

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belgaum, Karnataka - 590018



Department Of Computer Science And Engineering

A Project Report on

“HYPER SPECTRAL IMAGE CLASSIFICATION”

In partial fulfillment of the Final Year Project Phase-2

For the Academic Year 2022-23

Submitted by:

VARUN H S

(ISK20CS417)

Under The Guidance Of:

Dr. Shyleshchandra Gudihatti K N

Associate Professor

Dept of CSE



Govt. Sri Krishnarajendra Silver Jubilee Technological Institute

(Affiliated to Visvesvaraya Technological University, Belgaum)

KR Circle, Bangalore-560 001

Govt. Sri Krishnarajendra Silver Jubilee Technological Institute

(Affiliated to Visvesvaraya Technological University, Belgaum)

KR Circle, Bangalore-560001

Department Of Computer Science And Engineering



CERTIFICATE:

This is to certify that the Project entitled as “**Hyperspectral Image Classification**” is a bonafide work carried out by **VARUN H S (ISK20CS417)** as a partial fulfillment for the award of Bachelor’s Degree in **Computer Science And Engineering** for **Project Phase-2 Report** as prescribed by **Visvesvaraya Technological University, Belgaum** during the academic year **2022-23**.

Signature of Guide

Dr. Shyleshchandra Gudihatti K N

Associate Professor

Department of CSE

Signature of HOD

Dr. Pradeep Kumar K

Associate Professor and HOD

Department of CSE

Signature of Principal

Dr. M B Patil

Principal

G.S.K.S.J.T.I

Examiner Signature

1.

2.

ACKNOWLEDGEMENT

A unique opportunity like this comes very rarely. It is indeed a pleasure for us to work in this technical seminar. The satisfaction that accompanies the successful completion of this project is incomplete without the mention of the people whose guidance has made it possible for us to complete this project.

I am grateful to our institution **Govt. Sri Krishnarajendra Silver Jubilee Technological Institute** with its ideals and inspiration for providing us with facilities that have made this project successful.

I am grateful to our principal **Dr. M B Patil, Principal G.S.K.S.J.T.I** and **Dr. PRADEEP KUMAR K, Associate Professor and Head of the department of computer science and engineering** for providing excellent lab facilities that helped us in completing our project in time.

I would like to acknowledge **Dr Shyleshchandra Gudihatti K N, Associate Professor, Department of Computer science and Engineering** in helping us to understand the relevant concepts related to the project and providing guidance when necessary.

VARUN H S
(1SK20CS417)

ABSTRACT

Hyper-spectral image classification is a popular topic in the field of remote sensing. Hyper- spectral images captures light information from across the electromagnetic spectrum. This gives significantly large amount of information to perform classification tasks. With the advent of Deep Learning many neural networks have been proposed for Hyper-spectral Image Classification. Recently, many Convolutional Neural Network based models have been proposed. However, many of these frameworks use only 3D-CNN or only 2D CNN or the use of alternate 3D-2D CNN. They do not fully capture the spectral, spatial and the spectral-spatial features. To solve this issue a Novel 3DCNN with spectral-spatial feature extraction, spatial feature extraction and a spectral feature extraction method is proposed. Specifically use Principal Component Analysis to reduce the dimensions along the spectral dimension, later for each pixel, the surrounding neighborhood pixels are formed into a data cube and fed into the 3D convolutions to hierarchically extract high level spectral- spatial features. Then introduce a spectral feature extraction sub-network and a spatial feature extraction sub-network to extract useful features from the input. Then combine the outputs of these two sub-networks into a feature fusion layer.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ACKNOWLEDGEMENT	i
	ABSTRACT	ii
	LIST OF FIGURES	iv
	LIST OF TABLES	vi
1.	INTRODUCTION	1
	1.1 Introduction to Hyper-spectral Images	2
	1.2 Introduction to Deep Learning	4
	1.3 Problem Statement	6
	1.4 Objectives	7
	1.5 Purpose	7
	1.6 Scope	8
2.	LITERATURE SURVEY	10
3.	SYSTEM REQUIREMENT SPECIFICATION	13
	3.1 Hardware Requirements	14
	3.2 Software Requirements	14
4.	SYSTEM DESIGN AND IMPLEMENTATION	15
	4.1 System Architecture	16
	4.2 Data Flow Diagram	21
	4.3 Modules/APIs Used	22
5.	IMPLEMENTATION	25
	5.1 Pseudo Code	28
6.	TESTING	32
	6.1 Testing	33
	6.2 Test cases	34
7.	PERFORMANCE EVALUATION	36
	7.1 Screenshots	41
8.	CONCLUSIONS AND FUTURE WORK	47
9.	REFERENCES	49

LIST OF FIGURES

Figure No.	Caption	Page No.
Figure 1.1	Hyperspectral Image	2
Figure 1.2	Neural Network	4
Figure 4.1	Proposed System architecture diagram	16
Figure 4.2	Convolution Operation	17
Figure 4.3	Mapping of Mish activation function	18
Figure 4.4	Input and output Mapping of SoftMax activation function	19
Figure 4.5	Fully Connected Layer	19
Figure 4.6	Dropout Layer	20
Figure 4.7	Data flow diagram	21
Figure 7.1	Indian Pines Color Image , Ground Truth and Legend	36
Figure 7.2	Indian Pines Predicted Ground Truth after training on 30% of the labeled data	36
Figure 7.3	Pavia University Color Image , Ground Truth and Legend	37
Figure 7.4	Pavia University Predicted Ground Truth using 30% of the labeled data	37
Figure 7.5	Salinas Scene Color Image , Ground Truth and Legend	38
Figure 7.6	Salinas Scene Predicted Ground Truth using 30% of labeled data	39

Pseudo Code 1	Loading of Dataset	28
Pseudo Code 2	Scaling module	28
Pseudo Code 3	PCA module	29
Pseudo Code 4	Padding module	29
Pseudo Code 5	Module to create patches	29
Pseudo Code 6	Module to plot Variance Ratio	30
Pseudo Code 7.1	Convolutional Neural Network Module	30
Pseudo Code 7.2	Convolutional Neural Network Module	31
Screenshot 1	Dataset Description and Training Samples	41
Screenshot 2	Color Image of the dataset	41
Screenshot 3	Ground Truth with Legend	42
Screenshot 4	Spectral Signature of pixel	43
Screenshot 5	Variance Ratio Plot	43
Screenshot 6	Data split into training set and testing set	44
Screenshot 7	Accuracy Report	45
Screenshot 8	Distribution of pixel into classes	46

LIST OF TABLES

Table No.	Description	Page No.
Table 5.2.1	Test case 1	28
Table 5.2.2	Test case 2	28
Table 6.1	Accuracy	40

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 Introduction to Hyperspectral Images

Hyperspectral images unlike regular images can be visualized as a 3-Dimensional hyper cube that contains a 2 Dimensional pixel image called spatial image and 1-Dimensional array for each pixel called the spectral range that collects information from across the electromagnetic spectrum. The 1-Dimensional array is further divided into narrow channels or bands that have a width of 4 - 10nm over the range of 400nm – 2500nm. Unlike ordinary images Hyperspectral images are rich in spectral information, and this spectral information can reflect the physical structure and chemical composition of the objects of interest, and hence provide high discrimination ability. And with the development of remote sensing technology, research study on the use of hyperspectral images has become popular over the years. Although, it is not possible for a person to afford this technology, government organizations however can afford it. It has been found to have a wide range of applications in environment, medicine, military and mining fields.

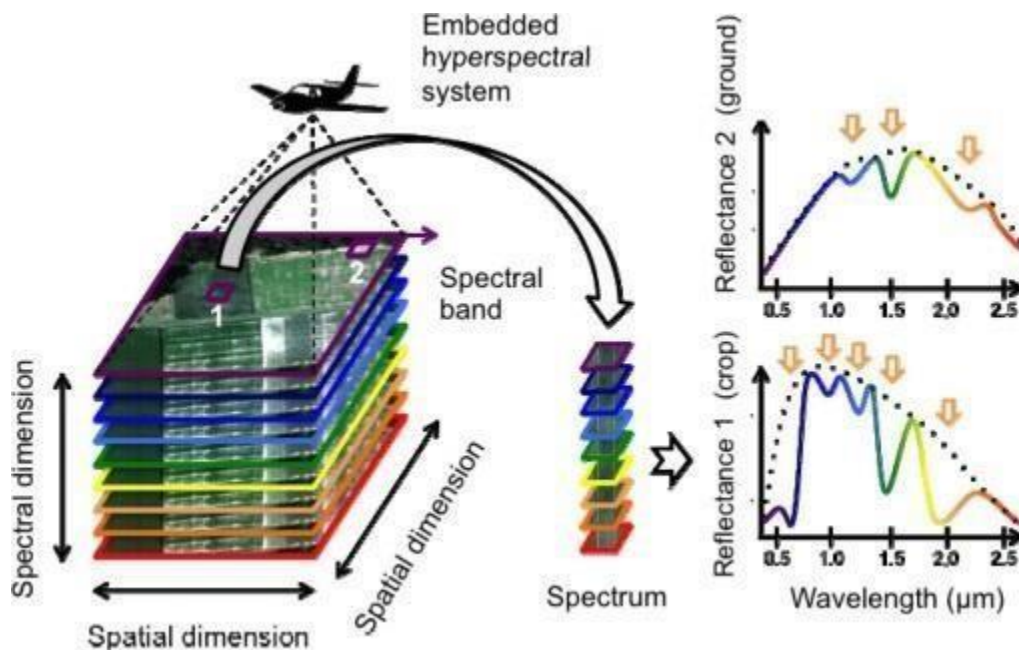


Figure 1.1 Hyperspectral Image

Remote sensing is the process of detecting and monitoring the physical characteristics of an area, by measuring the radiation reflected from a distance, usually from an aerial vehicle like a drone or from a satellite. Hyperspectral remote sensing, also known as imaging spectroscopy, is a relatively new technology that is currently being investigated by researchers and scientists with regard to the detection and identification of minerals, terrestrial vegetation, and man-made materials based on their spatial and spectral characteristics.

1.1.1 Advantages of Hyperspectral Images

High spectral resolution

Compared to multi-spectral images where the spectral distribution is uneven and contains 10 – 30 bands, HSI has narrow band length of 4 -10nm and has a continuous spectral distribution and has above 100 bands. Thus providing a rich set of information for accurate classification.

Image Segmentation

Since the bands in HSI are very narrow i.e 4-10nm and has a continuous spectral distribution. We can easily partition the images into adjacent bands and create multiple segments. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.

No need for domain knowledge

Since it has high spectral resolution, the discriminators needed to perform classification are within image itself. So a good feature extractor can perform classification without having any prior domain knowledge.

1.1.2 Disadvantages of Hyper-spectral

ImagesHigh Dimensional Data

High Dimension means that the number of dimensions are staggeringly high and hence calculations become extremely difficult. With high dimensional data, the number of features can exceed the number of observations. In HSI, there can be more than 100 bands per pixel.

Storage Requirements

Significant data storage is required for a single image. As an example, one image of size 1024*1024 pixels would add 1048576 samples to the data set. Using about 200 spectral channels, the size of the raw data received for one image, from the sensor would be over 1 GB.

Limited Available Training Samples

The use of hyperspectral sensors is not widely spread among the common people due to its high cost and hence limited training samples are available for use. And as a consequence it suffers from Hughes Phenomenon.

