Hyperspectral Image Classification Using Deep Neural Network

by

Yiding SHI

Advised by Professor PHILIP HENG WAI LEONG

A thesis submitted in partial fulfillment for the
degree of Bachelor of Engineering (Software)

in the
School of Electrical and Information Engineering

July 2018

Statement of Achievements

This thesis implements a hyperspectral image classification method based on deep neural network. It constructs a deep neural network model and implements processing and deep learning of image information with multi-dimensional spectral channels. It can classify the earth surface objects represented by image pixels with high accuracy and keep the complexity of time and memory within a relatively low level.

Abstract

Hyperspectral image is a commonly used technique for earth surface survey in ecology, mining, and hydrological application. Compared with other survey methods, it has the advantage of lower cost and faster data collection. Briefly, it is an image form that contains hundreds of narrow spectral channels in every single pixel, which is a measured value of the corresponding wavelength. By processing the spectral and spatial information in the hyperspectral image, each pixel and the surface objects they represent will be identified and classified. At present, the most commonly used method for processing hyperspectral images is based on deep learning. However, due to some reasons, such as the defects of neural network design and fewer available training samples, the performance of classification needs to be improved. My main contribution in this thesis is that: 1)A preprocessing of the presentation extraction on hyperspectral image data set is implemented, which is utilized to efficiently extract the spatial and spectral information of hyperspectral images. Propose an improved extraction technique on edge pixels, which includes more accurate spatial information. 2) Propose a deep convolutional neural network model for hyperspectral image classification. Many techniques, such as dropout and regularization, are applied to optimize the neural network performance, solving problems like overfitting, and the small mount of training samples. 3) Experiments are conducted on actual hyperspectral image data set, and compared with other peoples previous work. A high classification accuracy is obtained as the result.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor PHILIP HENG WAI LEONG, for his support and encouragement of my work. Without his help and expert advice, it is impossible to complete this thesis.

In addition, I would like to thank all my friends and classmates, such as Zhiwang Zhang who helped me in this project. Without their help, I could not well understand the application of hyperspectral image and the principle of neural network. The establishment of the neural network model in this thesis is also impossible.

Contents

Statement of Achievements ii

Abstract iii

Acknowledgements iv

List of Figures viii

List of Tables ix

Abbreviations x

1 Introduction 1

2 Literature review 3
2.1 Introduction . . . 3
2.2 Hyperspectral Images . . . 3
2.2.1 Introduction . . . 3
2.2.2 The Imaging process . . . 4
2.2.3 Curse of dimensionality . . . 5
2.2.4 Limited amount of labeled training data . . . 6
2.2.5 Complex light-scattering mechanisms . . . 7
2.3 Convolutional Neural Network . . . 8
2.4 The Development of HSI Classification . . . 12
2.5 The Significance and Purpose of This Research . . . 14

3 Methods 15
3.1 Introduction . . . 15
3.2 Development environment . . . 15
3.2.1 Python . . . 15
3.2.2 Tensorflow . . . 16
3.2.3 Keras . . . 16
3.3 Data set introduction and processing . . . 16
3.3.1 Feature extraction . . . 17
3.3.2 Edge problem in feature extraction . . . 20
3.3.3 Data set splitting . . . 22

3.3.4 Zero mean Normalization . . . 22
3.4 Structure . . . 23
3.4.1 Layers . . . 23
3.4.2 Relu and dropout . . . 25
3.4.3 L2 regularization . . . 26

4 Results **28**
4.1 Training . . . 28
4.2 Result of testing . . . 28
4.3 Compare with other people's work . . . 33
4.4 Structural details and their performance . . . 34
4.4.1 Patch size in preprocessing . . . 34
4.4.2 Edge problem in feature extraction . . . 36
4.4.3 Dropout method . . . 37
4.4.4 Learning rate decay . . . 38

5 Discussion **41**
5.1 Results evaluation . . . 41
5.2 Complexity evaluation . . . 42
5.2.1 Time complexity . . . 42
5.2.2 Memory complexity . . . 44
5.3 Improvement . . . 44

6 Conclusions **46**

Bibliography **47**

List of Figures

2.1 Hyperspectral image 4
2.2 Schematic diagram of Imaging process 5
2.3 Finite data set accuracy 6
2.4 Atmospheric transmittance versus wavelength 7
2.5 Aerial maps and initial classification maps 8
2.6 Shared weighs in CNN 9
2.7 Convolution layer formula 10
2.8 Brief example of convolutional layer 10
2.9 Pooling layer formula 11
2.10 Brief example of pooling layer 11

3.1 Aerial maps and groundtruth classification maps of Indian Pines 17
3.2 Single pixel feature extraction 19
3.3 Labeled data after extracting 19
3.4 Neighbouring pixels feature extraction 19
3.5 Labeled data after extracting 20
3.6 Extracting with discarding pixels on edge 20
3.7 Extracting with padding the centred pixel 21
3.8 Extracting with padding the pixels on image edge 22
3.9 Formula of zero mean normalization 23
3.10 Re-shift parameter phase in zero mean normalization 23
3.11 ReLU activation function 25
3.12 Normal neural network and neural network with Dropout 26
3.13 L2 regularization formula 26

4.1 Result: Loss versus epochs 29
4.2 Result: Classification accuracy versus epochs 29
4.3 The input data generated by different patch sizes performs on CNN . . . 35
4.4 Proposed CNN without Dropout 38
4.5 Loss changes without learning rate decay 39
4.6 Classification accuracy changes without learning rate decay(larger fluctuations appearing) 40

5.1 Time complexity of a single convolutional layer 43
5.2 Size of feature map 43
5.3 Time complexity of convolutional layers 43
5.4 Memory complexity of neural network 44

List of Tables

3.1 Classes in Inidan Pines data set 18
3.2 Architecture of the neural network in this thesis 24

4.1 Result: Confusion matrix 30
4.2 Accuracy on classes 31
4.3 Confusion matrix of the best classifier in proposed CNN 32
4.4 Accuracy on classes of the best classifier in proposed CNN 33
4.5 Compare with other people's work 34
4.6 Number of classes in Indian Pines when feature extracting 37

Abbreviations

HSI	**H**yper **S**pectral **I**mage
RGB color model	**R**ed **G**reen **B**lue color model
CNN	**C**onvolutional **N**eural **N**etwork
ReLU	**Re**ctified **L**inear **U**nit
SVM	**S**upport **V**ector **M**achine
API	**A**pplication **P**rogramming **I**nterface
PURR	**P**urdue **U**niversity **R**esearch **R**epository
AVIRIS	**A**irborne **V**isible **I**nfrared **R**adiation **I**maging **S**pectrometer
SGD	**S**tochastic **G**radient **D**escent
FLOPs	**FL**oating-point **OP**erations per **s**econd
RNN	**R**ecurrent **N**eural **N**etwork
ResNet	**R**esidential **N**etwork
DenseNet	**D**ensely Connected Convolutional **N**etworks
LSTM	**L**ong **S**hort **T**erm **M**emory

Chapter 1

Introduction

Hyperspectral image (HSI) refers to an image form that contains hundreds of narrow spectral channels in every single pixel. These data are collected from the surface of the Earth, simultaneously recorded by sensors of artificial satellites which has been placed into the planet orbit. This method, or called space-based remote sensing, relies mainly on the fact that information can be collected from electromagnetic energy fields radiated from the planet's surface, especially from the spatial and spectral signals [1]. The information collected by hyperspectral image sensors in pixels level accuracy is represented by vectors where each data is a measured value of the corresponding wavelength. The size of the vector is determined by the number of spectral channels of the sensors.

According to [2], HSI system sensors are mostly deployed to sense a wide range of spectral signals, especially the middle infrared wavelength. For hyperspectral images, the number of spectral data channels can usually reach hundreds. Basing on the fact that every object of material has its unique characteristic of spectral reflection, these detailed and complex data can provide the possibility for people to research associated imaging classification tools to accurately identify Earth surface objects[3][4]. This method does not distinguish objects by visual means but instead collects and senses a series of spectral mixtures emitted from different closely spaced objects. Then, by analyzing the difference in the distribution of energy at different wavelengths in each pixel, the objects of different materials are distinguished or classified.

The contribution I made in this thesis are as follows:

1)A preprocessing of the presentation extraction on hyperspectral image data set is implemented, which is utilized to efficiently extract the spatial and spectral information of hyperspectral images. Propose an improved extraction technique on edge pixels, which includes more accurate spatial information.

2) Propose a deep convolutional neural network model for hyperspectral image classification. Many techniques, such as dropout and regularization, are applied to optimize the neural network performance, solving problems like overfitting, and the small mount of training samples.

3) Experiments are conducted on actual hyperspectral image data set, and compared with other peoples previous work. A high classification accuracy is obtained as the result.

This thesis will introduce the background of the project, show the design details of the solution, and then give a comprehensive analysis. The chapter 2 will introduce the background and literature review of this field. It includes hyperspectral image and its features, as well as the classification methodologies of deep neural network, while enumerating and introducing the researches and achievements that have contributed during the forming process. In chapter 3, the detail of the proposed neural network design will be introduced. In chapter 4, I will analyze this model and compare it with others. In chapter 5, based on the strengths and weaknesses, I will point out the future development directions and the parts that can be considered to improve.

Chapter 2

Literature review

2.1 Introduction

Deep learning is a technology that implements machine learning. It can be understood as a neural network structure with multiple hidden layers[5]. Convolutional neural network is one kind of deep learning neural network, which has become a hot topic in the field of speech analysis and image recognition. The spectral information contained in the hyperspectral image not only greatly improves the information richness, but also provides a possibility of more reasonable and effective analysis and processing in the processing technology. This chapter will introduce hyperspectral image, neural networks and convolutional neural networks, including their features and applications. In addition, a literature review of hyperspectral image classification based on deep neural networks will also be introduced.

2.2 Hyperspectral Images

2.2.1 Introduction

Hyperspectral image is a form of image with a few relatively broad wavelength bands is produced by multispectral remote sensors[6]. Compared to normal color images (three wavelength information, RGB color model), dozens or hundreds of narrow, adjacent spectral bands data in each pixel are collected simultaneously. These measurements can

be derived as a continuous spectrum for each image pixel, as shown in the Figure 2.1. For different materials and objects on the earth's surface, sensors produce a different continuous spectrum. Through neural network classification techniques, measurements of certain types of ground materials such as vegetation or minerals can be identified and classified[7]. This is also the technical foundation of applications of hyperspectral images in many fields.

