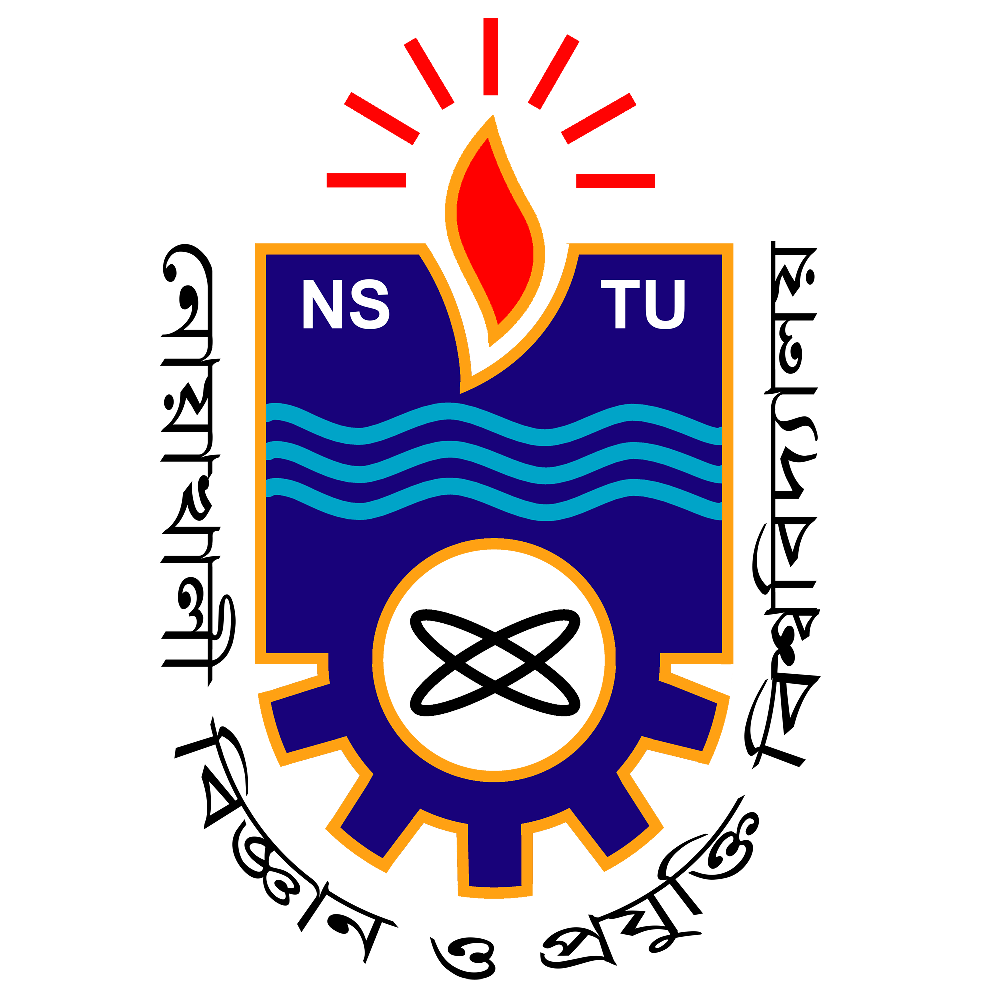
**Classification of Plant Leaf Diseases Using Different Convolutional Neural Network Models**



**Course Code: CSTE 4208**

**Course Title: Thesis and Project**

**Supervised By**

**Md. Hasnat Riaz**

**Assistant Professor**

**Department of CSTE, NSTU**

**Submitted By**

**Partha Pratim Mazumder**

**ASH1701033M**

**Session: 2016-2017**

**Department of Computer Science and Telecommunication Engineering**

**Noakhali Science and Technology University**

**Date of Submission: 22 February, 2022.**

The Thesis work entitled as **“Classification of Plant Leaf Diseases using different Convolutional Neural Network models”**, submitted by Partha Pratim Mazumder, ASH1701033M has been accepted Bachelors of Science and Computer Science and Telecommunication Engineering as B. Sc Engineering (CSTE) to be awarded by Noakhali Science and Technology University.

**Board Of Examiners**

**1st Examiner**

1…………………………………..

……………………………………

……………………………………

……………………………………

Department of Computer Science and Telecommunication Engineering, NSTU

**2nd Examiner**

2…………………………………..

……………………………………

……………………………………

……………………………………

Department of Computer Science and Telecommunication Engineering, NSTU

**3rd Examiner**

3…………………………………..

……………………………………

……………………………………

……………………………………

Department of Computer Science and Telecommunication Engineering, NSTU

**External**

4…………………………………..

Dr. Mohammad Sanaullah Chowdhury.

Professor

Department of Computer Science and Engineering

University of Chittagong, Chittagong.

**State of Originality**

It is hereby declared that the content of this thesis is original and any part of it has not been submitted elsewhere for the award of any degree or diploma of the university or other institute of higher learning.

………………………………….. ……………………………….. ………………………………….. ……………………………….

Signature of the Supervisor Signature of the Candidate

Date: Date:

ACKNOWLEDGEMENT

First, I would like to thank the Almighty for giving us the opportunity to complete this work successfully. My acknowledgement is meant to express my sincere gratitude to all those people who have been associated with this thesis and have helped us with it and by sharing their opinions and experiences through which we received the required information crucial for my thesis. I am thankful to our parents for their relentless support. Most importantly I am tremendously grateful to our honorable supervisor Md. Hasnat Riaz, Assistant Professor, Department of Computer Science and Telecommunication Engineering (CSTE), Noakhali Science and Technology University who took time out of his hectic schedule to guide us and provide us with all the necessary materials and sufficient knowledge that was the major requirement

Table of Contents

[Abstract 1](#_Toc96413319)

[Keywords 1](#_Toc96413320)

[**CHAPTER ONE** 2](#_Toc96413321)

[1.1 Introduction 2](#_Toc96413322)

[1.2 Objectives 3](#_Toc96413323)

[1.3 Literature Review 3](#_Toc96413324)

[**CHAPTER TWO** 5](#_Toc96413325)

[2.1 Convolutional Neural Network 5](#_Toc96413326)

[2.2 Different Types of CNN models 8](#_Toc96413327)

[2.2.1 VGG16 8](#_Toc96413328)

[2.2.2 VGG19 8](#_Toc96413329)

[2.2.3 ResNet50 9](#_Toc96413330)

[2.2.4 InceptionV3 10](#_Toc96413331)

[2.2.5 MobileNetV2 10](#_Toc96413332)

[2.2.6 Xception 11](#_Toc96413333)

[**CHAPTER THREE** 12](#_Toc96413334)

[3.1 Description of Data 12](#_Toc96413335)

[3.2 Data Preprocessing and establishment of model 13](#_Toc96413336)

[**CHAPTER FOUR** 16](#_Toc96413337)

[4.1 Data Preprocessing 16](#_Toc96413338)

[4.2 Model Building 17](#_Toc96413339)

[4.3 Programming Languages 17](#_Toc96413340)

[4.4 IDE 17](#_Toc96413341)

[4.5 Execution Platform 17](#_Toc96413342)

[**CHAPTER FIVE** 18](#_Toc96413343)

[5.1 Methodology 18](#_Toc96413344)

[5.1.1 Initial Xception model 19](#_Toc96413345)

[5.1.2 Applying Different Models 21](#_Toc96413346)

[5.2 Results 52](#_Toc96413347)

[**CHAPTER SIX** 53](#_Toc96413348)

[References 54](#_Toc96413349)

[**APPENDIX** 57](#_Toc96413350)

[APPENDIX A 57](#_Toc96413351)

[APPENDIX B 57](#_Toc96413352)

[APPENDIX C 58](#_Toc96413353)

[APPENDIX D 58](#_Toc96413354)

[APPENDIX E 59](#_Toc96413355)

# **Abstract**

Transfer learning using pre-trained deep neural network model has been largely used for plant disease identification. As detecting plant diseases are difficult without an expert’s knowledge fast and automated diagnostic methods are highly desired in agricultural fields. We have developed a Convolutional Neural Network (CNN) model for classifying different plant leaf images mostly of diseased plants and healthy plants with the help of different types of Convolutional Neural Network models. We used an open database from PlantVillage Dataset of **54, 306** images containing **14** different plants in a set of **38** distinct classes of (diseased plants and healthy plants) to train our model. We have used total of **6** different models *VGG16, VGG19, ResNet50, Inception V3, Xception, MobileNetV2* and we have found the best performance reaching **96.53%** of success rate using **0.25%** of data as testing among the whole dataset in identifying the corresponding (diseased plant leaf images and healthy plant leaf images) combination using *MobileNetV2* model. Also, our other models like *VGG16* with **91.46%** accuracy, *VGG19* with **90.23%** accuracy, *InceptionV3* with **77.60%** accuracy, *ResNet50* with **95.67%** accuracy, *Xception* with **96.41%** accuracy also ensure consistent amount of success rate with our very dataset. The significantly good amount of success rate makes the model a very useful advisory or early warning tool and also an approach that could be further extended to uphold an integrated plant disease identification system to operate in real cultivation conditions or a clear path toward smartphone – assisted crop disease diagnosis on a large scale of areas.

# **Keywords**

Plant disease classification, Neural Network, VGG16, VGG19, InceptionV3, Xception, ResNet50, MobileNetV2

# **CHAPTER ONE**

**PREFACE**

## **1.1 Introduction**

Plant diseases have a longer lasting effect on agricultural products. There are estimated more than $30-50 billion annually monetary loss caused by different plant diseases [1] Modern technologies have blessed human society the potential to produce ample rations to meet the request of more than 7 billion people. However after the lack of side effects of the food always remains intimidated by a number of factors such as significant amount of change in climate [2], the reduce in pollinators (from the [reports of Plenary of the Intergovernmental Science-Policy Platform on Biodiversity Ecosystem and Services of its 4th session, 2016](https://www.frontiersin.org/articles/10.3389/fpls.2016.01419/full#B23))[3], plant diseases [4], and also many others. Also there are disease causing agents called pathogens. However we can lessen crop losses and also can take different types of measures to overpower specific micro-organisms if plant diseases are efficiently diagnosed and distinguished early. Thus, plant pathologists have shared their knowledge with farmers through farming communities. That’s why machine learning comes into the picture. To improve the diagnostic results, several studies on machine learning-based automated plant diagnosis have been conducted. Convolutional neural networks (CNNs) are widely perceived as one of the most promising classification techniques among machine learning fields. The most attractive advantage of CNN is their ability to acquire requisite features for the classification from the images automatically during their learning processes. Recently, CNN have demonstrated excellent performance in large scale general image classification tasks [5], traffic sign recognition [6], leaf classification [7], and so on. Computer vision, and object recognition techniques in particular, has made immense advancements in the past few years. The PASCAL VOC Challenge [8], and more recently the Large Scale Visual Recognition Challenge (ILSVRC) [9] based on the ImageNet dataset [10] have been widely used as yardstick for a quantity of visualization-related problems in computer vision, including object classification. In 2012, a large, deep convolutional neural network achieved a top-5 error of 16.4% for the classification of images into 1000 possible categories [11]. In the next 3 years, different types of advancements in deep convolutional neural networks lessened the error rate up to 3.57% [12], [13], [14], [15],[16].

## **1.2 Objectives**

The objectives of this thesis study are:

* Develop a system that’s capable to dig up and appoint the different type of plant diseases based on blobs detection and statistical analysis.
* Analyzing different types of plant diseases as efficiently as possible.
* Predicting the possible pathogens of a certain leaf analyzing it’s affected region.
* To let people know the possibility of a certain malady only analyzing raw images from them.

## **1.3 Literature Review**

Research in agriculture section is focused towards improvement of the standards and the proportion of the product at less wasting with more take. The standard of the agricultural product may be debased due to different types of plant diseases. These diseases are caused by pathogens such as fungi, bacteria and viruses. With the help of different types of applications, many systems have been suggested to solve or at least to lower the problems faced by the farmers, by harnessing the use of image processing and different types of automatic classification tools.

Suhaili Kutty et al.[17] discussed the process to classify *Anthracnose* and *Downey Mildew*, watermelon leaf diseases using neural network analysis. They used a digital camera with specific calibration procedure under controlled environment. Their classification is based on color feature extraction from RGB color model where the RGB pixel color indices have been extracted from identified ROI(region of interest).To reduce noise from images and for segmentation median filter is used. And for classification of the image, neural network pattern recognition toolbox is utilized. Proposed method achieved 75.9% of accuracy based on its RGB mean color component.

Sanjeev Sannaki et al [18] identify the disease with the help of image processing and AI techniques on images of grape plant leaf. In their proposed system , complex background with grape leaf image is taken as input. Noise is removed using anisotropic diffusion also the segmentation is done by k-means clustering. After segmentation, feature extraction is happened by computing Gray Level Co-occurrence Matrix. And finally classification takes place using Feed Forward Back Propagation Network classifier. Also Hue feature is used for more accurate result.

Akhtar et al [19] have implemented the support vector machine(SVM) approach procedures for the classification and detection of rose leaf diseases as black spot and anthracnose. Authors have implied the threshold method for segmentation and Ostu’s algorithm was mainly used to establish the threshold values. In this approach, different features of DWT, DCT and texture based eleven haralick features are extricated which are afterwards merged with SVM approach and predicts quite efficient accuracy value.

The study of Usama Mokhtar et al [20] incorporated with method that involves Gabor Wavelet Transform technique to extract fitting features relevant to image of tomato leaf in coincidence with using Support Vector Machines (SVMs). They described technique of Tomato leaves diseases detection and diseases are: Powdery mildew and Early blight. Here Gabor Wavelet Transform is applied in feature extraction for feature vectors also in classification. Cauchy Kernel, Laplacian Kernel and Invmult Kernel methods are involved in SVM for output decision where tomato leaf infected with Powdery mildew or early blight. The proposed approach ensures excellent footnote with accuracy 99.5%.

Supriya et al [21] worked with the cotton leaves. They first captured the affected leaf and then pre-process converting into other color space. They also used Otsu’s global thresholding method during segmentation. Again, color-co-occurrence method is used for extracting different features such as color and texture. Multi SVM(Multi Support Vector Machine) classifier is used for detecting the diseases.

Ms. Kiran R. Gavhale et al [22] presented number of image processing techniques to extract diseased part of leaf. For Pre-processing, Image enhancement is completed using DCT domain and thus color space conversion is done. After that segmentation is done with the help of k-means clustering. Feature extraction is done using GLCM Matrix. For classifying canker and anthracnose disease of citrus leaf, the use of SVM with radial basis kernel and polynomial kernel is done.

N.J. Janwe and Vinita Tajane [23] suggested for their medical plants disease identification using Canny Edge detection algorithm, Histogram Analysis and CBIR. The identification of medical plants according to its edge features. The leaf image converts to gray scale and calculate the edge histogram. The algorithm that purposed is canny edge detection.

