HYPERSPECTRAL DATA CLASSIFICATION VIA CAPSULE NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ELMAS SOYAK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2019

Approval of the thesis:

HYPERSPECTRAL DATA CLASSIFICATION VIA CAPSULE NETWORKS

submitted by **ELMAS SOYAK** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. Gözde Bozdağı Akar
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Kemal Leblebicioğlu
Electrical and Electronics Engineering, METU _____

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Cüneyt Bazlamaçcı
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Seniha Esen Yüksel
Electrical and Electronics Engineering, Hacettepe University _____

Assist. Prof. Dr. Elif Vural
Electrical and Electronics Engineering, METU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Elmas Soyak

Signature :

ABSTRACT

HYPERSPECTRAL DATA CLASSIFICATION VIA CAPSULE NETWORKS

Soyak, Elmas
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Gözde Bozdağı Akar

September 2019, 64 pages

In this thesis, a novel deep architecture capsule networks are investigated for hyperspectral data classification purposes. Even though this algorithm resembles convolutional neural networks (CNN), which is one of the most successful methods in classification, capsule networks have been developed to overcome the limitations of it. CNN applies convolution operation to extract features in the samples and uses these features to classify them. However, it fails to measure the relationship between these features. Moreover, pooling operation that is used in CNN to reduce the number of parameters results in loss of position information and thus decreases the success of classifier. With the novelties proposed in capsule networks, it is intended to resolve the shortcomings of CNN mentioned above. Instantiation parameters such as position, orientation, scale of each feature are kept in a capsule, and both the presence of the relevant feature and the instantiation parameters of the feature are utilized in the classification step. In the experiments performed on hyperspectral data, the efficiency of capsule networks is evaluated by using different number and structure of training samples. A CNN algorithm with a similar structure is constructed and compared with capsule networks. Although the presented method yields successful results, it has

been observed that iteration is exhausting in terms of memory and processing time.

Keywords: capsule networks, hyperspectral data, machine learning, remote sensing, deep learning

ÖZ

KAPSÜL AĞLARI İLE HİPERSPEKTRAL VERİLERİN SINIFLANDIRILMASI

Soyak, Elmas
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Gözde Bozdağı Akar

Eylül 2019 , 64 sayfa

Bu tez kapsamında, yeni bir derin mimari olan kapsül ağlarının hiperspektral veri sınıflandırma üzerindeki başarısı incelenmiştir. Bu yöntem, günümüzde sınıflandırmada kullanılan en başarılı yöntemlerden biri olan evrişimsel sinir ağlarına benzemekle beraber, yöntemin kısıtlarını gidermek ve başarımını artırmak amacıyla geliştirilmiştir. Evrişimsel sinir ağları, evrişim operasyonunu kullanarak özniteliklerin ortaya çıkarılmasını sağlamaktadır. Daha sonra bu öznitelikleri kullanarak bir sınıflandırma gerçekleştirmektedir. Ancak, bu özniteliklerin kendi aralarındaki ilişkiyi ölçmekte yetersiz kalmaktadır. Ayrıca, evrişimsel sinir ağlarında parametre sayısını azaltmak amacıyla kullanılan havuzlama işlemi, özniteliğin konum bilgisinin kaybolmasına sebep olmakta ve yöntemin başarımını düşürmektedir. Kapsül ağlarında sunulan yeniliklerle, evrişimsel sinir ağlarının bahsedilen eksiklerinin giderilmesi amaçlanmıştır. Her bir özniteliğe ait konum, yönelim, ölçek gibi somutlaştırma parametreleri bir kapsül içinde tutulmakta ve sınıflandırma aşamasında hem ilgili özniteliğin varlığına hem de özniteliğin bahsedilen somutlaştırma parametrelerindan yararlanılmaktadır. Hiperspektral veriler üzerinde yapılan deneylerde, farklı sayıda ve ve yapı-

daki eğitim kümeleri kullanılarak, kapsül ağlarının verimi değerlendirilmiştir. Ayrıca, benzer yapıda bir CNN modeli oluşturulmuş ve performans açısından kapsül ağlarıyla karşılaştırılması yapılmıştır. Sunulan yöntemin başarılı sonuçlar verdiği görülmekle beraber, yineleme işleminin bellek ve işlem süresi açısından yorucu olduğu gözlemlenmiştir.

Anahtar Kelimeler: kapsül ağları, hiperspektral veri, makine öğrenmesi, uzaktan algılama, derin öğrenme

This thesis is dedicated to my family.

ACKNOWLEDGMENTS

I want to thank to my supervisor Prof. Dr. Gözde Bozdağı Akar for her supports. Moreover, I am also grateful to my previous thesis supervisor Assoc. Prof. Dr. Mehmet Mete Bulut for his assistance.

Lastly, I want to thank to my family, my friends and my fiance Kemal Gürkan Toker for their endless supports.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiii

LIST OF FIGURES xv

LIST OF ABBREVIATIONS xvi

CHAPTERS

1 INTRODUCTION 1

2 LITERATURE IN HYPERSPECTRAL IMAGE CLASSIFICATION USING DEEP LEARNING 7

3 CAPSULE NETWORKS 23

3.1 Artificial Neural Networks 23

3.2 Convolutional Neural Networks 26

3.2.1 Drawbacks of CNN 28

3.3 Capsule Networks: A Novel Deep Learning Algorithm 29

3.3.1 What is a Capsule? 30

3.3.1.1 Computing the magnitude of a capsule 30

3.3.1.2 Affine transformation of input vectors 31
3.3.2 Dynamic routing 32
3.3.3 Loss function 33
3.3.3.1 Margin loss 33
3.3.3.2 Reconstruction loss 34
3.3.4 Capsule Networks vs. CNN 34
4 EXPERIMENTS 37
4.1 Datasets 37
4.2 Preprocessing The Datasets 39
4.3 Architecture of the Proposed Method 42
4.4 Results and Comparison 45
4.4.1 Effect of Filter Number 46
4.4.2 Effect of Kernel Size 46
4.4.3 Effect of Number of Routing Iterations 47
4.4.4 Effect of Number of Training Samples 48
4.4.5 Effect of Window Size 50
4.4.6 Effect of Number of PCs 50
4.4.7 Effect of Data Augmentation 50
4.4.8 Comparing with CNN 53
5 CONCLUSION 57
REFERENCES 59

LIST OF TABLES

TABLES

Table 2.1 Brief explanation of the studies 9

Table 2.2 Brief explanation of the studies 10

Table 2.3 Performance of the studies on Pavia University dataset 12

Table 2.4 Performance of the studies on Pavia Centre dataset 14

Table 2.5 Performance of the studies on Indian Pines dataset 16

Table 2.6 Performance of the studies on Salinas dataset 18

Table 4.1 Size of Classes for Hyperspectral Datasets 38

Table 4.2 Spectral Information Preserved at Each Principal Component 42

Table 4.3 Results for CapsNet with varying number of filters, PaviaU, 200 samples 47

Table 4.4 Results for CapsNet with varying kernel sizes, 200 samples 47

Table 4.5 Results for CapsNet with varying number of iterations in dynamic routing, PaviaC, 400 samples 48

Table 4.6 Results for CapsNet with varying number of training data and window size 49

Table 4.7 Effect of PC Numbers on Classification Accuracy for All Datasets (window size = 7x7) 51

Table 4.8 Effect of data augmentation on classification Accuracy for all datasets, window size =7x7 52

Table 4.9 Confusion matrix of experiments using augmented data (x50) on Pavia University dataset 52

Table 4.10 Confusion matrix of experiments using augmented data (x50) on Pavia Centre dataset 53

Table 4.11 Comparison with CNN using 400 training samples each class (window size = 7x7) 54

Table 4.12 Comparison with CNN using data augmentation (window size = 7x7) 54

Table 4.13 Confusion matrix of experiments with CNN using augmented data (x50) on Pavia University dataset 55

Table 4.14 Confusion matrix of experiments with CNN using augmented data (x50) on Pavia Centre dataset 56

LIST OF FIGURES

FIGURES

Figure 1.1 Signatures of surface materials taken by AVIRIS sensor [1] . . . 2

Figure 1.2 Electromagnetic spectrum [2] 3

Figure 1.3 Atmospheric transmission with respect to wavelengths [6] 3

Figure 1.4 Spectral reflectance of vegetation showing the contributing materials [1] 5

Figure 3.6 The signatures of land types in Pavia University dataset. 31

Figure 4.1 False color maps and groundtruths of Pavia University, Pavia Centre, and Salinas datasets [48] 39

Figure 4.2 The signatures of land types in Pavia Centre dataset. 40

Figure 4.3 The signatures of land types in Salinas dataset. 41

Figure 4.4 Proposed capsule network architecture 43

Figure 4.5 Parameters at each layer of the proposed network 44

Figure 4.6 Capsule output of experiments on Pavia Uni. dataset. 45

Figure 4.7 Classification map for Pavia Centre with 400 training samples. . 48

LIST OF ABBREVIATIONS

1D	1-Dimensional
2D	2-Dimensional
3D	3-Dimensional
AA	Average Accuracy
ANN	Artifical Neural Networks
AVIRIS	Airborne Visible/Infrared Imaging Spectrometer
BLDE	Balanced Local Discriminant Embedding
CCD	Charge Couple Device
CNN	Convolutional Neural Networks
DBN	Deep Belief Nets
DR	Dimension Reduction
FE	Feature Extraction
FOV	Field of View
GPU	Graphical Processing Unit
HRS	Hyperspectral Remote Sensing
IFOV	Instantenous Field of View
KSC	Kennedy Space Center
LR	Logistic Regression
MLP	Multilayer Perceptron
NIR	Near Infrared
OA	Overall Accuracy
PC	Principal Component
PCA	Principal Component Analysis
RBF	Radial Basis Function

RBM	Restricted Boltzmann Machine
ReLU	Rectifying Linear Unit
ROSIS	Reflective Optics System Imaging Spectrometer
SAE	Stacked Auto-Encoders
SDA	Stacked Denoising Auto-Encoders
SLIC	Simple Linear Iterative Clustering
SNR	Signal-Noise-Ratio
SPP	Spatial Pyramid Pooling
SVM	Support Vector Machines
VS	Vector Stacking

CHAPTER 1

INTRODUCTION

Remote sensing is defined as sensing objects at a distance without making pyhsical contact in order to gather information about them. Aircrafts or satellites are deployed for remotely monitoring the objects on the Earth. Sensors on these devices monitor the earth's surface and collect necessary data. They operate in various spectral bands of electromagnetic spectrum from NIR to visible region (Figure 1.2). Remote sensing can be divided into 2 categories as active and passive. Active systems emit signals and measure the signals reflecting from the objects. Passive systems only gather radiation from other sources. The solar radiation is highly exposed to atmospheric emission at certain wavelengths. The bands outside these regions are called atmospheric windows and the solar energy passes through the atmosphere to the earth surface in these regions. Atmospheric transmission as a function of wavelength can be seen in Figure 1.3, . The molecules in the atmosphere such as H_2O and CO_2 are the main contributors of these windows [1].

In this thesis, we are interested in hyperspectral imaging which is a subclass of remote sensing. In recent years, due to the advances in technology, hyperspectral remote sensing systems have become an essential tool in monitoring the earth surface [3] [4]. Hyperspectral images contain hundreds of spectral bands (in infrared (IR) band of electromagnetic spectrum, 0.4-2.4μm) that help observe the spectral characteristics of the scene. The prefix "hyper-" comes from including large number (more than 100) of narrow spectral bands (10-20 nm wide). Solar energy at these spectral bands is reflected by the earth objects and the reflectivity value is used to discriminate the materials. Hyperspectral images can be represented as three-dimensional data cubes (hypercube) as shown in Figure 1.1. The hypercube contain spatial data including

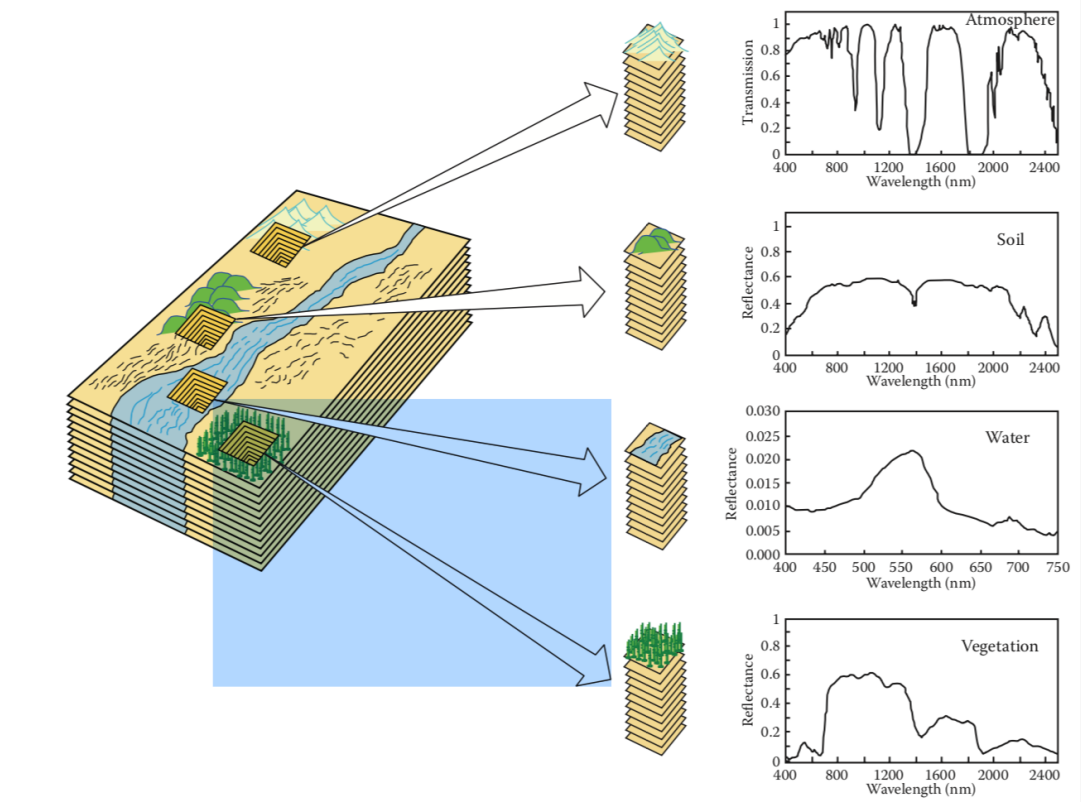


Figure 1.1: Signatures of surface materials taken by AVIRIS sensor [1]

pixels. And each pixel has spectral information [5].

