Heaven’s Light is Our Guide

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Rajshahi University of Engineering & Technology, Bangladesh

Spectral Spatial Classification of Hyperspectral Image Based on Stacked Autoencoder and Convolutional Neural Network

Author

Shafia Afrin

Roll: 1503022

Department of Computer Science & Engineering

Rajshahi University of Engineering & Technology

Supervised by

Prof. Dr. Boshir Ahmed

Department Head

Department of Computer Science & Engineering

Rajshahi University of Engineering & Technology

ACKNOWLEDGEMENT

First and foremost, praises and thanks to the Almighty Allah, for His showers of blessings throughout my research work to complete the research successfully.

I would like to express my deep and sincere gratitude to my research supervisor, Dr. Boshir Ahmed, Professor and Head of Department of Computer Science & Engineering, Rajshahi University of Engineering and Technology, Rajshahi, for giving me the opportunity to do research and providing invaluable guidance throughout this research. His dynamism, vision, sincerity and motivation have deeply inspired me to do better every day. He has taught me the methodology to carry out the research and to present the research works as clearly as possible. It was a great privilege and honor to work and study under his guidance. I am extremely grateful for what he has offered me. His continuous support was the most necessary tool that helped us to achieve the result.

I am also grateful to all the respective teachers of Department of Computer Science & Engineering, Rajshahi University of Engineering and Technology, for all the valuable suggestions and inspirations that helped me along the way.

I am extremely grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. I would also like to thank all my classmates, seniors and well-wishers for their constant inspiration and support.

Date: February, 2021

RUET, Rajshahi

Shafia Afrin

Heaven's Light is Our Guide

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Rajshahi University of Engineering & Technology, Bangladesh

CERTIFICATE

*This is to certify that this thesis report entitled "**Spectral Spatial Classification of Hyperspectral Image Based on Stacked Autoencoder and Convolutional Neural Network**" submitted by **Shafia Afrin, Roll: 1503022** in partial fulfillment of the requirement for the award of the degree of Bachelor of Science in Computer Science & Engineering of Rajshahi University of Engineering & Technology, Bangladesh is a record of the candidate own work carried out by him under my supervision. This thesis has not been submitted for the award of any other degree.*

Supervisor	External Examiner
--	--
Prof. Dr. Boshir Ahmed	
Professor, Head of Department	
Department of Computer Science & Engineering	
Rajshahi University of Engineering &Technology	
Rajshahi-6204	

ABSTRACT

Hyperspectral image (HSI) classification is the latest trend in the remote sensing community. Effectively extracting features from Hyperspectral images and classifying them still remains a big challenge. Over the years many dimension reduction methods are applied on Hyperspectral data but most of the classic approaches are linear in nature. With the advent of deep learning we have come to see deep learning methods performs surprisingly well in feature extraction and classification. That's why in this paper a new deep learning framework of spectral-spatial feature extraction and HSI classification is introduced. Firstly, the model of Stacked Autoencoder was exploited in our proposed hybrid network to extract various kinds of features. Secondly, the extracted features were fed to a 2D-CNN model that take both the spatial and spectral features into account. The experimental results show that this framework achieves the highest classification accuracy of 99.4% on KSC dataset. This is highest among all conventional methods, outperforming classical classifiers such as PCA-based SVM.

Index Terms—Convolutional neural network (CNN), deep feature extraction, dimension reduction hyperspectral image classification, Kennedy Space Centre (KSC), Principal Component Analysis (PCA), Stacked Autoencoder, Support Vector Machine (SVM).

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Overview

Hyperspectral remote sensing is a relatively new technique for the detection and identification of rocks, terrestrial plants, and man-made materials and histories. Researchers worldwide are investigating on this. That's why several techniques are being developed for feature extraction and classification in Hyperspectral data. In this chapter we will discuss about the Hyperspectral data and address the challenges that come with it. Later the motivation and objectives for this research will be discussed.

1.2 Remote Sensing

The history of Satellite remote sensing can be traced back to the early days of both Russian and American programs in the space era. It actually started as a dual approach of imaging surfaces high from spacecrafts using several types of sensors. After World War II, in 1946, Germans used automated still or movie camera [1].

Remote sensing is the combination of two words – remote and sensing. Here remote means something that is distant and sensing means to get information. The physical characteristics of a region can be detected and tracked from a distance by measuring its reflected and emitted radiation (normally from satellite or aircraft). Special cameras capture images that are remotely sensed, allowing researchers "sense" things about the Planet.

There are mainly two types of Remote sensing – active or passive. Active sensors are those that use internal stimuli of itself to collect data about Earth. For example, a laser-beam remote sensing system projects a laser onto the surface of Earth and tests the time that it takes for the laser to return back to its sensor. Passive sensors record natural energy that is reflected or emitted from the Earth's surface. Passive sensors mostly records reflected sunlight.

Hyperspectral Radiometer is a good example of passive sensor. It is an advanced multispectral sensor and it detects hundreds of very wide range of electromagnetic spectrum. It is the very high spectral resolution of this sensors that makes it possible to distinguish well between individual goals based on their spectral responses in each of the narrow bands.

1.3 Hyperspectral Remote Sensing

The concept of Hyperspectral remote sensing began in the mid-80. Historically it has been mostly used by geologists for the purpose of mapping of minerals. Aircraft often took Hyperspectral images of the surface of the Earth for various applications.

Hyperspectral imaging uses a device called spectrometer for collecting spectral information. This device is known as the Hyperspectral camera. We measure thousands or hundreds of spectra with a Hyperspectral camera. So, the information collected by a Hyperspectral imaging system is known as Hyperspectral cube or Hyper Cube, as it is actually three-dimensional and has volume. Along with height and width it has depth component associated with the wavelength at which each photograph was shot. That means that the Hyperspectral cube is simply a stack of several images.

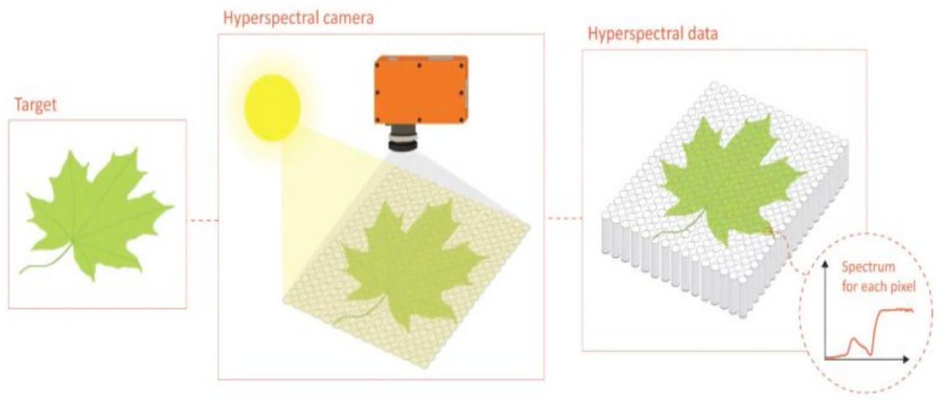


Fig. 1.1: How a Hyperspectral camera produces three dimensional data of target[2]

For example, a Hyperspectral cube of Moffett Field, California is shown in Fig. 1.2, here we can observe the landscape at different wavelengths by looking at the top face of the stack and gradually moving downward.

Hyperspectral imaging finds a wide range of applications in remote sensing. But it comes with many challenges too.

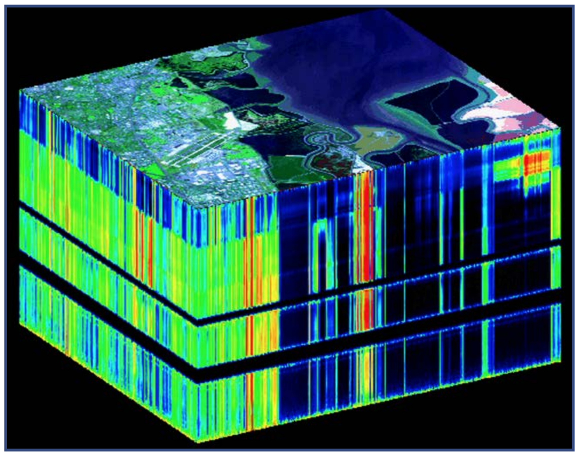


Fig. 1.2: A Hyperspectral cube of Moffett Field, California[3]

1.4 Challenges in Hyperspectral Image Classification

In the remote sensing community, the term classification is used to denote the process that assigns individual pixels to a previously known set of classes[4]. As Hyperspectral images contain a huge number of spectral bands so every pixel contains a very large amount of information. It can rise the classification complexity exponentially. It is challenging to develop reliable and efficient method for classification of hyperspectral data. Some of these challenges are described here –

- **Hughes Phenomenon:** With the increase of dimensions in hyperspectral images, for classification task the number of training samples needs to be increased. Otherwise, good parameter estimation accuracy is very difficult to ensure [5]. As the amount of spectral bands increases Classification accuracy increases gradually in the beginning, but decreases dramatically when the band number reaches some value. This phenomenon is referred as Hughes phenomenon or curse of dimensionality [6].

1h

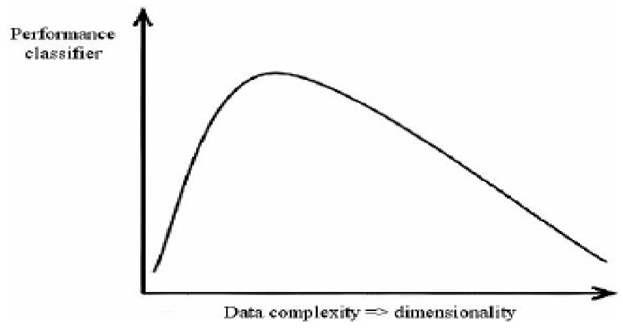


Fig. 1.3: Illustration of Hughes phenomenon[7]

- **High Data Correlation:** The dimensionality of spectral bands wouldn't create much obstacle if they were not highly correlated. In hyperspectral data cube, information can be overlapped among multiple spectral bands. As a result, there are many redundant bands in the data set. This high correlation can effect classification effectively. Figure: 1.4 shows the correlation matrix where the bright diagonal bright color indicates high correlation among spectral band-1 to band-40 in the first block. The dark block diagonal indicates the very low correlation among spectral band-42 to band-80 and so on. So, the data set contains a high degree of redundancy. In hyperspectral data, many of the adjacent pixels are often spatially correlated. So spatial correlation also exists in hyperspectral data cube.

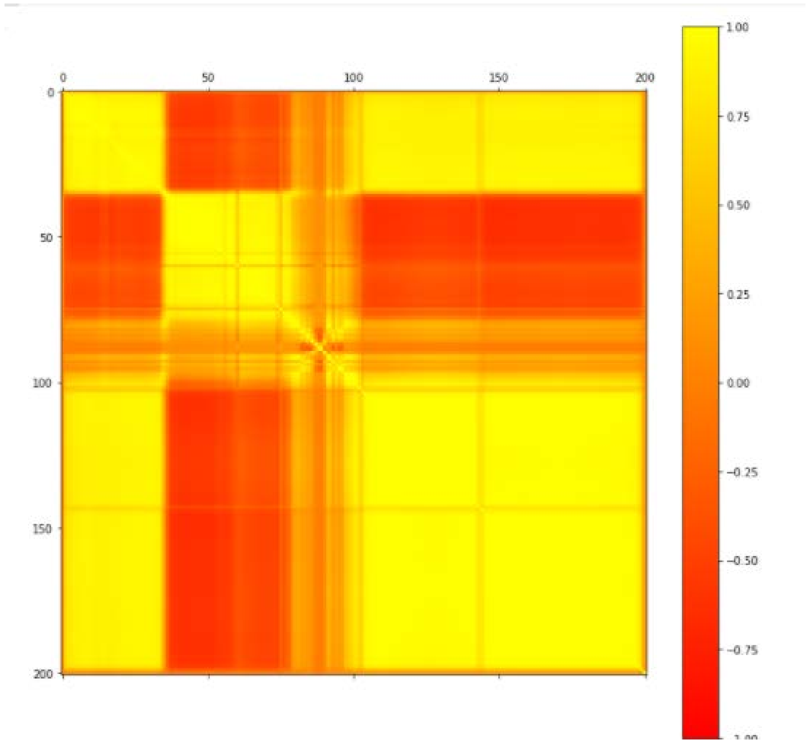


Fig. 1.4: Correlation map in Indian Pines dataset

- **Limited Training Samples:** In most HSI classification scenario, we can only work with one hyperspectral image for training a model. The reason is that creating multiple HSI of one area is computationally costly. That's why, creating a functional deep learning model that can properly classify hyperspectral data is very difficult.

- **Cost and Complexity:** Fast computers, sensitive detectors, and large data storage capacities are needed for analyzing hyperspectral data. Significant data storage capacity is necessary since uncompressed hyperspectral cubes are large, multidimensional datasets, potentially exceeding hundreds of megabytes. All of these factors greatly increase the cost of acquiring and processing hyperspectral data.[8]

1.5 Motivation

It is undeniable that, Hyperspectral data analysis is one of the fastest growing research field in today's world. Hyperspectral data contains huge amount of information and with the development of different deep learning methods our ability to correctly classifying them is also increasing.

In recent years we can see a rise of using deep learning methods in Hyperspectral images. But for feature extraction in hyperspectral images we still widely use linear feature extraction methods like- PCA, ICA, LDA. But in real world not all data is linear in manner. In nonlinear data these methods mentioned above perform very poorly.

That's why we need deep feature extraction methods that can work effectively on non-linear data. The motivation behind my research is –

- ❖ To effectively & efficiently reduce dimensions by Stacked Autoencoder.
- ❖ To Compare extracted features by Autoencoder with popular linear feature extraction technique - Principal Component Analysis or PCA
- ❖ Using Convolutional Neural Network for Classification

1.6 Outline

The report is organized in 8 chapters that discusses all the related topics needed for understanding the research work. The outline of the chapter are briefly discussed below:

- ❖ **Chapter 1**: Firstly, this chapter gives a short overview on our research field and problem. After that it tells us the changes in this research field and what motivated this research.
- ❖ **Chapter 2**: This chapter gives us the basics of Hyperspectral image and its use in several fields. Secondly, it gives us an insight in several popular methods.

- **Chapter 3:** In this chapter we talk about Deep Learning and Neural network. Deep Learning is the backbone of our prosed two networks.
- **Chapter 4:** In this chapter we talk about dimension reduction. Firstly this chapter gives us a theoretical background on Autoencoders and how they are better than PCA.
- **Chapter 5:** In this chapter we will talk about classification methods. Terminology and architecture of Non-linear Support Vector Machine, 2D-CNN, 3D-CNN will be discussed here.
- **Chapter 6:** The methodology and dataset is described in this chapter. The selected architecture of the Autoencoder and the CNNs are also discussed.
- **Chapter 7:** In this chapter Experimental results are shown & they are compared with several benchmark methods.
- **Chapter 8:** Conclusion of this research is drawn in this chapter. Drawbacks of the proposed method and possible ways of improvement is also discussed.

1.7 Conclusion

This chapter gave us a peak through what we will be doing in this research. In the later chapters the details will be discussed.

Chapter 2
Hyperspectral Imaging

2.1 Introduction

Our human eye has only three color receptors - blue, green and red. So, human eye can’t see in most of the spectrum of light. With spectral imaging technology we can detect what our eyes can't see. This technology enables us to collect and process information from across wide range of electromagnetic spectrum.

Every surface and material has its own specific fingerprint. This enables us to distinguish between surfaces that look totally similar to the human eye, such as- regular or fake banknotes, coffee beans or small stones, healthy skin spots, melanoma or land cover from far distance.

Unlike ordinary camera spectral imaging encompasses a wide variety of techniques that go beyond RGB (red green, blue light). Spectral imaging may obtain data from infrared, the visible spectrum, the ultraviolet, x-rays, or some combination of the other spectrum.

