LEAF DISEASE DETECTION USING CNN

A PROJECT REPORT

Submitted by

CHETAN G (180501029)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESWARA COLLEGE OF ENGINEERING
(An Autonomous Institution; Affiliated to Anna University, Chennai-600 025)

ANNA UNIVERSITY :: CHENNAI 600 025

JUNE 2022

LEAF DISEASE DETECTION USING CNN

A PROJECT REPORT

Submitted by

CHETAN G (180501029)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESWARA COLLEGE OF ENGINEERING
(An Autonomous Institution; Affiliated to Anna University, Chennai-600 025)

ANNA UNIVERSITY :: CHENNAI 600 025

JUNE 2022

SRI VENKATESWARA COLLEGE OF ENGINEERING
(An Autonomous Institution; Affiliated to Anna University, Chennai-600 025)

ANNA UNIVERSITY :: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"LEAF DISEASE DETECTION USING CNN"** is the bonafide work of **"CHETAN G (180501029)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.R. ANITHA	**SURESHKUMAR M**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
	ASSISTANT PROFESSOR
COMPUTER SCIENCE & ENGG	**COMPUTER SCIENCE & ENGG**

Submitted for the project viva-voce examination held on ………………..

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ABSTRACT

Agriculture plays an important role in the economy of our country. Plants are affected by various diseases which are caused by changes in the environment and use of chemical pesticides. Diseases in plants affect the growth of the plant which can lead to huge losses in the production. Identification of the plant disease by the naked eye requires a huge amount of work, knowledge and experience so as to detect the actual disease. The diseases can tend to look similar which further leads to confusion. Due to these challenges the plant disease can sometimes be predicted incorrectly. Therefore, early identification of the disease is very significant in order to avoid such issues. This is a problem that has a social impact and can help improvise on sustainability and provide feasible and easy solutions. Advanced developments in deep learning have helped improve the accuracy and performance of detection systems. This project aims to provide an effective means of timely detection and classification of the disease of a leaf.

ACKNOWLEDGEMENT

We thank our Principal **Dr. S. Ganesh Vaidyanathan,** Sri Venkateswara College of Engineering for being the source of inspiration throughout our study in this college.

We express our sincere thanks to **Dr. R. Anitha,** Head of the Department, Computer Science and Engineering for her encouragement accorded to carry this project.

With profound respect, we express our deep sense of gratitude and sincere thanks to our internal guide**, Mr. M Sureshkumar, Assistant Professor** for their valuable guidance and suggestions throughout this project.

We are also thankful to our project coordinators **Dr. R.Jayabhaduri, Associate Professor, Dr. N.M.Balamurugan, Associate Professor and Ms.V.Rajalakshmi, Assistant Professor** for their continual support and assistance.

We thank our family and friends for their support and encouragement throughout the course of our graduate studies.

CHETAN G

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**iii**
	LIST OF FIGURES	**viii**
1	**INTRODUCTION**	**1**
	1.1 BACKGROUND	1
	1.2 EXISTING SYSTEM	2
	1.3 ISSUES AND CHALLENGES	3
2	**RELATED WORK**	**5**
3	**PROPOSED SYSTEM**	**7**
	3.1 PROPOSED SYSTEM ARCHITECTURE	7
	3.2 MODULES	9
4	**DETECTION AND CLASSIFICATION USING CNN**	**10**
	4.1 INTRODUCTION TO DEEP LEARNING	10
	4.2 DEEP LEARNING AND NEURAL NETWORK	11
	4.2.1 CONVOLUTION	13
	4.2.2 RELU	14

4.2.3 POOLING 14

4.2.4 FULLY CONNECTED LAYER 15

4.3 INTRODUCTION TO TENSORFLOW 16

4.3.1 KEY FEATURES OF TENSOR FLOW 17

5 SYSTEM REQUIREMENT SPECIFICATION 18

5.1 HARDWARE REQUIREMENTS 18

5.1.1 LAPTOP/PC 18

5.1.2 ANDROID MOBILE PHONE 18

5.2 SOFTWARE REQUIREMENTS 19

5.2.1 GOOGLE COLAB 19

5.2.1.1 LIBRARIES USED 19

5.2.2 ANDROID STUDIO 21

6 IMPLEMENTATION AND RESULT 23

6.1 IMPLEMENTATION MODEL 23

6.1.1 DATA ACQUISITION 23

6.1.2 DATA PREPROCESSING 24

6.1.3 BUILD AND TRAIN CNN MODEL 25

6.1.3.1 CNN MODEL LAYERS 26

6.1.4 TESTING THE MODEL 32

6.1.5 DEPLOYING THE MODEL TO ANDROID APPLICATION 33

7 CONCLUSION AND FUTURE WORK 36

7.1 CONCLUSION 36

7.2 FUTURE WORK 37

REFERENCES 38

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1.1	Proposed system architecture	7
4.2.1	Example of a CNN model architecture	12
4.2.1.1	Convolution of 5x5 matrix using 3x3 filter	13
4.2.3.1	Maxpooling of 4x4 matrix	15
4.2.4.1	Fully Connected Layer	16
5.1.2.1	Android logo	19
5.2.1.1.1	TensorFlow Lite logo	20
6.1.1.1	Health and Diseased Corn leaf images	23
6.1.2.1	Data Preprocessing Steps	24
6.1.3.1.1	CNN model layers	26
6.1.3.1.2	Convolution Layer 1	27
6.1.3.1.3	Maxpooling Layer 1	28
6.1.3.1.4	Convolution Layer 2	28
6.1.3.1.5	Maxpooling Layer 1	29

6.1.3.1.6	Convolution Layer 3	29
6.1.3.1.7	Maxpooling Layer 1	30
6.1.3.1.8	Training and Validation Accuracy and Loss graph	31
6.1.4.1	Result of Test Image	32
6.1.4.2	Conversion to TFLite	32
6.1.5.1	Splash Screen of Cropfit App	33
6.1.5.2	Homepage of Cropfit App	34
6.1.5.3	Result on Cropfit App	34
6.1.5.4	Remedy and tools on Cropfit App	35

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Agriculture plays the role of the foundation of the Indian economy. The focus on commercialization as the growth of agriculture has created a harmful effect on our environment. The development in chemical industries and the need for quick remedies has led to the increase in use of chemical pesticides which in turn results in chemical buildup in the soil and water. In order to get a temporary solution we have been repeatedly using such chemicals but these tend to have a long term effect on our environment. The detection of plant diseases based on the visual appearance of the plant leaves through naked eye observation involves an escalating complexity. Plants are vulnerable to several problems and attacks caused by diseases. Numerous reasons can be used to explain the diseases of the plants, the disease may be caused because of changes in the environmental conditions such as the temperature or humidity. The diseased plants will have different physical features on the leaves, there may be a change in color or there may be patterns on the leaves. As these changes may be very minute and maybe difficult to observe with the naked eye.

This difficulty and also the large field of variety of crops and their diseases has resulted in the failure of even experts in the field and pathologists. This can cost a farmer on a huge scale in case of a wrong prediction or solution. An automated system designed to help predict plant diseases based on the plant leaves image could be of great benefit to farmers. It can prove to be a useful technique for farmers and can help with an earlier detection to treat the plant as soon as to avoid incurring any loss.

1.2 EXISTING SYSTEM

The existing system for plant disease detection is the naked eye observation by pathologists or experts in the field who identify and predict the diseases of the plant. This requires lots of knowledge and experience and can also be a failure if predicted wrong. It also involves periodic monitoring of plants which can be time consuming and also expensive in case of a large area. In rural areas farmers and those with less exposure do not have the knowledge required or do not even know that there are experts for such problems. The involvement of experts also does not prove to be a cost-efficient method. Moreover, it is a more tedious task and also less accurate and can be time consuming for a large area. In consideration of this the use of technology and development can prove to be a more efficient method for the detection of plant diseases. It is also a more cost-efficient method. This system requires less effort, less time and can prove to be more accurate. Though there have been a few systems already proposed that involve deep learning methods, the systems still lack accuracy and more optimization is needed. Moreover, these systems haven’t been made into a remotely usable version.