# **CHAPTER TWO**

**RELATED DISCUSSIONS**

We used Convolutional Neural Network (CNN) algorithm to generate our predictions and build the models. Our main challenge was to cope with the different sized images and make the best use of different learning rates. But using Convolutional Neural Network we can make the best use of our dataset and also CNN has shown favorable results in previous works related to plant disease detection with good accuracy and thus proves to be a good candidate for detecting various plant leaf diseases.

## **2.1 Convolutional Neural Network**

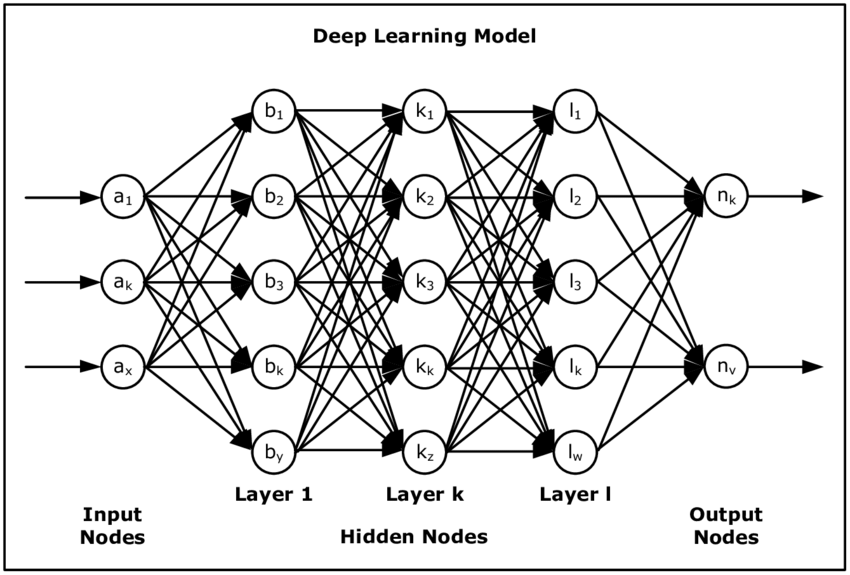
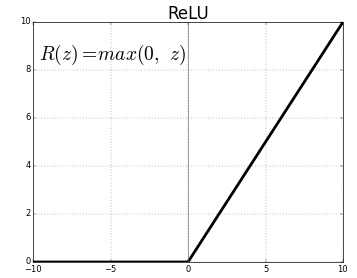
Convolutional Neural Network (Conv-Net) is a regularized version of multilayer perceptron (MLP) and uses a feedforward artificial neural network generating a set of outputs from a set of inputs. CNN is characterized by several layers of input nodes connected as a directed graph between the input and output layers. The pre-processing required in a Conv-Net is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, Conv-Nets have the ability to learn these filters / characteristics.

Fig 2.1: Deep learning model with k hidden layers [27]

Convolutional Neural Network is comprised of one or more convolutional layers and then followed by one or more fully connected layers in a standard multilayer neural network. The hidden layers are Convolution Layer, Pooling Layer, Flattening Layer, Fully Connected Layers.

Fig 2.2: Detailed Convolutional Neural Network using different hidden layers [28]

* ***Convolution + ReLU:*** Convolution layer serves as neural networks filter. This layer also provides feature mapping, learning the parameters of such maps and how the patterns are detected. Here the activation function is ReLU(Rectified Linear Unit) which provides the linearity of the model.
* ***Pooling:***We will do specific type of pooling max pooling where the kernel extracts the maximum value of the area it convolves.
* ***Flattening:***In this layer we use the data pooled from previous layer and convert into a 1-dimensional array for inputting it to the next layer.
* ***Full Connection****:* Here, previous layer inputs are connected together.
* ***Soft-max:*** Used as the activation function for multi-class classification problem where class membership is required on more than two class levels.

Here, we used ReLU (Rectified Linear Unit) activation function so that it can prevent the exponential growth in the computation required to operate the neural network. The main equation for ReLU activation function is:

…………… (2. 1) [35]

Fig 2.3: ReLU Activation Function.

Here, *f(z)* returns *0* if it receives any negative input, but for any positive value *z*, it returns the value back. Thus, it gives an output that has a range from *[0, ∞)*. It is most commonly used in different deep learning algorithms to accurately predict results. In our final output layer we have used Soft-max function as an activation function. The equation for Soft-max activation function is:

………………… (2. 2) [36]

It gives the output of each unit between *0* and *1,* just like Sigmoid function but it divides each unit such that the total sum of the unit is between *0* and *1*. Here, *a* represents the values from the neurons of the output layer. The exponential acts as the non-linear function. Later these values are divided by the sum of exponential values in order to normalize and then convert them into probabilities.

## 2.2 **Different Types of CNN models**

### 2.2.1 **VGG16**

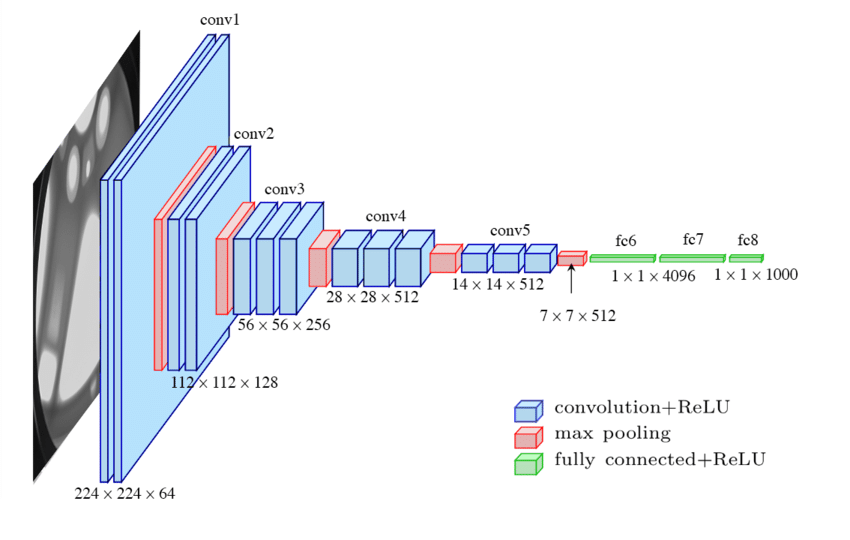


Fig 2.4: Architecture and Implementation of VGG16 and VGG19 [29]

* Developed by Karen Simonyan and Andrew Zisserman from University of Oxford, 2014 [30].
* Widely used CNN architecture for ImageNet dataset, a large visual database used in visual object recognition software research.
* “16” implies total number of layers is 16.
* Achieved accuracy is 92.7% in 14 million images from 1000 classes.
* The input size is fixed 224 x 224 RGB
* The end of the network has fully connected layer with 4096 neurons followed by a SoftMax layer.

### 2.2.2 **VGG19**

* Developed by Karen Simonyan and Andrew Zisserman in their “Very Deep Convolutional Networks for Large – Scale Image Recognition” article [30].
* Composed of 19 CNN layers (16 Convolution layers, 3 Fully Connected layer, 5 MaxPool layer and 1 SoftMax layer)
* Classify images up-to 1000 object categories.
* The input image size is similar to VGG16 224 x 224 RGB.
* Uses 2x2 max-pooling layers followed by some conv layers for down-sampling which eventually reduces the input size of later layer.

### 2.2.3 **ResNet50**

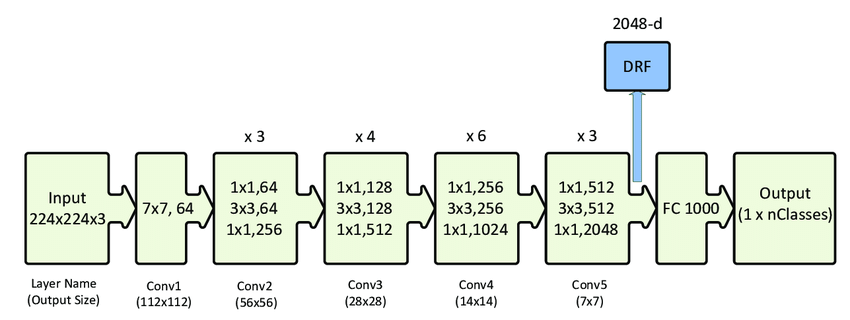


Fig 2.4: Architecture and Implementation of ResNet50 [31]

* Stands for Residual Network.
* First introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun in their 2015 computer vision paper “Deep Residual Learning for Image Recognition” [31].
* ResNet50 can work with 50 different layers.
* Accepts a 224x224 image as input.
* Consists of a stack of residual blocks which are feed forward neural networks with shortcuts.
* Here, shortcuts are connections skipping over some layers which are used to deal with the problem of vanishing – gradients as the network goes deeper.

### 2.2.4 **InceptionV3**

Fig 2.5: Architecture and Implementation of InceptionV3 [32]

* Released in 2015 developed by Google as module for Google-Net [32].
* Inception Module is a stack of a max-pooling layer and convolution layers.
* Has a total of 42 layers and a lower error rate than it’s pre-processors.
* Input of Inception V3 is a fixed-size 299x299 image.
* Widely used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset.
* Mainly used for assisting in image analysis and object detection so mainly in Computer Vision.

### 2.2.5 **MobileNetV2**

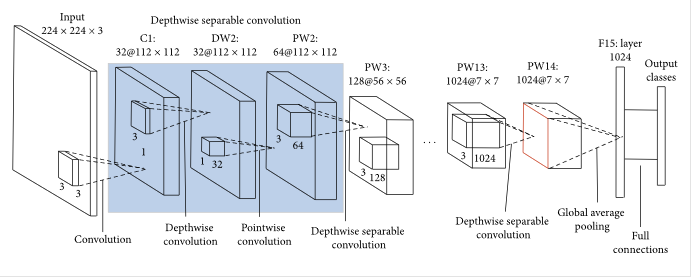


Fig 2.6: Architecture and Implementation of MobileNetV2[33]

* Released in the spring of 2017 by Menglong Zhu, Andrey Zhmoginov and Liang-Chigh Chen [33].
* A network model using depth-wise separable convolution as its basic unit.
* Supports any input size greater than 32 x 32
* Has fewer parameters and higher classification accuracy. That’s the reason it works so well in different dataset.
* Designed for mobile and embedded vision applications

### 2.2.6 **Xception**

Fig 2.6: Architecture and Implementation of Xception [34]

* Xception – the Extreme Inception, first published in year 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) by Francois Chollet from Google, Inc [34].
* Outperform Inception V3 on a large image classification dataset comprising 350 million images and 17,000 classes.
* Works better than Inception V3 not due to increased capacity as they both use same number of parameters but rather to a more efficient use of model parameters.
* Total number of layers are 71 and can load more than a million images as a pretrained version of the network trained.
* Standard input image size is 299x299 RGB images.

# **CHAPTER THREE**

**DESCRIPTION OF DATA AND PREPROCESSING**

## 3.1 **Description of Data**

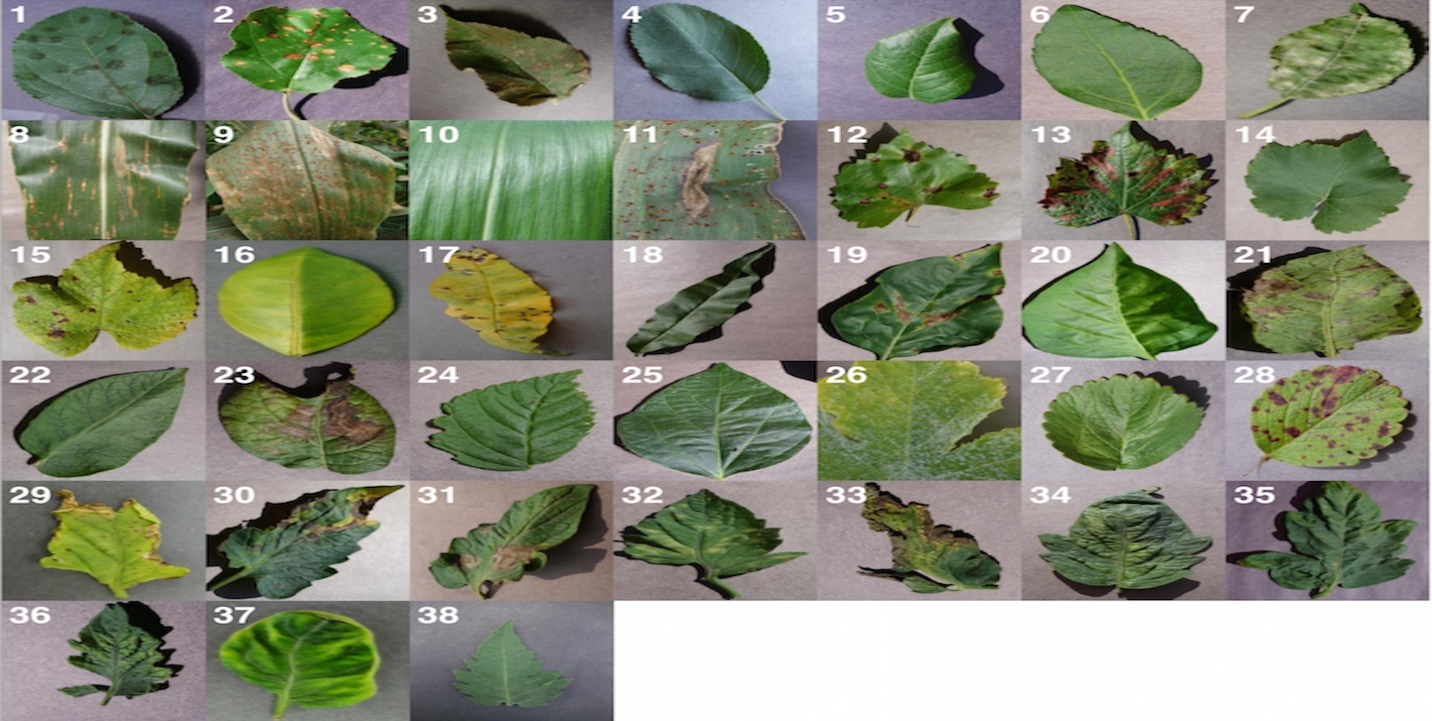
We use Plant-Village Dataset for completing this classifier. We inspect total 54,306 images of plant leaves, which have a spread of 38 class labels allotted to them. Each class label is a crop-disease pair, and we ensure an attempt to estimate the crop-disease pair given just the picture of the plant leaf. In all the methods used in this paper, we reduce the sizes of the images to 256x256 pixels, and we carry out both the model optimization and predictions on these downscaled images. Across all our experiments, we work with the colored version of the whole Plant-Village dataset

Fig 3.1: Example of leaf images from the Plant Village dataset, representing every crop-disease pair used. [24]

**(1) Apple Scab**, (*Venturiain aequalis)* **(2) Apple Black Rot**, (*Botryosphaeria obtuse)* **(3) Apple Cedar Rust**, (*Gymnosporangium juniperi-virginianae)* **(4) Apple healthy** **(5) Blueberry healthy** **(6) Cherry healthy** **(7) Cherry Powdery Mildew**, (*Podoshaera clandestine)* **(8) Corn Gray Leaf Spot**, (*Cercospora zeae-maydis)* **(9) Corn Common Rust**, (*Puccinia sorghi)* **(10) Corn healthy** **(11) Corn Northern Leaf Blight**, (*Exserohilum turcicum)* **(12) Grape Black Rot**, (*Guignardia bidwellii)*, **(13) Grape Black Measles(Esca)**, (*Phaeomoniella aleophilum),* (*Phaeomoniella chlamydospore)* **(14) Grape Healthy** **(15) Grape Leaf Blight**, (*Pseudocercospora vitis)* **(16) Orange Huanglongbing**(Citrus Greening), (*Candidatus Liberibacter spp.)* **(17) Peach Bacterial Spot**, (*Xanthomonas campestris)* **(18) Peach healthy** **(19) Bell Pepper Bacterial Spot**,(*Xanthomonas campestris)* **(20) Bell Pepper healthy** **(21) Potato Early Blight**, (*Alternaria solani)* **(22) Potato healthy** **(23) Potato Late Blight**, (*Phytophthora infestans)* **(24) Raspberry healthy** **(25) Soy bean healthy** **(26) Squash Powdery Mildew**, (*Erysiphe cichoracearum)* **(27) Strawberry Healthy** **(28) Strawberry Leaf Scorch**,(*Diplocarpon earlianum)* **(29)Tomato Bacterial Spot**, (*Xanthomonas campestris pv.vesicatoria)* **(30) Tomato Early Blight**,(*Alternaria solani)* **(31) Tomato Late Blight**, (*Phytophthora infestans)***(32)Tomato Leaf Mold**, (*Passalora fulva)* **(33)Tomato Septoria Leaf Spot**,(*Septoria lycopersici)* **(34) Tomato Two Spotted Spider Mite**, (*Tetranychus urticae)* **(35) Tomato Target Spot**,(*Corynespora cassiicola)* **(36) Tomato Mosaic** **(37) Tomato Yellow Leaf Curl (38) Tomato healthy**

Here, each class contains more than a thousand images with labeling in it. This dataset can be directly utilized as building multi-class disease identification models that a leaf image is labeled as one of thirty eight classes (healthy or diseased).

