A Project Report

on

Reduction of Overfitting in Hyper Spectral Images

Submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science & Engineering

by

C.R.ROJA SREE	**174G1A0556**
K.LAVANYA	**174G1A0534**
E. CHANDANA	**174G1A0514**
C.ANITHA	**174G1A0507**

Under the Guidance of

Mr. P. Veera Prakash, **M.Tech, (Ph.D),**
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY
(Affiliated to JNTUA, Approved by AICTE,Accrediated by NAAC with 'A' Grade & Accrediated by NBA (EEE,ECE & CSE))Rotarypuram Village, B K Samudram Mandal , Ananthapuramu- 515701

2020-2021

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY

(Affiliated to JNTUA, Approved by AICTE, Accrediated by NAAC with 'A' Grade & Accrediated by NBA (EEE,ECE & CSE)) Rotarypuram Village, B K Samudram Mandal , Ananthapuramu-515701

Certificate

This is to certify that a project report entitled REDUCTION OF OVERFITTING IN HYPER SPECTRAL IMAGES is the bonafide work carried out by **C.R.ROJA SREE** bearing Roll Number **174G1A0556, K.LAVANYA** bearing Roll Number **174G1A0534, E.CHANDANA** bearing Roll Number **174G1A0514, C.ANITHA** bearing Roll Number **174G1A0507** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** during the academic year 2020-2021.

Guide	**Head of the Department**
Mr. P. Veera Prakash, M.Tech, (Ph.D)	Dr. G. K. V. Narashima Reddy, Ph.D
Assistant Professor	Professor & HOD

Date :

EXTERNAL EXAMINAR

Place : Ananthapuramu

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we have now the opportunity to express our gratitude for all of them.

It is with immense pleasure that we would like to express our indebted gratitude to our Guide **Mr. P. Veera Prakash**, **M.Tech, (Ph.D), Assistant Professor**, **Computer Science & Engineering**, who has guided us a lot and encouraged us in every step of the project work. We thank him for the stimulating guidance, constant encouragement and constructive criticism which have made possible to bring out this project work.

We express our deep felt gratitude to **Dr. P. Chitralingappa**, **Ph.D, Associate Professor** and **Mrs. M. Soumya, M.Tech, Assistant Professor,** project coordinators valuable guidance and unstinting encouragement enabled us to accomplish our project successfully in time.

We are very much thankful to **Dr. G. K. V. Narasimha Reddy, Ph.D, Professor** and **Head of Department, Computer Science & Engineering,** for his kind support and for providing necessary facilities to carry out the work.

We wish to convey our special thanks to **Dr. G. BalaKrishna**, **Ph.D, Principal, Srinivasa Ramanujan Institute of Technology** for giving the required information in doing our project work. Not to forget, we thank all other faculty and non-teaching staff, and our friends who had directly or indirectly helped and supported us in completing our project in time.

We also express our sincere thanks to management for providing excellent facilities.

Finally, we wish to convey our gratitude to our families who fostered all the requirements and facilities that we need.

Project Associates

DECLARATION

We, Ms. C.R.Roja sree with reg no: 174G1A0556, Ms. K. Lavanya with reg no: 174G1A0534, Ms. E. Chandana with reg no: 174G1A0514, Ms. C. Anitha with reg no: 174G1A0507 students of SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, Rotarypuram, hereby declare that dissertation entitled REDUCTION OF OVERFITTING IN HYPER SPECTRAL IMAGES embodies the report of our project work carried out by us during IV year Bachelor of Technology in COMPUTER SCIENCE AND ENGINEERING, under the supervision of Mr. P. Veera Prakash, M.Tech, (Ph.D), Department of CSE, SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, ANANTHAPURAMU and this work has been submitted for the partial fulfillment of the requirements for the award of the Bachelor of Technology degree.

The results embodied in this project report have not been submitted to any other University of Institute for the award of any Degree or Diploma.

C.R.ROJA SREE	Reg No: 174G1A0556
K.LAVANYA	Reg No: 174G1A0534
E.CHANDANA	Reg No: 174G1A0514
C.ANITHA	Reg No: 174G1A0507

CONTENTS

CONTENTS

ABSTRACT

Hyper Spectral Image(HSI) provides a wide range of spectral information that can be used to address a variety of problems like crop analysis, geological mapping, mineral exploration, etc. Hyper Spectral Images are the images in which one continuous spectrum is measured for each pixel.

Hyper Spectral Images does have some training data samples which causes over fitting. In order to overcome this problem we are improving the accuracy in training data sets by using CROSS-VALIDATION . The Training data set is artificially increased by adding training samples that have been interpolated from the original training data . Hence, by applying the cross validation and data augmentation to the existing datasets, overcomes over fitting problem and improves the classification accuracy.

LIST OF FIGURES

LIST OF ABBREVIATIONS

HSI	Hyper Spectral Image
ML	Machine Learning
CV	Cross Validation
DA	Data Augumentation
CNN	Convolutional Neural Network
DL	Deep Learning
IDLE	Integrated Development Environment
DFD	Data Flow Diagram
RSI	Remote Sensing Images
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
OA	Overall Accuracy
AA	Average Accuracy

LIST OF TABLES

CHAPTER – 1

INTRODUCTION

The human eye is only able to see in a limited part of electromagnetic spectrum and can distinguish between objects based on their different spectral responses in that narrow spectral range. However, multispectral imaging sensors have been developed that are able to acquire an image in infrared and visible segments of electromagnetic spectrum (as shown in figure 1.1). This allows material identification on the basis of their unique spectral signature in a wide spectral range. Multispectral imaging exploits the property that each material has its own unique spectral signatures. Spectrum of a single pixel in a multispectral image provides information about its constituents and surface of the material.

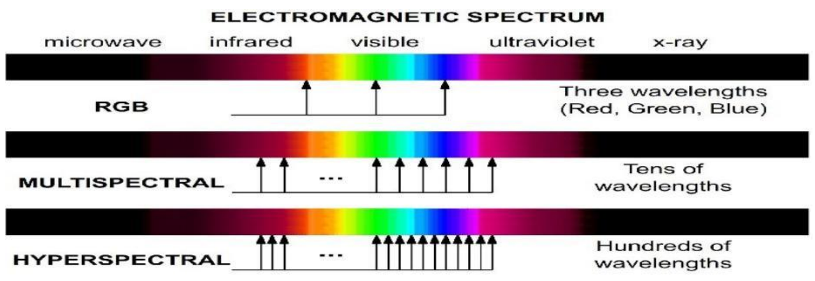


Fig 1.1: Different image visibility in Electromagnetic Spectrum

Multispectral imaging technology is being used for environment and land observation remote sensing in satellite and airborne systems. These systems acquire data in a small number of spectral bands by using parallel sensor arrays. Most of the multispectral imaging systems use three to six spectral bands with large optical band intervals, ranging from visible to near infrared regions of electromagnetic spectrum for scene observation. However, such low number of spectral bands is the limiting factor for discrimination of various materials. The development in hyper spectral sensing has made it possible to acquire several hundred spectral bands of observational scene in a single acquisition. The increased spectral resolution of these hyper spectral images allow for detailed examination of land surfaces and different materials present in the observational scene, which was previously not possible with low spectral resolution of multispectral imaging scanners .

1.1 Hyper Spectral Image

Hyper Spectral imaging or imaging spectrometry is a spectral sensing technique in which an object is photographed using several well defined optical bands in broad spectral range. Technological progression in optical sensors over the last few decades provides enormous amount of information in terms of attaining requisite spatial, spectral and temporal resolutions. Especially, the generous spectral information comprises of hyperspectral images (HSIs) establishes new application domains and poses new technological challenges in data analysis. With the available high spectral resolution, subtle objects and materials can be extracted by hyperspectral imaging sensors with very narrow diagnostic spectral bands for the variety of purposes such as detection, urban planning, agriculture, identification, surveillance, and quantification.

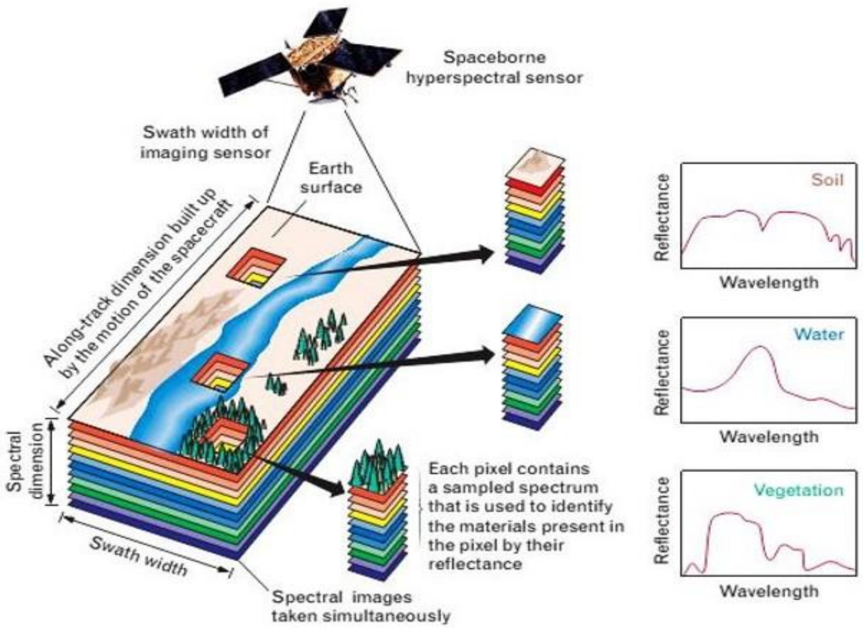

Spaceborne hyperspectral sensor
Swath width of imaging sensor
Earth surface
Along-track dimension built up by the motion of the spacecraft
Spectral dimension
Swath width
Spectral images taken simultaneously
Each pixel contains a sampled spectrum that is used to identify the materials present in the pixel by their reflectance
Soil
Reflectance
Wavelength
Water
Reflectance
Wavelength
Vegetation
Reflectance
Wavelength

Fig 1.2: Example of Hyper spectral Image

Hyperspectral image is a 3D data cube (as shown in figure 1.2), which contains two-dimensional spatial information (image feature) and one-dimensional spectral information . Especially, the spectral bands occupy very fine spectral wavelengths, while the image features such as Land cover features and shape features disclose the disparity and association among adjacent pixels from different directions at a confident wavelength.

1.2 Hyperspectral Image Classification

Hyperspectral Image (HSI) classification is a phenomenal mechanism to analyze diversified land cover in remotely sensed hyperspectral images. In the remote sensing community, the term classification is used to denote the process that assigns individual pixels to a set of classes. The output of the classification step is known as the classification map.

The two main challenges confront during classifying hyperspectral images first one is a curse of dimensionality which is also termed as Hughes phenomenon and another one is working with few training samples. Based on the use of training sample image classification can be made into supervised, unsupervised and semi -supervised hyperspectral image classification methods. In supervised classification only labeled data is used to train the classifier. In unsupervised method the computer or algorithm automatically group's pixels with similar spectral characteristics (means, standard deviations, etc.,) into unique clusters according to some statistically determined criteria. While in semi-supervised classification both labeled and unlabeled data can be used to train the classifier.

1.3 Objective

- Our aim is to find a best method for reducing the overfitting in hyper spectral remote sensing images.

In this project we performed classification on custom hyperspectral remote sensing data set and predicting the best method by comparing the accuracy indexes obtained from each method. In the field of hyperspectral image classification, a widely used way for objective performance evaluation of different classification methods is calculating two accuracy indexes, i.e., the validation accuracy, the training loss . The accuracy indexes are obtained by comparing the classification results with the data set.

1.4 Organization of the report

In this Chapter, we discuss why we use the hyper spectral image and what is Hyper spectral Image Classification. Chapter 2 outlines the exciting overfitting reduction methods and classification algorithms used. Chapter 3 we discuss about the software and hardware requirement specification , datasets used for the project. Chapter 4 we discuss about the installation of python and python libraries required. Chapter 5 we present the design and steps involved in design. Chapter 6 we discuss about machine learning algorithms. Chapter 7 we discuss about deep learning algorithms. Chapter 8 we present the classification maps obtained by using different classification algorithms. Chapter 9 comparing metrics obtained by using different classification algorithms.

CHAPTER – 2

LITERATURE REVIEW

Numerous literatures pertaining to classification of Hyper Spectral Image (HSI) have been published already and are available for public usage. A comparative study on HSI classification will help use to find the best method which will classify the images efficiently with maximum accuracy.

Hyper Spectral Image captured by hyper spectral sensors collect hundreds of narrow and contiguous spectral bands with a fine spectral resolution for each image. Each pixel in a Hyper Spectral Image (HSI) can be represented as a high-dimensional vector with a size corresponding to the number of spectral bands. HSI has a high spectral resolution which can provide rich information for the classification and identification of objects in a satellite scene. Hyper Spectral Images does have some training data samples which causes over fitting. In order to overcome this problem we are improving the accuracy in training data sets by using CROSS-VALIDATION. Hence, by applying the cross validation and data augmentation to the existing datasets, overcomes over fitting problem and improves the classification accuracy

2.1 Reduction of Overfitting

Overfitting is “The production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably”.

- **Methods to detect Overfitting** :

  1 . Cross-validation

  2 . Train with more data

  3 . Remove features

  4 . Early stopping

  5 . Regularization

  6 . Ensembling

  7 . Data Augmentation

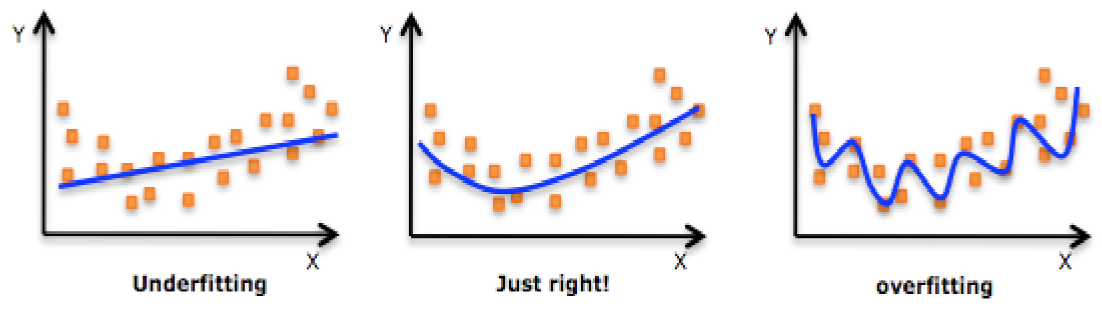


Fig 2.1 : Differentiation of Overfitting with remaining

Overfitting can be reduced in two ways : Cross validation and
Data Augmentation techniques.

- Cross validation approach is used to reduce the overfitting problem.
- Data augmentation approach is used to find out the classification accuracy.

2.1.1 Cross Validation

Cross-validation is a statistical method used to estimate the performance of machine learning models. It is used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited. In cross-validation, you make a fixed number of folds of the data, run the analysis on each fold, and then average the overall error estimate.

you have trained the model with the dataset available and now you want to know how well the model can perform. One approach can be that you are going to test the model on the dataset you have trained it on, but this may not be a good practice.

2.1.2 Data Augmentation

Data Augmentation is a process of increasing the available limited data to large meaningful and more diversity amounts. In other terms, we are artificially increasing the size of the dataset by creating different versions of the existing data from our dataset. Consider a car in an image, the car may not be at the center in all cases, sometimes it can be in the left side of the image or right. The image may be clicked on a bright sunny day or on a cloudy day. The image might be the left view of the car or the right view.

