

**Department of Computer Applications**

Mini Project Report

Plant Leaf Disease Detection System

Done by

### Name Mehafil Salim

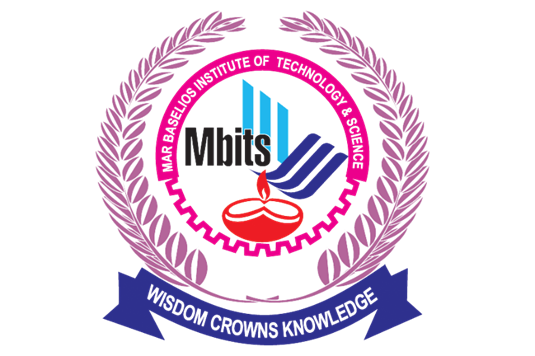
**Reg No: MBI23MCA-2027**

Under the guidance of

### Prof. Reenu Sebu

**2023-2024**

# CERTIFICATE



**Cyber Bully Comments Detector**

Certified that this is the bonafide record of project work done by

**Mehafil Salim**

#### Reg No: MBI23MCA-2027

##### During the academic year 2023-2024, in partial fulfilment of requirements for award of the degree,

###### Master of Computer Applications of

**APJ Abdul Kalam Technological University Thiruvananthapuram**

#### Faculty Guide Head of the Department

Prof. Reenu Sebu Prof. Reshma S

#### Project Coordinator External Examiner

Prof. Merin Joy M

# ACKNOWLEDGEMENT

First and foremost, I thank God Almighty for his divine grace and blessings in making all this possible. May he continue to lead me in the years to come.

I would like to express my special gratitude and thanks to my project guide Prof. Reenu Sebu, Assistant Professor, Department of Computer Applications for her guidance and constant supervision as well as for providing necessary information regarding the Main project & also for her support.

I am highly indebted to our project coordinator Prof. Merin M Joy Assistant Professor, Department of Computer Applications and Head of the Department Prof. Reshma S, Assistant Professor, Department of Computer Applications for her guidance and support.

I profusely thank other Professors in the department and all other staffs of MBITS, for their guidance and inspirations throughout my course of study. No words can express my humble gratitude to my beloved parents who have been guiding me in all walks of my journey. My thanks and appreciations also go to my friends and people who have willingly helped me out with their abilities.

# ABSTRACT

The exponential growth of online social media platforms has brought about new forms of interpersonal communication but has also given rise to harmful behaviors like cyberbullying. Cyberbullying, characterized by repetitive hostile and abusive language, can cause significant psychological and emotional harm to individuals, especially among adolescents and vulnerable groups. In response to this growing issue, this research proposes a robust cyberbullying detection system that leverages advanced machine learning techniques and natural language processing (NLP) to identify abusive content in real-time. By analyzing textual data from social media platforms, we aim to detect instances of bullying before they escalate, thereby contributing to safer online environments.

The system we developed employs several state-of-the-art classifiers, including support vector machines (SVM), random forests, and deep learning models like convolutional neural networks (CNN) and recurrent neural networks (RNN). These models are trained on large annotated datasets that contain various examples of cyberbullying behaviors, allowing the system to differentiate between harmful and benign interactions. To further enhance detection accuracy, we incorporate features such as sentiment analysis, user interaction patterns, and word embeddings like Word2Vec and GloVe, which provide richer semantic representations of the text. This multi-faceted approach enables the system to capture nuanced instances of cyberbullying that may otherwise go unnoticed by traditional keyword-based detection methods.

Our evaluation results show that the proposed system achieves high precision and recall in identifying cyberbullying across diverse datasets and platforms, making it a valuable tool for social media moderation and intervention efforts. By providing real-time alerts and flagging abusive content, this system has the potential to reduce the prevalence of cyberbullying and contribute to healthier online interactions. This work underscores the importance of combining technological innovation with ethical considerations to tackle one of the most pressing challenges of the digital age.

Dataset URL :

**LIST OF TABLES**

2.2 Summary Table 5

# LIST OF FIGURES

* + - 1. Pepper bell bacterial spot 8
      2. Pepper bell healthy 8
      3. Potato Early blight 9
      4. Potato healthy 9
      5. Potato late blight… 10
      6. Tomato target spot 10
      7. Tomato mosaic virus… 11
      8. Tomato yellow leaf curl virus… 11
      9. Tomato bacterial spot… 12
      10. Tomato early blight 12
      11. Tomato healthy… 13
      12. Tomato late blight 13
      13. Tomato leaf mold… 14
      14. Tomato septoria leaf spot… 14
      15. Tomato spider mites two spotted spider mites… 15

3.3.1 Data Visualization… 18

* + - 1. Block Diagram of Resnet-50… 19
      2. Skip Connection… 21
      3. Identity Block 21
      4. Convolution Block 22
      5. Max Pooling Layer… 23
      6. Fully Connected Layer… 24
      7. Convolution layer working with a stride of 1 26
      8. Working of Stride… 26
      9. Padding 28
      10. Working of Convolution Layer 29
      11. Graph of Leaky ReLU… 30
      12. Graph or ReLU… 30
      13. Dimension Table 31

3.5 Project Pipeline 32

* + - 1. Training code implementation… 41
      2. Training Epochs… 41

4.1.3.1 Testing 42

* 1. Confusion Matrix… 43
  2. Classification Report 44
  3. Accuracy Graph… 44
  4. Loss Graph… 45
  5. UI Design 46
  6. Result of Image Feed… 46
  7. Result of Real Time Image… 41

7.1 Git History… 48

# CONTENTS

###### [Introduction 1](#_bookmark0)

1. [**Supporting Literature 2**](#_bookmark1)
   1. [Literature Review 2](#_bookmark2)
   2. [Findings and Proposals 6](#_bookmark3)

###### [System Analysis](#_bookmark4) 7

* 1. Analysis of Dataset 7
     1. [About the Dataset](#_bookmark5) 7
     2. [Explore the Dataset](#_bookmark6) 7
  2. [Data Preprocessing](#_bookmark7) 16
  3. [Data visualization 17](#_bookmark8)
  4. Analysis of Architecture 19
     1. Block Diagram 19
     2. Description of Each Layers 21
     3. Proposed Algorithm 31
  5. [Project Pipeline 32](#_bookmark9)
  6. [Feasibility Analysis](#_bookmark10) 34
     1. [Technical Feasibility](#_bookmark11) 34
     2. Economic Feasibility 34
     3. [Operational Feasibility](#_bookmark12) 35
  7. [System Environment 35](#_bookmark13)
     1. [Software Environment 35](#_bookmark14)
     2. [Hardware Environment 39](#_bookmark15)

1. [System Design 40](#_TOC_250012)
   1. [Model Building 40](#_TOC_250011)
      1. [Model Planning 40](#_TOC_250010)
      2. [Training 40](#_TOC_250009)
      3. [Testing 42](#_TOC_250008)
2. [Results and Discussion 43](#_TOC_250007)
3. Model Deployment 46
4. [Git History 48](#_TOC_250006)
5. [Conclusions 49](#_TOC_250005)
6. [Future Work 50](#_TOC_250004)
7. [Appendix 51](#_TOC_250003)
   1. [Minimum Software Requirements 51](#_TOC_250002)
   2. [Minimum Hardware Requirements 51](#_TOC_250001)
8. [References 52](#_TOC_250000)
9. **INTRODUCTION**

The rapid expansion of online social media platforms has revolutionized the way people communicate and connect. While these platforms offer numerous benefits, they have also become breeding grounds for harmful behaviors, with cyberbullying emerging as a pervasive issue. Cyberbullying, characterized by repetitive and hostile language intended to harm, is particularly damaging to adolescents and vulnerable individuals. The psychological and emotional consequences of such online abuse can be profound, leading to anxiety, depression, and, in extreme cases, self-harm.

Given the scale and complexity of content on social media, detecting instances of cyberbullying in real-time has become a critical challenge. Traditional approaches, often based on simple keyword filtering, fail to capture the subtleties and context of harmful interactions. To address this issue, advanced computational methods are needed that can accurately and efficiently identify abusive behavior.

In this study, we propose a sophisticated cyberbullying detection system that leverages state-of-the-art machine learning techniques and natural language processing (NLP). By analyzing textual data from social media, our system aims to detect instances of cyberbullying before they escalate, contributing to safer and more supportive online environments. This research explores various models, including support vector machines (SVM), random forests, and deep learning architectures such as convolutional neural networks (CNN) and recurrent neural networks (RNN), combined with advanced features like sentiment analysis, user interaction patterns, and word embeddings (Word2Vec and GloVe). The goal is to develop a system that not only recognizes blatant abuse but also captures more subtle and context-dependent instances of bullying behavior.

## SUPPORTING LITERATURE

### Literature Review

**PAPER 1**: Shrestha, Garima, Majolica Das, and Naiwrita Dey, “Plant disease detection usingCNN” 2020 IEEE Applied Signal Processing Conference (ASPCON), IEEE, 2020.

Agricultural productivity is a key component of Indian economy. Therefore the contribution of food crops and cash crops is highly important for both the environment and human beings. Every year crops succumb to several diseases. Due to inadequate diagnosis of such diseases and not knowing symptoms of the disease and its treatment many plants die. This study provides insights into an overview of the plant disease detection using different algorithms. A CNN based method for plant disease detection has been proposed here. Simulation study and analysis is done on sample images in terms of time complexity and the area of the infected region. It is done by image processing technique. The study focuses on plant disease detection in agricultural crops, which is crucial for the Indian economy and the well-being of both the environment and human beings. The researchers propose a CNN (Convolutional Neural Network) based method for plant disease detection, utilizing image processing techniques.The study conducts a simulation and analysis using sample images, considering factors such as time complexity and the area of the infected region. A total of 15 cases are used for testing, including 12 cases of diseased plant leaves and 3 cases of healthy leaves. The diseased leaves consist of various diseases such as Bell Paper Bacterial Spot, Potato Early Blight, Potato Late Blight, Tomato Target Spot, Tomato Mosaic Virus, Tomato Yellow Leaf Curl Virus, Tomato Bacterial Spot, Tomato Early Blight, Tomato Late Blight, Tomato Leaf Mold, Tomato Septoria Leaf Spot, and Tomato Spider Mites.The proposed CNN model achieves a test accuracy of 88.80%, indicating its effectiveness in detecting plant diseases. Additionally, different performance metrics are derived for the model, which further assess its capabilities in disease detection. However, the specific performance metrics and their values are not mentioned in the summary.Overall, the study provides insights into the application of algorithms and image processing techniques for plant disease detection, highlighting the potential of the proposed CNN-based approach in this domain.

**PAPER 2** : Asif, Md Khalid Rayhan, Md Asfaqur Rahman, and Most Hasna Hena. “CNN based disease detection approach on potato leaves”,2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), IEEE, 2020.

Potatoes are a well-known vegetable to all of us. If the other countries are taken into consideration, it can be easily concluded that potatoes are the number one vegetable all over the world, which has been increasingly claimed by many Agricultural departments. Despite the hype, potato leaf disease causes significant damage to the potatoes. Various types of diseases such as early blight, late blight, septoria blight etc. will attack potato plants and exhibit their syndrome in the leaf of these disorders. The farmer would not face incurring major economic losses if these outbreaks are detected at the primary stage and sufficient action is taken. The proposed model will strongly identify and detect diseases of potato leaf stand on image processing methods in this research paper. The study focuses on potato leaf diseases, which can cause significant damage to potato crops worldwide. Diseases such as early blight, late blight, and septoria blight can affect potato plants and manifest symptoms in their leaves. Detecting these diseases at an early stage and taking appropriate action is crucial to avoid major economic losses for farmers. The research paper proposes a model that utilizes image processing methods to accurately identify and detect potato leaf diseases. Among various machine learning algorithms, the Convolutional Neural Network (CNN) model is employed due to its effectiveness in image classification tasks. Five algorithms are used in the study: AlexNet, VggNet, ResNet, LeNet, and a Sequential model, with the Sequential model being the proposed one.The model is trained using images of both normal and diseased potato leaves. Through the application of the algorithms, the model analyzes these images and classifies potato plant leaves as either diseased or normal. The provided model achieves a high precision rate of 97%, indicating its strong performance in accurately identifying potato leaf diseases.In summary, the research paper focuses on potato leaf diseases and proposes a model based on CNN and image processing methods for disease detection. The model achieves a precision rate of 97% and demonstrates the potential to effectively differentiate between normal and diseased potato leaves.

**PAPER 3** : Ajra, Husnul, Mst Khairun Nahar, Lipika Sarkar, and Md Shohidul Islam. “Disease Detection of Plant Leaf using Image Processing and CNN with Preventive Measures” ,In 2020 Emerging Technology in Computing, Communication and Electronics (ETCCE), pp.1-6. IEEE, 2020.

Agriculture is a very significant field for increasing population over the world to meet the basicneeds of food. Meanwhile, nutrition and the world economy depend on the growth of grains and vegetables. Many farmers are cultivating in remote areas of the world with the lack of accurate knowledge and disease detection, however, they rely on manual observation on grains and vegetables, as a result, they are suffering from a great loss. Digital farming practices can be an interesting solution for easily and quickly detecting plant diseases. To address such issues,this paper proposes plants leaf disease detection and preventive measures technique in the agricultural field using image processing and two well-known convolutional neural network (CNN) models as AlexNet and ResNet-50. Firstly, this technique is applied on Kaggle datasets of potato and tomato leaves to investigate the symptoms of unhealthy leaf. Then, the feature extraction and classification process are performed in dataset images to detect leaf diseases using AlexNet and ResNet- 50 models with applying image processing. The experimental results elicit the efficiency of the proposed approach where it achieves the overall 97% and 96.1% accuracy of ResNet- 50 and the overall 96.5% and 95.3% accuracy of AlexNet for the classification of healthy-unhealthy leaf and leaf diseases, respectively. Finally, a graphical layout is also demonstrated to provide a preventive measures technique for the detected leaf diseases and to acquire a rich awareness about plant health.

