifications of multi-class landscapes. We believe that the proposed method has great practical value for solving diverse classification problems.

There are also many applications can be implemented in the future, such as building age prediction by comparing buildings condition in same area via remote sensing images obtained in different year. As shown in Figure 6.1, if we detect new buildings in the newer HHRS image while vacant in older one, the building age can be easily predicted. And the similar application such as automatically building amount counting and building outline detection can also be conducted.

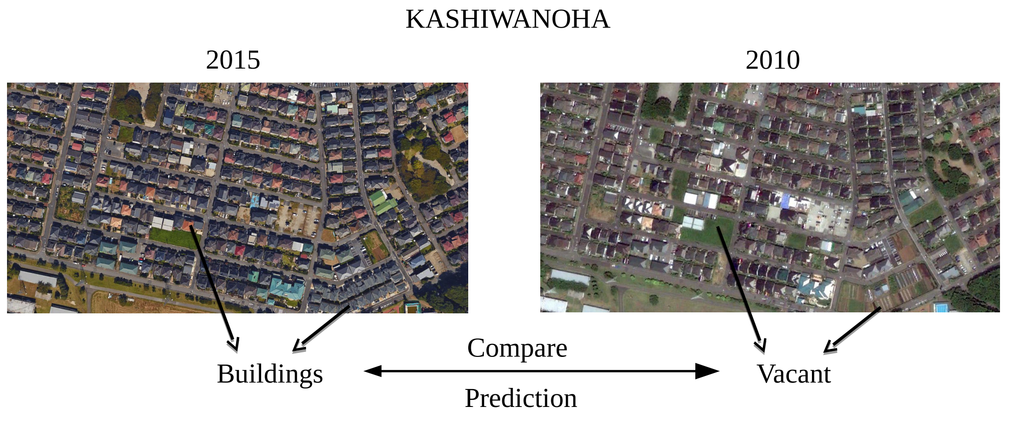


Figure 6.1: Building age predition.

Japan is one of the countries most affected by natural disasters; the catastrophes such as earthquake, Tsunami and landslide usually cause enormous losses. As an indispensable resource, maps used to illustrate land conditions after catastrophe are quite significant. In this research plan, a system for automatic and real-time generalization of catastrophe maps is proposed.

Rather than existing methodologies, which highly depend on human beings, here, by considering the characteristics and importance of catastrophe maps, we propose a brand new map generalization system based on combining geographic information system (GIS) and machine learning methods, which would be capable to automatically provide accurate, efficient and time-sequenced catastrophe maps.

As shown in Figure 6.2, By implementing unmanned aerial vehicles (UAV), machine learning methods and geographic sensors, the important land features such as safe roads, broken buildings can be identified; meanwhile, the digitalized map with accurate geographic coordinates can be generated as well. Base on the obtained results, life and property would be saved. Furthermore, not only in catastrophe, this system could also be used in many other map generalization conditions.

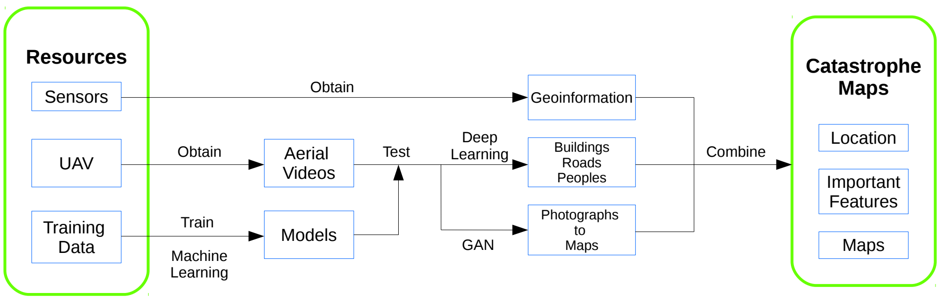


FIGURE 6.2: System for Automatic and Real-time Generalization of Catastrophe maps.

What's more, we can combine the proposed method with other dataset such as trajectory to implement more promising and interesting researches in the future.

Bibliography

[1] Nicolas H Younan and Selim Aksoy. Foreword to the special issue on pattern recognition in remote sensing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(5):1331–1333, 2012.