We can apply this technique at the time of the data generation after preprocessing and before training. We apply this technique only for the training dataset. At test time we use the test image directly without any transformations. For small datasets we can generate the transformations of the images and train the model with all the data at once. For large datasets we can generate unique transformed images for every batch of an epoch. Data augmentation techniques are fliping ,rotating, zoom in ,zoom out, and etc.

2.2 Techniques of Classification in Hyperspectral Image

Hyper spectral image classification fall under three categories supervised hyper spectral image classification, unsupervised hyper spectral image classification, semi supervised hyper spectral image classification to handle the various issues which are faced while classifying hyper spectral images such as large number of spectral channels, acquisition of labelled data etc.

Based on the use of training sample image classification can be done into supervised, unsupervised and semi-supervised manner. In Unsupervised classification, grouping of pixels is done on basis of unlabelled data. The supervised classification process involves grouping of samples of known labels to assign unclassified pixels to one of several informational classes on the basis of available labelled data. Supervised method performs the steps such as feature extraction, training and labelling processes. In semi-supervised classification conducted in the presence of labelled and unlabelled data.

In general, the complex characteristics of hyperspectral data make the accurate classification of such data challenging for traditional machine learning methods. In addition, hyperspectral imaging often deals with an inherently nonlinear relation between the captured spectral information and the corresponding materials. In recent years, deep learning has been recognized as a powerful feature extraction tool to effectively address nonlinear problems and widely used in a number of image processing tasks. Motivated by those successful applications, deep learning has also been introduced to classify HSIs and provide good performance.

In this project we will perform classification of hyperspectral image using traditional machine learning method Convolutional Neural Network (CNN). Then, we will compare metrics obtained from bothtraditional machine learning and deep learning method.

2.2.1 Convolutional neural network

Convolutional neural network (CNN) is a special type of feed forward artificial neural network in which connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. It is the foundation of most computer vision technologies. Unlike traditional multilayer perceptron architectures, it uses two operations called 'convolution' and pooling' to reduce an image into its essential features, and uses those feature to understand and classify the image.

The basic building blocks of CNN are:

Convolution layer: A "filter", sometimes called a "kernel", is passed over the image, viewing a few pixels at a time (for example, 3X3 or 5X5). The convolution operation is a dot product of the original pixel values with weightsdefined in the filter. The results are summed up into one number that represents all the pixels the filter observed.

Activation layer: The convolution layer generates a matrix that is much smaller in size than the original image. This matrix is run through an activation layer, which introduces non-linearity to allow the network to train itself via backpropagation. The activation function is typically ReLu.

Pooling layer: "Pooling" is the process of further downsampling and reducing the size of the matrix. A filter is passed over the results of the previous layer and selects one number out of each group of values (typically the maximum, this is called max pooling).

Fully connected layer: A traditional multilayer perceptron structure. Its input is a one-dimensional vector representing the output of the previous layers. Its output is a list of probabilities for different possible labels attached to the image (e.g. dog, cat, bird).

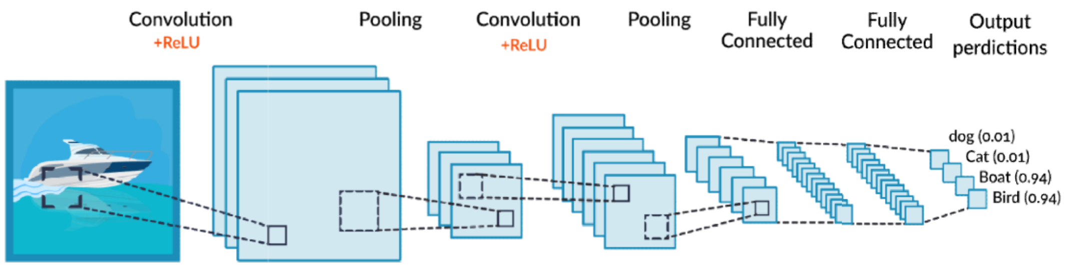


Fig 2.2: Example CNN architecture

CHAPTER – 3

REQUIREMENTS

3.1 Software and Hardware Requirement Specification

Hardware Requirements:

- RAM 8GB minimum, above 8 GB recommended
- Disk space required is 5GB for python, required packages and dataset installations
- Processor Intel core i5 or above

Software Requirements:

The Software requirements for developing this project is as follows

- Operating system – Windows, Linux and Mac OS
- Platform – Colab Notebook
- Python 3.7.5, Google colaboratory

3.2 Datasets

The Datasets used in this project are Custom dataset hyperspectral remote sensing datasets.

3.2.1 Custom Dataset

This hyperspectral image data set captured the Custom dataset . Custom dataset is 150*150 pixels, but some of the samples in both images contain no information and have to be discarded before the analysis. The geometric resolution is 1.3 meters. There are 140 hyperspectral bands in the image data set, with 150 × 150 pixels for the spatial dimensions.

In this project we downloaded two files from each dataset like corrected and cube file. Cube file contains the data related class labels. It will be stored in both the training dataset and test dataset.

Earth file contains the data related to grounding and it is also either stored in training dataset and also test dataset.

#	**Class**	**Samples**
1	Cube	20
2	Earth	60

Table 3.1: Earth and cube classes for the dataset and their respective samples number

CHAPTER – 4

TECHNOLOGY

4.1 Language used

The programming language that was used in this project on a comparative study on hyperspectral image classification is python. The implementation of source code was done through python. Python is an interpreted, interactive, object-oriented programming language which is suitable for implementing machine learning and deep learning algorithms in easier way.

Python is among the most developer-friendly Machine Learning and Deep Learning programming language, and it comes with the support of a broad set of libraries catering to your every use-case and project.

Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems. ... Python code is understandable by humans, which makes it easier to build models for machine learning.

First, you need Python installed. ... It is an industrial-strength Python implementation for Linux, OSX, and Windows, complete with the required packages for machine learning, including numpy, scikit-learn, and matplotlib. It also includes iPython Notebook, an interactive environment for many of our tutorials.

4.2 Installation of Python

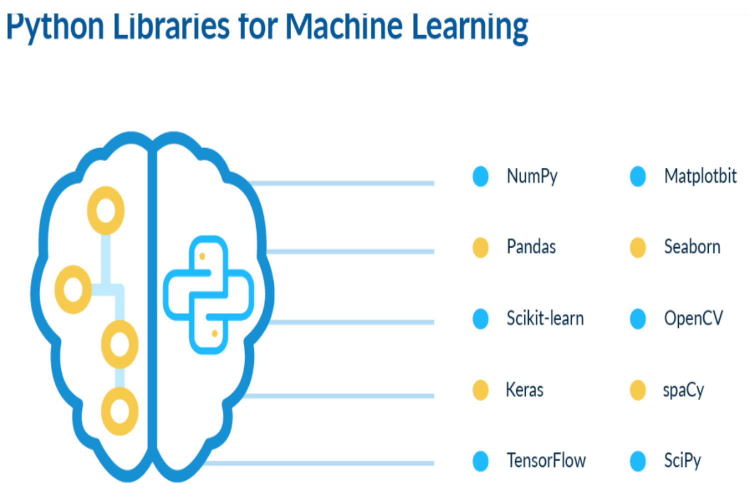
Step 1: Download the Python Installer binaries from the website:
https://www.python.org/downloads/windows/

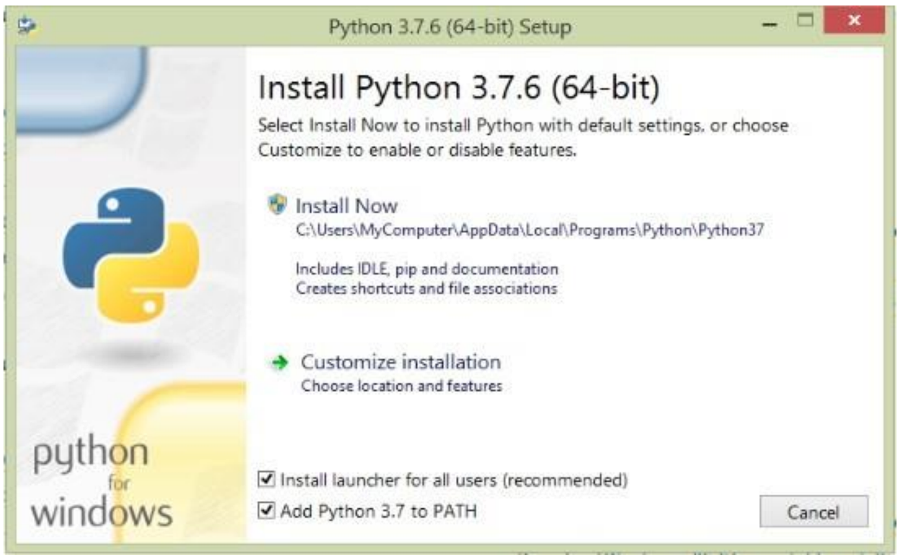
In website that "Wnidows x86-64 executable installer" under python 3.7.6 download that python installer binary if your system is of 64 bit or "Wnidows x86 executable installer" for 32 bit.

Step 2: Run the Executable Installer

- Once the installer is downloaded, run the Python installer. Check the Install launcher for all users check box. Further, you check the Add Python 3.7.6 to path check box to include the interpreter in the execution path and select Customize installation as shown in figure 4.1.
- After selecting it will take to you Optional Features as shown in figure 4.1.

  Then click next.
- After that it takes you to Advanced Options. Here you can customize installation location by click the browse button as shown in figure 4.1 then click install to start installation.





Install Python 3.7.6 (64-bit)

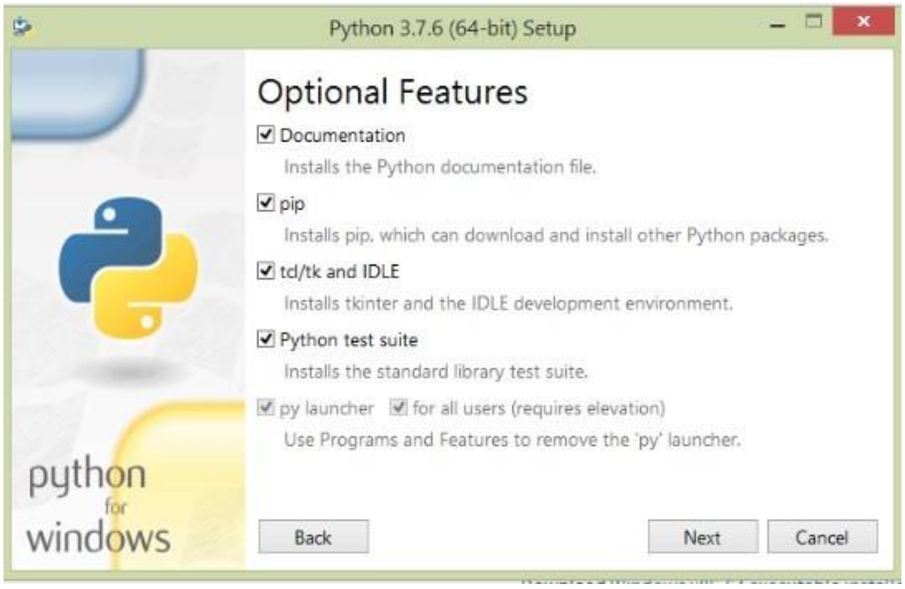
Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\MyComputer\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

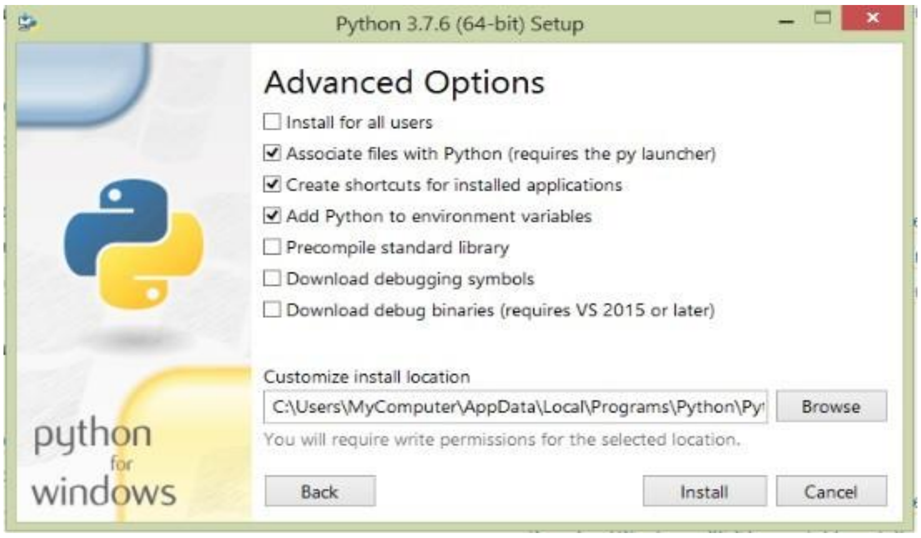
Customize installation
Choose location and features

- [x] Install launcher for all users (recommended)
- [x] Add Python 3.7 to PATH



Optional Features

- [x] Documentation
  Installs the Python documentation file.
- [x] pip
  Installs pip, which can download and install other Python packages.
- [x] td/tk and IDLE
  Installs tkinter and the IDLE development environment.
- [x] Python test suite
  Installs the standard library test suite.
- [x] py launcher [x] for all users (requires elevation)
  Use Programs and Features to remove the 'py' launcher.



Advanced Options

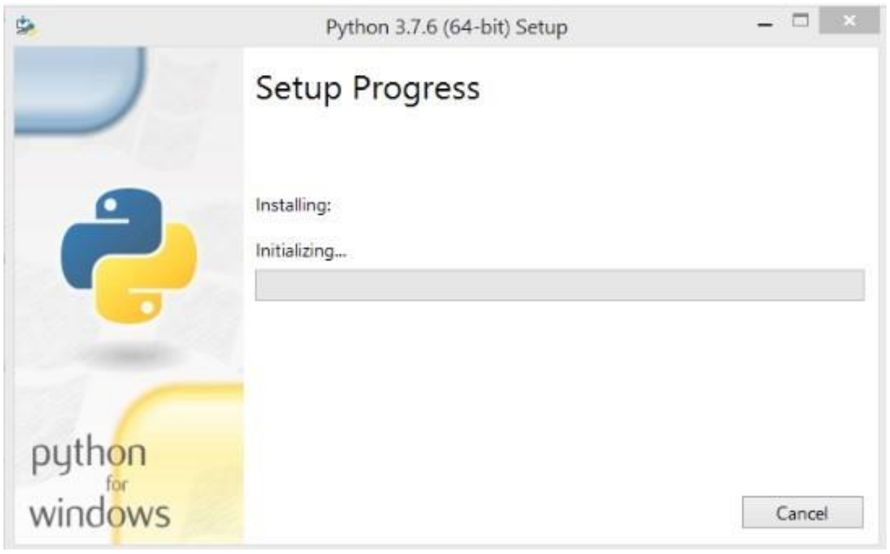
- [ ] Install for all users
- [x] Associate files with Python (requires the py launcher)
- [x] Create shortcuts for installed applications
- [x] Add Python to environment variables
- [ ] Precompile standard library
- [ ] Download debugging symbols
- [ ] Download debug binaries (requires VS 2015 or later)

Customize install location
C:\Users\MyComputer\AppData\Local\Programs\Python\Py
You will require write permissions for the selected location.

Fig 4.1: Running the Executable Installer

Step 3: Setup progress

- Once the installation is over, you will see a Python Setup Successful window.
- Then click close as shown in figure 4.2.