### Summary Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **TITLE** | **AUTHORS** | **PUBLICATION**  **DETAILS** | **FINDINGS** |
| 1 | Plant disease detection using CNN | Shrestha, Garima, Majolica Das, and Naiwrita Dey. | 2020 IEEE Applied Signal Processing Conference (ASPCON). IEEE, 2020. | * CNN based method for plant disease detection. * Kaggle dataset of tomato, potato and bellpepper are used. * Test accuracy is obtained as 88.8%. |
| 2 | CNN  based disease detection approach on potato leaves | .Asif, Md Khalid Rayhan, Md Asfaqur Rahman, and Most Hasna Hena. | 2020 3rd  International Conference on Intelligent Sustainable Systems (ICISS). IEEE, 2020. | * CNN method is used for both image classification and disease detection. * There are 5 algorithms are used namely AlexNet,ResNet,VggNet, LeNet & Sequential model. * Kaggle dataset of potato isused. * Sequential model established 97% of great precision. |
| 3 | Disease Detection of Plant Leaf  using Image Processing and CNN with Preventive Measures | Ajra, Husnul, Mst Khairun Nahar, Lipika Sarkar, and Md Shohidul Islam | In 2020 Emerging Technology in Computing, Communication and Electronics (ETCCE), pp. 1-6.  IEEE, 2020. | Uses image processing techniques aplplied on leaves and CNN for diseasedetection.   * AlexNet and ResNet 50 are the algorithms used. * Overall accuracy through ResNet 50 is 96.1% and for AlexNet is 95.3 %. * From all comparisons the performance of ResNet 50 is better than AlexNet . * Kaggle dataset of tomato and   potato leaves are used. |

Table 2.2.1 Summary Table

### Findings and Proposals

Our research focused on the development and evaluation of a robust cyberbullying detection system, utilizing various machine learning and NLP techniques. Through extensive experimentation on large annotated datasets, several key findings emerged:

**Findings**

1. **Model Performance**: Among the classifiers tested, deep learning models, particularly recurrent neural networks (RNNs) and convolutional neural networks (CNNs), consistently outperformed traditional models like support vector machines (SVMs) and random forests. The ability of RNNs to capture sequential dependencies in text and CNNs' effectiveness in detecting patterns within short textual segments proved critical in identifying abusive behaviour.
2. **Word Embeddings**: The use of word embeddings such as Word2Vec and GloVe significantly improved detection accuracy compared to traditional feature representations (e.g., bag-of-words or TF-IDF). These embeddings allowed the models to capture semantic similarities between words, enabling better detection of nuanced forms of cyberbullying, even when the exact abusive terms were not present.
3. **Sentiment and Interaction Features**: Integrating sentiment analysis and user interaction patterns into the models contributed to higher precision in identifying harmful content. Sentiment analysis helped to distinguish between negative and aggressive language, while user interaction patterns provided additional context, flagging interactions that showed repeated hostile behaviour over time.
4. **Real-time Detection**: Our system demonstrated the capability to analyze textual data in real-time, flagging abusive content as it emerged. This ability to provide immediate alerts is critical for platforms seeking to intervene early and prevent the escalation of cyberbullying incidents.
5. **False Positives and Context Sensitivity**: While the system achieved high accuracy, challenges remain with false positives, particularly in distinguishing between playful banter and actual abusive behaviour. Further refinement is needed to improve context sensitivity, especially in diverse cultural and linguistic contexts where expressions of humour or sarcasm may be misinterpreted as bullying.

**Proposals**

1. **Model Enhancement via Contextual Embeddings**: We propose the integration of advanced contextual word embeddings like BERT (Bidirectional Encoder Representations from Transformers), which consider the context of a word within a sentence, to further reduce false positives and enhance the system’s ability to understand subtle abusive language in its proper context.
2. **Multi-modal Data Analysis**: In future iterations of the system, we recommend expanding beyond textual data by incorporating other forms of communication such as images, videos, and audio. Cyberbullying often involves visual elements, and multi-modal analysis could improve detection by analyzing content holistically.
3. **Personalized Detection Models**: Cyberbullying patterns can vary significantly across different age groups, communities, and cultural contexts. We propose developing personalized detection models that
4. can be fine-tuned for specific user groups or platforms, allowing for more accurate detection tailored to the characteristics of different online environments.
5. **Ethical Considerations and Transparency**: While advancing the technological capabilities of cyberbullying detection, it is equally important to prioritize ethical considerations. We propose that future development of such systems include transparency mechanisms, allowing users to understand why certain content was flagged. This transparency will help build trust and ensure that the system is not perceived as overly restrictive or biased.
6. **Collaboration with Social Media Platforms**: To maximize the impact of the system, we suggest collaboration with social media companies to integrate the detection system directly into their platforms. This partnership would ensure that cyberbullying instances are flagged in real-time, enabling moderators to intervene swiftly. Additionally, platform-specific adjustments can be made to improve the performance and accuracy of the system in different environments.

## SYSTEM ANALYSIS

### Analysis of the dataset

* + 1. **About the dataset**

I collect my datasets from Kaggle open source library platform.

* + - Dataset URL: <https://www.kaggle.com/datasets/emmarex/plantdisease>

Kaggle Plant Village dataset is a collection of images of diseased and healthy plant leaves. The images were captured using a smartphone camera with a high resolution.The dataset consists of 15 classes, where 12 classes represent different plant diseases and the remaining 3 classes represent healthy plant leaves. Each class contains a variable number of images, with a total of 20,638 images in the dataset. The plant species included in this dataset are bell-pepper, potato, and tomato.The dataset is split into a training set and a validation set, with 70% of the images in the training set and the remaining 30% in the validation set. The images are stored in JPEG format and are organized into separate folders for each class.This dataset can be used for various tasks such as classification, segmentation, and object detection. It provides a valuable resource for researchers and practitioners to develop machine learning models for plant disease diagnosis and detection.

### Explore the dataset

The dataset available at <https://www.kaggle.com/datasets/emmarex/plantdisease> is a collection of images of various plant leaves affected by different diseases. The dataset contains a total of 20638 images belonging to 15 classes of plant diseases .Each classes contain variable number of images and stored as different folders.