Hyperspectral images are gathered by hyperspectral sensor systems. This sensor systems are divided into 2 groups depending on the data acquisition system: whiskbroom and pushbroom systems[1]. Whiskbroom imaging system usually consists of a single sensor and this sensor scans each line through a rotating scan mirror. Pushbroom imaging system consists of an array of detectors sweeping one line at a time. It scans the area in the direction of motion of aircraft. The presence of multiple sensors brings the necessity of calibration which is very complicated and time consuming. However, sensor dwell time (IT) of data could be longer in pushbroom systems and this helps to get a data with higher signal-to-noise-ratio (SNR). There are several sensor systems manufactured for hyperspectral imaging. The Pavia University, Pavia Centre, Salinas datasets that we use in this thesis are taken by the sensors listed below:

- AVIRIS (Airborne Visible/Infrared Imaging Spectrometer): was developed to gather data on an aircraft in 1989. The sensor operates in 224 bands in spectral

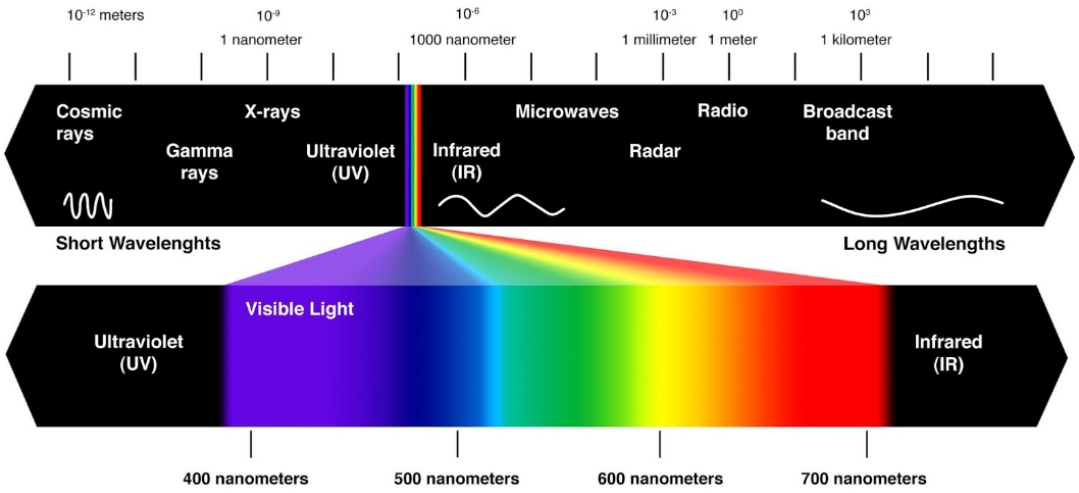


Figure 1.2: Electromagnetic spectrum [2]

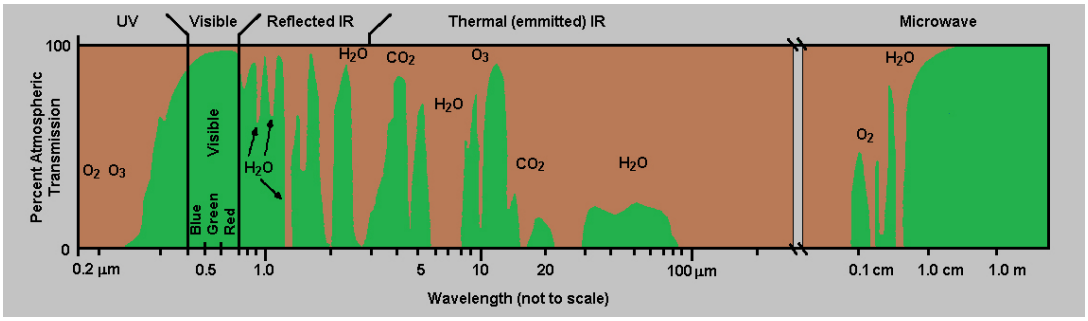


Figure 1.3: Atmospheric transmission with respect to wavelengths [6]

band of 0.35-2.5 µm with a spectral resolution of 10nm. This sensor system acquires image using whiskbroom scanning mode. Data obtained from this sensor has a dimension of 614*512*224. Each sensor has an IFOV of 0.05°. Total field of view (FOV) of the system is 30°.

- ROSIS (Reflective Optics System Imaging Spectrometer): was developed as a compact airborne imaging spectrometer for research purposes. The sensor operates in 115 bands in spectral band of 0.43-0.86 µm with a spectral resolution of 4nm. This sensor system acquires image using a 2D CCD array. Each sensor has an IFOV of 0.03°. Total field of view of the system is 32°.

Hyperspectral imaging technology provides rich spectral information about the land types. This information is utilized in a broad range of application areas[3] [4]. Some

of the application areas of this technology are:

- Military surveillance: Human or vehicle detection/classification.
- Agriculture and crop analysis: Monitoring the status of plants on the field or controlling the area whether forbidden crops such as cannabis are planted.
- Resource exploration: Geographical mapping of minerals, oil
- Environmental monitoring: Detecting forest fires or predicting weather by sensing clouds can be accomplished with remote sensing technology. Atmospheric ozone depletion, deforestation, global warming, urban growth are among the areas remote sensing can be used.

Variety of research fields are studied in these application areas. Some of those are unmixing [7][8], target detection [9], classification [10], anomally detection [11] etc. In this thesis, we focus on hyperspectral image classification. Each material has a unique reflectance behaviours in 0.4-2.5μm solar-reflective spectral region due to the molecular geometry and elements of the object. and this uniqueness provides discriminating power for classification. This spectral behaviour originates from the chemicals constituting the material. For instance, transition metals (e.g. Fe, Ti, Cr) contribute to the charge transfer and so determines the absorption feature. Besides, vibrational processes in H20 and OH- associated with water, hydroxyl, carbonate and sulfate define the absorption property ([1]). The parameters characterizing the diagnostic absorption features can be directly correlated with the chemistry and structure of the sample. For example, vegetation indices of hyperspectral bands are detected which are mainly NIR, red and blue bands. In Figure 1.4, impact of pigments such as chlorophyll-a (Chla), chlorophyll-b (Chlb), carotenoids (Cars), anthocyanins (Anths), water, and ligno-cellulose on the reflectance can be observed.

Some of the challenges that researchers encounter during classification are;

- Spectral variability during data acquisition: Atmospheric conditions have a big impact on the spectral variability. Atmospheric attenuation and scattering directly affect the signal reaching the sensor. The major molecules contributing

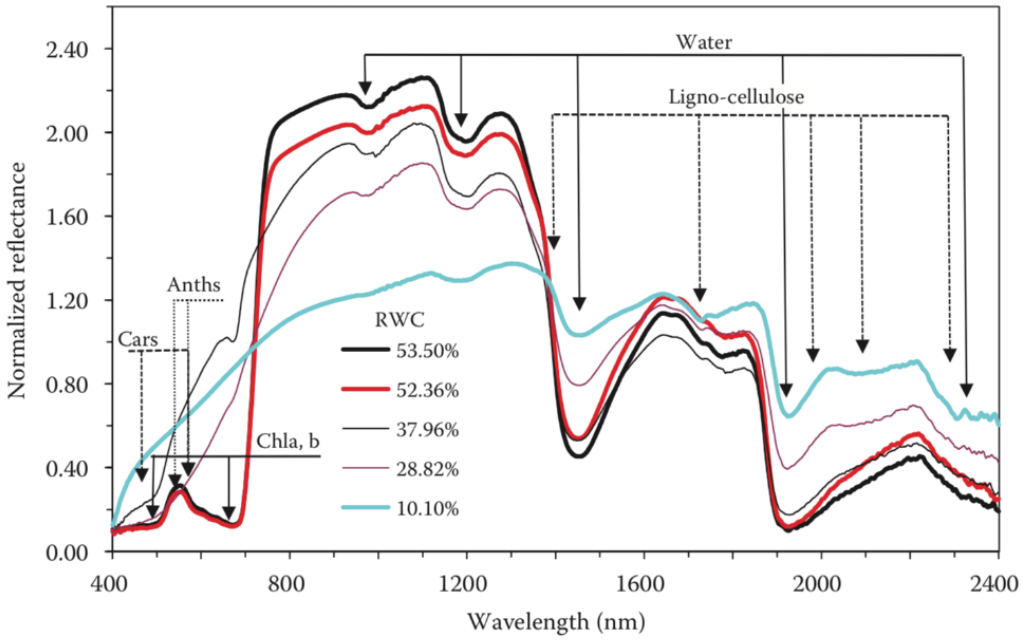


Figure 1.4: Spectral reflectance of vegetation showing the contributing materials [1]

to attenuation are water vapor, carbon dioxide, and oxygen. Aerosols in the air result in scattering of reflecting signals.

- Spectral mixing: One pixel may cover multiple objects. Signature of pixel area is a combination of signatures of multiple objects. This fact complicates the object categorization. Unmixing algorithms are needed to reveal the components in this unit pixel.
- Similarity between different class signatures. For instance, the chemical structure of vegetation may lead similar spectral behaviours.
- high dimensionality,
- limited training samples.

Feature extraction plays a key role to deal with these problems. The performance of classification is highly dependent on the learned features. That is why different methods have been proposed to extract effective features. Recently, deep methods such as CNN, autoencoders have been gaining a large amount of interest in hyperspectral image applications as feature extractor and classifier, due to the fact that (i) multi-layer

architecture of deep methods is able to extract more abstract and robust features than shallow methods and (ii) nonlinear activation functions at the end of each layer help to reveal nonlinear properties exhibited in the data. In this thesis, capsule network which is one of the popular deep architecture, is proposed for hyperspectral image classification. The efficiency of capsule networks were evaluated by using different number and structure of training samples in the experiments.

The rest of the thesis is organized as follows. In Chapter 2, studies that propose deep architectures for hyperspectral image classification are reviewed. In Chapter 3 theory of capsule network is depicted and it is compared with CNN. Experimental results with three hyperspectral data sets and discussions are provided in Chapter 4. Finally, Chapter 4 summarizes the observations and concludes the thesis.

CHAPTER 2

LITERATURE IN HYPERSPECTRAL IMAGE CLASSIFICATION USING DEEP LEARNING

Due to the fact that gathering/accessing data is becoming easy and computing power of the computers is becoming high, deep architectures have become useful and popular in machine learning field in recent years. And, deep methods have proven its effectiveness in many classification areas such as scene labeling [12], digit classification [13], character classification [14], face recognition [15], natural language processing [16]. Besides these fields, remote sensing field also utilized the effectiveness of the deep methods for feature extraction and classification. In this section, the studies that propose deep architectures for hyperspectral image classification will be discussed. Summary information about these studies is given in Table 2.1 and 2.2. Also, performances of the studies on Pavia University, Salinas, Pavia Centre and Indian Pines datasets are given separately in Table 2.3, 2.4, 2.5 and 2.6.

In [17], a 5-layer CNN is adopted to extract features and classify the data. Layers in the structure are input, convolution, max pooling, fully-connected and output layers, respectively. This simple structure is adopted since typical CNNs with many convolutional layers are not applicable for hyperspectral data. The algorithm is applied to 3 datasets: Indian Pines, Salinas and University of Pavia.

In [18], a joint spectral-spatial framework is proposed for hyperspectral image classification. In this paper, principal component alaysis (PCA) is utilized for dimension reduction purpose. The first several PCs of a neighborhood region are extracted and fed into CNN. Three PCs are transformed from the 103 channels by PCA, and the size of neighborhood region and the spectral feature map in pixels is 42x42. Deep CNN structure consists of 3 convolutional and 2 subsampling layers. The spectral feature

maps are generated by dividing the spectral vector into 42 subvectors and taking the square root of dot product of each subvector. At the classification step, LR is chosen.

In [19], a 3D CNN model is introduced which combines spectral and spatial features. Instead of one pixel, the neighborhood region of each pixel is provided as input to the system. By doing this, both spectral and spatial information is taken into account and better performance is observed. During training, dropout and L2 normalization are applied to increase robustness of the classifier. Moreover, virtual sample enhancement is developed to overcome the small training set problem. There exists 2 ways of creating virtual samples. One way is to multiply the data with a factor and adding random noise. The other way is to linearly combine two samples with varying ratios and adding noise. The algorithm achieves by far the best results on the experiments.

In [20], the model has 2 channels of CNN, designed for spectral and spatial features. The spectral channel takes 1D pixel as the input and applies convolution and pooling operations on this data. On the other way, for the spatial channel, the images over spectral bands are averaged in order to fuse the information from all bands and suppress the noise. The spatial neighboring patch of each pixel is extracted as input of the spatial channel and fed to CNN. These features are simultaneously fed to fully connected layers. After that, a softmax regression layer takes this joint feature as input and classifies the data. In this paper, transfer learning method is applied to the model. According to transfer learning theory, in order to improve performance in limited training sample case, the network is trained using samples from other remote sensed scenes. The bottom and mid layers are carried to the network of the current scene. Indian Pines dataset is used for tests whereas Salinas dataset is employed for purpose of transfer learning.

In [21], an efficient CNN architecture is improved to boost its discriminative capability for hyperspectral image classification. Parameter optimization is achieved using a small training set and 1*1 kernels are applied. The network structure consists of 3 convolutional layers, 2 normalization, 2 dropout and a global average pooling layer. 1*1 convolutional kernels can only extract features among the different bands instead of spatial features. Normalization layers and global average pooling layer can extract features in the spatial domain. In this paper, small training set is focused, so 3 to 15

Table 2.1: Brief explanation of the studies

Index	**Method**	**Year**	**Dimension Reduction Method**	**Novelty of Proposed Method**	Data Aug.
[17]	CNN	2015	-	simple CNN with 1-layer convolution	-
[18]	CNN	2015	PCA (3 PCs)	dividing data into 42 sub-vector and taking dot product of each sub-vector	-
[19]	CNN	2016	-	CNN with 3D kernels applied; data augmentation, regularization and dropout investigated	applied
[20]	CNN	2016	averaging images over spectral bands	spectral and spatial features in 2 channels, and transfer learning	-
[21]	CNN	2017	-	1*1 kernels applied to prevent over-fitting with 3*3 kernels; small training sets used	applied
[22]	CNN	2017	-	center pixel paired with each neighbor pixel to increase number of training data	-
[23]	CNN	2017	-	3D CNN with 3D kernel	-
[24]	CNN	2017	-	multi-scale CNN with kernels 1*1,3*3 and 5*5, residual learning and data augmentation	applied
[25]	CNN	2016	BLDE &PCA	PCA-applied spatial data is fed to CNN, the outputs of BLDE and CNN are stacked and sent to LR	-
[26]	CNN	2018	-	convolutional layer outputs reshaped to make a matrix, 2D image obtained to feed CNN, 3D kernels used	-
[27]	CNN	2018	-	very large kernel sizes, border mirroring, 5-layer CNN with 3D kernels	-
[28]	ANN	2017	-	distance to each class center is calculated and added to the soft-max loss	applied
[29]	SAE	2014	PCA	spectral vector and PCA-applied spatial features are stacked and fed to a SAE	-
[30]	SAE	2015	PCA (3 PCs)	multi-scale features are extracted using SAE from 3 images corresponding to 3 PCs and fed to CNN	-
[31]	SAE	2015	PCA (replacing Lab color space with PCs)	super-pixels generated and majority vote applied to each super-pixel to fine-tune the classification of SAE	-
[32]	SAE-CNN	2016	SAE	Features are extracted using SAE and sent to CNN, spatial pyramid pooling is applied to overcome scale variance	-

Table 2.2: Brief explanation of the studies

Index	**Method**	**Year**	**Dimension Reduction Method**	**Novelty of Proposed Method**	Data Aug.
[33]	SAE	2016	-	unsupervised training of each layer via SAE with no spatial information	-
[34]	SAE	2015	-	weighted average of features of neighbor pixels are added using an update layer, features of each test sample are described as linear combinations of training features (collaborative representation)	-
[35]	DBN	2015	DBN for spectral data, PCA for spatial data (5 PCs)	spectral features extracted using DBN and spatial features using PCA, then concatenated for classification using LR	-
[36]	RBM	2014	-	data smoothed via nonlinear diffusion, then features extracted with a 3-layer RBM and classified with OMP or SP algorithms	-
[37]	RBM	2014	PCA (3 PCs)	a conventional 2-layer DBN with spectral and spatial features concatenated as input	-
[38]	DBN	2017	-	weights are diversified to make hidden units behave uncorrelated and active	-
[39]	CAP	2018	-	conventional CapsNet with 3D kernels, spatial information added	-
[40]	CAP	2018	-	capsule network with 3D convolutional layer corporating 11*11 window region and all spectral bands	-

samples are chosen for each class. The datasets utilized to observe the performance of the algorithm are Indian Pines, Salinas and Pavia University. The proposed method is compared with the version where 3x3 kernels are selected for convolution. It is observed that overfitting problem occurs with 3x3 kernels whereas it does not occur with 1x1 kernels. Moreover, the effect of dropout is measured by setting dropout rate to 0 and 0.6, respectively. It is seen that both training and test losses are lower when dropout is applied.

In [22], a novel method is proposed which increases the number of training samples to ensure that the advantage of CNN can be actually offered. For each pixel, pixel-pairs, constructed by combining the center pixel and each one of the surrounding pixels, are classified by the trained CNN, and the final label is then determined by a majority voting strategy. Pair of samples belonging to the same class is labeled with their class number whereas that of samples from different classes is labeled as 0. In doing so, the amount of input data for training exhibits quadratic growth and the internal correlation of neighbors is utilized.

