NORTH SOUTH UNIVERSITY

Department of Electrical and Computer Engineering

Senior Design Project

AN INTELLIGENT APPROACH TO DETECT PLANT DISEASE

With Convolutional Neural Network

Submitted By:

NAME	ID
Sarker Md Tanzim	1420436042
A.M. Almarufuzzaman	1420469042
Md. Mohaiminul Haque	1310320642
MD. Mehedi Hasan	1320782042

Submitted To: Dr. Shazzad Hosain

<u>**LETTER OF TRANSMITTAL**</u>

April 2019

To

Dr. K. M. A. Salam

Professor and Chairman,

Department of Electrical and Computer Engineering

North South University, Dhaka

Subject: Submission of Capstone Project on “**An Intelligent Approach to Detect Plant Disease**”.

Dear Sir,

With due respect, we would like to submit Our Capstone Project Report on “**An Intelligent Approach to Detect Plant Disease**” as a part of our BSC program. The report deals with an image processing model (Convolutional Neural Network) to detect disease on vegetable leaf. We tried our level best to make the report meaningful and informative.

The Capstone project was very much valuable to us as it helped us to gain experience from practical field. It was a great learning experience for us. We tried to the maximum competence to meet all the dimensions required from this report.

We will be highly obliged if you are kind enough to receive this report and provide your valuable judgment. It would be our immense pleasure if you find this report useful and informative to have an apparent perspective on the issue.

Sincerely Yours,

....

Sarker Md Tanzim

ECE Department

North South University, Bangladesh

....

A.M. Almarufuzzaman

ECE Department

North South University, Bangladesh

....

MD. Mehedi Hasan

ECE Department

North South University, Bangladesh

....

Md. Mohaiminul Haque

ECE Department

North South University, Bangladesh

Declaration

This is to declare that no part of this report or the project has been previously submitted elsewhere for the fulfillment of any other degree or program. Proper acknowledgement has been provided for any material that has been taken from previously published sources in the bibliography section of this report.

....	....
Sarker Md Tanzim	A.M. Almarufuzzaman
ECE Department	ECE Department
North South University, Bangladesh	North South University, Bangladesh
....	....
MD. Mehedi Hasan	Md. Mohaiminul Haque
ECE Department	ECE Department
North South University, Bangladesh	North South University, Bangladesh

Approval

The Senior Design Project entitled "**An Intelligent Approach to Detect Plant Disease**", by Sarker Md Tanzim (ID#1420436042), A. M. Almarufuzzaman (ID#1420469042), Md. Mohaiminul Haque (ID#1310320642) and MD. Mehedi Hasan (ID#1320782042) has been accepted as satisfactory and approved for partial fulfillment of the requirement of BS in CSE degree program on April, 2019.

Supervisor's Signature	**Department Chair's Signature**
Dr. Shazzad Hosain	**Dr. K. M. A. Salam**
Associate Professor	**Professor**
Department of Electrical and Computer Engineering	Department of Electrical and Computer Engineering
North South University	North South University
Dhaka, Bangladesh.	Dhaka, Bangladesh.

Acknowledgement

First and foremost, we want to thank the Almighty for giving us the ability and knowledge to carry out this project and our family members for their utmost supports.

We deeply thank our respectful faculty and advisor **Dr. Shazzad Hossain**, for providing us an opportunity to do the project and giving us his support and guidance, which helped us complete the project duly.

We are also extremely thankful to **A.P.M. Alamgir Kabir** from **Giant Agro Processing Ltd.** at Bhaluka, Gazipur research center for giving us the opportunity to collect some of his hybrid vegetables leaf image for our dataset. We are truly grateful to him for giving us such a nice support and guidance, although he had busy schedule managing the research on hybrid vegetable.

Last but not the least, we want show our gratitude to our beloved university for providing us with all the facilities and the chance to showcase our project in front of other student and faculty members. This opportunity to present our work has made our hardship worthwhile.

Abstract

Vegetable and crop diseases are serious threat to food security for most of the agriculturally based country., but the identification is still very difficult specially in early stages. So, the idea of - fast, accurate and early detection of vegetable and crop's pests and infectious disease can be used to minimize the food and economical losses. Indication of plant of vegetable diseases can be identified from leaves, where the evidence of a disease or infection can be found, using Convolutional Neural Network (CNN) techniques. We proposed a CNN model inspired by VGG19 model trained on 24,162 images of corn, potato and tomato leaves, that could detect 15 district categories of diseases excluding the healthy leaves. At the end, an accuracy of 96.70% was obtained on test dataset with our CNN architecture. The idea of identifying the phase of an infection or disease is predicted by observing the prediction for result as healthy plant or vegetable when the final prediction was infected vegetable.

Table of Contents

Chapter 1: Introduction

1.1 Overview

Plant diseases is a critical issue in the field of agriculture. It can have many negative effects on the agricultural productivity of a country. Almost 16% of all crops are lost to plant diseases every year. New pathogens may yet increase this number. [6] Whenever a new pathogen arises, even if conditions are optimal and crops are produced in a large-scale monoculture, it can still cause an increase in crops death. Pathogens are the reason behind 12.5% of worldwide crop losses. [7] Most diseases are caused by bacteria, viruses and fungi. They infest the plants, cause serious damage to their productivity and quality and sometimes even kill their hosts.

When a plant is infected by any diseases the symptoms always appears on its leaf. So, leaf is the key to the detection and classification of plant diseases. Different plant leaf bears different diseases and the major categories of plant leaf diseases are based on viral, fungal and bacteria. The manual observation by the naked eye of experts is the main approach used to detect and identify the plant diseases. However, this requires continuous monitoring of experts. When there is a large farm, this method might be ridiculously expensive and time consuming.

Further, in some developing countries – like ours, farmers may have to go long distances to contact experts which makes consulting experts too expensive and time consuming and moreover farmers are unaware of non-native diseases. Automatic detection by using computer vision to identify plant diseases is an important topic as it may prove beneficial in monitoring large field of crops, and thus automatically detect diseases from symptoms that appear on plant leaves. Thus, automatic detection of plant disease with the help of computer vision provides more accurate. Reasonably, naked eye identification is less accurate and time consuming. This is why we decided to build an automated system to overcome the problems of manual techniques.

1.2 Motivation

Bangladesh is an agricultural country and agriculture sector is the backbone of its economy. Its economy has conventionally been based on agriculture. That is because nature and geo-location has consecrated it with vast areas of fertile land and abundant success of water, which are the principal components of an agro-based economy. Diseases and insect pests are the major problems in cultivation and these require careful diagnosis and timely handling to protect the crops from heavy losses. In plant, diseases can be found in various parts such as fruit, stem and leaves. Plant diseases cause episodic outbreak of diseases. Those outbreak of diseases leads to large scale death and famine. Diseases are the natural factor to cause some serious effects on plants and that ultimately decreases productivity, quality and quantity of crops.

In developed countries, handful of intensive care and management mechanisms are used in place to moderate the consequences of harmful diseases of crop plants. These mechanisms allow prompt application of control measures. In the so-called developed countries, agriculture is not without risks of epidemics. However, management systems are in place that moderate the economic and social effects of such very harmful diseases. Similar systems must be established urgently in developing countries to avert socio-economic disaster due to plant disease.

At first, we thought to use image processing to design the system model but after some research we found that CNN can generate more accuracy than Image processing so we decided to go with CNN to design our system model which can detect some selected diseases from some selected vegetables.

1.3 Why Convolutional Neural Network

Our initial plan was to use traditional image processing algorithms but latter we switched on CNN because of its performance and higher accuracy. Since, CNN is a part of AI, it can be thought of automatic feature extractors from the images. If, we used the traditional algorithms with pixel vector we lose a lot of 3-D interface between pixels. A CNN successfully uses adjacent pixel information to successfully down-sample the image first by convolution and then uses a prediction layer at the end and it has the advantage to learn the hierarchical features.

The main inspiration behind the arrival of CNNs has been to address many of the restrictions that traditional neural networks faced when applied to those problems. When used in areas like image classification, traditional fully-connected neural networks simply don't scale well due to their excessively large number of connections. CNNs bring a few new ideas that contribute to improve the efficiency of deep neural networks. Convolutional Neural Networks (CNNs) take advantage of local 3-D coherence in the input (sometimes pictures), which let them to have fewer weights as some parameters are shared. This process, taking the form of convolutions, makes them especially well-suited to extract relevant information at a low computational cost. [1]

The only downside to CNN is that it takes a lot of time to train. Although, this time depends on the dataset size and the hardware on which the model is being trained. However, the bright side is, once a model is built, the prediction is pretty fast. A CNN network can train with any number of inputs and hidden layers. It may take longer time to train with large dataset but CNNs accuracy becomes better with more data points.

