AIRCRAFT DETECTION

Industrial Training Report

Submitted in the partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Information Technology & Engineering

By

PALAK SINGHAL - 02196303116

Maharaja Surajmal Institute Of Technology
(Affiliated to Guru Gobind Singh Indraprastha University)
Janakpuri, New Delhi-58

October 2019

भारत सरकार
GOVERNMENT OF INDIA

रक्षा मंत्रालय
MINISTRY OF DEFENCE

DRDO/DTRL/HRD/ST/2019/58/2M

रक्षा अनुसंधान एवं विकास संगठन
Defence R&D Organization

DEFENCE TERRAIN RESEARCH LABORATORY
रक्षा भू-भाग अनुसंधान प्रयोगशाला
METCALFE HOUSE, DELHI – 110054

CERTIFICATE

This is to certify that **Ms. Palak Singhal** (B.Tech, IT) student of "Maharaja Surajmal Institute of Technology, has worked on **"Aircraft Detection using Deep Learning"** under the guidance of Sh. Ashwani Kumar, Scientist 'D' at DTRL, DRDO. She undertook her project from **3rd June 2019 to 16th July 2019** and has completed her project successfully.

Ashwani Kumar
Scientist-'D'
Project Guide

16/7/2019
Kamal Deep
Scientist-'D'
Head, HRD

MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY

Summer/Industrial Training Evaluation Form

F05 (MSIT-EXM-P-02)

(Year 20.19. -- 2020)

Details of the Student

Name....Palak Singhal.....
Roll No...02196303116.....
Branch and Semester..7th Sem..IT(E)
Mobile No...9315707093.....
E-mail ID...palak7013@gmail.com

Details of the Organisation

Name and address of organisation..Defence Research & Development org.....
Broader Area.....Terrain Mapping.....
Name of Instructor....A. Shrinivasan.....
Designation and Contact No..Sc 'D', 9899135794

Student Performance Record

	No. of days Scheduled for the training	Number of days actually attended	Curriculum Scheduled for the student	Curriculum actually covered by the student
Week 1	3	3	Introduction to Python	Introduction to Python
Week 2	3	3	Introduction to Keras and Spyder IDE	Introduction to Keras and Spyder IDE
Week 3	3	3	Introduction to Neural Networks	Introduction to Neural Networks
Week 4	3	3	Introduction to CNN and its layers	Introduction to CNN and its layers
Week 5	3	3	Understanding of Aircraft Detection & Dataset	Understanding of Aircraft Detection & Dataset
Week 6	3	3	Implementation of Aircraft Detection.	Implementation of Aircraft detection.

Palak Singhal
(Signature of the student)

Any comments or suggestions for the student performance during the training program (to be filled by instructor).....

.....

.....

(Signature of the Instructor)

along with S. KUMAR
HR/ASHWANI KUMAR
Sc 'D'/Scientist 'D'
CTRL' DRDO, Hqrs
Ministry of Defence, New Delhi-110054

Note : Every student has to fill and submit this Performa duly signed by his/her instructor to the faculty-

DECLARATION

I, Palak Singhal from IT(E), 7th Semester hereby, declare that the work which is being presented by me in the partial fulfilment of the degree of B. Tech submitted to **Maharaja Surajmal Institute of Technology, GGSIPU** is an authentic record of my own work on the project **"Aircraft Detection"** carried out from 3rd June, 2019 to 16th July, 2019 under the guidance of **Mr. Ashwani Kumar, Scientist 'D' at DTRL** from **DEFENSE RESEARCH AND DEVELOPMENT ORGANISATION**, Ministry of Defense, Metcalfe House, Delhi-54. I got sufficient knowledge in the field of Deep Learning and development from my mentor and understood the importance of self-work and the enthusiasm to do it.

PALAK SINGHAL
02196303116

ACKNOWLEDGEMENT

It gives me an immense pleasure in acknowledging the effort of my mentor for giving me his valuable time and full cooperation for undertaking this Practical Summer Training Program at **DRDO**. I am indebted to the members of the **DTRL department** for their wholehearted cooperation.

I would especially like to thank **Mr. Ashwani Kumar, Scientist 'D'** project mentor, without his guidance and support this training would not have been possible. His encouragement and experience helped to realize the practical aspect of Deep Learning and Python. He gave me ample support and help for accomplishment of my project. I feel grateful to him for giving me the opportunity to have a practical experience in this field. His knowledge and immense work experience helped me a lot in making this six weeks Practical Summer Training Program a great learning experience.

PALAK SINGHAL

TABLE OF CONTENT

LIST OF FIGURES

ABSTRACT

The program utilizes the complete understanding of Deep Learning for Aircraft Detection. It simply detects the probability of aircraft in the satellite's images dataset. The main challenge is to make the probability of detecting an object so accurate in order to get the exact and accurate result rate. In Aircraft Detection project a Dataset with 32,000 satellite images out of which 8,000 images are with aircraft and remaining 24,000 images are without aircraft images in them.

The program makes use of Python language as the core language to build Aircraft Detection. The whole work is carried on Spyder IDE. A Dataset is used to provide a set of training images to train the computer to make correct object detection from the given set of images. The main concept of Aircraft Detection is done with the help of object detection by using Convolutional Neural Networks (and all its layers).

The dataset is taken from kaggle (community of data scientists and machine learners).Dataset is divided into 28,000 (7,000 + 21,000) training images and 4,000 (1,000 + 3,000) test images. Aircraft Detection using Deep Learning is implemented in Python using Convolution Neural Networks (CNN) on Keras and IDE used is Spyder (Anaconda).

CHAPTER-1: COMPANY PROFILE

1.1 ABOUT DRDO

The Defence Research and Development Organisation (DRDO) is an agency of the Government of India, charged with the military's research and development, headquartered in New Delhi, India. DRDO was formed in 1958 by the merger of the Technical Development Establishment and the Directorate of Technical Development and Production of the Indian Ordnance Factories with the Defence Science Organisation. It is under the administrative control of the Ministry of Defence, Government of India. DRDO was then a small organization with 10 establishments or laboratories. Over the years, it has grown multi-directionally in terms of the variety of subject disciplines, number of laboratories, achievements and stature.

Today, DRDO is having more than 50 labs, engaged in developing Defence Technologies covering various disciplines like aeronautics, armaments, electronics, combat vehicles, engineering systems, instrumentation, missiles, advanced computing and simulation, special materials, naval systems, life sciences, training, information systems and agriculture. Over 5000 scientists and about 25,000 other scientific, technical and supporting personnel back DRDO. Several major projects are carried on for the development of arms and ammunitions, missiles, light combat aircrafts etc.

1.2 HISTORY

The DRDO was established in 1958 by amalgamating the Defence Science Organisation and some of the technical development establishments. A separate Department of Defence Research and Development was formed in 1980 which later on administered DRDO and its 50 laboratories/establishments. Most of the time the Defence Research Development Organisation was treated as if it was a vendor and the

Army Headquarters or the Air Headquarters were the customers. Because the Army and the Air Force themselves did not have any design or construction responsibility, they tended to treat the designer or Indian industry at par with their corresponding designer in the world market. If they could get a MiG-21 from the world market, they wanted a MiG-21 from DRDO.

DRDO started its first major project in surface-to-air missiles (SAM) known as Project Indigo in 1960s. Indigo was discontinued in later years without achieving full success. Project Indigo led to Project Devil, along with Project Valiant, to develop short-range SAM and ICBM in the 1970s. Project Devil itself led to the later development of the Prithvi missile under the Integrated Guided Missile Development Programme (IGMDP) in the 1980s.

IGMDP was an Indian Ministry of Defence programme between the early 1980s and 2007 for the development of a comprehensive range of missiles, including the Agni missile, Prithvi ballistic missile, Akash missile, Trishul missile and Nag Missile. In 2010, then defence minister A. K. Antony ordered the restructuring of the DRDO to give 'a major boost to defence research in the country and to ensure effective participation of the private sector in defence technology.

1.3 PROJECTS

- The DRDO is responsible for the ongoing **Light Combat Aircraft**. The LCA is intended to provide the Indian Air Force with a modern, fly by wire, multi-role fighter, as well as develop the aviation industry in India.
- The DRDO has also developed two unmanned aerial vehicles – the Nishant tactical UAV and the Lakshya (*Target*) Pilotless Target Aircraft (PTA). The Lakshya PTA has been ordered by all three services for their gunnery target training requirements. Efforts are on to develop the PTA further, with an improved all digital flight control system, and a better turbojet engine. The Nishant is a hydraulically launched short-ranged UAV for the tactical battle area. It is currently being evaluated by the Indian Navy and the Indian Paramilitary forces as well.

- DRDO has been responsible for the indigenization of key defence stores and equipment. DRDO has assisted Hindustan Aeronautics Limited and the IAF with the indigenization of spares and assemblies for several aircraft.
- The DRDO has steadily increased its radar development. The result has been substantial progress in India's ability to design and manufacture high power radar systems with locally sourced components and systems.
- DRDO has worked extensively on high speed computing given its ramifications for most of its defence projects. These include supercomputers for computational flow dynamics, to dedicated microprocessor designs manufactured in India for flight controllers and the like, to high speed computing boards built around Commercial off the Shelf (COTS) components, similar to the latest trends in the defence industry.

