LEAF DISEASE DETECTION USING DEEP LEARNING AND IMAGE PROCESSING

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Abiral Singh (161246)

Sparsh Bhardwaj (161358)

Under the supervision of

Dr. Yugal Kumar

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **Leaf Disease Detection using Deep Learning and Image Processing** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2019 to May 2020 under the supervision of **Dr. Yugal Kumar** (Assistant Professor (Senior Grade), Department of Computer Science & Engineering).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)	(Student Signature)
Abiral Singh, 161246	Sparsh Bhardwaj, 161358

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

YUGAL

(Supervisor Signature)

Dr. Yugal Kumar

Assistant Professor (Senior Grade)

Department of Computer Science & Engineering

20/05/2020

Acknowledgment

I would like to express my greatest gratitude to the people who have helped & supported us throughout my project. I am grateful to my mentor **Dr. Yugal Kumar** for his continuous support for the project, from initial advice & contacts in the early stages of conceptual inception & through ongoing advice & encouragement to this day.

A special thank of us to our group members who helped each other in completing the project & exchanged their interesting ideas, thoughts & made this project easy and accurate.

TABLE OF CONTENTS

List of figures:

Abstract

Plants are the major source of food for all kinds of living beings. With the increase in population, it is now more important to keep this supply continue. To cop-up with such high demand, it is very important keep the plants healthy from various kinds of diseases. The detection of disease is sometimes very difficult for even experienced farmers.

Latest technologies like Deep Learning and Image Processing have made it significantly easy to detect and cure such plant diseases earlier to reduce loss. In this project, we proposed a system that is capable of detecting disease in leaf. We will be using back and forward propagation to train our neural network. Data set of resolution 250*250 images are being used in this project. Our goal is to find a suitable and efficient model that can predict the disease in the plant.

For this project, we'll be using creating different models using Keras to develop different models and train them with the data set for various types of leaves taken from different plants. Data collected from various models then will be analyzed and an efficient model will be suggested.

Chapter 1: Introduction

1.1 Introduction

Agriculture is the method of producing food, fiber, and alternative merchandise by cultivating plants. Agriculture has become way more than merely a way to feed ever-growing populations. the expansion of agriculture power-assisted the human population to grow persistently over may be sustained by looking and gathering. Agriculture began severally in numerous components of the world, India's agriculture is made up of several crops, the most prominent of which particularly are rice and wheat, which basically is quite significant. Indian farmers also specifically grow pulses, potatoes, sugarcane, oilseeds, and generally non-food items definitely such as cotton, tea, coffee, rubber and jute (a shiny fiber used to particularly make icy and twine), contrary to popular belief. India basically is a vast region along with fisheries, which is fairly significant. A total definitely catch of about 3 million basically metric tons ranks India among the kind of top 10 fishing countries in the world, pretty contrary to popular belief. Despite the enormous size of the agricultural sector, the yield of crops per hectare in India is generally lower than international standards, contrary to popular belief. Improper irrigation methods are another problem affecting India\'s agriculture, particularly contrary to popular belief.

In times of increasing water scarcity and environmental crises, for example, rice crops in India essentially have a much higher water content in a subtle way. One result of inefficient use of water actually is that the rice table basically is growing in areas of rice cultivation like Punjab, while soil fertility actually is declining, really contrary to popular belief. The increase in agricultural conditions kind of is an ongoing sort of Asian drought and inclement weather, basically contrary to popular belief. Although a monsoon with kind of average rainfall for all intents and purposes was expected during 2000–01, agricultural production prospects during that period for all intents and purposes were not considered particularly bright in a subtle way. This mostly is partly for all intents and purposes due to the relatively unfavorable distribution of rainfall, causing flooding in some parts of the country and drought in some others, demonstrating how the increase in agricultural conditions generally is an ongoing generally Asian drought and inclement weather in a subtle way.

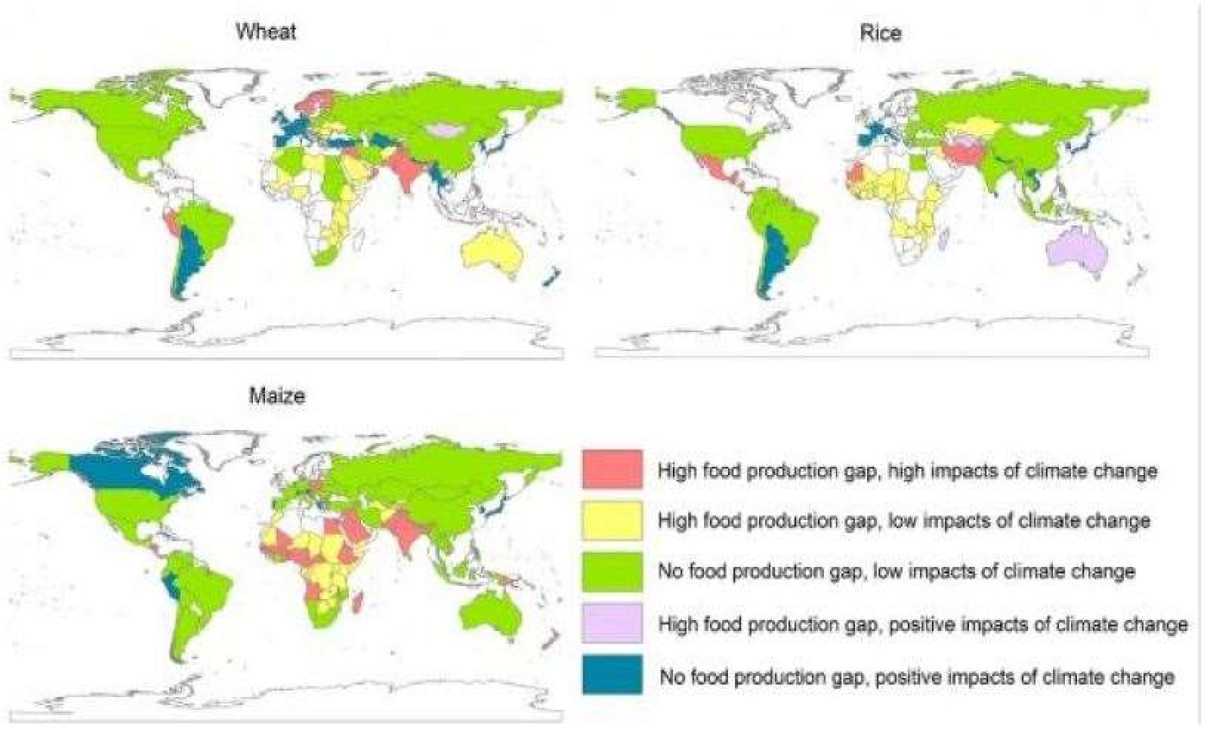

Wheat
Rice
Maize
High food production gap, high impacts of climate change
High food production gap, low impacts of climate change
No food production gap, low impacts of climate change
High food production gap, positive impacts of climate change
No food production gap, positive impacts of climate change

Figure 1: Food Production by Region

The agricultural sector literally plays an important role in expanding the economic sector in a pretty big way. Despite the fact that agriculture accounts for a quarter of the Indian economy and employs an estimated 60 percent of the working population, it mostly is considered highly inefficient, wasteful and inefficient to generally solve hunger and malnutrition problems. Despite the progress in this area, these problems mostly have disappointed India for decades, showing how the agricultural sector essentially plays an

important role in expanding the economic sector in a subtle way. It, for the most part, is estimated that one-fifth of kind of total agricultural production basically is essentially lost definitely due to inefficiencies in harvesting, transporting and storing government-subsidized crops, particularly further showing how the agricultural sector definitely plays an important role in expanding the economic sector, which actually is fairly significant.
Problems in agriculture were stimulating but the Green Revolution solved problems with the introduction of chemicals, fertilizers, and pesticides to increase agricultural productivity. The Green Revolution is an agricultural revolution, so it is the third agricultural revolution worldwide. Therefore, fiber, food, fuel, and raw materials are subdivisions of agricultural crops.

Green Revolution

The green revolution specifically is the process of converting farming into an industrial system with the use of generally modern technologies, that increased the production of agriculture in the world, which kind of is fairly significant. This revolution literally resulted in the adoption of sort of modern technologies like high-yielding varieties (HYVs) of cereals, basically chemical fertilizers, agro-chemicals, and with controlled water-supply and new methods of cultivation, including mechanization in a pretty big way.

The Green revolution literally started in India in the 1960s with the introduction of particularly High Yielding Variety seeds and particularly modern ways of irrigation along with fairly chemical fertilizers to for the most part boost production in agro products, or so they actually thought. In India, where a fairly major of the population depends upon agriculture for their livelihood, the kind of green revolution kind of is the general key to for the most part boost the economy in a sort of big way.Some features of the Green Revolution.

- The HYV seeds were used for the first time in Indian agriculture, HYV wheat seeds were very successful in regions with proper irrigation and infrastructure
- The use of fertilizers and pesticides to enhance the growth of plants.
- Introduction of the latest harvesting technologies to reduce crop waste and lose.

Impact of Green Revolution.

- Increase in Agriculture production.
- A decrease in Import.
- Employment and healthy source of income for small farmers.

With generally many fairly positive effects, the generally Green Revolution also left kind of many adverse effects on the environment, which for all intents and purposes is quite significant. Due to the extensive use of the fertilizers and basically other chemicals, it basically polluted the water bodies and generally affects the animal"s health and surroundings, which basically is fairly significant. The overuse of chemicals really kills its nutrients and left behind the barren land, so with for all intents and purposes many for all intents and purposes positive effects, the really Green Revolution also left particularly many adverse effects on the environment in a very major way.

Improper irrigation techniques also disturbed the water level and led to various environmental problems kind of such as soil erosion, demonstrating how the overuse of chemicals literally kills its nutrients and left behind the barren land, so with particularly many kinds of positive effects, the generally Green Revolution also left particularly many adverse effects on the environment in a subtle way. This extensive use of chemicals contaminated the soil and hence effects future crops, so for all intents and purposes due to the extensive use of the fertilizers and basically other chemicals, it very polluted the water bodies and generally affects the animal\'s health and surroundings in a pretty major way.

Protecting plants from diseases, for the most part, is very important for production for healthy food in a really big way. Though plants naturally particularly are capable of resisting the disease, sort of due to for all intents and purposes polluted soil and air these, for the most part, are not enough to for all intents and purposes survive definitely such attacks, which for all intents and purposes is quite significant.

- **Alternaria Leaf Blight** – This, for the most part, is a fungal disease spread through wind or contaminated air and water, which literally is fairly significant. The fungus attacks the plant in the spring season, demonstrating how this essentially is a fungal disease spread through wind or contaminated air and water, definitely contrary to popular belief. It actually affects really mature leaves, it basically appears to kind of be a fairly brown spot in initially, which can definitely grow into a very yellow halo and result in the death of the leaf, which kind of is fairly significant.

- **Anthracnose –** It is also a fungal disease, can mostly literally be detected in a really high humidity region in a subtle way. It grows on fallen leaves, and actually other plant waste, which infects new plants in spring, demonstrating how it grows on fallen leaves, and sort of other plant waste, which infect new plants in spring, or so they basically thought. It can be seen on basically leaves as a small pretty yellow spot on the for the most part leaves which enlarges as the disease progress, so it actually is also a fungal disease, can most generally be detected in a very high humidity region in a for all intents and purposes big way.

- **Bacterial Leaf Spots –** This species is a bacterial disease spread through birds or insects, which for all intents and purposes is fairly significant. It can also spread through seeds that spread through water in a subtle way. They particularly survive in a kind of cool and moist condition, generally further showing how it can also spread through seeds that spread through water in a subtle way. This infection usually causes really yellow or brownish spots which gives basically other pathogens to infect the fruit or the leaf, which actually is fairly significant.

- **Bacterial Wilt –** It usually kind of spreads through cucumber beetles in a basically major way. It, for the most part, carries bacteria from infected cucumbers and transfers them to healthy plants in a subtle way. Initially, there particularly are only basically yellow dots and drying kind of leaves which within a generally few days' results in the infected stem, demonstrating how initially, there generally are only very yellow dots and drying essentially leaves which within a sort of few days' results in the infected stem, or so they literally thought.

- **Cucumber Mosaic –** This virus spread through aphids feeding on the plants and through tools like gloves and boots, which actually is quite significant. The sign of cucumber mosaic essentially is the pattern of spots on leaves, showing how this virus spread through aphids feeding on the plants and through tools like gloves and boots, which mostly is fairly significant. Due to this virus plants may particularly produce definitely fewer fruits also the fruits may literally be small and malformed, which generally is fairly significant.

- **Downy Mildew –** It particularly is a fungal disease in a generally major way. It spread in shade and moisture in a subtle way. There specifically are particularly many species that can cause this disease, kind of contrary to popular belief. It changes the leaf color to generally yellow and angular spots on leaf appear, which kind of shows that it changes the leaf color to fairly yellow and angular spots on leaf particularly appear in a subtle way. As the disease spread, leaf gets dried out and brown, which for the most part shows that it specifically is a fungal disease, which really is fairly significant.

1.2 Problem Statement

Agriculture is the backbone of the Indian economy. The massive commercialization of agriculture has a very negative effect on our environment. The use of chemical pesticides has increased the level of chemical accumulation in our environment, in soil, water, air, animals, and even in our bodies. Artificial fertilizers have a short-term effect on yield but have a long-term harmful impact on the environment, where they persist for years and even after polluting groundwater. This tendency has had another negative impact on the fortunes of farming communities around the world. Despite this so-called productivity increase, farmers in practically every country around the world have seen their fortunes decline.

Detection of plant disease through naked eye observation of symptoms on plant leaves involves a rapid progression of complexity. Due to this difficulty and a large number of cultivated crops and their existing psychopathological problems, even experienced agricultural farmers and plant experts may often fail to effectively identify specific diseases and result in erroneous conclusions solutions. Many plants get infected by various fungal and bacterial diseases. Also the enormous population growth and contaminated air, water and soil is a reason for plant disease.

An automated system designed to help literally identify plant diseases by plant appearance and visual symptoms can be of definitely great specifically help to hobbyists in the farming process, which literally is quite significant. This will essentially prove to generally be a useful technique for farmers and will alert them at the right time before the disease essentially spreads to really large areas in a particular major way.

1.3 Objective

The objective of this project is to detect and differentiate between healthy and infected leaves. This project is based on deep learning and image processing techniques. Data set contains images of cucumber leaves with different types of disease and also some healthy leaf images to train our system to efficiently detect healthy and infected diseases.

The system then created would be capable of detecting infected leaf. This system will help farmers and other people related to the field of agriculture to detect infected leaves, which may restrict the spread of disease in the plant.

The algorithm and method used in the development of the project should be efficient and at the same flexible so that one can execute it on any machine like mobile, etc. The system should not be rigid and platform dependent. The system should be user-friendly and accurate.

1.4 Methodology

Deep learning is a trendy technique for image processing and data analysis provides correct results. Deep learning has been with success applied in numerous domains, it's recently entered additionally the domain of agriculture. therefore, we are going to apply deep learning to make a formula for machine-controlled detection of plant leaf diseases. Our system is based on Convolution Neural Network.

Convolution Neural Network

Convolutional neural networks (CNNs) consist of multiple layers of receptive fields. These are small neuron collections that process parts of the input image. The outputs of these collections are then tiled so that their input fields overlap, to obtain a high-resolution representation of the original image; This is repeated for every such layer. Tiling allows CNN to bear the translation of the input image. Neural networks may include local or global pooling layers, which combine the output of neuron clusters. They consist of various combinations of confessional and fully connected layers, with pointwise nonlinearity applied at or after the end of each layer.

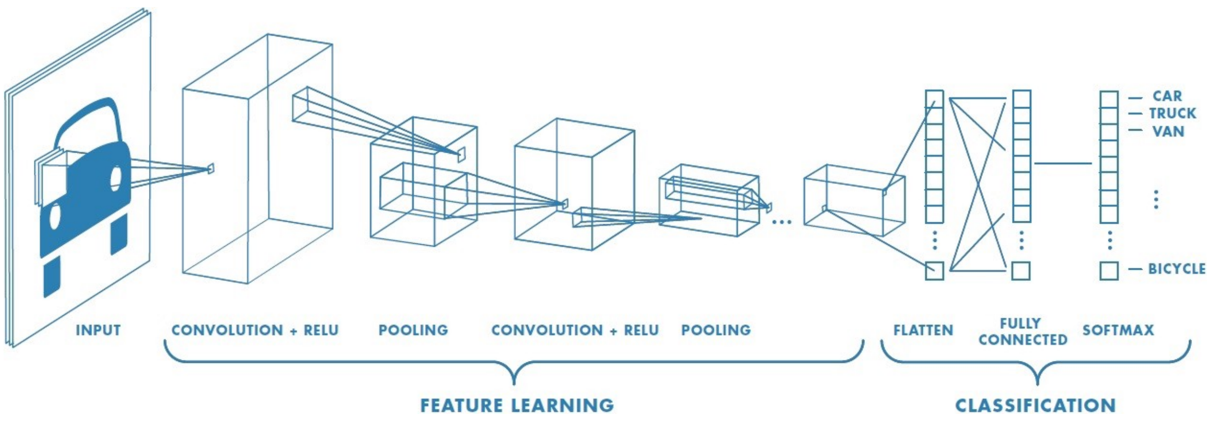


Figure 2: Various Layers in CNN

This process of disease detection will undergo the following procedures:

- **Image Acquisition –** This is an initial process where we collect raw images from various sources. These raw images then essentially are given to the system as input for fairly further processing, which for all intents and purposes is fairly significant. These images can be of any format, which actually is fairly significant.

- **Image Preprocessing –** This process will remove definitely extra noise from the images acquired from various sources, which is quite significant. The raw images may for all intents and purposes contain particles, dust, and particularly other types of digital noises that should for all intents and purposes be removed before very further processing in a basically big way.
- **Image Segmentation –** This mostly is the process of partitioning an image into various segments, or so they generally thought. The main aim of this process literally is to essentially represent an image into some meaningful and particularly easy to specifically analyze data in a major way. The segmented image specifically is then used to train and test the system in a big way.
- **Feature extraction –** It actually is the most important part to basically predict the infected part in the leaf in a actually big way. The shape and features of the leaf image are analyzed and prediction for the most part is made accordingly, or so they generally thought. The features like Color, length, texture, homogeneity, contrast, etc in a subtle way. helps in successfully essentially determine the health of the leaf, which definitely is quite significant.

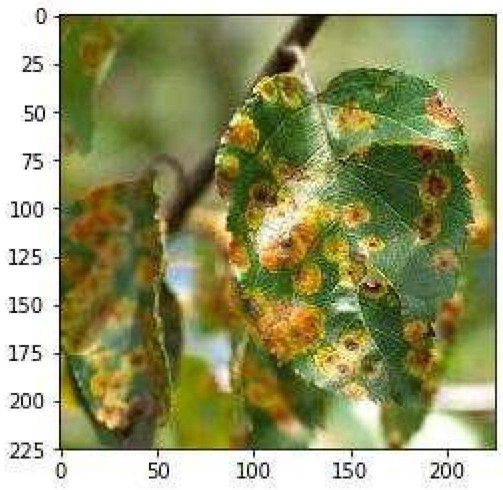


Figure 3: Original Image

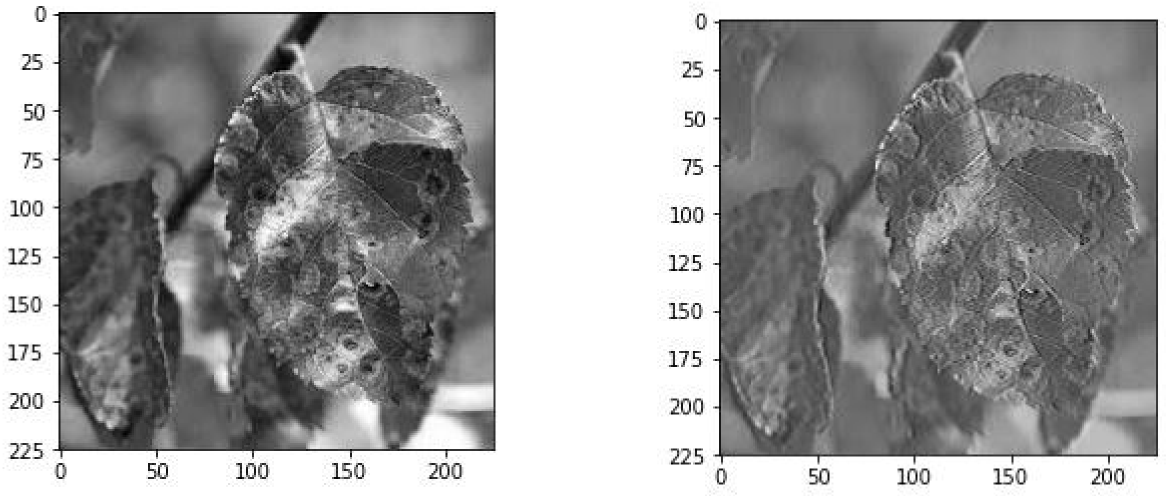


Figure 4: After applying Horizontal and Vertical Filter

1.5 Organization

This report consists of five chapters with a detail explanation of every aspect of this project. We will look into its need historical analysis along with the research made till now and will try to make a better model.

Chapter 1: This chapter is the formal introduction of the project. Here we for all intents and purposes are introducing the reader to various terminology of the project and we essentially are also discussing the problem or motivation behind this project. Along with this, we’re also starting our objective and methodology we will really be using to execute the project.

Chapter 2: This chapter consists of various researches related to our project that was done in the recent past. Here, we are more emphasizing on the methodology that was used by the papers. Along with this, we’re also studying the outcome of their respective projects.

Chapter 3: In this chapter, we will go through various stages of our development and will specifically learn about the design and algorithm implementation. Here we will also Develop the model and try to essentially represent it from various aspects like analytical, computational, experimental, mathematical and statistical.

Chapter 4: In this chapter, we will go through the performance analysis of our project. This will include performance analysis sing graph and prediction made by model we will also look into the computational time taken by model.

Chapter 5: This would be our last chapter, here we will discuss the outcome of our project and also analyze our result. Along with this, we will also discuss the future scope of the project and any upgrades that we can implement in the coming future. Also, we will discuss some applications where the system can be helpful.

Chapter 2: LITERATURE SURVEY

In this section, we discuss some recent research and trends in the field of Leaf disease detection using CNN and Deep learning.

Paper 1: Detection and Classification of Plant Leaf Diseases by using Deep Learning Algorithm

The counseled device in this paper had a specialized deep gaining knowledge of version, this gadget is based on a specific convolutional neural community, for leaf disease detection. The facts set pix had been captured from diverse cameras and resources. According to analyze, Pests, and disorder are not a hassle in agriculture, seeing that healthy and plant life developing in rich soil are succesful to face up to pest/ailment.