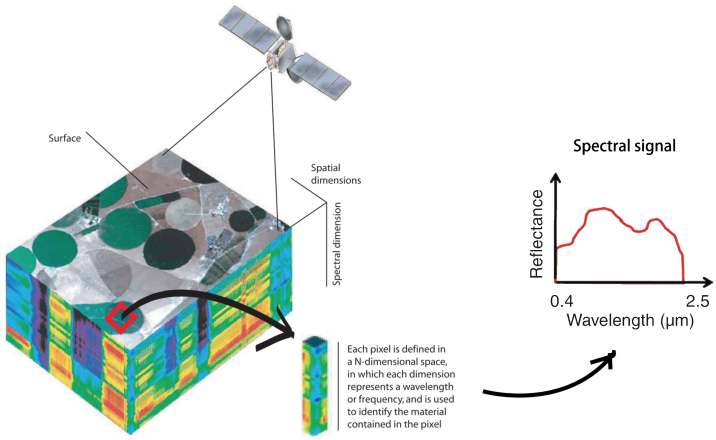

FIGURE 2.1: Hyperspectral image

2.2.2 The Imaging process

Hyperspectral images are collected by devices called spectrometers through the combined use of two technologies, spectroscopy and the remote imaging of Earth surfaces[8]. Spectroscopy is a science that studies reflected light, emitted light, and their wavelength changes from materials. The optical components in Spectrometers divide the light entering the lens into many different wavelength components. Each wavelength component is measured by different sensors. With the integration of a large amount of spectral information, the data of this measured ray in a spectral range can be obtained. Remote imaging refers to the process of collecting spatially related images by spatially moving sensor platforms over the Earth's surface[9]. In this way, hyperspectral images with both

spatial and spectral information are generated. The optical and sensing processes are shown in Figure 2.2.

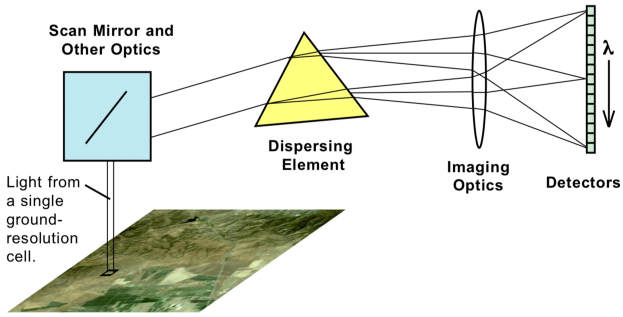

FIGURE 2.2: Schematic diagram of Imaging process

2.2.3 Curse of dimensionality

In the domain of hyperspectral images spectral information, vectors are used for representing pixels, whose components record corresponding specific wavelengths. The fixed number of spectral bands in sensors capability spectral bands determine the size of vectors. Compared to ordinary multispectral images, hyperspectral images have the ability to record hundreds of spectral channels. The direct problem caused by the high number of spectral channel is that computers process at high loads, which will inevitably lead to the drawbacks of larger operational energy consumption and a long time spent[7].

Apart from the unnecessary computational load, theoretical and practical problems will arise with the increasing spectral dimensionality. This kind of significant challenges machine learning faces is referred to as the Hughes effect[10]. In machine learning cases that trained by limited and fixed number of training samples, its predictive power will reduce as the dimensionality increases[11].

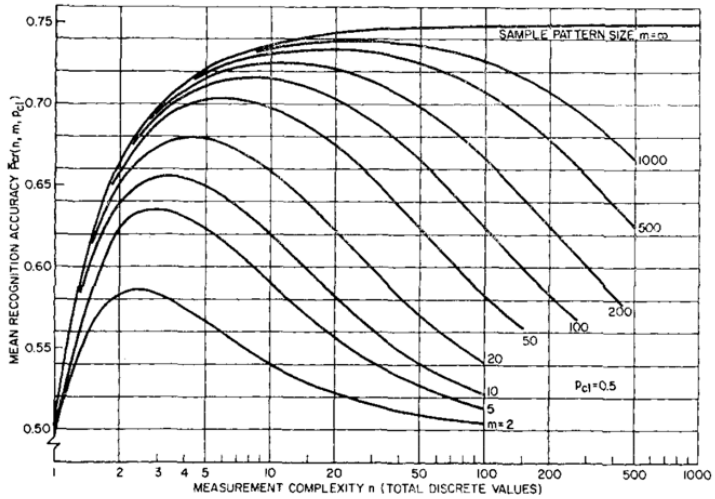

FIGURE 2.3: Finite data set accuracy

According to the theory of high-dimensional data analysis indicated by Hughes, G (1968)[12], for fixed sample pattern size n, the accuracy first begins to rise with measurement complexity m increases. Shown in Figure 2.3, it ultimately falls back as n/m due to the precision of the probability estimates mono-tonically degrades. Because of this fact of decreasing accuracy, compared to ordinary multispectral images, hyperspectral images is still a challenging task.

2.2.4 Limited amount of labeled training data

Another vital problem for hyperspectral images classification techniques is the limited amount of available training data (labeled data). Compared to the relatively simple collection process (collected by satellite sensors), the collection process of labeled training data is accompanied by ground campaigns, which is time and labor consuming[13]. In machine learning, overfitting is likely to occur in the case of insufficient training data, resulting in the result of learning to be too complicated (too many parameters relative to the number of observations) and too perfect to match the training data 14]. The appearance of overfitting also means this model has poor predictive performance applying to the unknown data. However, the model should apply to generalised situations rather than just the existing data used in training.

2.2.5 Complex light-scattering mechanisms

The effects of the atmosphere are ubiquitous in the imaging process, and even relatively clear atmospheres interact with the incoming light and light reflected by objects. These interactions reduce the energy of incoming light reaching the earth surface objects at some wavelengths and reduce the energy they transmit to satellite sensors. Because of absorption of gases and scatting of molecules, the transmittance rate of atmosphere is reduced.

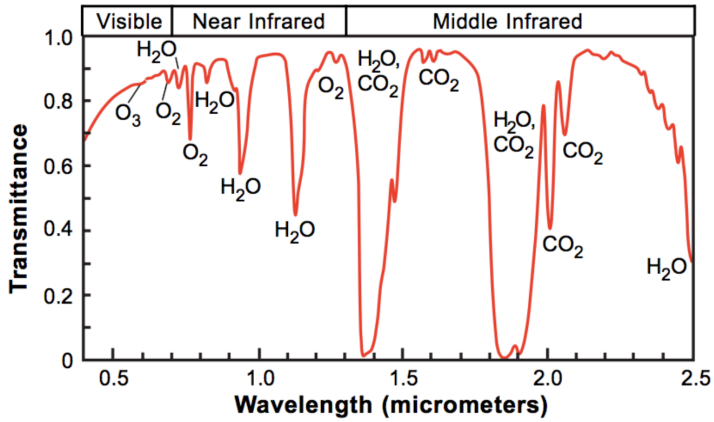

FIGURE 2.4: Atmospheric transmittance versus wavelength

Figure 2.4 illustrates atmospheric transmittance rate versus wavelength for atmospheric conditions[9]. Obviously, because of the combined effects of these, water and carbon dioxide cause the energy of some bands to be almost completely reduced. Therefore, hyperspectral sensors can then get so little useful information on this wavelength range. In the data set named "India pines" used in this thesis, the corruption problem of the image due to water absorption has been solved, i.e. these bands are usually manually removed in the data preprocessing step[14].

Figure 2.5 is the aerial maps over the earth surface (a) and the initial classification maps(b) for different data set (the above is the Indian Pines image, the below is the Pavia University image)[15]. Obviously, noisy scatter points appear in the images, which caused by some complex light-scattering effects.

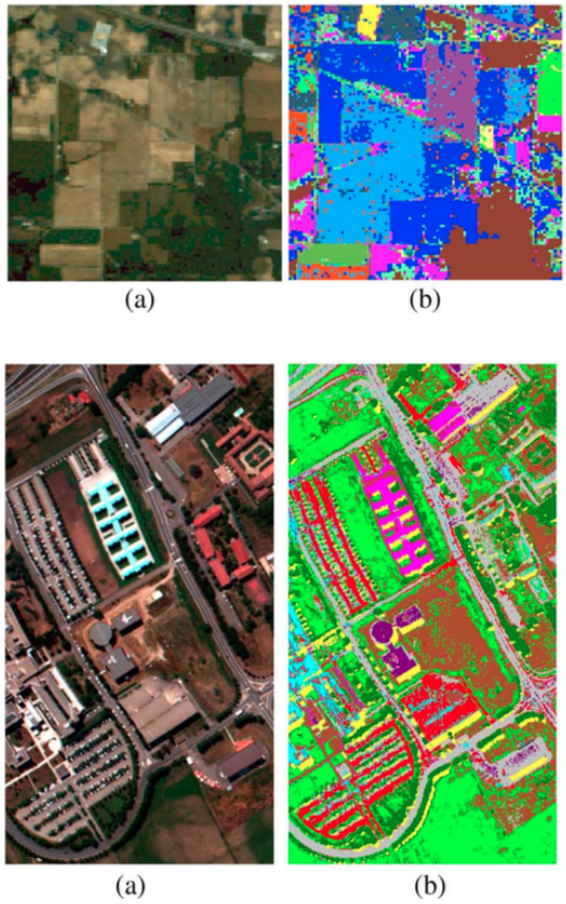

FIGURE 2.5: Aerial maps and initial classification maps

2.3 Convolutional Neural Network

The human brains do well in dealing with classification and identification-related tasks. For image classification and patterns recognition, feature extraction(FE) is one of the most core technologies[10]. A lot of researchers have attempted to build image FE systems which are comparable to or better than human brains. Deep learning is a new-fashioned artificial intelligence. The deep learning based models typically are built with multi-layers(greater than three layers). Through this multi-layers structure, deep learning models can express more complex data, such as images, text, and audio as input. A class of models is involved in deep learning method, including DBN[16], SAE[17] and CNN.

Inspired by neuroscience, convolution neural network (CNN) is one special type of deep learning model. In the field of image classification and recognition, CNN is considered as a more outstanding deep learning model, and it can have good performance in HSI feature extraction[15]. There are two unique aspects, local connections, and shared weights, make CNN architecture different from other learning patterns. Figure 2.6 shows the shared weights in CNN.

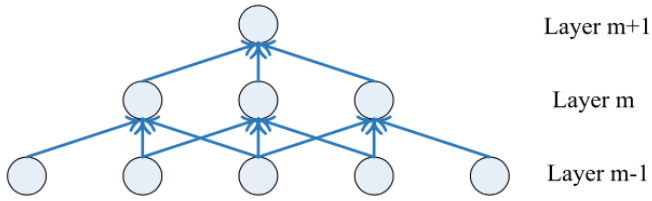

Fig. 1. Local connections in the architecture of the CNN.

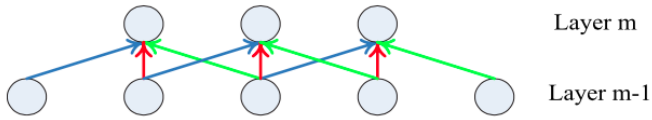

Fig. 2. Shared weights in the architecture of the CNN.

FIGURE 2.6: Shared weighs in CNN

According to Y. Chen (2016)[15], Local connections (illustrated in Figure 2.6-Fig. 1) mean that every three adjacent neurons on (m-1) th layer are connected to neurons on upper m th layer. Shared weights (illustrated in Figure 2.6-Fig. 2) means that same colour indicating the same weight. These features can help CNN architecture in image classification and extraction to have better generalisation ability. A general and simple CNN model consists of a convolution layer and a pooling layer. Deep CNN is contracted in a stacked manner by multiple layers of convolution and pooling.