1.3 ISSUES AND CHALLENGES

The various issues and challenges involved are

- Need of knowledge and experience of experts in the field, but this means a third-party involvement and dependency.
- Farmers seek advice and help at a cost and with their inadequate resources and financial struggle it makes it harder.
- Reduction in agricultural production potential as a result of loss due to diseases and the fight for land in fertile areas.
- Untimely predictions lead to loss and suffrage for the agriculture field.
- Increased demands to satisfy the requirements of the exponentially increasing population.

1.4 OVERVIEW

Several techniques have been used for the purpose of the detection and classification of crop diseases. Most of these techniques have involved machine learning techniques of image processing. Deep Learning techniques in particular those involving Convolutional Neural Networks (CNNs) have proved to be a success in image processing, recognition and classification. The convolution layers of a CNN can be viewed as matching filters that have been directly derived from the data. A model that undergoes CNN training is the result, it has a set of weights and biases based on the process involved. The best feature of CNN being its robustness towards unseen data helping with the generalization and making it a more efficient system. This model deployed to a user accessible version such as a mobile application can help with the early detection of plant leaf diseases to take measures to prevent the spread and also provide a solution to tackle the disease.

CHAPTER 2

RELATED WORK

1. In paper[1], a deep learning technique - CNN based method for plant disease detection has been proposed... A study and analysis is done on the sample data based on the area of the infected region. It is done by image processing technique. Different performance metrics are derived for the same. Test accuracy is obtained as 88.80%.

2. In paper[2], the proposed system is based on the segmentation method through simple linear iterative clustering to detect disease in plant leaves. It also shows visual attributes such as color, gradient, texture, and shape to describe the features of leaves. The system tends to have high complexity and poor accuracy.

3. In paper[3], the system is focused on identifying jute leaf disease using CNN (Convolutional Neural Network). CNN is one of the deep feed-forward artificial neural networks which have the features like local connection, augmentation and pooling operation that make it possible to effectively shorten the network complexity. The image processing algorithms and techniques which have been playing an important role in agricultural industries in recent years, especially in the area of leaf disease detection, are used.

4. In paper[4], the system focuses on development of an automated

system to detect plant diseases. Image processing toolbox is used for measuring the affected area of disease and to determine the difference in the color of the disease affected area. The algorithm is used to classify the leaves and the classified outcomes are separated using an Arduino based conveyor belt system. The involvement of hardware devices makes the system more expensive and space consuming.

CHAPTER 3

PROPOSED SYSTEM

3.1. PROPOSED SYSTEM ARCHITECTURE

The below figure depicts the system architecture for detection and classification of diseased corn leaves using convolutional neural networks.

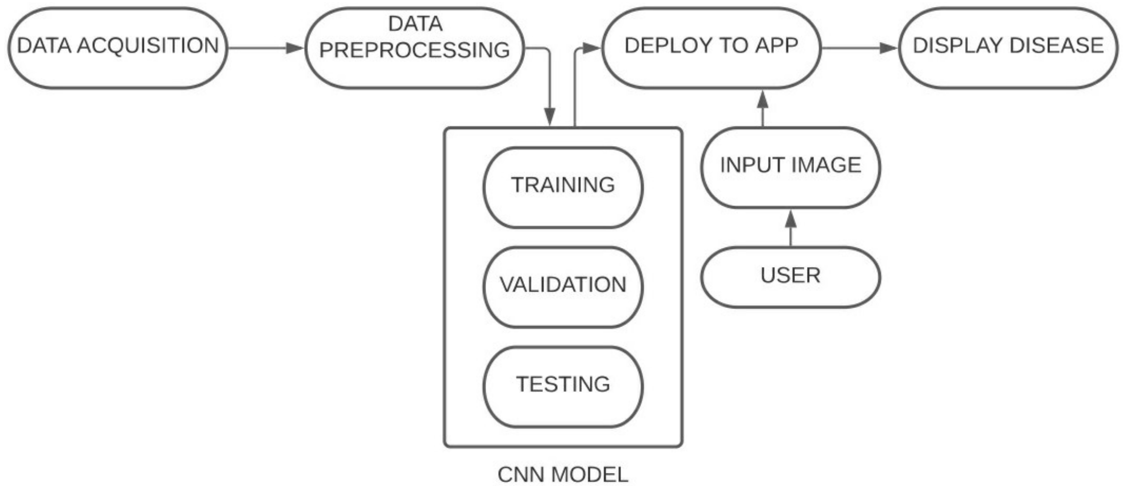


Figure 3.1.1 Proposed system architecture

Recent research and development in the field of technology has led to machine learning and deep learning techniques being employed for the detection and classification of images. In our proposed system, we are using a detector model based on Convolution Neural Networks for the detection and classification of diseases that affect corn plants.

The distinctive part of our approach is rather than only dealing with the detection of the disease we also give a solution in the form of a remedy or suggestion for those concerned diseases. A mobile application on which a trained convolutional neural networks model is deployed can be used to detect and classify the disease of a corn leaf by uploading the image from the local backup or by capturing a real time image. Initially data for the system is acquired from the various databases available online. For this project, we have taken the Corn Disease Dataset from the PlantVillage dataset in Kaggle. This data is preprocessed to clean the data and modify it to the required format. The preprocessed data is fed to a CNN model built with various layers for the purpose of training.

Convolutional Neural Networks consists of 2 main steps: (i) feature extraction and (ii) classification. The feature extraction step involves the input layer, the convolutional layers with a stride and padding value, rectified linear unit (ReLU), and the pooling layers and batch normalization layer. The classification step involves the fully connected layer, the softmax layer(in the case of more than two categorizations) and the output layer. By setting appropriate parameter values a generalized well fit model is built. The trained model is then used to test images that are unseen. Further, the trained model is then deployed to an android application. This application takes input from the user in the form of an image of the corn leaf and displays the disease of the leaf along with a remedy. The attack of pests by use of proper fungicides and remedies can be reduced. Through this system we expand the projects of the earlier mentioned authors such that the remedy of the disease is also shown in the app.

3.2. MODULES

The entire proposed architecture system is split into five modules:

- Data Acquisition
- Data Preprocessing
- Build & Train CNN Model
- Test Model
- Deploy Model to App

CHAPTER 4

DETECTION AND CLASSIFICATION USING CNN

4.1 INTRODUCTION TO DEEP LEARNING

Deep learning is a part of the broader family of machine learning methods which is based on learning data representations, as unlike task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. Deep learning models are somehow related to information processing and communication patterns similar to how our human biological nervous system works, such as neural coding. Biological neural system defines a relationship between various stimuli and associated neuronal responses in the brain. Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields such as computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics and drug design, where they have produced results which exceeds human experts.