## 3.2 **Data Preprocessing and establishment of model**

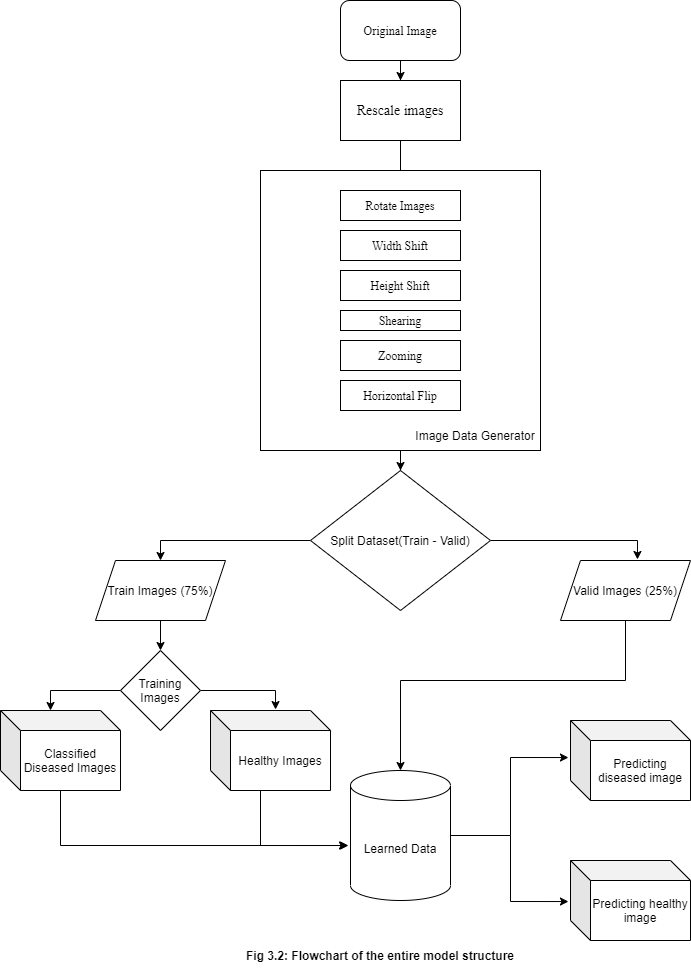
In this paper we designed our system to accept images from various types of digital cameras to cope with images captured in different sizes and aspect ratios. Here, we use Google Colab as platform to generate the model. But as Google Colab has it’s GPU limitaion and fixed amount of RAM so dataset preprocessing is necessary. Using dataset preprocessing we mainly perform data augmentation to generate “new” trainning samples from the original ones by applying random jitters and perturbations but also maintaining the original class labels. We can obtain the augmented data from the pure images by performing simple geometric transformation such as random – i.Translations, ii. Rotations, iii. Changes in scale, iv. Shering, v. Horizontal / Vertical flip.



Fig 3.3: Pre-processing data from dataset

To have a proper sense of how our working will perform on newly unseen data, and also to remain a track of if any of our approaches are overfitting with the new data, we go through all our experiments across a whole range of train-test set splits, namely 80–20 (80% of the whole dataset used for training, and 20% for testing), 60–40 (60% of the whole dataset used for training, and 40% for testing), 75–25 (75% of the whole dataset used for training, and 25% for testing), 40–60 (40% of the whole dataset used for training, and 60% for testing), 20–80 (20% of the whole dataset used for training, and 80% for testing). Among all the splits 75-25 performs better than most other splits.

We evaluate the applicability of deep convolutional neural networks for the classification problem described above. We focus on six popular architectures, namely *VGG16[30], VGG19[30], MobileNetV2[31], ResNet50[32], InceptionV3 [33]*, and *Xception [34].*

To summarize-

1. Choice of deep learning architecture:

1. *InceptionV3*
2. *Xception*
3. *MobileNetV2*
4. *VGG16*
5. *VGG19*
6. *ResNet50*

2.Choice of training mechanism:

1. *Transfer Learning.*
2. *Training from scratch*

3.Choice of dataset:

1. *Color*

4.Choice of training-testing set distribution:

1. *Train:80%, Test:20%*
2. *Train:75%, Test:25%*
3. *Train:60%, Test:40%*
4. *Train:40%, Test:60%*
5. *Train:20%, Test:80%*

To enable a fair comparison between the results of all the experimental configurations, we also tried to standardize the hyper-parameters across all the experiments, and we used the following hyper-parameters in all of the experiments:

* Base learning rate: 0.001
* Batch size: 32
* Default Image Size: (256X256) RGB image
* Epoch: 55
* Depth: 3
* Optimizer: Adam

# **CHAPTER FOUR**

**PROGRAMMING LANGUAGE AND ENVIRONMENT**

## 4.1 **Data Preprocessing**

* **NumPy** (python library): NumPy stands for Numerical Python. It’s a Python package and a library consisting of multidimensional array objects and a collection of routines to process those arrays. Logical and mathematical operations on the arrays can be performed with the help of NumPy library.
* **Pandas** (python library): It’s a python data analysis library. It provides fast, flexible and expressive data structures and also designed to make working with “relational” or “labeled” data which are both easy and intuitive. Pandas aims to be the fundamental high – level building block for having practical real world data analysis in Python.
* **Pickle** (python library): Pickle is mainly used in sterilizing and desterilizing a Python object structure. It’s basically the process of converting a Python object into a byte stream to store it in a file or database, also maintain program state across sessions or transport data over the network.
* **CV2** (python library): OpenCV-Python is mainly used in computer vision problems for binding designs. cv2.imread() method loads an image from a specific folder if the image present otherwise returns an empty matrix. It performs tasks like face detection, object detection tracking, landmark detection and so on.
* **Listdir** (python library): listdir() method is used to have the list of all the files and directories in the specific folder. It mainly returns a list containing names of the entries in the directory given by path and if no directory is specified then returns current directory.
* **MatPlotlib** (python plotting library): It’s a comprehensive library for making static, animated and interactive visualizations of numerical data in python. Matplotlib mainly provides an object oriented API for embedding plots into applications using general-purpose GUI like TKinter, wxPython etc.

## 4.2 **Model Building**

* **TensorFlow** framework (python): TensorFlow is Google’s open source AI framework for ML and high performance numerical calculations. It’s a python library that cites C++ to build and execute dataflow graphs. TensorFlow supports many classification and regression algorithms and more generally deep learning and neural networks.
* **Keras** framework (python): Keras is a neural network library. Where TensorFlow provides both high – level and low – level APIs while Keras provides only high – level APIs. Keras is capable of running on top of TensorFlow, CNTK or Theano.

## 4.3 **Programming Languages**

* **Python:** Python is a general-purpose programming language that can be used on any modern computer operating system. It can be used for processing text, numbers, images, scientific data and just about anything else you might save on a computer.

## 4.4 **IDE**

* **Jupyter notebook:** The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more

## 4.5 **Execution Platform**

* **Google Colab:** Colab is a Google internal research tool for data science. They have released the tool sometime earlier to the general public with a noble goal of dissemination of machine learning education and research.

# **CHAPTER FIVE**

**METHODOLOGY AND RESULTS**

## 5.1 **Methodology**

First we import all the necessary libraries and create directory path mounting our dataset from google drive with google colab. Then we preprocess the data and split the dataset between Train – test. We preprocess the dataset using ImageDataGenerator() method.

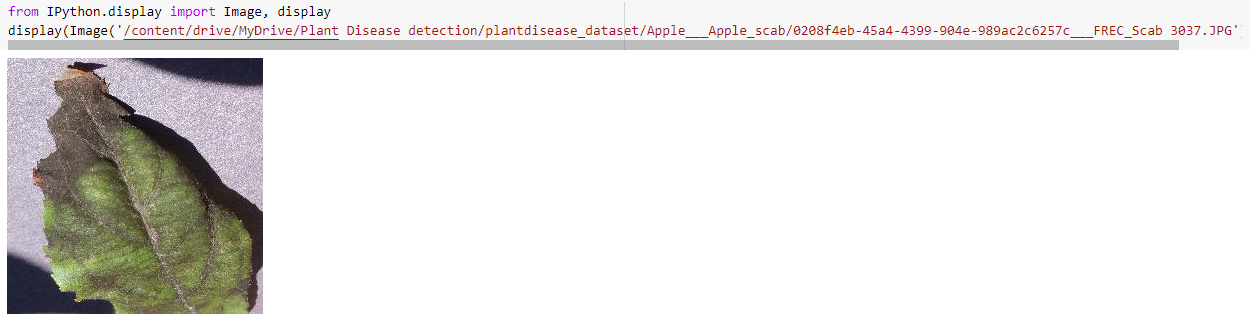


Fig 5.1: Images Before Preprocessing.

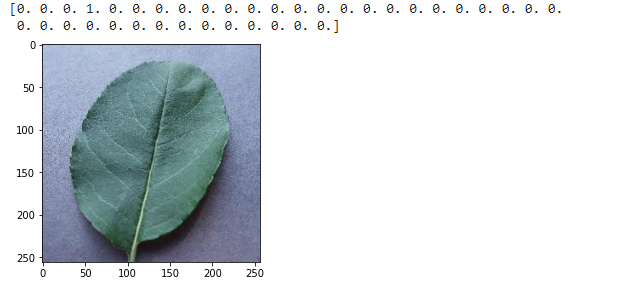


Fig 5.2: Images After Preprocessing.

### 5.1.1 **Initial Xception model**

Calling Xception model function:

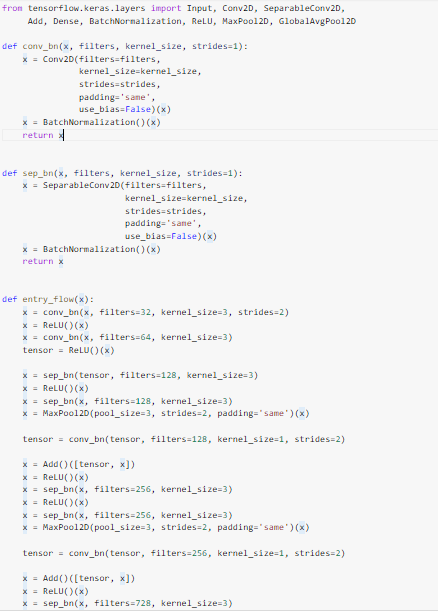


Fig 5.3: Xception Pre-trained Model.

Running epochs:

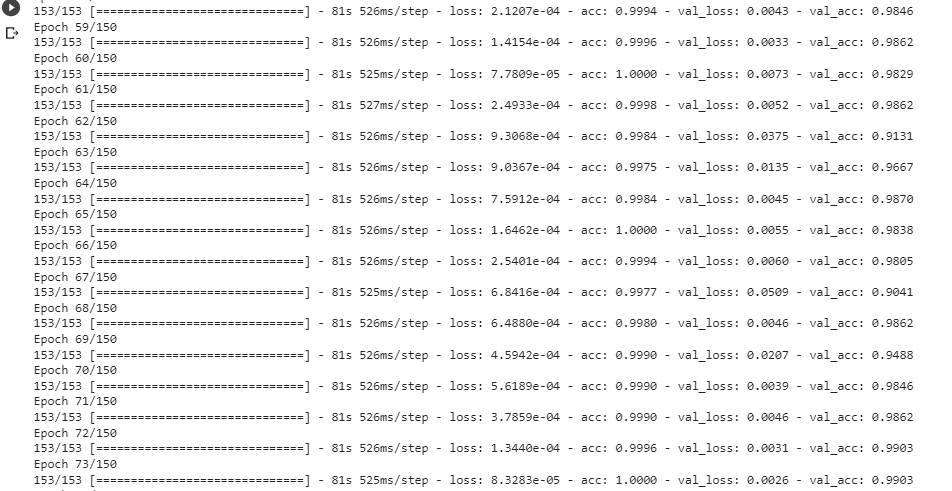


Fig 5.4: Running Epochs (150) in Xception Model.

Training - Validation Accuracy and Training - Validation Loss :

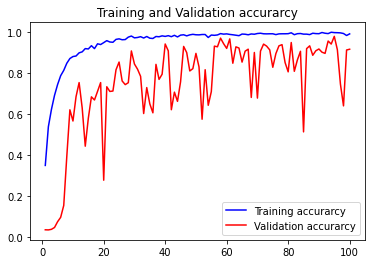
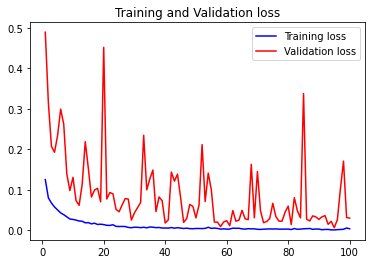
 

Fig 5.5: Accuracy and Loss graph using Xception model.

Predicting new images

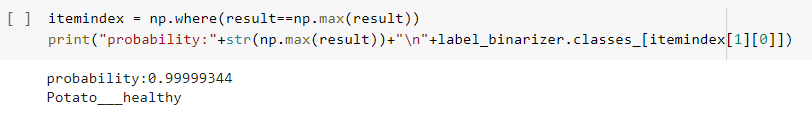


Fig 5.6: Successfully predicting image classes using Xception model.