Convolution layer:

$$v_{ij}^{x} = g\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1} w_{ijm}^{p} v_{(i-1)m}^{x+p}\right) \tag{1}$$

$$g(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2}$$

FIGURE 2.7: Convolution layer formula

The above Figure 2.7 formula shows a neuron v_{ij}^{x} at position x of the j th feature map in the i th layer; m index that connections between feature maps in the $(i-1)$ th layer and current map; w_{ijm}^{p} is the weight between p position and m feature map; p_i is the width of the kernel toward the spectral dimension; b_{ij} is the bias of j th feature map in the i th layer.

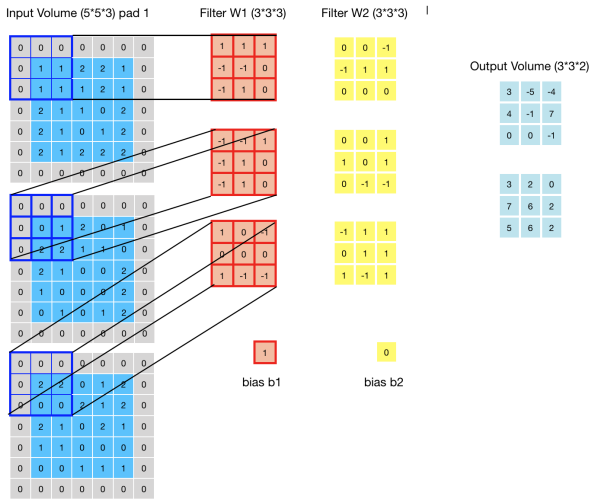

FIGURE 2.8: Brief example of convolutional layer

Figrue 2.8 shows a simple example of convolutional layer. Its input volume is (5*5*3), which is expened to 7*7*3 with zero value padding. The main part of this convolutional layer are 2 filter. The filters are in size of 3*3*3, with 2 biases. Inadditional stride of this convolutional layer is 2 , which defines filters move 2 pixels every time. By this way, the output has the size of 3*3*2.

Pooling layer:

$$a_j = \max_{N\times 1}\left(a_i^{n\times 1} u(n,1)\right) \qquad (3)$$

FIGURE 2.9: Pooling layer formula

Pooling layer provides invariance by reducing feature map resolution. The above Figure 2.9 formula shows the most common max pooling, is $u(n,1)$ the window function to the patch of the convolution layer; is the maximum a_j in the neighbourhood.

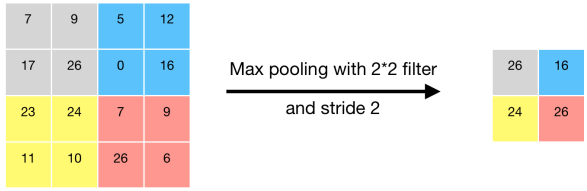

FIGURE 2.10: Brief example of pooling layer

Figrue 2.10 shows a simple example of pooling layer. Max pooling is one of the most common used pooling method. In this case it select the maximum value in every 2*2 matrix.

In addition, all layers in CNN models are trained by the back-propagation algorithm, which calculates the error contribution of each neuron after processing a batch of data. Back-propagation algorithm can calculate the gradient descent optimisation algorithm and distribute the error at output back through the model layers[18]. Since known and expected input of each value is required, the back-propagation algorithm is categorised as a supervised learning method.

To reduce or eliminate negative effects of the curse of dimensionality, feature extraction (FE) is deemed to be an effective method[15]. In the spectral representation, due to the high spectral correlation (neighbouring spectral data can represent similar materials), discriminating materials usually utilise this fact which group in this spectral space. On the other hand, in the spectral representation, due to the high spatial correlation (adjacent pixels can be most likely identified as similar or exactly same materials), discriminating materials can also apply this fact which group in spacial space[6]. The

methodology of feature extraction is based on these representations, which have been widely used in dimensionality reduction. The most commonly used methods are to physically select the most spectral information bands or use principal component analysis (PCA) and minimum noise fraction (MNF) transformation[8].

To avoid machine learning tends to overfitting, some additional techniques are considered to be necessary (e.g. cross-validation, regularisation[15], early stopping[19], pruning, Bayesian priors on parameters, model comparison, or dropout[20]). The main technical basis for these technologies is either (1) a set of clear mechanisms for penalising models that are too complex and potentially overfitting, or (2) a set of designed data not used for training, which is assumed to simulate the typical test data, are used for testing the generalisation ability of the model.

To address such problems that the spectral characteristics of the observed object are affected by the unknown atmospheric scattering, deep architecture models are considered as a helpful choice. Besides the interactions between atmospheric and lights, other factors such as undesired scattering from other objects and intraclass viability can increase difficulty level for effective hyperspectral data feature extracting. Inherently, in view of complex light scattering mechanisms of objects, hyperspectral data is nonlinear[8][7], which makes those methods originally designed for linear transformation no longer suitable. On the other hand, manifold learning is attempting to find the intrinsic structure of nonlinear data, which could have positive improvement effects to hyperspectral images feature extraction[21], namely the nonlinear data can be processed and presented by kernel-based algorithm. Kernel-based algorithm provides a possibility to convert linear problems to non-linear problems by mapping the raw data into higher dimensions. The advantage of deep architecture is known that it can potentially lead to more abstract features at high levels than other models. This ability is generally robust and invariant[15].

2.4 The Development of HSI Classification

In the development process of hyperspectral image development, initially, the vast majority of researchers put the focus on feature extraction by analysing spectral information. These methods include principal component analysis (PCA)[22], independent component

analysis (ICA)[23], linear discriminant analysis[24]. These methods use linear analysis to transform the input data and attempt to exploit potential features. However, considering the effects of atmospheric and natural objects in hyperspectral imaging (the spectral information of HSI is nonlinear), these linear analytical methods perform not outstanding.

Since 2000, the manifold method has been introduced into the study of HSI. The manifold method is for non-linear data, trying to dig out the intrinsic structure of nonlinear data, which is of great benefit to feature extraction and further improvement to the accuracy of classification. In other words, kernel-based algorithm is used for data representation, so that the nonlinear data can be addressed. In the higher latitude Hilbert space, the kernel-based method establishes a mapping of the raw input data, in which the nonlinear problem can be dealt with in a linear way[15].

Compared to the spectral data analysis, incorporating HSI spatial information into HSI analysis can help to achieve better FE performance (high spatial and spectral correlation). In recent years, the increasingly advanced hyperspectral imaging technology has the ability to obtain high spatial resolution images[7]. High-precision spatial information has had a greater impact on the performance of the spectral space feature extraction method in the classification[25]. A proven example is a study did by M. Fauvel (2008), who introduced a spacial method into his research based on the fusion of morphological operators and support vector machine (SVM). Then the classification accuracy of his model greatly increased. However, most of the current methods of hyperspectral images research are developed with only one-layer processing structure, which does not meet the needs of higher feature extraction capability. In the commonly used feature extraction classification methods, PCA and ICA can be regarded as single-layer methods; linear SVMs and logistic regression (LR) can be regarded as single-layer classifiers; decision tree can be considered as two-layers methods.

Recently, the deep learning method has been widely considered as an effective FE method implementation. By extracting the potential abstract feature on higher level, it can effectively solve the deformation caused by the scattering from atmosphere and other objects[15]. Some researchers have already done some work on deep models. In 2004, Y. Chen published a deep learning method named stacked autoencoder(SAE)[26]. Then in 2015, he proposed another deeper method called entitled deep belief network (DBN)[27].

Because the deep models have a strong ability to extract features, their classification accuracy is relatively high. However, due to the deep models require a large amount of training data, and the available HSI data is indeed limited, so there is still a great room for improvement.

2.5 The Significance and Purpose of This Research

Considering the application of hyperspectral images in various areas and the important role it plays in them, any improvement in performance in terms of hyperspectral images classification and identification can produce significant technical and economic value. In the past, some shortcomings of classification models are obvious. For example, some models cannot be applied to the multi-information channels of hyperspectral image, some models are lack of the ability to analyse nonlinear data, or some models are obstructed by a limited amount of training data. And all these problems in the CNN deep model are considered can be addressed.

Therefore, the purpose of this study is to establish a deep learning model based on CNN to deal with high-dimensional data of hyperspectral image in an efficient FE way, and this model needs to overcome the problem of high-dimensional data, the processing problem of nonlinear data, and rare labeled training data.

Chapter 3

Methods

3.1 Introduction

This chapter will illustrate the design of the program from the aspects of the development environment, data set selection, neural network model construction and data processing procedures. They will well reflect the design ideas and advantages of the application of deep neural network on hyperspectral image classification in this thesis.

3.2 Development environment

3.2.1 Python

For the neural network model that implements machine learning, several commonly used programming languages are Python, R, MATLAB, Ruby, and C++. Considering the ease of use of programming languages, learning costs, language performance, and the number of third-party libraries This project decided to use Python as a programming language for development. Python is an interpreted language, which greatly facilitates the process of program writing. Due to its dynamic type system and garbage collection features, it can greatly improve the efficiency of machine learning, which requires extensive prototyping and iterative research. Python also has a large number of high-quality third-party libraries, such as NumPy, SciPy for math operations, MatplotLib and SeaBorn for visualization, and TensorFlow, Theano for machine learning. This can

save development time and keep the focus of research on the structural design of the model.

3.2.2 Tensorflow

Considering reducing the amount of code lines and development complexity in the process of neural network development, this project chose to use an external open source software library. TensorFlow is arguably one of the best libraries in a neural network. TensorFlow is a Python external structure package developed by Google. It is also an open source software library that uses data flow diagrams for numerical computation. Its task is to train deep neural networks. By using TensorFlow, I can quickly Familiar with neural networks, greatly reduce the cost of development and development difficulties of deep learning. TensorFlow provides a Python API(Application programming interface) for compatibility with the use of the Python language, which can use an 'import' field to refer to the library. In addition, the open source nature of TensorFlow allows everyone to use and maintain it and consolidate it. It enables it to be updated quickly and its functionality can be improved.

3.2.3 Keras

Keras is an open source neural network library written for Python that performs very well in terms of modularity and scalability. The use of Keras can make the construction of a neural network model simple and fast, because he defines many layers in the neural network model, and has very detailed documentation to explain the specific usage. In addition, it provides the tf.keras API, which is well compatible with Tensorflow. Therefore, in this project, Keras will be used in conjunction with Tensorflow.

3.3 Data set introduction and processing

A good training set plays an important role in obtaining efficient neural networks. With the problem that the collection process of labeled training data is accompanied by expensive ground campaigns, this project has to select one from a limited number of public available hyperspectral image training sets. Therefore, this project selected a dataset

from the Purdue University Research Repository (PURR)[28]. This scene was acquired by the AVIRIS(Airborne Visible/Infrared Imaging Spectrometer) sensor at the Indian pine test site in northwestern Indiana and represents a 2 mile×2 mile area with 20 meters spatial resolution. The Hyperspectral image is 145×145 pixels and contains 220 spectral reflectance bands with wavelengths ranging from 0.4 to 2.5×10^{-6} meters.