4.2 DEEP LEARNING AND NEURAL NETWORKS

In machine learning, a convolutional neural network (CNN, or

ConvNet) is a class of deep, feed-forward artificial neural networks that have successfully been applied to analyzing visual imagery. CNN is one of the most efficient deep learning neural network algorithms in the field of computer vision. CNN stands out in the field of image recognition, image classification, and image segmentation and so on. The main goal of CNN is to process types of data such as 1D for signal or time series data, 2D for images or audio signals and 3D for video or depth images. As it has been achieving state-of-the-art for image classification, we have used traditional CNN to classify our corn leaf image dataset. CNN's use a number of multilayer perceptrons designed to achieve minimal pre-processing. They are otherwise known as shift invariant or space invariant artificial neural networks, based on their shared-weights architecture and translation invariance characteristics. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use comparatively less pre-processing compared to other image classification algorithms. This means that the network learns the filters whereas traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

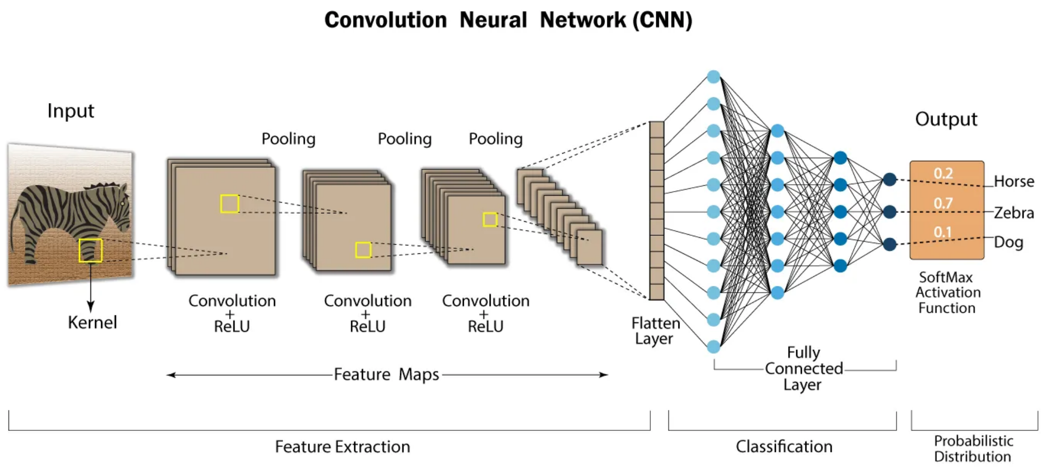


Figure 4.2.1 Example of a CNN model architecture

They have applications in image and video recognition, recommender systems and natural language processing.There are four main operations in the ConvNets: Convolution Non-Linearity (ReLU) Pooling or Sub Sampling Classification (Fully Connected Layer) These operations are the foundation of every Convolutional Neural Network, so understanding how this work is an important step to developing a sound understanding of ConvNets.

4.2.1 CONVOLUTION

ConvNets derive their name from the “convolution” operator. The primary purpose of Convolution in the case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. Convolution is similar to cross-correlation. For discrete real valued signals, they differ only in a time reversal in one of the signals.

For continuous signals, the cross-correlation operator is the adjoint operator of the convolution operator. Fig 5.2.1.1: Example matrices We slide the orange matrix over our original image (green) by 1 pixel (also called 'stride') and for every position, we compute element wise multiplication (between the two matrices) and add the multiplication outputs to get the final integer which forms a single element of the output matrix. Note that the 3×3 matrix "sees" only a part of the input image in each stride. In CNN 35 terminology, the 3×3 matrix is called a 'filter' or 'kernel' or 'feature detector' and the matrix formed by sliding the filter over the image and computing the dot product is called the 'Convolved Feature' or 'Activation Map' or the 'Feature Map'. It is important to note that filters act as feature detectors from the original input image.

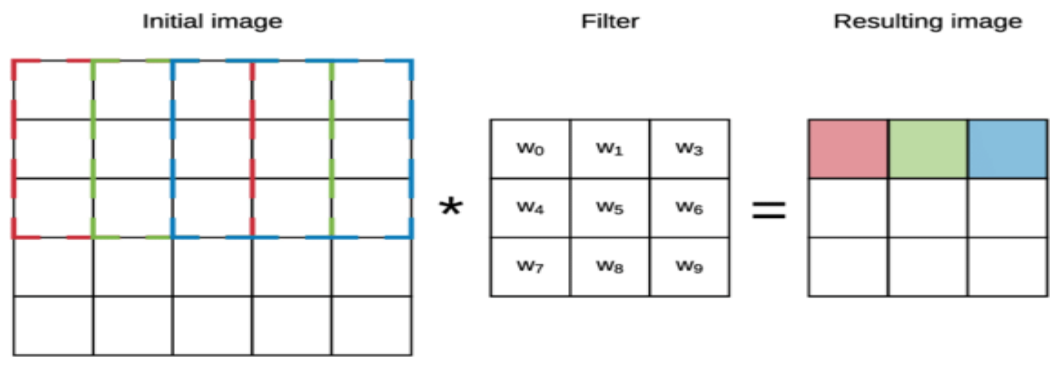


Figure 4.2.1.1 Convolution of 5x5 matrix using 3x3 filter

4.2.2 RELU

An additional operation called ReLU has been used after every Convolution operation. ReLU stands for Rectified Linear Unit and is a nonlinear operation. ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a nonlinear function like ReLU). Other nonlinear functions such as tanh or sigmoid can also be used instead of ReLU, but ReLU has been found to perform better in most situations.

4.2.3 POOLING

Spatial Pooling (also called subsampling or down sampling) reduces the dimensionality of each feature map but retains the most valuable information. Spatial Pooling can be of several types: Max, Average, Sum etc. In the case of Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also take the average (Average Pooling) or the sum of all elements in that window. In practice, Max 36 Pooling has been shown to work better. Fig 5.2.3.1: Max Pooling operation We slide our 2 x 2 window by 2 cells (also called 'stride') and take the maximum value in each region.

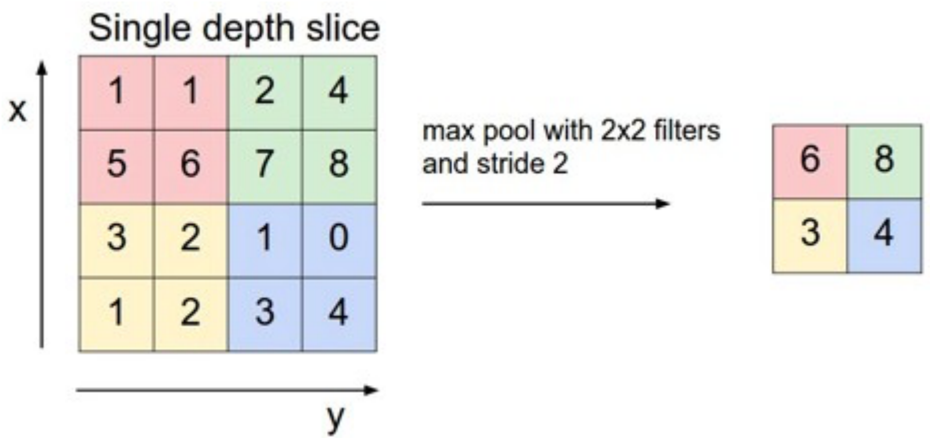


Figure 4.2.3.1 Maxpooling of 4x4 matrix

4.2.4 FULLY CONNECTED LAYER

The Fully Connected layer is a traditional Multi-Layer Perceptron that uses a SoftMax activation function in the output layer. The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer. The output from the convolutional and pooling layers represents high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset. Apart from classification, adding a fully-connected layer is also a way of learning non-linear combinations of these features. Most of the features from convolutional and pooling layers may be good for the classification task, but combinations of those features might be even better.

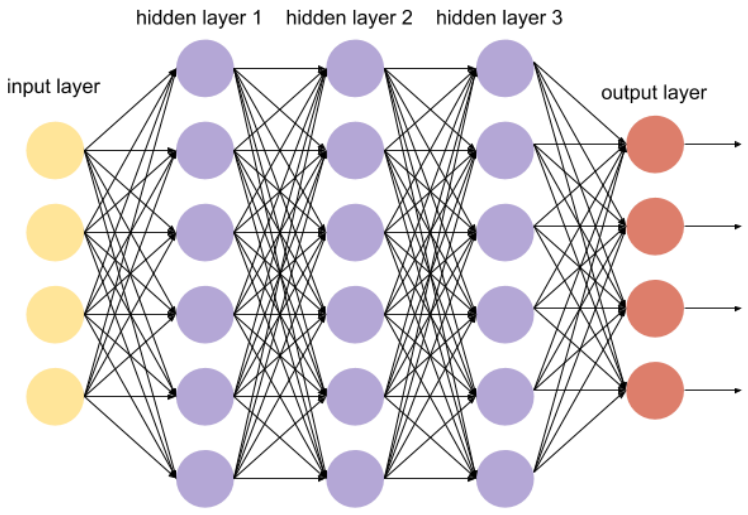


Figure 4.2.4.1 Fully Connected Layer

4.3 INTRODUCTION TO TENSORFLOW

TensorFlow is an open-source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows users to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

4.3.1 KEY FEATURES OF TENSORFLOW

Efficiently works with mathematical expressions involving multidimensional arrays. Good support of deep neural networks and machine learning concepts. GPU/CPU computing where the same code can be executed on both architectures. High scalability of computation across machines and huge data sets. Together, these features make TensorFlow the perfect framework for machine intelligence at a production scale.