[2] Jaewoong Choi, Junyoung Lee, Dongwook Kim, Giacomo Soprani, Pietro Cerri, Alberto Broggi, and Kyongsu Yi. Environment-detection-and-mapping algorithm for autonomous driving in rural or off-road environment. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):974–982, 2012.

[3] Huilin Xing and Xiwei Xu. *M8. 0 Wenchuan Earthquake*, volume 123. Springer, 2010.

[4] Jonathan J Davies, Alastair R Beresford, and Andy Hopper. Scalable, distributed, real-time map generation. *IEEE Pervasive Computing*, 5(4):47–54, 2006.

[5] Jianrong Fan, Jim X Chen, Bingwei Tian, Dong Yan, Genwei Cheng, Peng Cui, and Wen Zhang. Rapid assessment of secondary disasters induced by the wenchuan earthquake. *Computing in science & engineering*, 12(1):10–19, 2010.

[6] Lei Wang, Paul D Groves, and Marek K Ziebart. Urban positioning on a smartphone: Real-time shadow matching using gnss and 3d city models. The Institute of Navigation, 2013.

[7] Rebecca Bunnell, Jonathan Mermin, and Kevin M De Cock. Hiv prevention for a threatened continent: implementing positive prevention in africa. *Jama*, 296(7): 855–858, 2006.

[8] Nick Gallent, Meri Juntti, Sue Kidd, and Dave Shaw. *Introduction to rural planning.* Routledge, 2008.

[9] Joan Davidson and Gerald Wibberley. *Planning and the Rural Environment: Urban and Regional Planning Series.* Elsevier, 2016.

[10] Thomas Lillesand, Ralph W Kiefer, and Jonathan Chipman. *Remote sensing and image interpretation.* John Wiley & Sons, 2014.

[11] J Richards and X Jia. Remote sensing digital image analysis: An introduction springer-verlag. *Berlin Heidelberg*, 1999.

[12] Robert A Schowengerdt. *Remote sensing: models and methods for image processing.* Academic press, 2006.

[13] Agnès Bégué, Elodie Vintrou, Denis Ruelland, Maxime Claden, and Nadine Dessay. Can a 25-year trend in soudano-sahelian vegetation dynamics be interpreted in terms of land use change? a remote sensing approach. *Global Environmental Change*, 21(2):413–420, 2011.

[14] Wenbin Wu, Ryosuke Shibasaki, Peng Yang, Qingbo Zhou, and Huajun Tang. Remotely sensed estimation of cropland in china: A comparison of the maps derived from four global land cover datasets. *Canadian Journal of Remote Sensing*, 34(5): 467–479, 2008.

[15] Xinyang Yu, Anding Zhang, Xiyong Hou, Mingjie Li, and Yingxiao Xia. Multi-temporal remote sensing of land cover change and urban sprawl in the coastal city of yantai, china. *International Journal of Digital Earth*, 6(sup2):137–154, 2013.

[16] Hang Zhou, Elena Aizen, and Vladimir Aizen. Deriving long term snow cover extent dataset from avhrr and modis data: Central asia case study. *Remote sensing of environment*, 136:146–162, 2013.

[17] Jordan B Long and Chandra Giri. Mapping the philippines' mangrove forests using landsat imagery. *Sensors*, 11(3):2972–2981, 2011.

[18] Andrea S Laliberte, DM Browning, and Albert Rango. A comparison of three feature selection methods for object-based classification of sub-decimeter resolution ultracam-l imagery. *International Journal of Applied Earth Observation and Geoinformation*, 15:70–78, 2012.

[19] Shaohui Sun and Carl Salvaggio. Aerial 3d building detection and modeling from airborne lidar point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(3):1440–1449, 2013.

[20] Dennis C Duro, Steven E Franklin, and Monique G Dubé. A comparison of pixel-based and object-based image analysis with selected machine learning algorithms for the classification of agricultural landscapes using spot-5 hrg imagery. *Remote Sensing of Environment*, 118:259–272, 2012.

[21] Jianhong Guo, Lu Liang, and Peng Gong. Removing shadows from google earth images. *International Journal of Remote Sensing*, 31(6):1379–1389, 2010.

[22] David Potere. Horizontal positional accuracy of google earth's high-resolution imagery archive. *Sensors*, 8(12):7973–7981, 2008.