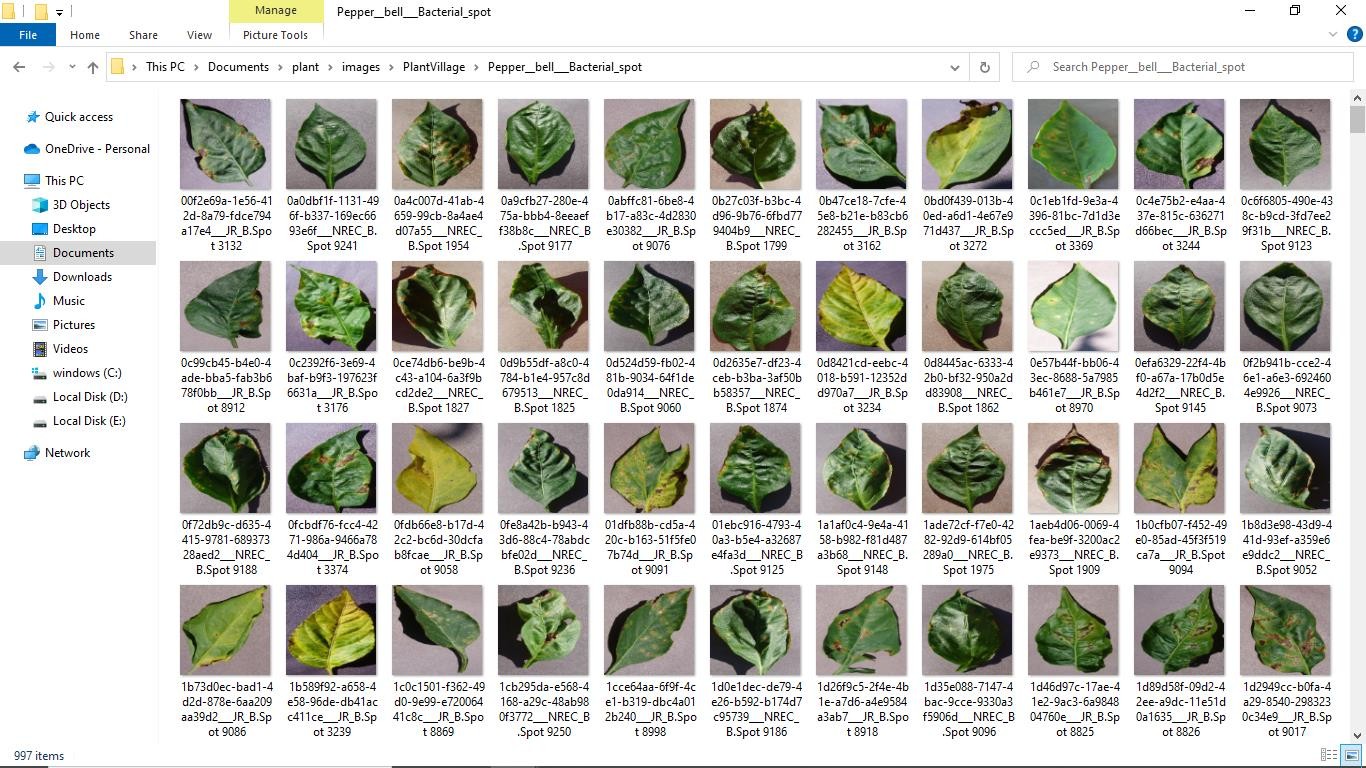


Fig 3.1.2.1 Pepper bell bacterial spot

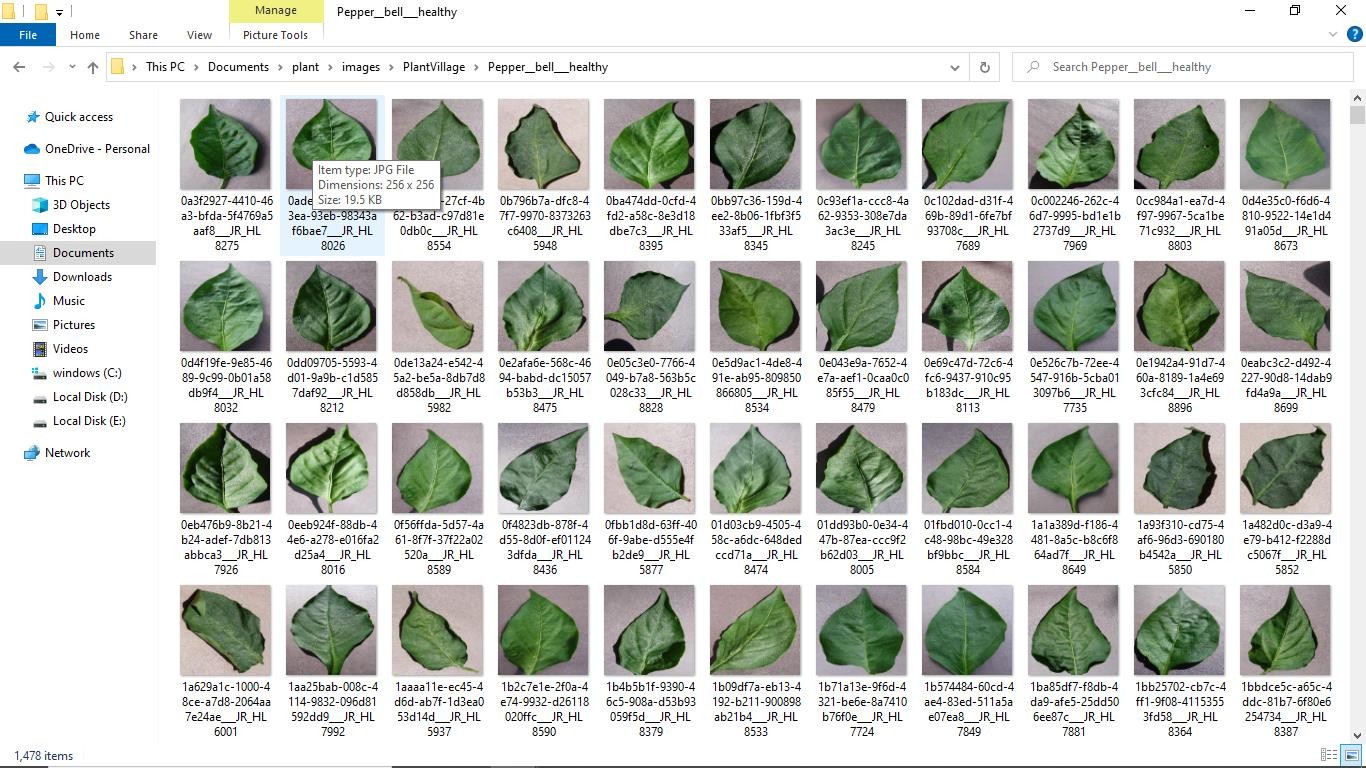


Fig 3.1.2.2 Pepper bell healthy

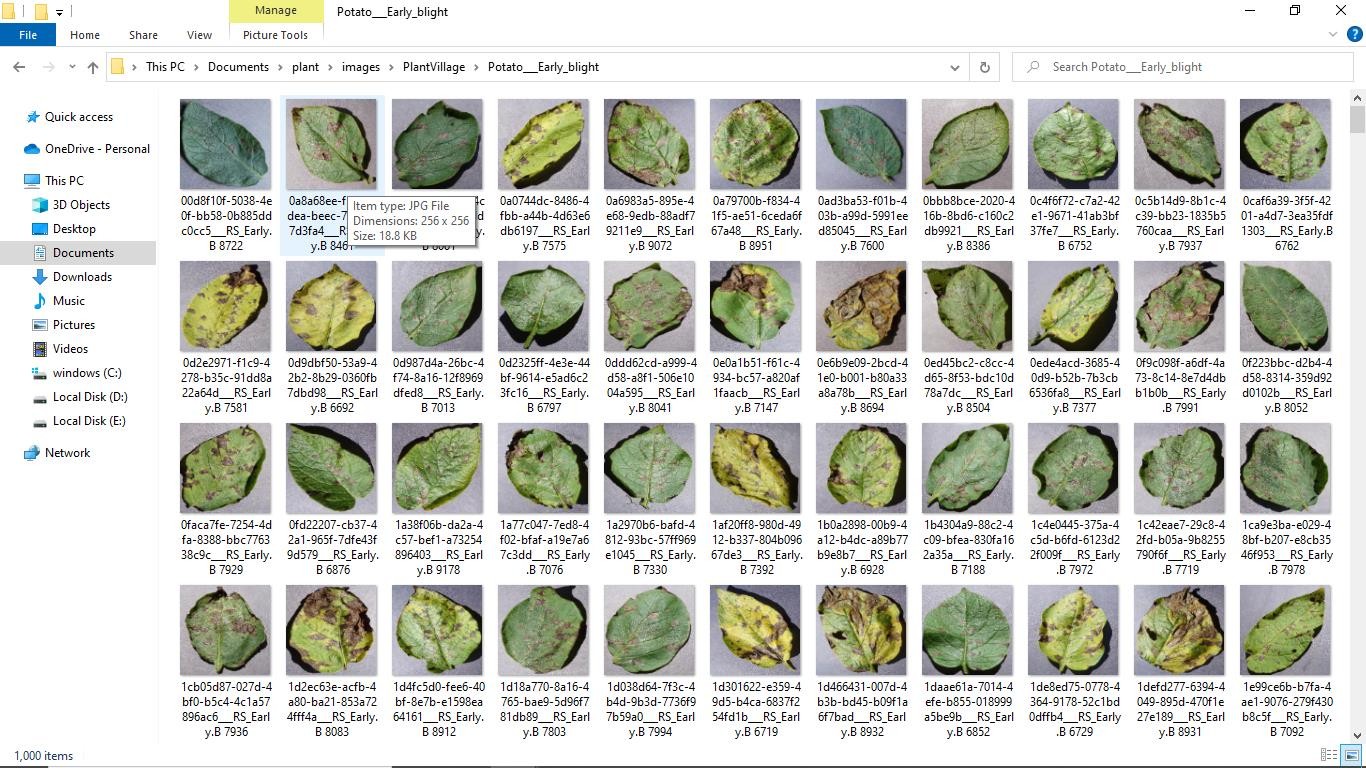


Fig 3.1.2.3 Potato Early blight

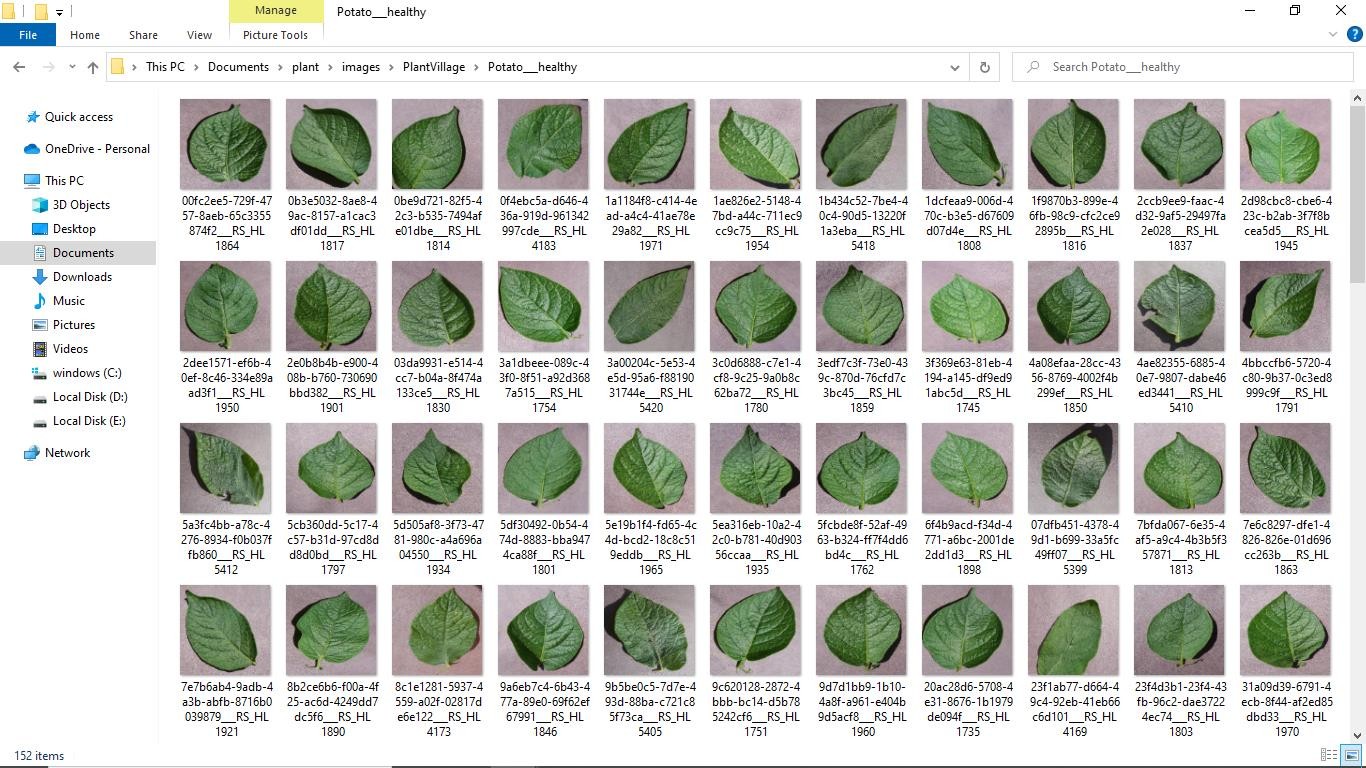


Fig 3.1.2.4 Potato healthy

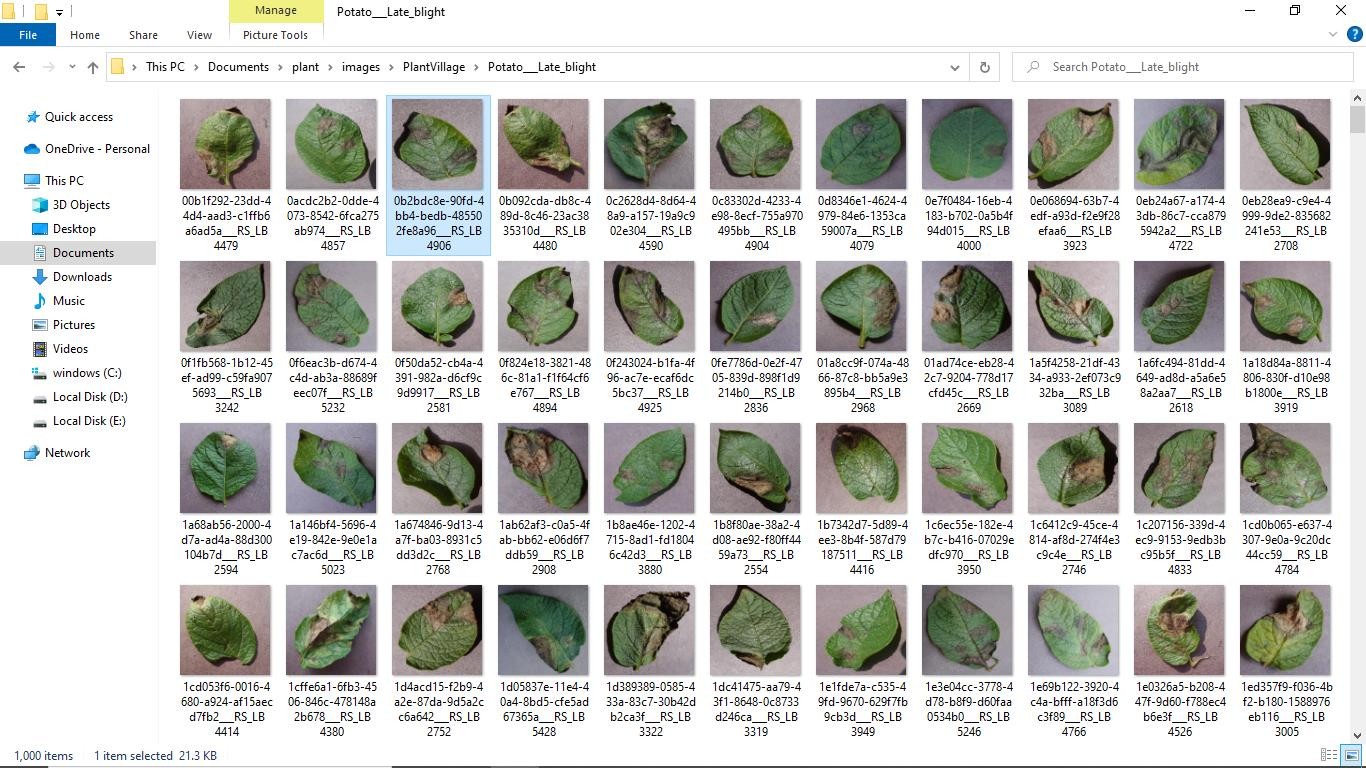


Fig 3.1.2.5 Potato late blight

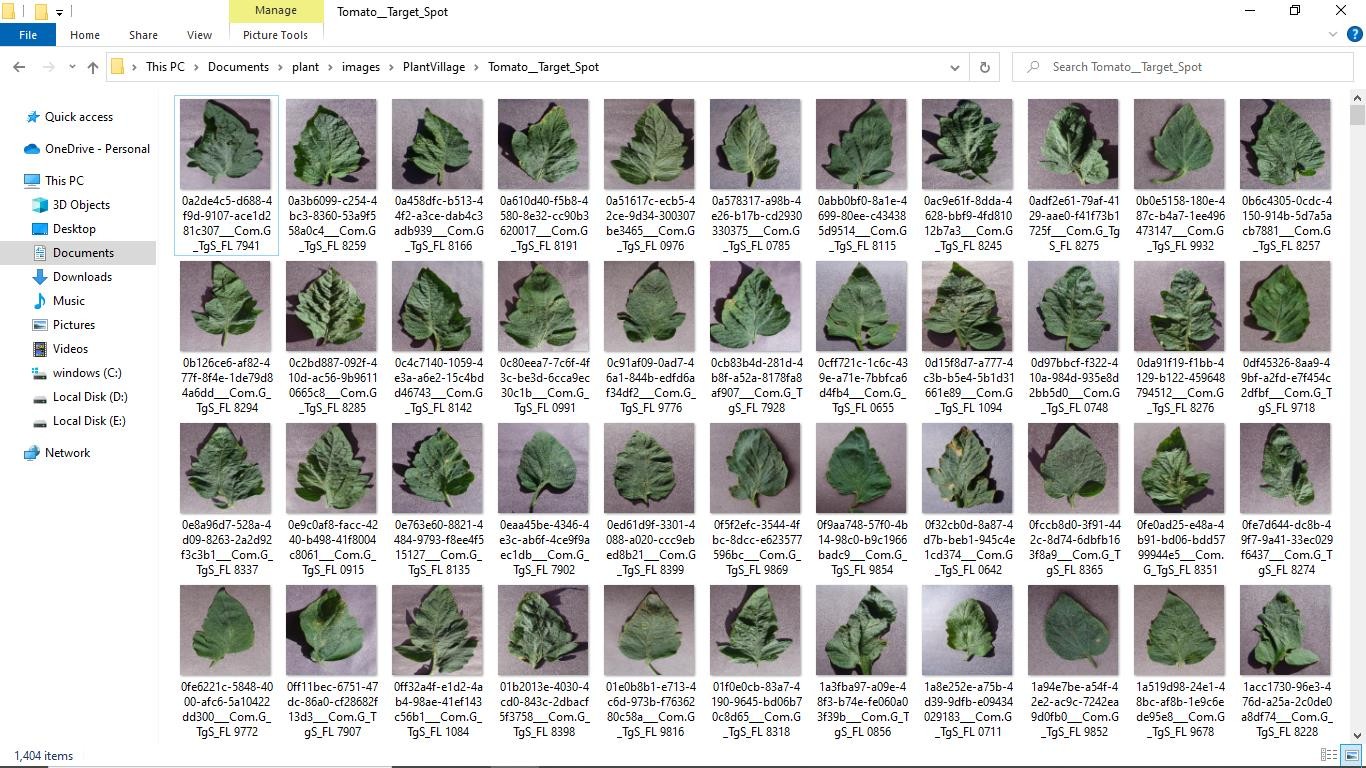


Fig 3.1.2.6 Tomato target spot

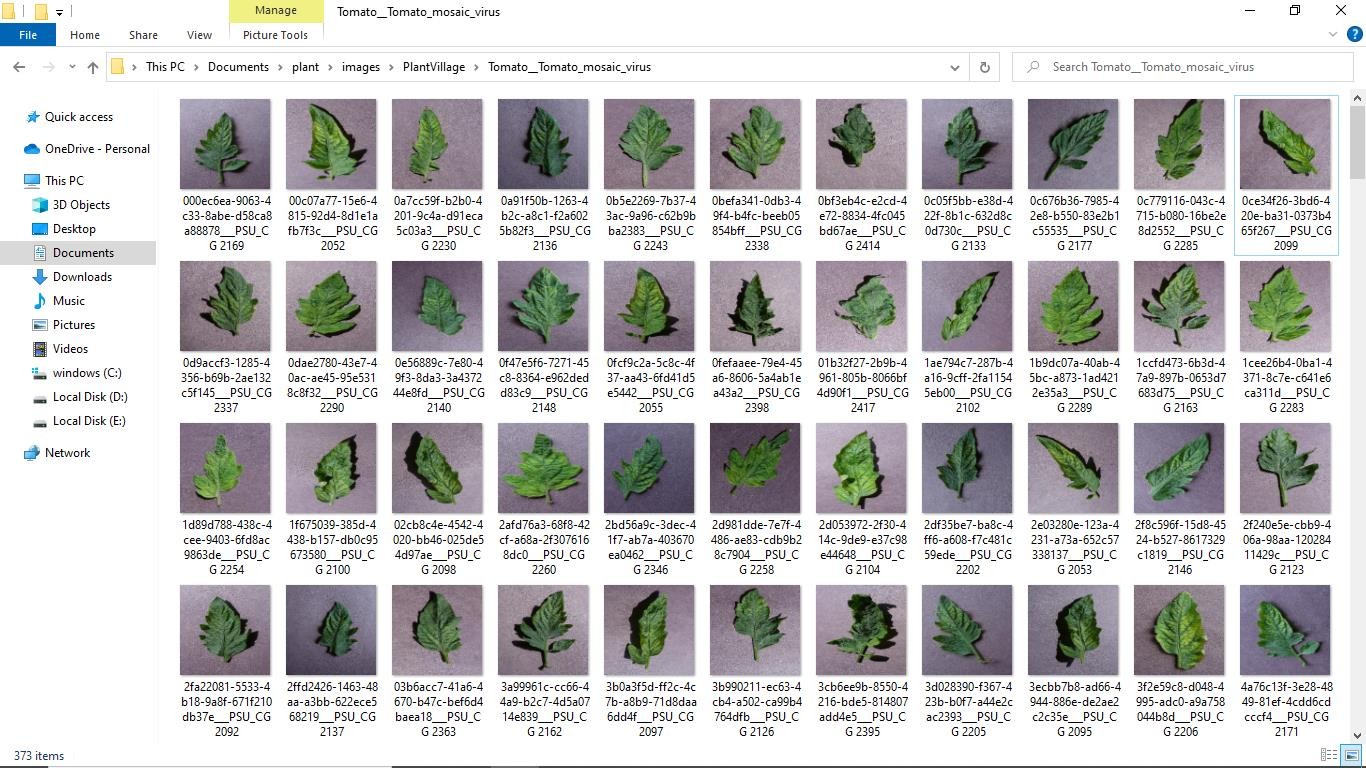


Fig 3.1.2.7 Tomato mosaic virus

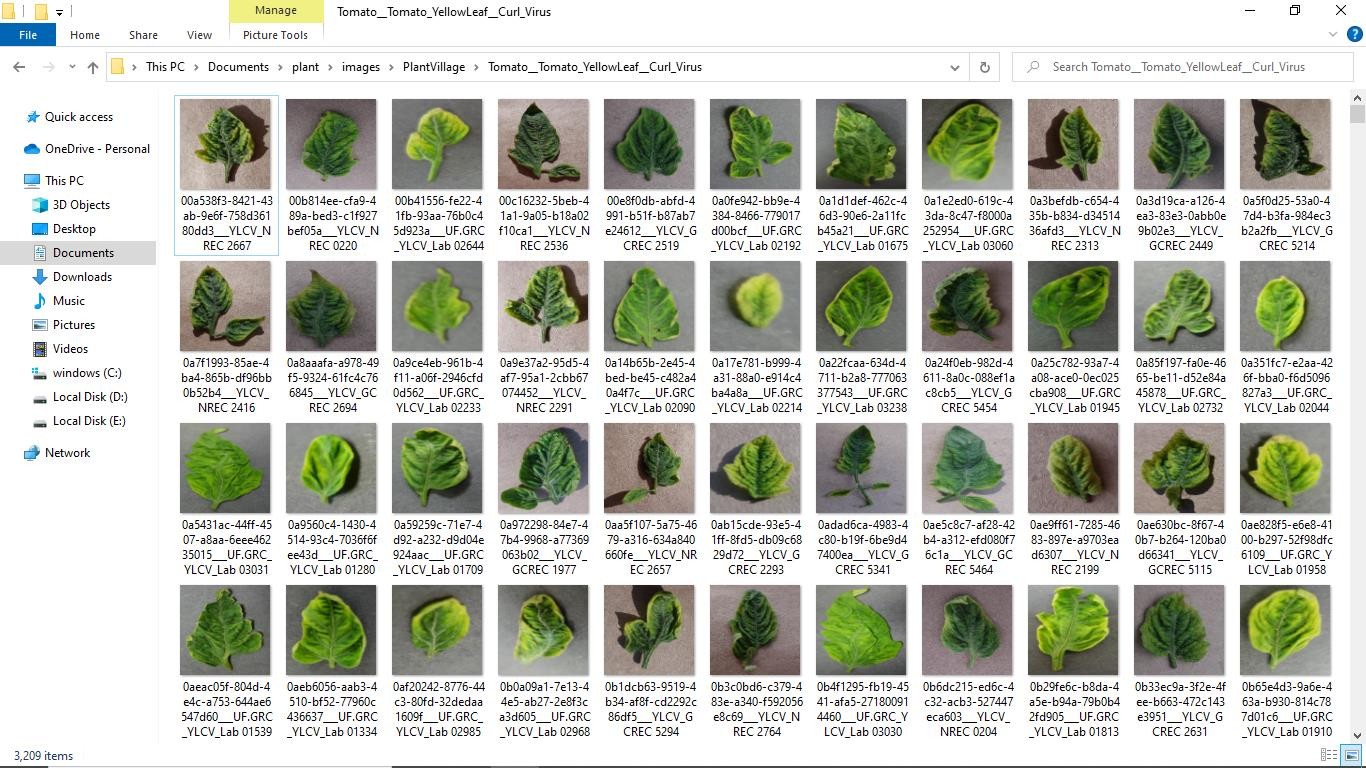


Fig 3.1.2.8 Tomato yellow leaf curl virus

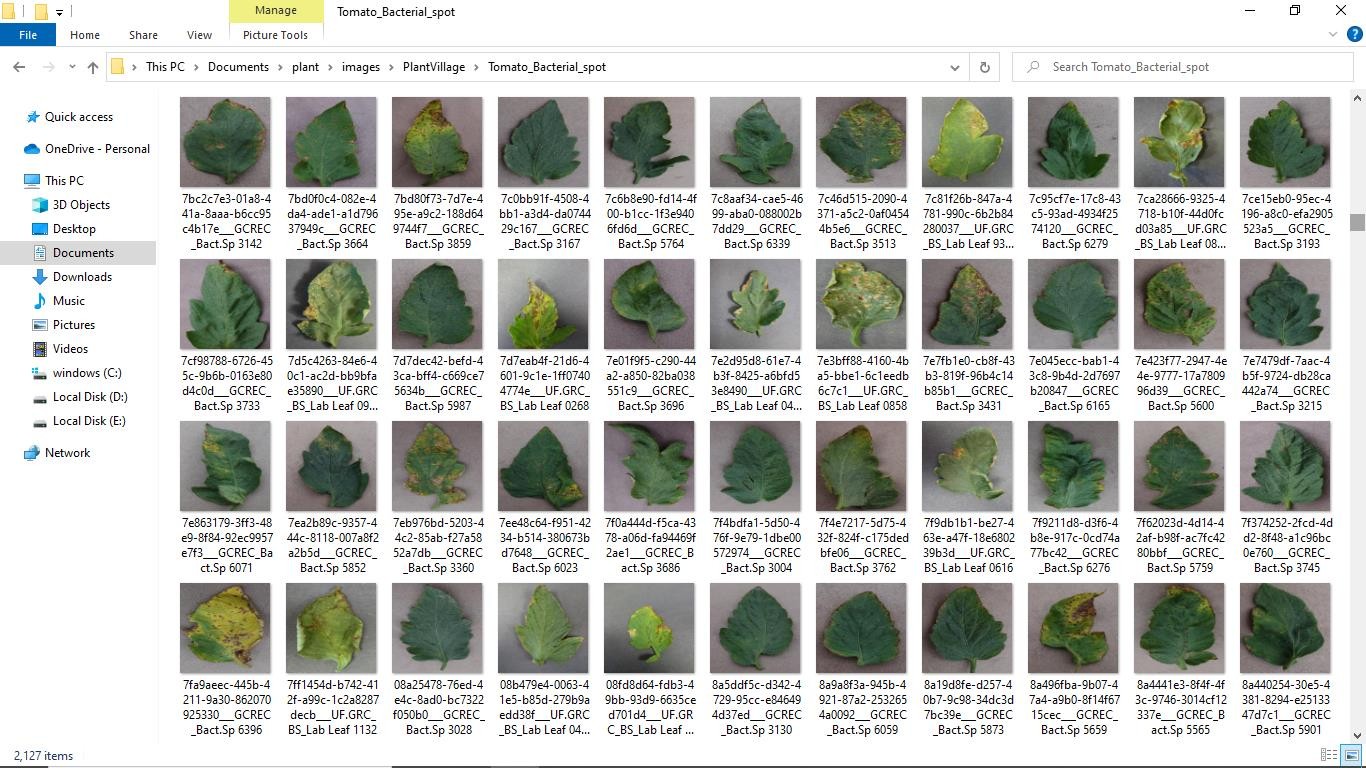


Fig 3.1.2.9 Tomato bacterial spot

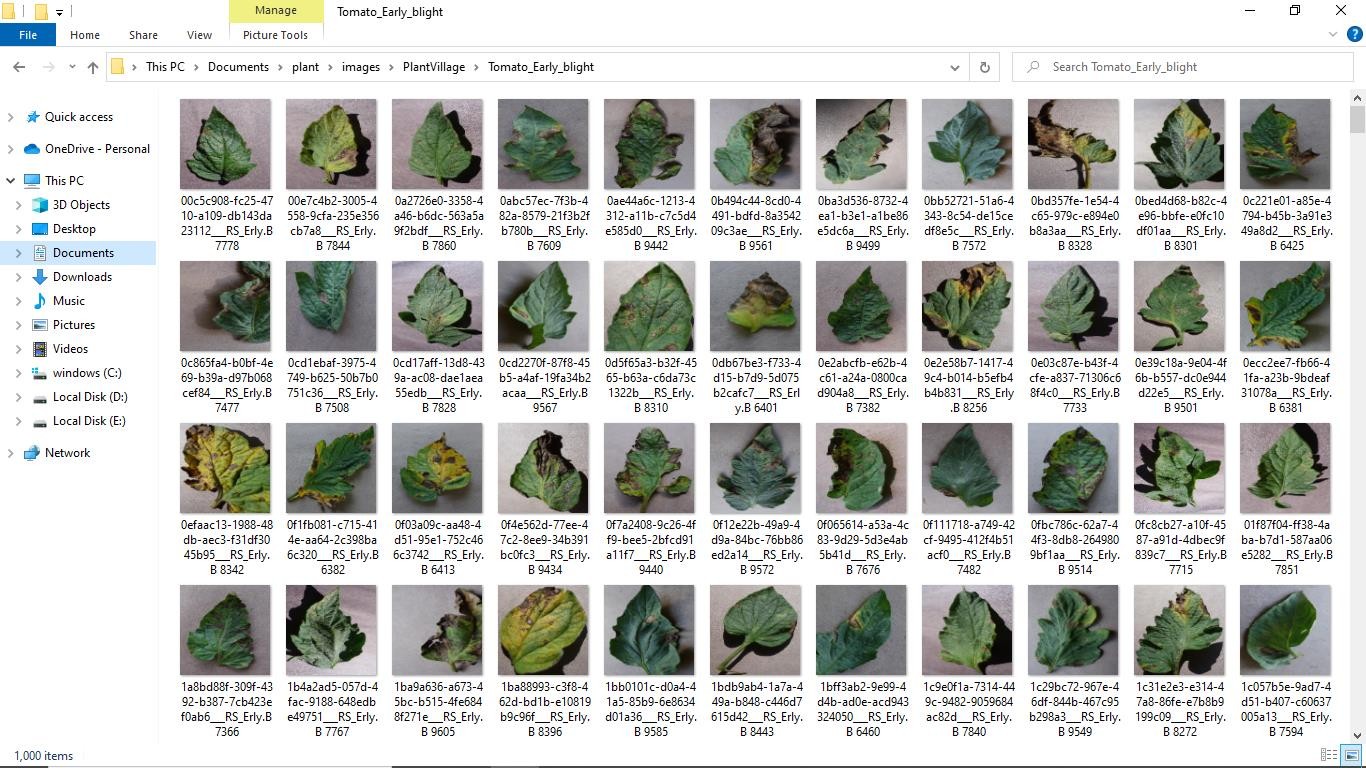


Fig 3.1.2.10 Tomato early blight

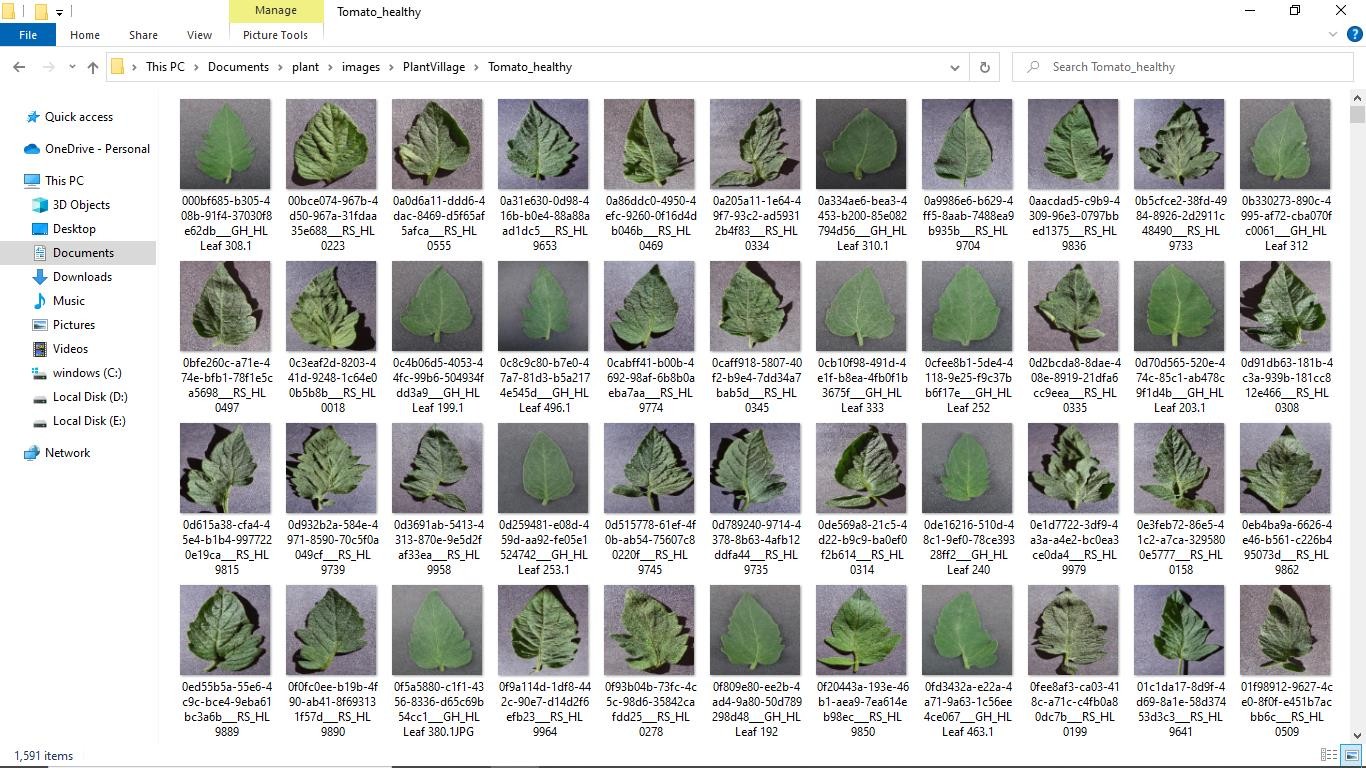


Fig 3.1.2.11 Tomato healthy

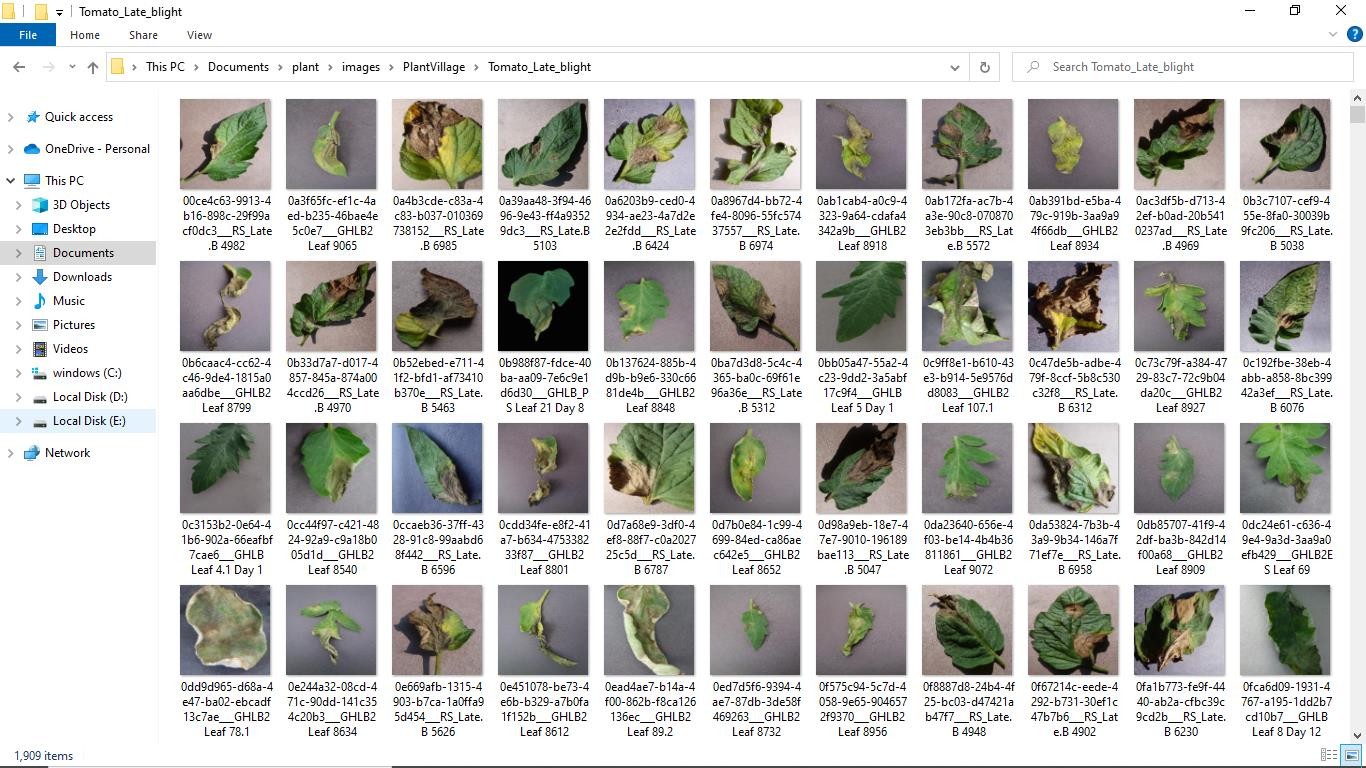


Fig 3.1.2.12 Tomato late blight

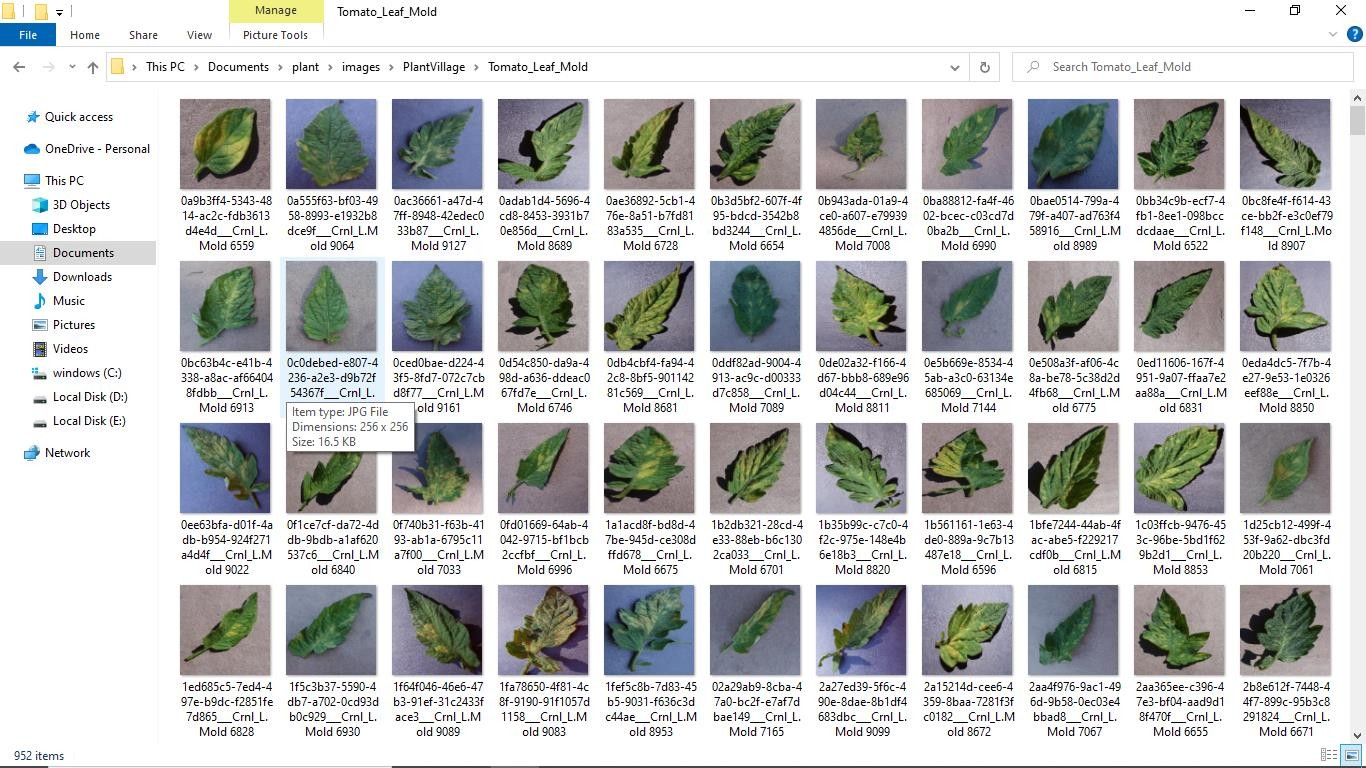


Fig 3.1.2.13 Tomato leaf mold

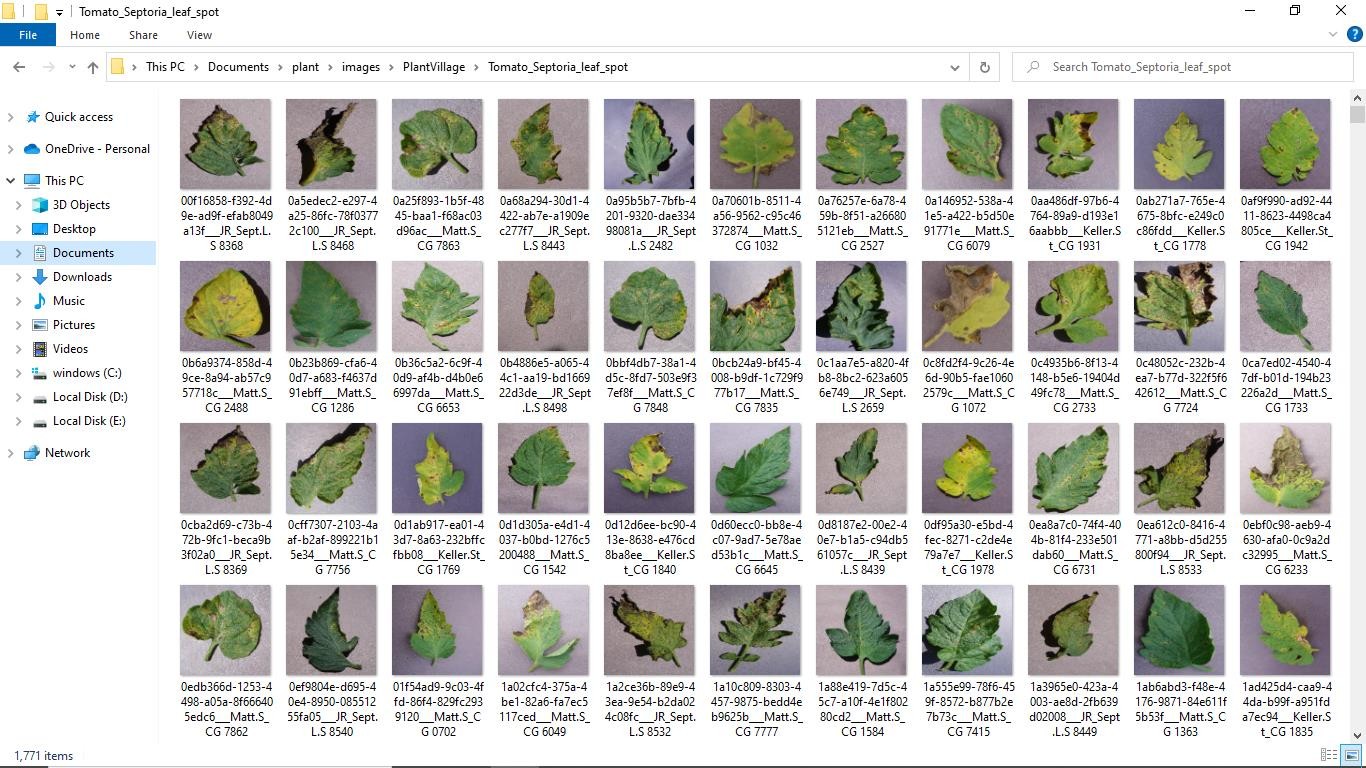


Fig 3.1.2.14 Tomato septoria leaf spot

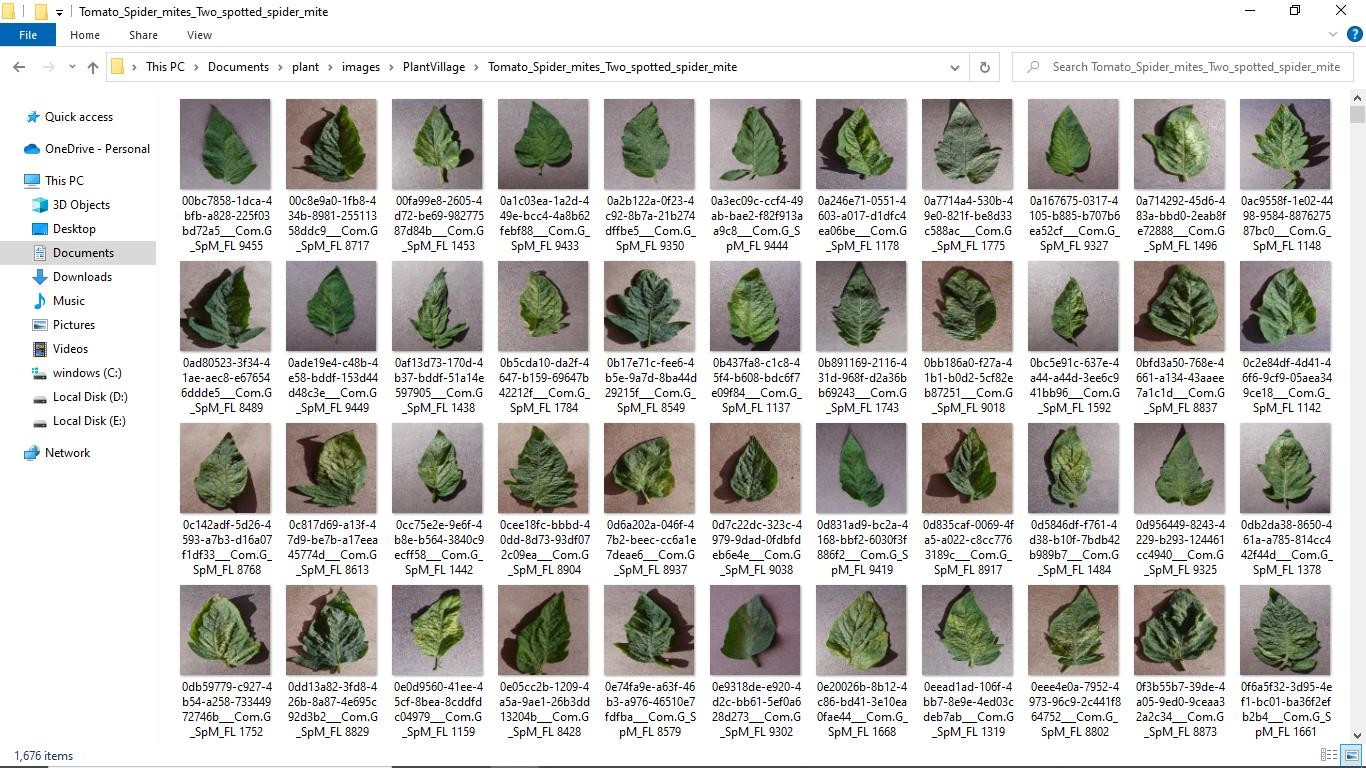


Fig 3.1.2.15 Tomato spider mites two spotted spider mites

### Data Preprocessing

* + 1. **Data Cleaning**

The plant village dataset is already preprocessed. So there is no preprocessing technique required in development phase. But in deployment phase, there is need for preprocessing the real time image captured through webcam. They are:

1. Resizing

Image resizing is made possible through OpenCV inter\_linear. To resize an image, scale it along each axis (height and width), considering the specified scale factors or just set the desiredheight and width.

