Department of Electrical and Computer Engineering
North South University

Senior Design Project

# Plant Disease Detection and Solution for Rural Farmers Using Computer Vision, Cloud Computing and Android Platform

| | |
|---|---|
| **Mohammad Rashedul Alam** | **ID # 1511384042** |
| **Sakib Mukter** | **ID # 1520268042** |
| **Aminul Islam** | **ID # 1520523042** |

Faculty Advisor:

Md. Shahriar Karim

Assistant Professor

ECE Department

Spring, 2019

# 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.

.........................................................
Mohammad Rashedul Alam
ECE Department
North South University, Bangladesh

........................................................
Sakib Mukter
ECE Department
North South University, Bangladesh

.........................................................
Aminul Islam
ECE Department
North South University, Bangladesh

# Approval

The Senior Design Project entitled "**Plant Disease Detection and Solution for Rural Farmers Using Computer Vision, Cloud Computing and Android Platform**" by Mohammad Rashedul Alam (ID#1511384042), Sakib Mukter (ID#1520268042) and Aminul Islam (ID#1520523042) has been accepted as satisfactory and approved for partial fulfillment of the requirement of BS in CSE degree program on May, 2019.

**Supervisor's Signature**

**Md. Shahriar Karim**
**Assistant Professor**
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh.

**Department Chair's Signature**

**Dr. K. M. A. Salam**
**Professor & Chair**
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh.

# Acknowledgement

First of all, we would like to express our profound gratitude to our honorable course instructor, **Md. Shahrirar Karim** for his constant and meticulous supervision, valuable suggestions, his patience and encouragement to complete the thesis work.

We would also like to thank the ECE department of North South University for providing us with the opportunity to have an industrial level design experience as part of our curriculum for the undergraduate program.

Finally, we would like to thank our families and everybody who supported us and provided with guidance for the completion of this project.

## Abstract

Bangladesh is a densely populated country with considerably low per capita arable land, which makes a daunting task to grow sufficient food grains for about its 160 million people. Diseases prevalence and the lack of close monitoring often results in crop loss as high as 30% in some cases. For instance, rice production reduces by about 10% because of diseases, whereas potato and tomato production decreases by 37% and 43% respectively because of leaf infection. Early and accurate detection of these diseases can prevent a large-scale yield loss. However, detection of these diseases is hard for farmers without the direct help of skilled people. To provide the farmers with the initial information, we develop a voice assisted mobile app that can predict a possible set of diseases from the images of leaf-infection. The App is optimized against low-resolution images and includes voice assistance at its every step to ensure usability for the farmers, who are generally are uncomfortable with digital platforms. Also, the App also suggests possible remedies for the affected crops, and the information of sellers and distributors of pesticides, fertilizers, and other relevant commodities. Together, this App attempts to increase crops yield and is expected to act as the bridge between the sellers, distributors and the farmers living in the farthest corner of our country.

## *Table of Content*

# List of Figures

# List of Tables

# Chapter 1
# Project Overview

## 1.1 Introduction

We rely upon edible plants similarly as we rely upon oxygen. Without yields, there is no food, and without food, there is no life. It is no mishap that human development started to flourish with the creation of agriculture. Today, current innovation enables us to develop crops in amounts important for a relentless nourishment supply for billions of individuals. However, diseases remain a noteworthy risk to this supply, and a huge amount of harvests is lost every year due to different plant diseases. The circumstance is especially desperate for the 500 million smallholder farmers around the world, whose occupations rely upon the healthy progression of their harvests. In Africa alone, 80% of the agrarian yield originates from smallholder farmers. Same goes for Bangladesh as a major portion of our crops gets damaged every year due to plant diseases. Most of these diseases show symptoms on their leaves and can be identified accordingly. But an expert on identifying these diseases are not available widely. With billions of mobile phones now present worldwide, it would be extraordinary if the mobile phone could be transformed into a disease diagnostics instrument, perceiving infections from pictures it catches with its camera.  So, we have come up with a solution to make the disease prediction available to the local farmers via low-end smartphones. Farmers can easily take pictures of the leaves. The photo is then analyzed via machine learning approaches to identify the diseases.

## 1.2 Machine Learning and Computer Vision Basics

As the name suggests, Machine Learning (ML) enables a machine to learn autonomously based on experience, observations and examining patterns from a set of data without the need of explicitly programming. A predefined program only follows a set of instructions and is unable to handle new cases. But in machine learning a set of data is provided to the machine and the machine analyzes the dataset to find similar patterns and take decisions autonomously based on its observations. Computer vision is a field of study that helps the machine see and understand the content of any digital image and videos. The objective of computer vision is to comprehend the substance of digital pictures. Commonly, this includes creating strategies that attempt to recreate the ability of human vision.

## 1.2.1 How Machine Learning works

Training the machines include a basic procedure where each stage produces a better variant of the machine. The process can be showed as the below figure

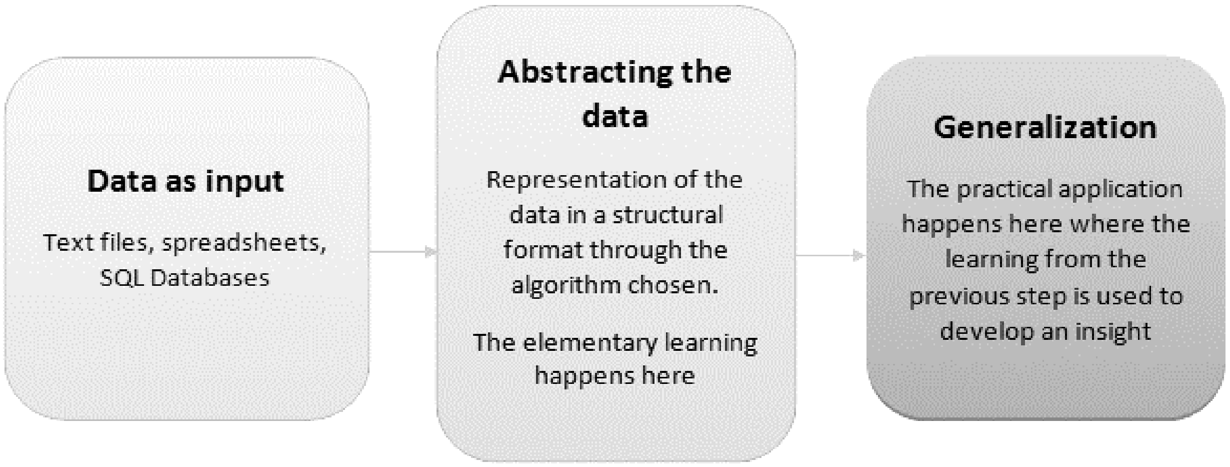

Data as input
Text files, spreadsheets, SQL Databases
Abstracting the data
Representation of the data in a structural format through the algorithm chosen.
The elementary learning happens here
Generalization
The practical application happens here where the learning from the previous step is used to develop an insight

Fig. 1.1. Machine learning working procedure

## 1.2.2 Applications of Machine Learning

Machine learning nowadays is used in most of the sector of our everyday life. We are using these products every day that utilizes the power of machine learning. Such examples are:

- Virtual Personal Assistants such as, Siri, Alexa, Google, are some of the popular virtual assistants that help to our searches, tasks, remainder, etc. Machine learning helps them getting better with our past involvement.
- Social Media Services uses machine learning for popular features such as face recognition and auto tag suggestion, suggesting contacts, etc.
- Email spam and malware filtering are done through machine learning, as most of the code for spam and malware are similar with negligible variations. Machine learning can easily detect spam and malware.

- Product Recommendations is one of popular feature for e-commerce. Machine learning helps to recommend relevant products to the customer by their activities and past actions. Each customer sees their unique necessary recommendations, in turn, rises up the sales.

### 1.2.3 Pros and Cons

As everything has its pros and cons machine learning also has its own advantages and disadvantages. The pros and cons are discussed below:

**Pros**:

- Easily identifies trends and patterns
- No human intervention needed (automation)
- Handling multi-dimensional and multi-variety data
- Wide Applications

**Cons:**

- Hard to acquire data
- Needs more time and computational power
- Higher error-susceptibility

### 1.2.4 List of popular Machine Learning Tools & Frameworks

Popular machine learning frameworks:

- TensorFlow
- Torch
- Caffe
- Theano
- Amazon Machine Learning
- Accord.Net
- Scikit-learn
- Apache Mahout
- Microsoft Cognitive Toolkit
- Keras

Popular machine learning tools:

- Kaggle
- Jupyter Notebook
- Anaconda

## 1.3 Our proposed project

The main idea of this project was to create an android app that can help the farmers of Bangladesh to easily detect plant diseases and take necessary steps. Low resolution images and less bandwidth usage had to be incorporated in the app for the farmer’s convenience. User friendly guideline and voice assistance is also incorporated to the ease discomfort of the farmers with digital platform.

### 1.3.1 Description of the idea:

The app that we have built can detect leaf disease with a high accuracy. Users can take photos of the affected area of the leaf or upload previous taken photos. Then a low-resolution version of the image is sent to servers. Machine learning model in the server analyzes and sends the output back to the app thus the less computing power is needed in mobile phone. After getting the result remedies and solutions are show to the user. Also, a few similar cases are shown in case of wrong detection.

**Features of the App:**

The android app has the following features

- Take Photos
- Voice Assistance
- Stepwise navigation tutorial
- Save Photos
- Choose from gallery
- Remedies and relevant information
- Low power and bandwidth consumption
- Video tutorial

### 1.3.2 Difficulties

The level of difficulty of this project was high, as we had to first experiment with different models and architecture to find out the approach that gives the best prediction. Different architecture produces distinct results with just a slight variation. So, we had to research different possibilities. Finding the best model that fits our dataset was a very daunting task. We explored different models and chose the best amongst them. Also, we had to pre-process and scale the image dataset to work with the model. Afterward, the model, the app and the database had to be synchronized to work smoothly for the disease prediction.

## 1.4 Motivation

With the recent development of the computation power of computers, machine learning techniques have become a popular field. We can accomplish many daunting and hard tasks with the help of machines easily. Detecting plant diseases from analyzing leaves is hard for uneducated farmers. Further complicating matter, skilled people aren’t always available. As the same leaf diseases follow a similar pattern, we can take the help of machines to solve this problem. The machine can find patterns and similarities for the same diseases and categorize them accordingly. When a machine is done learning patterns, it can help giving prediction to new unseen images. This idea motivated our project to take shape and encourage us to develop a complete solution for the farmers.

## 1.5 Summary

In this chapter, we have briefly described the basics of machine learning and computer vision. We also explained machine learning works, and the popular tools and frameworks are discussed too.