Despite the few drawbacks the reason behind choosing CNN is the influence of its advantages. Well, we could have chosen other Neural Networks such as LSTM, RRN. But LSTM, RRN works better with sequential Datasets such as video, sound. Since we were working with separate images, CNNs become the best choice to train our dataset to build the system model.

1.4 Objective

The main objective of our project is these –

1. Identify which plant the leaf belongs to.
2. Check if the leaf is infected or not.
    a. If yes, Identify the disease.
    b. Detect which phase the infection is in.

1.5 Summary

Plant diseases are a massive problem and a serious threat to food security for most of the agriculturally based country like Bangladesh. Mainly farmers suffer a lot, because they couldn't properly identify the disease and uses wrong medicine. To reduce these kinds of loss disease detection is mandatory. However, the identification is still very difficult in our country due to the absence of the necessary patronage. So, an idea of fast and accurate detection of vegetable and crop's infectious disease is required that can be used to minimize the food and economical losses. To solve that problem, we proposed this project based on Convolutional Neural Network.

Chapter 2: Literature Review

2.1 Overview

Before going ahead with our own project, we researched previous works done in this topic. We went through various papers that were related to our project. We gathered information that could help us in our works and got the basic idea about the steps we must take. We explored multiple ways of image processing and recognition, which provided us with many options. Not only these papers helped us to choose a suitable way for our project but also gave us many tips for achieving maximum efficiency.

2.2 Related Works

We went through many previous researched that were similar to our work. It gave us a clear view of which path we should take. We found a lot of work on similar problems to ours and they provided us with many useful solutions.

Our theory on using Neural Network being the best method for plat disease detection was firmly based on the review work of K. Golhani et all. Their review paper on advanced Neural Network (NN) techniques talked about different NN models and classifiers when used on plant disease detection. They reviewed different mechanisms and algorithms to process the plant disease data. They observed that Neural Network approach can be used as a powerful tool for identifying diseases. [10]

Machine learning approach to detect and diagnose plant diseases by processing leaf images is not an uncommon strategy. There are many studies that were done in this line. Most of the researches were easily carried out using a small number of images. They primarily focused on the extraction of features from image to classify the leaves. M. Brahimi, K. Boukhalfa and A. Moussaoui, used a handful of 14,828 images of tomato leaves including nine distinct categories of diseases infected leaves. They claimed this number to be enough and more than the number used in other studies. They took Convolutional Neural Network (CNN), just like us, as a learning algorithm. They argued that the advantage of CNN being, "The automatic extraction of features by processing directly the raw images." [11]

Their model achieved a significant 99.18% level of accuracy which was more then other shallow model that used small datasets. So, CNN model trained with large dataset can be a very powerful tool in disease detection. [11]. In another research, deep convolutional networks approach was also used in the development of plant disease recognition model. Their model was able to classify 13 different disease categories with the experimental results attaining on average 96.3%. [12]

We got the idea of using VGG model from a research done by K. Ferentinos. In his research [9] he used deep learning methods to developed a CNN models to perform plant disease detection and diagnosis. He used a database of 87,848 images, which included 58 categories of diseases from 25 different types of plants. He tried different architectures – AlexNetOWTBn and VGG to train the dataset, and then chose the one that had the best accuracy in identifying diseases which happen to be the VGG model. The model had an accuracy level of 99.53% when tested on 17,548 new set of images. This significantly high success rate of VGG model proved to be handy on detecting the diseases in early stage. This approach could be useful in detecting and identifying leaf diseases in real life scenario. [9]

2.3 Summary

From all these research papers and related works, we gained many valuable insights that later made it easier for us to decide how to conduct our own project. We understood each procedure, mechanism and usage very carefully and applied that knowledge as we crafted our own design. It's from these work that we decided to go with VGG model as our reference model and was encouraged to use a dataset of such a large size.

Chapter 3: Methodology

3.1 Overview

In this chapter we will give a systematic, theoretical analysis of each step of our project. We will go through the principals and theories behind our applied methods. We will only talk about the problems we needed to solve in those steps rather than the solutions. We will give an overview of our model and show which methods we used in each layer.

3.2 Our Approach

Before we went in to the details we first tried to figure out how our overall system would look like and how one part would interact with other. So, we discussed the design of our overall project using block diagrams as shown in Figure 01(a) below.

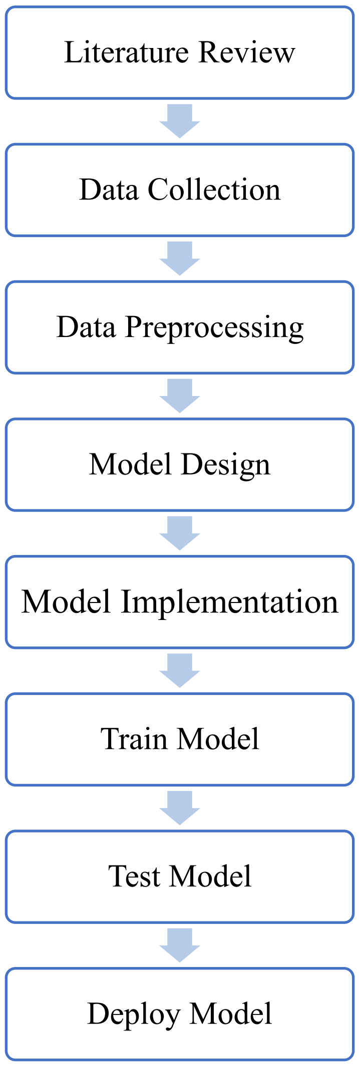


Figure 01 (a): Workflow

Our first move was to gather sufficient knowledge about our topic. To achieve that, we had to go through many related paper works, online articles, journals etc. Once we were ready to start our project, we began to collect our dataset. This took quite a while as we also needed a vast dataset to start our work. The collected dataset then went through several processes to make it more readable for the machine.

In the next step we began to design our model using the knowledge we gained from related work as well as using new ideas. Once we had a satisfactory model design we went ahead and start implementing the model in Python. We trained and tested the model over and over again each time trying to fix and improve the model until we had the desired accuracy. After that we deployed the model using flask so that any random user could use the model.

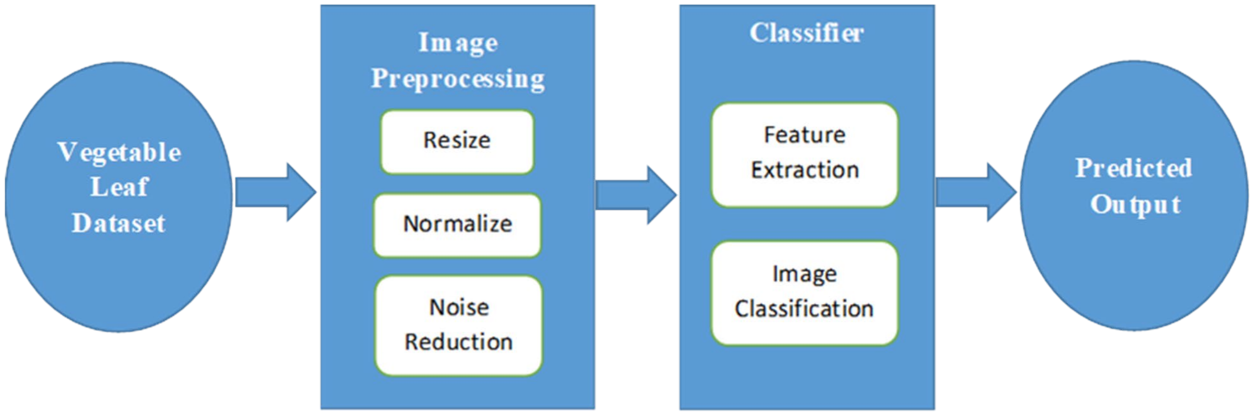


Figure 01 (b): Model Overview.

The basic working idea of our model is shown in Figure 01(b). We first preprocess the our dataset by resizing, rotating, and normalizing the images and send them through our model for feature extraction and classification, The trained model then can take in a new image as input and predict it's class as output.