1.4 CORE COMPETENCE

Department of Defence Research and Development (R&D) is working for indigenous development of weapons, sensors & platforms required by the three wings of the Armed Forces. To fulfill this mandate, Dep't of Defence Research and Development (R&D), is closely working with academic institutions, Research and Development (R&D) Centers and production agencies of Science and Technology (S&T) Ministries/Depts. in Public & Civil Sector including defence Public Sector Undertakings & Ordnance Factories.

Defense Research and Development organization is the most strategic organization that builds arms and ammunitions for the country's defense shield. It makes use of the most efficient technologies ever to create the best in order to protect the country on the enemy's ground. Also it comes up with new innovation every year.

1.5 ABOUT DTRL

Terrain evaluation activities started in DRDO on 19th February, 1964 with the creation of Terrain Assessment Cell (TEC).The basic objective of TEC was to develop techniques for assessing the evaluation of the terrain and the possibility of mobility in remote areas. TEC was renamed as Defense Lands Research Laboratory (DTRL),

giving status of full-fledged laboratory on December 10, 1981.DTRL was created on self-accounting unit on September 17, 1998.

1.5.1 CHARTER OF DUTIES

1) To develop a reliable system for prediction of terrain characteristics and derivation of military potential of various types of terrains based on modern techniques of terrain evaluation.
2) To develop infrastructure, competence and instrumentation in the latest techniques related to terrain research
3) To interact with the Services & other agencies including Universities for basic and applied aspects of terrain research.
4) To develop methods for automatic feature extraction from remotely sensed data.
5) To propagate the techniques of terrain evaluation in Defence services and conduct training in this field.
6) To become the nodal agency for acquiring and processing of high resolution optical, thermal and hyper spectral image.

CHAPTER-2: TOOLS AND TECHNOLOGY USED

2.1 INTRODUCTION

Aircraft Detection is an Object Detection based project. Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

Object Detection involves use of Neural Networks, Deep Learning. The project has been implemented using Convolutional Neural Networks in Keras (Python). Spyder has been used as IDE.

2.1.1 IMAGE RECOGNITION

Image recognition is the ability of software to identify objects, places, people, writing and actions in images. It is used to perform a large number of machine-based visual tasks, such as labeling the content of images with meta-tags, performing image content search, guiding autonomous robots, self-driving cars and accident avoidance systems.

Software for image recognition requires deep machine learning. Performance is best on convolutional neural networks as the specific task.

Current and future applications of image recognition include smart photo libraries, targeted advertising, the interactivity of media, accessibility for the visually impaired and enhanced research capabilities.

Google, Facebook, Microsoft, Apple and Pinterest are among the many companies that are investing significant resources and research into image recognition.

Image recognition is a part of computer vision and a process to identify and detect an object or attribute in a digital video or image. Computer vision is a broader term which includes methods of gathering, processing and analyzing data from the real world. The data is high dimensional and produces numerical or symbolic information in the form of decisions.

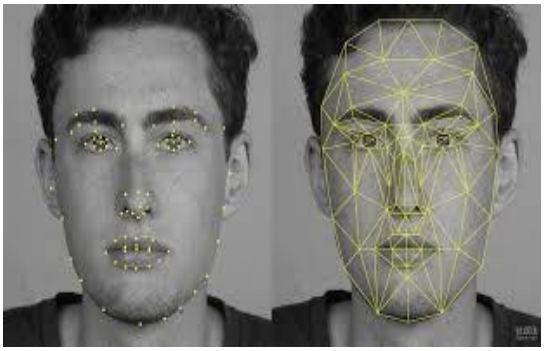

Figure 2.1: Image showing Face Recognition

2.1.2 OBJECT DETECTION

Object Detection is the process of finding real-world object instances like car, bike, TV, flowers, and humans in still images or Videos. It allows for the recognition, localization, and detection of multiple objects within an image which provides us with a much better understanding of an image as a whole.

Object detection is the craft of detecting instances of a certain class, like animals, humans and many more in an image or video. It is commonly used in applications such as image retrieval, security, surveillance, and advanced driver assistance systems (ADAS).

Object Detection can be done via multiple ways:

- Feature-Based Object Detection
- Deep Learning Object Detection etc.

Object Detection, a subset of Computer Vision, involves detecting instances of objects from a particular class in an image. Its use cases are endless, be it Tracking objects, Video surveillance, Pedestrian detection, Anomaly detection, People Counting, Self-driving cars or Face detection, the list goes on.

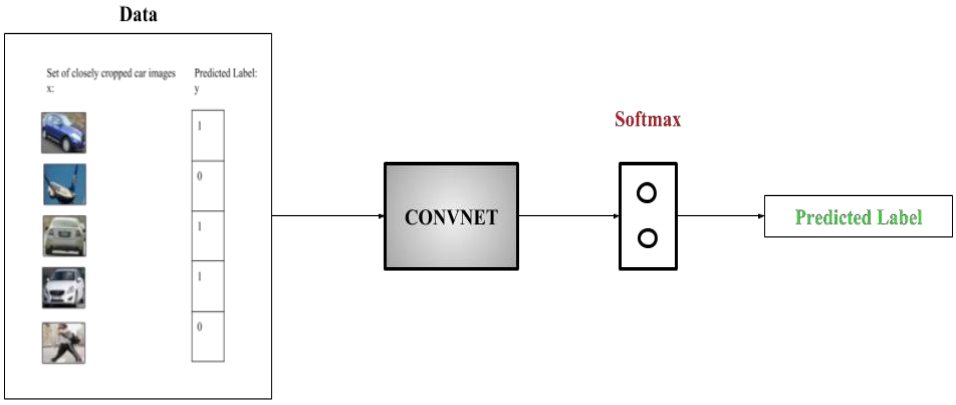

Figure 2.2: Visualization of Object Detection

There are various applications of object detection be it:

1. Face Recognition
2. Self-driving cars
3. Security
4. Industrial Quality check
5. People Counting etc.

2.2 WHAT IS DEEP LEARNING?

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on artificial neural networks. Learning can be supervised, semi-supervised or unsupervised.

Deep Learning architecture has a Deep Neural Network(NN), Recurrent Neural Networks(RNN), Convolutional Neural networks(CNN) have been applied to fields including Computer Vision, Speech Recognition, Object Recognition, Social Networking Filtering, Machine translation, Drug design, Medical image analysis etc.

Deep learning is a class of machine learning algorithms that use multiple layers to progressively extract higher level features from raw input. For example, in image processing, lower layers may identify edges, while higher layer may identify human-meaningful items such as digits/letters or faces. The "deep" in "deep learning" refers to the number of layers through which the data is transformed.

In an image recognition application, the raw input may be a matrix of pixels, the first representational layer may abstract the pixels and encode edges, the second layer may compose and encode arrangements of edges, the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level *on its own*. (Of course, this does not completely obviate the need for hand-tuning; for example, varying numbers of layers and layer sizes can provide different degrees of abstraction.)

2.3 NEURAL NETWORKS

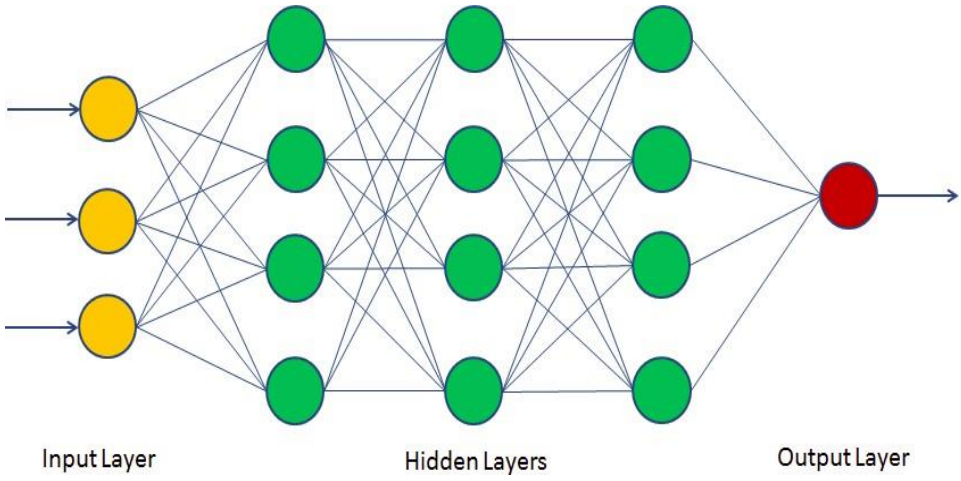

Figure 2.3: Diagram of Neural Network

A **neural network** is a network or circuit of neurons, or in a modern sense, a neural network is a biological neural network, made up of real biological neurons, for solving artificial intelligence (AI) problems. The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1.

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are

numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

Neural networks help us cluster and classify. You can think of them as a clustering and classification layer on top of the data you store and manage. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on. Neural networks can also extract features that are fed to other algorithms for clustering and classification; so you can think of deep neural networks as components of larger machine learning applications involving algorithms for reinforcement learning, classification and regression.

