GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Plant Disease Detection Using Convolutional Neural Networks

A Project Report of Capstone Project 2

Submitted by

DAMINI VERMA

(1613114018 /

16SCSE114043)

in partial fulfilment of the award of the degree of

Bachelor of Technology IN

Computer Science and Engineering

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Mr. G Nagarajan
Assistant Professor

May- 2020

GALGOTIAS UNIVERSITY

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"Plant Disease Detection Using Convolutional Neural Networks"** is the bonafide work of "DAMINI VERMA**"** who carried out the project work under my supervision.

<<Signature of the Head of the Department>>

<<Name>>

HEAD OF THE DEPARTMENT
<<Department Details>>

<<Signature of the Supervisor>>

Mr. G Nagarajan

<<Name>>

SUPERVISOR

Assistant Professor
School of Computing Science & Engineering

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1.	Abstract	1
2.	Introduction	2
3.	Existing System	5
4.	Proposed System	7
5.	Implementation	10
6.	Result	33
7.	Conclusion	36
8.	References	37

CHAPTER 1

ABSTRACT

When plants and crops are affected by pests it affects the agricultural production of the country. Usually farmers or experts observe the plants with naked eye for detection and identification of disease. But this method can be time processing, expensive and inaccurate. Automatic detection using image processing techniques provide fast and accurate results. This paper is concerned with a new approach to the development of plant disease recognition model, based on leaf image classification, by the use of deep convolutional networks. Advances in computer vision present an opportunity to expand and enhance the practice of precise plant protection and extend the market of computer vision applications in the field of precision agriculture. Novel way of training and the methodology used facilitate a quick and easy system implementation in practice. All essential steps required for implementing this disease recognition model will be fully described throughout the paper, starting from gathering images in order to create a database, assessed by agricultural experts, a deep learning framework to perform the deep CNN training. This method paper is a new approach in detecting plant diseases using the deep convolutional neural network trained and fine-tuned to fit accurately to the database of a plant's leaves that was gathered independently for diverse plant diseases. The advance and novelty of the developed model lie in its simplicity; healthy leaves and background images are in line with other classes, enabling the model to distinguish between diseased leaves and healthy ones or from the environment by using CNN.

CHAPTER 2

INTRODUCTION

The problem of efficient plant disease protection is closely related to the problems of sustainable agriculture Inexperienced pesticide usage can cause the development of long-term resistance of the pathogens, severely reducing the ability to fight back. Timely and accurate diagnosis of plant diseases is one of the pillars of precision agriculture. It is crucial to prevent unnecessary waste of financial and other resources, thus achieving healthier production in this changing environment, appropriate and timely disease identification including early prevention has never been more important. There are several ways to detect plant pathologies. Some diseases do not have any visible symptoms, or the effect becomes noticeable too late to act, and in those situations, a sophisticated analysis is obligatory. However, most diseases generate some kind of manifestation in the visible spectrum, so the naked eye examination of a trained professional is the prime technique adopted in practice for plant disease detection. In order to achieve accurate plant disease diagnostics a plant pathologist should possess good observation skills so that one can identify characteristic symptoms. Variations in symptoms indicated by diseased plants may lead to an improper diagnosis since amateur gardeners and hobbyists could have more difficulties determining it than a professional plant pathologist. An automated system designed to help identify plant diseases by the plant's appearance and visual symptoms could be of great help to amateurs in the gardening process and also trained professionals as a verification system in disease diagnostics.

Traditionally, identification of plant diseases has relied on human annotation by visual inspection. Nowadays, it is combined or substituted with various technologies such as immunoassays (e.g., enzyme-linked immunosorbent assay, ELISA) and PCR or RNA-seq to detect pathogen-specific antigens or oligonucleotides, respectively. Moreover, recent technical advances and dramatic cost reductions in the field of digital image acquisition have allowed the introduction of an array of image-based diagnosis methods at a practical level. However, as the acquired image encloses condensed information that is extremely difficult for the computer to process, it requires a pre-processing step to extract a certain feature (e.g., color and shape) that is manually predefined by experts. In such situations, deep learning is typically used because it allows the computer to autonomously learn the most suitable feature without human intervention. An initial attempt to use deep learning for image-based plant disease diagnosis was reported in 2016, where the trained model was able to classify 14 crops and 26 diseases with an accuracy of 99.35% against optical images. Since then, successive generations of deep-learning-based disease diagnosis in various crops have been reported.