# Chapter 2
# Related work

## 2.1 Introduction

This project uses the idea of image classification with the help of machine learning. Nowadays this field is widely recognized and a lot of work is being done in the image classification field. But there are not that many projects that incorporate a complete solution for plant disease detection. To obtain a clear idea some of the existing works about image recognition, studied and discussed in this chapter.

## 2.2 Existing Work Related to Image Recognition

Image recognition is the process of identifying and detecting an object or an image. A machine learning approach to image recognition involves identifying and extracting key features from images and using those as inputs to a machine learning model. Image classification for face recognition, cat, dog, car and different object detection is very popular nowadays. In contrary, plant classification is not done to an extensive amount.

## 2.3 Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification

The latest generation of convolutional neural networks (CNNs) has achieved impressive results in the field of image classification. Implementing the appropriate management strategies like fungicide applications, disease-specific chemical applications, and vector control through pesticide applications could lead to early information on crop health and disease detection. This could control the diseases and improve productivity.

## 2.4 Detection and Classification of Rice Plant Diseases

Studies based on image recognition on plant disease were not enough. One of the studies is plant disease detection and recognition of diseases from plant leaf images. This recognition system involves feature extraction and classification of diseases using K-NN (K-Nearest Neighbors) and SVM (Support Vector Machine). One of the articles presents a prototype system for detection and classification of rice diseases based on the images of infected rice plants. Their prototype system is developed after detailed experimental analysis of various techniques used in image processing operations. Neural networks, with their outstanding ability to derive meaning from a complex or imperfect dataset, can be applied for extracting patterns and detecting trends that are too difficult to notice by humans or computer techniques. Other advantages of ANNs (Artificial Neural Network) are adaptive learning, self-organization, real-time operations, and so forth.

## 2.5 Summary

The existing work related to image recognition on mobile apps, image preprocess, shape detection, plant disease detection that we found useful have been briefly described in this section.

# Chapter 3
# Architecture of the system

## 3.1 Introduction

In this chapter we discuss about the architecture of this project. Precisely we discuss about the procedure, workflow, techniques, functionality etc. in this chapter.

## 3.2 Procedure and Functionality

Before describing the workflow of the system, how the system works has been explained first.

### 3.2.1 Procedure

The model has been trained with different diseases of different plants. For training, we have used 38971 images of infected leaves. We also trained this model with healthy leaves so that it can detect the healthy plants. First, the model takes images as input which we can also call image acquisition. Then the image has been pre-processed using median filter. Then the model visualizes the data of the image. Then extracts feature from the image which helps to identify which disease is occurred. Then this model predicts based on our trained model.

### 3.2.2 Functions

This project has some other functions rather than prediction a disease. It also shows some similar diseases based on prediction probability. Then it shows the remedy of particular disease. This model can also differentiate whether there is a leaf in image or not. If user takes pictures of other things instead of leaf, the model shows a notification message to take images of leaves so that it can predict the diseases of the plant.

## 3.3 Model Learning Flow

This diagram shows how this model learns to predict about the diseases of the different plants.

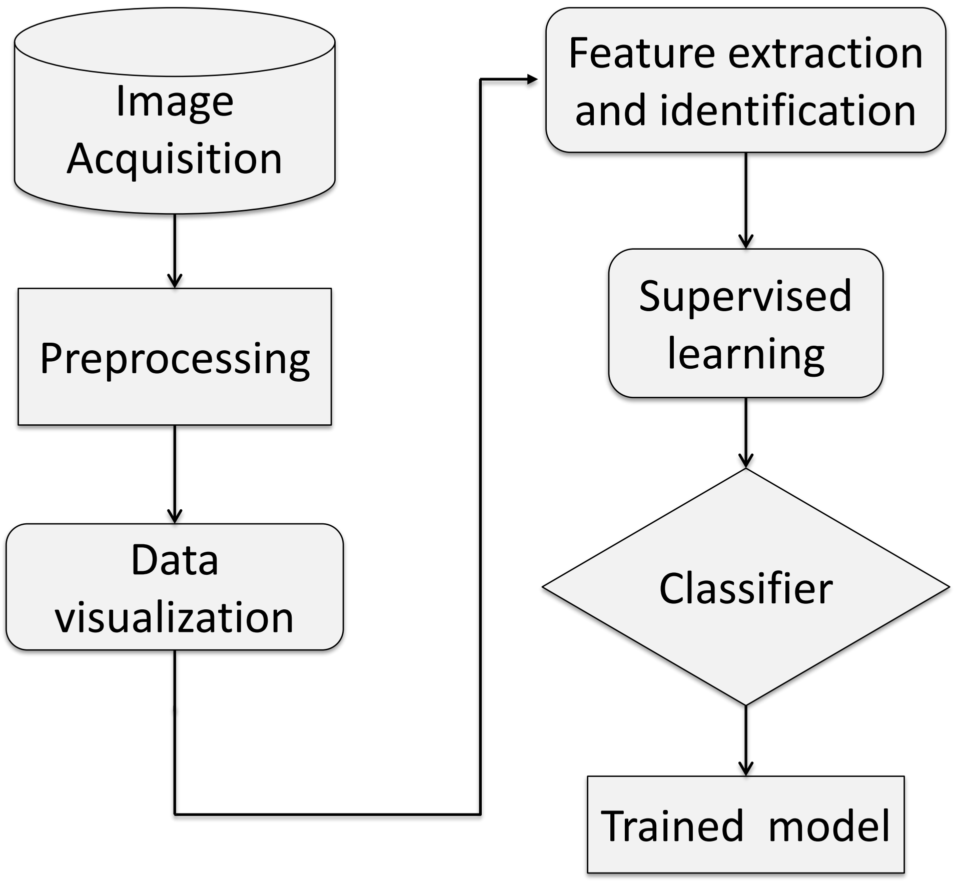


Fig. 3.1. Model learning method

Pre-labeled dataset of leaf images is pre-processed and scaled which is then used for visualization. Data visualization helps generating idea of feature extraction techniques. Lastly the processed data goes through a model which is discussed later on and a trained classifier is generated.

## 3.4 Overall System Architecture

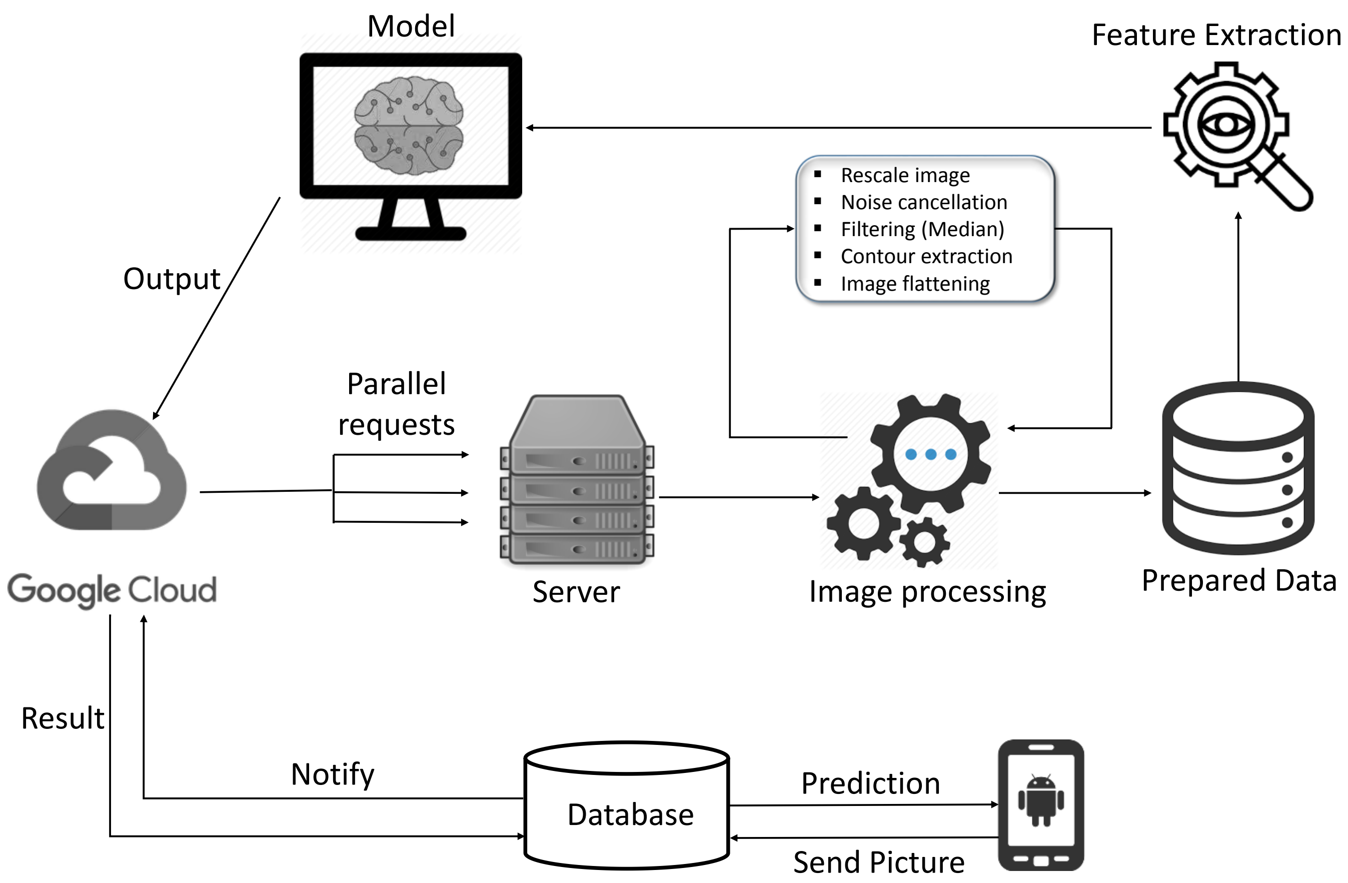


Fig. 3.2. Overall system architecture

Captured images from the device is sent to the database. The database notifies the cloud server. Then the cloud server fetches the images and does some pre-processing and scaling. The processed image is given as input to the model. The model does feature extraction and predicts the disease class, the output is then again sent to the database via cloud server. Android app show the result from the database. Detailed description of these components is explained in later chapters.

## 3.5 Summary

In this chapter we discussed about the architecture of this project. We tried to explain workflow briefly through the diagrams. We think that this a good way to know about the overall system of the project instead of writing an essay about the system architecture.

# Chapter 4
# The Model used in this project

## 4.1 Introduction

Different architecture and algorithm of machine learning is tested for this project. The machine learning model is the core of this project. The accuracy of the prediction depends on the model's architecture.

## 4.2 AlexNet

AlexNet is a popular architecture for image classification problems. AlexNet consists of 5 Convolutional Layers and 3 Fully Connected Layers. It has 60 million parameters and 650,000 neurons.