### 5.1.2 **Applying Different Models**

#### 5.1.1.1 **VGG16**

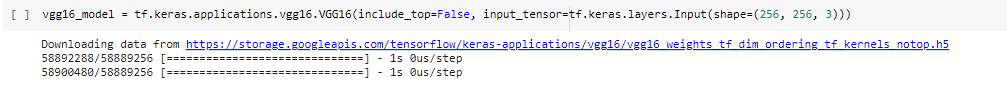
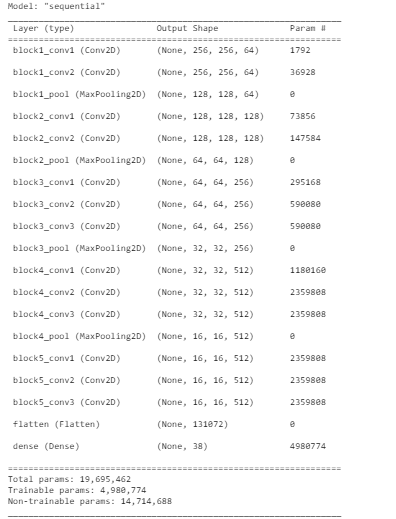
After preprocessing the dataset we then call built-in VGG16 model from google colab.

Fig 5.7.1: Calling VGG16 pretrained model from Google Colab..

As the VGG16 is a functional model so we must convert it into Sequential model and we then use Flattening layer to convert 2d array into 1d array and finally we mapped output of the model with our 38 classes.



Then we run model.fit() method to fit the dataset images with the above mentioned architecture. Here, we use total epoch 55 and batch size for each step is 10.

Fig 5.7.2: VGG16 model summarized.

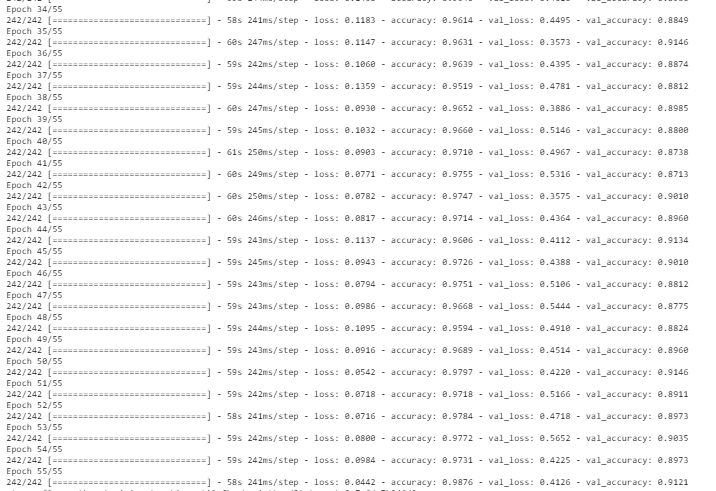


Fig 5.7.3: Running epochs in VGG16 model.

Using VGG16 we have our train accuracy is 98.76% and validation accuracy is 91.46%.

Graph for Train and Validation accuracy and Train and Validation loss is given below –

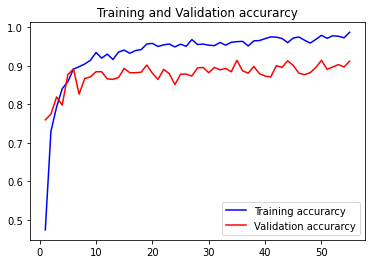
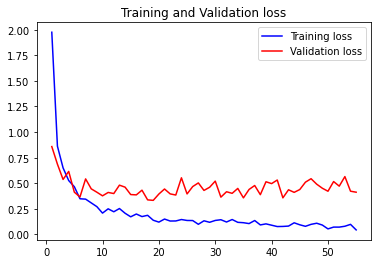


Fig 5.7.4: Accuracy and Loss graph in VGG16 model.

Confusion matrix:

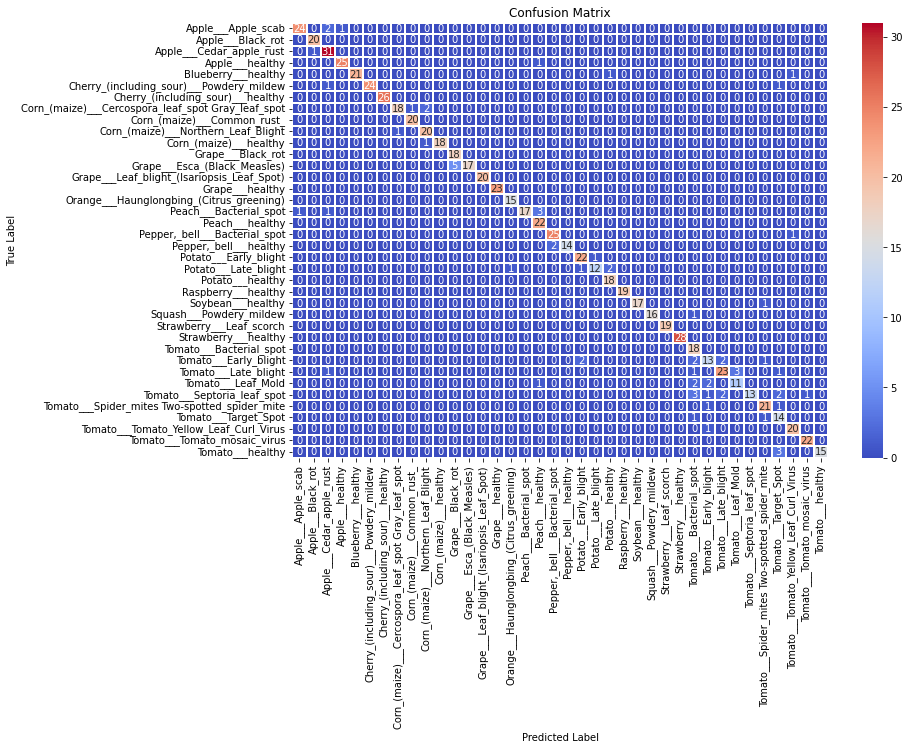


Fig 5.7.5: Confusion Matrix for 38 classes in VGG16 model.

Precision, Recall and F1 score:

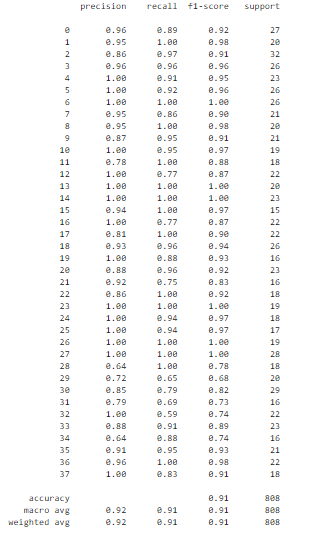


Fig 5.7.6: Precision, Recall and F1 score for 38 classes in VGG16 model.

ROC AUC score: is 0.9551831544276838

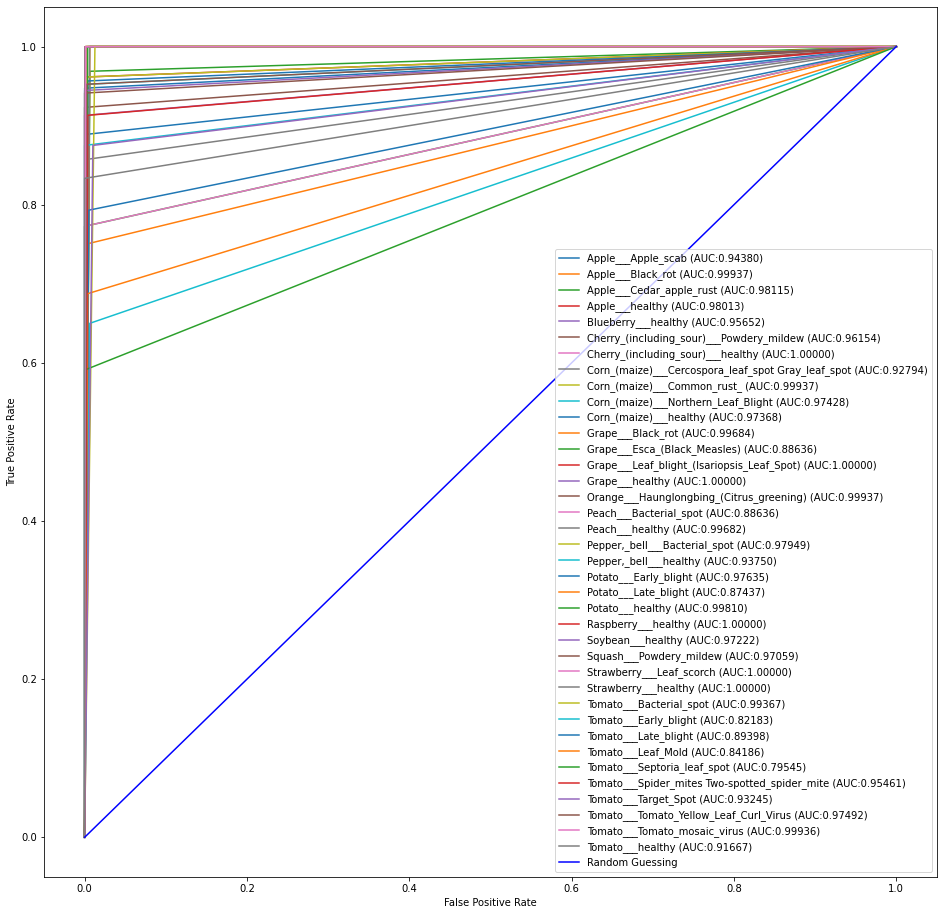


Fig 5.7.7: ROC AUC score for 38 classes in VGG16 model.

#### 5.1.1.2 **VGG19**

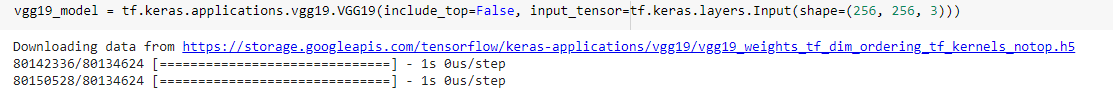
After preprocessing the dataset we then call built-in VGG19 model from google colab.

Fig 5.8.1: Calling VGG19 pretrained model from Google Colab.

As the VGG19 is a functional model so we must convert it into Sequential model and we then use Flattening layer to convert 2d array into 1d array and finally we mapped output of the model with our 38 classes.

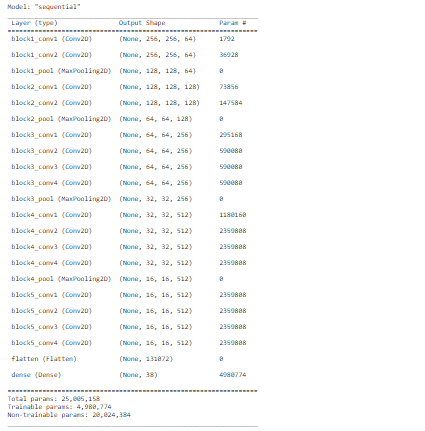


Fig 5.8.2: VGG19 model summarized.

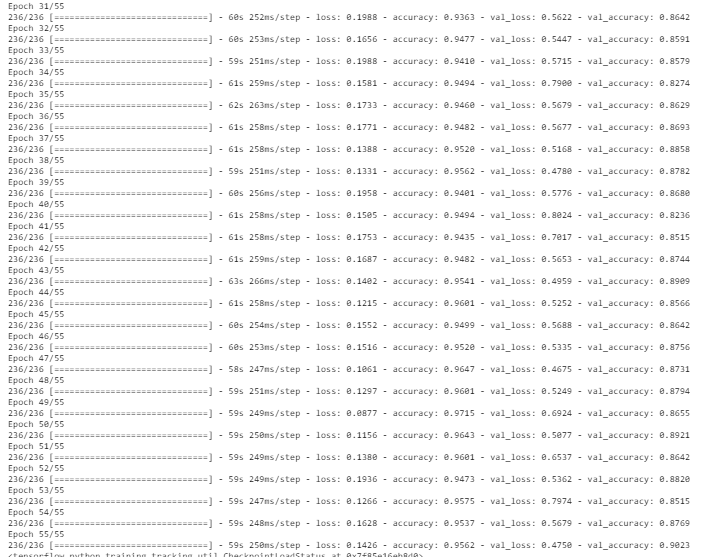
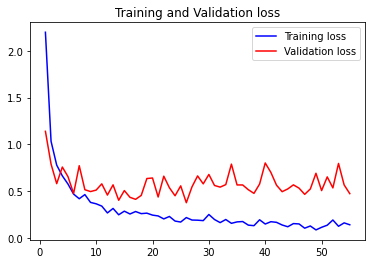
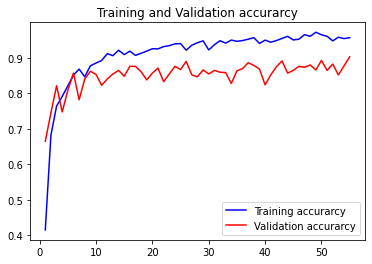
Then we run model.fit() method to fit the dataset images with the above mentioned architecture. Here, we use total epoch 55 and batch size for each step is 10.

Fig 5.8.3: Running epochs in VGG19 model.

Using VGG19 we have our train accuracy is 96.47% and validation accuracy is 90.23%.

Graph for Train and Validation accuracy and Train and Validation loss is given below –



Confusion matrix:

Fig 5.8.4: Accuracy and Loss graph in VGG19 model.

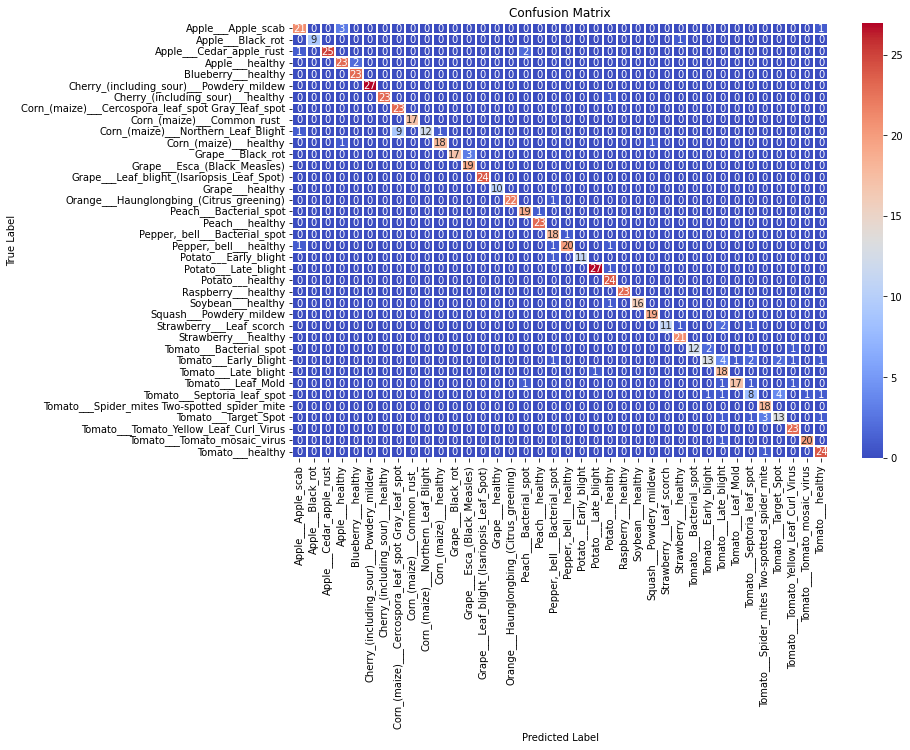


Fig 5.8.5: Confusion Matrix for 38 classes in VGG19 model.