In [23], a 3D-CNN framework is proposed that takes full advantage of both spectral and spatial information contained within hyperspectral data. Without any preprocessing and post-processing operations, the method directly applies 3D kernels on the input image. The architecture consists of 2 convolutional layers and a fully connected layer, no pooling operation is applied. The size of kernels are determined by experiments. The algorithm is applied to 3 different datasets: Pavia University, Botswana and Indian Pines.

In [24], a multi-scale fully-convolutional neural network is proposed that can optimize both spectral and spatial information. It is one of the first works to utilize a very deep fully CNN with for hyperspectral image classification. The input data is convolved with 3 different sizes of filters $1 * 1 * d$, $3 * 3 * d$, and $5 * 5 * d$, where the parameter d corresponds to the number of spectral bands. The output of each filter is combined together to be fed into the network. The concept of "residual learning" is also introduced to handle sub-optimality caused by limited number of training data. This module sums the values of input and output layers which has been proved to better optimize the weight of each layer. Multi-scale filtering bank and residual learning

Table 2.3: Performance of the studies on Pavia University dataset

Index	**Method**	**Training Data Size**	**Data Aug.**	**OA**	**AA**	κ
[17]	CNN	200x9	-	92.56	-	-
[18]	CNN	3,921	-	95.18	93.51	93.64
[19]	CNN	3,930	applied	0.9954	99.66	99.4
[20]	CNN	-	-	-	-	-
[21]	CNN	3x9	applied	67.85	-	60.4
[22]	CNN	200x9	-	96.48	-	-
[23]	CNN	20,925	-	99.39	98.85	99.2
[24]	CNN	200x9x4	applied	96.73	-	-
[25]	CNN	50x9	-	96.98	-	-
[26]	CNN	34,220 (80%)	-	99.52	-	-
[27]	CNN	200x9	-	98.06	98.61	97.44
[28]	ANN	200x9x400	applied	98.55	97.42	98.05
[29]	SAE	25,550	-	98.52	97.82	0.9807
[30]	SAE	3,921	-	96.37	95.06	95.0
[31]	SAE	21,388	-	96.7	95.4	95.0
[32]	SAE-CNN	3,921	-	96.28	96.31	95.1
[33]	SAE	21,388	-	95.97	-	-
[34]	SAE	-	-	-	-	-
[35]	DBN	21,900	-	99.05	98.48	98.75
[36]	RBM	-	-	-	-	-
[37]	RBM	-	-	-	-	-
[38]	DBN	-	-	93.11	93.92	90.82
[39]	CAP	60x9	-	95.9	96.2	94.56
[40]	CAP	8129 (15%)	-	**99.95**	**99.9**	**99.93**

contribute to the learning of the network with a small amount of training data. Moreover, data augmentation is performed via mirroring the training samples across the horizontal, vertical and diagonal axes. In order to verify the effectiveness of residual learning, a similar network without residual learning module is trained and it failed to converge in training due to the small size of training data.

In [25], a deep CNN model is proposed with balanced local discriminant embedding (BLDE) algorithm for dimension reduction. Since hyperspectral samples exhibit intraclass variation and similarity between different classes, the selection of dimension reduction method has a vital role on the classification success. It is known that supervised dimension reduction (DR) methods are more successful than unsupervised methods since they aim to separate each class. BLDE algorithm estimates a linear mapping that simultaneously maximizes the local margin between different classes and keeps the samples within-class stay close. On the other hand, CNN is applied to extract spatial-related deep features. Before applying CNN, the principal components of input data is extracted using PCA. Then, BLDE-based features are stacked with CNN-based features and fed into an LR classifier. The method is tested on Pavia Center and Pavia University datasets. Although DR operation takes considerable amount of time, the results are satisfying.

In [26], a 3D CNN algorithm is proposed. The structure consists of 2 convolutional layers, 1 reshape layer, 1 pooling layer and 3 fully connected layers where the last layer is softmax layer. It takes 3x3 neighborhood region with all spectral bands and feeds to the network. The vectors obtained after convolution are stitched into a matrix whose width is equal to the number of convolution filters. By doing this, a 2D data is obtained at the output of convolution. After that, the network behaves like a 2D image classifier. At the output layer, XGBoost is implemented instead of softmax layer in order to prevent overfitting. During experiments, 80% of all samples are used for training. Moreover, a capsule networks is implemented, however it does not give expected benefits.

In [27], a 5-layer CNN model which uses spectral-spatial information is introduced. 3D kernels are applied at convolutional layers to get discriminative properties. Very large patch sizes, like 9, 19 and 29, are adopted believing that it defines the sample

Table 2.4: Performance of the studies on Pavia Centre dataset

Index	**Method**	**Training Data Size**	**Data Aug.**	**OA**	**AA**	κ
[17]	CNN	-	-	-	-	-
[18]	CNN	-	-	-	-	-
[19]	CNN	-	applied	-	-	-
[20]	CNN	-	-	-	-	-
[21]	CNN	-	applied	-	-	-
[22]	CNN	-	-	-	-	-
[23]	CNN	-	-	-	-	-
[24]	CNN	-	applied	-	-	-
[25]	CNN	50x9	-	99.87	-	-
[26]	CNN	-	-	-	-	-
[27]	CNN	-	-	-	-	-
[28]	ANN	200x9x400	applied	99.73	99.25	0.996
[29]	SAE	-	-	-	-	-
[30]	SAE	-	-	-	-	-
[31]	SAE	74,076	-	99.8	99.4	0.991
[32]	SAE-CNN	-	-	-	-	-
[33]	SAE	-	-	-	-	-
[34]	SAE	2,138	-	**99.9**	**99.73**	**0.998**
[35]	DBN	-	-	-	-	-
[36]	RBM	-	-	-	-	-
[37]	RBM	-	-	-	-	-
[38]	DBN	-	-	-	-	-
[39]	CAP	-	-	-	-	-
[40]	CAP	-	-	-	-	-

better. During the experiments, the algorithm reaches a classification error with 1500 iterations when 9x9 neighborhood is extracted, on the other hand, it takes 1000 iteration to reach the same error value with 19x19 window size. While dividing the image into $n * n$ patches, pixels around the borders cannot be extracted. This algorithm replicates border pixels with mirroring and includes these pixels into classification. Varying training samples (50, 100 and 200) are used during the tests to observe its effect. Best results are obtained with 200 training samples and 29x29 patches.

In [28], an artificial neural network-based framework is introduced. In this paper, the center loss parameter is generated to enhance the separability of the deep features. To this end, it is obtained by averaging the distances between each input feature and the corresponding center of each class. Center of each class is determined by averaging the features of the corresponding class. The loss function of the proposed architecture is computed by adding center loss to the softmax loss. Another contribution of this paper is the center classifier approach, which assigns label to a given sample according to its nearest class center. Differing from other models which are trained using patch-based samples, this method feeds only spectral features at the training stage. At the testing stage, the neighborhood area is taken into consideration. Distance of the given sample to each class center is calculated for 8 scales of neighborhoods (3x3, 5x5, ..., 17x17). After gathering the neighborhoods with same label, the weight of each class is computed and the sample is assigned to the class with maximum weight.

There are several methods which make use of stacked auto-encoders (SAE) for hyperspectral image classification. In [29], a joint spectral-spatial multi-layer stacked auto-encoder structure is proposed. This method applies PCA in the preprocessing step in order to reduce the dimension of input data and first n principal components are taken. In order to include the spatial information, a neighborhood region of size $a * a$ is cropped from the n-dimensional data for each pixel. That matrix with size $a * a * n$ is flattened to size $a^2n * 1$ and concatenated to the raw pixel data which is not transformed with PCA. This vector provides input to the deep architecture. At the classification step, logistic regression is preferred. This algorithm is tested on Pavia and KSC datasets. For KSC dataset, the overall accuracy, average accuracy and Kappa coefficient values are 98.76%, 97.9% and 0.9862, respectively. For Pavia University dataset, these values are 98.52%, 97.82%, and 0.9807.

Table 2.5: Performance of the studies on Indian Pines dataset

Reference	**Method**	**Training Data Size**	**Data Aug.**	**OA**	**AA**	κ
[17]	CNN	200x8	-	90.16	-	-
[18]	CNN	-	-	-	-	-
[19]	CNN	1,765	applied	97.56	99.23	97.0
[20]	CNN	200x8	-	95.58	-	-
[21]	CNN	3x8	applied	64.19	-	59.9
[22]	CNN	200x8	-	94.34	-	-
[23]	CNN	5,043	-	99.07	98.66	-
[24]	CNN	200x8x4	applied	94.24	-	-
[25]	CNN	-	-	-	-	-
[26]	CNN	8,199 (80%)	-	99.09	-	-
[27]	CNN	200x16	-	98.37	99.27	98.15
[28]	ANN	-	applied	-	-	-
[29]	SAE	-	-	-	-	-
[30]	SAE	-	-	-	-	-
[31]	SAE	5,124	-	91.9	-	-
[32]	SAE-CNN	-	-	-	-	-
[33]	SAE	5,124	-	92.06	-	-
[34]	SAE	1,036	-	99.22	98.57	99.11
[35]	DBN	5,100	-	95.95	95.45	95.39
[36]	RBM	-	-	81.3	85.33	-
[37]	DBN	-	-	-	-	-
[38]	DBN	200x8	-	91.03	93.54	89.07
[39]	CAP	-	-	-	-	-
[40]	CAP	1537 (15%)	-	**99.45**	**99.34**	**99.37**

In [30], a multiscale convolutional auto-encoder (MCAE) is developed to extract deep features. The structure is divided into 2 main components: feature extraction and classification. In the initial step, 2 kinds of features are extracted: spectral and MCAE features. Spectral features are obtained by applying PCA and taking 3 principal components (PC). MCAE operation is applied after spectral features are obtained. For 3 images corresponding to each PC, pyramid pooling is applied by downsampling the images at 3 scales. The resulting images are normalized and then trained in CNN. The CNN consists of 2 layers with filter size 7*7. Sigmoid functions and 2*2 max pooling operations are applied to each layer. At the output, 315 feature maps are obtained. This corresponds to 315 dimensions for each pixel. Then, a logistic regression classifier is employed to label each sample using these features.

In [31], a deep model based on stacked denoising autoencoders (SDA) is introduced to learn spectral feature representations of the data. Moreover, superpixels are deployed to generate the spatial constraints for the refinement of spectral classification results. The structure of SDA consists of 5 layers: one input layer, 3 hidden SDA layers, and one output layer. The pixels are fed directly to the network as input layer. Then, PCA is applied to the output of the network. After replacing Lab color space with principal components, superpixels are generated using SLIC algorithm. For pixels in the boundary of each superpixel, classification is performed individually. Finally, class labels are assigned by majority vote in each superpixel.

In [32], proposed framework serves as an engine for merging the spatial and spectral features via SAE and CNN followed by a LR classifier. SAE is aimed to get high level features by reducing the dimension of data. Since traditional CNN is intolerant to the scale variance of the objects, SPP is introduced to hyperspectral image classification for the first time by pooling the spatial feature maps of the top convolutional layers into a fixed-length feature. The algorithm includes 4 steps: generating the deep spectral features via SAE, training a CNN model and pooling the top convolutional layers, determining the spectral-spatial feature adjustment parameter and concatenating the spectral-spatial features in order to feed into LR classifier. The adjustment parameter is determined through experiments and arranges the ratio of weight of features.

In [33], a SDAE based deep method is proposed where unsupervised training is

Table 2.6: Performance of the studies on Salinas dataset

Reference	**Method**	**Training Data Size**	**Data Aug.**	**OA**	**AA**	κ
[17]	CNN	200x16	-	92.6	-	-
[18]	CNN	-	-	-	-	-
[19]	CNN	-	applied	-	-	-
[20]	CNN	-	-	-	-	-
[21]	CNN	3x16	applied	85.24	-	83.6
[22]	CNN	200x8	-	94.8	-	-
[23]	CNN	-	-	-	-	-
[24]	CNN	200x16x4	applied	95.42	-	-
[25]	CNN	-	-	-	-	-
[26]	CNN	43,303 (80%)	-	98.95	-	-
[27]	CNN	-	-	-	-	-
[28]	ANN	200x9x400	applied	96.98	98.81	96.62
[29]	SAE	-	-	-	-	-
[30]	SAE	-	-	-	-	-
[31]	SAE	27,064	-	95.5	-	-
[32]	SAE-CNN	-	-	-	-	-
[33]	SAE	-	-	-	-	-
[34]	SAE	-	-	-	-	-
[35]	DBN	-	-	-	-	-
[36]	RBM	-	-	-	-	-
[37]	DBN	-	-	-	-	-
[38]	DBN	-	-	-	-	-
[39]	CAP	60x6	-	**99.94**	**99.95**	**99.92**
[40]	CAP	8200 (15%)	-	99.81	99.92	99.79

achieved via stacked denoising auto-encoders and supervised finetuning via logistic regression (LR). ReLU is chosen as the activation function since it increases the separability of the features. Each DAE layer is trained independently and decoding layers are removed after training. At the output layer, logistic regression is added for classification. The spatial information of the hyperspectral image is not utilized in the paper. Three datasets are used in experiments: Indian Pines, Botswana and Pavia University. Three networks parameters, number of layers, number of units in each layer and the standard deviation of Gaussian noise are determined according to the optimal classification results on the validation data. The computation time of the algorithm is compared with linear-kernel SVM and RBF-kernel SVM methods. It is observed that the proposed method is much faster than RBF-kernel SVM and slightly slower than linear-kernel SVM.

In [34], a spectral-spatial classification method is proposed which deploys spatial updated deep auto-encoder (SDAE) and a collaborative representation-based classification. The algorithm consists of 3 parts: feature representation, classification, and spatial regularization. In the feature extraction step, both spectral and spatial features are obtained. While extracting spectral features, correlation between each sample is calculated and this similarity regularization term is added to the energy function of the auto-encoder. By doing so, it is aimed to keep the correlation between the similar samples during encoding operation. In order to take the spatial information into consideration, an update layer is inserted after the hidden layer. In the update layer, each feature is replaced with the weighted average of the features computed from the surrounding samples. The weight of each sample exponentially varies with respect to its Euclidean distance from the center. After pre-training all the layers, multinomial logistic regression (MLR) is added to the output layer in order to supervise the training. At the classification step, collaborative representation-based classification is applied. Here, the features of each test sample is described as linear combinations of training features. Using a classical sparse representation-based classification, class label of each test sample is determined. The framework is designed as to succeed when the samples are limited, so the experiments are carried out with small training sets.

Deep belief net is another deep network which is preferred for hyperspectral image

classification. In [35], a DBN-based architecture is proposed, which combines the spectral-spatial FE and classification together to get high accuracy. The framework is a hybrid of PCA, hierarchical learning-based FE and logistic regression (LR). In spectral feature extraction step, the raw data of all the spectral channels are used and several layers of DBN are applied to extract robust features. In parallel, PCA is applied along the spectral dimension for dimension reduction and first few PCs are taken only. Then, for each pixel, a $w * w$ neighborhood region is extracted and flattened. The size of flattened data is $w^2 * n$, where n is the number of PCs. Spectral and spatial features are combined using a vector stacking (VS) approach. The 1-D vector obtained in spatial classification part is added to the end of spectral vector. LR layer produces the class labels using these hybrid features.

In [36], Restricted Boltzmann Machine (RBM) is utilized to construct a deep model. At the preprocessing step, nonlinear diffusion is applied to smoothen the data without losing edge information. The diffusion type used is Perona malik diffusion. The output of diffusion is fed as input to a 3-layer RBM network and features are extracted. The 1st layer of the network is trained according to contrastive divergence (CD). The number of neurons for each unit is 200, 60, 60 and 200, respectively. After that, samples are labeled via Orthogonal Matching Point (OMP) and Subspace Pursuit (SP) classification algorithms. The experiments are conducted on Indian Pines dataset.