Hyperspectral imaging is a subcategory of spectral imaging. In hyperspectral imaging, a complete spectrum is collected for every pixel in a 2D-image plane.

2.2 Hyperspectral Image

Hyperspectral remote sensing combines imaging and spectroscopy in a single system. It collects hundreds of images at different set of wavelengths. Hyperspectral data sets are generally composed of about 100 to 200 spectral bands of relatively narrow bandwidths (5-10 nm).It is typically collected (and represented) as a data cube or hypercube with spatial information collected in the X-Y plane, and spectral information represented in the Z-direction.

While spectrometer records a reflectance spectrum for each pixel in the image, the measurements are made at many narrow contiguous wavelength bands, resulting in a complete spectrum for each pixel. In Figure-2.3, we can see can see, an aircraft is creating a hyperspectral image data cube of a land cover using special Hyperspectral camera. Here it is recording spectral reflectance of all the elements that are present in the scene like – tree, water, and land in various bands of the spectrum based on their spectral reflectance. From the graph attached beside we can see that for each element reflectance band is different.

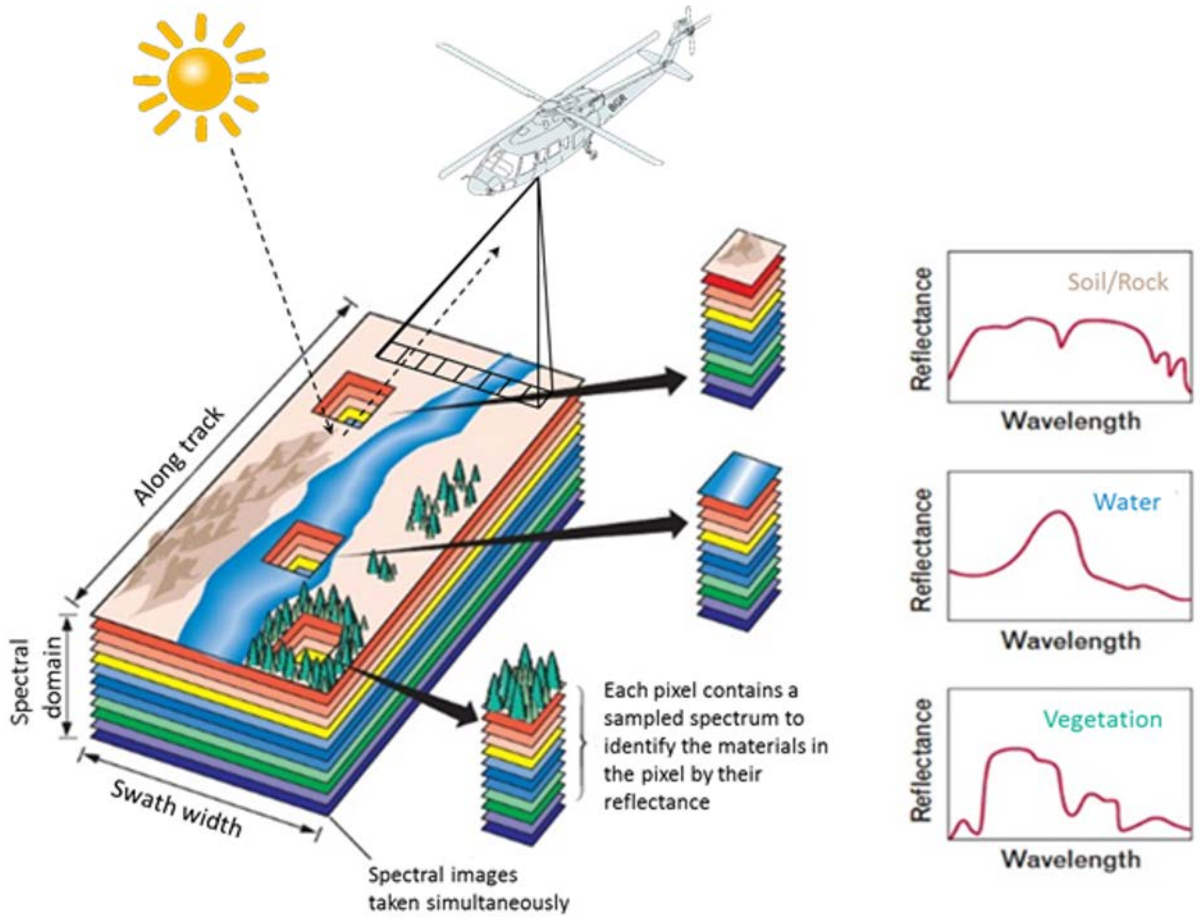

Along track
Spectral domain
Swath width
Spectral images taken simultaneously
Each pixel contains a sampled spectrum to identify the materials in the pixel by their reflectance
Soil/Rock
Water
Vegetation
Reflectance
Wavelength

Fig. 2.1 : Different elements in a scene producing different set of spectrum[9]

2.3 Application of Hyperspectral Images

Hyperspectral remote sensing is used in a wide array of applications. Although originally was developed for mining and geology, it has now spread into fields as widespread as ecology and surveillance, as well as historical manuscript research. Some application of hyperspectral image is summarized below:

- **Target Detection:** Involves distinction between targets from similar backgrounds and locating examples of targets that are smaller than the nominal pixel size.
- **Remote Sensing:** In remote sensing technology, distinguishing earth surface features is very important because each feature has a different spectrum band. Multi-spectral satellite can capture images up to a few bands, e.g. Landsat 7 with 8 bands. But multi-spectral imaging satellites can capture Earth's surface in more than 200 bands, enabling scientists to differentiate objects that could not be detected with multi-spectral data.
- **Biotechnology:** In biological and medical applications, hyperspectral technology has become popular. They are mostly used in the study of wound analysis, fluorescence microscopy and cell biology [10].
- **Agriculture**: For specific crops and in specific climates, usage of hyperspectral remote sensing is increasing for monitoring the development and health of crops. In Australia, research is going on to use imaging spectrometers to detect grape variety and develop an early warning system for disease outbreaks[11].
- **Pharmaceuticals:** Hyperspectral imaging technique is being widely used to enhance the quality control. It is used widely for controlling the counterfeit or illegal drugs, managing the packaging of medicine and mixing of the powder.
- **Oil Spill Detection**: Hyperspectral imaging systems carrying aircrafts and spacecrafts can detect hydrocarbons and thus plays an important role in the response to the deep water Horizon oil spill.
- **Hazardous Waste Monitoring:** data of mine waste can be rapidly analyzed to search for minerals that produce acids that pollute rivers. Pictures of mine waste can be rapidly analyzed for acid-polluting minerals. Hyperspectral imaging is also used to detect heavy metals, such as cadmium, plum and arsenic. Interestingly, hyperspectral data may also be employed to classify vegetation and the mineral composition of the underlying soil can then be inferred.
- **Food Processing:** In the food processing industry, hyperspectral imaging is used to identify and remove defects and foreign material (FM) that are invisible to traditional camera and laser sorters[12].
- **Mineralogy:** Many minerals can be identified from airborne images, and their relation to the presence of precious minerals, including gold and diamonds, is also recorded.

Currently, many research are currently underway to understand the relationship between oil and gas leakages from pipelines and natural wells, and their effects on the vegetation and the spectral signatures.

- **Chemical Imaging**: A wide range of chemical threats can be exposed to soldiers. Mostly invisible but through hyperspectral imaging technology detectable are these risks. This was demonstrated in distance up to 5 km by the Hyper-Cam Telops, introduced in 2005[8].
- **Astronomy:** In astronomy hyperspectral imaging is widely used to determine a spatially-resolved spectral image. Since a spectrum is an important diagnostic for celestial components, having a spectrum for each pixel allows more science cases to be addressed. In astronomy, this technique is used for thousand years which is commonly referred to as integral field spectroscopy.

Most important applications are mentioned here. The truth is with passing time, we are having newer and newer use of Hyperspectral Imaging technology in various field.

2.4 Hyperspectral Image Classification

In the remote sensing community, the term classification is used to denote the process that assigns individual pixels to a set of classes. The output of the classification step is known as the classification map. With respect to the availability of training samples, classification approaches can be split into two categories: supervised and unsupervised classifiers.

- **Unsupervised Classification**: In order to deal with Hughes phenomenon, feature extraction methods are applied to reduce the dimensionality by selecting only the prominent features that represents the dataset well. In unsupervised methods, the algorithm or method automatically groups pixels with similar spectral characteristics (means, standard deviations, etc.) into different clusters according to some statistical criteria. Further, unsupervised classification methods do not require any prior knowledge or label to train the data. The familiar unsupervised methods are principal component analysis (PCA) and independent component analysis (ICA).
- **Supervised Classification:** Supervised approaches classify input data for each class using a set of previously seen representative samples known as training samples. The power of classification increases with the increase of available training samples. Supervised

classifier can face challenge about the imbalance between high dimensionality and incomplete accessibility of training samples or the presence of noise in the data.

The conventional workflow of supervised hyperspectral data classification is:

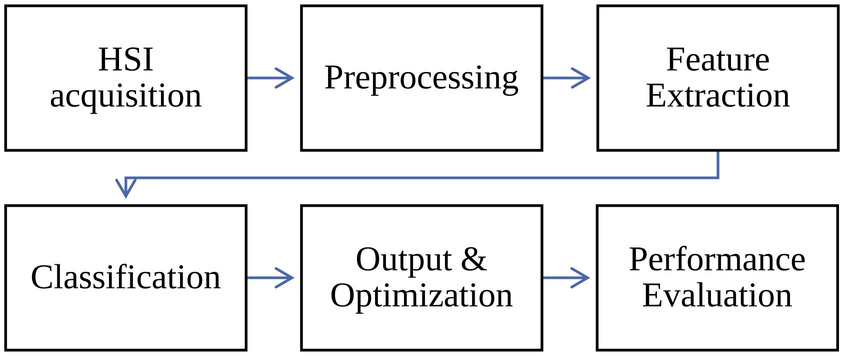


Fig. 2.2: Flowchart of HSI supervised classification process

2kahla

2.7 Conclusion

In this chapter the basics of hyperspectral Image and the applications of hyperspectral images were discussed. After that the work flow of hyperspectral classification was introduced. These information will help us understanding the upcoming chapters.

Chapter 3

Neural Network & Deep Learning

3.1 Introduction

Deep learning is a sub-division of machine learning algorithms that uses multiple layers to progressively extract higher-level features from the raw input. Artificial neural networks are the building blocks of deep learning. An Artificial Neural Network (ANN) or neural network or multilayer perceptron is a computational model which is inspired from the functions of biological neural network. Just like a neuron forms the basic element of our brain it forms the basic structure of a neural network.

3.2.1 Neurons

To describe neural networks, we will begin by describing the simplest possible neural network. It has only one 'neuron'. We will use the following diagram to denote a single neuron:

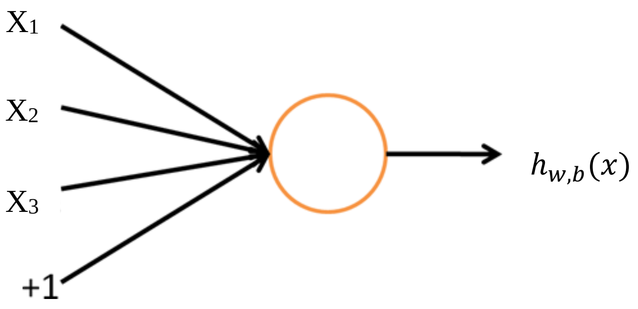


Fig. 3.1: A single neuron

This neuron is a computational unit that takes as input *x1*, *x2*, *x3* and one intercept term, and outputs,

$$h_{w,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b),$$

Where *W* and *b* are weight and bias parameters, which are initialized randomly and $f : \mathbb{R} \rightarrow \mathbb{R}$ is called the activation function.

The activation function is used for determining whether the output of neuron is "yes" or "no". It maps the result between 0 to 1 or -1 to +1, depending on the type of activation function. There are several types of activation functions which will be discussed later. Here we have used Sigmoid activation function.

$$f(z) = \frac{1}{1 + \exp(-z)}$$

Thus, our single neuron corresponds exactly to the input-output mapping defined by logistic regression.

3.2.2 Feed Forward Neural Network

A neural network is put together by hooking together many of simple "neurons," so that the output of a neuron can be the input of another. For example, here is a small neural network:

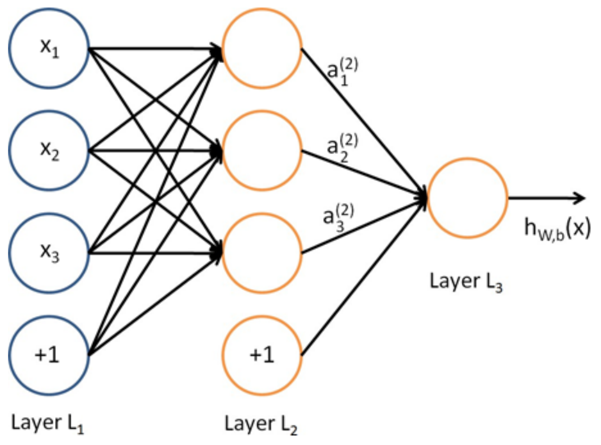


Fig. 2.2: A neural network combined of many simple neurons

In this figure, we have used blue circles to also denote the inputs to the network. The circles labeled "+1" are called bias units, and correspond to the intercept term. The leftmost layer of the network is called the input layer, and the rightmost layer the output layer (which, in this example, has only one node). The middle layer of nodes is called the hidden layer, because its values are not observed in the training set. This neural network has 3 input units (not counting the bias unit) or nodes, 3 hidden units, and 1 output unit.

Here n_l denotes the number of layers in our network; thus $n_l = 3$ in our example. We label layer l as L_l , so layer L_l is the input layer, and layer L_{nl} is the output layer.

Our neural network has parameters $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$, where we write W_{ij}^{l} to denote the parameter (or weight) associated with the connection between unit j in layer l, and unit i in layer $l+1$. Also, $b_i^{(l)}$ is the bias associated with unit i in layer $l+1$. So,W_{12}^{2} will mean a connection between unit 2 in layer 2 (here, the hidden layer) and unit 1 in layer 3(here, the output layer).

Let's assume the total number of input feature size is n_{i-1} and the hidden layer size is n_i . Since each connection is associated with a weight parameter, the number of weight parameters will be $n_{i-1} \times n_i$. Each output neuron is also associated with one bias parameter, hence the number of bias parameters is n_i. The total number of trainable parameters = $n_{i-1} \times n_i + n_i$

We will write $a_i^{(l)}$ to denote the activation (meaning output value) of unit i in layer l. For $l = 1$, we also use $a_i^{1} = x_i$ to denote the i-th input. Given a fixed setting of the parameters W, b, our neural network defines a hypothesis $h_{w,b}(x)$ that outputs a real number. Specifically, the computation that this neural network represents is given by:

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)$$

$$h_{w,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})\ldots..(5)$$

In the sequel, we also let $z_i^{(l)}$ denote the total weighted sum of inputs to unit i in layer l, including the bias term (e.g., $z_i^{(2)} = \sum_{j=1}^{n} W\, ij\; x_j + b_i^{(1)}$), from which we get,

$a_i^{(l)} = f(z_i^{(l)})$

From this it easily lends itself to a more compact notation. From equation (2-5) we can write:

$$z^{(2)}= W^{(1)}x + b^{(1)}$$

$$a^{(2)} = f(z^{(2)})$$

$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$

$$h_{w,b}(x) = a^{(3)} = f(z^{(3)})$$

More generally, recalling that we also use $a^{(1)} = x$ to also denote the values from the input layer, then given layer l's activations $a^{(l)}$, we can compute layer $l + 1$'s activations $a^{(l+1)}$ as:

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)} \ldots\ldots\ldots\ldots\text{(6)}$$

$$a^{(l+1)} = f(z^{(l+1)}) \ldots\ldots\ldots\ldots\ldots\ldots\text{(7)}$$

So, this is one example of a feed-forward neural network, since the connectivity graph does not have any directed loops or cycles.