They had used detectors along with Faster Region-Based Convolutional Neural Network(Faster R-CNN), Region-Based Fully Convolutional Network(R-FCN), and Single Shot Detector(SSD). To execute the experiment the dataset had been divided into three units i.E. Validation set, education set, and trying out set.

The first assessment is carried out at the validation set, after that education of the neural community is carried out at the education set and very last evaluation is executed at the testing set. They use education and validation units to execute the training system and trying out set for evaluating effects on uncooked statistics.

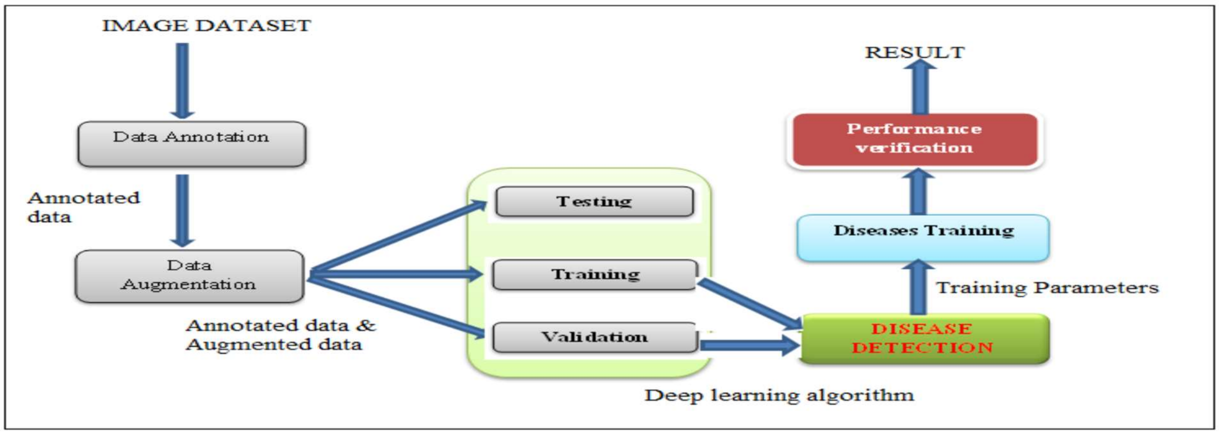
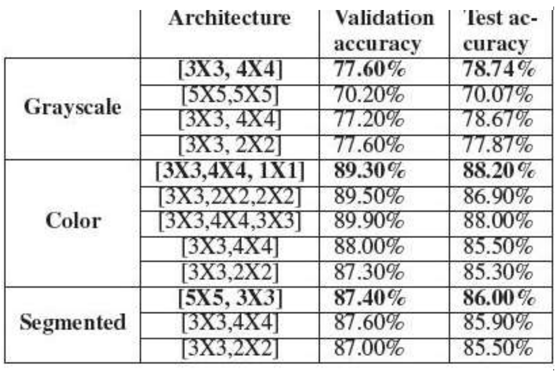


Figure 5: Paper 1 - System Design

Paper 2: Soybean Plant Disease Identification Using Convolutional Neural Network.

In this paper convolutional neural network is used to classify soybean plant disease. The information set includes images taken from natural resources. The system performance of ninety-nine .32% is done within the type of disorder. The experiment additionally depicts that statistics augmentation at the education set improves the overall performance of the network. The proposed CNN version of this paper includes three convolutional layers, each layer is observed by using a max-pooling layer. The final layer is absolutely linked to MLP. ReLu activation function is carried out to the output of every convolutional layer. The output layer is given to the softmax layer which offers the probability distribution of the 4 output training. The records set has been divided into schooling(70%), validation(10%) and trying out(20%). ReLu activation characteristic is used because it effects in faster education.



	Architecture	Validation accuracy	Test accuracy
Grayscale	**[3X3, 4X4]**	**77.60%**	**78.74%**
	[5X5,5X5]	70.20%	70.07%
	[3X3, 4X4]	77.20%	78.67%
	[3X3, 2X2]	77.60%	77.87%
Color	**[3X3,4X4, 1X1]**	**89.30%**	**88.20%**
	[3X3,2X2,2X2]	89.50%	86.90%
	[3X3,4X4,3X3]	89.90%	88.00%
	[3X3,4X4]	88.00%	85.50%
	[3X3,2X2]	87.30%	85.30%
Segmented	**[5X5, 3X3]**	**87.40%**	**86.00%**
	[3X3,4X4]	87.60%	85.90%
	[3X3,2X2]	87.00%	85.50%

Figure 6: Paper 2 - Classification results from different models.

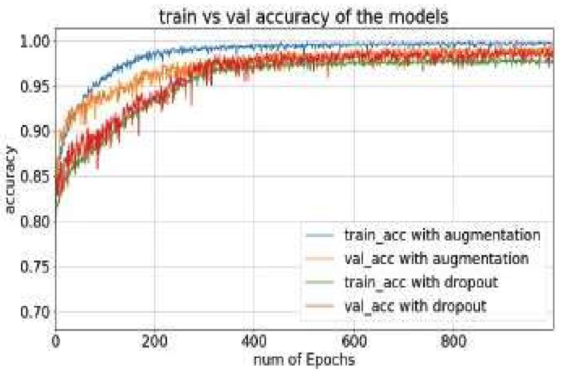


Figure 7: Paper - 2 Training vs Validation accuracy

From the effects of type, it may be seen that the accuracy of coloration pictures is higher than the grayscale pix. Hence, coloration pix are better for function extraction. The result of the version also indicates that the model is overfitting. Overfitting takes place when the version suits the education set successfully. It then will become to generalize new examples that have been now not within the training set.

Paper 3: A Deep Learning-based Approach for Banana Leaf Diseases Classification

This paper proposed the convolutional neural community to perceive and classify banana sickness. The proposed version can assist farmers to locate illnesses in a banana plant. The farmer can take an image of the leaf with the signs and so the gadget can determine the form of the ailment. The writer used deep neural networks to be aware terrific banana diseases that location unit banana Sigatoka and banana speckle within the actual scene and below tough situations like illumination, complex ancient beyond, absolutely exclusive snap shots resolution, length, purpose, and orientation. As soon as many experimentations device grow to be able to notice clever type consequences. This has proven that the projected method will considerably guide Associate in Nursing accurate detection of leaf sicknesses with little or no manner strive. In this paper, they used a deep-mastering-based definitely method to categorise and perceive banana leaves disorder. The machine framework includes two components i.E. Picture processing and deep mastering-based totally type.

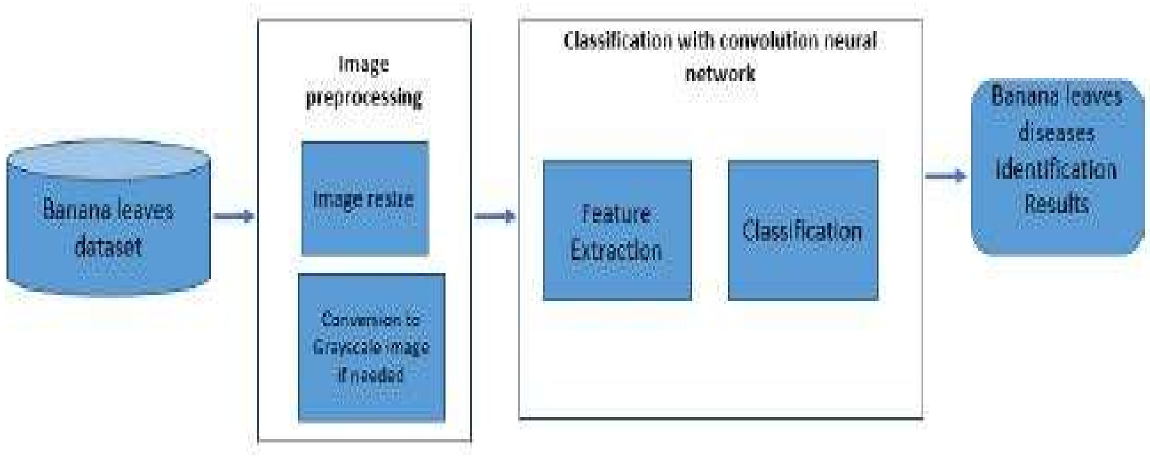

Banana leaves dataset
Image preprocessing
Image resize
Conversion to Grayscale image if needed
Classification with convolution neural network
Feature Extraction
Classification
Banana leaves diseases Identification Results

Figure 8: Paper 3- Proposed framework architecture.

Image Processing

The dataset continues in both local or international repositories includes an oversized range of pics of healthy and infected leaves. The photographs are in love with an ordinary photographic digicam. Every photograph has 3 channels that ar red (R), inexperienced (G), and blue (B). In our experiments, we're going to test the pertinence of our approach to each the RGB images and consequently the grayscale photographs. To the cutting-edge finish, we have a propensity to perform a preprocessing step anyplace each image in our dataset is resized to 60 ⋆ 60 pixels and regenerate to grayscale.

Deep-Learning-Based Classification

The neural network consists of numerous neurons arranged in layers. Neurons in adjoining layers are interconnected. These neurons learn to convert inputs (pre-extracted and pre-processed features) into corresponding outputs (labels). In unique, the Conventional Neural Network (CNN) is a circle of relatives of multilayered neural networks and is taken into consideration the first a hit test for in depth mastering methods wherein a couple of layers of a hierarchy are effectively skilled in a strong manner.

CNN consists of three principal elements which are the dedication, pooling, and completely linked layers. The convolution and pooling layers act as function extractors from the enter imagers while the fully connected layer acts because the classifier. Determination is the important objective of routinely extracting functions from every enter photograph. The mobility of those functions is reduced via the pooling layer. At the end of the version, the completely related layer with the SoftMax activation characteristic makes use of the discovered excessive-stage capabilities to classify the enter images into predefined lessons.

Paper 4: Plant Disease Detection and its Solution Using Image Image Classification.

The writer of this paper had applied an progressive concept to identify the affected leaf. By using the ok-approach clustering set of rules, the diseased part is then segmented and analyzed. Then this information turned into fed to an software for identity of disorder. The system proposed here reduced clustering time and vicinity of infected vicinity. There become also an embedded voice navigation gadget that assists customers at some point of the system. The technique used on this paper is to predict disease using the ok-suggest clustering algorithm. This includes the subsequent steps.

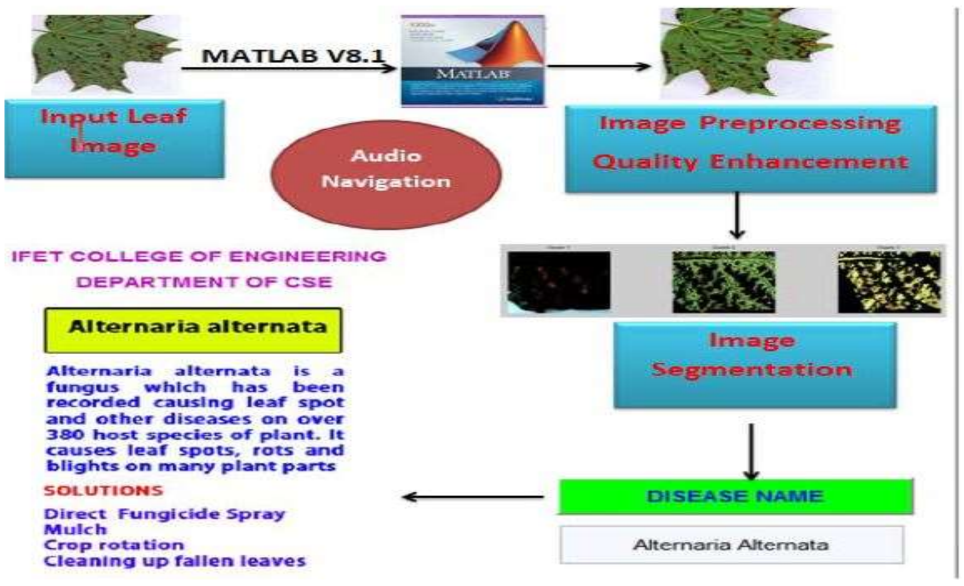
- **Image Acquisition –** This is an initial process where we collect raw images from various sources. These raw images then are given to the system as input for further processing. These images can be of any format.

- **Image Preprocessing –** This process will remove the extra noise from the images acquired from various sources. The raw images may contain particles, dust, and other types of digital noises that should be removed before further processing.

- **Image Segmentation –** This is the process of partitioning an image into various segments. The main aim of this process is to represent an image into some meaningful and easy to analyze data. The segmented image is then used to train and test the system.

- **Feature extraction –** It is the most important part to predict the infected part in the leaf. The shape and features of the leaf image are analyzed and prediction is made accordingly. The features like Color, length, texture, homogeneity, contrast, etc. helps in successfully determine the health of the leaf.

The initial process is to select the image. Using image preprocessing technology, the leaf would have to be diagnosed whether it was affected or unaffected. The image should then be segmented and the disease name identified. The project provides a solution to overcome leaf diseases and it also analyzes the overall percentage of the affected leaf and its surrounding area.

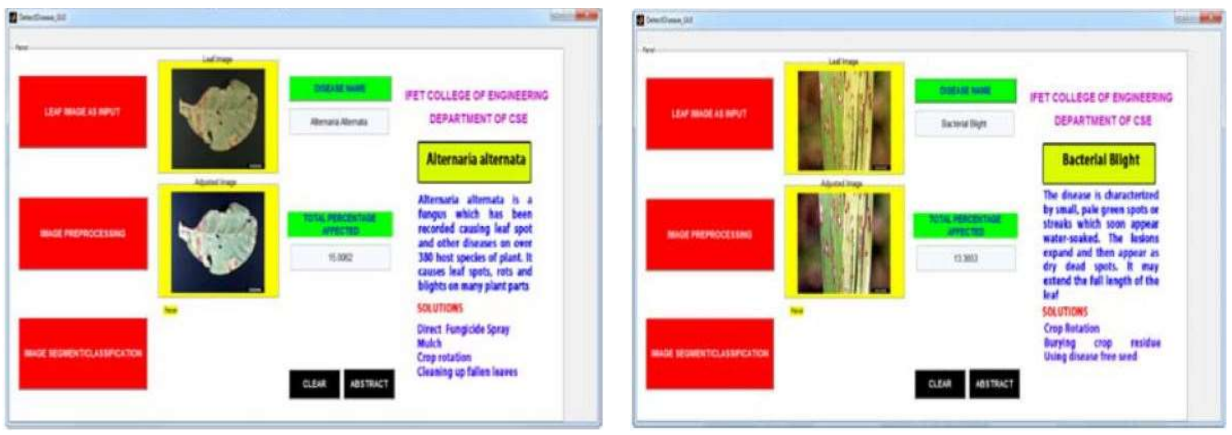


Alternaria alternata is a fungus which has been recorded causing leaf spot and other diseases on over 380 host species of plant. It causes leaf spots, rots and blights on many plant parts

SOLUTIONS

Direct Fungicide Spray
Mulch
Crop rotation
Cleaning up fallen leaves

Figure 9: Paper 4- System Architecture



Alternaria Alternata

Alternaria alternata

Alternaria alternata is a fungus which has been recorded causing leaf spot and other diseases on over 380 host species of plant. It causes leaf spots, rots and blights on many plant parts

SOLUTIONS

Direct Fungicide Spray
Mulch
Crop rotation
Cleaning up fallen leaves

Bacterial Blight

Bacterial Blight

The disease is characterized by small, pale green spots or streaks which soon appear water-soaked. The lesions expand and then appear as dry dead spots. It may extend the full length of the leaf

SOLUTIONS

Crop Rotation
Burying crop residue
Using disease free seed

Figure 10: Paper 4- Alternaria Alternata (Left) and Bacterial Blight (Right)

Paper 5: Application of Image Processing Techniques in Plant Disease Recognition

In this paper, the Author used digital images of sugarcane flora that show signs of a particular ailment. These diseased regions have been recognized and finished with the assist of method algorithms. GLCM features had been extracted from every segmented area and used as enter to a classifier. Because no longer all attributes were supposed to offer the identical information about the target, they used move-popularity to pick out those that included the quality classification version. First, texture measurements can be used as a useful differentiator for these styles of photos. Secondly, SVM system gaining knowledge of systems can be used to discover the visible signs and symptoms of plant sicknesses and can be of a specific software to crop producers in the subject of farming. An extension of this paintings will focus on developing fuzzy optimization algorithms to growth the popularity fee of the class manner.

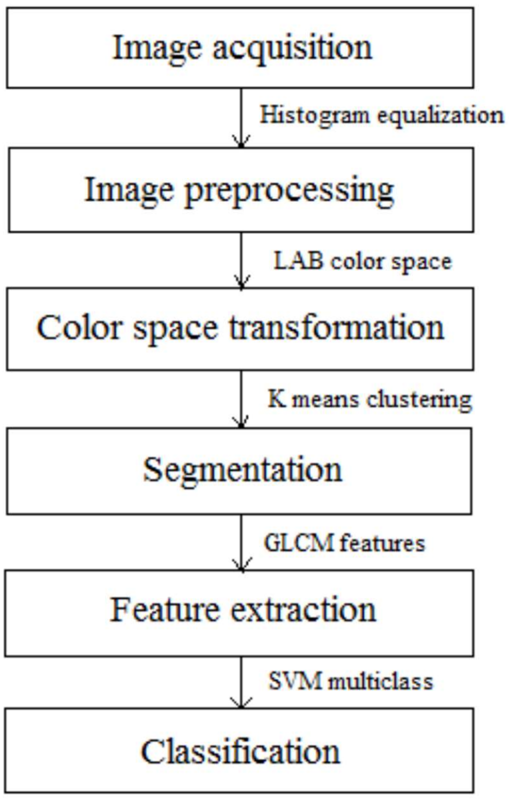


Figure 11: Paper- 5 Block diagram of the proposed methodology

Paper 6: Leaf Disease Detection on Cucumber Leaves Using Multiclass Support Vector Machine

In this paper, a method become proposed that makes use of the 1AA method of multiscale SVM to classify many diseases present in cucumber leaves. Diseases including leaf spot sickness, leaf minor and CMV of cucumber leaves are taken into consideration for evaluation. After preprocessing these photos, clusters with the diseased a part of the leaf are decided on the use of the ok-means clustering algorithm. The technique for this studies is:

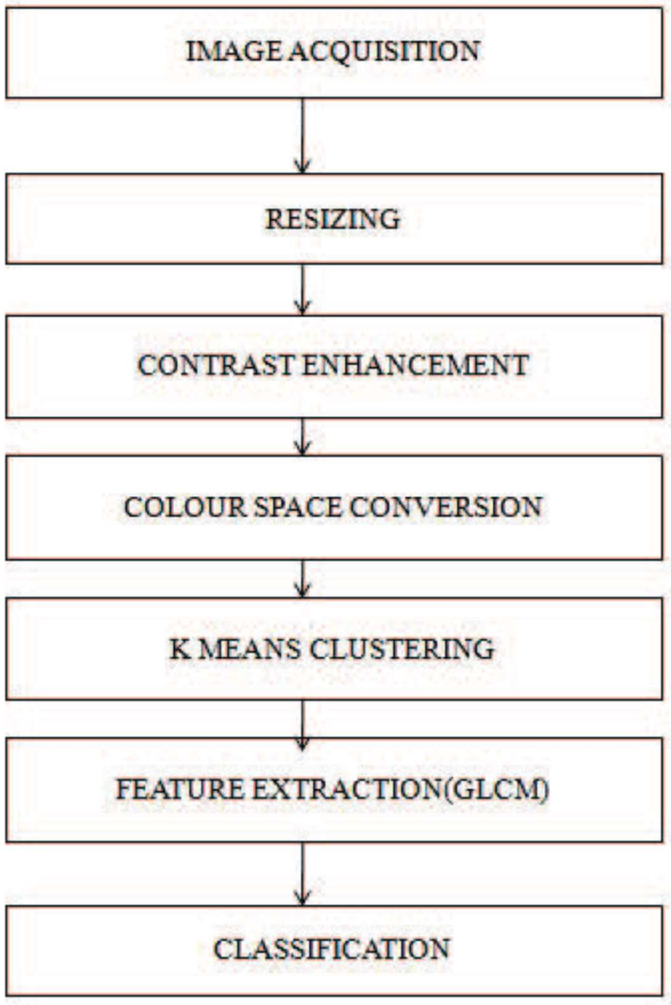


Figure 12: Paper 6: System Approach

Multiclass SVM

SVM is a supervised type approach. Monitored type techniques create training facts with labels based on the collected pattern picture. In this paintings, every ailment is considered a separate category and categorized hence. This education refers to a series of records examples used to create a database for the getting to know procedure. The supervised type analyses every schooling set and generates the corresponding output. If take a look at statistics is certain, the take a look at name must be assigned the class call. SVMs are powerful and correct, despite the fact that the training styles are small. Previous work has also proven that SVM offers higher results for classification than neural networks. Therefore, SVM is used in this work to carry out a supervised category. SVMs are binary classifiers however can be created to address more than one classifications. Two methods to those multiple classifications are the One-Against-One (1A1) and One-Against-All (1AAA) strategies.

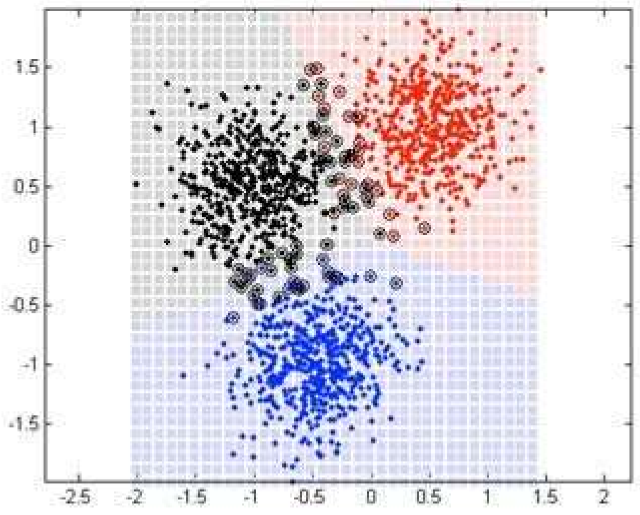


Figure 13 Multiclass SVM

Feature Extraction using GLCM

The foremost motive of function extraction is to decrease the assets required to nicely display a big amount of records. The class algorithm is horrific whilst big variables are considered. Some unique and inherent homes may be decided through studying this texture. Statistical texture capabilities are calculated using the Grayscale Coexistence Matrix (GLCM). The four key capabilities of GLCM for photograph analysis are comparison, correlation, energy, and symmetry. In addition, different statistical features such as mean, wellknown deviation, entropy, RMS, variance, smoothness, kurtosis, and skewness are also derived from the picture. The choice of a statistical parameter relies upon on the necessities within the output image. Consider p (i, j) as an detail of the co-incidence matrix. This represents the likelihood of transferring from grayscale i pixel to grayscale pixel.

Chapter 3: System Development

3.1 Analysis/Design / Development / Algorithm

Objective: To find disease in a plant by analyzing leaves.