1.1.3 Definitions and Acronyms

High Dimensional Data refers to the situation where the number of features are higher than the number of observations.

Hughes Phenomenon dictates that, with increased number of hyperspectral bands the number of samples (i.e., training pixels) required to maintain minimum statistical confidence and functionality in hyperspectral data for classification purposes grows exponentially, making it very difficult to address accuracy.

List of acronyms:

HSI - Hyperspectral Image

HSIC – Hyperspectral Image Classification

1.2 Introduction to Deep Learning (DL)

Deep Learning (also known as Deep Structured Learning) is part of a broader family of Machine Learning methods that are based on Artificial Neural Networks with representation learning. Deep Learning uses multiple hierarchical layers to progressively extract higher-level features from the input.

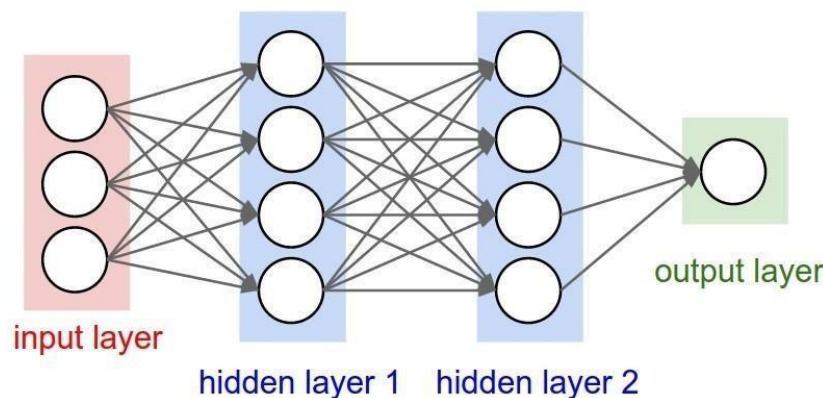


Figure 1.2 Neural Network

In the above figure 1.2, we see the different layers in neural network. The first few layers are used

to represent simple features like edges and texture and with each passing layer the features are extracted and abstracted to represent more complex features. The term “Deep” in Deep Learning usually refers to the number of layers in a neural network while shallow neural networks consists of 2-30 layers, Deep Neural Networks can have up to 150 hidden layers.

1.2.1 Advantages of Deep Learning

Automatic Feature Recognition

Deep Learning techniques can extract informative features from the original data via a series of hierarchical layers. Specifically, initial layers extract some simple features like texture and edge information. Furthermore, the deeper layers are able to represent more complicated features with the help of previous layers. The learning process is totally automatic, which makes deep learning more suitable for coping with the wide varieties of situations.

Higher Classification Accuracy

Compared to other classification algorithms (e.g., neural networks , support vector machines (SVM) , multi-nomial logistic regression and dynamic or random sub space), it has been shown that Deep Learning methods outperform these traditional Machine Learning methods.

Better performance on complex data

Complex features usually have high computation cost and suffer from high dimensionality. Feature Extraction is an effective way in both reducing high dimensionality and computation cost. Deep Learning techniques are known to be effective feature extractors and thus are more suitable when the characteristics of the data is complex.

1.2.2 Disadvantages of Deep Learning

High Chance of Overfitting

Overfitting is a concept in data science, which occurs when a statistical model fits exactly against its training data. When the model memorizes the noise and fits too closely to the training set, the model becomes “overfitted” to the extent that it negatively impacts the performance of the model and hence unable to generalize well to new data.

Deep Learning techniques are known to overfit the data when there are limited available training samples.

Unclear Implementation

Deep Learning techniques tends to learn features automatically so it's difficult to see the evolution of a system over time. And as the number of layers increases in the network, it also becomes harder to describe the model and it becomes difficult to understand it. Significant computing power in terms of Memory, CPU and GPU is needed for substantial computing power to train a Deep Learning model. This is due to the requirement of large training samples and complex architecture of the deep neural network.

1.2.3 Definitions and Acronyms

Machine Learning is the study of algorithms that provide a system the ability to learn and improve from experience without being explicitly programmed.

Deep Learning is a sub-branch of Machine Learning that gives systems the ability to learn like human beings through examples by the use of Artificial Neural Networks.

Artificial Neural Networks are systems vaguely inspired by the biological neural networks that constitute the animal brain. It consists of a collection of connected units called artificial neurons which loosely model the biological brain. Each of these neurons are connected to each other and they receive, process and pass information to other neurons.

Feature Extraction is a technique where an initial set of raw data is used to derive values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps.

List of acronyms:

ML - Machine Learning

DL - Deep Learning

ANN - Artificial Neural Network

FE - Feature Extraction

1.3 Problem Statement

The main challenge in this task is the high dimensionality of the data, as well as the presence of noise and spectral variability, which can make it difficult to accurately classify individual pixels. The objective of hyperspectral image classification in a single point is to develop algorithms that can effectively exploit the spectral information at the pixel level to achieve high accuracy and robustness in classifying individual pixels.

1.4 Objectives

The objectives for a hyperspectral image classification project can vary depending on the specific application and goals of the project. However, some general objectives for such a project may include:

- Develop an accurate and efficient algorithm for hyperspectral image classification that can classify each pixel in the image into one of several predefined classes.
- Evaluate the performance of the classification algorithm using appropriate metrics such as overall accuracy, confusion matrix, and Kappa coefficient.
- Investigate different feature extraction and dimensionality reduction techniques to reduce the computational cost and improve the classification accuracy.
- Investigate different machine learning and deep learning models such as Support Vector Machines (SVM), Random Forest (RF), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) for hyperspectral image classification.
- Perform data preprocessing, including noise reduction, atmospheric correction, and image registration, to improve the quality of the hyperspectral data and enhance the accuracy of the classification.

1.5 Purpose

Hyperspectral Images is being researched for its use in remote sensing. Many models have been proposed which do not fully capture the characteristics of hyperspectral images. The purpose of this project is to develop a effective neural network model that can take into consideration the spectral-spatial, spatial and the spectral characteristics.

1.6 Scope

Using Hyperspectral Image Classification, system will enable the government officials to monitor the growth of urban area by finding the distribution of man made materials (e.g buildings, roads etc) and analyzing the distribution of vegetation.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

Paper 1: Zilong Zhong, Jonathan Li, Lingfei Ma, Han Jiang, He Zhao worked on **DEEP RESIDUAL NETWORKS FOR HYPERSPECTRAL IMAGE CLASSIFICATION**. Deep neural networks can learn deep feature representation for hyperspectral image (HSI) interpretation and achieve high classification accuracy in different datasets. However, counter-intuitively, the classification performance of deep learning models degrades as their depth increases. To study the influence of deep learning model son HSI classification accuracy, this paper applied ResNets and CNNs with different depth and width using two challenging datasets. Moreover, they tested the effectiveness of batch normalization as a regularization method with different model settings. The experimental results demonstrate that ResNets mitigate the declining-accuracy. They adopted the improved residual networks for HSI classification. In this work it was shown that batch normalization enhances the HSI interpretation performance of both CNNs and ResNets. The ResNets have alleviated but not fully overcome the decreasing-accuracy effect. the proposed ResNets have learned a more discriminative representation of HSIs than those of CNNs. Moreover, the deep learning models with wider architectures tend to deliver higher classification accuracy under the same regularization methods, but the increase is not obvious when kernel number is larger than 24. It is worth noting that the ResNets with 4 layers perform the best in both HSI datasets, owing to small numbers of training samples and land cover categories.

Paper 2: A. N. Abbasi and Mingyi He have worked on **Convolutional Neural Network with PCA and Batch Normalization for Hyperspectral Image Classification**. Proposed a deep learning method which uses spectral reduction as preprocessing and batch normalization in every layer of the deep network. The PCA is used to reduce the spectral dimensionality, to avoid overfitting and regularize the network, batch normalization and dropout combination is used which also increases the accuracy. Moreover the data augmentation used to create further variation (flipping and rotating) in labeled training data for classification improvement and also

enhancement of data for the training to overcome the limited training samples to some extent. The training process is regularized and the overfitting is avoided by using combination of batch normalization and dropout. Moreover, oversampling and augmentation in training data is used to expand the training data and to create some variation in available training data. Finally the experimental results demonstrated the performance of their method in comparison to other methods especially for hyperspectral classification tasks.

Paper 3: S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri worked on **Hybridsn: Exploring 3-d–2-d CNN feature hierarchy for hyperspectral image classification..** The use of CNN for HSI classification is also visible in recent works. These approaches are mostly based on 2D CNN. Whereas, the HSI classification performance is highly dependent on both spatial and spectral information. Very few methods have utilized the 3D CNN because of increased computational complexity. This paper proposes a Hybrid Spectral Convolutional Neural Network for HSI classification. Basically, the HybridSN is a spectral-spatial 3D-CNN followed by spatial 2D-CNN. The 3D-CNN facilitates the joint spatial-spectral feature representation from a stack of spectral bands. The 2D-CNN on top of the 3D-CNN further learns more abstract level spatial representation. Moreover, the use of hybrid CNNs reduces the complexity of the model compared to 3D-CNN alone. HybridSN for HSI classification model basically combines the complementary information of spatial-spectral and spectral in the form of 3D and 2D convolutions, respectively. The experiment is conducted over three benchmark datasets. The proposed model made use of a very large patch size to make up for its simplistic network.

Paper 4: Konstantinos Makantasis, Konstantinos Karantzas, Anastasios Doulami and Nikolaos Doulamis worked on **DEEP SUPERVISED LEARNING FOR HYPERSPECTRAL DATA CLASSIFICATION THROUGH CONVOLUTIONAL NEURAL NETWORKS.** Proposed a deep learning based classification method that hierarchically constructs high-level features. Their method exploits a 2-Dimensional Convolutional Neural Network to encode pixel's spatial information and a Multi-Layer Perceptron to conduct the classification task. For dimensionality reduction, Randomized PCA (R-PCA) is introduced along the spectral dimension to condense the whole image. During the

experimentation process run on widely-used hyperspectral datasets, the amount of information is preserved by using the first 10 to 30 principal components, reducing this way up to 15 times the dimensionality of the raw input. Then take a 5x5 patch for each pixel in the image. The CNN consists of 2 Convolutional Layers that extracts the spatial features. And two fully connected layers that fuse the features of the 2DCNN.

Paper 5: Yanan Luo, Jie Zou, Chengfei Yao, Tao Li, Gang Bais worked on **HSI-CNN: A Novel Convolution Neural Network for Hyperspectral Image**. Proposed a network structure that possesses reshape 1-dimension array data to image-like 2-dimension matrix, which can make complete use of the spectral and spatial information hidden in the original data. By referring to the idea of cube selection of input data, 8-neighbor pixels labeled as the central pixel's categories are used as the input of their model. HSI-CNN. The network model is composed of a 3D Convolution, 1 reshape layer, 2D - convolution layers, 1 pooling layer and 3 fully connected layers where the last layer is softmax layer. The most important part of the model is the reshape layer. Since each feature vector is convoluted from the same original data with different convolution kernels which make these vectors have different representations of the particular features, so there is a strong correlation between these vectors. All the original vectors are directly stitched into a matrix whose height is constant and whose width changes from the original one to the number of feature vectors. After the reshape operation the data can be input as normal 2D image classification, in this way the scarce HSI data amounts to a multiplied increase.

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

In this chapter discuss about the hardware and software requirements for the system.