[23] Qiong Hu, Wenbin Wu, Tian Xia, Qiangyi Yu, Peng Yang, Zhengguo Li, and Qian Song. Exploring the use of google earth imagery and object-based methods in land use/cover mapping. *Remote Sensing*, 5(11):6026–6042, 2013.

[24] Le Yu and Peng Gong. Google earth as a virtual globe tool for earth science applications at the global scale: progress and perspectives. *International Journal of Remote Sensing*, 33(12):3966–3986, 2012.

[25] Lucian Drăguţ, Dirk Tiede, and Shaun R Levick. Esp: a tool to estimate scale parameter for multiresolution image segmentation of remotely sensed data. *International Journal of Geographical Information Science*, 24(6):859–871, 2010.

[26] Qian Yu, Peng Gong, Nick Clinton, Greg Biging, Maggi Kelly, and Dave Schirokauer. Object-based detailed vegetation classification with airborne high spatial resolution remote sensing imagery. *Photogrammetric Engineering & Remote Sensing*, 72(7):799–811, 2006.

[27] Chisa Shinsugi, Masaki Matsumura, Mohamed Karama, Junichi Tanaka, Mwatasa Changoma, and Satoshi Kaneko. Factors associated with stunting among children according to the level of food insecurity in the household: a cross-sectional study in a rural community of southeastern kenya. *BMC public health*, 15(1):441, 2015.

[28] Sankar K Pal and Amita Pal. *Pattern recognition: from classical to modern approaches*. World Scientific, 2001.

[29] Mariana Belgiu and Lucian Drăguţ. Random forest in remote sensing: A review of applications and future directions. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:24–31, 2016.

[30] Fei Lv, Min Han, and Tie Qiu. Remote sensing image classification based on ensemble extreme learning machine with stacked autoencoder. *IEEE Access*, 2017.

[31] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[32] Peter M Atkinson and ARL Tatnall. Introduction neural networks in remote sensing. *International Journal of remote sensing*, 18(4):699–709, 1997.

[33] Farid Melgani and Lorenzo Bruzzone. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on geoscience and remote sensing*, 42(8):1778–1790, 2004.

[34] Alessia Mammone, Marco Turchi, and Nello Cristianini. Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):283–289, 2009.

[35] Jian Zheng, Zhanzhong Cui, Anfei Liu, and Yu Jia. A k-means remote sensing image classification method based on adaboost. In *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, volume 4, pages 27–32. IEEE, 2008.

[36] Ugur Zongur, Ugur Halici, Orsan Aytekin, and Ilkay Ulusoy. Airport runway detection in satellite images by adaboost learning. In *Image and Signal Processing for Remote Sensing, Proceedings of SPIE*, volume 7477, page 747708, 2009.

[37] Rui Li, Jiulin Sun, Juanle Wang, Lijun Zhu, and Rui Liu. The study on dynamic extraction of urban land use cover with remote sensing image based on adaboost algorithm. In *Sixth International Symposium on Multispectral Image Processing and Pattern Recognition*, pages 74981U–74981U. International Society for Optics and Photonics, 2009.

[38] Melih Cetin, Ugur Halici, and Örsan Aytekin. Building detection in satellite images by textural features and adaboost. In *Pattern Recognition in Remote Sensing (PRRS), 2010 IAPR Workshop on*, pages 1–4. IEEE, 2010.

[39] Jie Dou, Kuan-Tsung Chang, Shuisen Chen, Ali P Yunus, Jin-King Liu, Huan Xia, and Zhongfan Zhu. Automatic case-based reasoning approach for landslide detection: integration of object-oriented image analysis and a genetic algorithm. *Remote Sensing*, 7(4):4318–4342, 2015.

[40] Xianju Li, Xinwen Cheng, Weitao Chen, Gang Chen, and Shengwei Liu. Identification of forested landslides using lidar data, object-based image analysis, and machine learning algorithms. *Remote Sensing*, 7(8):9705–9726, 2015.

[41] Sara Attarchi and Richard Gloaguen. Classifying complex mountainous forests with l-band sar and landsat data integration: A comparison among different machine learning methods in the hyrcanian forest. *Remote Sensing*, 6(5):3624–3647, 2014.

[42] Wenlong Jing, Yaping Yang, Xiafang Yue, and Xiaodan Zhao. Mapping urban areas with integration of dmsp/ols nighttime light and modis data using machine learning techniques. *Remote Sensing*, 7(9):12419–12439, 2015.