Among various network architectures used in deep learning, convolutional neural networks (CNN) are widely used in image recognition.

Plant disease has long been one of the major threats to food security because it dramatically reduces the crop yield and compromises its quality. Accurate and precise diagnosis of diseases has been a significant challenge. Timely and accurate diagnosis of plant diseases is one of the pillars of precision agriculture. It is crucial to prevent unnecessary waste of financial and other

resources, thus achieving healthier production in this changing environment, appropriate and timely disease identification including early prevention has never been more important.

An automated system designed to help identify plant diseases by the plant's appearance and visual symptoms could be of great help to amateurs in the gardening process and also trained professionals as a verification system in disease diagnostics.

Timely and accurate diagnosis of plant diseases is one of the pillars of precision agriculture. It is crucial to prevent unnecessary waste of financial and other resources, thus achieving healthier production in this changing environment, appropriate and timely disease identification including early prevention has never been more important.

Some diseases do not have any visible symptoms, or the effect becomes noticeable too late to act, and in those situations, a sophisticated analysis is obligatory.

However, most diseases generate some kind of manifestation in the visible spectrum, so the naked eye examination of a trained professional is the prime technique adopted in practice for plant disease detection.

In order to achieve accurate plant disease diagnostics a plant pathologist should possess good observation skills so that one can identify characteristic symptoms.

CHAPTER 3

EXISTING SYSTEM

Traditionally, identification of plant diseases has relied on human annotation by visual inspection. Numerous procedures are currently in use for plant disease detection applying computer vision. One of them is disease detection by extracting colour feature as authors in [17] have presented. In addition, plant disease detection could be achieved by extracting shape features method. There are some approaches which apply the feed-forward back propagation of neural networks consisting of one input, one output, and one hidden layer for the needs of identifying the species of leaf, pest, or disease. Also, detection and differentiation of plant diseases can be achieved using Support Vector Machine algorithms. This technique was implemented for sugar beet diseases, where, depending on the type and stage of disease, the classification accuracy was between 65% and 90%.

Likewise, there are methods that combine the feature extraction and Neural Network Ensemble (NNE) for plant disease recognition. Through training a definite number of neural networks and combining their results after that, NNE offers a better generalization of learning ability. Such method was implemented only for recognizing tea leaf diseases with final testing accuracy of 91%.

Another approach based on leaf images and using ANNs as a technique for an automatic detection and classification of plant diseases was used in conjunction with K-means as a clustering procedure. ANN consisted of 10 hidden layers. The number of outputs was 6 which was the number of classes representing five diseases along with the case of a healthy leaf. On average, the accuracy of classification using this approach was 94.67%.

Nowadays, it is combined or substituted with various technologies such as immunoassays (e.g., enzyme-linked immunosorbent assay, ELISA) and PCR or RNA-seq to detect pathogen-specific antigens or oligonucleotides, respectively. Moreover, recent technical advances and dramatic cost reductions in the field of digital image acquisition have allowed the introduction of an array of image-based diagnosis methods at a practical level. However, as the acquired image encloses condensed information that is extremely difficult for the computer to process, it requires a pre-processing step to extract a certain feature (e.g., colour and shape) that is manually predefined by experts.

In such situations, deep learning is typically used because it allows the computer to autonomously learn the most suitable feature without human intervention. An initial attempt to use deep learning for image-based plant disease diagnosis was reported in 2016, where the trained model was able to classify 14 crops and 26 diseases with an accuracy of 99.35% against optical images. Since then, successive generations of deep-learning-based disease diagnosis in various crops have been reported.

CHAPTER 4

PROPOSED SYSTEM

Plant Disease Detection Model

An automated system designed to help identify plant diseases by the plant's appearance and visual symptoms could be of great help to amateurs in the gardening process and also trained professionals as a verification system in disease diagnostics.

Advances in computer vision present an opportunity to expand and enhance the practice of precise plant protection and extend the market of computer vision applications in the field of precision agriculture.