3.1 Hardware Requirements

This program can run in most systems as long as all the necessary libraries are installed. The notebook can also be opened in Google's cloud platform Colaboratory. A system with good CPU and GPU is ideal.

- Processor : Intel Core i5 or above
- RAM : 8GB
- Hard Disk Space : 100GB
- Operating System: Windows, Linux

3.2 Software Requirements

- Language : Python
- Libraries : numpy, pandas, sklearn, scipy, spectral, matplotlib, keras
- IDE : Jupyter Notebook, Google Colaboratory

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

4.1 System Architecture

The architecture diagram is shown in the figure 4.1. It can be classified into the following steps. Data collection, data loading, scaling, dimension reduction, patching, spectral-spatial feature extraction, spatial feature extraction and spectral feature extraction, feature fusion, predictions.

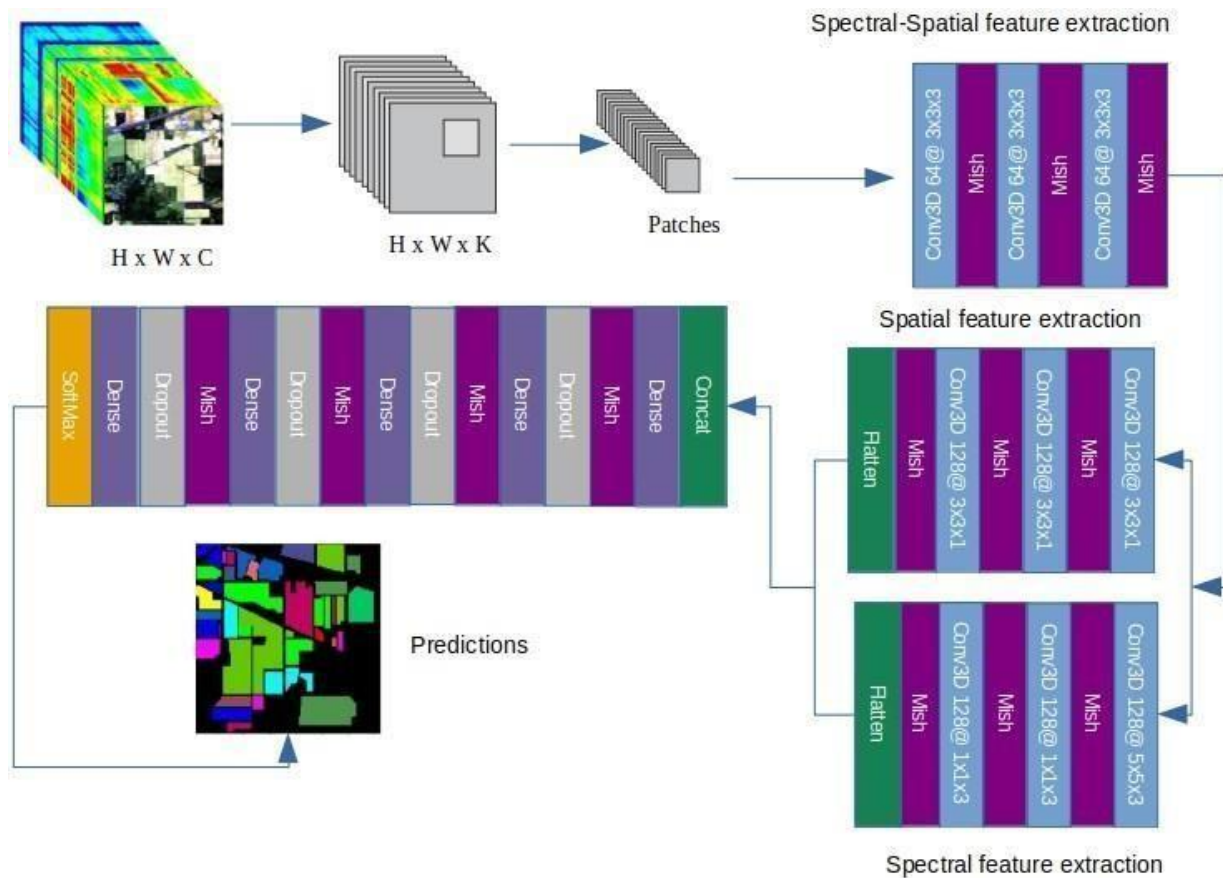


Fig 4.1 Proposed System architecture diagram

Convolutional Neural Networks

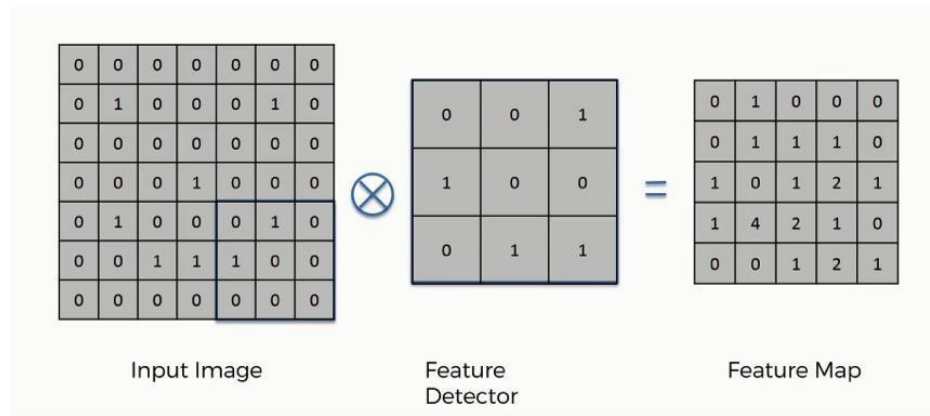


Fig 4.2 Convolution Operation

A Convolutional Neural Network is a Deep Learning algorithm which can take an image as input, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The preprocessing required in a ConvNet is much lower than other classification algorithms. A convolution in CNN is a filter applied to different positions of the input to generate a feature map which represents patterns in the image. This feature map is then passed on to the next layer which will then combine and extract more abstract feature like edges, so on and so forth. In our work we make use of 3D convolution, which applies a filter of size $F @ K1i \times K2i \times K3i$ to the input image cube where F represents the number of filters, i represents the i -th layer in the network.

Activation Function

In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs. A standard integrated circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on the input. Activation functions are a critical part of the design of a neural network. The choice of activation function in the hidden layer will control how well the network model learns the training dataset. The choice of activation function in the output layer will define the type of predictions the model can make. As such, a careful choice of activation function must be made for each deep learning neural network project.

Mish

It is an activation function that is defined as

$$f(x) = x * \tanh(\text{softmax}(x))$$

Where,

$$\text{softplus}(x) = \ln(1 + e^x)$$

Where,

x is the input

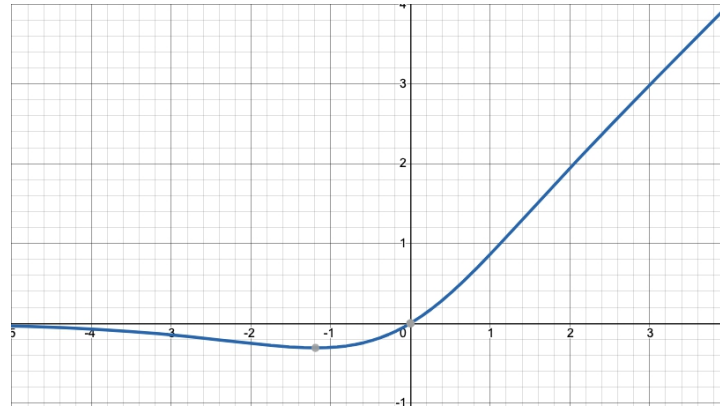


Fig 4.3 Mapping of Mish activation function

SoftMax

It is given by the expression.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Where,

Z is the input vector

$e^{(z_i)}$ = Standard exponential function for input vector

K = number of classes in the multi-class classifier

$e^{(z_j)}$ = standard exponential function for output vector

It is an activation function that takes in input and outputs a probability distribution as shown in the below figure 4.3.

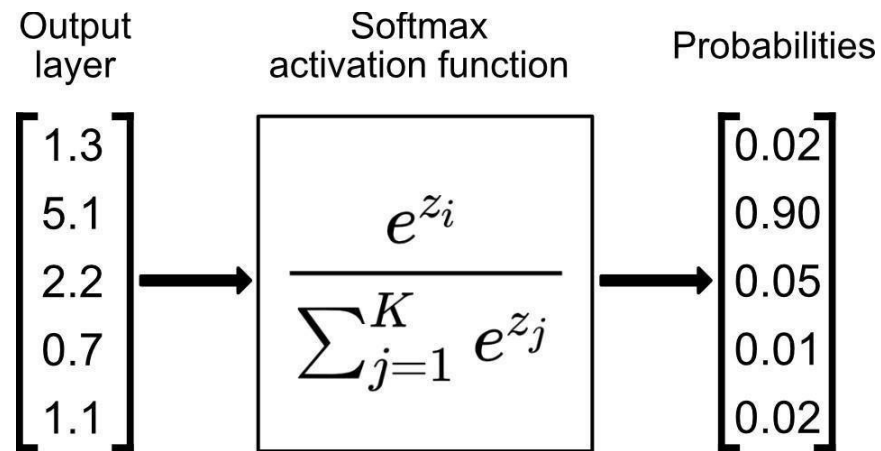


Fig 4.4 Input and output Mapping of SoftMax activation function

Dense Layer

Dense Layer or fully connected layer is the final layer of the CNN. In Dense Layer all the inputs of the previous layer are fully connected to each neuron in the current layer. In our project we have included 4 dense layers which can be seen in the architecture diagram in figure 4.1. The Dense layer is applied so that high level features extracted from the CNN, mix and match together and hence all the features are considered when making the prediction.

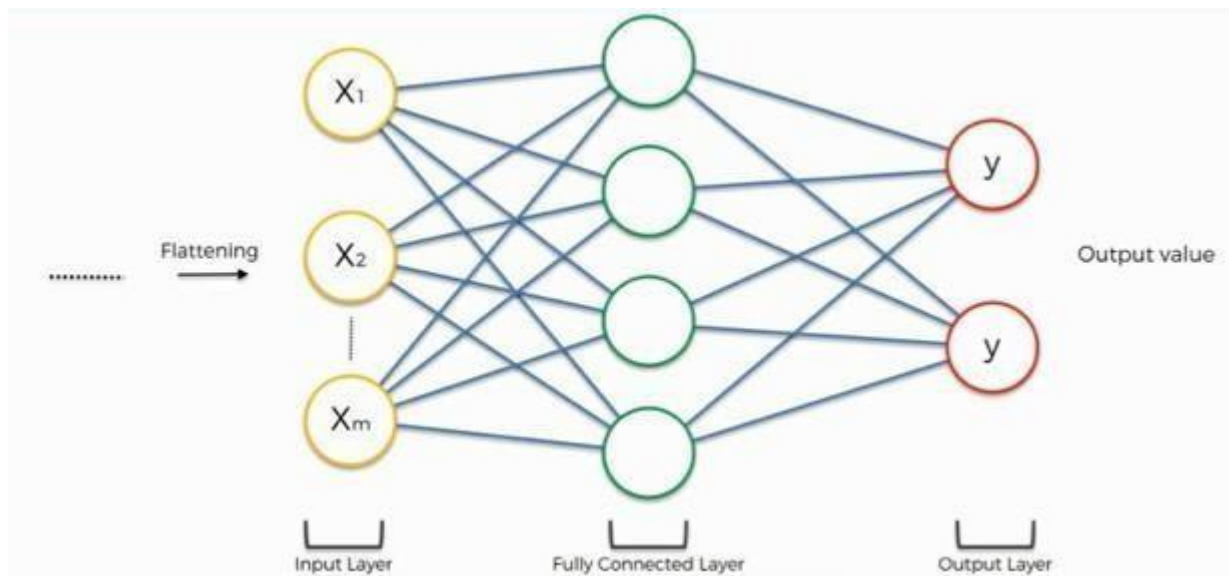


Fig 4.5 Fully Connected Layer

Dropout Layer

Deep Learning models are known to enter a situation called overfitting. Over fitting is a situation where the model learns the training data and the noise too well in a way that it affects the performance negatively while predicting new data. To alleviate this issue we strategically introduce a dropout layer in the neural network. Dropout is a generalization technique used during training that randomly sets the output of some nodes of a layer to zero and hence the name dropout. This introduces randomness in the data and forces the model to generalize.