3.2.3 Backpropagation

When training the network there are mostly two passes that need to be completed one after another. The forward pass computes the values from inputs to outputs and the backward pass performs the backpropagation which starts at the end in a recursive manner like a chain rule. It computes the gradients to the inputs of the circuit.

Let's assume we have a training set of m examples like this – $\{(x^{(1)},y^{(i)}),\ldots..(x^{(m)},y^{(m)})\}$. Here $x^{(1)}$, $x^{(2)},\ldots,x^{(m)}$ input values which will be fed to the network. On the other hand $y^{(1)},\ldots,y^{(m)}$ are the target values. After one forward pass we will calculate the error between output and target value. We will use mean square error as cost function. We will define the overall cost function as:

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{w,b}\left(x^{i}\right) - y^{i}\right\|^{2}\right)\right]$$

Our goal is to minimize $J(W,b)$ as a function of W and b. To train our neural network, we will initialize each parameter $W_i^{(l)}$ and each $b_i^{(l)}$ to a small random value near zero. It is very important to initialize all the parameters randomly, rather than 0 to all nodes. If all the start with an identical values, then all the hidden layer unit will end up learning the same function. The random initialization is known as symmetry breaking.

After iteration of gradient descent updates the parameters as follows:

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

$$b_{i}^{l} := b_{i}^{l} - \alpha \frac{\partial}{\partial b_{i}^{l}} J(W, b)$$

Here α is the learning rate. The key step here is computing the partial derivatives. The partial derivative terms can be computed as:

$$\frac{\partial}{\partial W_{ij}^{l}} J(W, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial W_{ij}^{l}} J(W, b, x^{i}, y^{i})$$

$$\frac{\partial}{\partial b_{i}^{l}} J(W, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial b_{i}^{l}} J(W, b, x^{i}, y^{i})$$

Now we are ready to describe the backpropagation algorithm.

Let's assume we are given a training example *(x, y)*, we will first run a forward pass to compute all the activations throughout the network. Thus we will get the output value of the hypothesis $h_{w,b}(x)$. Then, for each node *i* in layer *l*, we would like to compute an error term (δ_i^l) . That measures how much that node was "responsible" for any errors in our output. For an output node, we can directly measure the difference between the network's activation and the true target value, and use that to define δ_i^{nl}, where layer *nl* is the output layer). For hidden units we will compute δ_i^l based on a weighted average of the error terms of the nodes that uses a_i^l as an input.

In detail, here is the backpropagation algorithm:

1. Perform a forward pass, computing the activations for Layers L_2, L_3 and so on up to the output layer $L_{nl.}$

2. For each output unit *i* in layer *nl* (the output layer), set

$$\delta_i^{nl} = \frac{\partial}{\partial z_i^{nl}} \frac{1}{2} \|y - h_{w,b}(x)\|^2 = -\left(y_i - a_i^{nl}\right). f'(z_i^{nl})$$

3. For *l* = *nl* -1,*nl* -2 ,…,2 ,

   For each node *i* in layer *l*, set

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

4. Compute the partial derivatives :

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

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

5. Update the parameters :

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

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

3.3 Activation Functions

In a neural network inputs are fed into the neurons in the input layer. Each neuron has a weight, and multiplying the input number with the weight gives the output of the neuron, which is transferred to the next layer. The activation function is a mathematical ‘gate’ in between the input feeding the current neuron and its output going to the next layer. It can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold. Or it can be a transformation that maps the input signals into output signals that are needed for the neural network to function.

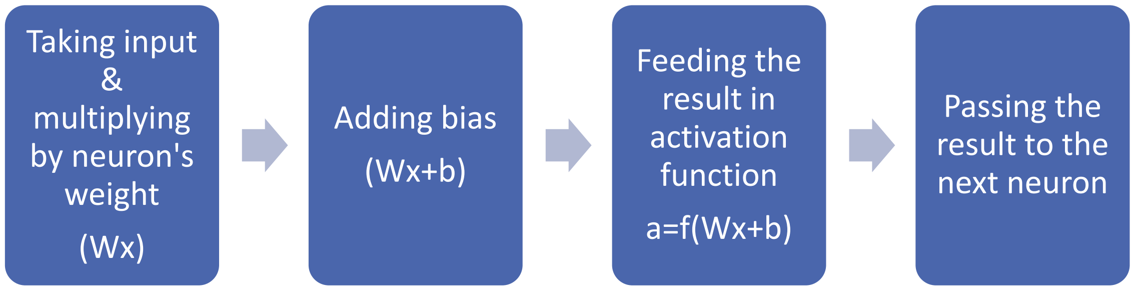

Taking input & multiplying by neuron's weight
(Wx)
Adding bias
(Wx+b)
Feeding the result in activation function
a=f(Wx+b)
Passing the result to the next neuron

Fig. 3.3: The basic process carried out by a neuron in forward pass

Modern neural network models use non-linear activation functions as they can help the network learn complex data. They also allow backpropagation because they have a derivative function which is related to the inputs. They allow ‘stacking’ of multiple layers of neurons to create a deep neural network too.

Some of the most popular non-linear activation functions are described below:

- **Sigmoid or Logistic Activation function**: Sigmoid function is one of the most popular activation functions. It can be defined as :

$$g(x) = \frac{1}{1 + e^{-x}}$$

It gives value between 0 and 1. It is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.

Its major disadvantage is for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.

- **Hyperbolic Tangent or Tanh function**: It’s also like the sigmoid function but it is better. Its range is between -1 to 1. The advantage is that the negative inputs are mapped strongly negative and the zero inputs are mapped near zero in the tanh graph. It can be defined as :

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

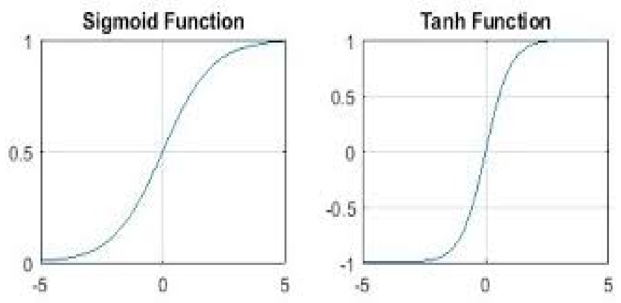


Fig. 3.4: Pictorial representation of Sigmoid and Tanh activation function[13]

- **Rectified Linear Unit (ReLU) function**: ReLU is the most used activation function in the world right now. It is computationally efficient and converges faster than any other functions. The definition of ReLU is :

$$g(x) = \max(0, x) = \begin{cases} x \; if \; x \geq 0 \\ 0 \; if \; x < 0 \end{cases}$$

Its major disadvantage is when inputs approach zero, or are negative, the gradient of the function becomes zero then the network cannot perform backpropagation and cannot learn. This problem can be solved with Leaky ReLU function.

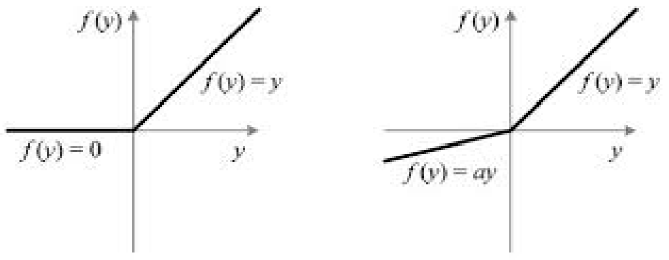


Fig. 3.5: Pictorial representation of ReLU and Leaky ReLU activation function[14]

The leak helps to increase the range of the ReLU function. Usually the value of *a* is 0.01.

- **Softmax function**: The Softmax function produces output in the range of values between 0 and 1,with the sum of the probabilities equal to 1. The Softmax function can be presented as :

$$f(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Softmax function is used in multiclass models where it returns probabilities of each class, with the target class having the largest probability. The Softmax function mostly appear at the outermost layer at the deep learning architecture.

3.4 Loss Function or Error Function

Loss function is used as a method of evaluating how well any algorithm models the given dataset. If the predictions are totally off, loss function will output a higher number. If they're pretty good, it'll output a lower number. As continuous changes are done in the attempt of improving the model, the loss function will tells us whether we are progressing or not.

There are various kinds of loss functions. Most popular ones are discussed below:

- **Mean Squared Error (MSE):** Mean Squared Error or MSE loss function is the default loss to use for regression problems. Mean squared error is calculated as the average of the squared differences between the predicted and actual values. It can be mathematically defined as:

  $$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2$$

  The result is always positive regardless of the sign of the predicted and actual values and a perfect value is 0.0. The squaring means that larger mistakes result in more error than smaller mistakes, meaning that the model is punished for making larger mistakes.
- **Binary Cross-Entropy:** Binary Cross-entropy is the default loss function to use for binary classification problems where the target values are between 0 and 1. This function is also known as Log loss. This function is defined as:

  $$L = \frac{1}{n}\sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{y_i}\log(\mathrm{y_i'}) + (1-\mathrm{y_i})\log(1-\mathrm{y_i'}))$$

  Here y is the expected probability & y' is the probability of each class label predicted by the model. Here the class labels are 0 and 1. This is a discrete probability distribution with two events and a certain probability for one event and an impossible probability for the other event. Binary Cross Entropy function mainly measures the divergence between these two probability distribution. If the value is large it means the difference between two distributions is very large and vice versa.

- **Categorical Cross Entropy:** It is an extension of Binary cross entropy for multiclass classification problem. It can be mathematically defined as the following equation:

$$L = -\sum_{j=0}^{M} \sum_{i=0}^{N} (y_{ij} log(y'_{ij}))$$

Choosing a loss function is an important part of neural network construction. The loss function is mainly application specific. In general we use MSE for regression problem, binary cross entropy for binary classification problem and categorical cross entropy for multiclass classification problem.

3.5 Overfitting

Overfitting happens when the model learns the signal as well as noise in the training data and wouldn't perform well on new data which it has never seen. It will increase the variance of a model which is not good for its performance. There are many ways to overcome this phenomenon. Sometimes creating bigger network, adding more hidden units, dropping out some units in every iteration and training on more data helps overcoming this problem. There is another solution to this and the solution is adding more information. This is called Regularization.

3.6 Regularization

In this technique some additional information is added to the network which is called Regularization parameter. Regularization parameter (lambda) penalizes or constrains all the coefficients so that model generalizes the data and won't overfit. In other words this technique discourages very complex models and thus it avoids the risk of Overfitting.

Here we will discuss about the L1 and L2 regularization. The equation of L1 and L2 regularization is very similar.

L1 regularization is also known as Lasso regularization. It can be expressed as:

$$Loss = error(y, y') + \frac{\lambda}{2m}\sum_{i=1}^{N}|w_i|$$

L2 regularization is also known as Ridge regularization. It can be expressed as:

$$Loss = error(y, y') + \tfrac{\lambda}{2m}\textstyle\sum_{i=1}^{N} w_i^2$$

Here the key difference between these two equations is the penalty term. Second term on the right side of each of the equation penalizes the weight matrices being too large. Ridge regression adds “squared magnitude” of coefficient as penalty term to the loss function. Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds “absolute value of magnitude” of coefficient as penalty term to the loss function.

3.7 Conclusion

In this chapter a brief discussion on neural network architecture and some related terminologies are discussed. This will help us to understand the concepts of Autoencoders and Convolutional Neural Networks.

Chapter 4
Feature Extraction

4.1 Introduction

Hyperspectral images typically record material’s hundreds of thousands of spectral wavelengths for each pixel in the image. Although the rich spectral signatures can provide useful information for data analysis, the high dimensionality of HSI data presents some new challenges like - curse of dimensionality problem. This phenomenon reduces the generalization capability of classifiers and deteriorate the classification performance, especially when the available labeled samples are limited.

Owing to the dense sampling of spectral wavelengths and as the spectral reflectance of most materials changes only gradually over certain spectral bands, many contiguous bands are highly correlated and not all features or spectral bands are expected to contribute useful information for the data classification/analysis task at hand. As one of the typical method to alleviate this problem, dimensionality reduction is widely used as a preprocessing step to remove the highly correlated and redundant measurements in the original high-dimensional HSI spectral space and preserve essential information in a low dimensional subspace. It has attracted increasing attentions in recent years.

There are primarily two methods for dimensionality reduction. They are:

a) **Feature Selection:** Feature selection is for filtering irrelevant or redundant features from dataset. There are several techniques for that, such as: Variance Thresholds, Correlation Thresholds, Genetic Algorithms (GA), etc.

b) **Feature Extraction:** Feature extraction is for creating a new, smaller set of features that stills captures most of the useful information. Feature extraction techniques can be broadly classified in four types:

    i. **Knowledge Based Feature Extraction:** Knowledge-based feature extraction methods enhance specific characteristics of spectral bands by performing arithmetic operations on the relevant original bands. These techniques are straightforward and determine features using spectral knowledge of the classes. Example: Normalized Difference

Vegetation Index (NDVI), Normalized Difference Water Index (NDWI), Soil Adjusted Vegetation Index (SAVI), etc. [15].

ii. **Statistical feature extraction:** The statistical feature extraction refers to the process of transforming the high dimensional data to a lower dimensional space enhancing the class separability without significant loss of information. A new set of features is generated by the re-distribution of underlying information. Statistical feature extraction can be of two types – supervised or unsupervised [15].

    Supervised methods require a prior knowledge in the form of labelled samples to determine some metrics that separate data points in different classes. For example: Generalized Discriminant Analysis (GDA), Linear Discriminant Analysis (LDA).

    Unsupervised feature extraction methods do not require a prior information or training data. Projection pursuit (PP) is an unsupervised mechanism that transforms high dimensional data into lower dimensional space retaining most of the information into a new set of orthogonal variables. For example: Principal component analysis (PCA), Independent Component Analysis (ICA), Maximum Noise Fragment (MNF), etc. [15]

iii. **Wavelet-based feature extraction:** Wavelet transforms are widely used in signal analysis. Wavelets have the ability to separate fine-scale and large-scale details of the signal, preserving the energy and spatio-geometrical information at different scales. Several methods based on wavelet transforms have been developed for feature extraction from hyperspectral data. Most of these methods use Discrete Wavelet Transform (DWT).

iv. **Deep feature extraction:** Some of the conventional feature extraction techniques have performed well in spectral domain. However, their ability to deal with the nonlinear nature of hyperspectral data is limited. In recent times, deep learning architectures observed remarkable success in dealing with the nonlinear input data. Deep learning techniques employ hierarchical learning framework to extract high level features with very deep neural networks. Typically neural networks deeper than three layers are considered as deep networks. Deep models progressively learn abstract and complex features from lower ones at higher layers, which are typically invariant to local changes of input data. Some popular techniques of this category are : Autoencoder, Convolutional Neural Network (CNN)

Following is diagram that clearly summarizes the feature extraction methods;

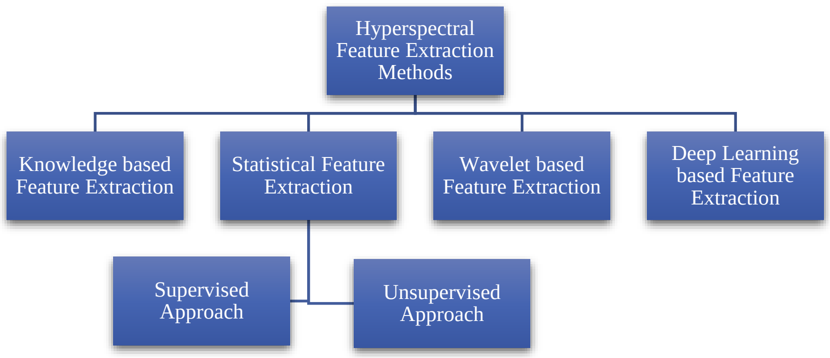

Hyperspectral Feature Extraction Methods
Knowledge based Feature Extraction
Statistical Feature Extraction
Wavelet based Feature Extraction
Deep Learning based Feature Extraction
Supervised Approach
Unsupervised Approach

Fig. 4.1: Schematic representation of Feature extraction Techniques in Hyperspectral Images

4.2 Literature Review

In the early stage study (mid 1980's) of Hyperspectral feature extraction, the main focus was purely on spectral-based methods. These methods either enhances relevant bands by arithmetic operations or projects the data onto a new feature space preserving the discriminative information. The most popular methods were - principal component analysis (PCA)[16], independent component analysis (ICA), linear discriminate analysis (LDA), etc.