In a regular Neural Network there are three types of layers:

- Input Layer
- Hidden Layer
- Output Layer

2.3.1 WHAT ARE NEURONS?

As the name suggests, neural networks were inspired by the neural architecture of a human brain, and like in a human brain the basic building block is called a Neuron. Its functionality is similar to a human neuron, i.e. it takes in some inputs and fires an output. In purely mathematical terms, a neuron in the machine learning world is a placeholder for a mathematical function, and its only job is to provide an output by applying the function on the inputs provided.

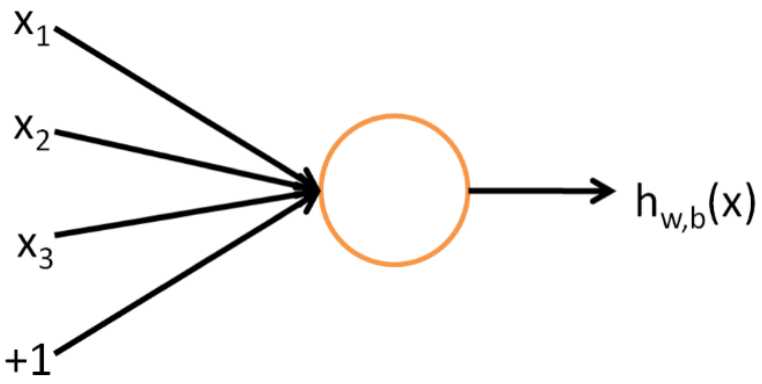

Figure 2.4: Structure of Neuron

2.3.2 TYPES OF NEURAL NETWORKS

There are seven types of Neural Networks:-

1) Multilayer perceptron (MLP)
2) Convolutional neural network (CNN)
3) Recursive neural network (RNN)
4) Recurrent neural network (RNN)
5) Long short-term memory (LSTM)
6) Sequence-to-sequence models
7) Shallow neural networks.

2.4 CONVOLUTIONAL NEURAL NETWORKS (CNN)

Convolutional Neural Networks, like neural networks, are made up of neurons with learnable weights and biases. Each receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output.

The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on Convolutional Neural Networks. Neural networks, as its name suggests, is a machine learning technique which is modeled after the brain structure. It comprises of a network of learning units called neurons. These neurons learn how to convert input signals (e.g. picture of a cat) into corresponding output signals (e.g. the label "cat"), forming the basis of automated recognition.

Let's take the example of automatic image recognition. The process of determining whether a picture contains a cat involves an activation function. If the picture resembles prior cat images the neurons have seen before, the label *"cat"* would be activated. Hence, the more labeled images the neurons are exposed to, the better it learns how to recognize other unlabeled images. We call this the process of training neurons.

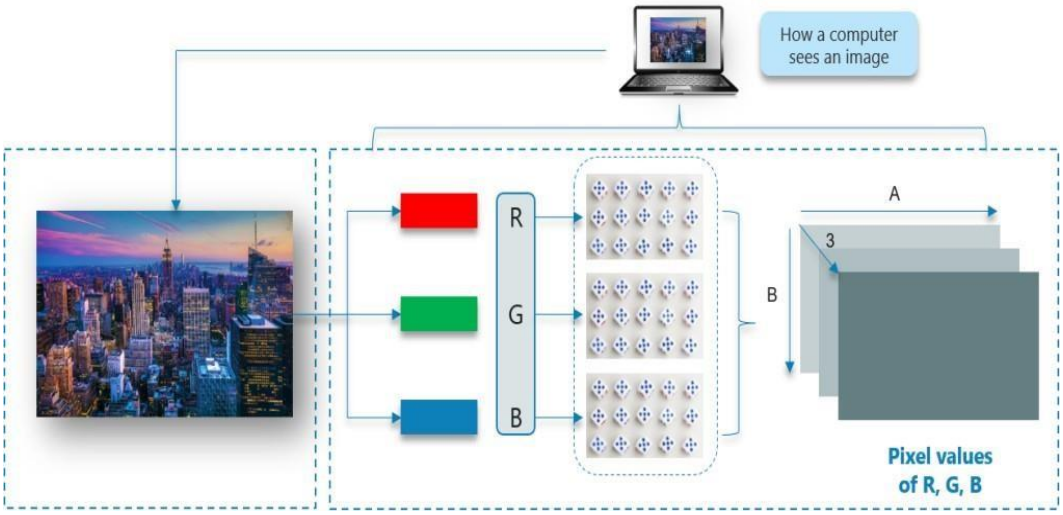

Figure 2.5: How a computer percepts an image

A Convolutional Neural Network consists of 4 layers are:

1. Convolution Layer
2. Activation Layer
3. Pooling Layer
4. Full Connectedness (Fully Connected Layer).

2.4.1 CONVOLUTION LAYER

Convolution layer is the layer where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user-specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on

It computes the output volume by computing dot product between all filters and image patch. Suppose we use total 12 filters for this layer we'll get output volume of dimension 32 x 32 x 12 (Input is 32 x 32 x 3).

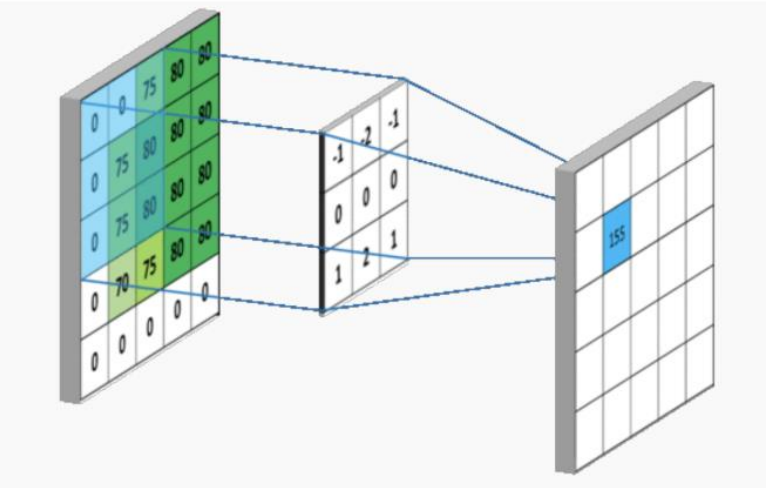

Figure 2.6: Visualizing Convolution Layer

We have **4 steps** for convolution:

- Line up the feature and the image.
- Multiply each image pixel by corresponding feature pixel.
- Add the values and find the sum.
- Divide the sum by the total number of pixels in the feature.

In a convolutional layer, we are basically applying multiple filters at over the image to extract different features. But most importantly, we are learning those filters

2.4.2 ACTIVATION LAYER

Activation Layer transform function only activates a node if the input is above a certain quantity, while the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship with the dependent variable.

There are various types of activation functions i.e. Tanh, ReLU, Leaky ReLU, Sigmoid function etc.

The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back. So, it can be written as f(x) = max (0, x).

The Leaky ReLU is one of the most well-known. It is the same as ReLU for positive numbers. But instead of being 0 for all negative values, it has a constant slope (less than 1.).

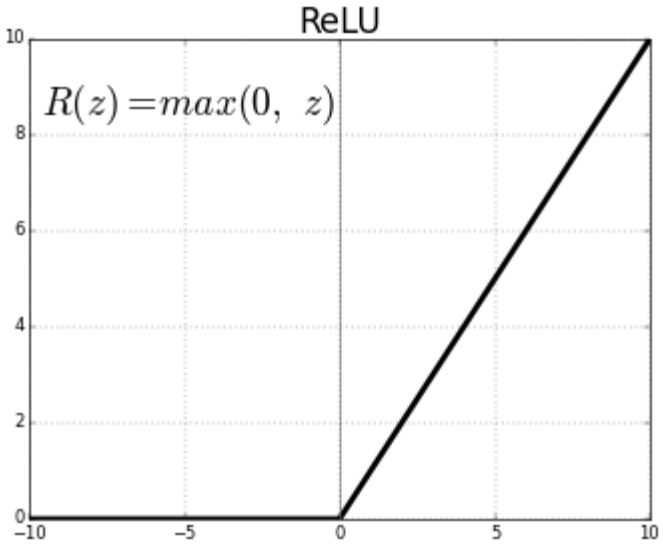

Figure 2.7: ReLU Activation Function

The tanh(z) function is a rescaled version of the sigmoid, and its output range is [− 1,1] instead of [0,1]. The general reason for using a Tanh function in some places instead of the sigmoid function is because since data is centered around 0, the derivatives are higher. A higher gradient helps in a better learning rate.

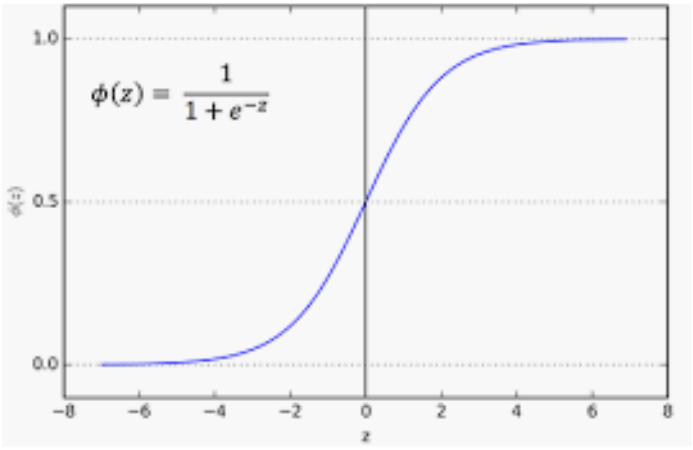

Figure 2.8: Tanh Function

There are other alternatives, but both practitioners and researchers have generally found an insufficient benefit to justify using anything other than ReLU. In general practice as well, ReLU has found to be performing better than sigmoid or tanh functions.