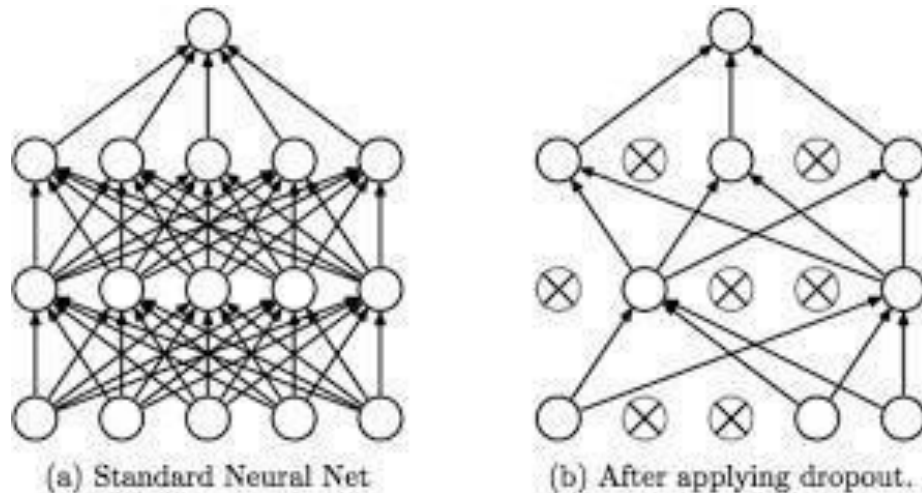


Fig 4.6 Dropout Layer

4.2 Data Flow Diagram

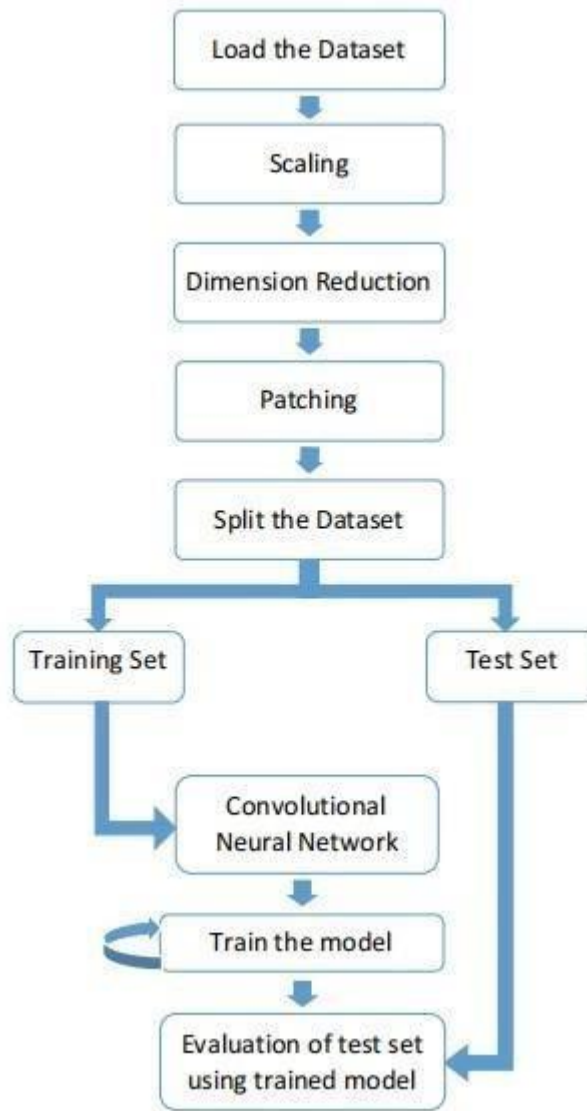


Fig. 4.7 Data flow diagram

In our proposed method, the image is in the form of a Matlab file. First load the image into python as an array. Then apply the MinMax scaling to the image as previously discussed. This changes the original range of each band in the image to a range of 0 - 1.0. Then send the image to the dimension reduction module implemented using PCA, where the image's dimension is reduced along with the spectral dimension. An image of dimension $H \times W \times C$ where H is height, W is the Width and C is the channels is reduced to $H \times W \times K$, where K represents the number of principal components. Afterwards, the image is passed to the patching

module where in a data cube is generated for each pixel with the pixel in the centre and the neighbouring pixels are selected to ensure CNN takes into consideration both the spectral and spatial characteristics. The labeled data is then split into training and testing samples. The training samples is then fed into the CNN. The network is trained for a certain number of epochs, after which it is tested using the testing samples.

4.3 Modules/APIs Used

Modules

The in-built packages in Python are called modules. A module can define functions, classes and variables. Following are the modules used in our work:

Module Name : numpy

numpy (Numeric Python) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Functions:

- a. reshape():** By reshaping we can add or remove dimensions or change number of elements in each dimension.
- b. zeros():** function returns a new array of given shape and type, with zeros.
- c. cumsum():** function is used to compute the cumulative sum of array elements over a given axis.
- d. unique():** This function returns an array of unique elements in the input array.
- e. ravel():** This function returns a 1-D array, containing the elements of the input.
- f. argmax():** This function returns indices of the max element of the array in a particular axis.
- g. diag():** This function extracts the diagonal elements in the array.
- h. sum():** This function returns the sum of an array across a axis.
- i. mean():** Returns the average of the input.
- j. arange():** Returns an array if the specified length.

Module Name : scipy.sio

Scipy (Scientific Python) is a Python library used for scientific computing and technical computing.

Functions:

a. loadmat(): A function used to load the Matlab file into a python array.

Module Name : matplotlib.pyplot

matplotlib.pyplot is a plotting library used for 2D graphics in Python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

Functions:

a. show(): The show() function in pyplot module of matplotlib library is used to display all figures.

Module Name : spectral

Spectral Python (SPy) is a pure Python module for processing hyperspectral image data.

Functions:

a. imshow(): this function is used to display the color image from the hyperspectral data.

Module Name : sklearn

Sklearn (scikit learn) is a machine learning library for the Python programming language.

Functions:

a. MinMaxScaler(): This function transforms each feature in the input by scaling each feature to a given range.

b. PCA(): Applies Linear dimensionality reduction to the data so as to project it to a lower dimensional space.

c. test_train_split(): This function splits input data into a training set and a testing set.

Module Name : keras

Keras is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models.

Functions:

- a. **Conv3D()**: This function applies a 3 Dimensional Convolution to the input.
- b. **Dropout()**: This function applies Dropout to the input.
- c. **Flatten()**: This function takes a n-dimensional input and returns a 1-Dimensional array.
- d. **Concatenate()**: This function concatenates arrays of similar dimensions.
- e. **Dense()**: This function applies the fully connected layer to the input.

CHAPTER 5

IMPLEMENTATION

CHAPTER 5

IMPLEMENTATION

1) Scaling module

Scaling is a technique used to change the range of the entire dataset. It is applied so that the machine learning algorithm does not consider higher values of the input with more importance and lower values of the input with lesser importance. The expression is as show below.

$$X_{\text{new}} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$

=

Where,

X represents the features,

X_i an instance of the the feature.

2) Dimension Reduction module

PCA is used to reduce the dimensions of the dataset. Thus reducing computation costs. It takes the entire dataset as input and converts them to principal components with successively high variance. Here we take a HSI with dimensions of $H \times W \times C$ where H stands for height, W for width and C is the number of spectral channels and convert the image to $H \times W \times K$ where K is the number of principal components.

3) Patching module

To consider both the spectral and spatial characteristics of the image, the input is sent in the form a data cube. For each pixel which needs to be classified, Consider that pixel as the center of cube and populate the remainder of the cube with its surrounding pixels. It is given by the expression shown below.

$$M = (W - 1) / 2$$
$$P(I,J) = \text{HSI}(I-M:I+M, J-M:j+M)$$

Where,

W is the window size,

M is the margin,

P is the image patch and (I,J) is the position of the pixel in the HSI

4) Convolutional Neural Network

The Convolutional Neural Network that we designed can be seen in figure 4.1. It consists of 3 3D-Convolutional Layers with kernel values 64@3 x 3 x 3. The activation function used is mish as described previously. The output of this 3 convolution layers is then passed to two sub-networks one focuses on extracting spatial characteristics with kernel values 128@ 3 x 3 x 1 and the other focuses on extracting the spectral characteristics with kernel values 128@ 1 x 1 x 3. The outputs of these two networks are then flattened to a linear array and then concatenated. This array is then passes to 4 fully connected layer as described in the figure 4.1. Dropout is placed here so that the model does not overfit. Finally it is passed to the final layer whose activation function is SoftMax which outputs the probability distribution for each class in the dataset.

5.1 CODE SNIPPETS:

1) Loading of dataset

```

1 def loadDataSet(name):
2     if name == 'IP':
3         img = sio.loadmat('Indian_pines_corrected.mat')['indian_pines_corrected']
4         gt = sio.loadmat('Indian_pines_gt.mat')['indian_pines_gt']
5         labels = ['unknown', 'Alfalfa', 'Corn-notill', 'Corn-mintill', 'Corn',
6                 'Grass-pasture', 'Grass-trees', 'Grass-pasture-mowed',
7                 'Hay-windrowed', 'Oats', 'Soybean-notill', 'Soybean-mintill',
8                 'Soybean-clean', 'Wheat', 'Woods', 'Buildings-Grass-Trees-Drives',
9                 'Stone-Steel-Towers']
10        rgb_bands = (29, 19, 9)
11        spatial_resolution = 20
12
13    elif name == 'SAL':
14        img = sio.loadmat('Salinas_corrected.mat')['salinas_corrected']
15        gt = sio.loadmat('Salinas_gt.mat')['salinas_gt']
16        labels = ['unknown', 'Brocoli_green_weeds_1', 'Brocoli_green_weeds_2',
17                'Fallow', 'Fallow_rough_plow', 'Fallow_smooth', 'Stubble',
18                'Celery', 'Grapes_untrained', 'Soli_vinyard_develop', 'Corn_senesced_green_weeds',
19                'Lettuce_romaine_4wk', 'Lettuce_romaine_5wk', 'Lettuce_romaine_6wk', 'Lettuce_romaine_7wk',
20                'Vinyard_untrained', 'Vinyard_vertical_trellis',]
21        rgb_bands = (29, 19, 9)
22        spatial_resolution = 3.7
23
24    elif name == 'PU':
25        img = sio.loadmat('PaviaU.mat')['paviaU']
26        gt = sio.loadmat('PaviaU_gt.mat')['paviaU_gt']
27        labels = ['unknown', 'Asphalt', 'Meadows', 'Gravel', 'Trees', 'Painted metal sheets',
28                'Bare Soil', 'Bitumen', 'Self-Blocking Bricks', 'Shadows',]
29        rgb_bands = (53, 31, 8)
30        spatial_resolution = 1.3
31
32    elif name == 'KSC':
33        img = sio.loadmat('KSC.mat')['KSC']
34        gt = sio.loadmat('KSC_gt.mat')['KSC_gt']
35        labels = ['Undefined', 'Scrub', 'Willow swamp', 'Cabbage palm/hammock',

```

0s completed at 15:26

Pseudo Code 1: Loading of Dataset

Loading of dataset involves importing dataset into the CNN network. Hyperspectral image classification pipeline provides functions for reading and importing hyperspectral image and associated ground truth labels from an external source or file. It typically includes implementation of various file formats and may also provide additional functions for preprocessing and partitioning the dataset into training and testing sets.