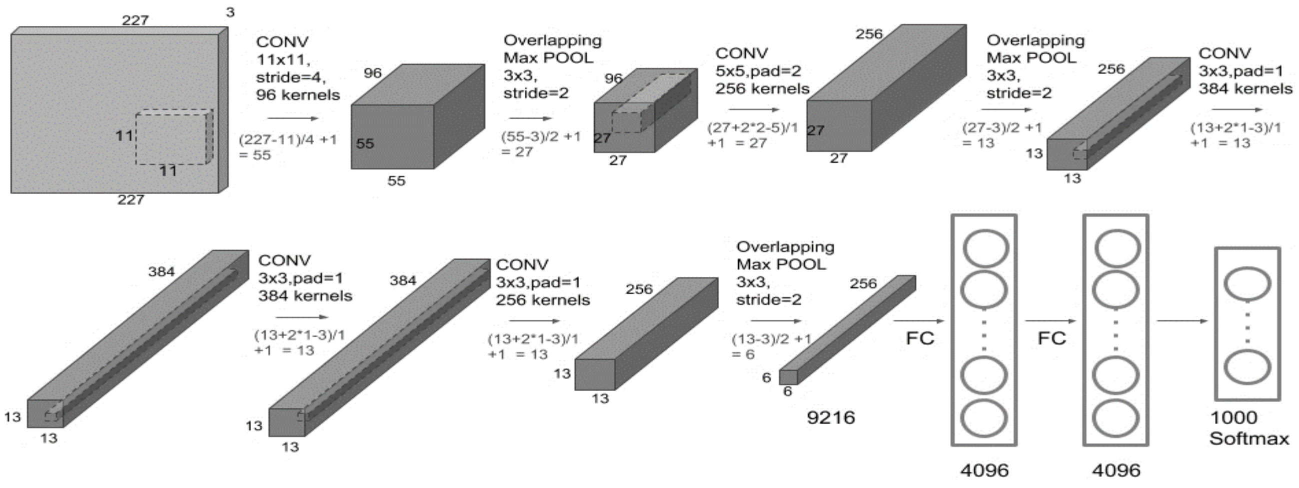


Fig. 4.1. Architecture of AlexNet

Various Convolutional Kernels (a.k.a kernels) extract features in a picture. In a solitary convolutional layer, there are generally numerous kernels of a similar size. For instance, the first Conv Layer of AlexNet contains 96 parts of size 11x11x3. Note the width and height of the kernel are typically the same and the depth is equivalent to the quantity of channels.

Using AlexNet for this project we got an accuracy of 86.08%.

## 4.3 Transfer learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It's a popular mechanism used in machine learning as the pre-trained models are used as starting point for new models.

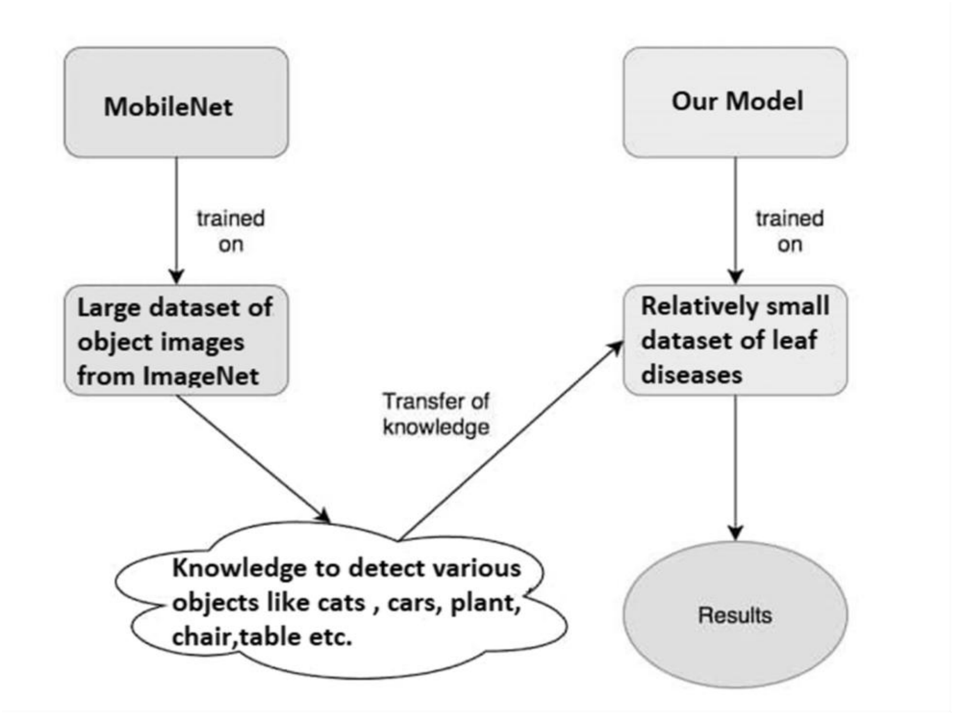

MobileNet
trained on
Large dataset of object images from ImageNet
Our Model
trained on
Relatively small dataset of leaf diseases
Transfer of knowledge
Knowledge to detect various objects like cats , cars, plant, chair,table etc.
Results

Fig. 4.2. How transfer learning works

## 4.3 MobileNet

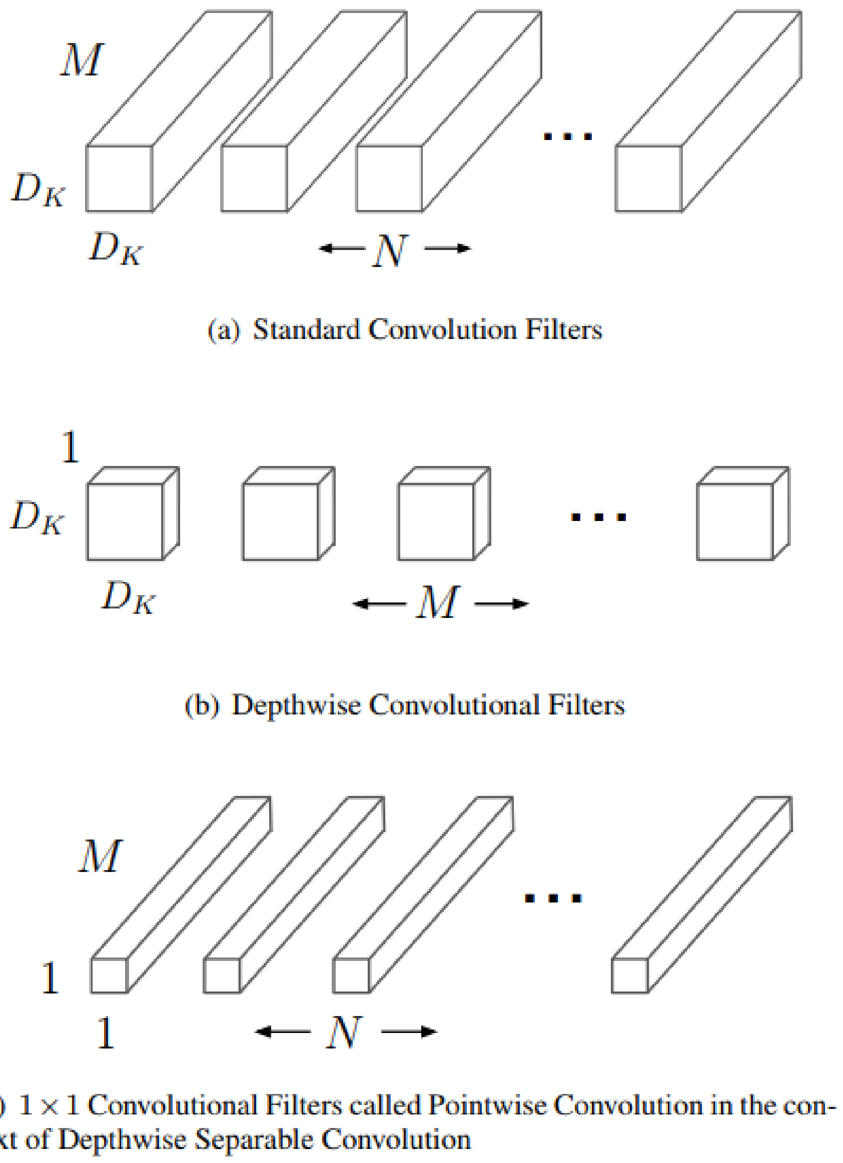
MobileNet uses depthwise separable convolutions. This convolution block was at first introduced by Xception. A depthwise separable convolution is made of two operations: a depthwise convolution and a pointwise convolution. MobileNet is pre-trained on the ImageNet dataset.

The overall architecture of the Mobilenet is as follows, having 30 layers with:

1. Convolutional layer with stride 2
2. Depthwise layer
3. Pointwise layer that doubles the number of channels

4. Depthwise layer with stride 2

5. Pointwise layer that doubles the number of channels



$M$
$D_K$
$D_K$
$\leftarrow N \rightarrow$
(a) Standard Convolution Filters
1
$D_K$
$D_K$
$\leftarrow M \rightarrow$
(b) Depthwise Convolutional Filters
$M$
1
1
$\leftarrow N \rightarrow$
(c) $1 \times 1$ Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Fig. 4.3. Difference between pointwise and depth wise convolutions

## 4.4 Architecture of MobileNet

MobileNet has the following architecture:

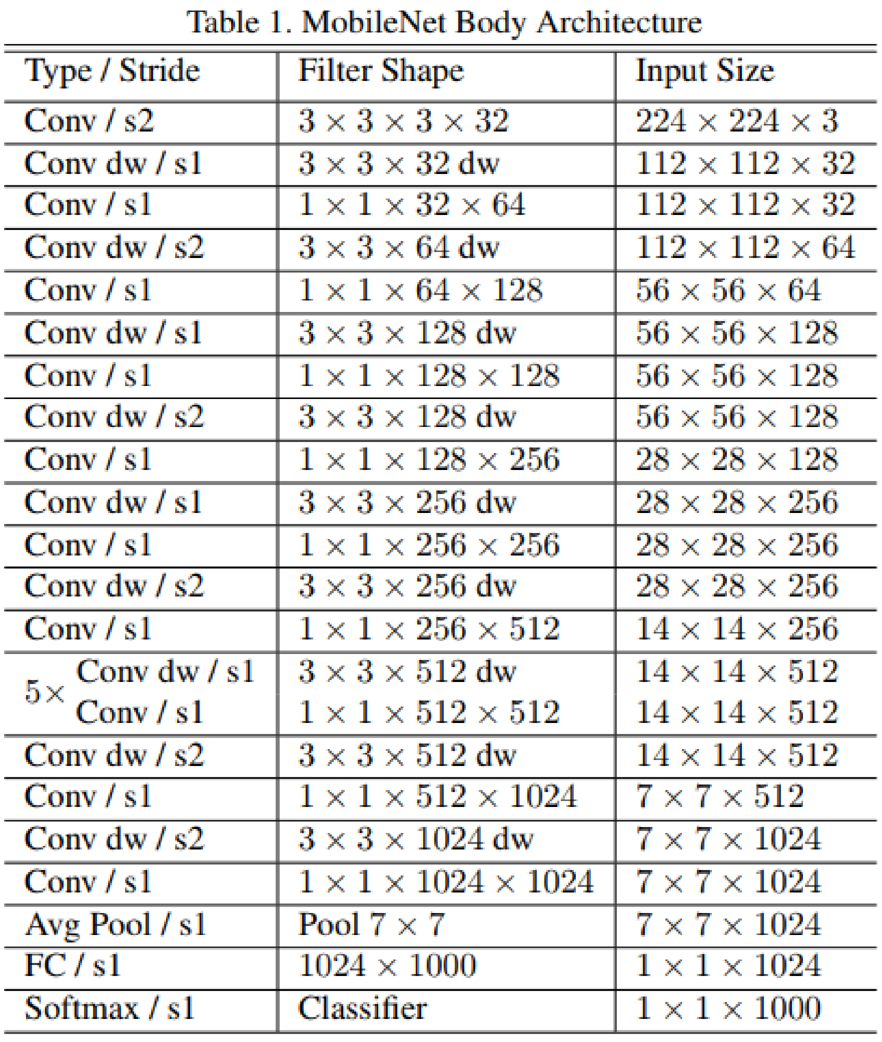


Table 1. MobileNet Body Architecture

| Type / Stride | Filter Shape | Input Size |
|---|---|---|
| Conv / s2 | $3 \times 3 \times 3 \times 32$ | $224 \times 224 \times 3$ |
| Conv dw / s1 | $3 \times 3 \times 32$ dw | $112 \times 112 \times 32$ |
| Conv / s1 | $1 \times 1 \times 32 \times 64$ | $112 \times 112 \times 32$ |
| Conv dw / s2 | $3 \times 3 \times 64$ dw | $112 \times 112 \times 64$ |
| Conv / s1 | $1 \times 1 \times 64 \times 128$ | $56 \times 56 \times 64$ |
| Conv dw / s1 | $3 \times 3 \times 128$ dw | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 128$ | $56 \times 56 \times 128$ |
| Conv dw / s2 | $3 \times 3 \times 128$ dw | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 256$ | $28 \times 28 \times 128$ |
| Conv dw / s1 | $3 \times 3 \times 256$ dw | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 256$ | $28 \times 28 \times 256$ |
| Conv dw / s2 | $3 \times 3 \times 256$ dw | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 512$ | $14 \times 14 \times 256$ |
| 5× Conv dw / s1 | $3 \times 3 \times 512$ dw | $14 \times 14 \times 512$ |
| Conv / s1 | $1 \times 1 \times 512 \times 512$ | $14 \times 14 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 512$ dw | $14 \times 14 \times 512$ |
| Conv / s1 | $1 \times 1 \times 512 \times 1024$ | $7 \times 7 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 1024$ dw | $7 \times 7 \times 1024$ |
| Conv / s1 | $1 \times 1 \times 1024 \times 1024$ | $7 \times 7 \times 1024$ |
| Avg Pool / s1 | Pool $7 \times 7$ | $7 \times 7 \times 1024$ |
| FC / s1 | $1024 \times 1000$ | $1 \times 1 \times 1024$ |
| Softmax / s1 | Classifier | $1 \times 1 \times 1000$ |

Fig. 4.4. MobileNet body architecture

## 4.5 Project model architecture

The architecture of the model for this project is based on the MobileNet. The last 1000 neuron layer are discarded from the original model. We trained those last layers with our dataset with 26 classes. The model summary is given below:

```
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, None, None, 3)     0         
_________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, None, None, 3)     0         
_________________________________________________________________
conv1 (Conv2D)               (None, None, None, 32)    864       
_________________________________________________________________
conv1_bn (BatchNormalization (None, None, None, 32)    128       
_________________________________________________________________
conv1_relu (ReLU)            (None, None, None, 32)    0         
_________________________________________________________________
conv_dw_1 (DepthwiseConv2D)  (None, None, None, 32)    288       
_________________________________________________________________
conv_dw_1_bn (BatchNormaliza (None, None, None, 32)    128       
_________________________________________________________________
conv_dw_1_relu (ReLU)        (None, None, None, 32)    0         
_________________________________________________________________
conv_pw_1 (Conv2D)           (None, None, None, 64)    2048      
_________________________________________________________________
conv_pw_1_bn (BatchNormaliza (None, None, None, 64)    256       
_________________________________________________________________
conv_pw_1_relu (ReLU)        (None, None, None, 64)    0         
_________________________________________________________________
conv_pad_2 (ZeroPadding2D)   (None, None, None, 64)    0         
_________________________________________________________________
conv_dw_2 (DepthwiseConv2D)  (None, None, None, 64)    576       
_________________________________________________________________
conv_dw_2_bn (BatchNormaliza (None, None, None, 64)    256       
_________________________________________________________________
conv_dw_2_relu (ReLU)        (None, None, None, 64)    0         
_________________________________________________________________
conv_pw_2 (Conv2D)           (None, None, None, 128)   8192      
_________________________________________________________________
conv_pw_2_bn (BatchNormaliza (None, None, None, 128)   512       
_________________________________________________________________
conv_pw_2_relu (ReLU)        (None, None, None, 128)   0         
_________________________________________________________________
conv_dw_3 (DepthwiseConv2D)  (None, None, None, 128)   1152      
_________________________________________________________________
conv_dw_3_bn (BatchNormaliza (None, None, None, 128)   512       
_________________________________________________________________
conv_dw_3_relu (ReLU)        (None, None, None, 128)   0         
_________________________________________________________________
conv_pw_3 (Conv2D)           (None, None, None, 128)   16384     
_________________________________________________________________
conv_pw_3_bn (BatchNormaliza (None, None, None, 128)   512       
_________________________________________________________________
conv_pw_3_relu (ReLU)        (None, None, None, 128)   0         
_________________________________________________________________
conv_pad_4 (ZeroPadding2D)   (None, None, None, 128)   0         
_________________________________________________________________
conv_dw_4 (DepthwiseConv2D)  (None, None, None, 128)   1152      
_________________________________________________________________
conv_dw_4_bn (BatchNormaliza (None, None, None, 128)   512       
_________________________________________________________________
conv_dw_4_relu (ReLU)        (None, None, None, 128)   0         
```

```
conv_pw_4 (Conv2D)           (None, None, None, 256)   32768
_________________________________________________________________
conv_pw_4_bn (BatchNormaliza (None, None, None, 256)   1024
_________________________________________________________________
conv_pw_4_relu (ReLU)        (None, None, None, 256)   0
_________________________________________________________________
conv_dw_5 (DepthwiseConv2D)  (None, None, None, 256)   2304
_________________________________________________________________
conv_dw_5_bn (BatchNormaliza (None, None, None, 256)   1024
_________________________________________________________________
conv_dw_5_relu (ReLU)        (None, None, None, 256)   0
_________________________________________________________________
conv_pw_5 (Conv2D)           (None, None, None, 256)   65536
_________________________________________________________________
conv_pw_5_bn (BatchNormaliza (None, None, None, 256)   1024
_________________________________________________________________
conv_pw_5_relu (ReLU)        (None, None, None, 256)   0
_________________________________________________________________
conv_pad_6 (ZeroPadding2D)   (None, None, None, 256)   0
_________________________________________________________________
conv_dw_6 (DepthwiseConv2D)  (None, None, None, 256)   2304
_________________________________________________________________
conv_dw_6_bn (BatchNormaliza (None, None, None, 256)   1024
_________________________________________________________________
conv_dw_6_relu (ReLU)        (None, None, None, 256)   0
_________________________________________________________________
conv_pw_6 (Conv2D)           (None, None, None, 512)   131072
_________________________________________________________________
conv_pw_6_bn (BatchNormaliza (None, None, None, 512)   2048
_________________________________________________________________
conv_pw_6_relu (ReLU)        (None, None, None, 512)   0
_________________________________________________________________
conv_dw_7 (DepthwiseConv2D)  (None, None, None, 512)   4608
_________________________________________________________________
conv_dw_7_bn (BatchNormaliza (None, None, None, 512)   2048
_________________________________________________________________
conv_dw_7_relu (ReLU)        (None, None, None, 512)   0
_________________________________________________________________
conv_pw_7 (Conv2D)           (None, None, None, 512)   262144
_________________________________________________________________
conv_pw_7_bn (BatchNormaliza (None, None, None, 512)   2048
_________________________________________________________________
conv_pw_7_relu (ReLU)        (None, None, None, 512)   0
_________________________________________________________________
conv_dw_8 (DepthwiseConv2D)  (None, None, None, 512)   4608
_________________________________________________________________
conv_dw_8_bn (BatchNormaliza (None, None, None, 512)   2048
_________________________________________________________________
conv_dw_8_relu (ReLU)        (None, None, None, 512)   0
_________________________________________________________________
conv_pw_8 (Conv2D)           (None, None, None, 512)   262144
_________________________________________________________________
conv_pw_8_bn (BatchNormaliza (None, None, None, 512)   2048
_________________________________________________________________
conv_pw_8_relu (ReLU)        (None, None, None, 512)   0
_________________________________________________________________
conv_dw_9 (DepthwiseConv2D)  (None, None, None, 512)   4608
```

```
________________________________________________________________________
conv_dw_9_bn (BatchNormaliza (None, None, None, 512)   2048
________________________________________________________________________
conv_dw_9_relu (ReLU)        (None, None, None, 512)   0
________________________________________________________________________
conv_pw_9 (Conv2D)            (None, None, None, 512)   262144
________________________________________________________________________
conv_pw_9_bn (BatchNormaliza (None, None, None, 512)   2048
________________________________________________________________________
conv_pw_9_relu (ReLU)        (None, None, None, 512)   0
________________________________________________________________________
conv_dw_10 (DepthwiseConv2D) (None, None, None, 512)   4608
________________________________________________________________________
conv_dw_10_bn (BatchNormaliz (None, None, None, 512)   2048
________________________________________________________________________
conv_dw_10_relu (ReLU)       (None, None, None, 512)   0
________________________________________________________________________
conv_pw_10 (Conv2D)          (None, None, None, 512)   262144
________________________________________________________________________
conv_pw_10_bn (BatchNormaliz (None, None, None, 512)   2048
________________________________________________________________________
conv_pw_10_relu (ReLU)       (None, None, None, 512)   0
________________________________________________________________________
conv_dw_11 (DepthwiseConv2D) (None, None, None, 512)   4608
________________________________________________________________________
conv_dw_11_bn (BatchNormaliz (None, None, None, 512)   2048
________________________________________________________________________
conv_dw_11_relu (ReLU)       (None, None, None, 512)   0
________________________________________________________________________
conv_pw_11 (Conv2D)          (None, None, None, 512)   262144
________________________________________________________________________
conv_pw_11_bn (BatchNormaliz (None, None, None, 512)   2048
________________________________________________________________________
conv_pw_11_relu (ReLU)       (None, None, None, 512)   0
________________________________________________________________________
conv_pad_12 (ZeroPadding2D)  (None, None, None, 512)   0
________________________________________________________________________
conv_dw_12 (DepthwiseConv2D) (None, None, None, 512)   4608
________________________________________________________________________
conv_dw_12_bn (BatchNormaliz (None, None, None, 512)   2048
________________________________________________________________________
conv_dw_12_relu (ReLU)       (None, None, None, 512)   0
________________________________________________________________________
conv_pw_12 (Conv2D)          (None, None, None, 1024)  524288
________________________________________________________________________
conv_pw_12_bn (BatchNormaliz (None, None, None, 1024)  4096
________________________________________________________________________
conv_pw_12_relu (ReLU)       (None, None, None, 1024)  0
________________________________________________________________________
conv_dw_13 (DepthwiseConv2D) (None, None, None, 1024)  9216
________________________________________________________________________
conv_dw_13_bn (BatchNormaliz (None, None, None, 1024)  4096
________________________________________________________________________
conv_dw_13_relu (ReLU)       (None, None, None, 1024)  0
________________________________________________________________________
conv_pw_13 (Conv2D)          (None, None, None, 1024)  1048576
________________________________________________________________________
conv_pw_13_bn (BatchNormaliz (None, None, None, 1024)  4096
_________________________________________________________________
conv_pw_13_relu (ReLU)       (None, None, None, 1024)  0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_3 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_4 (Dense)              (None, 26)                13338
=================================================================
```