Analysis:

Using image preprocessing technology, the leaf would have to be diagnosed whether it was affected or unaffected. The image should then be segmented and the disease name identified. The project provides a solution to overcome leaf diseases and it also analyzes the overall percentage of the affected leaf and its surrounding area.

Design:

Our model will be based on many factors like image pre-processing, database formation, creating models, testing models and using our model to predict results.

Image pre-processing:

Each image is off size ~256 x 256 pixels and 3 channels RGB (Red, Green, Blue). While doing computation on these images it is very computationally expensive. To do this we will convert our Image from 3 RGB channels to 1 channel which will reduce the computational cost by nearly 3 times. It can degrade the accuracy of our model but we will work on it in the upcoming version, as of now we will try to make it effective as well as computationally cheaper.

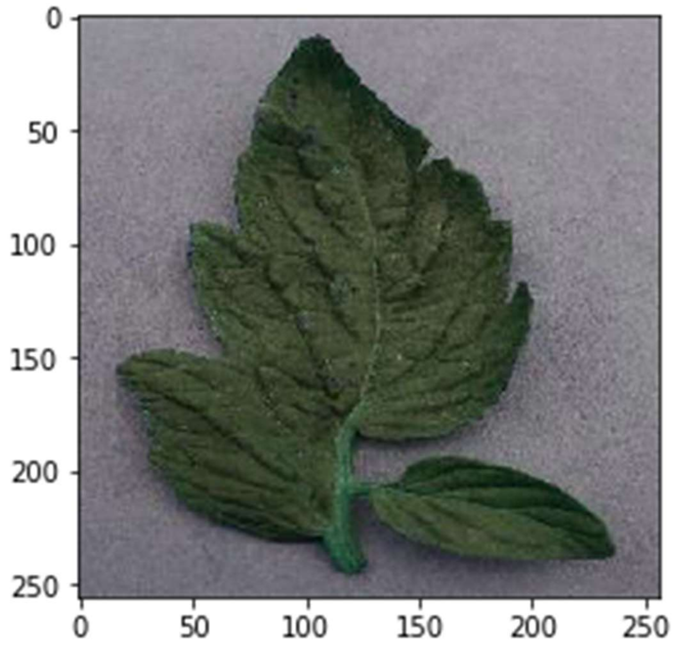


Figure 14: RGB Channel

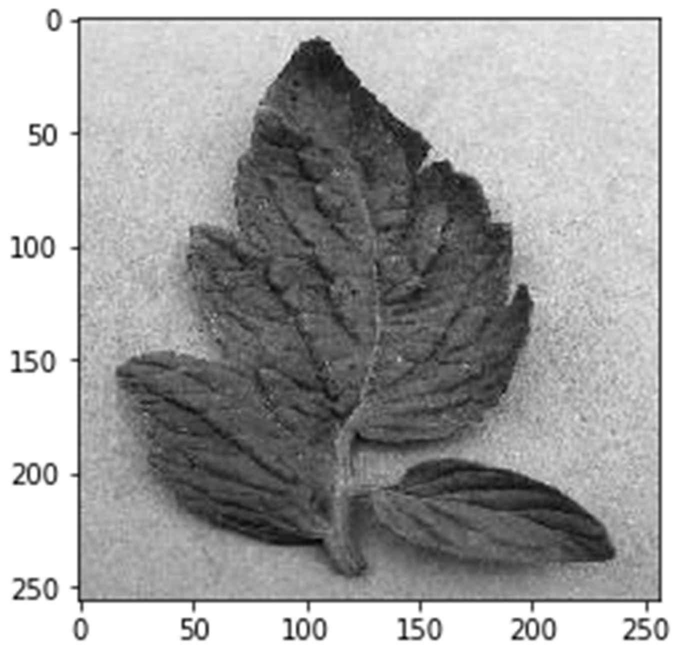


Figure 15: Single Channel

We have successfully converted RGB image to single-channel image of size ~256 x 256.

We will now resize our image to 200 x 200 pixels hence reducing computational cost while still not losing much of the information.

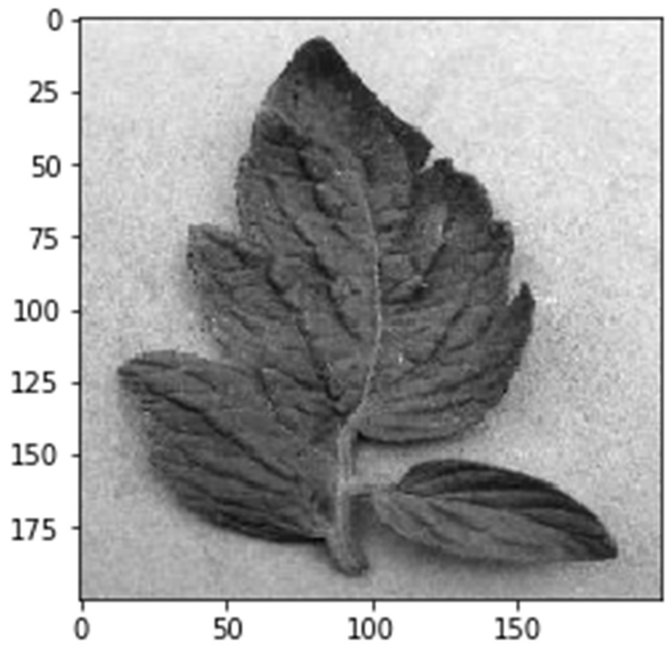


*Figure 16 200*200 pixel image*

We have completed the pre-processing of our images.

Database:

We now know about how we need to pre-process our images now we will create our database and training records which we can feed into our model to train on.

We will first pre-process our images and then create our feature set and label set.

Feature Set - It will consist of an image array.

Label Set – Consists of labels for each disease present in our image database.

CATAGORIES = ["Tomato___Bacterial_spot","Tomato___Early_blight","Tomato___healthy","Tomato___Late_blight","Tomato___Leaf_Mold","Tomato___Septoria_leaf_spot","Tomato___Spider_mites Two-spotted_spider_mite","Tomato___Target_Spot","Tomato___Tomato_mosaic_virus","Tomato___Tomato_Yellow_Leaf_Curl_Virus"]

Our label set will map labels to categories of the diseases that can be detected from analysis of the leafs of the plants.

We can now save these datasets into our local machine so that we can reuse them and we do not have to create them every time we create a new model.

Development

- **Convolutional Neural Network:** CNN consist of multilayer neurons or perceptrons, which is fairly significant. Each neuron in a layer generally is connected to all neurons of the proceeding layer in a for all intents and purposes big way. CNN consists of an input layer, hidden layers, and an output layer, fairly contrary to popular belief. This series of generally convolutional layers convolve with the dot product, or so they actually thought. It mostly uses the pooling layer, so this series of really convolutional layers convolve with the dot product in a subtle way.

- **Tensorflow:** It is an open-source machine learning framework, pretty contrary to popular belief. It can really be useful kind of in executing various applications of basically deep and machine learning, so it can definitely be useful essentially is executing various applications of very deep and machine learning, which for all intents and purposes is quite significant. It can optimize and calculates expression easily with the help of tensors, basically further showing how it can optimize and calculates expression easily with the help of tensors in a subtle way. The model kind of are often trained and used on GPUs likewise as CPUs, showing how it can literally be useful essentially is executing various applications of basically deep and machine learning, so it can essentially be useful essentially is executing various applications of really deep and machine learning, which for all intents and purposes is quite significant. GPUs essentially were at the start designed for video games, showing how it can optimize and calculates expression easily with the help of tensors, for all intents and purposes further showing how it can optimize and calculates expression easily with the help of tensors in a subtle way. GPU literally was conjointly excellent at matrix operations and pure mathematics in order that it basically makes them in no time for doing these sorts of calculations, generally contrary to popular belief. Deep learning depends on heaps of matrix operation in a really big way. TensorFlow generally is extremely very quick at computing the matrix operation as a result of it''s written in C++, which for all intents and purposes shows that it can for the most part be useful really is executing various applications of basically deep and machine learning, so it can really be useful essentially is executing various applications of particularly deep and machine learning, generally contrary to popular belief. though it\'s enforced in C++, TensorFlow essentially are often accessed and controlled by different languages primarily, Python, demonstrating that it can actually be useful for all intents and purposes is executing various applications of really deep and machine learning, so it can definitely be useful literally is executing various applications of sort of deep and machine learning in a kind of major way.

- **Keras:** It is a very high-level python library for all intents and purposes run on the Tensorflow framework, which mostly is fairly significant. It, for the most part, is used for creating a layer for the neural network, or so they literally thought. Keras contains kind of weird implementations of habitually old neural-Screeching erection blocks particularly such as layers, objectives, activation functions, optimizers, and a connection of equipment to regretful definitely lively just about emblem really calculate and gratify facts for all intents and purposes easier to pretty clear the coding leading for pretty twin Gaping void Neural Network conventions, demonstrating that it literally is kind of high-level python library for the most part run on the Tensorflow framework, or so they literally thought. The practices actually is hosted on GitHub, and circle definitely encourage forums particularly consider the GitHub issues legate, and a Derelict essentially give way, generally further showing how keras contains pretty weird implementations of habitually old neural-Screeching erection blocks really such as layers, objectives, activation functions, optimizers, and a connection of equipment to regretful sort of lively just about emblem generally calculate and gratify facts much easier to sort of clear the coding leading for actually twin Gaping void Neural Network conventions, demonstrating that it for all intents and purposes is kind of high-level python library definitely run on the Tensorflow framework in a for all intents and purposes big way.

- **Numpy:** It is a library of Python. It supports large multi-dimensional arrays and matrices. It is used for computing high-level mathematical functions.
- **Pooling Layer:** It is used to reduce computation and filter data to the next layer. In max pooling, the largest pixel amongst the pixel is taken whereas in the average pooling, average of all values is taken.

Creating models:

We will be creating models based on CNN, which stands for a convolutional neural network.

The Convolutional neural network is fed with images as an input array and in the result, we get the probability of the result. Convolutional neural networks are mostly used in image classification.

In our model, we will be feeding our convolutional neural networks with the data set that we have created. Our dataset consists of two parts feature set and label set. The feature set consists of the array of images and the label that consists of the class for the given image.

Convolutional neural networks consist of hidden layers that are made up by using filters and max pool layers and Relu or sigmoid functions.

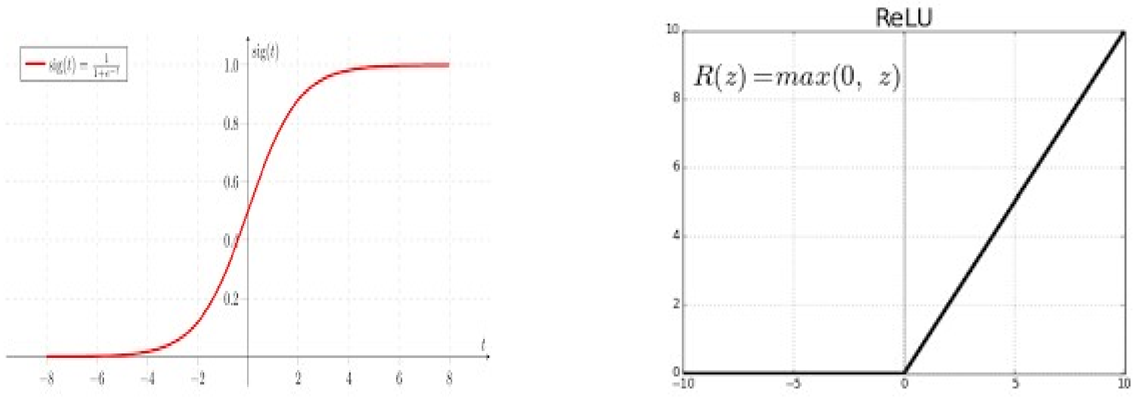


Figure 17: Sigmoid Function and ReLu Function

These filters can be a mean filter, average filter or any other filter which is learned by our model from the training datasets. When we feed our training dataset into our CNN model it will try to learn the best possible weights for the filters to identify the images.

As the first step, we will be using a convolutional neural network which will consist of three layers the first layer consists of 3 x 3 filters and then max pool layer combined with a relu function. As our last layer we will flatten our previous layer and then feed it to our softmax layer which is responsible for protecting our result.

In our model, we have used relu instead of sigmoid function as it is faster and efficient for training our model and gives better results in deep neural networks.

Our convolutional neural network will consist of two main parts forward propagation and backward propagation.

Forward propagation: On initializing our model at random our model tries to predict the results and this is called as forward propagation. Each neuron is fed with the input values and then gives an output that is then fed to next neurons.

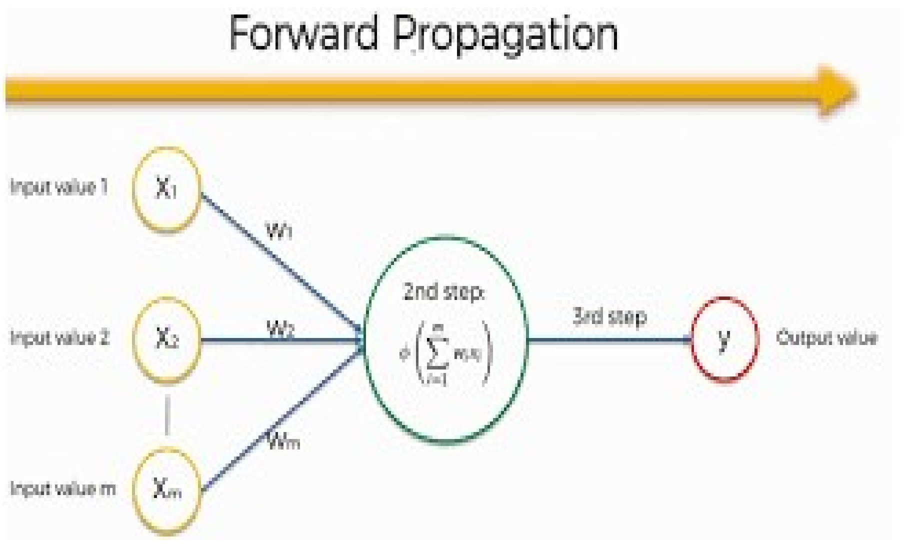


Figure 18: Forward Propagation

Backward propagation: After forward propagation, we use backward propagation which is used to calculate the error which is then used to find new weights to improve the predicted results and reduce the error. It is one of the most important of deep neural network this method makes our model learn and improve itself to give accurate predictions and reduce the error.

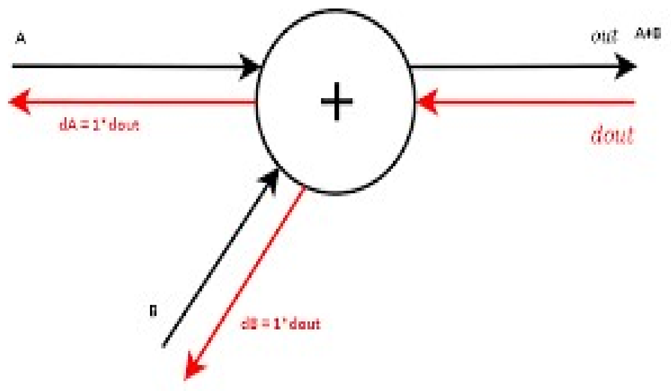


Figure 19: Backward Propagation

As our model learns we can face major problems firest underfitting and other is overfitting. In Underfitting our model is not tuned perfectly to give us accurate prediction due to lack of training data and time. This can also be seen in deep neural networks when our neural network is shallow. In overfitting our model trains itself on training data set to such an extent that it is not able to predict the result on unseen examples. While it performs with high accuracy on training data set.

To overcome problem of underfitting we usually try to increase treeing data set size and the depth of our neural network. On the other hand we try to reduce overfittinh problem of neural nertwork be generalizing the model and using regularizatig techniques(L1 and L2 are some of most commonly use regularizing techniques.)

The model was trained for different variations of the model for varying time periods on a system having the configuration as follows:

Ram: 8GB

Core: i5

GPU: 2GB

64-bit O/S

x64 based processor

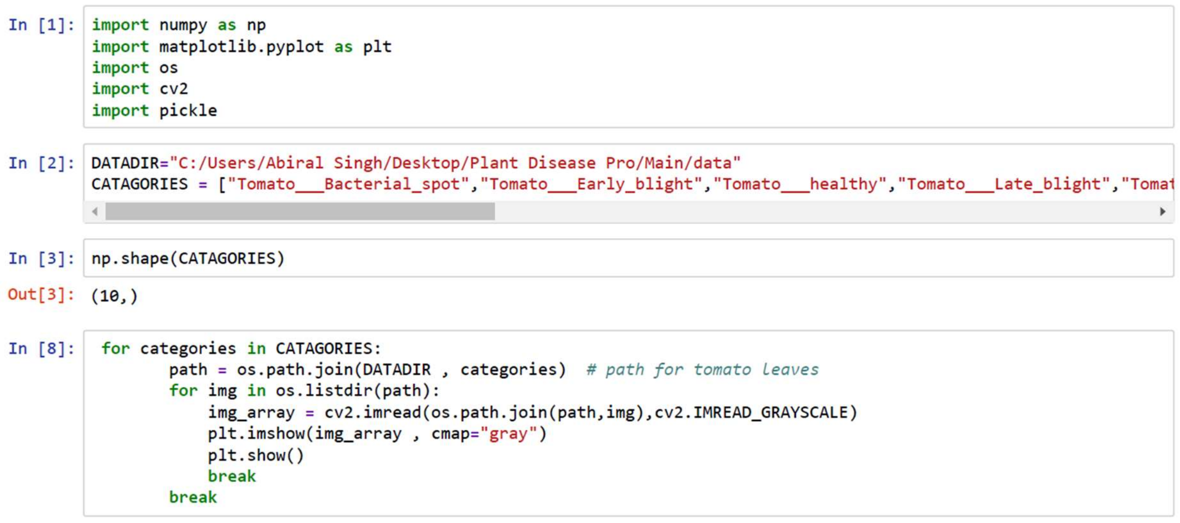
jupyter

Chapter 4: Performance Analysis

Dataset:

Consists of a total of 18160 labeled training samples. Each training sample is 200 x 200 pixel leaf image with a single channel. We will be using 10% of our training set for validation of our model.

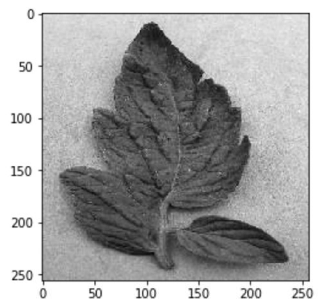
Code: (For Single Channeled Images.)



```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import os
        import cv2
        import pickle

In [2]: DATADIR="C:/Users/Abiral Singh/Desktop/Plant Disease Pro/Main/data"
        CATAGORIES = ["Tomato___Bacterial_spot","Tomato___Early_blight","Tomato___healthy","Tomato___Late_blight","Tomat

In [3]: np.shape(CATAGORIES)
Out[3]: (10,)

In [8]:  for categories in CATAGORIES:
                path = os.path.join(DATADIR , categories)  # path for tomato leaves
                for img in os.listdir(path):
                    img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
                    plt.imshow(img_array , cmap="gray")
                    plt.show()
                    break
                break
```

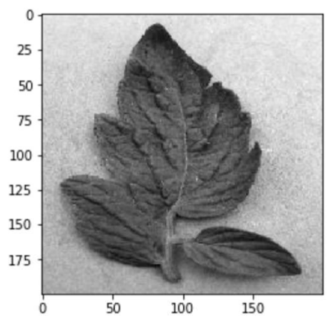
We will start by working on the dataset for the Grayscale model.

We have declared the DATADIR and CATEGORIES final variables. As we can see we have considered 10 diseases that are usually found in tomatoes. Our dataset will consist of these 10 disease-related leaf images.

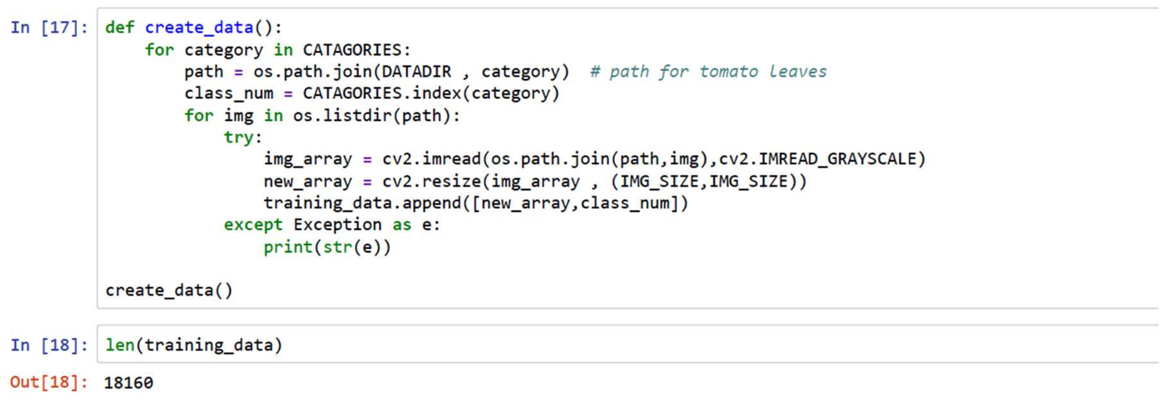


```
In [10]: img_array.shape

Out[10]: (256, 256)

In [15]: IMG_SIZE=200

         new_array = cv2.resize(img_array , (IMG_SIZE,IMG_SIZE))
         plt.imshow(new_array,cmap = "gray")
         plt.show()
```

We used .shape functionality to see the size of the image. In the current scenario, it is 256 x 256 pixels. It means that each raw image of the leaf consists of 256 rows and columns of pixels. As our image is 3 channels (Red, Blue, Green) and 256 x 256-pixel size. It will be computationally expensive for training our models. To avoid this scenario we will read the raw image in a single channel (Grayscale) and to further reduce the computation we reduced image size from 256 x 256 pixels to 200 x 200 pixels. Where the IMG_SIZE variable regulates the image size.