2) Apply Scaling

```

1 def applyScaling(X):
2     newX = np.reshape(X, (-1, X.shape[2]))
3     sc = MinMaxScaler()
4     newX = sc.fit_transform(newX)
5     newX = np.reshape(newX, (X.shape[0], X.shape[1], X.shape[2]))
6     return newX, sc

```

Pseudo Code 2: Scaling module

The scaling module provides functions for scaling the spectral reflectance values of the input hyperspectral data. It typically includes implementation of common scaling techniques such as Min-Max scaling normalization and may also provide additional functions for preprocessing, including partitioning the data into training and testing sets.

3) Apply PCA

```

1 def applyPCA(X, numComponents=75):
2     newX = np.reshape(X, (-1, X.shape[2]))
3     pca = PCA(n_components=numComponents, whiten=True)
4     newX = pca.fit_transform(newX)
5     newX = np.reshape(newX, (X.shape[0], X.shape[1], numComponents))
6     return newX, pca

```

Pseudo Code 3: PCA module

The PCA (Principal Component Analysis) module provides functions for performing PCA on the input hyperspectral data. It typically includes implementation of common PCA techniques, such as dimensionality reduction and feature extraction, and may also provide additional functions for visualizing the PCA results and selecting the optimal number of principal components to retain.

4) Apply Padding

```

1 def padWithZeros(X, margin=2):
2     newX = np.zeros((X.shape[0] + 2 * margin, X.shape[1] + 2 * margin, X.shape[2]))
3     x_offset = margin
4     y_offset = margin
5     newX[x_offset:X.shape[0] + x_offset, y_offset:X.shape[1] + y_offset, :] = X
6     return newX

```

Pseudo Code 4: Padding module

The padding module provides functions for adding additional pixels to the input hyperspectral image to increase its size or to align it with other images. It typically includes implementation of various padding techniques, including zero-padding, edge-padding, and reflection-padding, and may also provide additional functions for cropping the image and adjusting its size. Padding is a critical step in hyperspectral image classification as it can prevent edge effects and improve the accuracy and stability of the classification algorithm.

5) Create Patches

```

1 def createPatches(X, y, windowSize=5, removeZeroLabels = True):
2     margin = int((windowSize - 1) / 2)
3     zeroPaddedX = padWithZeros(X, margin=margin)
4
5     # split patches
6     patchesData = np.zeros((X.shape[0] * X.shape[1], windowSize, windowSize, X.shape[2]))
7     patchesLabels = np.zeros((X.shape[0] * X.shape[1]))
8     patchIndex = 0
9
10    for r in range(margin, zeroPaddedX.shape[0] - margin):
11        for c in range(margin, zeroPaddedX.shape[1] - margin):
12            patch = zeroPaddedX[r - margin:r + margin + 1, c - margin:c + margin + 1]
13            patchesData[patchIndex, :, :, :] = patch
14            patchesLabels[patchIndex] = y[r - margin, c - margin]
15            patchIndex = patchIndex + 1
16
17    if removeZeroLabels:
18        patchesData = patchesData[patchesLabels>0, :, :, :]
19        patchesLabels = patchesLabels[patchesLabels>0]
20        patchesLabels -= 1
21
22    return patchesData, patchesLabels

```

Pseudo Code 5: Module to create patches

The patch extraction module provides the functions for dividing the input hyperspectral image into smaller patches of a predefined size. This module is typically used to create training and testing samples for machine learning algorithms that require input data in the form of smaller image patches. The patch extraction module may also include additional functions for visualizing the extracted patches and selecting the optimal patch size for the classification algorithm.

6) Plot Variance Ratio

```

1 def plotExplainedVarianceRatio(X, numComponents=75):
2     #Find best number of principal components to apply to the dataset
3     newX = np.reshape(X, (-1, X.shape[2]))
4     pca = PCA(n_components=numComponents, whiten=True)
5     newX = pca.fit_transform(newX)
6     ev = pca.explained_variance_ratio_
7
8     #plot a chart to see Cumulative variance ratio
9     plt.figure(figsize=(12, 6))
10    plt.plot(np.cumsum(ev))
11    plt.xlabel('Number of components')
12    plt.ylabel('Cumulative explained variance')
13
14    plt.show()

```

Pseudo Code 6: Module to plot Variance Ratio

The variance ratio plotting module in a hyperspectral image classification pipeline provides functions for visualizing the amount of variance captured by each principal component generated from the PCA module. It typically includes implementation of a scree plot or variance ratio plot that shows the proportion of variance explained by each principal component. This module is useful for selecting the optimal number of principal components to retain for classification and feature extraction.

7) Convolutional Neural Network

```

1 # input layer
2 input = Input((windowSize, windowSize, numPCA, 1))
3
4 x = Conv3D(filters=64, kernel_size=(3, 3, 3), padding=pad, activation=mish)(input)
5 print(f"spectral-spatial : {x.shape}")
6
7 x = Conv3D(filters=64, kernel_size=(3, 3, 3), padding=pad, activation=mish)(x)
8 print(f"spectral-spatial : {x.shape}")
9
10 x = Conv3D(filters=64, kernel_size=(3, 3, 3), padding=pad, activation=mish)(x)
11 print(f"spectral-spatial : {x.shape}")
12
13 # Spatial feature extraction sub network
14 x2 = Conv3D(filters=128, kernel_size=(3, 3, 1), padding='same', activation=mish)(x)
15 print(f"spatial : {x2.shape}")
16
17 x2 = Conv3D(filters=128, kernel_size=(3, 3, 1), padding='valid', activation=mish)(x2)
18 print(f"spatial : {x2.shape}")
19
20 x2 = Conv3D(filters=128, kernel_size=(3, 3, 1), padding='valid', activation=mish)(x2)
21 print(f"spatial : {x2.shape}")
22
23 # Spectral feature extraction sub network
24 xs = x.shape
25 x3 = Conv3D(filters=128, kernel_size=(xs[1], xs[2], 3), padding='valid', activation=mish)(x)
26 print(f"spectral : {x3.shape}")
27
28 x3 = Conv3D(filters=128, kernel_size=(1, 1, 3), padding='valid', activation=mish)(x3)

```

Pseudo Code 7.1: Convolutional Neural Network Module

```

30
31 x3 = Conv3D(filters=128, kernel_size=(1, 1, 3), padding='valid', activation=mish)(x3)
32 print(f"spectral : {x3.shape}")
33
34 x2 = Flatten()(x2)
35 x3 = Flatten()(x3)
36
37 x = concatenate([ x2, x3], axis=-1)
38 print(x.shape[1])
39
40 x = Dense(units=1024, activation=mish)(x)
41 x = Dropout(0.4)(x)
42
43 x = Dense(units=1024, activation=mish)(x)
44 x = Dropout(0.4)(x)
45
46 x = Dense(units=512, activation=mish)(x)
47 x = Dropout(0.4)(x)
48
49 x = Dense(units=512, activation=mish)(x)
50 x = Dropout(0.4)(x)
51
52 output = Dense(units=numOfClasses, activation='softmax')(x)

```

```

spectral-spatial : (None, 5, 5, 60, 64)
spectral-spatial : (None, 5, 5, 60, 64)
spectral-spatial : (None, 5, 5, 60, 64)
spatial : (None, 5, 5, 60, 128)
spatial : (None, 3, 3, 60, 128)
spatial : (None, 1, 1, 60, 128)
spectral : (None, 1, 1, 58, 128)
spectral : (None, 1, 1, 56, 128)
spectral : (None, 1, 1, 54, 128)
14592

```

Pseudo Code 7.2: Convolutional Neural Network Module

The Convolutional Neural Network (CNN) module in a hyperspectral image classification provides functions for constructing and training a deep learning model based on the CNN architecture. This module typically includes implementation of various layers such as convolutional, pooling, and fully connected layers, and may also provide additional functions for selecting the optimal hyperparameters, visualizing the model architecture, and evaluating the model's performance. CNNs are a popular approach for hyperspectral image classification due to their ability to automatically learn discriminative features from the input data.

CHAPTER 6

TESTING

CHAPTER 6

TESTING

6.1 Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirement or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. Testing is one of the most challenging steps of the software development process. It requires close attention to detail and cannot be completed if you don't apply a methodical approach

Unit Testing

Unit testing is the first stage of software testing levels. During this stage, testers evaluate individual components of the system to see if these components are functioning properly on their own. Each of the units selected is then tested to check whether or not it's fully functional. By applying unit testing in code changes, you can make sure that all issues are resolved quickly.

Integration Testing

Testers perform integration testing in the next phase of testing. Here, they test individual components of the system and then test them as a collective group. This allows software testers to determine the performance of individual components as a group and identify any problems in the interface between the modules and functions.

System Testing

System testing is the final stage of the verification process. In this stage, testers see whether or not the collective group of integrated components is performing optimally. The process is crucial for the quality life cycle, and testers strive to evaluate if the system can fulfill the quality standards and complies with all major requirements.

Acceptance Testing

Acceptance testing is the final stage of the QA test cycle. It helps evaluate if the application is ready to be released for user consumption. Typically, testers carry out this phase with the help of the representatives of the customer who test the application by using it. Therefore, they will check if the application can perform all the functions specified

6.2 Test Cases

Functionality: To check if the entered dataset name is present.				
Input	Criteria	Expected Output	Actual Output	Status
'IP'	Is Defined?	Return dataset	Returns Dataset	Pass
'UP'	Is Defined?	Return dataset	Returns Dataset	Pass
'SAL'	Is Defined?	Return dataset	Returns Dataset	Pass
'KC'	Is Defined?	Returns error message	Returns Error Message	Pass

Table 5.2.1: Test Case 1

Functionality: To check patch size.				
Input	Criteria	Expected Output	Actual Output	Status
11	Is greater than or equal to 11 ?	True	True	Pass
5	Is greater than or equal to 11 ?	False	False	Pass
15	Is greater than or equal to 11 ?	True	True	Pass

Table 5.2.2: Test Case 2

A test case is a specific set of inputs, execution conditions, and expected results used to verify the functionality and accuracy of a software application or system. It provides a detailed description of the test scenario, including the preconditions, steps, and expected outcomes.