In 2002, Craig Rodarmel and Jie Shan applied PCA in Hyperspectral Image classification[17]. Since then PCA is the most popular dimension reduction that is being used on Hyperspectral images. PCA has been applied successfully on Landsat-7 ETM imaginary for a high dimensionality reduction method and also image enhancement. Combination PCA with classification method Support Vector Machine (SVM) has been succeeded applied for handling image fusion with filtering called image fusion and recursive filtering (IFRF)[18].

Since 2000, manifold learning research became very active. Manifold learning has influenced many research areas, including hyper spectral remote sensing. This type of learning extracts the intrinsic structure of nonlinearly distributed data, which is considered to be very useful for hyper spectral feature extraction.[19]

Literature studies around 2007 have suggesting that more useful features can be extracted by incorporating spatial information into a spectral-based feature extraction system. With the development of imaging technology, hyper spectral sensors could provide good spatial resolution. Literature study's revealing that classification performance is well improved by using both spectral–spatial feature extraction methods. In 2007 Mathieu Fauvel and Jocelyn Chanussot introduced a method based on the fusion of morphological operators and support vector machine (SVM), which leads to high classification accuracy.[20]

In 2006, in a paper by Hinton and Salakhutdinov paper it was shown that effective dimensionality reduction of data can be achieved with Artificial Neural Network (ANN)[21]. They used non-linear deep auto-encoders, trained by back-propagation, to reduce the dimensionality of high-dimensional data [22]. Later Autoencoders have shown significantly good performance in Hyperspectral data that are non-linear in nature. In 2014, Yushi Chen and Zhouhan L in their paper proposed a dimension reduction technique on Hyperspectral image using Stacked Autoencoder [23] ,[22]. Besides that, learning features with convolution neural networks (CNN) has shown great significance and shown its importance in remote sensing applications from 2012. CNN also provided state-of-art performance in hyper spectral image classification problem[24].

In 2019, Heena Patel & Kishor P. Upla proposed a method combining Autoencoder and CNN [9].

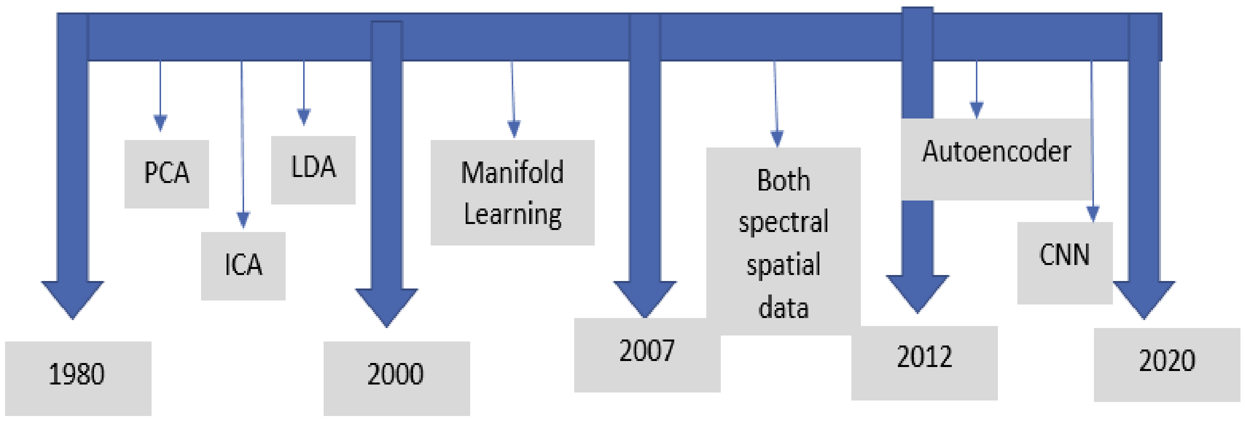


Fig. 4.2: Research trends in Hyperspectral Image Feature extraction method

4.3.1 Autoencoders

An autoencoder is an unsupervised learning algorithm. A basic autoencoder (AE) is a deep learning architecture model in which original data at the input is reconstructed at the output by going through one or multiple intermediate layer with reduced number of hidden nodes. The AE model tries to learn deep and abstract features in those reduced hidden nodes, so a reconstruction is feasible from them[25].

The generalized autoencoder consists of two parts, an encoder and a decoder. Encoder part has one visible layer of d inputs and and one hidden layer of h units with an activation function$f$. During training, it first maps the input $x\epsilon R^d$ to the hidden layer and get the latent representation $y\epsilon R^h$; and then y is mapped to an output layer that has the same size with input layer. This process is called "reconstruction." The reconstruction is denoted as $z\epsilon R^d$ (Fig. 15).

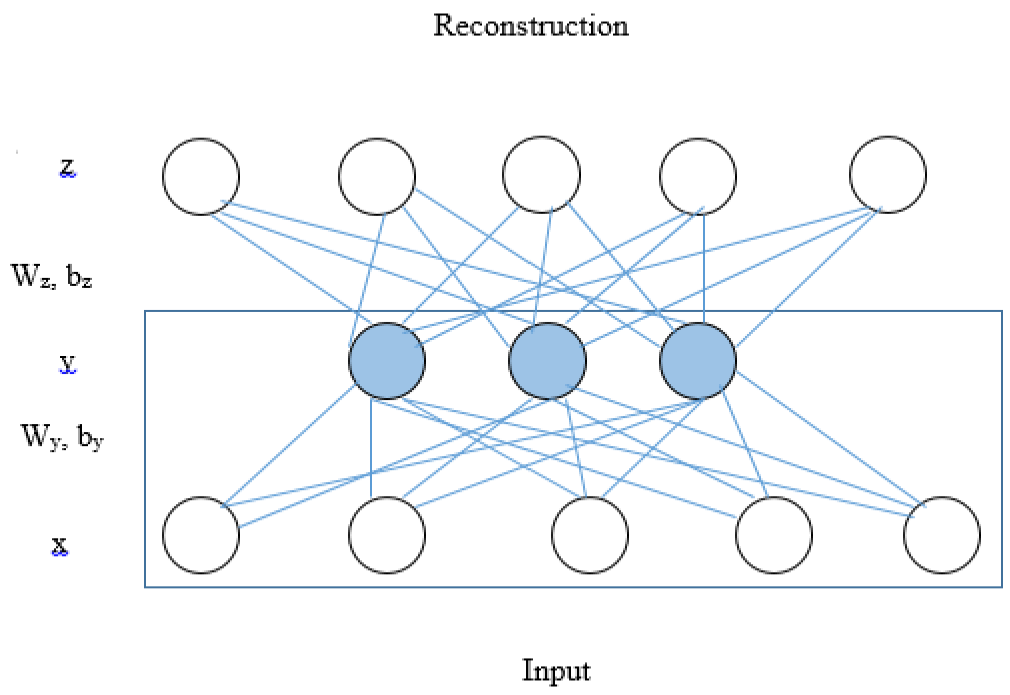


Fig. 4.3: A single layer autoencoder. The model learns a hidden representation "y" from input "x" by reconstructing it on "z". Corresponding parameters are denoted in the network [26].

An autoencoder is a typical feed-forward neural network that can hook together many simple neurons. The output of a neuron can be the input of another, and the parameters can be estimated using a back-propagation algorithm. A forward pass is first run to compute the activations throughout the network, including the output value of the hypothesis.

Mathematically this procedure can be shown as,

$$y = f\left(W_y x + b_y\right)$$

And,

$$z = f(W_z y + b_z)$$

Where W_y, W_z denotes input-to-hidden and hidden-to-output weights respectively and b_y, b_z denotes the bias of hidden and output units, and (·) denotes the identity function or the activation function, which apply element-wise to its arguments. The goal of training is to minimize the "error" between input and reconstructed input or output, such that,

$$argmin_{w,b_y,b_z}(cost(x,z))$$

Where z is dependent on parameters W, by, bz while x is given. (x, z) stands for the error. Generally for loss function we use MSE loss function or Cross Entropy los function.

Autoencoder applies backpropagation, setting the target values to be equal to the inputs. It uses, z = x equation for that. The weight updating rules can be defined as,

$$W = W - \alpha \frac{\delta cost(x,z)}{\delta W}$$

$$b_y = b_y - \alpha \frac{\delta\ cost(x,z)}{\delta W}$$

$$b_z = b_z - \alpha \frac{\delta cost(x,z)}{\delta W}$$

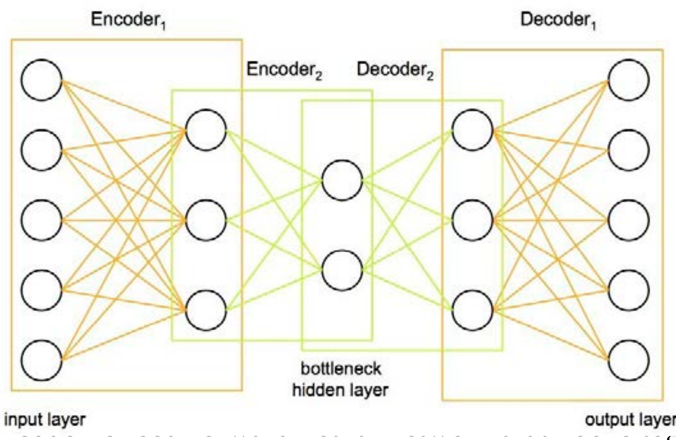
Where α is the learning rate.

The power of AE lies in this form of reconstruction-oriented training. During reconstruction, it only uses the information in hidden layer activity, which is encoded as features from input. If the model can recover original input perfectly from *y*, it means that retains enough information of the input. And the learned nonlinear transformation, which is defined by those weights and biases, can be deemed as a good feature extraction step. That's how autoencoder can be used as effective feature extraction technique.

4.

4.3.2 Stacked Autoencoder

When a number of autoencoders are stacked as shown in Figure 16. The hidden layer in one autoencoder becomes the input to the next one. This arrangement is known as Stacked Autoencoder.



We can see an auto ther many For

Fig. 4.4: Architecture of Stacked autoencoder [27].

In this multilayer network weights are generally learned in greedy fashion. The parameters are fine-tuned with back propagation.

Here, the Encoder$_1$ maps inputs to the first hidden layer or Encoder$_2$. In this manner, subsequent layers of AEs are trained via the output of its previous layer. In decoder layer we try to reconstruct the output of the Decoder$_2$ according to the activity of the Bottle neck layer. After that, the output of the whole network is constructed in Decoder$_1$ using the output of Decoder$_2$ as input.

4.3.3 Regularizing Stacked Autoencoder

There are two major problems in implementing stacked autoencoder architecture. They are:

1) In Figure. 16 we can see in a stacked autoencoder there many parameters that need to be learned. Sometimes it becomes computationally challenging when the network is too big.
2) Learning so many parameters (network weights) can lead the network to over-fitting [28].

One solution to these problems can be Regularization. We don't want our network to learn the noise of the data and simply copy it. We want our network to learn the inner structure of data. To exploit the inner structure of data, an additional regularization, the sparsity constraint on the hidden units, is introduced. In this technique, a neuron whose output is close to one is active, while one whose output is close to zero is inactive. A sparse autoencoder aims to limit neurons so that they are inactive most of the time.

There are several ways to regularize an autoencoder. Here, we will be using L1 regularization. We can do this by adding L1 regularization term in the loss function. Then our cost function becomes:

$$argmin_{w^s}(cost(x,z) + \lambda\|f(x)\|_1$$

Here, the L1 activity regularizer that will apply a penalty to the loss function during the optimization phase. It will act as weight decay term that tends to decrease the magnitude of the weights, prevents over-fitting. Now during each iteration randomly highest valued nodes (neurons) will be fired and the rest will be dropped.

It will force the autoencoder to use all of their neurons. It can no longer just memorize the input through certain nodes because, in each run, those nodes may not be the ones active. That's how, we can extract important features of data by using regularized stacked autoencoders.

4.4 PCA and Hyperspectral data

PCA is one of the most commonly used methods of dimensionality reduction. In 2003, Anil Cheriyadat and Lori Mann Bruce in their paper [29] tried to analyze whether Principal Component Analysis is the most suitable feature extraction technique for Hyperspectral data or not.

PCA is an unsupervised linear feature extraction technique. PCA maximizes data variance in the dimensionality reduced space. Mathematically what it does is:

Let’s assume our given data matrix is D of the size M × N where M denotes the number of pixels and N denotes the number of bands then,

Step 1. Standardization

Firstly, the data volume has to be moved to be recentered around the reference origin point so that each one of them contributes equally to the analysis. Otherwise, the features or bands with a higher value will dominate the principal components. The standardization can be performed via the given equation:

$$d'_{iJ} = \frac{d_{ij} - \mu}{\sigma}$$

Here, d_{ij} is the i’th sample value for j’th feature of the dataset D. μ is the mean and σ is the standard deviation of the dataset D. finally $d_{ij}{}'$ is the standardized value of d_{ij} .

Step 2. Covariance Matrix Computation

The main purpose of this step is to see if there is any relationship between them the bands or features. Covariance between two feature vector x and y can be calculated as following if the mean of x and y is respectively x’ $and\ y$’. The equation is:

$$Cov(x, y) = \frac{\sum_{i=1}^{M}(x_i - x')(y_i - y')}{M}$$

The covariance that we have as entries of the matrix tells us about the correlations between the variables is actually the sign of the covariance that matters. If positive then the two variables increase or decrease together or correlated. If negative then One increases when the other decreases (Inversely correlated).

Step 3. Computing Eigen value and Eigen vector

PCA is based on Eigen value decomposition of the co-variance matrix. Which takes the form of:

$$C_x = ADA^T$$

Where, $D = diag(\lambda_1, \lambda_2, \ldots, \lambda_N)$ is the diagonal matrix composed of the eigenvalues $\lambda_1, \lambda_2 \ldots \lambda_N$ of the covariance matrix C_x and A is the orthonormal matrix composed of the corresponding N dimension eigenvectors $a_k (k = 1,2, \ldots, N)$ of C_x as follows:

$$A = (a_1, a_2, \ldots, a_N)$$

The linear transformation defined by:

$$y_i = A^T x_i \ (i = 1,2, \ldots, M)$$

This generates the PCA pixel vector. All these pixel vectors form the PCA (transformed) bands of the original images.

Step 4. Feature Vector

In the previous step we got the principal components in order of significance. In this step, we decide that if we want to keep all these components or discard those of lesser significance and form with the remaining ones a matrix of vectors that is called feature vector.

Step 5. Recasting the dataset along the principal component axis

The purpose of this step is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components. This is done by multiplying transpose of the original standardized dataset, $D'$by the transpose of the feature vector, F.

$$Final\ Dataset, D^R = F^T * D'^T$$

By following these steps, we can obtain a reduced low dimensional dataset D^R of the original high dimensional dataset D which store almost the same amount of information as the original dataset.