3.3 Project Requirements

- Machine Learning
- Convolutional Neural Network
- Infected Vegetable Leaf Dataset
- Image Preprocessing
- Python

3.4 Summary

This part discusses not a detailed plan but gives us an overview. We have given a building block view of the steps taken by us throughout the process of building this project. To summarize our entire work

Chapter 4: Data

4.1 Overview

This chapter is dedicated to our data. For our project, we had to deal with vegetable plant leaves images. We had to collect these images ourselves and label them. We also had to preprocess the entire dataset to prepare them for training. These were simple but lengthy processes. We'll go over them and explain or steps as well as provided detailed descriptions of our dataset in the following sections.

4.2 Data Labeling

Since our project was to identify whether a vegetable has a disease or not by observing a leaf image, we had to collect data (Vegetable leaf images) physically. The researcher on hybrid vegetable, Mr. A.P.M Alamgir Kabir from Giant Argo Processing Limited helped us to collect healthy leaf images of Corn, Potato and Tomato. For the infected images we used our smartphones camera to capture video of a particular disease and plant at a time. Latter we split the frames from the videos as images to categorize the leaf as plants and diseases.

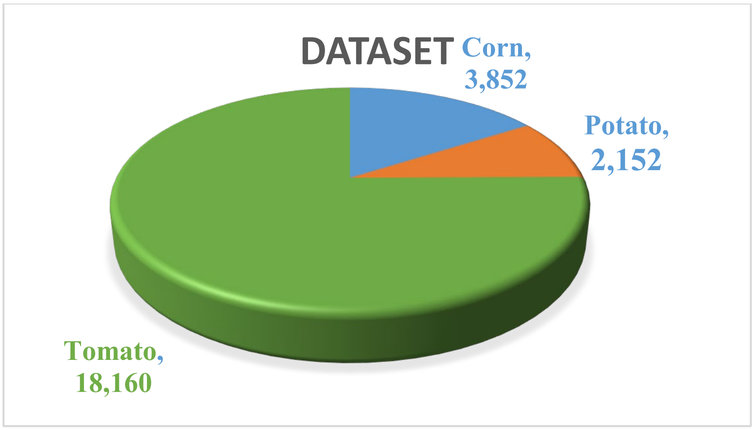


Figure 02: Dataset overview

We were able to assemble 24,162 healthy and infected images for Corn, Potato and Tomato. The ratio of each category is shown in Figure 02. The data needed to be labeled for the machine to recognize them. So, we manually labeled each type of data. First, we sorted the leaf data into three broad categories depending on what type of plant (vegetable) they belonged to. Then we sorted for them by each diseased and healthy leaf.

4.3 Data Visualization

In our dataset, there were three major categories: Corn, Potato & Tomato. Each of these categories contain several diseases as well as the healthy leaf category. A detailed description of these three types of leaves are given below.

<u>Corn:</u>

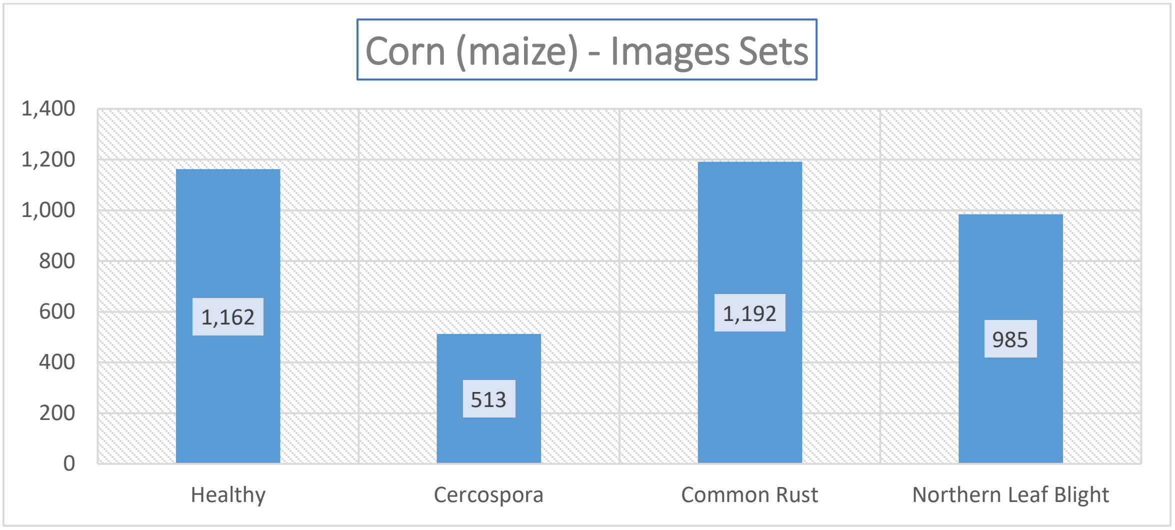
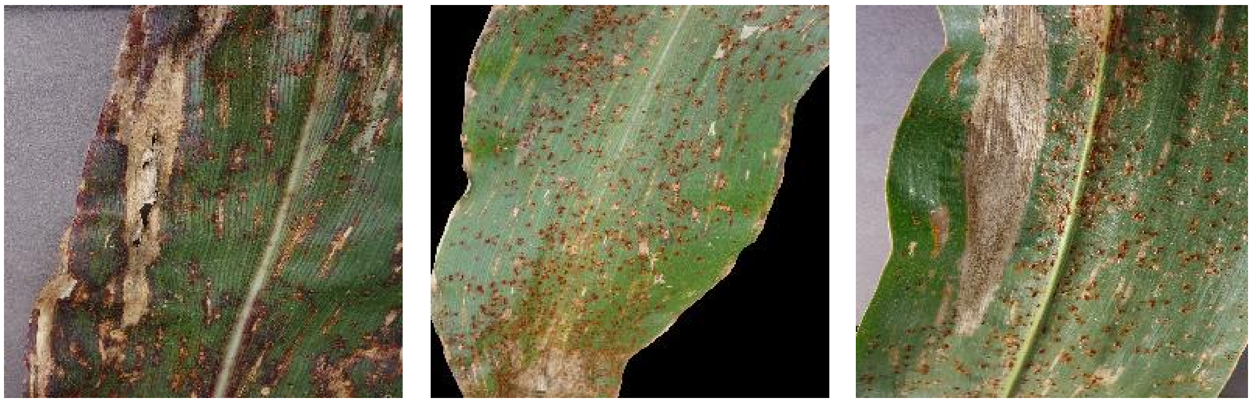


Figure 03: Corn (maize) Leaf Dataset.



4 (a): Cercospora **4 (b): Common Rust** **4 (c): Northern Leaf Blight**

Figure 04: Corn (maize) Disease Categories.

Figure 03 shows the total number of corn leaf images for each category. As we can see, there are four total categories including the healthy one. So, there are three types of diseases for corn leaves. A short description on each category are provided below.

- **Cercospora:**

Figure 4(a) shows a Cercospora or gray leaf spot disease infected leaf of corn. It is caused by *Cercospora zeae maydis* named fungus. Symptomatic includes small, tan, rectangular lesions surrounded with yellow halos. In late it becomes long, grey to tan in color, and rectangular.

- **Common Rust:**

Figure 4(b) shows a Common rust infected corn leaf. This is caused by the fungus *Puccinia sorghi*. These diseases can be easily recognized and distinguished from other diseases by the development of dark, reddish-brown pustules scattered over both the upper and lower surfaces of the corn leaves.

- **Northern Leaf Blight:**

Figure 4(c) shows Northern leaf blight affected corn leaf. This is caused by Exserohilum turcicum. The symptoms include brown spots with long gray halos.