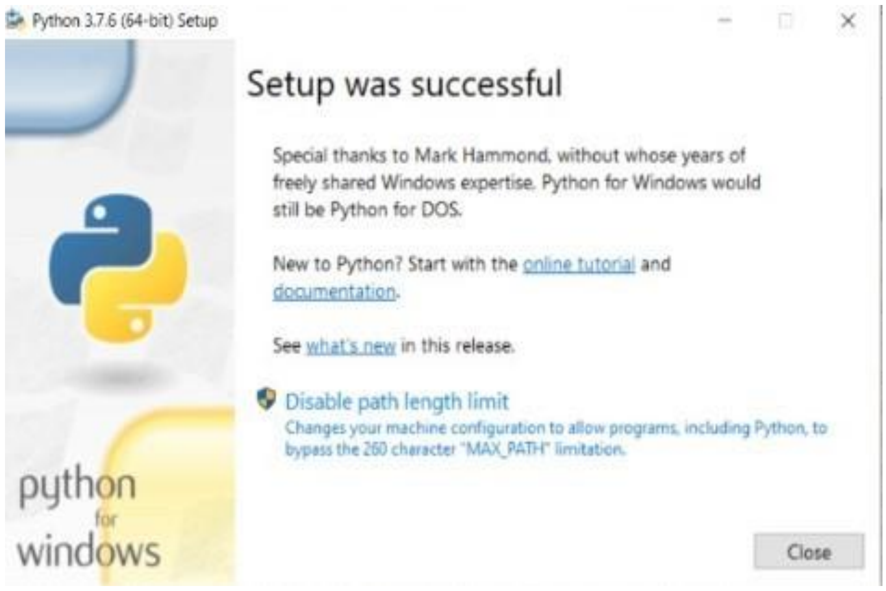


Python 3.7.6 (64-bit) Setup

Setup Progress

Installing:

Initializing...



Python 3.7.6 (64-bit) Setup

Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

Disable path length limit

Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

Fig 4.2: Setup progress

Step 4: Verify the Python Installation

- Search for the command prompt and type “python”. You can see that

Python 3.7.6 is successfully installed.

- An alternate way to reach python is to search for "Python" in the start menu and clicking on IDLE (Python 3.7 64-bit). You can start coding in Python using the Integrated Development Environment (IDLE).

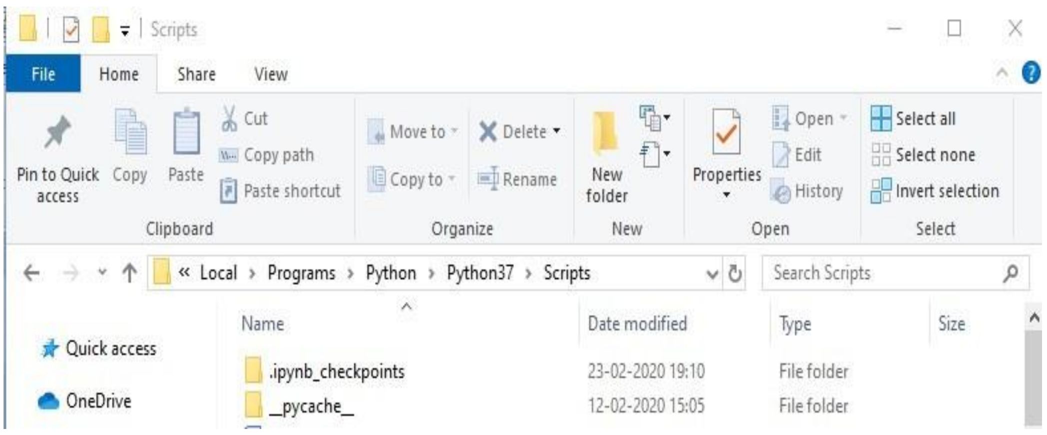
4.3 Libraries Used

Python is increasingly being used as a scientific language. The necessary libraries for our project should be installed before implementation. The libraries required for our project are: Numpy, Pandas, Sklearn, Matplotlib, Scipy, Tensorflow, Keras, Spectral.

4.3.1 Numpy

NumPy or Numerical Python is linear algebra developed in Python. Why do a large number of developers and experts prefer it to the other Python libraries for machine learning?

Almost all Python machine-learning packages like Matplotlib, SciPy, Scikit-learn, etc rely on this library to a reasonable extent. It comes with functions for dealing with complex mathematical operations like linear algebra, Fourier transformation, random number and features that work with matrices and n-arrays in Python. NumPy Python package also performs scientific computations. It is widely used in handling sound waves, images, and other binary functions.



« Local › Programs › Python › Python37 › Scripts

Name	Date modified	Type	Size
.ipynb_checkpoints	23-02-2020 19:10	File folder	
__pycache__	12-02-2020 15:05	File folder	

Fig 4.3: Path that should be in Command Prompt

Installation of all packages will be done in the path as shown in figure 4.3.

For installation of this package we need to enter the command in the command prompt opened in path specified in the above fig 4.3.

- Command for installation: pip install numpy
- In order to verify the Installation enter the command: import numpy

4.3.2 Pandas

In machine learning projects, a substantial amount of time is spent on preparing the data as well as analyzing basic trends & patterns. This is where the Python Pandas receives machine learning experts' attention. Python Pandas is an open-source library that offers a wide range of tools for data manipulation & analysis. With this library, you can read data from a broad range of sources like CSV, SQL databases, JSON files, and Excel.

It enables you to manage complex data operation with just one or two commands. Python Pandas comes with several inbuilt methods for combining data, and grouping & filtering time-series functionality. Overall, Pandas is not just limited to handle data-related tasks; it serves as the best starting point to create more focused and powerful data tools.

For installation of this package we need to enter the command in the command prompt opened in path specified in the above fig 4.3.

- Command for installation: pip install pandas
- In order to verify the Installation enter the command: import pandas

4.3.3 Sklearn

Scikit-learn or Sklearn is another prominent open-source Python machine learning library with a broad range of clustering, regression and classification algorithms. DBSCAN, gradient boosting, random forests, vector machines, and k-means are a few examples. It can interoperate with numeric and scientific libraries of Python like NumPy and SciPy.

It is a commercially usable artificial intelligence library. This Python library supports both supervised as well as unsupervised ML.

For installation of this package we need to enter the command in the command prompt opened in path specified in the above fig 4.3.

- Command for installation: pip install sklearn
- In order to verify the Installation enter the command: import sklearn

4.3.4 Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

For installation of this package we need to enter the command in the command prompt opened in path specified in the above fig 4.3.

- Command for installation: pip install matplotlib
- In order to verify the Installation enter the command: import matplotlib

4.3.5 Scipy

Scipy is a library that uses Numpy for more mathematical functions. Scipy uses Numpy arrays as the basic data structure, and comes with modules for various commonly used tasks in scientific programming, including linear algebra, integration (calculus), ordinary differential equation solving, and signal processing.

For installation of this package we need to enter the command in the command prompt opened in path specified in the above fig 4.3.

- Command for installation: pip install scipy
- In order to verify the Installation enter the command: import scipy

4.3.6 Keras

Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer. Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend. So Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano.

Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

For installation of this package we need to enter the command in the command prompt opened in path specified in the above fig 4.3.

- Command for installation: pip install keras
- In order to verify the Installation enter the command: import keras

In this project, **tensorflow** is used backend for keras. So, before installing keras library tensorflow is installed.

TensorFlow is a very popular open-source library for high performance numerical computation developed by the Google Brain team in Google. As the name suggests, Tensorflow is a framework that involves defining and running computations involving tensors. It can train and run deep neural networks that can be used to develop several AI applications. TensorFlow is widely used in the field of deep learning research and application.

For installation of this package we need to enter the command in the command prompt opened in path specified in the above fig 4.3.

- Command for installation: pip install tensorflow
- In order to verify the Installation enter the command:
  import tensorflow

4.3.7 Spectral

Spectral Python (SPy) is a pure Python module for processing hyperspectral image data. It has functions for reading, displaying, manipulating, and classifying hyperspectral imagery. It can be used interactively from the Python command prompt or via Python scripts.

For installation of this package we need to enter the command in the command prompt opened in path specified in the above fig 4.3.

- Command for installation: pip install spectral
- In order to verify the Installation enter the command: import spectral

4.4 Colab Notebook or Google Colabaratory

As an effort to make an integrated interactive computing environment for multiple languages, **Colaboratory**, or **"Colab"** for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them.Colab notebooks are Jupyter notebooks that are hosted by Colab.

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow

- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

Colaboratory is a free Jupyter notebook environment provided by Google where you can use free GPUs and TPUs which can solve all these issues.

How to use Google Colab

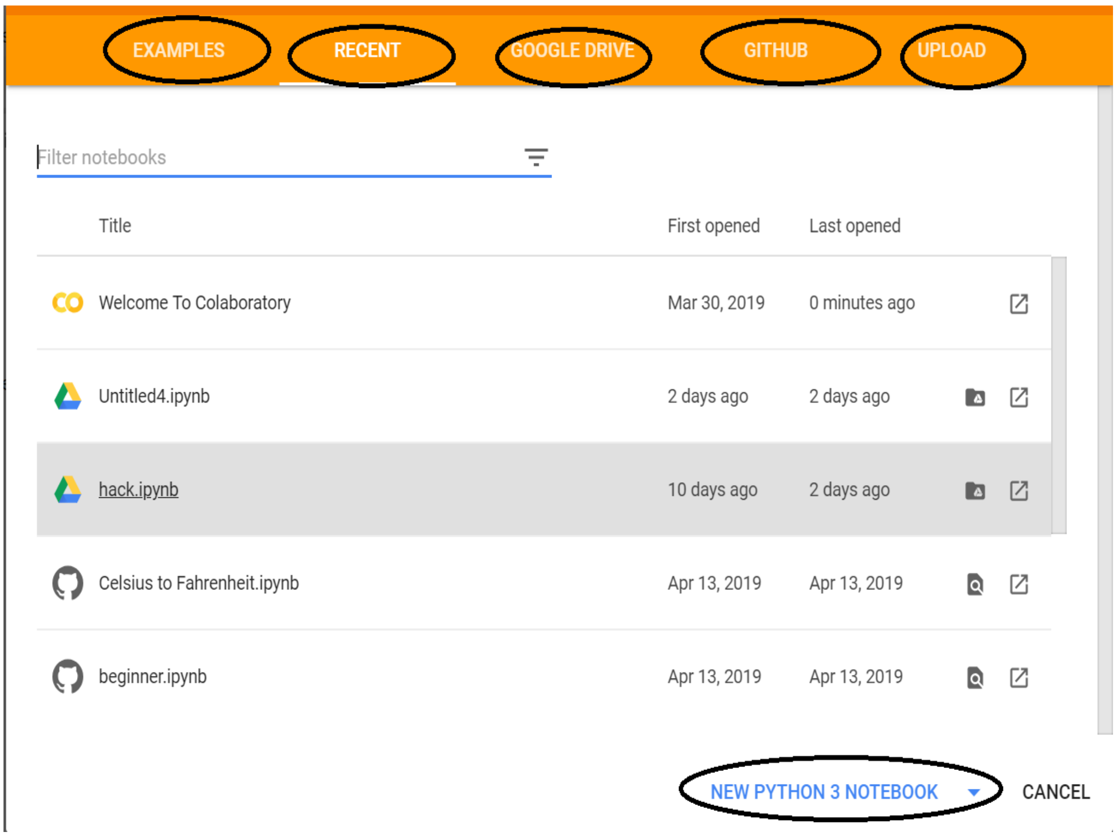
If you want to create a machine learning model but say you don't have a computer that can take the workload, Google Colab is the platform for you. Even if you have a GPU or a good computer creating a local environment with anaconda and installing packages and resolving installation issues are a hassle.

Getting Started

To start working with Colab you first need to log in to your google account, then go to this link https://colab.research.google.com.

Opening Colab Notebook:

On opening the website you will see a pop-up containing following tabs –



Title	First opened	Last opened
Welcome To Colaboratory	Mar 30, 2019	0 minutes ago
Untitled4.ipynb	2 days ago	2 days ago
hack.ipynb	10 days ago	2 days ago
Celsius to Fahrenheit.ipynb	Apr 13, 2019	Apr 13, 2019
beginner.ipynb	Apr 13, 2019	Apr 13, 2019

Fig 4.4 Opening Colab Notebook in google chrome

EXAMPLES: Contain a number of Jupyter notebooks of various examples.

RECENT: Jupyter notebook you have recently worked with.

GOOGLE DRIVE: Jupyter notebook in your google drive.

GITHUB: You can add Jupyter notebook from your GitHub but you first need to connect Colab with GitHub.

UPLOAD: Upload from your local directory.

Else you can create a new Jupyter notebook by clicking New Python3 Notebook or New Python2 Notebook at the bottom right corner.

Notebook's Description:

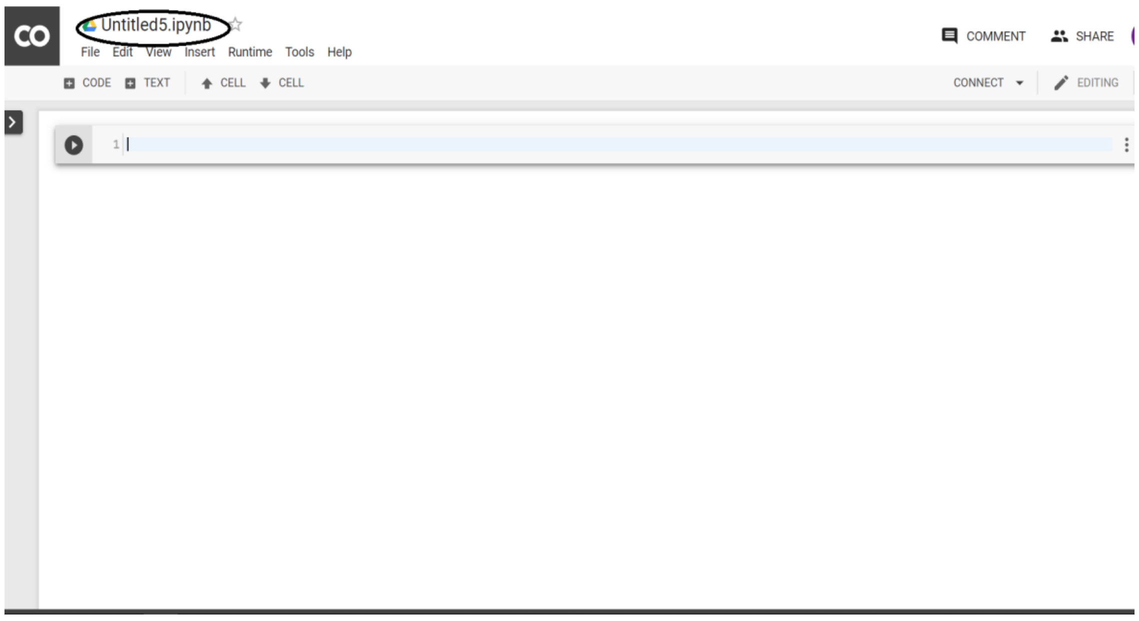


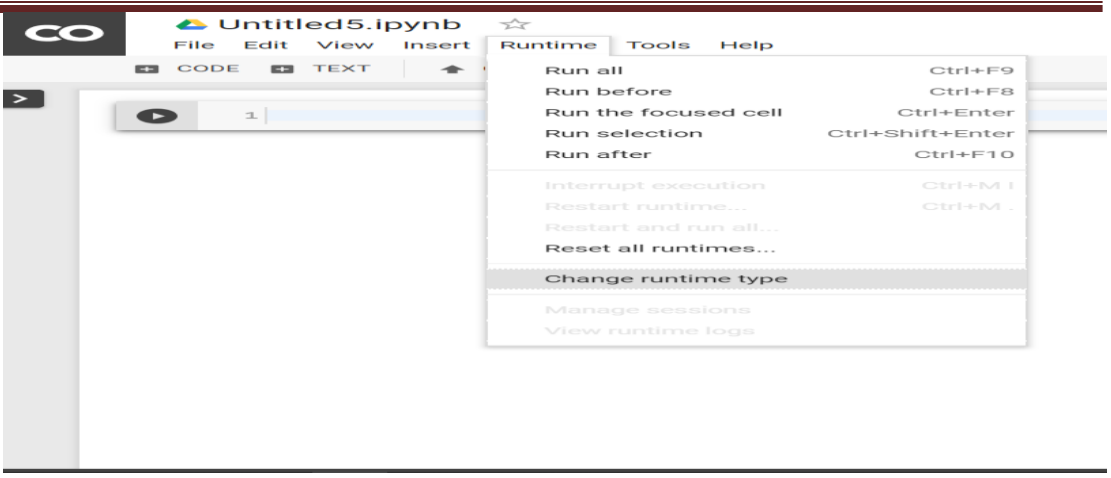
Fig 4.5 Creating a New Notebook

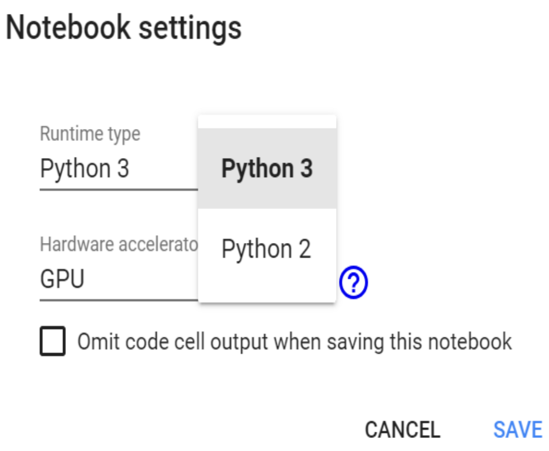
On creating a new notebook, it will create a colab notebook with Untitled0.ipynb and save it to your google drive in a folder named Colab Notebooks. Now as it is essentially a Jupyter notebook, all commands of Jupyter notebooks will work here. Though, you can refer the details in Getting started with colab Notebook.

Let's talk about what different here.

Change Runtime Environment:

Click the "Runtime" dropdown menu. Select "Change runtime type". Select python2 or 3 from "Runtime type" dropdown menu.




Notebook settings

Runtime type
Python 3

Hardware accelerator
GPU

Use GPU and TPU:

Click the “Runtime” dropdown menu. Select “Change runtime type”.

Now select anything(GPU,CPU,None)you want in the “Hardware accelerator” dropdown menu.

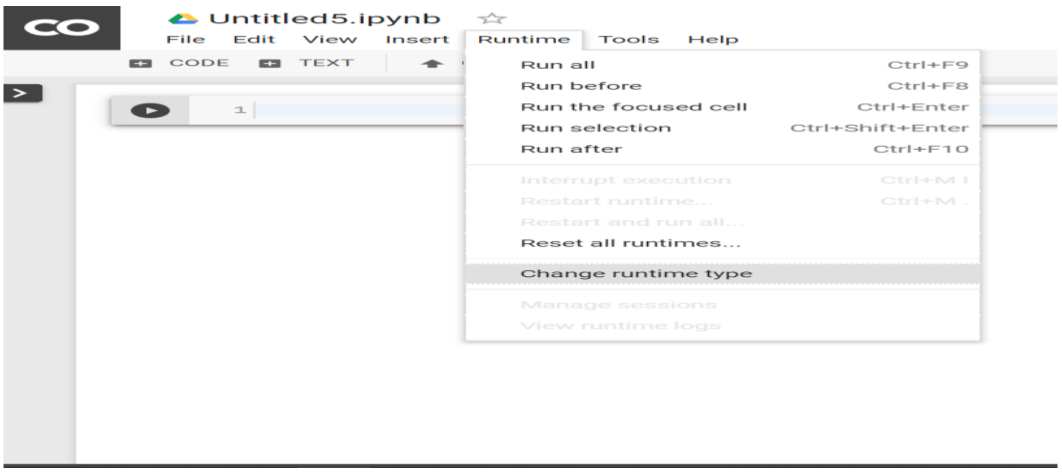
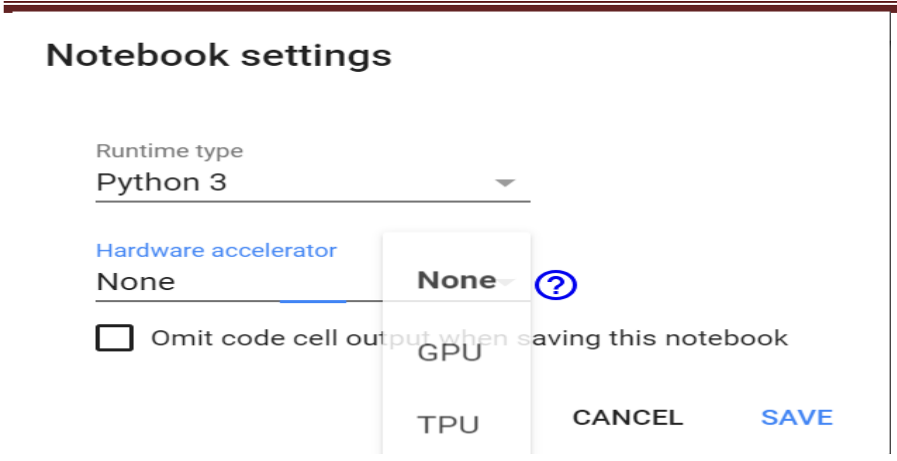


Fig 4.6 Running GPU and TPU in Colab



Install Python packages –

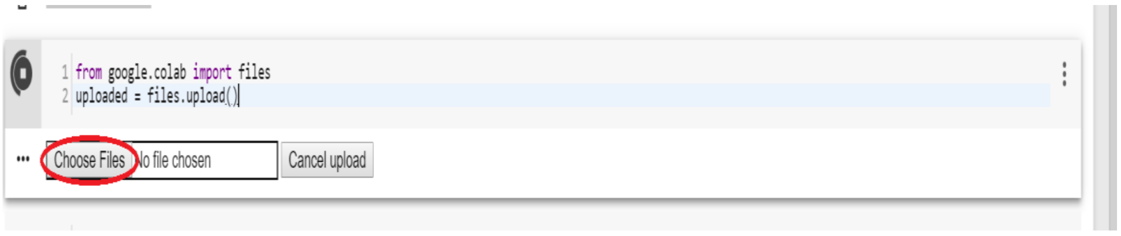
Use can use pip to install any package. For example:

```
! pip install pandas
```

Upload File:

```
from google.colab import files
uploaded = files.upload()
```

Select “Choose file” and upload the file you want. Enable third-party cookies if they are disabled.



```
from google.colab import files
uploaded = files.upload()
```

Then you can save it in a dataframe.

CHAPTER-5

DESIGN

5.1 Data Flow Diagram

Data flow diagram will describe the flow of the project in a cleared manner. The data flow diagrams (DFD) are used for modeling the requirements and it focus on flow of the data but not order of the data.DFD can be represented in the form of levels.

5.1.1 Level – 0

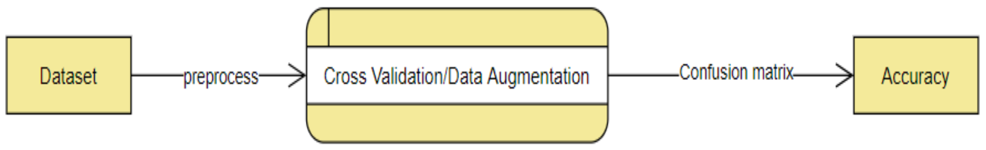


Fig 5.1: DFD level -0

In this level, there will the highest abstraction of data. Only source, process and output are provided in this level. Datasets can be obtained as: downloading from website.

5.1.2 Level – 1

In this level, the flow diagram will explain some more detailed view of the project.

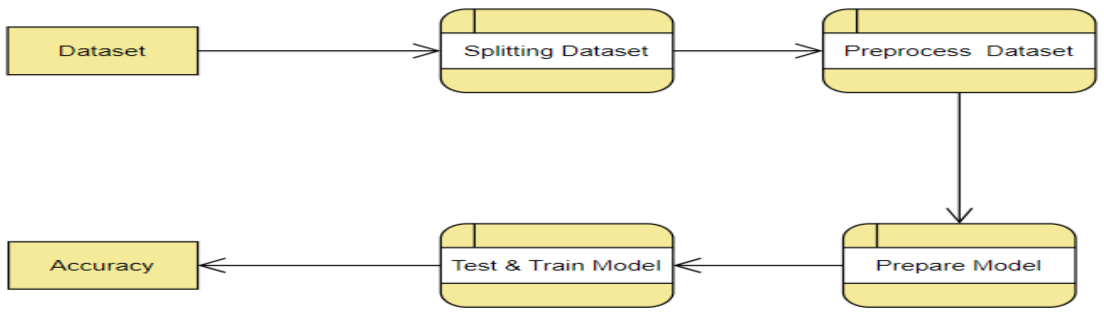


Fig 5.2: DFD level – 1

5.1.3 Level -2

Initially obtain the datasets from any websites. After obtaining the datasets, perform data transformation to it in such a way that there shouldn't be any integration problem or any redundancy issue.

Now, the actual task begins. By using machine learning and deep learning techniques, apply the classification algorithms. Select the classifiers that are to be applied to the project and apply them to the obtained dataset. Applying the classifiers to the dataset actually mean that needs to train the model with the classifiers and test the data so that the model will be fit.

Thus after the model is completed, it provides some metrics that describe the performance of each classifier that we have used above. Through those metrics, we can compare and select the best classifier that performs well only for the given input dataset. The best classifier can be varied when the dataset be changed. It is not generalized best classifier, but specific to the given dataset.

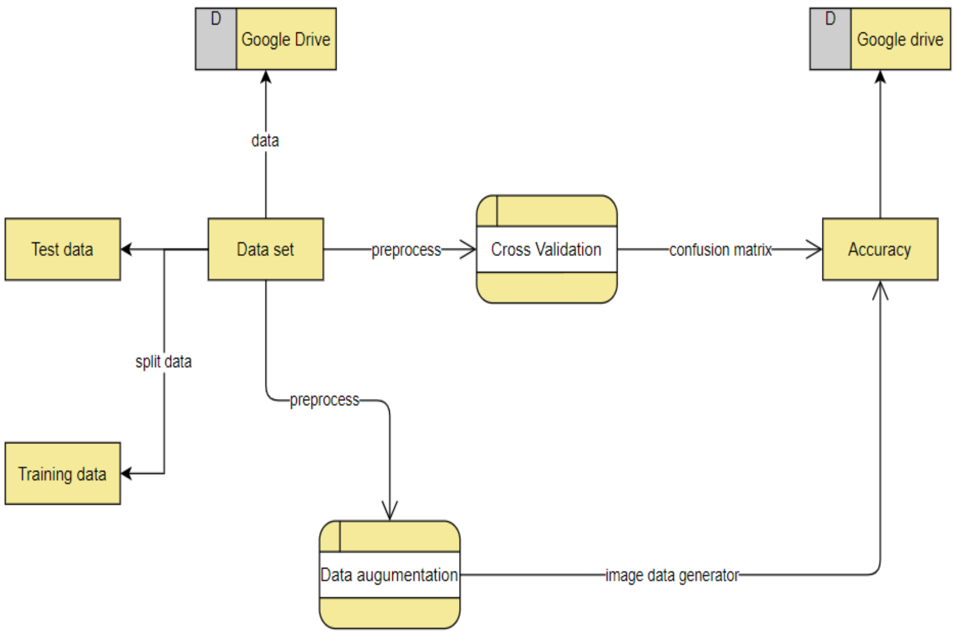


Fig 5.3: DFD level - 2

5.2 Steps involved in Design

1. Data Collection: Collecting the Datasets required for performing the project

2. Data Preparation: In this step, we will overcome the two challenges of hyperspectral image dataset like reducing overfitting, increasing accuracy.

To overcome the challenge of reducing overfitting in hyper spectral image dataset can be done by using a method called CV. After that dataset is divided into train and test data by using inbuilt function of sklearnsplitTrainTestSet() by giving test ratio or train ratio. T

To overcome the challenge of increasing acccuracy for training, data augmentation process is used. Data augmentation is the process of increasing the amount and diversity of data. We do not collect new data, rather we transform the already present data. It is an integral process in deep learning, as in deep learning we need large amounts of data and in some cases it is not feasible to collect thousands or millions of images, so data augmentation comes to the rescue. It helps us to increase the size of the dataset and introduce variability in the dataset.

For data augmentation some of the operations are performed like rotation, shearing, zooming cropping, flipping and changing the brightness level etc. In this project, for data augmentation flipping operation is performed on training data.

3. Model Building: , CNN model (Sequential Model) from deep learning.

4. Applying models: Deep learning models are applied to the Custom Dataset. Obtain the metric values from each model on dataset then compare those metric values.

CHAPTER – 6

MACHINE LEARNING ALGORITHMS

6.1 Machine Learning

Machine Learning (ML) plays a key role in many scientific disciplines and its applications are part of our daily life. It is used for example to filter spam email, for weather prediction, in medical diagnosis, product recommendation, face detection, fraud detection, etc. Machine Learning studies the problem of learning, which can be defined as the problem of acquiring knowledge through experience.

This process typically involves observing a phenomenon and constructing a hypothesis on that phenomenon that will allow one to make predictions or, more in general, to take rational actions. For computers, the experience or the phenomenon to learn is given by the data, hence we can define ML as the process of extracting knowledge from data.

Machine learning is closely related to the fields of Statistics, Pattern Recognition and Data Mining. At the same time, it emerges as a subfield of computer science and gives special attention to the algorithmic part of the knowledge extraction process. In summary, the focus of ML is on algorithms that are able to learn automatically the patterns hidden in the data.

6.1.1 Supervised Learning

This project is uses supervised learning algorithms, where ML algorithms are trained on some annotated data (the training set) to build predictive models, or learners, which will enable us to predict the output of new unseen observations. It is called supervised because the learning process is done under the supervision of an output variable, in contrast with unsupervised learning where the response variable is not available.

Supervised learning assumes the availability of labeled samples, i.e. observations annotated with their output, which can be used to train a learner. In the training set we can distinguish between input features and an output variable that is assumed to be dependent on the inputs.

The output, or response variable, defines the class of observations and the input features are the set of variables that have some influence on the output and are used to predict the value of the response variable. Depending on the type of output variable we can distinguish between two types of supervised task:

- classification (Confusion Matrix)

The first assumes a categorical output, while the latter a continuous one. Fraud detection belongs to the first type since observations are transactions that can be either genuine or fraudulent, while in other problems such as stock price predictions the response is a continuous variable. On the other hand, in both classification and regression tasks, input features can include both quantitative and qualitative variables. In a classification problem an algorithm is assessed on its overall accuracy to predict the correct classes of new unseen observation.