From the calculations above we can say that the principal component analysis is based on the fact that neighboring bands of hyperspectral images are highly correlated and often convey almost the same information about the object. The analysis above is used to transform the original data to remove the correlation among the bands. In the process, the optimum linear combination of the original bands accounting for the variation of pixel values in an image is identified [17].

There are several reasons for which PCA is not the best feature extraction technique for HSI data. They are discussed below:

1) Hyperspectral data tend to be distributed in the shape of a hyper ellipsoid, which demonstrates the fact that higher dimensional data is highly correlated. In PCA, second order statistics, the co-variance matrix, is used as the basis of transformation. But not all hyperspectral data is highly co-related. In those scenarios PCA performs very poorly.
2) Many hyperspectral data distributions exhibit large within class variance, S_w, which is caused by several factors like natural variation in the target material, environmental conditions, sensor angle, etc. Although some of these variances can be compensated, within-class variance is still an alarming characteristic of remote sensing data, which degrades the classification accuracy.

3) PCA may not perform better with multi-class data distributions. Since PCA is applied on the entire data space, it tries to maximize some of the global statistics of the data space, ignoring some of the local statistics that can discriminate the classes.

These are several reasons for which PCA is not the best feature extraction technique. But this method is very popular as it requires very low computational power and easy to understand. Now we will compare the autoencoder and the PCA technique.

4.5 Comparison between Autoencoder & PCA

When in autoencoder we use linear activation functions (example: Sigmoid function) and MSE as the loss function, then it can be shown that the autoencoder reduces to PCA. When nonlinear activation functions (example ReLU) are used, autoencoders provide nonlinear generalizations of PCA.

PC's are totally uncorrelated with each other because each component is projected onto different orthogonal axes. But autoencoded features may have correlations since they are just trained for accurate reconstruction.

For classification which technique to use hugely depends on the nature of the data. If the features have non-linear relationship with each other than autoencoder will be able to compress the information better into low dimensional latent space leveraging its capability to model complex non-linear functions.

PCA performs better when there is high correlation in the dataset. This high correlation can be exploited through correlation map. If we see our data is not highly correlated, we should use autoencoder for feature extraction.

If the first Principal component of a dataset expresses less that 30% of its total variance then we shouldn't use PCA too. Then autoencoder becomes the better option for dimension reduction.

4.6 Conclusion

In this chapter we discussed about different types of feature extraction techniques and their history. We also got a detailed idea on how autoencoder encodes data. After that we drew a comparison between autoencoder and most popular feature extraction technique PCA. This discussion will help us to understand the later chapters.

Chapter 5
Convolutional Neural Network

5.1 Introduction

In recent years, Convolutional Neural Networks (CNN) have shown remarkable performance in image classification and computer vision because they are very good at detecting edges, patterns in an image. CNNs are being widely used in Hyperspectral image classification too.

CNNs are feed-forward neural networks. Often CNNs have up to 20 or 30 layers. CNNs work so good because of a special kind of layer called the convolutional layer. CNNs contain many convolutional layers, which are stacked on top of each other. These convolutional layers are capable of recognizing different kinds of sophisticated shapes. This convolutional layers are followed by pooling layers and fully connected layers. Together these layers can recognize handwritten digits or human faces.

The use of convolution layers in a CNNs represents the layout of the human visual cortex, where a series of layers process an input image and classify increasingly more complex features.

5.2 Literature Review

The first CNN model was developed in 1989 combined with backpropagation model by Ye. Le. Cun and B. Boser [30]. In 2010 Xavier Glorot and Antoine Bordes introduced Rectified Linear Functiontion (ReLU) as the activation function for CNN [31] which improved CNN models a lot.

In the 2015,in a paper by Yangyu Huang and W hu, they proposed a novel CNN architecture for Hyperspectral image classification[32]. This architecture also had five common layers - input layer, the convolutional layer, the max pooling layer, the full connection layer and the output layer.

In 2017, Xuefeng Liu and Qiaoqiao Sun in [33] proposed their hybrid classification method combining PCA with CNN for dimension reduction .

Heena PatelKishor and UplaKishor Upla in their paper published in 2019 used Autoencoder before CNN architecture for enhancing the features [9].In 2020, Madhumitha Ramamurthy and Y. Harold

Robinson used autoencoder for in their paper for dimensionality reduction and CNN for classification in HSI data[34].

5.3 General Architecture of CNN

There are three basic components of defining a basic convolutional network. They are:

1. The convolutional layer
2. The Pooling layer(optional)
3. The Fully Connected layer

In the first layer or convolution layer, input image is convoluted using many types of filters for producing a feature map. On the second layer or pooling layer translation and scaling is done. This layer basically reduces the size of the feature map exponentially. Fully connected layer is the same as the hidden layer in the neural network. In the fully connected layer all the inputs from one layer are connected to every unit of the next layer. It is usually connected to output layer.

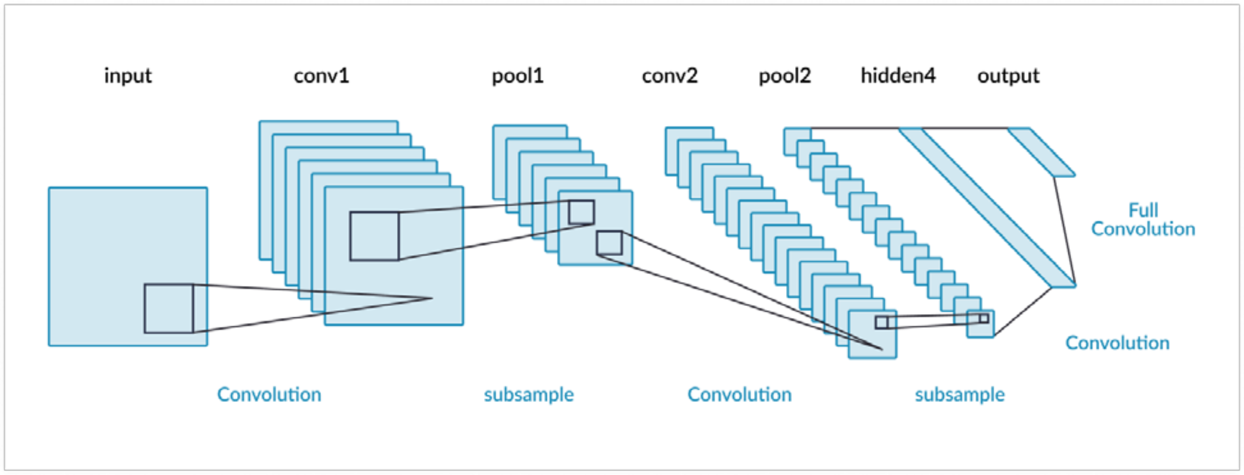


Fig. 5.1: General architecture of Convolutional Neural Network [35]

5.4.1 Convolution Operation

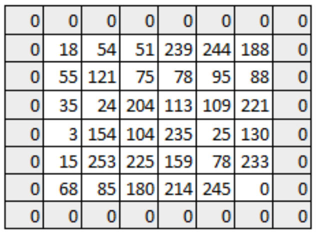
Convolution is a very powerful tool. The purpose of doing convolution is to extract and use most important features from the data. Various types of filters are used in image processing. Each type of filters helps to extract different aspects or features from the input image. Some filters detect horizontal edges, some filters can detect vertical edges. These filters combined can detect complex feature from data. Similarly, in Convolutional Neural Network, different features are extracted through convolution using filters.

Mathematically convolution function is defined as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

It is defined as the integral of the product of the two functions after one is reversed and shifted. For understanding convolution operation in a image we have to at first learn some terminologies. They are described below:

- **Kernel size:** The kernel is defined as the filter that is used to extract features from the images. The kernel matrix moves over the input data, performs the dot product with the sub-region or patch of input data, and gets the output as the matrix of dot products.
  In 2D CNN, kernel moves in 2 directions where as in 3D CNN kernel moves in 3 directions.
- **Number of Kernels:** Corresponds to the number of filters that are used in each learning or iteration. This is done for seeking something new in data.
- **Padding:** In some cases the neurons at the borders can't process a whole receptive field. We can solve this by adding a border of zeros around the image. It helps us keep more of the information at the border of an image. Without padding, very few values at the next layer would be affected by pixels as the edges of an image. It also allows us to use a convolution layer without necessarily shrinking the height and width of the volumes. In deep learning we use "valid" padding to denote no padding was used and we use "same" padding to denote that the output dimension is the same as the input dimension.



0	0	0	0	0	0	0	0
0	18	54	51	239	244	188	0
0	55	121	75	78	95	88	0
0	35	24	204	113	109	221	0
0	3	154	104	235	25	130	0
0	15	253	225	159	78	233	0
0	68	85	180	214	245	0	0
0	0	0	0	0	0	0	0

Fig. 5.2: one layer of zero padding added to input image

- **Stride:** Stride governs how many cells the filter is moved in the input to calculate the next cell in the result. The larger the stride the lesser the output height and width.

We can apply a simple formula to calculate the output dimensions. The spatial size of the output image can be calculated as,

$$\left\lfloor \frac{W - F + 2P}{S} \right\rfloor + 1$$

Here, W is the input volume size, F is the size of the filter, P is the number of padding applied and S is the number of strides. Suppose we have an input image of size 32*32*3, we apply 10 filters of size 3*3*3, with single stride and no zero padding. Here W=32, F=3, P=0 and S=1. The output depth will be equal to the number of filters applied which is 10. The size of the output volume will be ([32-3+0]/1) +1 = 30. Therefore the output volume will be 30*30*10.

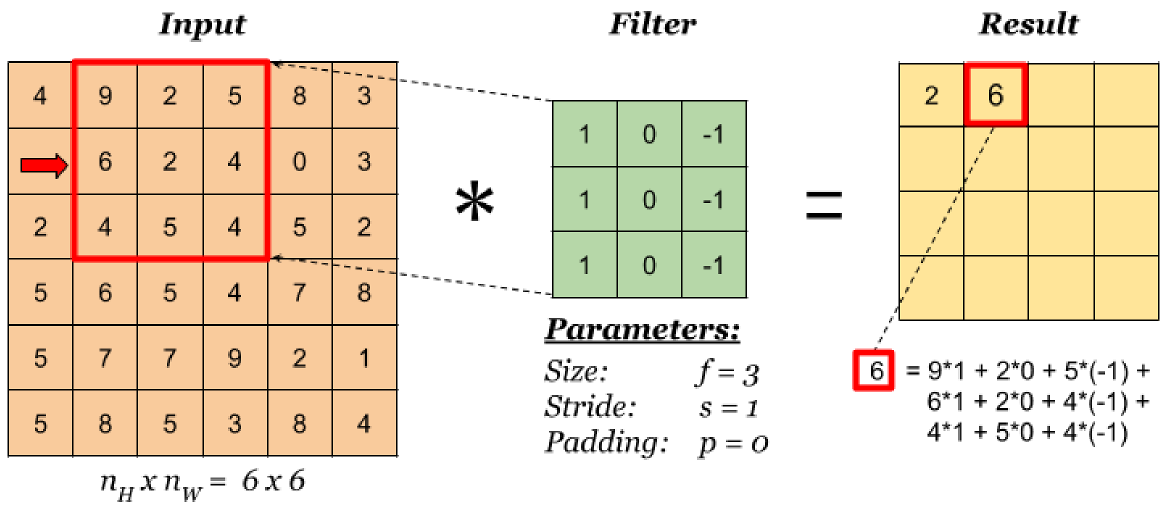


Fig. 5.3: Convolution operation [36]

5.4.2 2D-CNN:

It is called 2 dimensional CNN because the kernel slides along only 2 dimensions. The kernel in 2D-CNN has 3 dimensions – height, width and the number of channels in input data.

Suppose, that we are given a hyperspectral image of size N × M × D, where N and M are the width and the height of the image and D denotes the number of spectral bands. If we want to use a K sized kernel its dimension will be K × K × D. It’s demonstrated in Figure 5.4.

For example when we apply 2D-CNN to an RGB image, the filter also has matching number of channels. For calculating one output cell, convolution on each matching channel should be performed and added.

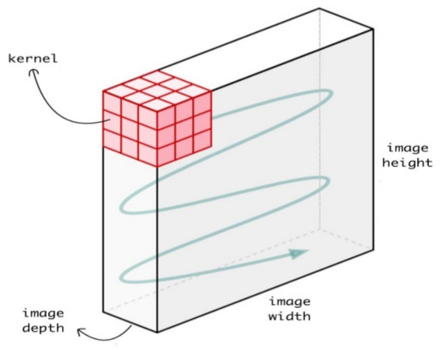


Fig. 5.4: 2D convolution Kernel [37]

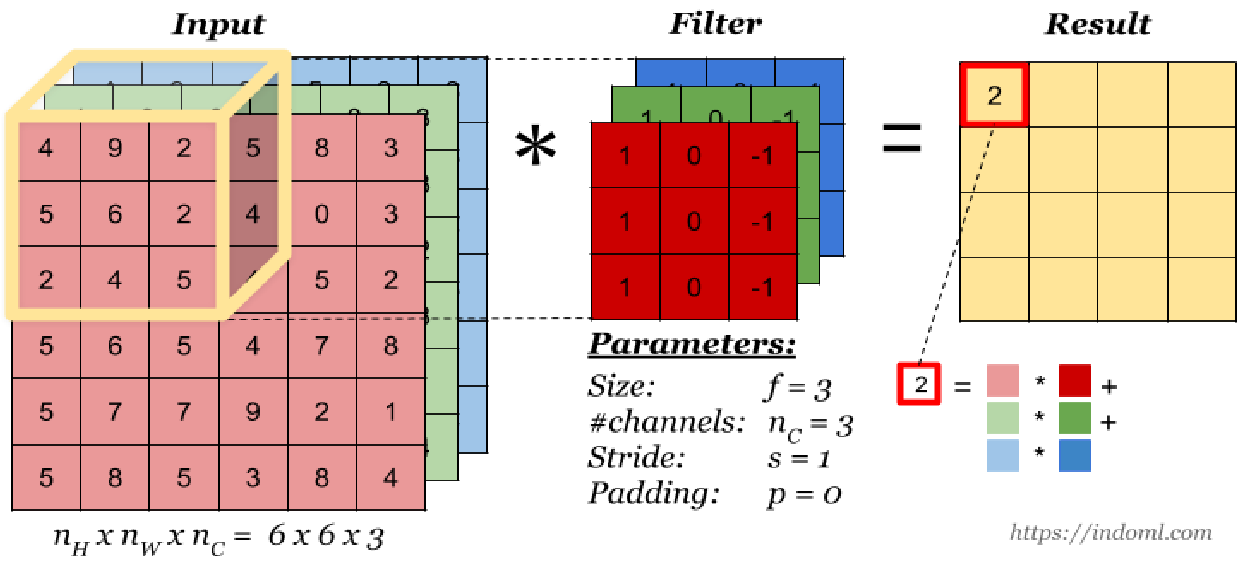


Fig. 5.5: 2D convolution on multiple channel [36]

5.4.3 3D-CNN:

Training 3D-CNN is similar to the training of 2D CNN. 3D CNNs use 3D convolutional kernels for producing a volumetric patch of a scan. This three dimensional kernel can slide in 3 dimensions as shown in Figure 23.