Potato:

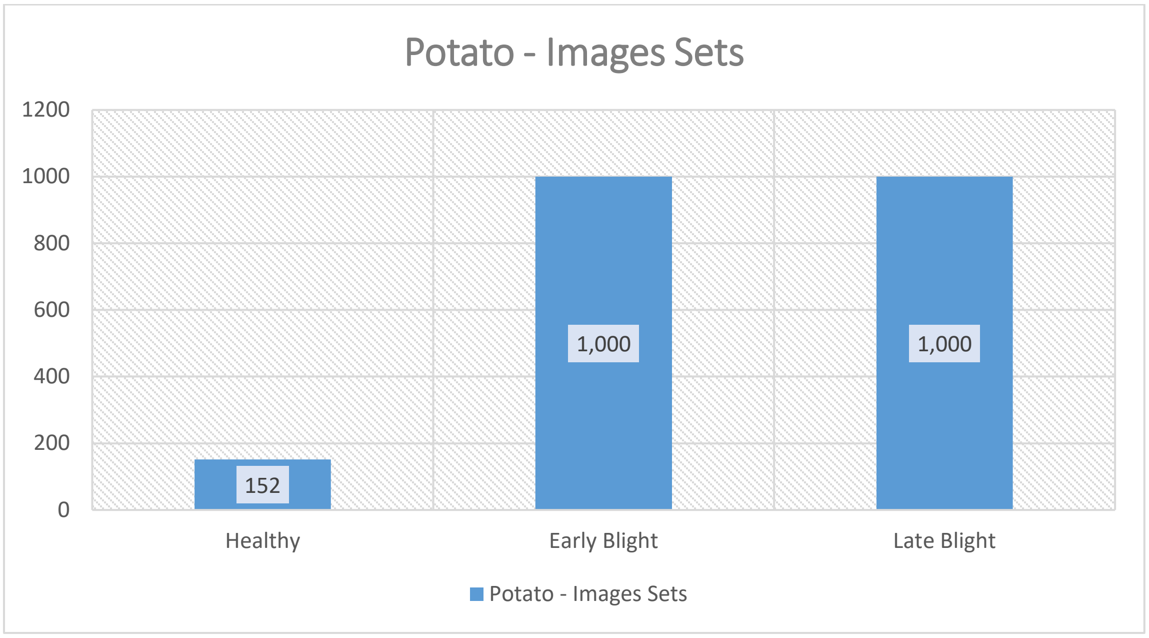
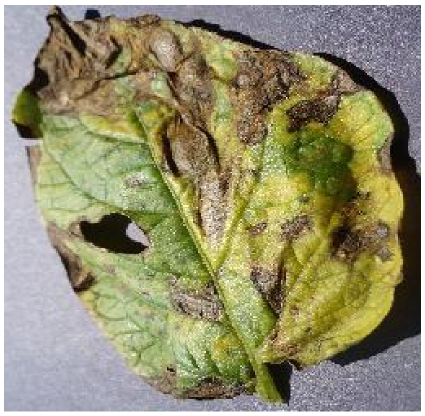


Figure 05: Potato Leaf Dataset.


6 (a): Early Blight

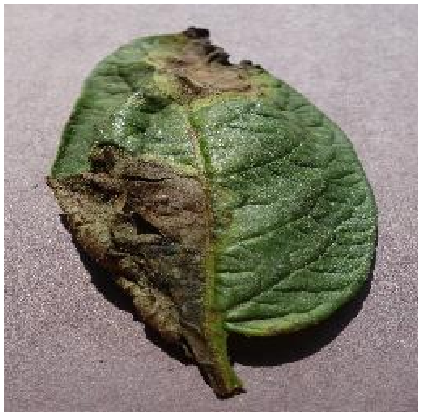

6 (b): late Blight

Figure 06: Potato Leaf Diseases Categories.

Figure 05 shows the total number of potato leaf images for each category. From the graph we can see there are three total categories including the healthy one. So, there are two types of diseases for potato leaves. A short description on each category are provided below.

- **Early Blight**

In Figure 6(a) we can see an Early blight infected potato leaf. Initial infection occurs on older leaves, with concentric dark brown spots developing mainly in the leaf center. Infected leaves turn yellow with the underlying flesh turning dry, leathery and brown.

- **Late Blight**

In Figure 6(a) we can see an Late blight infected potato leaf. This disease first appears as a water-soaked, gray-green spots. As the disease matures, these spots darken and a white fungal growth forms on the undersides. Eventually the entire plant will become infected.

<u>Tomato:</u>

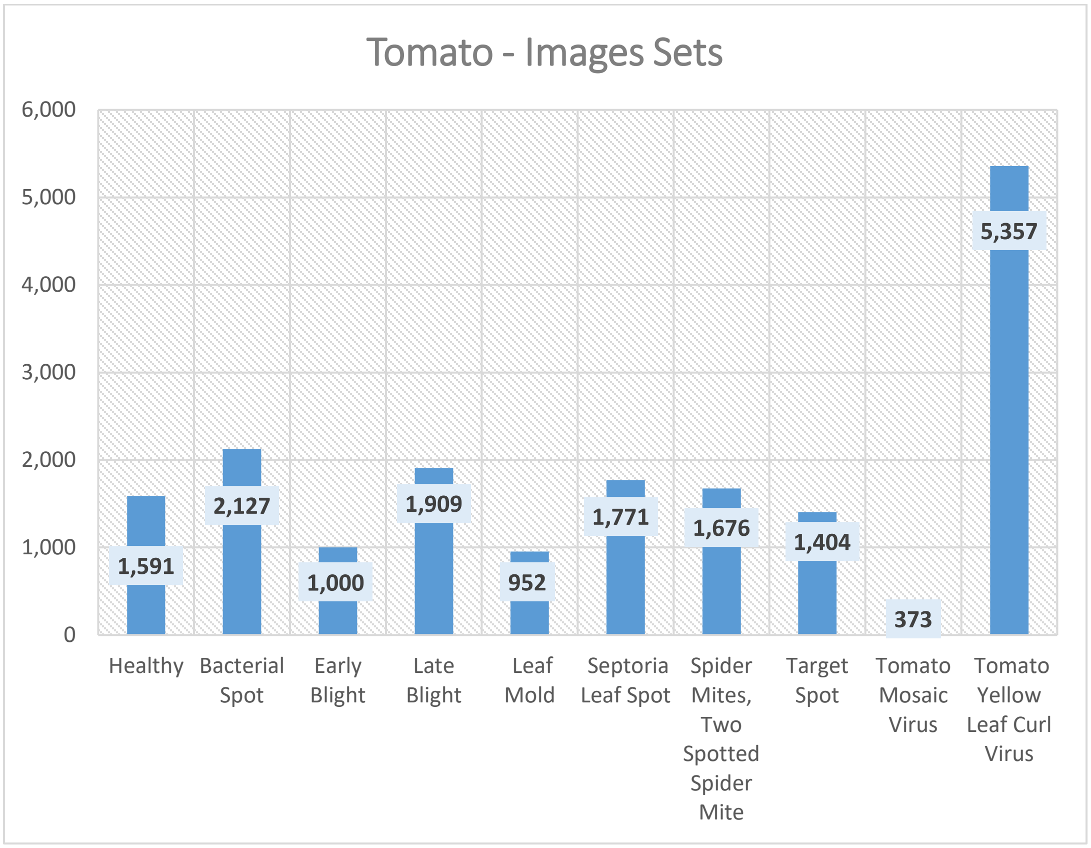
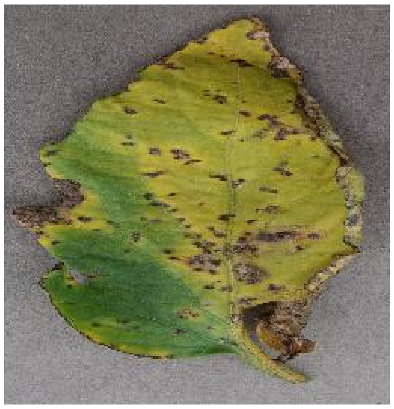
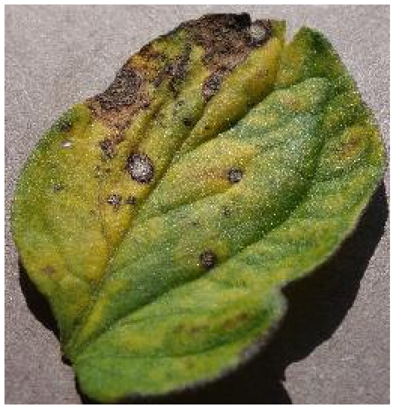
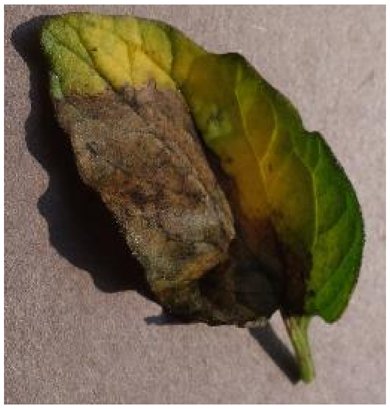


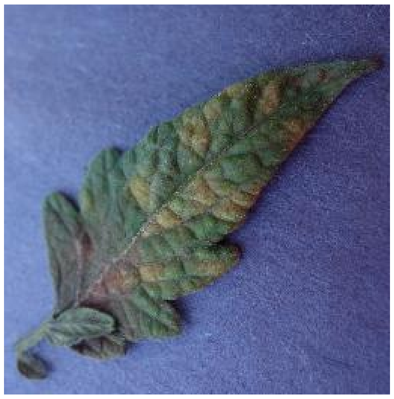
Figure 07: Tomato Leaf Dataset.