Precision, Recall and F1 score:

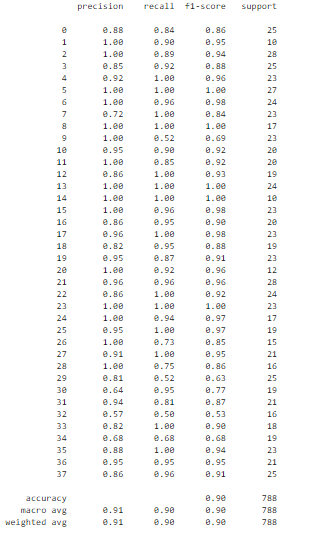


Fig 5.8.6: Precision, Recall and F1 score for 38 classes in VGG19 model.

ROC AUC score: is 0.9484857392674255

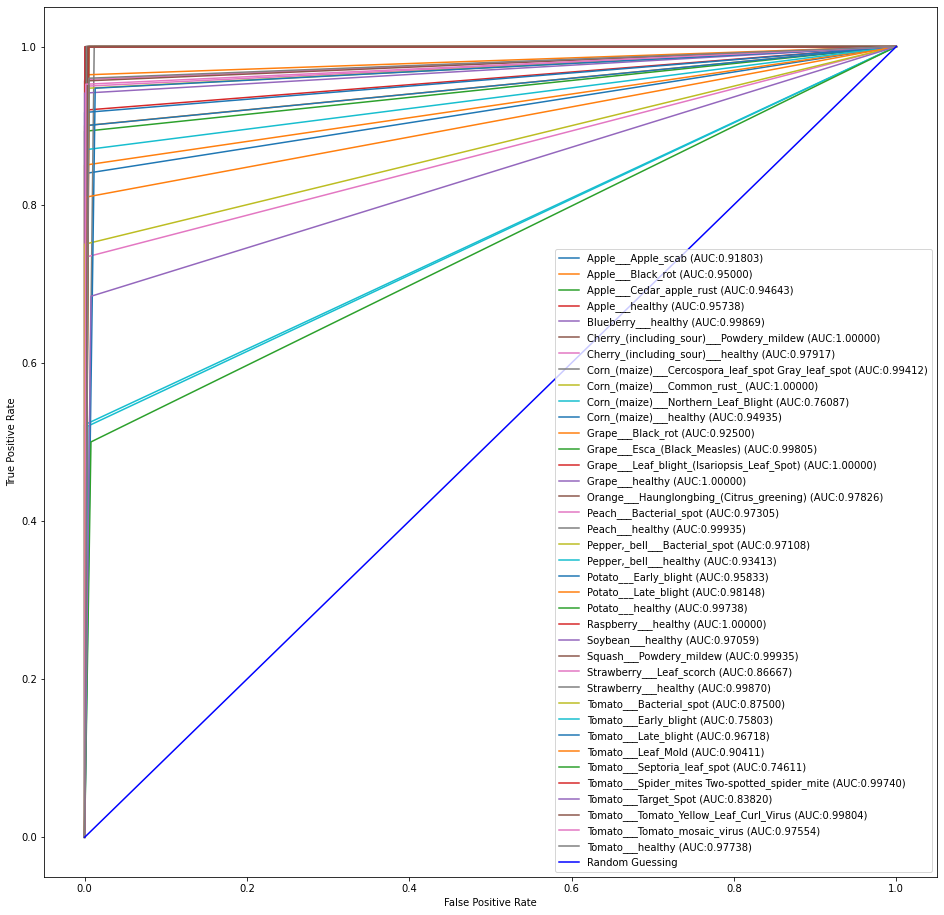


Fig 5.8.7: ROC AUC score for 38 classes in VGG19 model.

#### 5.1.1.3 **ResNet50**

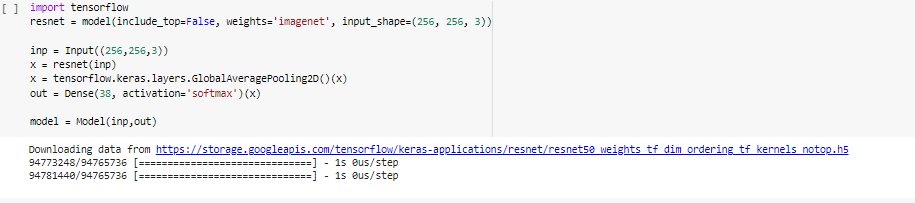
After preprocessing the dataset we then call builtin ResNet50 model from google colab.

Fig 5.9.1: Calling ResNet50 pretrained model from Google Colab..

As the ResNet50 is a functional model so we must convert it into Sequential model and we then use Flattening layer to convert 2d array into 1d array and finally we mapped output of the model with our 38 classes.

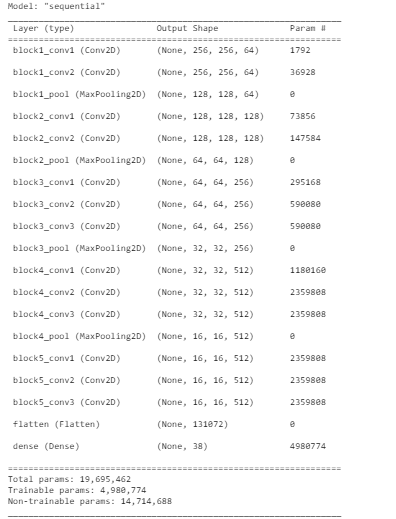
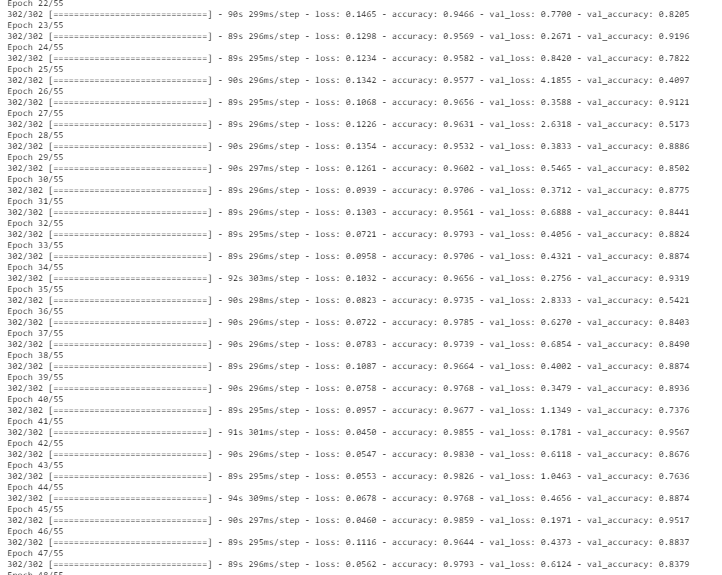


Fig 5.9.2: ResNet50 model summarized.

Then we run model.fit() method to fit the dataset images with the above mentioned architecture. Here, we use total epoch 55 and batch size for each step is 10.

Fig 5.9.3: Running epochs in ResNet50 model.



Using ResNet50 we have our train accuracy is 98.55% and validation accuracy is 95.67%.

Graph for Train and Validation accuracy and Train and Validation loss is given below –

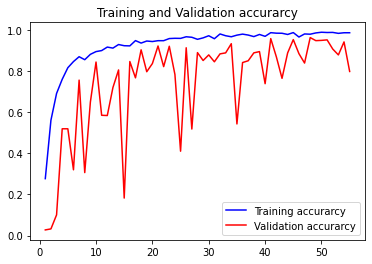
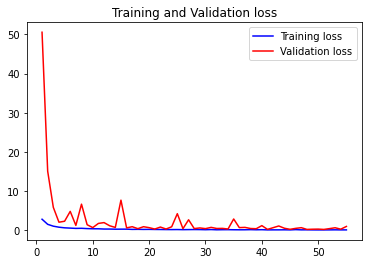


Fig 5.9.4: Accuracy and Loss graph in ResNet50 model.

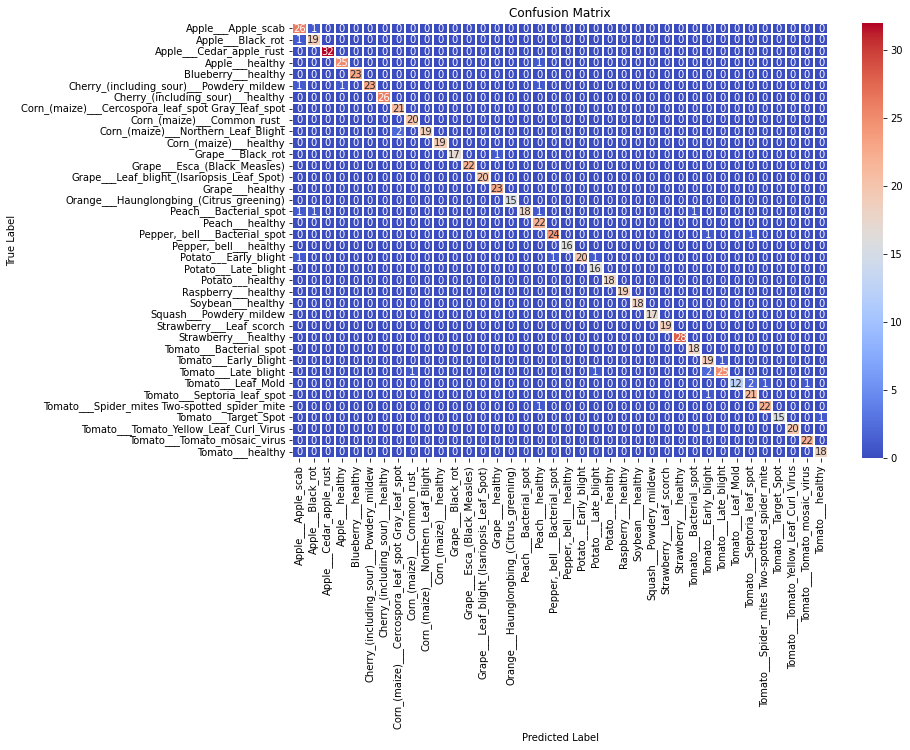
Confusion matrix: 

Fig 5.9.5: Confusion Matrix for 38 classes in ResNet50 model.

Precision, Recall and F1 score:

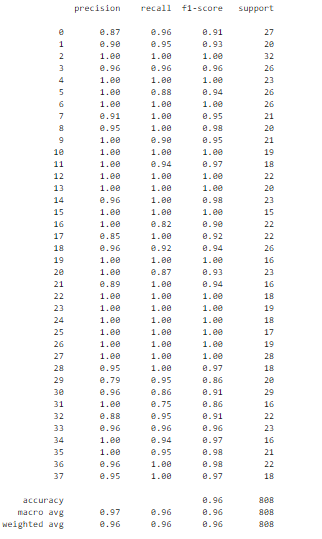


Fig 5.9.6: Precision, Recall and F1 score for 38 classes in ResNet50 model.

ROC AUC score: is 0.9808253287059144

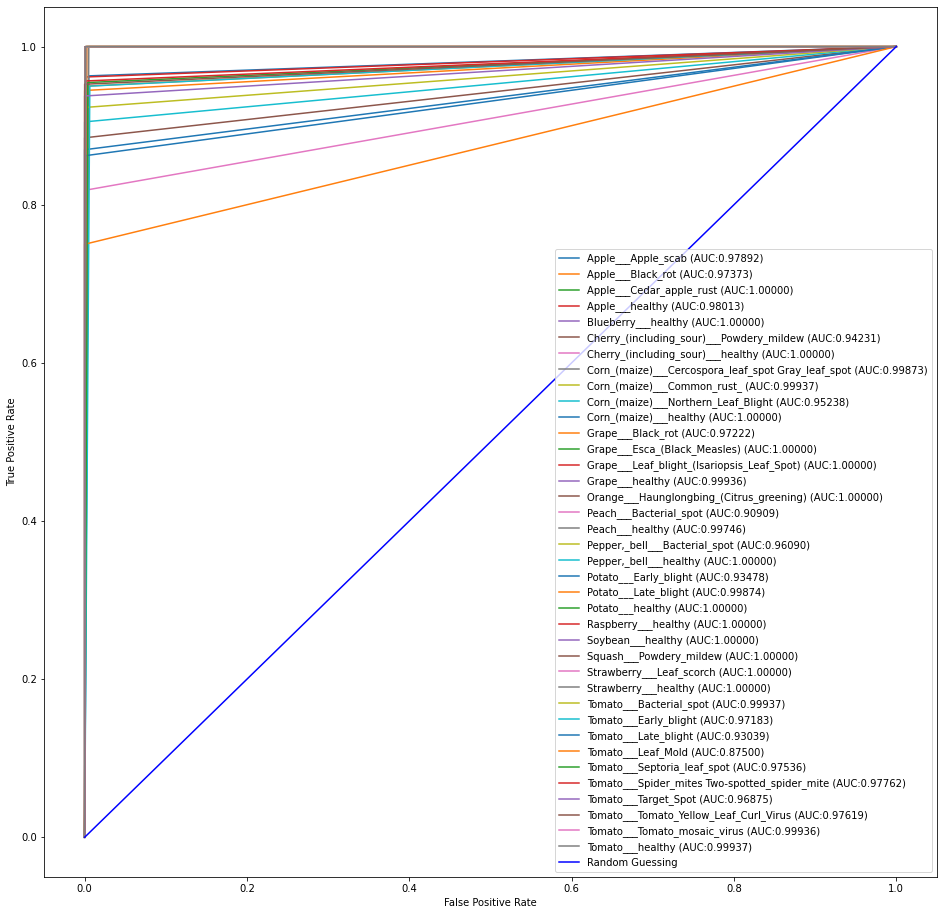


Fig 5.9.7: ROC AUC score for 38 classes in ResNet50 model.

#### 5.1.1.4 **InceptionV3**

After preprocessing the dataset we then call built-in InceptionV3 model from google colab.

Fig 5.10.1: Calling InceptionV3 pretrained model from Google Colab..



As the InceptionV3 is a functional model so we must convert it into Sequential model and we then use Flattening layer to convert 2d array into 1d array and finally we mapped output of the model with our 38 classes.