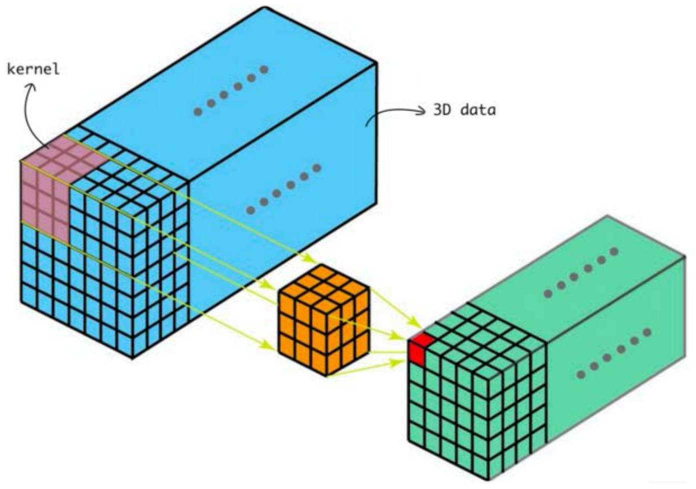


Fig. 5.6: 3D convolutional kernel [38]

Hyperspectral image is captured by scanning the same region with different spectral bands. The formed image by this method can have some correlations as close hyperspectral bands may result in similar images. It is desirable to take into account hyperspectral correlations. Though the 2D-CNN model can only utilize the spatial context, it ignores the hyperspectral correlations. On the other hand 3D-CNN can extract the spectral data too.

5.5 Pooling Layer

Often when the images are too large, we might need to reduce the number of training parameters. Pooling is performed separately for each bands, so the number of bands of the image remains unchanged. The pooling layer not only reduces the size of the representations it can also speed up the calculations. It can make some of the detected features more robust. . The most common type of pooling layer commonly used is max pooling and average pooling.

Pooling layer has 3 hyper parameters. They are –

1. Size(f)
2. Stride(s)
3. Type (max or average pooling)

Let's assume we have an input of $n_h \times n_w \times n_c$. Here n_h is the height, n_w is the width and n_c is the number of channels in data. If we apply the mentioned hyperparameters for pooling the output dimension will be of size:

$$\left\lfloor \frac{n_h - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_w - f}{s} + 1 \right\rfloor \times n_c$$

Where $\left\lfloor \frac{n_h - f}{s} + 1 \right\rfloor$ is the height, $\left\lfloor \frac{n_w - f}{s} + 1 \right\rfloor$ is the width and n_c is the number of channels.

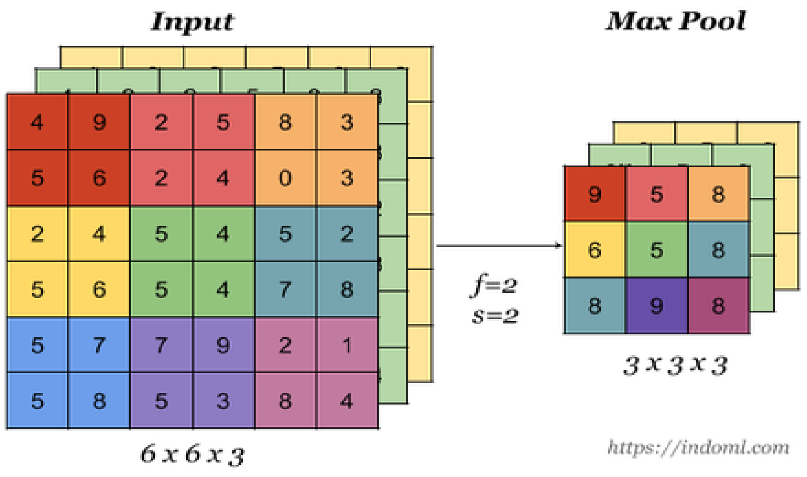


Fig. 5.7: Max pooling [36]

5.6 Fully Connected Layer

Convolutional layer and pooling layer can only extract the features but for classification we need fully connected layer for generating an output equal to the number of classes we need. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

Fully Connected Layer is a feed forward neural networks. Fully Connected Layers are usually connected to the output of the network. The output layer uses the Softmax activation (instead of ReLU) after going through the fully connected layers. The probabilities or output of Softmax function assigns the input being in a specific class (classification).

5.7 Conclusion

In this chapter a brief review on CNN architecture was given. We've also discovered how 2D-CNN or 3D-CNN functions. This information will help us to the feature extraction methods that are discussed later.

Chapter 6

Proposed Methodology

6.1 Introduction

Classification of Hyperspectral image has some unique challenges. Its high dimensionality and high correlation among bands that make it hard to classify them. Moreover, not every type of hyperspectral data is same. Because of uncontrolled changes in the reflectance captured by the spectrometer (normally because of changes in atmospheric conditions, occlusions due to the presence of clouds, and variations in illumination, among other environmental interferers) not every type of Hyperspectral data has same properties.

Considering all these factors a methodology was proposed that can capture the most important features of hyperspectral data and classify them accordingly.

6.2 Overview of Proposed Methodology

Hyperspectral images are volumetric data. It contains many redundant bands. These massive amount of redundant data can mess with the performance of any classifier. That's why before feeding this data into any classifier it's better to pick out the bands that best represent the data. That's why we use dimensionality reduction methods.

For dimensionality reduction we will use Autoencoder. On the first step we will normalize image between values 0-1. Autoencoders are unsupervised in manner. They tries to find the best reconstruction of data in reduced dimension space. We will feed our whole data in Stacked Autoencoder. When we get desired accuracy for reconstruction or tolerable threshold error we will stop the training. We reduce the dimensions such a way that spatial dimensions are preserved only the spectral bands are reduced. Now we have got the best representation of our data in desired number of spectral bands.

After that we will split the data for train, validation and split. We will train our 2 dimensional Convolutional Neural Network (2D-CNN) with the training data. In the training stage 2D-CNN extracts the deep features from compressed data which would be later used for classification. We will use the validation data to observe the training progress. After that we will test the model using the test data.

An overview of proposed method is given below:

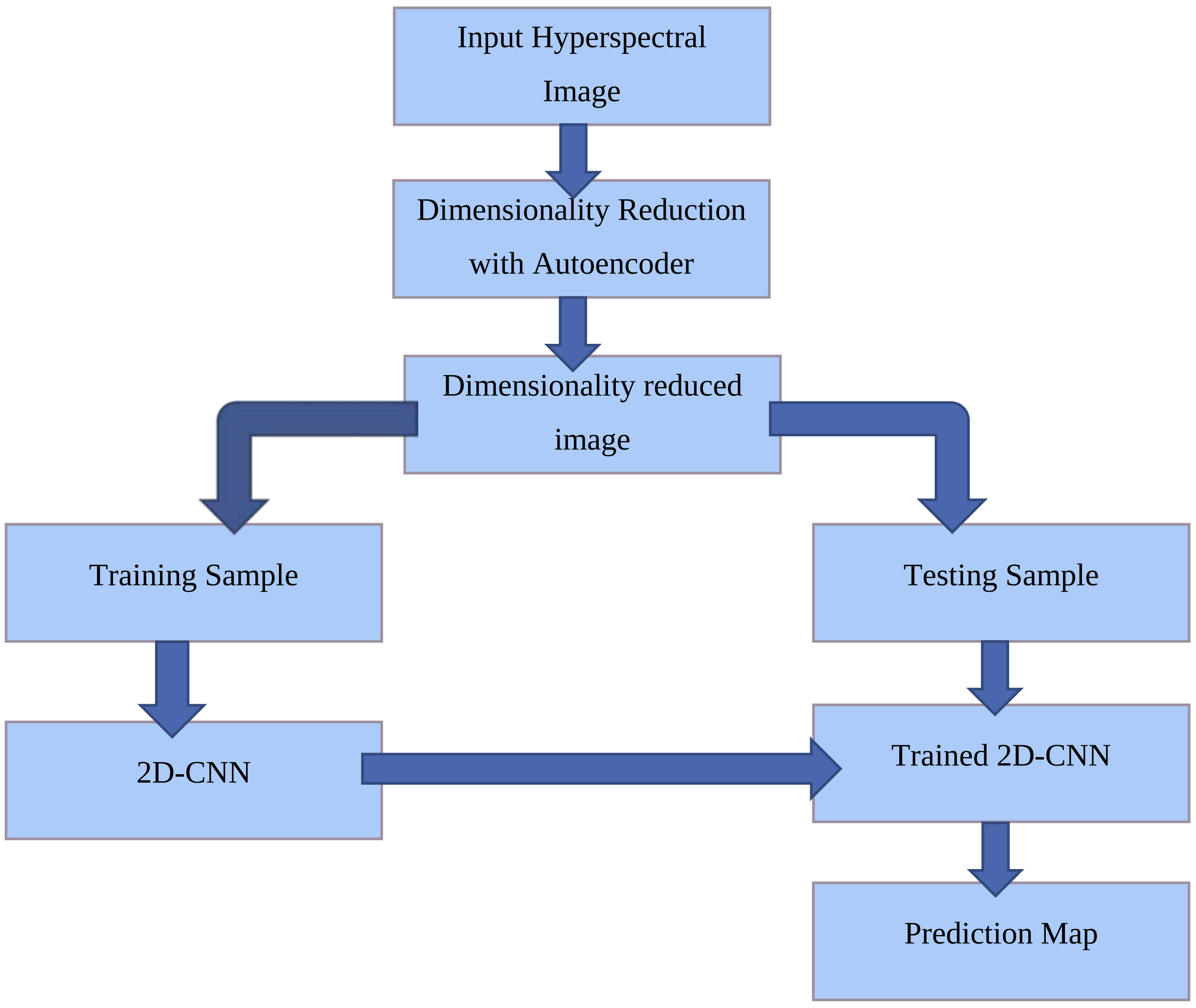

Input Hyperspectral Image
Dimensionality Reduction with Autoencoder
Dimensionality reduced image
Training Sample
Testing Sample
2D-CNN
Trained 2D-CNN
Prediction Map

Fig. 6.1: An overview of Stacked Autoencoder-CNN approach

6.2.1 Dimensionality Reduction using Stacked Autoencoder

The first part of our proposed approach is dimensionality reduction. Dimensionality reduction using Autoencoder is an unsupervised nonlinear method. Autoencoder is known to be very good at enhancing non-linear characteristics on high-dimensional data.

Let's denote the volume of hyperspectral data as $I \in R^{M\times N\times L}$, where I is the original input, M is the width, N is the height and L is the number of spectral bands. These high volume of pixels are classified in 10-16 classes. There are high intra-class variability and interclass similarity between the classes. It is a big challenge for any classification model to tackle this. That’s why we need to reduce the spectral dimension in such a way that it represents the HSI data well. Autoencoder will keep the spatial dimensions intact but reduce the spectral dimension form L to D. The spatial bands are kept intact because they are important for recognizing objects. We represent Autoencoder reduced data cube by $X \in R^{M \times N \times D}$, where X is the modified input, M is the width, N is the height and D is the number of bands after feeding the data into autoencoder. As the label of each pixel is not used here that’s why it’s an unsupervised method.

The Stacked Autoencoder model is very simple. It is primarily constructed using multilayer backpropagation neural network. This model is consisted of one input layer, one output layer and multiple hidden layers. The number of input nodes is equal to the number of pixels in Hyperspectral image. For our proposed model we are using four hidden layers- two for encoder portion and another two for decoder portion. The output of final encoder layers is fed to the decoder layers. In the decoder portion the data is reconstructed. Our goal is to reduce the reconstruction error.

Each hidden layer is a densely connected neural network layer. Each hidden layer can contains 100 neurons except for the bottleneck layer. This formation has high risk of Overfitting. To avoid that L1 regularizer is attached to each hidden layer except for the bottleneck layer. The number of neurons in bottleneck layer is equal to the number of desired reduced dimensions or D so we want to fire every node in this layer. The primary goal is that it takes input I and tries to reconstruct in on the output using D number of bands. Where $D \ll L$. That’s how Stacked Autoencoder reduces dimensions keeping the information of data intact. We try to keep the reduced dimension close to 15 to 25 as it best represents our data.

Let’s assume, $P = M \times N$

Where M = width of the HS data cube

N= height of the HS data cube

The architecture of our model is:

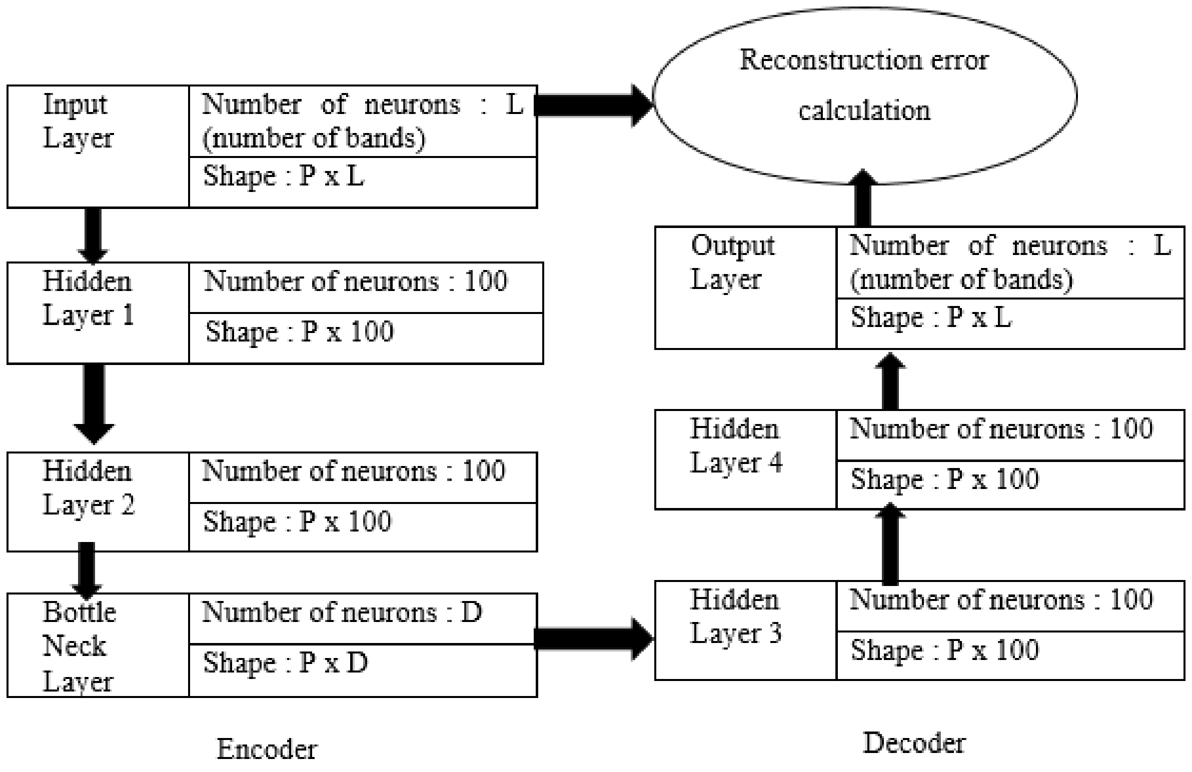


Fig. 6.2: Architecture of proposed Stacked Autoencoder

For KSC dataset number of spectral bands is 176. If reduce the dimension to 25 using this model, the subsequent layers will be of size 176-100-100-25-100-100-176.

6.2.2 Classification using Convolutional Neural Network

Convolutional Neural Networks shows surprisingly good performance in various computer vision area, HSI classification is one of those areas. CNN models can gracefully integrate spectral features with spatial features of HS data cube.

As Autoencoder beautifully summarizes spectral information we don't need to use complex Convolution like 3D-CNN. So we will be using 2D-CNN. The whole process is divided into two parts. They are – Feature Extraction (FE) net and Classifier net.

Feature Extraction net is composed of multiple convolutional layer, activation function and Pooling layers. In 2D-CNN process the input data is convolved by 2D kernels. It performs spectral-spatial analysis by accepting spatial patches over all available spectral bands. So, to match with the CNN architecture the input is decomposed into spatial patches. We make $S \times S$ size patches where S can be 3, 5, 7, 9, 11, etc. Mostly odd number of pixels are used to create patches. Each patch represents 3D kernel of size $S \times S \times D$, where D is the number of reduced spectral bands. In 2D Convolution the dot products between kernel weights and input patches are integrated across spectral bands or channels. The kernel is strides on the input data to cover full all the spatial dimensions. The output volume has same depth as the number of applied filters. The convoluted features are passed through the activation function, introducing non-linearity in the model. We have used ReLU activation function in our model. After that we applied Pooling layer. The pooling layer applies non-linear down-sampling to activation maps. In model 2D Max Pooling is used which captures the maximum pixel value of each 4 pixel cluster.