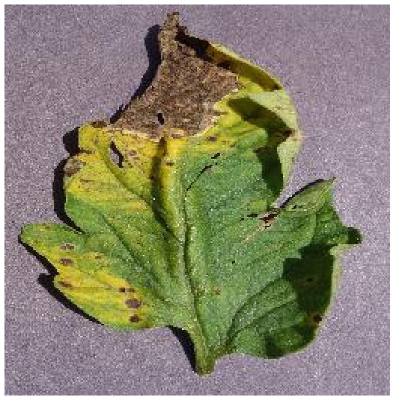
Figure 07 shows the total number of tomato leaf images for each category. As the graph shows, there are ten total categories including the healthy one. So, there are nine types of diseases for tomato leaves.

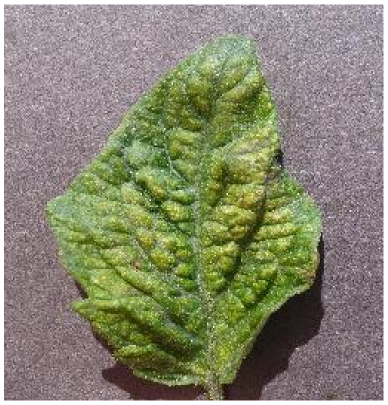

8 (a): Bacterial Spot

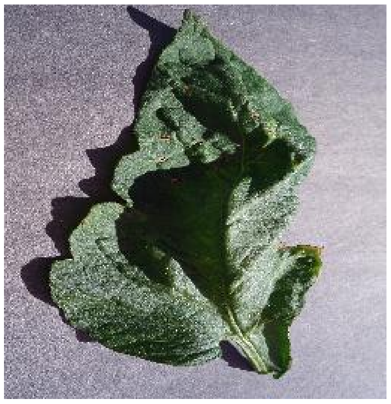

8 (b): Early Blight

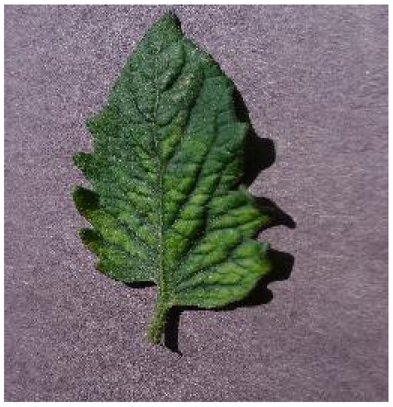

8 (c): Late Blight


8 (d): Leaf Mold


8 (e): Septoria Leaf Spot


8 (f): Spider Mites


8 (g): Target Spot


8 (h): Mosaic Virus

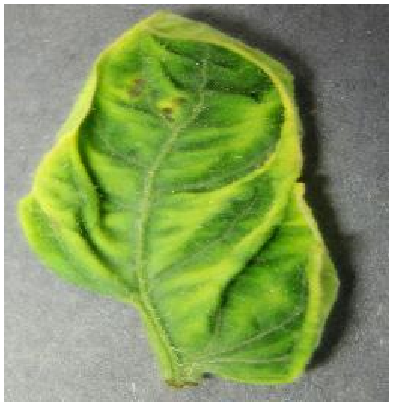

8 (i): Yellow Leaf Curl Virus

Figure 08: Tomato Leaf Diseases Categories.

A short description on each category are provided below.

- **Bacterial Spot**

Bacterial spot of tomato is a result of the bacteria xanthomonas campestris pv. vesicatoria. Symptoms of bacterial spot first appear as small, greasy, and irregular marks under the tomato plant's leaves. The spots start as a dark green, then gradually become purple and gray with black ce nters, possibly within a white or yellow outer circle as shown in Figure 8(a).

- **Early Blight**

Early blight of tomato is also caused by Alternaria solani as Early blight of potato. From Figure 8(b) we can see the symptoms includes small necrotic spots that appear dry and papery. As lesions enlarge they usually produce concentric rings giving the lesion a target-like appearance. Lesions forming on the leaf rachis or petiole may cause entire leaves to turn brown and shrivel.

- **Late Blight**

Late blight of tomato is also caused by Phytophthora infestansI as Late blight of potato. And it has the same symptomps and also favored the same environme nt as Late blight of potato. Figure 8(c) shows a late blight infected tomato leaf.

- **Leaf Mold**

Leaf mold of tomato is caused by Passalora fulva. The primary symptom as shown in in Figure 8(d), appears on the upper surface of infected leaves as a small spot pale green or yellowish with indefinite margins, and on corresponding area of the lower surface, the fungus begins to sporulate. On the lower surface as an olive green to grayish purple and velvety appearance, which are composed of spores. Continuously, the color of the infected leaf chang es to yellowish brown and the leaf begins to curl and dry.

- **Septoria Leaf Spot**

In Figure 8(e) of tomato is one of the most devastating disease for tomato. It is caused by a fungus named *Septoria lycopersici*. Once a plant has been infected with it, the leaves will begin to show random wet-looking spots all over, which will grow to a larger size and change to a light brown or tan color. The leaves may even have small spore-producing masses on the spots.

- **Spider Mites**

Spider mites is another disease of tomato as show in figure 8(f). Leaves of mite infested plants may turn yellow and dry up, and plants may lose vi gor and die when infestations are severe. The undersides of affected leaves appear tan or yellow and have a crusty texture.

- **Target Spot**

Target spot of tomato is caused by the fungus Corynespora cassiicola. As we can see in Figure 8(g) the symptoms include irregular-shaped spots with a yellow margin. Spread to all leaflets and to other leaves is rapid, causing the leaves to turn yellow, collapse and die. Spots also occur on the stems. They are long and thin.

- **Tomato Mosaic Virus**

Tomato Mosaic Virus is a plant pathogenic virus and there is no cure for it. ToTomato mosaic virus causes yellow mosaic symptoms on the leaves and fruits as seen in Figure 8(h). The primary symptoms are seen as a general mottling or mosaic appearance on foliage. Light and darker green mosaic leaf mottle, sometimes with distortion of younger leaves.

- **Yellow Leaf Curl Virus**

From figure 8(i) we can see a yellow leaf curl virus affected tomato leaf. It is the most damaging and threatening virus for tomato production worldwide. It make the leaf to turn yellow and curled up along the edges.

4.4 Data Preprocessing

To make a great model the dataset needs to be properly prepared before training. For that purposed we carried out some steps of pre-processing. First of all, we split the dataset into a training set and test set. The split ratio was 80%-20%. and then applied the following methods:

- Random Rotation

On the training set, we randomly rotated some of the data from each category and performed horizontal flip on some of them.

- Resize

All the images were different in sizes. This could have caused a problem while training. So, we resized all the images to insure all of them are of the same size, where height and width both were 224 unit. We did this for both train and test sets.

- Noise reduction

Noise means unnecessary and redundant information in the images. It can cause overfitting problem. So, we blurred some of the images to reduce the noise in images and avoid overfitting.

- Normalization

We did normalization to increase the diversity of the data. We performed normalization in the following manner.

image = (image - mean) / std………………… (1)

We applied the above formula (1) on each image for all three colors (red, blue, green) so that their values stay in the range [-1, 1]. The reason we did this is because normalization can help CNN perform more efficiently by containing the values within a range and reducing the skewness of the data. This allows the CNN model to read the data faster and get a learning rate.

4.5 Summary

This chapter includes the detailed information about our dataset. It talks about how we collected the data and how we labeled them. It gives can brief description of each category of diseases. Finally, it shows how we prepared the data before passing it through our model for training.

Chapter 5: Model

5.1 Overview

In this chapter we will briefly discuss about the most crucial part of our project, the model architecture. Here we will talk about how we designed and trained our model, which includes how many layers we added and what their purposes were. We know CNN takes a set of input and passes them through a series of hidden layers and each of those layers can be different and serve various purpose. Except the input layer and the last fully connected layer which is the output layer, there are no restriction to how many and what kind of hidden layers there can be inside a model. So, the main challenge was to select the number and type of layers we want to put in our system. That's what we will go through here and explain how many layers we chose and why.

5.2 Model Background

Before we started designing our own model, we looked into some of the models that already existed and had a excellent performance in image recognition. We studied pretrained models like DenseNet, Vgg19, AlexNet, GoogLeNet carefully. These CNN models helped us to understand how each layer works and how we should arrange them. We studied the pros and cons of each layers so that we could decide how and where we needed to put them in our own model. Among the pretrained models, ours is very close to the VGG19 pretrained model. So, let's take a brief look in VGG19 models architecture before we start explaining our own model.