[43] Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 2015.

[44] Jake Bouvrie. Notes on convolutional neural networks. 2006.

[45] Xueyun Chen, Shiming Xiang, Cheng-Lin Liu, and Chun-Hong Pan. Vehicle detection in satellite images by parallel deep convolutional neural networks. In *Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on*, pages 181–185. IEEE, 2013.

[46] Bao-Qing Li and Baoxin Li. Building pattern classifiers using convolutional neural networks. In *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, volume 5, pages 3081–3085. IEEE, 1999.

[47] Jun Yue, Wenzhi Zhao, Shanjun Mao, and Hui Liu. Spectral–spatial classification of hyperspectral images using deep convolutional neural networks. *Remote Sensing Letters*, 6(6):468–477, 2015.

[48] Stefan Lee, Haipeng Zhang, and David J Crandall. Predicting geo-informative attributes in large-scale image collections using convolutional neural networks. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 550–557. IEEE, 2015.

[49] Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3288–3291. IEEE, 2012.

[50] Dimitrios Marmanis, Mihai Datcu, Thomas Esch, and Uwe Stilla. Deep learning earth observation classification using imagenet pretrained networks. *IEEE Geoscience and Remote Sensing Letters*, 13(1):105–109, 2016.

[51] Jun Ding, Bo Chen, Hongwei Liu, and Mengyuan Huang. Convolutional neural network with data augmentation for sar target recognition. *IEEE Geoscience and Remote Sensing Letters*, 13(3):364–368, 2016.

[52] Fan Hu, Gui-Song Xia, Jingwen Hu, and Liangpei Zhang. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sensing*, 7(11):14680–14707, 2015.

[53] Martin Längkvist, Andrey Kiselev, Marjan Alirezaie, and Amy Loutfi. Classification and segmentation of satellite orthoimagery using convolutional neural networks. *Remote Sensing*, 8(4):329, 2016.

[54] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pages 958–962, 2003.

[55] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[56] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques, 2007.

[57] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[58] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[59] Ivo F Sbalzarini and Petros Koumoutsakos. Feature point tracking and trajectory analysis for video imaging in cell biology. *Journal of structural biology*, 151(2): 182–195, 2005.

[60] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[61] Derek Greene, Pádraig Cunningham, and Rudolf Mayer. Unsupervised learning and clustering. *Machine learning techniques for multimedia*, pages 51–90, 2008.

[62] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[63] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[64] O Martınez Mozos, Cyrill Stachniss, and Wolfram Burgard. Supervised learning of places from range data using adaboost. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1730–1735. IEEE, 2005.

[65] Freund Yoav and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, 1995. *CiteSeerX*, 10(1.56):9855.

[66] Simon S Haykin, Simon S Haykin, Simon S Haykin, and Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, NJ, USA:, 2009.

[67] B Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[68] David John Finney. Probit analysis: a statistical treatment of the sigmoid response curve. 1952.

[69] Andrew Ng, Jiquan Ngiam, Chuan Yu Foo, Yifan Mai, Caroline Suen, Adam Coates, Andrew Maas, Awni Hannun, Brody Huval, Tao Wang, et al. Unsupervised feature learning and deep learning, 2013.

[70] Rasmus Berg Palm. Prediction as a candidate for learning deep hierarchical models of data. *Technical University of Denmark*, 5, 2012.

[71] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[72] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

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

[74] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[75] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[76] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[77] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[78] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[80] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2001.

[81] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[82] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[83] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[84] George Forman. Bns feature scaling: an improved representation over tf-idf for svm text classification. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 263–270. ACM, 2008.

[85] Engui Fan. Extended tanh-function method and its applications to nonlinear equations. *Physics Letters A*, 277(4):212–218, 2000.

[86] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[87] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[88] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[89] Sylvia Frühwirth-Schnatter. Data augmentation and dynamic linear models. *Journal of time series analysis*, 15(2):183–202, 1994.

[90] Armin Gruen and Haihong Li. Road extraction from aerial and satellite images by dynamic programming. *ISPRS Journal of Photogrammetry and Remote Sensing*, 50(4):11–20, 1995.

[91] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[92] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[93] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[94] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

[95] Jean Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational linguistics*, 22(2):249–254, 1996.

２０１７年度　修士論文　CNNを利用した非都市地域における建物の抽出方法に関する研究

郭　直霊