When resizing an image:

* + It is important to keep in mind the original aspect ratio of the image (i.e. widthby height), if you want to maintain the same in the resized image too.
  + Reducing the size of an image will require resampling of the pixels.
  + Increasing the size of an image requires reconstruction of the image. This meansyou need to interpolate new pixels.
  + So in-order to provide it as the input to a resnet architecture,we must convert itinto a fixed size. ie, 128 x 128x 3.

### Analysis of Feature Variables

Kaggle plant village dataset is image dataset. Features of image include properties like size, dimension, resolution, color etc. Properties and values in dataset are;

Dimension - 256\*256 Width - 256

Height - 256 Horizontal Resolution - 96 dpi

Vertical Resolution - 96 dpi Bit depth - 24

### Analysis of Class Variables

* Class of the dataset
* Healthy Leaves

|  |  |  |
| --- | --- | --- |
| Bell Paper Healthy | Potato Healthy | Tomato Healthy |

* Diseased Leaves

|  |  |  |
| --- | --- | --- |
| Bell Paper Bacterial Spot | Potato Early Blight | Potato Late Blight |
| Tomato Target Spot | Tomato Early Blight | Tomato Late Blight |
| Tomato Mosaic Virus | Tomato Leaf Mold | Tomato Septoria Leaf Spot |
| Tomato Yellow Leaf Curl Virus | Tomato Spider Mites | Tomato Bacterial Spot |

### Data Visualization

Data visualization is the graphical representation of information and data in a pictorial or graphical format (Example: charts, graphs, and maps). Data visualization tools provide an accessible way to see and understand