CHAPTER 7

PERFORMANCE EVALUATION

CHAPTER 7

PERFORMANCE EVALUATION

Experiments were conducted on the publicly available Indian Pines (IP), University of Pavia (UP), and Salinas Scene (SA) datasets.

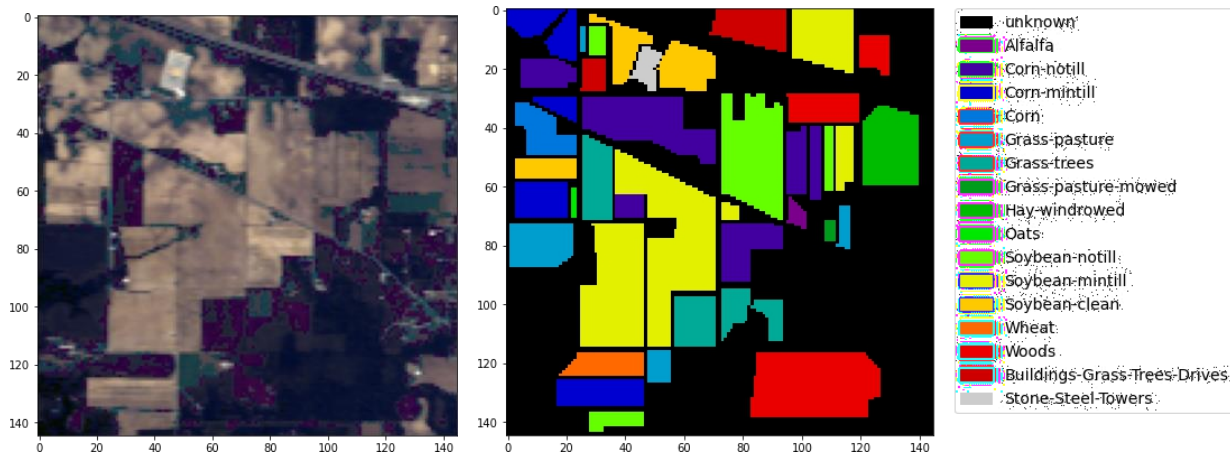


Figure 7.1 Indian Pines Color Image , Ground Truth and Legend

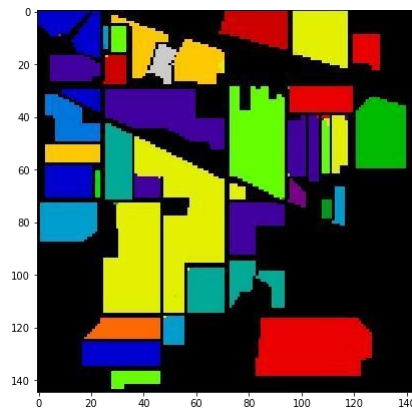


Figure 7.2 Indian Pines Predicted Ground Truth after training on 30% of the labeled data

The Indian Pines(IP) dataset was taken by NASA's Airborne Visible / Infrared Imaging Spectrometer (AVRIS) over North Western Indiana. This dataset contains 145×145 pixels, with 220 spectral bands in the wavelength range from 0.4 to 2.5 μm and is mainly composed of multiple agricultural fields with a spatial resolution of 20 meters per pixel. After removing the background pixels, 10,249 pixels were reserved, which contains 16 classes each representing the different land-cover types. Fig. 6.1 shows the color image, ground truth and the legend. Fig 6.2 shows predicted ground truth of the proposed model.

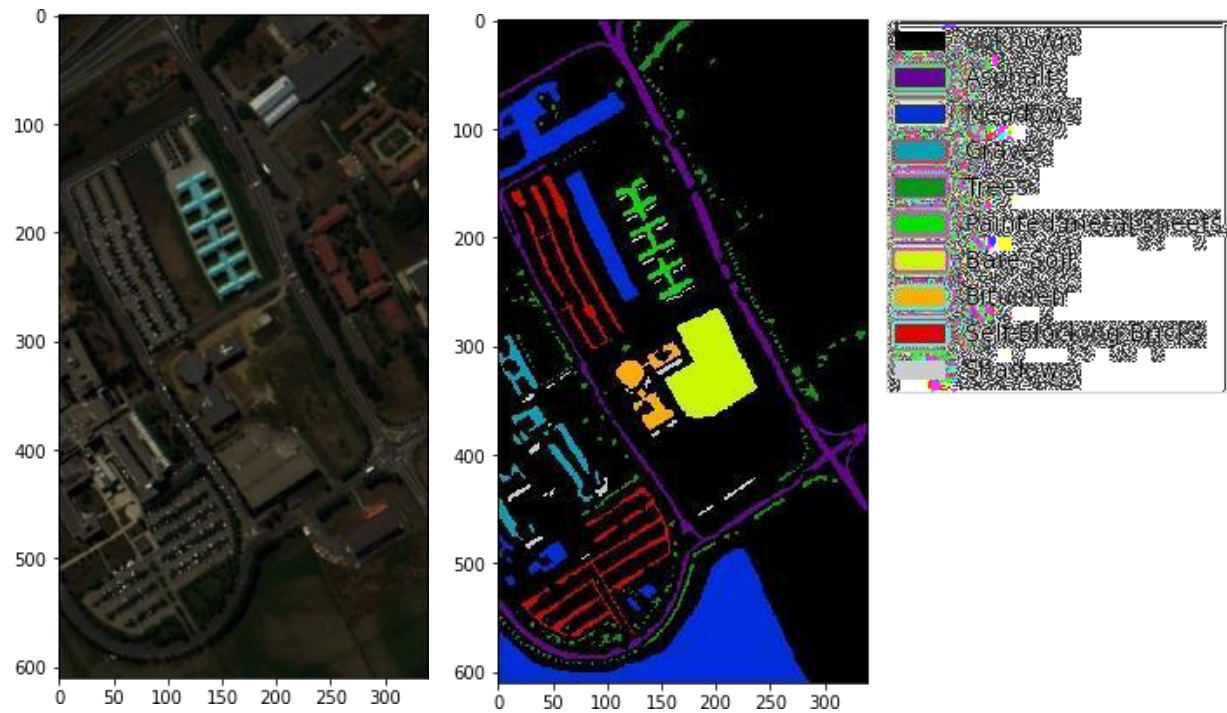


Figure 7.3 Pavia University Color Image , Ground Truth and Legend

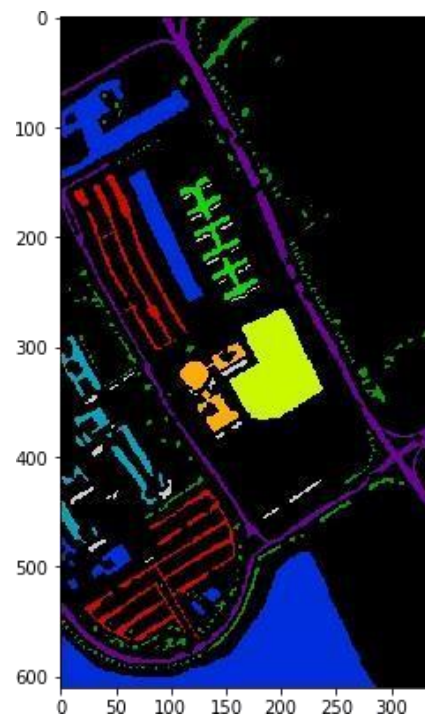


Figure 7.4 Pavia University Predicted Ground Truth using 30% of labeled data

The Pavia University dataset was acquired by the Reflective Optics System Imaging Spectrometer (ROSIS) sensor over the University of Pavia in 2001. After removing the water absorption bands only 103 bands of the data were retained, and the spectral range was from 0.43 to 0.86 μm and with a resolution of 1.3 meters per pixel. This dataset has 610×340 pixels. After removing the background pixels, 42,776 pixels were reserved, which contained nine classes representing the different land-cover types. Fig. 6.3 shows the color image, ground truth and the legend. Fig 6.4 shows predicted ground truth of the proposed model.

The Salinas scene dataset was taken by the AVIRIS sensor over Salinas Valley, California in 1998. The Salinas Scene dataset contains the image with 512×217 spatial dimension and 224 spectral bands with 16 classes. After removing the water absorption bands 200 bands were retained. After removing the background pixels 54129 pixels were reserved, which contains 16 classes representing agriculture fields. Fig. 6.5 shows the color image, ground truth and the legend. Fig 6.6 shows predicted ground truth of the proposed model.

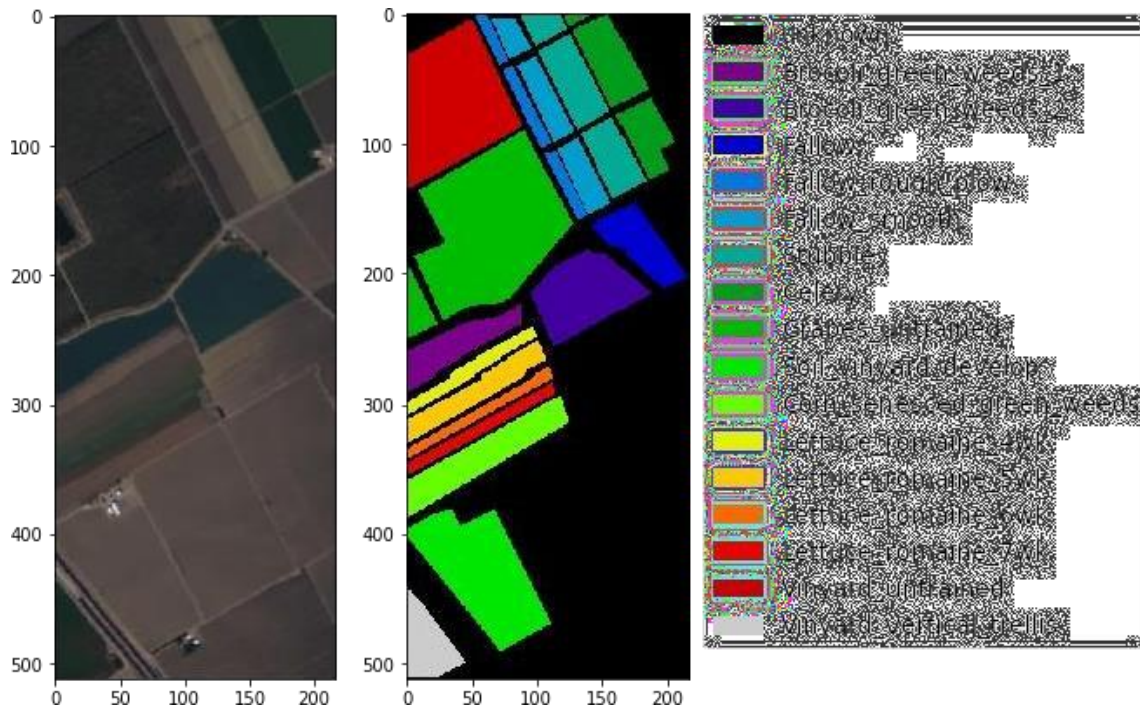


Figure 7.5 Salinas Scene Color Image , Ground Truth and Legend

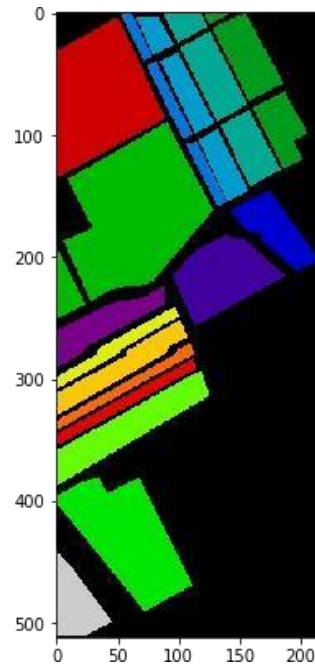


Figure 7.6 Salinas Scene Predicted Ground Truth using 30% of labeled data

To measure performance three metrics are used. **Overall Accuracy** which represents the ratio of number of correct predictions and the total number of predictions. **Each Class Accuracy** is the accuracy obtained for each category in the dataset. **Average Accuracy** is the average of Each Class Accuracy. Finally **Kappa co-efficient** which represents the agreement between the ground truth(the labeled data) and our model (predicted data).