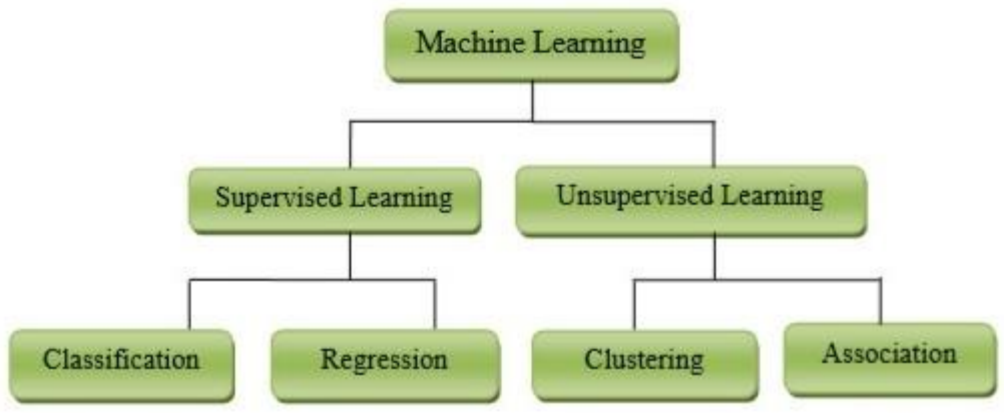


Fig 6.1: Types of ML

Supervised learning as the name indicates a presence of supervisor as teacher. Basically supervised learning is a learning in which we teach or train the machine using data which is well labeled that means some data is already tagged with correct answer. After that, machine is provided with new set of examples (data) so that supervised learning algorithm analyses the training data (set of training examples) and produces a correct outcome from labelled data.

For instance, suppose you are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all different fruits one by one like this:

- If shape of object is rounded and depression at top having color Red then it will be labeled as Apple.

- If shape of object is long curving cylinder having color Green-Yellow then it will be labeled as Banana.

Since machine has already learnt the things from previous data and this time have to use it wisely. It will first classify the fruit with its shape and color, and would confirm the fruit name as banana and put it in Banana category. Thus machine learns the things from training data (basket containing fruits) and then apply the knowledge to test data (new fruit).

6.1.2 Unsupervised Learning

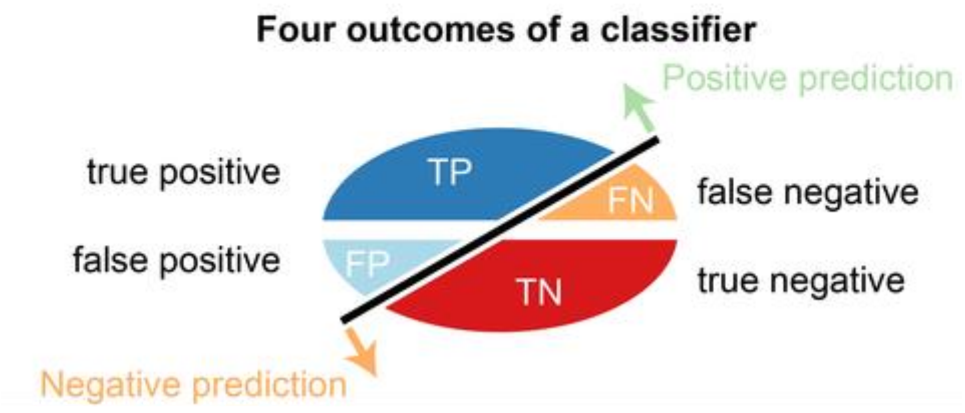
Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore machine is restricted to find the hidden structure in unlabeled data by our-self. For instance, suppose it is given an image having both A Comparative Study on Hyperspectral Image Classification.

dogs and cats which have not seen ever. Thus machine has no any idea about the features of dogs and cat so we can't categorize it in dogs and cats. But it can categorize them according to their similarities, patterns and differences i.e., we can easily categorize the above picture into two parts. First may contain all pictures having dogs in it and second part may contain all pictures having cats in it.

6.2 Confusion Matrix

Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.



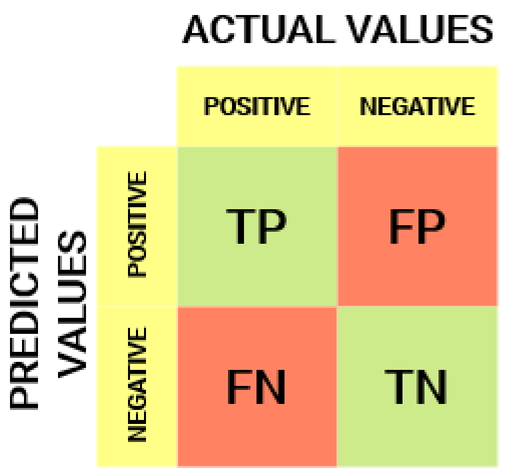


Fig.6.2 Confusion Matrix

Let's decipher the matrix:

The target variable has two values: Positive or Negative

The columns represent the actual values of the target variable

The rows represent the predicted values of the target variable

Understanding True Positive, True Negative, False Positive and False Negative in a Confusion Matrix

True Positive (TP)

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

True Negative (TN)

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

False Positive (FP) – Type 1 error

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the Type 1 error

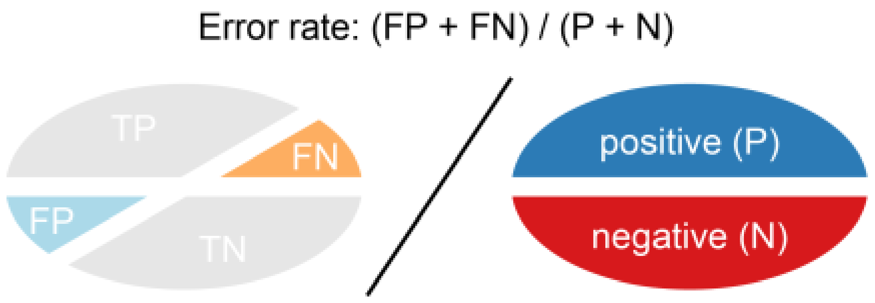
False Negative (FN) – Type 2 error

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the Type 2 error

Error rate (ERR) and accuracy (ACC) are the most common and intuitive measures derived from the confusion matrix.

Error rate

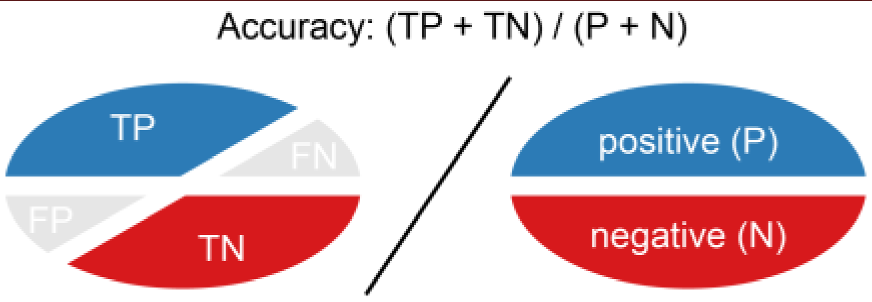
Error rate (ERR) is calculated as the number of all incorrect predictions divided by the total number of the dataset. The best error rate is 0.0, whereas the worst is 1.0.



Error rate is calculated as the total number of two incorrect predictions (FN + FP) divided by the total number of a dataset (P + N).

Accuracy

Accuracy (ACC) is calculated as the number of all correct predictions divided by the total number of the dataset. The best accuracy is 1.0, whereas the worst is 0.0. It can also be calculated by 1 – ERR.



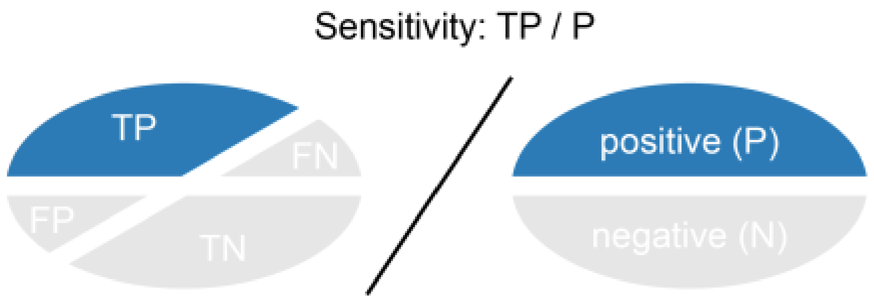
Accuracy is calculated as the total number of two correct predictions (TP + TN) divided by the total number of a dataset (P + N).

Other basic measures from the confusion matrix

Error costs of positives and negatives are usually different. For instance, one wants to avoid false negatives more than false positives or vice versa. Other basic measures, such as sensitivity and specificity, are more informative than accuracy and error rate in such cases.

Sensitivity (Recall or True positive rate)

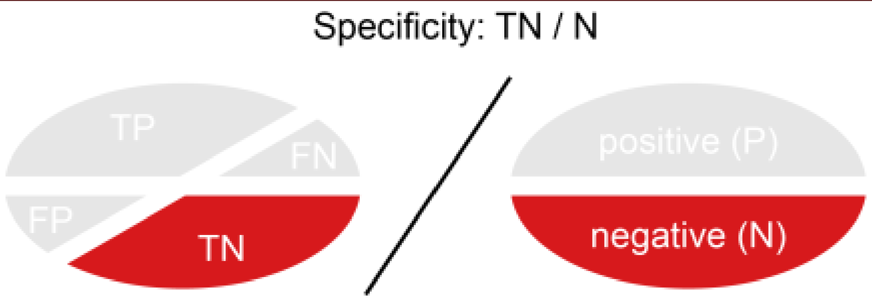
Sensitivity (SN) is calculated as the number of correct positive predictions divided by the total number of positives. It is also called recall (REC) or true positive rate (TPR). The best sensitivity is 1.0, whereas the worst is 0.0.



Sensitivity is calculated as the number of correct positive predictions (TP) divided by the total number of positives (P).

Specificity (True negative rate)

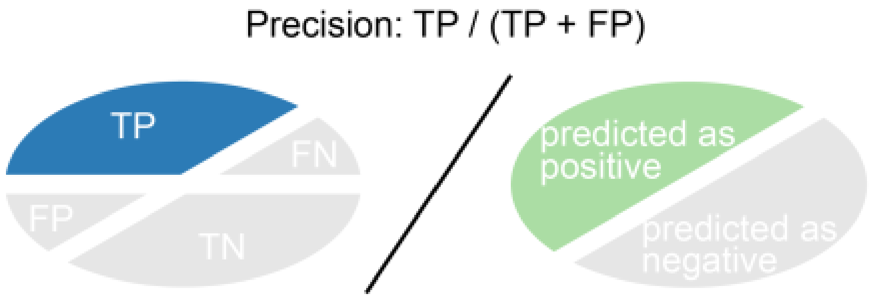
Specificity (SP) is calculated as the number of correct negative predictions divided by the total number of negatives. It is also called true negative rate (TNR). The best specificity is 1.0, whereas the worst is 0.0.



Specificity is calculated as the number of correct negative predictions (TN) divided by the total number of negatives (N).

Precision (Positive predictive value)

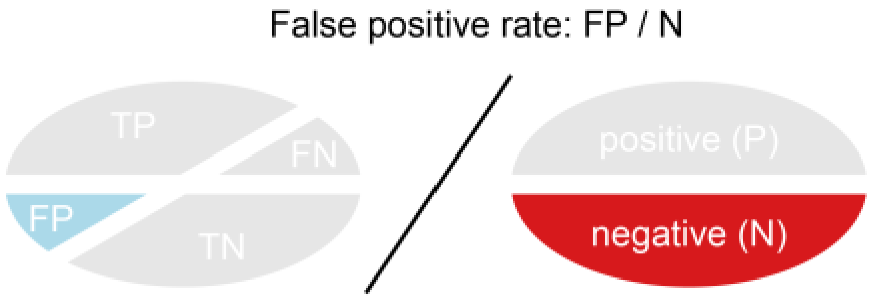
Precision (PREC) is calculated as the number of correct positive predictions divided by the total number of positive predictions. It is also called positive predictive value (PPV). The best precision is 1.0, whereas the worst is 0.0.



Precision is calculated as the number of correct positive predictions (TP) divided by the total number of positive predictions (TP + FP).

False positive rate

False positive rate (FPR) is calculated as the number of incorrect positive predictions divided by the total number of negatives. The best false positive rate is 0.0 whereas the worst is 1.0. It can also be calculated as 1 – specificity.



False positive rate is calculated as the number of incorrect positive predictions (FP) divided by the total number of negatives (N).

CHAPTER -7

DEEP LEARNING ALGORITHMS

7.1 Deep Learning

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

Deep learning, a subset of machine learning, utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

In this project, we are using Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN) Classifier’s from deep learning algorithms.

7.2 Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered,with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

In this project we used a CNN architecture with layers as shown in figure 7.5

In CNN all layers are organized in 3 dimensions: width, height and depth.

Convolution layer: The main building block of CNN is the convolution layer. Convolution is a mathematical operation to merge two sets of information. In our case the convolution is applied on the input data using a convolution filter/ kernel to produce a feature. In figure 7.4 taken Input data which is an image with dimension 5(height)x5(width)x1(number of channels) performed convolution operation using 3x3x1 kernel/filter by sliding the kernel throughout the image/input data and obtain a convoluted feature/feature map. This layer performs feature extraction on input image by convolution operation.

After convolution operation the flatten function is used to convert 3D data to 1D in order to pass data to fully connected layer which serve as a classifier on top of the extracted features