Table. 4.1. Project model architecture

Total parameters: 5,866,202

Trainable parameters: 5,844,314

Non-trainable parameters: 21,888

Using this model, we got the maximum accuracy of 96.08%.

## 4.6 Data Augmentation

Machine learning models such as Convolutional Neural Network (CNN) requires a lot images train. Typically, any dataset less than 10000 is considered a small dataset. Acquiring this large dataset isn't always possible. We overcame this with the help of data augmentation. Data augmentation is done by applying random crops, mirroring, flipping the images. This way a large number of images can be created. We created 31088 images from our original 7500 images belonging to 26 classes. Data augmentation also help reducing overfitting.

## 4.7 Summary

In this chapter we discussed about the models we used for this project. We tried to explain MobileNet, AlexNet, transfer learning, data augmentation briefly.

# Chapter 5
# Theory

## 5.1 Introduction

The details of the theory of our system are discussed in this chapter. The theoretical explanation is divided into following sections:

- Machine Learning
- Convolutional Neural Network (CNN)

## 5.2 Machine Learning

Tom Mitchell (1998) provides a modern definition for machine learning: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

There are mainly two kind of machine learning techniques:

- Supervised machine learning: The program is given and trained on a known dataset with labels.
- Unsupervised machine learning: The program is given a bunch of data and must find patterns and relationships therein.
- Others: Reinforcement learning, recommender systems.

The machine learning technique used here is Convolutional Neural Network (CNN) a supervised machine learning technique.

## 5.3 Convolutional Neural Network (CNN)

The Convolutional Neural Networks (CNNs) is a sort of mathematical structure for investigation of datasets, pictures, etc. CNN take an input image, assign importance (weights and biases) to various objects in the image and can differentiate one from the other. The architecture of CNN was inspired by the organization of the visual cortex of human brain.

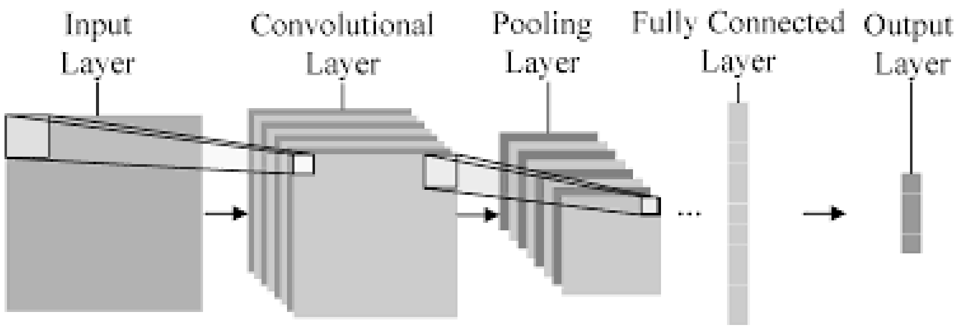


Fig. 5.1. Convolutional Neural Network

The Convolutional Neural Network has many layers. They serve different purposes. The layers are discussed below:

## 5.3.1 Convolutional Layers

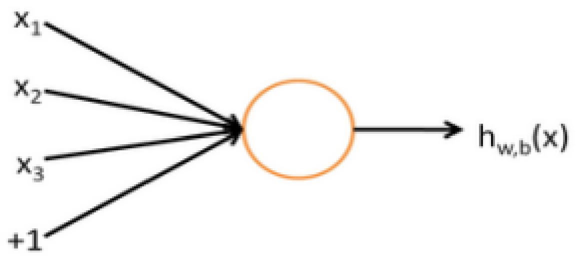


Fig. 5.2. Single computing unit in convolutional layer

Here, “Xn” is the input function. The circle is a computing unit and has a weighted function which can be regarded as a filter. The equation of the function is:

$$h_{W,b}(x) = f(W^T x) = f\left(\sum_{i=1}^{3} W_i x_i + b\right) \quad (1)$$

In the equation (1), “W” is weighted functions designed by programmers, for some time it is a “convolutional window”, which is a small matrix and elements of the matrix are depended on processes requests. “b” is a fixed element belonged to weighted function, which can be removed sometimes.

Convolution layer has kernels or convolution window which are designed as smaller pixel maps according to need.

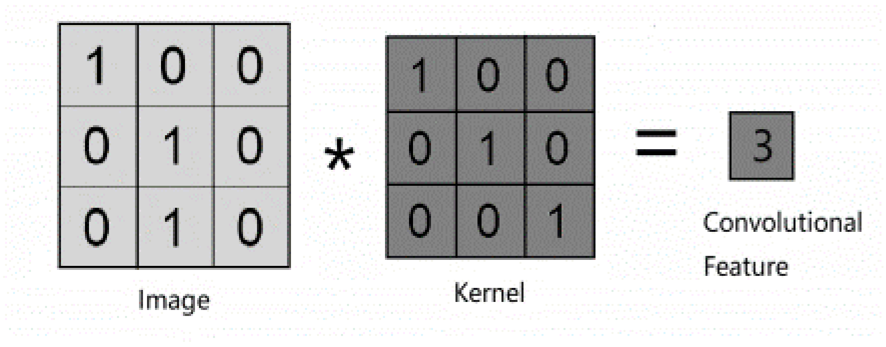


Fig. 5.3. An example of convolution

The convolution is the total of the items between each picture pixel and kernel pixel. For instance, in figure-5.3, every one of 9 pixels can be separated as 1 pixel to extract feature value. The higher estimation of result implies this current region's information has a higher association with a convolutional window so that kernel can be regarded as a filter. Other than different qualities in kernels pixels, the convolution procedure is likewise changed by edge and stride. Edge is straightforward it is only the length of the side of convolution window. It depicts the measure of portions. At the point when a kernel completes a convolution procedure with a piece of input data, it should move to somewhere else for doing the following convolution.

### 5.3.2 Pooling Layers

Like the Convolutional Layer, the Pooling layer oversees decreasing the spatial size of the Convolved Feature. This is to diminish the computational power required to process the information through dimensionality decrease. Besides, it is helpful for extracting dominant features which are rotational and positional invariant, along these lines keeping up the procedure of successfully training of the model.

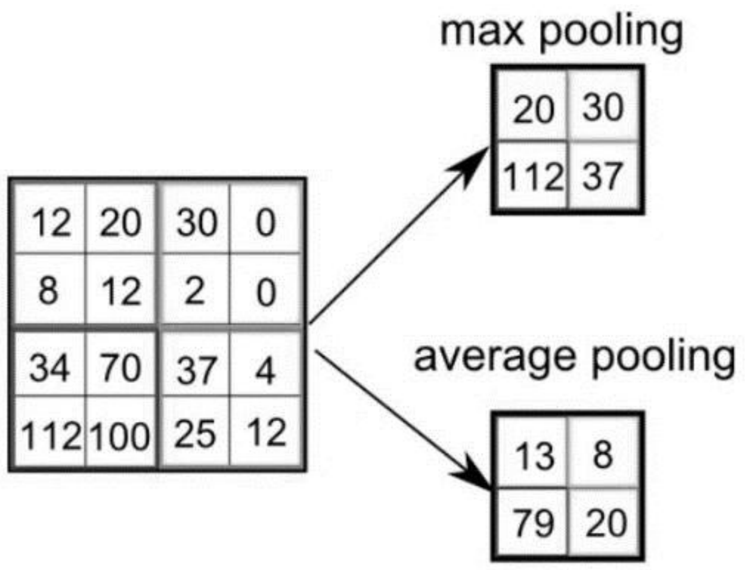


Fig. 5.4. Types of pooling

## 5.4 Summary

In this chapter we have discussed about the theories we used in this project. We tried to discuss about machine learning, neural network, convolutional neural network, pooling layers thoroughly.

# Chapter 6
# Dataset

## 6.1 Introduction

The core element of the project the machine learning model needs a huge amount of dataset for training and learning. The model finds similarities and learns patterns from the dataset and accordingly classifies them. Human beings also learn to identify things by observing the object first. Machine learning inspired by human brain follows this same nature.

## 6.2 Plant Village Dataset

PlantVillage is a nonprofit project by Penn State University in the US and EPFL in Switzerland. They have collected - and are continuing to collect - tens of thousands of images of diseased and healthy crops. The goal of PlantVillage Disease Classification challenge is to develop algorithms that can accurately diagnose a disease based on an image.

## 6.3 Plant Village Dataset Description

PlantVillage dataset has 38 classes of crops disease pairs. The data records contain 54,309 images. The images span 14 crop species: Apple, Blueberry, Cherry, Corn, Grape, Orange, Peach, Bell Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, Tomato. In contains images of 17 fungal diseases, 4 bacterial diseases, 2 mold (oomycete) diseases, 2 viral disease, and 1 disease caused by a mite. 12 crop species also have images of healthy leaves that are not visibly affected by a disease.