In this scene, one-third of the vegetation is crops (mainly corn and soybean), and the remaining two-thirds are forests and shrubs. This Hyperspectral image was taken in June and the crop is in the early stages of plant growth. Therefore, the plant coverage rate is below 5 percent. In the upper part of the scene, there are two two-way highways. In the middle, there is a railway across. In addition, there are some houses, infrastructures and roads are scattered and distributed in the scene(shown in Figure 3.1). Although there are spectral information variations caused by occlusion between objects, the objects in the scene basically can be divided into 16 classes. The training data set classifies pixels as following Table 3.1:

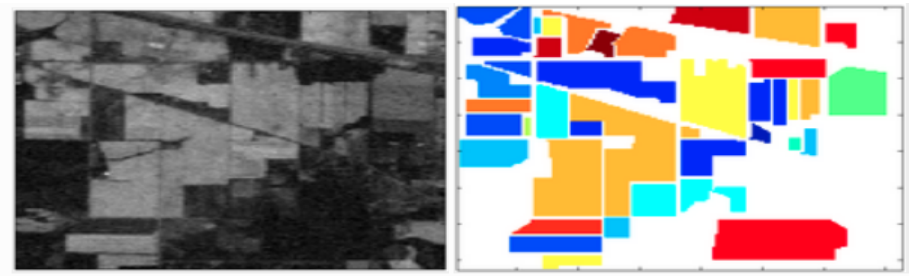

FIGURE 3.1: Aerial maps and groundtruth classification maps of Indian Pines

3.3.1 Feature extraction

From the dataset obtained from the publicly available open project, a hyperspectral image and an image of the actual ground classification of each pixel are provided. This thesis implements the extraction on the hyperspectral image and the method of training the data with the labeled data set[28].

If only one pixel is selected eevey time for feature extraction, and all its spectral channel information is extracted. The specific operation is shown in Figure 3.2 and Figure 3.3. In this method example, there is an 11*11 pixel image with 7 spectral channels is performed by the single-pixel feature extraction. On the other hand, a groundtruth classification

#	Class	Samples
1	Alfalfa	46
2	Corn-notill	1428
3	Corn-mintill	830
4	Corn	237
5	Grass-pasture	483
6	Grass-trees	730
7	Grass-pasture-mowed	28
8	Hay-windrowed	478
9	Oats	20
10	Soybean-notill	972
11	Soybean-mintill	2455
12	Soybean-clean	593
13	Wheat	205
14	Woods	1265
15	Buildings-Grass-Trees-Drives	386
16	Stone-Steel-Towers	93

TABLE 3.1: Classes in Inidan Pines data set

image indicates that all pixels can be classified into three classes (Class 1, 2, 3) and a none-classification class. After extraction, a data group with the size of 1*1, which has 7 spectral channels information, is obtained. By comparing the classification situation of the same position in the labeled dataset, we can know that this single-pixel data belongs to class 2. At the same time, any pixels that cannot be classified will be ignored. After traversing all positions in the image, we know that 36 sets of data belong to class 1, 36 sets of data belong to class 2, 30 sets of data belong to class 3, and 19 sets of data cannot be classified into class 1, 2 or 3. These data groups will then be further processed as input to the convolutional neural network model.

In this thesis, another feature extraction method called Neighbourhood Pixel Extraction are applied[29]. That is, not only a single pixel and its spectral information is processed, but also a certain range of adjacent pixels around the selected pixel. In Figure 3.4 and Figure 3.5, the same multi-channel image with the size of 11*11*7 is extracted using this method. In this example, all the pixels in the 3*3 range around the selected pixels and their spectral channels information is extracted to output a data set of size 3*3*7.

During the traversal extraction process, some of the points at the edge of the image obviously cannot directly use the neighboring pixel extraction method in which the selected pixel is centered. The processing method for the edge pixels will be described later.

Compared to the single pixel extraction method, the neighbouring pixels are more able to retain the spatial information of the hyperspectral image. In addition to a large amount of spectral information, the surface objects represented by each pixel in the hyperspectral image also have spatial continuity (Such as minerals are always concentrated in a certain area, and plants always appear in certain areas.) By combining these two kinds of information, the advantages of hyperspectral images in spectral information and spatial resolution can be fully utilized.

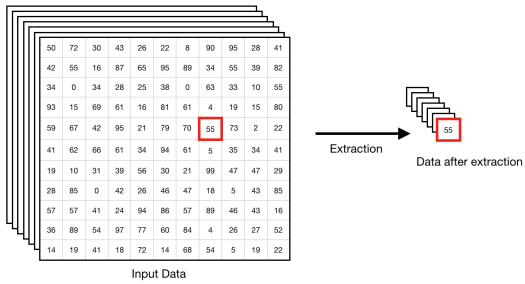

FIGURE 3.2: Single pixel feature extraction

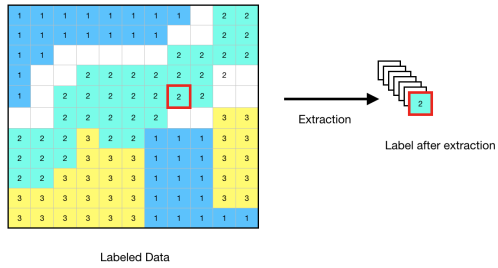

FIGURE 3.3: Labeled data after extracting

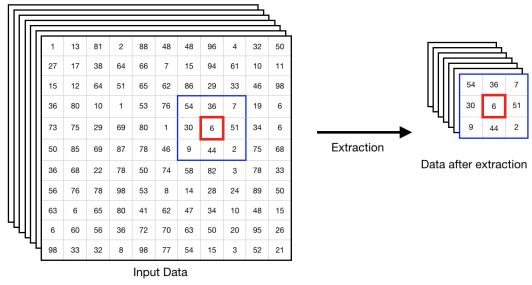

FIGURE 3.4: Neighbouring pixels feature extraction

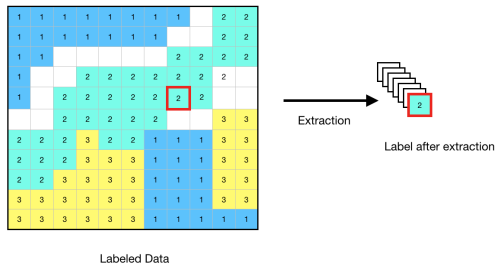

FIGURE 3.5: Labeled data after extracting

3.3.2 Edge problem in feature extraction

When the traversed patch window moves to the edge of the image for feature extraction, the problem of the edge extracting needs to be solved. The edge of the patch has exceeded the edge of the actual image, some pixels in the patch cannot be copied from the image. Currently, there are two ways to solve this problem: 1)Ignore the pixels near the edge[15]. 2)Padding in the remaining patch sections with the selected central pixel(As my schoolmate Zhiwang Zhang's thesis on 2017[30]).

The first method brings the problem that a large amount of pixel information is discarded, which leads to a decrease in the learning effect of the neural network and a decrease in the classification accuracy. In addition, the problem of not enough available training data of hyperspectral image will be more serious. This method of extracting with discarding pixels on edge is shown in Figure 3.6. We can see that the green pixels represent the pixels that can be extracted by the patch, and other parts are discarded. In the figure, an 11*11 image, if a patch with a size of 5 is used for extraction, 40.5% of the pixels are available, about 59.5% of pixels were discarded.

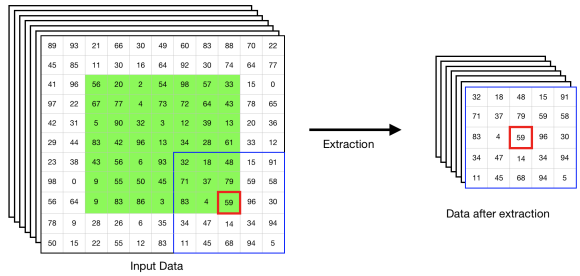

FIGURE 3.6: Extracting with discarding pixels on edge

The second method uses the selected center pixel to fill in the non-coinciding parts between patch and image. This method partially solves the edge extraction problem and obtains a patch containing edge pixels, but the rationality of this arbitrary filling worth to be reconsidered. The information of the filled patch part is consistent with the selected pixel of the center, but it is not necessarily spatially and spectrally related to the real image edge. The actual situation is that, the actual surface objects outside of the edges of image highly likely to be identical to the objects represented by the image edge pixels. By this method, the spatial information contained in patchs could be seen as wrong, which will affect the training of neural networks. This method of extracting with padding the centred pixel is shown in Figure 3.7.

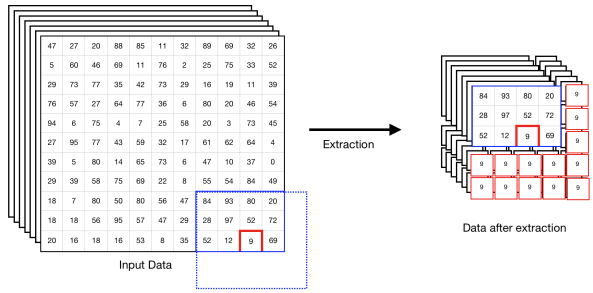

FIGURE 3.7: Extracting with padding the centred pixel

The thesis introduces a new method to solve the edge processing problem. It is based on patch pixel filling as well, but uses the image edge pixels to fill. The spectral information of the pixels at the edges of the image will extend parallel to or vertically (copying) into the patch. If patch appears in the four corners of the image, the four pixels in the corner of the image will be copied along with the diagonal direction. The method is shown in Figure 3.8.

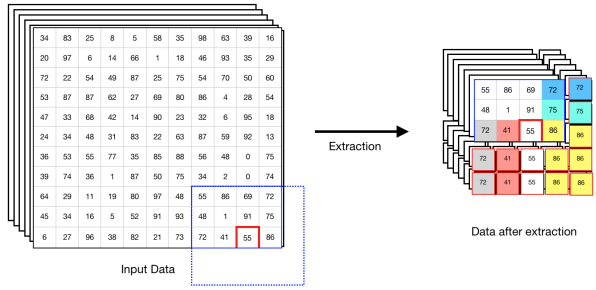

FIGURE 3.8: Extracting with padding the pixels on image edge

3.3.3 Data set splitting

Because of the limited number of training data sets, this 145×145 pixel hyperspectral image needs to be fully utilized. By setting a patch size of 11 pixels, the Indian pines scene is extracted as spatially adjacent smaller images. The length and width of the pixel are 11 pixels and the depth is 220 (that is, 220 spectral channel data representing different wavelengths). Every patche will be labeled into one specific class, while ignoring all patches with unknown land-cover types for the central pixel. It embodies the design of incorporating spatial information of pixels into a neural network for analysis. Then, based on the ratio of 20%, 40%, and 40%, make a split for training, validation and testing data set for classes. With this method of processing data set, the training, validation, and testing data sets obtained occupy 2.18×10^9 bytes (2.18GB) of space on disk, while the original hyperspectral image of Indian pines occupies only6.29×10^6 bytes (6.3MB). of storage space.