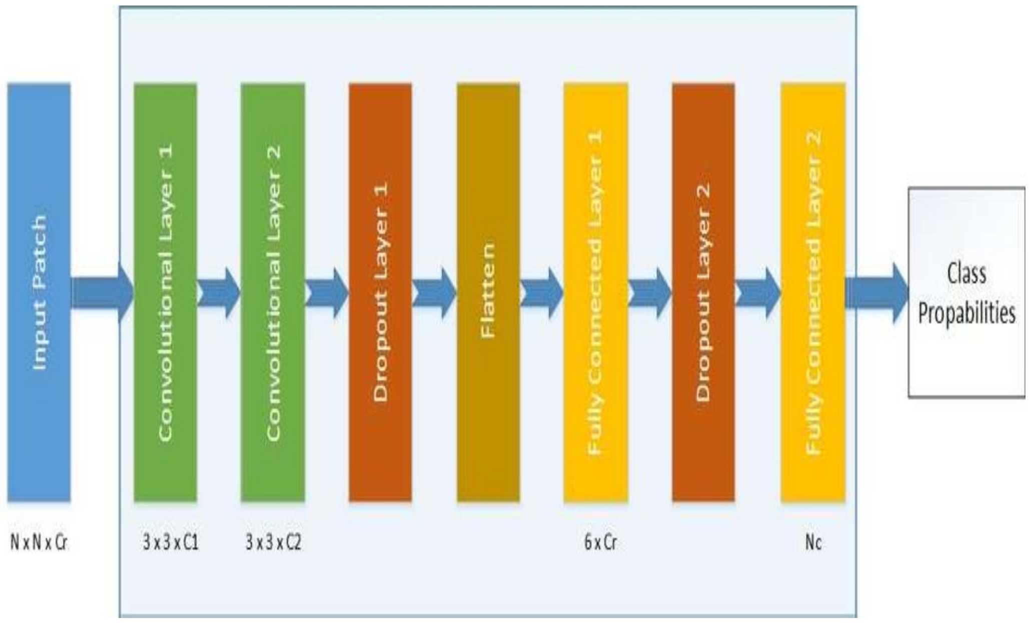


Fig7.1:CNNclassifier

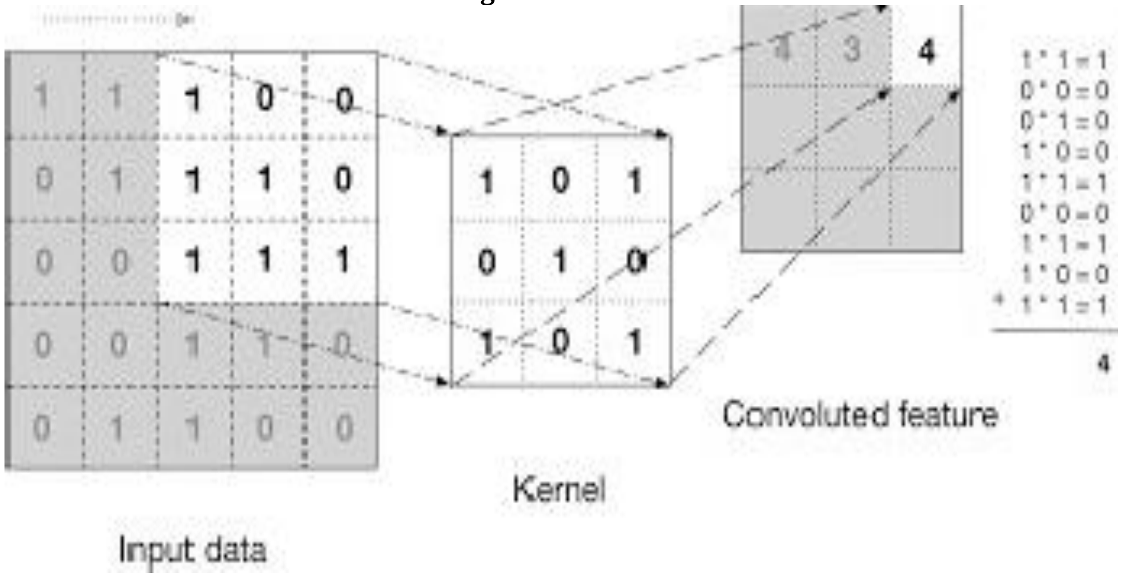


Fig 7.2: Convolution Operation

7.2.1 Building CNN Classifier

- Loading the datasets
- Reducing the dimensions of dataset using a feature extraction method PCA. Because hyperspectral dataset posses high dimensionality.
- Splitting the dataset into train & test data.
- Creating the input and target tensors that are compatible with the type of neural network model that is used for classification.
- CNN Classifier is Created with input layer, two convolution hidden layers, one hidden flatten layer, two dropout layer, two fully connected layers and output layer for performing classification using following code snippets:

```
model = Sequential()

model.add(Conv2D(C1, (3, 3), activation='relu',
input_shape=input_shape))

model.add(Conv2D(3*C1, (3, 3), activation='relu'))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(6*numPCAcomponents, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(16, activation='softmax'))
```

- Compiling and fitting the Classifier using the following code snippets:

```
sgd = SGD(lr=0.0001, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='categorical_crossentropy', optimizer=sgd,
metrics=['accuracy'])

model.fit(X_train, y_train, batch_size=32, epochs=30)
```

- Finally, evaluate the classifier by comparing groundtruth and predicted.

CHAPTER – 8

EXECUTION & RESULTS

This project has been successfully executed its source code. The source code of this project is written in colab notebook in python. We execute all the code written in the notebook by selecting the option "Run all" under "Cell" tab as shown in figure 8.1. The output of the source code will be display in the same page as shown in figure 8.2.

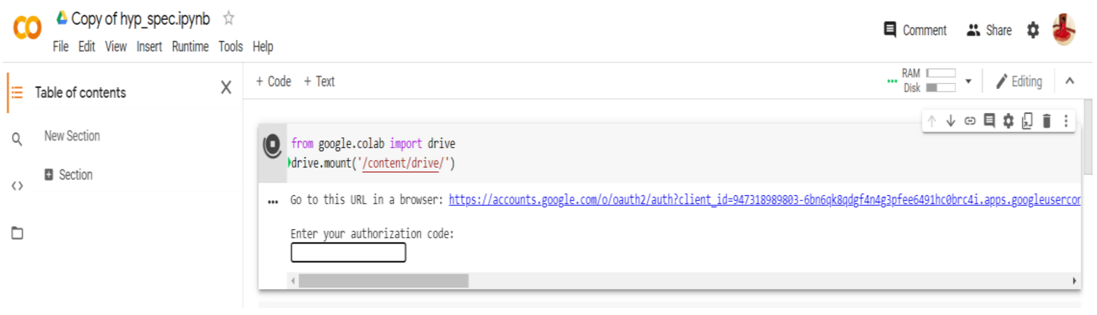
Upload File By Mounting Google Drive:

To mount your drive inside "mntDrive" folder execute following –

```
from google.colab import drive
drive.mount('/content/drive/')
```

Then you'll see a link, click on link, then allow access, copy the code that pops up, paste it at "Enter your authorization code:".

Now to see all data in your google drive you need to execute following:



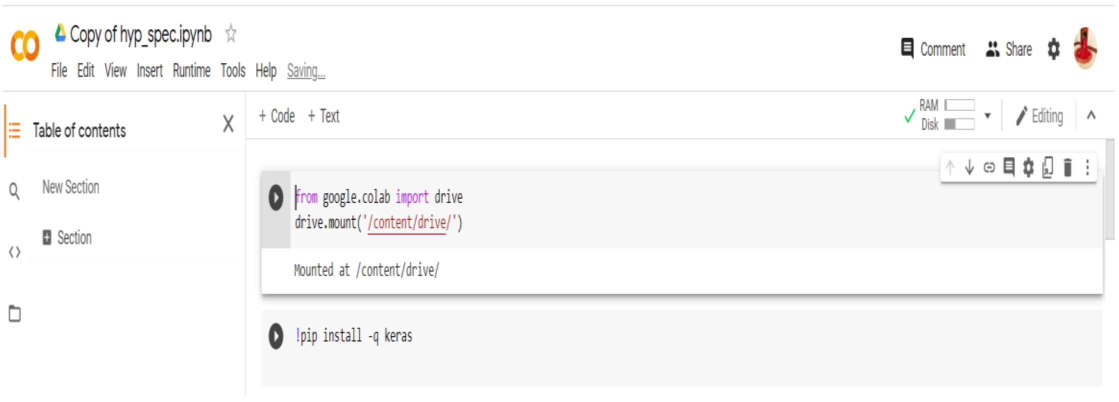
```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercon

Enter your authorization code:
```

Fig 8.1 Mounting Google Drive with Colab

File Hierarchy:

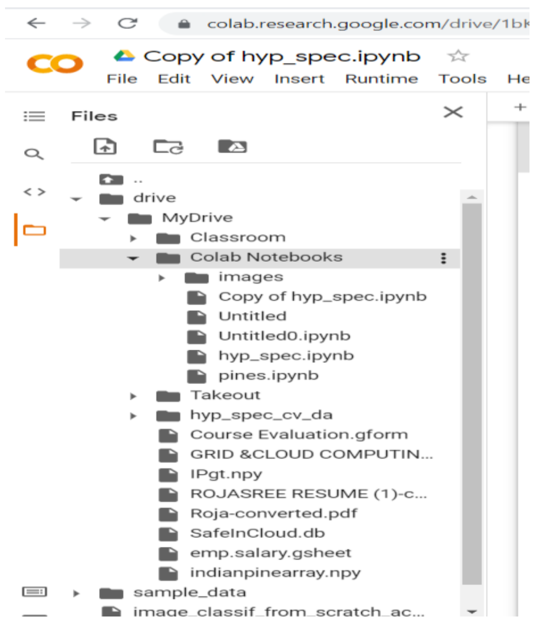
You can also see file hierarchy by clicking ">" at top left below the control buttons (CODE, TEXT, CELL).



```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

```
!pip install -q keras
```


colab.research.google.com/drive/1bk
Copy of hyp_spec.ipynb
File Edit View Insert Runtime Tools He
Files

```
..
drive
  MyDrive
    Classroom
    Colab Notebooks
      images
      Copy of hyp_spec.ipynb
      Untitled
      Untitled0.ipynb
      hyp_spec.ipynb
      pines.ipynb
    Takeout
    hyp_spec_cv_da
    Course Evaluation.gform
    GRID &CLOUD COMPUTIN...
    IPgt.npy
    ROJASREE RESUME (1)-c...
    Roja-converted.pdf
    SafeInCloud.db
    emp.salary.gsheet
    indianpinearray.npy
sample_data
image_classif_from_scratch_ac...
```

Fig 8.2 Opening Colab Notebooks in Colab

The file will be saved at “colab notebook” name. Now we can directly download from there, Or, you can just open file hierarchy and right clicking will give download option.

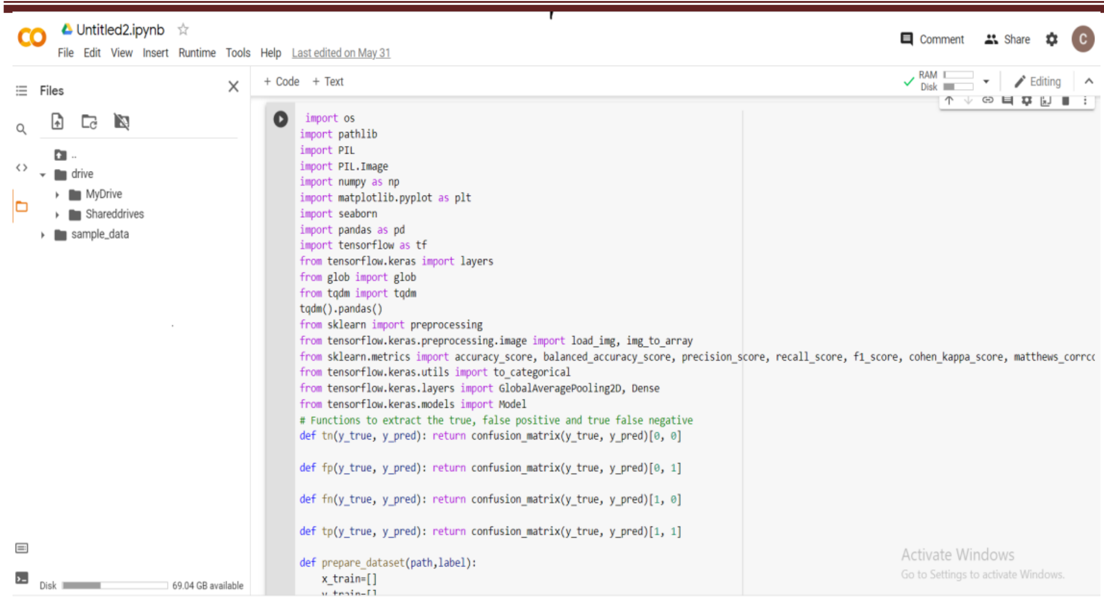
Now we need to open the coding terminal and need to import all the packages that are needed to execute the code. The package or module that is needed to be installed by executing the following code.

```
!pip install -q keras
```

To check whether that is installed or not execute the following code

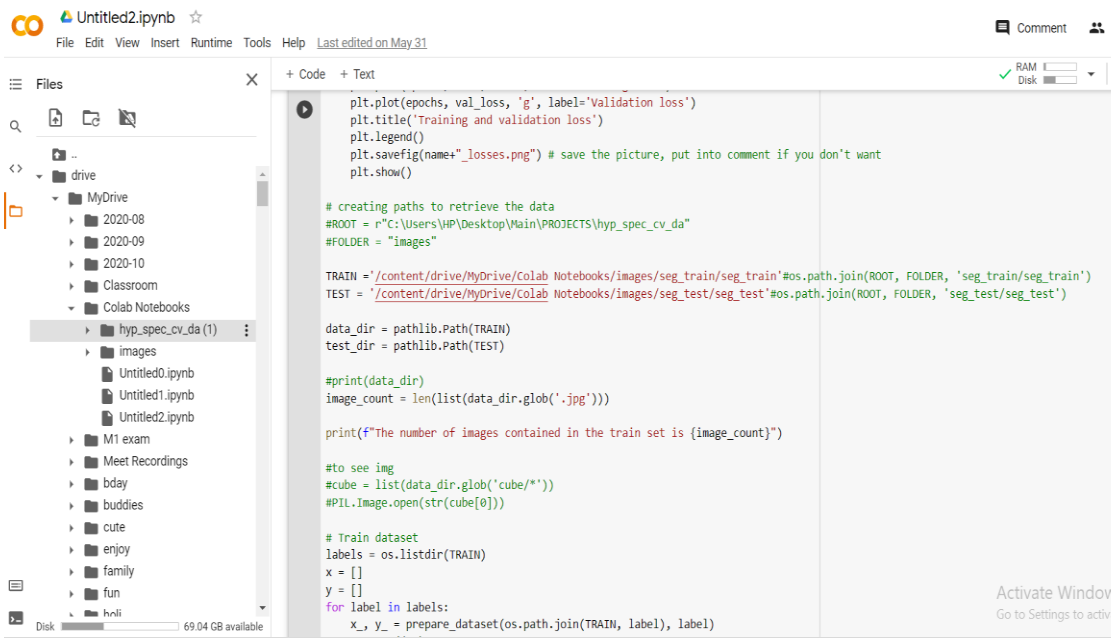
```
import keras
```

Now we need to the coding terminal and type the entire code in the coding terminal.Next we copy the path from seg-test and paste it over Test line and copy the another path from seg-train and paste it over Train line in the code.

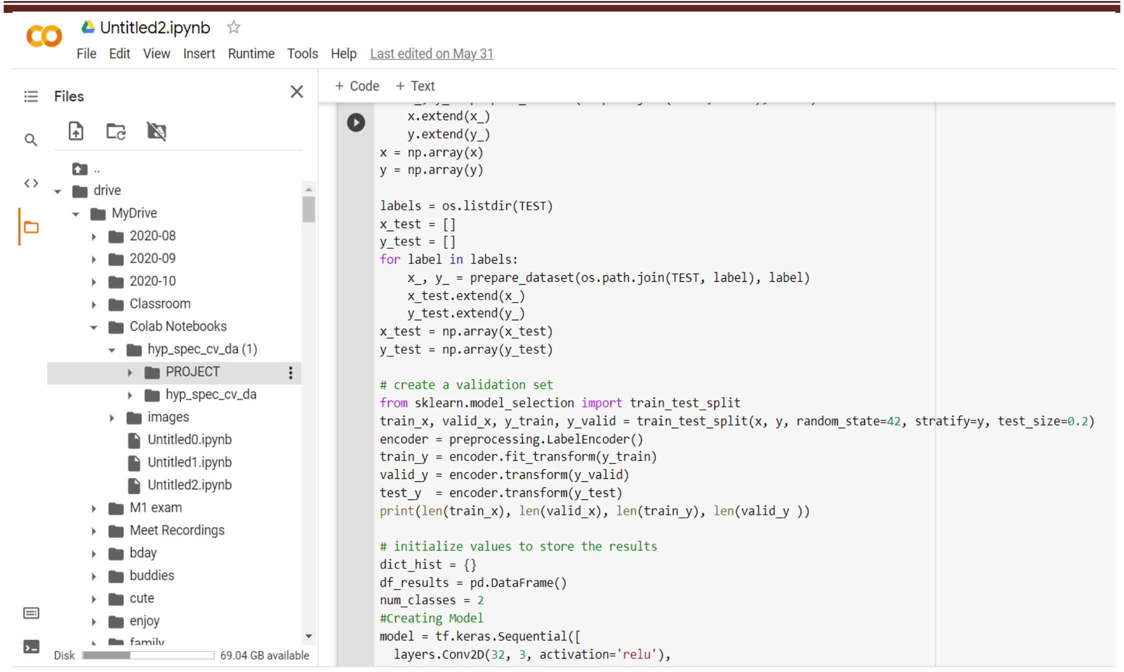