The classifier net performs the final classification based on features obtained on Feature Extraction portion. Normally Fully Connected layer followed by Softmax activation function is applied. The Softmax function gives the probability of each pixel belonging to each of the classes.

For our model we reduced the spectral dimension to 25 and took patches of size 11x11. The summary of our model is:

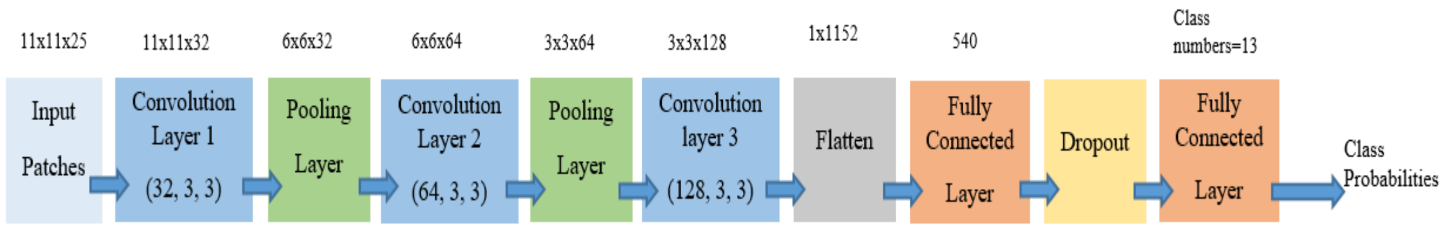


Fig. 6.3: Architecture of proposed 2D-CNN approach

The first dimension of 2D convolution kernel is $32 \times 3 \times 3$, where 32 is the number of 2D kernels and 3×3 represents the spatial dimension of 2D-kernel. We have experimented with many types of kernel sizes, among the sizes 3×3 gives the optimal result.

The parameters of the network is described below:

Table 6.1: Parameters of proposed 2D-CNN network

Layers	Convolution Layer 1	Convolution Layer 2	Convolution Layer 3	Dense Layer 1	Dense Layer 2
Kernel Size	32x3x3	64x3x3	128x3x3	540	13
Feature Maps	7232	18494	73856	622620	7033

6.3 Conclusion

In this chapter the methodology of proposed system is briefly explained with necessary diagrams and tables. In the later chapter we will explore how this dimension reduction and classification techniques work on real life data.

Chapter 7
Result & Performance Analysis

7.1 Introduction

Result and performance evaluation of a method is a crucial part. In the previous chapter the proposed methodology was explained. In this chapter we will see how our model performs on real world data. The dataset we will be using is briefly explained in this chapter. On our dataset we will apply autoencoder for dimension reduction and compare the performance with other classic dimension reduction method PCA. Following that we will apply 2D-CNN for classification task and again compare it with a widely used classifier Support Vector Machine (SVM). The performances will be compared using a different set of accuracy and performance metric.

7.2 Dataset Description

Various type of Hyperspectral datasets are publicly available. Each datasets are different from one another. One classification method that works on one dataset may not work on another. Here we have chosen Kennedy Space Center dataset for our experiments.

The NASA AVIRIS (Airborne Visible/Infrared Imaging Spectrometer) instrument acquired data over the Kennedy Space Center (KSC), Florida, on March 23, 1996. AVIRIS acquires data in 224 bands of 10 nm width with center wavelengths from 400 - 2500 nm. The KSC data, acquired from an altitude of approximately 20 km, have a spatial resolution of 18 m. After removing water absorption and low SNR bands, 176 bands were used for the analysis. Training data were selected using land cover maps derived from color infrared photography provided by the Kennedy Space Center and Landsat Thematic Mapper (TM) imagery. The vegetation classification scheme was developed by KSC personnel in an effort to define functional types that are discernable at the spatial resolution of Landsat and these AVIRIS data. Discrimination of land cover for this environment is difficult due to the similarity of spectral signatures for certain vegetation types. For classification purposes, 13 classes representing the various land cover types that occur in this environment were defined for the site. The number of labeled pixels is 5211.

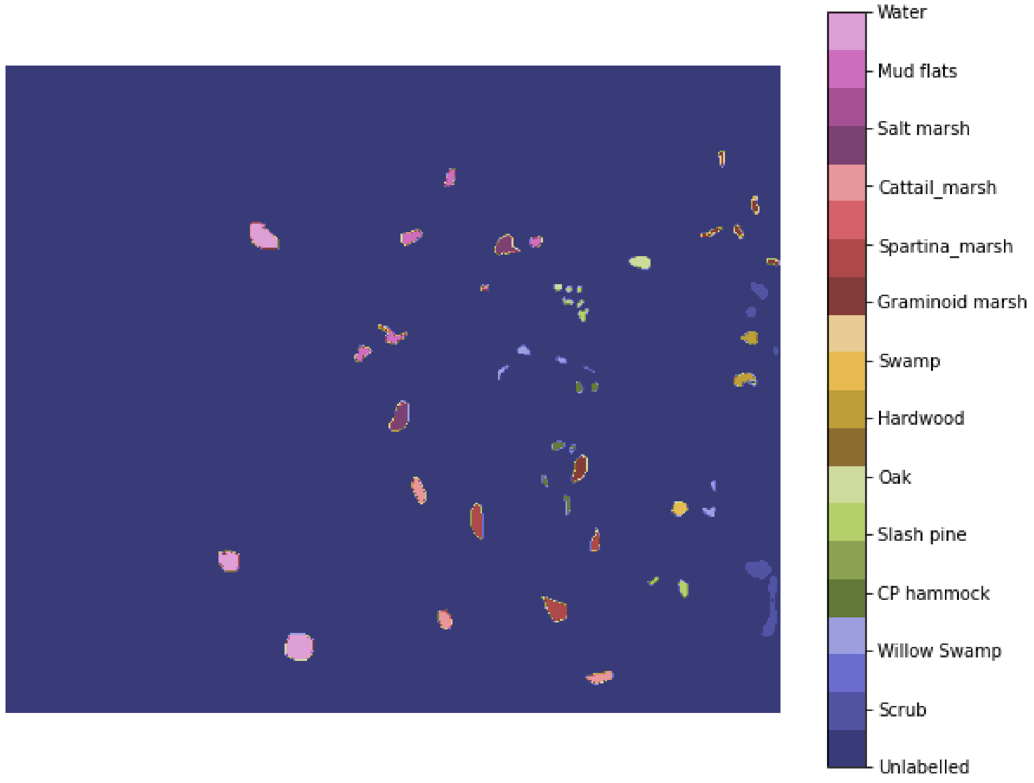


Fig. 7.1: Ground Truth of Kennedy Space Center Dataset

We can see there are 13 different classes in KSC dataset. Different classes have different number of pixels. Ground truth classes along with their respective sample numbers are given below:

Table 7.1 Ground truth classes for the Kennedy Space Center dataset and their respective sample numbers

Order No.	Class Name	Number of Samples
1	Scrub	761
2	Willow Swamp	243

3	CP hammock	256
4	Slash Pine	252
5	Oak/broadleaf	161
6	Hardwood	229
7	Swamp	105
8	Graminoid marsh	431
9	Spartina Marsh	520
10	Cattail Marsh	404
11	Salt Marsh	419
12	Mud Flats	503
13	Water	927
	Total	5211

7.3 Experimental Setup

All the necessary programs were written on Colab notebook. Colab is a product of Google that enables everyone to write and execute machine learning code on browser. Colab gives each user 2.3 GHz CPU, 12GB of RAM and NVIDIA-SMI 450.36.06 GPU.

The proposed Autoencoder and 2D-CNN both architectures were implemented on Keras framework which is built on top of Tensorflow. Tensorflow is an open-source library developed by Google which is specialized on Machine Learning. The PCA dimensionality reduction algorithm and SVM classifier both are implemented in scikit-learn library.

7.4 Result Analysis

We have proposed a model with two subsequent neural network. Let’s call it SAE-2DCNN model. We will take different hyperparameters for each of deep learning models and compare their overall accuracy.

For Autoencoder we have taken dimensions D = 7,15,20,25. We want to see how well our input was reconstructed for each dimensions. For that for each band we calculated the total intensity of KSC data.

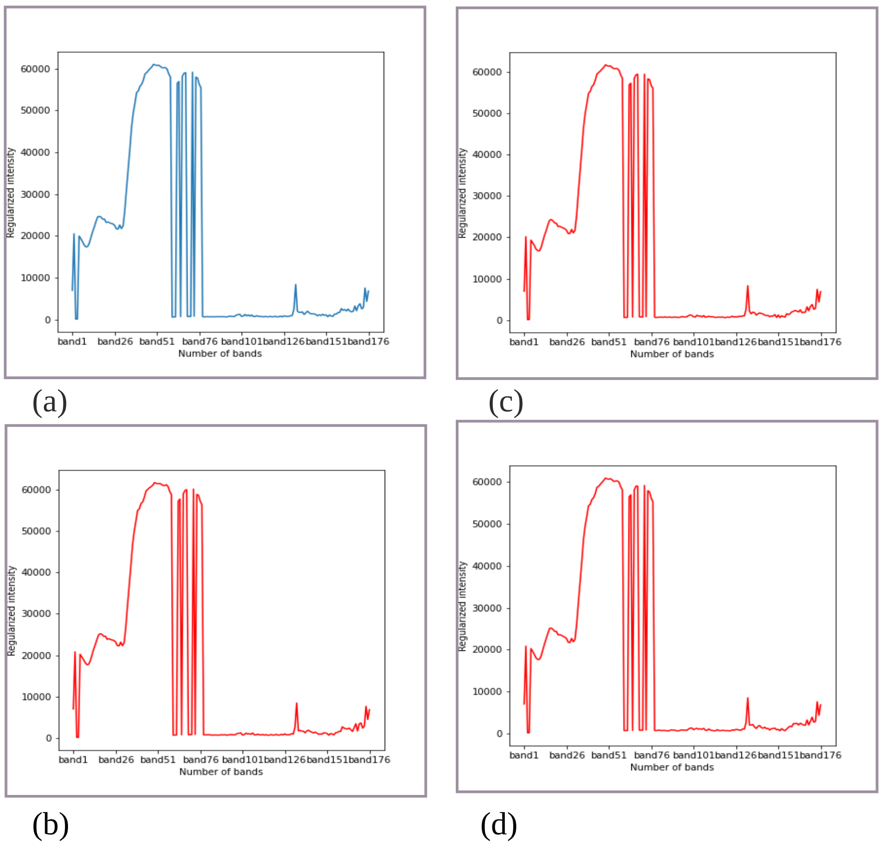


Fig. 7.2: Reconstructions of KSC data in different reduced dimension. (a) Input Spectrum (b) ~ (d) Reconstruction of (a) in dimension 7, 15 and 25 respectively.

Here we can see our Stacked Autoencoder can replicate the input pretty accurately for different dimensions.

We can vary the spatial context of 2D-CNN too. We have used patches of size 3, 5, 7 & 11. We have seen that on larger dimensions and on larger patches our classifier performs well. We have represented the performance of our model in various dimensions and patch size in a graph below:

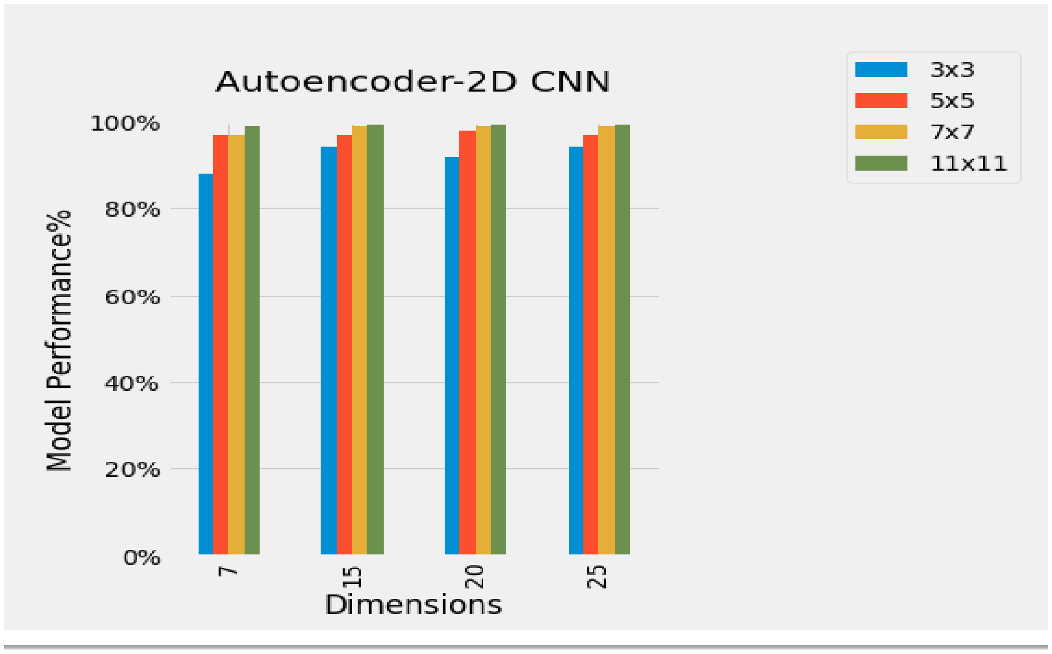


Fig. 7.3: Performance of SAE-2DCNN model on different dimension and patch size

From the graph we can see our hybrid model AE-2DCNN performs best when patches are of size 11 with a classification accuracy 98.8%. There is $\pm 0.20\%$ deviation associated with each accuracy measure.

99.4% is the average accuracy. But for evaluating a model class specific performance measure is needed. That's why we use Overall accuracy (OA) and Cohen Kappa accuracy.
Overall accuracy denotes the ratio of the samples recognized as correctly by the classification model among the total number of samples. On the other hand Cohen's Kappa accuracy is a measurement that is defined on the difference between the actual agreement in the confusion matrix and the chance agreement. The higher the Kappa score the better the classifier. That's why Kappa accuracy is very much useful metric in multiclass classification.

Classification accuracy of each class for dimension = 25 and window size 11 is shown below:

Table 7.2 Classification accuracy of combined approach AE-2DCNN on Kennedy Space Centre dataset

Class name	Number of Train samples	Number of Test samples	AE-2D CNN accuracy For dimension = 25, Patch size = 11x11
Scrub	609	152	100%
Willow Swamp	194	49	100%
CP hammock	205	51	97.5%
Slash Pine	202	50	96%
Oak / broadleaf	129	32	84%
Hardwood	183	46	96%
Swamp	84	21	100%
Graminoid marsh	345	86	100%
Spartina Marsh	416	104	100%
Cattail Marsh	323	81	100%
Salt Marsh	335	84	100%
Mud Flats	402	101	100%
Water	741	186	100%
Average Accuracy			**99%**
Overall Accuracy			**98%**
Kappa Accuracy			**99%**

Here we can see our hybrid classifier classifies most classes accurately securing accuracy score of on average 99.4%.

7.5 Competitive Comparison

In this section we have compared our proposed approach we have our proposed model with other bench mark approaches. Experimental result has shown has our proposed approach has outperformed previous hyperspectral classification on KSC dataset.