3.3.4 Zero mean Normalization

Thesis performs zero-mean normalization on the raw data in the data set[31]. The deep neural network involves many stacking layers, and the updating of parameters on each layer leads to the change of the input data distribution on the upper layer. The distribution of data will change very dramatically, which is considered to have a bad influence on the performance of deep neural network. Therefore, in order to obtain a higher performance model, the input data needs to be handled with care, and the distribution of the data needs to be normalized.

In the normalization method, the data is processed in two steps, re-scale parameter, and the re-shift parameter. In the re-scale parameter phase, all spectral data is scaled into the range of 0 to 1 (the original minimum value is set to 0, the original maximum value is set to 1, and then all data is remapped to this range in the original ratio.) In fact, in the Indian Pines data set, the spectral data range is already in the 0 to 1 range.

In the re-shift parameter phase, all data is slid vertically to the x-axis to make their mean value is 0 (shown in Figure 3.9 and 3.10). The formula to calculate the new merit formula as follows:

$$\mathrm{V}_{new} = \frac{(V - U_x)}{S_x}$$

FIGURE 3.9: Formula of zero mean normalization

V_new is the new value, U_x is the mean value, S_x is the standard deviation of all data. Both of them can be calculated from data set. After zero-mean normalizing, all data in same band will have a mean value of 0.

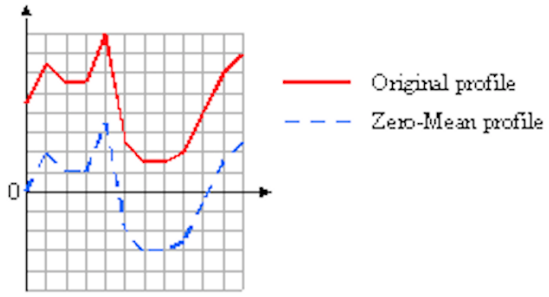

FIGURE 3.10: Re-shift parameter phase in zero mean normalization

3.4 Structure

3.4.1 Layers

By processing the data set before, we select the K neighborhood of the current pixel to form the input of the deep CNN model. A complete CNN consists of a convolutional layer and a pooling layer. The deep CNN used in this project is constructed by stacking

multiple convolutional and pooling layers. Because the size of the image is not large and the resolution is not high, this neural network selects a 3×3 filter for convolution and a 2×2 filter for the pooling. The image passes through the stacked multi-layer convolution and pooling and is converted into the form of feature maps. These features contain spatial and spectral information of all input images. Then, with flatten, dense and dropout operations, these features are processed by the fully connected layer and finally classified into 16 classes. This project uses the LR classifier structure with softmax as the output layer activation to obtain the activation of all output elements whose sum is 1. Through this method, all outputs appear as conditional probabilities. The size of the output of the LR is same to the number of classification classes (the number of classification classes is 16 according to the training data), and the input layer is determined by the size of the output layer of the upper deep CNN structure.

The following Table 3.2 shows the structure of this deep CNN model, the output shape of each layer (the input shape is the output shape of the previous layer), and the number of parameters.

Layer (type)	Output Shape	Number of Param
conv2d_1 (Conv2D), * Input Shape = (220,11,11)	(128, 11, 11)	253568
activation_1 (Activation)	(128, 11, 11)	0
conv2d_2 (Conv2D)	(128, 11, 11)	147584
activation_2 (Activation)	(128, 11, 11)	0
max_pooling2d_1(MaxPooling)	(128, 5, 5)	0
dropout_1 (Dropout)	(128, 5, 5)	0
conv2d_3 (Conv2D)	(64, 5, 5)	73792
activation_3 (Activation)	(64, 5, 5)	0
conv2d_4 (Conv2D)	(64, 5, 5)	36928
activation_4 (Activation)	(64, 5, 5)	0
max_pooling2d_2 (MaxPooling2)	(64, 2, 2)	0
dropout_2 (Dropout)	(64, 2, 2)	0
flatten_1 (Flatten)	(256)	0
dense_1 (Dense)	(128)	32896
activation_5 (Activation)	(128)	0
dropout_3 (Dropout)	(128)	0
dense_2 (Dense)	(16)	2064
activation_6 (Activation)	(16)	0
Total params: 546,832 Trainable params: 546,832 Non-trainable params: 0		

TABLE 3.2: Architecture of the neural network in this thesis

Among these structures, layers are carefully designed and have significant effects on the learning performance of the model. The following paragraphs will list these designs and introduce their principles.

3.4.2 Relu and dropout

The activation function introduces nonlinear factors to the neuron so that the neural network can fit any nonlinear function well, so that the neural network can be applied to many nonlinear models. There are many different activation functions available to apply, including Sigmoid, Tanh, and Softmax. In this neural network model, an activation function called ReLu is in use[**?**]. It will return 0 for all negative inputs. All positive inputs are returned directly to neurons. Many previous studies have shown that the application of ReLu in CNN can significantly improve performance [35]. The ReLU function has a low computational complexity (no division is involved). The output of some neurons is 0, which results in the sparsity of the network and reduces the interdependence of parameters, thus alleviating the occurrence of overfitting problems. Figure 3.11 shows the ReLu activation function in cartesian coordinate system.

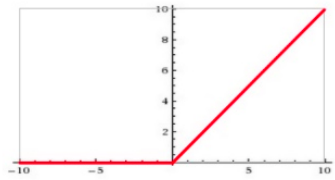

FIGURE 3.11: ReLU activation function

Another method to reduce the possibility of overfitting is Dropout. According to Hinton's 2010 description of the Dropout method, neural network elements are temporarily dropped from the network during the training of deep learning networks. Based on a certain probability, it sets the output of some hidden neurons to zero, which means that the discarded neurons do not work in the forward propagation and they do not participate in the backpropagation process. This model uses the Dropout method multiple times. The parameters are set to 0.5, which means that each neuron has a 50 percent

probability of being removed, which in this way can make each neuron's training independent of the others, and also makes the mutual influence between features weakened. Figure 3.12 shows a simple neural network and the same neural network with Dropout

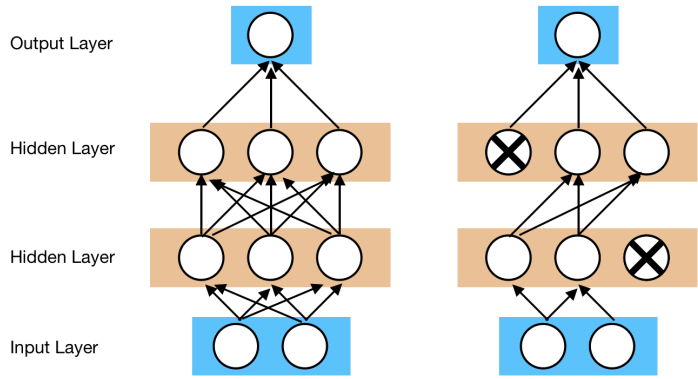

FIGURE 3.12: Normal neural network and neural network with Dropout

3.4.3 L2 regularization

Due to the limited amount of available training samples in the classification of HSI using deep neural networks, overfitting problems often occur. Overfitting refers to using excessive parameters when the model tries to fit a data set. Compared to the total amount of data available, an overfitting model is sufficiently complicated to fit the data perfectly. The overfitting problem means that the neural network performs extremely well when it is running on the training data set, but it runs poorly on the test data, which leads to low classification accuracy. To avoid this situation, one technique called L2 regularization is applied to this deep CNN model. The idea of L2 regularization is to add an additional regularization term to the cost function. The L2 regularization method reduces the sum of the squares of the following equations to reduce cost function. LR is used as a method for adjusting weights and deviations in backward propagation. In this paper, the L2 regularization method is optional. Figure 3.13 shows the formula of L2 regularization.

$$c = c_0 + \frac{\lambda}{2m} \sum_{j=1}^{N} w_j^2$$

FIGURE 3.13: L2 regularization formula

C_0 represents the original cost function, and the following item is the L2 regularization term. m is the size of mini-batch. N represents the number of weights, and the parameter λ can be change according empiric.

Chapter 4

Results

4.1 Training

Throughout the deep CNN structure, the back-propagation algorithm is used for training model, from the convolutional layer to the pooling layer. This method calculates the gradient of the cost function for all the weights in the network. This gradient is fed back to the optimization method to update the weights and minimize the cost function. Backpropagation requires to know the expected output for each input value to calculate the cost function gradient. Therefore, it is considered as a supervised learning method. With the powerful ability of constructing the neural network model, Keras library can easily set the parameters required for back-propagation training, including the learning rate, momentum (parameter that accelerates SGD(Stochastic gradient descent) in the relevant direction and dampens oscillations) and learning rate decay over each update(It can make the rate of gradient descent is gradually reduced. The Cost function keep steady within a very small range to obtain a satisfactory value).

4.2 Result of testing

A total of 300 epochs were processed in the model. During each epoch, all training data experiences a forward and back propagation and all parameters are updated once time. We can derive the trend of loss and classification accuracy in 300 epochs(shown in Figure 4.1 and 4.2). In the figure, the classification accuracy rapidly increases, reaching

more than 90% after the 30 th epoch, and then gradually approaches 1.0. During the 300 epoch periods, the highest value reached 0.9943. At the same time, the value of loss gradually approaches to 0.This indicates that this model has relatively good performance on classification.