CHAPTER 5

SYSTEM REQUIREMENT SPECIFICATION

5.1 HARDWARE REQUIREMENTS

5.1.1 Laptop/PC

A laptop/personal computer is required with an internet connection to run the google colab platform through a browser.

Core: Intel Core i5
CPU: 2.30 GHz Quad-Core
RAM: 8.00 GB

5.1.2 Android Mobile Phone

An android mobile phone is required to run the android application on which the model is deployed.

Android Version: Android 6.0 and above

Figure 5.1.2.1. Android logo

5.2 SOFTWARE REQUIREMENTS

5.2.1. GOOGLE COLAB

Colaboratory (Colab) is a platform product provided by Google Research. This platform allows anybody to write and execute python code through any compatible browser with an internet connection. It is being used very well for machine learning and data analytics and coding. In a technical way, Colab is a hosted Jupyter notebook service that needs no setup to use, at the same time providing free access to computing resources including GPUs.

5.2.1.1 LIBRARIES USED

NumPy

NumPy provides a high-performance multidimensional array and basic tools to compute with and manipulate these arrays.

Matplotlib

Matplotlib is used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing features to control line styles, font properties, formatting axes etc. It supports various graphs and plots such as histograms, bar charts, power spectra etc. 'matplotlib.pyplot' is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting areaetc.

Keras

Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3 Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, R, Theano, and PlaidML.

TensorFlow Lite

TensorFlow Lite is TensorFlow's lightweight solution for mobile and embedded devices. It lets you run machine-learned models on mobile devices with low latency, so you can make use of them to do classification, regression without the use of a server.

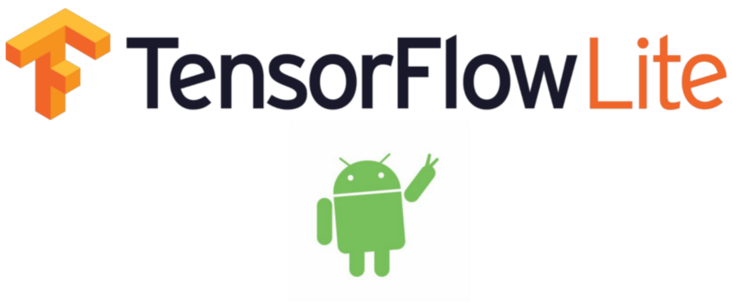


Figure 5.2.1.1.1. TensorFlow Lite logo

It’s presently supported on Android and iOS, as well as having a Java Wrapper for Android Developers. TensorFlow Lite consists of a runtime on which you can run pre-existing models, and a suite of tools that you can use to prepare your models for use on mobile and embeddeddevices.

5.2.2 ANDROID STUDIO

Android Studio is the official Integrated Development Environment (IDE) for Android app development. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:(i) A flexible Gradle-based build system. (ii)A fast and feature-rich emulator. (iii)A unified environment where you can develop for all Android devices. It is simple to apply changes to push code and resource changes to your running app without having to uninstall and restart the app. There are a resourceful of code templates to be used. The GitHub integration allows you to import sample code from repositories which enable us to use it for common features.

There is also built-in support for the Tensorflow lite platform.

Kotlin

Kotlin is a cross-platform, statically typed, general-purpose programming language that supports both object-oriented and functional programming with type inference. Kotlin’s interoperability with Java is core to its growth. Kotlin's interoperability with Java means that you don’t have to adopt Kotlin all at once. You can have projects with both Kotlin and Java code. Kotlin is fully supported in Android Studio. All new releases of Android Studio ship with support for creating new projects with Kotlin files, converting Java language code to Kotlin, debugging Kotlin code, and more.

CHAPTER 6

IMPLEMENTATION & RESULT

6.1 IMPLEMENTATION MODULES

6.1.1 DATA ACQUISITION

The collection and preparing of data to be taken as input for a CNN model is a significant step to build an efficient model that generalizes. For the detection of corn leaf diseases, all the data imagery is taken from Kaggle Corn Disease Dataset. The dataset used consists of 4188 images of Corn plant leaves.

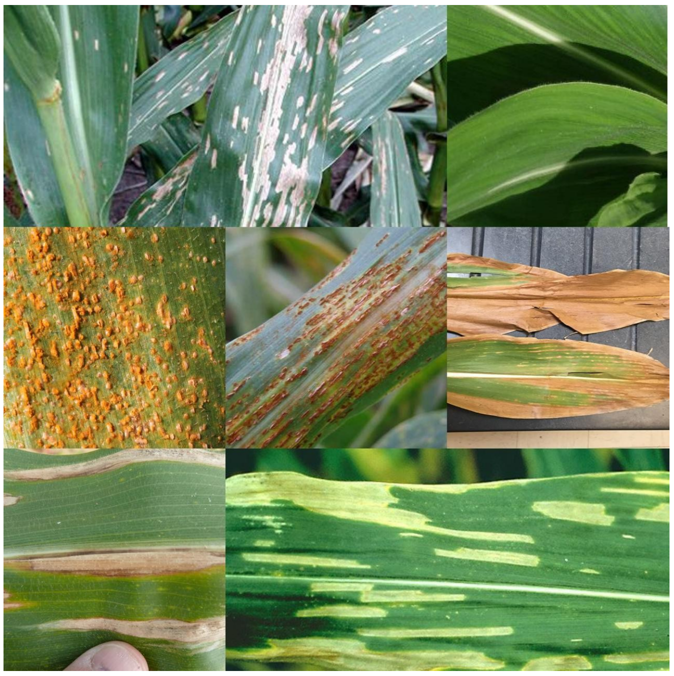
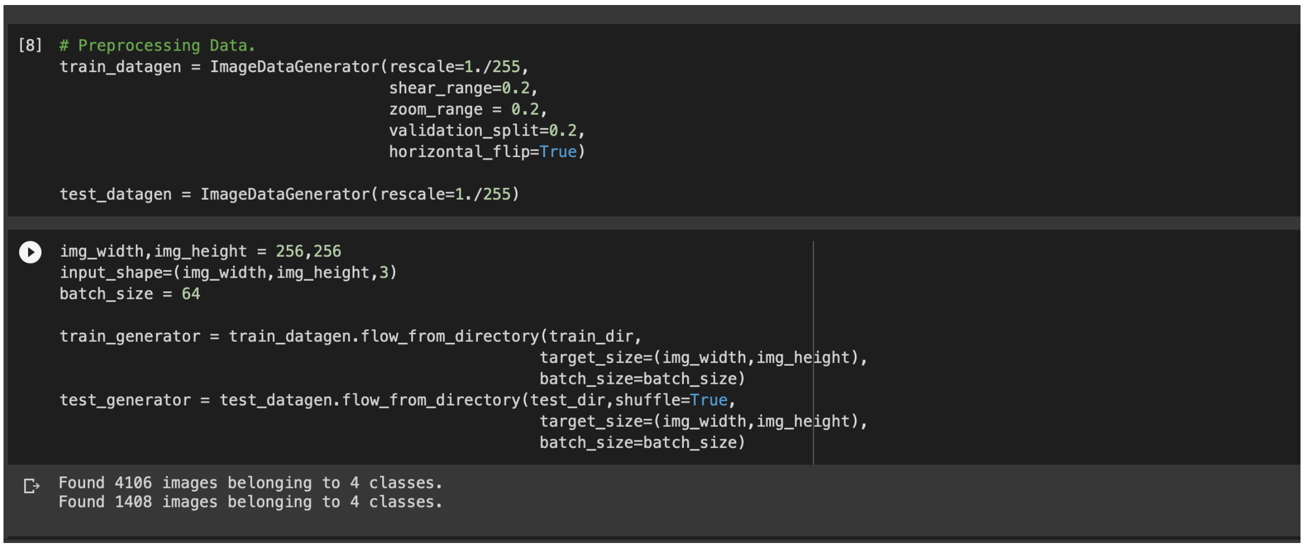


Figure 6.1.1.1 Healthy and Diseased Corn leaf images

The images consist of both fresh and diseased leaves. Each image comes under a class that is used as the label to identify the disease. There are four classes: Blight, Common Rust, Gray Leaf Spot and Healthy. This data is used to train, validate and test the model. The data is checked to make sure it is clean and there is no unnecessary or irrelevant data. In respect of this, it helps build a more efficient and precise model.