On our first comparison we have compared Stacked Autoencoder with PCA. PCA is a well-known and widely used dimension reduction technique. For PCA we have taken dimensions- 7, 15, 20, 15; just like we have taken for Stacked Autoencoder. For Classifier we have used SVM and 2D-CNN. The result:

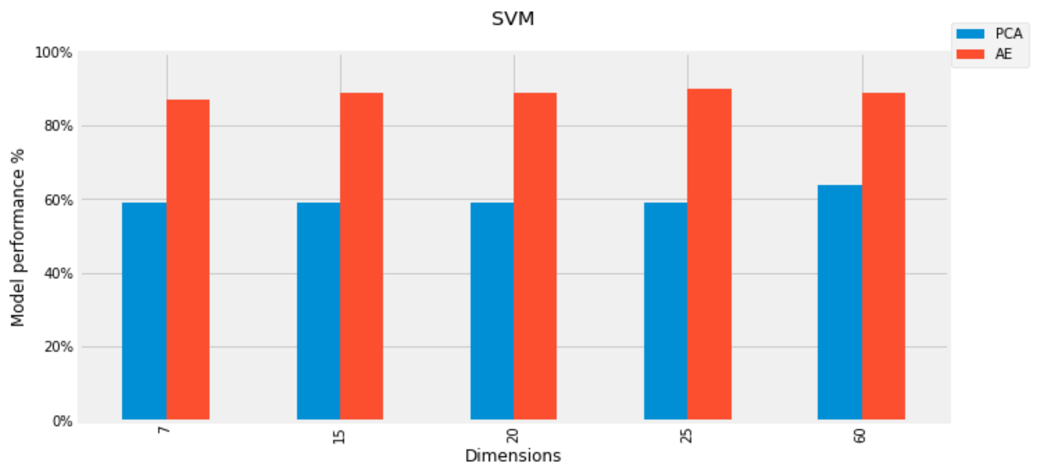


Fig. 7.4: Performance comparison of SAE and PCA on SVM classifier

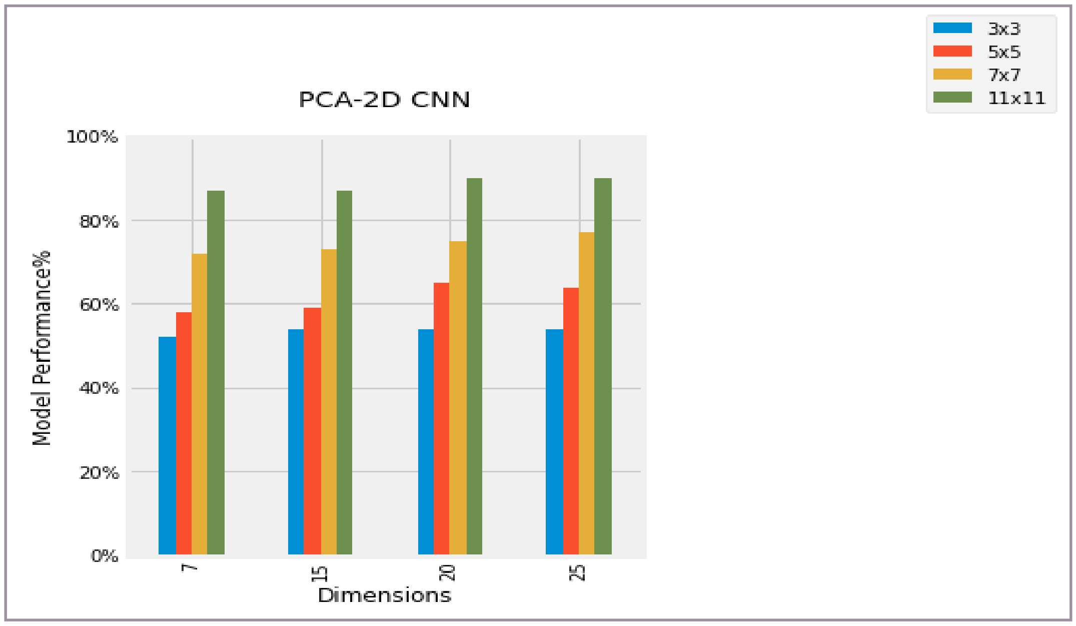


Fig. 7.5: Performance comparison of SAE and PCA on 2D-CNN classifier

Lastly, I have we have taken the best performance of each model and Some Benchmark performance on KSC dataset and plotter them we can see our model performs the best.

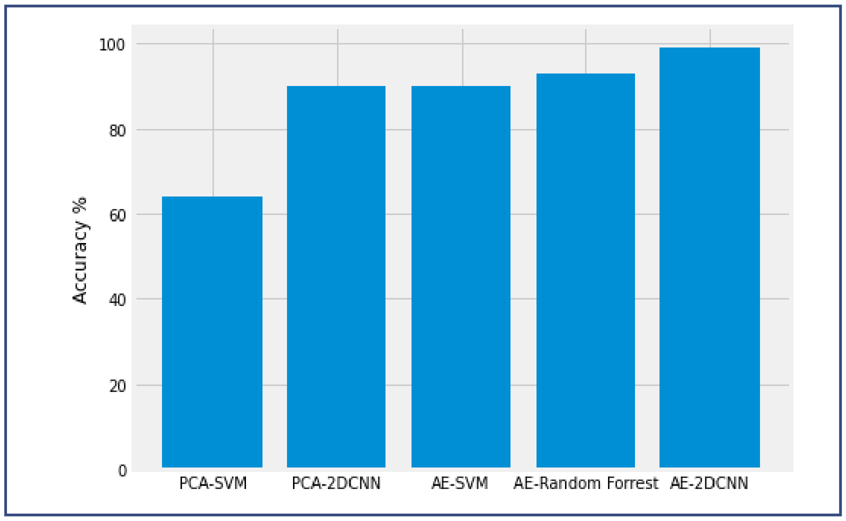


Fig. 7.6: Comparison between proposed hybrid methods along with other benchmark methods

7.6 Conclusion

Observing all the results we see that for KSC dataset it's best to use Deep Learning based approach both in dimension reduction and classification technique. Here we have introduced SAE-2DCNN approach which outperforms all the previous performance on this dataset.

Chapter 8
Conclusion & Future Work

8.1 Summary

In this paper, we have proposed a hyperspectral image classification framework using both Stacked Autoencoder and 2D-Cnvolutional Neural Network. We have also seen combination of these two deep learning approach can extract spectral and spatial information and classify them most efficiently.

PCA is a widely used dimension reduction technique in Deep Learning research field. Autoencoder outperforms PCA in every sector for KSC dataset. From this we can conclude linear dimension reduction technique is not suitable for all types of Hyperspectral data, some of them need non linearity. Our experiments also confirms that autoencoder-extracted representations help lowering error rate of SVM, a classical classifier previously considered as state-of-the-art in this field

Despite PCA being a bad dimension reduction technique for KSC dataset with combination of 2D-CNN it performs somewhat well. Our experiments suggest that deeper representations always lead to better classification accuracies.

8.2 Limitations

The complexity of deep learning models are very high. Combining two deep learning models leads to too many parameters that are needed to be trained.

Both the models take very high time to be trained. Which is a constraint for commercial use.

8.3 Future work

In future we will apply this hybrid classification model to other datasets and see how it performs on different datasets. We will apply our model to other classification tasks and see how it performs. There are various other deep learning models. Applying them on our dataset can produce good result in less time.

8.4 Conclusion

Our proposed deep framework SAE-2DCNN performed very well and succeeded in classifying hyperspectral images with highest accuracies for KSC dataset. Individually SAE or 2D-CNN combined with other liner classifiers can improve their performance too.

References

[1] ‘Remote Sensing: History, Principles and Types’, *Biology Discussion*, May 12, 2016. https://www.biologydiscussion.com/plant-taxonomy/remote-sensing-history-principles-and-types/30587 (accessed Dec. 06, 2020).

[2] ‘hyperspectral-camera-800x290.jpg (800×290)’. https://www.specim.fi/wp-content/uploads/2020/10/hyperspectral-camera-800x290.jpg (accessed Dec. 13, 2020).

[3] ‘Hyperspectral Imaging and its Applications’. http://large.stanford.edu/courses/2015/ph240/islam1/ (accessed Dec. 06, 2020).

[4] R. Gogineni and A. Chaturvedi, ‘Hyperspectral Image Classification’, *Process. Anal. Hyperspectral Data*, Dec. 2019, doi: 10.5772/intechopen.88925.

[5] ‘Feature Extraction for Hyper Spectral Image’. https://a-a-r-s.org/proceeding/ACRS1999/Papers/HSP99-2.htm (accessed Dec. 07, 2020).

[6] G. Hughes, ‘On the mean accuracy of statistical pattern recognizers’, *IEEE Trans. Inf. Theory*, vol. 14, no. 1, pp. 55–63, 1968.

[7] T. Decourselle, J.-C. Simon, L. Journaux, F. Cointault, and J. Miteran, *Noise Robustness of a Texture Classification Protocol for Natural Leaf Roughness Characterisation*. 2012, p. 68.

[8] ‘Hyperspectral imaging’, *Wikipedia*. Dec. 03, 2020, Accessed: Dec. 09, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Hyperspectral_imaging&oldid=992015359.

[9] https://www.whymap.org/EN/Themen/GG_Fernerkundung/Bilder/hyperspektrale_Fe_p_en.html?view=render (Accessed Feb. 07, 2021).

[10] ‘10 Important Applications of Hyperspectral Image’. https://grindgis.com/remote-sensing/10-important-applications-of-hyperspectral-image (accessed Dec. 10, 2020).

[11] F. Lacar, M. Lewis, and I. Grierson, ‘Use of hyperspectral imagery for mapping grape varieties in the Barossa Valley, South Australia’, 2001, doi: 10.1109/IGARSS.2001.978191.

[12] K. T. H. Editor Managing, ‘Five New Technologies for Inspection’, *Food Processing*. https://www.foodprocessing.com/articles/2013/inspection-technologies/ (accessed Dec. 10, 2020).

[13] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, ‘Activation Functions: Comparison of trends in Practice and Research for Deep Learning’, *ArXiv181103378 Cs*, Nov. 2018, Accessed: Dec. 14, 2020. [Online]. Available: http://arxiv.org/abs/1811.03378.

[14] ‘Activation Functions in Neural Networks | by SAGAR SHARMA | Towards Data Science’. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (accessed Dec. 14, 2020).

[15] B. Kumar, O. Dikshit, A. Gupta, and M. K. Singh, ‘Feature extraction for hyperspectral image classification: a review’, *Int. J. Remote Sens.*, vol. 41, no. 16, pp. 6248–6287, Aug. 2020, doi: 10.1080/01431161.2020.1736732.

[16] G. Licciardi, P. R. Marpu, J. Chanussot, and J. A. Benediktsson, ‘Linear Versus Nonlinear PCA for the Classification of Hyperspectral Data Based on the Extended Morphological Profiles’, *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 3, pp. 447–451, May 2012, doi: 10.1109/LGRS.2011.2172185.

[17] C. Rodarmel and J. Shan, ‘Principal Component Analysis for Hyperspectral Image Classification’, *Surv Land Inf Syst*, vol. 62, Jan. 2002.

[18] A. Setiyoko, I. G. W. S. Dharma, and T. Haryanto, ‘Recent development of feature extraction and classification multispectral/hyperspectral images: a systematic literature

review', *J. Phys. Conf. Ser.*, vol. 801, p. 012045, Jan. 2017, doi: 10.1088/1742-6596/801/1/012045.

[19] R. Vaddi and M. Prabukumar, 'Comparative study of feature extraction techniques for hyper spectral remote sensing image classification : A survey', in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, Jun. 2017, pp. 543–548, doi: 10.1109/ICCONS.2017.8250521.

[20] 'Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles - IEEE Conference Publication'. https://ieeexplore.ieee.org/document/4423943 (accessed Dec. 11, 2020).

[21] G. E. Hinton and R. R. Salakhutdinov, 'Reducing the Dimensionality of Data with Neural Networks', *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006, doi: 10.1126/science.1127647.

[22] H. Petersson, D. Gustafsson, and D. Bergstrom, 'Hyperspectral image analysis using deep learning — A review', in *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, Dec. 2016, pp. 1–6, doi: 10.1109/IPTA.2016.7820963.

[23] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, 'Deep Learning-Based Classification of Hyperspectral Data', *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014, doi: 10.1109/JSTARS.2014.2329330.

[24] 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 Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016, doi: 10.1109/TGRS.2016.2584107.

[25] H. Patel and K. P. Upla, 'AECNN: Autoencoder with Convolutional Neural Network for Hyperspectral Image Classification', in *Computer Vision Applications*, Singapore, 2019, pp. 115–128, doi: 10.1007/978-981-15-1387-9_10.

[26] J. Zabalza *et al.*, 'Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging', *Neurocomputing*, vol. 185, pp. 1–10, Apr. 2016, doi: 10.1016/j.neucom.2015.11.044.

[27] Zhouhan Lin, Yushi Chen, Xing Zhao, and Gang Wang, 'Spectral-spatial classification of hyperspectral image using autoencoders', in *2013 9th International Conference on Information, Communications Signal Processing*, Dec. 2013, pp. 1–5, doi: 10.1109/ICICS.2013.6782778.

[28] J.-P. Briot, G. Hadjeres, and F. Pachet, 'Deep Learning Techniques for Music Generation - A Survey', Sep. 2017.

[29] K. Gupta and A. (Advisor) Majumdar, 'Regularized autoencoders', Nov. 2016, Accessed: Dec. 18, 2020. [Online]. Available: https://repository.iiitd.edu.in/xmlui/handle/123456789/493.

[30] A. Cheriyadat and L. M. Bruce, 'Why principal component analysis is not an appropriate feature extraction method for hyperspectral data', in *IGARSS 2003. 2003 IEEE International Geoscience and Remote Sensing Symposium. Proceedings (IEEE Cat. No.03CH37477)*, Jul. 2003, vol. 6, pp. 3420–3422 vol.6, doi: 10.1109/IGARSS.2003.1294808.

[31] Y. Le Cun *et al.*, 'Handwritten digit recognition with a back-propagation network', in *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, Jan. 1989, pp. 396–404, Accessed: Dec. 18, 2020. [Online].

[32] X. Glorot, A. Bordes, and Y. Bengio, *Deep Sparse Rectifier Neural Networks*, vol. 15. 2010.

[33] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, ‘Deep Convolutional Neural Networks for Hyperspectral Image Classification’, *Journal of Sensors*, Jul. 30, 2015. https://www.hindawi.com/journals/js/2015/258619/ (accessed Dec. 18, 2020).

[34] X. Liu, Q. Sun, B. Liu, B. Huang, and M. Fu, ‘Hyperspectral image classification based on convolutional neural network and dimension reduction’, in *2017 Chinese Automation Congress (CAC)*, Oct. 2017, pp. 1686–1690, doi: 10.1109/CAC.2017.8243039.

[35] M. Ramamurthy, Y. H. Robinson, S. Vimal, and A. Suresh, ‘Auto encoder based dimensionality reduction and classification using convolutional neural networks for hyperspectral images’, *Microprocess. Microsyst.*, vol. 79, p. 103280, Nov. 2020, doi: 10.1016/j.micpro.2020.103280.

[36] ‘LeNet-5-1998.png (1552×595)’. https://missinglink.ai/wp-content/uploads/2019/08/LeNet-5-1998.png (accessed Dec. 19, 2020).

[37] ‘Student Notes: Convolutional Neural Networks (CNN) Introduction’, *Belajar Pembelajaran Mesin Indonesia*, Mar. 07, 2018. https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/ (accessed Dec. 19, 2020).

[38] ‘Beginer: The difference between 1D, 2D, and 3D convolution turns out to be this - Programmer Sought’. https://www.programmersought.com/article/63014851723/ (accessed Dec. 19, 2020).

[39] S. Verma, ‘Understanding 1D and 3D Convolution Neural Network | Keras’, *Medium*, Jul. 11, 2020. https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610 (accessed Dec. 19, 2020).