trends, patterns in data, and outliers. Data visualization tools and technologies are essential to analysing massive amounts of information and making data-driven decisions. The concept of using pictures is to understand data that has been used for centuries. General types of data visualization are charts, tables, graphs, maps, and dashboards.

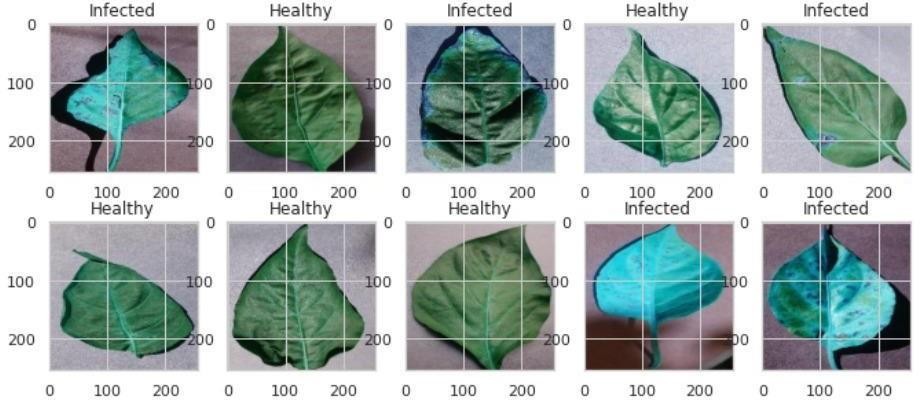
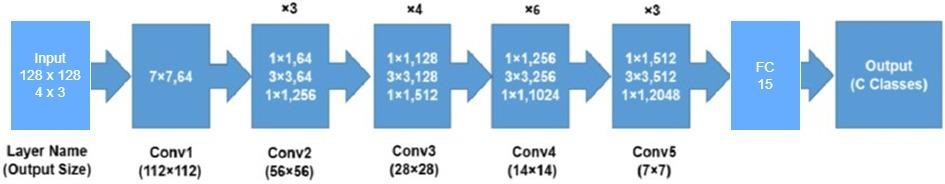


Fig 3.3.1 Dataset Visualization

### Analysis of Architecture

* + 1. **Block diagram**



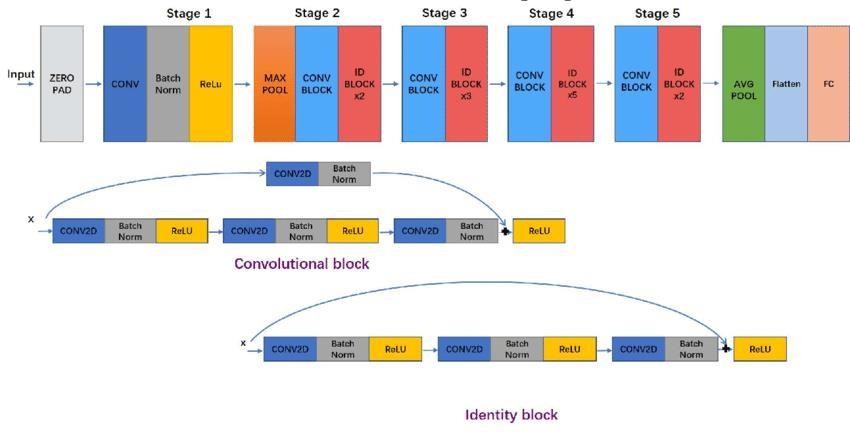


Fig 3.4.1.1 Block Diagram of Resnet-50

* A convolution with a kernel size of 7 \* 7 and 64 different kernels all with a stride of size 2 giving us **1 layer**.
* Next we see max pooling with also a stride size of 2.
* In the next convolution there is a 1 \* 1,64 kernel following this a 3 \* 3,64 kernel and atlast a 1 \* 1,256 kernel, These three layers are repeated in total 3 time so giving us **9 layers** in this step.
* Next we see kernel of 1 \* 1,128 after that a kernel of 3 \* 3,128 and at last a kernel of 1\* 1,512 this step was repeated 4 time so giving us **12 layers** in this step.
* After that there is a kernel of 1 \* 1,256 and two more kernels with 3 \* 3,256 and 1 \* 1,1024 and this is repeated 6 time giving us a total of **18 layers**.
* And then again a 1 \* 1,512 kernel with two more of 3 \* 3,512 and 1 \* 1,2048 and this was repeated 3 times giving us a total of **9 layers**.
* After that we do a average pool and end it with a fully connected layer containing 1000nodes and at the end a softmax function so this gives us **1 layer**.

