UNIVERSITÀ DEGLI STUDI DI CATANIA
Dipartimento di Matematica e Informatica
Dottorato di Ricerca in Matematica e Informatica XXXII Ciclo

DEVARAM RAMI REDDY

Deep Learning on Hyperspectral Image Classification

Doctor of Philosophy Thesis

Supervisor
Prof. Sebastinao Battiato *Ph.D.*

Anno Accademico 2019 - 2020

Abstract

Classification of Hyperspectral images is one of the main problems in the research field of Remote Sensing and other applications developed through computer vision. With the advantage of spectral and spatial information, it is possible to distinguish effectively different materials on the surface. Since last decade, the intensive employment of Convolutional Neural Networks (CNN) for classification and segmentation tasks led to high-quality results in the field of Hyperspectral Imagery Classification. However, these works are not able to perform satisfactorily on data acquired from various Hyperspectral Imaging Sensors. In this thesis, we propose a novel CNN architecture for HSI pixel-wise classification to improve the robustness and stability of the model to the data obtained from various sensors, thus giving state-of-the-art results. The proposed approach focuses on feature extraction through Dilated Convolution and Transposed Convolution. Moreover, the ELU activation function also played an essential role by activating the neurons with negative input values. Since, to face dataset imbalance problem, we adopt an oversampling strategy that increases the samples in minority classes. To prove the validity of the proposed framework, we tested it on five different HSI datasets and compared the performance with the most successful previous works. Training of the neural network has been performed on various ratios of the train, validation, and test data distribution. The evaluation of the model has been done by Three and Five-Fold cross-validation, and the performances have proven that our approach is competitive with the state-of-art and exhibits the best results on all the employed datasets, which prove that the proposed model is very robust under various Hyperspectral datasets irrespective of their characteristics.

Keywords Deep Learning, Machine Learning, Image Processing, Remote Sensing, Hyperspectral Images, Image Classification, Convolutional Neural Networks, Dilation Convolution, Transposed Convolution, Exponential Linear Unit, Optimization, Resmapling, Principal Component Analysis.

Acknowledgements

Firstly, I would like to express my deepest gratitude and special thanks to the University of Catania and ERSU.

First and foremost, I would like to express my gratitude and appreciation to my supervisors, Prof. Sebastiano Battiato and Prof. Giovanni Gallo, for their great patience, precious guidance, and generous support throughout my doctoral work. All their constant encouragement helped me to focus on my research for which I sincerely thank them. It has been my honor to work with them. I want to express my sincere appreciation to Dr. Dario Allegra for his collaboration and constructive comments in most of the developments presented in this thesis. I have always been impressed by his scientific intensity and active thinking.

I also gratefully acknowledge and appreciate Prof. Filippo Stanco, Prof. Giovanni Maria Farinella, Dr Antonino Furnari and Dr. Daniele Di Mauro; Mostly and especially Dr. Filippo Milotta and Dr. Emilano Spera for their great advice and assistance during my research.

A personal note, a special word of thanks goes to my parents who always stand behind me and support me spiritually throughout my life. My special thanks to Devaram Narsi Reddy, Devaram Prameela, and Devaram Srinivas Reddy, my Father, Mother, and Brother, whose encouragement and guidance always helped me in my personal and professional life. I am grateful for the love, encouragement, and tolerance of Anantha Laxmi and Rami Reddy, who have helped me make a path to higher studies in my life. Without their patience and love, I may not enter into higher studies.

I am especially grateful to Praveen Reddy and Shyam Sundar Reddy Muskula, my cousins, who have been there whenever economical need they helped me a lot. They always have the time to hear about my successes and failures and their suggestions brings me to think beyond the boundaries and has always been a faithful listeners.

Contents

List of Figures

List of Tables

List of Abbreviations

HS	**Hyperspectral**
HSI	**Hyperspectral Imagery or Images**
IMSPC	**Image Spectroscopy**
EMR	**Electromagnetic Radiation**
DL	**Deep Learning**
ML	**Machine Learning**
AI	**Artificial Intelligence**
UV	**Ultra Violet**
NN	**Neural Networks**
CNN	**Convolutional Neural Networks**
ANN	**Artificial Neural Networks**
SVM	**Support Vector Machine**
KNN	**K-Nearest Neighbor**
SL	**Supervised Learning**
LR	**Linear Regression**
LogR	**Logistic Regression**
ELU	**Exponential Linear Unit**
ReLU	**Rectified Linear unit**
LTU	**Linear Threshold Unit**
MLP	**Multi Layer Perceptron**
RMSE	**Root Mean Squared Error**
MSE	**Mean Squared Error**
RGB	**Red Green Blue**
TC	**Transposed Convolution**
MAX	**Maximum**
LDA	**Linear Discriminant Analysis**
PCA	**Principal Component Analysis**

DBN	**Deep Believe Networks**
MAPE	**Mean Absolute Percentage Error**
OSCP	**Orthogonal Complement Subspace Projection**
SMOTE	**Synthetic Minority Over-Sampling**
SNOCC	**Sigma Nearest Oversampling Convex Combination**
DEA	**Divergence-Encouraging Autoencoder**

Chapter 1

Introduction

The objective of this thesis is to develop new and robust Deep Learning (DL) models for the classification of Hyperspectral Imagery (HSI) without regard to the spatial, spectral resolution, and scale of the image. However, the previous research shows us there is a vast variation in the performance of the Deep Learning models concerning the datasets. Moreover, reduce the complexity of the network to maintain memory computation cost without compromising the performance of the model is a challenging task.

Since Convolutional Neural Networks are studied on visual-related tasks. Therefore limited literature on the methods with multiple(small) layers for HSI classification is available.

According to our analysis, standard CNNs, such as only standard convolutional and pooling layers, are not able to extract relevant and robust features for Hyperspectral image classification.

Although Deep Learning models have shown their abilities for HSI classification, still some limitations exist, which reduce the performance of such techniques. In general, DL models require a considerable number of training samples to train a large number of parameters in the networks reliably. On the other hand, having insufficient training samples is a common problem in remotely sensed image classification.

Furthermore, classical CNNs usually use a pooling layer to capture invariant features from the input data, but the pooling operation drops the explicit positional relationship of features.

In Hyperspectral remote sensing, sufficient spectral information and the positional relationship in a pixel vector are the essential factors for accurate spectral

classification. Therefore, it is essential to keep the precise positional relationship in the feature extraction stage.

Besides, when it comes to extracting spectral-spatial features from HSI, it is also essential to hold the positional relationship of spectral-spatial features.

Moreover, most of the existing DL techniques use a scalar value to describe the intensity of a feature. In contrast, proposed networks use vectors to represent features. The usage of vectors improves the feature representation and is a more assuring method for feature learning. These network properties will align with the goals of this study and the current requirements in the hyperspectral community.

To address the above statements, in this thesis, we have found that CNN's can be effectively employed to classify hyperspectral imagery after building appropriate CNN architecture with the help of different layers.

Though a productive fusion of extraction of spectral-spatial feature by standard, transposed, and dilation convolution layers achieve superior performance than standard spatial or spectral only features extraction in the task of remote sensing land cover classification. For various tasks with different hyperspectral data and objectives, relevant spectral and spatial features shall satisfy their individual needs. While spectral information is primary, spatial information is supplementary for remote sensing hyperspectral image classification. However, spatial and spectral structure information is essential for the classification task. Even though nowadays many hyperspectral datasets are available, each of them has different spatial and spectral structures and also concerning to their scales. Spatial and spectral feature extraction approaches arise from computer vision and machine learning, respectively. Deep Learning techniques well aligned for spectral-spatial features fusion, which serve as input to classification and other tasks. Our ultimate aim is to develop efficient and robust hyperspectral image classifier and feature representation methods for hyperspectral imagery concerning different characteristics.

1.1 Contribution

This research objective has been accomplished by specific techniques; listed below.

- It has proved that predicting hyperspectral data to a precise subspace (or union of subspace) can increase the separability of the classes. In addition to dealing

with the problem of high dimensionality, Principal Component Analysis can expertly handle the selection of principal components of high dimensional data. Consequently, we evaluated our model performance with a various number of principal components. Considering fewer principal components may reduce computation cost. However, there is a chance of information loss from the original data. The performance stability of the model may also depend on the precise quantity of principal components concerning the given data. The selection of specific principal components has considered as a contribution of this thesis work.

- The proposed approach has several advantages over the standard deterministic classification, which obtains the "best" class label for each pixel on different levels of hyperspectral data sets. In other words, our approach allows us to assess classification uncertainty. Most importantly, we train the network with small image patches instead of the entire image. This technique proves that it easily transfers local information instead of global information. Because for the pixel level classifiers, local information is more significant. Train the network from small patches also prevents the over-fitting problem.

- Hyperspectral Imagery data sets are mostly unbalanced in terms of samples in individual categories. Data imbalance leads to a reduction in the performance of the classifier to address this problem, and therefore we use a valid resampling or oversampling technique to balance the data set concerning the following category. The effectiveness of this resampling method shows the desired performance of the model.

- Efficiently exploiting the information coming from multiple features is a challenging issue in multiple feature learning approaches. Therefore, to address this issue is a particular case. Specifically, we propose various feature extraction levels to exploit different types of features through the standard, transposed, and dilation convolutional layers with specific parameters being considered.

- Additionally, Exponential Linear Unit (ELU) is one of the main element which can alleviate the vanishing gradient problem during training the network by

identification of positive values. The output values of the ELU with the negative inputs become negative values, so the mean of the outputs of activation function goes near to zero; since the network still able to learn from negative values, which is an advantage during learning. Implementation of ELU is robust to the noise of the input to the network. ELU also reduce the computational complexity by reducing the bias shift effect, since, ELU brings the average gradient closer to the unit natural gradient.

- Finally, We trained on our model with different level data partitioning, such as we used training, validation, and testing samples from 5% and 10% to 90% with the difference of 10% of data and a various number of principal components. To evaluate the model robustness, 3, and 5−Fold cross-validation is an essential technique, in this work the low standard deviation on $5-Fold$ and $3-Fold$ Cross-validation confirm our approach is stable for each of the datasets.

1.2 Thesis Structure

The rest of the thesis is structured as follows. Chapter 2 presents background concepts of Remote Sensing, Deep Learning and Machine Learning. Chapter 3 explores related works on hyperspectral imaging technology, feature extraction in both remote sensing, and classification of hyperspectral imagery. Chapter 4 describes about datasets used for this research. A complete description of proposed approach, algorithms/techniques and technical details are presented in chapter 5. In Chapter 6 gives the experiments and evaluation results of this thesis. Chapter 7 presents details of accepted publication. At last, the conclusion of the thesis and future works are described in Chapter 8.

Chapter 2

Background

2.1 Hyperspectral Imagery on Remote sensing

Over the past several decades, Remote Sensing has experienced dramatic improvements in spatial resolution and acquisition rates. In this thesis, we focused on Remote Sensing imagery classification of Hyperspectral (HS) Images, which differ from traditional images taken by SLR, Mobile, and other cameras. Hyperspectral images(HSI) characteristically described as a spectral sensing method, which is also known as imaging spectroscopy (IMSPC). IMSPC is the simultaneous acquisition of images in numerous narrow contiguous spectral bands, typically in the number of hundreds, and therefore is a three-dimensional cube.

However, RS(Remote-sensing) imagery classification is a complicated method and requires consideration of many factors. The significant steps of image classification and analysis may include perception of a suitable classification system, feature extraction, image pre-processing, selection of training samples, selection of suitable classification approaches, post-classification processing, and accuracy estimation. The users require the computation of the field of study, financial condition, and interpreter's skills are essential factors guiding the selection of RS data, the design of the classification method, and the quality of the classification results. This section describes the understanding of Hyperspectral Images and significant achievements that may be involved in image classification.

2.1.1 Hyperspectral Imaging

Hyperspectral Imaging (HSI) is a method that combines digital imaging with spectroscopy. HSI collects and processes information from over the electromagnetic

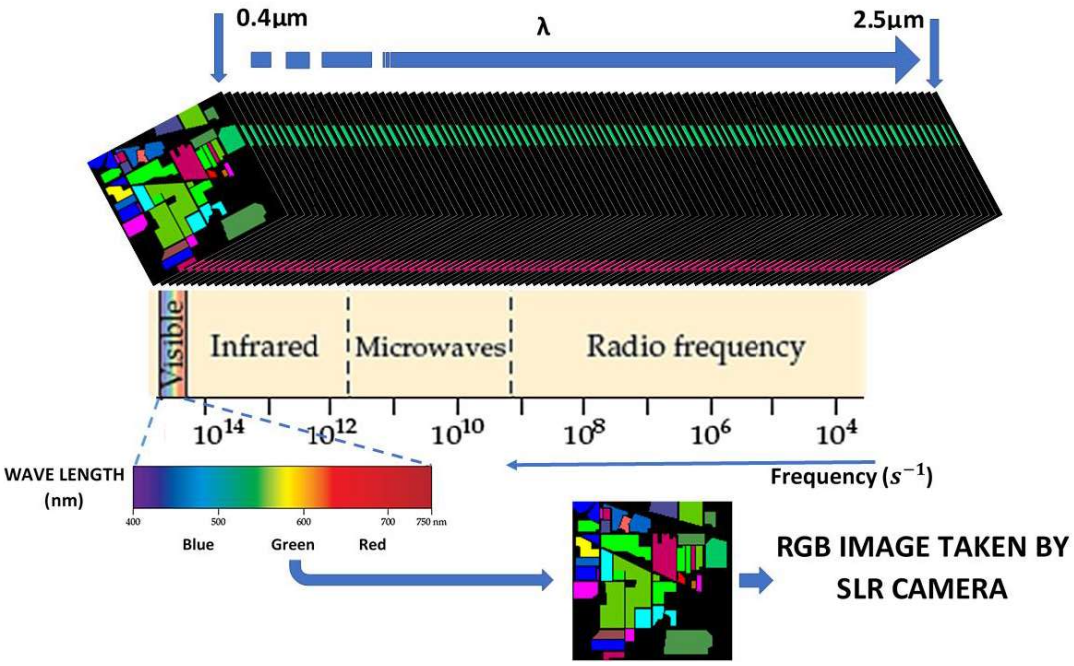


Figure 2.1: spectrum

spectrum as a function of the wavelength, and produces hyperspectral images with instruments called imaging spectrometers. Spectral images form and analyze spectral radiances at each pixel in the scene, where the objects characterized by their spatial shape and their spectral radiance. The spectral radiance of an object is its reflected light intensity as a function of wavelength, which is indicative of the material composing the object. Although the human eye can recognize spatial shape very well, it does not notice spectral radiance characteristics nearly as accurately. Instead, the human eye observes only a dominant part of spectral radiance, which observed as the color of the object. The spectral reflectance of an object, considered

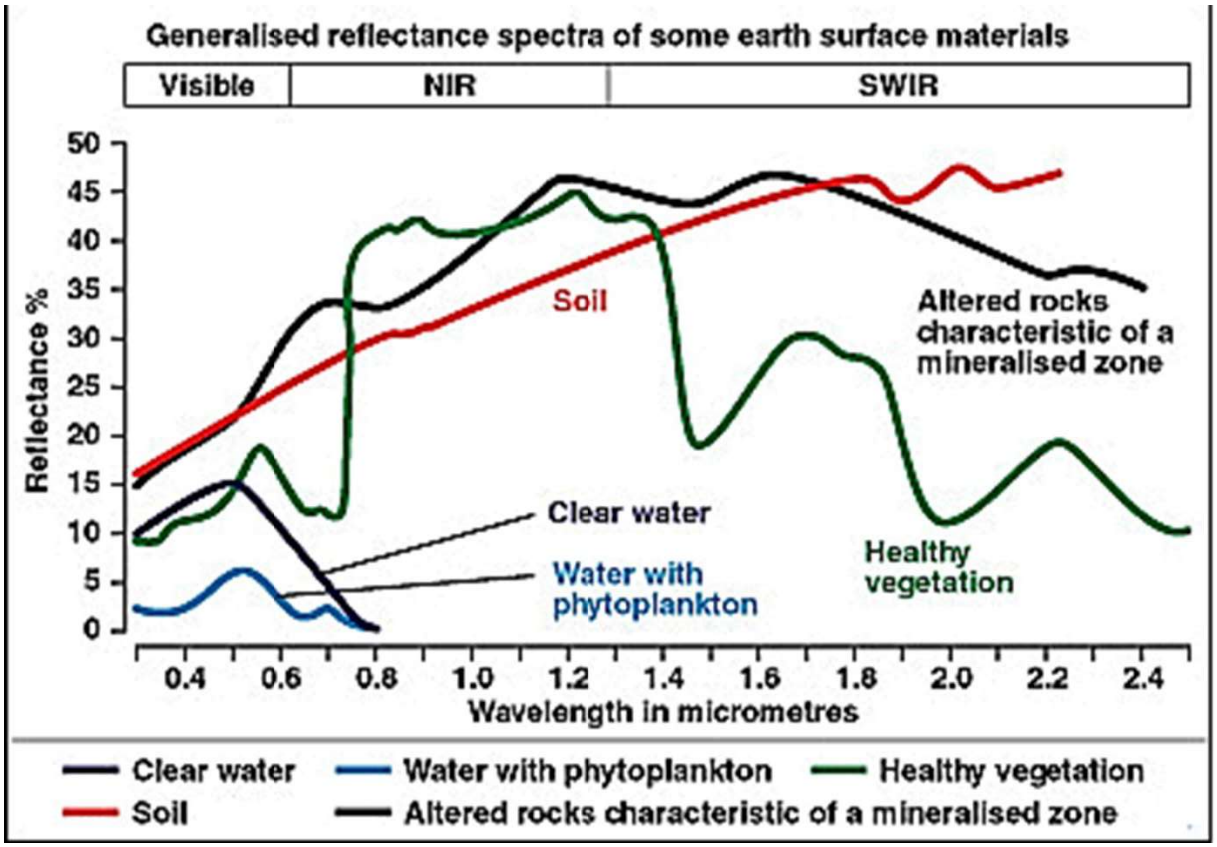


Figure 2.2: spectral signatures

by the value, whcih is the ratio between radiance and the illumination of scene . In the system, most of the objects identified through their spectral reflectance alone. The choice of the appropriate spectral bands is essential for the analysis. Indeed, the deflections of the spectral curves mark the wavelength ranges. Figure: 2.1 for which the material selectively receives the incident energy. These features are generally called absorption bands. The overall shape of a spectral curve and the position and strength of absorption bands in many cases used to identify and assign different materials. For example, vegetation has higher reflectance in the near-infrared range and lower reflectance of red light than soils. However, because the human eye is

a relatively inexperienced observer, it is not always possible to distinguish objects based upon their observed colors. Multispectral remote sensors such as the Landsat Thematic Mapper and SPOT XS produce images with a few moderately broad wavelength bands. Hyperspectral remote sensors, on the other hand, collect image data simultaneously in hundreds or thousands of narrow, adjacent spectral bands. These measurements make it reasonable to assume a continuous spectrum for each image cell. Hyperspectral images contain a wealth of data, but interpreting them requires an understanding of precisely what properties of materials we are trying to measure, and how they relate to the measurements made by the hyperspectral sensor. The essential elements of an imaging spectrometer are shown in Figure: 2.2. The development of these sophisticated sensors has involved the convergence of two related but different technologies: spectroscopy and the remote imaging of Earth and planetary surfaces. Spectroscopy is the study of light emitted by or reflected from materials and its variation in energy with wavelength as applied to the field of optical remote sensing, spectroscopy trades with the spectrum of sunlight that is diffusely reflected (scattered) by materials at the Earth's surface. Instruments called spectrometers (or spectroradiometers) are used to make ground-based or laboratory measurements of the light reflected from the test material.

An optical dispersing element such as a grating or prism in the spectrometer splits this light into many narrow, adjacent wavelength bands, and a separate detector measures the energy in each band. By using hundreds or even thousands of detectors, spectrometers can make spectral measurements of bands as narrow as 0.01 m over a wide wavelength range, typically at least 0.4 to 2.4 m (visible through middle infrared wavelength ranges) in remote-sensing applications. For other applications, such as astronomy, different wavelength ranges can use (e.g., UV, visible, infrared, and radio wavelength ranges). Remote sensing images are designed to focus and measure the light reflected from many adjacent areas on the Earth's surface. Recent advances have allowed the design of images that have spectral ranges and resolutions comparable to ground-based spectrometers.

2.1.2 Spectral Reflectance

In reflected-light spectroscopy, the significant property that we need to obtain is spectral reflectance: the ratio of reflected energy to incident energy as a function

of wavelength. Reflectance changes with wavelength for most materials because the energy at specific wavelengths is scattered or absorbed to different degrees. These reflectance differences are evident when we compare spectral reflectance curves (plots of reflectance versus wavelength) for different materials, as in Figure: 2.2. The overall appearance of a spectral curve and the position and strength of absorption bands in many cases can be used to identify and discriminate different materials. For example, vegetation has higher reflectance in the near-infrared range and lower reflectance in the red range than soils.

2.1.3 Representation of Hyperspectral Images

By the characteristics of HS data, each pixel is a vector; typically, the data represented by a hyperspectral cube. Because of this cubic representation of the data, it is fundamental to consider the use of tensors of order three as a mathematical model for the interpretation of hyperspectral images. Typically, the spatial dimensions respectively associated with 1-mode and 2-mode of the tensor and the spectral dimension associated with the 3-mode of the tensor. The folding matrix in the spectral mode (3-mode) of a tensor is of significant interest in the study of data compared to the folding matrices of the spatial modes. Indeed, the folding matrix in the spectral mode allows a concrete physical representation of the spectral data where each column of the folding matrix represents the spectrum of a pixel, unlike the two folding matrices of spatial modes that are more difficult to interpret, see Figure: 2.1. Indeed, the folding 3-mode matrix is currently using for spectral analysis. However, introducing spatial information often allows for increasing the performances.

To distinguish the material on the surface of the earth depends on the wavelength of the reflected EMR to the sensor, in Figure: 2.2 explores the reflected EMR was having different wavelengths of propagated waves concerning the interaction of the material.

2.2 Machine Learning and Deep Learning

Deep learning (DL) uses a distinct technology, named Multi-Layer Neural Networks, which implementing Machine learning (ML). DL is the field where the machine

can perform various tasks requiring human intelligence; simply machines can learn themselves by mimicking the human brain. The areas of Machine Learning and Deep Learning are both fit into the broader scope of Artificial Intelligence. For better understanding, before going with DL, there are some essential concepts we might know about AI and ML.

Fig-1 illustrates how AI, ML, and DL are related to each other, which includes three concentric circles; consider each circle is one of the three fields. Deep learning is a subset of Machine Learning, which is also a subset of Artificial Intelligence.

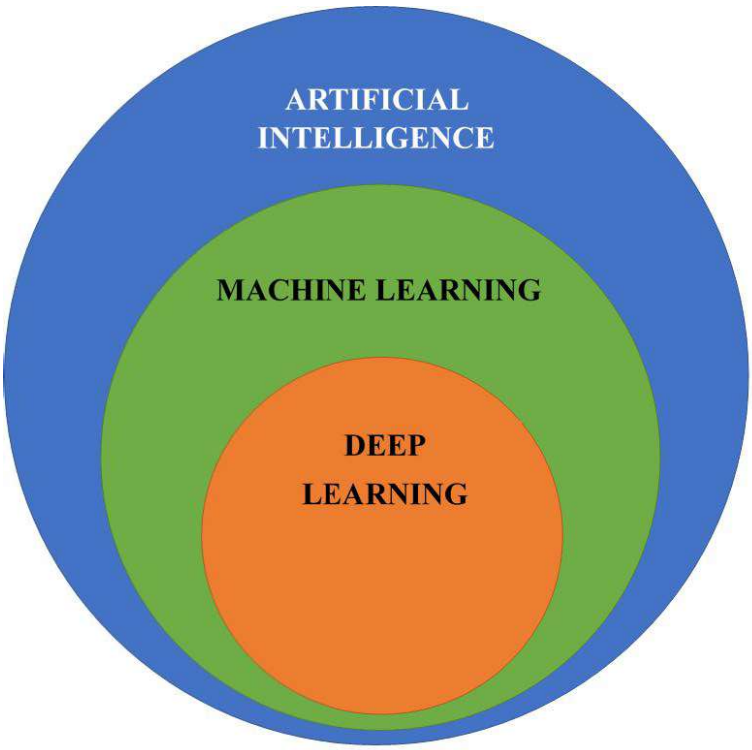


Figure 2.3: Visual understanding the difference between
Artificial Intelligence, Machine Learning, and Deep Learning.

The definition of AI gives us a complete intuition about it, which is, " Artificial Intelligence allows a machine or computer to perform like a human brain, with the help of a predefined computer program." With the influence of AI, machines"

can mimic human behavior in some way. However, AI, including ML, consists of numerous techniques that allow machines or computers to figure things out through the data and produce AI applications. Artificial Intelligence introduced in 1956 by John McCarthy. The aim of AI is a computer or machine to execute tasks, which are the things that required intelligence are perceived uniquely as humans.

However, AI is a fast-growing advancing technology, made potential by the Internet, that may shortly have significant influences on our daily lives. Although, with fundamental techniques of AI, already achieved great success in many applications like Email filtering, Fraud detection, Speech recognition, and personalization. The solution to these problems turns out to be an AI system not only imitate the behavior of humans but mimic how humans can learn.

Here, let us discuss some terms during the implementation of AI, ML, and DL; Techniques, Algorithms, and Models. Techniques are a way of solving problems. For instance, classification is a technique for grouping similar things, and the Decision tree is one of the well-known algorithms which we employed initially.

Initially, researchers work with AI to solve problems like playing Chess and a kind of logic problem. AI techniques contain many rule-based or expert systems; one of the most widely used techniques in machine learning, which introduced in the 1980s. However, Primary techniques of AI like rule-based systems are not able to work well for advanced applications like Image recognition and extract meaning from text.

In this section, we explored distinct algorithms and models referred to as machine learning. Algorithms are a series of instructions and rules developed by programmers to instruct machines to solve the different task-based problems. Technically, Machine learning algorithms differ from traditional algorithms. ML algorithms allow the computers or machines used for new and complicated tasks; for instance, image recognition, classification, or translate pictures into speech.

The significant process of ML is to provide training data to leaning algorithms. Based on inferences from data, learning algorithms produce a new set of rules. The principle of generating a new algorithm with a set of appropriate rules referred to as the ML model. However, different ML models generated with the same learning algorithm by changing instructions from input or training data. For illustration, computers can learn how to translate text from one to another language or predict

the stock market by the same type of learning algorithm.

Although the Internet is one of the keys to the technology behind the tremendous growth of AI, the Internet Community realizes the challenges and opportunities associated with AI is essential to improving the secured Internet that people trust. For example, credit card transaction declined, reduced the personalized emails or ads, and getting recommendations, which we might interest in when online shopping and promotions from companies where we are as a customer, all these situations caused by the working of ML algorithms behind the Internet. Although, a machine learning model may employ incorporate with three standard techniques, such as Supervised Learning(SL), Unsupervised Learning(UL), and Reinforcement Learning(RL).

2.2.1 Supervised Learning

Supervised Learning: SL algorithms such as linear regression and logistic regression, support vector machine (SVM), k-nearest neighbors, decision trees, and artificial neural networks are using in many applications in the field of Data Science and computer vision than unsupervised Learning.

SL learns a function by utilizing available data, that function maps an input to outputs. This type of problem technically we can call it as a function approximation. We can use different kinds of algorithms (e.g., SVM and Neural Networks) and the configuration of algorithms (e.g., network architecture or topology and hyperparameter) to establish the scope of a possible solution that model may produce. Briefly, SL is required when we previously know the response for some input representations, and we need a model that reproduces specific responses for new inputs. If we consider a function y=f(x), where x is input sample, and y is output, the aim is to obtain the function f(), which fits some known samples of x and y.

Besides, SL problems assorted into classification and regression problems.

Classification

Classification: A classification problem is when the output variable is categorized, and classifiers can be Binary classifiers and Multi-Class Classifier. Binary Classifiers performs only on two different classes or two possible outcomes, e.g., such as 'cat' or 'dog,' 'true' or 'false,' and 'yes' or 'no,' 'Positive' or 'Negative.' Multi-Class Classifier

performs on two or more distinct classes, e.g., Classification of Soil Types, Vehicle Types, Object Types, Crop Types.

We have a distinct type of classifiers in the field of ML, such as Naive Bayes Classifier, Support Vector Machine(SVM), K-Nearest Neighbour(KNN), Decision Tree, and Random Forest. SL requires training labeled data. To classify data into labeled groups is the aim of the SL; for this, we need to do label the data initially to train the model. In technical explanation, for classification where each class or label has some probability, that can be weighted through associated cost with each label, than which gives us a final value based on which we can decide to put it some label or not.

Regression

Regression: A regression problem is when the output variable is a real or continuous value, such as 'Weight,' 'height,' 'price,' and 'distance.' The regression problem which predicts the value from continuous data and classification problem aim is to predict the input data which belongs to a class, and both are related to prediction. Regression is a kind of predictive technique that examines the relationship between a target(dependent) and predictor(independent) variables. The simplest model that can apply is linear regression, and it tries to fit data with the best hyperplane, which goes within the points. To modeling and analyze the data, regression is an essential method. We have many types of regression techniques in the literature, such as Linear regression(LR), Logistic regression(LogR), polynomial regression, Ridge regression, Step-wise regression, Lasso regression, and ElasticNet regression. Linear regression and Logistic regression are the usual techniques.

Linear regression concentrates on the conditional probability distribution of the response of the values given by the predictors. The only drawback of the Linear regression is the problem of overfitting. The formulation of LR is $Y' = bX + A$.

Logistic Regression: In early twentieth-century logistic regression used in biological sciences, then applications of social sciences also adopted. The logistic regression aims to find the best fitting model to describe the relationship between the response of the outcome variable(dependent) and predictor variables (Independent). Dependent variables must be binary and are the values we need to predict. Independent

variables are that we expect to affect the dependent variable. When the dependent variables or targets are categorical, logistic regression played an important role, for instance, email filtering, whether the email is spam or not. Consider the following example; if we need to classify the above task with linear regression, there is a need for introducing a threshold value based on which classification of the task. For example, if the actual class is spam and predicted constant value is 0.3 and the value we fixed as the threshold is 0.5, the classified data point is not-spam, this kind of situation leads to severe consequence in real-time; if in the case of the medical field. However, to address the above problem, logistic regression used for the classification problem since the range of the value strictly between 0 to 1.

2.2.2 Unsupervised Learning

Unsupervised Learning is the training of machine utilizing data that is neither classified nor labeled, which letting the algorithm to work on that data without guidance. Unlike SL, there no training provided for the machine. Therefore the machine is bound to observe the hidden structure in unlabeled information by our-self. UL categorized into two types of algorithms, which are Clustering and Association.

Since these algorithms solving many problems through ML, it is about how humans learn, but some tasks like speech and handwritten recognition were still hard for machine intelligence employing AI and ML compare with human intelligence. Here, one idea changes the future; machines can mimic how humans learn, why not try to mimic the human brain.

The above idea is the primary source behind developing Neural Networks (NN).

2.3 Artificial Neural Network

Neural Networks(NN) or Artificial Neural Networks are a sort of algorithms modeled loosely similar to the form of how the brain works. ANN is a computational model based on the functions and structure of biological Neural Networks. Neural networks attained great success in many applications. The fundamental advantage of NNs, they can learn how to recognize patterns. For instance, the NN algorithm can learn to recognize the given image contains an object, whether it is a Cat or Dog.

The biological neural system motivates artificial neural networks (or neural networks). Neural networks consist of an input layer, hidden layers, and an output layer. [1] described that in a neural network, a single hidden layer, including a finite number of neurons, could approximating any continuous function to any desired accuracy.

In this part, we can understand basic intuition behind the NN, how it works, and the impact of NN in the digital world.

Artificial neural networks (ANN) assume the essential characteristics of neurons and their interconnections. We then typically program a computer to mimic these characteristics. ANNs typically made of interconnected "blocks or units." ANN is an artificial human neuro system to receive, process, and transmit information in terms of computer science.

Artificial Neural Networks are at the heart of Deep Learning. ANNs are very powerful and scalable, and which are ideal for handling complex ML tasks similar to classification, recognition, recommendations, and also for new generation gaming technology. For example, ANN able to classify massive or billions of images.

ANNs are first introduced in 1943 by Warren McCulloch and Walter Pitts. The fist ANN architecture composed concerning how biological neurons might work mutually in animals to make complex computations employing propositional logic. The early-stage 1960s of using ANNs might reduce the popularity, but in the early1980s, there was a rebirth of attention in ANN as new network architecture and better training methods invented. Though by the 1990s, there was a powerful machine learning technique introduced, were favored by most researchers such as SVM. SVM offers more reliable results and strong theoretical foundations. However, since the advantage of Data availability, the usage of ANNs is expanding drastically, as compared to ANNs often outperform on complex problems than other ML techniques.

Let us see a simple ANN architecture; called Perceptron.

Before going to know about Perceptron, Let us start with the most significant five essential basic building blocks to design an ANN, which are Neuron, Input, Weights, Bias, and Output.

Neuron: what does a neuron do?

Initially, look at the Figure: 2.4 it takes the input and multiplies them by their

weights, then add all multiplied values. Later, the activation function applies to the sum.

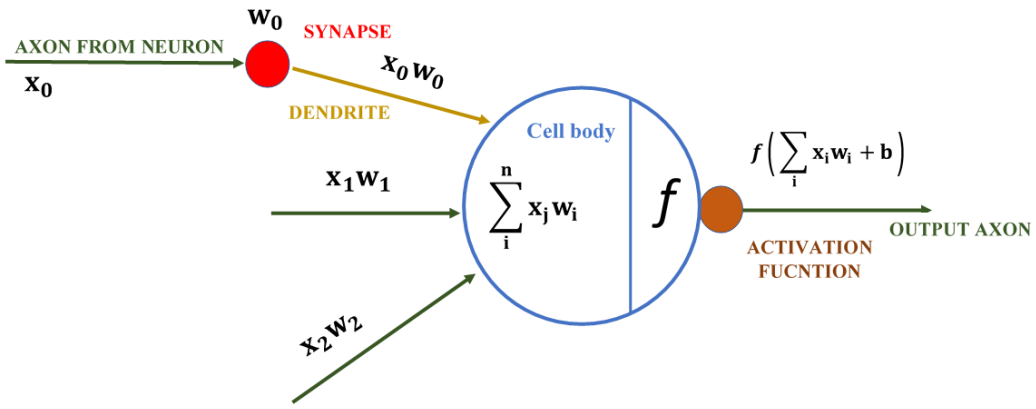


Figure 2.4: Neuron

The neuron aims to modify the weights, based on numerous samples of inputs and outputs. So let us say we show the neuron thousands of instances of DOG images and NON-DOG images, and for each of these examples, we explain what features are present in the images and how well we are assured they occur.

Based on the data we gave as an input neuron determines, Some features are crucial and decisive (for instance, every drawing of the dog had a tailpiece in it, so the weight must be positive and substantial). Some features are not necessary (for instance, only a few images had two eyes, so the weight must be small). However, some features are essential and detrimental (for instance, every image containing a horn has been a picture of a Unicorn, not a Dog, so the weight must be significant and negative).

2.3.1 Perceptron

Perceptron is an Early Deep Learning Technique Figure: 2.5, which introduces in 1957 by Frank Rosenblatt, it has Linear Threshold Unit(LTU).

Consider input and output are numbers, and Individual weights are associated with all input connections. A weighted sum of its inputs Equation: 2.2 estimated by LTU then utilizes step function(Activation function) to the sum and produce the outcomes Equation: 2.3

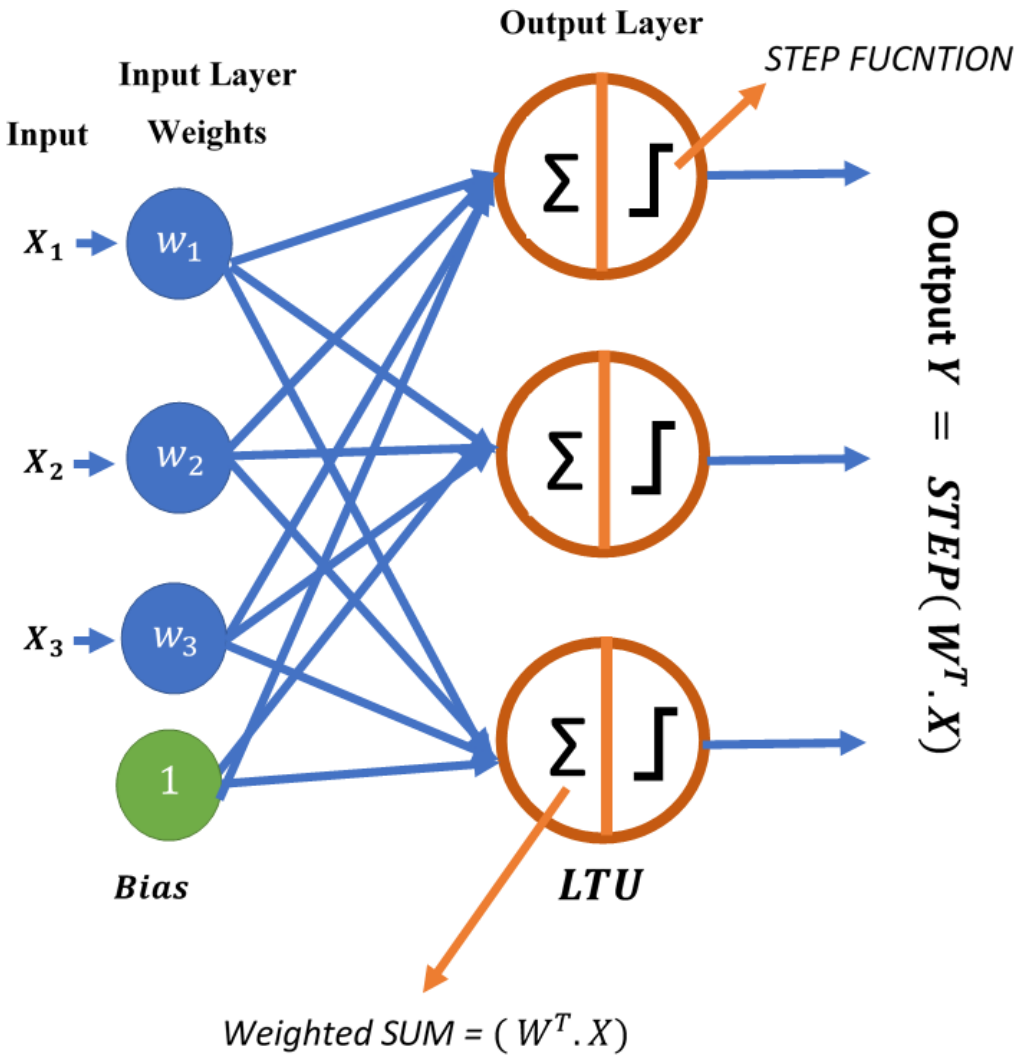


Figure 2.5: Perceptron

$$WeightedSum = w_1x_1 + w_2x_2 + +w_nx_n = w^T \cdot x \tag{2.1}$$

$$output(Y) = step(WeightedSum) = step(w^T \cdot x). \tag{2.2}$$

Figure: 2.5 illustrates, it contains three inputs and outputs, also called a multi-output classifier, which classifies multiple samples simultaneously into three binary classes. Learn the rule of the perceptron inspired by Hebbian Learning (Hebb's

Rule). Hebb's Rule: The connection between the two neurons increased whenever they have the same output. However, perception implements modified Hebb's Rule Equation: 2.3, which considered a single sample at a time and made predictions individually and the wrong prediction of each neuron; it reinforces the connection of the weight from the input, which would have contributed to the correct prediction.

$$w_{i,j}^{(NextStep)} = w_{i,j} + \eta(y_j - \widehat{y_j})x_i \tag{2.3}$$

Where x_i is the i^{th} input of the existing training sample and $w_{i,j}$ is the weights association between the i^{th} input neuron and j^{th} output neuron. y_j is target output, and $\hat{y_j}$ is the output of the i^{th} neuron for the existing training sample, and η is the learning rate of the training network.