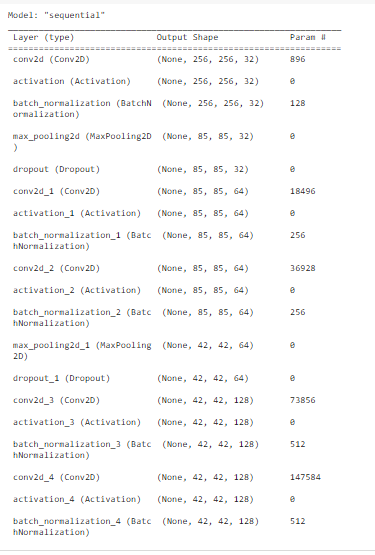


Fig 5.10.2: InceptionV3 model summarized.

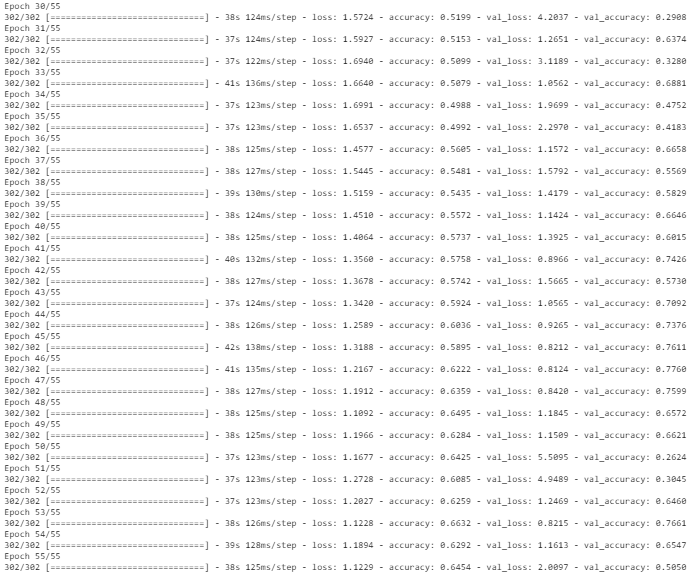
Then we run model.fit() method to fit the dataset images with the above mentioned architecture. Here, we use total epoch 55 and batch size for each step is 10.

Fig 5.10.3: Running epochs in InceptionV3 model.

Using InceptionV3 we have our train accuracy is 66.32% and validation accuracy is 77.60%.

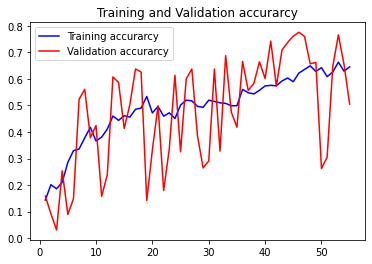
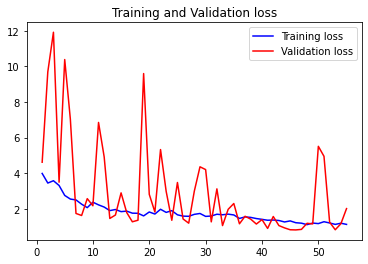
Graph for Train and Validation accuracy and Train and Validation loss is given below –

Fig 5.10.4: Accuracy and Loss graph in InceptionV3 model.

Confusion matrix:

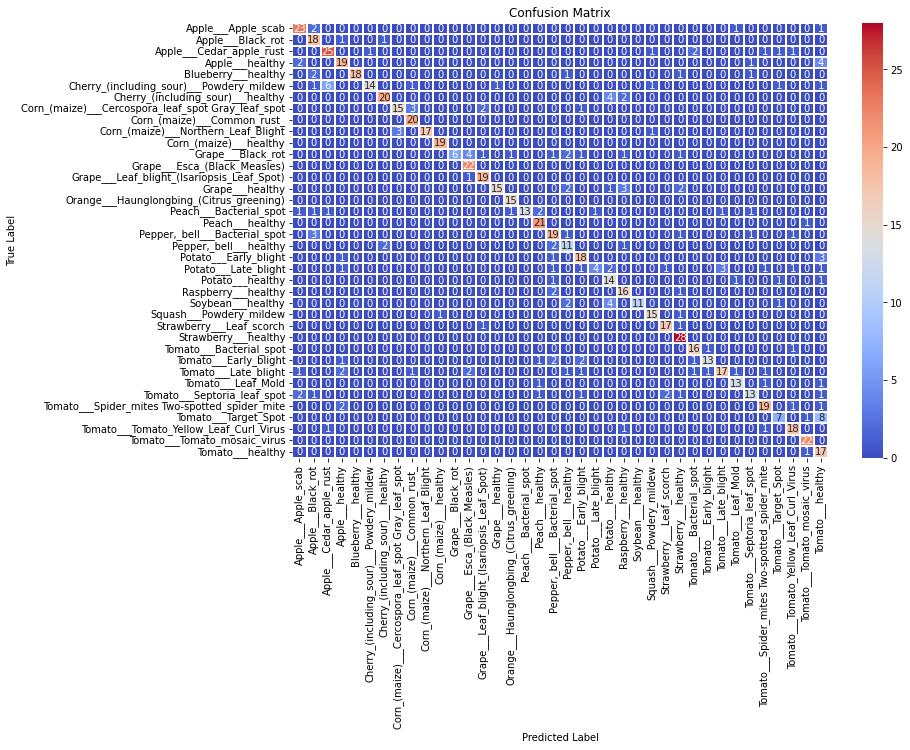


Fig 5.10.5: Confusion Matrix for 38 classes in InceptinV3 model.

Precision, Recall and F1 score:

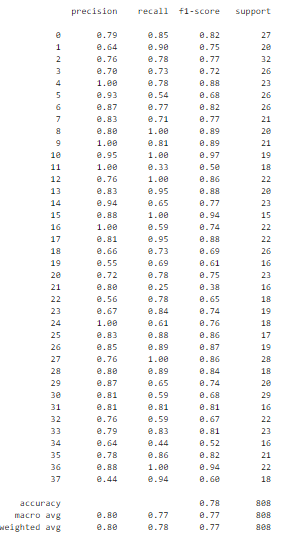


Fig 5.10.6: Precision, Recall and F1 score for 38 classes in IncetionV3 model.

ROC AUC score: is 0.8839659736902374

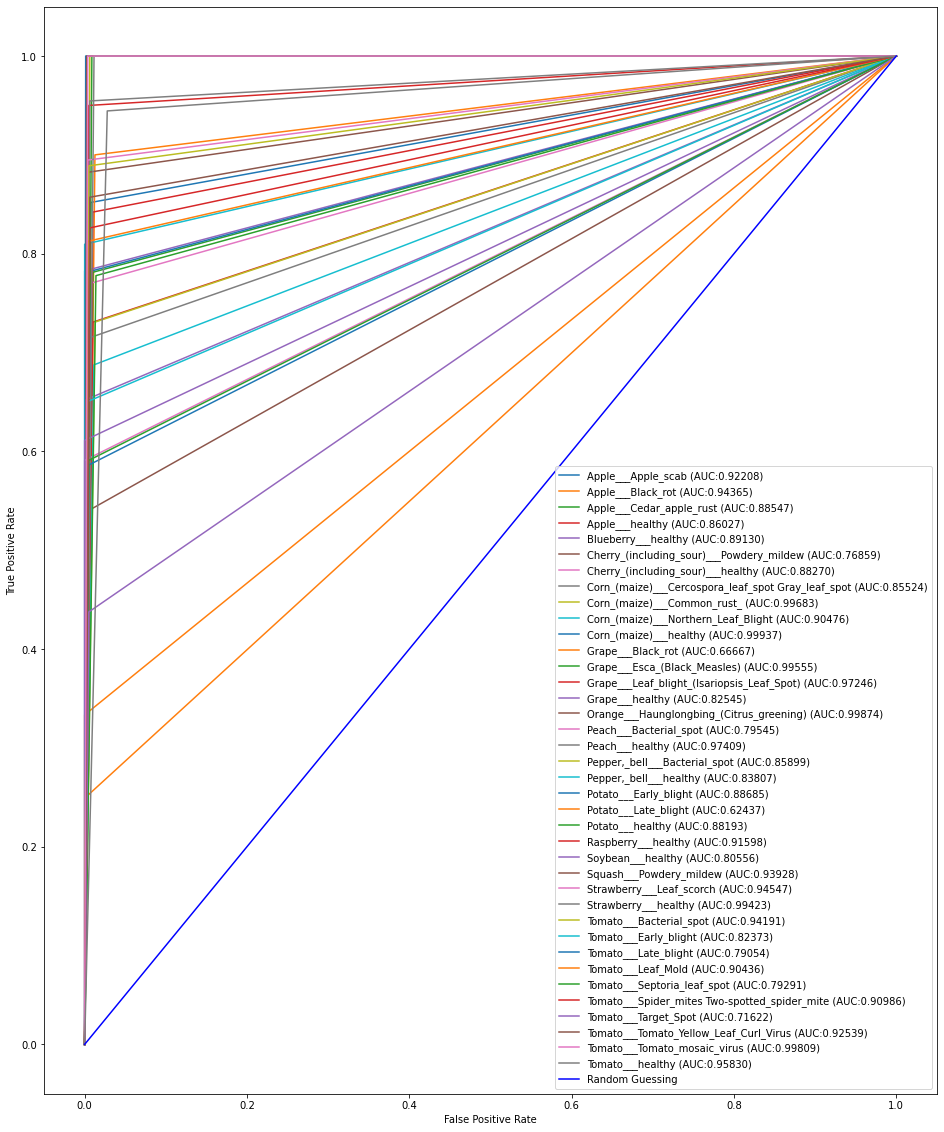


Fig 5.10.7: ROC AUC score for 38 classes in InceptionV3 model.

#### 5.1.1.5 **MobileNetV2**

After preprocessing the dataset we then call built-in MobileNetV2 model from google colab.

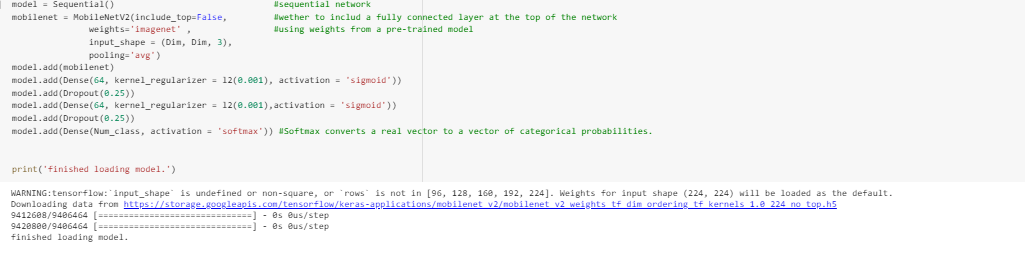


Fig 5.11.1: Calling MobileNetV2 pretrained model from Google Colab..

As the MobileNetV2 is a functional model so we must convert it into Sequential model and we then use Flattening layer to convert 2d array into 1d array and finally we mapped output of the model with our 38 classes.

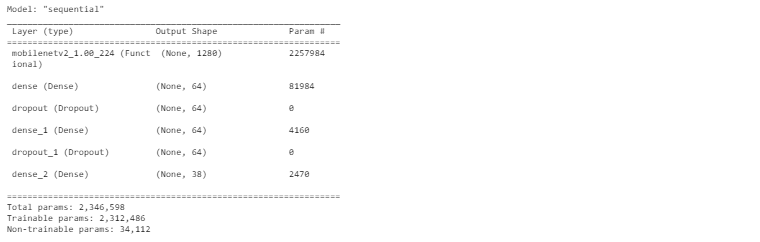


Fig 5.11.2: MobileNetV2 model summarized.

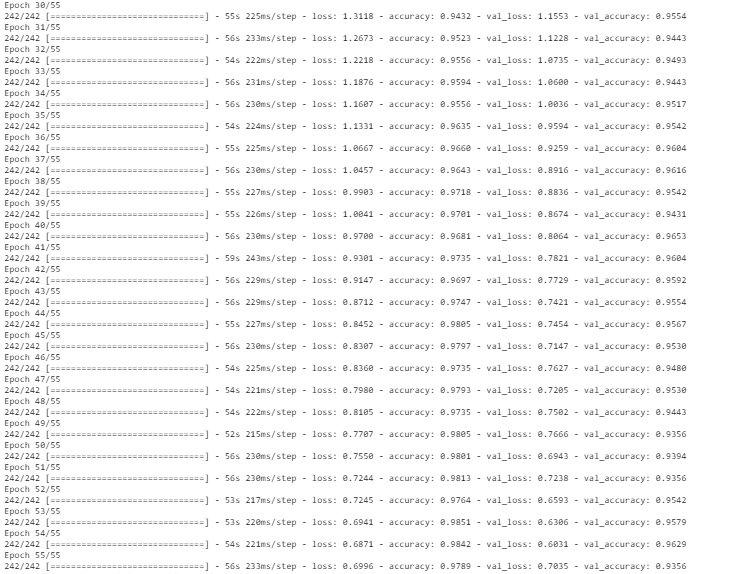
Then we run model.fit() method to fit the dataset images with the above mentioned architecture. Here, we use total epoch 55 and batch size for each step is 10.

Fig 5.11.3: Running epochs in MobileNetV2 model.

Using MobileNetV2 we have our train accuracy is 98.51% and validation accuracy is 96.53%.

Graph for Train and Validation accuracy and Train and Validation loss is given below –

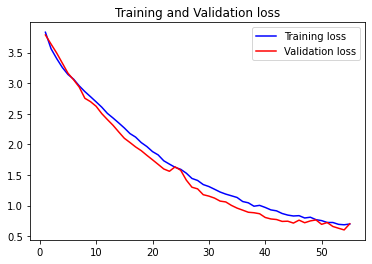
 

Fig 5.11.4: Accuracy and Loss graph in MobileNetV2 model.

Confusion matrix:

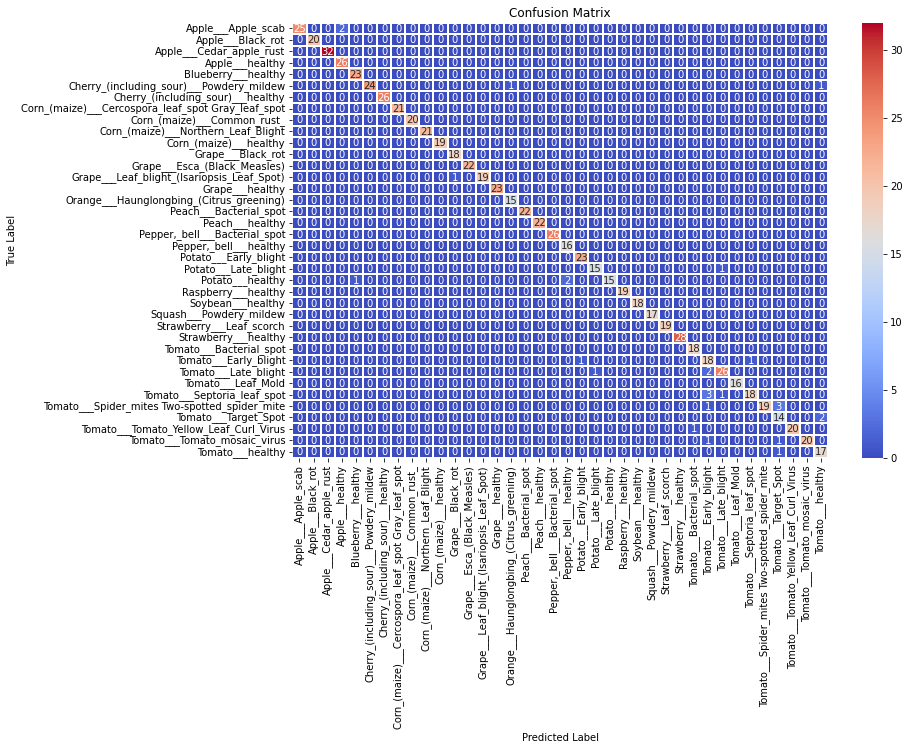


Fig 5.11.5: Confusion Matrix for 38 classes in MobileNetV2 model.