- VGG19

This VGG is a pre-trained Convolutional Neural Network (CNN) model that is trained on more than a million images from the ImageNet database. The network is 47 layers deep and the VGG network can classify images into 1000 object categories – such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The image input size of the VGG is 224-by-224. [3]

VGG network contains 16 convolutional layers and is very interesting because of its very same architecture. Similar to AlexNet, only 3x3 convolutions but contains a lot of filters. The VGG network trained on 4 GPUs for 2–3 weeks. It is presently the most favored choice in the computer vision community for extracting features from images. The weight configuration of the VGG network is openly available to use and has been used in many other applications and challenges as a baseline feature extractor. However, VGG network consists of 138 million parameters and that can be a bit challenging to handle. VGG network has 47 layers. There are 19 layers with learnable weights: 16 convolutional layers, and 3 fully connected layers. [3]

5.3 Model Design

We now move on to our own model. As we have stated before the main challenge for designing a model is to determine the number and type of layers that should be used to maximize efficiency. Taking the decisions about what to use was not so easy because there are huge number of combinations that could be applied. That's why we studied the pre-built models which provided us some insights.

The Input layer takes a color image as input. A color image is actually a multi-dimensional array or a tensor which has the following values: [Height (H), Width (W), RGB channels (C_{in})]. So, the Input layer needs to hold the information about the height, width and RGB color channel of an image. For our case the value of the images was [224, 224, 3]. [2]

The Convolution Layer is the heart of a CNN architecture. This where the features of an input images are extracted. The extraction process is done automatically with the help of a filter. The filter, which is actually a matrix, is called a kernel. The kernel extracts a 2D activation map known as the feature map. In training phase, this map is used to detect patterns in images and later on, with the help of this map, the model searches for similarities between images and the map. This how the model recognizes patterns. [4]

Conv layer comes with a linear rectifier (RelU) module that allowed the layer to perform an activation function to each element of the input and threshold their values at zero. However, it didn't affect the size. So, we required something to bring the size down to make it more manageable. This where the Pooling layer came in. Pooling layer helped us downsize the volume. It applied down-sampling operation to reduce the width and length. [2]

The fully-connected (FC) layer can compute class score and return a volume size [1x1xC], where c is the number of class size. The final fc layer is where the output class is decided. It So, we had to put a FC layer in the tail end of our model. This layer had to be connected with all the neurons of previous layer. [4]

We played around with these layers for a while, mixing them up and trying different combination. After going through many failed and undesirable attempt we found a few combos that were worth using. We picked the architecture that gave us the best accuracy for our leaf dataset. This model had layers. Once we decided which of them we want in our model it was time to calculate their sizes and number of parameters.

We started of the input layer. It's the starting point of our model. As there's no parameter in this layer there was no need for any calculation. We just provided the input image size which was [224, 224, 3]. The next layer was our 1st convolution layer. The learning process started form here. So, we needed to consider the weight matrices. To calculate the parameters we followed the following equation:

$$P = ((m * n)+1)*k) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \; 2$$

Equation 2 helped us calculate the number of parameters, P by multiplying shape of width of filter, m and shape of height filter, n. Then, we added 1 for the bias element and again multiply the result with the number of filters, k. The pooling, dropout and rectifiers layer didn't need any parameters so there wasn't any calculation needed.

The trickiest part is to calculate the forward and backward propagations. It's not possible to show these calculations by hand as it takes a huge amount of computational power. Thankfully, python automatically computes these when a CNN model is called. We merely selected the arrangement of layer and the value for some other parameter and let python take care of the rest. Here is the summary of our model:

Input features: 12288 (as each image had the size of [64, 64, 3])

Output classes: 23

Kernel size: 3 by 3

Stride: 1 by 1

Layers:

The below Table 01 gives a overview of our model and it’s layers.

Layer No.	Layer Name	Input Parameters	Output Parameters
1.	input	12288	0
2.	conv_0	12288	4096
3.	relu_0	4096	4096
4.	drop_0	4096	4096
5.	relu_1	4096	4096
6.	drop_1	4096	4096
7.	FC1	4096	1024
8.	relu_2	1024	1024
9.	drop_2	1024	1024
10.	FC2	1024	256
11.	relu_3	256	256
12.	drop_3	256	256
13.	Output	256	23

Table 01: Model Layer Summary.

Table 01 gives us the overview of our model layers. Note that the Number of parameter doesn’t change on *relu* and *drop* layers as they only work on the activation size and hence keep the parameters unchanged.

5.4 Implementation

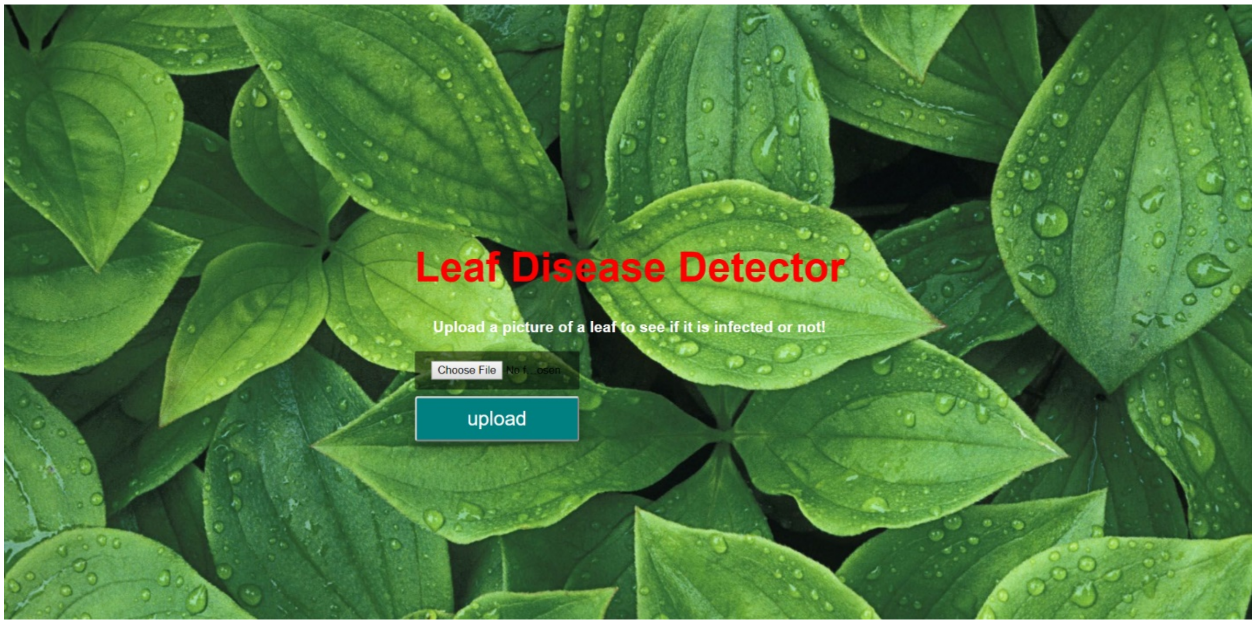
Once we had our model design fixed it was time for implementation. Up till this point we were designing our model theoretically. To make it working we moved on to coding. For the coding language, we preferred Python3 language to build the system. Its libraries provided a great help in building and training the model. We used two python frameworks in this project -

- pyTorch: For building the model
- flusk: For deploying the model

pyTorch provided many handful modules for designing and building a model of our own. It allowed the flexibility we needed to implement our model just the way we wanted it to. Before passing the data through training we first loaded the dataset using the *dataloader* function. The *transform* module helped us pre-process our data to make it more readable. With the help of this module, we resized, rotated and normalized the dataset. Then we went on to implement the model. pyTorch allowed us to add each layer with the options to choose the parameter by ourselves. We created our classifier using the *nn* module provided by pyTorch.

Once we created the model, we started to pass the dataset through the model for training and validating. We split the dataset and sent in batches to reduce the computational expenses. We repeat this process for 30 times and observed the accuracy in each step to see how the model was performing. After we reached the desired accuracy the model was ready for deployment.

For deployment we used python’s flask framework. This micro web framework was the suitable option as we only wanted to deploy the model just for presenting our project in the final presentation. Using flask, we built a simple site where an user can upload an image and get the result and the phase as well.