However, the perception is not able to learn intricate patterns because the decision boundary every output neuron is linear. Perceptron may not be able to explore class probability as output; it just delivers predictions based on the strict threshold. For this reason, logistic regression is the right choice for complex tasks.

2.3.2 Multi-Layer Perceptron.

To address the limitations of the perceptron later works implemented a new architecture. The following work with a multi-layer perceptron (MLP), also called the feed-forward neural network, has proved that they can approximate an XOR operator as well as many other non-linear functions. The MLP is the welcome environment of deep learning: an excellent place to start when we are learning about deep learning. Figure: 2.6, which stack multiple perceptron together. Note that contrary to Perceptron, MLP able to solve the problem with XOR. Formally, a single-hidden-layer MLP is a function $f : R^D \rightarrow R^L$, where D is the size of the input vector X and L is the size of the output vector $F(x)$, such that, in matrix notation:

$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)}) + w^{(1)}x))) \tag{2.4}$$

where G and Ss are activation functions, $w^{(1)}w^{(2)}$ are weight matrices and $(b^{(2)}(b^{(2)}$ represents the bias vectors.

MLP is a deep artificial neural network. It consists of more than one perceptron, which is composed of an input layer to collect the data and an output layer that provides a prediction about the input, and in between those two, an optional number of hidden layers that are the actual computational processors or workers of the MLP. MLPs with one hidden layer able to approximate any continuous function.

To train an MLP, we get familiar with all parameters of the model, and here we utilize Stochastic Gradient Descent with mini-batches. To obtain the gradients, the set of parameters $\Theta = W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}$ should learn through back-propagation algorithm.

Multi-Layer perceptron often used over supervised learning problems; they train on a collection of input-output combinations and learn to model the relationship among those inputs and outputs. Training includes altering the parameters(weights and biases) of the model to minimize loss. Back-propagation used to make those adjustments within parameters relative to the error, and the error itself measured with some techniques, including by root mean squared error (RMSE).

Training a perceptron. The several basic deep learning algorithms for supervised training of the MLP is known as back-propagation. The fundamental procedure to train the perceptron. Initially, A training example presented and propagated ahead through the network. Then the output error is determined by the usual metric called mean squared error(MSE):

$$Error = \frac{1}{2}(y - \widehat{y})^2 \tag{2.5}$$

Where y is the target value, and $\hat{y}$ is the actual output. MSE is an excellent choice to calculate the error than other methods. To minimize the network error, we can use Stochastic Gradient Descent or other optimizers, in early works, SGD is a standard optimization method.

The aim is to update the weights of the model to minimize the loss function. The weights are updated using a back-propagation algorithm, which we will ll study next.

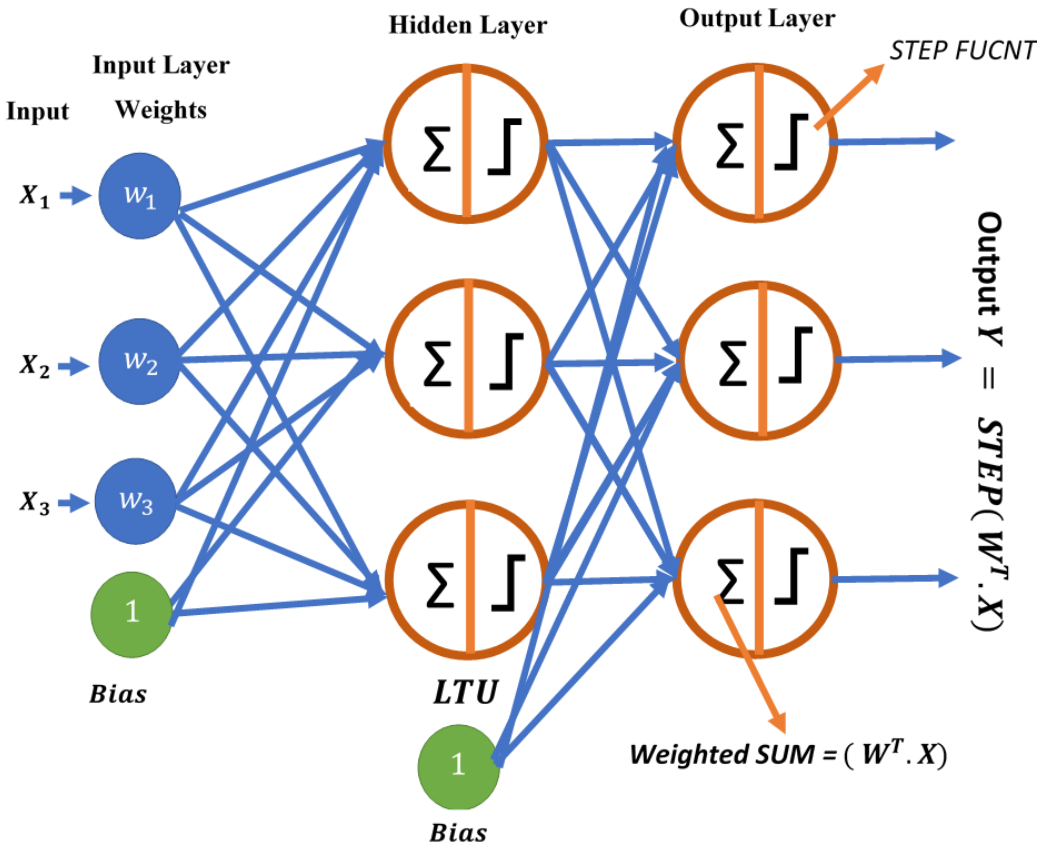


Figure 2.6: Multi-Layer Peceptron

2.3.3 Back-Propagation

Back-Propagation (BP) algorithms operate by discovering the loss (or error) at the output and then propagating it back into the network. The weights are updated to decrease the loss resulting from each unit. Let us try to give some intuition into how BP works. The primary step in minimizing the error is to determine the gradient of each node concerning the final output. Since it is a multi-layer network, identifying the gradient is not very simple. To get the gradients for multi-layer networks. Let's take a step back of neural networks and consider a straightforward system as following Equation: 2.7,2.10, and 2.13: To understand more specifically, The CNN requires to modify and update its kernel parameters, or weights, for the given training data. Back-propagation is an efficient method for computing gradients required to perform gradient-based optimization of the weights in neural

networks [2]. The distinct combination of weights that minimize the loss function (or error function) is the solution to the optimization problem. The method requires the computation of the gradient of the error function at each iteration. Since the loss function should be both continue and differentiable at every iteration step. The initial weights of an untrained CNN randomly chosen. Consequently, before training, the neural network cannot make meaningful predictions for network input, as there is no relation between an image and its labeled output yet. By exposing the network to a training data set, comprising images and their labeled outputs with correct classes, the weights adjusted. Training is the adaptation of the weights in such a way that the difference between the desired output and network output minimized, which means that the network trained to find the right features required for classification. There are two computational phases in a neural network, the forward pass and the backward pass in which the weights adapted.

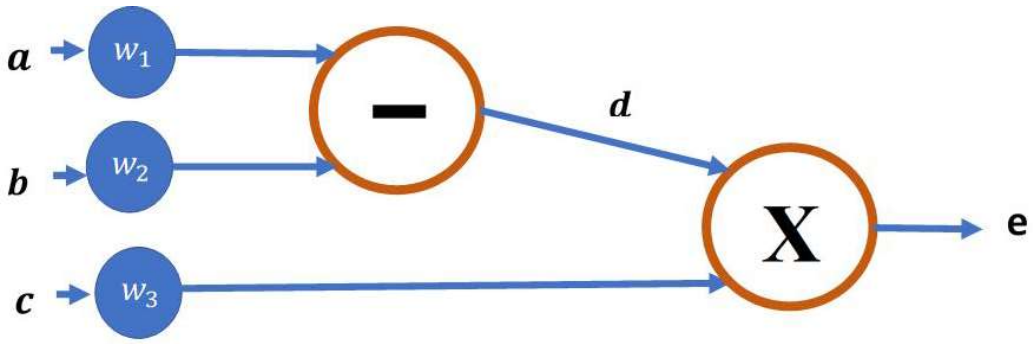


Figure 2.7: Processing of data

Consider the Figure: 2.7 we have three inputs which simple processing as:

$$d = a - b \tag{2.6}$$

$$e = d * c = (a - b) * c \tag{2.7}$$

Now we require to determine the gradients of a, b, c, and d concerning the output e. The following cases are very straight forward:

$$\frac{\partial e}{\partial d} = c | \frac{\partial e}{\partial c} = d \tag{2.8}$$

$$\tag{2.9}$$

$$\frac{\partial d}{\partial a} = 1 | \frac{\partial d}{\partial b} = -1 \tag{2.10}$$

However, for determining the gradients for a and b, we need to apply the chain rule.

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial d} * \frac{\partial d}{\partial a} = c \tag{2.11}$$

$$\tag{2.12}$$

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial d} * \frac{\partial d}{\partial b} = -c \tag{2.13}$$

To compute gradients through multiplying the input gradient, over a node with that of the output of that node.

How error Back propagate to the network. The error for layer L-1 should be defined using the following equation:

$$e_{L-1}^{(i)} = \left(\sum_{k=1}^{N_L} W_{ik}^{L-1} \cdot e_L^{(i)} \right) * f^{'}(x)^i \tag{2.14}$$

The error for layer $L - 1$ should be defined using the following equation: • We recognize that the gradient of a node is a function of gradients of all nodes from the next layer. Here, error in a particular node based on the weighted sum of errors on

all the nodes of the next layer, which take the output of this node as input. Since errors are calculated using the gradients of each node, the factor comes into the picture. • $f'(x)(i)$ relates to the derivative of the activation function for the inputs getting into that node. Note that x refers to the weighted sum of all inputs in the present node, before applying the activation function. • By using the chain rule, the multiplication of the gradient of the current node, i.e., $f'(x)(i)$ with that of following nodes which comes from the first half of the right-hand side of the equation.

This process has to be repeated consecutively from the $L - 1^{th}$ layer to the $2nd$ layer. Note that the first layer is just the inputs.

2.3.4 Weights

(Parameters) — A weight describes the strength of the connection between neurons or units. If the weight from node one to node two has a higher magnitude, it means that neuron one has a more critical influence over neuron two. A weight takes down the effect of the input value. Weights near zero means, changing this input will not modify the output. Negative weights indicate that raising this input will decrease the output. Weight determines how much impact the input will have on the output.

2.3.5 Bias

It is an extra input to neurons, and it is always one and has its connection weight. It makes sure that when all input data are none (all 0's), there is going to be activation in the neuron.

2.4 Convolutional Neural Networks

Convolutional Neural Networks(CNN) or Convolutional Networks are particularly existing and a standard class of feed-forward networks. CNN's are very well-suited for image classification, image recognition, and other computer vision applications.

Before going for the exact structure of convolutional networks, first, describe an image kernel or filter with associated weights. A kernel utilized over a complete

input image and often utilizes multiple filters. For instance, if we use four 6x6 filters to a given input image, and the output pixel with coordinates 1,1 is the weighted sum of a 6x6 square of input pixels with the top left corner 1,1, and the weights of the filter are also 6x6 matrix. Output pixel 2,1 is the result of an input square with the top left corner 2,1 and so on.

For better intuition, let us look at the following example. In the following Figure: 2.8 considers the green color represents an Image Dimensions of 5 (Height) x 5 (Breadth) x 1 (Number of channels, e.g. Grayscale Image, Image with the single-channel). The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in color yellow. We have selected K Dimensions = 3 (Height) x 3 (Breadth) x 1 (Number of channels) matrix.

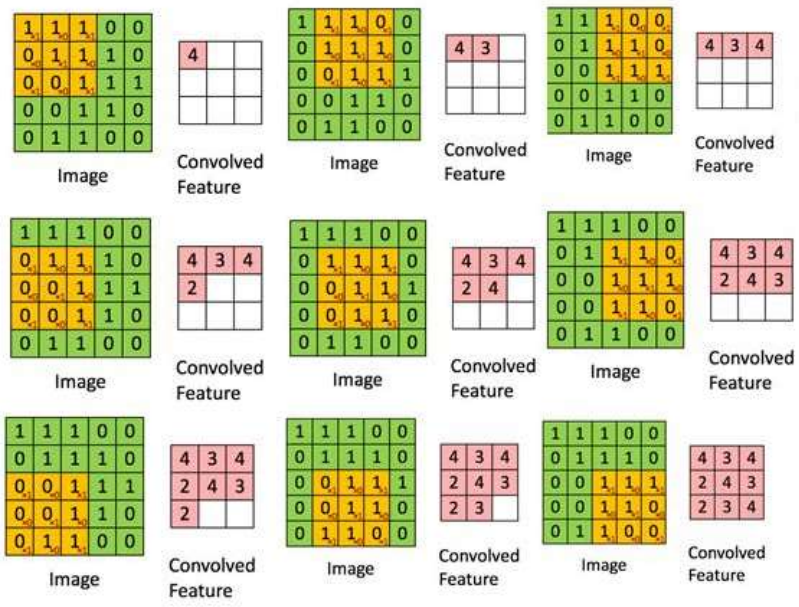


Figure 2.8: Convoluting $5 \times 5 \times 1$ image with a $3 \times 3 \times 1$ kernel

$$Kernel/Filter, K = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{2.15}$$

Each neuron operates in its receptive field and connected to the following neurons to reach the entire visual field. Precisely, each neuron reacts to motivates only in the limited region of the visual field called the receptive field. Individual neurons in CNN processes information only in its receptive field as well. The layers are organized in such a way so that they recognize more modest patterns (lines, curves) and more elaborate patterns (faces, objects).

Let us see a simple CNN architecture in Figure :2.9.

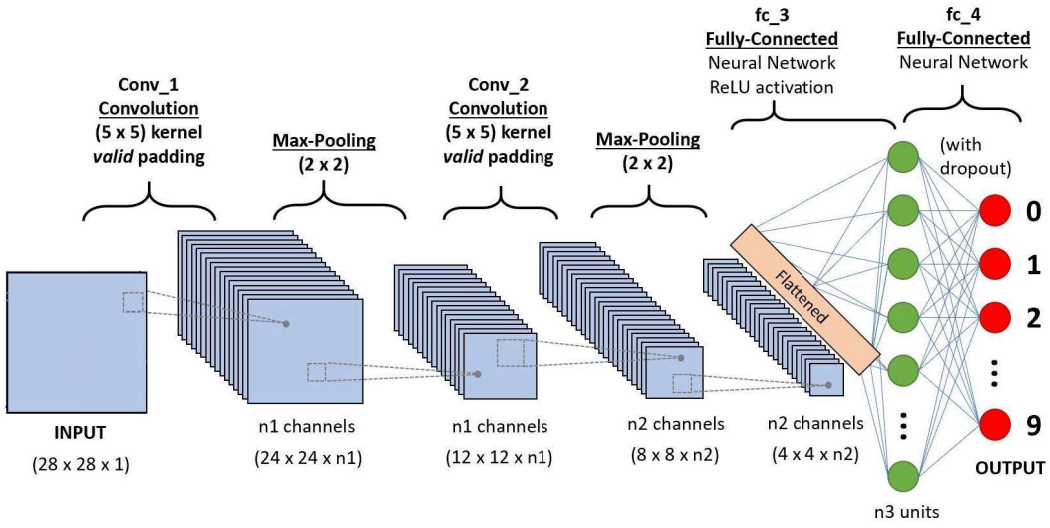


Figure 2.9: An example CNN with two convolutional layers, two pooling layers, and a fully connected layer which decides the final classification of the image into one of several categories

A more specific summary of what CNN's do, it would take the image as input,

pass it into a sequence of convolutional, non-linear, pooling, and fully connected layers, and generate an output. The output can be a single class or a probability(multi) of classes that best represents the image. Now, the hard part is knowing about what each of these layers does. So 'let us get into know about some elements on CNN.

2.4.1 Input

When a computer takes an image as input, the computer understood as an array of pixel values. Depending on the resolution and size of the image, it sees a $32 \times 32 \times 3$ array of numbers (The three refers to RGB values). Let us say we have an image(RGB, color), and 32×32 dimensions. The representative array is $32 \times 32 \times 3$. Each of these numbers is given a value from $0to255$, which describes the pixel intensity at a specific coordinate. These numbers, while insignificant for human, when we perform image classification, are the only inputs available to the computer. The idea is that give those array of numbers to the computer, and it produces numbers that represent the probability of the image being a distinct class.

The CNN in Figure: 2.9 is related in architecture to the original LeNet, and for instance, it classifies an image into four categories: cat, dog bird, or boat (the original LeNet was used mainly for character recognition tasks). From the input image, the network accurately specifies the highest probability of an object, e.g., cat(0.94) amongst all four classes. Technically the sum of all probabilities in the output layer should be one.

There are four primary operations in the CNN shown in Figure:2.9

1. Convolution 2. Non Linearity (ReLU) 3. Pooling or Sub Sampling 4. Classification (Fully Connected Layer) The above operations are the fundamental building blocks of every ConvNet, so discovering how this work is an essential start to developing an in-depth knowledge of ConvNets. Let us get the intuition behind each of these operations.

Channel is a general term used to refer to a specific component of an image. An image from a regular digital camera has three channels – red, green, and blue – imagine those as three two dimensional-matrices stacked over each other (one for

each color), each holding pixel values in the range $0 to 255$.

A grayscale image has only one channel. Consider grayscale images until this explanation, so we have a single $2d$ matrix representing an image. The value of every pixel in the image or matrix between $0 to 255$ – zero represents black, and 255 represents white.

2.4.2 Convolution

The first layer in a ConvNet is always a Convolutional Layer. Convolution layer presents to extract features from an input. Convolution collects the relationship between pixels by learning image features using small squares of input data. Convolution is a mathematical operation that takes two inputs, such as an image matrix and a filter or kernel. Consider a 5 x 5 green matrix is an image, with pixel values, are 0, 1, and the green matrix is a filter matrix 3 x 3, as shown in Figure: 2.9.

The Convolution Operation aims to extract high-level features such as edges, through input image. CNN's also operate with more than one convolutional layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation. By adding more layers(hidden layers), the architecture can extract to the High-Level features as well.

As we explained above, every image represents a matrix of pixel values. Consider a 5 x 5 image with pixel values between 0 and 1. The convolution of the 5 x 5 image and the 3 x 3 matrix can compute, as shown in the Figure: 2.8.

Take a bit to understand how the computation in Figure: 2.8 is being done. We slide the orange matrix(consider the orange matrix is a kernel or filter) over our green matrix (consider the green matrix is an original image) by individual pixel ('stride'). For every pixel, we compute element-wise multiplication (between the two matrices) and add the multiplication outputs to perceive the final entity, which forms a single element of the output pink color matrix (which represents as a feature map). It is evident from the Figure: 2.8 that various values of the kernel matrix produce distinct Feature Maps for the corresponding input image. It is necessary to understand that the Conv operation obtains the local dependencies from the original or input image. Consider we have two different kernels/filters and one input image.

Additionally, see how these two filters generate distinct feature maps from the same input image. Consider, the image and the filters are entirely numeric matrices.

In practice, while training the network, a CNN can learn the values of those kernels (although we still need to specify parameters such as the number of filters, size of the filter, and network architecture). The large number of filters we have, the more image features can be extracted, and the better our network improves at recognizing patterns in unseen images, it leads to increase computation complexity.

The Feature Map (Convolved Feature) size depends on three parameters that we require to choose before the convolution operation performed.

2.4.3 Convolution Operation

The Convolution Operation aims to extract high-level features such as edges, through input image. CNN's also operate with more than one convolutional layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation. By adding more layers(hidden layers), the architecture can extract to the High-Level features as well.

Strides Stride is the number of pixels shifts across the input image. When the stride is one, then we move the filters to one pixel at a time from to right side. When the stride is two, then we move the filters to two pixels at a time and so on. The below fig: Conv_1 shows convolution would work with a stride of one

Padding Sometimes filter does not fit perfectly with the input image. We have two options: Pad the picture with zeros (called zero-padding). Sometimes, it is beneficial to pad the input matrix with zeros around the border so that we can apply the filter to bordering elements of our input image matrix. Zero-padding also called wide-convolution, which allows us to control the size of the feature map. Leave the portion of the image wherever the filter does not fit, called valid padding, which holds only a valid part of the image.

Depth Depth corresponds to the number of kernels we use for the convolution operation. In the network shown in fig: depth, we are performing a convolution of the original image using three different filters, which offers three separate feature

maps. These three feature maps stacked two-dimensional matrices; therefore, the 'depth' of the feature map would be three.

2.4.4 Pooling

Comparable to the Convolutional Layer, the Pooling layer is capable of reducing the spatial dimension of the convolved Feature. It leads to reduce the computational cost required to process the data through dimensionality reduction. Furthermore, it is beneficial for extracting robust features that are rotational and positional invariant, thus keeping the process of effectively training of the model.

There are two types of Pooling layers: Max and Average Pooling. Max Pooling yields the maximum value, and Average Pooling yields the average of all the values from the portion of the image covered by the filter. Max Pooling also acts as a Noise Suppressant. It rejects the noisy activation's collectively and also makes de-noising along with dimensionality reduction. In the case of Max Pooling, we define a spatial region (for example, a 22 window) and take the most significant element from the rectified feature map within that window. Instead of taking the most significant element, we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has shown to work better.

Average pooling performs dimensionality reduction as a noise suppressing mechanism. Therefore, we can say that Max Pooling performs better than Average Pooling.

2.4.5 Loss Function

There are different loss functions possible for different purposes. In this part, we go through some of the very often used loss functions, with a set of examples.

The loss function assists in optimizing the parameters of the neural networks. The loss function aims to minimize the loss for a neural network by optimizing its parameters or weights. The loss is estimated using loss function by matching the target or actual value and predicted value by a neural network. Then we use the gradient descent method to optimize the weights of the network such that the loss minimized. It is the way to how we train a neural network.

Mean Squared Error(MSE) When we have a regression problem, one of the loss function we can use ahead is MSE. As the name implies, MSE loss is measured by using the mean of squared differences between actual and predicted values. For instance, we have a neural network; it takes some data related to the house and predicts its price. In this case, we can use MSE loss. Primarily, in the case where the output is a real number, we should use the MSE loss function.

Binary Crossentropy(BCE) When we have a binary classification task, one of the loss function, we can use that is BCE. If we use the BCE loss function, we need one output node to classify the data into two classes Equation: 2.16 The output value should be pass through a sigmoid activation function, so the output is in the range of between 0 and 1.

$$-(y \log(p) + (1 - y) \log(1 - p)) \tag{2.16}$$

For example, we have a neural network that takes data related to the atmosphere and predicts whether if there is rain or not. If the output is more than 0.5, the network classifies there is rain, and if the output is less than 0.5, the network classifies there is no rain. More the probability score value, the more the chance of raining. During training the network, the target value fed to the network should be one if it is raining otherwise zero. One important thing, if we are using the BCE loss function, the output of the node should be between 0 and 1. It means we have to use a sigmoid activation function on our final output. Since sigmoid converts any real value in the range between 0 and 1. What if we are not using sigmoid activation on the final layer? Then we can pass an argument called from logits as true to the loss function, and it internally applies the sigmoid to the output value.

Categorical Crossentropy(CCE) When we have a multi-class classification problem, CCE is one of the loss function we can use. If we are using CCE loss function, there must be the same number of output nodes as the classes. Moreover, the final layer output should be passed through a softmax activation so that each node output a probability value between 0 and 1. For example, we have a neural network that takes an image and classifies it into a cat or dog, i.e., If N¿ 2, we determine a separate loss for every class label per observation and sum the result by Equation: 2.17. If the cat node has a high probability score, then the image is

classified into a cat otherwise dog. Primarily, whichever class node has the highest probability score, the image is classified into that class.

$$-\sum_{c=1}^{N} y_{o,c} \log(p_{o,c}) \tag{2.17}$$

For feeding the target value at the time of training, we have to encode them one-hot. If the image is of the cat, then the target vector would be (1, 0), and if the image is of dog, the target vector would be (0, 1). Mostly, the target vector would be of the same size as the number of classes, and the index position corresponding to the actual class would be 1, and all others would be zero. What if we are not using softmax activation on the final layer? Then we can pass an argument called from logits as true to the loss function, and it internally applies the softmax to the output value. Same as in the above case.

2.5 Image Classification

2.5.1 Classification Approaches and selection of training data

A satisfactory classification system and an adequate number of training samples are pre-requirements for a successful classifier. There are three significant obstacles identified in [3] when medium spatial resolution data used to the classification of vegetation: adequate hierarchical levels settings for mapping, representing discrete land-cover factors perceptible by selected RS, and choosing representative training approaches. In usual, a classification approach is designed based on the user's requirement, a spatial resolution of selected RS, adaptability with previous work, image-processing and classification algorithms accessible, and time limitations. Such a system should be instructive, exhaustive, and detachable [4]. In various instances, a hierarchical classification approach utilized to take different requirements into account. An adequate number of training samples and their representatives is essential for image classification [5]. Training samples usually collected from fieldwork or by high spatial resolution aerial and satellite images. Various collection approaches, such as single-pixel, seed, and polygons, may be used, but they would affect classification results, particularly for classifications with high spatial resolution image data

[6]. When the perspective of a study area is complex and varied, selecting sufficient training samples becomes complicated. This problem will be complicated if medium or coarse spatial resolution data are used for classification, because a massive volume of mixed pixels may happen. Hence, training sample choice must consider the spatial resolution of the RS data used, availability of ground reference data, and the complexity of perspectives in the study area.

2.5.2 Data Selection for Remote Sensing

Remote Sensing includes both airborne and spaceborne sensor data; they are different in spatial, radiometric, spectral, and temporal resolutions. Understanding the intensities and flaws of different types of sensor data is necessary for the selection of proper remotely sensed data for image classification. Previous research [7] has examined the characteristics of significant types of remote-sensing data. For example, [8] studied the properties of different remote-sensing data in spectral, radiometric, spatial, and temporal resolutions, polarization, and angularity. Choosing proper sensor data is the first substantial move to successful classification for a specific purpose [7] [9] [10]. It needs considering such factors as user's requirement, the scale, and aspects of a study area, the availability of multiple image data and their properties, cost and time limitations, and the interpreter's experience in using the selected image. Scale, image resolution, and the user's need are the most crucial factors influencing the selection of RS. The user's need defines the nature of classification and the scale of the research area, thus influencing the selection of proper spatial resolution of RS. Prior research has explored the consequences of scale and resolution on RS image classification [11]. Overall, a fine-scale classification system needed at a local level. Thus high spatial resolution data such as IKONOS and SPOT 5 HRG data are helpful. At a regional scale, medium spatial resolution data such as Landsat TM/ETM+, and Terra ASTER data are frequently using. Though, continental or global scale, coarse spatial resolution data like AVHRR, MODIS, and SPOT Vegetation are preferable. Another major factor influencing the selection of sensor data is the climatic condition. The hazy natural circumstances in the humid tropical areas are usually an obstacle for obtaining high-quality optical sensor data. Therefore, different kinds of radar data serve as an essential supplementary data source. Since many sources of sensor data are now quickly available, image analysts

have more alternatives to select suitable RS for a particular type of study. A blend of multi-sensor data with various image characteristics is often beneficial to the research [7]. In this situation, economic condition is often a crucial factor that affects the selection of RS and the time and effort that can be devoted to the classification procedure, hence affecting the quality of the classification results.

2.5.3 Data pre-processing

Image preprocessing may introduce the discovery and restoration of split lines, geometric rectification or image registration, radiometric calibration and atmospheric change, and topographic improvement. If various ancillary data used, data exchange with several sources or formats, and quality evaluation of these data are also required before they can consolidate into a classification method. Precise geometric rectification or image registration of remotely sensed data is a requirement for a mixture of diverse source data in a classification method. Many textbooks and articles have reported this topic in detail [4] [12]. If a single-date image for classification, an atmospheric restoration may not be needed [13]. When multi-temporal or multisensor data utilized, atmospheric calibration is necessary; It is particularly reliable when multisensor data, like Landsat TM and SPOT or Landsat TM and radar data, are integrated for image classification. A kind of methods, varying from mild relative calibration and dark-object subtraction to calibration strategies based on complicated models, have been exploited for radiometric and atmospheric normalization and correction [14] [15] [16]. The topographic improvement would be another essential aspect if the study area determined in rugged or mountainous regions [17].

2.5.4 Feature Extraction and Selection

Choosing proper variables is an essential step for successfully performing an image classification method. Several latent variables utilized for image classification, such as spectral signatures, vegetation records, reconstructed images, textural or contextual learning, multi-temporal images, and ancillary data. Due to different directions in land-cover separability, utilizing many variables in a classification system may decrease classification accuracy [18] [19]. It is essential to choose only the variables that are most effective for separating land-cover or vegetation classes, mainly when

employed hyperspectral or multi-source data. Many approaches, such as principal component analysis, minimum noise fraction transform, discriminant analysis, decision boundary feature extraction, non-parametric weighted feature extraction, wavelet transform, and spectral mixture analysis [20] [21] [22]employed for feature extraction, in order to reduce the data redundancy inherent in remotely sensed data or to extract specific land cover information. The optimal choice of spectral bands for classifications discussed in [22] [4]. Visual analysis (e.g., bar graph spectral plots, co-spectral mean vector plots, two-dimensional feature space plot, and ellipse plots) and statistical approaches (e.g., average divergence, transformed divergence, Bhattacharyya distance, Jeffreys– Matusita distance) have utilized to recognize an optimal subset of bands [4]. Fuzzy-Logic expert system for feature selection described [23]. Three strategies for optimizing the choice of multi-source data, and found that these approaches (exhaustive search by recursion, isolated independent search, and sequential dependent search) applied to a variety of data analyses examined in [24]. In practice, a comparison of different combinations of chosen variables implemented frequently, and a useful reference dataset is essential. In particular, a good representative dataset for each class is critical for implementing a supervised classification. The divergence-related algorithms frequently utilized to evaluate the class separability and then to filter the training samples for each class.

2.5.5 Selection of a Precise Classification Approach

Several factors, such as the spatial resolution of the RS, various sources of data, a classification approach, and the availability of classification software, must be taken into account when choosing a classification approach for use. Several classification techniques have their advantages. The query of which classification approach is proper for a specific study is not easy to answer. Distinctive classification results may be achieved depending on the classifier(s) chosen. Comprehensive summarizing of primary classification methods described in 3.2

2.5.6 Post-Classification Processing

Initial per-pixel classification approaches may lead to ‘salt and pepper’ impacts in classification maps. A primary filter frequently utilized to overcome those noises

— most of the image classification based on remotely spectral response senses. By cause of the complexity of biophysical surroundings, spectral uncertainty is prevalent among land-cover classes. Therefore, additional data often utilized to change the classification image based on confirmed expert rules. E.g., forest distribution in mountainous areas is similar to elevation, slope, and features. Data describing territory properties used to modify classification results based on the information of specific vegetation categories and topographic factors. In residential zones, housing or population density associated with residential land-use distribution patterns, and such data can utilize to correct some classification challenges happen among commercial and high-intensity rural areas or between recreational grass and crops. Even though commercial and high-intensity urban areas have similar spectral signatures, their population densities are considerably distinctive. Furthermore, recreational grass often appears in urban areas, but grassland and crops are located mainly far from residential areas, with sparse houses and a less population density. Consequently, expert insight can be caused based on the associations between housing or population masses and residential land-use classes to support separate recreational grass from grassland and crops. Prior research has indicated that post-classification processing is a necessary step in enhancing the quality of classifier performance.

2.5.7 Classifier Performance Evaluation

Evaluation of a trained classifier is a crucial process in the classification system. Several procedures may be applied, ranging from a qualitative evaluation based on expert insight to a quantitative accuracy evaluation based on sampling approaches. To evaluate the performance of classifiers, [25] introduced six principles: robustness, allow to fully use the information content of the data, accuracy, reproducibility, equal applicability, and objectiveness. Indeed, no classification algorithm can satisfy all these requirements nor apply to all studies, due to various environmental settings and datasets applied. [26] recommends the use of multiple criteria to evaluate the appropriateness of algorithms. These rules incorporate with computational resources, classification accuracy, the stability of the algorithm, and robustness to noise in the training data. Classification accuracy evaluation is, yet, the most common approach for estimation of classification performance, which is presents in the following subsection-2.5.8.

2.5.8 Classification Accuracy Estimation

Before performing a classification accuracy estimation, one necessitates knowing the causes of errors [27] [28]. Further to errors from the classification itself, additional sources of errors, such as position errors resulting from the registration, interpretation errors, and lower quality of training or test samples, all these errors are influencing classification accuracy. In the process of accuracy evaluation, considered that the difference between an image classification result and the reference data is the cause of classification error. Nevertheless, to provide a reliable report on classification accuracy, non-image classification errors should also be examined, particularly when reference data not collected from a field study. A classification accuracy evaluation usually includes three necessary ingredients: responsive design, sampling design, and estimation, and research procedures [29]. The choice of a proper sampling strategy [30] is a significant move. The main components of a sampling strategy include sampling unit (pixels or polygons), sampling design, and sample size [31]. Potential sampling designs include random, stratified random, systematic, double, and cluster sampling. Detailed description of sampling techniques discovered in previous literature [32] [33]. The confusion matrix strategy is one of the most widely used in accuracy estimations [34]. To accurately produce a confusion matrix, one of the requirements consider the following factors: (1) source data collection, (2) classification design, (3) sampling system, (4) spatial autocorrelation, and (5) sample size and sample unit. The subsequent generation of a confusion matrix, other essential accuracy estimation components, such as overall accuracy, omission error, commission error, and kappa coefficient, can be derived. According to earlier research has described the purposes and presented computation techniques for these elements [30] [33] [34] [29] [35] [36] [37] [38] [39] [40] [41]. They have estimated the status of accuracy estimation of image classification and discussed concerning issues. [33] systematically examined the theory of fundamental accuracy estimation and some high-level issues involved in fuzzy-logic and multilayer estimations, and described principles and practical concerns in designing and managing accuracy estimation of remote-sensing data. The Kappa coefficient is a measure of overall statistical agreement of a confusion matrix, which takes non-diagonal components into account. Kappa analysis recognized as a powerful method for analyzing a single confusion matrix and for examining the differences between various confusion matrices [30]

[40] [41]. Revised kappa coefficient and tau coefficient have developed as improved measures of classification accuracy [42] [43]. Furthermore, accuracy estimation based on a normalized confusion matrix has been conducted, which regarded as a more reliable performance than the conventional confusion matrix [30] [44] [45]. The confusion matrix strategy is only suitable for 'hard' classification, considering that the map classes are jointly independent and exhaustive and that each position belongs to a single class. This hypothesis is frequently disrupted, particularly for classifications with coarse spatial resolution imagery. 'Soft' classifications have implemented to depreciate the mixed-pixel problem using fuzzy logic. The usual confusion matrix strategy is not suitable for evaluating soft classification outcomes. Therefore, many dissimilar measures developed, such as conditional entropy and mutual information [46] [15], fuzzy-set approaches [47] [48] [49] and parametric generalization of Morisita's index [50]. Nevertheless, one significant problem in assessing fuzzy classifications is the burden of collecting reference data. Further research is thus needed to find a suitable method for evaluating fuzzy classification issues. In summary, the confusion matrix approach is the most traditional accuracy estimation strategy for individual classes. Uncertainty and confidence study of classification outcomes has obtained significant attention recently [51] [52], and spatially precise data on mapping confidence observed as an essential perspective in effectively using classification results for decision-making [51] [52].

2.6 Materials

2.6.1 Transposed Convolution Layer

Transposed Convolutional(TC) layer, some references use the name deconvolution, which is unsuitable because it is not a deconvolution. The deconvolution layer expands the resolution of the feature maps of previous layers of CNN to get the same resolution as the input image. An easy resizing of the maps is an alternative as we do for the resizing of an image. However, since a transparent upsampling accidentally lose details, a more reliable choice is to have a trainable upsampling convolutional layer, whose parameters are going to change during training.