In [37], restricted Boltzmann machine (RBM) model and its deep structure deep belief networks (DBN) are constructed for feature extraction and classification of hyperspectral images. The aim of using these structures is to avoid the loss of detailed information due to the dimension reduction operations. In this method, PCA is applied as dimension reduction step and first 3 principal components are preserved. For each pixel in this new 3-channel image, $7 * 7$ neighborhood area is extracted. Then, it is reshaped into a $147 * 1$ column vector and stacked in series behind the spectral curve. This spectral-spatial feature provides input to the DBN which consists of 2 layers of RBM. In the experiments, the hyperspectral image acquired in 2012 over the University of Houston and the neighboring urban area is used.

In [38], a deep diversified DBN-based classification algorithm is proposed. Training of conventional DBNs make many hidden units behave similarly or perform as

'dead' or 'over-tolerant'. Diversifying operation is introduced to encourage neurons to be uncorrelated and prevent these drawbacks. Typical DBN consists of unsupervised pretraining where spectral features are determined and supervised fine-tuning where class information is used for fine-tuning of parameters. Diversifying operation is applied to both steps for better performance. A diversity promoting prior $p(w^l)$ is incorporated into training equation and the weights are updated according to that equation. For each layer, diversifying is performed separately. Experiments are conducted on the proposed method for various numbers of training data. It is concluded that the accuracy proportionally increases with number of training samples. Moreover, effect of the depth of algorithm is also analyzed. It is found that best depths are 2 and 4 for Indian Pines and University of Pavia data sets.

In [39], a modified capsule network is proposed. In this work, a 2-layer network is thought to be sufficient to classify data. A 7x7 neighborhood region with all spectral bands included constitutes the input of the architecture. 2 datasets Pavia University and Salinas are taken as the benchmark dataset. However, only largest 6 classes of Salinas are employed. To show the efficiency of the proposed method, 60 samples from each class are used for training, and satisfying results are obtained. The method is compared with a CNN algorithm that is very similar to the capsule networks in terms of structure. Although it takes longer time to train CapsNet, it outclasses CNN algorithm.

In [40], a 3D capsule network is constructed. A patch of 11*11 is taken along with all spectral bands and fed to the 1st convolutional layer. 15% of dataset is used for training. Here, capsule units are specialized as to cover both spectral and spatial information. Proposed network consists of a 3D convolutional layer, primary capsule layer, dense capsule layer, and fully connected layers. In this structure, first 3 layers act as encoder and fully connected layers act as decoder.

After analyzing these articles, some key points are extracted on using deep methods for hyperspectral data classification. One and most important finding is the effect of number of training samples. These papers can be divided into 2 groups. One group takes equal number of samples from each class like 50, 200, etc. The other group uses particular percentage of whole data just as 60% or 80%. Comparing methods with the

ones using same amount of training data will give a better idea about the classifier. For hyperspectral datasets, there are large differences in number of samples of each class. When equal number of samples are taken from each class, this corresponds to a small proportion of whole data. On the other hand, when this restriction is ignored and a vast proportion of data is used for training, classifier better generalizes the input data. It is observed that second group commonly gets a better performance ([26]). Several papers aimed to solve this by creating virtual data using augmentation and obtained satisfying results as in [28]. Moreover, some papers intend to exhibit remarkable performance with very limited training data as in [21]. Various data representations employed in these papers have a decisive impact on the results. Most methods extract several PCs and then utilize $n * n$ window region to feed the network. However, best results are obtained by including all spectral bands along with spatial information as in [23]. Utilizing this representation comes with a price on computational complexity and execution time. In [31], employing SLIC as an alternative technique to get spatial information brings outstanding success in classification. Unlike other areas that use deep networks, using very deep networks does not give the best results as in [24]. To sum up, training data number, data representation, and data augmentation are among the most critical factors affecting the classifier performance.

Deep methods used in classification of hyperspectral images are mainly based on CNN, due to the fact that this architecture gives the most satisfying classification performance. However, it has several limitations. It fails to measure the relationship between the features. Moreover, pooling operation that is used in CNN to reduce the number of parameters results in loss of position information and thus decreases the success of classifier. In this thesis, capsule networks which are proposed to overcome the shortcomings of CNN is investigated for hyperspectral data classification.

CHAPTER 3

CAPSULE NETWORKS

In this thesis, we focus on capsule network which is a method based on artificial neural networks. Therefore, artificial neural networks will be mentioned firstly on the purpose of setting up a substructure for capsule networks. And then, CNN architecture which is widely used in hyperspectral image processing field, will be discussed in Section 3.2. Furthermore, the limitations of CNN, differences between capsule network and CNN, the benefits, drawbacks and theory of capsule networks are discussed in the following sections.

3.1 Artificial Neural Networks

The idea of artificial neural networks is inspired from human's central nervous system, since human brain is good at cognitive science. The variations of artificial neural networks (ANN) are extracting features from data automatically by experience like human brain does. The network consists of nodes. The nodes in networks correspond to human neuron cells and they perform small arithmetic operations. The output of each layer is fed as input to the next layer [41] [42]. Thus layered structure is constructed with connected nodes to learn abstract representations. Some of the concepts in artificial neural networks are given below:

Nodes: A neural network can be defined as connection of nodes. They behave like neuron cells in brain. Each node takes input from many other nodes and computes its own activation value (Figure 3.1). A node is activated only when sufficient input value is obtained. Moreover, each node has its own bias added to the summation. From Equation 3.1.1, it is seen that nodes have a simple working principle. In this equation,

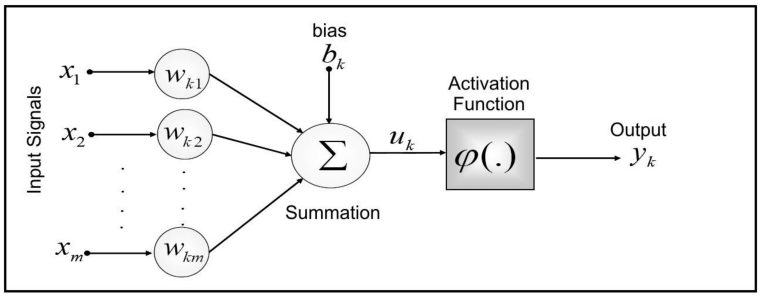


Figure 3.1: Nodes in an ANN [43]

x_i represents the signals from the input. If there is a layer before, it represents the node output of previous layer. Here, m number corresponds to the number of signals coming from the previous layer. w_{ki} describes the connection weight between nodes i and k. Initially, random values are assigned to this parameter and it is tuned via backpropagation. b_k parameter in given equation represents the bias value for node k. Calculation of bias is achieved in a similar fashion with weight parameter. Aim of a neural network is to learn the weight and bias parameters in training phase and this learnt parameters are utilized in class prediction of test samples.

$$u_k = \sum_{1}^{m} x_i w_{ki} + b_k \tag{3.1.1}$$

Activity functions are applied to the output of each node for normalization. Output of this function equals to the probability of detection of a particular feature. In a biological neuron, this corresponds to the firing of neuron. The motivation behind choosing a nonlinear function is to transform samples into a linearly separable space so that discriminability increases. Nonlinearity is necessary for linearly inseparable cases. Hyperbolic tangent (tanh), rectified linear unit (ReLU) and logistic sigmoid functions are the most commonly chosen activation functions. Since ReLU has less computational complexity, it is highly preferred. In Equation 3.1.2, y_k value represents the activation value of function Φ. Weighted summation (u_k) of previous layer output is put into an activation function and its output indicates the detection of a particular feature.

$$y_k = \Phi(u_k) \tag{3.1.2}$$

Output Function: Output functions are employed to represent a probability distribution over a discrete variable [44].

Learning process: Learning is basically adjusting the connection weights in the network. Cost functions are deployed to measure the classification error. Using back-propagation algorithm, cost function is minimized iteratively by tuning the weights. At each iteration, connection weights and bias are updated as to decrease the classification error. This decrease is achieved with gradient-descent algorithm. According to this, cost function is represented as a function of training parameters. Derivative of this function is computed and the parameters are updated in the opposite direction of derivative. One problem with this algorithm is that the function can get stuck in local minima while searching for minimum error. In Equation 3.1.3, calculation of error value (e) at the last layer of a network can be seen. For a n-class problem, y_i corresponds to the expected output for class i. It equals to 1 for true class and to 0 for all other classes. Sum of squares of differences between expected output and actual output constitutes the error.

$$e = \sum_{1}^{n} (y_i - f(x_i))^2 \tag{3.1.3}$$

Architecture Design: Architecture of the network has a role on the performance. Number of layers, number of nodes, etc. must be selected according to the problem. A larger network requires more training samples in order to tune the parameters and they are harder to optimize. On the other hand, a smaller network can cause underfitting. Since there are no exact rules for architecture design, an ideal one can be found by trial and error. For the case of hyperspectral data classification, a small-size network is usually adopted to avoid overfitting.

Dropout:Dropout is a technique generated to increase the classification performance. Its theory stands on the belief that overtraining can degrade classification accuracy. To solve this issue, several nodes in the network are canceled, i.e. zero output is attained from these nodes. By the experiments, it is proved that dropout prevents memorization with no computational cost. Even though training performance deteriorates with dropout, test performance gets better. The nodes that dropout is applied are chosen randomly at each layer.

Regularization: It is applied to prevent network from overfitting. This is the state where training error is small and test error is large. This happens when the networks learns the training data, but does not perform well on the new test data. A penalty term is added to the cost function as regularization term.

Data Augmentation: Data augmentation term is referred to artificially increasing the number of training data using available ones. When training data is limited, this method is applied to better generalize the model. For the case of hyperspectral data, this is achieved in several ways:

- adding random noise,
- combining 2 samples linearly with different ratios and then adding noise.

3.2 Convolutional Neural Networks

CNN is a specialized type of ANN which is developed for image classification. It uses convolution operation instead of matrix multiplication in one or more layers (Figure 3.2). The kernels used in convolution are determined via learning. These learnt kernels are scanned through 2D data. By doing so, a high output occurs where a feature appears in the input. If that feature is translated in the image, the same high output occurs with the same amount of translation.

CNN has 3 principal layers: convolution, pooling and fully-connected layers:

- Convolution layer convolves image with 2D features. When a similar 2D feature is detected, a large output value is obtained. This output value is sent to a nonlinear activation function such as ReLU.
- Pooling layer takes the output of activation function and downscales data by using pooling operation such as max pooling or average pooling (Figure 3.5). By doing so, only important information is transferred to the next layer and computation time decreases. There are 3 parameters determining the output size: filter size, stride and padding. Filter size specifies the block size that pooling will be achieved. In Figure 3.3 and Figure 3.4, 3x3 filters are deployed.

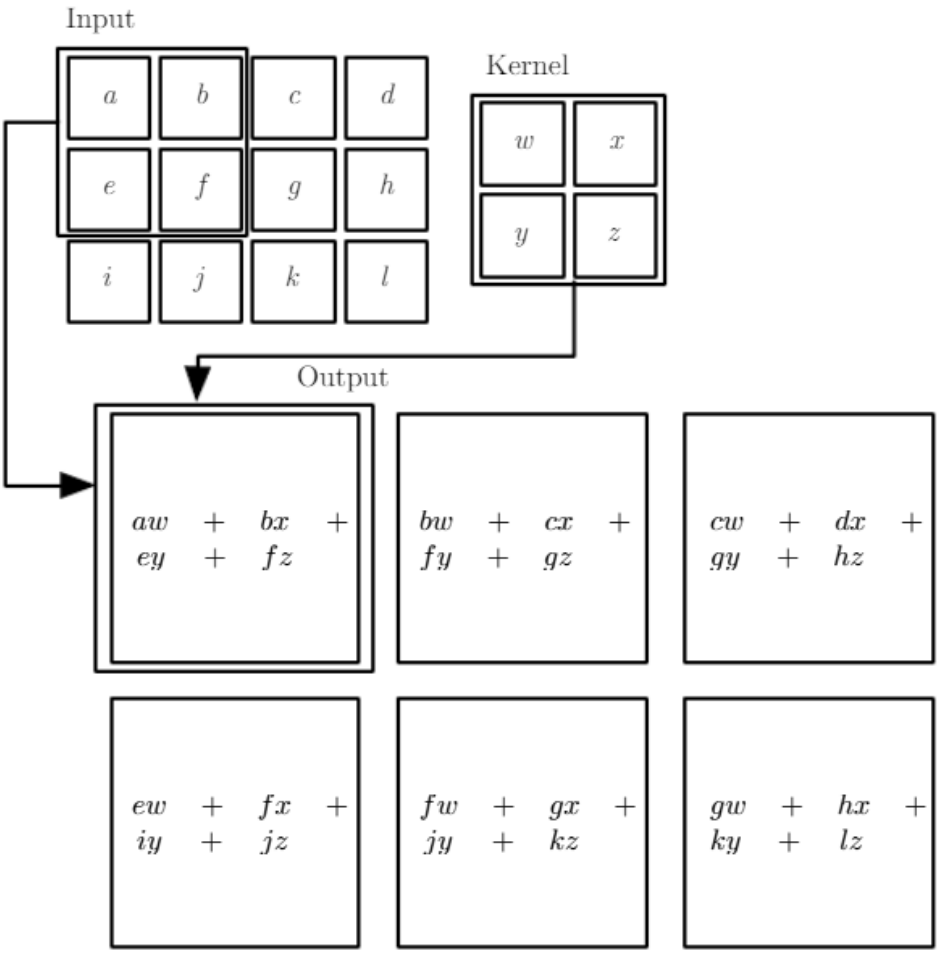


Figure 3.2: Convolution operation [45]

Stride defines how many block will be shifted at successive pooling operations. In these figures, stride values are 1 and 2 respectively. Stride directly affects the output size. The last parameter is the padding. Input volume is padded with numbers around the border in order to apply convolution to the edge pixels.

- Fully connected layer is no different than an ordinary artificial neural network layer. Each neuron in this layer is connected to each neuron in the previous layer.

Although CNN is invariant to translation, it is not invariant to rotation or scale change. The solution to rotation change is to rotate the training data and feed it to the network. One solution invented for scale changes is to apply pyramid pooling. This method

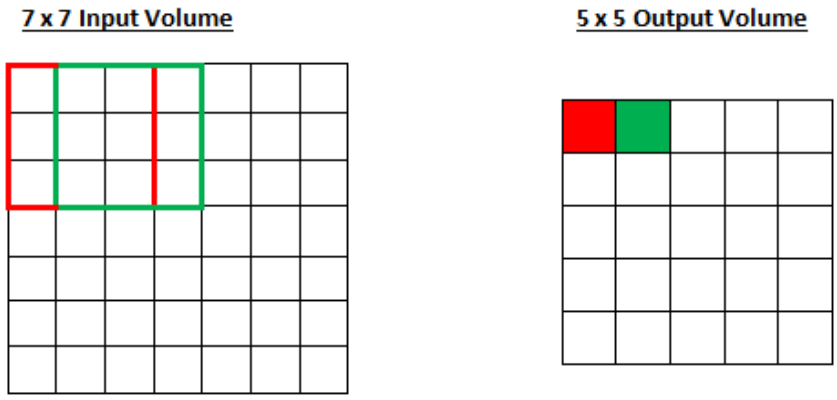


Figure 3.3: Convolution operation [46]

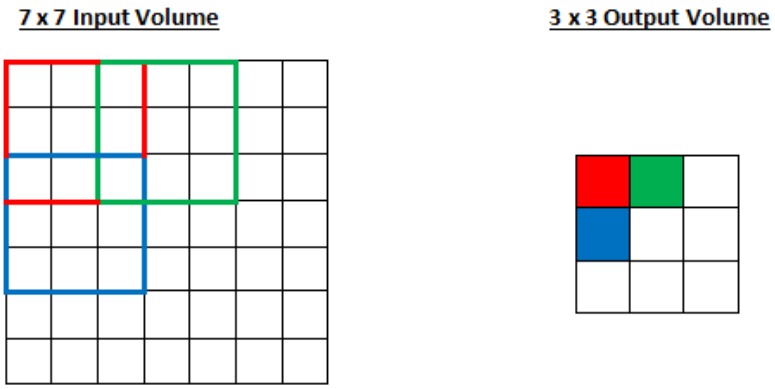


Figure 3.4: Convolution operation [46]

scales the training data in several ratios and feed all of them to the network so that network learns features at each scale. Both these methods have a drawback. When a feature at low scale and another at high scale occur at the same, the CNN accepts the presence of both features and this might mislead the algorithm.