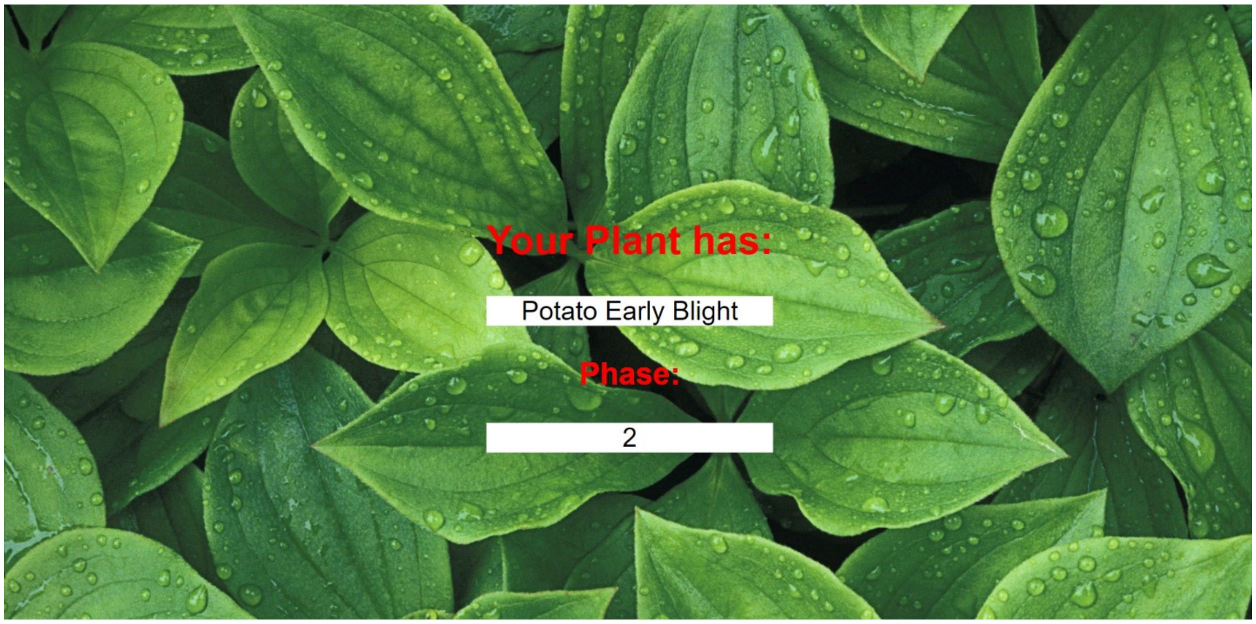


Leaf Disease Detector

Upload a picture of a leaf to see if it is infected or not!

Figure 09: Project Interface (Home Screen)

Figure 09 shows the home display of our project. Users can use this page to upload their infected vegetable leaf image for detection. By clicking upload button our model will take in the image and output the predicted class providing the user with the name of the disease (if any) the leaf is infected with and the phase it is in, in the following manner as shown in Figure 10, where we can see the output or the result screen.


Your Plant has:

Potato Early Blight

Phase:

2

Figure 10: Project Interface (Output Screen)

5.5 Summary

We designed our model taking inspersions from the vgg19 model. We changed and updated the model until we acquired the desired accuracy. Then we implemented the model using python's *pyTorch* framework and once it was ready, we deployed it using *flask* framework. We created a simple interface where users can test their vegetable leaf images for any possible diseases.

Chapter 6: Results

6.1 Overview

We are going to give a summary of the test result we achieved in this section. We will discuss and evaluate the outcome we have got from our model. We will analyze the findings from each steps of our project with graphs and figures. We will also compare our results with prebuilt models here.

6.2 Evaluation

Once we managed to make the model work, we needed to know if it was running properly. It could be manually tested that our model was able to predict the classes accurately. However, that wasn't enough to prove that our model indeed could perform efficiently and so we decided to apply some evaluation methods. For that purpose, we chose two methods:

I. Accuracy

II. Cross Entropy Loss

- Model Accuracy

The most common metric used for evaluating a model's classification efficiency is accuracy. It gives us the percentage of correctly predicted class by the model. It derives the results by taking the ratio of correctly predicted class by total prediction. In Figure below, we showed the model accuracy through each epoch. The highest accuracy our model managed to achieve was 96.70%.

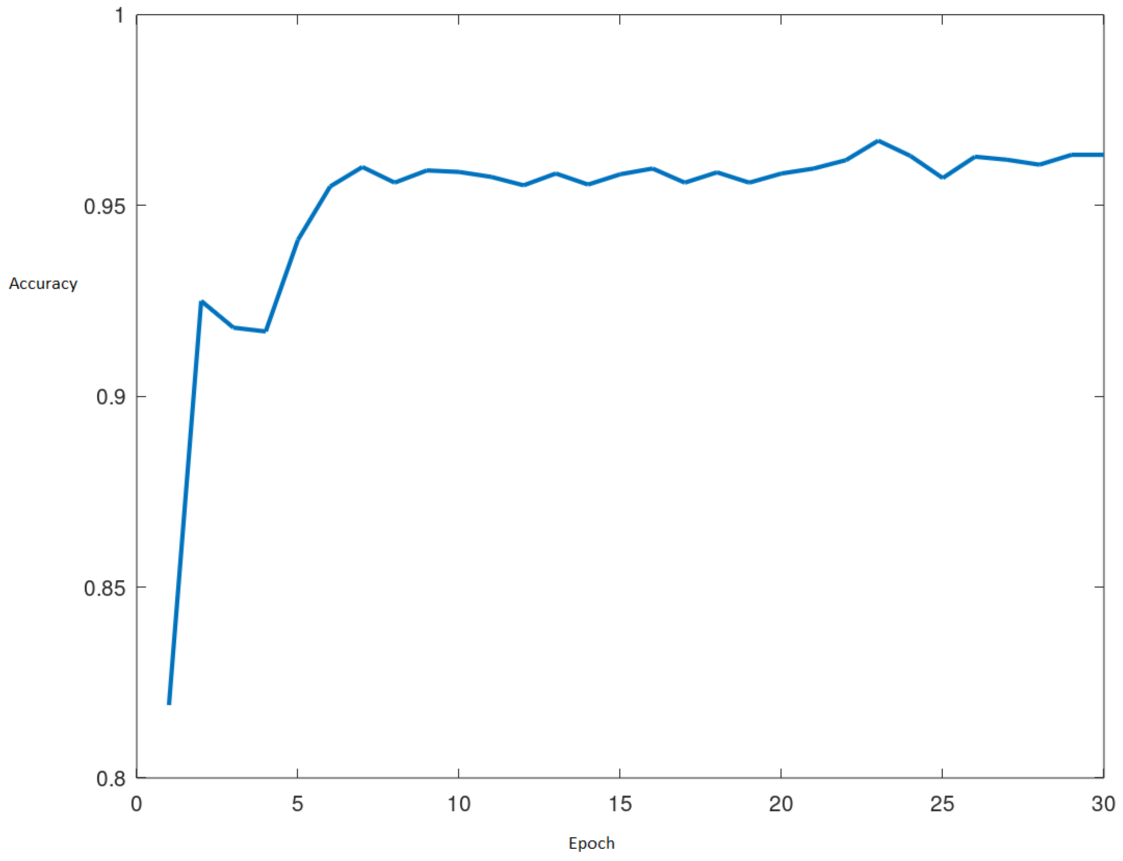


Figure 11: Model Accuracy for each epoch.

- Model Loss

Loss is a metric to measure the negative side of a model. It penalizes the model for wrong predictions. So, the less the loss, the better. The loss of our model through epoch 1 to 30 is shown below in Figure 12. For the accuracy of 96.70% our model yielded a loss of 13.61%.