6.1.2 DATA PREPROCESSING

Preprocessing refers to all the transformations done on the raw data before it is fed to the CNN model. If this step is not done it can lead to very bad classification. Image data preprocessing is also used to scale up the training dataset in order to improve the performance and ability of the CNN model to generalize. From the acquired dataset, 80% of the images is used for training and 20% is used for testing.



```
[8] # Preprocessing Data.
    train_datagen = ImageDataGenerator(rescale=1./255,
                                       shear_range=0.2,
                                       zoom_range = 0.2,
                                       validation_split=0.2,
                                       horizontal_flip=True)

    test_datagen = ImageDataGenerator(rescale=1./255)
```

```
    img_width,img_height = 256,256
    input_shape=(img_width,img_height,3)
    batch_size = 64

    train_generator = train_datagen.flow_from_directory(train_dir,
                                                        target_size=(img_width,img_height),
                                                        batch_size=batch_size)
    test_generator = test_datagen.flow_from_directory(test_dir,shuffle=True,
                                                        target_size=(img_width,img_height),
                                                        batch_size=batch_size)
```

```
Found 4106 images belonging to 4 classes.
Found 1408 images belonging to 4 classes.
```

Figure 6.1.2.1 Data Preprocessing Steps

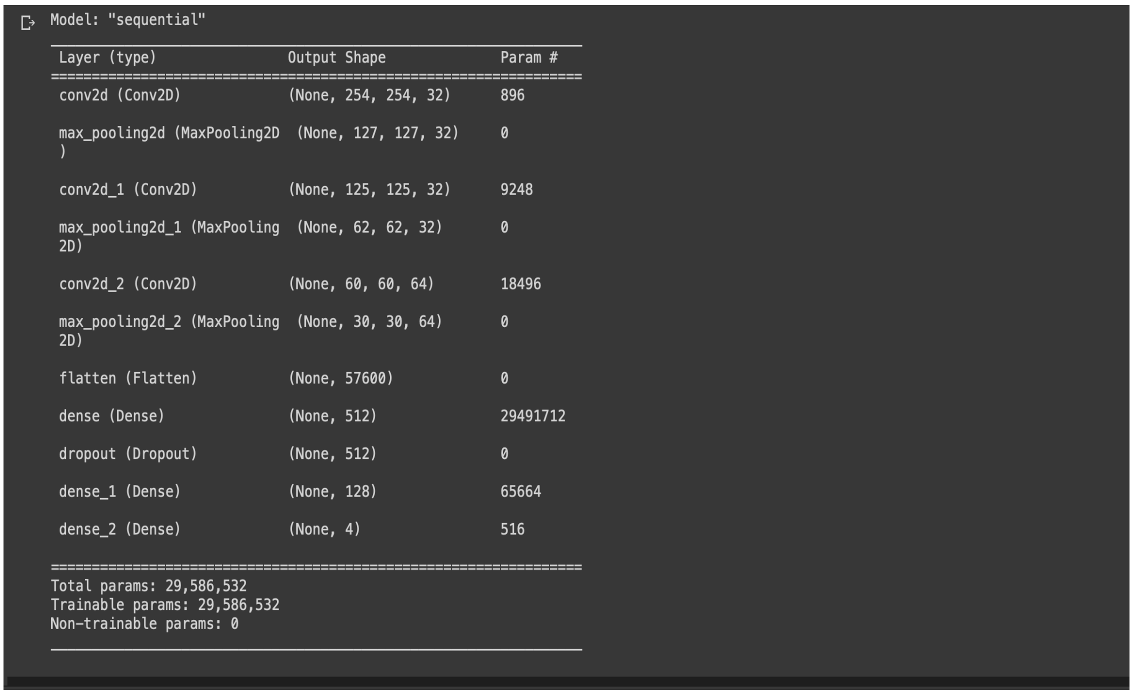
The preprocessing involves steps such as reshaping, flipping and scaling. This can be done using various image preprocessing libraries. For this purpose we have used the ImageDataGenerator from the Keras library. It yields batches of 256 × 256 RGB images of shape (64, 256, 256, 3). The preprocessing done on the training set is carried out in parallel on the testing set.

6.1.3 BUILD & TRAIN CNN MODEL

After preprocessing the images are used to train the model, a convolutional neural networks model is built with various layers. The training data in specific is used to build and train the model. We use the Keras API and import the convolution, max pooling, flatten, dense and dropout packages to build the layers of the model. The input layer takes an input image of size 256 x 256 which is coloured in batches of 64. The output layer gives one of the four classes as an output.

6.1.3.1 CNN MODEL LAYERS

The Convolutional Neural Network model is built using convolutional layers with ReLU activation, maxpooling layers and then a flatten layer which serves as a bridge to the inner layers. The inner layers consist of dense and dropout layers and then the final layer which serves as the output layer.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 32)      9248

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 64)       0
 2D)

 flatten (Flatten)           (None, 57600)             0

 dense (Dense)               (None, 512)               29491712

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 128)               65664

 dense_2 (Dense)             (None, 4)                 516

=================================================================
Total params: 29,586,532
Trainable params: 29,586,532
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.1.3.1.1 CNN model layers

CONVOLUTION

The aim of this step is to reduce the size of the image and make processing faster and easier. Some of the features of the image are lost in this step. However, the main features of the image that are important in image detection are retained. These features are the ones that are unique to identifying that specific object.

The convolution layers are added with ReLU activation. The convolutional layers take images in batches of 64 and the kernel size/ filter is taken as 3 x 3 window.

MAX-POOLING

Pooling enables the CNN to detect features in various images irrespective of the difference in lighting in the pictures and different angles of the images. Max pooling works to preserve the main features while also reducing the size of the image. This helps reduce overfitting, which would occur if the CNN is given too much information, especially if that information is not relevant in classifying the image. The max pooling layers use a window of size 2 x 2.