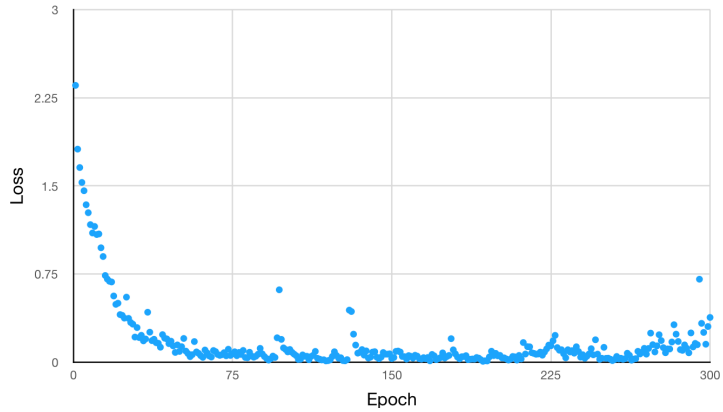

FIGURE 4.1: Result: Loss versus epochs

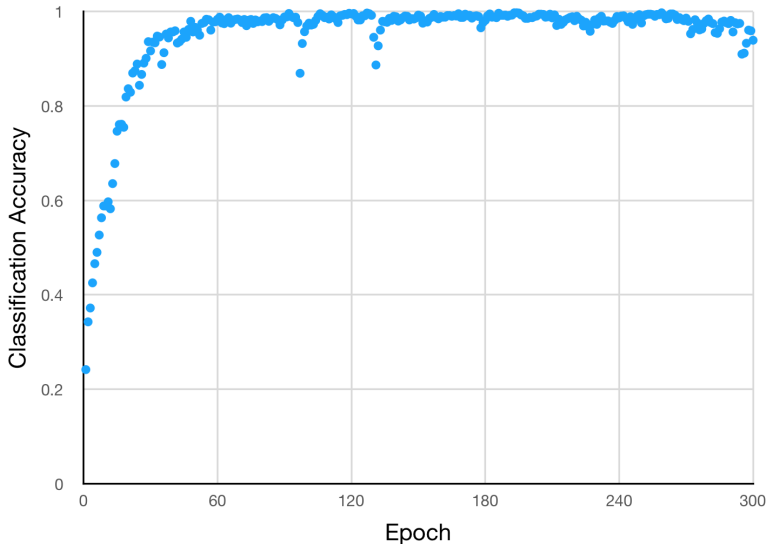

FIGURE 4.2: Result: Classification accuracy versus epochs

The classification performance of this network can also be represented by a fusion matrix as shown in Table 4.1. Table 4.1 shows the result after 300 epochs. The Confusion matrix is a visual tool for supervised learning. Each column of the matrix represents the prediction of a class, and each row represents the actual situation of a class. It

can show the confusion level between different classes. In this Table 4.1 fusion matrix, non-zero data is mainly concentrated near the main diagonal, which means that most of the predictions for the samples are consistent with the actual situation. At the same time, there are also some non-zero data appearing at other locations far away from main diagonal, which indicates the prediction is wrong. However, compared to the non-zero data on the main diagonal, the values of these data are low, which represents the error rate of classification is relatively low as well.

	Predicted															
Actual	35	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	0	1011	24	2	6	0	0	0	0	1	43	54	0	0	1	0
	0	4	655	0	0	0	0	0	0	0	0	5	0	0	0	0
	0	2	57	115	1	0	0	0	0	0	0	13	0	0	0	0
	0	0	13	0	364	0	0	2	0	1	0	0	0	6	0	0
	0	1	0	0	0	583	0	0		0	0	0	0	0	0	0
	0	0	8	0	0	0	9	0	0	5	0	0	0	0	0	0
	0	0	2	0	1	0	0	371	0	2	0	0	0	1	5	0
	0	0	4	0	0	0	0	0	12	0	0	0	0	0	0	0
	0	0	9	0	0	0	0	0	0	745	11	10	0	1	0	0
	0	30	15	0	1	0	0	0	0	10	1903	5	0	0	0	0
	0	0	4	0	0	0	0	0	0	0	0	467	0	0	0	3
	0	0	0	0	0	0	0	0	0	1	0	0	163	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0	1011	0	0
	0	0	0	0	2	5	0	1	1	7	0	0	0	3	289	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	74

TABLE 4.1: Result: Confusion matrix

According to Table 4.1 confusion matrix, the classification prediction accuracy of this neural network can be calculated for each class adn all classes. For each class, its accuracy is calculated as:

$$Acc_i = \frac{P_i^{True}}{\sum P_i}$$

For class i, its classification accuracy Acc_i is related to the true prediction P_i^{True} and total prediction $\sum P_i$ in this class.

For all classes, the total accuracy calculation formula is:

$$Acc = \frac{\sum P^{True}}{\sum P}$$

Which shows the total accuracy Acc is related to the sum of all true predictions in all classes $\sum P^{True}$ and all predictions $\sum P$.

	Accuracy of each class
1	0.9722222222222222
2	0.8852889667250438
3	0.9864457831325302
4	0.6117021276595744
5	0.9430051813471503
6	0.9982876712328768
7	0.4090909090909091
8	0.9712041884816754
9	0.75
10	0.9600515463917526
11	0.9689409368635438
12	0.9852320675105485
13	0.9939024390243902
14	0.9990118577075099
15	0.9383116883116883
16	1.0
Overall Accuracy	0.9530029297

TABLE 4.2: Accuracy on classes

The calculation results in Table 4.2 show that after training with 300 epochs, although the classification results for each class are not the same, this neural network model shows quite good ability for the classification of all classes.

	Predicted															
Actual	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1129	0	0	0	0	0	0	1	0	12	0	0	0	0	0
	0	0	658	0	0	0	0	0	1	0	0	5	0	0	0	0
	0	0	0	188	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	384	0	0	0	0	0	0	0	0	2	0	0
	0	0	0	0	0	584	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	21	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	382	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0
	0	4	0	0	0	0	0	0	0	768	0	2	0	0	2	0
	0	1	0	0	0	0	0	0	0	1	1955	0	0	6	1	0
	0	0	0	0	0	0	0	0	0	2	0	470	0	0	0	2
	0	0	0	0	0	0	0	0	0	0	0	0	164	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	1011	1	0
	0	0	0	0	0	2	0	0	0	0	0	0	0	0	306	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	74

TABLE 4.3: Confusion matrix of the best classifier in proposed CNN

The Figure 4.3 and 4.4 show the best classifier of the proposed CNN, which overall accuracy reached 99.43%. It can be seen that in this classifier, both the overall accuracy and the classification accuracy of each class have reached a very high level.

	Accuracy of each class
1	1.000
2	0.9886164623
3	0.9909638554
4	1.000
5	0.9948186528
6	1.000
7	0.9545454545
8	1.000
9	1.000
10	0.9896907216
11	0.9954175153
12	0.9915611814
13	1.000
14	0.9990118577
15	0.9935064935
16	1.000
Total Accuracy	0.9943847656

TABLE 4.4: Accuracy on classes of the best classifier in proposed CNN

4.3 Compare with other people's work

Compare the best classifier of my proposed CNN structure with experiments data from other people's papers. According to Chen's 2016 paper[15], their two models, 3D-EMP-RBF-SVM and 3D-CNN-LR on the Indian Pines data set have overall classification accuracy(OA) values of 96.92±0.81 and 97.56±0.43 respectively. According to the paper of Lichao Mou in 2017[32], their two models, RNN-LSTM and RNN-GRU-PRetanh on the Indian Pines dataset, have overall classification accuracy(OA) values of 80.52 and 88.63, respectively. According to the paper of Xiaorui Ma in 2015[33], their three models, SOMP, MPM-LBP, and CDL-MLR on the Indian Pines data set, have overall classification accuracy(OA) values of 92.00, 95.12, and 98.26 respectively.

The specific comparison and classification accuracy of each class are shown in Table 4.5. It can be seen that both the overall accuracy and the average accuracy of all classes, the CNN model proposed in this thesis has certain advantages.

	This thesis	Chen's 2016 paper		Lichao Mou's 2017 paper		Xiaorui Mas 2015 paper		
Class	Proposed CNN	3D-EMP-RBF-SVM	3D-CNN-LR	RNN-LSTM	RNN-GRU-PRetanh	SOMP	MPM-LBP	CDL-MLR
1	100.0	95.94	100.0	46.03	70.59	99.90	100.0	100.0
2	98.86	91.39	96.34	61.73	70.28	99.97	99.92	96.65
3	99.10	92.53	99.49	86.96	81.52	92.77	96.20	99.95
4	100.0	100.0	100.0	87.02	90.16	95.76	98.64	98.98
5	99.48	98.03	99.91	86.66	91.97	99.49	98.64	98.78
6	100.0	92.35	99.75	97.49	96.13	99.32	99.97	99.27
7	95.45	95.93	100.0	59.69	84.75	100.0	99.94	100.0
8	100.0	96.35	100.0	64.89	59.64	86.61	88.64	96.62
9	100.0	100.0	100.0	60.46	86.17	98.81	99.58	99.92
10	98.97	87.92	98.72	98.77	99.38	95.21	92.27	98.57
11	99.54	92.23	95.52	75.32	84.97	97.53	96.54	96.78
12	99.16	96.85	99.47	71.82	77.58	99.95	98.27	98.37
13	100.0	99.23	100.0	91.11	95.56	97.73	96.47	98.58
14	99.90	96.85	99.55	79.49	84.62	95.00	97.97	98.87
15	99.35	89.28	99.54	90.91	90.91	71.31	88.83	95.64
16	100.0	96.24	99.34	100.0	100.0	99.94	99.28	99.48
OA	99.43	96.92	97.56	80.52	88.63	92.00	95.12	98.26
AA	99.36	95.07	99.23	78.65	85.26	95.58	96.94	98.72

TABLE 4.5: Compare with other people's work

4.4 Structural details and their performance

4.4.1 Patch size in preprocessing

In the processing process, the patch size determines the the mount of spatial and spatial information of adjacent data are extracted every time. Because the purpose of this extraction method is to obtain spectral and spatial consistency information within a pixel and its neighboring pixels, the patch value should theoretically be determined by the size of the ground object. For example, in order to classify the pixels representing buildings in the image, a patch window that can contain buildings of a general size and certain range of neighboring pixels is considered to be the most suitable. However, due to the training speed and the occupation of computer resources, the setting of the patch value is usually defined freely by the designer.

The thesis compares the CNN classification performance under three cases, where patch values are 7, 11, and 15. The specific data is shown in the figure below.

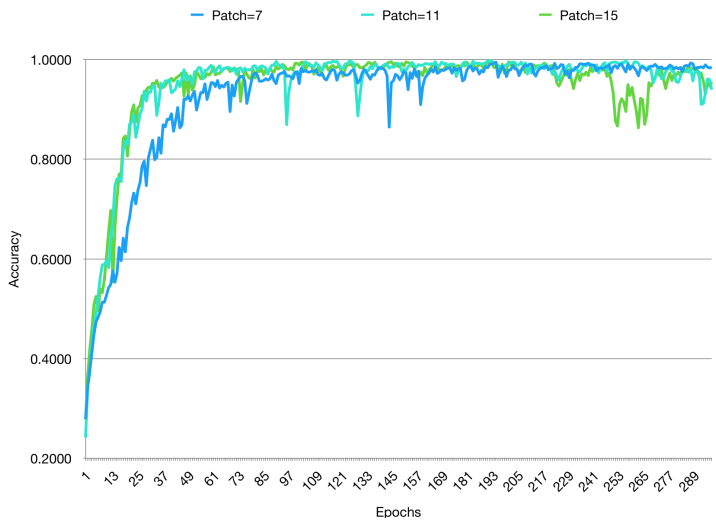

FIGURE 4.3: The input data generated by different patch sizes performs on CNN

In Figure 4.3, it can be observed that the classification accuracy variation pattern of the extracted data generated by different patch sizes on the CNN is basically the same. However, the larger the patch size indicates a faster-increasing accuracy change trend (ie, approaching a high classification accuracy in the fewer epochs period). In the figure, the line representing the patch size of 7 basically stopped the increase of accuracy near the 85th epoch, while the line representing the patch size of 11 and the line representing the patch size of 15 stopped rising near the 30th epoch.