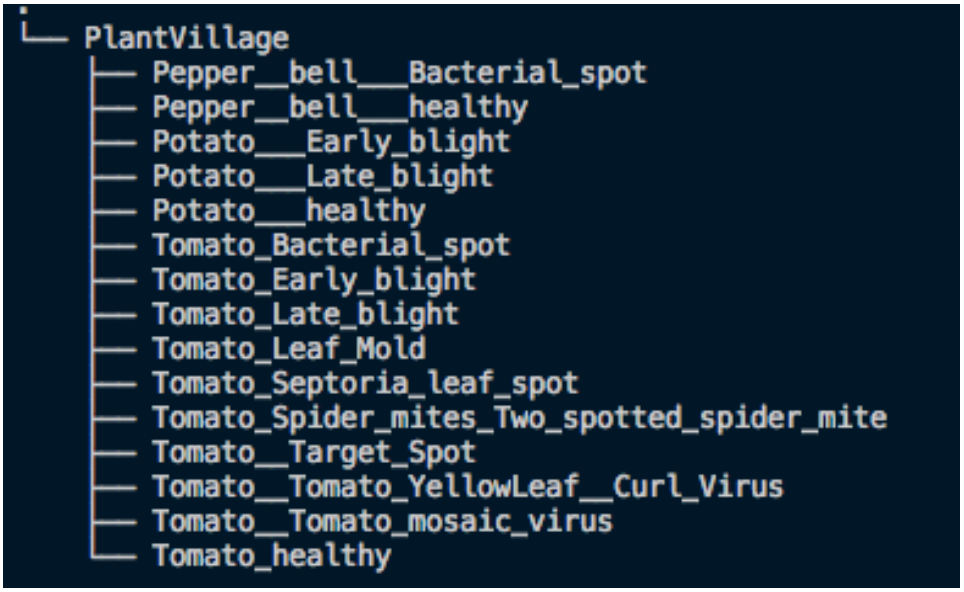
Exploiting common digital image processing techniques such as colour analysis and thresholding will be used with the aim of detection and classification of plant diseases.

The acquired image encloses condensed information that is extremely difficult for the computer to process, it requires a pre-processing step to extract a certain feature (e.g., colour and shape) that is manually predefined by experts. In such situations, deep learning is typically used because it allows the computer to autonomously learn the most suitable feature without human intervention.

Among various network architectures used in deep learning, convolutional neural networks (CNN) are widely used in image recognition.

Dataset Used

The dataset that was used is the Plant Village dataset taken from Kaggle.