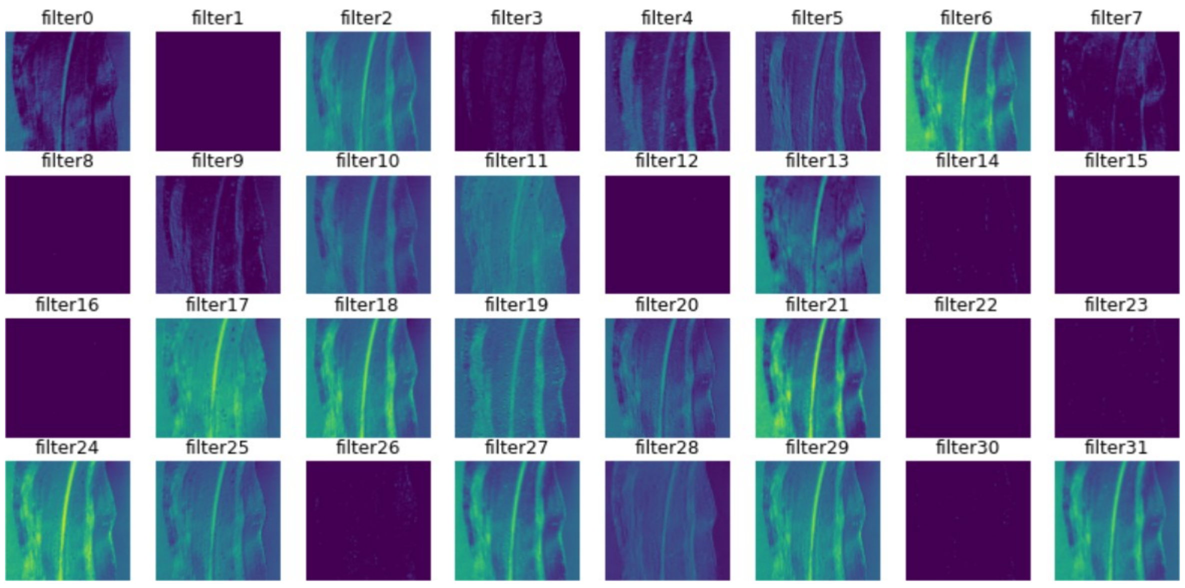


Figure 6.1.3.1.2 Convolution Layer 1

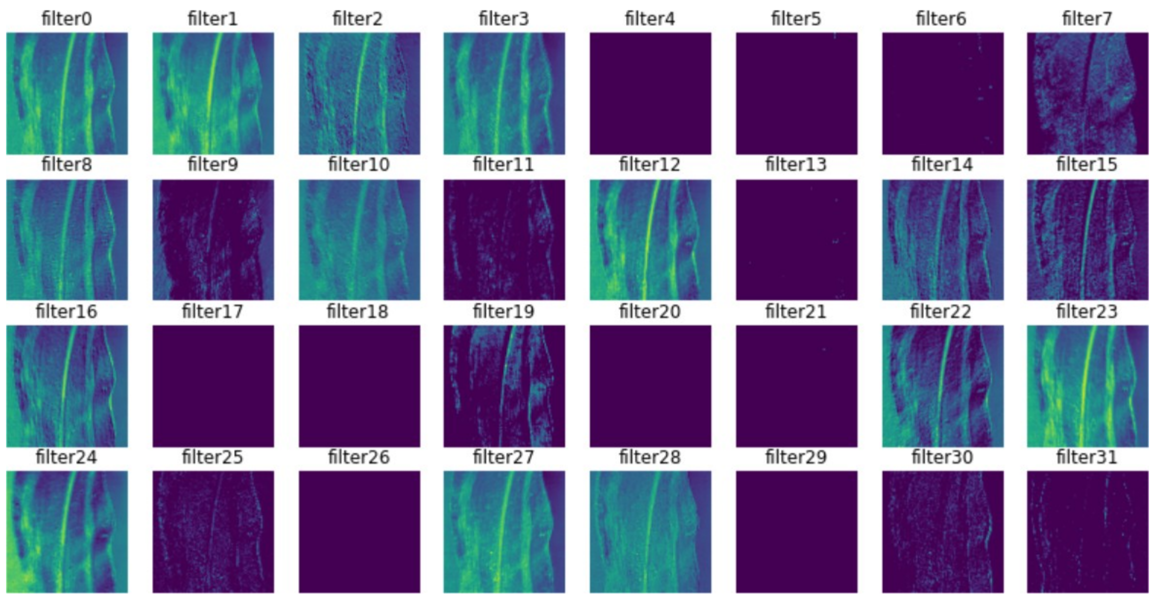


Figure 6.1.3.1.3 Maxpooling Layer 1

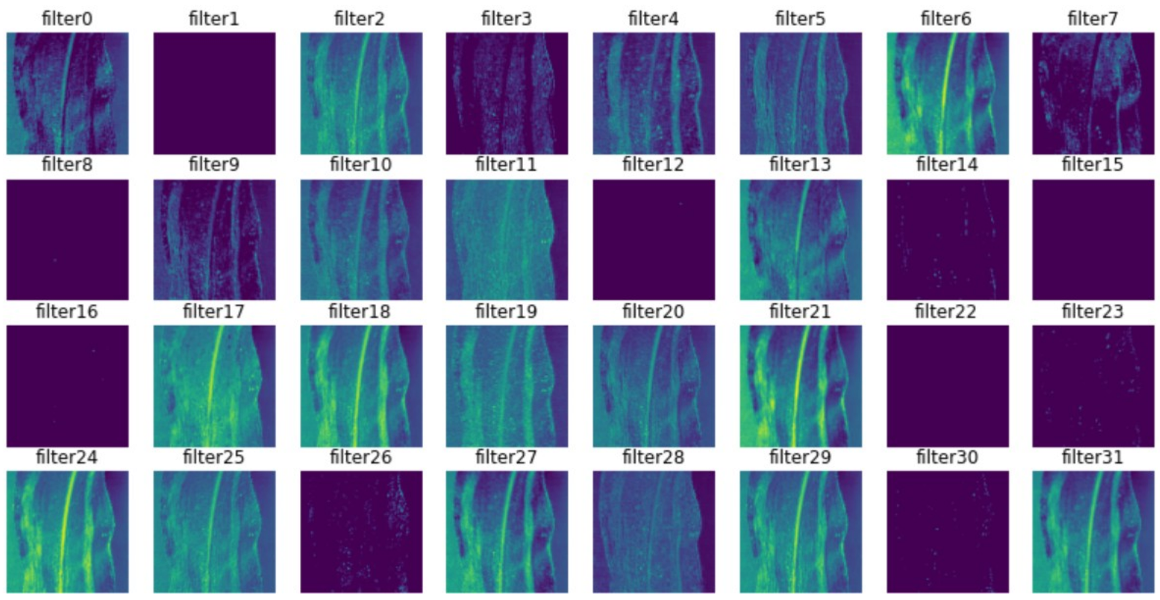


Figure 6.1.3.1.4 Convolution Layer 2

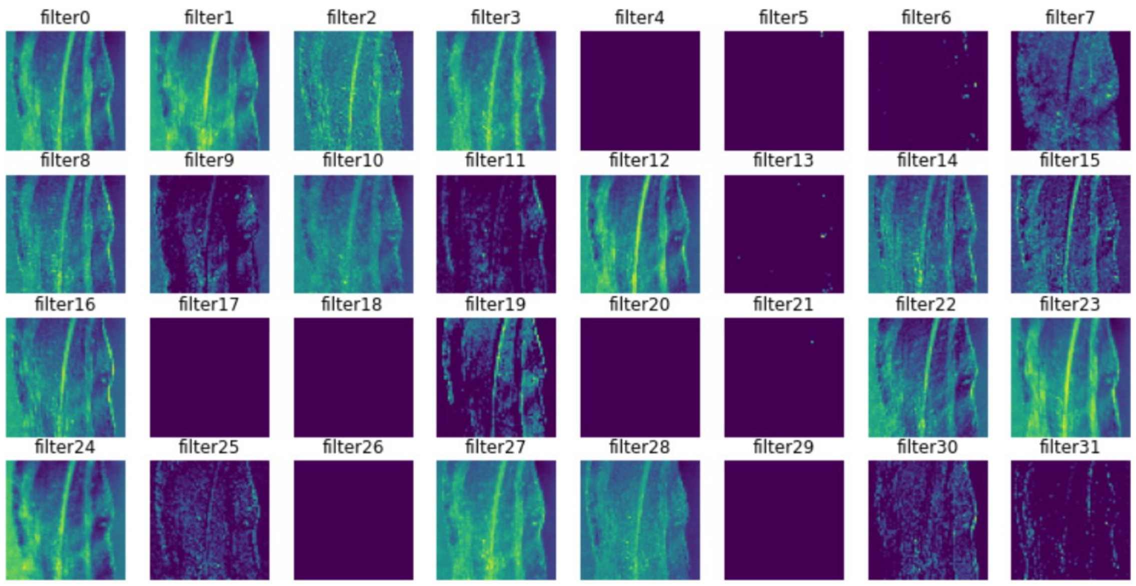


Figure 6.1.3.1.5 Maxpooling Layer 2

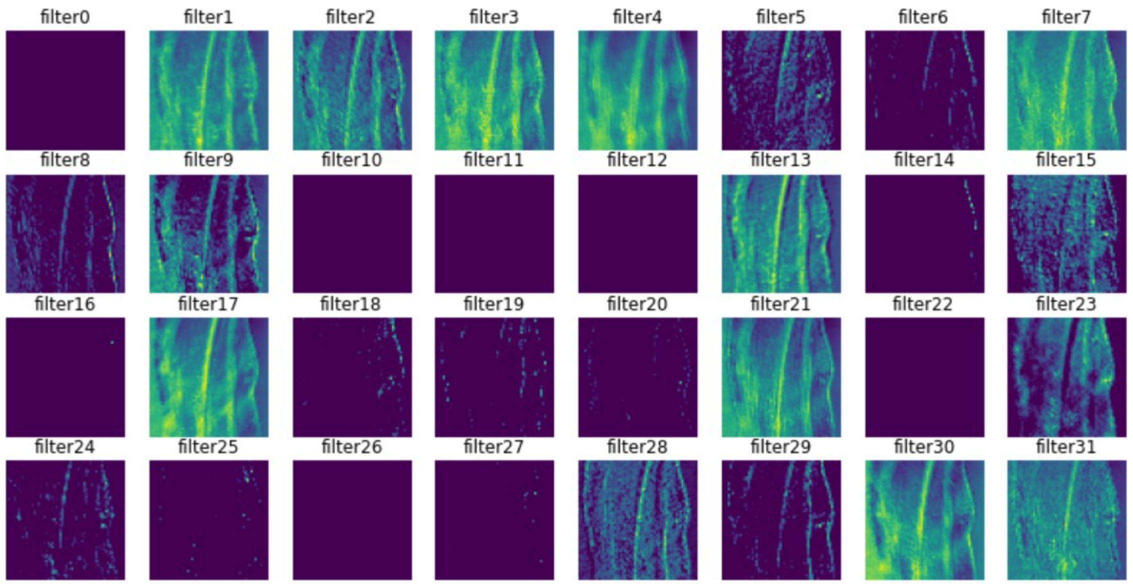


Figure 6.1.3.1.6 Convolution Layer 3

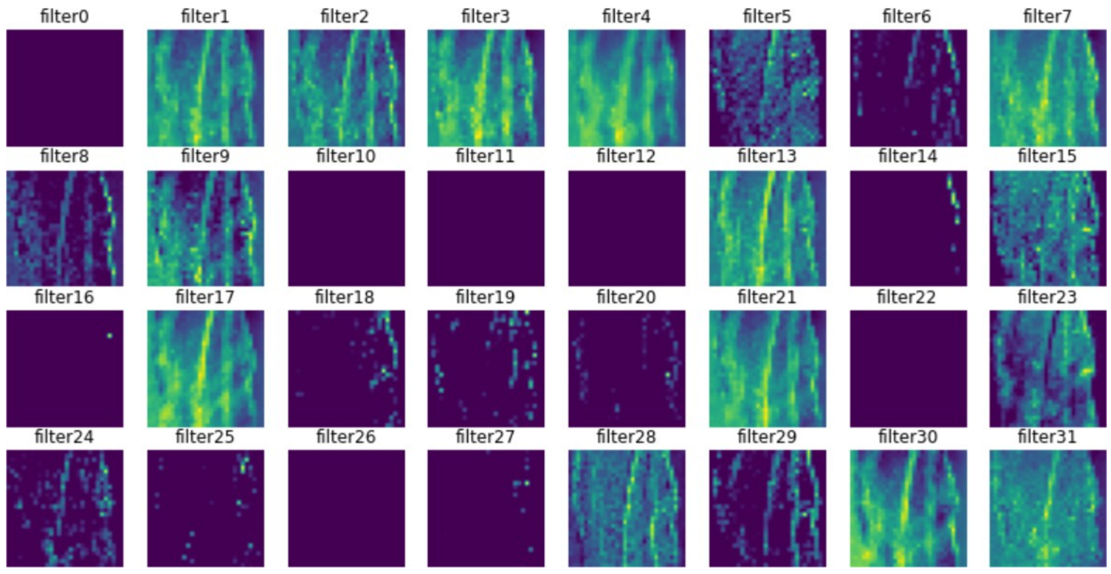


Figure 6.1.3.1.7 Maxpooling Layer 3

INNER LAYERS

The Fully Connected layers are the inner layers of a CNN model. They consist of the parameters such as weights and biases along with the neurons and are used to connect the neurons between two different layers. The fully connected layers are usually added before the output layer and therefore make the last few layers of a CNN model. In this model we have used a dense layer with 512 units and ReLU activation and then a dropout layer, and a dense layer with 128 units and ReLU activation.

OUTPUT LAYER