The TC operation creates the related connectivity as the standard convolution, though in the reversed direction. We utilize it for up-sampling. Furthermore, the

weights in the TC are learnable. So we do not need a predefined interpolation method. Even though TC does not mean to take a convolution matrix and apply transpose operation. The central intent is that the connection between the input and the output manipulated in a backward way compared with a regular convolution matrix. To up-sample the input by adding zeros within the input matrix in a way that the direct convolution allows the same outcome as the TC. TC performs the upsample operation and represents the coarse input data to fill in the details during upsampling, and it the combination of the UpSampling2D and Conv2D layers into a single layer. However, it is efficient for the dense pixel classification due to adding zeros to up-sample the input image/matrix before the convolution. Therefore, for Pixel-wise image classification, a deconvolutional layer is placed in the initial layers of regular CNN. The down-sampled maps from CNN are upsampled by the deconvolution layer, producing the feature to predict class labels of the individual pixel at their location. These predictions compared with the ground truth labels available, and a loss function is determined, which leads the network towards accurate prediction by updating the parameters associated with backward propagation. To obtain things may critical, deconvolutions do exist, but those are not standard in the field of DL. A transposed convolution is slightly related because it produces the same spatial resolution a possible deconvolutional layer would. However, the actual mathematical operation that made on the values is distinctive. A TC convolutional layer carries out a regular convolution but reverts its spatial transformation.

A network to learn how to up-sample optimally, TC is the best choice. It does not employ a predefined interpolation technique. TC contains learnable parameters.

In this thesis, initially, we utilize regular convolutions to reduce the input data into an abstract spatial representation, and then use transposed convolutions to expand the abstract representation into something of use. TC performs the upsample operation and represents the coarse input data to fill in the details during upsampling, and it the combination of the UpSampling2D and Conv2D layers into a single layer.

For example, for super-resolution, the aim is to upscale the input image to higher resolutions. Therefore TC can be used as an option to traditional approaches like bicubic interpolation.

The TC method is similar to the gradient estimate for a traditional convolution, i.e., the backward pass of a regular convolution and vice versa.

Unquestionably TC's are more adaptable than classical upsampling methods such as bicubic interpolation or nearest-neighbor interpolation. There are a few limitations, such as there can be checkerboard artifacts in the output. Transposed Convolutions are not able to perform without learning the optimal kernel weights, compare with classical upsampling methods.

Consider a single value in the input, and distribute it to a neighborhood of points at the output. The value of the filter defines precisely, and for each output pixel, it multiplies the input pixel value by the corresponding weight of the filter, then repeats this manner for every pixel value in the input data, and aggregate values in each output cell. Since, which is essential to understand, that the kernel values are defining a significant amount about the input value, then continuing to distribute over every output cells in the neighborhood.

Let us examine the first case that both input and output channels are 1, with 0 padding and 1 stride. Figure: 2.10 TC1 explains how transposed convolution with a 22 kernel computed on the 22 input matrix.

We can perform TC operation by given a matrix C(Filter or Kernel) and input matrix X. Convolution computes results by the Equation: 2.18

$$Y[i,j] = (X[i:i+h, j:j+w] * K).sum() \tag{2.18}$$

Where i and j are the pixel coordinates, and h and w are the dimensions of the input matrix or image. Convolution operation compiles the input values through the kernel while the TC distributes input values through the kernel, which produces the shape of output to larger than input shape.

The multi-channel expansion of the TC is identical to the convolution. When there are multiple input channels, denoted by ci, the TC assigns a khkw kernel matrix to the individual input channel. If the output has a channel size co, then we have a cikhkw kernel for each output channel.

As a result, if we feed X into a convolutional layer function f to compute Y=f(X) and create a transposed convolution layer g with the same hyper-parameters as f

except for the output channel set to be the channel size of X, then g(Y) should have the same shape as X. Let us verify this statement.

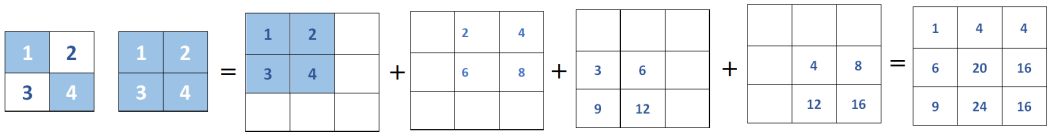


Figure 2.10: Transposed convolution layer with a 2×2 kernel.

2.6.2 Dilation Layer

Dilated convolutions help to produce a greater receptive field without negotiating on the resolution. Hence, the fusion of dilation with other standard convolution was considered significant [53]. In this work, We perform dilated convolution operation on the last layer or before fully connected layer of the network with default dilation factor, i.e., '1' to capture context features and the main idea behind to preserve spatial resolution in convolutional networks for image classification. Dilated convolutions layers help while producing more visually appealing outputs. The CNN-based approaches usually produce boundary artifacts, distorted structures, and blurry textures incompatible with surrounding areas. Because CNNs are not capable of modeling the long-term correlations between the several depth regions and contextual information. The proposed architecture uses dilated convolutions to aggregate multi-scale contextual information without losing resolution. There are two main reasons for the utilization of the dilation layer. Firstly, The architecture based on the fact that dilated convolutions support the exponential expansion of the receptive field without loss of resolution [54], which can capture high-level interpretation with broader information. On the other hand, although apparently, features from last layers of a network have broad receptive fields on the input image, in practice, they are much shorter [55]. This problem alleviated by dilated convolution. Figure: 2.11 is an instance of dilated convolution. To obtain the size of the feature map after dilated convolution unchanged, in our work, we used the default padding rate and the dilation rate that is '1'. More details regarding dilated convolution explained in [54]. The reason behind placing the dilation convolution in the last layer of the architecture is high-level context, semantics and broader visual cues are

available. However, by the presence of convolution layers, instead, dilation layers manage the loss of information. Meanwhile, the captured feature maps with multi-scale contexts can be adjusted automatically due to their similar resolution. Due to the loss of spatial information by using only standard convolution may be harmful to classifying natural images and it leads to significantly reduce the performance of the classifier when transfer to other tasks those are involved spatially detailed image understanding. To address this problem, by the fusion of features extracted by the standard convolution and dilation convolution can apply all kinds of imagery data.

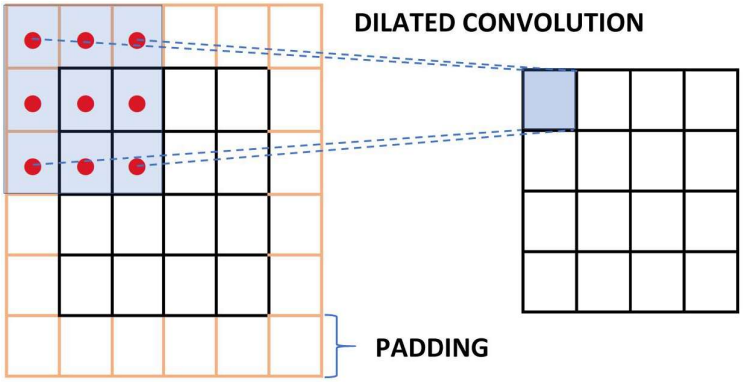


Figure 2.11: An illustration of dilated convolution used to capture context, where the size of feature map and convolution of a kernel is $5 \times 5 and 1 \times 1$, respectively, both the dilation rate and the padding rate equal 1.

2.6.3 Exponential Linear Unit(ELU)

In this work, the non-linear activation function used in the network is the Exponential Linear Unit (ELU) function with all corresponding transposed convolution layers. However, remaining all standard Convolution and Dilation Donvolution layers used ReLU non-linear activation function.

The most popular activation function conceivably is ReLU. It contains non-negative activation; therefore, the mean activation of ReLU is more significant than zero, which Causes a bias shift for a unit in the next layer. Various machine learning

methods work better with zero-centered, normalized features. Since ELUs contains negative values, it pushes the mean of the activations closer to zero. They have mean activations closer to zero; it leads to learning and converges the loss faster than other activation functions. ELU activation function prevents the problem of network performance degrading due to so-called Dying Gradients. A dead Rectified Linear Unit (ReLU) always outputs a zero value, for example, caused by a significant gradient update. For ReLUs, this is an irreversible process. Recovering a dead weight is very unlikely since weights are not able to update for a zero gradient. As the ReLU gradient at zero is zero, weights remain unchanged. Dead weights do not contribute to the learning process. [56] explains ELU guides not only to faster learning but also to significantly better generalization performance than ReLUs and LReLUs on networks with more than five layers. However, ELU shows significant performance because for dense pixel classification, assign ELU in initial layers may provide lager information to the following layers. Since appointing ReLU in initial layers may have a possibility to lose important information by negative values, it leads to reduce the robustness of the classifier. However, ELU becomes smooth slowly until its output equal to $-\alpha$, whereas RELU sharply smoothes. Notice that is equal to $+1$ in the following illustration.

The Math The math is not too hard. The equation describes below,

$$f_{ELU}(h_k) = \begin{cases} h_k & if(h_k > 0) \\ \alpha(e^{h_k} - 1) & if(h_k < 0) \end{cases} \tag{2.19}$$

If we have input x-value, which is higher than zero, then it is similar to the ReLU – the outcome going to be a y-value corresponding to the x-value. However, this time, if the input value x is less than 0, we have a value slightly under 0.

We have the y-value, which depends both on the value of input x and the parameter α. We can modify it when we need it. Moreover, we introduce an exponential operation ^{x}, which means the ELU is more computationally expensive than the ReLU.

The unique information we need to know is that the derivative of the exponent function is the exponent itself.

However, ELU becomes smooth slowly until its output equal to $-\alpha$, whereas ReLU sharply smoothes. Notice that α is equal to $+1$ in the following illustration.

$$\frac{\partial f_{ELU}(h_k)}{\partial \boldsymbol{w}_k} = \frac{\partial f_{ELU}(h_k)}{\partial h_k} \frac{\partial h_k}{\partial \boldsymbol{w}_k}. \tag{2.20}$$

(2.21)

The derivative of the activation function served to backpropagation algorithm during learning. That is why both the service and its derivative should have low computation cost.

$$\frac{\partial f_{ELU}(h_k)}{\partial h_k} = \begin{cases} 1 & (h_k > 0) \\ f_{ELU}(h_k) + \alpha & (h_k \leq 0) \end{cases} \tag{2.22}$$

It seems simple enough. The value of$y$ output is 1 if x is larger than 0. The output is the ELU function (not differentiated) plus the α value if the input $x < 0$. The plot for it looks like Figure: 2.12

The unique information we need to know is that the derivative of the exponent function is the exponent itself.

$F_{ELU}(h_k)$ is the forward pass, and its derivative is $F`(x)$ for calculating its backward gradients.

The ELU hyperparameter examines the value to which an ELU saturates for negative net inputs (see Figure: 2.13. ELU reduces the vanishing gradient influence as rectified linear units (ReLUs), and leaky ReLUs (LReLUs) do. The vanishing gradient problem is alleviated because the positive part of these functions is the identity; therefore, their derivative is one and not contractive. In contrast, Tanh and Sigmoid activation functions are contractive almost everywhere [56]

If we see carefully here, we bypass the dead relu problem while still having some of the computational speed gained by the ReLU activation function; that is, we still have some dead components in the network.

However, still, ELU has some limitations, such as it carries longer computation time, because of the exponential operation included, it does not avoid the exploding gradient problem, and The neural network does not learn the alpha value.

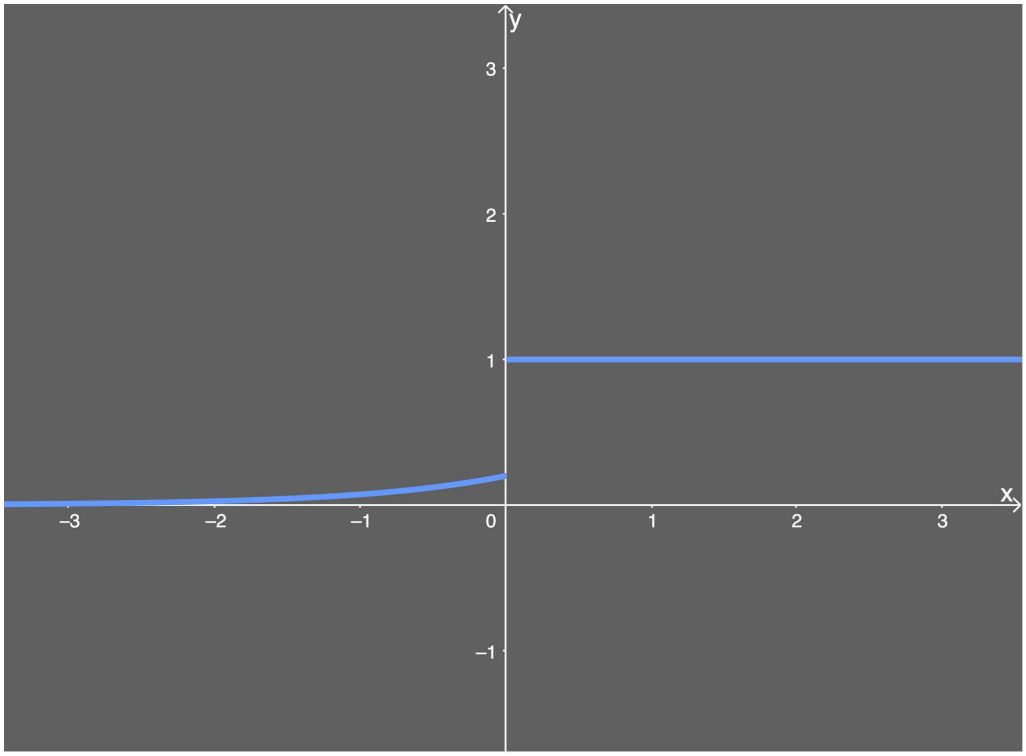


Figure 2.12: The differentiated ELU activation function

2.6.4 Dimensionality Reduction

The decrease in the hyperspectral representation space can carry utilizing feature selection or extraction techniques. In both approaches, the aim is to decrease the number of spectral bands, without loss of information. The process of feature selection is to pick a representative subset of features from the original data by assessing its discrimination capabilities according to statistical distance measures among classes (e.g., Bhattacharyya distance, Jeffries-Matusita distance, and the transformed divergence measure). The feature extraction method addresses the problem of dimensionality reduction by projecting the data from the original feature space into a low-dimensional subspace, which contains most of the original information. The most widely-known feature extraction method corresponds to Principal Component

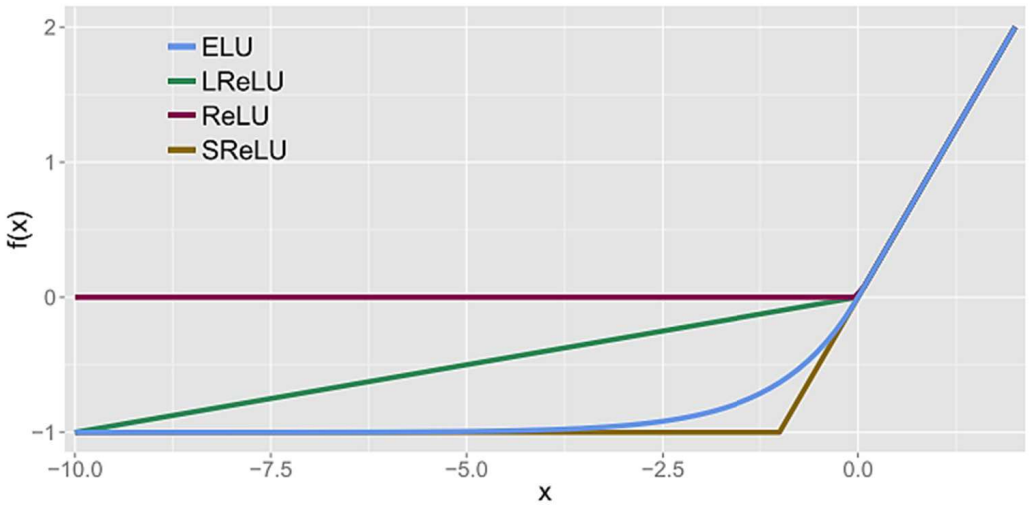


Figure 2.13: The rectified linear unit (ReLU),the leaky ReLU (LReLU, $\alpha = 0.1$), the shifted ReLUs (SReLUs), and the exponen tial linear unit (ELU, $\alpha = 1.0$).

Analysis (PCA), which seeks to reduce the dimension of the data by finding a few orthogonal linear combinations of the original variables with the most significant variance. It includes a mathematical procedure that transforms several (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components. Principal component analysis (PCA) is a procedure used to maintain variation and bring out healthy patterns in a dataset. It is frequently used to make data easy to examine and visualize.

For example, consider having data with two dimensions, like (height, weight). This dataset can be plot as points in a plane. However, if we require to bother out variation, PCA finds a new coordinate system in which every point has a new (x, y) value. The axes do not mean anything physical; they are combinations of height

and weight called "principal components"; those are choosing to give one axis with lots of variation.

Moderately unsurprisingly, lessening the dimension of the feature space is called "dimensionality reduction". There are many ways to achieve dimensionality reduction, but most of these techniques fall into one of two classes, which are Feature Elimination and Feature Extraction

In this thesis, data to feed the network, we first use Principal Components analysis to reduce the high dimensionality of the original image (i.e., hundreds of spectral bands). PCA is a mathematical procedure to move N-dimensional data in another N-dimensional space where the dimensions are linearly uncorrelated [57]. Additionally, it allows selecting the M most representative dimensions. Consequently, a HSI pixel $x_{i,j}$ with N spectral bands, can be reduced to a $\overline{x}_{i,j}$ vector in M-dimensional space, with $M < N$. Then, for each pixel in the reduced space, we extract a $P \times P$ patch to use for its classification. Hence, the pixel-wise dataset for HSI pixels classification includes a collection of patches in the new M-dimensional space.

2.6.5 Resampling or Oversampling

When performing classification algorithms, the structure of our data is of great importance. Precisely, the balance between the number of observations for each possible output heavily influences our prediction's performance. Let us see the illustration Figure: 2.15 of that with a plot from imbalanced-learn.

As we can see, we are dealing with data that has three different classes that we are trying to predict with the help of a linear support vector machine algorithm. In the top-left plot, the yellow class is overlooking the other classes, and the decision boundaries for the other classes are barely recognizable. As we keep adding more observations to the underrepresented classes, the decision boundaries change dramatically. Thus, when trying to make predictions with regards to a minority class, we need to find ways to avoid being misled by the majority class.
However, a popular way to undertake the issue of imbalanced data is over-sampling/resampling. Over-sampling refers to numerous methods that aim to increase the number of samples from the rare class in the data set. In our case, these techniques increase the number of fraudulent transactions in our data (usually to 50:50). We might ask why

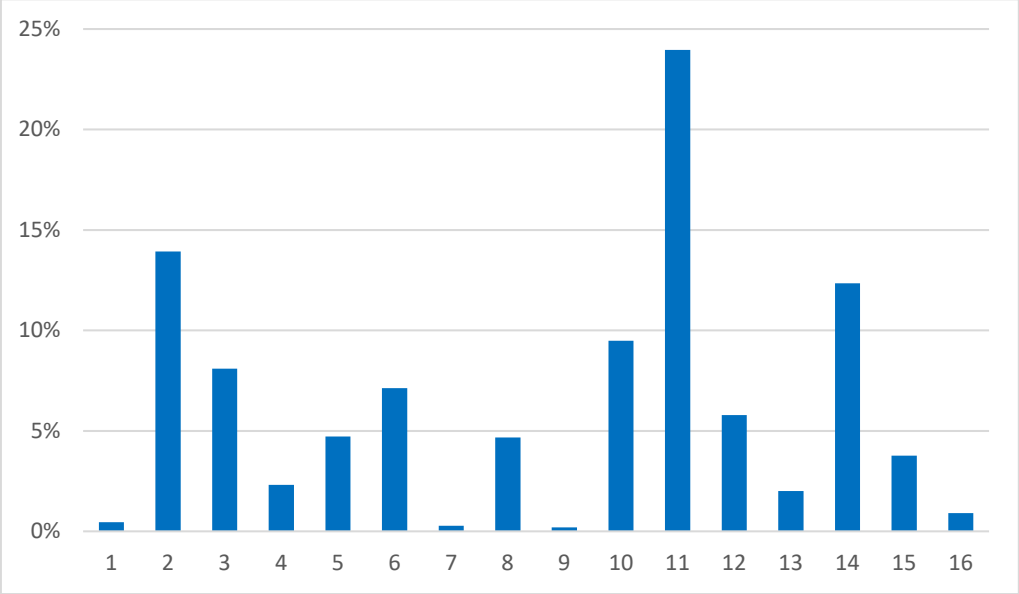


Figure 2.14: Imbalanced Data Distribution

one would even do that in the first place. Good question. If we do not balance the number of samples, most classification algorithms massively focus on the majority class. As a result, it might seem like our algorithm is achieving superb results when, in reality, it is merely always predicting the majority class. One popular way to undertake the issue of imbalanced data is over-sampling/resampling. Over-sampling refers to numerous methods that aim to increase the number of samples from the rare class in the data set. In our case, our data (usually to 50:50). We might ask why one would even do that in the first place. Good question. If we do not balance the number of samples, most classification algorithms massively focus on the majority class. As a result, it might seem like our algorithm is achieving superb results when, in reality, it is merely always predicting the majority class.

A better choice to using random naive over-sampling is Synthetic Minority Over-Sampling Technique(SMOTE). While SMOTE still oversamples the minority class, it does not rely on reusing previously existing observations. Instead, SMOTE creates new (synthetic) observations based on the observations in your data. How does SMOTE do that? To illustrate our point, we have put together a fictional data set.

As we can see in the Figure: 2.16, there are way more triangles than squares in

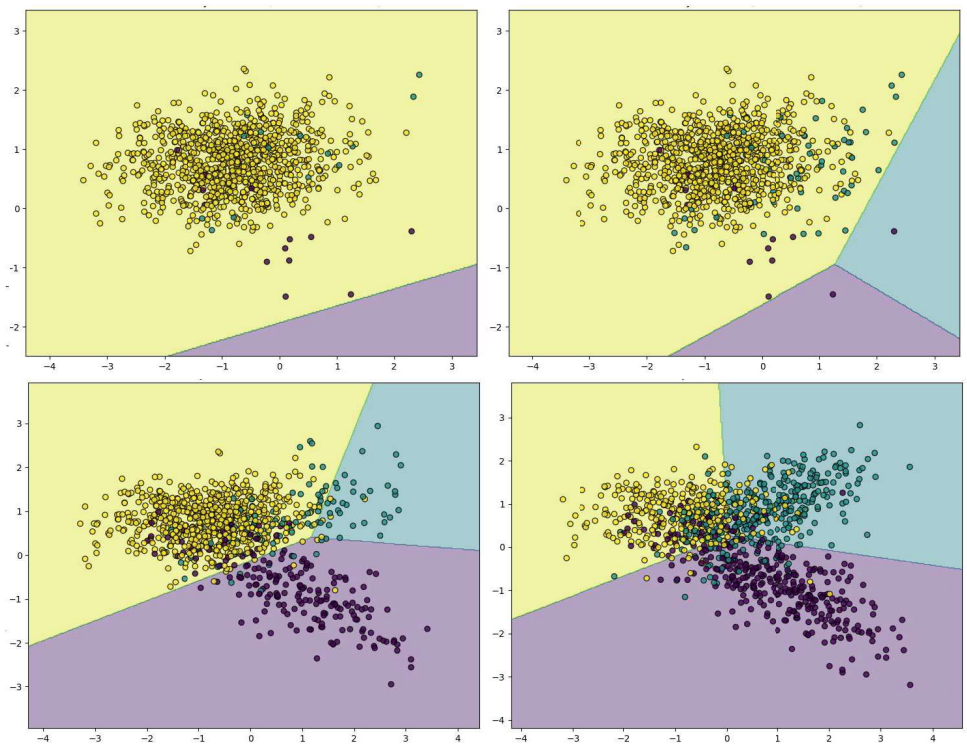


Figure 2.15: Imbalanced Data Distribution

this fictional data set. Now, in order to train a more accurate classifier, we would like to utilize SMOTE to oversample the squares. First, SMOTE finds the k-nearest-neighbors of each member of the minority class. Let us visualize that for one of the squares and assume that $k = 3$.

The visualization of Figure: 2.17 we have identified the three nearest neighbors of the orange square. Now, depending on how much oversampling desired, one or more of these nearest neighbors are going to be used to create new observations. For this explanation, let us assume that we are going to use two of the three nearest neighbors to create new observations. The next and final step is to create new observations by randomly choosing a point on the line connecting the observation with its nearest neighbor.

The dashed lines in the Figure: 2.18 represent the connection between the orange

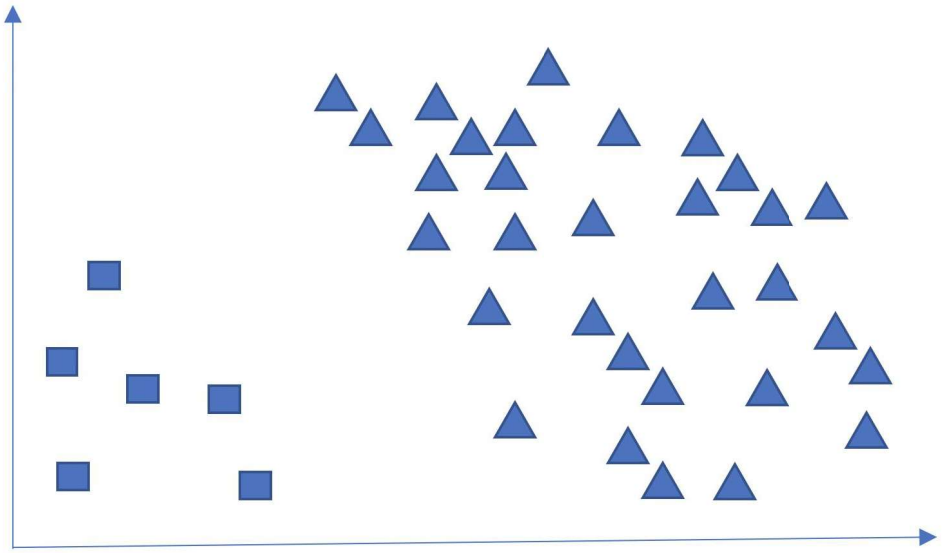


Figure 2.16: Visualization Imbalanced Data Distribution 1

square and its green nearest neighbors. The two red squares denote the new observations added to the data set by SMOTE. The main advantage of SMOTE compared to traditional random naive over-sampling is that by creating synthetic observations instead of reusing existing observations, our classifier is less likely to overfit. At the same time, we should always make sure that the observations created by SMOTE are realistic. The technique SMOTE uses to create new observations only helps if the synthetic observations are realistic and could have observed in reality. However, samples generated by SMOTE make the network more complex.

2.6.6 K-Fold Cross Validation and Data Partition

A cross-validation algorithm is a fundamental tool in the field of Data Science. It enables us to maintain our data properly. Let us go ahead about what cross-validation is.

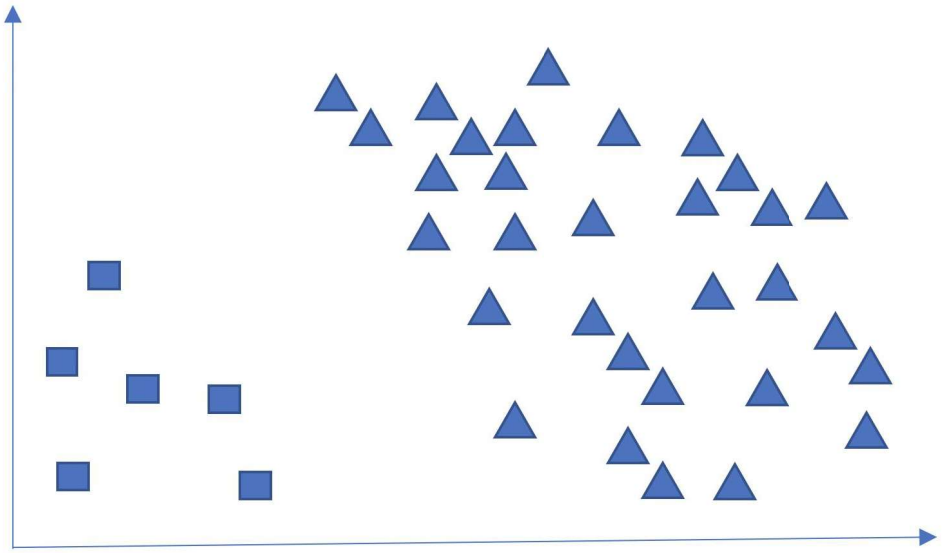


Figure 2.17: Visualization Imbalanced Data Distribution 2

In Machine learning, we frequently distribute the dataset into Training, Validation, and Test datasets. Cross-validation is a kind of resampling technique for train and validates machine learning models. Dividing the dataset with a distinct ratio in terms of the number of data points or examples to train and validate machine learning models is a primary feature of cross-validation.

K-fold cross-validation is a simple system to generate a K number of folds randomly. Obtaining the optimal value of K is a significant task. Optimize the value of K, many kinds of research conveyed using different datasets on parametric and non-parametric models [58]. In [59] explored leave $'p - out'$ cross-validation for density estimation by the optimization of cross-validation split-ratio; in this thesis, we focused on finding the value of $'p'$ the number of data points to be used get train and validation sets. More recently, the same author [59] attained optimal split-ratio for density estimation by the implementation of a $L2 - loss$ measure. [60] examines the predominance and importance of partition-induced covariate shift on different k-fold cross-validation designs, and the degree of partition-induced covariate shift

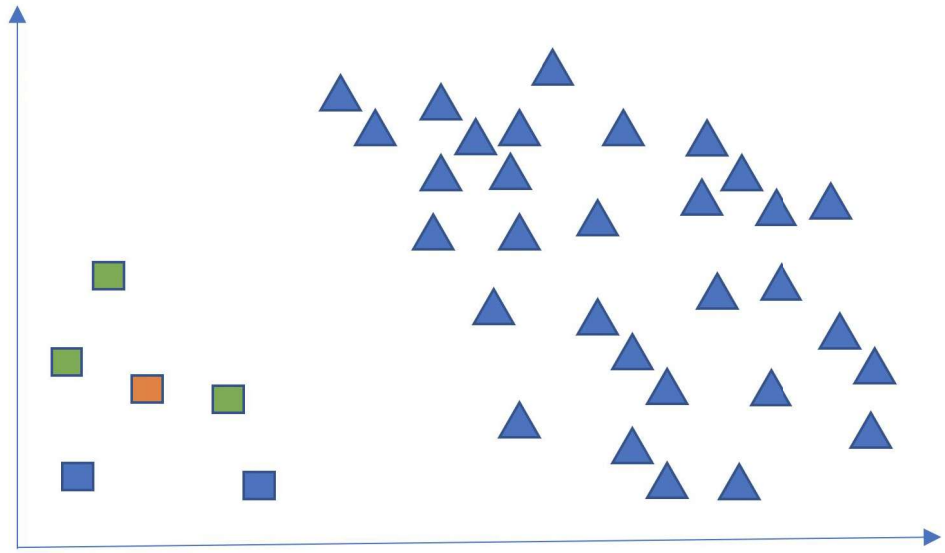


Figure 2.18: Visualization Imbalanced Data Distribution 3

depends on the cross-validation idea considered. In this way, worse schemes may wreck the precision of a single-classifier performance estimation and also increase the required number of repetitions of cross-validation to reach a stable performance estimation.

Training data set — used to train the model, it can vary, yet typically we use 5% to 90% of the potential data for training.

Validation data set — Once we choose the model that performs well on training data, we run the model on the validation data set. Which is a subset of the data usually ranges from $10\% to 30\%, 40\%$, and so on it depends on the training set. The validation data set helps implement an unbiased evaluation of the model fitness. If the error on the validation dataset increases, then the model might suffer from overfitting.

Test dataset — Also called a holdout data set. This dataset includes data that has never used during model training. The test data set serves with the final evaluation of the model. Typically the required size of the test set would depend on the training and validation sets.

When we are going to build a machine learning model using some data, we often split our data into training, validation, and test sets. The training set used to train the model, and the validation set used to validate it on data use during training to converge the loss of model using the back-propagation algorithm. Finally, the test set used to evaluate the robustness and performance of the model on data it has never seen before. The classic approach is to do a simple $60\% - 20\%$ split, sometimes with different values like $50\% - 30\%$ or $80\% - 10\%$ for train and validation set, remaining data used to test the model. In cross-validation, we do more than one split. We can do 3, 4, 6, or K number of splits. Those separations are called Folds, and there are several approaches we can build these folds. For instance,

K fold cross validation The method, Figure: 2.19, involves randomly splitting the dataset into K groups or folds of approximately equal size. The first fold kept for testing, and the model trained on $K - 1$ folds.

The process repeated K times, and each time different fold or a different group of data points used for validation.

For instance, a simple K-Folds — We split complete data into K folds; consider $K = 5$, for example. If we have 5000 examples in our dataset, We split it into 5 folds, fold 1, fold 2, fold 3, fold 4, and fold 5. Then we build five different models; each model is trained on 4 folds and tested on the $5th$ fold. Our first model is trained on part $1, 2, 3$, and 4, and tested on fold 5. Our second model is trained to on fold $1, 2, 3$, and fold 5 and tested on fold 4 and so on.

Leave One Out Cross-Validation — This is the high-level method to do cross-validation. For each data point in our dataset, we build a model using all other data points and then test it on the selected instance. **Stratified Cross-Validation** — Split the complete data into folds, make sure that each fold is a valid representative of the complete data. The necessary fundamental thing is that we need the equivalent proportion of several classes in each fold. Most of the time,

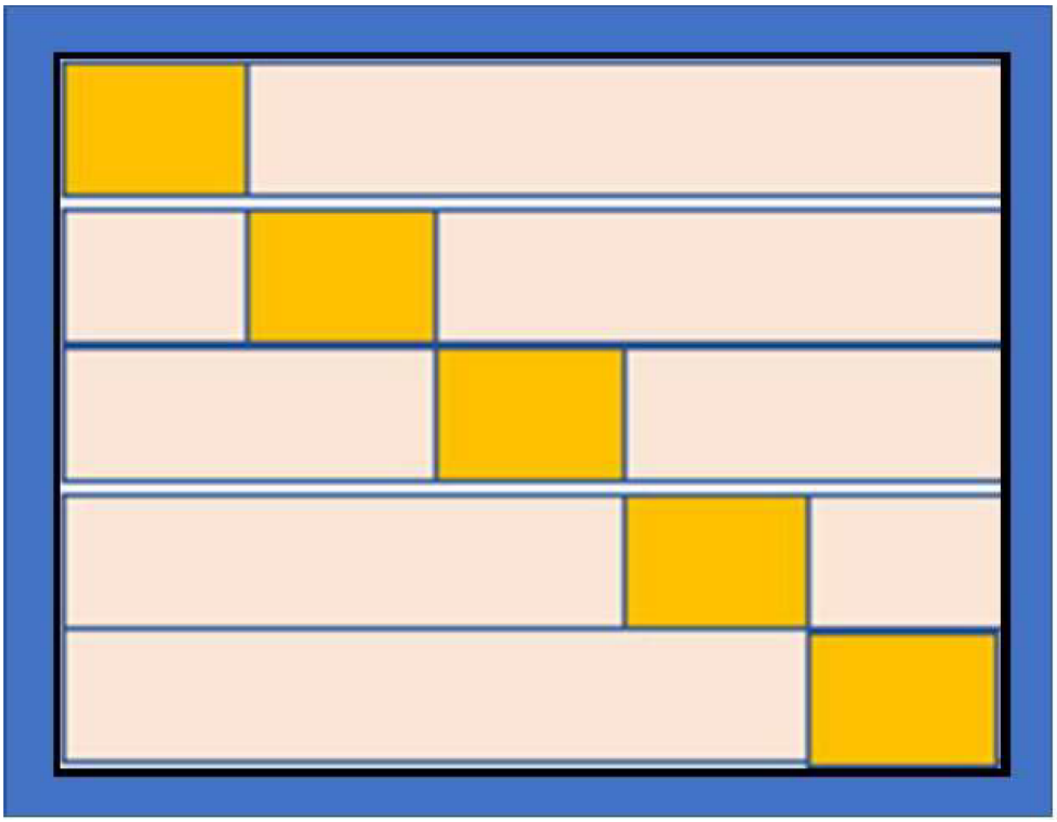


Figure 2.19: Visualization Imbalanced Data Distribution 1

it occurs by arranging them randomly, but sometimes, in complicated datasets, we have to expect a proper distribution for each fold.

let us look at some practical reasons for the implementation of cross-validation.

Employing Complete Data When we have a small amount of data, splitting it into training and test set might leave us with a small test set. Suppose we have only 100 examples if we do a simple 80–20 split, we will get 20 samples in our test set. It is not enough. We can get almost any performance on this set only due to chance. The problem is even worse when we have a multi-class problem. If we have 10 classes and only 20 examples, It leaves us with only 2 examples for each class on average. Testing anything on only 2 examples cannot lead to any real conclusion. If we use cross-validation in this case, we build K different models so that we can make predictions on all of our data. For each instance, we predict by a model that did not see this example, and so we are getting 100 examples in our test set. For the

multi-class problem, we get 10 examples for each class on average, and it is much better than just 2. After we evaluated our learning algorithm, we are now can train our model on all our data because if our 5 models had similar performance using different train sets, we assume that by training it on all, the data may get similar performance. By doing cross-validation, we can use all our 100 examples both for training and for testing while evaluating our learning algorithm on samples we have it has never seen before.