Key Features of ResNet

* ResNet uses Batch Normalization at its core. The Batch Normalization adjusts the input layer to increase the performance of the network. The problem of covariate shift is mitigated.
* ResNet makes use of the Identity Connection, which helps to protect the network from vanishing gradient problem.
* Deep Residual Network uses bottleneck residual block design to increase the performance of the network.

Skip Connection

In ResNet architecture, a “shortcut” or a “skip connection” allows the gradient to be directly backpropagated to earlier layers:



Fig 3.4.1.2 Skip Connection

The image on the top shows the “main path” through the network. The image on the bottom adds a shortcut to the main path. By stacking these ResNet blocks on top of each other, you can form a very deep network. There are two main types of blocks are used in a ResNet, depending mainly on whether the input/output dimensions are the same or different.

### Description of Each Layers

* + - 1. **Identity Block** - The identity block is the standard block used in ResNets and corresponds to the case where the input activation has the same dimension as the output activation.

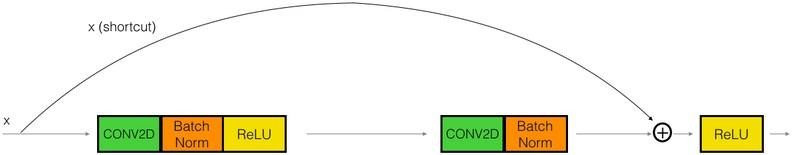


Fig 3.4.2.1 Identity block

* + - 1. **Convolutional Block –** We can use this type of block when the input and output dimensions don’t match up. The difference with the identity block is that there is a CONV layer in the shortcut path.

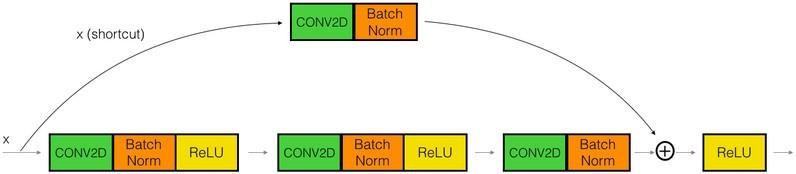


Fig 3.4.2.2 Convolution Block

1. Convolutional Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respectto the size of the filter (MxM).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

1. Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs.This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.

In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually servesas a bridge between the Convolutional Layer and the FC Layer.

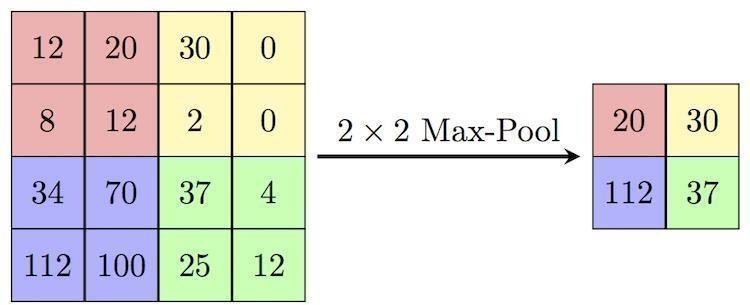


Fig 3.4.2.3 Process in max pooling layer

1. Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture. In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

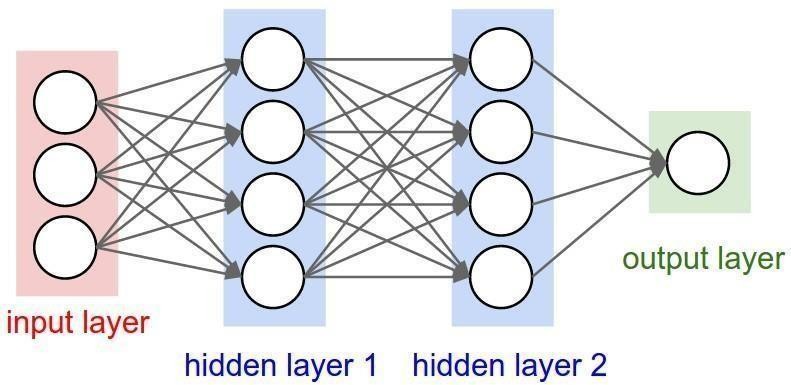


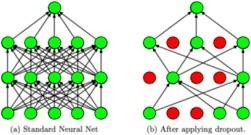
Fig 3.4.2.4 Fully Connected Layer

1. Dropout

Usually, when all the features are connected to the FC layer, it can cause overfitting

in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model’s performance when used on a new data.

To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropoutof 0.3, 30% of the nodes are dropped out randomly from the neural network.



1. Activation Functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information ofthe model should fire in the forward direction and which ones should not at the end of the network.

It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred an for a multi-class classification, generally softmax is used Before learning each layer, there are two parameters which are important in the working ofconvolutional neural network layers, stride and padding.

1. Stride

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and soon. Stride is a component of [convolutional neural networks,](https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network) or [neural](https://deepai.org/machine-learning-glossary-and-terms/neural-network) networks tuned for the compression of images and video data.

Stride is a parameter of the neural network's filter that modifies the amount of movement over the image or video. For example, if a neural network's stride is set to 1, the filter will move one pixel, or unit, at a time. The size of the filter affects the encoded output volume, so stride is often set to a whole integer, rather than a fraction or decimal.

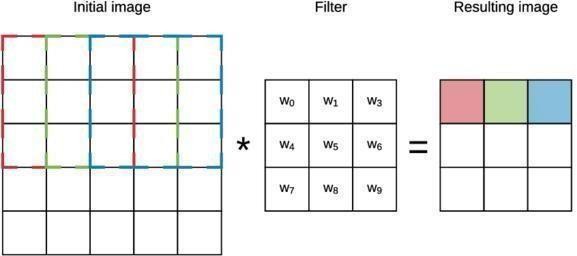


Figure 3.4.2.5 : Convolution layer working with a stride of 1

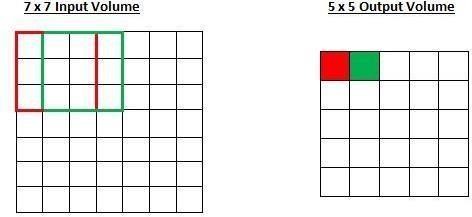


Figure 3.4.2.6 Working of stride

Imagine a convolutional neural network is taking an image and analyzing the content. If the filter size is 3x3 pixels, the contained nine pixels will be converted down to 1 pixel in the output layer. Naturally, as the stride, or movement, is increased, the resulting output willbe smaller. Stride is a parameter that works in conjunction with [padding,](https://deepai.org/machine-learning-glossary-and-terms/padding) the feature that adds blank, or empty pixels to the frame of the image to allow for a minimized reduction of size in the output layer. Roughly, it is a way of increasing the size of an image, to counteract the fact that stride reduces the size. Padding and stride are the foundational parameters of any convolutional neural network.

1. Padding

Sometimes filter does not perfectly the input image. we have two options :

* + Pad the picture with zeros (zero-padding) so that it fits.
  + Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

Padding is a term relevant to [convolutional neural networks](https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network) as it refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN. For example,if the padding in a CNN is set to zero, then every pixel value that is added will be of value zero. If, however, the zero padding is set to one, there will be a one pixel border added to

the image with a pixel value of zero. Padding works by extending the area of which a convolutional [neural](https://deepai.org/machine-learning-glossary-and-terms/neural-network) network processesan image. The kernel is the neural networks filter which moves across the image, scanning each pixel and converting the data into a smaller, or sometimes larger, format. In order to assist the kernel with processing the image, padding is added to the frame of the image to allow for more space for the kernel to cover the image. Adding padding to an image processed by a CNN allows for more accurate analysis of images.

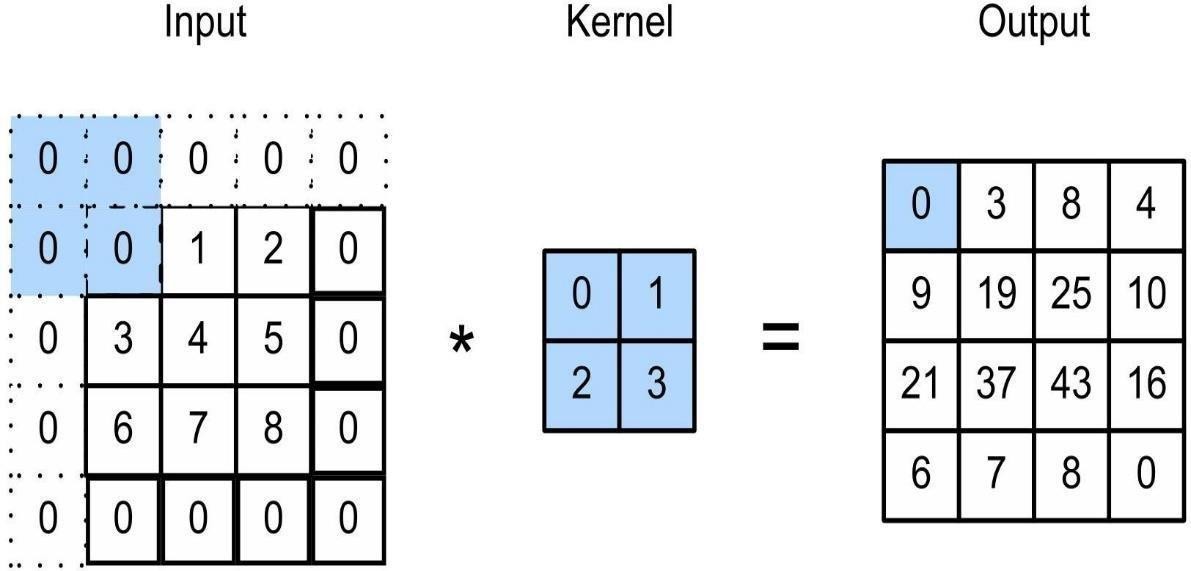


Figure 3.4.2.7 padding

The key building block in a convolutional neural network is the convolutional layer. We can visualize a convolutional layer as many small square templates, called convolutional kernels, which slide over the image and look for patterns. Where that part of the image matches the kernel’s pattern, the kernel returns a large positive value, and when there is no match, the kernel returns zero or a smaller value.

Convolution layers used trainable kernels or filters to perform convolution operations, sometimes including an optional trainable bias for each kernel. These convolution operations involved moving the kernels over the input in steps called strides. Generally, the larger the stride was, the more spaces the kernels skipped between each convolution. This led to less

overall convolutions and more miniature output size. For each placement of a given kernel, amultiplication operation was performed between the input section and the kernel, with the bias summed to the result. This produced a feature map containing the convolved result. The feature maps were typically passed through an activation function to provide input for the subsequentlayer.

Size of the feature map = [(input\_size − kernel\_size + 2 × p adding) / stride] + 1

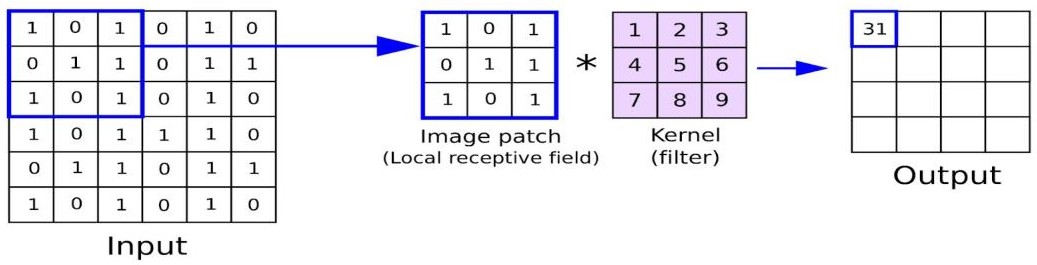


Figure 3.4.2.8 : Working of convolutional layer.

1. ReLU

The rectified linear activation function or ReLU for short is a linear function that will

output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. Here I am using Leaky ReLU.

Difference between ReLU and Leaky ReLU

* + Leaky ReLU is a type of activation function that helps to prevent the function from becoming saturated at 0. It has a small slope instead of the standard ReLU which has an infinite slope
  + Leaky ReLU is a modification of the ReLU activation function. It has the same form as the ReLU, but it will leak some positive values to 0 if they are close enough to zero.
  + it is a variant of the ReLU activation function. It uses leaky values to avoid dividing by zero when the input value is negative, which can happen with standard ReLU when training neural networks with gradient descent.

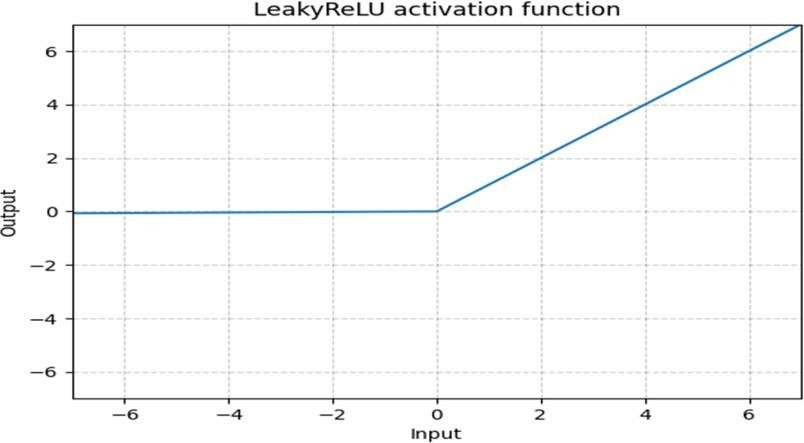


Fig 3.4.2.9 Graph of Leaky ReLU

**Softmax** is often used as the activation for the last layer of a classification network because the result could be interpreted as a probability distribution.