3.2.1 Drawbacks of CNN

CNN has been the state-of-the-art method in various classification tasks. However, it has several drawbacks. Firstly, it works by collecting the features in the sample. It only seeks for the existence of an entity in the image. It is not capable of preserving properties of the entity. Other properties of an entity such as rotation, scale, etc. are

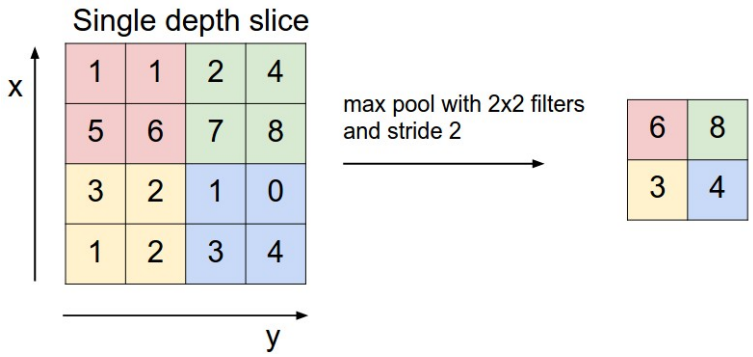


Figure 3.5: Max pooling operation [47]

discarded in CNN. To be more illustrative, for a face detection problem, it searches for the existence of nose, mouth, and eyes. The problem is that presence of these features does not mean that it is a face. The algorithm should consider the distance, orientation and scale relationships between these entities. It does not take into account the spatial and orientational relationships between these entities. Besides, CNN requires excessive amount of training samples. This brings problems for datasets with limited samples. Another drawback of CNN is utilization of max pooling operation between layers. The motivation behind max pooling is to reduce parameter number via downsampling. This brings translational invariance to the architecture. To be more illustrative, when a low-level feature is moved in the data, same output will be obtained in the high-level neurons due to translational invariance. Position information is lost with max pooling. This has a negative effect on the classification success. However, max pooling is widely used in convolutional neural networks.

3.3 Capsule Networks: A Novel Deep Learning Algorithm

Capsule network is a deep framework developed for classification purposes. It consists of a convolutional layer, primary capsule layer, capsule layer, mask layer, and decoder network. First layer is a convolutional layer and extracts the basic features such as edges and color variations. After that, primary capsule layer produces capsules from neuron outputs. This layer behaves like an inverse rendering operation. Given an image or other data, it computes the internal parameters of features such as rotation, scale, etc. Then, capsule layer extracts more abstract features in the

data. Here, dynamic routing is applied for determining weights between low-level and high-level features. The last capsule layer has capsules with the same number of classes. After applying softmax operation to the output of capsules, class probabilities are obtained. Class predictions are achieved by selecting the class with highest probability. The layers mentioned thus far are make up the encoder network. Additionally, decoder network is constructed for regularization. The capsule output of correct class is given to a 2-layer fully connected network to rebuild the original data. Difference between original data and rebuilt data is provided as a regularization term to the error function. Calculated error value is carried backwards using back propagation and learning is achieved. In the following parts, a detailed explanation of capsule networks is given.

3.3.1 What is a Capsule?

A capsule corresponds to a set of neurons. Capsules encapsulate valuable information about the state of the feature they are detecting. This information is stored in vector form. Each neuron in a capsule seeks different instantiation parameters such as the position, size and orientation. The magnitude of activity vector of each capsule represents the probability that the entity exists. Its orientation exposes the instantiation parameters. If the detected feature changes its state, the probability stays still, but its orientation changes. This instantiation parameters are determined by algorithm using training data.

3.3.1.1 Computing the magnitude of a capsule

Activation of a single neuron is calculated using a nonlinear activation function which limits the output value between 0 and 1. For the capsules, output is in a vector form, therefore a "squashing" function is developed in order to downscale the vector without changing its direction.

$$v_j = \frac{||s_j||^2}{1 + ||s_j||^2} \frac{s_j}{||s_j||} \tag{3.3.1}$$

In Equation 3.3.1, s_j refers to total input of the capsule j and v_j refers to the output of that capsule. The magnitude of the vector implies the probability of existence of an entity and the orientation implies some internal state of that entity.

3.3.1.2 Affine transformation of input vectors

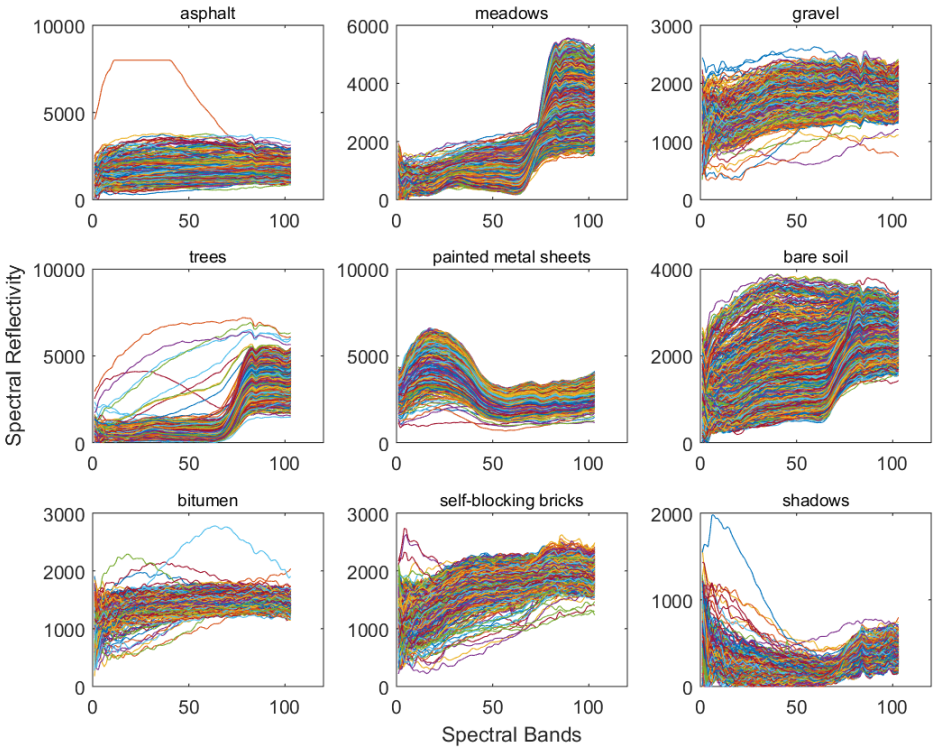


Figure 3.6: The signatures of land types in Pavia University dataset.

The output of capsules is multiplied with a weight matrix. This matrix encodes the relationship between low-level and high-level capsules. States of low-level capsules with respect to high-level capsules is contained in this affine transformation matrix. In the case of face detection, this corresponds to the relation between a nose and a face. The affine transformation matrix includes the information about the scale, position, etc. of the nose with respect to the face. In the case of hyperspectral data, this might be the location of a peak point in the signature. The form of a peak is detected on a lower capsule and the properties of that peak such as the location, slope, etc. might be stored in the weight matrix. Figure 3.6 shows the signature of land types in Pavia University

dataset. The location of peaks and curvatures defines the characteristic signature of that land type. This operation has no analogue in traditional neural networks.

$$\hat{u}_i = W_{ij} u_i \tag{3.3.2}$$

In Equation 3.3.2, u_i and $\hat{u}_i$ refer to the output of lower-level capsule and the transformed output, respectively. The variable W_{ij} represents the weight matrix.

3.3.2 Dynamic routing

Capsules receive multi-dimensional prediction vectors from the lower-level capsules. Dynamic routing algorithm measures the similarity between each lower-level and upper-lower capsule and assigns weights accordingly. In Figure 3.7, pseudo code of dynamic routing is given. Each capsule in lower level i is connected to each capsule in higher level j. However, the weights of these connections are computed iteratively. For each capsule at level j, it checks all the lower level capsules and assigns greater weight to the more similar one in an iterative way. The resemblance of capsules is determined via dot product of the capsule outputs.

Procedure 1 Routing algorithm.

procedure ROUTING($\hat{\boldsymbol{u}}_{j|i}$, r, l)
  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` computes Eq. 3
    for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$
    for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` computes Eq. 1
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} . \mathbf{v}_j$
  return $\mathbf{v}_j$

Figure 3.7: Algorithm of dynamic routing [46]

In Equation 3.3.3, weight update operation can be seen. The weight between capsules i and j is updated by adding the result of dot product to the capsule weight. Initially, all weights from level i to the capsule j are equal. Step by step, the weights increase for the lower-level capsules similar to the given high-level capsule to make sure that the output is sent only to the appropriate parent capsules. These computed weights are normalized at each iteration. The aim is to send the capsules only to the appropriate

parent capsule. In CNN, max pooling is the analogue of dynamic routing. Weights determined via dynamic routing are not model parameters. At testing step, these weights are recalculated using the procedure.

$$b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i}.v_j \tag{3.3.3}$$

In Equation 3.3.4, $\hat{u}_i$ symbolizes the output of lower-level capsule i. The term c_i represents the scalar weight of the capsule i which is determined iteratively by dynamic routing operation. Lastly, the variable u_j stands for the output of higher-level capsule j.

$$u_j = squash(\sum(c_i\hat{u}_i)) \tag{3.3.4}$$

3.3.3 Loss function

It is known that loss function (or error function) provides the necessary feedback during the learning process. Capsule networks use weighted sum of margin loss and reconstruction loss for a more efficient parameter adjustment. In Equation 3.3.5, total loss equation is given. Here, L_{total}, L_{margin} and L_{Recons} represent total loss, margin loss and reconstruction loss, respectively. α is taken as 0.0005 in the original paper. Due to this, reconstruction loss has a minor contribution to the total loss calculation.

$$L_{total} = L_{margin} + \alpha L_{Recons} \tag{3.3.5}$$

3.3.3.1 Margin loss

At the last layer of capsule networks, number of capsules is equal to number of classes. Softmax operation is applied after capsule layer gives the class probability for each class. These computed probabilities are used for margin loss calculation. The class probabilities are expected to satisfy 2 requirements defined below:

- the probability obtained at the last layer after softmax operation must be greater than 0.9 for the corresponding class of the input,
- the probability must be less than 0.1 for other classes.

Margin loss equation given in Equation 3.3.6 checks whether these conditions are met and computes an error with respect to this. In this equation, L_c represents the margin loss for class c. Here, T_c equals 1 if the sample belongs to class c, otherwise equals 0. m^+, m^- and λ constants are equal to 0.9, 0.1 and 0.5, respectively.

$$L_c = T_c max(0, m^+ - ||v_c||)^2 + \lambda(1 - T_c)max(0, ||v_c|| - m^-)^2 \tag{3.3.6}$$

As can be seen in Equation 3.3.7, margin loss (L_{margin}) is calculated by adding each class loss (L_c) in an n-class dataset.

$$L_{margin} = \sum_{c=1}^{n} L_c \tag{3.3.7}$$

3.3.3.2 Reconstruction loss

Similar to traditional deep methods, capsule network uses regularization to prevent the network from overfitting. Reconstruction loss is developed for this purpose. A 2-layer neural network, called decoder, is designed to reconstruct the input data using the capsule outputs. To be more illustrative, reconstruction loss indicates how well the input data is reconstructed using output of capsule network. Here, only the capsule output corresponding to the class of input is utilized and the capsule outputs from other classes are masked. Capsule network constitutes the "encoder" part and feedforward neural network constitutes the "decoder" part. Reconstruction loss is computed as the squared difference between the reconstructed data and the input.

3.3.4 Capsule Networks vs. CNN

In this section, similarities and differences of CNN and capsule networks are summarized. First of all, both methods apply convolution operation at the first layer to

extract basic features. These basic features include edges and color changes. After the convolutional layer, these methods utilize the extracted features in different ways. Capsule networks replace scalar neurons used in CNN with capsules which are in vector form. While the value of neuron output indicates the existence of a feature in CNN, the magnitude of capsule output vector indicates the same for capsule networks. Additionally, direction of the capsule stores pose parameters of that feature. It is expected that low-level features belonging to a high-level feature have similar pose parameters. Secondly, capsule networks take into account the existing hierarchies between simple and complex features by using affine transformation matrix. Output of a capsule at a lower level is multiplied by this matrix so that the predicted state of the high-level feature is obtained as encoded in the transformed capsule vector. Thirdly, dynamic routing in capsule networks takes the place of pooling operation in CNN. By dynamic routing, the lower level capsule sends its output to the higher level capsule whose output is similar. This is achieved by updating weights iteratively between low-level and high-level capsules. Here, the connection weight between capsules having similar pose parameters gets higher. Low-level capsules with pose parameters similar to the high-level capsule has more weight. Therefore, the pose parameters have a significant effect on the learning (weight update) process. For both algorithms, model parameters are updated via back-propagation.

Although there are several common structures in CNN and capsule networks, some critical differences make CapsNet more preferable than CNN. With the novelties proposed by capsule networks such as capsule structure and dynamic routing, features are processed in a more intelligent way than CNN does. However, nested structure of dynamic routing makes capsule network more costly.

CHAPTER 4

EXPERIMENTS

4.1 Datasets

Performance of the proposed capsule network is investigated on three public hyperspectral datasets: Salinas, Pavia University and Pavia Center [48].

Salinas: This data is collected by AVIRIS sensor over Salinas Valley, California. The sensor operates in 0.4-2,5um band region. Although it consists of 224 spectral bands, 24 water absorption bands are removed. The image size is 512x217. The dataset includes 16 classes most of which are plants. It has 3.7m geometric resolution. Spectral signatures of the classes in Salinas dataset are shown in Figure 4.3

Pavia Center: This data is taken by ROSIS sensor over Pavia, Italy. The sensor operates in 0.43-0.86um band region. It has 102 spectral reflectance bands. The image size is 1096x1096 pixels. Spatial resolution of a pixel is 1.3 meters. The dataset includes 9 classes. Spectral signatures of the classes in Pavia Centre dataset are shown in Figure 4.2

Pavia University: This data is taken by ROSIS sensor over Pavia, Italy. The sensor operates in 0.43-0.86um band region. It has 103 spectral reflectance bands. The image size is 610x610 pixels. Spatial resolution of a pixel is 1.3 meters. The dataset includes 9 classes. Spectral signatures of the classes in Pavia University dataset are shown in Figure 3.6

False color images and the groundtruths for the datasets that are used in experiments are given in Figure 4.1 a), b), c) for Pavia University, Pavia Centre and Salinas datasets, respectively. Only a small percentage of pixels in the image are used since

Table 4.1: Size of Classes for Hyperspectral Datasets

Class	**Salinas**	**Pavia Center**	**Pavia Uni.**
1	2,009	65,971	6,631
2	3,726	7,598	18,649
3	1,976	3,090	2,099
4	1,394	2,685	3,064
5	2,678	6,584	1,345
6	3,959	9,248	5,029
7	3,579	7,287	1,330
8	11,271	42,826	3,682
9	6,203	2,863	947
10	3,278	-	-
11	1,068	-	-
12	1,927	-	-
13	9,16	-	-
14	1,070	-	-
15	7,268	-	-
16	1,807	-	-
Total	**54,129**	**148,152**	**42,776**