```
 import os
import pathlib
import PIL
import PIL.Image
import numpy as np
import matplotlib.pyplot as plt
import seaborn
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers
from glob import glob
from tqdm import tqdm
tqdm().pandas()
from sklearn import preprocessing
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from sklearn.metrics import accuracy_score, balanced_accuracy_score, precision_score, recall_score, f1_score, cohen_kappa_score, matthews_corrc
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense
from tensorflow.keras.models import Model
# Functions to extract the true, false positive and true false negative
def tn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 0]

def fp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 1]

def fn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 0]

def tp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 1]

def prepare_dataset(path,label):
    x_train=[]
```

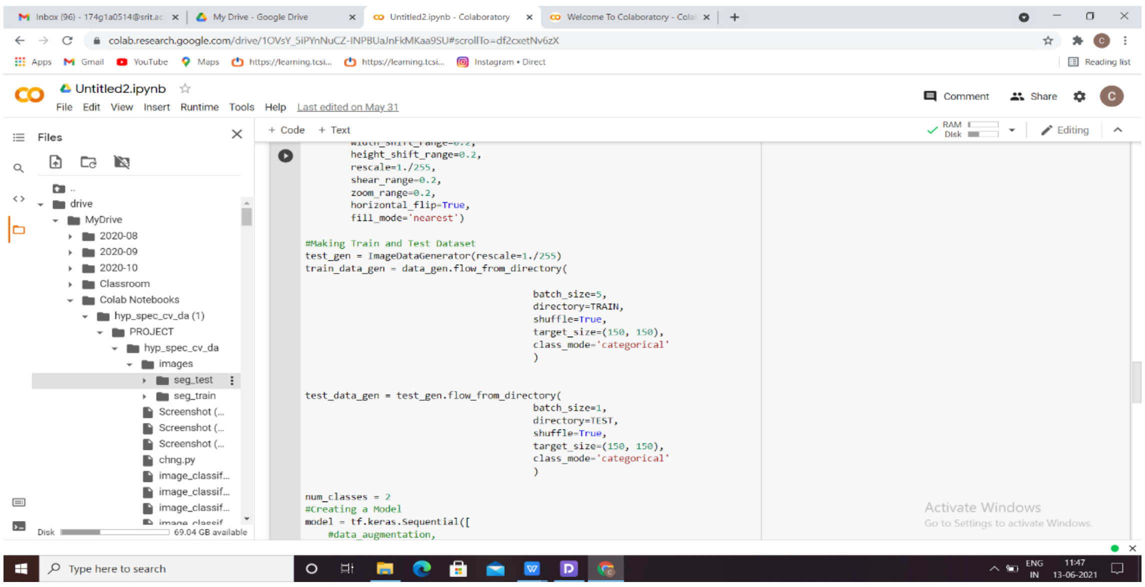
Fig 8.3 Executing python code in the section



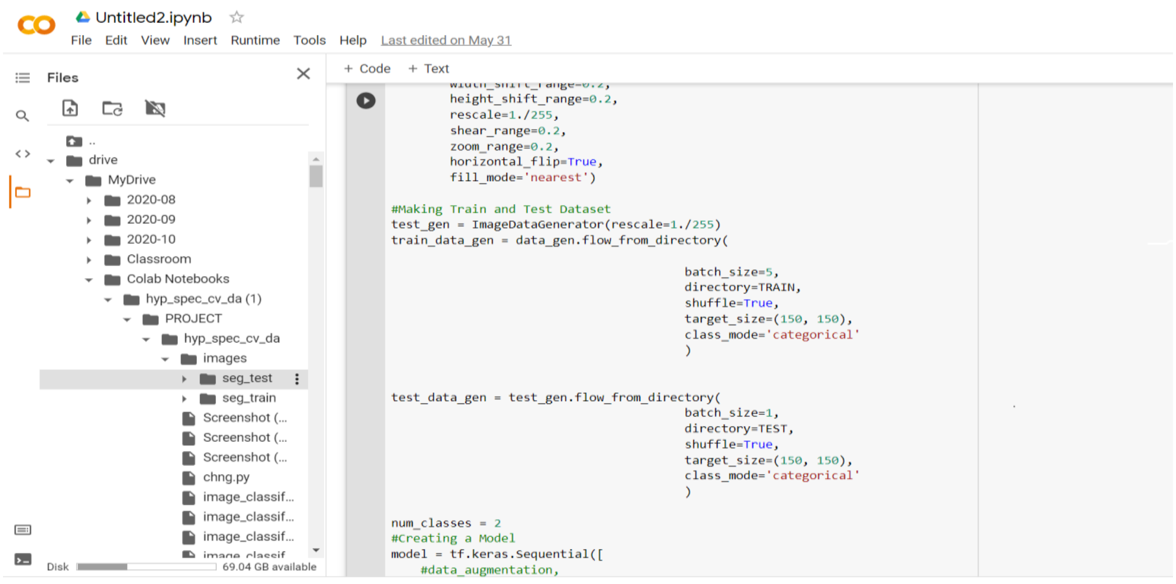
```
    plt.plot(epochs, val_loss, 'g', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()
    plt.savefig(name+"_losses.png") # save the picture, put into comment if you don't want
    plt.show()

# creating paths to retrieve the data
#ROOT = r"C:\Users\HP\Desktop\Main\PROJECTS\hyp_spec_cv_da"
#FOLDER = "images"

TRAIN ='/content/drive/MyDrive/Colab Notebooks/images/seg_train/seg_train'#os.path.join(ROOT, FOLDER, 'seg_train/seg_train')
TEST = '/content/drive/MyDrive/Colab Notebooks/images/seg_test/seg_test'#os.path.join(ROOT, FOLDER, 'seg_test/seg_test')

data_dir = pathlib.Path(TRAIN)
test_dir = pathlib.Path(TEST)

#print(data_dir)
image_count = len(list(data_dir.glob('.jpg')))

print(f"The number of images contained in the train set is {image_count}")

#to see img
#cube = list(data_dir.glob('cube/*'))
#PIL.Image.open(str(cube[0]))

# Train dataset
labels = os.listdir(TRAIN)
x = []
y = []
for label in labels:
    x_, y_ = prepare_dataset(os.path.join(TRAIN, label), label)
```



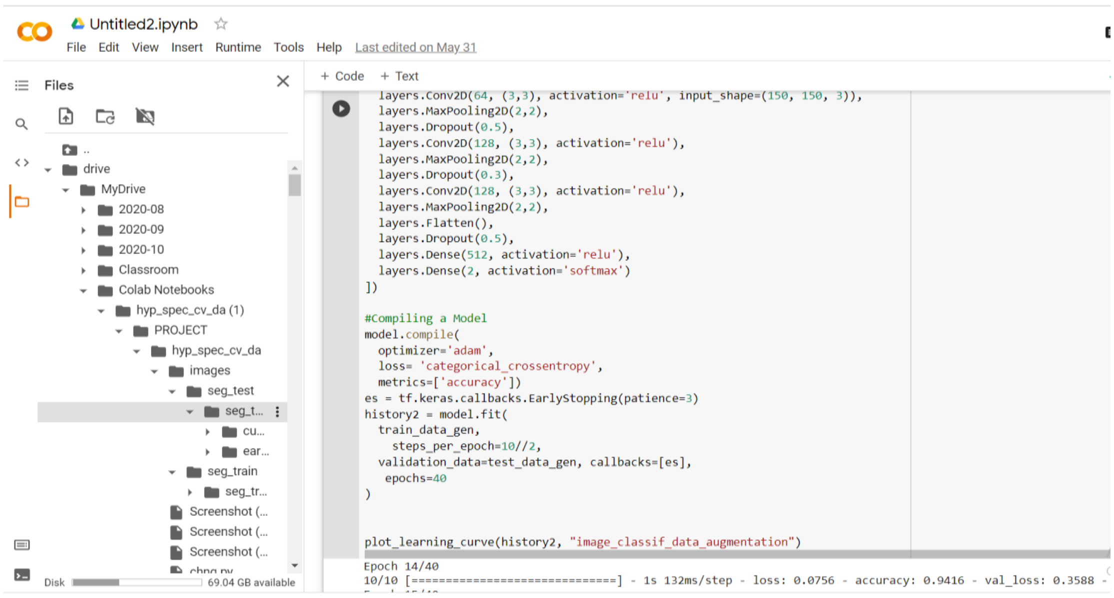
```
    x.extend(x_)
    y.extend(y_)
x = np.array(x)
y = np.array(y)

labels = os.listdir(TEST)
x_test = []
y_test = []
for label in labels:
    x_, y_ = prepare_dataset(os.path.join(TEST, label), label)
    x_test.extend(x_)
    y_test.extend(y_)
x_test = np.array(x_test)
y_test = np.array(y_test)

# create a validation set
from sklearn.model_selection import train_test_split
train_x, valid_x, y_train, y_valid = train_test_split(x, y, random_state=42, stratify=y, test_size=0.2)
encoder = preprocessing.LabelEncoder()
train_y = encoder.fit_transform(y_train)
valid_y = encoder.transform(y_valid)
test_y  = encoder.transform(y_test)
print(len(train_x), len(valid_x), len(train_y), len(valid_y ))

# initialize values to store the results
dict_hist = {}
df_results = pd.DataFrame()
num_classes = 2
#Creating Model
model = tf.keras.Sequential([
  layers.Conv2D(32, 3, activation='relu'),
```



```
        height_shift_range=0.2,
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')

#Making Train and Test Dataset
test_gen = ImageDataGenerator(rescale=1./255)
train_data_gen = data_gen.flow_from_directory(

                                        batch_size=5,
                                        directory=TRAIN,
                                        shuffle=True,
                                        target_size=(150, 150),
                                        class_mode='categorical'
                                        )


test_data_gen = test_gen.flow_from_directory(
                                        batch_size=1,
                                        directory=TEST,
                                        shuffle=True,
                                        target_size=(150, 150),
                                        class_mode='categorical'
                                        )

num_classes = 2
#Creating a Model
model = tf.keras.Sequential([
    #data_augmentation,
```



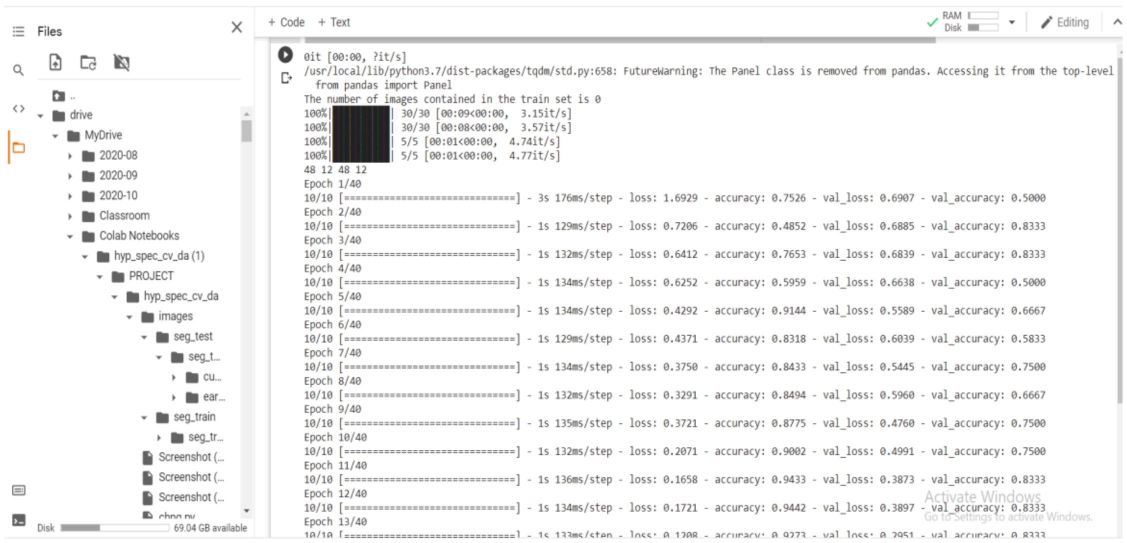
```
        height_shift_range=0.2,
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')

#Making Train and Test Dataset
test_gen = ImageDataGenerator(rescale=1./255)
train_data_gen = data_gen.flow_from_directory(

                                        batch_size=5,
                                        directory=TRAIN,
                                        shuffle=True,
                                        target_size=(150, 150),
                                        class_mode='categorical'
                                        )


test_data_gen = test_gen.flow_from_directory(
                                        batch_size=1,
                                        directory=TEST,
                                        shuffle=True,
                                        target_size=(150, 150),
                                        class_mode='categorical'
                                        )

num_classes = 2
#Creating a Model
model = tf.keras.Sequential([
    #data_augmentation,
```



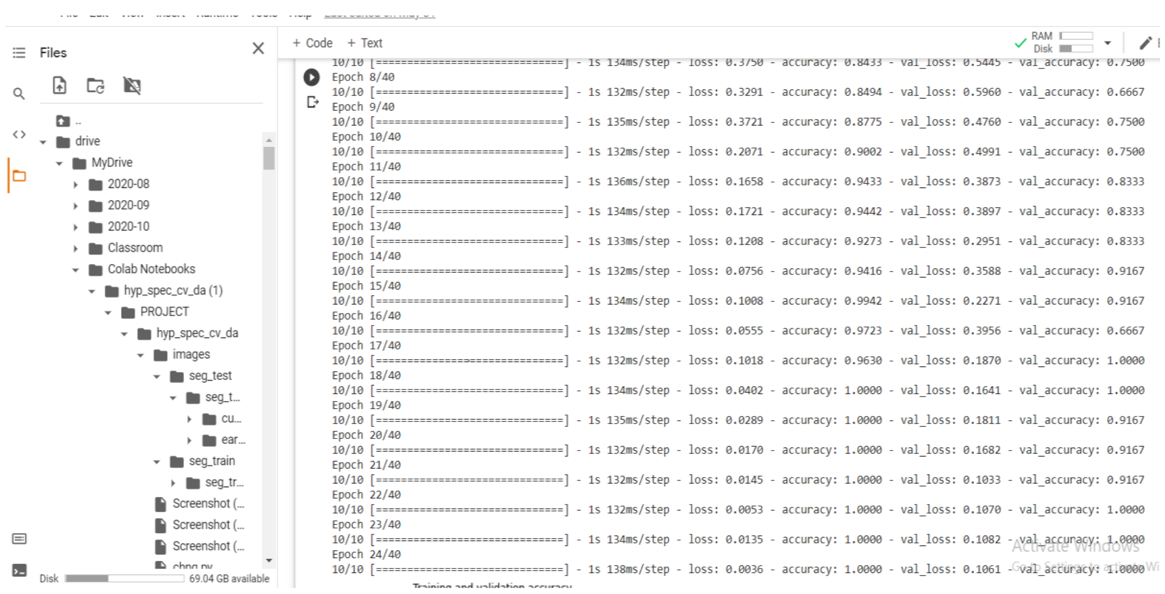
```
  layers.Conv2D(64, (3,3), activation='relu', input_shape=(150, 150, 3)),
  layers.MaxPooling2D(2,2),
  layers.Dropout(0.5),
  layers.Conv2D(128, (3,3), activation='relu'),
  layers.MaxPooling2D(2,2),
  layers.Dropout(0.3),
  layers.Conv2D(128, (3,3), activation='relu'),
  layers.MaxPooling2D(2,2),
  layers.Flatten(),
  layers.Dropout(0.5),
  layers.Dense(512, activation='relu'),
  layers.Dense(2, activation='softmax')
])