## 6.4 Filtered Dataset

As the plant village dataset has a lot of irrelevant crops which aren't important for crop production Bangladesh we have disregarded those and arranged the dataset according to our needs. We have kept 5 major crops and their disease containing 25 classes. These crops are Apple, Tomato, Potato, Corn, Grape. For detecting irrelevant pictures and leaves we have added one more class with non-leaf objects and pictures.  So, we have a total of 26 classes for our dataset containing healthy and disease effected images. The total images in this dataset is 7500 images.

## 6.5 Dataset Example

The dataset has clear close pictures of the leaves and diseases as below. Each disease is identifiable at different stage.

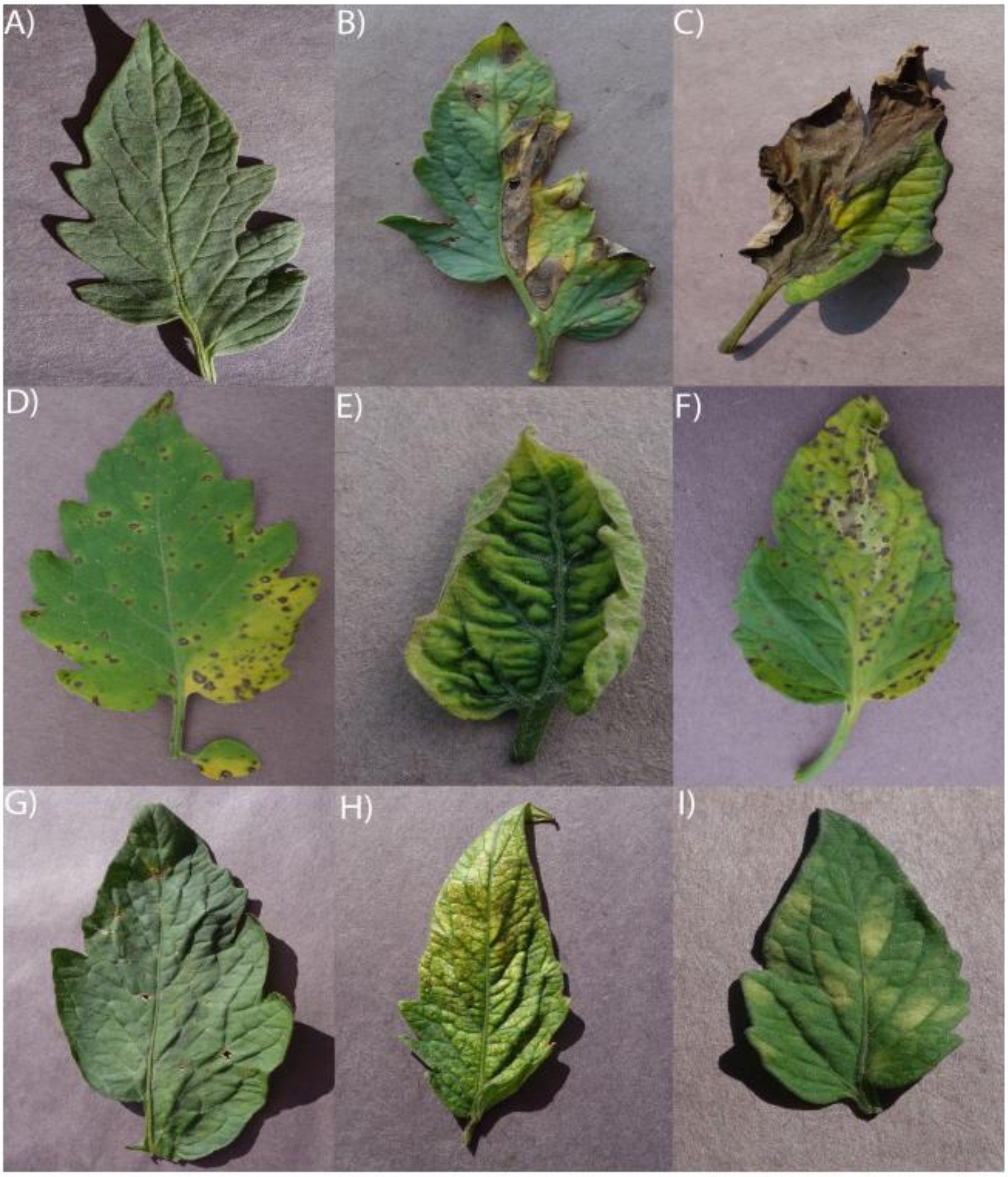


Fig. 6.1. Different tomato diseases

Examples of different phenotypes of tomato plants. A) Healthy leaf B) Early Blight (Alternaria solani) C) Late Blight (Phytophthora Infestans) D) Septoria Leaf Spot (Septoria lycopersici) E) Yellow Leaf Curl Virus (Family Geminiviridae genus Begomovirus) F) Bacterial Spot (Xanthomonas campestris pv. vesicatoria) G) Target Spot (Corynespora cassiicola) H) Spider Mite (Tetranychus urticae) Damage

## 6.6 Dataset Processing

The images in the dataset needs preprocessing to remove backgrounds. We have done that through segmentation which is discussed in later chapter. Also, noisy and blurry images in the dataset are corrected using median filtering. As the images are similar type the models have a high probability of overfitting. We have avoided such cases with the help of Data Augmentation and Dropout.

## 6.7 Summary

In this chapter the dataset is described and how the dataset is arranged and processed is also described.

# Chapter 7
# Image Preprocessing

## 7.1 Introduction

Image pre-processing is a technique to remove noise from images. Most of the time images contains noise. Images may be blurred and distorted. These noises need to be pre-processed to get better result.

## 7.2 Image processing techniques used

We have tested and used different kinds of image processing techniques in this project. These are discussed below:

## 7.3 Deconvolution

Deconvolution corrects the systematic error of blur (loss of contrast in smaller features) in optical systems such as fluorescence microscopy images. A good point spread function (PSF) increases the accuracy of deconvolution. The PSF in many contexts can be thought of as the extended blob in an image that represents an unresolved object.

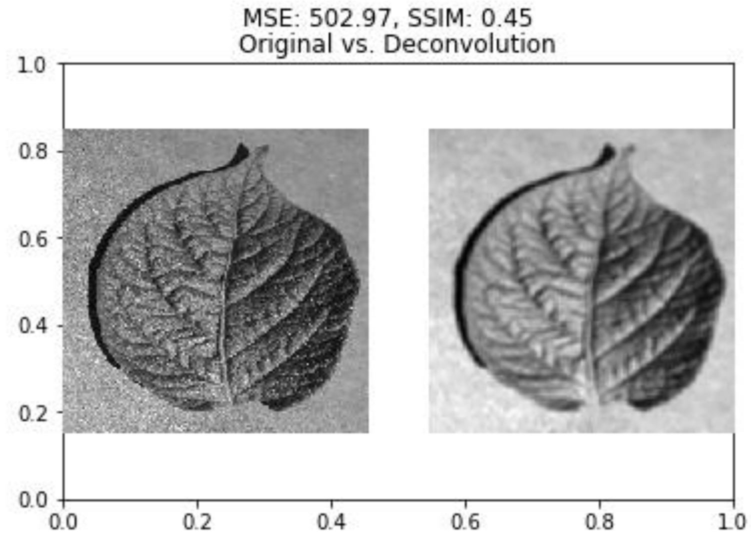


Fig. 7.1. Original vs Deconvolution

The result of applying deconvolution in our dataset we got an Mean Squared Error (MSE) of 502.97 and The Structural Similarity Index (SSIM) of 0.45. Deconvolution works well for microscopic images. But didn’t produce the best result for this project.

## 7.4 Median Filtering

Median Filter is a non-linear digital filtering technique, often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). The median is calculated by first sorting all the pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value.

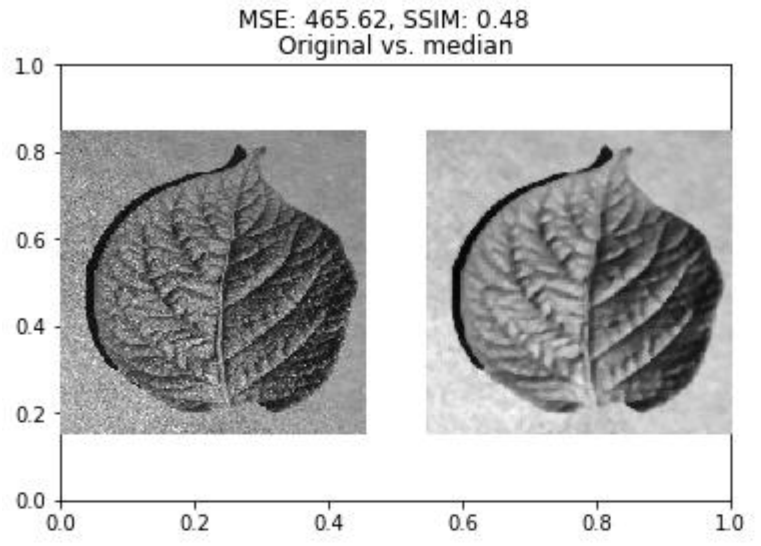


Fig. 7.2. Original vs Median

The result of applying median filtering in our dataset we got an Mean Squared Error (MSE) of 465.62 and The Structural Similarity Index (SSIM) of 0.48. This technique performs best for our dataset.

## 7.5 Image Segmentation

Image segmentation is a technique to extract objects from an image. Objects can be identified and extracted by using image segmentation. Typically, images are stored in RGB (red, green, and blue) color space. HSV (hue, saturation, value) is an alternative representation of the RGB color model which is easier to perform segmentation. By converting color and applying threshold we can segment the image which contains the leaves.

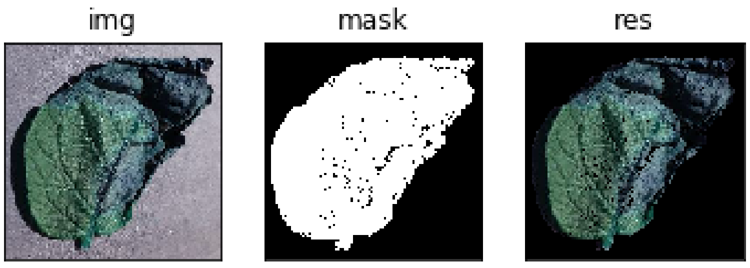


Fig. 7.3. Segmented leaf images

We also have a non-leaf class in the machine learning model that further does leaf detection for segmentation.

## 7.6 Summary

In this chapter, the image processing techniques and result are discussed.

# Chapter 8
# Tensorflow & Keras API

## 8.1 Introduction

This chapter discusses the TensorFlow and Keras API, which has been used in this project. We'll also discuss the differences between related technologies in this chapter.