2.4.3 POOLING LAYER

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or down sampling which reduces the dimensionality of each map but retains the important information

In this layer, we shrink the image stack into a smaller size. Pooling is done after passing through the activation layer.

Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling take the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling. On the other hand, Average pooling take the average of the rectified feature map.

We do this by implementing the following 4 steps:-

- Pick a window size (usually 2 or 3)
- Pick a stride (usually 2)
- Walk your window across your filtered images.
- From each window, take the maximum value.

It is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast, reduces memory and prevents from over fitting. Two common types of are Max. Pooling and Average Pooling. Max. Pooling is used here.

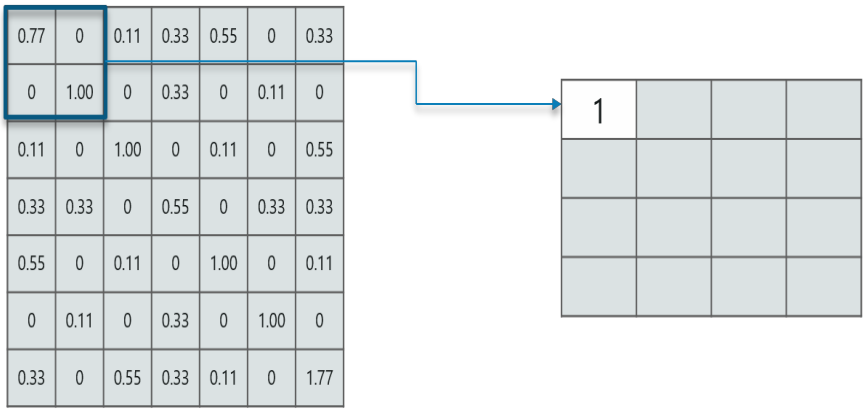

Figure 2.9: Picking a Window size and stride size of 2

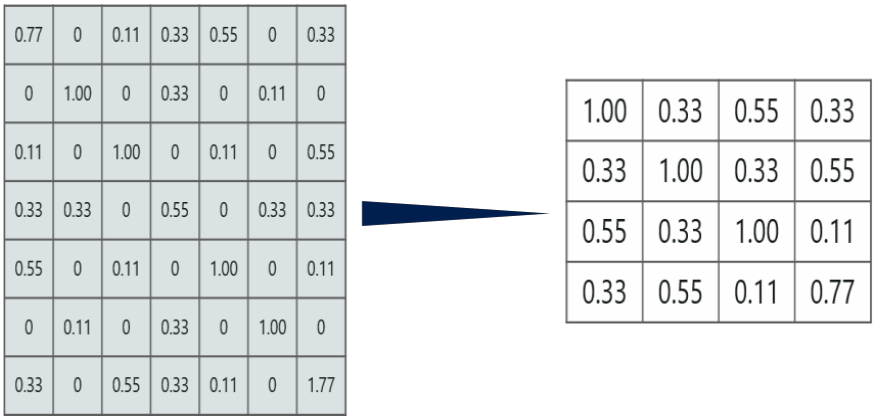

Figure 2.10: After applying filter

2.4.4 FULLY CONNECTED LAYER

Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP. It is regular neural network layer which takes input from the previous layer and computes the class scores and outputs the 1-D array of size equal to the number of classes.

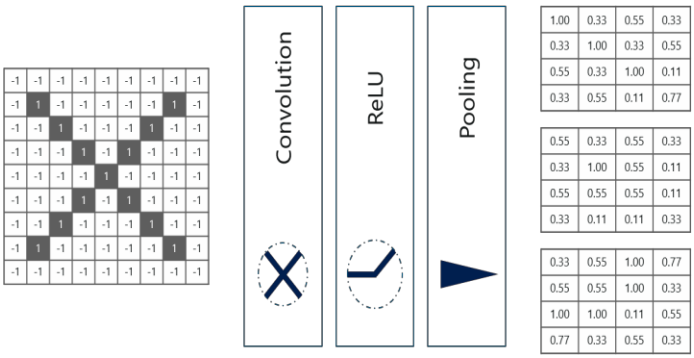

Figure 2.11: Last layers in the network are fully connected

We need to perform the 3 operations in an iteration after the first pass. So after the second pass we arrive at a 2×2 matrix as shown above: The last layers in the network are fully connected, meaning that neurons of preceding layers are connected to every neuron in subsequent layers. Also, fully connected layer is the final layer where the classification actually happens. Here we take our filtered and shrinked images and put them into one single list.

2.4.5 WHAT DO CNN LAYERS LEARN?

- Each CNN layer learns filters of increasing complexity.
- The first layers learn basic feature detection filters: edges, corners, etc.
- The middle layers learn filters that detect parts of objects. For faces, they might learn to respond to eyes, noses, etc.
- The last layers have higher representations: they learn to recognize full objects, in different shapes and positions.

2.5 PYTHON

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear logical code for small and large-scale projects.

2.5.1 CHARACTERISTICS OF PYTHON

Python is one of the most widely used language over the web. Few of them are here:

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

2.6 KERAS LIBRARY

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit etc. Designed to enable fast experimentation with deep neural networks. It focuses on being user- friendly, modular and extensible. It was developed as a part of the research effort of the project ONEIROS.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics Processing Units (GPU) and Tensor processing units (TPU).

CHAPTER-3: DEMONSTRATION OF TECHNOLOGY

3.1 INTRODUCTION

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. It is one of the most used field of Image Recognition. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

It is widely used in computer vision tasks such as face detection, face recognition, video object co-segmentation. It is also used in tracking objects, for example tracking a ball during a football match, tracking movement of a cricket bat, or tracking a person in a video.

Aircraft Detection is an object detection project. It simply detects the probability of aircraft in the satellite's images dataset. It is based on Convolution neural networks (CNN), deep learning. "Planes in Satellite Imagery" is used as dataset taken from Kaggle (community of data scientists and machine learners). It has 32,000 satellite images with 8,000 images with aircraft and 24,000 images without images. It makes use of Python language as the core language. Keras is used as implementation library. The whole work is carried on Spyder IDE in Anaconda.

3.2 ABOUT DATASET

3.2.1 CONTEXT

Satellite imagery provides unique insights into various markets, including agriculture, defense and intelligence, energy, and finance. New commercial imagery providers, such as Planet and BlackSky, are using constellations of small satellites to exponentially increase the amount of images of the earth captured every day. This flood of new imagery is outgrowing the ability for organizations to manually look at each image that gets captured, and there is a need for machine learning and computer vision algorithms to help automate the analysis process.

The aim of this dataset is to help address the difficult task of detecting the location of airplanes in satellite images. Automating this process can be applied to many issues including monitoring airports for activity and traffic patterns, and defense intelligence. Continuously updates will be made to this dataset as new Planet imagery released. Current images were collected as late as July 2017.

3.2.2 CONTENT

Provided is a zipped directory planesnet.zip that contains the entire dataset as .png image chips. Each individual image filename follows a specific format: {label} __ {scene id} __ {longitude} _ {latitude}.png.

Label: Valued 1 or 0, representing the "plane" class and "no-plane" class, respectively. Scene id: The unique identifier of the PlanetScope visual scene the image chip was extracted from. The scene id can be used with the Planet API to discover and download the entire scene.

longitude_latitude: The longitude and latitude coordinates of the image center point, with values separated by a single underscore.

The dataset is also distributed as a JSON formatted text file planesnet.json. The loaded object contains data, label, scene_ids, and location lists.

The pixel value data for each 20x20 RGB image is stored as a list of 1200 integers within the data list. The first 400 entries contain the red channel values, the next 400 the green, and the final 400 the blue. The image is stored in row-major order, so that the first 20 entries of the array are the red channel values of the first row of the image. The list values at index *i* in labels, scene_ids, and locations each correspond to the *i*-th image in the data list.

3.2.3 CLASS LABELS

The "plane" class includes 8000 images. Images in this class are near-centered on the body of a single airplane, with the majority of the plane's wings, tail, and nose also visible. Examples of different aircraft sizes, orientations, and atmospheric collection conditions are included. Example images from this class are shown below.