However, the computer resources and training time of different epochs input data set need to be considered. After data set preprocessing, the original Indian Pines data set with a size of 6.3MB is extracted using the methods with patch sizes of 7, 11 and 15. They are extracted into training + verification + test data sets of 882.9MB, 2.18GB, and 4.05GB, respectively. Running on my laptop with a 2.4 GHz Intel Core i7 processor (approximately 32.08 GFLOPs), the time consumption for each epoch is about 6.2 seconds, 18 seconds, and 23 seconds. Taking into account the occupation of computing resources and the consumption of training time, the thesis selects a patch size of 7*7 pixels. The network model can not only obtain high classification accuracy but also reduce training and verification time.

4.4.2 Edge problem in feature extraction

Compared to the traditional approach of discarding edge pixels, the edge extraction method proposed by this thesis can retain more pixel space and spectral information. For any extraction method with N as patch size and hyperspectral image with H*W, $\frac{(H-N+1)(W-N+1)}{HW}$% Pixels were extracted, and $(1 - \frac{(H-N+1)(W-N+1)}{HW})$% of the pixels were abandoned. In the method described by Chen in the 2016 paper, they used a patch of size 27 to extract the Indian Pines data set (145*145), then 67.35% of the pixels Was extracted, and 32.65% of pixels were abandoned. If this edge processing method is used in the thesis with patch of size 11, 86.68% of the pixels Was extracted, and 13.31% of pixels were abandoned.

Considering that the distribution of each labeled class is different in the image data set, this method of discarding the edge data has different effects on each class. For example, the pixels representing class 15 and class 3 that are mostly distributed in the edges area of the image, then there will be only 69% and 83% of the pixels respectively will be retained to generate feature extraction by this way. However, considering some pixels are not classified into 16 classes, the figure is different from this simple expectation. The detail is given in the Table 4.6 below.

Class	Actual object	No Discarding	Discarding pixels in FE	Discarding rate
1	Alfalfa	46	46	0%
2	Corn-notill	1428	1428	0%
3	Corn-mintill	830	685	17.46%
4	Corn	237	221	6.75%
5	Grass-pasture	483	423	12.42%
6	Grass-trees	730	730	0%
7	Grass-pasture-mowed	28	28	0%
8	Hay-windrowed	478	478	0%
9	Oats	20	20	0%
10	Soybean-notill	972	924	4.93%
11	Soybean-mintill	2455	2350	4.28%
12	Soybean-clean	593	561	5.40%
13	Wheat	205	205	0%
14	Woods	1265	1265	0%
15	Buildings-Grass-Trees-Drives	386	265	31.34
16	Stone-Steel-Towers	93	93	0%
Total		10249	9712	5.24%

TABLE 4.6: Number of classes in Indian Pines when feature extracting

Compared to the method used by Zhiwang in the thesis in 2017, this thesis adopts the same patch size, but adopts different patch padding methods. My proposed method more emphasizes that the spectral and spatial information of the image edge pixels extends into the patch. In his thesis, using the input patchs generated by that method (padding patchs with centred pixel) as input, the overall classification accuracy of his neural network model reached 99.2%. Compared to my best classification accuracy of 99.43%, the overall classification ability of two neural network is very close.

4.4.3 Dropout method

To solve the problem of overfitting, the Dropout method is introduced into the thesis. The neural network unit was temporarily dropped from the network with a certain probability. In this section, the proposed neural network model (using Dropout technique) and the same CNN model without Dropout technique are tested. Their experimental results are shown in Figure 4.4 below.

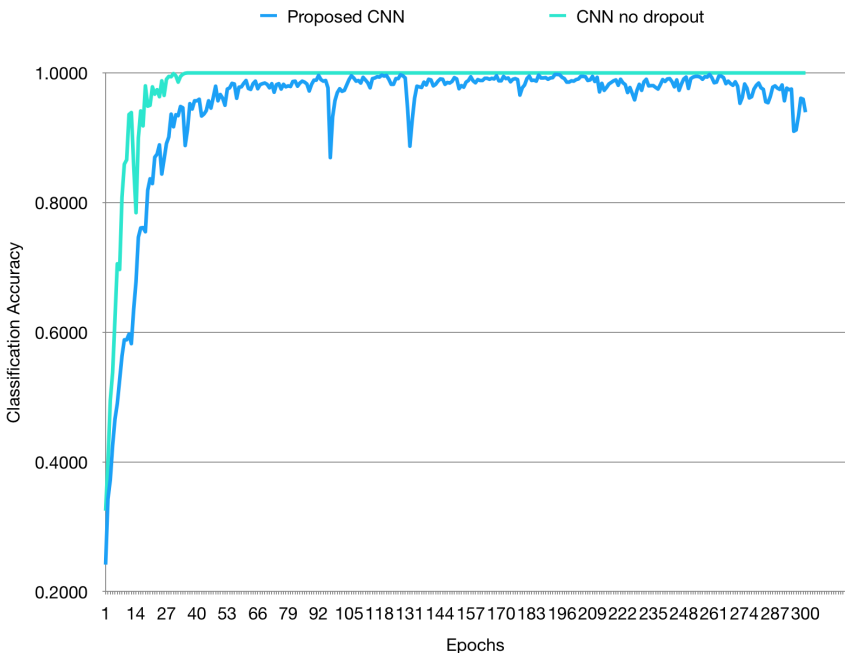

FIGURE 4.4: Proposed CNN without Dropout

It can be observed that this model without Dropout achieves a higher classification accuracy within a shorter epochs period. Even at the 36th epoch, the accuracy rate reaches 1.0, and this value is maintained continuously. The learning situation could be regarded as overfitting. At this time, the training accuracy is 100%, and testing accuracy is 97.80%. It can be seen that the Dropout method can prevent overfitting in the case that there is not enough labeled data available for hyperspectral images.

4.4.4 Learning rate decay

In this network, one problem that can be expected and solved is the fluctuation of classification performance, that is, the unfavorable and abnormal fluctuation that appears in the trend of loss and accuracy. In addition to the random fluctuations in the trend, some large fluctuations are considered to be related to the network's learning rate. In the case of using a fixed learning rate, the solution represented by the network will converge towards the optimal solution, but after many times of training, it will wander around, and never really converge. Therefore, the concrete manifestation of this phenomenon is the fluctuation that appears in classification performance.

To solve this problem, in this neural network model, we have adopted a method called learning rate decay, which is to set a floating point number for the learning rate decay over each update. Through this method of slowly reducing the learning rate, the network will gradually converge near the optimal solution with smaller learning rate, and produces smaller fluctuations in classification performance. In Figure 4.5 and 4.6, a same neural network model with fixed learning rate is trained and tested. Some larger fluctuations appear in the stable trends. The classification accuracy has drastically decreased, and the value of loss has risen sharply.

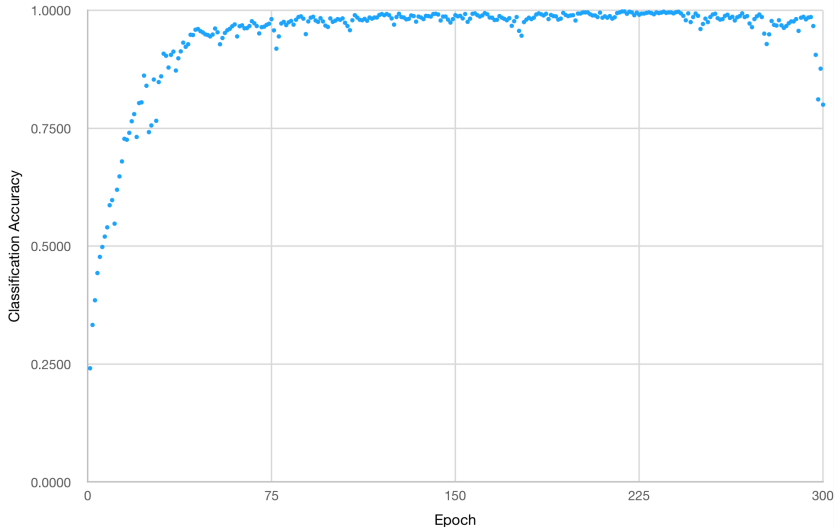

FIGURE 4.5: Loss changes without learning rate decay

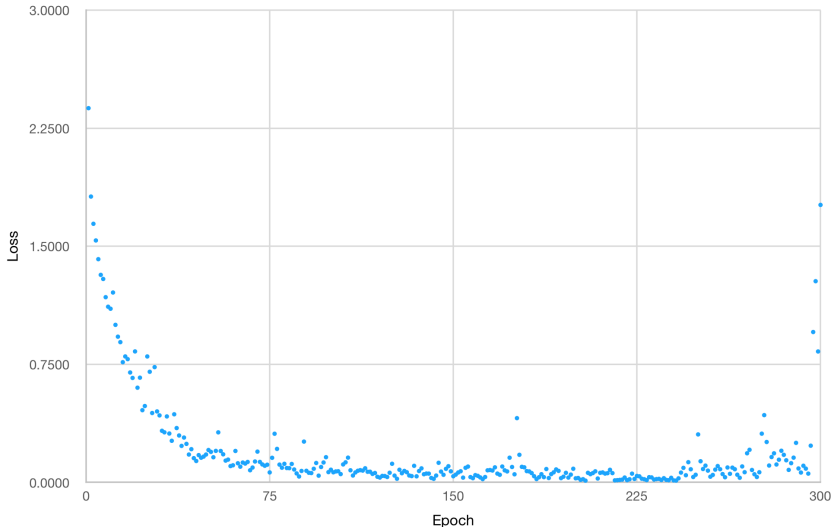

FIGURE 4.6: Classification accuracy changes without learning rate decay(larger fluctuations appearing)

Chapter 5

Discussion

This chapter will discuss this deep convolutional neural network model from various perspectives. Firstly, the result of testing will be evaluated. Then, the time and memory complexity of the program will be discussed. Lastly, this model will be compared with other models to obtain the directions for possible future improvement.

5.1 Results evaluation

The trend of classification accuracy and the value of loss in the 300 epochs are shown in Figure 4.1 and Figure 4.2. Since the model adopts random numbers to set initial parameters, we can see that after the 1st epoch, the initial value of classification accuracy is very low (0.2416). After that, the classification accuracy continued to increase and approached a relatively high value at a fairly rapid rate. After 30 epochs, the classification accuracy rises to about 90%. Then the effect of learning gradually slows down. Finally, near the 60 epochs, the classification accuracy almost stopped improving. In the reminding epochs, there are only a few random fluctuations. We can observe that in the 300 epochs period, the accuracy reaches up to highest value 99.76%, which means that this convolutional neural network model is quite good for training samples.

In addition, the value of loss shows a trend of variations that is different from the classification accuracy. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances. In the backward propagation, the value of loss will determine how much to change the parameters in network.

Similarly, due to the random initialization of the network, the loss value of shows a high level(2.3599) after the 1st epoch, which means that there is a large difference between the initial predictions results and the actual values. After 30 epochs, the loss value drops to about 0.2. Then the decline of the loss gradually slows down. Finally, loss nearly stops falling around 60th epoch. In the reminding epochs, there are only a few random fluctuations. It can be seen that the trend of loss value and classification accuracy are inversely related, and there is consistency on the horizontal axis (within the same epoch, the change of accuracy is consistent with the change of the loss value).