## 8.2 TensorFlow

**T**ensorFlow is a free and open source library for dataflow and differentiable programming across a spread of tasks. it's a symbolic scientific discipline library and is additionally used for machine learning applications like neural networks. It's used for each analysis and production at Google. It was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015. It offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by exploitation the high-level Keras API, that makes obtaining started with TensorFlow and machine learning straightforward

## 8.3 Keras API

Keras is an open-source neural-network library written in Python. It can be running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. For deep neural networks, it allows faster experimentation. It is user-friendly and extensible. In 2017, it took place in TensorFlow's core library. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. It supports Convolutional Neural Network (CNN) and Recurrent Neural Networks (RNN). It also supports other common utility layers like dropout, batch normalization, and pooling.

## 8.4 Tensorflow vs PyTorch

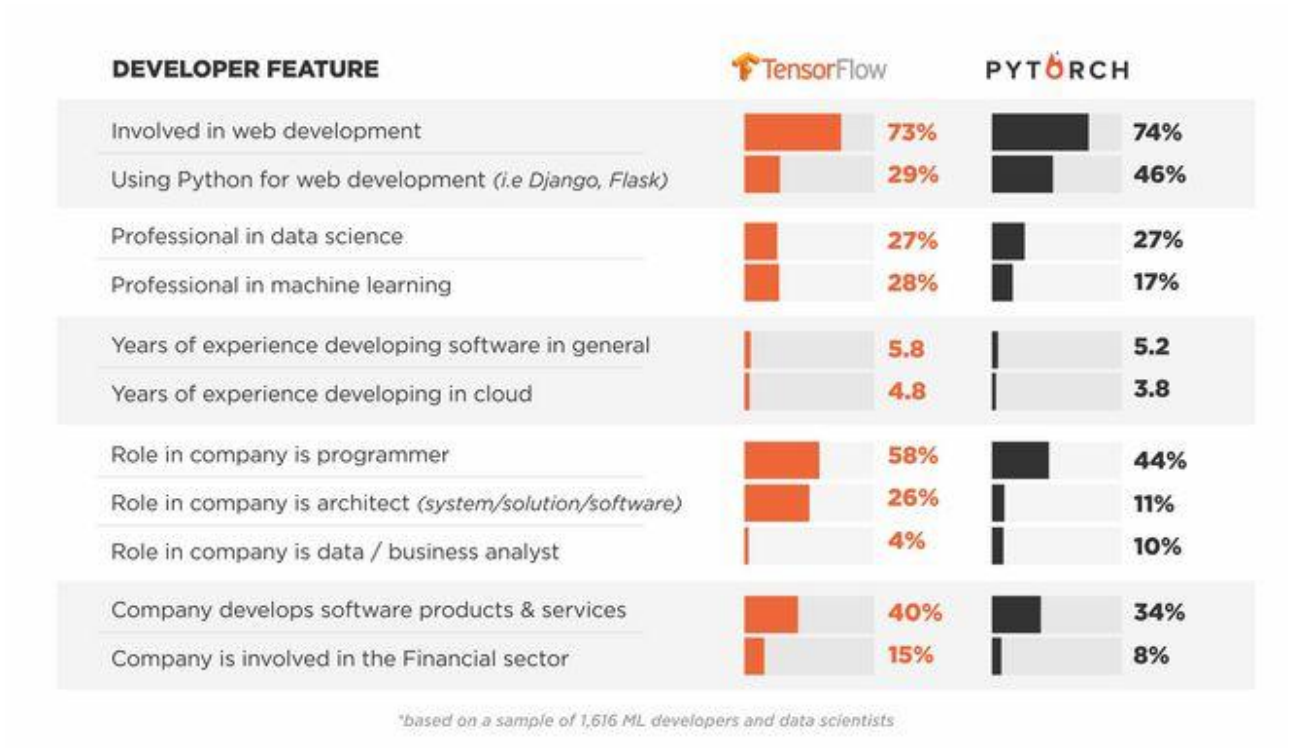


Fig. 8.1. TensorFlow vs PyTorch

## 8.5 Tensorflow vs Caffe

| TensorFlow | Caffe |
|---|---|
| 1. TensorFlow is easy to deploy as users need to install the python pip manager. | 1. In Caffe, we don’t have any straightforward method to deploy. |
| 2. TensorFlow offers high-level API’s for model building so that we can experiment easily with TensorFlow API’s. | 2. Caffe doesn’t have higher level API’s due to which it will be hard to experiment with Caffe, the configuration in a non-standard way with low-level API’s. |
| 3. In TensorFlow, configuration of jobs is straightforward for multi-node tasks by setting the tf.Device to the number of jobs need to run. | 3. In Caffe, we need to use MPI library for multi-node support and it was initially used to break apart of massive multi-node supercomputer applications. |
| 4. TensorFlow framework has less performance than Caffe in the internal benchmarking of Facebook. It has a steep learning curve and it works well on images and sequences. | 4. Caffe framework has a performance of 1.2 to 5 times more than TensorFlow in internal benchmarking of Facebook. It has a steep learning curve for beginners. It works well for deep learning on images but doesn’t work well on RNN and sequence models. |

Table. 8.1. TensorFlow vs Caffe

## 8.6 TensorFlow vs Microsoft Cognitive Toolkit

| **TensorFlow** | **Microsoft Cognitive Tool** |
|---|---|
| 1. Model building is easy in TensorFlow. | 1. It delivers good performance and scalability. |
| 2. It has some high-level API’s such as Keras. | 2. It features a lot of highly optimized components. |
| 3. It offers support for Android, iOS etc. | 3. It offers support for Apache Spark. |
| 4. It needs large datesets. | 4. It’s very efficient in terms of resource usage. |

Table. 8.2. TensorFlow vs Microsoft Cognitive Tool

## 8.7 Summary

In this chapter, we have briefly described the basics of TensorFlow and Keras API’s features and then some differences between TensorFlow vs PyTorch, TensorFlow vs Caffe and Tensorflow vs Microsoft Cognitive Tool. We have developed the model of this project with TensorFlow framework and Keras is running top of the TensorFlow.

# Chapter 9
# Tools used in this system

## 9.1 Introduction

The different tools we used in our system and their functions are described in this chapter.

The following tools have been used in the development of this project:

- Google Cloud Platform.
- GitHub.
- Firebase.
- Jupyter Notebook.
- Anaconda.

## 9.2 Google Cloud Platform

Google Cloud Platform (GCP) is a cloud service from Google. This platform contains several engines like Google App Engine, Google Compute Engine, Google Cloud Console, Google Cloud Datastore, Google Cloud Storage, Google BigQuery, Google Cloud SQL, etc. We used Google Cloud Computing for this project. As the model of this project is on the server, we have used GPU, CPU, and RAM from GCP which makes our model run faster. This service costs on hour basis. The more powerful GPUs and CPUs cost more.

## 9.3 GitHub

GitHub is a famous repository hosting service for software development. It is a web-based service for version control using Git. It has over 28 million users all over the world and it the largest host of source code in the world right now. It allows us to make an update of our system from any corner of the world. It allows merging of source codes from different developers of the same project. A developer can clone their project from anywhere in the world. GitHub makes our projects more portable.

## 9.4 Firebase

Firebase is a mobile and internet app development platform that gives developers with an excess of tools and services to assist them to develop high-quality apps, grow their user base, and earn additional profit. It is a service from Google Mobile Platform. It offers Firebase Cloud Messaging, Firebase Auth, Real-time Database, Firebase Storage, Firebase Hosting, etc.

## 9.5 Summary

In this chapter, we have described the different tools used in our system – Google Cloud Platform, GitHub, Firebase as clearly and briefly as possible. The following chapter describes the tools we have used in this project.

# Chapter 10
# Compliance with standards

## 10.1 Introduction

In this chapter the compliance of our design model to all these standards. There are many international standards which a system should meet, among which the IEEE, US and European standards are noteworthy to mention.

## 10.2 Compliance with IEEE standards

There are a few distinct guidelines put forward by IEEE Standards affiliation. The models we have used for our project are also used in most of the IEEE standard projects like Apps Requirements Specifications, Apps Design Description, and Open Firmware etc. Our system is safe, beneficial for the disabled, and uses equipment that do not pose any threat to the environment. We could also find some exact standard related to our project.

## 10.3 Compliance with US standards

In case of US standards, we could not find exact standard for our system. However, there were some standards that could be attributed to our project. Like ANSI/ISEA 138-2019 standard this is specifically to address the gaps in appropriately evaluating performance of a glove's dorsal protection and assist employers in making informed product selections. Though this does not entirely match our project, it is the closest we could find that relates to our work.

## 10.4 Compliance with European standards

GOST R 21.1101-2013 is a European standard for System of design documents for construction, main requirements for design and working documents. Our model has meets this standard.

## 10.5 Summary

In this chapter, analyzed different standards of safety, regulations and low and how the model we designed meets the relevant regulations set by the IEEE, US and European standards has been discussed.

# Chapter 11
# Design Impact

## 11.1 Introduction

The different ways in which this designed project leaves an impact, and how its manufacturability and sustainability may be discussed in this chapter.

## 11.2 Economic impact

Bangladesh has considerably low per capita arable land, which makes it hard to grow sufficient food grains for about 160 million people. Diseases prevalence and the lack of close monitoring often result in crop loss. For instance, Potato, Tomato, and Rice production decrease by 37%, 43%, and 10% respectively because of leaf infection. Early and accurate detection of these diseases can prevent a large-scale yield loss. It will create a good impact on our economy.

## 11.3 Environmental impact

As this project is software based, it is free from all sorts of materials and chemicals that may harm the wildlife or the environment. This project may aid environmental development by preventing a large-scale yield loss. So it is safe to say that this project does not have any negative environmental impact.

## 11.4 Social impact

This project can be put to a wide range of uses like poultry, fisheries, etc. This project may help our farmers to take the necessary steps of their crops before it’s too late. It may open up a new era in the agricultural sector in Bangladesh. Thus we can say that it is expected to have a remarkable social impact in our country.

## 11.5 Political impact

This project does not have any direct impact on the political aspect.

## 11.6 Ethical impact

The aim of this project is to help rural farmers of Bangladesh and make their life easier and open up new possibilities for research and development in this field. It intends to help rural farmers. Thus it can be said that the project has a positive ethical impact.

## 11.7 Health and safety impact

This project doesn't relate to any health or safety issues. As it is an app, it doesn't impact any health issues. For safety purpose, this project doesn't ask for any personal information of the users. So that we can say that, this project doesn't harm any health or safety issues.

## 11.8 Manufacturability

This project doesn't require any hardware components. All the tools, APIs are widely available on the internet. The model is developed by us. Without those tools and APIs, it would be a complex system to manufacture.

## 11.9 Sustainability

This project has fulfilled all the terms and conditions of the Google app store and we mentioned all the references. Thus it is expected to sustain on the internet.

## 11.10 Summary

The different aspects of this project's impact and its manufacturability and sustainability have been discussed in this chapter.