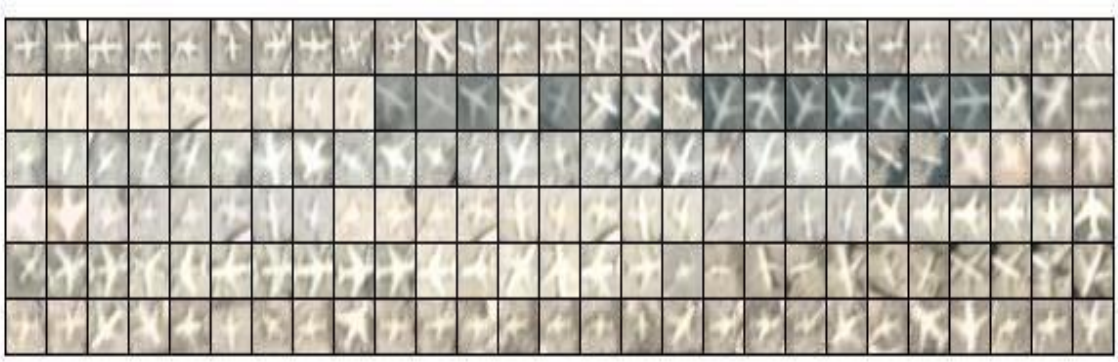

Figure 3.1: Dataset Images with Airplane

The "no-plane" class includes 24000 images. A third of these are a random sampling of different landcover features - water, vegetion, bare earth, buildings, etc. - that do not include any portion of an airplane. The next third are "partial planes" that contain a portion of an airplane, but not enough to meet the full definition of the "plane" class. The last third are "confusers" - chips with bright objects or strong linear features that resemble a plane - that have previously been mislabeled by machine learning models. Example images from this class are shown below.

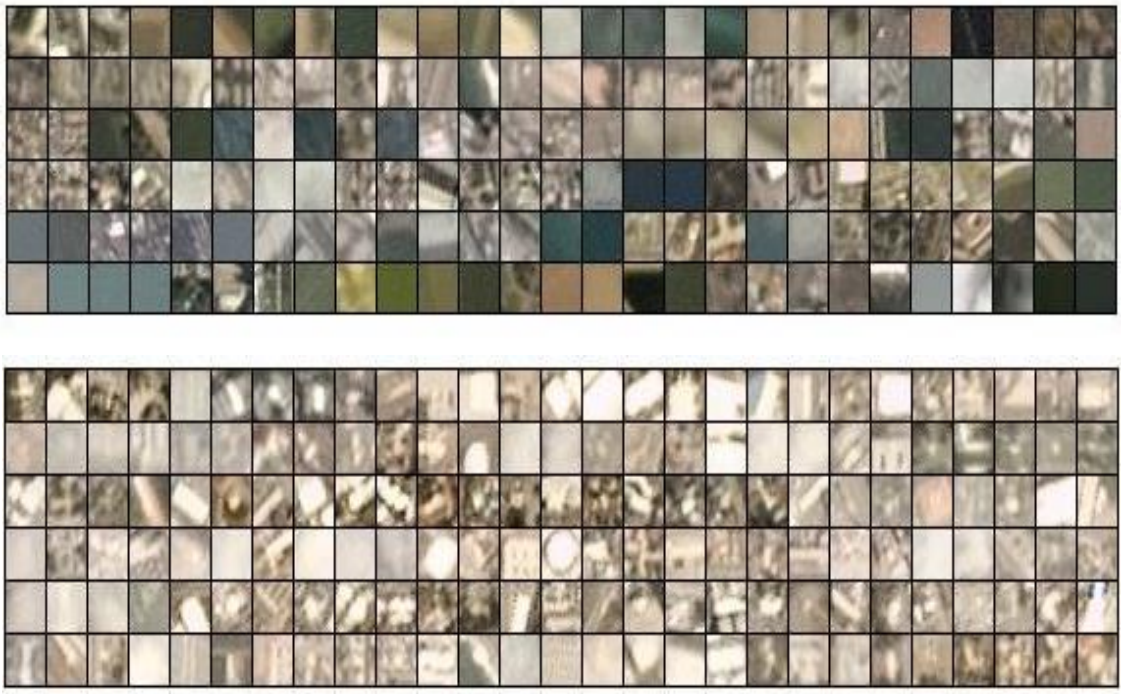

Figure 3.2: Dataset Images without Airplane

3.2.4 ACKNOWLEDGEMENT

Satellite imagery used to build PlanesNet is made available through Planet's Open California dataset, which is openly licensed. As such, this dataset is also available under the same CC-BY-SA license. Users can sign up for a free Planet account to search, view, and download their imagery and gain access to their API.

3.3 IMPLEMENTING PROJECT

A great way to build a Convolutional neural network (CNN). The Keras library in Python makes it pretty simple to build a CNN. Computers see images using pixels. Pixels in images are usually related. For example, a certain group of pixels may signify an edge in an image or some other pattern. Convolutions use this to help identify images. A convolution multiplies a matrix of pixels with a filter matrix or 'kernel' and sums up the multiplication values. Then the convolution slides over to the next pixel and repeats the same process until all the image pixels have been covered.

3.3.1 ARRANGING DATASET

The satellite's dataset, taken from kaggle, contained the mixed images of both with and without airplanes. Before starting the actual implementation of the project, rearrangement of dataset was done. The format of dataset was as follows:-

Dataset was divided into subfolders:

1. Train (Training Images)
2. Test (Testing Images)

Each subfolder was further divided into subfolders:

1. One (With Airplane)
2. Zero (Without Airplane)

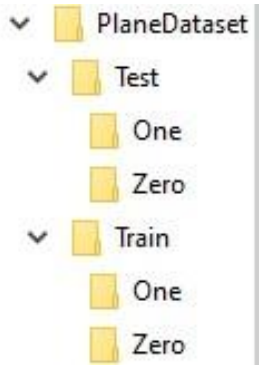

Figure 3.3: Dataset Arrangement

3.3.2 BUILDING THE CNN MODEL

Keras modules for CNN model and layers are imported first. The model type that we will be using is Sequential. Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer.

We use the 'add()' function to add layers to our model. Our model has 2 layers of Convlution2D layers. These are convolution layers that will deal with our input images, which are seen as 2-dimensional matrices.32 in both the layers, are the number of nodes or filter in each layer. This number can be adjusted to be higher or lower, depending on the size of the dataset. In our case, 32 work well, so we will stick with this for now.

```
# Initialising the CNN
classifier = Sequential()

# step 1 - Convolution and Activation function
classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Adding second layer
classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Step 3 - Flattening
classifier.add(Flatten())

# step 4 - Full Connection
classifier.add(Dense(output_dim = 128, activation = 'relu'))
classifier.add(Dense(output_dim = 1, activation = 'sigmoid'))

```

Figure 3.4: Building CNN Model

Kernel size is the size of the filter matrix for our convolution. So a kernel size of 3 means we will have a 3x3 filter matrix i.e. 3 rows and 3 columns in filters.

Activation is the activation function for the layer. The activation function we will be using for our first 2 layers is the ReLU, or Rectified Linear Activation to remove the negative pixels to increase non-linearity. This activation function has been proven to work well in neural networks. Our first layer also takes in an input shape. This is the shape of each input image, 64,64,3 as seen earlier on, with the 3 signifying that the images are colored (rgb).

Now come to the Pooling layer, here we have used Maxpooling as MaxPooling2D with pool_size or stride 2. There are 2 pooling layers, 1 after each Convolution2D layer. In between the Convolution2D layers and the dense layer, there is a 'Flatten' layer. Flatten serves as a connection between the convolution and dense layers.

'Dense' is the layer type we will use in for our output layer. Dense is a standard layer type that is used in many cases for neural networks. We have 2 dense layers 1st for specifying nodes of hidden layer i.e. 128 and ReLu is used as activation function and 2nd for specifying nodes of final output with activation function "Sigmoid" for binary outcome.

3.3.3 COMPILING THE MODEL

Compiling the model takes three parameters: optimizer, loss and metrics. The optimizer controls the learning rate. We will be using 'Adam' as our optimizer.

Adam is generally a good optimizer to use for many cases. The Adam optimizer adjusts the learning rate throughout training. The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer.

We used 'binary_crossentropy' for our loss function. This is the most common choice for classification. A lower score indicates that the model is performing better. To make things even easier to interpret, we will use the 'accuracy' metric to see the accuracy score on the validation set when we train the model.

3.3.4 IMPORTING AND PRE-PROCESSING DATASET

We have used ImageDataGenerator from Keras' preprocessing module to preprocess the images. It takes various parameters. We have specified four of them i.e. rescale to rescale pixel values of images between 0 and 1, shear_range as 0.2, zoom_range as 0.2 and horizontal_flip. Imported test_set and training_set through ImageDataGenerator model directly from the directory, specifying target_size as (64, 64), batch_size as 32, class_mode as binary.

3.3.5 TRAINING THE DATASET

To train, we will use the 'fit()' function on our model with the following parameters: training data (training_set), target data (test_set), validation data, and the number of epochs

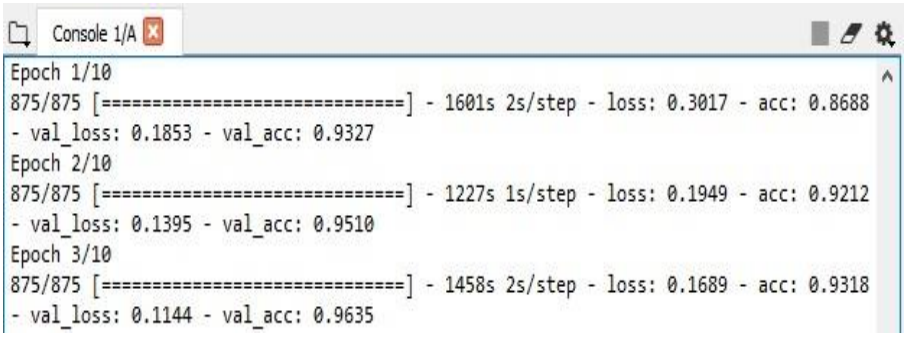

Figure 3.5: Training Dataset

For our validation data, we will use the test_set provided to us in our dataset, which we have split into images with and without aircraft.