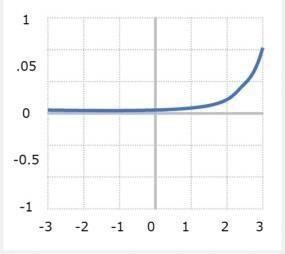


Fig 3.4.2.10 : Graph of RELU

### 3.4.3 Dimension table of the architecture

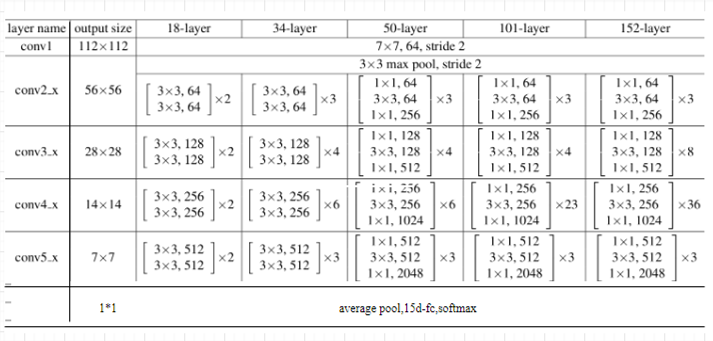


Fig 3.4.2.11 Dimension table

### Project Pipeline

4

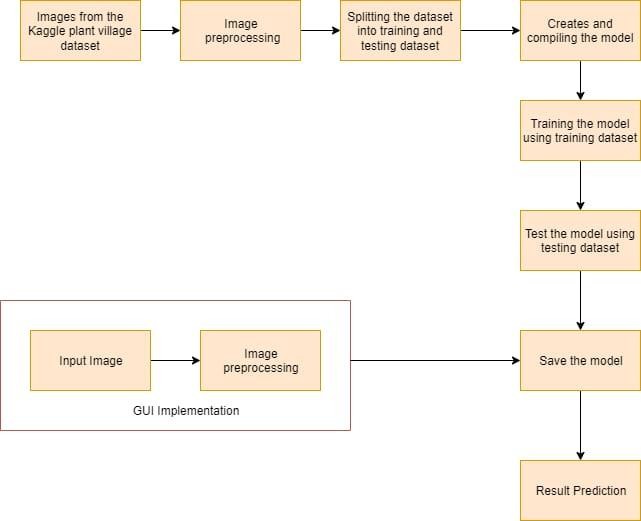


Fig 3.5 Project Pipeline

### 3.5.1 Detailed Pipeline Analysis

* **Dataset collection**: Dataset is collected from kaggle plant village dataset.
* **Image Preprocessing**: Image preprocessing is a crucial step in deep learning tasks, particularly in computer vision applications, to enhance the quality and suitability of images for training deep neural networks. Preprocessing techniques aim to standardize and optimize the input images before feeding them into the network, improving the

network's performance and efficiency.

* + Resizing: Images are often resized to a fixed size to ensure consistent dimensions across the dataset.

o Augmentation: Data augmentation techniques generate additional training samples by applying random transformations such as rotation, flipping, translation, and adding noise. This approach helps in preventing overfitting, increasing the diversity of the training set, and improving the model's generalization ability.

* **Data Splitting**: Divide the dataset into separate subsets for training, validation, and testing. The training set is used to train the deep learning model, the validation set is used to fine-tune hyperparameters and monitor model performance, and the testing set is used for the final evaluation of the trained model's generalization ability.

###### Working of the project pipeline :

* + Initially , Dataset is collected from kaggle open source python library and the dataset is plant village dataset.
  + After the dataset collection, then perform image preprocessing technique include resizing and augmentation.
  + Then the preprocessed dataset is divided into 75:25 for testing and training.
  + We created a model using Resnet-50 architecture and fed the preprocessed dataset to predict the disease classes
  + The model is then validated and evaluated
  + Next step is to save the model.
  + The project is estimated to predict real time image.

### Feasibility Analysis

A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing system or proposed system, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects for success.

Evaluated the feasibility of the system in terms of the following categories:

* Technical Feasibility
* Economical Feasibility
* Operational Feasibility

###### Technical Feasibility

Proposed system is technically feasible since all the required tools are easily available. Technical issues involved are the necessary technology existence, technical guarantees of accuracy, reliability, ease of access, data security, and aspects of future expansion.

The application is technically feasible because all the technical resources required for the development and working of the application is easily available and reliable. The project is implemented in Python. Since Python supports a various libraries and packages that make theproject development easier, the project was technically feasible. The codes are written in Google Colab, therefore all the libraries will be available, no need to install or import each ofthose. These requirements are easily available, reliable, and will make the system more time saving and require less manpower.

###### Economical Feasibility

In our proposed system ”Plant Leaf Disease Detection System”, the development cost of the application is optimum. The system requires only a computer for working. The code is working on Google Colab and Jupyter notebook , So the colab consumes an amount of internet. The development of the system will not need a huge amount of money. It will be economically feasible. But in Jupyter Notebook it needs high memory and time. But it doesn’t need internet.

###### Operational Feasibility

Operational feasibility assesses the extent to which the required system performs a series of steps to solve business problems and user requirements. Operational feasibility is mainly concerned with issues like whether the system will be used if it is developed and implemented. The developed system is completely driven and user friendly. Since the code iswritten on Google Colab, no need for worrying about importing or installing the libraries required. There is no need of skill for a new user to open this application and use it. The interface contain only a file upload option and a submit button. Users also need to be aware of the application initially. Then they can use it easily. So it is feasible. But sometimes it have a GPU issues.If we use jupyter notebook, then we must need a GPU in our system to get a faster user input and output.

* 1. **System Environment**

System environment specifies the hardware and software configuration of the new system. Regardless of how the requirement phase proceeds, it ultimately ends with the software requirement specification. A good SRS contains all the system requirements to a level of detailsufficient to enable designers to design a system that satisfies those requirements. The system specified in the SRS will assist the potential users to determine if the system meets theirneeds orhow the system must be modified to meet their needs.

###### Software Environment

Various software used for the development of this application are the following :

###### Python

Python is a high-level programming language that lets developers work quickly and integrate systems more efficiently. This model is developed by using many of the Python libraries and packages such as :

###### Pandas

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures andoperations for

manipulating numerical data and time series. This library is built on top of the NumPy library.

###### Matplotlib

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.

In this application, its used for plotting the graph.

###### Numpy

NumPy is a Python library used for working with arrays. NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays.

In this application, its used for handling arrays.

###### Tensorflow

TensorFlow is an open-source library developed by Google primarily for deep learning applications.

In this application, its used for creating and handling the model.

###### Keras

Keras is a powerful and easy-to-use free open source Python library for

developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code.

In this application, its used for creating and handling the model.

###### OS

The OS module in Python provides functions for interacting with the operating

system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system-dependent functionality.

In this application, its used for saving the model.

###### Google Colab

Colab is a free Jupyter notebook environment that runs entirely in the cloud. We can write and execute code in Python. Colab supports many machine learning libraries which can be easily loaded in the colab notebook.

###### Jupyter Notebook

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience. TheJupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installedon a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

###### Visual Studio Code

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflowsto fuller featured IDEs, such as [Visual Studio IDE](https://visualstudio.microsoft.com/)

* **HTML** and **CSS**

Hyper Text Markup Language is used for creating web pages.HTML describes the structure of the web page. Here, the user interface of my project is done using HTML. Cascading Style Sheet is used with HTML to style the web pages.

###### PHP

Here php is used to connect to the backend. PHP is an acronym for "PHP: Hypertext Preprocessor" PHP is a widely-used, open source scripting language. PHP scripts are executed on the server. PHP is free to download and use.

###### Github

Git is an open-source version control system that was started by Linus Torvalds. Git is similar to other version control systems Subversion, CVS, and Mercurial to name a few. Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute. Git is the preferred version control system of most developers, since it has multiple advantages over the other systems available.It stores file changes more efficiently and ensures file integrity better.

The social networking aspect of GitHub is probably its most powerful feature, allowing projects to grow more than just about any of the other features offered. Project revisions can be discussed publicly, so a mass of experts can contribute knowledge and collaborate to advance a project forward.

###### Flask

Flask is a small and light weight python web framework that provides useful tools and features that make creating web applications in python easier. It gives developers flexibility and is more accessible framework for new developers since you can build a web application quickly using one single Python file.

###### Hardware Environment

Selection of hardware configuration is very important task related to the software development. The hardware configuration of project done system is :

* + - * Processor : Intel core i3
      * Memory : 8 GB RAM or greater
      * Disk space : 40 GB or greater
      * GPU : Nvidia (Kaggle )
      * Good internet connectivity

## SYSTEM DESIGN

### Model Building

Model building in deep learning refers to the process of designing and developing a neural network architecture that can accurately perform a specific task. Model building involves Model Planning, Training and Testing.

#### Model Planning

Model is generated using Resnet-50 architecure which provides more accuracy. By splitting the dataset ,a portion is used for training the model and the other is used for testing the model.75% of dataset is used for training and remaining 25% used as testing data.

#### Training

Training a dataset in deep learning involves feeding the data to a deep neural network model and adjusting the model's parameters iteratively to learn from the data and make accurate predictions. The training process is typically an iterative and resource- intensive task, requiring powerful hardware (e.g., GPUs or TPUs) and significant computational resources. Deep learning frameworks like TensorFlow or PyTorch provide libraries and APIs that simplify the implementation and management of the training process.By iteratively updating the model's parameters based on the training data, deep neural networks can learn to make accurate predictions and generalize to new, unseen examples.



Fig 4.1.2.1 Training code implemenatation

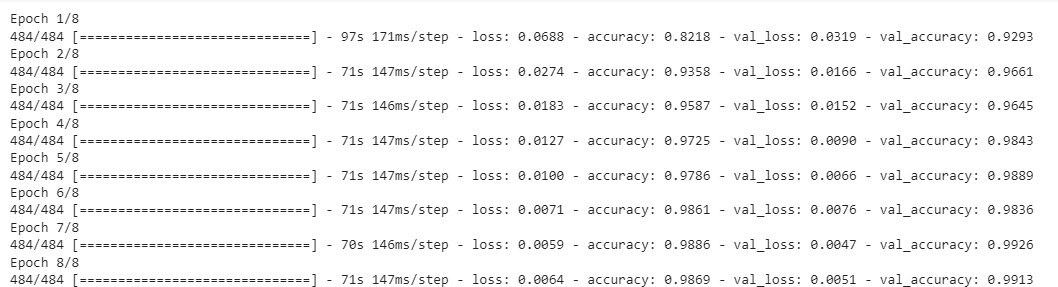


Fig 4.1.2.2 Training Epochs

#### Testing

The model testing is referred to as the process where the performance of a fully trained model is evaluated on a testing set. The testing set consisting of a set of testing samples should be separated from both training and validation sets, but it should follow the same probability distribution as the training set.

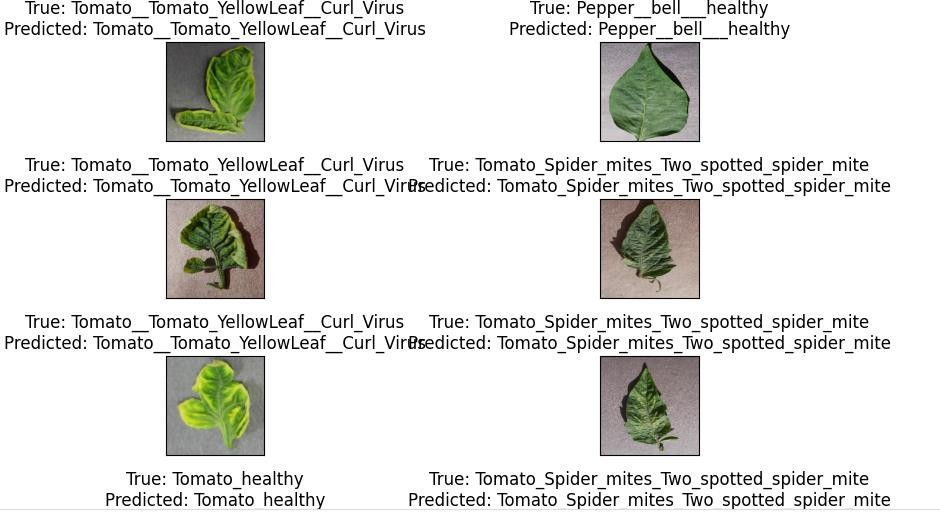


Fig 4.1.3.1 Testing

## RESULTS AND DISCUSSION

The accuracy is the metrics used in the training of the dataset. Accuracy is a measurement of observational error. It defines how close or far off a given set of measurement are to their true value.

#### Confusion Matrix

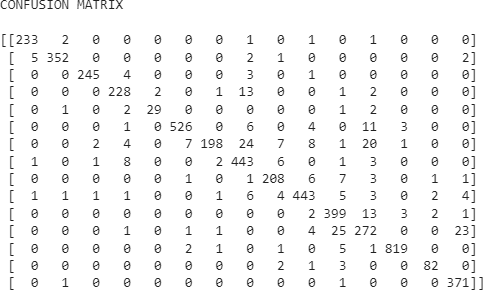


Fig 5.1 Confusion matrix

#### Classification Report



Fig 5.2 Classification Report

#### Accuracy / Loss Graphs

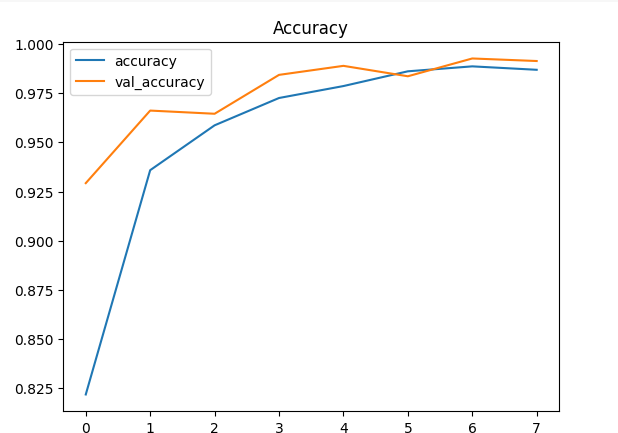


Fig 5.3 Accuracy Graph

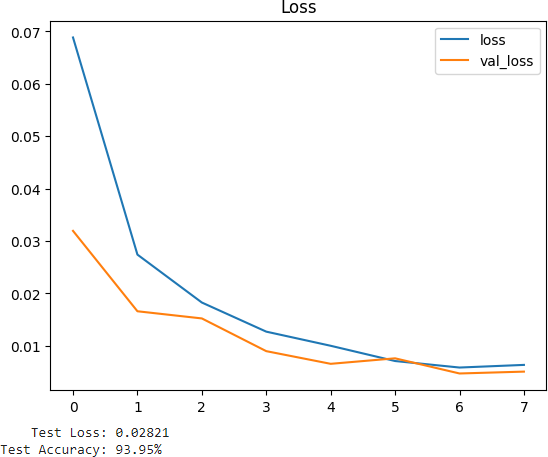


Fig 5.4 Loss Graph

## MODEL DEPOLYMENT

This figure shows the user interface of this application. The interface is very simple and easy to understand. There are only some elements displayed on the screen. There is a file upload option provided. The user can choose an image file. Then there is a submit button. On click of the button, the image uploaded will be given to the model. After processing for few seconds, the image we given and the prediction by the model will be displayed as output as shown below :