Getting More Matrices As discussed in Employing Complete Data, when we create five distinct models utilizing our learning algorithm and test it on diverse test sets, we can make more convinced with the performance of our algorithm. If we begin with a single evaluation on the test set, we get simply one result. We may get this result because of an opportunity or a biased test set for some reason. By training 5 or more different models, we should get sufficient. Suppose we train on five models, and we consider accuracy as our measurement. We could conclude in various states. The best aspect is that our accuracy is comparable in all our folds, say $91.0, 90.5, 91.8, 92.2$ and 91.1, which suggests that our algorithm and our data is compatible, and we can be sure that by training the model on all the data set we have and extend it in result lead to similar performance. However, we could complete it in a slightly modified scenario, say $91.0, 54.0, 92.5, 90.5$, and 86.8. These results look strange. It seems like one of 5 folds is from a separate distribution; we have to move back and make sure that given data is something we expect it is. The critical outline we can settle up in is when we have a significant difference in our results, say $70, 54, 98, 500$, and 97. In this case, it seems like that our algorithm or our data is neither compatible, it could be that the algorithm is inefficient to learn, or the data is very confused or complicated. We need to get more metrics and brings a necessary outcome of both algorithm and r data via Cross-Validation

Use Models Stacking We need to build a proper pipeline for the models to determine what we expect. Consider, Neural Networks, for instance. It is possible to build many layers. Every layer having an output of the previous layer, and the network able to learn a unique representation from the given data, so ultimately, it yields reliable predictions. Through the back-propagation algorithm, the training

of several layers is possible. Every layer estimates its error and transfers it back to the preceding layer. When implementing similar without Neural Networks, we are not able to train the network in the same way, yet always not possible to get clear derivative or error, which we give back. For instance, consider the Random Forest algorithm, which predicts something, and then we require to do a Linear Regression that will commit to earlier predictions and generate some real number. The significant task here, the following model, need to learn on the predictions of the previous model. The most reliable solution is to use two separate datasets for each model. We can train the Random Forest on the first dataset. Then, we use the second dataset to produce a prediction using it. Then using second dataset predictions to train the second model, which is the logistic regression, and finally, we use the third dataset to evaluate the complete solution. We make predictions using the primary model, pass those predictions to the second model, and then compare that to the provided ground truth, which is the third dataset. If we have limited data, we cannot prepare it. Additionally, we may not train both models on the equivalent dataset because the second model learns upon predictions from the first model already seen. It may lead to over-fit, or it may have reliable results than on another dataset. It suggests that the second model is trained, but not on the test set. It may manage to another effect on final evaluation, and it is hard to understand. By the implementation of the cross-validation technique, it is possible to make predictions on the given dataset in the same way as described before, and so our second's models input will be real predictions on data that our first model never seen before.

Group The Data When we implement a random train-test split of our data, we believe that our samples are independent. That means that by identifying some samples may not help us recognize other samples. However, that is not a regular situation. Suppose a speech recognition scheme. Given that data may combine different speakers, saying several words. Let us look at verbal digits recognition. In this dataset, for instance, there are three speakers and 900 recordings (300 for each speaker). If we do a random split, our training and test set will share the same speaker saying the same words! Of course, it is an improvement within our algorithm performance, but if we test the model on a new speaker, our results will be much worse. The conventional way to do it is to divide the speakers, i.e., use

two speakers for training and use the third for testing, then we will test the trained algorithm only on the third speaker. It may not be enough, we want to know how our algorithm performs on different speakers. We can use cross-validation on the speaker's level. We will train 3 models, each time using one speaker for testing and remain two for training. In this way, we able to evaluate our algorithm better and finally build our model on all speakers.

Fine-Tune The Parameters Parameter fine-tune is one of the usual and definite reasons to do cross-validation. Most of the learning algorithms need to tune the parameters. In case, the number of trees in Gradient Boosting classifier, in a Neural Network hidden layer size or activation functions, kernel types in an SVM, and several more. We need to obtain the best parameters for our problem. We do it by trying different values and choosing the best ones. There are many techniques to do this. It could be a manual search, a grid search, or some more sophisticated optimization. However, in all those cases, we cannot do it on our training test and not on our test set, of course. We have to use a third set, a validation set. By splitting our data into three sets somewhat of two, we will tackle all the same issues we talked about before, particularly if we do not have much data. By doing cross-validation, we can do all those steps using a single set.

The advantage of utilizing K-fold cross-validation is.

Computation time reduced as we repeated the process only 5 times when the value of k is 5. Reduced bias Every data points get to be tested precisely once and employed in training k-1 times The variance of the resulting estimate reduced as k increases However, the drawback is, the training algorithm is computationally intensive as the algorithm has to be rerun from scratch k times

2.7 Metrics for Imbalanced data Classification

Classification task having multiple classes, including imbalanced dataset, shows a complex challenge than a binary classification problem. The skewed distribution makes many standard machine learning algorithms less effective, particularly in predicting minority class examples. To address this problem, let us first explain the problem at round and then consider the ways to overcome it.

Multiclass Classification: A classification task with more than two classes; e.g., classify a set of images of fruits which may be apples, banana, or peaches. Multiclass classification assumes that each sample assigned to a single label: a fruit can be either an apple or a peach but not both at the same time.

Imbalanced Dataset: Imbalanced data typically introduces to a problem with classification problems where the classes did not represent equally. For instance, we have a 3-class classification problem of a set of fruits to classify as bananas, apples, or peaches with a total of 100 samples. A total of 80 instances assigned label with Class-1 (Banana), 10 samples with Class-2 (Apples), and the remaining 10 samples labeled with Class-3 (Peaches). This kind of distribution called an imbalanced dataset and the ratio of $8 : 1 : 1$. Most of classification data sets do not have an exactly equal number of samples in each class, but a small difference usually does not matter. There are problems where the class imbalance is not just ordinary. For instance, in datasets like those that characterize fraudulent transactions are imbalanced. The vast majority of the transactions are in the "Not-Fraud" class, and tiny minority samples are in the "Fraud" class.

measure model performance: Let us think that we trained our model on imbalanced data of earlier example of fruits, and as data is heavily biased towards Class-1 (Bananas) because bananas are 80% of total samples, the model over-fits on the Class-1 label and predict it in most of the cases. We achieve an accuracy of 80%, which seems very good at first but looks closely, it may never be able to classify apples or peaches accurately. Now the question is if the Accuracy, in this case, is not the right metric to choose, then what metrics to use to measure the performance of the model?

2.7.1 Confusion Matrix

Confusion Matrix is a performance measurement for a classification algorithm where output can be two or more classes.

With imbalanced classes, it is easy to get a high accuracy without actually making useful predictions. So, Accuracy as an evaluation metrics performs sense only if the class labels uniformly distributed. In the case of imbalanced classes, confusion-matrix is an excellent technique for summarizing the performance of a classification algorithm.

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Figure 2.20: Parts of of Confusion Matrix

n = 75	PREDICTED NO	PREDICTED YES
ACTUAL NO	20	10
ACTUAL YES	5	40

Figure 2.21: Structure of Confusion Matrix

The confusion matrix is a table that often used to represent the performance of the classifier on a set of test data for which the actual known values. The confusion matrix itself is comparatively simple to understand, but the related terminology can be confusing.

Let us start with an example confusion matrix for a binary classifier to understand quickly.

What can we learn from the matrix-2.21 There are two possible predicted classes: "yes" and "no." If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease. The classifier made a total of 75 predictions (e.g., 75 patients tested for the presence of that disease). Out of those 75 cases, the classifier predicted "yes" 50 times, and "no" 25 times. In reality, 45 patients in the sample have the disease, and 30 patients do not. Let us now define the most basic terms, which are whole numbers (not rates):

true positives (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.

true negatives (TN): We predicted no, and they don't have the disease.
false positives (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
false negatives (FN): We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

We have added these terms to the confusion matrix, and also added the row and column totals:

2.7.2 Accuracy

Accuracy is one of the metrics concerning evaluating classification models. Informally, Accuracy is the fraction of predictions our model got right. Formally, Accuracy has the following definition:

$$Accuracy = \frac{NumberofCorrectPredictions}{TotalNumberofPredictions}$$

For binary classification, Accuracy can also calculate in terms of positives and negatives. Figure: 2.20 as follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Where
TP = True Positives,
TN = True Negatives,
FP = False Positives,
FN = False Negatives.

Overall Accuracy(OA) Overall Accuracy is telling us out of all of the samples what proportion outlined accurately. The overall Accuracy expressed as a percent, with 100% accuracy being perfect classification where all samples were classified correctly. Overall, Accuracy is not much challenging to calculate and understand

		ORIGINAL DATA			
		Water	**Forest**	**Urban**	**Total**
CLASSIFIED	**Water**	**21**	**6**	**0**	**27**
	Forest	**5**	**31**	**1**	**37**
	Urban	**7**	**2**	**22**	**31**
	Total	**33**	**39**	**23**	**95**

Figure 2.22: Representation of a Confusion Matrix of classifier evaluation with three categories

but eventually only provides the map user and producer with necessary accuracy information. The diagonal elements in the confusion matrix represent the areas that were correctly classified. To calculate the overall Accuracy, we add the number of correctly classified samples and divide it by the total number of samples. For instance consider the Figure: 2.22

Number of correctly classified site: $21 + 31 + 22 = 74$ Total number of reference sites $= 95$

Overall Accuracy $= 74/95 = 77.9\%$ We could also represent this as an error percentage, which would be the complement of accuracy: error + accuracy $= 100\%$. In the above example, the error would be the number of samples incorrectly classified divided by $95 or 21/95 = error, = 22.1\%$. We could also define the overall error by subtracting the accuracy percentage from$100 : 100 - 77.9 = 22.1\%$.

	cat	dogs
Cats	10	7
dogs	5	8

Figure 2.23: A Simple Structure of Confusion Matrix with two class classification results

2.7.3 kappa Coefficient

The Kappa statistic (or value) is a metric that compares an Observed Accuracy with an Expected Accuracy. The kappa statistic used not only to evaluate a single classifier but also to evaluate classifiers amongst themselves. Besides, it takes into account agreement with a random classifier, which generally means it is less misleading than only using accuracy as a metric, for instance, an Observed Accuracy of 80% is a lot less impressive with an Expected Accuracy of 75% versus an Expected Accuracy of 50%. Computation of Observed Accuracy and Expected Accuracy is integral to the comprehension of the kappa statistic and represented through the use of a confusion matrix. Before going to move further, Let us look at section: 2.7.1 and the following example Figure: 2.23 Consider that built a model using supervised machine learning on labeled data. It does not always have to be the case; the kappa statistic frequently used as a measure of reliability within two human rater/estimator/examiner/judges. Although, columns correspond to one "rater," while rows correspond to another "rater." In supervised machine learning, one "rater" returns ground truth (the actual values of every instance to be classified), acquired from labeled data, and the other "rater" is the machine learning classifier used to perform the classification. Ultimately it does not matter which is which to compute the kappa statistic, but for clarity's sake, let us assume that the columns reflect ground truth, and the rows reflect the machine learning classifier classifications.

Before we get to the equation for the kappa statistic, one more value is needed: the Expected Accuracy—this value defined by the accuracy any random classifier would expect to achieve based on the confusion matrix. The Expected Accuracy is

directly related to the number of instances of each class (Cats and Dogs), along with the number of instances that the machine learning classifier agreed with the ground truth label. To calculate Expected Accuracy for our confusion matrix, first, multiply the marginal frequency of Cats for one "rater" by the marginal frequency of Cats for the second "rater" and divide by the total number of instances. The marginal frequency for a certain class by a certain "rater" is just the sum of all instances the "rater" indicated were that class. In our case, $15(10 + 5 = 15)$ instances labeled as Cats according to ground truth, and $17(10+7 = 17)$ instances were classified as Cats by the machine learning classifier. This results in a value of $8.5(15 * 17/30 = 8.5)$. Then this is made for the second class as well (and can be repeated for each additional class if there are more than 2). $15(7 + 8 = 15)$ instances labeled as Dogs according to ground truth, and $13(8 + 5 = 13)$ instances classified as Dogs by the machine learning classifier. This results in a value of $6.5(15 * 13/30 = 6.5)$. The final step is to add all these values together, and finally divide again by the total number of instances, resulting in an Expected Accuracy of $0.5((8.5 + 6.5)/30 = 0.5)$. In our example, the Expected Accuracy turned out to be 50%, as will always be the case when either "rater" classifies each class with the same frequency in a binary classification (both Cats and Dogs contained 15 instances according to ground truth labels in our confusion matrix).

The kappa statistic can then calculated using both the Observed Accuracy (0.60) and the Expected Accuracy (0.50) and the formula:

$Kappa = (observedaccuracy - expectedaccuracy)/(1 - expectedaccuracy)$

So, in our case, the kappa statistic equals: $(0.60 - 0.50)/(1 - 0.50) = 0.20$.

2.7.4 F1-Score

F1 score is a classifier metric based on Precision and Recall, which calculates an average of Precision and Recall in a way that indicates the lowest value, however, why an F1-score is essential to measure the quality of the model. Before going to understand F1-score, let me introduce two new metrics called precision and Recall. What are these precision and Recall? lets consider the parts of confusion matrix from Figure: 2.20

If we look at Wikipedia, the formula for calculating Precision and Recall is as follows:

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Figure 2.24: Total Predicted Positives

Precision

Lets consider Figure: 2.24

$$Precision = \frac{TruePositive}{TruePositive+FalsePositive}$$

From the above equation, we can see that the precision expresses regarding how accurate our model is out of those predicted positive, how many of them are actual positive.

Precision is an exceptional measure to decide when the values of False Positive is high. For example, email spam detection. In email spam detection, a false positive means that a non-spam email (actual negative) has identified as spam (predicted spam). The email user might lose relevant emails if the precision is not high for the spam detection model.

Recall

Let us consider the same logic of Precision for Recall. How Recall has calculated. Lets consider Figure: 2.25.

$$Precision = \frac{TruePositive}{TruePositive+FalseNegative}$$

Here we see, the Recall calculates how many of the Actual Positives our model takes label it as Positive (True Positive). Applying the same belief, we know that

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Figure 2.25: Actual Positives

Recall shall be the model metric we use to select our best model when there is a high value incorporated with False Negative. For example, consider fraud detection or infected patient detection. If a fraud transaction (Actual Positive) predicted as non-fraud (Predicted Negative), the outcome could be not good for the bank. Similarly, in infected patient detection, if an infected patient (Actual Positive) went through the test and predicted as not infected (Predicted Negative), the value associated with False Negative will be much high if the infection is contagious.

F1-Score

When we consider the Precision and Recall from the literature, we cannot avoid the additional measure, F1-Score, which is a function of Precision and Recall.

$$Precision = 2 * \frac{Precision * Recall}{Precision + Recall}$$

F1-Score is required when we need to explore a balance between Precision and Recall. However, what is the main difference between F1-Score and Accuracy? We have previously noted that accuracy can primarily be contributed by a vast number of True Negatives, which in most business matters, we do not focus on much. In contrast, False Negative and False Positive usually has business costs (real hypothetical). Thus F1-Score might be a more reliable measure in practice if we need to investigate a balance between Precision and Recall, and there is an irregular class distribution (a massive number of Actual Negatives).

Chapter 3

Related Works

3.1 Traditional Approaches

With the advantage of wealth information provided by the hyperspectral imagery, various kind of applications include geographical research, wetland mapping, environmental mapping, and global change research, crop analysis and plant, mineral identification, and other remote sensing applications are adopted. Since the classification of each pixel in the hyperspectral image is one crucial requirement for all these applications. HS Images typically contain precious spectral evidence and high-resolution spatial structure of materials within hundreds of continuous observation bands all over the electromagnetic spectrum at each pixel, covering a wide range of wavelengths captured by satellite spectrometers [61] [62]. In the initial stage, to solve the classification problem of hyperspectral imagery, most of the research community focus on, discovering the importance of spectral signatures of hyperspectral images [63] [64]. However, according to the relation of non-linearity between the captured spectral information and the corresponding material, these methods are not suitable to analyze the attributes of HS imagery. Later many feature extraction methods discovered, including Principal Component Analysis (PCA) [65]. PCA extracts discriminative features to reduce the set of relevant bands since given spatial resolution, and neighboring pixels are highly correlated, in this case, spectral signatures contain vast redundant information. Independent component analysis [66] and linear discriminant analysis (LDA) [67] are used to extract spectral features. Those features were fed into classifiers to produce the final classification result. HS data contains structural spectral and spatial features. In terms of data classification accuracy, the structure of features is fundamental. However, classification with above

traditional methods, there is a possibility to lose the structural information about original features, and increasing the quantity and variability of available data, the computational cost may increase exponentially. In literature, we have two types of classification methods for Hyperspectral imagery: Pixel-level and Image-level classification. In the last two decades, hyperspectral imagery research community heavily focused on pixel-level HSI classification algorithms [68], and Kernel learning methods [69] [70] [71] [72] are achieved satisfactory performance with less computational cost compared with the traditional methods. The supervised, semi-supervised [73], and unsupervised [74] machine learning techniques discovered towards HSI classification, including K-means clustering [75], K-nearest neighbors [76] [77], maximum likelihood [78], support vector machine (SVM) [63] [79], random forest [80], logistic regression, and neural network are played a significant role to solve the classification problem. Nevertheless, the SVM has undoubtedly become the most extensively used approach in HSI classification research [81], then other non- parametric approaches. In general, compare with unsupervised classifiers, supervised classifiers attain higher classification results. The most commonly used supervised classifiers present in [82]. Since kernel learning methods [83] can learn intricate patterns with a few parameters. Kernel methods are also suitable for datasets, which are limited to training samples [84]. Multiple kernel learning methods [85] can extract the features based on groups of bands, contextual, or textural features. However, kernel learning methods also had some limitations, saying choose an appropriate kernel to solve a specific problem is tricky. However, according to the accuracy of the system, kernel methods are not suitable or outperformed in many domains under deep learning approaches.

3.2 Modern Classification Methods

For two decades, numerous advanced classification approaches, including fuzzy-sets, artificial neural networks, and expert systems, have extensively employed for image classification. In [86] explained the status and research preferences of land-cover

mapping for broad areas. [87]assessed land-cover classification strategies with average spatial resolution remotely sensed data. [88] [89] particularly concentrate on image processing strategies and classification algorithms. In usual, image classification methods assorted as parametric and non- parametric or hard and soft fuzzy classification, or supervised and unsupervised, or per- field, per-pixel and sub-pixel. A brief description of advanced classification strategies provided in the following subsections. A comprehensive summary of a particular classification method should refer to cited references.

3.3 Per-pixel classification

Initial per-pixel classifiers typically exhibit a signature by merging the spectra of all pixels in a training-set for a given feature. The resulting signature contains the contributions of all materials present in the training pixels but neglects the influence of the mixed pixels. Per-pixel classification approaches can be parametric or non-parametric. The parametric classifiers consider that a normally distributed dataset exists and that the statistical parameters like mean vector and covariance matrix generated from the training samples are representative. Nevertheless, the hypothesis of normal spectral distribution is frequently disrupted, especially in complex landscapes. Besides, insufficient, non-representative, or multi-mode distributed training samples can further present uncertainty to the image classification system. A different major drawback of the parametric classifiers lies in the problem of integrating spectral data with additional data. The maximum likelihood may be the most commonly used parametric classification approach in practice, because of its robustness and its easy availability in about any image-processing software.

With a non-parametric classification approach, the hypothesis of a normal distribution of the dataset may not require. No analytical parameters are required to distribute image categories. Non-parametric classification approaches are consequently particularly suitable for the association of non-spectral data into a classification system. Significant prior researches have shown that non-parametric classifiers may produce better outcomes than parametric classifiers in heterogeneous landscapes [34] [90]. Between the most usually applied non-parametric classification strategies are neural networks, decision trees, support vector machines, and expert systems.

In particular, for a decade, the neural network approaches are widely utilizing. The neural network has diverse advantages, including its non- parametric nature, arbitrary decision boundary capability, easy adaptation to different types of data and input structures, fuzzy output values, and generalization for use with multiple images, making it a promising system for land-cover classification [90]. The multilayer perceptron (MLP) is the popular type of neural network in image classification [91]. However, the variations in the dimensionality of a dataset and the features of training and testing sets may reduce the accuracy of image classification in [92]. Bagging, boosting, or a hybrid of both methods used to enhance classification performance in a non-parametric classification procedure. These methods have been used in decision trees [27] [26] [93] and a support vector machine [94] to enhance classifications.

3.4 Sub-pixel classification

Most classification methods based on per-pixel information, in which every pixel classified into one class or category and the land-cover classes are generally particular. Due to the heterogeneity of landscapes and the lack of spatial resolution of RS imagery, mixed pixels are usual in medium and coarse spatial resolution data. The appearance of mixed pixels has been identified as a significant problem, influencing the effective use of RS data in per-pixel classifications [95], [96]. Sub-pixel classification strategies had exploited to produce a more relevant representation and accurate area estimation of land covers than per-pixel approaches, mainly when inferior spatial resolution data used [48] [97][49] [98]. A fuzzy description, in which each location formed of multiple and partial associations of all input classes, is needed. Several methods have been employed to derive a soft classifier, including fuzzy-set theory, Dempster–Shafer theory, certainty factor [99], softening the outcome of a hard classification from maximum likelihood [100], IMAGINE's sub-pixel classifier [**Huguenin1997**], and neural networks [97] [101] [102]. The fuzzy-set strategy [103] [104] [105], [106] and spectral mixture analysis (SMA) classification [20] [107][108] are the most famous methods used to succeed the mixed pixel problem. One important drawback of sub-pixel classification lies in difficulty in estimating accuracy, as

discussed in 2.5.8. Spectral mixture analysis has identified as an efficient approach for dealing with the mixed pixel problem. It evaluates the individual pixel spectrum as a linear combination of a set of endmember spectra [107] [108]. The outcome of spectral mixture analysis typically presented in the form of parts of images, with a single image for each endmember spectrum, serving the area dimensions of the endmembers within the pixel. Endmember choice is one of the most essential aspects in spectral mixture analysis, and many prior studies [109] [110] [111] [112] [113] [114] [115] [116] has investigated the approaches. Earlier research has proved that SMA helps to improve the classification accuracy [107] [117], and is particularly important to improve the area estimation of land-cover categories based on coarse spatial resolution data.

3.5 Contextual classification

Contextual classification approaches have been extended to cope with the difficulty of intra-class spectral variations Gong1992 [118] [119] [120] [121] [122]. Contextual classification employs spatial information between neighboring pixels to enhance classification performance [5] [119] [123] [122]. A contextual classification approach may employ smoothing strategies, Markov random fields, spatial statistics, fuzzy logic, segmentation, or neural networks [124] [125] [126] [121] [122]. Usually, pre-smoothing classification strategies include contextual information as extra bands, and classification is then accompanied using normal spectral classifiers, during post-smoothing classification conveyed on classified images earlier developed utilizing spectral-based classifiers. The Markov random field-based contextual classifiers, such as iterated conditional modes, are the most commonly used methods in contextual classification [125] [122] and have demonstrated to be efficient in enhancing classification outcomes.

3.6 Deep Learning Approaches:

The awareness of CNN primarily introduced in [127] then simplified in [128]. Deep learning(DL) methods actively examined in image classification [129] [130], natural language processing [131], speech recognition [132], and other computer vision and pattern recognition tasks. These methods contain two more hidden layers to extract invariant and discriminant features from input data. For a decade, remote sensing communities showing great interest in the DL approach. The first time, HS Image classification using DL in [133]. CNN appeared in [134], where convolutional kernels learned automatically through data clustering. Later deep believe networks (DBN) used for HSI classification in [135]. Sparse theory and Manifold learning techniques attained cognitive performance [76] [136] using the mechanism of a shallow layer. However, these methods are not able to deal with the complex classification problem. However, there are two types of CNN based methods, which are spectral and spectral-spatial classifiers. Most of the HSI classification methods used spectral-spatial information. In [133] combination of PCA, CNN, and Logistic regression used for spectral and spatial classification. If only spectral approaches might produce enough results, it does not benefit from the spatial structure of the HS image. Indeed, neighbor pixels likely share structural relationship information. The efficiency, robustness of the model may improve due to the utilization of spatial information during the analysis.

3.7 Transposed Convolution:

Transposed Convolution (TC) also called Fractionally Strided Convolution or Deconvolution in literature [137]. However, TC different from Deconvolution, mathematically Deconvolution defined as the inverse of the convolution. Deconvolution used as a Decoder in DL methods, mostly for segmentation tasks to map the latent features back to input space. For the segmentation task, every Deconvolutional layer in the DL framework contains Deconvolution operation and unpooling. Unpooling up-samples the input feature map based on unpooling shifts, defined by the corresponding pooling operation of convolutional layers. In [138] proposed a

framework for semantic segmentation by learning a deconvolution network since Deconvolution densify sparse activation maps obtained by unpooling via convolutional operation through multiple filters. In [139] proposed an unsupervised deconvolution network approach based on the convolutional decomposition of images under sparsity constraints to extract features to capture the mid-level cues impulsively appears in image data. In [140] implemented a weakly supervised semantic segmentation framework, to build discriminative feature sets, by feature maps generated from the multiple deconvolutional layers which are robust against false positives, because the feature maps generated from the deconvolution layers contain less noise than convolutional layers. In [141] proposed Laplacian pyramid model for reconstructing sub-band residuals from the high-resolution image, in this work, every pyramid takes coarse-resolution feature maps as input, and predict high-frequency resolution residuals by up-sampling the input feature maps using transposed convolutions. At different scales by the contexts of objects, transposed convolutions [142] used to predict the lost spatial information for the segmentation of cell instances. To reduce the computation cost, concerning the trainable parameter, [143] proposed TCNet (Transposed Convolutional Network), TC is useful for real-time applications, because they need a simple and effective network for fast computations, TC can significantly reduce the size of the layers still it goes deeper. In order to improve the performance of real-time image super-resolution [144] proposed a novel hourglass-shape structured network called up-down network (UDNet). Usually, all the existing fast CNN with transposed convolution models able to transform low-resolution feature representations to high-resolution output. In this work, they are combining transposed convolution and spatial aggregation, which allows the network to transfers feature representations between low-resolution space and high-resolution space. For better mapping, the network can learn multiple times in both low resolution and high-resolution spaces. In [145], transposed convolution helps to force the model to focus on learning more details from the image in terms of visual perception by upscale the feature maps in a residual branch of the proposed network instead of identity branch, which also stabilize the learning process. Transposed convolution suffering from the checkboard problem, briefly, there is no direct relationship between neighboring pixels during learning; this is a unique advantage for pixel-level

classification problems. However, for object-level classification or recognition applications learning from neighboring pixels is a great advantage, to learn the network from adjacent or neighboring pixels on the up-sampling feature maps [146] proposed an approach called pixel transposed convolution layer (PixelTCL).

3.8 Dilated Convolutions

In this thesis, we adopted Dilation convolution or Atrous-convolutions. Dilated convolutions layer demonstrated in various tasks like classification, semantic segmentation, image super resolutions with significant improvement of accuracy's and which can transfer to various similar datasets, irrespective of their scales, spatial and channel-wise resolutions. Dilated convolutions are the best alternative to pooling operations because they contain sparse kernels. Dilated convolutions are Mathematically dilation convolution operation considered from [137]. Dilated convolution first introduced in [54], to aggregates multi-scale contextual information, this approach achieved great success in dense prediction or semantic segmentation task, and also explained how it achieves the exponential expansion of the output receptive field through dilation convolution. In order to segment small and crowded object in RS Imagery as well as for image classification in [147] proposed a semi-supervised architecture called local feature extraction module to contain dilation layer on the top of the network, to address this problem, the combination of the local feature due to sparsity of the kernel by decreasing the dilation factor. In [136] proposed a residual network with dilation layer, called dilated residual network. The difficulties from low-resolution feature maps, this work motivates the standard convolution layer, which produces low spatial resolution feature maps, and reduce classification accuracy. Since, trained models are not able to afford for best performance on subsequent applications, because which require wide-ranging scene understanding, these problems erased by dilation convolution, dilation layers can increase the resolution of output feature maps without reducing the receptive field of individual neurons. Image segmentation tasks, objects appeared on different scales with in the same

class; those variations are depending on the distance between the object and position of the camera. To address this kind of problem in [148] introduced a CNN architecture with multiple dilated convolution blocks. Since, by dilated convolution, features are separated by a certain distance during the convolution process. Not only for classification, even dilated convolutions are playing a significant role in generating density maps. In [149], suggested a CNN based network with dilation layers. In this work, they used dilation layers at the back-end of the network to extract in-depth features to preserve the spatial information. In [150] recommended an approach for weakly and semi-supervised segmentation. In this work, they investigate the advantages of dilation layers, and classical CNN models carry performance differences by their limitation on learning to generate significant image-level dense object localization maps. Since, by altering the dilation rates, dilation layers effectively enlarge the receptive fields of convolutional kernels and transfer the neighbor discriminative information to non-discriminative object regions, those regions are localization maps in a particular object. Most of the traditional image classifications used convolutional layers. Nevertheless, the drawback with the CNN contrast with Dilation Convolutions, which consuming enormous computing resources, to address this problem in [151] proposed the Dilated CNN model with two modules. Firstly they replaced convolutional kernels of traditional CNN by dilated convolutional kernels, and tested with handwritten digital recognition data sets, following they discovered, some information loss due to simple dilated convolutions. Then designed Hybrid Dilation Convolutional Neural Networks to solve the problem of information loss with varying dilation factor or rates, this hybrid model forded state of the art results with remote sensing dataset. In [152] proposed an approach toward semantic segmentation of a satellite image, in this task they used a mechanism by using dilation convolutions called serial-parallel combination dilated convolution, which assembles multi-scale features by enlarging the receptive fields.

3.9 Exponential Linear Unit(ELU)

Exponential Linear Unit (ELU) first introduced in [56] and explored, ELU implements exponential function about negative inputs, ELU can alleviate the vanishing gradient problem during training phase thorough identify positive values. ELU has the same shape as ReLU for the positive inputs. However, the output values of the ELU with the negative inputs become negative values, so the mean of the outputs of activation function goes near to zero, which is an advantage during learning and robust to noise. ELU also reduce the computational complexity [153] by reducing the bias shift effect, since, ELU brings the average gradient to closer to the unit natural gradient. By decreasing the forward propagated information and variation, ELU ensures a noise-robust deactivation state. Not only of computer vision tasks like segmentation or classification, deep learning adopted in every field, in [154] explored deep learning and machine learning algorithms in the field of Electrical systems. Specifically for predicting the behavior of energy systems, in this work, they proposed multi-layered deep neural network for short-term forecasting about the electric grid, in this work they used different activation function to reach significant results since ELU gives better performance in terms of Mean Absolute Percentage Error (MAPE) over other activation functions. They tested ELU with various combination of hidden layers like ELU with single and two hidden layers in both cases they attained required performance, and they explored ELU does not saturate with lager input values, and the average of the ELU function closer to zero because of the characteristics of the negative portion of the function guarantee faster and accurate learning. In [155] presented a report about a comparative study on batch normalization and different activation functions with various networks, they concluded ELU activation function with Residual and VGG networks achieved desired performance over others activation function; however, all the networks and datasets are not suitable to attain better performance with ELU activation function.

3.10 Patch Wise training

Most of the traditional deep learning algorithms use full image train CNN. However, according to devices which we are using to train neural networks having limited memory, in such cases, we are not able to feed larger images to CNN. However, we can split the entire image into small patches from the training dataset as well as a validation set. In [156] explored a new architecture called Random Patches Network (RPNet), to reduce the computational burden as well as to capture the objects tend to have different scales, which take the random image patches from a single image as convolutional kernels without any training process. In [157] proposed a framework to classify hyperspectral images; in this work, they designed two shallow networks, PCA Whitening and stacking spectral patches. In this work, they split the image into patches as spectral cubes concerning spectral information, since to avoid computational cost while retaining all spectral bands, they stack spectral patches by same color channels. In such a case, they extract unique texture patterns from original information for the individual color channels. For segmentation task, patch-wise training showing grate performance. However, size of the patch is a matter for some cases, in [158] explored the effectiveness of the patch size for segmentation task, in this work they discovered increasing the size of the patch leads to improve the classification, performance. [159] proves patch-level classification achieved significant results than an image-level classification by Decision Fusion Model, and which automatically locates discriminative patches by utilizing the spatial relationship of patches. Patch Based Classifier implemented using convolutional neural networks in [160] for automatic classification of medical images, in this work due to limited data they extracted image patches from a single image and the augmented individual patches to augment entire image, they witnessed patch wise training gives relevant diagnostic information.

3.11 Imbalanced Data

Class imbalanced data classification has been a hot topic in the research and academic community. There are many solutions recommended for imbalanced data

classification and other tasks, from data-level to algorithm-level. In [161] proposed the most popular data-level approach called Synthetic Minority Over-sampling/re-sampling Technique (SMOTE) to alleviate the problem of class imbalance and which prevents the over-fitting problem. However in [162] explored there are severe flaws with SMOTE, to overcome those conflicts they proposed an approach called Sigma Nearest Oversampling Based Convex Combination (SNOCC), which reproduce the original sample distribution naturally by finding nearest neighbor samples, mainly which renders the new samples by increasing the number of seed samples. However, especially for minor classes in the imbalanced dataset, in [163] proposed two approaches, which are borderline-SMOTE1 and borderline-SMOTE2, the approaches oversample the data from near borderline(near neighbor samples) minority samples. HS imagery differs from traditional imagery datasets; most of the HS datasets are imbalanced. [149] proposed an approach, especially for imbalanced HS imagery datasets based on orthogonal complement subspace projection(OCSP). It works on major and minor classes of HS datasets, which depends on the size of the dataset, if dataset larger enough, in order to avoid over-fitting OCSP eliminate or reduce significant class data to balance with minor classes, if small dataset OCSP create artificial samples from minor classes until balance all the class distribution in the dataset. [164] explored a review of problems with imbalanced data, and he explained different solutions for highly and partially imbalanced datasets for different real-time applications. Many initial works consider data class imbalance is a challenge for the classification task, and most of the work proposed data resampling or over-sampling and under-sampling techniques, but all these considered a data pre-processing. However, in [165] proposed an approach for the problem of class imbalance in a two-class classification task called divergence-encouraging auto-encoder (DEA), which explicitly learn the feature during training from both majority and minority classes simultaneously.

3.12 A summary of classification Strategies

Whereas many classification strategies are still developing, which strategies are convenient for features of interest in previous study areas are not adequately surmised.

Classification algorithms can be per-pixel, sub-pixel, and contextual based. Per-pixel classification is yet most generally utilized in practice. Nevertheless, the accuracy may not satisfy the provision of research because of the influence of the mixed pixel problem. Sub-pixel approaches have the potential to trade with the mixed pixel complication and may attain higher accuracy for medium and coarse spatial resolution images. For higher quality spatial resolution data, though mixed pixels might lessen, the spectral variation within land categories may also reduce the classification accuracy. Per- field classification strategies are most convenient for high spatial resolution images. When utilizing multi-source information, such as a mixture of spectral signatures, texture and context information, and additional data, advanced non-parametric classifiers, like a neural network, decision tree, and knowledge-based classification, maybe more adapted to handle these complicated data processes, and thus have obtained growing attention in the remote-sensing community in current years. The choice of a relevant classifier needs consideration of many factors, such as accuracy, algorithm performance, and computational resources [26] [119] reviewed three criteria—the objective of classification, available computer resources, and efficient detachment of the categories. In practice, the spatial resolution of the RS data, the use of additional data, the classification technique, the accessible software, and the expertise analysts may all influence the decision of choosing a classifier. A related study of diverse classifiers is frequently managed to find the best classification outcome for a distinct study [5] [91] [119] [121] [166] [167] [168] [169]. In several instances, contextual-based classifiers, per-field strategies, and machine-learning algorithms produce a more reliable classification outcome than MLC, even if some compromises between classification accuracy, time consumption, and computing resources.

Chapter 4

Used Datasets

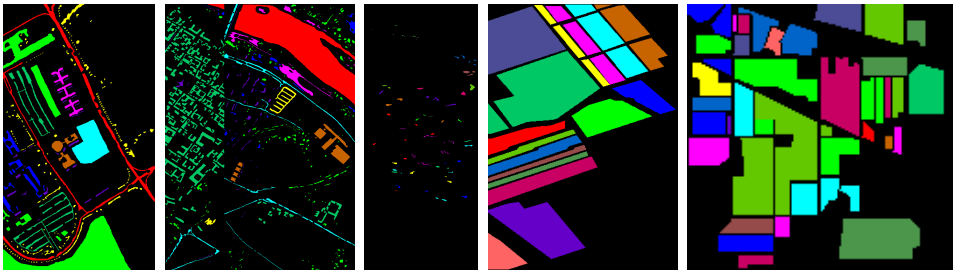
The proposed approach has been evaluated on five benchmark datasets, namely University of Pavia, Pavia Center, Botswana, Salinas and Indian Pines. All these datasets are publicly available and employed in many researches in the field. In this section we report the basic information about Datasets we used.

4.1 University of Pavia

The **University of Pavia** dataset, has been acquired using Reflective Optics System Imaging Spectrometer (ROSIS) over Pavia, in north Italy. This dataset includes a 610×610 pixels image with 103 bands. The spatial resolution is 1.3 meters per pixel. This area presents 9 different kinds of terrains; hence, each pixel annotated across 9 classes. The ground truth image is shown in the Figure. 4.1(a) and The class names and corresponding numbers of ground truth observations used in the experiments listed in the Table:4.1.

4.2 Pavia Center

The **Pavia Center** dataset has acquired by ROSIS and presents the same number of classes. The hyperspectral image consists of 102 bands; image size is 1096×1096 and spatial resolution 1.3 meters. Figure. 4.1(b) depicts the ground truth and the class names and corresponding numbers of ground truth observations used in the experiments listed in the Table:4.2.



(a) (b) (c) (d) (e)

Figure 4.1: (a) Ground truth image of Pavia University Dataset; (b) Ground truth image of Pavia Center Dataset; (c) Ground truth image of Botswana Dataset; (d) Ground truth image of Salinas Dataset; (e) Ground truth image of Indian Pines Dataset.