The number of epochs is the number of times the model will cycle through the data. The more epochs we run, the more the model will improve, up to a certain point. After that point, the model will stop improving during each epoch. For our model, we will set the number of epochs to 10.

3.3.6 MAKING PREDICTIONS

To see the actual predictions that our model has made for the test data, we can use the predict function. The predict function will give an array with 2 numbers. These numbers

are the probabilities that the input image represents each digit (0 or 1). The array index with the highest number represents the model prediction. The sum of each array equals 1 (since each number is a probability). The one with probability greater than 0.5 is considered as image with airplane otherwise without airplane.

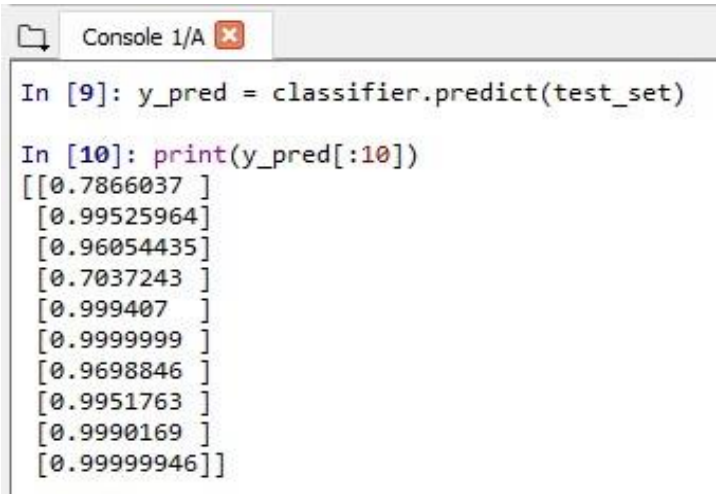

Figure 3.6: Predicting Results

CHAPTER-4: PROJECT SCREENSHOTS

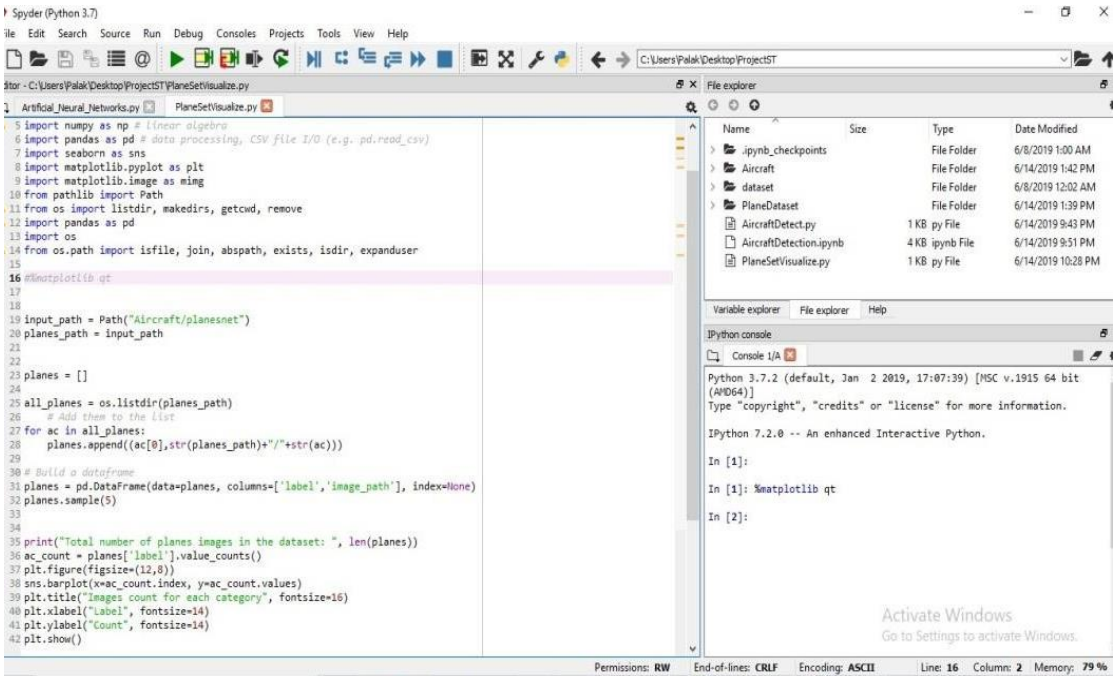

Figure 4.1: Spyder IDE

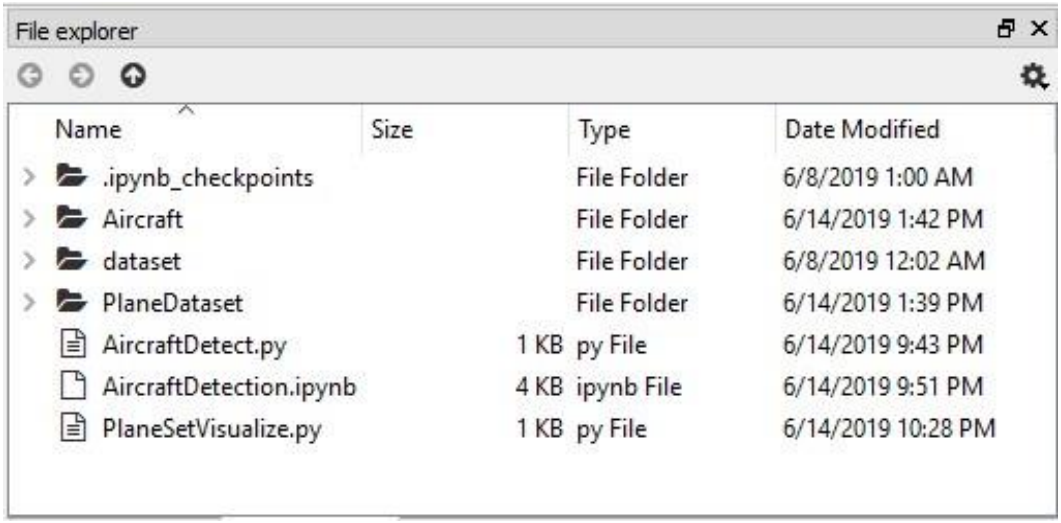

Figure 4.2: File Explorer

Variable explorer

Name	Type	Size	Value
ac	str	1	1__20170729_233954_0c42__-11…
ac_count	Series	(2,)	Series object of pandas.core.series module
all_planes	list	32000	['0__20140723_181317_0905__-…
ax	object	(5, 4)	ndarray object of numpy module
i	int	1	19
item	tuple	2	(25748, Series)
planes	DataFrame	(32000, 2)	Column names: label, image_path
random_samples	list	20	[('1', 'Aircraft\planesnet/1…
sample	tuple	2	('1', 'Aircraft\planesnet/1_…