To perform the experiments, Google Colaboratory with runtime set to GPU was used. For the optimizer, Adam was chosen with learning rate set to 0.0001 and decay set to 1e-05. The loss function used is categorical cross entropy. The model was trained for 100 epochs.

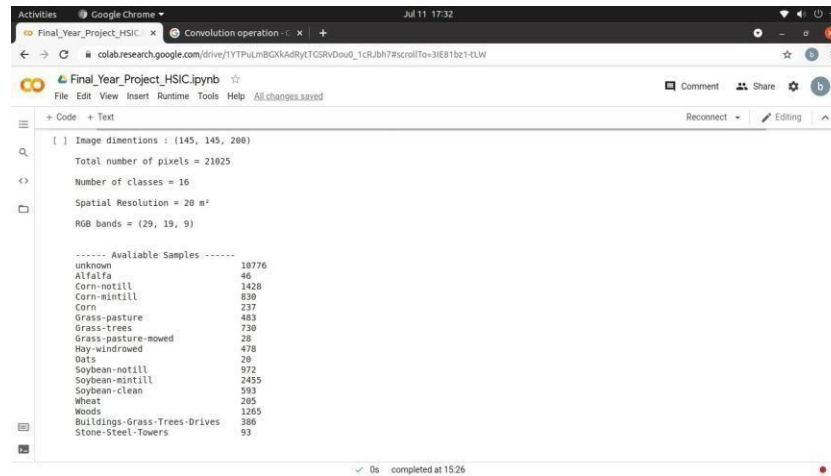
Train /Test Ratio	Indian Pines			Pavia University			Salinas Scene		
	OA	AA	KA	OA	AA	KA	OA	AA	KA
30/70	99.58	98.91	99.52	99.93	99.87	99.91	99.90	99.94	99.89
10/90	97.95	97.24	97.66	99.62	99.35	99.50	99.63	99.79	99.59

Table 7.1: Accuracy

7.1 SCREENSHOTS:

This section contains different screenshots of the project, indicating the various stages of its implementation.

1) Dataset Description



The screenshot shows a Jupyter Notebook interface with the following content:

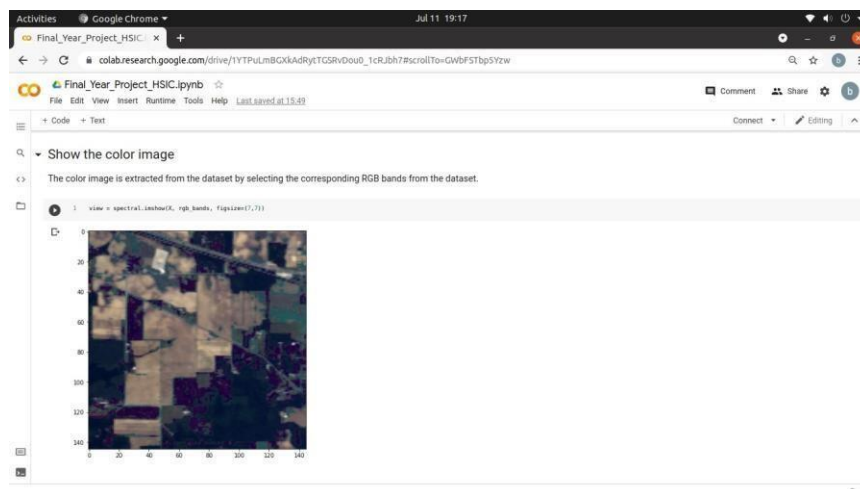
```
[ ] Image dimensions : (145, 145, 200)
Total number of pixels = 21025
Number of classes = 16
Spatial Resolution = 20 m
RGB bands = (29, 19, 9)

----- Available Samples -----
unknown          10776
Alfalfa           46
Corn-notill       1428
Corn-mintill       830
Corn              237
Grass-pasture      483
Grass-trees        730
Grass-pasture-mowed  28
Hay-windrowed      478
Oats              20
Soybean-notill     972
Soybean-mintill    2455
Soybean-clean      593
Wheat             265
Woods             1265
Buildings-Grass-Trees-Drives 386
Stone-Steel-Towers 93
```

Screenshot 1: Dataset Description and Training Samples

A dataset is a collection of data used to train and evaluate machine learning models, including hyperspectral images and ground truth labels. Training samples are a representative subset of the dataset used to train the model, with the number of samples depending on the problem complexity and dataset size

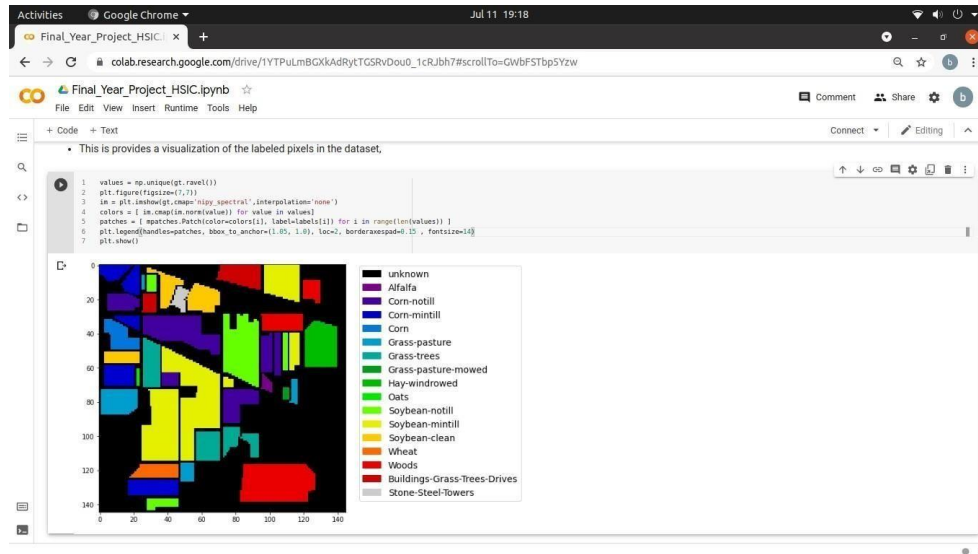
2) Color Image



Screenshot 2: Color Image of the dataset

A color image dataset is a collection of images with red, green, and blue color channels that are used for training and evaluating machine learning models. These images can be used for a variety of applications, such as object recognition, image segmentation, and classification.

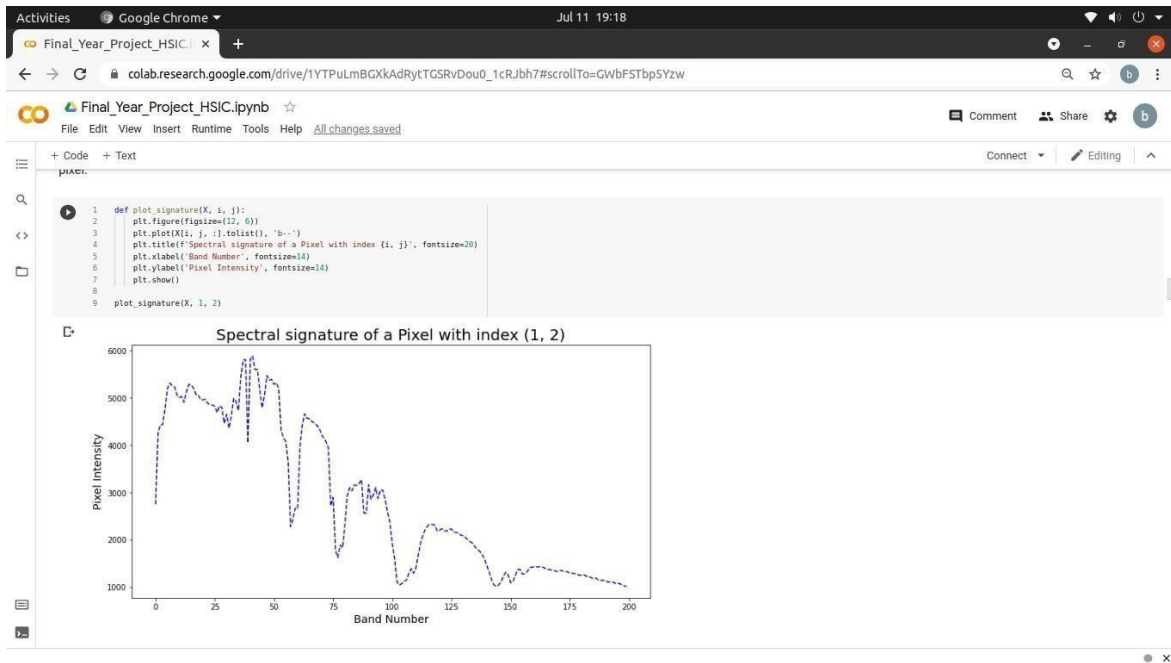
3) Ground Truth



Screenshot 3: Ground Truth with Legend

Ground truth with legend is a set of labels that indicate the true class or category of each pixel or object in an image. The legend provides a key that maps the label to the corresponding class or category, which is used to train and evaluate machine learning models for image classification, object recognition, and other computer vision tasks.

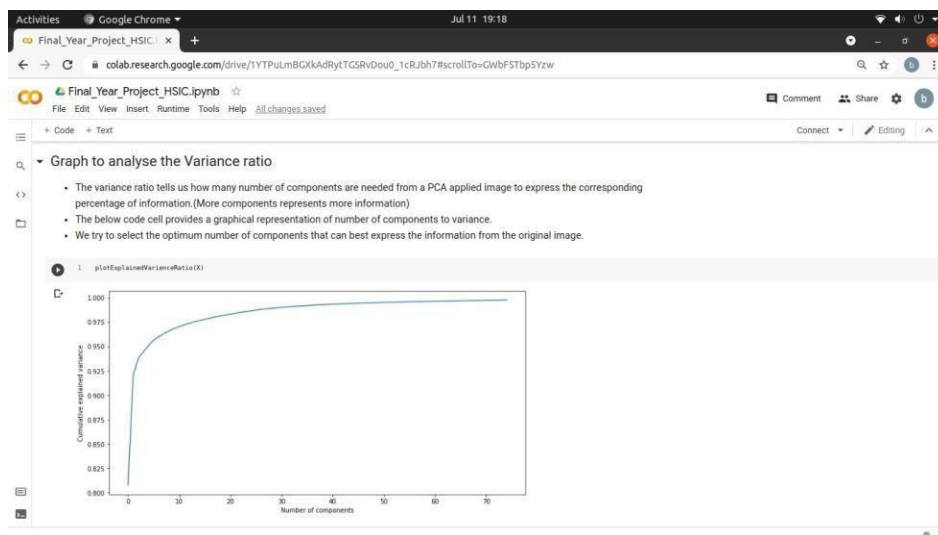
4) Visualization of Pixel's Spectral Signature



Screenshot 4: Spectral Signature of pixel

Visualization of Pixel's Spectral Signature involves plotting the intensity of light reflected from the surface at each wavelength using a line graph. This method can help identify materials and objects in hyperspectral images by analyzing the spectral properties of individual pixels.