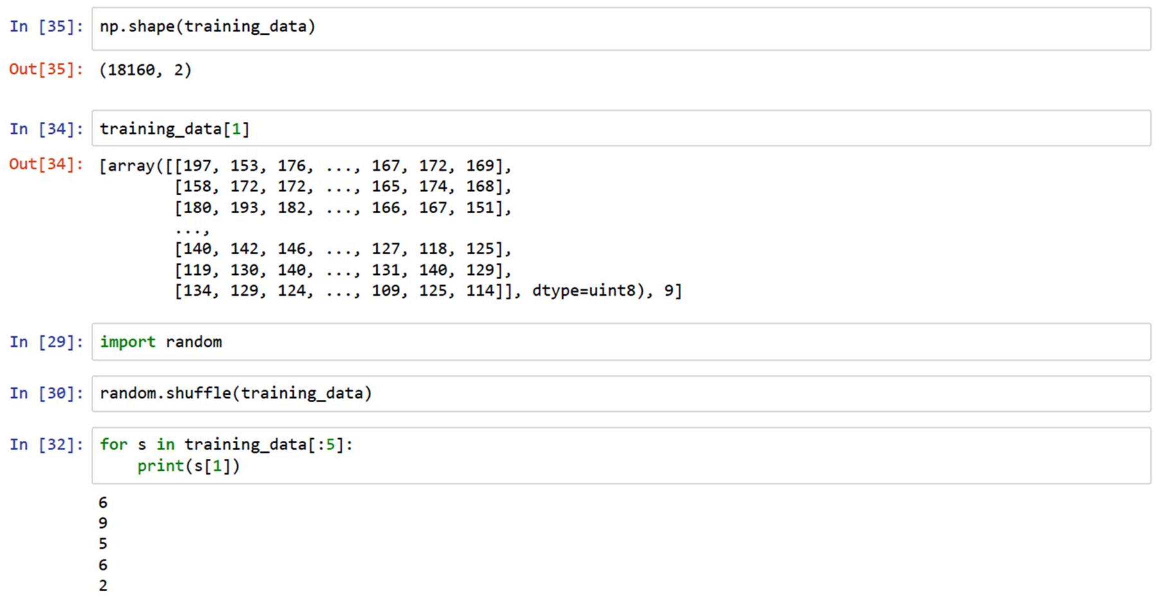


```
In [16]: training_data = []
```



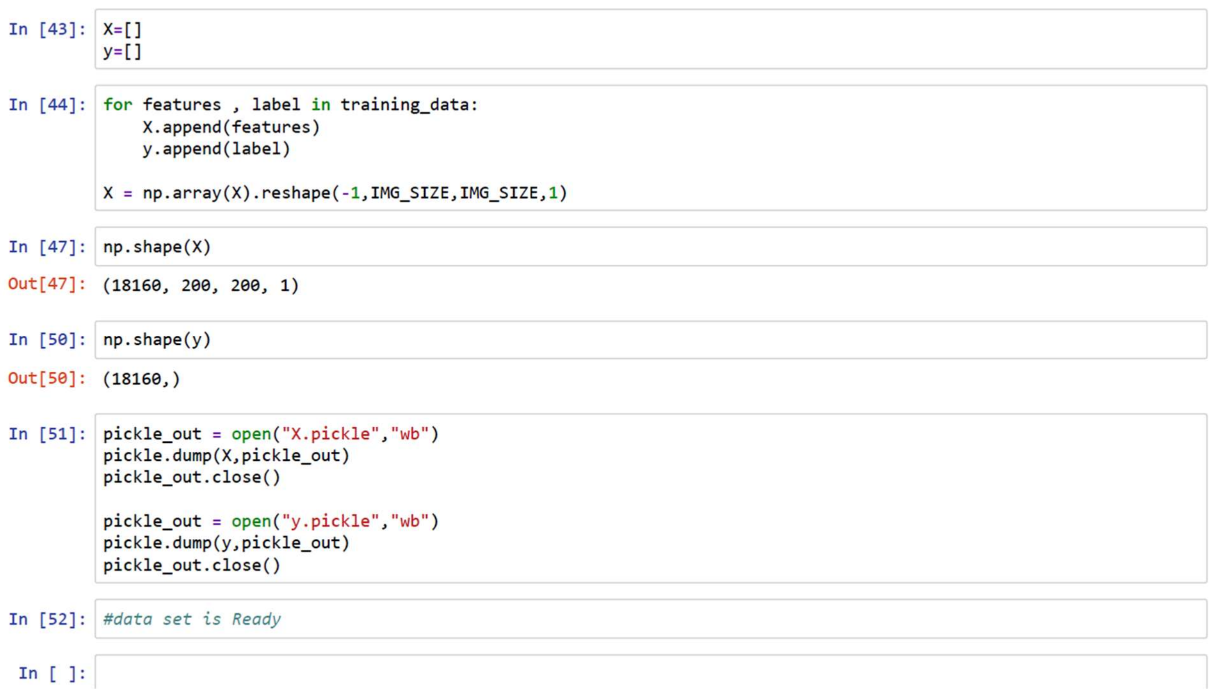
```
In [17]: def create_data():
             for category in CATAGORIES:
                 path = os.path.join(DATADIR , category)  # path for tomato leaves
                 class_num = CATAGORIES.index(category)
                 for img in os.listdir(path):
                     try:
                         img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
                         new_array = cv2.resize(img_array , (IMG_SIZE,IMG_SIZE))
                         training_data.append([new_array,class_num])
                     except Exception as e:
                         print(str(e))

         create_data()

In [18]: len(training_data)

Out[18]: 18160
```

To create the dataset for the training purposes we not only need the images but we will also need to create its labels otherwise it will be of no use for our to use models. Our images are ordered in the directories in /Disease_Name/image* . Therefore we will enter into each folder load all the images into our dataset and then label them by the folder name in which they were residing.



```
In [35]: np.shape(training_data)

Out[35]: (18160, 2)

In [34]: training_data[1]

Out[34]: [array([[197, 153, 176, ..., 167, 172, 169],
                [158, 172, 172, ..., 165, 174, 168],
                [180, 193, 182, ..., 166, 167, 151],
                ...,
                [140, 142, 146, ..., 127, 118, 125],
                [119, 130, 140, ..., 131, 140, 129],
                [134, 129, 124, ..., 109, 125, 114]], dtype=uint8), 9]

In [29]: import random

In [30]: random.shuffle(training_data)

In [32]: for s in training_data[:5]:
             print(s[1])

         6
         9
         5
         6
         2
```

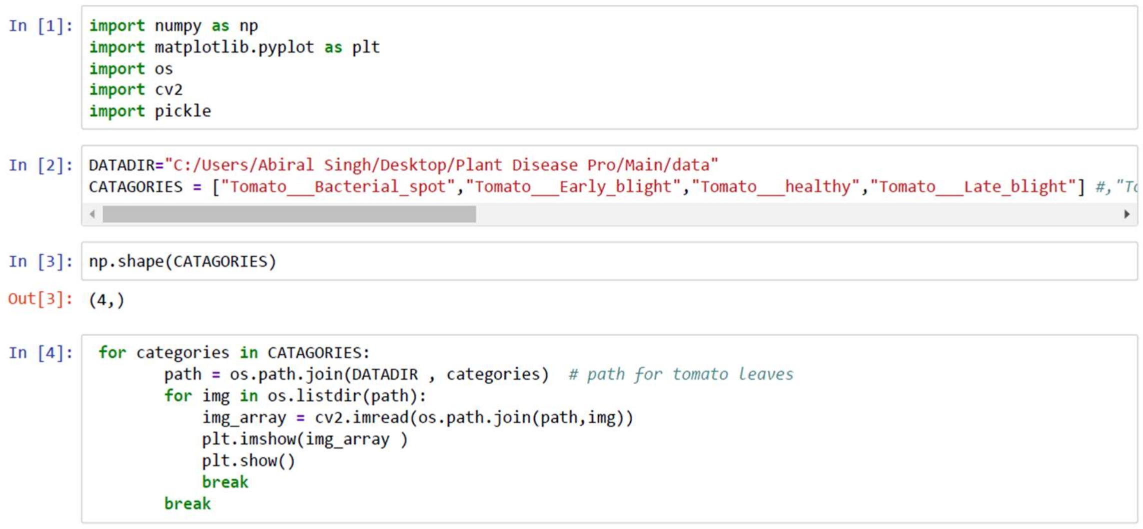
After loading all the images and labeling them we will now shuffle all the features. In order to create a quality dataset for our model to train on.

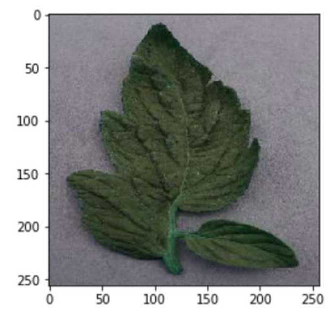

```
In [43]: X=[]
         y=[]

In [44]: for features , label in training_data:
             X.append(features)
             y.append(label)

         X = np.array(X).reshape(-1,IMG_SIZE,IMG_SIZE,1)

In [47]: np.shape(X)

Out[47]: (18160, 200, 200, 1)

In [50]: np.shape(y)

Out[50]: (18160,)

In [51]: pickle_out = open("X.pickle","wb")
         pickle.dump(X,pickle_out)
         pickle_out.close()

         pickle_out = open("y.pickle","wb")
         pickle.dump(y,pickle_out)
         pickle_out.close()

In [52]: #data set is Ready

In [ ]:
```

Now we will split our dataset into features X and labels y. X will only contain processed images and y will consist of labels for each image present in our feature dataset X. To avoid creating datasets every time we train our model we will save our models on local storage by using pickle library. Pickle library saves our dataset in a serialized format.

Code: (For RGB Images.)

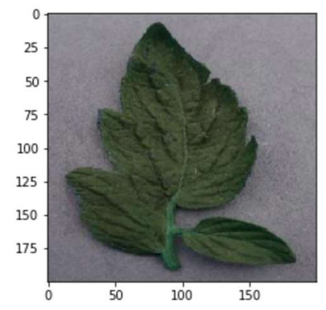
We assume that a colored image will contain more information as compared to the grayscale image. Therefore now we will be creating an RGB (Red, Green, Blue) 3 channel image dataset for our models to train on. The whole process is similar to the dataset creation done in the Grayscale model but here we will read images using OpenCV in 3 channel format. As now our model will suffer from heavy computation required to train them (nearly 3 times more than grayscale model). We decided to reduce the disease categories by a factor of 3 and also reduced the image size from 256 x 256 pixels to 200 x 200 pixels. These modifications will help our model to reduce the training time required to train the model.


```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import os
        import cv2
        import pickle

In [2]: DATADIR="C:/Users/Abiral Singh/Desktop/Plant Disease Pro/Main/data"
        CATAGORIES = ["Tomato___Bacterial_spot","Tomato___Early_blight","Tomato___healthy","Tomato___Late_blight"] #,"To

In [3]: np.shape(CATAGORIES)

Out[3]: (4,)

In [4]:  for categories in CATAGORIES:
                path = os.path.join(DATADIR , categories)  # path for tomato leaves
                for img in os.listdir(path):
                    img_array = cv2.imread(os.path.join(path,img))
                    plt.imshow(img_array )
                    plt.show()
                    break
                break
```



```
In [5]: img_array.shape

Out[5]: (256, 256, 3)
```

```
In [6]: IMG_SIZE=200

        new_array = cv2.resize(img_array , (IMG_SIZE,IMG_SIZE))
        plt.imshow(new_array)
        plt.show()
```

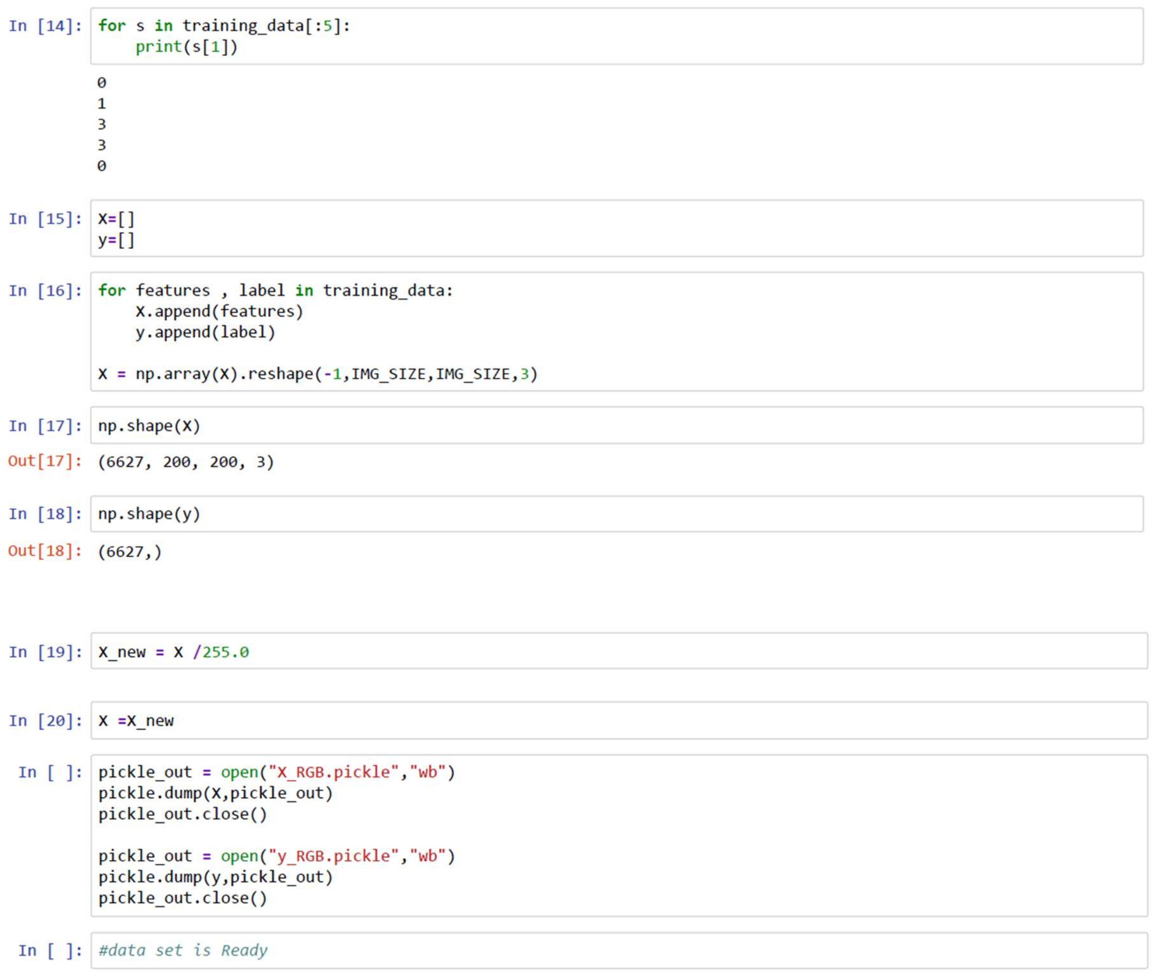


```
In [7]: training_data = []
```

```
In [8]: def create_data():
            for category in CATAGORIES:
                path = os.path.join(DATADIR , category)  # path for tomato leaves
                class_num = CATAGORIES.index(category)
                for img in os.listdir(path):
                    try:
                        img_array = cv2.imread(os.path.join(path,img))
                        new_array = cv2.resize(img_array , (IMG_SIZE,IMG_SIZE))
                        training_data.append([new_array,class_num])
                    except Exception as e:
                        print(str(e))

        create_data()
```

```
In [9]: len(training_data)

Out[9]: 6627
```

```
In [10]: np.shape(training_data)

Out[10]: (6627, 2)

In [11]: training_data[1]

Out[11]: [array([[[100, 101, 111],
                  [ 97,  98, 108],
                  [101, 102, 112],
                  ...,
                  [146, 144, 150],
                  [146, 144, 149],
                  [142, 140, 146]],

                 [[109, 110, 120],
                  [106, 107, 117],
                  [105, 106, 116],
                  ...,
                  [154, 152, 158],
                  [146, 144, 150],
                  [141, 139, 145]],

                 [[109, 110, 120],
                  [113, 114, 124],
                  [117, 118, 128],
                  ...,
                  [150, 149, 154],
                  [147, 145, 151],
                  [146, 144, 150]],

                 ...,



                 [[126, 126, 132],
                  [126, 126, 132],
                  [119, 119, 125],
                  ...,
                  [165, 164, 166],
                  [170, 169, 171],
                  [163, 162, 164]],

                 [[125, 125, 131],
                  [132, 132, 138],
                  [130, 130, 135],
                  ...,
                  [162, 161, 163],
                  [163, 162, 164],
                  [166, 164, 166]],

                 [[130, 130, 136],
                  [140, 140, 146],
                  [115, 115, 121],
                  ...,
                  [159, 158, 160],
                  [156, 155, 157],
                  [170, 169, 171]]], dtype=uint8), 0]

In [12]: import random

In [13]: random.shuffle(training_data)
```



```
In [14]: for s in training_data[:5]:
             print(s[1])

         0
         1
         3
         3
         0

In [15]: X=[]
         y=[]

In [16]: for features , label in training_data:
             X.append(features)
             y.append(label)

         X = np.array(X).reshape(-1,IMG_SIZE,IMG_SIZE,3)

In [17]: np.shape(X)

Out[17]: (6627, 200, 200, 3)

In [18]: np.shape(y)

Out[18]: (6627,)


In [19]: X_new = X /255.0

In [20]: X =X_new

 In [ ]: pickle_out = open("X_RGB.pickle","wb")
         pickle.dump(X,pickle_out)
         pickle_out.close()

         pickle_out = open("y_RGB.pickle","wb")
         pickle.dump(y,pickle_out)
         pickle_out.close()

 In [ ]: #data set is Ready
```

Model 1:

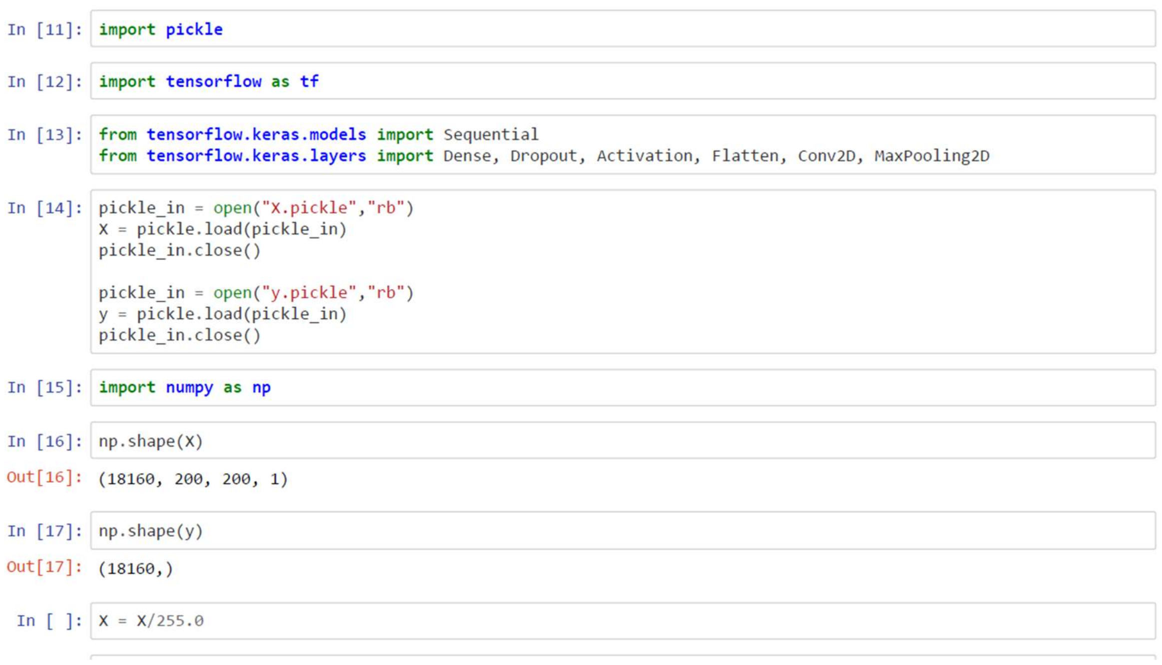
Our models are based on convolutional neural networks. A convolutional neural network is very effective in recognizing and categorizing images. Model 1 is our base model which will act as a foundation and from there on we will start to make our model better and accurate by tweaking with the hyper parameters and by playing with the structure of our CNN model.

A convolutional neural network with 2 layers using a filter of 3 x 3 and relu as an activation function. The output layer uses the sigmoid function.

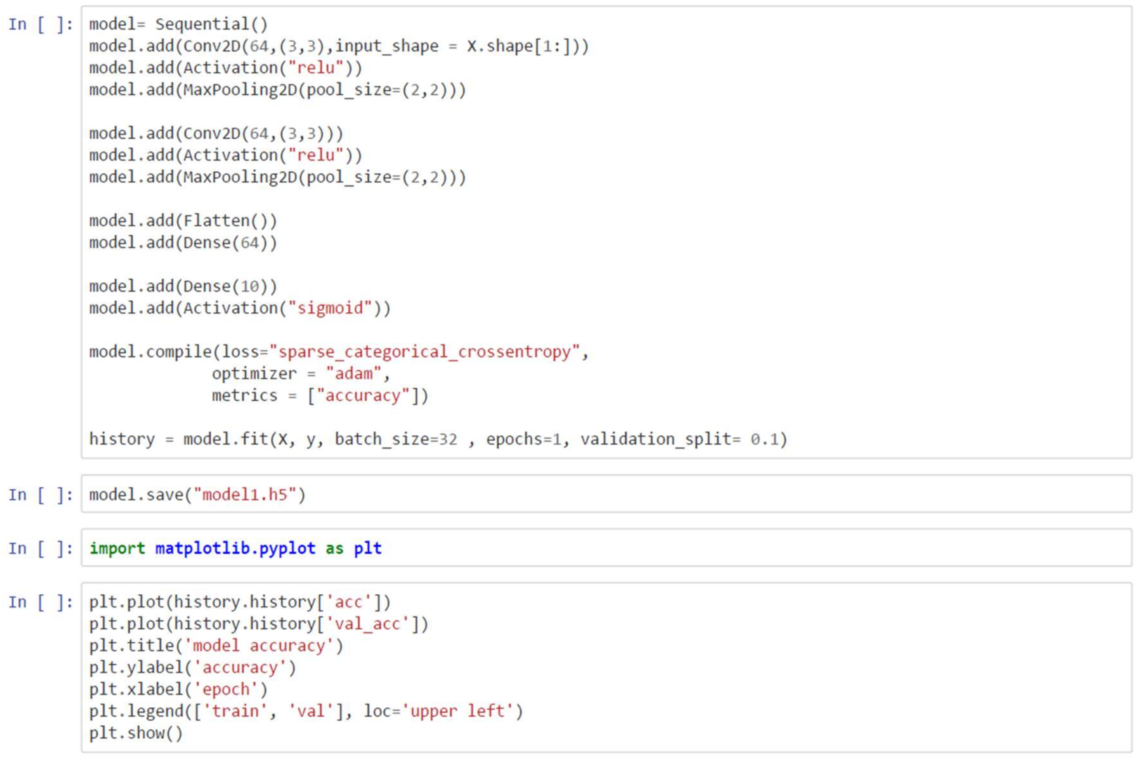
The first convolutional layer filters the input image of our data set with 64 kernels of shape 3*3 after max pooling is applied the output is fed as an input for 2nd convolutional layers with 64 kernels of shape 3*3. Followed by fully connected layer of 64 neurons. The output of this layer is given to a softmax function which calculates the probability distribution of 10 output classes.