Figure 4.3: Variable Explorer

Figure 4.4: Dataset with Aircraft images

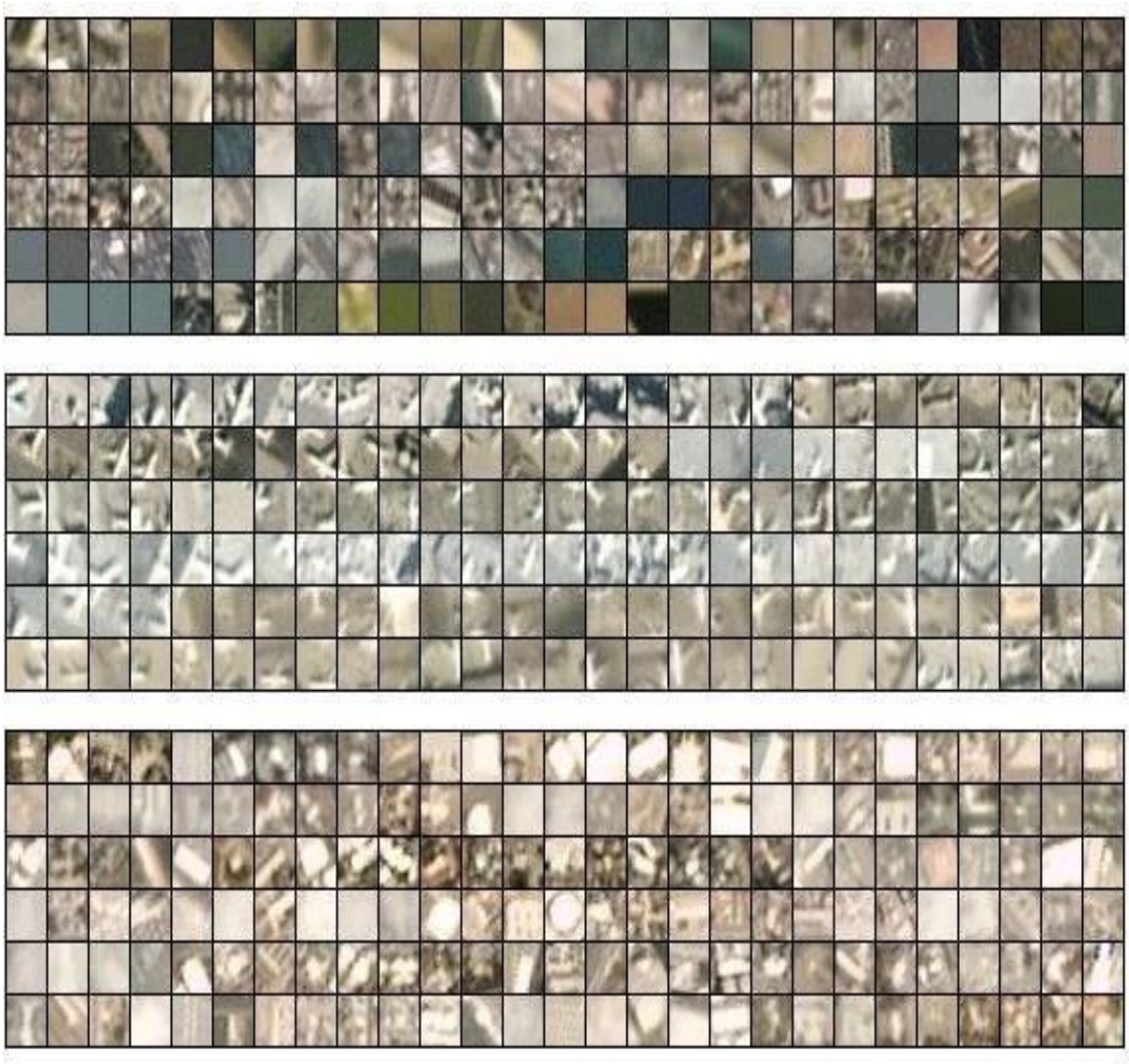

Figure 4.5: Dataset without Aircraft images

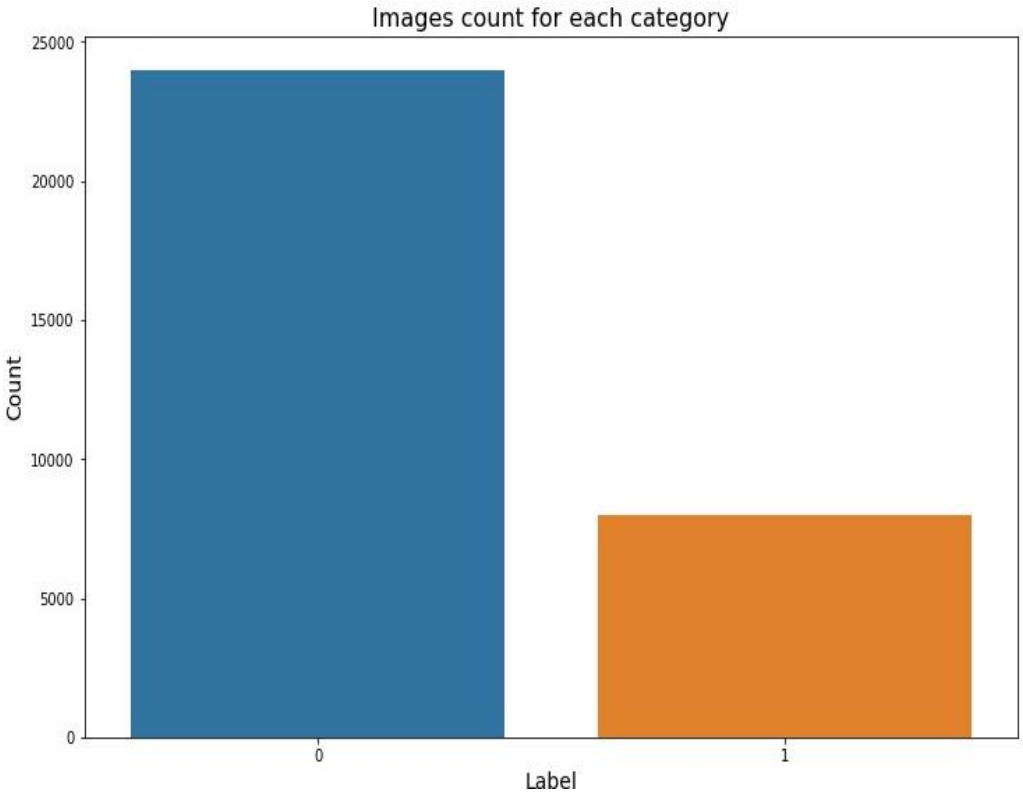

Figure 4.6: Categorical representation of the dataset

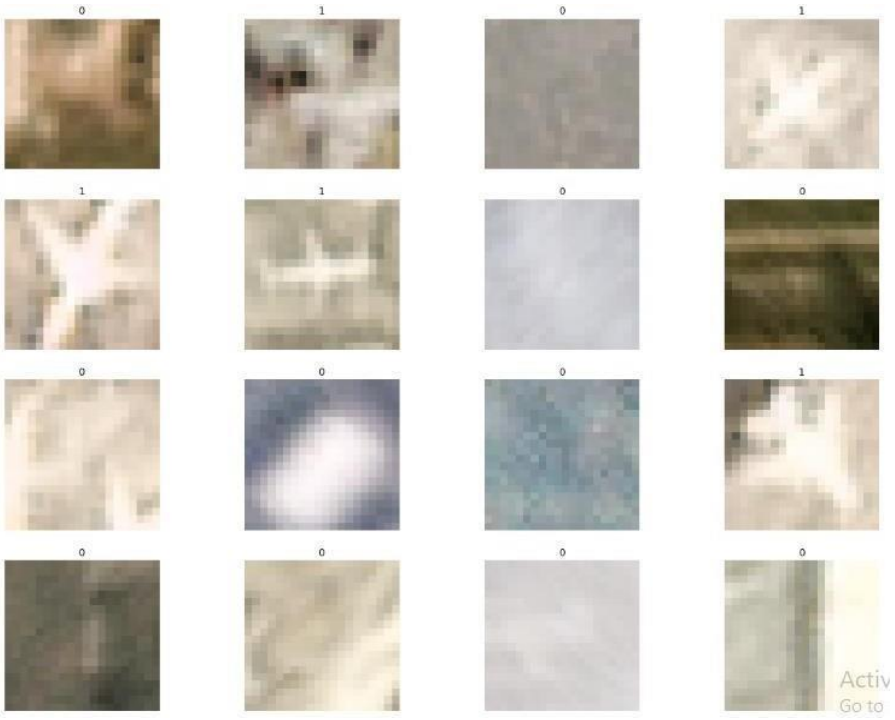

Figure 4.7: Magnified images of dataset

AircraftDetect.py

```
# Convolution Neural Network

# Part-1 Building the CNN

# Importing the Keras Libraries and Packages
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

# Initialising the CNN
classifier = Sequential()

# step 1 - Convolution and Activation function
classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Adding second layer
classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Step 3 - Flattening
classifier.add(Flatten())

# step 4 - Full Connection
classifier.add(Dense(output_dim = 128, activation = 'relu'))
classifier.add(Dense(output_dim = 1, activation = 'sigmoid'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# Part-2 Fitting the CNN to the images
from keras.preprocessing.image import ImageDataGenerator