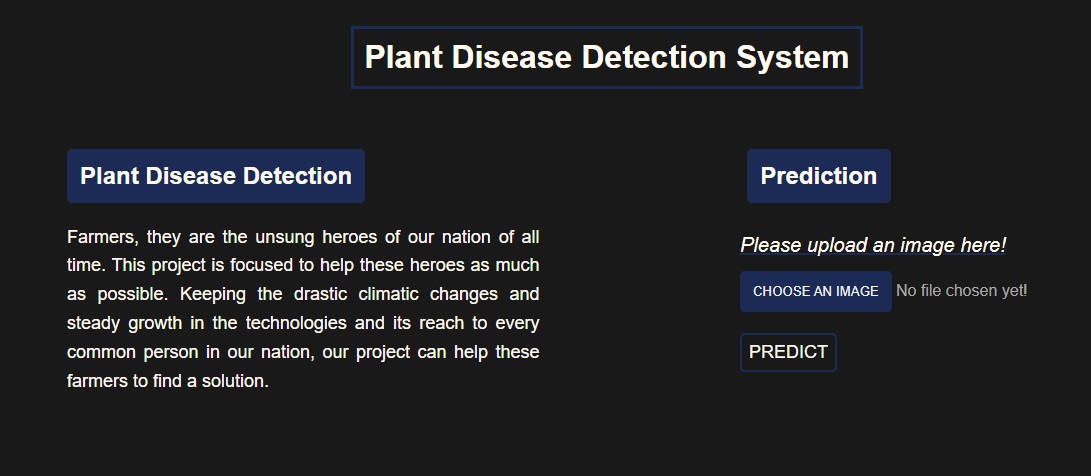


Fig 6.1: UI Design

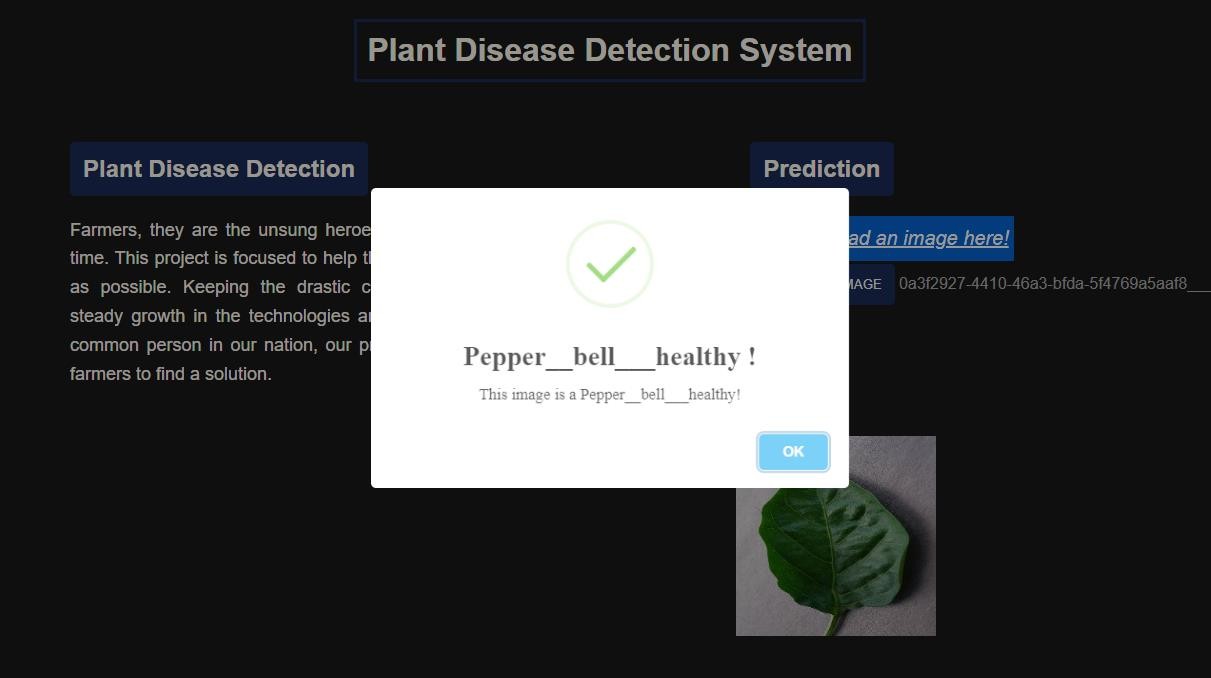


Fig 6.2: Result of Image Feed

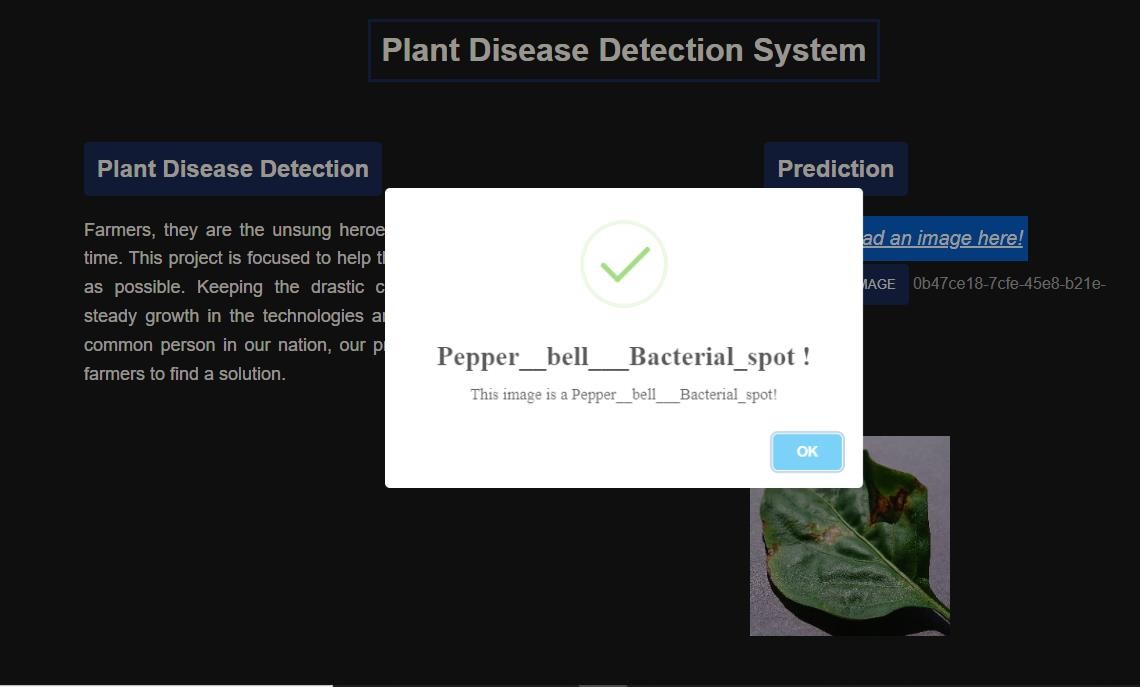


Fig 6.3: Result of real time image

## GIT HISTORY

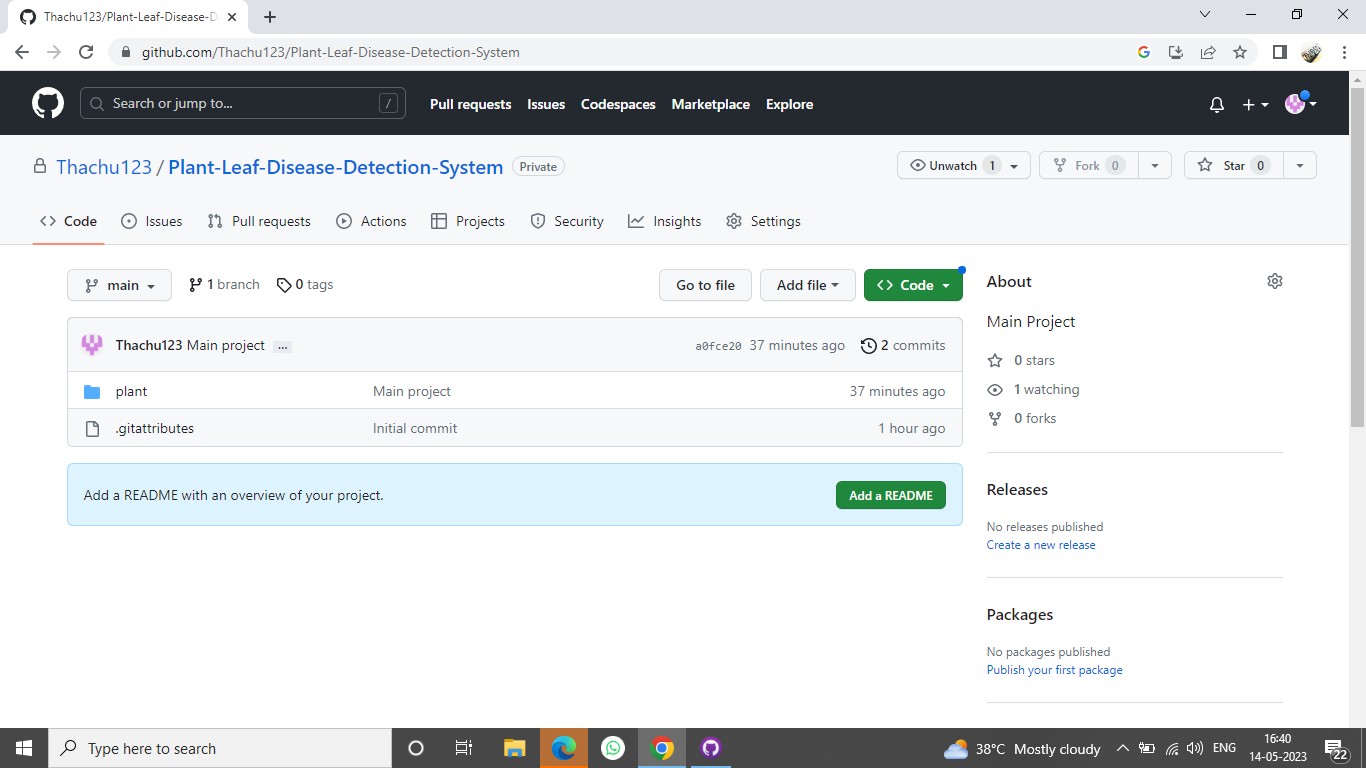


Fig 7.1 Git History

## CONCLUSIONS

Plant leaf disease detection systems using image processing and deep learning techniques have shown promising results in automating disease diagnosis and assisting farmers in early detection and prevention. These systems have the potential to significantly reduce crop losses, increase agricultural productivity, and promote sustainable farming practices.By leveraging advancements in computer vision, machine learning, and data collection techniques, plant disease detection systems can become more accurate, efficient, and accessible. They have the capacity to revolutionize the agricultural sector by providing farmers with timely and reliable information to combat plant diseases. However, further research and development are needed to address challenges such as dataset availability, model generalization, and real-time monitoring. Collaborative efforts between researchers, farmers, and agricultural stakeholders are essential for the successful implementation and adoption of plant leaf disease detection systems in real-world farming practices.

This project is a deep learning project, which aims to detect plant leaf disease from the image. The idea to develop this application as my academic project came to my mind after reading an IEEE paper. For getting more understanding I referred other papers on the same topic also. As the paper suggested it is done using CNN. So I use most recent CNN architecture ResNet50 for model building. And also I made a real time image processing, but becauseof real life constraints and my webcam , it doesn’t shows an accurate result.The entire project has been developed from the requirements to a complete system alongside evaluation and testing. This task concentrates on building up a computerized plant disease detection system. It reduces time and effort. In order to obtain the disease of leaves and to record their disease, the project proposed the plant disease detection can be done by observing the spot on the leaves of the affected plant.. The result of our project shows improved performance in the estimation of the diseases compared to the traditional black and white disease detecting systems. Current work is focused on the disease detection algorithms from images .The system developed have achieved its aim and objectives. There will be changes during the implementation to achieve more which could lead to more contribution in the future.I used Google Colab and Jupyter Notebook for developing this application. It made thework so easy and efficient.

## FUTURE WORK

Currently our project is working on single login at a time for the recognition . we designed a web application. In the future we design a mobile application that collaborate with one of the best disease detection system .so there will be interlinked monitoring and no further manipulation of disease of leaves can be done. There will be more contribution and implementation towards this disease detection in the future. Project can be modernized in nearby future as, when a responsibility for the same arises, as it is very flexible in positions of growth. And the enrichment approach of camera formation based on the result of the position valuation in order to progress the disease detection effectiveness.

1. Enhanced Accuracy: Continued research and development can focus on improving the accuracy of plant leaf disease detection systems. This can involve exploring advanced deep learning models, incorporating ensemble techniques, or integrating multiple sources of data for more accurate predictions.
2. Real-time Monitoring: Future systems can aim to provide real-time monitoring of plant diseases, enabling early detection and timely intervention. This can be achieved by leveraging technologies such as Internet of Things (IoT), remote sensing, and mobile applications.
3. Robustness and Generalization: Improving the robustness and generalization capability of disease detection systems is crucial. This can involve training models on diverse datasets from different geographical regions and plant species to ensure effective detection across various scenarios.
4. Disease Progression Tracking: Expanding the capabilities of plant disease detection systems to track the progression of diseases over time can be beneficial. This would enable farmers to monitor the development of diseases, assess the effectiveness of treatments, and make informed decisions.
5. Decision Support Systems: Integration with decision support systems can provide farmers with personalized recommendations for disease management. These systems can leverage historical data, environmental factors, and disease patterns to suggest appropriate prevention or treatment strategies.

## APPENDIX

### Minimum Software Requirements

* + - Software : Google Colab, Kaggle
    - Operating System : Windows

### Minimum Hardware Requirements

* + - Hardware capacity : 256 GB (minimum)
    - RAM : 8 GB
    - Processor : Intel Core i5 preferred
    - Display : 1366 \* 768
    - GPU : Nvidia ( Kaggle )

## REFERENCES

1.[https://colab.research.google.com](https://colab.research.google.com/) 2.<https://iq.opengenus.org/resnet50-architecture/> 3.<https://cv-tricks.com/keras/understand-implement-resnets/>

1. [https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-](https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/) [regularization-techniques/](https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/)
2. <https://iq.opengenus.org/relu-activation/> 6.<https://ieeexplore.ieee.org/search/advanced/citation>
3. Ajra, Husnul, Mst Khairun Nahar, Lipika Sarkar, and Md Shohidul Islam “Disease Detection of Plant Leaf using Image Processing and CNN with Preventive Measures” In 2020 Emerging Technology in Computing,Communication and Electronics (ETCCE), pp. 1-6. IEEE, 2020.
4. Shrestha, Garima, Majolica Das, and Naiwrita Dey. “Plant disease detection using CNN” 2020 IEEE Applied Signal Processing Conference (ASPCON). IEEE, 2020.
5. Asif, Md Khalid Rayhan, Md Asfaqur Rahman, and Most Hasna Hena “CNN based disease detection approach on potato leaves” 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS). IEEE, 2020.
6. https:/[/www.kaggle.com/datasets/](http://www.kaggle.com/datasets/emmarex/plantdisease.The)e[mmarex/plantdisease](http://www.kaggle.com/datasets/emmarex/plantdisease.The)