# Chapter 12
# Results and Discussion

## 12.1 Introduction

The machine learning models described here have different test and training accuracy. To evaluate and analyze machine learning models we need compare the result for accuracy.

## 12.2 Result of different Models

Comparison of different models:

| Name | Accuracy |
|---|---|
| MobileNet | 96.08% |
| AlexNet | 86.08% |
| Custom CNN | 90.05% |

Fig. 12.1. Comparison of different models

The three classifiers we tested with MobileNet gave the best testing accuracy with 96.08%.

## 12.3 Training accuracy of different Models

Training accuracy details:

| Name | Value | Steps | Time |
|---|---|---|---|
| MobileNet | 0.96 | 8900 | 1 hr. 30 min |
| AlexNet | 0.86 | 7000 | 2 hr. 15 min |
| Custom CNN | .90 | 1100 | 31 min |

Fig. 12.2. Training accuracy

Steps comparison of different models with the required time for training.

## 12.4 Model Size

Model size comparison:

| Name | Size (MB) |
|---|---|
| MobileNet | 40 MB |
| AlexNet | 150 MB |
| Custom CNN | 780 MB |

Fig. 12.3. Model sixe comparison

As we can see that MobileNet performs better than the other models and is comparatively of lower size.

## 12.5 Difficulties

The models are not perfect as any other machine learning models. For some odd cases the input gives wrong prediction. And for training purpose each model requires large amount of training data. For less data the model doesn’t perform well.

## 12.6 Summary

This chapter describes about the results and the difficulties we have faced during the development of this project. We showed some comparisons because we also followed those comparisons to select the best technology among them.

# Chapter 13
# Android App

## 13.1 Introduction

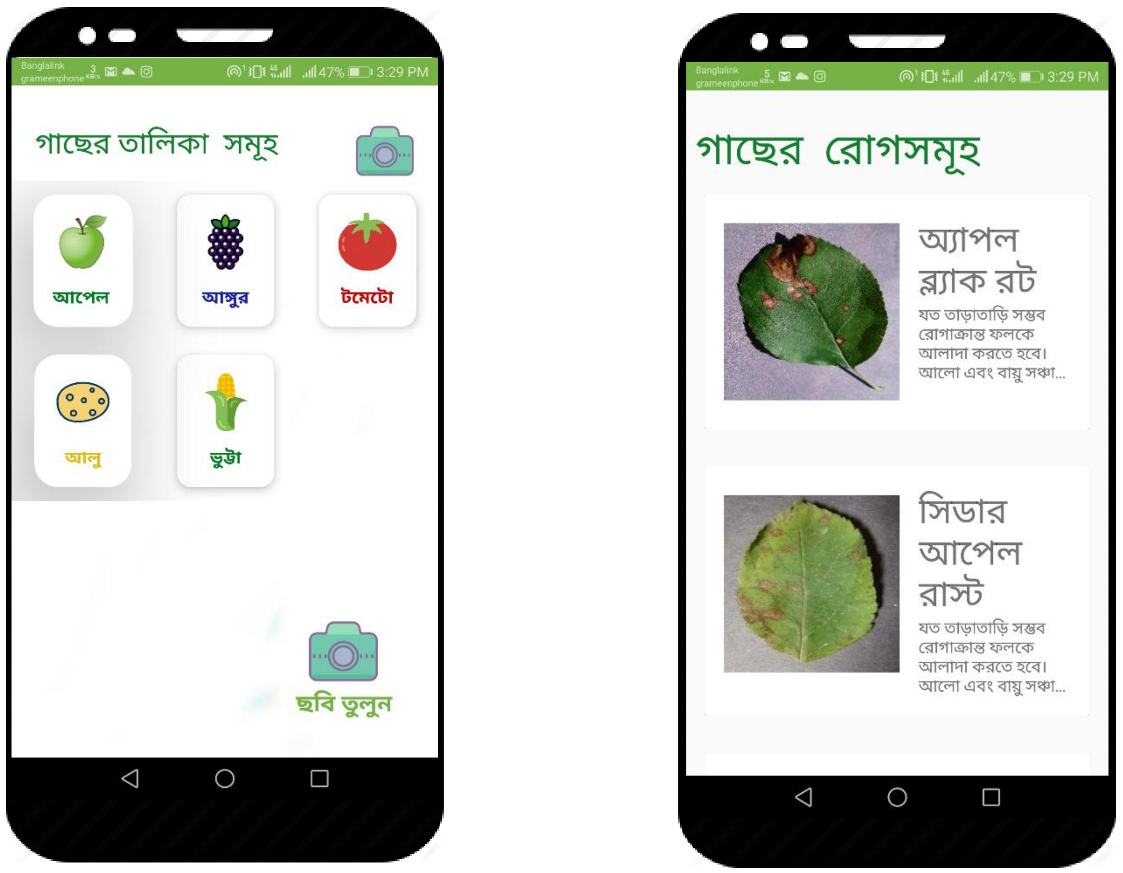
We have developed an Android app for this project. This is a lightweight app as all the image processing tasks have been done on the server end. In this chapter, we discuss the user interface (UI), power and mobile data consumption, user manual and tutorial of this app.

## 13.2 User Interface

We have made this app as simple as possible so that rural farmers can operate this app easily.

### 13.2.1 Graphical User Interface (GUI)

We have used Bangla font for our Android app, because it will be easy for the rural farmers to understand what he/she should do for their desired function. We also used related icons of the plants. We have used voice navigation to make them understand better. The background color is soothing so that it can be seen under the direct sunlight.



যত তাড়াতাড়ি সম্ভব রোগাক্রান্ত ফলকে আলাদা করতে হবে। আলো এবং বায়ু সঞ্চা...

যত তাড়াতাড়ি সম্ভব রোগাক্রান্ত ফলকে আলাদা করতে হবে। আলো এবং বায়ু সঞ্চা...

Fig. 13.1. User interface of the App

### 13.2.2 Bangla Voice Assistance

As our rural farmers do not know much about how to operate smartphones, we tried to help them by using Bangla voice assistance in our application. It will assist farmers in every step when he/she wants to take photos of the affected leaves. We expect that it may make their tasks easier.

## 13.3 Power and Mobile Data Consumption

The app is optimal against power consumption and data usages. The App only takes camera and storage permission from the users. It only takes 14-15 MB of storage. Data usages of this app also low because we resize the pictures first and then send them to our server. About 150 pictures can be processed by using only 2MB of data package. The app is also friendly in terms of battery drainage and data usages. The app requires low energy because most of the computationally heavy functions are done on our server. The app is mainly used for taking the inputs and showing the output.

## 13.4 User Manual and Tutorial

There is a written manual in our android app which will help our rural farmers to make them understand about the app operational steps. Also, there is a video tutorial that makes the task easier by showing how to use this app.

Fig. 13.2. Video Tutorial

## 13.5 Summary

In this chapter, we have discussed about the Android app of this project. Details and usage of the android interface is also discussed.

# Chapter 14
# Conclusion

In this project, we have focused on the rural farmers of Bangladesh. Farmers will be able to detect diseases of potatoes, tomatoes, apples, etc. by using this app. As these crops are most affected due to leaf diseases in Bangladesh, Farmers will be able to take necessary steps before it is too late. We would save million tons of crops every year by taking precautions at the early stage. This is an ideal neural network model, where it allows to add more crop diseases as an extension of the model. For this, we need the datasets of specific plant diseases. This model tries to predict the disease accurately. But as it is a Machine Learning model, the prediction will not be accurate all the time. The architecture of the project is also adaptable for other kinds of crops and poultry diseases. We can, in the future, expand the system to detect fisheries and poultry related diseases. If we are able to incorporate and introduce this app widely to the rural farmers, it can help them in great amount and may increase our crop production.

# Bibliography

[1] Ferentinos, Konstantinos P. “Deep Learning Models for Plant Disease Detection and Diagnosis.” Computers and Electronics in Agriculture, vol. 145, 2018,pp.11–318.,doi:10.1016/j.compag.2018.01.009.

[2] Ferentinos, Konstantinos P. “Deep Learning Models for Plant Disease Detection and Diagnosis.” Computers and Electronics in Agriculture,vol.145,2018,pp.311–318., doi:10.1016/j.compag.2018.01.009.

[3] Tucker C. Red and photographic infrared linear combinations for monitoring vegetation. Remote Sens Environ. 2016.10.16

[4] González, Andrés, and Andrés González. "Basic Machine Learning Concepts -". Cleverdata.Io, 2019, https://cleverdata.io/basic-machine-learning-concepts/.

[5] J. X. Du, X. F. Wang, and G. J. Zhang, “Leaf shape based plant species recognition,” Applied Mathematics and Computation, vol. 185, no. 2, pp. 883-893, 2007. http://dx.doi.org/10.1016/j.amc.2006.07.072

[6] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, 2005, pp. 886-893. http://dx.doi.org/10.1109/CVPR.2005.177

[7] D. G. Lowe, “Distinctive image features from scaleinvariant keypoints,” International Journal of Computer Vision, vol. 60, no. 2, pp. 91-110, 2004. http://dx.doi.org/ 10.1023/B:VISI.0000029664.99615.94

[8] Y. Freund and R. E. Schapire, “Experiments with a new boosting algorithm,” Machine Learning: Proceedings of the 13th International Conference, vol. 96, pp. 148-156, 1996.

[9] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, no. 3, pp. 273-297, 1995. http://dx.doi.org/10.1007/BF00994018

[10] E. Chum, “Drones and artificial intelligence– AI,” Available

http://www.focus.kr/view.php?key= 2016041200101856440

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradientbased learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 2002. http://dx.doi.org/10.1109/5.726791

[12] Huang, J., Rathod, V., Sun, C.; Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z.,Song, Y., Guadarrama, S., et al. Speed/accuracy trade-offs for modern convolutional object detectors. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 22–25 July 2017.

[13] Alvaro Fuentes , Sook Yoon , Sang Cheol Kim and Dong Sun Park A Robust Deep Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition, Sensors 2017, 17, 2022; doi:10.3390/s17092022.

[14] Ren, S., He, K., Girshick, R., Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans. Pattern Anal. Mach. Intell. 2016, 39, 1137–1149.

[15] Liu, W., Anguelov, D, Erhan, D., Szegedy, C., Reed, S.; Fu, C.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the European Conference on Computer Vision ECCV, Amsterdam, The Netherlands, 8–16 October 2016; pp. 21–37.

[16] Dai, J., Li, Y., He, K., Sun, J. R-FCN: Object Detection via Regionbased Fully Convolutional Networks. arXiv 2016, arXiv:1605.06409v2.

[17] Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. Int. Comput. Vis. 2010, 88, 303–338.

[18] R. Girshick. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, pages 1440–1448, 2015.

[19] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. arXiv preprint arXiv:1512.04412, 2015.

[20] R. Girshick. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, pages 1440–1448, 201