The output layer in this model is a dense layer with units as the number of classes in the dataset and softmax activation. The four classes in this model represent the three diseases Common Blight, Gray Spot, Common Rust and Healthy images. Softmax activation is used as the model is used to predict a multi-class classification. The softmax activation will output one value for each of the four nodes representing the four classes of images in the output layer.

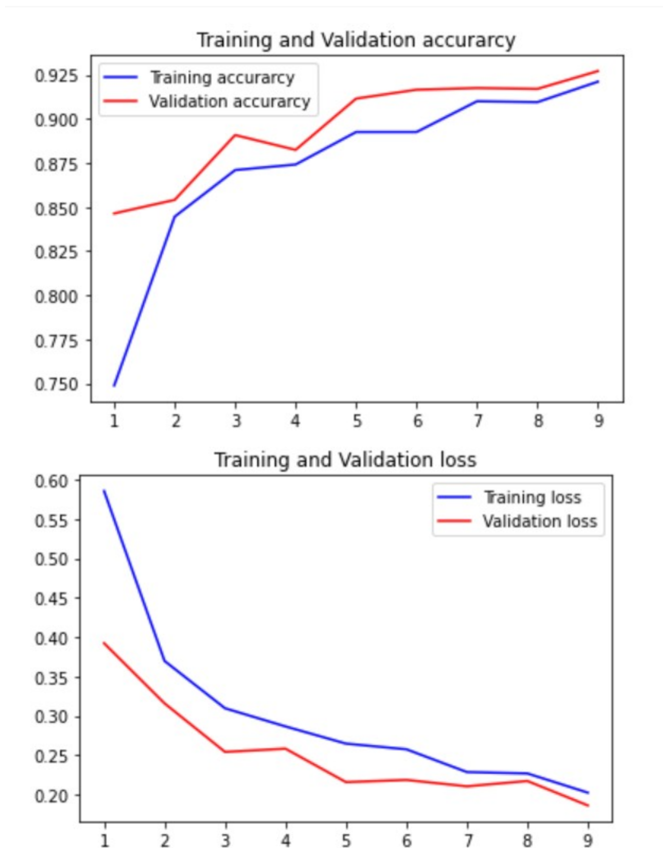


Figure 6.1.3.1.8 Training and Validation Accuracy and Loss graph

6.1.4 TESTING THE MODEL

The trained model is then used on the testing set to verify the model. A sample unseen image can also be used to test the model. Further after testing, the model is converted to a tflite format to be deployed to an android app. To convert the model we use the TFLiteConvertor from the Tensorflow library. It takes a keras model as the input and after conversion the output file is of tflite format.