```

Figure 4.8: Project Code

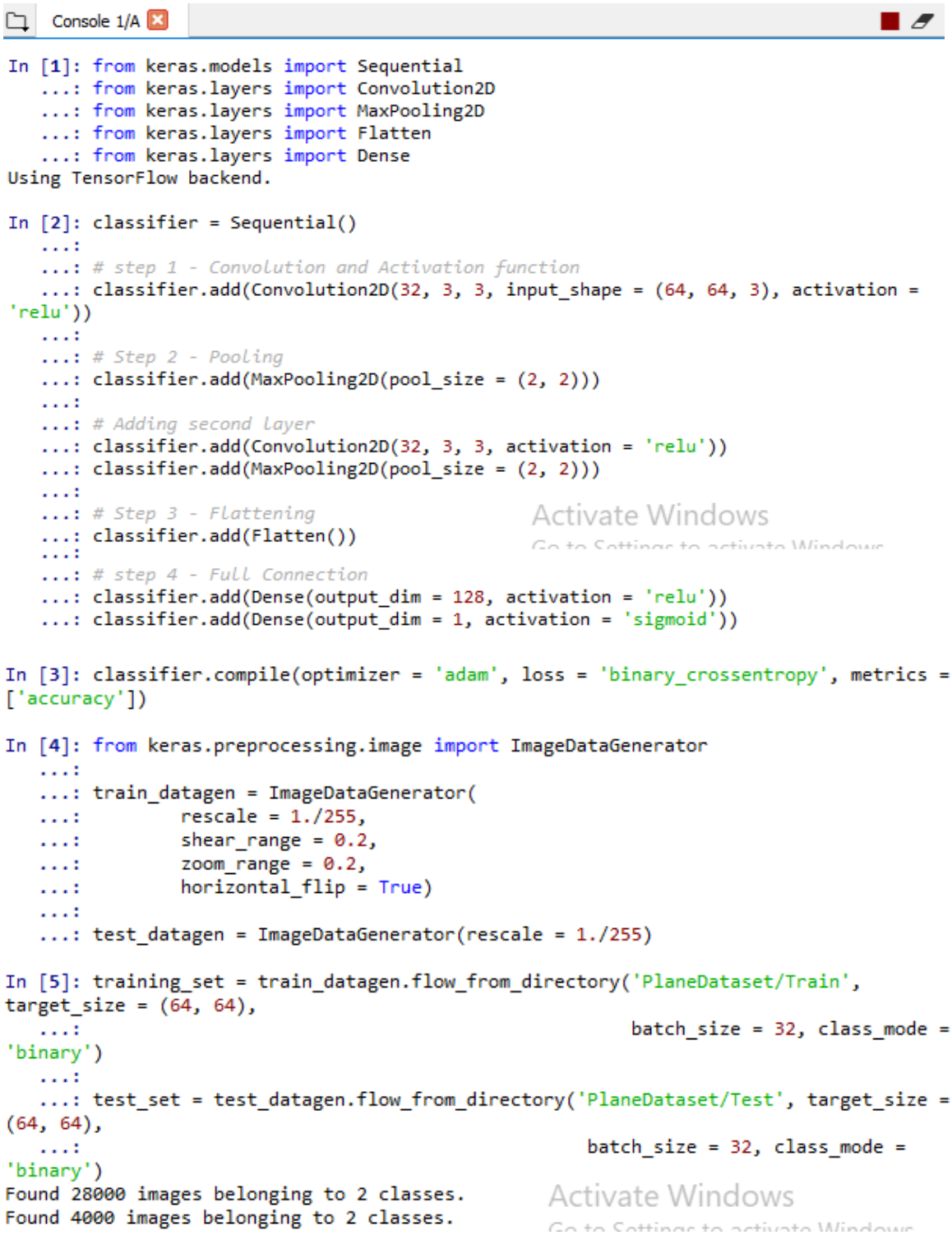

Figure 4.9: Executing Code in Console

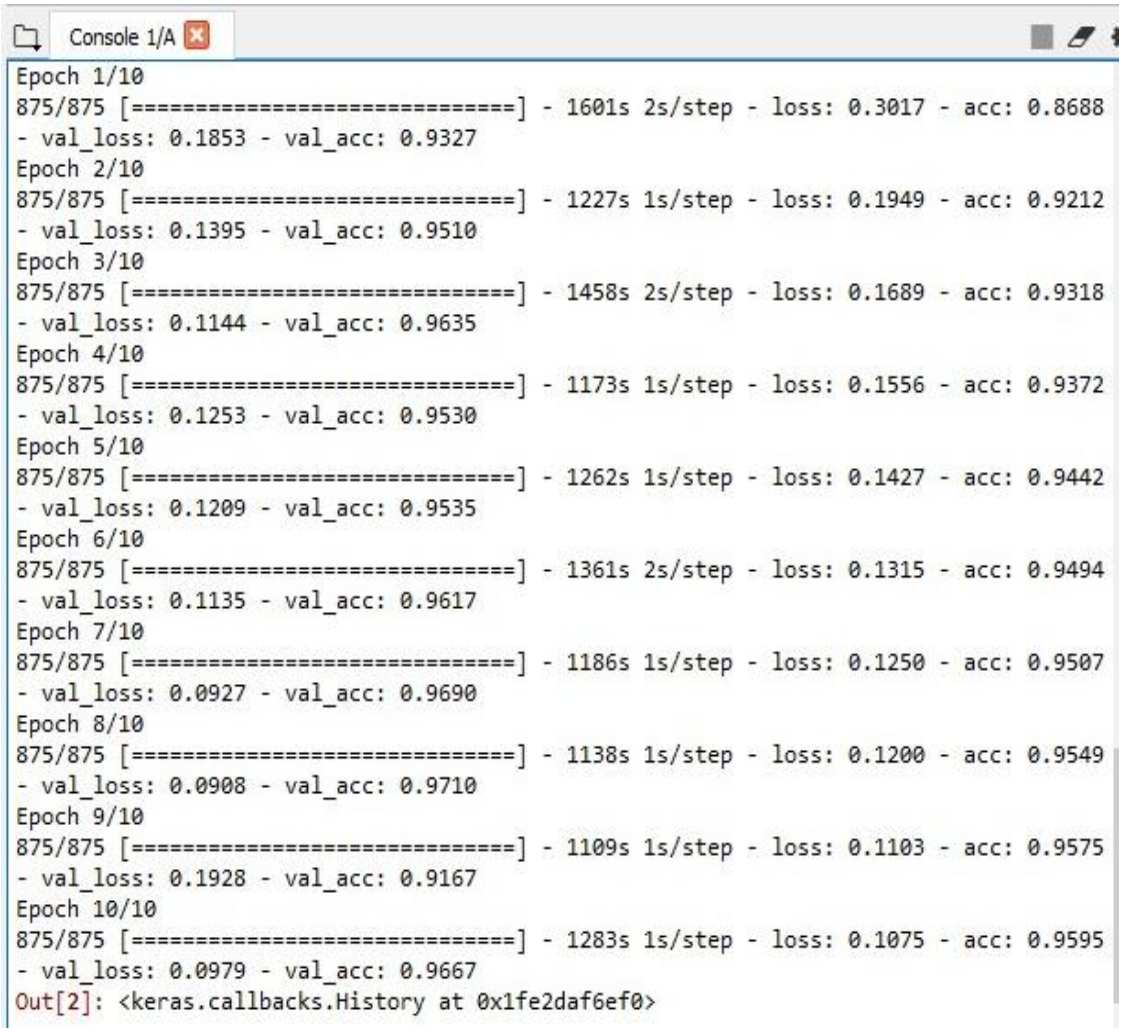

Figure 4.10: Training the model

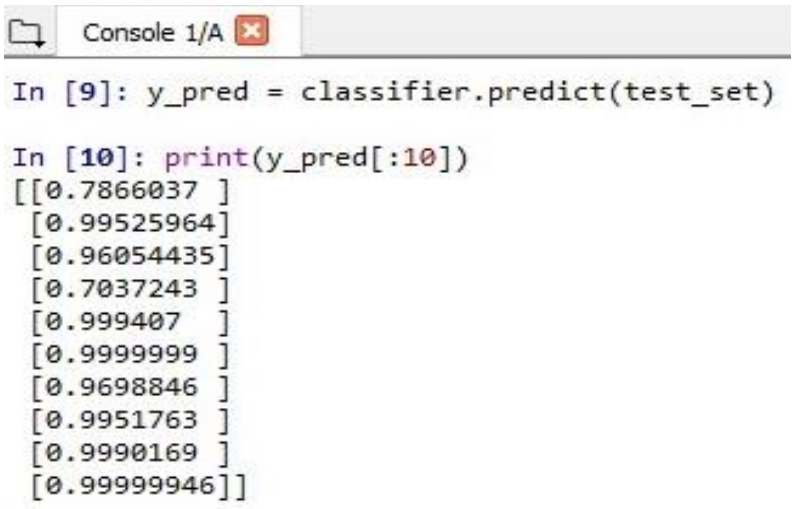

Figure 4.11: Predicting Results

CHAPTER-5: CONCLUSION

Training at DRDO was very beneficial for both of me, as I got the opportunity to enhance our knowledge and hence providing something fruitful to the organization. During this training, I learned about:

- Deep Learning
- How Deep Learning can be implemented in real world.
- Object Detection
- Image Recognition
- Convolutional Neural Networks
- Implementation of CNN and its layers.
- Anaconda software and SpyderIDE
- Learnt about Python programming
- Got to know about Keras library

Before joining this training we both didn't have any idea about the above technologies and their implementation but these two months of training provided us a deep exposure to the technologies that we came to know about while working with. Now we can step out to the professional world with this cutting edge technology. In this Artificial intelligence oriented era with the knowledge of deep learning we can make more new inventions and can help the world to cure its problems.

BIBLIOGRAPHY

- https://en.wikipedia.org/
- https://www.edureka.co
- https://skymind.ai
- https://kaggle.com
- https://towardsdatascience.com
- https://keras.io
- https://www.anaconda.com
- https://www.geeksforgeeks.org

APPENDICES

CNN- Convolutional Neural Network

Convolutional Neural Networks, like neural networks, are made up of **neurons** with **learnable weights** and **biases**. Each **neuron** receives several inputs**,** takes a weighted sum over them, pass it through an **activation function** and responds with an output**.** The whole network has a **loss function** and all the tips and tricks that we developed for neural networks still apply on Convolutional Neural Networks.

Deep Learning

Deep learning is a class of machine learning algorithms that use multiple layers to progressively extract higher level features from raw input.

Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit etc. Designed to enable fast experimentation with deep neural networks.

Object Detection

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos

THANK YOU