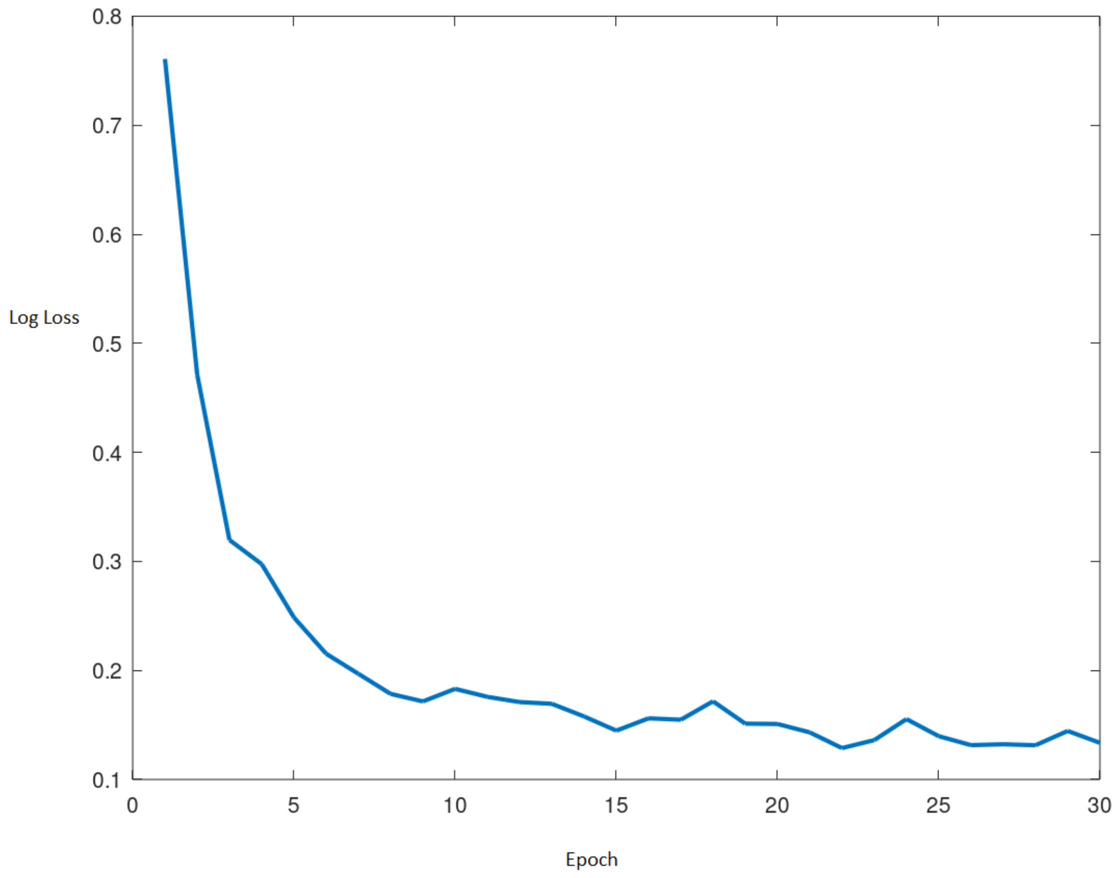


Figure 12: Model Loss for each epoch

6.3 Comparison

The evaluation methods told us the model was working fine, but it didn't make it clear exactly how well the model was performing. To know that we required a reference model so that we can compared our model with it to know where or model stood. We selected the VGG19 model as our reference model. We trained this pre-built model with our own dataset and ran the same evaluation methods on the results. Using the pertained vgg19 model we could achieve 97.63% accuracy whereas our model achieved a 96.70% accuracy. The loss on vgg19 was 13.14% and our model managed a loss of 13.61%. Table 02 gives the summary of the comparison

Model	**Accuracy (%)**	**Loss (%)**
Vgg19	97.63	13.14
Our Model	96.70	13.61

Table 02: vgg19 Model vs Our Model

6.4 Summary

The various test results have been thoroughly discussed in this chapter. This chapter shows us how well we have performed in building our model. We discuss not only our complete model but also all the failed attempts before our final models.

Chapter 7: Discussion

7.1 Overview

Making this project wasn't as easy as it looks. We face many obstacles in our way. We had to deal with many challenges and difficulties during this project. In this section, we will discuss those problems and explain briefly how we managed to overcome them. We'll also talk about some of the limitations related to this project. Finally, we will look at what we achieved from our work and how we can further progress our project in the future.

7.2 Challenges

- Data Collecting

Selecting the project, we are doing is the easiest task. Researching about the project and how we are going to approach to solve the problem was time-consuming but also an easy job. But We faced a tremendous challenge to collect data.

Our project's main agenda was to detect a disease from an image. For that agenda, we couldn't find any reliable source on the internet. We tried on our own to collect data from some farm near Dhaka. But everybody was unable to provide us any workable amount of data. Finally, Mr. A.P.M Alamgir Kabir from Giant Argo Processing Limited was able to help us assemble 24,162 healthy and infected images for Corn, Potato and Tomato.

- Layer Selection

Selecting the right number of layers was the most difficult decisions we had to take, as there no constrains or rule as to exactly how many layers there should be in a model. We had to go through a long and repetitive process to carefully pick the layers that we wanted to put in the model. We also had to calculate how many parameters there should be in each layer. This whole process took the majority of our project building time.

- Number of Epoch

Passing the dataset completely through a model sums up to one epoch. Selecting the number of epochs was another crucial decision we had to take. Because one epoch that is passing the dataset once wasn't enough as there was risk of underfitting. On the other hand, too many epochs could cause overfitting. A good place to stop is when passing the dataset doesn't bring much significant improvement. We observed 30 epoch was the appropriate place to stop.

7.3 Limitation

Training a model requires a huge amount of computation power. In each layer hundred lines of computation take place for each image. So, the machine that's training the model needs to be very powerful.

Unfortunately, the machines that we had were not as powerful as we required. The whole process proved to be very computationally expensive for our machines (laptops & pcs). So, we had to find an alternative way to avoid this problem.

- Training in Colab

One way was to run the process on graphics processing unit (GPU) rather than on central processing unit (CPU). Google Colab provided a great platform for GPU training. This gave us a significant boost on computation power. We were able to complete the training much faster this way.

- Batch Size

To make the computation even more easy we divided the dataset into some batches. This way we didn't have to pass the dataset at one go. Instead we sent a fraction of the dataset to reduce the pressure of computation. This divide and conquer method worked like a charm and we were able to complete the training process without any further obstacle.

7.4 Conclusion

Our model achieved the desired level of accuracy. It performed really well compared to other efficient models. It had almost the same accuracy level [96.70%] as the VGG model that we used as reference. The model was even able to identify disease in the early phases. We used a test set of images unknown to the model from each category to evaluate the model's precision and it passed on each of those. So, our goal to build a CNN classifier to detect plant disease was successful.

7.5 Future Work

Our model works perfectly for detecting a disease from an infected plant leaf. But there are many cases where the same plant can be affected by more than one disease. In such cases our model falls short as it can only detect one type of disease at a time. So, our future works includes adding functionality to our model so that if a leaf is infected by more than one disease, it can detect all of them separately. We also want to include a real-time disease detecting by building a mobile app.

Reference

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions”, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[2] "CS231n Convolutional Neural Networks for Visual Recognition", Cs231n.github.io, 2019. [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed: 02- Mar- 2019].

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[4] G. Huang, Z. Liu, L. van der Maaten and K. Weinberger, "Densely Connected Convolutional Networks", arXiv.org, 2016. [Online]. Available: https://arxiv.org/abs/1608.06993. [Accessed: 12- May- 2019].

[5] J. Amara, B. Bouaziz, and A. Algergawy, “A Deep Learning-based Approach for Banana Leaf Diseases Classification.”, Lecture Notes in Informatics (LNI), 2017. pp.79 - 85.

[6] E. C. Oerke, "Crop Losses to Pests.", The Journal of Agricultural Science., vol. 144, pp. 31 - 43, DOI. 10.1017/S0021859605005708, 2006.

[7]]R. Sheane, "The impact of plant pathogens on food security – 3Keel", 3keel.com, 2017. [Online]. Available: https://www.3keel.com/the-impact-of-plant-pathogens-on-food-security/. [Accessed: 27- March - 2019].

[8] A. Saxena, "Convolutional Neural Networks (CNNs): An Illustrated Explanation - XRDS", XRDS, 2019. [Online]. Available: https://blog.xrds.acm.org/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/. [Accessed: 28- May- 2019].

[9] K. Ferentinos, "Deep learning models for plant disease detection and diagnosis", Computers and Electronics in Agriculture, vol. 145, pp. 311-318, 2018.

[10] K. Golhani, S. Balasundram, G. Vadamalai and B. Pradhan, "A review of neural networks in plant disease detection using hyperspectral data", Information Processing in Agriculture, vol. 5, no. 3, pp. 354-371, 2018.

[11] M. Brahimi, K. Boukhalfa and A. Moussaoui, "Deep Learning for Tomato Diseases: Classification and Symptoms Visualization", Applied Artificial Intelligence, vol. 31, no. 4, pp. 299-315, 2017.

[12] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk and D. Stefanovic, "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification", Computational Intelligence and Neuroscience, vol. 2016, pp. 1-11, 2016.