For the results, what we can expect is that with the number of epochs experienced by the network increases, the accuracy of the network will increase as well, and eventually even reach to 100%. However, in that case, the network may not be well applied to other data set. So the learning network at this case is meaningless. We call that network overfitting or overtraining.

5.2 Complexity evaluation

Looking back at previous studies and models of convolutional neural networks, many innovations and improvements are closely related to improving time and memory complexity. Therefore, analyzing the time and memory complexity of convolutional neural networks can help future improvements and make the model becomes more efficient.

5.2.1 Time complexity

Time complexity is a function that describes the running time of the algorithm. The time complexity can be called asymptotic, that is, it describes the output value when the input value approaches infinity. Its unit of measure is FLOPs (Floating-point operations per second). The time complexity determines the training/prediction time of the model. If the complexity is too high, it will lead to a lot of time spent on model training and predicting. It can neither quickly improve models, nor can it make rapid predictions. Time complexity for a single convolutional layer is shown in Figure 5.1:

M is the side length of the feature map generated from each convolution kernel; K is the length of each convolution kernel; C_{in} is the number of channels for each convolution

$$\textbf{Time} \sim O(M^2 \cdot K^2 \cdot C_{in} \cdot C_{out})$$

FIGURE 5.1: Time complexity of a single convolutional layer

kernel, that is, the number of input channels, or the number of output channels of the upper layer; C_{out} is the number of convolution kernels that the convolution layer has, that is, the number of output channels. It can be seen that the time complexity of each convolutional layer is determined by the output feature map area M^2, the convolution kernel area K^2, the input C_{in}, and the output channel number C_{out}.

Among them, the output feature map size M is determined by the four parameters of input matrix size X, convolution kernel size K, Padding, and Stride. The mathematical definition of M is shown in Figure 5.2

$$M = (X - K + 2 * Padding)/Stride + 1$$

FIGURE 5.2: Size of feature map

Thus, time complexity for the entire convolutional neural network is shown in Figure 5.3 below:

$$\textbf{Time} \sim O\left(\sum_{l=1}^{D} M_l^2 \cdot K_l^2 \cdot C_{l-1} \cdot C_l\right)$$

FIGURE 5.3: Time complexity of convolutional layers

D is the number of convolution layers this neural network has, which is the depth of the network; l is the l th convolutional layer of the neural network; C_l is the the number of output channel(C_{out}) of the l th convolutional layer in the neural network, that is, the number of convolution kernels in the layer; For the l th convolutional layer, the input channel number C_{in} is the output channel number of the $l-1$ convolutional layer.

5.2.2 Memory complexity

Time complexity is a function that describes the resource of memory space of the algorithm. Memory complexity includes the number of parameters of the model (the volume of the model itself) and the size of the feature map of the output of each layer (this will affect the memory consumption of the model during operation). Memory complexity determines the number of parameters of the model. Due to the limitations of the dimensional curse, the more parameters of the model, the greater the amount of data needed to train the model.

$$\textbf{Space} \sim O\left(\sum_{l=1}^{D} K_l^2 \cdot C_{l-1} \cdot C_l\right)$$

FIGURE 5.4: Memory complexity of neural network

Figure 5.4 shows the truth that the number of parameters(time complexity) of a neural network is only related to K (the size of the convolution kernel), C (the number of channels), and D (the depth of the neural network). and are independent of the size of the input data. The input data does not affect the memory complexity of the neural network.

5.3 Improvement

With the development of neural network technology, deep neural networks have developed many variants, such as CNN, RNN(Recurrent neural network), Resnet(Residential network), DenseNet(Densely Connected Convolutional Networks)[34]. These variants often incorporate many of unique structures, such as convolutional layers or LSTM(Long short-term memory) elements. These new neural network models are mainly designed to solve two problems that prevent further deepening of the neural network. They are overfitting (the optimization function falls into the local optimal solution) and the vanishing gradient problem.

As the network gets deeper and deeper, the vanishing gradient problem becomes more obvious. In the backpropagation process, the updated gradient cannot be effectively passed to the previous layer. Because of this, the parameters of the previous layers

cannot be changed, training and testing result get worse. A deep neural network model ResNet proposes a solution that adds a mechanism, which is called identity mapping. Identity mapping passes the current output directly to the next level of the network (without adding any extra parameters). It makes a shortcut for data to skip the current layer. This model greatly improves the effectiveness of the backpropagation algorithm. Another neural network called DenseNet can be seen as an improved version of this design. Its skip-connection not only connects the upper and lower layers but also directly realizes the cross-layer connection. The gradient obtained by each layer is the sum of gradients from all the previous layers.

In fact, regardless of using which kind of network, they are often mixed in practical applications. It is hard to tell which kind a network belongs to. We can expect that with the progress of deep learning research in the future, more flexible combinations and more network structures will be developed. Although these variants are diverse in variety and structure, the purpose of all model designs is to solve specific problems. In the future, the deep neural network model constructed in this thesis needs to focus on feature extraction of high dimensional data and neural network gradient problem.

Chapter 6

Conclusions

Hyperspectral image has great value in many fields, especially its application on ecological science, mineral industry, and military. Through deep neural network processing, the ground or underground targets on the hyperspectral image can be automatically identified and classified efficiently. Although there are still many unfavorable factors that affect the accuracy of the classification, including the low number of labeled hyperspectral image data sets, the mutual interference of spectral data from objects in the images, and the problems caused by the deep neural network's own characteristics, the deep neural network model in this thesis has already obtained relatively high accuracy of classification. Additionally, this model has high scalability and ease of modification, which means that its structure and function can be further improved in the future. In short, the model described in this thesis can be regarded as an experimental attempt, which applies deep neural network on hyperspectral image classification.

References

[1] Landgrebe, D. (2002). Hyperspectral image data analysis. IEEE Signal processing magazine, 19(1), 17-28.

[2] Benediktsson, Jon Atli, and Pedram Ghamisi. Spectral-Spatial Classification of Hyperspectral Remote Sensing Images. Artech House, 2015.

[3] Landgrebe, David A. Signal theory methods in multispectral remote sensing. Vol. 29. John Wiley Sons, 2005.

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

[5] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. nature, 521(7553), 436.

[6] Bioucas-Dias, Jos M., et al. "Hyperspectral remote sensing data analysis and future challenges." IEEE Geoscience and remote sensing magazine 1.2 (2013): 6-36.

[7] Fauvel, Mathieu, et al. "Advances in spectral-spatial classification of hyperspectral images." Proceedings of the IEEE 101.3 (2013): 652-675.

[8] Camps-Valls, Gustavo, et al. "Remote sensing image processing." Synthesis Lectures on Image, Video, and Multimedia Processing5.1 (2011): 1-192.

[9] Smith, R. B. "Introduction to hyperspectral imaging, MicroImages." Inc., USA (2012).

[10] Jia, Xiuping, Bor-Chen Kuo, and Melba M. Crawford. "Feature mining for hyperspectral image classification." Proceedings of the IEEE 101.3 (2013): 676-697.

[11] Donoho, David L. "High-dimensional data analysis: The curses and blessings of dimensionality." AMS Math Challenges Lecture 1 (2000): 32.

[12] Hughes, Gordon. "On the mean accuracy of statistical pattern recognizers." IEEE transactions on information theory 14.1 (1968): 55-63.

[13] Bovolo, Francesca, Lorenzo Bruzzone, and Lorenzo Carlin. "A novel technique for subpixel image classification based on support vector machine." IEEE Transactions on Image Processing 19.11 (2010): 2983-2999.

[14] Sun, K., Geng, X., Ji, L., Lu, Y. (2014). A new band selection method for hyperspectral image based on data quality. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 7(6), 2697-2703.

[15] Chen, Yushi, et al. "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks." IEEE Transactions on Geoscience and Remote Sensing 54.10 (2016): 6232-6251.

[16] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.

[17] Vincent, Pascal, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." Journal of Machine Learning Research 11.Dec (2010): 3371-3408.

[18] Hecht-Nielsen, Robert. "Theory of the backpropagation neural network." Neural Networks 1.Supplement-1 (1988): 445-448.

[19] Prechelt, Lutz. "Automatic early stopping using cross validation: quantifying the criteria." Neural Networks 11.4 (1998): 761-767.

[20] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of machine learning research15.1 (2014): 1929-1958.

[21] Tetko, Igor V., David J. Livingstone, and Alexander I. Luik. "Neural network studies. 1. Comparison of overfitting and overtraining." Journal of chemical information and computer sciences 35.5 (1995): 826-833.

[22] Licciardi, Giorgio, et al. "Linear versus nonlinear PCA for the classification of hyperspectral data based on the extended morphological profiles." IEEE Geoscience and Remote Sensing Letters 9.3 (2012): 447-451.

[23] Villa, Alberto, et al. "Hyperspectral image classification with independent component discriminant analysis." IEEE transactions on Geoscience and remote sensing 49.12 (2011): 4865-4876.

[24] Bandos, Tatyana V., Lorenzo Bruzzone, and Gustavo Camps-Valls. "Classification of hyperspectral images with regularized linear discriminant analysis." IEEE Transactions on Geoscience and Remote Sensing 47.3 (2009): 862-873.

[25] Tarabalka, Yuliya, et al. "SVM-and MRF-based method for accurate classification of hyperspectral images." IEEE Geoscience and Remote Sensing Letters 7.4 (2010): 736-740.

[26] Chen, Yushi, et al. "Deep learning-based classification of hyperspectral data." IEEE Journal of Selected topics in applied earth observations and remote sensing 7.6 (2014): 2094-2107.

[27] Chen, Yushi, Xing Zhao, and Xiuping Jia. "Spectralspatial classification of hyperspectral data based on deep belief network." IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 8.6 (2015): 2381-2392.

[28] Hyperspectral Images- Indian Pines.https://engineering.purdue.edu/ biehl/MultiSpec/hyperspectral.html.

[29] Plaza, Antonio, Javier Plaza, and Gabriel Martin. "Incorporation of spatial constraints into spectral mixture analysis of remotely sensed hyperspectral data." Machine Learning for Signal Processing, 2009. MLSP 2009. IEEE International Workshop on. IEEE, 2009.

[30] Zhiwang, Zhang. "Deep Learning on Hyperspectral Image Classification using Spatial-Spectral Feature Extraction." Bachelor thesis (2017).

[31] Ioffe, S., Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[32] Mou, L., Ghamisi, P., Zhu, X. X. (2017). Deep recurrent neural networks for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, 55(7), 3639-3655.

[33] Ma, Xiaorui, Jie Geng, and Hongyu Wang. "Hyperspectral image classification via contextual deep learning." EURASIP Journal on Image and Video Processing 2015.1 (2015): 20.

[34] Huang, G., Liu, Z., Weinberger, K. Q., van der Maaten, L. (2017, July). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (Vol. 1, No. 2, p. 3).