	Class	**Samples**
1	Asphalt	6631
2	Meadows	18649
3	Gravel	2099
4	Trees	3064
5	Painted metal sheets	1345
6	Bare Soil	5029
7	Bitumen	1330
8	Self-Blocking Bricks	3682
9	Shadows	947

Table 4.1: Groundtruth classes for the Pavia University scene dataset and their respective samples number

	Class	**Samples**
1	Water	824
2	Trees	820
3	Asphalt	816
4	Self-Blocking Bricks	808
5	Bitumen	808
6	Tiles	1260
7	Shadows	476
8	Meadows	824
9	Bare Soil	820

Table 4.2: Groundtruth classes for the Pavia centre scene dataset and their respective samples number

4.3 Botswana

The **Botswana** has been collected by the Hyperion sensor NASA EO-1 satellite over the Okavango Delta in Botswana. This image presents 145 bands of size 1476×256 pixels; spatial resolution is 30 meters per pixel and wavelengths covering $400nm$ to $2500nm$. It includes 14 distinct land cover types. The ground truth data shown in Figur: 4.1(c), and the class names and corresponding numbers of ground truth observations used in the experiments listed in the Table:4.3.

	Class	**Samples**
1	Water	270
2	Hippo grass	101
3	Floodplain grasses 1	251
4	Floodplain grasses 2	215
5	Reeds	269
6	Riparian	269
7	Firescar	259
8	Island interior	203
9	Acacia woodlands	314
10	Acacia shrublands	248
11	Acacia grasslands	305
12	Short mopane	181
13	Mixed mopane	268
14	Exposed soils	95

Table 4.3: Ground-Truth classes for the Botswana dataset and their respective samples number

4.4 Salinas

The **Salinas** data have been collected by AVIRIS sensor with a spatial resolution of 3.7 meters. Image size is 512×217 and each pixel is labeled across 16 classes. Ground truth reported in Figure. 4.1(d) and The class names and corresponding numbers of ground truth observations used in the experiments listed in the Table:4.4.. Original data consisted of 224 bands, but the 20 ones related to water absorption have

discarded. Hence, it includes the 204 remaining bands.

	Class	**Samples**
1	Brocoli$_g reen_w eeds_1$	2009
2	Brocoli$_g reen_w eeds_2$	3726
3	Fallow	1976
4	Fallow$_r ough_p low$	1394
5	Fallow$_s mooth$	2678
6	Stubble	3959
7	Celery	3579
8	Grapes$_u ntrained$	11271
9	Soil$_v inyard_d evelop$	6203
10	Corn$_s enesced_g reen_w eeds$	3278
11	Lettuce$_r omaine_4 wk$	1068
12	Lettuce$_r omaine_5 wk$	1927
13	Lettuce$_r omaine_6 wk$	916
14	Lettuce$_r omaine_7 wk$	1070
15	Vinyard$_u ntrained$	7268
16	Vinyard$_v ertical_t rellis$	1807

Table 4.4: Groundtruth classes for the Salinas Scene dataset and their respective samples number

4.5 Indian Pines

The **Indian Pines** data have been acquired by AVIRIS sensor over the agricultural Indian Pines site in North-western Indiana with a spatial resolution of 20m per pixel and consists of 145×145 pixels and 224 spectral reflectance bands across the spectral range from 0.2 to $2.4 \mu m$. This scene is a subset of a larger one. It consists of 16 classes. However, the number of channels has been reduced to 200 by removing bands covering the region of water absorption. We report its ground truth in Fig. 4.1(e) and The class names and corresponding numbers of ground truth observations used in the experiments listed in the Table:4.5.

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

Table 4.5: Groundtruth classes for the Indian Pines Scene dataset and their respective samples number

Chapter 5

Proposed methodology

This chapter presents a proposed methodology of this thesis and thoroughly discussed. Moreover, further sections we discussed data pre-processing techniques and the learning process of the proposed network presented. The Figure: 5.2 helps us to understand the pipeline of the thesis

5.1 Summary

With the evolution of advanced imaging machines, hyperspectral imaging provides an alternative approach to analyze and tackle traditional problems in remote sensing and computer vision. In order to use sufficient spectral and spatial information from hyperspectral images, it is necessary to develop a series of methods to extract a highly comprehensive and discriminative representation of given objects, thus promoting fundamental pattern recognition tasks, such as object detection and recognition, and image classification. Existing techniques of remote sensing and computer vision focus on both grayscale based spatial feature or pixel-wise spectral feature. Although research has been work on the fusion of spatial and spectral information, there is a high demand for developing novel, effective, and efficient spectral-spatial feature extraction methods for various applications from industry and society. This thesis presents a novel spectral-spatial feature extraction methods for hyperspectral image classification using Deep Learning framework. Which focuses on a fundamental topic in computer vision or remote sensing. These methods are obtained from conventional two-dimensional methods and then prolonged to hyperspectral images. It is essential to point out that the spatial or spectral information to be obtained varies depending on different tasks or objectives. The final performance of the

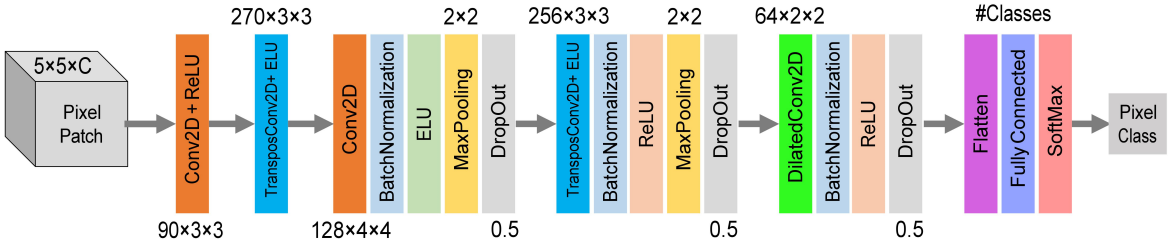


Figure 5.1: Proposed CNN Architecture

model entirely depends on the input data and information from the previous layers for convolutional neural networks. To extract valuable information through the entire network, we used three different feature extraction layers and well-implemented activation functions. However, the evaluation of the model is also an important task. Since high variance in the data leads to miss-classification on unseen data, to address this kind of situation, train the network with different data distribution, then we should know how our method efficient for further applications.

5.2 Architecture

In this section, we describe a new CNN architecture to classify individual pixels of hyperspectral imagery. Proposed architecture aims to classify pixels by starting from $P \times P$ patches. The reduced size of the input, motivate to employ Transposed Convolutions Layers for up-sampling and the Dilated Convolution layer to exponentially expanding receptive fields without losing resolution or coverage. The architecture consists of six layers. We found out that in the first two layers, the network suffered from neurons dying problem. Hence we decided to use Exponential Linear Unit (ELU) activation instead of Rectified Linear Unit (ReLU). This problem, also known as "dying ReLU", happens when the neuron gets stuck and always outputs 0. Hence, no gradients flow backward through the neuron, and it can define as "dead". In other words, this stops the learning process of the CNN [56].

The layers include standard Convolution followed by Transposed Convolutions layers (proposed by [170] as Deconvolution). For more explanation about the effectiveness of the Transposed Convolution explained in sec:2.6.1. Transposed Convolution is the process of going in the opposite direction of a standard convolution. This

work is done by preserving the connectivity of the pattern. They aim to densify the sparse activation of pooling layers and give a dense activation map as output.

In the fourth layer, instead of standard Convolution, we use Dilated Convolution to create a filter that presents spaces between each pixel. In order to not decrease the feature map resolution, we choose the Dilated Convolution layer with a dilation factor of 2. Batch Normalization used to scale the activation and to stabilize the network by normalizing the output of the previous layer. Moreover, to reduce network overfitting, we introduce Drop Out modules with a factor of 0.5. As in most classification architectures, the last layer is Fully-Connected, and SoftMax units provide the output probability for each class.

Finally, we use ADAM (Adaptive Moment Estimation) optimizer [171] because it presents the advantages of both Adaptive Gradient and Root Mean Square Propagation. We selected the following parameters: learning rate 0.001; $\beta_1 = 0.9$; $\beta_2 = 0.99$. These values control the decay rates of the exponential past gradients and past squared gradients, respectively. The proposed architecture reported in Figure.5.1.

5.3 Data pre-processing

5.3.1 Data Preparation

To prepare data, initially, we used Principal Components Analysis 2.6.4 to reduce the high dimensionality of the original image (i.e., hundreds of spectral bands). PCA is a mathematical procedure to move N-dimensional data in another N-dimensional space where the dimensions are linearly uncorrelated [57]. Additionally, it allows selecting the M most representative dimensions. Consequently, a HSI pixel $x_{i,j}$ with N spectral bands, can be reduced to a $\overline{x}_{i,j}$ vector in M-dimensional space, with $M < N$. Then, for each pixel in the reduced space, we extract a $P \times P$ patch to use for its classification. Hence, the pixel-wise dataset for HSI pixels classification includes a collection of patches in the new M-dimensional space.

Secondly, we randomly split the patch dataset into Training, Validation, and Test sets. Intuitively, the presence of adjacent patches in both, Training and Test sets, could drive biased classification. Nevertheless, we ignore this fact similar to the state-of-art works in order to perform a proper comparison with them. To face the problem of the imbalanced dataset, we operate an oversampling on the

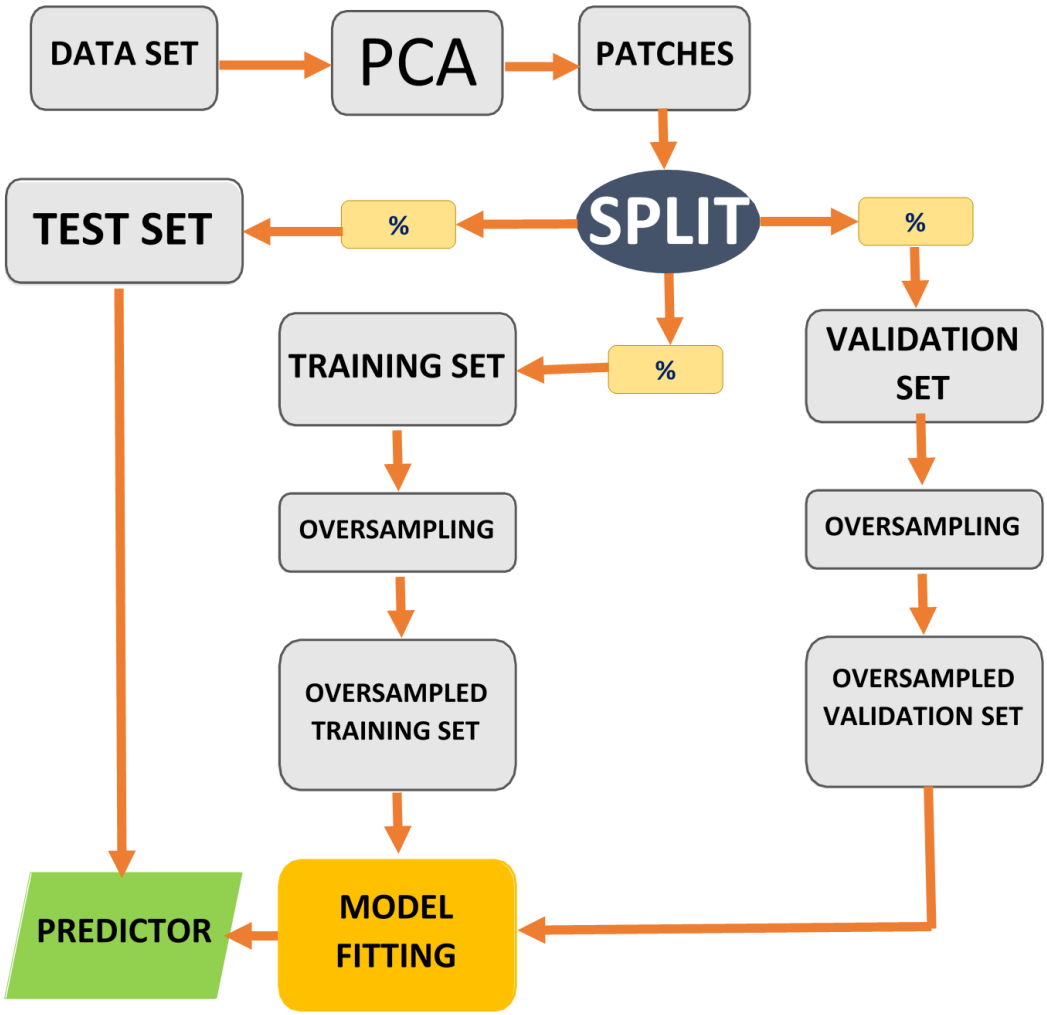


Figure 5.2: Pipeline of proposed methodology

underrepresented classes of Training and Validation sets. To remark the problem of class imbalance, in Figure: 2.14, report the percentage distribution across the 16 classes of the Indian Pines dataset. It can be noted the severe imbalance: while almost the 25% of the samples belong to the class 11, the classes 1, 7, and 9 includes less than 1% of the samples. The employed oversampling strategy [172] aims to replicate samples in minority classes. If the largest class includes S samples, after this process, all the classes will include about S samples.

5.3.2 Patch Selection

Most of the Neural Networks learned or trained by feeding the entire image or video for classification, segmentation, or recognition tasks. However, Neural networks have some limitations; for example, computationally, there is limited memory to feed images or videos with specific dimensions. For traditional RGB image contains spatially high resolution, but there are fixed channels. In the case of hyperspectral or multispectral images, spatially as well as spectrally contains enormous resolution. Due to large spatial and spectral resolution, neural networks suffered from a heavy computational burden. During training, there is an information loss, particularly in the pooling process, due to larger images, to address these problems, most of the works in literature implemented an important technique, which is called SPLIT AND RULE. Precisely split the image into random image patches (crop the large image into many non-overlapped images), then use these patches as input to CNN instead large image.

Patch-wise Training Image size to train a neural network have some limitation concerning the computer memory and Graphics Processing Unit (GPU). The images in the HS data set have huge dimensions so that the full images cannot feed to NN for training. Train a neural network with a patch-wise manner is a way to deal with a huge amount of data; in this method, many small patches taken from the original HS images from the training data set. After training the network, test image patches fed into the network to yield a classification performance. In remote sensing applications, satellite images or HS images have high dimensions, comparable to the aerial image dataset. In the case of classification using neural networks, the training process implemented using patches of the original image. In patch-wise training, the global scene context is not applicable here; consider the task is individual pixel classification. The scene composition remains the same overall data; for example, the location of all the classes never moves over time. As this information is not going to be present during training, these global spatial features are not able learned. Hence, pixel-wise classification/prediction only based on features about spatial and spectral structure covering an area as the patches. Patch-wise trained networks have more significant generalizing properties in a sense they can apply on scenes with slightly different spatial structures.

5.3.3 Data Set Partitioning

Initially, the patches are selected randomly from the entire HS data set then separated into two distinct sets. The first one is the test set. The second one is to train the network. The second set also separated into two parts, which are train and validation set. The process we used to create the data sets as described in Figure: 5.3. It is necessary to keep training and test data separate in predictive modeling, as it restricts further positive results or data leakage.

Neural network performance may improve by increasing the size of the training data. However, limitations are inflicted by computer memory and computational time. Maintaining the proper number of input samples, as well as the size of the patch or input image, is necessary to obtain the optimal result. The following results Table; 6.9 presents training two networks with the same training data sets and sizes, using patch size is 55, and Both training data sets have similar compositions regarding the percentage of pixels per class. Resampling/ oversampling is applied in this experiment as well. Less patch size, training is more stable. Moreover, when training a network with small training data set, the difference between training and validation accuracy increases by every epoch, which is the evidence for overfitting the training data. However, the other side of using more training is slower and broader memory usage. Both classification maps show misclassification.

5.3.4 Data Augmentation

We randomly split the patches from the whole image as Training, Validation, and Test sets. Intuitively, the presence of adjacent patches in both, Training and Test sets, could drive biased classification. Nevertheless, we ignore this fact similar to the state-of-art works in order to perform a proper comparison with them. To face the problem of the imbalanced dataset, we operate an oversampling on the under presented classes of Training and Validation sets. To remark the problem of class imbalance, in Fig. 2.14 we report the percentage distribution across the 16 classes of Indian Pines dataset. It can be noted the severe imbalance: while almost the 25% of the samples belong to class 11, the classes1, 7,, and 9 include less than 1% of the samples. The employed oversampling strategy [172], aims to replicate samples

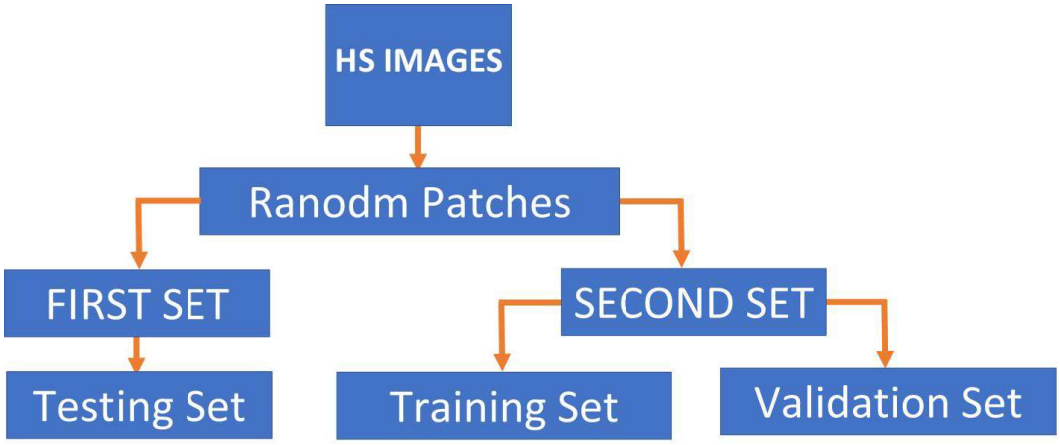


Figure 5.3: Test,Train and Validation Data Splitting

in minority classes. If the largest class includes S samples, after this process, all the classes will include about S samples.

After oversampling on Training and Validation, we train the proposed network. For an unbiased test, oversampling is not used on the Test set. Figure 5.2 shows the pipeline we adopt.

After oversampling on Training and Validation, we train the proposed network. For an unbiased test, oversampling is not used on the Test set. Figure 5.2 shows the pipeline we adopt.

5.3.5 Network Learning for Classification

In this section, we describe how the network learns to obtain a robust classifier. Initially, we describe the technical details which we used for whole research.

Technical Details: We design the neural network using Keras. Keras is a high-level neural networks Application Programming Interface (API) written in Python and capable of works using TensorFlow as backend. Keras, having deep learning libraries for machine learning, which allows for easy and fast prototyping, and it supports both convolutional neural networks and recurrent neural networks and also a combination of both. The network was designed by stacking network layers on top of each other, as each layer type has its function in Keras. Training process implemented on NVIDIA Tesla K80, which contains 12 Gigabytes of RAM.

Learning Process We randomly select a certain number of patches(cropped input images) from the Hyperspectral image for the training and validation sets, and we use the test data to evaluate the performance of the proposed network. For each training patch, we crop non-overlapped surrounding 5×5 neighboring pixels from the whole image for learning convolutional layers. The proposed network contains approximately 2000K parameters, which are learned from several hundreds of training pixels from each material category. To neglect overfitting and class imbalance, we augment the training and validation samples by the oversampling technique.

To learning the proposed network, Adaptive Moment Estimation(ADAM) with a batch size of 256 samples used with 100 epochs, a momentum of 0.9, a weight decay of 0.0006, and a gamma of 0.9. We initially set a base learning rate of 0.001. To learn the network, the last layer of the network is using by a softmax layer, commonly used for learning convolutional layers. The All layers initialized with a zero-mean Gaussian distribution with a standard deviation of 0.005. Biases of all convolutional layers initialized with 1.

Initially, the first layer of the network contains a convolutional layer with the ReLU activation function, which gen low-level feature maps from the input image. However, those feature maps do not have complete information. Because when ReLU receives the negative inputs, neurons able to activate since weights are not able to update for a zero gradient. As the ReLU gradient at zero is zero, weights remain unchanged Dead weights do not contribute to the learning process. For this reason, we lose some information from the initial layers. However, the convolution

layer reduces the feature maps to the following layer. By the cause of less information passes from initial layers, it is difficult to reach optimal performance, especially for dance pixel classification. To address this problem, we introduce Transposed convolution in the second layer of the network with the ELU activation function. Because the ELU activation function prevents the problem of network performance degrading due to so-called Dying Gradients. Assign ELU in initial layers may provide significant information to following layers.

Since adopting the Transposed convolution layer, upsample the feature maps. Consider, dense pixel classification, especially for hyperspectral data classification most of the neighboring pixels are the same class. In this case, Transposed convolution split the neighboring pixels into different positions, by shifting the neighboring pixel network can put the effort to learn individual pixels spectrally and spatially. ELU fixed with the Transposed convolution layer; ELU activates on the zero inputs also because bias also exists there. For more information about the ELU activation function, please refer to the sec: 2.6.3.

Eventually, Dilation layer helps us achieve stable performance. Dilation layer improves input receptive fields without losing information and resolution; in the last layers of CNN has high-level feature maps, which are essential to achieve stable classification performance.

	Conv_1	**Transposed 2Dconv_1**	**Conv_2**	**Transposed 2Dconv_2**	**Dilation Conv**
Number of filters	C	3 X C	128	256	64
Kernel size	3 x 3	3 x 3	4 x 4	4 x 4	2 x 2
			Max-Pool	Max-Pool	
Kernel size			2 x 2	2 x 2	

Table 5.1: Parameters Used to Design The Proposed CNN Architecture (C = Number of Principal Components)

Chapter 6

Experiments and Results

Note : In this chapter we presents about the experimental procedures and discused about Results of respected experiments. In section-6.1 and section 6.2 presents the results which are obtained in initial experiments; those results are published in a conference 7. Section-6.3 we discussed about all the experiments we done in this thesis.

6.1 Summary of Experiments

This section gives a brief observation of the following experiments and results. The lack of a common experimental setting in literature makes very time consuming a rigorous comparison with state-of-art approaches. For instance, dataset splitting is often different and this may influence the performances. Moreover, in some works, no test set is employed. Experimental results undoubtedly prove that the proposed method is competitive. Actually, a stronger argument in this sense, would require a different experimental settings for each state-of-art work or the implementation of all of them. Since, this is beyond the scope of previous methodologies, this thesis and considered as an achievement.

Initially, the most important observation is the feature extraction stage. Since, feature extraction to learn the network via standard, transposed convolution and dilation layers are played most important role in this work.

However, in order to understand about the stability of model, the initial observation from the presented results, which is increasing the performance of the model with respect to the number of training samples. Since there is small difference in

the performance with respect to the number of training samples from 5%*to*90% of its data.

The following sections describes experiments with a little deep representation of results are presented.

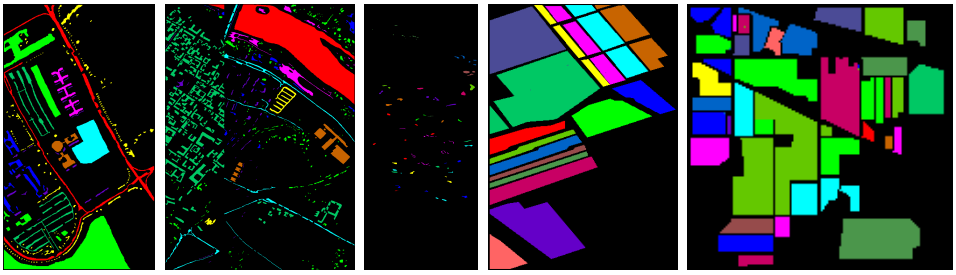
6.2 Published Experiment

To demonstrate the validity of the proposed approach we perform a 5-Fold Cross validation test on the datasets described in chapter-4. PCA is used to reduce the original number of spectral bands to 30 dimensions, while a patch size of 5×5 is chosen. Then, we randomly select 60% of the patches for Training, 20% for Validation and 20% for Test. Training and Validation set are oversampled and then the network is trained from scratch. As in most of the literature works, we adopted overall accuracy (OA), average accuracy (AA), and kappa coefficient (Kappa) for performance evaluation. OA is the ratio between the number of correctly classified pixels and the number of all classified pixels. AA is the average of classification accuracy of all classes, while the Cohen's Kappa coefficient is used to measure the agreement of classification for all the classes.

The results show an overall accuracy of 99.93%, 99.99%, 100.00%, 99.99% and 99.81% for Pavia University, Pavia Center, Botswana, Salinas and Indian Pines dataset respectively.

Finally, for each dataset, we report the results of the most recent works. Since in the state-of-art works, not all the datasets have been used, we report a different table for each of them. Moreover, we do not report some AA and Kappa values because they are missing in the original works. As shown in Tables 6.1, 6.2, 6.3, 6.4, 6.5 the proposed approach outperforms all the other methods for all the datasets. In Botswana dataset, we even achieve the 100.00% of performance on Botswana dataset. Moreover, the low standard deviation on 5-Fold Cross validation confirm our approach is stable for each of the datasets.

For a qualitatively evaluation, we also report the predicted classes in Figures 6.1(a), 6.1(b), 6.1(c), 6.1(d) and 6.1(e).



(a) (b) (c) (d) (e)

Figure 6.1: (a) Predicted classes of Pavia University Dataset; (b) Predicted classes of Pavia Center Dataset; (c) Predicted classes of Botswana Dataset; (d) Predicted classes of Salinas Dataset; (e) Predicted classes of Indian Pine Dataset.

Method	[173]	[174]	[175]	[176]	[177]	**Our**
OA(%)	99.62	99.64	99.18	98.90	99.39	**99.93 ±0.020**
AA(%)	-	99.61	98.75	98.49	98.85	**99.94 ±0.028**
KAPPA(%)	-	99.53	98.95	98.52	99.20	**99.91 ±0.027**

Table 6.1: Overall accuracy (OA), Average accuracy (AA) and Kappa coefficient for Pavia University dataset.

Method	[173]	[178]	[179]	[176]	[180]	**Our**
OA(%)	99.91	97.81	99.73	99.75	98.85	**99.99 ±0.001**
AA(%)	-	92.81	99.25	99.40	98.43	**99.99 ±0.005**
KAPPA(%)	-	96.88	99.61	99.64	97.90	**99.99 ±0.002**

Table 6.2: Overall accuracy (OA), Average accuracy (AA) and Kappa coefficient for Pavia Center dataset.

Method	[181]	[182]	[183]	[184]	[177]	**Our**
OA(%)	97.44	88.19	89.44	97.93	99.55	**100.00 ±0.000**
AA(%)	97.80	89.53	90.60	-	99.60	**100.00 ±0.000**
KAPPA(%)	-	-	88.57	96.30	99.51	**100.00 ±0.000**

Table 6.3: Overall accuracy (OA), Average accuracy (AA) and Kappa coefficient for Botswana dataset.

6.3 Experiments

Note : In this section we present results in many tables. Please consider the main motivation behind to present all those results, to show the robustness and stability of the model trained on various experimental settings and quantity of data distribution.

Methods	[173]	[174]	[185]	[176]	[181]	**Our**
OA(%)	99.53	98.34	94.76	99.38	99.37	**99.99 ±0.014**
AA(%)	-	99.33	94.75	99.76	99.67	**99.98 ±0.009**
KAPPA(%)	-	98.15	94.16	99.30	-	**99.99 ±0.010**

Table 6.4: Overall accuracy (OA), Average accuracy (AA) and Kappa coefficient for Salinas dataset.

Methods	[173]	[174]	[175]	[186]	[177]	**Our**
OA(%)	98.88	97.57	96.87	90.08	99.07	**99.81 ±0.065**
AA(%)	-	98.46	96.75	93.09	98.66	**99.83 ±0.068**
KAPPA(%)	-	97.23	95.67	88.75	98.93	**99.79 ±0.074**

Table 6.5: Overall accuracy (OA), Average accuracy (AA) and Kappa coefficient for Indian Pines dataset.

Finally, we pick the lowest (%) of training data i.e., 5% of training data and the evaluation results presented in tables from 6.11 to 6.30 and the failure results with the implementation of various standard elements in the literature are presented in tables from tables: from 6.6 to 6.10.

All the following experiments are implemented by a methodology as section: 6.2 except cross-validation technique(in 6.2 we implemented 5-Fold cross-validation remaining all the experiments done by 3-Fold cross-validation) and we obtain satisfactory results. Though, according to literature many researchers have developed best and accurate models of Hyperspectral Image Classification. Nevertheless, those model are not robust to hyperspectral data obtained from different sensors. However, The ultimate aim of this thesis is to develop a unique and robust classifier using Deep Learning and machine learning algorithms for hyperspectral data obtained from different sensor.

To demonstrate the validity of the proposed approach we implemented several training and evaluation strategies. In this experiments we mainly concentrate to find the stability and robustness of classifier. We perform a 3-fold Cross validation on the datasets described in Section 4. PCA helps us to reduce the dimensionality of given data. Since model performance is depends on the size and the complexity of the input data. Therefore, most of the state of are methods used 30 principal components, but, we evaluate our model on different number of PCA components because the dimensionality of used dataset are not unique. We implement the experiments with

various quantity of principal components such as 15%, 25%, 50% and 100% on all the datasets 4 to find is stability point of proposed model with respect to performance and computation complexity depends on the dimensions of the data. Since, if observe the these results with lowest is 15% PCA components table:6.11, 6.15, 6.23, 6.19 and 6.27 are very stable with different train and test data distribution and also whcih consumes very less computation time.

According to literature image classifier performance improves with size of the training samples and size of input image. However, We perform experiments with different patch size(croped original image size) 5×5 and 15×15 to understand how learing performance depends on the size of the input image. However, Training and Validation set are oversampled to address the problem of data imbalance and then the network is trained from scratch. As per most of the literature works, we adopted overall accuracy (OA), average accuracy (AA), and kappa coefficient (Kappa) for performance evaluation. Over All is the ratio between the number of correctly classified pixels and the number of all classified pixels. Average Accuracy is the average of classification accuracy of all classes, while the Cohen's Kappa coefficient is used to measure the agreement of classification for all the classes.

However, computational cost increased with the size of input image. Still, there is huge advantage with larger input image for image classification. Though, for pixel wise classification works well with smaller input images. Since, if task is less complex, larger input lead to over fitting.

6.4 Experimental Results and Compare with Failure Cases

In this section, we presented the experimental results obtained from the principal elements and hyperparameters we used, which are the Transposed Convolution layer 2.6.1, the Dilation layer 2.6.2 and Exponential Linear Unit 2.6.3 comparing with the results obtained from by using the absence of these elements.

These results explain how the proposed model is robust to various hyperspectral datasets. In these experiments, we consider all the datasets and 50% 20% and 30% of data used for train, validation, and testing. Table: 6.6 show the classification results proposed model in terms of Average Accuracy (AA), in the proposed approach

used Exponential Linear Unit 2.6.3 we obtained satisfactory results. However, most of the state of the art research explains that ReLU obtains the optimal results. However, usage of ELU in initial layers may extract negative features, which prevents information loss. Since, ELU may increase the computational cost, though, combination with ReLU may converge the generalization error faster. Figure: 6.4(a) obtained from 6.6 shows the stability of the proposed compare with the presence of ReLU instead ELU, here the implementation of ELU gives us better performance then ReLU. However, Figure: 6.4 presents the visual understanding about model stability. To Table: 6.7 presents the performance results of the proposed approach compare with the standard convolution layer replaced in place of the dilation layer. Nevertheless, Tables: 6.8, 6.9 and 6.10 presents the performance results of without resampling, input image patch size of 15×15 instead 5×5 and replace the transposed, dilation convolutions with standard convolutions respectively. To deeper understanding, look at the Figures: 6.4 shows visual understanding about the stability of proposed model in terms of performance, and the results with respect these plots are presented in Tables: 6.6 to 6.10. If we observe the the plots of learning curves 6.2 and 6.3 express there is no overfitting during training.

	Botswana	**Pavia University**	**Indian Pines**	**Pavia Center**	**Salinas**
Proposed Approach	1	0.9996	0.9994	0.9997	0.9998
ReLU	0.9978	0.996385	0.98918	0.99816	0.99915

Table 6.6: Evaluation Results (Average Accuracy) comparing with proposed approach with the same topology replaced position of ELU With ReLU, and 50% data used to train the network, 30% data used to test the model

	Botswana	**Pavia University**	**Indian Pines**	**Pavia Center**	**Salinas**
Proposed Approach	1	0.9996	0.9994	0.9997	0.9998
Convolution	0.997	0.99625	0.99329	0.99857	0.99919

Table 6.7: Evaluation Results (Average Accuracy) comparing with proposed approach with the same topology replaced Dilation Layer With Standard Convolution Layer

	Botswana	**Pavia University**	**Indian Pines**	**Pavia Center**	**Salinas**
Proposed Approach	1	0.9996	0.9994	0.9997	0.9998
Without Resampling	0.984759	0.99759	0.71082	0.99843	0.90495

Table 6.8: Evaluation results (Average Accuracy) proposed approach with the same topology and without implementation of Resampling or Oversampling

	Botswana	**Pavia University**	**Indian Pines**	**Pavia Center**	**Salinas**
Proposed Approach	1	0.9996	0.9994	0.9997	0.9998
PatchSize = 15	0.99975	0.95984	0.99277	0.99893	0.99951

Table 6.9: Evaluation results (Average Accuracy) proposed approach with the same topology and replaced Patch Size of 5 With 15×15

	Botswana	**Pavia University**	**Indian Pines**	**Pavia Center**	**Salinas**
Proposed Approach	1	0.9996	0.9994	0.9997	0.9998
Only Convolution	0.97777	0.99555	0.72755	0.99854	0.99946

Table 6.10: Evaluation Results (Average Accuracy) comparing with proposed approach with the same topology replaced Transposed convolution and Dilation Convolution With Standard Convolution layers

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.99892	1.11E-16	0.9986	1.11E-16	0.9985	1.11E-16	139.6	1.0
20_20_60	0.99940	0.0005	0.99931	0.0006	0.9992	0.00069	180.7	1.0
30_20_50	1	0	1	0	1	0	203.4	1.0
40_20_40	1	0	1	0	1	0	232.9	1.0
50_20_30	1	0	1	0	1	0	269.3	1.0
60_20_20	1	0	1	0	1	0	308.1	1.0
70_10_20	1	0	1	0	1	0	327.8	1.0
80_10_10	1	0	1	0	1	0	352.5	1.0
90_5_5	1	0	1	0	1	0	386.7	1.0

Table 6.11: Botswana Data set 15 % PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9986	0.00039	0.9983	0.00041	0.9982	0.0004	160.2	1.0
20_20_60	0.9996	0.0002	0.9996	0.0002	0.9996	0.0002	210.8	1.0
30_20_50	0.9994	1.11E-16	0.9993	0.00E+00	0.9993	0.00E+00	92.2	1.0
40_20_40	1	0	1	0	1	0	273.3	1.0
50_20_30	1	0	1	0	1	0	319.0	1.0
60_20_20	1	0	1	0	1	0	362.3	1.0
70_10_20	1	0	1	0	1	0	392.8	1.0
80_10_10	1	0	1	0	1	0	427.4	1.0
90_5_5	1	0	1	0	1	0	461.1	0.9

Table 6.12: Botswana Data set 25%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9970	2.86E-05	0.9964	0	0.9961	1.20E-08	244.1	1.0
20_20_60	1	0	1	0	1	0	318.6	1.0
30_20_50	1	0	1	0	1	0	371.2	1.0
40_20_40	1	0	1	0	1	0	430.4	1.0
50_20_30	1	0	1	0	1	0	500.1	1.0
60_20_20	1	0	1	0	1	0	569.2	1.1
70_10_20	1	0	1	0	1	0	611.0	1.0
80_10_10	1	0	1	0	1	0	677.7	1.0
90_5_5	1	0	1	0	1	0	747.0	1.0

Table 6.13: Botswana Data set 50%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9979	0.0002	0.9976	0.0002	0.9974	0.0002	518.9	1.0
20_20_60	0.9982	0.0005	0.9981	0.0006	0.9979	0.00069	653.7	1.0
30_20_50	0.9995	0.0003	0.9995894	0.0002	0.9995	0.0003	777.4	1.0
40_20_40	0.9990	0.0003	0.9989	0.0003	0.9988	0.0003	910.0	1.0
50_20_30	1	0	1	0	1	0	1050.0	1.1
60_20_20	0.9991	6.12 E-04	0.9989	7.25 E-04	0.9988	7.86 E-04	1192.5	1.1
70_10_20	1	0	1	0	1	0	1297.6	1.0
80_10_10	1	0	1	0	1	0	1442.4	1.0
90_5_5	1	0	1	0	1	0	1587.1	1.0

Table 6.14: Botswana Data set 100%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9880	0.9829	0.9829	0.0008	0.9805	0.0009	380.6	1.2
20_20_60	0.9910	0.0010	0.9906	0.0004	0.9893	0.0004	504.9	1.3
30_20_50	0.9949	0.0012	0.9933	0.0011	0.9924	0.0012	623.8	1.4
40_20_40	0.9988	0.0004	0.9980	0.0005	0.9977	0.0006	750.7	1.5
50_20_30	0.9990	0.0003	0.9986	0.0007	0.9985	0.0008	863.6	1.6
60_20_20	0.9989	0.0003	0.9986	0.0002	0.99851	0.0002	976.2	1.7
70_10_20	0.9994	0.0005	0.9988	0.0008	0.9987	0.0009	1066.9	1.3
80_10_10	1	0	1	0	1	0	1197.0	1.3
90_5_5	1	0	1	0	1	0	1294.5	1.1

Table 6.15: Indian Pines Data set 15%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9920	0.0001	0.9909	0.0003	0.9896	0.0003	510.0	1.2
20_20_60	0.9950	0.0001	0.9949	0.0003	0.9941	0.00043	667.4	1.4
30_20_50	0.9952	0.0008	0.9963	0.0002	0.9958	0.0002	834.3	1.4
40_20_40	0.9996	0.0002	0.9992	0.0003	0.9991	0.0003	997.1	1.6
50_20_30	0.9992	0.0001	0.9991	0.0003	0.999	0.0003	1149.5	1.6
60_20_20	0.9986	0.0006	0.9991	0.0002	0.99907	0.0002	1314.3	1.7
70_10_20	0.9994	0	0.9995	0	0.9994	1.11 E-16	1423.0	1.3
80_10_10	1	0	1	0	1	0	1601.1	1.4
90_5_5	1	0	1	0	1	0	1755.1	1.2