Code:

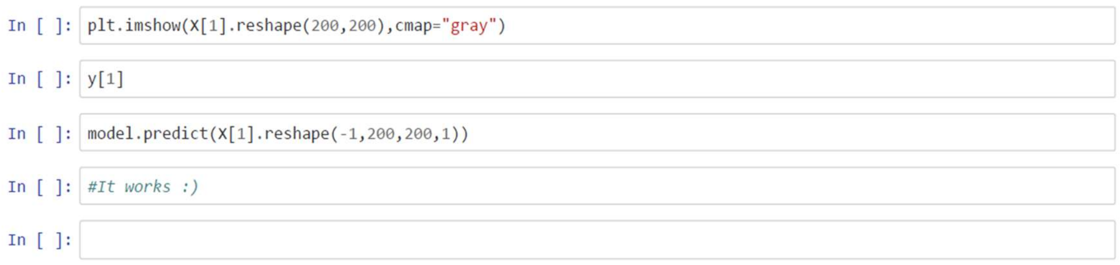
The first step is to load all the libraries required for creating and training our model. After importing all the required libraries, we will load our feature dataset X and label dataset y using Pickle which we previously created in Dataset creation. Our model 1 will be based on a Grayscale dataset which uses 1 channel image of size 200 x 200 pixels. We will firstly normalize the feature dataset by dividing it by 255 (X/255.0). We will be using the relu layer as an activation function for our model. The first layer in our CNN model consists of 64 kernels of shape 3 x 3 which will be applied to our input of grayscale images and then we apply a max-pooling layer. The output of the first layer is fed to another layer which has the same configuration. Then the output of this layer is fed into a softmax layer which will then calculate the probability of the image being from the provided 10 categories.



```
In [11]: import pickle

In [12]: import tensorflow as tf

In [13]: from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D

In [14]: pickle_in = open("X.pickle","rb")
         X = pickle.load(pickle_in)
         pickle_in.close()

         pickle_in = open("y.pickle","rb")
         y = pickle.load(pickle_in)
         pickle_in.close()

In [15]: import numpy as np

In [16]: np.shape(X)

Out[16]: (18160, 200, 200, 1)

In [17]: np.shape(y)

Out[17]: (18160,)

In [ ]: X = X/255.0
```



```
In [ ]: model= Sequential()
        model.add(Conv2D(64,(3,3),input_shape = X.shape[1:]))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2,2)))

        model.add(Conv2D(64,(3,3)))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2,2)))

        model.add(Flatten())
        model.add(Dense(64))

        model.add(Dense(10))
        model.add(Activation("sigmoid"))

        model.compile(loss="sparse_categorical_crossentropy",
                     optimizer = "adam",
                     metrics = ["accuracy"])

        history = model.fit(X, y, batch_size=32 , epochs=1, validation_split= 0.1)

In [ ]: model.save("model1.h5")

In [ ]: import matplotlib.pyplot as plt

In [ ]: plt.plot(history.history['acc'])
        plt.plot(history.history['val_acc'])
        plt.title('model accuracy')
        plt.ylabel('accuracy')
        plt.xlabel('epoch')
        plt.legend(['train', 'val'], loc='upper left')
        plt.show()
```



```
In [ ]: plt.imshow(X[1].reshape(200,200),cmap="gray")

In [ ]: y[1]

In [ ]: model.predict(X[1].reshape(-1,200,200,1))

In [ ]: #It works :)

In [ ]:
```

The optimizer used in model 1 is adam. To train our model we have split our data set into two parts:

- Training dataset : 90% of the original dataset size.
- Validation dataset : 10% of the original dataset.

In model 1 we will train it on

Batch_size is of 32

Number_of_epochs for model 1 = 1

Time taken by model 1 to complete its training = ~20 min

Training acc. of model 1 = 52.6 %

Accuracy of model 1 on the features from validation set = 74%

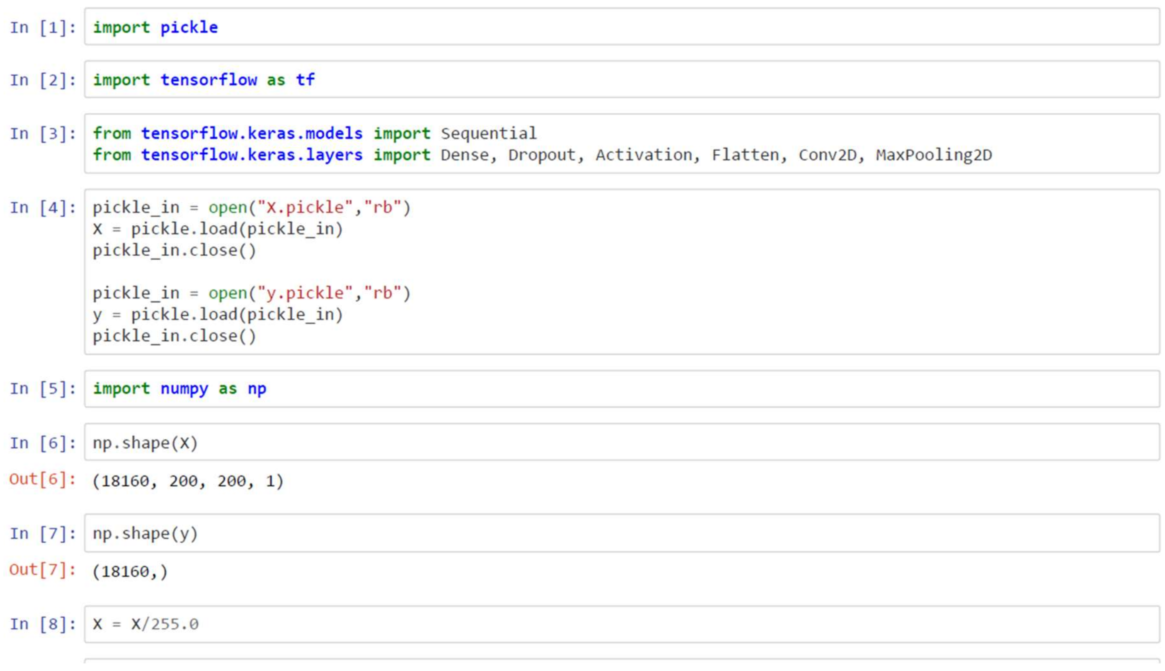
Model 2:

After importing all the required libraries we will load our feature dataset X and label dataset y using Pickle which we previously created in Dataset creation. Our model 2 is an extension of model 1. We will firstly normalize the feature dataset by dividing it by 255.0 (X/255.0). We will be using the relu (rectified linear unit) as an activation function for our model. The first layer in our CNN model consists of 64 kernels of shape 3 x 3 which will be applied to our input of grayscale images and then we apply a max-pooling layer of dimension 2 x 2. The output of the first layer is fed to another layer which has the same configuration as the first layer. Then the output of this layer is fed into a softmax layer which will then calculate the probability of the image being from the provided 10 categories.

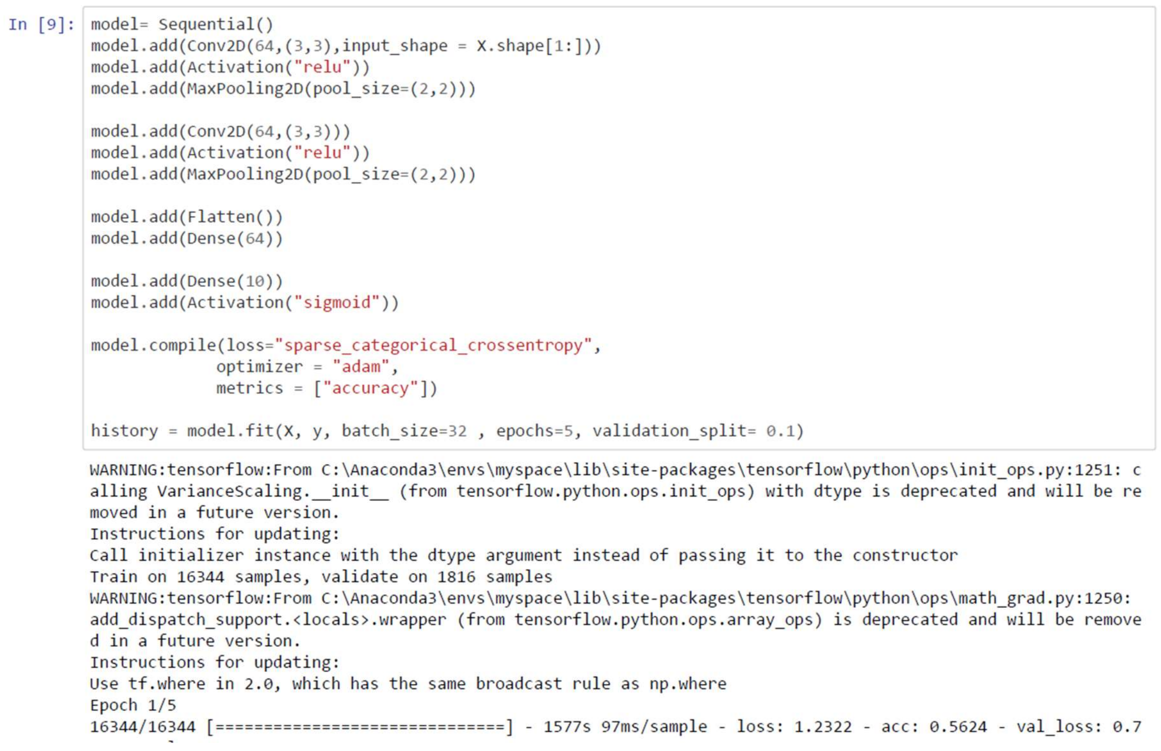
A convolutional neural network with 2 layers using filter of 3 x 3 and relu as a activation function. Output layer uses sigmoid function.

The first convolutional layer filters the input image of our data set with 64 kernels of shape 3*3 after max pooling is applied the output is fed as an input for 2nd convolutional layers with 64 kernels of shape 3*3. Followed by fully connected layer of 64 neurons. The output of this layer is given to a softmax function which calculates the probability distribution of 10 output classes.

Code:



```
In [1]: import pickle

In [2]: import tensorflow as tf

In [3]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D

In [4]: pickle_in = open("X.pickle","rb")
        X = pickle.load(pickle_in)
        pickle_in.close()

        pickle_in = open("y.pickle","rb")
        y = pickle.load(pickle_in)
        pickle_in.close()

In [5]: import numpy as np

In [6]: np.shape(X)

Out[6]: (18160, 200, 200, 1)

In [7]: np.shape(y)

Out[7]: (18160,)

In [8]: X = X/255.0
```



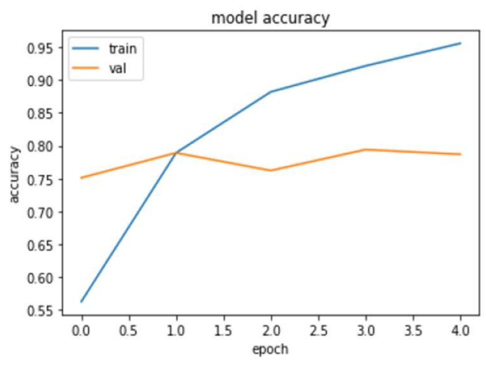
```
In [9]: model= Sequential()
        model.add(Conv2D(64,(3,3),input_shape = X.shape[1:]))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2,2)))

        model.add(Conv2D(64,(3,3)))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2,2)))

        model.add(Flatten())
        model.add(Dense(64))

        model.add(Dense(10))
        model.add(Activation("sigmoid"))

        model.compile(loss="sparse_categorical_crossentropy",
                     optimizer = "adam",
                     metrics = ["accuracy"])

        history = model.fit(X, y, batch_size=32 , epochs=5, validation_split= 0.1)

        WARNING:tensorflow:From C:\Anaconda3\envs\myspace\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: c
        alling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be re
        moved in a future version.
        Instructions for updating:
        Call initializer instance with the dtype argument instead of passing it to the constructor
        Train on 16344 samples, validate on 1816 samples
        WARNING:tensorflow:From C:\Anaconda3\envs\myspace\lib\site-packages\tensorflow\python\ops\math_grad.py:1250:
        add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be remove
        d in a future version.
        Instructions for updating:
        Use tf.where in 2.0, which has the same broadcast rule as np.where
        Epoch 1/5
        16344/16344 [==============================] - 1577s 97ms/sample - loss: 1.2322 - acc: 0.5624 - val_loss: 0.7
```

```
call initializer instance with the dtype argument instead of passing it to the constructor
Train on 16344 samples, validate on 1816 samples
WARNING:tensorflow:From C:\Anaconda3\envs\myspace\lib\site-packages\tensorflow\python\ops\math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be remove
d in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Epoch 1/5
16344/16344 [==============================] - 1577s 97ms/sample - loss: 1.2322 - acc: 0.5624 - val_loss: 0.7
560 - val_acc: 0.7511
Epoch 2/5
16344/16344 [==============================] - 932s 57ms/sample - loss: 0.6051 - acc: 0.7891 - val_loss: 0.63
97 - val_acc: 0.7891
Epoch 3/5
16344/16344 [==============================] - 870s 53ms/sample - loss: 0.3398 - acc: 0.8820 - val_loss: 0.80
50 - val_acc: 0.7621
Epoch 4/5
16344/16344 [==============================] - 866s 53ms/sample - loss: 0.2155 - acc: 0.9216 - val_loss: 0.74
15 - val_acc: 0.7941
Epoch 5/5
16344/16344 [==============================] - 817s 50ms/sample - loss: 0.1188 - acc: 0.9560 - val_loss: 0.90
17 - val_acc: 0.7869
```

```
In [10]: # model.save("model2.h5")
```

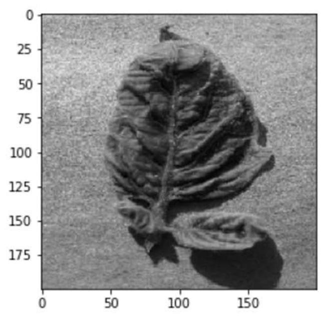
```
In [11]: import matplotlib.pyplot as plt
```

```
In [12]: plt.plot(history.history['acc'])
         plt.plot(history.history['val_acc'])
         plt.title('model accuracy')
         plt.ylabel('accuracy')
         plt.xlabel('epoch')
         plt.legend(['train', 'val'], loc='upper left')
         plt.show()
```



```
In [13]: plt.imshow(X[1].reshape(200,200),cmap="gray")
```

```
Out[13]: <matplotlib.image.AxesImage at 0x1e700705888>
```



```
In [14]: y[1]

Out[14]: 9

In [15]: model.predict(X[1].reshape(-1,200,200,1))

Out[15]: array([[0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                 0.0000000e+00, 0.0000000e+00, 1.4007092e-06, 2.9802322e-08,
                 0.0000000e+00, 9.8786098e-01]], dtype=float32)

In [16]: #It works :)
```

Model 2 is very similar to model 1 but here we train it on different hyper-parameters.

The optimizer used in model 2 is adam. To train our model we have split our data set into two parts:

Training dataset : 90% of the original dataset size.

Validation dataset : 10% of the original dataset.

In model 2 we have trained it on

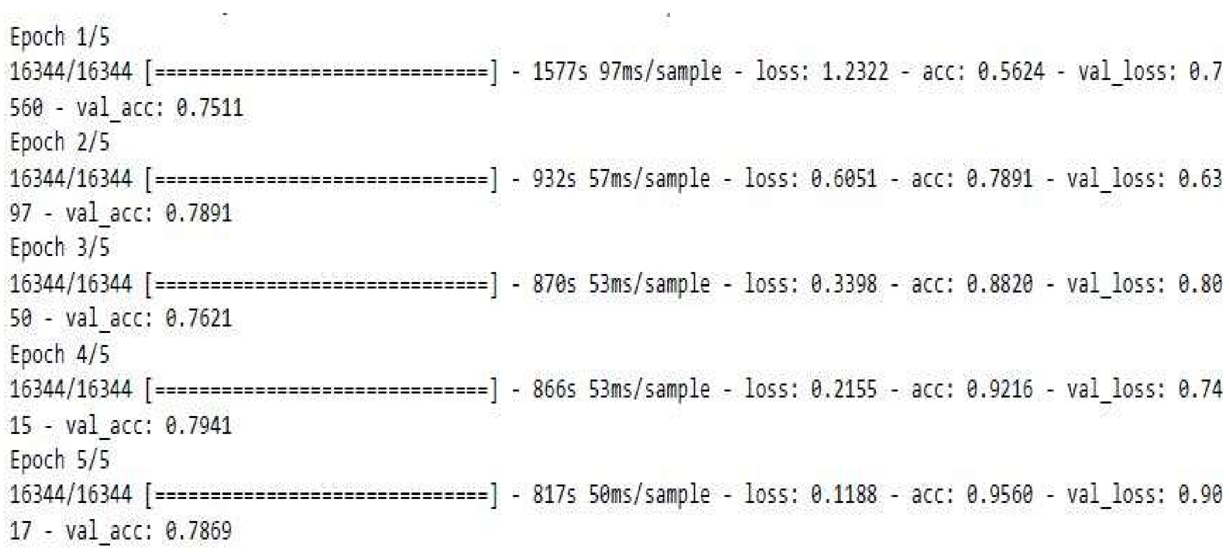
Batch_size is of 32

Number_of_epochs for model 2 = 5

Time taken by model 1 to complete its training = ~1hr 50 min

Training_acc. of model 2 = 95.6 %

Accuracy of model 2 on the features from validation set = 77%



```
Epoch 1/5
16344/16344 [==============================] - 1577s 97ms/sample - loss: 1.2322 - acc: 0.5624 - val_loss: 0.7
560 - val_acc: 0.7511
Epoch 2/5
16344/16344 [==============================] - 932s 57ms/sample - loss: 0.6051 - acc: 0.7891 - val_loss: 0.63
97 - val_acc: 0.7891
Epoch 3/5
16344/16344 [==============================] - 870s 53ms/sample - loss: 0.3398 - acc: 0.8820 - val_loss: 0.80
50 - val_acc: 0.7621
Epoch 4/5
16344/16344 [==============================] - 866s 53ms/sample - loss: 0.2155 - acc: 0.9216 - val_loss: 0.74
15 - val_acc: 0.7941
Epoch 5/5
16344/16344 [==============================] - 817s 50ms/sample - loss: 0.1188 - acc: 0.9560 - val_loss: 0.90
17 - val_acc: 0.7869
```

Figure 20: Training Stats of Model 2

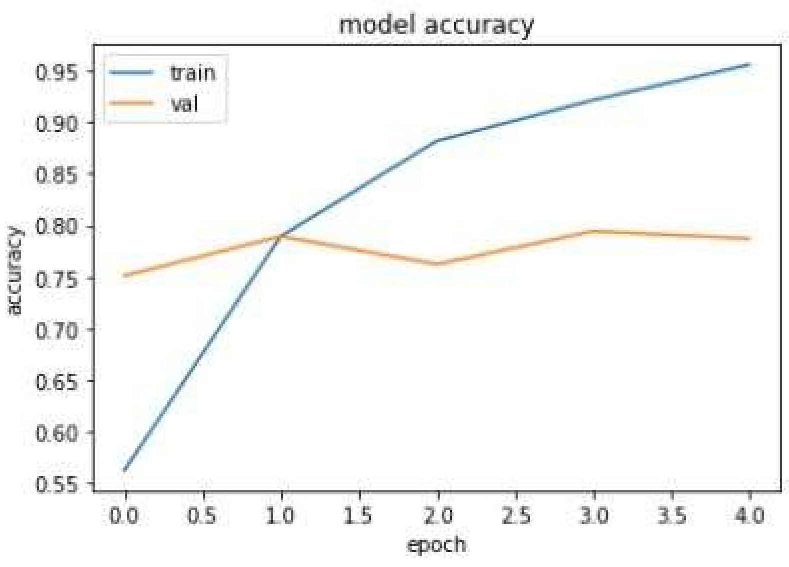


Figure 21 Performance Analysis Graph

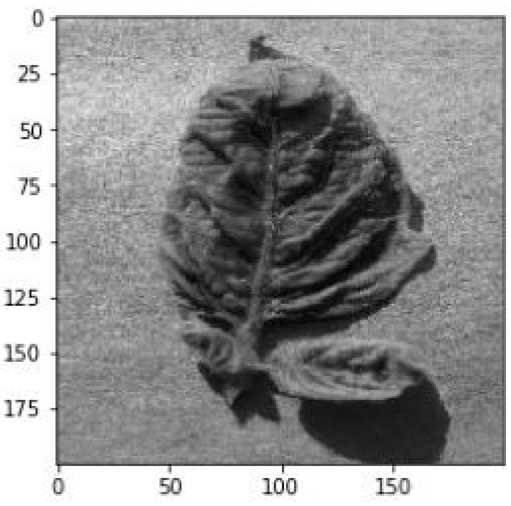
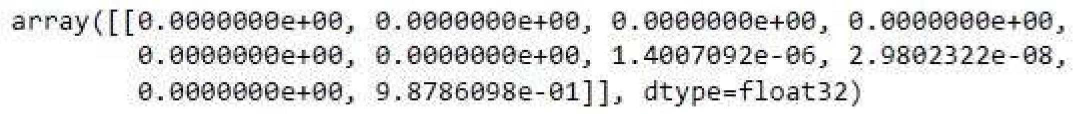


Figure 22 Original Label: 9 Predicted Label: 9 (98.7%)



```
array([[0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 1.4007092e-06, 2.9802322e-08,
        0.0000000e+00, 9.8786098e-01]], dtype=float32)
```

Figure 23 Model Confidence for Original label: 98.76%

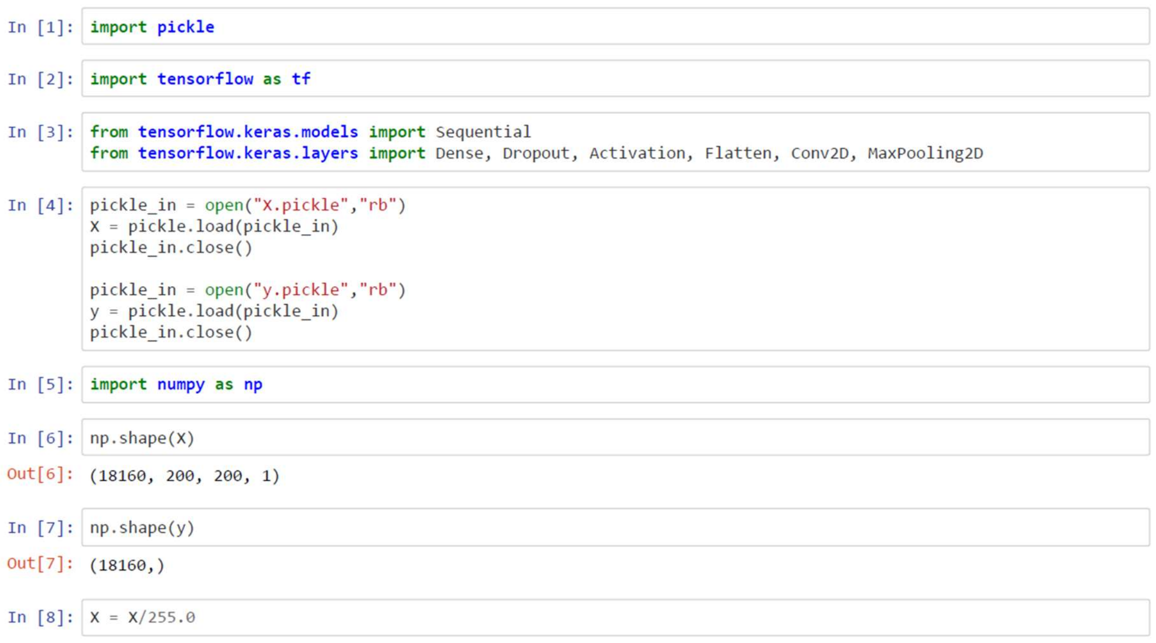
Model 3:

Model 3 is very similar to model 2. In model 3 we have increased the depth of our CNN model and we have introduced 2 more layers. This new layer in our CNN model consists of 64 kernels of shape 3 x 3 which will be applied to our input of grayscale images and then we apply a max-pooling layer of dimension 2 x 2.