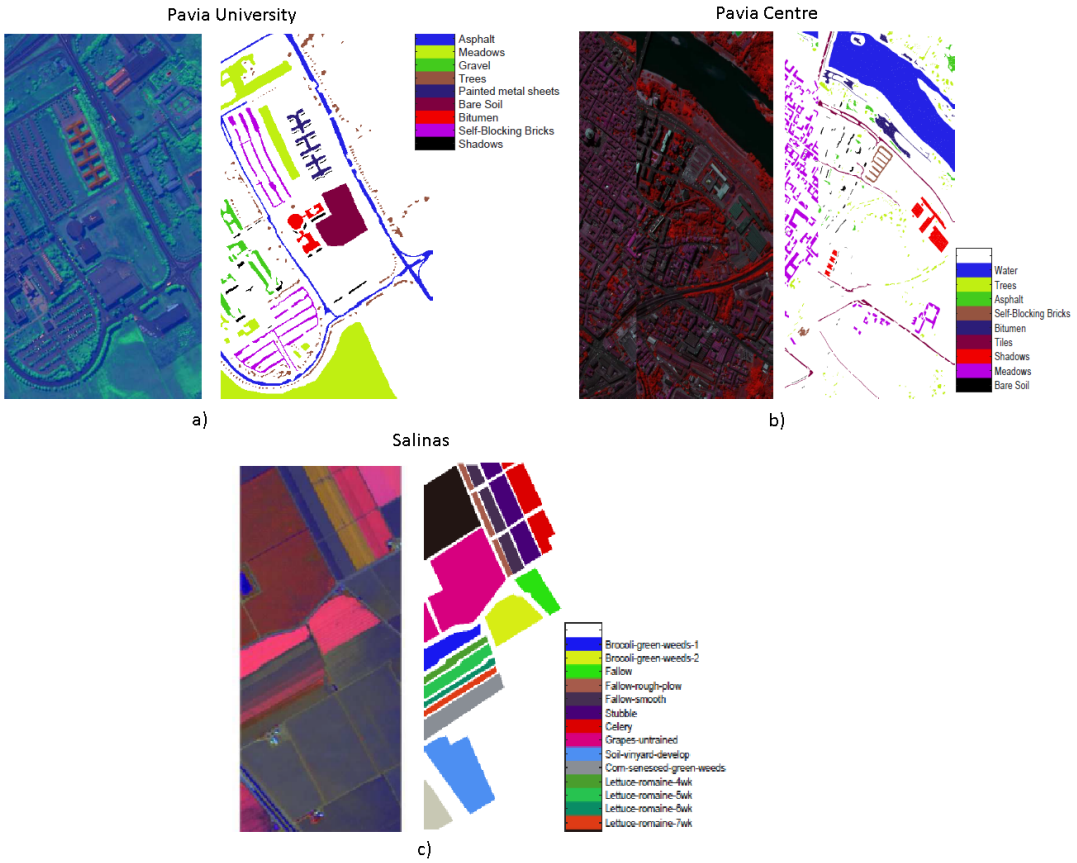


Figure 4.1: False color maps and groundtruths of Pavia University, Pavia Centre, and Salinas datasets [48]

most of them are unlabeled. In Table 4.1, the number of labeled samples in each class is listed. It is seen that the class sample sizes are not well-balanced. For this reason, equal number of training samples are taken from each class so that large classes will not overwhelm the small ones during training. Cross-validation with 4 different data is applied to obtain reliable results.

4.2 Preprocessing The Datasets

Hundreds of spectral bands in hyperspectral images provide significant information about the scene. On the other hand, high dimensionality of the data may lead to curse of dimensionality problem. Moreover, noisy and correlated bands exist in these spectral bands. In order to deal with these problems, the studies in hyperspectral

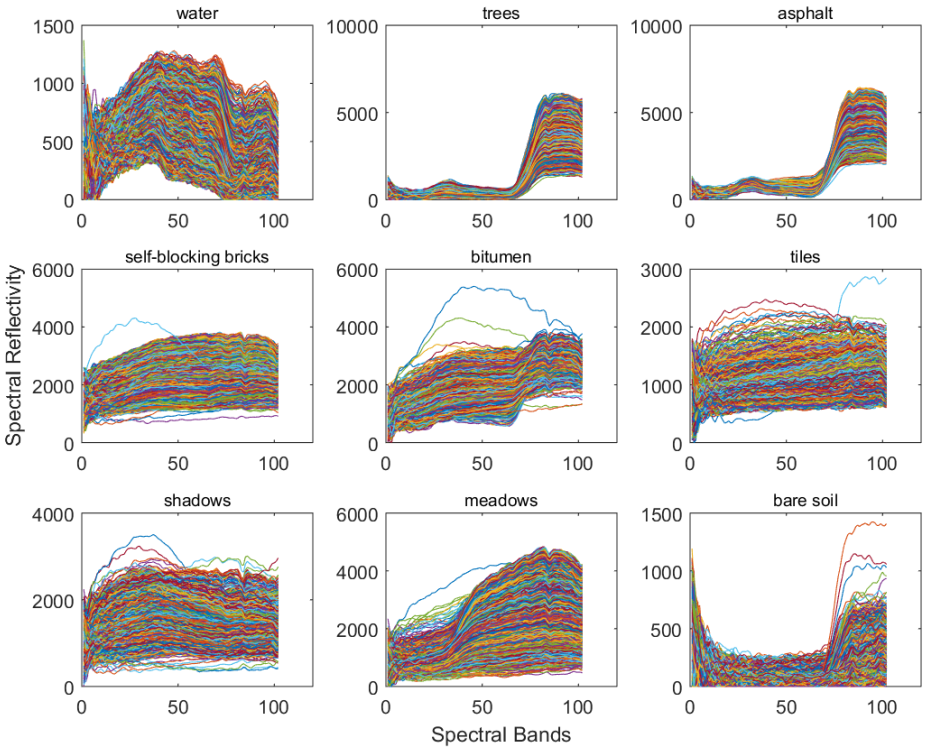


Figure 4.2: The signatures of land types in Pavia Centre dataset.

image processing utilize dimension reduction techniques. One common approach is to apply principal component analysis (PCA). It deploys orthogonal transformation to convert correlated data into uncorrelated one in an unsupervised way. Although, PCA does not guarantee to obtain separable classes, it is frequently used in hyperspectral image classification with success.

In this thesis, we utilized PCA to reduce dimension of datasets. Let C be the covariance matrix of the data. Here, the entire dataset (d dimensions) is projected onto a new subspace (k dimensions where $k < d$). To obtain this subspace, firstly, eigenvectors and eigenvalues of matrix C are computed as in Equation 4.2.1, 4.2.2 and 4.2.3.

$$C * x = \lambda * x \tag{4.2.1}$$

$$(C - \lambda I) * x = 0 \tag{4.2.2}$$

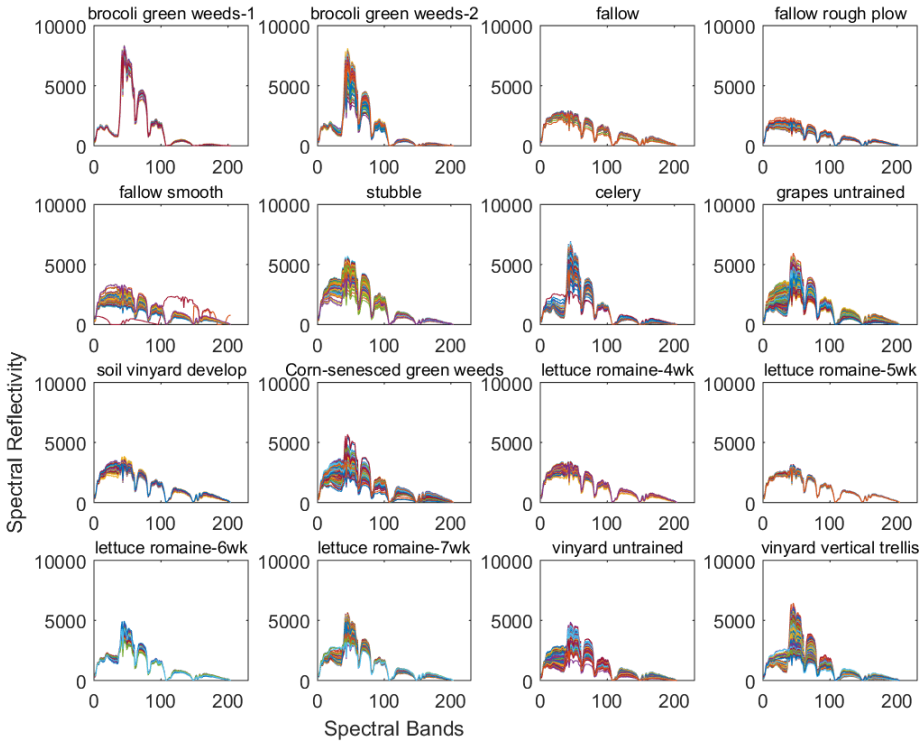


Figure 4.3: The signatures of land types in Salinas dataset.

$$det(C - \lambda I) = 0 \tag{4.2.3}$$

After that, eigenvalues are sorted in descending order and their corresponding eigenvectors are arranged in this order, too. First d eigenvectors constitute the bases of the new subspace. For hyperspectral datasets, PCA is applied and spectral information preserved at each principal component is computed as in Table 4.2. It is observed that 4 PCs contain more than 99% of information (variance).

As an alternative, Fisher's linear discriminant analysis is also implemented as a supervised dimension reduction method. However, it did not outperform PCA and not included in this thesis.

After dimension reduction, a neighboring region of size $n * n$ is cropped from this s-dimensional data for each point to deal with inter-class spectral similarities. Since some classes have very similar spectral signatures, the spectral signatures of neighbor pixels are also taken into account to handle this issue. It originates from the idea that

Table 4.2: Spectral Information Preserved at Each Principal Component

PCs	Spectral Info.			Cumulative Spectral Info.		
	Pavia C.	Salinas	Pavia U.	Pavia C.	Salinas	Pavia U.
1	0.702	0.745	0.583	0.702	0.745	0.583
2	0.260	0.235	0.361	0.963	0.98	0.944
3	0.028	0.011	0.044	0.991	0.991	0.988
4	0.003	0.005	0.003	0.994	0.997	0.991

adjacent pixels usually belong to the same class. After extracting an $n * n$ window region, the sample size becomes (n, n, d). Then this 3-dimensional sample is flattened to an array of size of $(n*n*d, 1)$. Thus, dimension reduction is performed and spatial data is taken into account in preprocessing step.

4.3 Architecture of the Proposed Method

The architecture is very similar to the one in the original paper ([49]). MNIST dataset in the paper has a size of 28*28. However, our hyperspectral data is flattened to 1D. For this reason, first layer is designed as a 1D convolutional layer instead of a 2D layer. In Figure 4.4, visualization of proposed architecture can be seen. Layers of this architecture are described below:

- Convolutional layer: For an input sample with 4 PCs and 7*7 neighborhood region, the sample size becomes 7*7*4. After flattening, it becomes a 1D data of size 196*1. After convolutional layer with 256 filters, kernel size and stride being 9 and 2, the output data has a size of 94*256.

  At this layer, number of convolution parameters are 2304, which equals multiplication of number of filters (256) with filter size (9). 256 bias parameters are applied to each convolution. Total number of parameters at this layer is 2,560.

- Primary capsule layer: Primary capsule consists of 1D convolutional layer of kernel size, filters and strides being 3, 256 and 2. The output data has a size

of 46*256. After that, a reshape layer is applied to make data fit into a 8-dimensional capsule. The resulting data dimension is 1472*8.

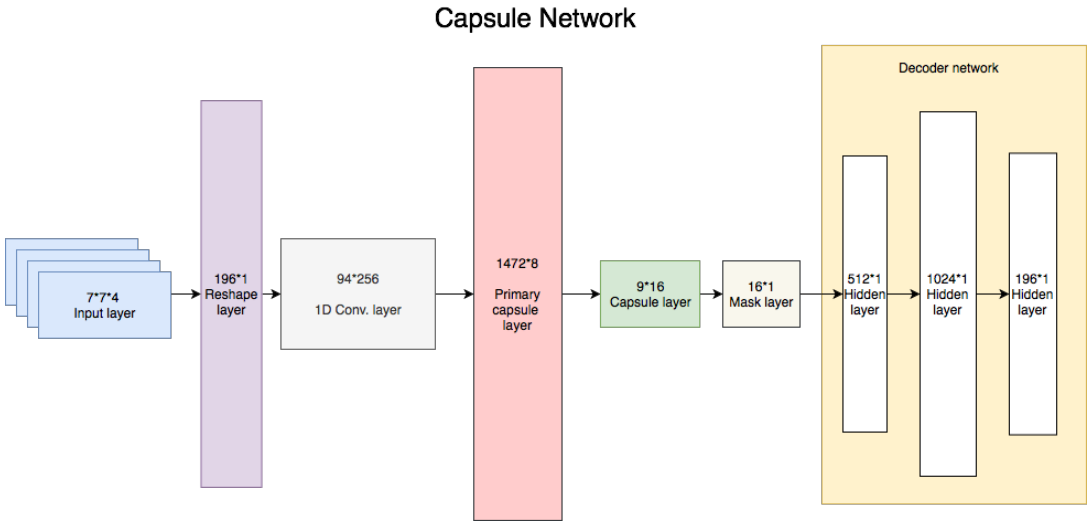
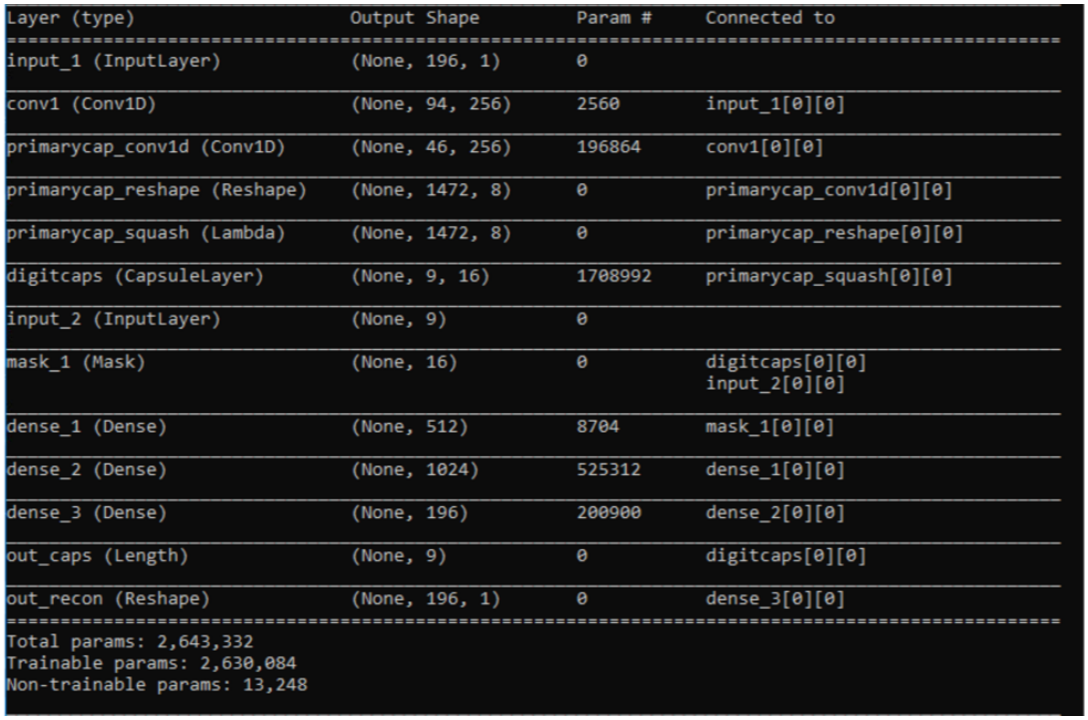


Figure 4.4: Proposed capsule network architecture

At this layer, number of convolution parameters are 196,608, which equals multiplication of number of filters (256) with input data size (256) and with filter size (3). 256 bias parameters are applied to each convolution. Total number of parameters at this layer is 196,684.

- Capsule layer: Each capsule in this layer has 16 nodes, i.e. 16 dimensions. The number of capsules is equal to the number of classes in the dataset. Each capsule corresponds to a class. Therefore, for a 9-class dataset, the output is a 9*16 matrix. The length of each capsule is calculated and used for classification error calculation. In Figure 4.6, outputs of capsule layer are displayed. Here, 4 samples belonging to meadows classes of Pavia University dataset are fed to the network. In capsule layer, 9 capsules are employed each with 8 dimensions. Each row in the matrix corresponds to the vector output of a capsule.