Table 6.16: Indian Pines Data set 25%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9942	0.0003	0.9922	0.0004	0.9912	0.0005	930.3	1.4
20_20_60	0.9948	6.31 E-05	0.9961	0.0003	0.9956	0.0003	1217.4	1.5
30_20_50	0.9951	0.0001	0.9970	0.0001	0.9966	0.0001	1510.9	1.6
40_20_40	0.99728	0.0007	0.9993	0.0003	0.9992	0.0003	1814.7	1.8
50_20_30	0.9994	0.0002	0.9992	0.0004	0.9991	0.0004	2096.2	1.9
60_20_20	0.9986	0.0012	0.9995	0	0.9994	3.77 E-09	2381.6	2.0
70_10_20	0.9994	0	0.9995	0	0.9994	1.11 E-16	2611.3	1.5
80_10_10	1	0	1	0	1	0	2921.7	1.5
90_5_5	1	0	1	0	1	0	3206.9	1.3

Table 6.17: Indian Pines Data set 50%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9950	0.0004	0.9935	0.0001	0.9926	0.0001	2151.1	1.7
20_20_60	0.9948	0.0004	0.9967	0.0003	0.9962	0.0004	2831.1	1.9
30_20_50	0.9945	0.0007	0.9972	0.00015	0.996	0.0001	3503.9	2.1
40_20_40	0.9981	0.0001	0.9991	0.0004	0.9989	0.0004	4198.1	2.4
50_20_30	0.9998	0.0002	0.9997	0.0003	0.9997	0.00034	4855.6	2.6
60_20_20	0.9977	0.0016	0.9995	0.0003	0.9994	0.0004	5539.4	2.9
70_10_20	0.9994	0	0.9995	0	0.9994	1.11 E-16	6078.3	1.9
80_10_10	1	0	1	0	1	0	6851.4	2.1
90_5_5	1	0	1	0	1	0	7536.6	1.5

Table 6.18: Indian Pines Data set 100%PCA Components

Split. Ratio Tr_Vl_Tt	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9977	0.0001	0.9980	5.68 E-05	0.9974	7.56 E-05	1372.3	1.9
20_20_60	0.9986	0.0003	0.9989	0.00016	0.9985	0.0002	1807.4	2.3
30_20_50	0.9989	5.11 E-05	0.9992	3.82 E-05	0.9989	5.06E-05	2262.8	2.6
40_20_40	0.9994	0.0001	0.9993	0.0001	0.9990	0.0002	2712.3	3.0
50_20_30	0.9992	0.0002	0.9994	0.0001	0.9992	0.0002	3147.3	3.3
60_20_20	0.9995	0.0001	0.9994	0.00029	0.99927	0.0003	3614.2	3.6
70_10_20	0.9996	0.0001	0.9995	0.0001	0.9994	0.0001	3866.3	2.3
80_10_10	0.9995	0.0002	0.9997	0	0.9996	3.48 E-09	4353.6	2.5
90_5_5	0.9994	0.0004	0.9996	0.0002	0.9995	0.0002	4735.4	1.7

Table 6.19: Pavia University Data set 15%PCA Components

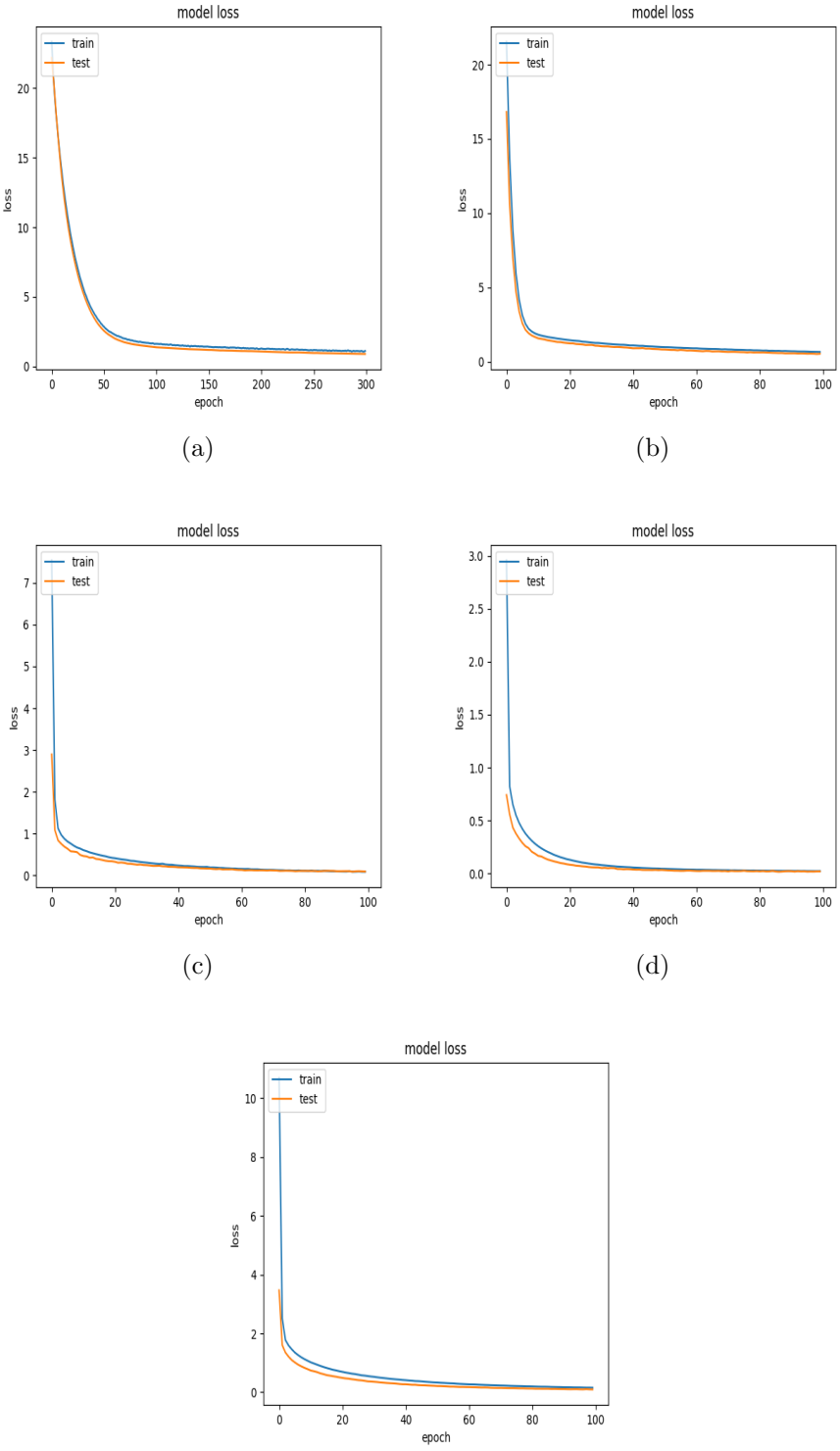


Figure 6.2: Loss Convergence Learning Curve of the (a)BOTSWANA (b)INDIAN PINES. (c)Pavia_University, (d)Pavia_Center, (e)Salinas Datasets

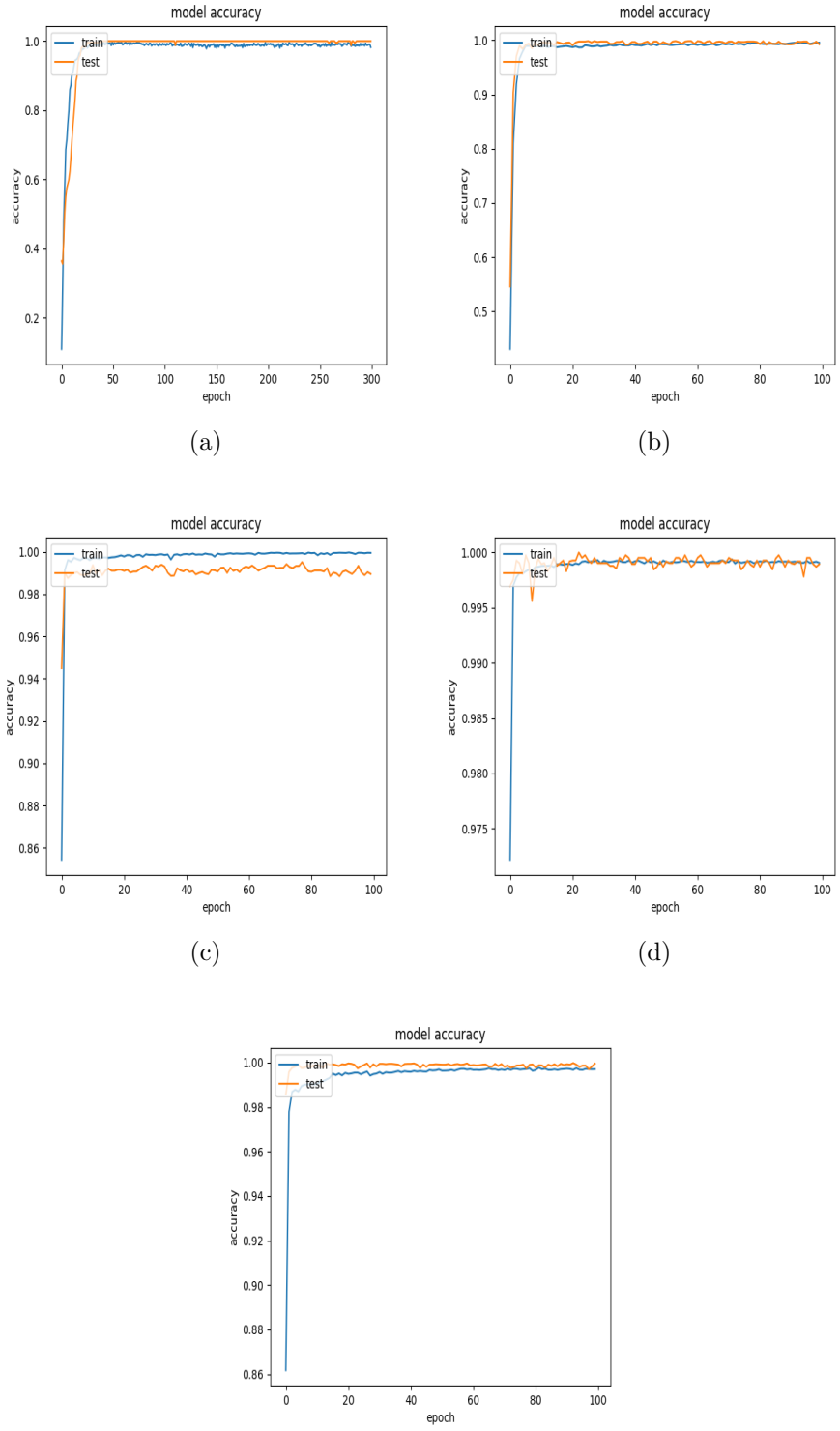
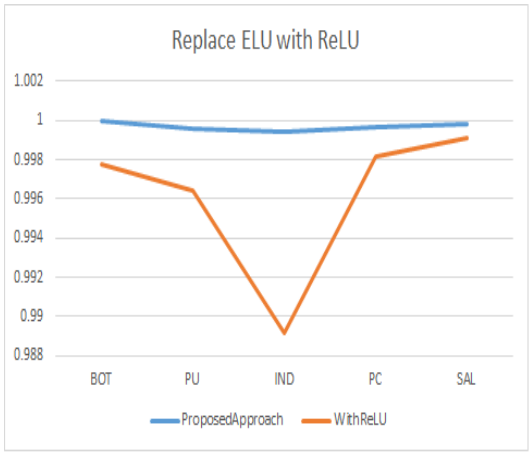
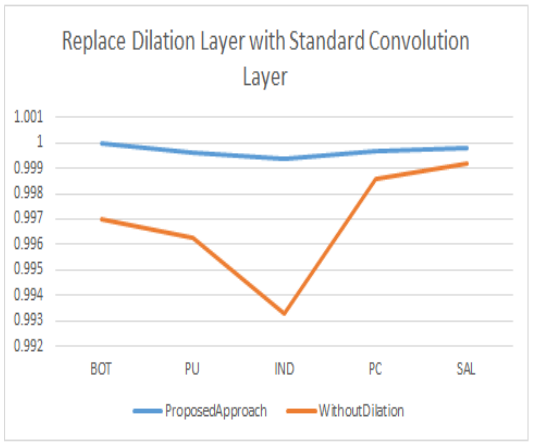


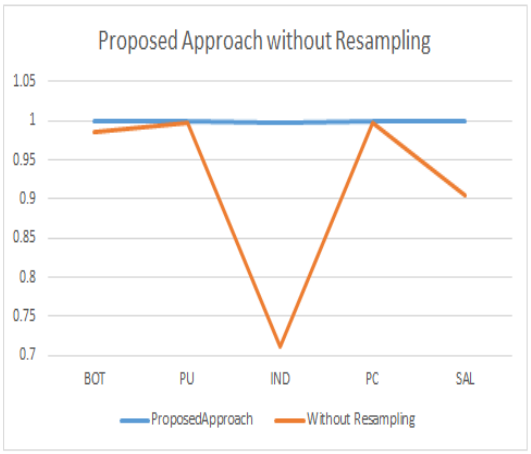
Figure 6.3: Training Accuracy Learning Curve Visualization of the (a)Botswana (b)Indian Pines. (c)Pavia University, (d)Pavia Center, (e)Salinas Datasets



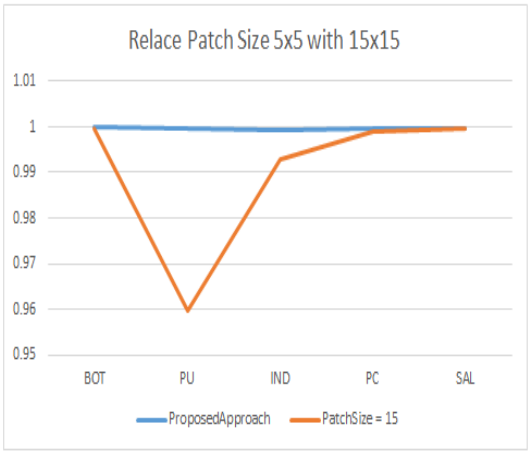
(a)



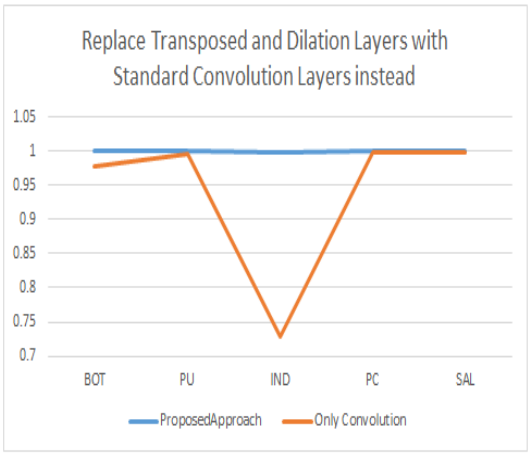
(b)



(c)



(d)



(e)

Figure 6.4: Stability of Proposed approach compared with other failure of other standard elements using for evaluation on 50 principal components from each of 5 datasets.

Split. Ratio Tr_Vl_Tt	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9973	0.0003	0.9978	4.73 E-04	0.9971	6.27E-04	1395.8	2.0
20_20_60	0.9986	0.0002	0.9989	0.0001	0.9986	0.0002	1874.2	2.3
30_20_50	0.9989	1.82 E-04	0.9990	3.31 E-04	0.9987	4.38E-04	2309.823	2.727
40_20_40	0.9992	0.0002	0.99	2.20 E-04	0.9991	2.92 E-04	2779.8	3.1
50_20_30	0.9994	9.55 E-05	0.9996	6.36 E-05	0.9994	8.43 E-05	3353.0	3.6
60_20_20	0.9993	3.10 E-04	0.9994	3.31 E-04	0.9992	4.38 E-04	4169.0	4.4
70_10_20	0.9993	0.0004	0.9994	2.75 E-04	0.9993	3.65 E-04	4513.9	2.8
80_10_10	0.9998	0.0001	0.9999	0.0001	0.9998	0.0001	5083.8	3.0
90_5_5	0.999	2.83 E-04	0.9998	2.20 E-04	0.9997	2.92 E-04	5546.6	2.0

Table 6.20: Pavia University Data set 25%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9963	0.0007	0.9967	0.0005	0.9956	0.0007	2299.4	2.6
20_20_60	0.9981	0.0001	0.9987	0.0001	0.9983	0.0002	3027.9	3.1
30_20_50	0.9985	0.0003	0.9989	0.0002	0.9985	0.0003	3804.5	3.5
40_20_40	0.9996	0.0001	0.9994	0.0001	0.9993	0.0002	4546.2	4.1
50_20_30	0.9996	0.0001	0.9996	9.71E-05	0.9995	0.0001	5267.8	4.7
60_20_20	0.9996	0.0001	0.9995	0.0001	0.9993	0.0002	5773.7	4.5
70_10_20	0.9995	0.0001	0.9995	0.0001	0.9994	0.0001	5955.1	2.6
80_10_10	1	0	1	0	1	0	6716.2	2.8
90_5_5	0.9996	0.0004	0.9996	0.0004	0.9995	0.0005	7380.1	1.9

Table 6.21: Pavia University Data set 50%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9960	0.0001	0.9966	0.0003	0.9955	0.0004	3948.1	2.8
20_20_60	0.9981	0.0002	0.9987	5.51 E-05	0.9983	7.30 E-05	5199.9	3.3
30_20_50	0.9987	0.0003	0.9989	0.0003	0.9986	0.0005	6479.7	3.8
40_20_40	0.9996	9.38 E-05	0.9994	0.0001	0.9992	0.0001	8157.1	5.3
50_20_30	0.9995	1.20 E-04	0.9996	9.72 E-05	0.9995	1.29 E-04	9546.8	6.0
60_20_20	0.9994	8.17 E-05	0.9994	5.51 E-05	0.9992	7.30 E-05	10815.9	6.7
70_10_20	0.9995	0.0002	0.9996	0.0002	0.9994	0.0003	11505.2	3.5
80_10_10	0.9997	3.80 E-05	0.9997	0	0.9996	1.43 E-09	12633.0	3.5
90_5_5	1	0	1	0	1	0	13954.9	2.3

Table 6.22: Pavia University Data set 100%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9982	9.40 E-05	0.9992	5.12 E-05	0.9989	7.25 E-05	8779.8	10.6
20_20_60	0.9990	0.0002	0.9996	6.25 E-05	0.9995	8.85 E-05	11851.9	13.7
30_20_50	0.9994	0.0001	0.9998	1.91 E-05	0.9997	2.70 E-05	14927.2	16.2
40_20_40	0.9992	0.0001	0.9997	2.87 E-05	0.9997	4.06 E-05	17764.6	18.7
50_20_30	0.9997	1.17 E-04	0.9999	2.12 E-05	0.9999	3.00 E-05	20791.0	21.5
60_20_20	0.9999	8.46 E-05	0.9999	1.59 E-05	0.9999	2.25 E-05	23700.0	24.7
70_10_20	0.9998	0.0001	0.9999	4.21 E-05	0.9998	5.96 E-05	25030.4	13.4
80_10_10	0.9998	0.0002	0.9999	6.36 E-05	0.9999	9.01E-05	28436.1	14.8
90_5_5	0.9994	0.0001	0.9998	0	0.9998	1.05 E-09	30401.9	8.9

Table 6.23: Pavia Center Data set 15%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9989	5.83 E-05	0.9995	2.84 E-05	0.9994	4.02 E-05	8246.3	10.7
20_20_60	0.9993	0.0001	0.9997	4.53 E-05	0.9996	6.42 E-05	11124.1	13.7
30_20_50	0.9995	0.0001	0.9999	3.37 E-05	0.9998	4.77 E-05	13597.4	13.6
40_20_40	0.9997	0.0001	0.9999	2.87 E-05	0.9998	4.06 E-05	17712.3	20.5
50_20_30	0.9996	1.03 E-04	0.9998	2.81 E-05	0.9998	3.97 E-05	21798.9	24.8
60_20_20	0.9999	6.11 E-06	0.9999	1.59 E-05	0.9999	2.25 E-05	25698.1	28.5
70_10_20	0.9999	6.11 E-06	0.9999	1.59 E-05	0.9999	2.25 E-05	26591.7	15.3
80_10_10	1	0	1	0	1	0	27870.6	15.9
90_5_5	1	0	1	0	1	0	29927.6	9.4

Table 6.24: Pavia Center Data set 25%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9988	0.0001	0.9995	6.88 E-05	0.9993	9.74 E-05	7558.6	5.65
20_20_60	0.9988	0.0003	0.9996	6.36 E-05	0.9995	9.01 E-05	9982.5	7.1
30_20_50	0.9997	0.0001	0.9999	5.44 E-05	0.9998	7.70 E-05	12643.0	8.6
40_20_40	0.9996	0.0001	0.9998	4.43 E-05	0.9998	6.27 E-05	15224.1	10.5
50_20_30	0.9997	6.37 E-05	0.9999	2.81 E-05	0.9999	3.97 E-05	17728.1	11.7
60_20_20	0.9999	6.11 E-06	0.9999	1.59 E-05	0.9999	2.25 E-05	20261.3	13.1
70_10_20	0.9999	6.11 E-06	0.9999	1.59 E-05	0.9999	2.25 E-05	21965.8	7.2
80_10_10	1	0	1	0	1	0	24969.1	7.9
90_5_5	0.9999	2.45 E-05	0.9999	6.36 E-05	0.9999	9.01 E-05	27841.4	6.8

Table 6.25: Pavia Center Data set 50%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9985	0.0002	0.9993	0.0001	0.9990	0.0001	14544.6	9.0
20_20_60	0.9995	6.02 E-05	0.99981	2.3 1E-05	0.9997	3.27 E-05	19371.3	11.704 8
30_20_50	0.9997	0.0001	0.9998	6.88 E-05	0.9998	9.75 E-05	24250.4	14.1
40_20_40	0.9997	9.40 E-05	0.9999	2.39 E-05	0.9999	3.38 E-05	29272.3	16.5
50_20_30	0.9999	8.79 E-05	0.9999	3.18 E-05	0.9999	4.51E-05	33803.3	18.3
60_20_20	0.99998	6.11E-06	0.99995	1.59 E-05	0.99993	2.25 E-05	37146.4	18.5
70_10_20	0.99993	7.92 E-05	0.99996	2.76 E-05	0.99995	3.90 E-05	40238.7	9.5
80_10_10	0.99999	1.22 E-05	0.99997	3.18 E-05	0.99996	4.51 E-05	45282.4	11.0
90_5_5	1	0	1	0	1	0	49867.4	6.1

Table 6.26: Pavia Center Data set 100%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9993	4.06 E-05	0.9983	5.42 E-05	0.9981	6.04 E-05	1707.2	2.1
20_20_60	0.9997	9.03 E-05	0.9993	0.0002	0.9992	0.0002	2237.8	2.6
30_20_50	0.9998	7.66 E-05	0.9996	0.0001	0.9995	0.0001	2784.4	2.9
40_20_40	0.9997	0.0001	0.9995	0.0001	0.9995	0.0002	3332.7	3.3
50_20_30	0.9999	1.64 E-05	0.99991	2.90 E-05	0.9999	3.23 E-05	3921.8	3.7
60_20_20	0.9998	8.63 E-05	0.9999	7.54 -05	0.9998	8.40 E-05	4450.4	4.1
70_10_20	0.99999	1.31 E-05	0.99996	4.35 E-05	0.99996	4.85 E-05	4858.6	2.6
80_10_10	1	0	1	0	1	0	5401.8	2.9
90_5_5	1	0	1	0	1	0	5930.2	1.9

Table 6.27: Salinas Dataset 15%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9994	4.83 E-05	0.9985	0.0001	0.9983	0.0001	2278.1	2.4
20_20_60	0.9997	3.71 E-05	0.9994	0.0001	0.9994	0.0002	3025.5	2.9
30_20_50	0.9998	3.51 E-05	0.9997	9.05 E-05	0.9996	0.0001	3743.9	3.2
40_20_40	0.9998	4.29 E-05	0.9996	7.54 E-05	0.9996	8.40 E-05	4546.1	3.7
50_20_30	0.9998	6.04 E-05	0.9998	8.71 E-05	0.9997	9.70 E-05	5280.5	4.2
60_20_20	0.99999	1.31 E-05	0.99996	4.35 E-05	0.99996	4.85 E-05	5985.4	4.4
70_10_20	0.99997	1.78 E-05	0.9999	4.35 E-05	0.99993	4.85 E-05	6476.6	2.8
80_10_10	0.99997	4.05 E-05	0.99993	8.71 E-05	0.99993	9.70 E-05	7373.8	3.0
90_5_5	0.9998	0.0001	0.9998	0.0001	0.9998	0.0001	8017.1	2.1

Table 6.28: Salinas Dataset 25%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9989	0.0005	0.9970	0.0018	0.9967	0.0020	4143.7	3.0
20_20_60	0.9997	5.93 E-05	0.9992	0.0002	0.9992	0.0002	5488.7	3.4
30_20_50	0.9998	8.18 E-06	0.9997	0.0001	0.9997	0.0001	6870.1	4.2
40_20_40	0.9998	2.68 E-05	0.9998	7.85 E-05	0.9998	8.74 E-05	8217.2	4.7
50_20_30	0.9998	1.24 E-04	0.9999	7.68 E-05	0.9999	8.55 E-05	9570.1	5.4
60_20_20	1	0	1	0	1	0	10941.8	6.1
70_10_20	1	0	1	0	1	0	11878.7	3.4
80_10_10	0.99997	4.05 E-05	0.99993	8.71 E-05	0.99993	9.70 E-05	13469.2	3.9
90_5_5	1	0	1	0	1	0	14811.4	2.5

Table 6.29: Salinas Dataset 50%PCA Components

Split. Ratio Tr_Vl_Ts	**AA**	**AA_Std.** (±)	**oA**	**OA_Std.** (±)	**K**	**K_Std.** (±)	**Train Time**	**Test Time**
10_20_70	0.9992	9.28 E-06	0.9979	0.0001	0.9977	0.0001	9751.8	4.46
20_20_60	0.9996	0.0001	0.9991	0.0003	0.9990	0.0004	12867.3	5.60
30_20_50	0.9996	0.0001	0.9993	0.0004	0.9992	0.0005	16267.5	7.07
40_20_40	0.9998	2.29 E-05	0.9997	7.54 E-05	0.9997	8.40 E-05	19656.8	9.33
50_20_30	0.9998	4.40 E-05	0.9998	5.03 E-05	0.9997	5.60 E-05	22882.5	10.07
60_20_20	0.99996	4.64 E-05	0.9999	0.0001	0.9998	0.0001	26150.7	12.68
70_10_20	0.9999	1.31 E-05	0.9999	4.35 E-05	0.9999	4.85 E-05	28165.1	5.64
80_10_10	1	0	1	0	1	0	34465.4	8.22
90_5_5	1	0	1	0	1	0	41229.6	7.54

Table 6.30: Salinas Dataset 100%PCA Components

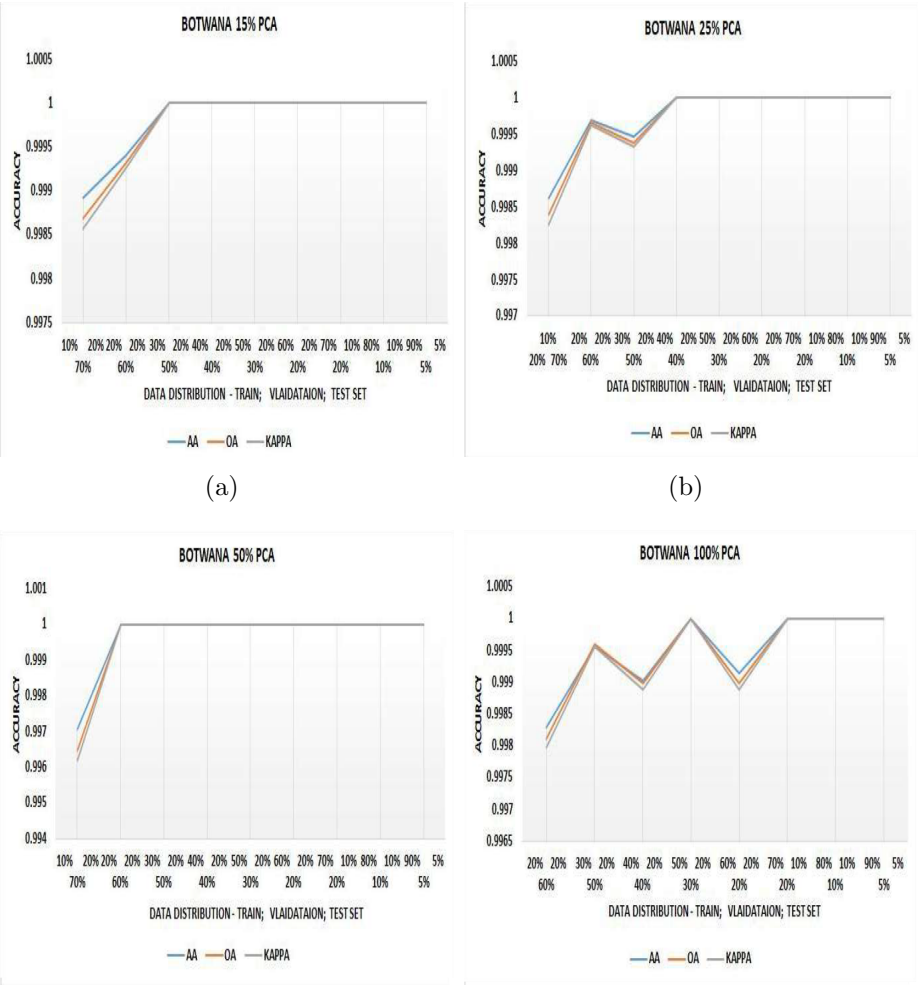


Figure 6.5: Stability of Proposed Model on BOTSWANA Dataset With Respect to PCA and Number of Samples Used For Training, Validation and Testing

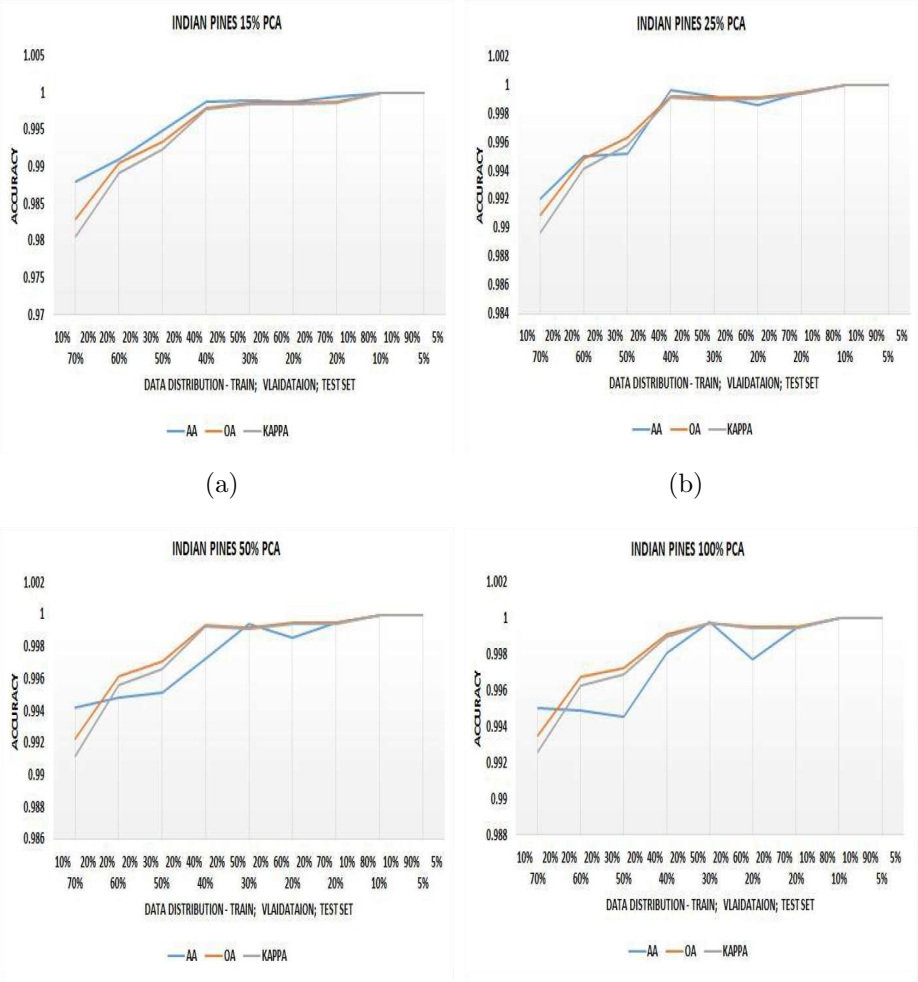


Figure 6.6: Stability of Proposed Model on INDIAN PINES Dataset With Respect to PCA and Number of Samples Used For Training, Validation and Testing

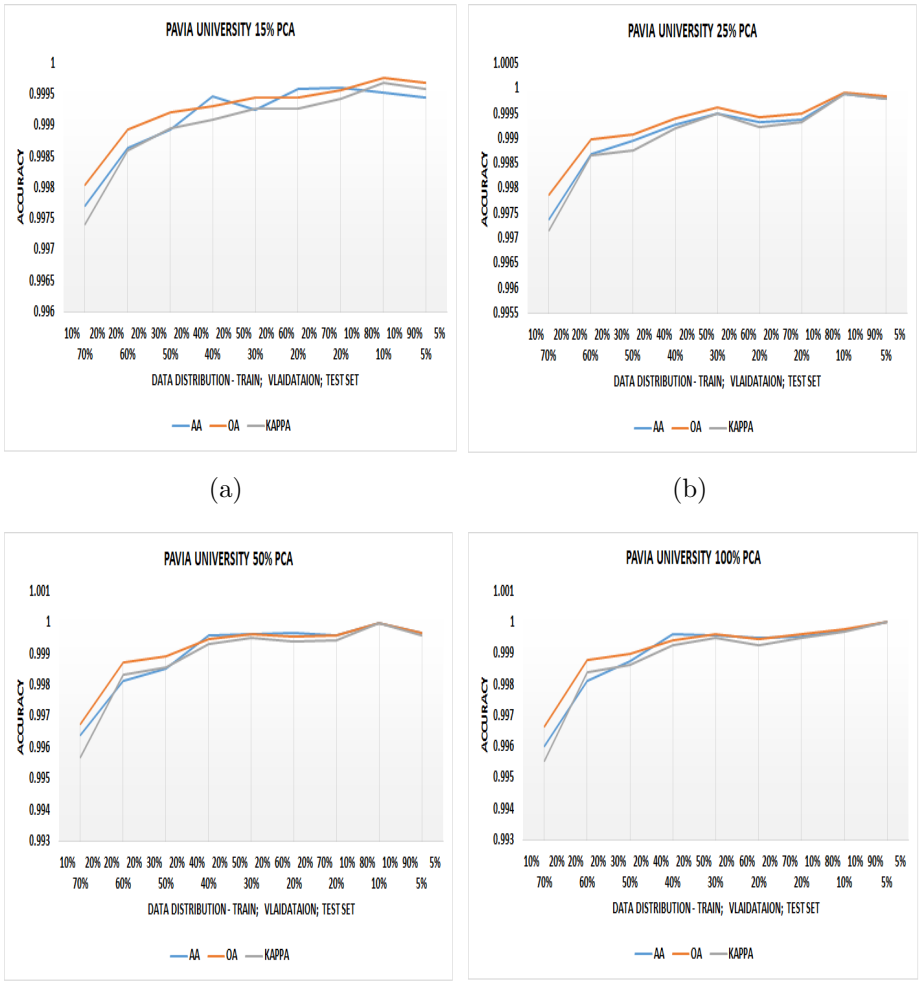
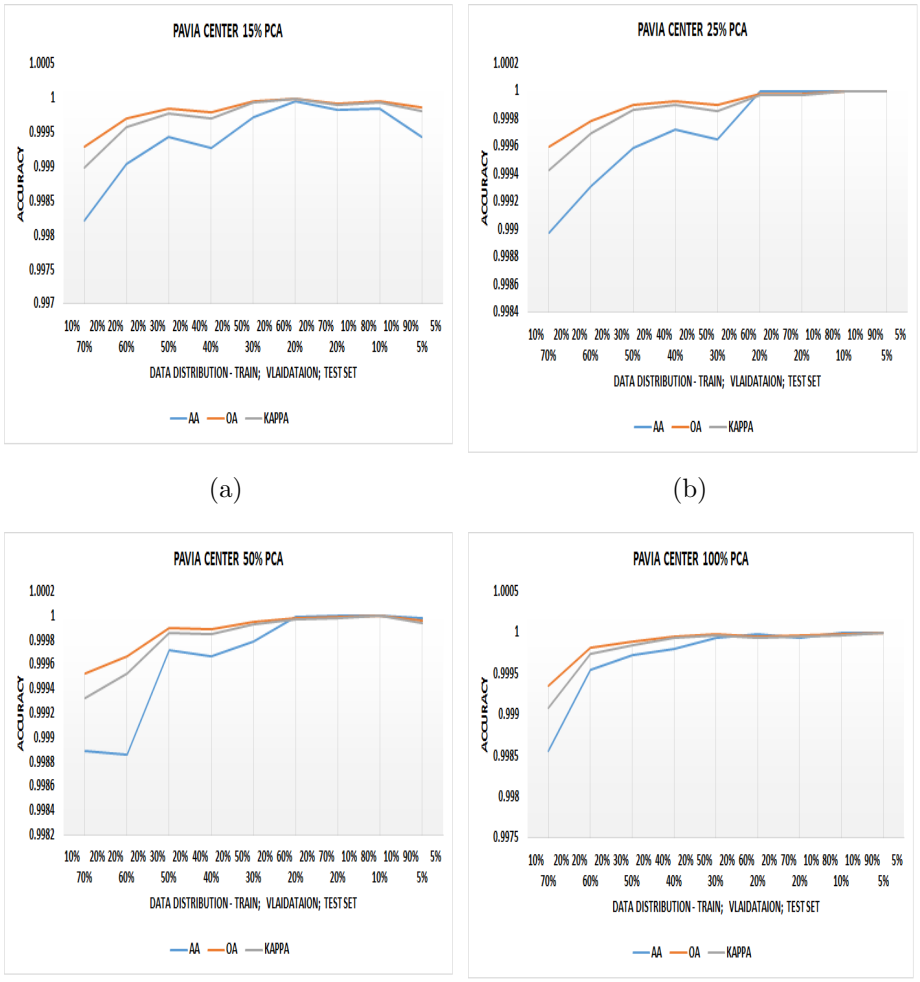