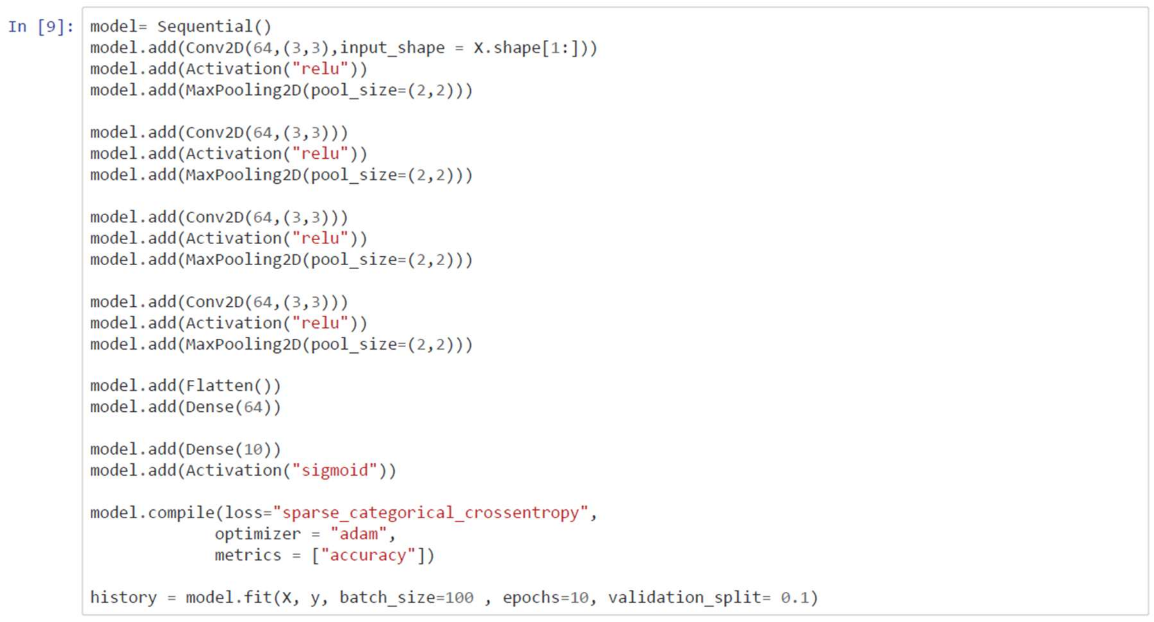
A convolutional neural network with 4 layers using a filter of 3 x 3 and relu as an activation function. The output layer uses sigmoid function.

The first convolutional layer filters the input image of our data set with 64 kernels of shape 3*3 after max pooling is applied the output is fed as an input for 2nd convolutional layers with 64 kernels of shape 3*3, which is then again fed as an input for 3rd convolutional layers with 64 kernels of shape 3*3 Followed by fully connected layer of 64 neurons, which is then again fed as an input for 4th convolutional layers with 64 kernels of shape 3*3 Followed by fully connected layer of 64 neurons. The output of this layer is given to a softmax function which calculates the probability distribution of 10 output classes.

Code:



```
In [1]: import pickle

In [2]: import tensorflow as tf

In [3]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D

In [4]: pickle_in = open("X.pickle","rb")
        X = pickle.load(pickle_in)
        pickle_in.close()

        pickle_in = open("y.pickle","rb")
        y = pickle.load(pickle_in)
        pickle_in.close()

In [5]: import numpy as np

In [6]: np.shape(X)

Out[6]: (18160, 200, 200, 1)

In [7]: np.shape(y)

Out[7]: (18160,)

In [8]: X = X/255.0
```



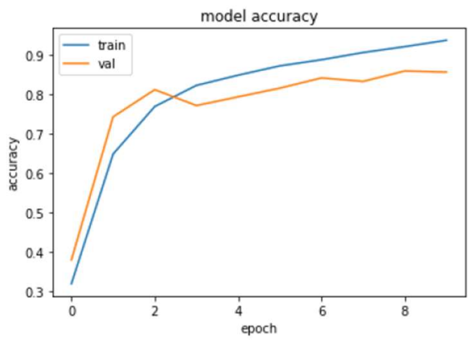
```
In [9]: model= Sequential()
        model.add(Conv2D(64,(3,3),input_shape = X.shape[1:]))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2,2)))

        model.add(Conv2D(64,(3,3)))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2,2)))

        model.add(Conv2D(64,(3,3)))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2,2)))

        model.add(Conv2D(64,(3,3)))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2,2)))

        model.add(Flatten())
        model.add(Dense(64))

        model.add(Dense(10))
        model.add(Activation("sigmoid"))

        model.compile(loss="sparse_categorical_crossentropy",
                      optimizer = "adam",
                      metrics = ["accuracy"])

        history = model.fit(X, y, batch_size=100 , epochs=10, validation_split= 0.1)
```

```
WARNING:tensorflow:From C:\Anaconda3\envs\myspace\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: c
alling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be re
moved in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Train on 16344 samples, validate on 1816 samples
WARNING:tensorflow:From C:\Anaconda3\envs\myspace\lib\site-packages\tensorflow\python\ops\math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be remove
d in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Epoch 1/10
16344/16344 [==============================] - 1271s 78ms/sample - loss: 1.6445 - acc: 0.3194 - val_loss: 1.4
746 - val_acc: 0.3800
Epoch 2/10
16344/16344 [==============================] - 1442s 88ms/sample - loss: 0.9871 - acc: 0.6479 - val_loss: 0.7
391 - val_acc: 0.7423
Epoch 3/10
16344/16344 [==============================] - 1482s 91ms/sample - loss: 0.6631 - acc: 0.7685 - val_loss: 0.5
728 - val_acc: 0.8111
Epoch 4/10
16344/16344 [==============================] - 1501s 92ms/sample - loss: 0.5176 - acc: 0.8220 - val_loss: 0.7
247 - val_acc: 0.7709
Epoch 5/10
16344/16344 [==============================] - 1680s 103ms/sample - loss: 0.4361 - acc: 0.8478 - val_loss: 0.
6155 - val_acc: 0.7930
Epoch 6/10
16344/16344 [==============================] - 1424s 87ms/sample - loss: 0.3737 - acc: 0.8714 - val_loss: 0.5
668 - val_acc: 0.8150
Epoch 7/10
16344/16344 [==============================] - 1420s 87ms/sample - loss: 0.3205 - acc: 0.8870 - val_loss: 0.5
369 - val_acc: 0.8409
Epoch 8/10
16344/16344 [==============================] - 1399s 86ms/sample - loss: 0.2727 - acc: 0.9053 - val_loss: 0.5
714 - val_acc: 0.8320
```

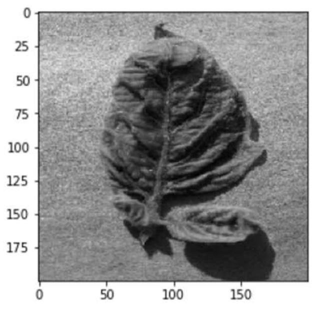
```
Epoch 9/10
16344/16344 [==============================] - 1411s 86ms/sample - loss: 0.2299 - acc: 0.9202 - val_loss: 0.4
426 - val_acc: 0.8585
Epoch 10/10
16344/16344 [==============================] - 1438s 88ms/sample - loss: 0.1835 - acc: 0.9362 - val_loss: 0.5
075 - val_acc: 0.8557
```

```
In [10]: #model.save("model3.h5")
```

```
In [11]: import matplotlib.pyplot as plt
In [12]: plt.plot(history.history['acc'])
         plt.plot(history.history['val_acc'])
         plt.title('model accuracy')
         plt.ylabel('accuracy')
         plt.xlabel('epoch')
         plt.legend(['train', 'val'], loc='upper left')
         plt.show()
```



```
In [13]: plt.imshow(X[1].reshape(200,200),cmap="gray")

Out[13]: <matplotlib.image.AxesImage at 0x17802385408>
```



```
In [14]: y[1]

Out[14]: 9

In [15]: model.predict(X[1].reshape(-1,200,200,1))

Out[15]: array([[0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                 0.0000000e+00, 0.0000000e+00, 3.5196543e-05, 1.7881393e-06,
                 1.4027135e-07, 8.9661920e-01]], dtype=float32)

In [16]: #It works :)
```

The optimizer used in model 3 is adam. To train our model we have split our data set into two parts:

Training dataset : 90% of the original dataset size.

Validation dataset : 10% of the original dataset.

In model 3 we have trained it on

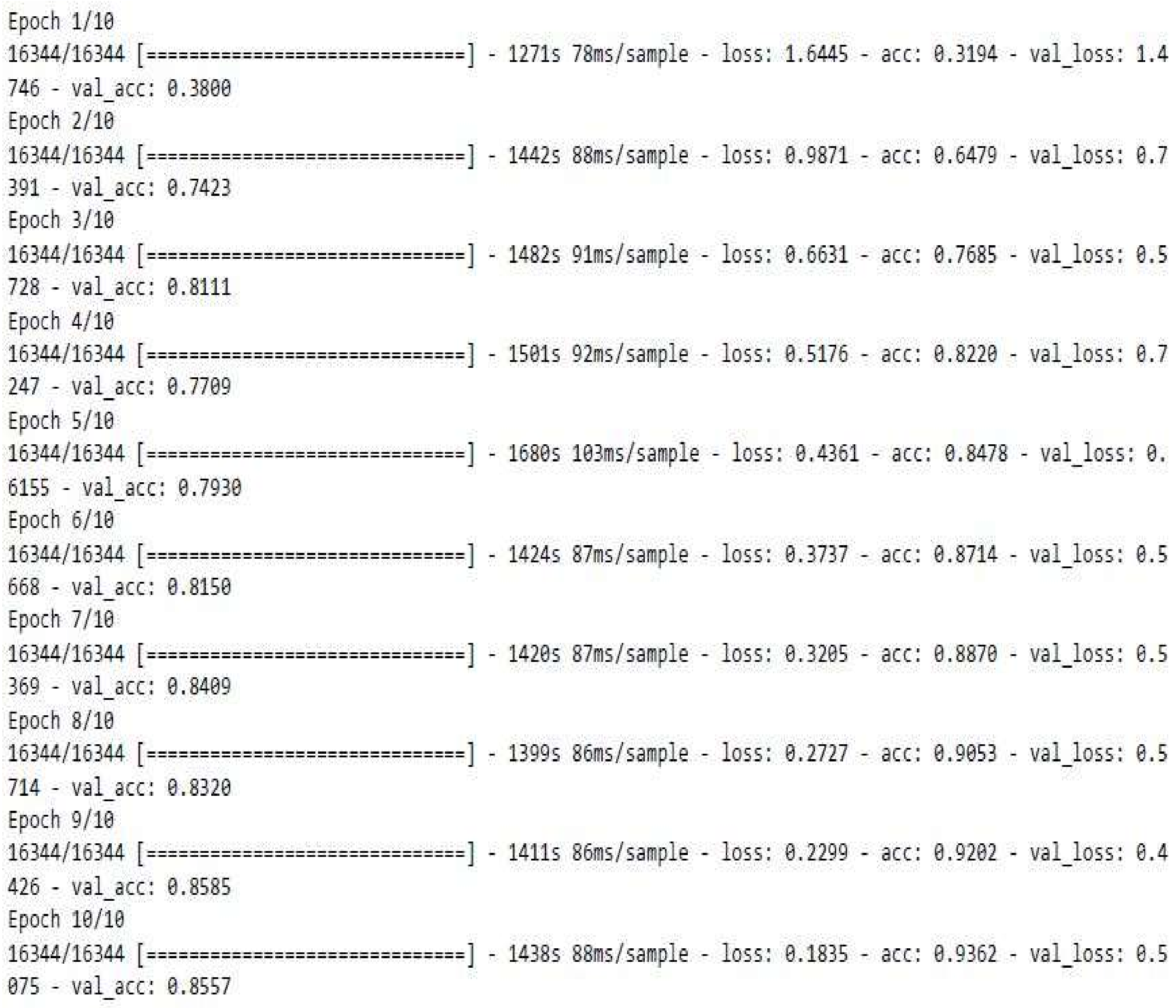
Batch_size is of 100

Number_of_epochs for model 3 = 10

Time taken by model 1 to complete its training = ~6hr

Training_acc. of model 2 = 93.6 %

Accuracy of model 2 on the features from validation set = 85.5%



```
Epoch 1/10
16344/16344 [==============================] - 1271s 78ms/sample - loss: 1.6445 - acc: 0.3194 - val_loss: 1.4
746 - val_acc: 0.3800
Epoch 2/10
16344/16344 [==============================] - 1442s 88ms/sample - loss: 0.9871 - acc: 0.6479 - val_loss: 0.7
391 - val_acc: 0.7423
Epoch 3/10
16344/16344 [==============================] - 1482s 91ms/sample - loss: 0.6631 - acc: 0.7685 - val_loss: 0.5
728 - val_acc: 0.8111
Epoch 4/10
16344/16344 [==============================] - 1501s 92ms/sample - loss: 0.5176 - acc: 0.8220 - val_loss: 0.7
247 - val_acc: 0.7709
Epoch 5/10
16344/16344 [==============================] - 1680s 103ms/sample - loss: 0.4361 - acc: 0.8478 - val_loss: 0.
6155 - val_acc: 0.7930
Epoch 6/10
16344/16344 [==============================] - 1424s 87ms/sample - loss: 0.3737 - acc: 0.8714 - val_loss: 0.5
668 - val_acc: 0.8150
Epoch 7/10
16344/16344 [==============================] - 1420s 87ms/sample - loss: 0.3205 - acc: 0.8870 - val_loss: 0.5
369 - val_acc: 0.8409
Epoch 8/10
16344/16344 [==============================] - 1399s 86ms/sample - loss: 0.2727 - acc: 0.9053 - val_loss: 0.5
714 - val_acc: 0.8320
Epoch 9/10
16344/16344 [==============================] - 1411s 86ms/sample - loss: 0.2299 - acc: 0.9202 - val_loss: 0.4
426 - val_acc: 0.8585
Epoch 10/10
16344/16344 [==============================] - 1438s 88ms/sample - loss: 0.1835 - acc: 0.9362 - val_loss: 0.5
075 - val_acc: 0.8557
```

Figure 24: Training Stats of Model 2

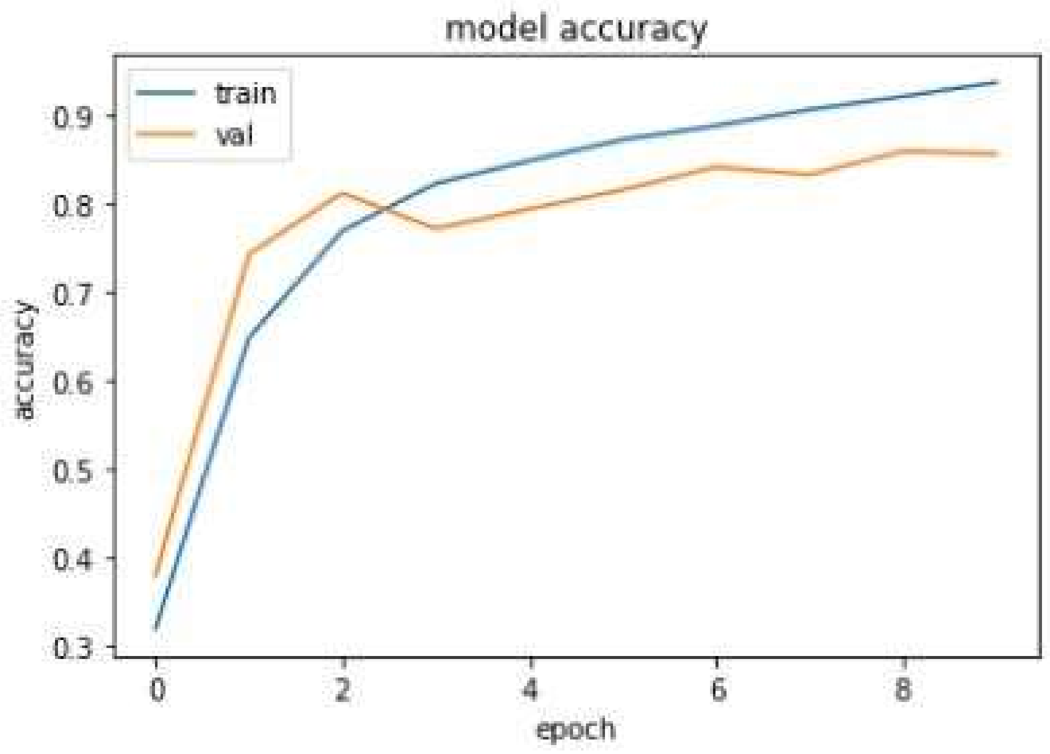


Figure 25 Performance Analysis Graph

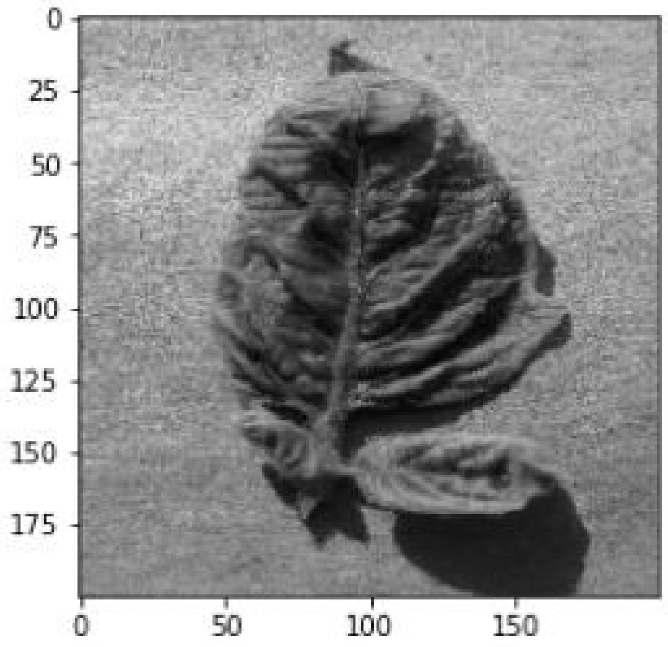
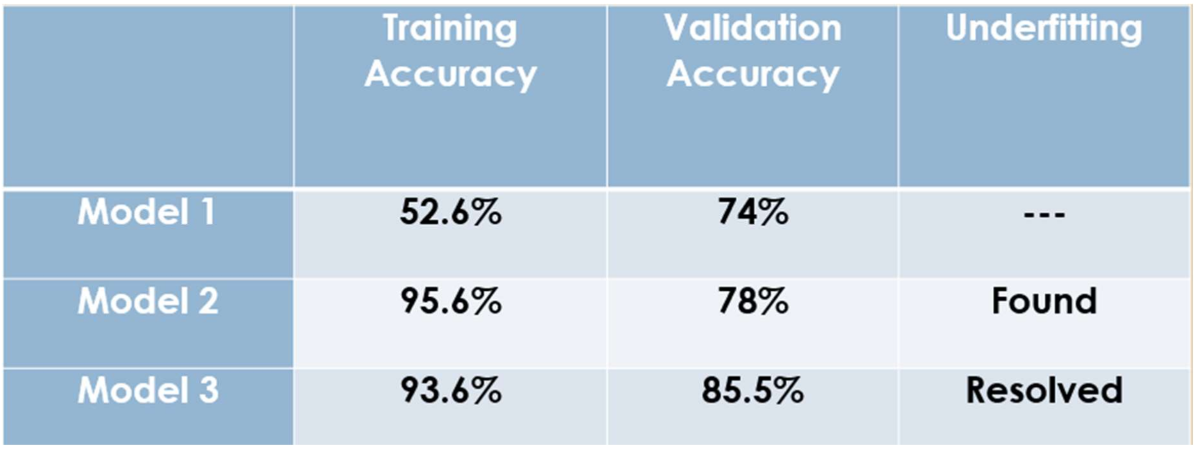


Figure 26 Original Label: 9 Predicted Label: 9 (89.6%)



	Training Accuracy	Validation Accuracy	Underfitting
Model 1	52.6%	74%	---
Model 2	95.6%	78%	Found
Model 3	93.6%	85.5%	Resolved

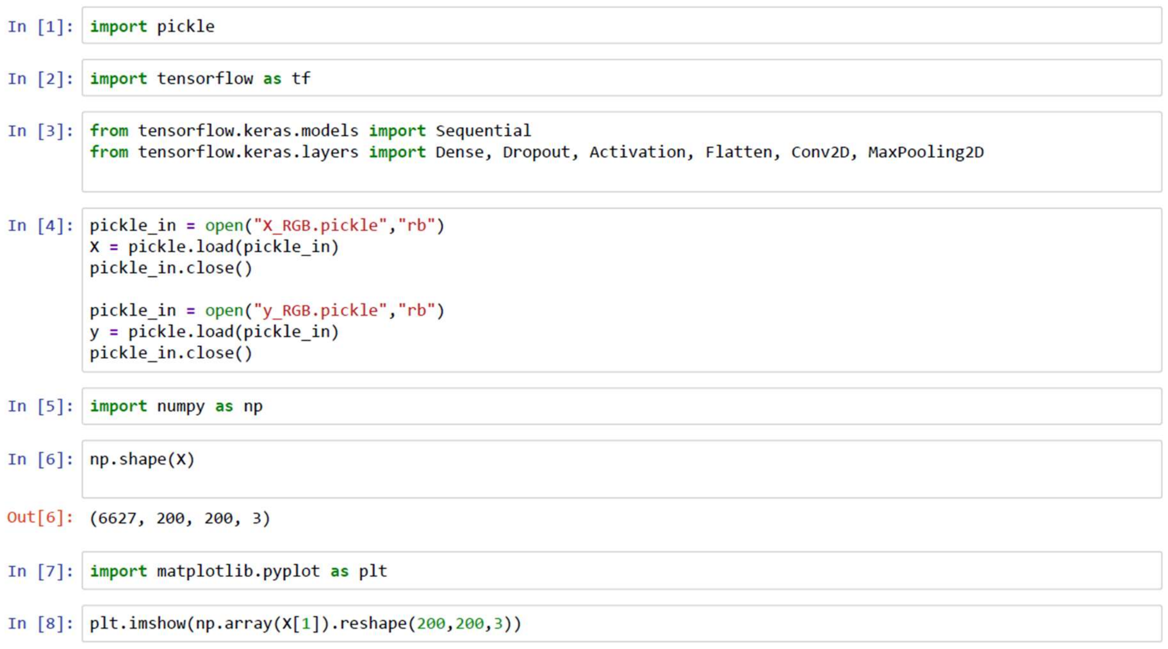
Figure 27: Models Comparison

Our first system consists of a single epoch and it has a training efficiency of ~52% which jumps to 95% in our second model which comprises 5 epochs. Then we developed other models to obtain a better result. So, In this model, we increased the number of hidden layers, the batch size is changed to 100 and for improving the training results epochs are changed to 10. The training accuracy in this model is obtained 95 % and validation accuracy is 85.5 %. Hence we concluded that model 3 is better from both the first and second models and has satisfactorily achieved the results for us. In this model, we also have solved the problem of underfitting which was observed in models 1 and 2. We have also observed that by increasing the depth of neural network underfitting can be removed significantly.