#Compiling a Model
model.compile(
  optimizer='adam',
  loss= 'categorical_crossentropy',
  metrics=['accuracy'])
es = tf.keras.callbacks.EarlyStopping(patience=3)
history2 = model.fit(
  train_data_gen,
    steps_per_epoch=10//2,
  validation_data=test_data_gen, callbacks=[es],
   epochs=40
)


plot_learning_curve(history2, "image_classif_data_augmentation")
```

```
Epoch 14/40
10/10 [==============================] - 1s 132ms/step - loss: 0.0756 - accuracy: 0.9416 - val_loss: 0.3588 -
```

After compiling this code ,these are the blue print’s of our project.

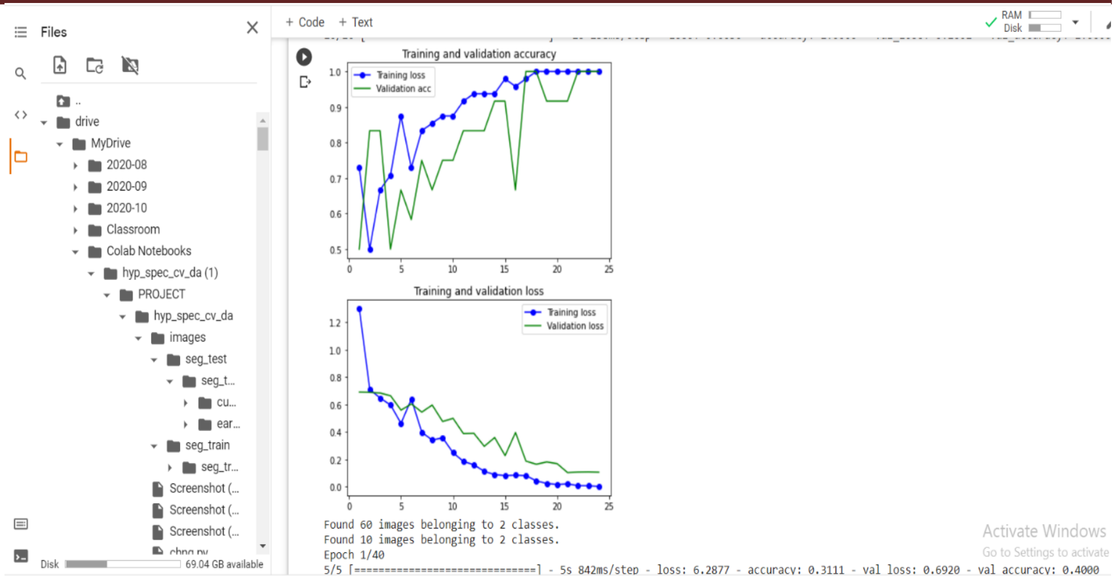


```
0it [00:00, ?it/s]
/usr/local/lib/python3.7/dist-packages/tqdm/std.py:658: FutureWarning: The Panel class is removed from pandas. Accessing it from the top-level
  from pandas import Panel
The number of images contained in the train set is 0
100%|██████████| 30/30 [00:09<00:00,  3.15it/s]
100%|██████████| 30/30 [00:08<00:00,  3.57it/s]
100%|██████████| 5/5 [00:01<00:00,  4.74it/s]
100%|██████████| 5/5 [00:01<00:00,  4.77it/s]
48 12 48 12
Epoch 1/40
10/10 [==============================] - 3s 176ms/step - loss: 1.6929 - accuracy: 0.7526 - val_loss: 0.6907 - val_accuracy: 0.5000
Epoch 2/40
10/10 [==============================] - 1s 129ms/step - loss: 0.7206 - accuracy: 0.4852 - val_loss: 0.6885 - val_accuracy: 0.8333
Epoch 3/40
10/10 [==============================] - 1s 132ms/step - loss: 0.6412 - accuracy: 0.7653 - val_loss: 0.6839 - val_accuracy: 0.8333
Epoch 4/40
10/10 [==============================] - 1s 134ms/step - loss: 0.6252 - accuracy: 0.5959 - val_loss: 0.6638 - val_accuracy: 0.5000
Epoch 5/40
10/10 [==============================] - 1s 134ms/step - loss: 0.4292 - accuracy: 0.9144 - val_loss: 0.5589 - val_accuracy: 0.6667
Epoch 6/40
10/10 [==============================] - 1s 129ms/step - loss: 0.4371 - accuracy: 0.8318 - val_loss: 0.6039 - val_accuracy: 0.5833
Epoch 7/40
10/10 [==============================] - 1s 134ms/step - loss: 0.3750 - accuracy: 0.8433 - val_loss: 0.5445 - val_accuracy: 0.7500
Epoch 8/40
10/10 [==============================] - 1s 132ms/step - loss: 0.3291 - accuracy: 0.8494 - val_loss: 0.5960 - val_accuracy: 0.6667
Epoch 9/40
10/10 [==============================] - 1s 135ms/step - loss: 0.3721 - accuracy: 0.8775 - val_loss: 0.4760 - val_accuracy: 0.7500
Epoch 10/40
10/10 [==============================] - 1s 132ms/step - loss: 0.2071 - accuracy: 0.9002 - val_loss: 0.4991 - val_accuracy: 0.7500
Epoch 11/40
10/10 [==============================] - 1s 136ms/step - loss: 0.1658 - accuracy: 0.9433 - val_loss: 0.3873 - val_accuracy: 0.8333
Epoch 12/40
10/10 [==============================] - 1s 134ms/step - loss: 0.1721 - accuracy: 0.9442 - val_loss: 0.3897 - val_accuracy: 0.8333
Epoch 13/40
10/10 [==============================] - 1s 133ms/step - loss: 0.1208 - accuracy: 0.9273 - val_loss: 0.2951 - val_accuracy: 0.8333
```



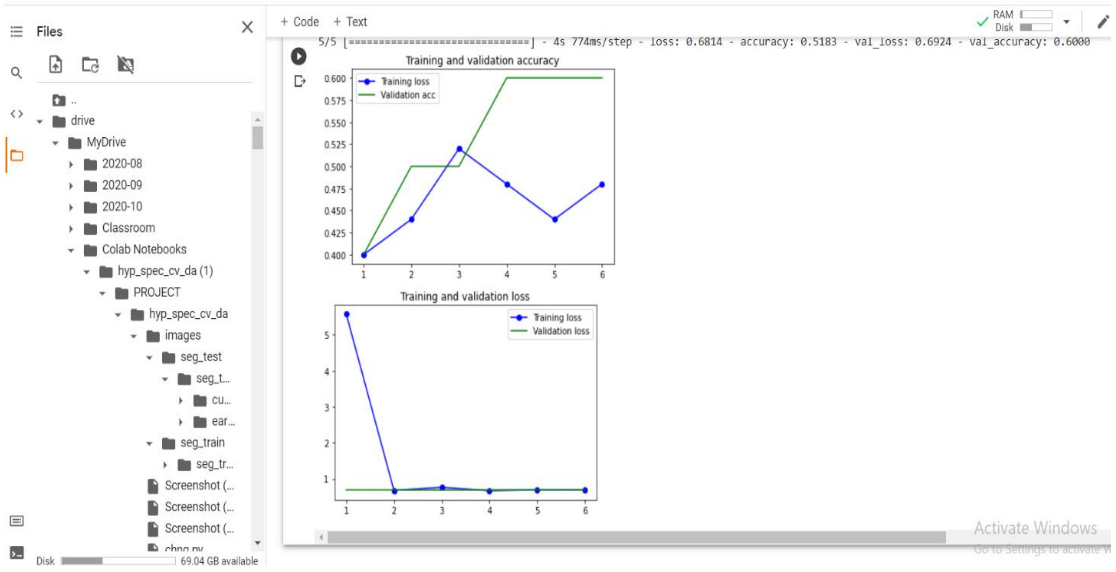
```
10/10 [==============================] - 1s 134ms/step - loss: 0.3750 - accuracy: 0.8433 - val_loss: 0.5445 - val_accuracy: 0.7500
Epoch 8/40
10/10 [==============================] - 1s 132ms/step - loss: 0.3291 - accuracy: 0.8494 - val_loss: 0.5960 - val_accuracy: 0.6667
Epoch 9/40
10/10 [==============================] - 1s 135ms/step - loss: 0.3721 - accuracy: 0.8775 - val_loss: 0.4760 - val_accuracy: 0.7500
Epoch 10/40
10/10 [==============================] - 1s 132ms/step - loss: 0.2071 - accuracy: 0.9002 - val_loss: 0.4991 - val_accuracy: 0.7500
Epoch 11/40
10/10 [==============================] - 1s 136ms/step - loss: 0.1658 - accuracy: 0.9433 - val_loss: 0.3873 - val_accuracy: 0.8333
Epoch 12/40
10/10 [==============================] - 1s 134ms/step - loss: 0.1721 - accuracy: 0.9442 - val_loss: 0.3897 - val_accuracy: 0.8333
Epoch 13/40
10/10 [==============================] - 1s 133ms/step - loss: 0.1208 - accuracy: 0.9273 - val_loss: 0.2951 - val_accuracy: 0.8333
Epoch 14/40
10/10 [==============================] - 1s 132ms/step - loss: 0.0756 - accuracy: 0.9416 - val_loss: 0.3588 - val_accuracy: 0.9167
Epoch 15/40
10/10 [==============================] - 1s 134ms/step - loss: 0.1008 - accuracy: 0.9942 - val_loss: 0.2271 - val_accuracy: 0.9167
Epoch 16/40
10/10 [==============================] - 1s 132ms/step - loss: 0.0555 - accuracy: 0.9723 - val_loss: 0.3956 - val_accuracy: 0.6667
Epoch 17/40
10/10 [==============================] - 1s 132ms/step - loss: 0.1018 - accuracy: 0.9630 - val_loss: 0.1870 - val_accuracy: 1.0000
Epoch 18/40
10/10 [==============================] - 1s 134ms/step - loss: 0.0402 - accuracy: 1.0000 - val_loss: 0.1641 - val_accuracy: 1.0000
Epoch 19/40
10/10 [==============================] - 1s 135ms/step - loss: 0.0289 - accuracy: 1.0000 - val_loss: 0.1811 - val_accuracy: 0.9167
Epoch 20/40
10/10 [==============================] - 1s 132ms/step - loss: 0.0170 - accuracy: 1.0000 - val_loss: 0.1682 - val_accuracy: 0.9167
Epoch 21/40
10/10 [==============================] - 1s 132ms/step - loss: 0.0145 - accuracy: 1.0000 - val_loss: 0.1033 - val_accuracy: 0.9167
Epoch 22/40
10/10 [==============================] - 1s 132ms/step - loss: 0.0053 - accuracy: 1.0000 - val_loss: 0.1070 - val_accuracy: 1.0000
Epoch 23/40
10/10 [==============================] - 1s 134ms/step - loss: 0.0135 - accuracy: 1.0000 - val_loss: 0.1082 - val_accuracy: 1.0000
Epoch 24/40
10/10 [==============================] - 1s 138ms/step - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.1061 - val_accuracy: 1.0000
```

Fig 8.4 Obataining the results of the code



```
Found 60 images belonging to 2 classes.
Found 10 images belonging to 2 classes.
Epoch 1/40
5/5 [==============================] - 5s 842ms/step - loss: 6.2877 - accuracy: 0.3111 - val loss: 0.6920 - val accuracy: 0.4000
```

Fig 8.5 Output of CV



```
5/5 [==============================] - 4s 774ms/step - loss: 0.6814 - accuracy: 0.5183 - val_loss: 0.6924 - val_accuracy: 0.6000
```

Fig 8.6 Output of DA

CHAPTER – 9

COMPARING METRICS

In the field of hyperspectral image classification, a widely used way for objective performance evaluation of different classification methods is calculating three accuracy indexes, i.e., the validation accuracy.

The accuracy indexes are obtained by comparing the classification results with the remaining data set .

Validation Accuracy:

VA or Validation accuracy is the number of accurately classified samples divided by the number of test samples, which can be calculated as follows:

$$VA = \sum_{i=1}^{C} Mii/N$$

where C is the number of classes. The confusion matrix M is obtained by comparing the classification map with the ground truth. Mii represents the number of samples belonging to class i and labeled as class i. N is the total number of test samples.

Training Loss:

TL represents the mean value of correctly classified pixels for each class, which is defined as follows:

$$\text{TL} = (\sum_{i=1}^{C} (Mii/ \sum_{j=1}^{C} Mij))/C$$

Datasets/ Classifiers	**CV**	
	LOSS	**ACC**
Custom Dataset	70.49	98.05

Table 9.1: Accuracy indexes of CV

Datasets/ Classifiers	**DA**	
	LOSS	**ACC**
Custom Dataset	68.93	52.03

Table 9.2: Accuracy indexes of DA

By comparing the Table 9.1 and Table 9.2 we can come to a conclusion that cross validation method is very appropriate for finding the accuracy and data augumentaion is for finding reducing the overfittting and obtaining the loss are better classifiers for classification of hyperspectral image.

CONCLUSION

In this project we performed classification of hyperspectral image by using machine learning and deep learning classifiers. The two metrics VA, TL are adopted for performance evaluation of classification methods. By comparing the metrics obtained from classification method we came to conclusion that deep learning methods are better for reducing the overfitting and increasing the accuracy in hyper spectral images.

BIBLIOGRAPHY

[1] M. J. Khan, H. S. Khan, A. Yousaf, K. Khurshid and A. Abbas, "Modern Trends in Hyperspectral Image Analysis: A Review," in IEEE Access, vol. 6, pp. 14118-14129, 2018.

[2] Rajesh Gogineni and Ashvini Chaturvedi, “Hyperspectral Image Classification” DOI: 10.5772/intechopen.88925 From the Edited Volume “Processing and Analysis of Hyperspectral Data”

[3] S. Li, Q. Hao, G. Gao and X. Kang, "The Effect of Ground Truth on Performance Evaluation of Hyperspectral Image Classification," in IEEE Transactions on Geoscience and Remote Sensing, vol. 56, no. 12, pp. 7195-7206, Dec. 2018.

[4] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi and J. A. Benediktsson, "Deep Learning for Hyperspectral Image Classification: An Overview," in IEEE Transactions on Geoscience and Remote Sensing, vol. 57, no. 9, pp. 6690-6709, Sept. 2019.

[5] Hosseiny, Benyamin & Rastiveis, Heidar & Daneshtalab, Somaye. (2019). Hyperspectral Image Classification By Exploiting Convolutional Neural Networks. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLII-4/W18. 535-540. 10.5194/isprs-archives-XLII-4-W18-535-2019.

[6] A. Thakur and D. Mishra, "Hyper spectral image classification using multilayer perceptron neural network & functional link ANN," 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, Noida, 2017, pp. 639-642.

[7] Ortaç, Gizem & Ozcan, Giyasettin. (2018). A Comparative Study For Hyperspectral Data Classification With Deep Learning And Dimensionality Reduction Techniques. Uludağ University Journal of The Faculty of Engineering. 23. 73 - 90. 10.17482/uumfd.435723.

[8] B. T. Abe, O. O.Olugbara and T. Marwala Hyperspectral Image Classification using Random Forests and Neural Networks Proceedings of the World Congress on Engineering and Computer Science 2012 Vol I

[9] https://medium.com/@prasad.pai/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9

[10] https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms /tutorial-random-forest-parameter-tuning-r/tutorial/