Figure 6.7: Stability of Proposed Model on PAVIA UNIVERSITY Dataset With Respect to PCA and Number of Samples Used For Training, Validation and Testing



(a) (b)

(c) (d)

Figure 6.8: Stability of Proposed Model on PAVIA CENTER Dataset With Respect to PCA and Number of Samples Used For Training, Validation and Testing

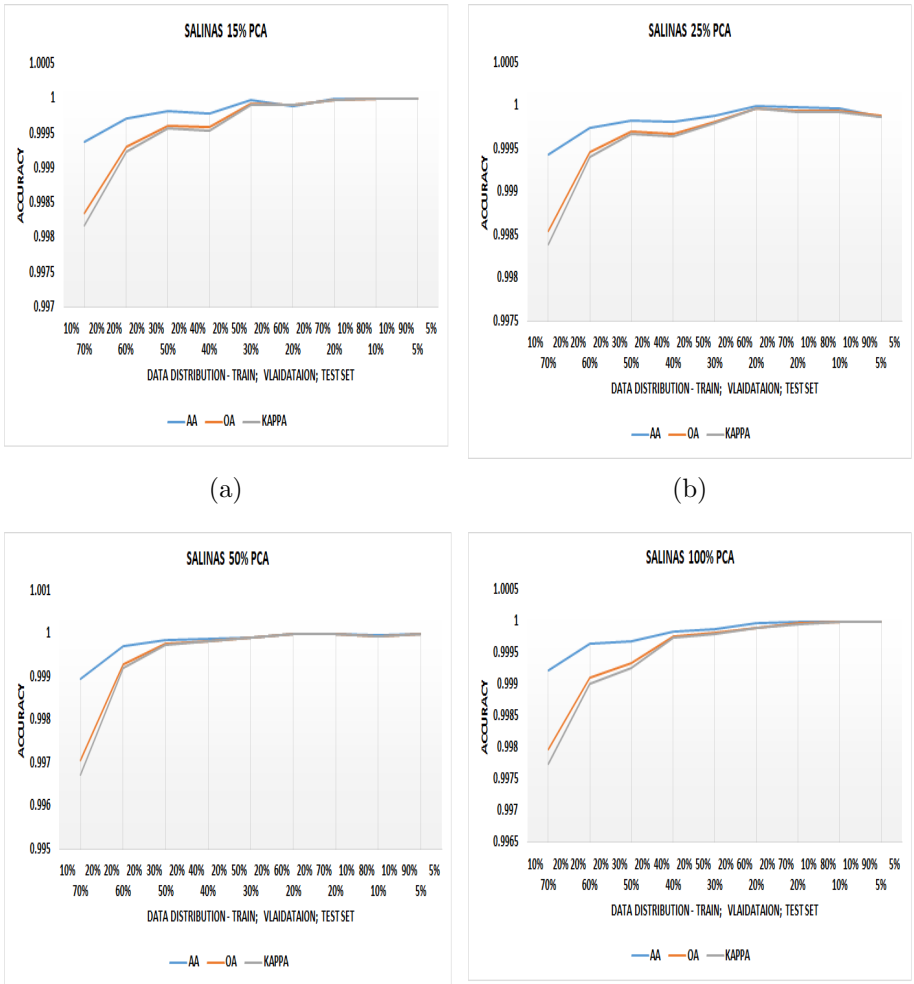


Figure 6.9: Stability of Proposed Model on SALINAS Dataset With Respect to PCA and Number of Samples Used For Training, Validation and Testing

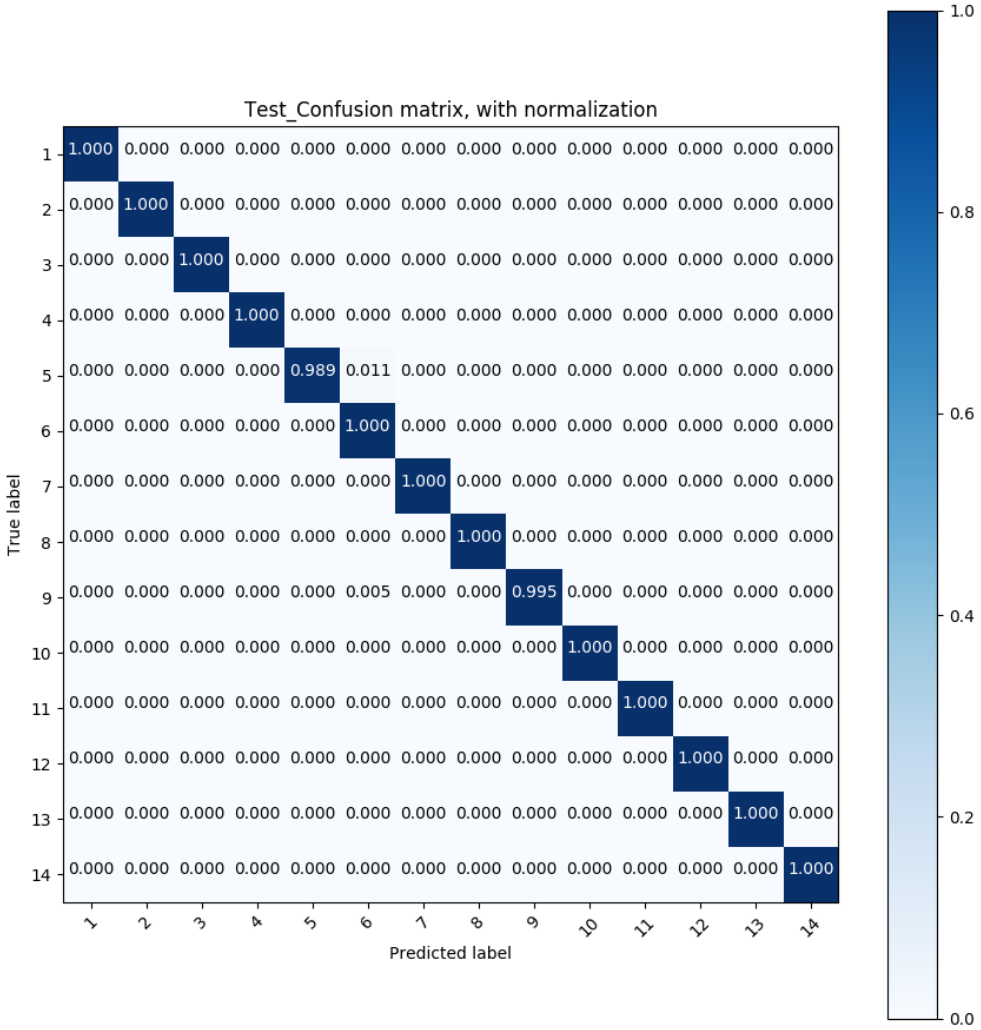


Figure 6.10: Normalize Confusion Matrix of BOTSWANA Dataset of 10% Training Samples and 90% Testing Samples

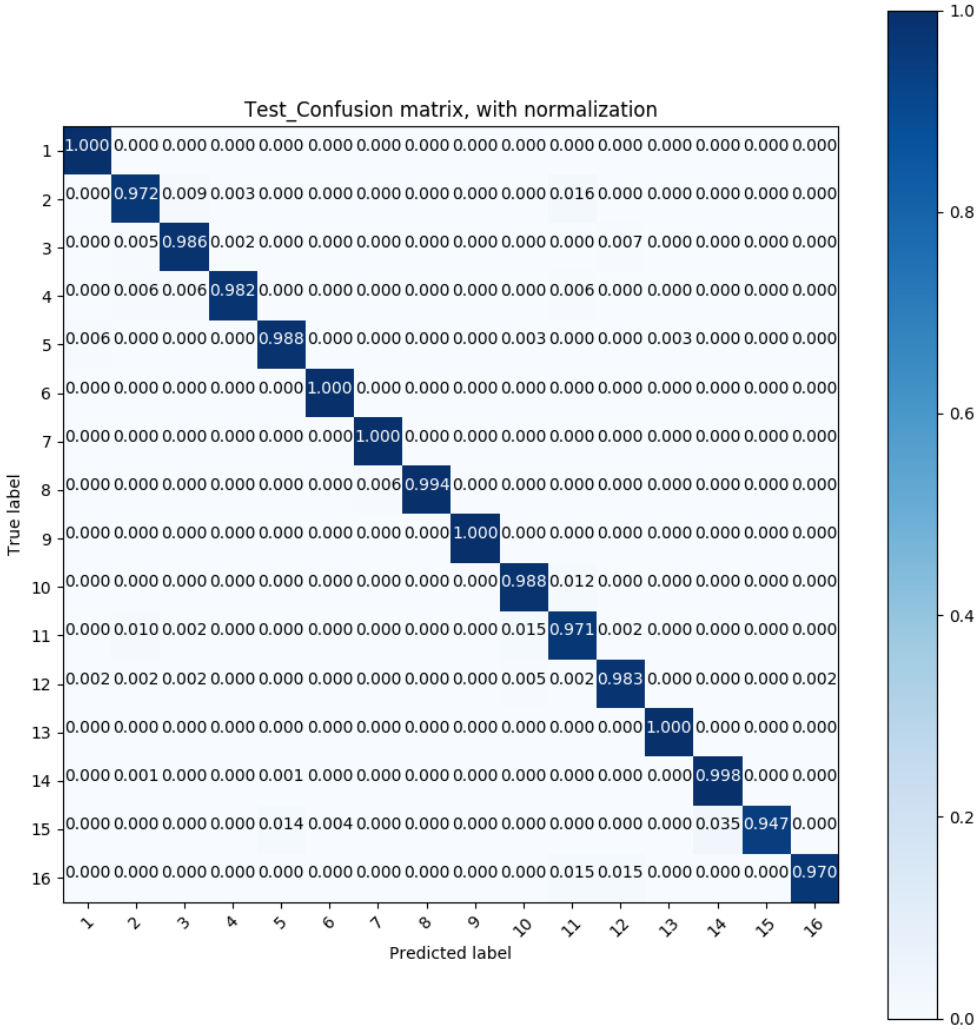


Figure 6.11: Normalize Confusion Matrix of INDIAN PINES Dataset of 10% Training Samples and 90% Testing Samples

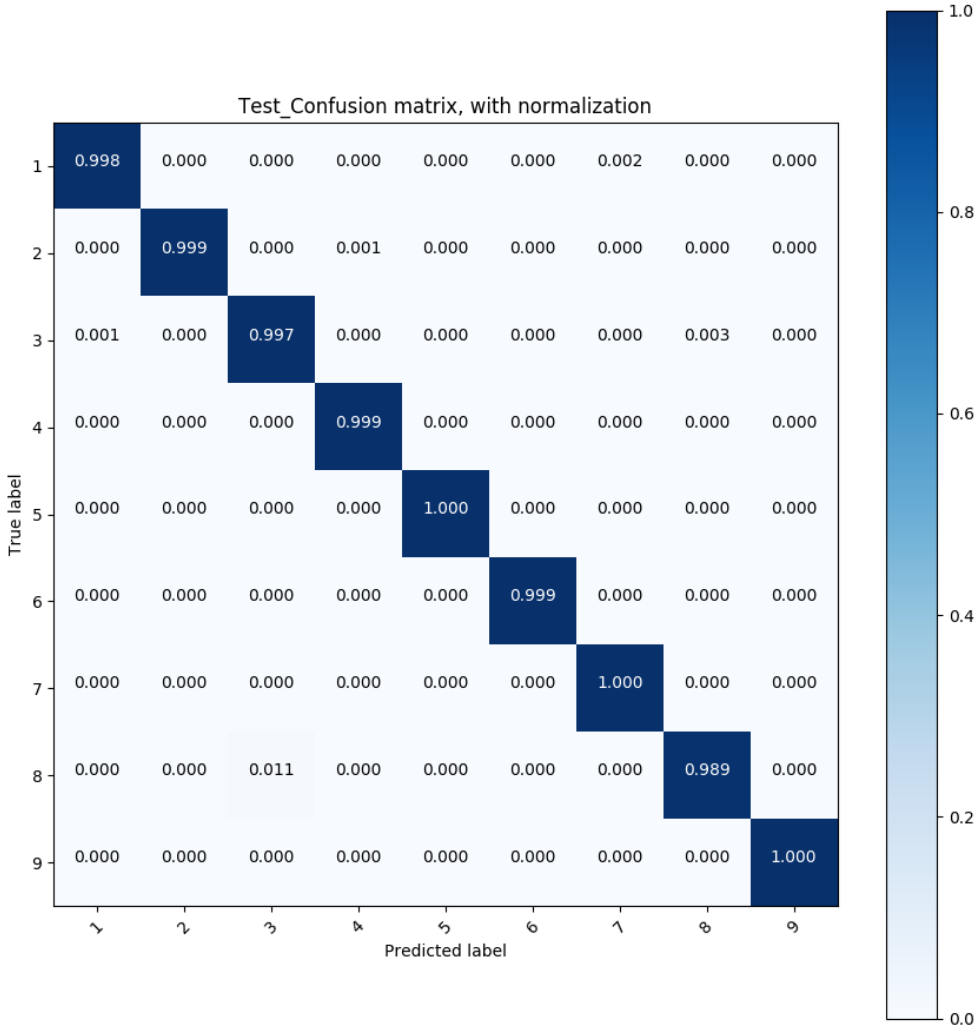


Figure 6.12: Normalize Confusion Matrix of PAVIA UNIVERSITY Dataset of 10% Training Samples and 90% Testing Samples

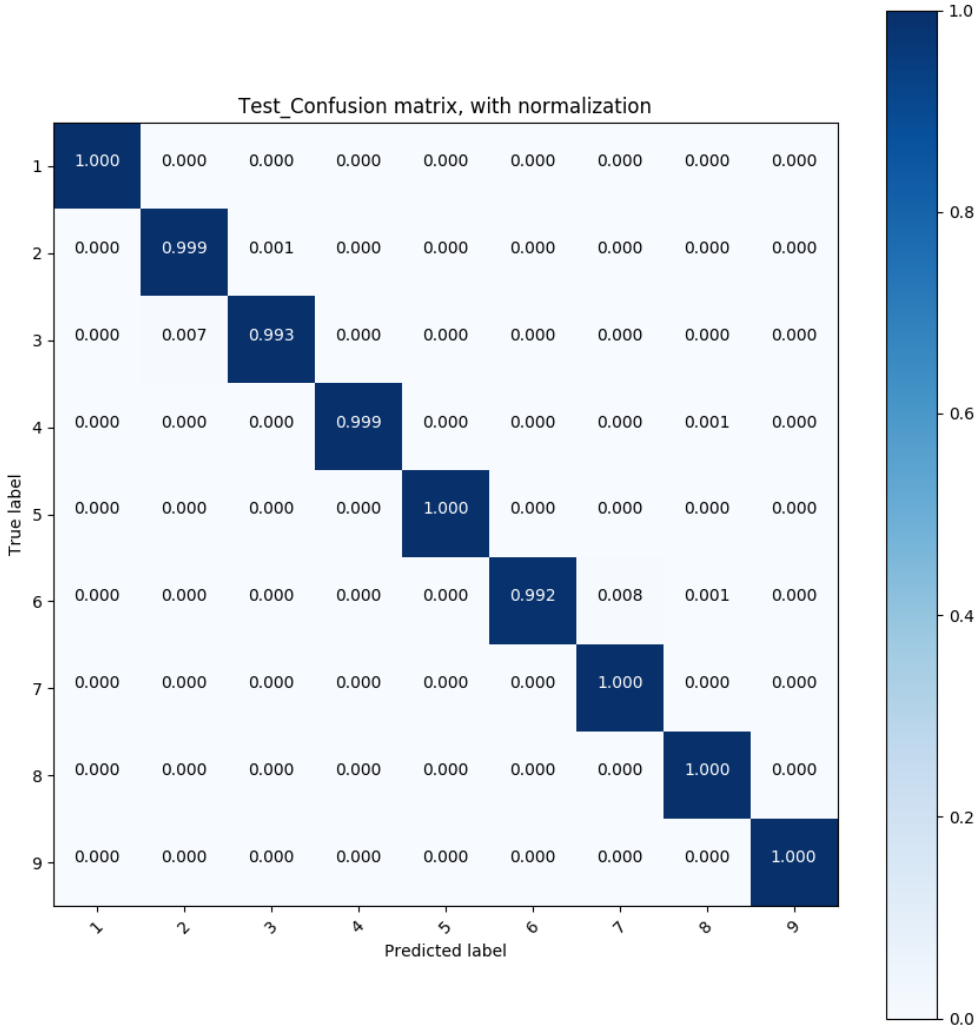


Figure 6.13: Normalize Confusion Matrix of PAVIA CENTER Dataset of 10% Training Samples and 90% Testing Samples

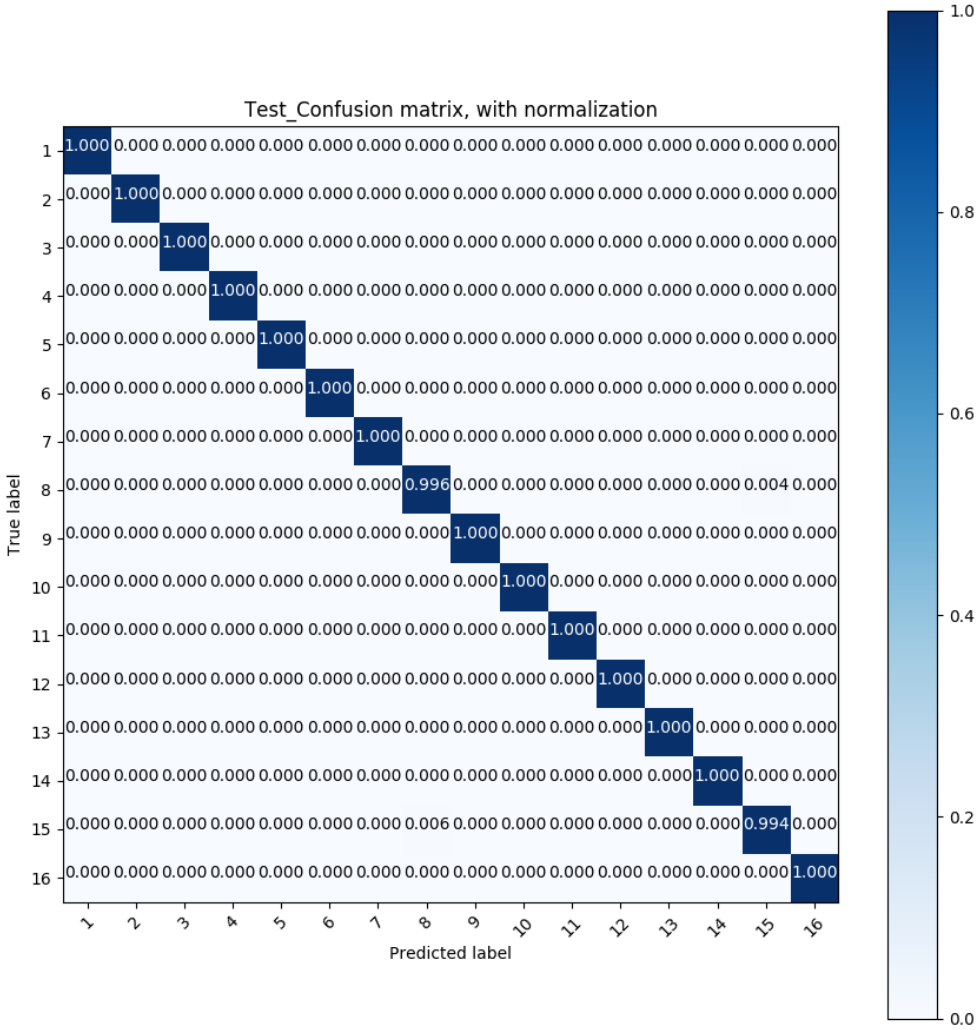


Figure 6.14: Normalized Confusion Matrix of SALINAS Dataset of 10% Training Samples and 90% Testing Samples

Chapter 7

Published Paper

NOTE : I presented here title of submitted and accepted paper with the ICIAP-2019 20^{th} International Conference on Image Analysis and Processing. All concepts, details and results are included in the previous chapters.

[1]. Devaram, R.R.; Allegra, D.; Gallo, G.; Stanco, F. Hyperspectral image classification via convolutional neuralnetwork based on dilation layers. In Proceedings of the International Conference on Image Analysis andProcessing (ICIAP), Trento, Italy, 9–13 September 2019; pp. 378–387; DOI: 10.1007/978-3-030-30642-7_34.

Chapter 8

Conclusion

In this work we explore the problem of pixel-wise HSI classification for data acquired from various sensors. The propose a framework which includes an oversampling stage to make benchmark datasets balanced, and a new CNN architecture based on Dilated and Transposed Convolutions makes model more robust for various hyperspectral data. Despite the already high performance achieved by state-of-art works, our method outperforms all of them and achieves the best performance for each benchmark dataset. Based on this research, in future works, we are strongly recommends that the feature extraction stage is significantly important. However, while extraction of features from CNN, includes transposed Convolution or Dilation Convolution along with Exponential Linear Unit in Initial layers of the network may extract valuable features without loss of information. Finally, we conclude this thesis with, considering to use our architecture for similar problems, like pixel-wise semantic segmentation on different domains gives better performance.

Bibliography

[1] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* 2.4 (Dec. 1989), pp. 303–314. DOI: 10.1007/bf02551274. URL: https://doi.org/10.1007/bf02551274.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[3] A CINGOLANI. "Mapping vegetation in a heterogeneous mountain rangeland using landsat data: an alternative method to define and classify landcover units". In: *Remote Sensing of Environment* 92.1 (July 2004), pp. 84–97. DOI: 10.1016/j.rse.2004.05.008. URL: https://doi.org/10.1016/j.rse.2004.05.008.

[4] J. R. Jensen and K. Lulla. "Introductory digital image processing: A remote sensing perspective". In: *Geocarto International* 2.1 (Mar. 1987), pp. 65–65. DOI: 10.1080/10106048709354084. URL: https://doi.org/10.1080/10106048709354084.

[5] L Hubert-Moy. "A Comparison of Parametric Classification Procedures of Remotely Sensed Data Applied on Different Landscape Units". In: *Remote Sensing of Environment* 75.2 (Feb. 2001), pp. 174–187. DOI: 10.1016/s0034-4257(00)00165-6. URL: https://doi.org/10.1016/s0034-4257(00)00165-6.

[6] D. Chen and D. Stow. "The Effect of Training Strategies on Supervised Classification at Different Spatial Resolutions". In: *Photogrammetric Engineering and Remote Sensing* 68 (Nov. 2002).

[7] M. A. Lefsky and W. B. Cohen. "Selection of Remotely Sensed Data". In: *Remote Sensing of Forest Environments*. Springer US, 2003, pp. 13–46. DOI:

10.1007/978-1-4615-0306-4_2. URL: https://doi.org/10.1007/978-1-4615-0306-4_2.

[8] M BARNSLEY. "Digital remotely-sensed data and their characteristics". In: *New Developments in Geographical Information Systems: Principles, Techniques, Management and Applications*. Wiley, 2005, pp. 451–466.

[9] S. R PHINN. "A framework for selecting appropriate remotely sensed data dimensions for environmental monitoring and management". In: *International Journal of Remote Sensing* 19.17 (Nov. 1998), pp. 3457–3463. DOI: 10.1080/014311698214136. URL: https://doi.org/10.1080/014311698214136.

[10] S. R. Phinn, C. Menges, G. J. Hill, and M. Stanford. "Optimizing Remotely Sensed Solutions for Monitoring, Modeling, and Managing Coastal Environments". In: *Remote Sensing of Environment* 73.2 (Aug. 2000), pp. 117–132. DOI: 10.1016/s0034-4257(00)00087-0. URL: https://doi.org/10.1016/s0034-4257(00)00087-0.

[11] "Scale in Remote Sensing and GIS (Mapping Sciences)". In: (1997). URL: https://www.amazon.com/Scale-Remote-Sensing-Mapping-Sciences/dp/156670104X?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=156670104X.

[12] T. Toutin. "Review article: Geometric processing of remote sensing images: models, algorithms and methods". In: *International Journal of Remote Sensing* 25.10 (May 2004), pp. 1893–1924. DOI: 10.1080/0143116031000101611. URL: https://doi.org/10.1080/0143116031000101611.

[13] C. Song, C. E. Woodcock, K. C. Seto, M. P. Lenney, and S. A. Macomber. "Classification and Change Detection Using Landsat TM Data". In: *Remote Sensing of Environment* 75.2 (Feb. 2001), pp. 230–244. DOI: 10.1016/s0034-4257(00)00169-3. URL: https://doi.org/10.1016/s0034-4257(00)00169-3.

[14] B. L. MARKHAM and J. L. BARKER. "Thematic Mapper bandpass solar exoatmospheric irradiances". In: *International Journal of Remote Sensing* 8.3 (Mar. 1987), pp. 517–523. DOI: 10.1080/01431168708948658. URL: https://doi.org/10.1080/01431168708948658.

[15] M. A. GILABERT, C. CONESE, and F. MASELLI. "An atmospheric correction method for the automatic retrieval of surface reflectances from TM images". In: *International Journal of Remote Sensing* 15.10 (July 1994), pp. 2065–2086. DOI: 10.1080/01431169408954228. URL: https://doi.org/10.1080/01431169408954228.

[16] D. G. Hadjimitsis, C. R. I. Clayton, and V. S. Hope. "An assessment of the effectiveness of atmospheric correction algorithms through the remote sensing of some reservoirs". In: *International Journal of Remote Sensing* 25.18 (Sept. 2004), pp. 3651–3674. DOI: 10.1080/01431160310001647993. URL: https://doi.org/10.1080/01431160310001647993.

[17] P. Meyer, K. I. Itten, T. Kellenberger, S. Sandmeier, and R. Sandmeier. "Radiometric corrections of topographically induced effects on Landsat TM data in an alpine environment". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 48.4 (Aug. 1993), pp. 17–28. DOI: 10.1016/0924-2716(93)90028-l. URL: https://doi.org/10.1016/0924-2716(93)90028-l.

[18] J. C. Price. "Comparing MODIS and ETM data for regional and global land classification". In: *Remote Sensing of Environment* 86.4 (Aug. 2003), pp. 491–499. DOI: 10.1016/s0034-4257(03)00127-5. URL: https://doi.org/10.1016/s0034-4257(03)00127-5.

[19] G. Hughes. "On the mean accuracy of statistical pattern recognizers". In: *IEEE Transactions on Information Theory* 14.1 (Jan. 1968), pp. 55–63. DOI: 10.1109/tit.1968.1054102. URL: https://doi.org/10.1109/tit.1968.1054102.

[20] T. Rashed, J. R. Weeks, M. S. Gadalla, and A. G. Hill. "Revealing the Anatomy of Cities through Spectral Mixture Analysis of Multispectral Satellite Imagery: A Case Study of the Greater Cairo Region, Egypt." In: *Geocarto International* 16.4 (Dec. 2001), pp. 7–18. DOI: 10.1080/10106040108542210. URL: https://doi.org/10.1080/10106040108542210.

[21] G. P. Asner and K. B. Heidebrecht. "Spectral unmixing of vegetation, soil and dry carbon cover in arid regions: Comparing multispectral and hyperspectral observations". In: *International Journal of Remote Sensing* 23.19 (Jan. 2002),

pp. 3939–3958. DOI: 10.1080/01431160110115960. URL: https://doi.org/10.1080/01431160110115960.

[22] L. Biehl and D. Landgrebe. "MultiSpec—a tool for multispectral–hyperspectral image data analysis". In: *Computers & Geosciences* 28.10 (Dec. 2002), pp. 1153–1159. DOI: 10.1016/s0098-3004(02)00033-x. URL: https://doi.org/10.1016/s0098-3004(02)00033-x.

[23] M. A. Penaloza and R. M. Welch. "Feature selection for classification of polar regions using a fuzzy expert system". In: *Remote Sensing of Environment* 58.1 (Oct. 1996), pp. 81–100. DOI: 10.1016/0034-4257(95)00260-x. URL: https://doi.org/10.1016/0034-4257(95)00260-x.

[24] D. R. Peddle and D. T. Ferguson. "Optimisation of multisource data analysis: an example using evidential reasoning for GIS data classification". In: *Computers & Geosciences* 28.1 (Feb. 2002), pp. 45–52. DOI: 10.1016/s0098-3004(01)00012-7. URL: https://doi.org/10.1016/s0098-3004(01)00012-7.

[25] J. Cihlar, Q. Xiao, J. Chen, J. Beaubien, K. Fung, and R. Latifovic. "Classification by progressive generalization: A new automated methodology for remote sensing multichannel data". In: *International Journal of Remote Sensing* 19.14 (Jan. 1998), pp. 2685–2704. DOI: 10.1080/014311698214451. URL: https://doi.org/10.1080/014311698214451.

[26] R DeFries. "Multiple Criteria for Evaluating Machine Learning Algorithms for Land Cover Classification from Satellite Data". In: *Remote Sensing of Environment* 74.3 (Dec. 2000), pp. 503–515. DOI: 10.1016/s0034-4257(00)00142-5. URL: https://doi.org/10.1016/s0034-4257(00)00142-5.

[27] R Lawrence. "Classification of remotely sensed imagery using stochastic gradient boosting as a refinement of classification tree analysis". In: *Remote Sensing of Environment* 90.3 (Apr. 2004), pp. 331–336. DOI: 10.1016/j.rse.2004.01.007. URL: https://doi.org/10.1016/j.rse.2004.01.007.

[28] K. CONGALTON R.G.and GREEN. "A practical look at the sources of confusion in error matrix generation". In: *PEandRS, Photogrammetric Engineering and Remote Sensing* 59.5 (1993), pp. 641–644. URL: https://www.

asprs.org/wp-content/uploads/pers/1993journal/may/1993_may_641-644.pdf.

[29] M. Kalkhan, R. Reich, and R. Czaplewski. "Variance Estimates and Confidence Intervals for the Kappa Measure of Classification Accuracy". In: *Canadian Journal of Remote Sensing* 23.3 (Sept. 1997), pp. 210–216. DOI: 10.1080/07038992.1997.10855203. URL: https://doi.org/10.1080/07038992.1997.10855203.

[30] R. G. Congalton. "A review of assessing the accuracy of classifications of remotely sensed data". In: *Remote Sensing of Environment* 37.1 (July 1991), pp. 35–46. DOI: 10.1016/0034-4257(91)90048-b. URL: https://doi.org/10.1016/0034-4257(91)90048-b.

[31] S. Muller, D. Walker, F. Nelson, N. Auerbach, J. Bockheim, S. Guyer, and D. Sherba. "Accuracy assessment of a land-cover map of the Kuparuk River Basin, Alaska: Considerations for remote regions". In: *Photogrammetric Engineering and Remote Sensing* 64 (June 1998), pp. 619–628.

[32] S. V. Stehman and R. L. Czaplewski. "Design and Analysis for Thematic Map Accuracy Assessment". In: *Remote Sensing of Environment* 64.3 (June 1998), pp. 331–344. DOI: 10.1016/s0034-4257(98)00010-8. URL: https://doi.org/10.1016/s0034-4257(98)00010-8.

[33] R. Congalton and K. Green. *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*. Aug. 2019. ISBN: 9780429052729. DOI: 10.1201/9780429052729.

[34] G. M. Foody. "Status of land cover classification accuracy assessment". In: *Remote Sensing of Environment* 80.1 (Apr. 2002), pp. 185–201. DOI: 10.1016/s0034-4257(01)00295-4. URL: https://doi.org/10.1016/s0034-4257(01)00295-4.

[35] W. D. Hudson and C. W. Ramm. "Correct Formation of the Kappa Coefficient of Agreement". In: 1987.

[36] R. Cangalton. *A Quantitative Method to Test for Consistency and Correctness in Photointerpretation*. 1983. URL: https://books.google.it/books?id=vb1DNQAACAAJ.

[37] L. Janssen and F. Vanderwel. "Accuracy assessment of satellite derived land-cover data: A review". In: *Photogrammetric Engineering and Remote Sensing; (United States)* (). ISSN: 0099-1112.

[38] S. V. Stehman. "Estimating the Kappa Coefficient and its Variance under Stratified Random Sampling". In: 1996.

[39] S. V. Stehman. "Selecting and interpreting measures of thematic classification accuracy". In: *Remote Sensing of Environment* 62.1 (Oct. 1997), pp. 77–89. DOI: 10.1016/s0034-4257(97)00083-7. URL: https://doi.org/10.1016/s0034-4257(97)00083-7.

[40] P. C. Smits, S. G. Dellepiane, and R. A. Schowengerdt. "Quality assessment of image classification algorithms for land-cover mapping: A review and a proposal for a cost-based approach". In: *International Journal of Remote Sensing* 20.8 (Jan. 1999), pp. 1461–1486. DOI: 10.1080/014311699212560. URL: https://doi.org/10.1080/014311699212560.

[41] G. M. Foody. "Thematic map comparison: evaluating the statistical significance of differences in classification accuracy". In: *Photogrammetric Engineering & Remote Sensing* 70.5 (May 2004), pp. 627–633. DOI: 10.14358/pers.70.5.627. URL: https://doi.org/10.14358/pers.70.5.627.

[42] G. M. Foody. "On the compensation for chance agreement in image classification accuracy assessment". In: 1992.

[43] Z. Ma and R. L. Redmond. "Tau coefficients for accuracy assessment of classification of remote sensing data". In: 1995.

[44] P. Hardin and J. M. Shumway. "Statistical significance and normalized confusion matrices". In: *Photogrammetric Engineering and Remote Sensing* 63 (Jan. 1997).

[45] S. V. Stehman. "A Critical Evaluation of the Normalized Error Matrix in Map Accuracy Assessment". In: *Photogrammetric Engineering & Remote Sensing* 70.6 (June 2004), pp. 743–751. DOI: 10.14358/pers.70.6.743. URL: https://doi.org/10.14358/pers.70.6.743.

[46] J. T. Finn. "Use of the average mutual information index in evaluating classification error and consistency". In: *International journal of geographical information systems* 7.4 (July 1993), pp. 349–366. DOI: 10.1080/02693799308901966. URL: https://doi.org/10.1080/02693799308901966.

[47] S. Gopal and C. Woodcock. "Theory and methods for accuracy assessment of thematic maps using fuzzy sets". In: *Photogrammetric Engineering and Remote Sensing; (United States)* 60,3 (Feb. 1994). ISSN: 0099-1112.

[48] C. E. Woodcock and S. Gopal. "Fuzzy set theory and thematic maps: accuracy assessment and area estimation". In: *International Journal of Geographical Information Science* 14.2 (Mar. 2000), pp. 153–172. DOI: 10.1080/136588100240895. URL: https://doi.org/10.1080/136588100240895.

[49] E. Binaghi, P. A. Brivio, P. Ghezzi, and A. Rampini. "A fuzzy set-based accuracy assessment of soft classification". In: *Pattern Recognition Letters* 20.9 (Sept. 1999), pp. 935–948. DOI: 10.1016/s0167-8655(99)00061-6. URL: https://doi.org/10.1016/s0167-8655(99)00061-6.

[50] C. Ricotta. "Evaluating the classification accuracy of fuzzy thematic maps with a simple parametric measure". In: *International Journal of Remote Sensing* 25.11 (June 2004), pp. 2169–2176. DOI: 10.1080/01431160310001618130. URL: https://doi.org/10.1080/01431160310001618130.

[51] W. Liu, S. Gopal, and C. E. Woodcock. "Uncertainty and Confidence in Land Cover Classification Using a Hybrid Classifier Approach". In: *Photogrammetric Engineering & Remote Sensing* 70.8 (Aug. 2004), pp. 963–971. DOI: 10.14358/pers.70.8.963. URL: https://doi.org/10.14358/pers.70.8.963.

[52] M. A. Friedl, K. C. McGwire, and D. K. McIver. "An Overview of Uncertainty in Optical Remotely Sensed Data for Ecological Applications". In: *Spatial Uncertainty in Ecology*. Springer New York, 2001, pp. 258–283. DOI: 10.1007/978-1-4613-0209-4_12. URL: https://doi.org/10.1007/978-1-4613-0209-4_12.

[53] M. Nitzberg, D. Mumford, and T. Shiota. *Filtering, Segmentation and Depth*. Springer Berlin Heidelberg, 1993. DOI: 10.1007/3-540-56484-5. URL: https://doi.org/10.1007/3-540-56484-5.

[54] F. Yu and V. Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". In: *CoRR* abs/1511.07122 (2015).

[55] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. "Object Detectors Emerge in Deep Scene CNNs". In: *International Conference on Learning Representations (ICLR)*. 2015.

[56] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *CoRR* abs/1511.07289 (2015).

[57] K. Pearson. "On lines and planes of closest fit to systems of points in space". In: *Philosophical Magazine* 2.6 (1901), pp. 559–572.

[58] A. Dhurandhar. "Insights into Cross-validation". In: 2008.

[59] A. Celisse. "Optimal cross-validation in density estimation". working paper or preprint. Oct. 2008. URL: https://hal.archives-ouvertes.fr/hal-00337058.

[60] J. G. Moreno-Torres, J. A. Saez, and F. Herrera. "Study on the Impact of Partition-Induced Dataset Shift on formula formulatype="inline"tex Notation="TeX"k/tex Fold Cross-Validation". In: *IEEE Transactions on Neural Networks and Learning Systems* 23.8 (Aug. 2012), pp. 1304–1312. DOI: 10.1109/tnnls.2012.2199516. URL: https://doi.org/10.1109/tnnls.2012.2199516.