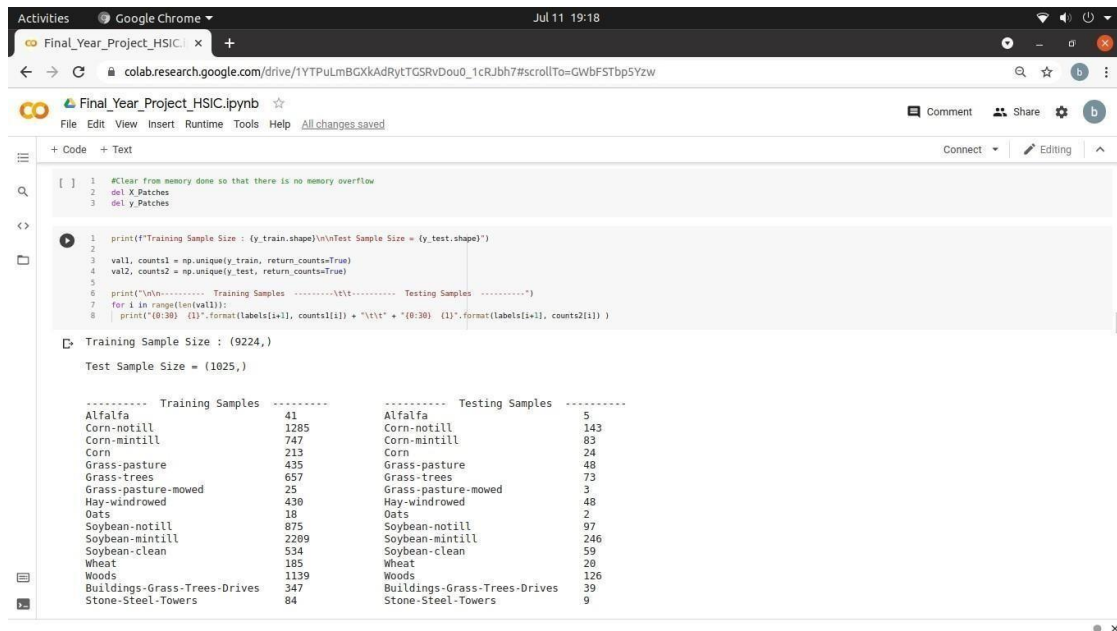
5) Variance Ratio Plot



Screenshot 5: Variance Ratio Plot

Variance Ratio Plot is a graphical technique used to determine the optimal number of spectral bands or features to use in a hyperspectral image analysis. This method involves plotting the ratio of between-class variance to within-class variance against the number of bands or features, allowing researchers to identify the most informative bands for classification.

6) Splitting Data into Training Set and Testing Set



```

1 #Clear from memory done so that there is no memory overflow
2 del X_Patches
3 del y_Patches

4 print("Training Sample Size : (y_train.shape)\nTest Sample Size = (y_test.shape)")
5 val1, count1 = np.unique(y_train, return_counts=True)
6 val2, count2 = np.unique(y_test, return_counts=True)
7 print("\n\n----- Training Samples ----- \n\n----- Testing Samples -----")
8 for i in range(len(val1)):
9     print("{0:30} {1}".format(labels[i+1], count1[i]) + "\n\t" + "{0:30} {1}".format(labels[i+1], count2[i]) )

```

Training Sample Size : (9224,)
Test Sample Size = (1025,)

Training Samples	Testing Samples
Alfalfa	41
Corn-notill	1285
Corn-mintill	747
Corn	213
Grass-pasture	435
Grass-trees	657
Grass-pasture-mowed	25
Hay-windrowed	430
Oats	18
Soybean-notill	875
Soybean-mintill	2209
Soybean-clean	534
Wheat	185
Woods	1139
Buildings-Grass-Trees-Drives	347
Stone-Steel-Towers	84

Screenshot 6: Data split into training set and testing set

Splitting the data into training and testing sets is a common practice in machine learning. The training set is used to train the model, while the testing set is used to evaluate the model's performance on new, unseen data. This technique helps to prevent overfitting and provides an estimate of how well the model will perform on new data.

7) Accuracy Report

```

1 classification, confusion, Test_loss, Test_accuracy, aa, each_acc, aa, kappa = reports(X_test, y_test, labels[i:i])
2 print()
3 print('Overall Accuracy = ' + str(aa) + "Average Accuracy = " + str(aa) + "kappa coefficient = " + str(kappa))
4
5 class_name = labels[i:i]
6
7
8 print("\n\n----- Each Class Accuracy -----")
9 for i in range(len(each_acc)):
10     print("%30s" % class_name[i], each_acc[i])

```

17/17 [=====] - 2s 39ms/step - loss: 0.0126 - accuracy: 0.9951

Overall Accuracy = 99.60975609756098
Average Accuracy = 98.5739910862964
Kappa coefficient = 99.55510076967566

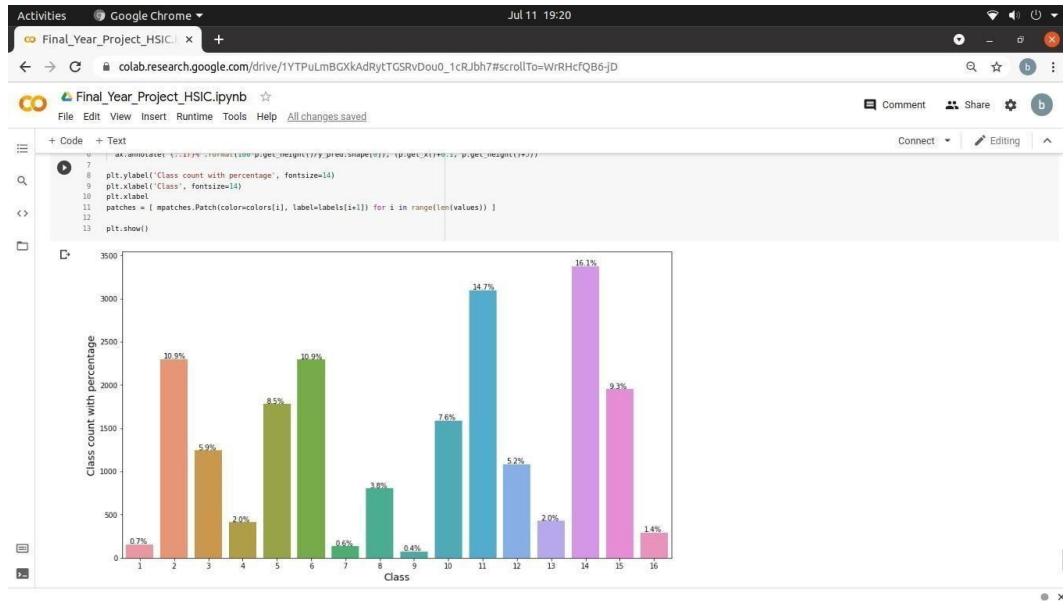
----- Each Class Accuracy -----

Alfalfa	80.0
Corn-notill	100.0
Corn-mintill	97.59036144578313
Corn	100.0
Grass-pasture	100.0
Grass-trees	100.0
Grass-pasture-mowed	100.0
Hay-windrowed	100.0
Oats	100.0
Soybean-notill	100.0
Soybean-mintill	99.59349593495935
Soybean-clean	100.0
Wheat	100.0
Woods	100.0
Buildings-Grass-Trees-Drives	100.0
Stone-Steel-Towers	100.0

Screenshot 7: Accuracy Report

An accuracy report is a summary of a machine learning model's performance on a testing dataset. It typically includes metrics such as accuracy, precision, recall, and F1 score, which measure how well the model performs in correctly classifying different classes. The accuracy report provides insight into the strengths and weaknesses of the model, and can be used to fine-tune the model and improve its performance.

8) Distribution of pixel into classes



Screenshot 8: Distribution of pixel into classes

Distribution of pixels into classes refers to the process of assigning each pixel in an image to a specific class or category based on its spectral signature or other features. The distribution of pixels into classes can be visualized using a histogram or other graphical techniques, which show the frequency of pixels in each class.

The distribution of pixels into classes is a critical step in many machine learning applications, such as hyperspectral image classification and object recognition. It helps researchers to understand the composition of image to identify areas of interest, such as regions with high concentrations of a particular material or object. By analyzing the distribution of pixels into classes, researchers can also evaluate the accuracy of the classification model and identify areas where the model may need further improvement.

CHAPTER 8

CONCLUSIONS AND FUTURE ENHANCEMENTS

CHAPTER 8

CONCLUSIONS AND FUTURE ENHANCEMENTS

CONCLUSIONS

Hyperspectral Image Classification is an emerging topic in the field of remote sensing. It has wide application ranges from agriculture, urban area analysis, mining and even in the forest area analysis. Due to the complex nature of Hyperspectral images and spectral-spatial correlation, high dimensionality and low sample size it is a challenging task for classification. In this work, a framework for Hyperspectral Image Classification using 3D Convolutional Neural Network is proposed which extracts the spectral-spatial, spatial and the spectral characteristics. It is evaluated with three datasets. The proposed method made use of the spectral and the spatial sub-networks to extract more specialized features and has obtained good performance.

FUTURE ENHANCEMENTS

- To further improve upon our model an unsupervised approach in which the model tries to learn the data needs to be further explored.
- The current implementation support only Matlab files as an input, this framework need to expanded to support other file extensions like .lan. .tif etc
- For better visualization of the output, a User Interface shall be added to invoke the implemented modules.

CHAPTER 9

REFERENCES

CHAPTER 9

REFERENCES

- 1) Makantasis, K. Karantzalos, A. Doulamis and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2015, pp. 4959-4962, doi: 10.1109/IGARSS.2015.7326945.
- 2) 4A. Ben Hamida, A. Benoit, P. Lambert and C. Ben Amar, "3-D Deep Learning Approach for Remote Sensing Image Classification," in IEEE Transactions on Geoscience and Remote Sensing, vol. 56, no. 8, pp. 4420-4434, Aug. 2018, doi: 10.1109/TGRS.2018.2818945.
- 3) M. He, B. Li, and H. Chen, "Multi-scale 3d deep convolutional neural network for hyperspectral image classification," in Proceedings of the IEEE International Conference on Image Processing (ICIP), Sept 2017, pp. 3904–3908.
- 4) S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri, "Hybridsn: Exploring 3-d–2-d cnn feature hierarchy for hyperspectral image classification," IEEE Geoscience and Remote Sensing Letters, vol. 17, no. 2, pp. 277–281, 2020.
- 5) Misra Diganta, "Mish: A self regularized non-monotonic neural activation function", arXiv preprint arXiv:1908.08681, 2019.
- 6) D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in Proceedings of the International Conference on Learning Representations (ICLR), 2015, arXiv: 1412.6980.
- 7) [http://www.ehu.eus/ccwintco/index.php/Hyperspectral Remote Sensing Scenes](http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes).