Precision, Recall and F1 score:

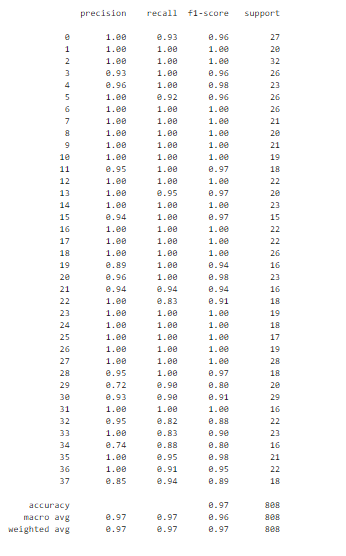


Fig 5.11.6: Precision, Recall and F1 score for 38 classes in MobileNetV2 model.

ROC AUC score: is 0.9823164948171881

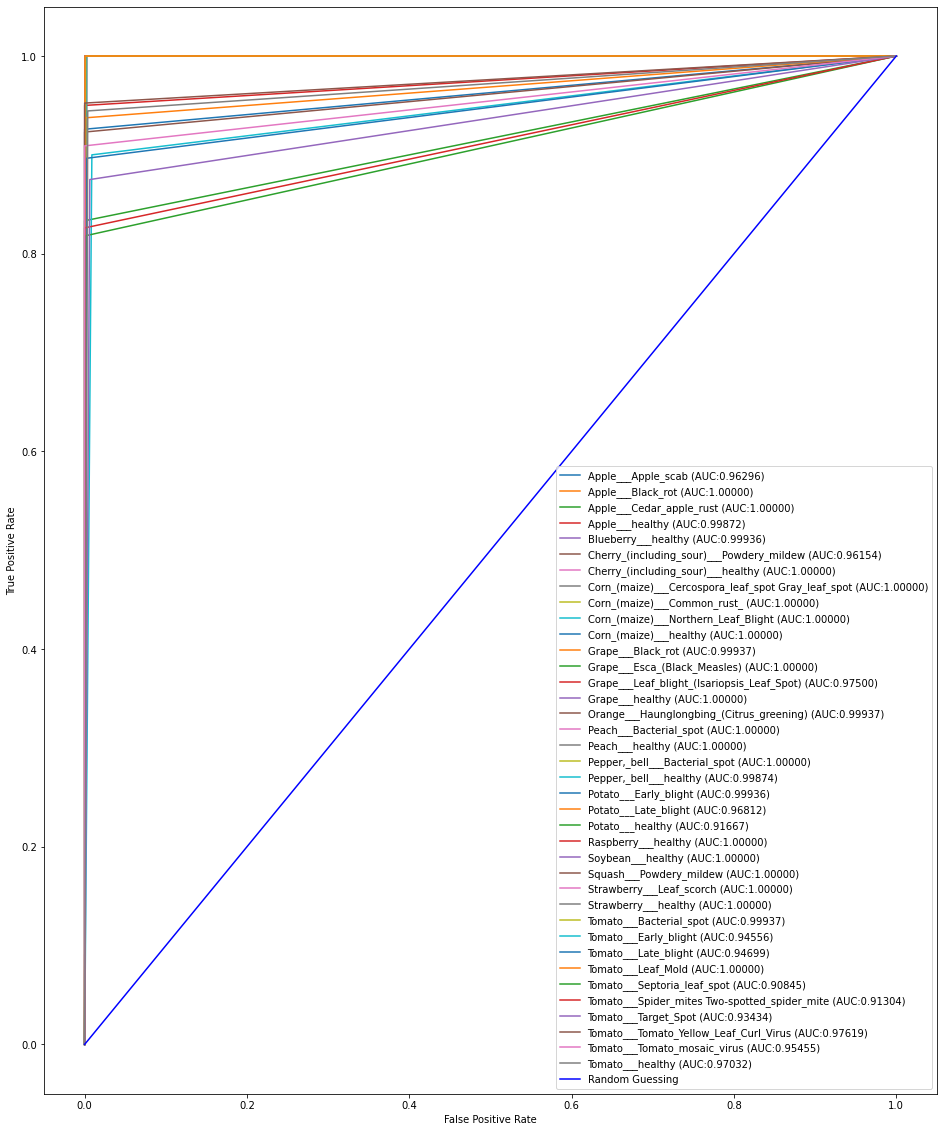


Fig 5.11.7: ROC AUC score for 38 classes in MobileNetV2 model.

#### 5.1.1.6 **Xception**

After preprocessing the dataset we then call built-in Xception model from google colab.

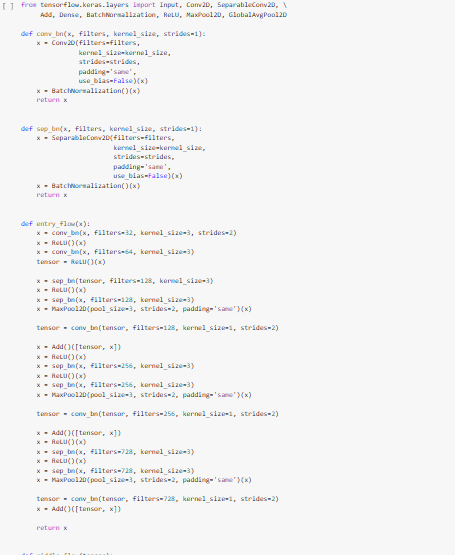


Fig 5.12.1: Calling Xception pretrained model.

As the Xception is a functional model so we must convert it into Sequential model and we then use Flattening layer to convert 2d array into 1d array and finally we mapped output of the model with our 38 classes.

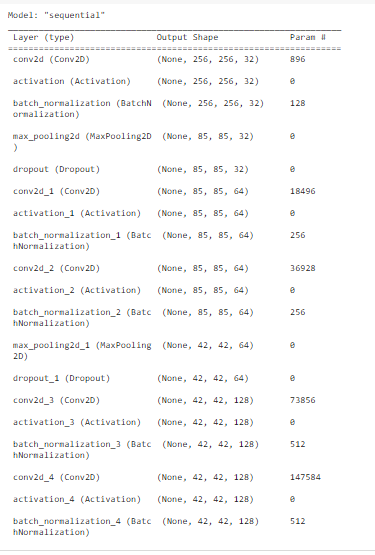
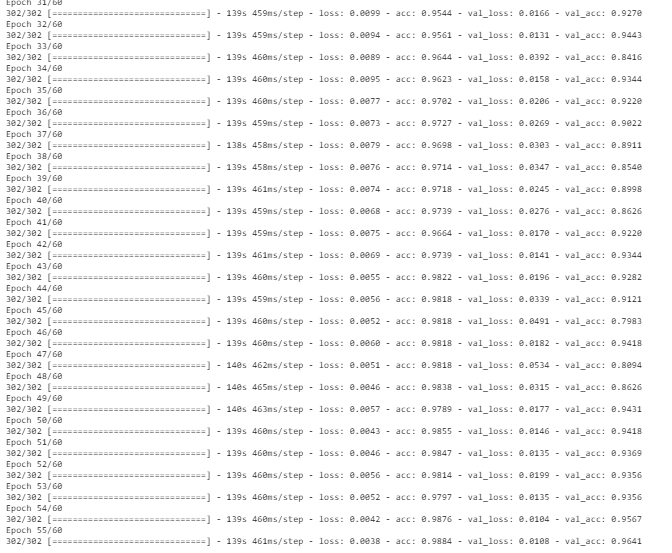


Fig 5.12.2: Xception model summarized.

Then we run model.fit() method to fit the dataset images with the above mentioned architecture. Here, we use total epoch 60 and batch size for each step is 10.

Fig 5.12.3: Running epochs in Xception model.



Using Xception we have our train accuracy is 98.84% and validation accuracy is 96.41%.

Graph for Train and Validation accuracy and Train and Validation loss is given below –

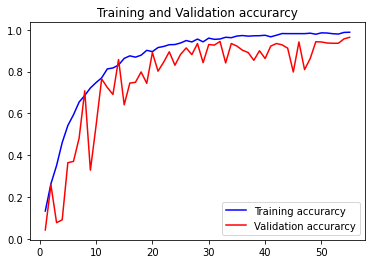
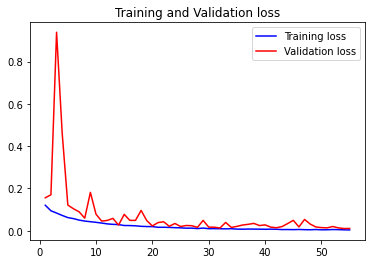


Fig 5.12.4: Accuracy and Loss graph in Xception model.

Confusion matrix:

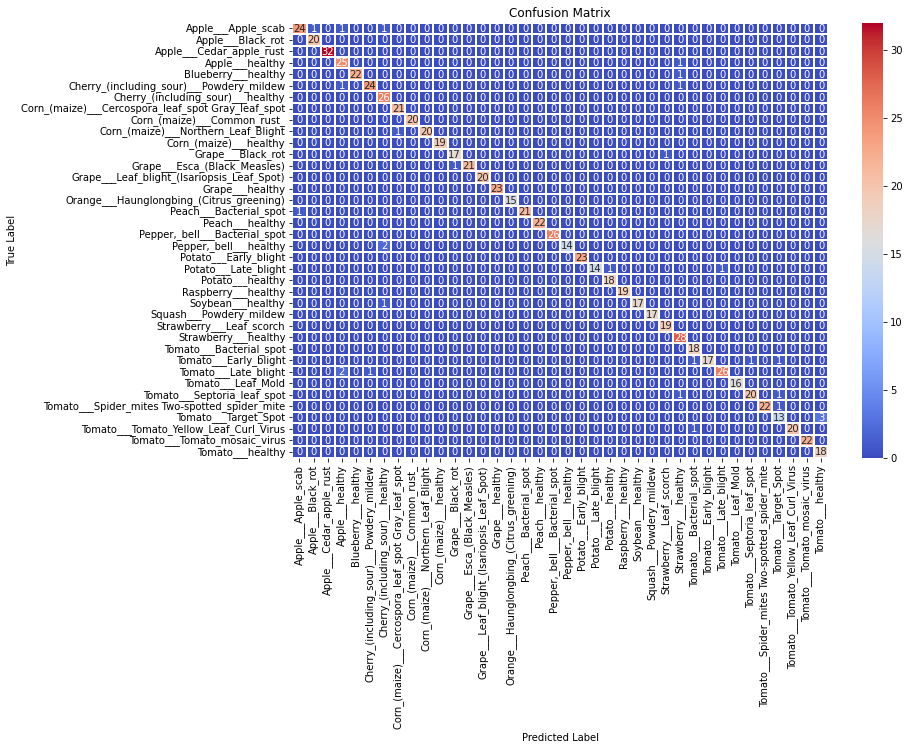


Fig 5.12.5: Confusion Matrix for 38 classes in Xception model.

Precision, Recall and F1 score:

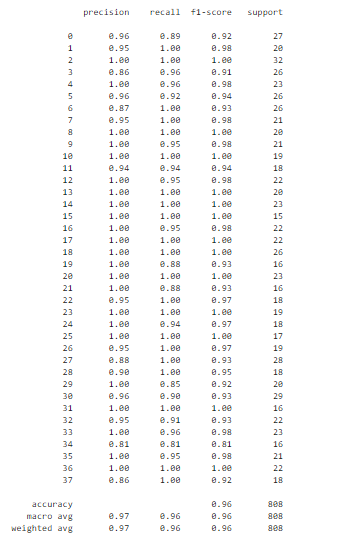


Fig 5.12.6: Precision, Recall and F1 score for 38 classes in Xception model.

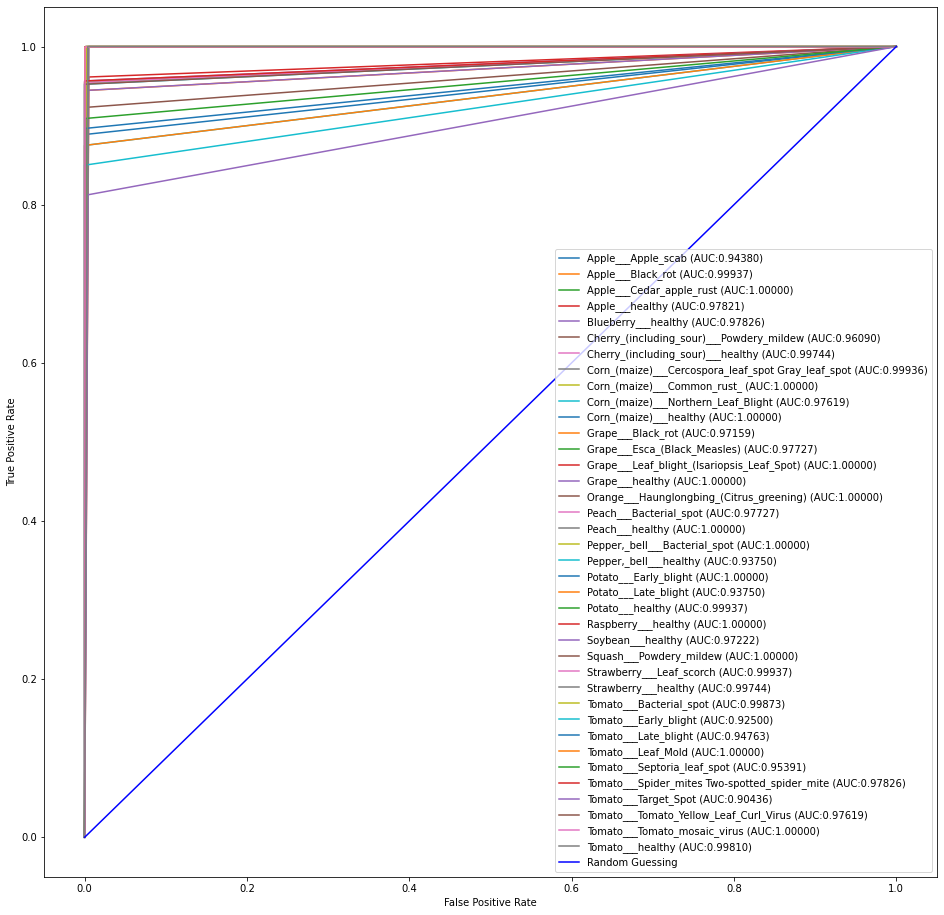
ROC AUC score: is 0.9811908264582

Fig 5.12.7: ROC AUC score for 38 classes in Xception model.