[61] D. B. Heras, F. Argüello, and P. Quesada-Barriuso. "Exploring ELM-based spatial–spectral classification of hyperspectral images". In: *International Journal of Remote Sensing* 35.2 (Jan. 2014), pp. 401–423. DOI: 10.1080/01431161.2013.869633. URL: https://doi.org/10.1080/01431161.2013.869633.

[62] W. Li, C. Chen, H. Su, and Q. Du. "Local Binary Patterns and Extreme Learning Machine for Hyperspectral Imagery Classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 53.7 (July 2015), pp. 3681–3693. DOI: 10.1109/tgrs.2014.2381602. URL: https://doi.org/10.1109/tgrs.2014.2381602.

[63] F. Melgani and L. Bruzzone. "Classification of hyperspectral remote sensing images with support vector machines". In: *IEEE Transactions on Geoscience and Remote Sensing* 42.8 (Aug. 2004), pp. 1778–1790. DOI: 10.1109/tgrs.2004.831865. URL: https://doi.org/10.1109/tgrs.2004.831865.

[64] Y. Zhong and L. Zhang. "An Adaptive Artificial Immune Network for Supervised Classification of Multi-/Hyperspectral Remote Sensing Imagery". In: *IEEE Transactions on Geoscience and Remote Sensing* 50.3 (Mar. 2012), pp. 894–909. DOI: 10.1109/tgrs.2011.2162589. URL: https://doi.org/10.1109/tgrs.2011.2162589.

[65] 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". In: *IEEE Geoscience and Remote Sensing Letters* 9.3 (May 2012), pp. 447–451. DOI: 10.1109/lgrs.2011.2172185. URL: https://doi.org/10.1109/lgrs.2011.2172185.

[66] T.-M. Tu. "Unsupervised signature extraction and separation in hyperspectral images: a noise-adjusted fast independent component analysis approach". In: *Optical Engineering* 39.4 (Apr. 2000), p. 897. DOI: 10.1117/1.602461. URL: https://doi.org/10.1117/1.602461.

[67] T. Bandos, L. Bruzzone, and G. Camps-Valls. "Classification of Hyperspectral Images With Regularized Linear Discriminant Analysis". In: *IEEE Transactions on Geoscience and Remote Sensing* 47.3 (Mar. 2009), pp. 862–873. DOI: 10.1109/tgrs.2008.2005729. URL: https://doi.org/10.1109/tgrs.2008.2005729.

[68] C.-I. Chang. *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. Springer, 2003. ISBN: 0306474832. URL: https://www.amazon.com/Hyperspectral-Imaging-Techniques-Detection-Classification/dp/0306474832?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0306474832.

[69] P. Gurram and H. Kwon. "Sparse Kernel-Based Ensemble Learning With Fully Optimized Kernel Parameters for Hyperspectral Classification Problems". In: *IEEE Transactions on Geoscience and Remote Sensing* 51.2 (Feb.

2013), pp. 787–802. DOI: 10.1109/tgrs.2012.2203603. URL: https://doi.org/10.1109/tgrs.2012.2203603.

[70] J. Liu, Z. Wu, J. Li, A. Plaza, and Y. Yuan. "Probabilistic-Kernel Collaborative Representation for Spatial–Spectral Hyperspectral Image Classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 54.4 (Apr. 2016), pp. 2371–2384. DOI: 10.1109/tgrs.2015.2500680. URL: https://doi.org/10.1109/tgrs.2015.2500680.

[71] F. de Morsier, M. Borgeaud, V. Gass, J.-P. Thiran, and D. Tuia. "Kernel Low-Rank and Sparse Graph for Unsupervised and Semi-Supervised Classification of Hyperspectral Images". In: *IEEE Transactions on Geoscience and Remote Sensing* 54.6 (June 2016), pp. 3410–3420. DOI: 10.1109/tgrs.2016.2517242. URL: https://doi.org/10.1109/tgrs.2016.2517242.

[72] L. Yang, M. Wang, S. Yang, R. Zhang, and P. Zhang. "Sparse Spatio-Spectral LapSVM With Semisupervised Kernel Propagation for Hyperspectral Image Classification". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 10.5 (May 2017), pp. 2046–2054. DOI: 10.1109/jstars.2016.2647640. URL: https://doi.org/10.1109/jstars.2016.2647640.

[73] J. Li, J. M. Bioucas-Dias, and A. Plaza. "Semisupervised Hyperspectral Image Segmentation Using Multinomial Logistic Regression With Active Learning". In: *IEEE Transactions on Geoscience and Remote Sensing* (Nov. 2010). DOI: 10.1109/tgrs.2010.2060550. URL: https://doi.org/10.1109/tgrs.2010.2060550.

[74] X. Ma, H. Wang, and J. Geng. "Spectral–Spatial Classification of Hyperspectral Image Based on Deep Auto-Encoder". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 9.9 (Sept. 2016), pp. 4073–4085. DOI: 10.1109/jstars.2016.2517204. URL: https://doi.org/10.1109/jstars.2016.2517204.

[75] J. M. Haut, M. Paoletti, J. Plaza, and A. Plaza. "Cloud implementation of the K-means algorithm for hyperspectral image analysis". In: *The Journal of Supercomputing* 73.1 (Oct. 2016), pp. 514–529. DOI: 10.1007/s11227-016-1896-3. URL: https://doi.org/10.1007/s11227-016-1896-3.

[76] L. Ma, M. M. Crawford, and J. Tian. "Local Manifold Learning-Based k -Nearest-Neighbor for Hyperspectral Image Classification". In: *IEEE Transactions on Geoscience and Remote Sensing* (Nov. 2010). DOI: 10.1109/tgrs.2010.2055876. URL: https://doi.org/10.1109/tgrs.2010.2055876.

[77] J.-M. Yang, P.-T. Yu, and B.-C. Kuo. "A Nonparametric Feature Extraction and Its Application to Nearest Neighbor Classification for Hyperspectral Image Data". In: *IEEE Transactions on Geoscience and Remote Sensing* 48.3 (Mar. 2010), pp. 1279–1293. DOI: 10.1109/tgrs.2009.2031812. URL: https://doi.org/10.1109/tgrs.2009.2031812.

[78] H. Z. M. Shafri, A. Suhaili, and S. Mansor. "The Performance of Maximum Likelihood, Spectral Angle Mapper, Neural Network and Decision Tree Classifiers in Hyperspectral Image Analysis". In: *Journal of Computer Science* 3.6 (June 2007), pp. 419–423. DOI: 10.3844/jcssp.2007.419.423. URL: https://doi.org/10.3844/jcssp.2007.419.423.

[79] G. Mercier and M. Lennon. "Support vector machines for hyperspectral image classification with spectral-based kernels". In: *IGARSS 2003. 2003 IEEE International Geoscience and Remote Sensing Symposium. Proceedings (IEEE Cat. No.03CH37477)*. IEEE. DOI: 10.1109/igarss.2003.1293752. URL: https://doi.org/10.1109/igarss.2003.1293752.

[80] J. Ham, Y. Chen, M. Crawford, and J. Ghosh. "Investigation of the random forest framework for classification of hyperspectral data". In: *IEEE Transactions on Geoscience and Remote Sensing* 43.3 (Mar. 2005), pp. 492–501. DOI: 10.1109/tgrs.2004.842481. URL: https://doi.org/10.1109/tgrs.2004.842481.

[81] G. Camps-Valls and L. Bruzzone, eds. *Kernel Methods for Remote Sensing Data Analysis*. John Wiley & Sons, Ltd, Oct. 2009. DOI: 10.1002/9780470748992. URL: https://doi.org/10.1002/9780470748992.

[82] P. Ghamisi, J. Plaza, Y. Chen, J. Li, and A. J. Plaza. "Advanced Spectral Classifiers for Hyperspectral Images: A review". In: *IEEE Geoscience and Remote Sensing Magazine* 5.1 (Mar. 2017), pp. 8–32. DOI: 10.1109/mgrs.2016.2616418. URL: https://doi.org/10.1109/mgrs.2016.2616418.

[83] G. Camps-Valls and L. Bruzzone. "Kernel-based methods for hyperspectral image classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 43.6 (June 2005), pp. 1351–1362. DOI: 10.1109/tgrs.2005.846154. URL: https://doi.org/10.1109/tgrs.2005.846154.

[84] R. Roscher and B. Waske. "Shapelet-Based Sparse Representation for Landcover Classification of Hyperspectral Images". In: *IEEE Transactions on Geoscience and Remote Sensing* 54.3 (Mar. 2016), pp. 1623–1634. DOI: 10.1109/tgrs.2015.2484619. URL: https://doi.org/10.1109/tgrs.2015.2484619.

[85] Q. Wang, Y. Gu, and D. Tuia. "Discriminative Multiple Kernel Learning for Hyperspectral Image Classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 54.7 (July 2016), pp. 3912–3927. DOI: 10.1109/tgrs.2016.2530807. URL: https://doi.org/10.1109/tgrs.2016.2530807.

[86] J. Cihlar. "Land cover mapping of large areas from satellites: Status and research priorities". In: *International Journal of Remote Sensing* 21.6-7 (Jan. 2000), pp. 1093–1114. DOI: 10.1080/014311600210092. URL: https://doi.org/10.1080/014311600210092.

[87] S. E. Franklin and M. A. Wulder. "Remote sensing methods in medium spatial resolution satellite data land cover classification of large areas". In: *Progress in Physical Geography: Earth and Environment* 26.2 (June 2002), pp. 173–205. DOI: 10.1191/0309133302pp332ra. URL: https://doi.org/10.1191/0309133302pp332ra.

[88] D. A. Landgrebe. *Signal Theory Methods in Multispectral Remote Sensing*. John Wiley & Sons, Inc., Jan. 2003. DOI: 10.1002/0471723800. URL: https://doi.org/10.1002/0471723800.

[89] P. Mather and B. Tso. *Classification Methods for Remotely Sensed Data*. CRC Press, 2009. ISBN: 1420090720. URL: https://www.amazon.com/Classification-Methods-Remotely-Sensed-Data/dp/1420090720?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1420090720.

[90] J. D. PAOLA and R. A. SCHOWENGERDT. "A review and analysis of backpropagation neural networks for classification of remotely-sensed multi-spectral imagery". In: *International Journal of Remote Sensing* 16.16 (Nov. 1995), pp. 3033–3058. DOI: 10.1080/01431169508954607. URL: https://doi.org/10.1080/01431169508954607.

[91] P. M. Atkinson and A. R. L. Tatnall. "Introduction Neural networks in remote sensing". In: *International Journal of Remote Sensing* 18.4 (Mar. 1997), pp. 699–709. DOI: 10.1080/014311697218700. URL: https://doi.org/10.1080/014311697218700.

[92] G. M. Foody and M. K. Arora. "An evaluation of some factors affecting the accuracy of classification by an artificial neural network". In: *International Journal of Remote Sensing* 18.4 (Mar. 1997), pp. 799–810. DOI: 10.1080/014311697218764. URL: https://doi.org/10.1080/014311697218764.

[93] M. Friedl, C. Brodley, and A. Strahler. "Maximizing land cover classification accuracies produced by decision trees at continental to global scales". In: *IEEE Transactions on Geoscience and Remote Sensing* 37.2 (Mar. 1999), pp. 969–977. DOI: 10.1109/36.752215. URL: https://doi.org/10.1109/36.752215.

[94] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, and S. Y. Bang. "Constructing support vector machine ensemble". In: *Pattern Recognition* 36.12 (Dec. 2003), pp. 2757–2767. DOI: 10.1016/s0031-3203(03)00175-4. URL: https://doi.org/10.1016/s0031-3203(03)00175-4.

[95] P. Fisher. "The pixel: A snare and a delusion". In: *International Journal of Remote Sensing* 18.3 (Feb. 1997), pp. 679–685. DOI: 10.1080/014311697219015. URL: https://doi.org/10.1080/014311697219015.

[96] A. P. Cracknell. "Review article Synergy in remote sensing-whats in a pixel?" In: *International Journal of Remote Sensing* 19.11 (Jan. 1998), pp. 2025–2047. DOI: 10.1080/014311698214848. URL: https://doi.org/10.1080/014311698214848.

[97] D. R. Peddle, G. M. Foody, A. Zhang, S. E. Franklin, and E. F. LeDrew. "Multi-Source Image Classification II: An Empirical Comparison of Evidential Reasoning and Neural Network Approaches". In: *Canadian Journal of*

Remote Sensing 20.4 (Dec. 1994), pp. 396–407. DOI: 10.1080/07038992.1994.10874582. URL: https://doi.org/10.1080/07038992.1994.10874582.

[98] C. Ricotta. "The influence of fuzzy set theory on the areal extent of thematic map classes". In: *International Journal of Remote Sensing* 20.1 (Jan. 1999), pp. 201–205. DOI: 10.1080/014311699213712. URL: https://doi.org/10.1080/014311699213712.

[99] I. Bloch. "Information combination operators for data fusion: a comparative review with classification". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 26.1 (1996), pp. 52–67. DOI: 10.1109/3468.477860. URL: https://doi.org/10.1109/3468.477860.

[100] R. Schowengerdt. "On the estimation of spatial-spectral mixing with classifier likelihood functions". In: *Pattern Recognition Letters* 17.13 (Nov. 1996), pp. 1379–1387. DOI: 10.1016/s0167-8655(96)00094-3. URL: https://doi.org/10.1016/s0167-8655(96)00094-3.

[101] A. D. Kulkarni and K. Lulla. "Fuzzy Neural Network Models for Supervised Classification: Multispectral Image Analysis". In: *Geocarto International* 14.4 (Dec. 1999), pp. 42–51. DOI: 10.1080/10106049908542127. URL: https://doi.org/10.1080/10106049908542127.

[102] B. Mannan and A. K. Ray. "Crisp and fuzzy competitive learning networks for supervised classification of multispectral IRS scenes". In: *International Journal of Remote Sensing* 24.17 (Jan. 2003), pp. 3491–3502. DOI: 10.1080/0143116021000053805. URL: https://doi.org/10.1080/0143116021000053805.

[103] G. M. FOODY. "Approaches for the production and evaluation of fuzzy land cover classifications from remotely-sensed data". In: *International Journal of Remote Sensing* 17.7 (May 1996), pp. 1317–1340. DOI: 10.1080/01431169608948706. URL: https://doi.org/10.1080/01431169608948706.

[104] G. M. Foody. "Sharpening fuzzy classification output to refine the representation of sub-pixel land cover distribution". In: *International Journal of Remote Sensing* 19.13 (Jan. 1998), pp. 2593–2599. DOI: 10.1080/014311698214659. URL: https://doi.org/10.1080/014311698214659.

[105] F. MASELLl, A. RODOLFI, and C. CONESE. "Fuzzy classification of spatially degraded Thematic Mapper data for the estimation of sub-pixel components". In: *International Journal of Remote Sensing* 17.3 (Feb. 1996), pp. 537–551. DOI: 10.1080/01431169608949026. URL: https://doi.org/10.1080/01431169608949026.

[106] B. Mannan, J. Roy, and A. K. Ray. "Fuzzy ARTMAP supervised classification of multi-spectral remotely-sensed images". In: *International Journal of Remote Sensing* 19.4 (Jan. 1998), pp. 767–774. DOI: 10.1080/014311698215991. URL: https://doi.org/10.1080/014311698215991.

[107] J Adams. "Classification of multispectral images based on fractions of endmembers: Application to land-cover change in the Brazilian Amazon". In: *Remote Sensing of Environment* 52.2 (May 1995), pp. 137–154. DOI: 10.1016/0034-4257(94)00098-8. URL: https://doi.org/10.1016/0034-4257(94)00098-8.

[108] D. Roberts, M. Gardner, R. Church, S. Ustin, G. Scheer, and R. Green. "Mapping Chaparral in the Santa Monica Mountains Using Multiple Endmember Spectral Mixture Models". In: *Remote Sensing of Environment* 65.3 (Sept. 1998), pp. 267–279. DOI: 10.1016/s0034-4257(98)00037-6. URL: https://doi.org/10.1016/s0034-4257(98)00037-6.

[109] M. O. Smith, S. L. Ustin, J. B. Adams, and A. R. Gillespie. "Vegetation in deserts: I. A regional measure of abundance from multispectral images". In: *Remote Sensing of Environment* 31.1 (Jan. 1990), pp. 1–26. DOI: 10.1016/0034-4257(90)90074-v. URL: https://doi.org/10.1016/0034-4257(90)90074-v.

[110] D. Roberts, M. Smith, and J. Adams. "Green vegetation, nonphotosynthetic vegetation, and soils in AVIRIS data". In: *Remote Sensing of Environment* 44.2-3 (May 1993), pp. 255–269. DOI: 10.1016/0034-4257(93)90020-x. URL: https://doi.org/10.1016/0034-4257(93)90020-x.

[111] J. J. SETTLE and N. A. DRAKE. "Linear mixing and the estimation of ground cover proportions". In: *International Journal of Remote Sensing* 14.6 (Apr. 1993), pp. 1159–1177. DOI: 10.1080/01431169308904402. URL: https://doi.org/10.1080/01431169308904402.

[112] A. Bateson and B. Curtiss. "A method for manual endmember selection and spectral unmixing". In: *Remote Sensing of Environment* 55.3 (Mar. 1996), pp. 229–243. DOI: 10.1016/s0034-4257(95)00177-8. URL: https://doi.org/10.1016/s0034-4257(95)00177-8.

[113] S Tompkins. "Optimization of endmembers for spectral mixture analysis". In: *Remote Sensing of Environment* 59.3 (Mar. 1997), pp. 472–489. DOI: 10.1016/s0034-4257(96)00122-8. URL: https://doi.org/10.1016/s0034-4257(96)00122-8.

[114] F. Garcı
a-Haro, M. Gilabert, and J. Meliá. "Extraction of Endmembers from Spectral Mixtures". In: *Remote Sensing of Environment* 68.3 (June 1999), pp. 237–253. DOI: 10.1016/s0034-4257(98)00115-1. URL: https://doi.org/10.1016/s0034-4257(98)00115-1.

[115] F. V. D. Meer. "Iterative spectral unmixing (ISU)". In: *International Journal of Remote Sensing* 20.17 (Jan. 1999), pp. 3431–3436. DOI: 10.1080/014311699211462. URL: https://doi.org/10.1080/014311699211462.

[116] F Maselli. "Definition of Spatially Variable Spectral Endmembers by Locally Calibrated Multivariate Regression Analyses". In: *Remote Sensing of Environment* 75.1 (Jan. 2001), pp. 29–38. DOI: 10.1016/s0034-4257(00)00153-x. URL: https://doi.org/10.1016/s0034-4257(00)00153-x.

[117] Y. E. Shimabukuro, G. T. Batista, E. M. K. Mello, J. C. Moreira, and V. Duarte. "Using shade fraction image segmentation to evaluate deforestation in Landsat Thematic Mapper images of the Amazon Region". In: *International Journal of Remote Sensing* 19.3 (Jan. 1998), pp. 535–541. DOI: 10.1080/014311698216152. URL: https://doi.org/10.1080/014311698216152.

[118] B. KARTIKEYAN, B. GOPALAKRISHNA, M. H. KALUBARME, and K. L. MAJUMDER. "Contextual techniques for classification of high and low resolution remote sensing data". In: *International Journal of Remote Sensing* 15.5 (Mar. 1994), pp. 1037–1051. DOI: 10.1080/01431169408954132. URL: https://doi.org/10.1080/01431169408954132.

[119] A.-M. Flygare. "A comparison of contextual classification methods using Landsat TM". In: *International Journal of Remote Sensing* 18.18 (Dec. 1997), pp. 3835–3842. DOI: 10.1080/014311697216649. URL: https://doi.org/10.1080/014311697216649.

[120] K. M. S. Sharma and A. Sarkar. "A Modified Contextual Classification Technique for Remote Sensing Data". In: 1998.

[121] J. Keuchel, S. Naumann, M. Heiler, and A. Siegmund. "Automatic land cover analysis for Tenerife by supervised classification using remotely sensed data". In: *Remote Sensing of Environment* 86.4 (Aug. 2003), pp. 530–541. DOI: 10.1016/s0034-4257(03)00130-5. URL: https://doi.org/10.1016/s0034-4257(03)00130-5.

[122] S. Magnussen, P. Boudewyn, and M. Wulder. "Contextual classification of Landsat TM images to forest inventory cover types". In: *International Journal of Remote Sensing* 25.12 (June 2004), pp. 2421–2440. DOI: 10.1080/01431160310001642296. URL: https://doi.org/10.1080/01431160310001642296.

[123] J. Stuckens, P. Coppin, and M. Bauer. "Integrating Contextual Information with per-Pixel Classification for Improved Land Cover Classification". In: *Remote Sensing of Environment* 71.3 (Mar. 2000), pp. 282–296. DOI: 10.1016/s0034-4257(99)00083-8. URL: https://doi.org/10.1016/s0034-4257(99)00083-8.

[124] E. Binaghi, P. Madella, M. G. Montesano, and A. Rampini. "Fuzzy contextual classification of multisource remote sensing images". In: *IEEE Transactions on Geoscience and Remote Sensing* 35.2 (Mar. 1997), pp. 326–340. DOI: 10.1109/36.563272. URL: https://doi.org/10.1109/36.563272.

[125] F. J. Cortijo and N. P. D. L. Blanca. "Improving classical contextual classifications". In: *International Journal of Remote Sensing* 19.8 (Jan. 1998), pp. 1591–1613. DOI: 10.1080/014311698215379. URL: https://doi.org/10.1080/014311698215379.

[126] B. Kartikeyan, A. Sarkar, and K. L. Majumder. "A segmentation approach to classification of remote sensing imagery". In: *International Journal of Remote Sensing* 19.9 (Jan. 1998), pp. 1695–1709. DOI: 10.1080/014311698215199. URL: https://doi.org/10.1080/014311698215199.

[127] K. Fukushima. "Neocognitron: A hierarchical neural network capable of visual pattern recognition". In: *Neural Networks* 1.2 (Jan. 1988), pp. 119–130. DOI: 10.1016/0893-6080(88)90014-7. URL: https://doi.org/10.1016/0893-6080(88)90014-7.

[128] D. Ciresan, U. Meier, J. Masci, L. Gambardella, and J. Schmidhuber. "Flexible, High Performance Convolutional Neural Networks for Image Classification." In: July 2011, pp. 1237–1242. DOI: 10.5591/978-1-57735-516-8/IJCAI11-210.

[129] W. Rawat and Z. Wang. "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review". In: *Neural Computation* 29.9 (Sept. 2017), pp. 2352–2449. DOI: 10.1162/neco_a_00990. URL: https://doi.org/10.1162/neco_a_00990.

[130] P. Tang, X. Wang, B. Feng, and W. Liu. "Learning Multi-Instance Deep Discriminative Patterns for Image Classification". In: *IEEE Transactions on Image Processing* 26.7 (July 2017), pp. 3385–3396. DOI: 10.1109/tip.2016.2642781. URL: https://doi.org/10.1109/tip.2016.2642781.

[131] P. Xu and R. Sarikaya. "Contextual domain classification in spoken language understanding systems using recurrent neural network". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2014. DOI: 10.1109/icassp.2014.6853573. URL: https://doi.org/10.1109/icassp.2014.6853573.

[132] M. Price, J. Glass, and A. P. Chandrakasan. "A Low-Power Speech Recognizer and Voice Activity Detector Using Deep Neural Networks". In: *IEEE Journal of Solid-State Circuits* 53.1 (Jan. 2018), pp. 66–75. DOI: 10.1109/jssc.2017.2752838. URL: https://doi.org/10.1109/jssc.2017.2752838.

[133] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu. "Deep Learning-Based Classification of Hyperspectral Data". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7.6 (June 2014), pp. 2094–2107. DOI: 10.1109/jstars.2014.2329330. URL: https://doi.org/10.1109/jstars.2014.2329330.

[134] and, and and. "Convolutional Neural Networks Based Hyperspectral Image Classification Method with Adaptive Kernels". In: *Remote Sensing* 9.6 (June 2017), p. 618. DOI: 10.3390/rs9060618. URL: https://doi.org/10.3390/rs9060618.

[135] Y. Chen, X. Zhao, and X. Jia. "Spectral–Spatial Classification of Hyperspectral Data Based on Deep Belief Network". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8.6 (June 2015), pp. 2381–2392. DOI: 10.1109/jstars.2015.2388577. URL: https://doi.org/10.1109/jstars.2015.2388577.

[136] C. Yu, B. Xue, Y. Wang, M. Song, L. Wang, S. Li, S.-Y. Chen, and C.-I. Chang. "Multi-class constrained background suppression approach to hyperspectral image classification". In: *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, July 2017. DOI: 10.1109/igarss.2017.8127717. URL: https://doi.org/10.1109/igarss.2017.8127717.

[137] V. Dumoulin and F. Visin. "A guide to convolution arithmetic for deep learning". In: *ArXiv* abs/1603.07285 (2016).

[138] H. Noh, S. Hong, and B. Han. "Learning Deconvolution Network for Semantic Segmentation". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Dec. 2015. DOI: 10.1109/iccv.2015.178. URL: https://doi.org/10.1109/iccv.2015.178.

[139] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. "Deconvolutional networks". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, June 2010. DOI: 10.1109/cvpr.2010.5539957. URL: https://doi.org/10.1109/cvpr.2010.5539957.

[140] H.-E. Kim and S. Hwang. "Deconvolutional Feature Stacking for Weakly-Supervised Semantic Segmentation". In: 2016.

[141] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. "Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[142] J. Yi, P. Wu, D. J. Hoeppner, and D. Metaxas. "Pixel-wise neural cell instance segmentation". In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE, Apr. 2018. DOI: 10.1109/isbi.2018.8363596. URL: https://doi.org/10.1109/isbi.2018.8363596.

[143] S. Kim, K. Lee, S. Doo, and B. Shim. "Moving Target Classification In Automotive Radar Systems Using Transposed Convolutional Networks". In: *2018 52nd Asilomar Conference on Signals, Systems, and Computers*. IEEE, Oct. 2018. DOI: 10.1109/acssc.2018.8645406. URL: https://doi.org/10.1109/acssc.2018.8645406.

[144] C. Chen, X. Tian, F. Wu, and Z. Xiong. "UDNet: Up-Down Network for Compact and Efficient Feature Representation in Image Super-Resolution". In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. IEEE, Oct. 2017. DOI: 10.1109/iccvw.2017.130. URL: https://doi.org/10.1109/iccvw.2017.130.

[145] Y. Lu, Y. Zhou, Z. Jiang, X. Guo, and Z. Yang. "Channel Attention and Multi-level Features Fusion for Single Image Super-Resolution". In: *2018 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, Dec. 2018. DOI: 10.1109/vcip.2018.8698663. URL: https://doi.org/10.1109/vcip.2018.8698663.

[146] H. Gao, H. Yuan, Z. Wang, and S. Ji. "Pixel Transposed Convolutional Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), pp. 1–1. DOI: 10.1109/tpami.2019.2893965. URL: https://doi.org/10.1109/tpami.2019.2893965.

[147] R. Hamaguchi, A. Fujita, K. Nemoto, T. Imaizumi, and S. Hikosaka. "Effective Use of Dilated Convolutions for Segmenting Small Object Instances in Remote Sensing Imagery". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2018. DOI: 10.1109/wacv.2018.00162. URL: https://doi.org/10.1109/wacv.2018.00162.

[148] T. Yamashita, H. Furukawa, and H. Fujiyoshi. "Multiple Skip Connections of Dilated Convolution Network for Semantic Segmentation". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, Oct.

2018. DOI: 10.1109/icip.2018.8451033. URL: https://doi.org/10.1109/icip.2018.8451033.

[149] Y. Li, X. Zhang, and D. Chen. "CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00120. URL: https://doi.org/10.1109/cvpr.2018.00120.

[150] Y. Wei, H. Xiao, H. Shi, Z. Jie, J. Feng, and T. S. Huang. "Revisiting Dilated Convolution: A Simple Approach for Weakly- and Semi-Supervised Semantic Segmentation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00759. URL: https://doi.org/10.1109/cvpr.2018.00759.

[151] X. Lei, H. Pan, and X. Huang. "A dilated CNN model for image classification". In: *IEEE Access* (2019), pp. 1–1. DOI: 10.1109/access.2019.2927169. URL: https://doi.org/10.1109/access.2019.2927169.

[152] M. Wu, C. Zhang, J. Liu, L. Zhou, and X. Li. "Towards Accurate High Resolution Satellite Image Semantic Segmentation". In: *IEEE Access* 7 (2019), pp. 55609–55619. DOI: 10.1109/access.2019.2913442. URL: https://doi.org/10.1109/access.2019.2913442.

[153] Y. Zhang, Z. Xiao, L. Wu, X. Lu, and Y. Wang. "Deep learning for subsurface penetrating super-resolution imaging". In: *2017 10th UK-Europe-China Workshop on Millimetre Waves and Terahertz Technologies (UCMMT)*. IEEE, Sept. 2017. DOI: 10.1109/ucmmt.2017.8068492. URL: https://doi.org/10.1109/ucmmt.2017.8068492.

[154] T. Hossen, S. J. Plathottam, R. K. Angamuthu, P. Ranganathan, and H. Salehfar. "Short-term load forecasting using deep neural networks (DNN)". In: *2017 North American Power Symposium (NAPS)*. IEEE, Sept. 2017. DOI: 10.1109/naps.2017.8107271. URL: https://doi.org/10.1109/naps.2017.8107271.

[155] V. Thakkar, S. Tewary, and C. Chakraborty. "Batch Normalization in Convolutional Neural Networks — A comparative study with CIFAR-10 data".

In: *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*. IEEE, Jan. 2018. DOI: 10.1109/eait.2018.8470438. URL: https://doi.org/10.1109/eait.2018.8470438.

[156] Y. Xu, B. Du, F. Zhang, and L. Zhang. "Hyperspectral image classification via a random patches network". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 142 (Aug. 2018), pp. 344–357. DOI: 10.1016/j.isprsjprs.2018.05.014. URL: https://doi.org/10.1016/j.isprsjprs.2018.05.014.

[157] L. Shu, K. McIsaac, and G. R. Osinski. "Hyperspectral Image Classification With Stacking Spectral Patches and Convolutional Neural Networks". In: *IEEE Transactions on Geoscience and Remote Sensing* 56.10 (Oct. 2018), pp. 5975–5984. DOI: 10.1109/tgrs.2018.2829400. URL: https://doi.org/10.1109/tgrs.2018.2829400.

[158] J. Hamwood, D. Alonso-Caneiro, S. A. Read, S. J. Vincent, and M. J. Collins. "Effect of patch size and network architecture on a convolutional neural network approach for automatic segmentation of OCT retinal layers". In: *Biomedical Optics Express* 9.7 (June 2018), p. 3049. DOI: 10.1364/boe.9.003049. URL: https://doi.org/10.1364/boe.9.003049.

[159] L. Hou, D. Samaras, T. M. Kurc, Y. Gao, J. E. Davis, and J. H. Saltz. "Patch-Based Convolutional Neural Network for Whole Slide Tissue Image Classification". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.266. URL: https://doi.org/10.1109/cvpr.2016.266.

[160] K. Roy, D. Banik, D. Bhattacharjee, and M. Nasipuri. "Patch-based system for Classification of Breast Histology images using deep learning". In: *Computerized Medical Imaging and Graphics* 71 (Jan. 2019), pp. 90–103. DOI: 10.1016/j.compmedimag.2018.11.003. URL: https://doi.org/10.1016/j.compmedimag.2018.11.003.

[161] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (June 2002), pp. 321–357. DOI: 10.1613/jair.953. URL: https://doi.org/10.1613/jair.953.

[162] Z. Zheng, Y. Cai, and Y. Li. "Oversampling Method for Imbalanced Classification". In: 34 (2015), 1017—1037. URL: https://cai.type.sk/content/2015/5/oversampling-method-for-imbalanced-classification/.

[163] H. Han, W.-Y. Wang, and B.-H. Mao. "Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 878–887. DOI: 10.1007/11538059_91. URL: https://doi.org/10.1007/11538059_91.

[164] B Santoso, H Wijayanto, K. A. Notodiputro, and B Sartono. "Synthetic Over Sampling Methods for Handling Class Imbalanced Problems : A Review". In: *IOP Conference Series: Earth and Environmental Science* 58 (Mar. 2017), p. 012031. DOI: 10.1088/1755-1315/58/1/012031. URL: https://doi.org/10.1088/1755-1315/58/1/012031.

[165] R. Luo, Q. Feng, C. Wang, X. Yang, H. Tu, Q. Yu, S. Fei, and X. Gong. "Feature Learning With a Divergence-Encouraging Autoencoder for Imbalanced Data Classification". In: *IEEE Access* 6 (2018), pp. 70197–70211. DOI: 10.1109/access.2018.2879221. URL: https://doi.org/10.1109/access.2018.2879221.

[166] G. Li, D. Lu, E. Moran, and S. J. S. Sant'Anna. "Comparative analysis of classification algorithms and multiple sensor data for land use/land cover classification in the Brazilian Amazon". In: *Journal of Applied Remote Sensing* 6.1 (Dec. 2012), p. 061706. DOI: 10.1117/1.jrs.6.061706. URL: https://doi.org/10.1117/1.jrs.6.061706.

[167] F. J. Cortijo and N. P. D. L. Blanca. "A comparative study of some non-parametric spectral classifiers. Applications to problems with high-overlapping training sets". In: *International Journal of Remote Sensing* 18.6 (Apr. 1997), pp. 1259–1275. DOI: 10.1080/014311697218403. URL: https://doi.org/10.1080/014311697218403.

[168] D Michelson. "Comparison of Algorithms for Classifying Swedish Landcover Using Landsat TM and ERS-1 SAR Data". In: *Remote Sensing of Environment* 71.1 (Jan. 2000), pp. 1–15. DOI: 10.1016/s0034-4257(99)00024-3. URL: https://doi.org/10.1016/s0034-4257(99)00024-3.

[169] M. Pal and P. M. Mather. "An assessment of the effectiveness of decision tree methods for land cover classification". In: *Remote Sensing of Environment* 86.4 (Aug. 2003), pp. 554–565. DOI: 10.1016/s0034-4257(03)00132-9. URL: https://doi.org/10.1016/s0034-4257(03)00132-9.

[170] M. D. Zeiler and R. Fergus. "Visualizing and Understanding Convolutional Networks". In: *European Conference on Computer Vision*. 2014, pp. 818–833. DOI: 10.1007/978-3-319-10590-1_53.

[171] D. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014).

[172] H. He and E. A. Garcia. "Learning from Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284. DOI: 10.1109/tkde.2008.239.

[173] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis. "Deep supervised learning for hyperspectral data classification through convolutional neural networks". In: *International Geoscience and Remote Sensing Symposium*. 2015. DOI: 10.1109/igarss.2015.7326945.

[174] Q. Gao, S. Lim, and X. Jia. "Hyperspectral Image Classification Using Convolutional Neural Networks and Multiple Feature Learning". In: *Remote Sensing* 10 (2018), p. 299. DOI: 10.3390/rs10020299.

[175] A. Mughees, A. Ali, and L Tao. "Hyperspectral image classification via shape-adaptive deep learning". In: *International Conference on Image Processing*. 2017. DOI: 10.1109/icip.2017.8296306.

[176] A. J. X. Guo and F. Zhu. "A CNN-based Spatial Feature Fusion Algorithm for Hyperspectral Imagery Classification". In: *CoRR* abs/1801.10355 (2018).

[177] Y. Li, H. Zhang, and Q. Shen. "Spectral–Spatial Classification of Hyperspectral Imagery with 3D Convolutional Neural Network". In: *Remote Sensing* 9.1 (2017). DOI: 10.3390/rs9010067.

[178] Y. Ju, L. Li, L. Jiao, R. Zhongle, B. Hou, and S. Yang. "Modified Diversity of Class Probability Estimation Co-training for Hyperspectral Image Classification". In: *CoRR* abs/1809.01436 (2018).

[179] A. J. X. Guo and F. Zhu. "Spectral-Spatial Feature Extraction and Classification by ANN Supervised With Center Loss in Hyperspectral Imagery". In: *IEEE Transactions on Geoscience and Remote Sensing* (2018), pp. 1–13. DOI: 10.1109/tgrs.2018.2869004.

[180] Z. Wang, N. M. Nasrabadi, and T. S. Huang. "Spatial–Spectral Classification of Hyperspectral Images Using Discriminative Dictionary Designed by Learning Vector Quantization". In: *IEEE Transactions on Geoscience and Remote Sensing* 52 (2014), pp. 4808–4822. DOI: 10.1109/tgrs.2013.2285049.

[181] S. A. Medjahed, T. A. Saadi, A. Benyettou, and M. Ouali. "A new post-classification and band selection frameworks for hyperspectral image classification". In: *The Egyptian Journal of Remote Sensing and Space Science* 2 (2016), pp. 163–173. DOI: 10.1016/j.ejrs.2016.09.003.

[182] S. Ranjan, D. R. Nayak, K. S. Kumar, R. Dash, and B. Majhi. "Hyperspectral image classification: A k-means clustering based approach". In: *International Conference on Advanced Computing and Communication Systems*. 2017. DOI: 10.1109/icaccs.2017.8014707.

[183] J. Xia, N. Yokoya, and A. Iwasaki. "Hyperspectral image classification with partial least square forest". In: *International Geoscience and Remote Sensing Symposium*. 2017. DOI: 10.1109/igarss.2017.8127790.

[184] H. Gao, S. Lin, Y. Yang, C. Li, and M. Yang. "Convolution Neural Network Based on Two-Dimensional Spectrum for Hyperspectral Image Classification". In: *Journal of Sensors* 2018 (2018), pp. 1–13. DOI: 10.1155/2018/8602103.

[185] C. Li, S. X. Yang, Y. Yang, H. Gao, J. Zhao, X. Qu, Y. Wang, D. Yao, and J. Gao. "Hyperspectral Remote Sensing Image Classification Based on Maximum Overlap Pooling Convolutional Neural Network". In: *Sensors* 18 (2018), p. 3587. DOI: 10.3390/s18103587.

[186] A. Mughees and L. Tao. "Hyper-voxel based deep learning for hyperspectral image classification". In: *International Conference on Image Processing*. 2017. DOI: 10.1109/icip.2017.8296399.