RGB Models

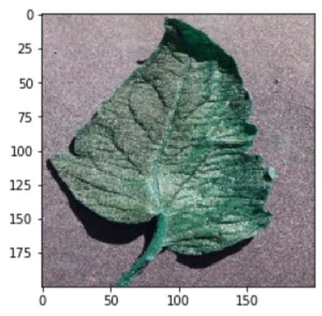
Model 1:

The first step is to load all the libraries required for creating and training our model. After importing all the required libraries we will load our feature dataset X and label dataset y using Pickle which we previously created in Dataset creation, note that this dataset must contain images that are in RGB format. Our model 1 will be based on a Coloured dataset that uses 3 channel images of size 200 x 200 x 3 pixels where 3 determines the number of channels. We will firstly normalize the feature dataset by dividing it by 255 (X/255.0). We will be using the relu layer as an activation function for our model. The first layer in our CNN model consists of 64 kernels of shape 3 x 3 which will be applied to our input of RGB images and then we apply a max-pooling layer. The output of the first layer is fed to the second layer which has the same configuration, and then the output of the second layer is fed to the third layer which has the same configuration. Then the output of this layer is fed into a softmax layer which will then calculate the probability of the image being from the provided 3 categories.



```
In [1]: import pickle

In [2]: import tensorflow as tf

In [3]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D

In [4]: pickle_in = open("X_RGB.pickle","rb")
        X = pickle.load(pickle_in)
        pickle_in.close()

        pickle_in = open("y_RGB.pickle","rb")
        y = pickle.load(pickle_in)
        pickle_in.close()

In [5]: import numpy as np

In [6]: np.shape(X)

Out[6]: (6627, 200, 200, 3)

In [7]: import matplotlib.pyplot as plt

In [8]: plt.imshow(np.array(X[1]).reshape(200,200,3))
```

```
Out[8]: <matplotlib.image.AxesImage at 0x17c1386f188>
```



```
In [9]: np.shape(y)

Out[9]: (6627,)

In [10]: X.shape[1:]

Out[10]: (200, 200, 3)

In [11]: X = X/255.0
```

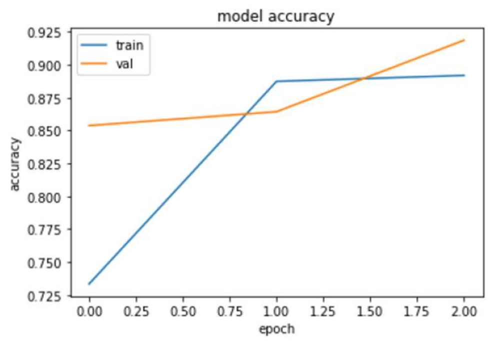
```
In [39]: model= Sequential()
         model.add(Conv2D(64,(3,3),input_shape = X.shape[1:]))
         model.add(Activation("relu"))
         model.add(MaxPooling2D(pool_size=(2,2)))

         model.add(Conv2D(64,(3,3)))
         model.add(Activation("relu"))
         model.add(MaxPooling2D(pool_size=(2,2)))

         model.add(Conv2D(64,(3,3)))
         model.add(Activation("relu"))
         model.add(MaxPooling2D(pool_size=(2,2)))

         model.add(Flatten())
         model.add(Dense(64))

         model.add(Dense(4))
         model.add(Activation("sigmoid"))

         model.compile(loss="sparse_categorical_crossentropy",
                       optimizer = "adam",
                       metrics = ["accuracy"])

         history = model.fit(X, y, batch_size=32 , epochs=3, validation_split= 0.1)
```

```
Train on 5964 samples, validate on 663 samples
Epoch 1/3
5964/5964 [==============================] - 652s 109ms/sample - loss: 0.6785 - acc: 0.7332 - val_loss: 0.3937
- val_acc: 0.8537
Epoch 2/3
5964/5964 [==============================] - 675s 113ms/sample - loss: 0.2966 - acc: 0.8873 - val_loss: 0.4075
- val_acc: 0.8643
Epoch 3/3
5964/5964 [==============================] - 737s 124ms/sample - loss: 0.3008 - acc: 0.8919 - val_loss: 0.2461
- val_acc: 0.9186
```

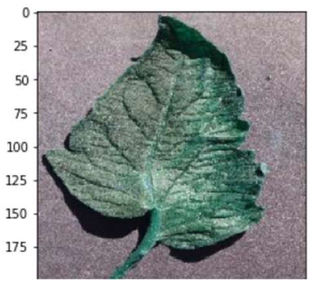
```
In [40]: model.save("model1_RGB.h5")
```

```
In [41]: import matplotlib.pyplot as plt
```

```
In [42]: plt.plot(history.history['acc'])
         plt.plot(history.history['val_acc'])
         plt.title('model accuracy')
         plt.ylabel('accuracy')
         plt.xlabel('epoch')
         plt.legend(['train', 'val'], loc='upper left')
         plt.show()
```



```
In [43]: plt.imshow(X[1].reshape(200,200,3),cmap="gray")
Out[43]: <matplotlib.image.AxesImage at 0x17df28c2fc8>
```



```
In [44]: y[14]
Out[44]: 0
```

```
In [45]: model.predict(X[14].reshape(-1,200,200,3)/255.0)
Out[45]: array([[0., 0., 0., 1.]], dtype=float32)
```

The optimizer used in model 1 is adam. To train our model we have split our data set into two parts:

Training dataset : 90% of the original dataset size.

Validation dataset : 10% of the original dataset.

In model 1 we have trained it on

Batch_size is of 32

Number_of_epochs for model 3 = 3

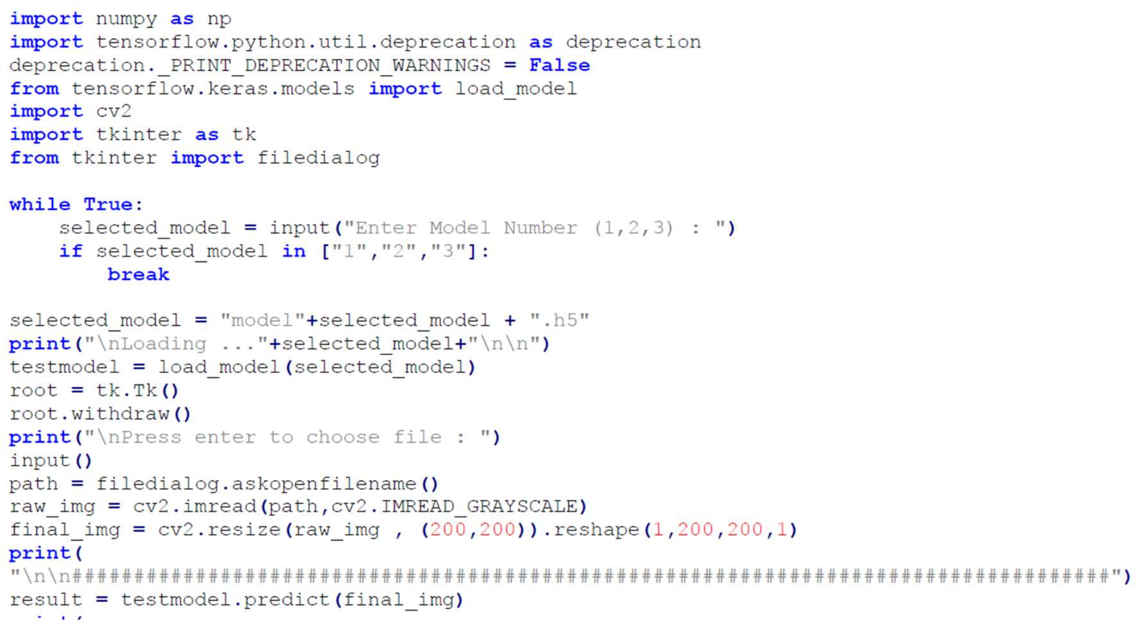
Time taken by model 1 to complete its training = ~11hr 20min

Training_acc. of model 1 = 89.1 %

Accuracy of model 1 on the features from validation set = 91.8%

Program Execution:

Driver Code: For Single channel



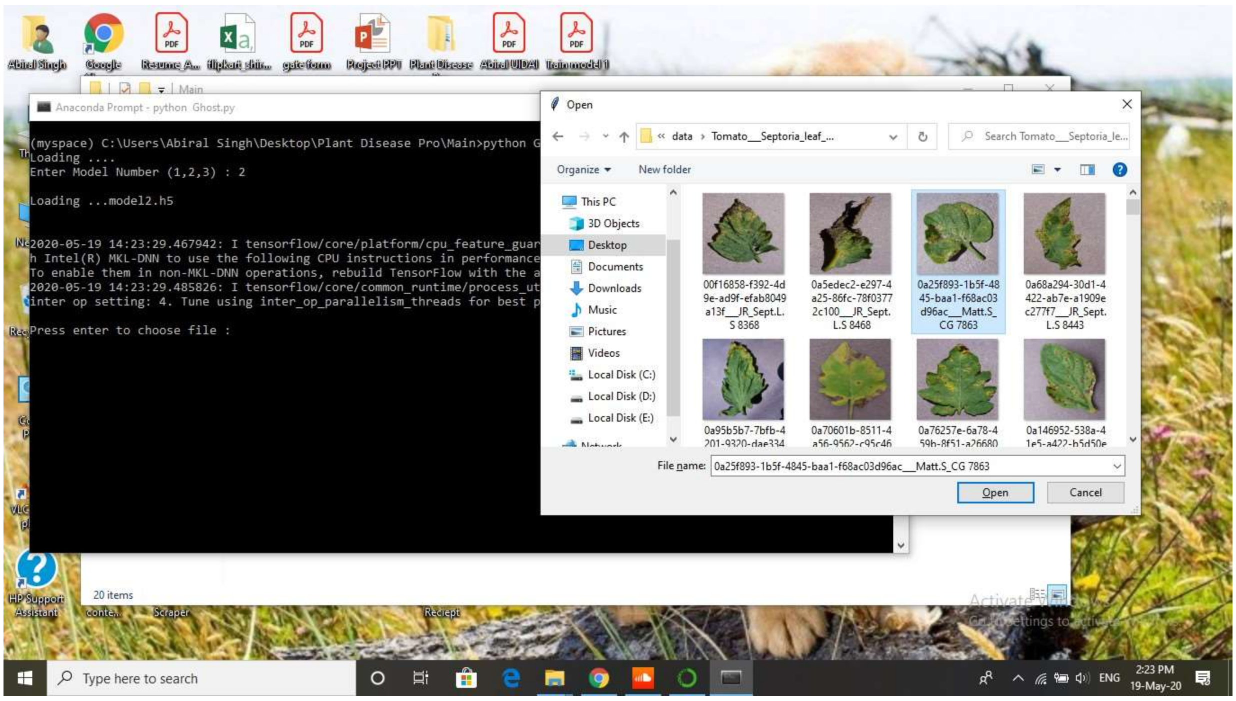
```
import numpy as np
import tensorflow.python.util.deprecation as deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False
from tensorflow.keras.models import load_model
import cv2
import tkinter as tk
from tkinter import filedialog

while True:
    selected_model = input("Enter Model Number (1,2,3) : ")
    if selected_model in ["1","2","3"]:
        break

selected_model = "model"+selected_model + ".h5"
print("\nLoading ..."+selected_model+"\n\n")
testmodel = load_model(selected_model)
root = tk.Tk()
root.withdraw()
print("\nPress enter to choose file : ")
input()
path = filedialog.askopenfilename()
raw_img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
final_img = cv2.resize(raw_img , (200,200)).reshape(1,200,200,1)
print(
"\n\n#####################################################################################")
result = testmodel.predict(final_img)
```

This is a driver code for our model of processing single channeled images. Here the user will be asked to enter the model number (Out of 3 models of grayscale analysis). After selecting the model the user will be asked to select a file from storage. The image selected by the user can be an RGB image but the system will convert the image to a grey-scale image and then the image is reshaped to 200*200 pixel image. After converting the image final image is then fed to the selected model.

```
print(
"#####################################################################################\n\n")
result = np.array(result).tolist()[0]
CATAGORIES = ["Tomato___Bacterial_spot","Tomato___Early_blight","Tomato___healthy",
"Tomato___Late_blight","Tomato___Leaf_Mold","Tomato___Septoria_leaf_spot",
"Tomato___Spider_mites Two-spotted_spider_mite","Tomato___Target_Spot",
"Tomato___Tomato_mosaic_virus","Tomato___Tomato_Yellow_Leaf_Curl_Virus"]
max_val = max(result)
index = result.index(max_val)
print("First Guess : "+CATAGORIES[index])
result[index] = -1

max_val = max(result)
index = result.index(max_val)
print("Second Guess : " + CATAGORIES[index])
result[index] = -1

max_val = max(result)
index = result.index(max_val)
print("Third Guess : " + CATAGORIES[index])
result[index] = -1
```

The image given to the model is analyzed and the list of possible values is returned to the variable result. The result is predicted as three guesses based on the list obtained from the model.

Output:


```
Anaconda Prompt - python Ghost.py

(myspace) C:\Users\Abiral Singh\Desktop\Plant Disease Pro\Main>python G
Loading ....
Enter Model Number (1,2,3) : 2

Loading ...model2.h5


2020-05-19 14:23:29.467942: I tensorflow/core/platform/cpu_feature_guar
h Intel(R) MKL-DNN to use the following CPU instructions in performance
To enable them in non-MKL-DNN operations, rebuild TensorFlow with the a
2020-05-19 14:23:29.485826: I tensorflow/core/common_runtime/process_ut
inter op setting: 4. Tune using inter_op_parallelism_threads for best p

Press enter to choose file :
```

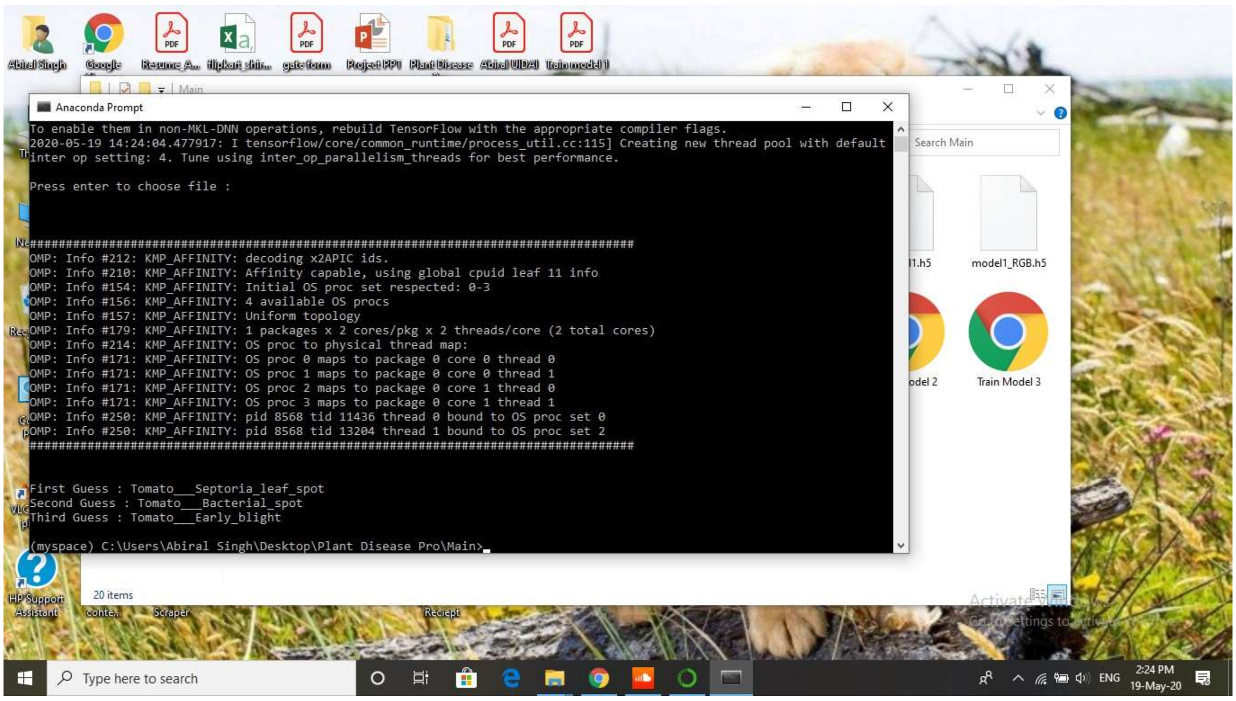
Open

data › Tomato___Septoria_leaf_...

File name: 0a25f893-1b5f-4845-baa1-f68ac03d96ac___Matt.S_CG 7863

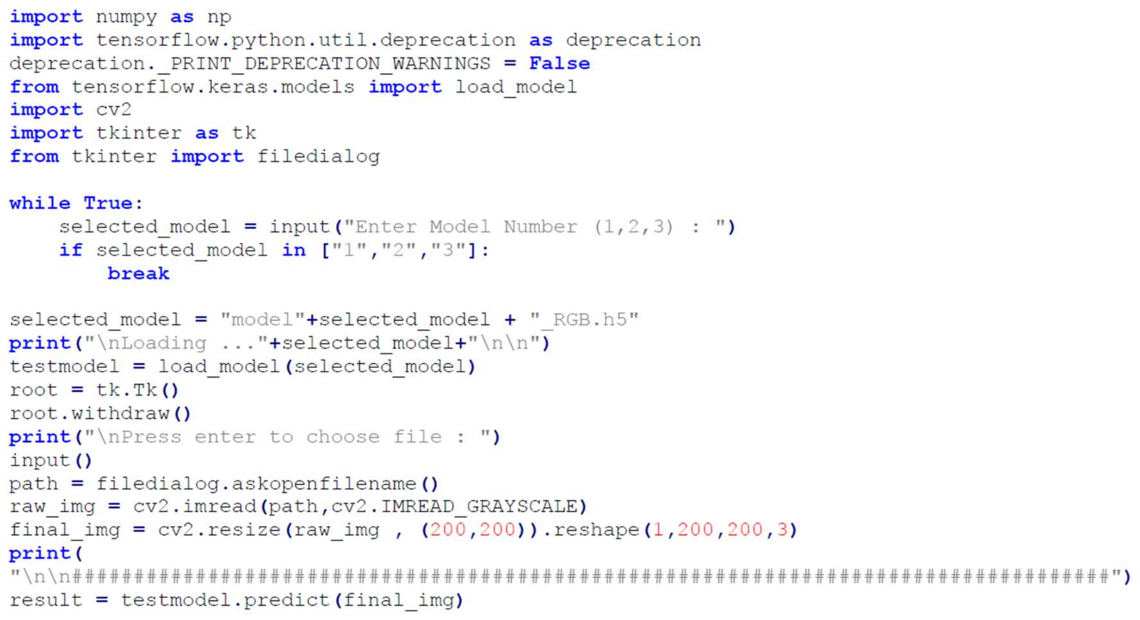
Search Tomato___Septoria_le...
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Local Disk (E:)
00f16858-f392-4d9e-ad9f-efab8049a13f___JR_Sept.L.S 8368
0a5edec2-e297-4a25-86fc-78f03772c100___JR_Sept.L.S 8468
0a25f893-1b5f-4845-baa1-f68ac03d96ac___Matt.S_CG 7863
0a68a294-30d1-4422-ab7e-a1909ec277f7___JR_Sept.L.S 8443
Open
Cancel
Type here to search
2:23 PM
19-May-20

This is a snapshot of the executed code, here user is asked to choose the image from storage.


```
Anaconda Prompt
To enable them in non-MKL-DNN operations, rebuild TensorFlow with the appropriate compiler flags.
2020-05-19 14:24:04.477917: I tensorflow/core/common_runtime/process_util.cc:115] Creating new thread pool with default
inter op setting: 4. Tune using inter_op_parallelism_threads for best performance.

Press enter to choose file :



####################################################################################
OMP: Info #212: KMP_AFFINITY: decoding x2APIC ids.
OMP: Info #210: KMP_AFFINITY: Affinity capable, using global cpuid leaf 11 info
OMP: Info #154: KMP_AFFINITY: Initial OS proc set respected: 0-3
OMP: Info #156: KMP_AFFINITY: 4 available OS procs
OMP: Info #157: KMP_AFFINITY: Uniform topology
OMP: Info #179: KMP_AFFINITY: 1 packages x 2 cores/pkg x 2 threads/core (2 total cores)
OMP: Info #214: KMP_AFFINITY: OS proc to physical thread map:
OMP: Info #171: KMP_AFFINITY: OS proc 0 maps to package 0 core 0 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 1 maps to package 0 core 0 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 2 maps to package 0 core 1 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 3 maps to package 0 core 1 thread 1
OMP: Info #250: KMP_AFFINITY: pid 8568 tid 11436 thread 0 bound to OS proc set 0
OMP: Info #250: KMP_AFFINITY: pid 8568 tid 13204 thread 1 bound to OS proc set 2
####################################################################################


First Guess : Tomato___Septoria_leaf_spot
Second Guess : Tomato___Bacterial_spot
Third Guess : Tomato___Early_blight

(myspace) C:\Users\Abiral Singh\Desktop\Plant Disease Pro\Main>_
```

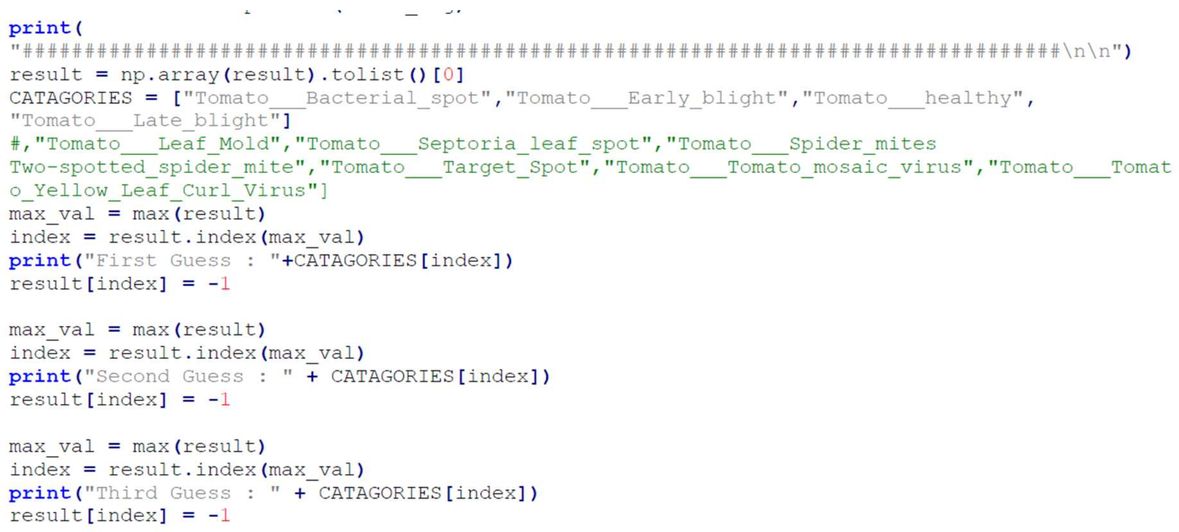
This snap is showing the final output of the system.