## 5.2 **Results**

Our initial work with this topic has accuracy of **99.22%** using **0.4%** of validation data with the model *Xception* which has already been published in [*4th International Conference on Intelligent Computing and Optimization 2021(ICO2021)*](Mazumder%20P.P.,%20Hossain%20M.,%20Riaz%20M.H.%20(2022)%20Classification%20and%20Detection%20of%20Plant%20Leaf%20Diseases%20Using%20Various%20Deep%20Learning%20Techniques%20and%20Convolutional%20Neural%20Network.%20In:%20Vasant%20P.,%20Zelinka%20I.,%20Weber%20GW.%20(eds)%20Intelligent%20Computing%20&%20Optimization.%20ICO%202021.%20Lecture%20Notes%20in%20Networks%20and%20Systems,%20vol%20371.%20Springer,%20Cham.%20https:/doi.org/10.1007/978-3-030-93247-3_14) [37] from Springer as part of the “[*Lecture Notes in Networks and Systems*](https://www.springer.com/series/15179)” book series. When we trained our model Google Colab had the feature of 25GB of free RAM and unlimited GPU. That’s the reason we were able to use most of our data images to train the model so the validation accuracy is well enough to predict images with maximum accuracy. But after the recent update of Google Colab the feature is restricted to 12GB RAM and 6hr GPU. So, we are forced to use less image to train our data and that’s why the validation accuracy is less than the original.

Previously, with InceptionV3 we obtained *(Train from scratch)* **80.28%** (epoch=25, Optimizer=Adam) to **92.04%** (epoch=100, Optimizer=Adam) and (*Train using pre-trained model)* **85.53%** (epoch=25, Optimizer=Adam)). Also using Xception model the accuracy varies from **98.625%- 99.22%**.

In summarizing our recent work -

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Deep learning Architecture** | **Parameters**  **(in million)** | **Epoch required** | **Training Accuracy** | **Validation Accuracy** | **Training Loss** | **Validation Loss** | **Precision** | **Recall** | **F1-Score** |
| VGG16 | 138M | 55 | 98.76 | 91.46 | 0.04 | 0.41 | 0.92 | 0.91 | 0.91 |
| VGG19 | 138M | 55 | 96.47 | 90.23 | 0.14 | 0.47 | 0.91 | 0.90 | 0.90 |
| ResNet50 | 23.6M | 55 | 98.55 | 95.67 | 0.04 | 0.16 | 0.97 | 0.96 | 0.96 |
| InceptionV3 | 41.2M | 55 | 66.32 | 77.60 | 1.12 | 2.00 | 0.80 | 0.77 | 0.77 |
| MobileNetV2 | 3.2M | 55 | 98.51 | 96.53 | 0.62 | 0.59 | 0.97 | 0.97 | 0.96 |
| Xception | 22.8M | 55 | 98.84 | 96.41 | 0.00 | 0.01 | 0.97 | 0.96 | 0.96 |

Table 5.1: Precision, Recall, F1-Score, Accuracy, Loss of Different Deep learning architecture

# **CHAPTER SIX**

**CONCLUSION AND FUTURE WORK**

Pre-trained models have been greatly used in machine learning and computer vision applications also including plant disease identification. The achievements of convolutional neural networks in object recognition and image classification have made immense advancement in the past few years. The main purpose of this system is to improve the efficiency of the automatic plant disease detection. Our above-mentioned results show that a large, deep convolutional neural network can achieve significant results on a highly challenging dataset with the help of purely supervised learning. Experimental results show that the proposed system can successfully detect and classify the plant disease with accuracy of 96.41%. In future work, we will extend our database for more plant disease identification and use large number of data as training data as training purpose in classification. As we increase the training data, the accuracy of the system will be high and then we can compare the accuracy rate and speed of system.

# **References**

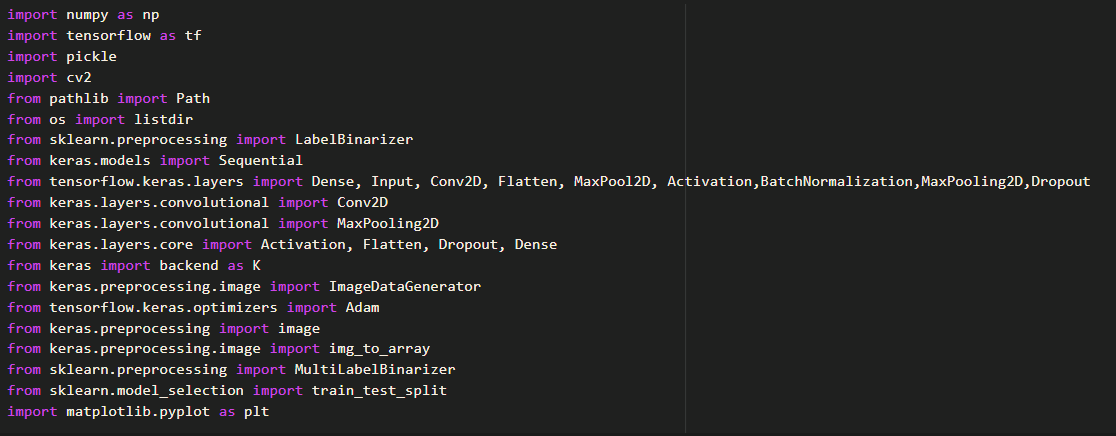
|  |  |
| --- | --- |
|  | Sastry, K.S.: Plant Virus and Viroid Diseases in the Tropics, vol. II. Springer, Heidelberg (2013) |
|  | Tai, A. P., Martin, M. V., and Heald, C. L. (2014). Threat to future global food security from climate change and ozone air pollution. *Nat. Clim. Chang* 4, 817–821. doi: 10.1038/nclimate2317 |
|  | Report of the Plenary of the Intergovernmental Science-PolicyPlatform on Biodiversity Ecosystem Services on the work of its fourth session (2016). *Plenary of the Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services Fourth session*. Kuala Lumpur. Available online at: <http://www.ipbes.net/sites/default/files/downloads/pdf/IPBES-4-4-19-Amended-Advance.pdf>, last accessed 2021/01/04. |
|  | Strange, R. N., and Scott, P. R. (2005). Plant disease: a threat to global food security. *Phytopathology* 43, 83–116. doi: 10.1146/annurev.phyto.43.113004.133839 |
|  | Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances In Neural Information Processing Systems,pp. 1–9 (2012) |
|  | Hall, D., McCool, C., Dayoub, F., Sunderhauf, N., Upcroft, B.: Evaluation of features for leaf classification in challenging conditions. In: 2015 IEEE Winter Conference on Applications of Computer Vision, pp. 797–804 (2015) |
|  | Jin, J., Fu, K., Zhang, C.: Traffic sign recognition with hinge loss trained convolutional neural networks. IEEE Trans. Intell. Transp. Syst. **15**, 1991–2000 (2014) |
|  | Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* 88, 303–338. doi: 10.1007/s11263-009-0275-4 |
|  | Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* 88, 303–338. doi: 10.1007/s11263-009-0275-4 |
|  | Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei L. (2009). “Imagenet: A large-scale hierarchical image database,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. (IEEE).* |
|  | Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, eds F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Curran Associates, Inc.), 1097–1105. |
|  | Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, eds F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Curran Associates, Inc.), 1097–1105. |
|  | Simonyan, K., and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556. |
|  | Zeiler, M. D., and Fergus, R. (2014). “Visualizing and understanding convolutional networks,” in *Computer Vision–ECCV 2014*, eds D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Springer), 818–833. |
|  | He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. arXiv:1512.03385. |
|  | Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. |
|  | Kutty, S. B., Abdullah, N. E., Hashim, H., Kusim, A. S., Yaakub, T. N. T., Yunus, P. N. A. M., & Abd Rahman, M. F. (2013, April). Classification of watermelon leaf diseases using neural network analysis. In *2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)* (pp. 459-464). IEEE. |
|  | Sannakki, S. S., Rajpurohit, V. S., Nargund, V. B., & Kulkarni, P. (2013, July). Diagnosis and classification of grape leaf diseases using neural networks. In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)* (pp. 1-5). IEEE. |
|  | Akhtar, A., Khanum, A., Khan, S. A., & Shaukat, A. (2013, December). Automated plant disease analysis (APDA): performance comparison of machine learning techniques. In *2013 11th International Conference on Frontiers of Information Technology* (pp. 60-65). IEEE. |
|  | Mokhtar, U., Ali, M. A., Hassenian, A. E., & Hefny, H. (2015, December). Tomato leaves diseases detection approach based on support vector machines. In *2015 11th International Computer Engineering Conference (ICENCO)* (pp. 246-250). IEEE. |
|  | Patki, S. S., & Sable, G. S. (2016). Cotton leaf disease detection & classification using multi SVM. *International Journal of Advanced Research in Computer and Communication Engineering*, *5*(10), 165-168. |
|  | Gavhale, K. R., & Gawande, U. (2014). An overview of the research on plant leaves disease detection using image processing techniques. *IOSR Journal of Computer Engineering (IOSR-JCE)*, *16*(1), 10-16. |
|  | Tajane, Vinita, and N. J. Janwe. 2014. Medicinal Plants Disease Identification Using Canny Edge Detection Algorithm, Analysis and CBIR. *International Journal of Advance Research in Computer Science and Software Engineering*. 4(6): 530-536. |
|  | Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in plant science, 7, 1419. (pg-3) |
|  | Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 31, No. 1). |
|  | Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258). |
|  | https://www.researchgate.net/figure/Artificial-Neural-Network-Deep-Learning-model\_fig8\_313408173 |
|  | Lu, J., Tan, L., & Jiang, H. (2021). Review on convolutional neural network (CNN) applied to plant leaf disease classification. Agriculture, 11(8), 707. |
|  | Ferguson, M., Ak, R., Lee, Y. T. T., & Law, K. H. (2017, December). Automatic localization of casting defects with convolutional neural networks. In 2017 IEEE international conference on big data (big data) (pp. 1726-1735). IEEE. |
|  | Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. |
|  | He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778). |
|  | Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826). |
|  | Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520). |
|  | Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258). |
|  | Agarap, A. F. (2018). Deep learning using rectified linear units (relu). ArXiv Preprint ArXiv:1803.08375. |
|  | Banerjee, K., Gupta, R. R., Vyas, K., & Mishra, B. (2020). Exploring alternatives to softmax function. arXiv preprint arXiv:2011.11538. |
|  | Mazumder P.P., Hossain M., Riaz M.H. (2022) Classification and Detection of Plant Leaf Diseases Using Various Deep Learning Techniques and Convolutional Neural Network. In: Vasant P., Zelinka I., Weber GW. (eds) Intelligent Computing & Optimization. ICO 2021. Lecture Notes in Networks and Systems, vol 371. Springer, Cham. https://doi.org/10.1007/978-3-030-93247-3\_14 |

# **APPENDIX**

## APPENDIX A

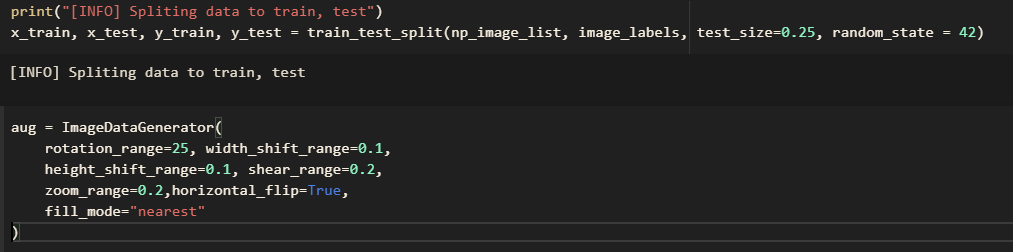
Library Functions:

We’ve used a bunch of library functions from Python Library. These are –



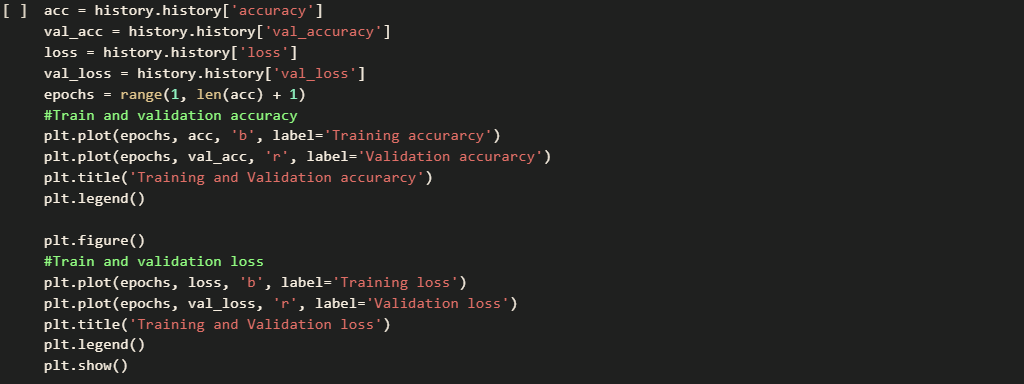
## APPENDIX B

Preprocessing Data:



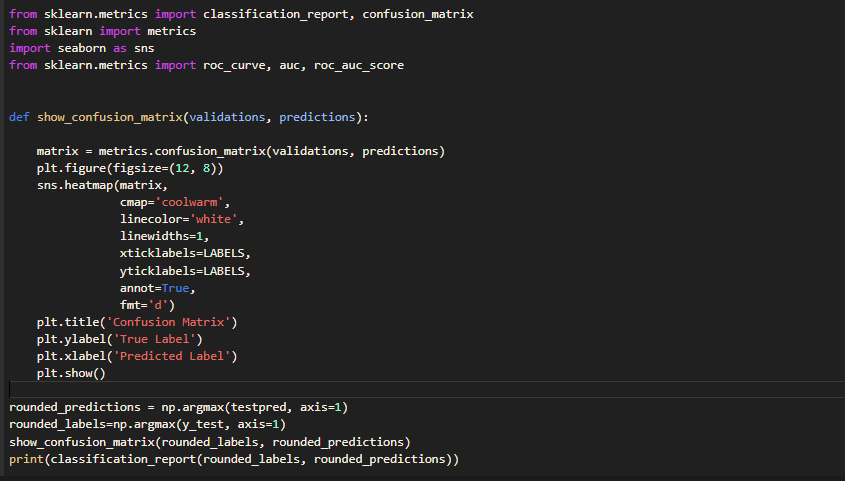
## APPENDIX C

Calculating Accuracy and Loss



## APPENDIX D

Calculating Confusion Matrix



## APPENDIX E

Calculating ROC AUC score