  At this layer, number of trainable parameters are 1472*8*9*16 = 1,695,744; which equals multiplication of input data size (1472, 8) and with output data size (9,16). Number of coupling coefficients (c_{ij}) equal to 13,248 (= 1472*9). Since these are iteratively computed with dynamic routing, they are not trainable. Total number of parameters at this layer is 1,708,992.



Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 196, 1)	0	
conv1 (Conv1D)	(None, 94, 256)	2560	input_1[0][0]
primarycap_conv1d (Conv1D)	(None, 46, 256)	196864	conv1[0][0]
primarycap_reshape (Reshape)	(None, 1472, 8)	0	primarycap_conv1d[0][0]
primarycap_squash (Lambda)	(None, 1472, 8)	0	primarycap_reshape[0][0]
digitcaps (CapsuleLayer)	(None, 9, 16)	1708992	primarycap_squash[0][0]
input_2 (InputLayer)	(None, 9)	0	
mask_1 (Mask)	(None, 16)	0	digitcaps[0][0] input_2[0][0]
dense_1 (Dense)	(None, 512)	8704	mask_1[0][0]
dense_2 (Dense)	(None, 1024)	525312	dense_1[0][0]
dense_3 (Dense)	(None, 196)	200900	dense_2[0][0]
out_caps (Length)	(None, 9)	0	digitcaps[0][0]
out_recon (Reshape)	(None, 196, 1)	0	dense_3[0][0]

Total params: 2,643,332
Trainable params: 2,630,084
Non-trainable params: 13,248

Figure 4.5: Parameters at each layer of the proposed network

- Mask layer: The aim of adding mask layer is to feed only the output of the capsule corresponding to the true class of the input sample. Capsule outputs from other classes are masked at this layer.

- Decoder network: It consists of 3 dense layers. First 2 layers have size 512 and 1024, respectively. Last hidden layer has same size with input layer, which is 196.

  First dense layer has 8192 weight parameters, which equals to multiplication in input size (16) with size of layer (512). 512 bias parameters are applied to each node in the layer. Total number of parameters at this layer is 8704. Second dense layer has 524,288 weight parameters, which equals to multiplication in input size (512) with size of layer (1024). 1024 bias parameters are applied to each node in the layer. Total number of parameters at this layer is 525,312. Last dense layer has 200,704 weight parameters, which equals to multiplication in input size (1024) with size of layer (196). 196 bias parameters are applied to each node in the layer. Total number of parameters at this layer is 200,900.

Total number of parameters in this network equals to 2,643,332. While 13,248 of

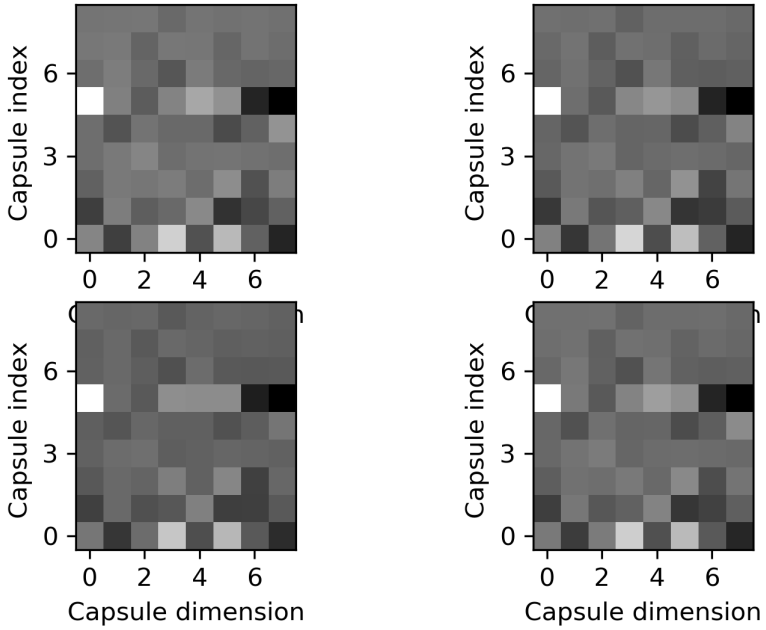


Figure 4.6: Capsule output of experiments on Pavia Uni. dataset.

them are calculated via dynamic routing, rest of them are updated with learning process. For different input data size, these number change. For a 5*5 neighborhood region, total number of parameters becomes 1,653,284. Parameters at each layer of the proposed structure is given in Figure 4.5.

4.4 Results and Comparison

The performance of the proposed capsule networks is investigated with various values of training data size, window size, PCs and filter size in experiments. In addition to these, data augmentation is performed and the impact of data augmentation is investigated. Proposed architecture is constructed using Keras library of Python which runs Tensorflow at backend. This library is specialized for building neural network models and can run tests on GPU effectively with parallel processing. The source codes given in [50] are modified for the implementation of the proposed method on Python. The tests are run on Nvidia GeForce GTX 960 graphics card. 500 epochs

In this study, classification success is measured with 3 metrics:

- Overall accuracy (OA) gives the percentage of correctly predicted test samples from the total number of samples (Equation 4.4.1).

$$Overall\ Accuracy = \frac{Number\ of\ Correct\ Predictions}{Number\ of\ Total\ Predictions} \tag{4.4.1}$$

- Average accuracy (AA) is the average of per class accuracy (Equation 4.4.2). When the classes are evenly distributed, OA and AA give same results. However, hyperspectral data usually has very different class sizes and these metric give information about the performance of the classifier on individual classes.

$$Average\ Accuracy = \frac{Sum\ of\ Class\ Accuracies}{Number\ of\ Classes} \tag{4.4.2}$$

- Kappa coefficient (κ) measures the agreement between 2 different raters. In Equation 4.4.3, p_o is observed level of agreement. It is equal to the sum of probability of true predictions for each class. p_e corresponds to expected agreement of classifiers. It is equal to the sum of multiplied probability of each class for each rater. Kappa metric is applied for multiple imbalanced classes. It takes values in the interval [0, 1].

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{4.4.3}$$

4.4.1 Effect of Filter Number

For the convolutional layer, the effect of number of filters is investigated. Utilizing more filters helps to extract more features. However, with limited training samples, these filters do not fit data sufficiently. Considering these trade-offs, optimum number of filters must be searched based on classifier performance. In Table 4.3, best results are obtained with filter numbers 128 and 256. Based on this result, 256 filters are used throughout the tests.

4.4.2 Effect of Kernel Size

The first 2 layers of capsule networks are 1D convolutional layers. In this section, the effect of kernel size in these layers is observed. Since kernel size has a crucial effect

Table 4.3: Results for CapsNet with varying number of filters, PaviaU, 200 samples

No. of Filters	OA	AA	κ	Time (sec)
16	79.71	80.51	73.94	5,313
32	79.13	80.89	73.32	5,487
64	81.19	82.88	75.9	5,515
128	81.26	**83.3**	75.88	5,510
256	**81.72**	82.34	**76.4**	5,606
512	80.32	82.91	74.79	5,451

on the success of CNN, it is expected to act similarly for the proposed algorithm. For this reason, optimum kernel size is searched for capsule networks. In the experiments with Pavia University datasets, best performance is observed with kernel size 3. Based on this result, kernel size of 3 is used throughout the tests.

Table 4.4: Results for CapsNet with varying kernel sizes, 200 samples

Datasets	Kernel Size	OA	AA	κ	Time (sec)
Pavia University	3	**81.72**	**82.34**	**76.40**	5,606
	7	79.67	80.93	74.02	5,350
	11	77.54	81.24	71.78	5,310

4.4.3 Effect of Number of Routing Iterations

In the original paper [49], it is recommended that the number of iterations be kept between 3-5. Following this rule, the experiments are conducted on Pavia Centre dataset. In Figure 4.5, it is seen that best result is obtained with 3 iterations. It is concluded that increasing the iterations causes the network to overfit the data. Changes in iterations do not directly effect the training time. Based on this result, 3 iterations are applied throughout the tests.

Table 4.5: Results for CapsNet with varying number of iterations in dynamic routing, PaviaC, 400 samples

No. of Iterations	OA	AA	κ	Time (sec)
3	**98.35**	**94.0**	**97.63**	135,055
4	97.78	93.47	97.11	137,952
5	98.09	93.64	97.26	147,188

4.4.4 Effect of Number of Training Samples

When few training samples are used, neuron weights cannot update sufficiently to fit the training data. This fact affects the success of classifier. On account of this, the number of training samples for each class is set to 200 and 400 and their outcomes are analyzed. As can be seen in Table 4.6, accuracies have increased by 1-2% with the increase in training samples. We can conclude that the classifier needs more samples to characterize the classes. It almost has no effect on computation time. In Figure 4.7, classification map for Pavia Centre dataset is visualized together with its groundtruth.

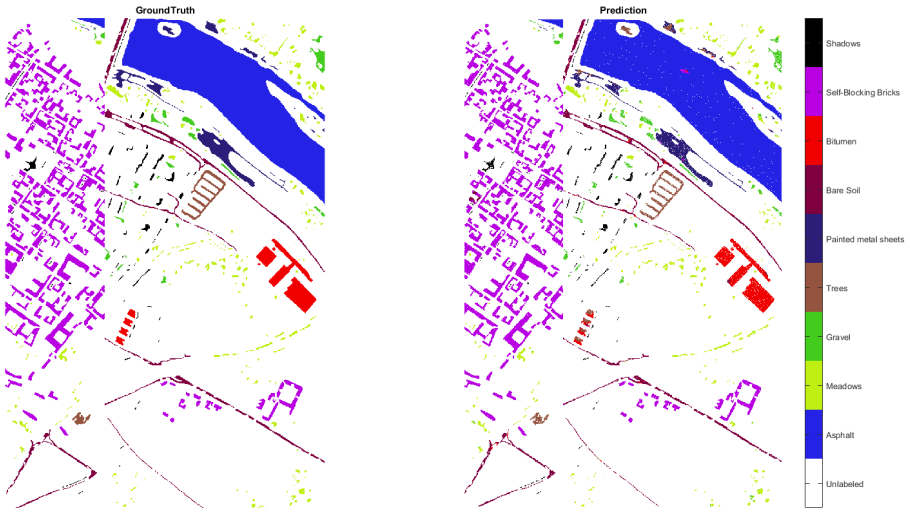


Figure 4.7: Classification map for Pavia Centre with 400 training samples.

Table 4.6: Results for CapsNet with varying number of training data and window size

Datasets	Training Data	Window size	OA	AA	κ	Time (sec)
Pavia Uni.	200	3x3	74.09	78.48	67.26	1,716
		5x5	75.75	78.67	69.39	3,286
		7x7	**81.72**	**82.34**	**76.35**	5,606
	400	3x3	77.87	80.1	71.2	1,753
		5x5	76.92	80.71	70.39	3,539
		7x7	**80.16**	**85.14**	**74.93**	6,055
Pavia Centre	200	3x3	96.48	89.76	95.00	21,190
		5x5	97.47	92.09	96.41	54,730
		7x7	**97.93**	**93.04**	**97.05**	130,797
	400	3x3	97.49	91.53	96.41	22,439
		5x5	97.91	92.83	97.00	66,384
		7x7	**98.35**	**94.0**	**97.63**	135,055
Salinas	200	3x3	84.66	87.99	82.94	11,630
		5x5	86.96	89.81	**85.46**	23,842
		7x7	**86.63**	**90.94**	85.09	39,600
	400	3x3	88.72	89.91	85.07	12,138
		5x5	88.52	91.69	**87.08**	27,403
		7x7	**88.28**	**92.58**	86.8	45,557

4.4.5 Effect of Window Size

As mentioned earlier, an $n * n$ neighborhood region is extracted for each pixel and fed into the network. The effect of size of neighborhood region is observed in Table 4.6 . The window sizes are chosen as 3x3, 5x5, and 7x7. Choosing a larger neighborhood region increases the input data size. This leads to a greater network and thus more trainable parameters. For example, with a 5*5*4 data size, number of network parameters becomes 1,653,284. On the other hand, for a 7*7*4 data size, this number increases to 2,643,332.

The experiments show that larger spatial region gives better classification accuracy. For Salinas dataset, increasing window region from 5x5 to 7x7 has a slight effect in the accuracy. Since it has a higher spatial resolution than other datasets, taking larger neighborhood results in assimilation of classes with small region. As the size of neighborhood region increases, the computation time of both algorithms linearly increases. Compared to other parameters, this has a major effect on training time. This can be explained with the increase in the network size.

4.4.6 Effect of Number of PCs

Although hyperspectral data consists of hundreds of channels, they can be sufficiently represented by several principal components. In the literature, up to 6 PCs are deployed for the hyperspectral data classification. In Table 4.2, the spectral information stored for different number of PCs can be observed. Although increasing number of PCs provides more spectral information, it affects computation time of the classifier negatively as can be seen in Table 4.7. Increasing PC numbers brings a wider network and this network requires longer time to train.

4.4.7 Effect of Data Augmentation

In order to search the impact of increasing training data virtually, number of training data is increased with varying ratios (4, 10, and 50). This is achieved by linearly combining 2 samples and adding random noise. The evident increase in classification

Table 4.7: Effect of PC Numbers on Classification Accuracy for All Datasets (window size = 7x7)

Datasets	PCs	OA	AA	κ	Time (sec)
Pavia University	2	81.26	81.56	75.53	3,334
	3	80.9	82. 16	75.33	4,468
	4	**81.72**	**82.34**	**76.35**	5,606
Pavia Centre	2	90.91	84.67	87.3	58,495
	3	98.15	93.69	97.35	102,506
	4	**98.35**	**94.0**	**97.63**	135,055
Salinas	2	82.57	88.88	80.6	6,616
	3	87.4	91.45	**85.9**	9,077
	4	**86.87**	**92.67**	85.3	14,256

performance proves that the network better generalizes with data augmentation. In Table 4.8, a satisfying performance increase is observed with adding virtual data. A 5% increase in overall accuracy of Salinas dataset is obtained, which is more effective than all other parameters. For Pavia University and Pavia Centre datasets, this increase equals to 9% and 1%, respectively. Since Pavia Centre results are almost saturated, this has a slighter affect on its performance. Computation time increases by 12.6, 16, and 3.42 times for these datasets, respectively.

Confusion matrix of experiments with Pavia University is given in Table 4.9. It is seen that asphalt, bare soil and meadows classes have the smallest class accuracy. Similarity between meadows and bare soil classes results in this poor performance.10% of meadows and 15% of bare soil samples are mislabeled as each other. Moreover, 18.83% of asphalt samples are misclassified as bitumen, bricks and gravel. When class signatures are considered (Figure 4.2), extracted features suffer from lack of details to discriminate these classes. This indicates the deficiency of preprocessed data and proposed algorithm in discriminating similar classes.