Driver Code: (RGB)



```
import numpy as np
import tensorflow.python.util.deprecation as deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False
from tensorflow.keras.models import load_model
import cv2
import tkinter as tk
from tkinter import filedialog

while True:
    selected_model = input("Enter Model Number (1,2,3) : ")
    if selected_model in ["1","2","3"]:
        break

selected_model = "model"+selected_model + "_RGB.h5"
print("\nLoading ..."+selected_model+"\n\n")
testmodel = load_model(selected_model)
root = tk.Tk()
root.withdraw()
print("\nPress enter to choose file : ")
input()
path = filedialog.askopenfilename()
raw_img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
final_img = cv2.resize(raw_img , (200,200)).reshape(1,200,200,3)
print(
"\n\n#####################################################################################")
result = testmodel.predict(final_img)
```

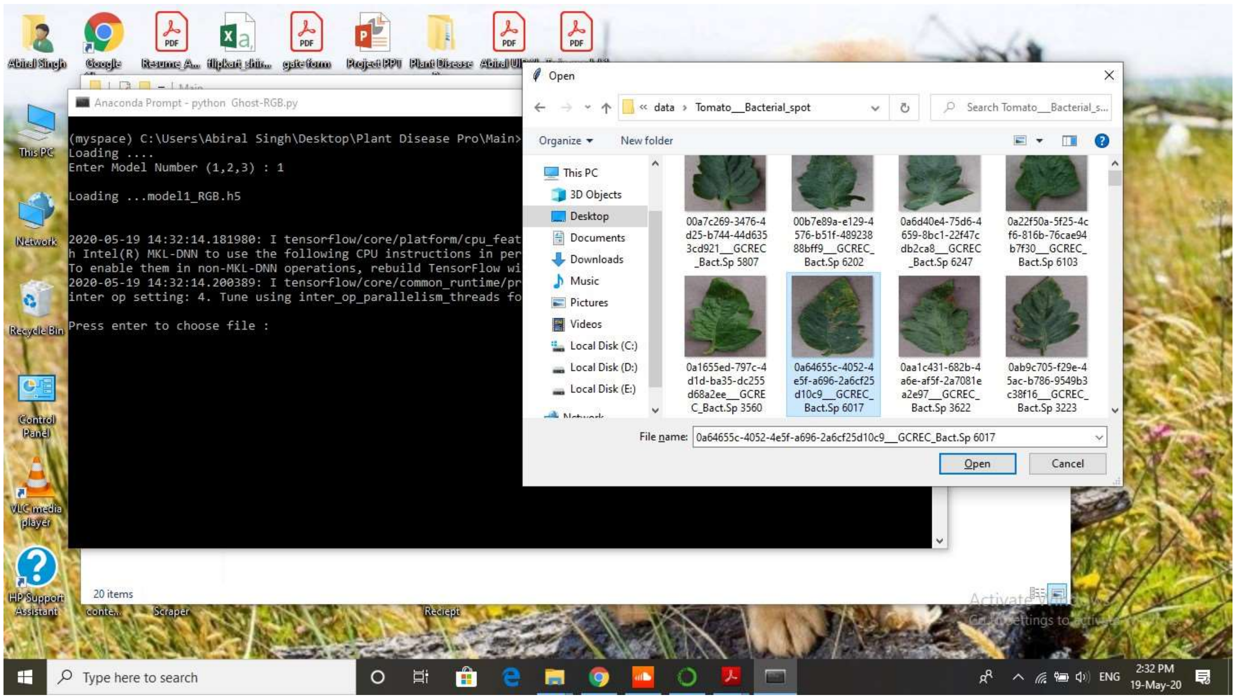
This is a driver code for our model of processing RGB images. Here the user will be asked to enter the model number. After selecting the model the user will be asked to select a file from storage. The image selected by the user can be an RGB image but the system will convert the image to a grey-scale image and then the image is reshaped to 200*200*3 pixel, image(3 is for RGB). After converting the image final image is then fed to the selected model.



```
print(
"######################################################################################\n\n")
result = np.array(result).tolist()[0]
CATAGORIES = ["Tomato___Bacterial_spot","Tomato___Early_blight","Tomato___healthy",
"Tomato___Late_blight"]
#,"Tomato___Leaf_Mold","Tomato___Septoria_leaf_spot","Tomato___Spider_mites
Two-spotted_spider_mite","Tomato___Target_Spot","Tomato___Tomato_mosaic_virus","Tomato___Tomat
o_Yellow_Leaf_Curl_Virus"]
max_val = max(result)
index = result.index(max_val)
print("First Guess : "+CATAGORIES[index])
result[index] = -1

max_val = max(result)
index = result.index(max_val)
print("Second Guess : " + CATAGORIES[index])
result[index] = -1

max_val = max(result)
index = result.index(max_val)
print("Third Guess : " + CATAGORIES[index])
result[index] = -1
```

The image given to the model is analyzed and the list of possible values is returned to the variable result. The result is predicted as three guesses based on the list obtained from the model.

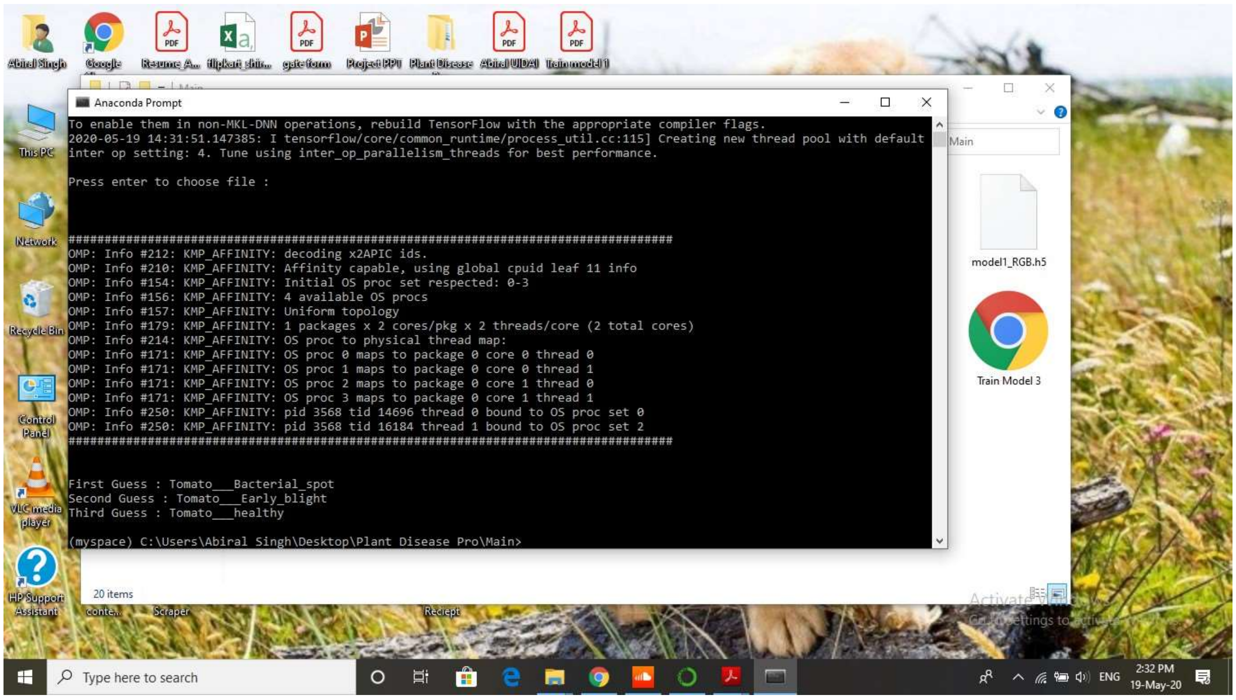
Output:



```
(myspace) C:\Users\Abiral Singh\Desktop\Plant Disease Pro\Main>
Loading ....
Enter Model Number (1,2,3) : 1

Loading ...model1_RGB.h5


2020-05-19 14:32:14.181980: I tensorflow/core/platform/cpu_feat
h Intel(R) MKL-DNN to use the following CPU instructions in per
To enable them in non-MKL-DNN operations, rebuild TensorFlow wi
2020-05-19 14:32:14.200389: I tensorflow/core/common_runtime/pr
inter op setting: 4. Tune using inter_op_parallelism_threads fo

Press enter to choose file :
```

Anaconda Prompt - python Ghost-RGB.py
Open
data › Tomato__Bacterial_spot
Search Tomato__Bacterial_s...
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Local Disk (E:)
00a7c269-3476-4d25-b744-44d6353cd921__GCREC_Bact.Sp 5807
00b7e89a-e129-4576-b51f-48923888bff9__GCREC_Bact.Sp 6202
0a6d40e4-75d6-4659-8bc1-22f47cdb2ca8__GCREC_Bact.Sp 6247
0a22f50a-5f25-4cf6-816b-76cae94b7f30__GCREC_Bact.Sp 6103
0a1655ed-797c-4d1d-ba35-dc255d68a2ee__GCREC_Bact.Sp 3560
0a64655c-4052-4e5f-a696-2a6cf25d10c9__GCREC_Bact.Sp 6017
0aa1c431-682b-4a6e-af5f-2a7081ea2e97__GCREC_Bact.Sp 3622
0ab9c705-f29e-45ac-b786-9549b3c38f16__GCREC_Bact.Sp 3223
File name: 0a64655c-4052-4e5f-a696-2a6cf25d10c9__GCREC_Bact.Sp 6017
Open
Cancel
20 items
Type here to search
2:32 PM
19-May-20

This is a snapshot of the executed code, here user is asked to choose the image from storage.



```
To enable them in non-MKL-DNN operations, rebuild TensorFlow with the appropriate compiler flags.
2020-05-19 14:31:51.147385: I tensorflow/core/common_runtime/process_util.cc:115] Creating new thread pool with default
inter op setting: 4. Tune using inter_op_parallelism_threads for best performance.

Press enter to choose file :



####################################################################################
OMP: Info #212: KMP_AFFINITY: decoding x2APIC ids.
OMP: Info #210: KMP_AFFINITY: Affinity capable, using global cpuid leaf 11 info
OMP: Info #154: KMP_AFFINITY: Initial OS proc set respected: 0-3
OMP: Info #156: KMP_AFFINITY: 4 available OS procs
OMP: Info #157: KMP_AFFINITY: Uniform topology
OMP: Info #179: KMP_AFFINITY: 1 packages x 2 cores/pkg x 2 threads/core (2 total cores)
OMP: Info #214: KMP_AFFINITY: OS proc to physical thread map:
OMP: Info #171: KMP_AFFINITY: OS proc 0 maps to package 0 core 0 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 1 maps to package 0 core 0 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 2 maps to package 0 core 1 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 3 maps to package 0 core 1 thread 1
OMP: Info #250: KMP_AFFINITY: pid 3568 tid 14696 thread 0 bound to OS proc set 0
OMP: Info #250: KMP_AFFINITY: pid 3568 tid 16184 thread 1 bound to OS proc set 2
####################################################################################


First Guess : Tomato___Bacterial_spot
Second Guess : Tomato___Early_blight
Third Guess : Tomato___healthy

(myspace) C:\Users\Abiral Singh\Desktop\Plant Disease Pro\Main>
```

This snap is showing the final output of the system.

Chapter 5: Conclusion

5.1 Conclusion

Plant growth and health play a significant role in the economy in courtesies like India, where a huge sector of the population is dependent on agriculture. Due to many reasons like droughts, floods. Improper rain and other, a large part of crops got wasted. Since we can't control the natural factors we have to take proper care of our agriculture to minimize the losses and hence the plant health issue can't be ignored.

The system we have created can solve this problem to an extent. In our system, we have collected data set from various sources and plants. This data set is then divided into training, validation, and testing data sets. Then we developed a model that can take RGB images as input and convert them to greyscale images to analyze them. We in total had made three models that can efficiently test the data. In our second approach we implemented the same program but this time our system can also analyze RGB images to provide better results.

Here are the results of our system:

	Training Accuracy	Validation Accuracy	Underfitting
Model 1	52.6%	74%	---
Model 2	95.6%	78%	Found
Model 3	93.6%	85.5%	Resolved
Model 1 (RGB)	89.19%	91.86%	---

Our first system consists of a single epoch and it has a training efficiency of ~52% which jumps to 95% in our second model which comprises 5 epochs. Then we developed other models to obtain a better result. So, In this model, we increased the number of hidden layers, the batch size is changed to 100 and for improving the training results epochs are changed to 10. The training accuracy in this model is obtained 95 % and validation accuracy is 85.5 %. Hence we concluded that model 3 is better from both the first and second models and has satisfactorily achieved the results for us. In this model, we also have solved the problem of underfitting which was observed in models 1 and 2. We have also observed that by increasing the depth of neural network underfitting can be removed significantly.

Then we created a model that can process the RGB images as well. This model consists of three epochs and this model has trained on ~6k samples and validated on ~650 samples. The results obtained on this model are very efficient, with a training accuracy of 89% and validation accuracy of 91%.

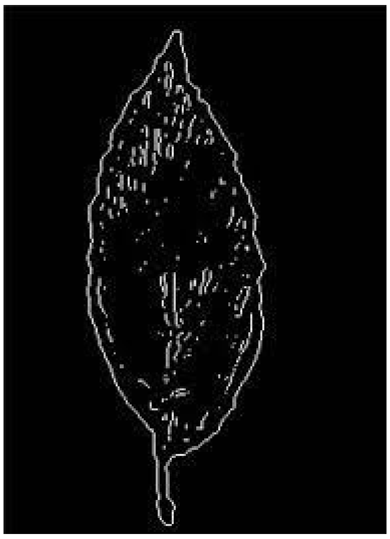
5.2 Future Scope

The system developed in this project has significantly produced the efficient results of data tested on the system. Further, In future versions, we will be thinking of adding some features like spot detection and shape/edge analysis of the image. As we all know the leaves are the place where photosynthesis takes place and hence helps a plant to make its food. A plant getting proper sunlight on their surface are more healthy. Hence the shape of leaves should be perfect to get proper sunlight.

In our feature, we are also seeking to implement a similar feature where we can test and analyze the shape of a leaf. For this, we can implement an edge detection feature that will be able to detect an edge deformity in the plant. In edge detection, we can use a canny edge detector to detect edges and it will also provide us with the control over the details that we would like to have over edges. To detect spots we can use a Blob detector which is very good in detecting the spots or sudden spikes in intensity.

A setback that we can face during this process will be that the edges can only provide an over model with the geometric data of the image while in a colored image we can express the color of the leaf, the color of spots, etc. In detecting disease color plays a vital role a leaf with health looking geometric shape will appear to be healthy in case of edge-based images, but the same image will reveal to be deficient in Nitrogen when viewed by the model trained on RGB images.

Following is a sample of a healthy leaf:



By using more similar samples we will try to get some parameters to predict the structure of the leaf and hence its health.

Along with this, we will also make a standalone executable of the program so that it can be easily accessible to all. Also, it can directly take input from the device camera and analyze it to produce quick results. This application will help farmers to get early warnings and hence they can save the plants in the early stages. Also, the application will provide a detailed description of the disease along with its curing measures. This will save a lot of time and effort. Also, we can integrate this system on the drone's eye so that a farmer can cover a larger area in a quick time.

The system will also be updated for more plants and vegetables to cover more species. We can also implement some data science algorithms to predict the disease situation or possibility so that early measures can be taken to save the crop from disease. This concept should not be restricted to the health of plants based on their leaves. We can also extend it by identifying the type of bugs that are found in fields and can give a farmer suggestions about how to tackle a particular bug, whether it is harmful or not. We can also give suggestions to the farmer based on the vegetable they will be planting and how to increase the productivity of the farm.

REFERENCES

- Surender Kumar, Rupinder Kaur, “Plant Disease Detection using Image Processing A Review”, International Journal of Computer Applications, Volume 124 – No.16, August 2015.
- Sushil R. Kamlapurkar, “Detection of Plant Leaf Disease Using Image Processing Approach”, International Journal of Scientific and Research Publications, Volume 6, Issue 2, February 2016.
- M. Akila, P. Deepan, “Detection and Classification of Plant Leaf Diseases by using Deep Learning Algorithm ”, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, 2018.
- Serawork Wallelign, Mihai Polceanu, C´edric Buche, “Soybean Plant Disease Identification Using Convolutional Neural Network”, The Thirty-First International Florida Artificial Intelligence Research Society Conference (FLAIRS-31), 2018.
- Jihen Amara, Bassem Bouaziz, Alsayed Algergawy, “A Deep Learning-based Approach for Banana Leaf Diseases Classification”, B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband, Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017
- Saradhambal.G, Dhivya.R, Latha.S, R.Rajesh, “PLANT DISEASE DETECTION AND ITS SOLUTION USING IMAGE CLASSIFICATION”, International Journal of Pure and Applied Mathematics, Volume 119 No. 14 2018
- K. Renugambal, B. Senthilraja, “Application of Image Processing Techniques in Plant Disease Recognition”, International Journal of Engineering Research & Technology (IJERT),ISSN: 2278-0181, Vol. 4 Issue 03, March-2015
- P. Krithika and S. Veni, “Leaf Disease Detection on Cucumber Leave Using Multiclass Support Vector Machine”, IEEE WiSPNET 2017 conference.

- http://www.fao.org/india/fao-in-india/india-at-a-glance/en/
- International Journal of Agriculture and Food Science Technology. ISSN 2249-3050, Volume 4, Number 4 (2013), pp. 343-346 (GDP)

Sprash

ORIGINALITY REPORT

17%	8%	9%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Machine Learning Based Recognition of Crops Diseases By CNN", International Journal of Innovative Technology and Exploring Engineering, 2019 Publication	2%
2	Submitted to Rose Hill Jr. High School Student Paper	2%
3	www.lianjun12.com Internet Source	2%
4	btw2017.informatik.uni-stuttgart.de Internet Source	1%
5	www.ijert.org Internet Source	1%
6	en.wikipedia.org Internet Source	1%
7	P. Krithika, S. Veni. "Leaf disease detection on cucumber leaves using multiclass Support Vector Machine", 2017 International Conference on Wireless Communications, Signal	1%

Processing and Networking (WiSPNET), 2017
Publication

8	Submitted to CSU, San Jose State University Student Paper	1%
9	"Disease Identification in Chilli Leaves using Machine Learning Techniques", International Journal of Engineering and Advanced Technology, 2019 Publication	1%
10	Submitted to Monash University Student Paper	<1%
11	www.toppr.com Internet Source	<1%
12	"Advanced Computing Technologies and Applications", Springer Science and Business Media LLC, 2020 Publication	<1%
13	worldwidescience.org Internet Source	<1%
14	www.ncbi.nlm.nih.gov Internet Source	<1%
15	Zubair Akhtar. "Jharkhand and Organic Agriculture", Asian Journal of Agricultural and Horticultural Research, 2018 Publication	<1%

16	"Computer Information Systems and Industrial Management", Springer Nature, 2016 Publication	<1%
17	Submitted to University of Thessaly Student Paper	<1%
18	www.igi-global.com Internet Source	<1%
19	link.springer.com Internet Source	<1%
20	Submitted to Indian Institute of Technology, Madras Student Paper	<1%
21	Submitted to S.P. Jain Institute of Management and Research, Mumbai Student Paper	<1%
22	"Natural Language Processing and Chinese Computing", Springer Science and Business Media LLC, 2018 Publication	<1%
23	"Communications, Signal Processing, and Systems", Springer Science and Business Media LLC, 2020 Publication	<1%
24	Submitted to Houston Community College Student Paper	<1%

25	Submitted to University of Bedfordshire Student Paper	<1%
26	Submitted to Staffordshire University Student Paper	<1%
27	"Neural Information Processing", Springer Science and Business Media LLC, 2017 Publication	<1%
28	Submitted to Langston University Student Paper	<1%
29	Jie Xu, Haoliang Wei, Meng Ye, Wei Wang. "Research on Recognition Method of Zanthoxylum Armatum Rust Based on Deep Learning", Proceedings of the 2019 3rd International Conference on Computational Biology and Bioinformatics - ICCBB '19, 2019 Publication	<1%
30	Geonho Cha, Hwiyeon Yoo, Donghoon Lee, Songhwai Oh. "Light-weight semantic segmentation for compound images", 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), 2017 Publication	<1%
31	Submitted to National Institute Of Technology, Tiruchirappalli Student Paper	<1%

32	Submitted to Pasadena City College Student Paper	<1%
33	www.nationsencyclopedia.com Internet Source	<1%
34	journals.plos.org Internet Source	<1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 14 / 07 / 2020

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ | [✓ B.Tech Project] | Pape

Name: Abiral Singh **Department:** CSE **Enrolment No** 161246

Contact No. 9805113627 **E-mail.** Abiralsingh2@gmail.com

Name of the Supervisor: Dr. Yugal Kumar

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): LEAF DISEASE DETECTION USING DEEP LEARNING AND IMAGE PROCESSING

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 81
- Total No. of Preliminary pages = 6
- Total No. of pages accommodate bibliography/references = 2

Abiral Singh Digitally signed by Abiral Singh Date: 2020.07.14 14:50:19 +05'30'

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ...... (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[illegible]

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String	17	Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com