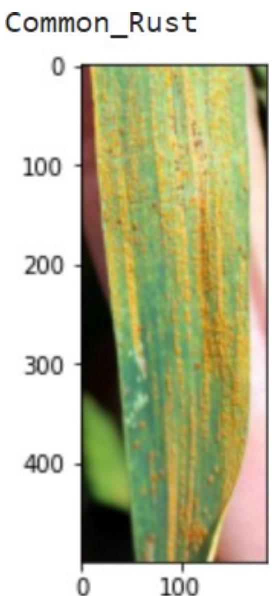
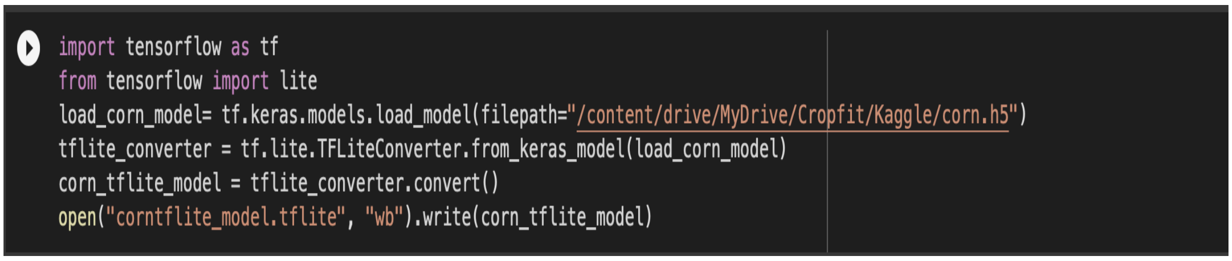


Figure 6.1.4.1 Result of Test Image



```
import tensorflow as tf
from tensorflow import lite
load_corn_model= tf.keras.models.load_model(filepath="/content/drive/MyDrive/Cropfit/Kaggle/corn.h5")
tflite_converter = tf.lite.TFLiteConverter.from_keras_model(load_corn_model)
corn_tflite_model = tflite_converter.convert()
open("corntflite_model.tflite", "wb").write(corn_tflite_model)
```

Figure 6.1.4.2 Conversion to TFLite

6.1.5 DEPLOYING MODEL TO ANDROID APPLICATION

The tflite model is then deployed to the app. An android application called Cropfit is developed in Android Studio using Java language. Using this app, the user can upload an image from local storage of the device or capture an image using the camera. This image after being processed to fit the requirements of the model input is fed to the model as input. The model processes the image then the result i.e the name of the disease of the plant will be displayed. An additional feature is that the app displays a remedy for the specific disease. This remedy medicine can be added to the cart and later can be purchased from the app.

Figure 6.1.5.1 Splash Screen of Cropfit App

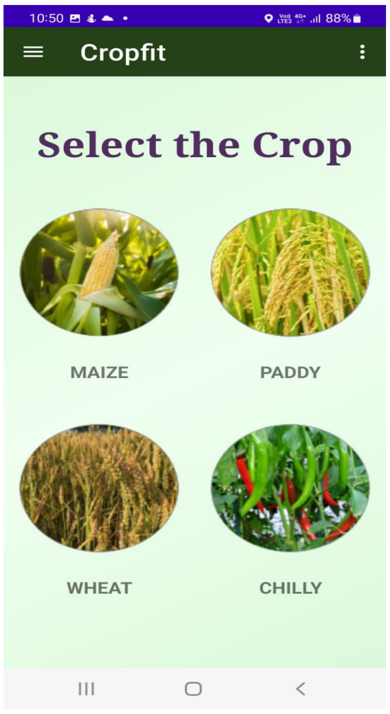
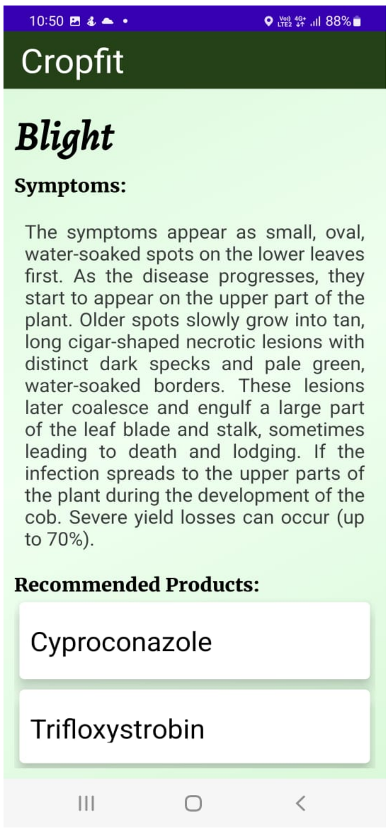


Figure 6.1.5.2 Homepage of Cropfit App


Cropfit

Blight

Symptoms:

The symptoms appear as small, oval, water-soaked spots on the lower leaves first. As the disease progresses, they start to appear on the upper part of the plant. Older spots slowly grow into tan, long cigar-shaped necrotic lesions with distinct dark specks and pale green, water-soaked borders. These lesions later coalesce and engulf a large part of the leaf blade and stalk, sometimes leading to death and lodging. If the infection spreads to the upper parts of the plant during the development of the cob. Severe yield losses can occur (up to 70%).

Recommended Products:

Cyproconazole

Trifloxystrobin

Figure 6.1.5.3 Result on Cropfit App

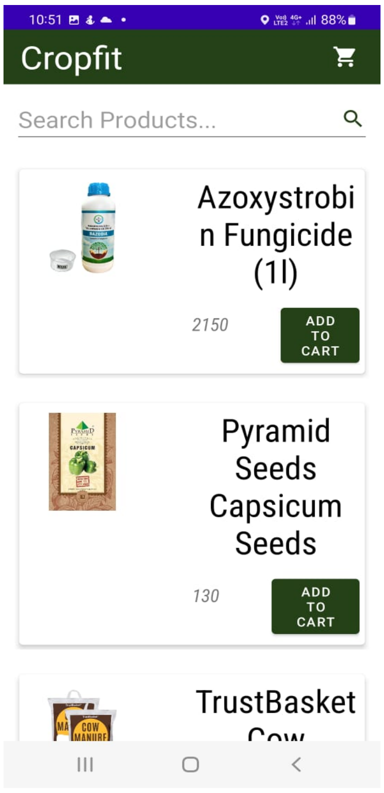

10:51 88%
Cropfit
Search Products...
Azoxystrobin Fungicide (1l)
2150
ADD TO CART
Pyramid Seeds Capsicum Seeds
130
ADD TO CART
TrustBasket Cow

Figure 6.1.5.4 Remedy and tools on Cropfit App

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

The proposed system was based on the identified issues and limitations of work of existing systems to identify crop diseases. In addition to the identification of the disease we provide a solution thus improving the features of the system. Moreover, with the help of an app farmers with basic education can make use of this system. Crop disease identification is not a simple task. It depends on an in-depth knowledge of the crops and their diseases and causes. In spite of there being plenty of research and study in this area, the systems proposed based on CNNs are not currently being used for field use due to uncertainty. The growth in technology has proved to be a step ahead in finding a solution. The proposed system and similar work is surely a development towards new technical agricultural tools that could contribute to a more sustainable and manageable crop cultivation. In our project an application is developed for the detection of corn plant diseases through the images of healthy and diseased plants by deploying a deep learning model using convolutional neural networks. The model took input images captured in the field or uploaded by the user. I hope this proposed system will make a contribution to the agriculture field.

7.2 FUTURE WORK

The primary goal for the future work of our proposed system would be to develop a complete application with help in case of more information needed and a means of communicating with a plant pathologist. This application would be an aid to farmers allowing quick and accurate disease predictions with an immediate remedy and expert support. Furthermore, future work will involve expanding the focus to more crops and their diseases.

REFERENCES

1. G. Shrestha, Deepsikha, M. Das and N. Dey, (2020),"Plant Disease Detection Using CNN", IEEE Applied Signal Processing Conference, Kolkata, India, pp. 109-113

2. KV Kumar and T Jayasankar, (2019), “An identification of crop disease using image segmentation”, International Journal Pharm Sci & Res, Vol.10, No. 3, pp. 1054-64

3. MD Hasan & Aniruddha Rakshit, (2019),”Recognition of Jute Disease by Leaf Image Classification Based on Convolutional Neural Network”, 10th International Conference on Computing, Communication and Networking Technologies, Kanpur, India, pp. 1-5

4. M. S. Arya, K. Anjali and D. Unni, (2018), "Detection of unhealthy plant leaves using image processing and genetic algorithm with Arduino", International Conference on Power, Signals, Control and Computation, Thrissur, India, pp. 1-5