Confusion matrix of experiments with Pavia Centre dataset is given in Table 4.10. Tree and asphalt signatures resemble each other and CapsNet misclassifies them as

Table 4.8: Effect of data augmentation on classification Accuracy for all datasets, window size =7x7

Datasets	Sample Size After Augmentation	OA	AA	κ	Time (sec)
Pavia University	200	81.72	82.34	76.35	5,606
	800	79.41	82.56	73.86	8,332
	2,000	83.66	86.63	79.27	13,730
	10,000	**90.67**	**89.67**	**87.86**	90,032
Pavia Centre	200	97.93	93.04	97.05	130,797
	800	97.34	91.44	96.21	196,317
	2,000	98.63	95.19	98.06	263,701
	10,000	**98.94**	**95.94**	**98.49**	544,716
Salinas	200	86.63	90.94	85.09	39,600
	800	88,16	92,3	86.81	80,111
	2,000	91.22	95.22	90.17	142,094
	10,000	**91.41**	**95.71**	**90.38**	270,097

Table 4.9: Confusion matrix of experiments using augmented data (x50) on Pavia University dataset

	Predictions								
Classes	Asphalt	Meadows	Gravel	Trees	Metal s.	B. Soil	Bitumen	Bricks	Shadows
Asphalt	5,021	11	267	8	0	2	693	251	9
Meadows	0	15,191	6	261	0	1,907	0	0	0
Gravel	14	0	1,732	0	0	0	5	66	0
Trees	3	4	0	2,813	0	1	1	0	1
Metal sheets	0	0	0	0	1,145	0	0	0	0
Bare Soil	24	722	9	2	5	4,059	7	1	0
Bitumen	7	0	3	0	0	0	1,120	0	0
Bricks	4	1	107	1	0	2	6	3,360	1
Shadows	0	0	0	0	0	0	0	0	747
Class Acc. (%)	80.18	87.48	95.32	99.65	100	84.05	99.12	96.5	100

Table 4.10: Confusion matrix of experiments using augmented data (x50) on Pavia Centre dataset

	Predictions								
Classes	Water	Trees	Asphalt	Bricks	Bitumen	Tiles	Shadows	Meadows	B. Soil
Water	65,116	0	0	0	0	53	0	0	0
Trees	0	7,090	244	0	0	0	0	0	0
Asphalt	0	110	2,626	0	0	0	0	4	0
Bricks	0	1	0	2,449	18	14	2	1	0
Bitumen	0	0	2	7	6,358	1	2	0	0
Tiles	0	0	0	14	2	8,939	19	5	0
Shadows	0	0	0	153	2	168	6,763	1	0
Meadows	0	0	8	19	1	53	3	42,067	28
B. Soil	0	0	0	0	0	2	0	12	2,643
Class Acc. (%)	99.92	96.67	95.84	98.55	99.67	99.55	95.43	99.73	99.47

each other. 3.3% of tree samples are classified as asphalt. 3.8% of asphalt samples are categorized as tree. Providing more spectral information is essential to avoid this problem. Moreover, 4.6% of shadow samples are labeled as bricks and tiles. Remaining classes have achieved more than 99% precision in classification.

4.4.8 Comparing with CNN

A convolutional neural network is implemented to compare with our proposed method. The network consists of 1 convolutional layer, 1 max pooling layer, and 2 fully-connected layers, respectively. Parameters of the convolutional layer are kept same as in CapsNet. First fully-connected layer has 32 nodes and the other layer has as many nodes as the number of classes. For both methods, 400 samples are used in training. Results obtained with CNN are given in Table 4.11. It is observed that CNN outclasses CapsNet for all datasets in terms of accuracy and computation time.

The experiments are repeated with augmented training data. In this case, number of training samples per class has increased to 10,000. In Table 4.12, it is seen that CapsNet performs slightly better than CNN. However, capsule networks have much longer training time with respect to CNN. This proves that CapsNet has a higher learning capacity when sufficient data is provided.

Table 4.11: Comparison with CNN using 400 training samples each class (window size = 7x7)

Datasets	Method	OA	AA	κ	Time (sec)
Pavia University	CNN	**89.86**	**89.44**	**86.48**	774
	CapsNet	80.16	85.14	74.93	6,055
Pavia Centre	CNN	**98.66**	**95.05**	**98.08**	2,477
	CapsNet	98.35	94.0	97.63	135,055
Salinas	CNN	**91.15**	**94.8**	**90.05**	998
	CapsNet	88.28	92.58	86.8	45,557

Table 4.12: Comparison with CNN using data augmentation (window size = 7x7)

Datasets	Method	OA	AA	κ	Time (sec)
Pavia University	CNN	89.31	87.62	86.04	4,639
	CapsNet	**90.67**	**89.67**	**87.86**	90,032
Pavia Centre	CNN	98.26	93.82	97.53	6,426
	CapsNet	**98.94**	**95.94**	**98.49**	544,716
Salinas	CNN	**92.17**	95.2	**91.24**	8,054
	CapsNet	91.41	**95.71**	90.38	270,097

Table 4.13: Confusion matrix of experiments with CNN using augmented data (x50) on Pavia University dataset

	Predictions								
Classes	Asphalt	Meadows	Gravel	Trees	Metal s.	B. Soil	Bitumen	Bricks	Shadows
Asphalt	5,742	4	127	8	7	5	253	97	19
Meadows	1	15,915	0	243	0	1148	0	58	0
Gravel	52	15	1,606	3	2	5	12	121	1
Trees	7	9	0	2,798	4	1	3	0	1
Metal sheets	0	0	0	0	1,145	0	0	0	0
Bare Soil	0	717	0	1	0	4,097	8	6	0
Bitumen	19	0	0	0	3	0	1,105	0	3
Bricks	69	26	147	3	15	23	23	3,166	0
Shadows	0	0	0	0	0	0	0	0	747
Class Acc. (%)	91.7	91.65	88.39	99.11	100	84.84	97.79	90.92	100

In Table 4.13, confusion matrix of experiments on Pavia University dataset using augmented data can be seen. When compared to the ones with CapsNet, CNN classifies asphalt samples better compared to CapsNet. However, gravel and bricks are better recognized with CapsNet. CNN algorithm has difficulty in discriminating gravel and bricks. Both algorithms have classified trees, metal sheets and shadows with almost no error. This indicates that improvements must be on extracting higher-level features in order to better separate similar classes. Moreover, applying PCA on samples is another cause of this problem.

In Table 4.14, confusion matrix of experiments on Pavia Centre dataset using augmented data is shown. Similar to CapsNet; trees, bricks and shadows classes have the smallest class accuracy. 6% of asphalt samples are classified as tree and 3.8% of tree samples are classified as asphalt. 7.7% of shadow samples are misclassified as bricks and tiles. Although same data representation is utilized in both methods, CapsNet makes better use of the available data.

Table 4.14: Confusion matrix of experiments with CNN using augmented data (x50) on Pavia Centre dataset

	Predictions								
Classes	Water	Trees	Asphalt	Bricks	Bitumen	Tiles	Shadows	Meadows	B. Soil
Water	64,989	0	0	0	0	180	0	0	0
Trees	0	6,887	446	0	0	0	0	1	0
Asphalt	0	111	2,605	0	10	4	0	10	0
Bricks	0	2	1	2,330	139	11	2	0	0
Bitumen	0	1	12	32	6,319	0	0	6	0
Tiles	9	25	3	56	66	8,708	72	30	10
Shadows	0	10	0	345	60	199	6,418	54	1
Meadows	0	21	7	53	253	91	0	41,719	35
B. Soil	0	5	12	0	5	7	0	7	2,621
Class Acc. (%)	99.72	93.91	95.07	93.76	99.2	96.98	90.56	97.63	98.65

CHAPTER 5

CONCLUSION

In this thesis, we have addressed the problem of hyperspectral data classification by introducing capsule networks. This network originates from CNN and aims to solve its shortcomings. It encapsulates the state of extracted features in a vector form and utilizes this information for a better classification. The network originally takes 2D images as input and is modified as to take 1D hyperspectral data. First layer of capsule networks (convolutional layer) is adapted to apply 1D convolutions. After combining spatial and spectral information, data becomes 3D and it is flattened to 1D to feed the network. After that, proposed method is tested on Pavia University, Pavia Centre, and Salinas datasets and proved its success by overall accuracies 90.67%, 98.94% ve 91.41%, respectively. The same dataset is applied on a CNN framework with a similar structure. When 400 training samples are used for each class, CNN exhibits better performance in terms of accuracy and training time. However, CapsNet outclasses CNN when augmented data is used at training. Confusion matrices show that CapsNet is more successful at distinguishing similar classes. Nonetheless, it is thought that better results can be obtained with some modifications. During the tests, increasing neighborhood region size and number of PCs apparently increased the classifier performance. It is concluded that data content is not sufficient for capsule network to characterize the classes. On the light of this assumption, giving all the spectral bands as input may help characterize the classes and better discriminate them. Combining full spectral bands with an $n * n$ window region increases the data size by a vast amount and therefore lead to curse of dimensionality. On the other hand, feeding all spectral bands to the network without including spatial information led to poor performance. For this reason, it is predicted that utilizing spatial information after classification such as segmentation would be more practical. Furthermore,

affine transform matrices learned during training can be utilized in creating virtual samples. This matrix stores relationship between basic and complex features. Lastly, spectral data can be divided into groups and trained in different networks as a solution to the high-dimensionality problem. Contributions of these modifications can be investigated as a future work.

REFERENCES

[1] R. Pu, *Hyperspectral Remote Sensing*. Taylor and Francis Group, 2017.

[2] Concerning the light. remote sensing basics. [Online]. Available: http://www.50northspatial.org/concerning-the-light/

[3] C. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*, ser. Hyperspectral Imaging: Techniques for Spectral Detection and Classification. Springer US, 2003, no. 1. c. [Online]. Available: http://books.google.com.tr/books?id=JhBbXwFaA6sC

[4] H. Grahn and P. Geladi, *Techniques and Applications of Hyperspectral Image Analysis*. Wiley, 2007. [Online]. Available: http://books.google.com.tr/books?id=DqmWQk01mlIC

[5] L. Ma, M. Crawford, and J. Tian, "Local manifold learning-based k -nearest-neighbor for hyperspectral image classification," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 48, no. 11, Nov 2010.

[6] Introduction to remote sensing. [Online]. Available: http://employees.oneonta.edu/baumanpr/geosat2/RS-Introduction/RS-Introduction.html

[7] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 354–379, April 2012.

[8] N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE Signal Processing Magazine*, vol. 19, no. 1, pp. 44–57, Jan 2002.

[9] D. G. Manolakis, D. Marden, and G. A. Shaw, "Hyperspectral image processing for automatic target detection applications," 2003.

[10] G. Mercier and M. Lennon, "Support vector machines for hyperspectral image classification with spectral-based kernels," in *IGARSS 2003. 2003 IEEE International Geoscience and Remote Sensing Symposium. Proceedings (IEEE Cat. No. 03CH37477)*, vol. 1. IEEE, 2003, pp. 288–290.

[11] D. Stein, S. G. Beaven, L. E. Hoff, E. M. Winter, A. P. Schaum, and A. Stocker, "Anomaly detection from hyperspectral imagery," *Signal Processing Magazine, IEEE*, vol. 19, pp. 58 – 69, 02 2002.

[12] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[13] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 3288–3291.

[14] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Convolutional neural network committees for handwritten character classification," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, 2011, pp. 1135–1139.

[15] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[16] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[17] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, vol. 2015, pp. 1–12, 07 2015.

[18] J. Yue, W. Zhao, S. Mao, and H. Liu, "Spectral spatial classification of hyperspectral images using deep convolutional neural networks," *Remote Sensing Letters*, vol. 6, no. 6, pp. 468–477, 2015.

[19] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks,"

IEEE Transactions on Geoscience and Remote Sensing, vol. 54, no. 10, pp. 6232–6251, 2016.

[20] J. Yang, Y. Zhao, J. C. W. Chan, and C. Yi, "Hyperspectral image classification using two-channel deep convolutional neural network," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 7 2016, pp. 5079–5082.

[21] S. Yu, S. Jia, and C. Xu, "Convolutional neural networks for hyperspectral image classification," *Neurocomputing*, vol. 219, no. Supplement C, pp. 88 – 98, 2017.

[22] W. Li, G. Wu, F. Zhang, and Q. Du, "Hyperspectral image classification using deep pixel-pair features," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 844–853, 2 2017.

[23] Y. Li, H. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery with 3d convolutional neural network," *Remote Sensing*, vol. 9, p. 67, 01 2017.

[24] H. Lee and H. Kwon, "Going deeper with contextual cnn for hyperspectral image classification," *IEEE Transactions on Image Processing*, vol. 26, no. 10, pp. 4843–4855, 10 2017.

[25] W. Zhao and S. Du, "Spectral - spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4544–4554, 8 2016.

[26] Y. Luo, J. Zou, C. Yao, X. Zhao, T. Li, and G. Bai, "Hsi-cnn: A novel convolution neural network for hyperspectral image," in *2018 International Conference on Audio, Language and Image Processing (ICALIP)*, 7 2018, pp. 464–469.

[27] M. E. Paoletti, J. M. Haut, J. Plaza, and A. G. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, no. Part A, pp. 120–147, 2017.

[28] F. Z. Alan J.X. Guo, "Spectral-spatial feature extraction and classification by ann supervised with center loss in hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–13, 9 2018.

[29] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2094–2107, 6 2014.

[30] W. Zhao, Z. Guo, J. Yue, X. Zhang, and L. Luo, "On combining multiscale deep learning features for the classification of hyperspectral remote sensing imagery," *International Journal of Remote Sensing*, vol. 36, no. 13, pp. 3368–3379, 2015.

[31] Y. Liu, G. Cao, Q. Sun, and M. Siegel, "Hyperspectral classification via deep networks and superpixel segmentation," *International Journal of Remote Sensing*, vol. 36, no. 13, pp. 3459–3482, 2015.

[32] J. Yue, S. Mao, and M. Li, "A deep learning framework for hyperspectral image classification using spatial pyramid pooling," *Remote Sensing Letters*, vol. 7, no. 9, pp. 875–884, 2016.

[33] C. Xing, L. Ma, and X. Yang, "Stacked denoise autoencoder based feature extraction and classification for hyperspectral images," *Journal of Sensors*, vol. 2016, pp. 1–10, 01 2016.

[34] X. Ma, H. Wang, and J. Geng, "Spectral spatial classification of hyperspectral image based on deep auto-encoder," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 9, pp. 4073–4085, 9 2016.

[35] Y. Chen, X. Zhao, and X. Jia, "Spectral spatial classification of hyperspectral data based on deep belief network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2381–2392, 6 2015.

[36] M. E. Midhun, S. R. Nair, V. T. N. Prabhakar, and S. S. Kumar, "Deep model for classification of hyperspectral image using restricted boltzmann machine," in *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*, ser. ICONIAAC '14. New York, NY, USA: ACM, 2014, pp. 35:1–35:7.

[37] T. Li, J. Zhang, and Y. Zhang, "Classification of hyperspectral image based on deep belief networks," in *2014 IEEE International Conference on Image Processing (ICIP)*, 10 2014, pp. 5132–5136.

[38] P. Zhong, Z. Gong, S. Li, and C. B. Schonlieb, "Learning to diversify deep belief networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 6, pp. 3516–3530, 6 2017.

[39] F. Deng, S. Pu, X. Chen, Y. Shi, T. Yuan, and S. Pu, "Hyperspectral image classification with capsule network using limited training samples," *Sensors*, vol. 18, no. 9, 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/9/3153

[40] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. Plaza, J. Li, and F. Pla, "Capsule networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 4, pp. 2145–2160, April 2019.

[41] R. E. Uhrig, "Introduction to artificial neural networks," in *Industrial Electronics, Control, and Instrumentation, 1995., Proceedings of the 1995 IEEE IECON 21st International Conference on*, vol. 1, Nov 1995, pp. 33–37 vol.1.

[42] S. Haykin, *Neural Networks: A Comprehensive Foundation*, ser. International edition. Prentice Hall, 1999.

[43] M. R. Veronez, S. Florêncio de Souza, M. T. Matsuoka, A. Reinhardt, and R. Macedônio da Silva, "Regional mapping of the geoid using gnss (gps) measurements and an artificial neural network," *Remote Sensing*, vol. 3, no. 4, pp. 668–683, 2011. [Online]. Available: https://www.mdpi.com/2072-4292/3/4/668

[44] A. C. Ian Goodfellow, Yoshua Bengio, *Deep Learning*. MIT Press, 2016.

[45] Convolutional neural networks. [Online]. Available: http://www.khshim.com/archives/681

[46] A beginner's guide to understanding convolutional neural networks part 2. [Online]. Available: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

[47] A. Andrej Karpathy. (2015) Cs231n convolutional neural networks for visual recognition. [Online]. Available: http://cs231n.github.io/convolutional-networks/

[48] "Hyperspectral remote sensing scenes
," http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes, [Accessed: 21-February-2018].

[49] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 3856–3866.

[50] A keras implementation of capsnet in nips2017 paper dynamic routing between capsules. [Online]. Available: https://github.com/XifengGuo/CapsNet-Keras