CHAPTER 5

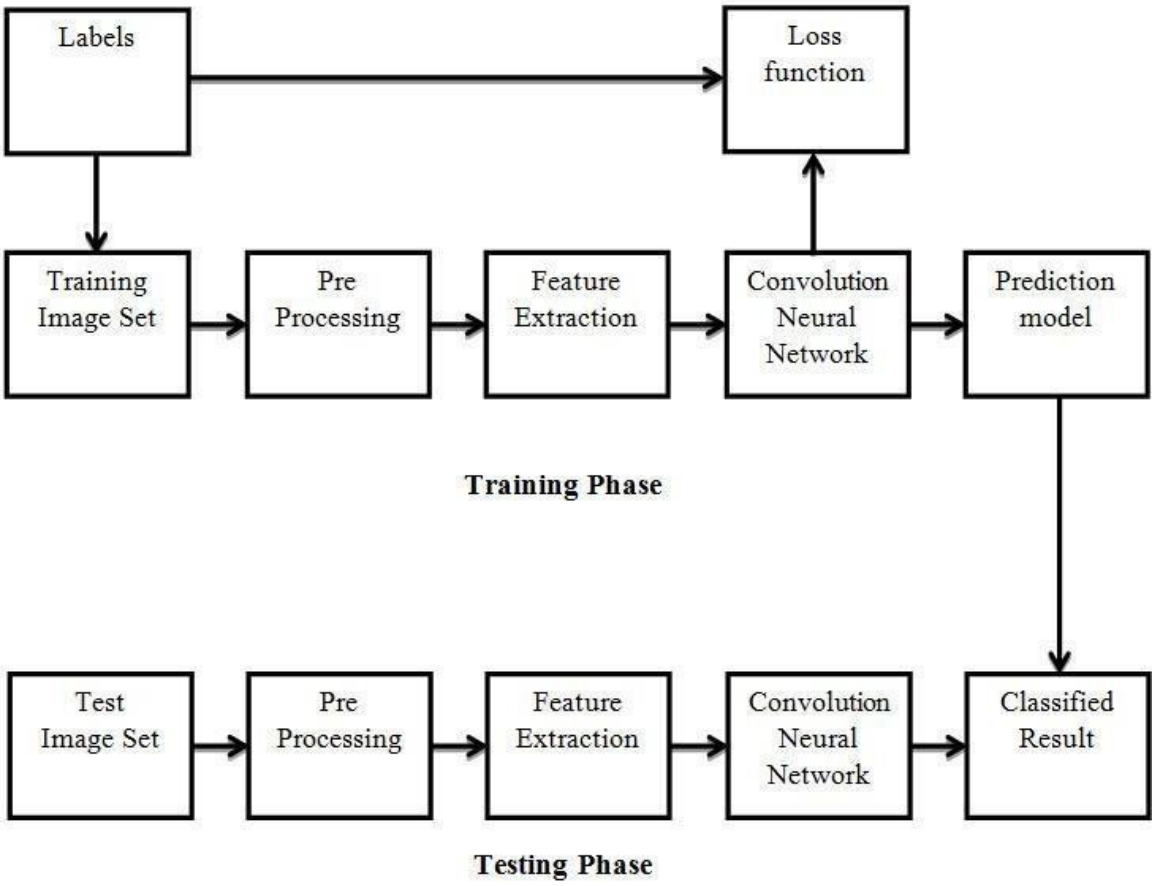
IMPLEMENTATION

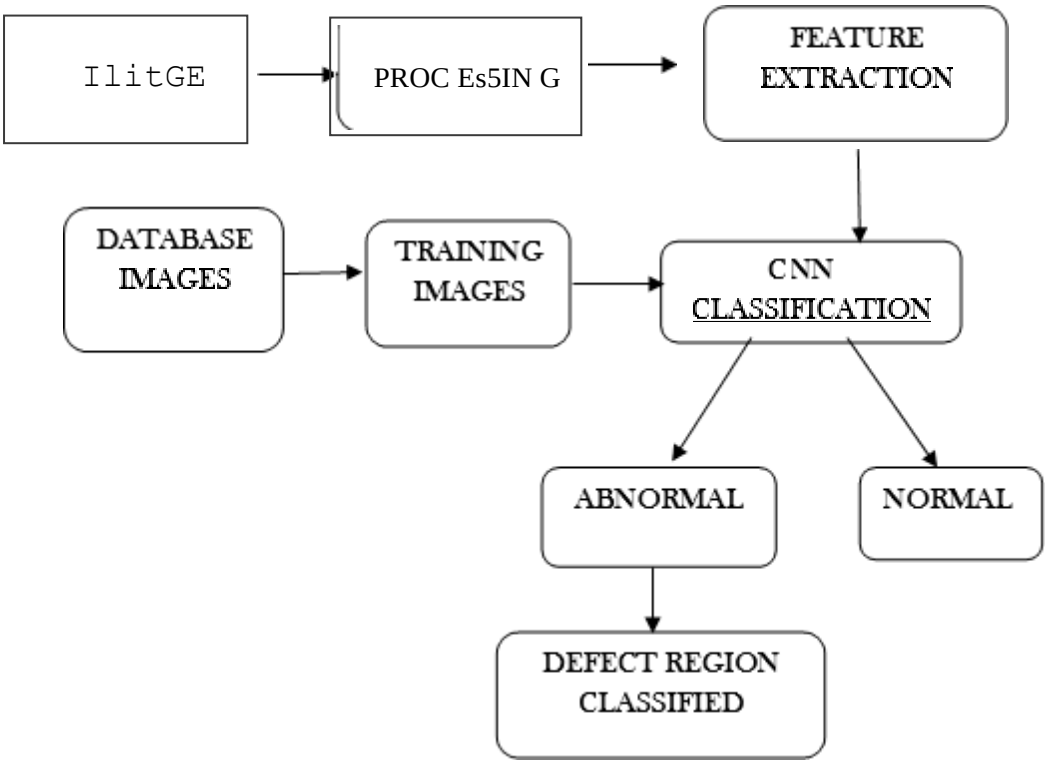
Architectural Diagrams

1. System Design

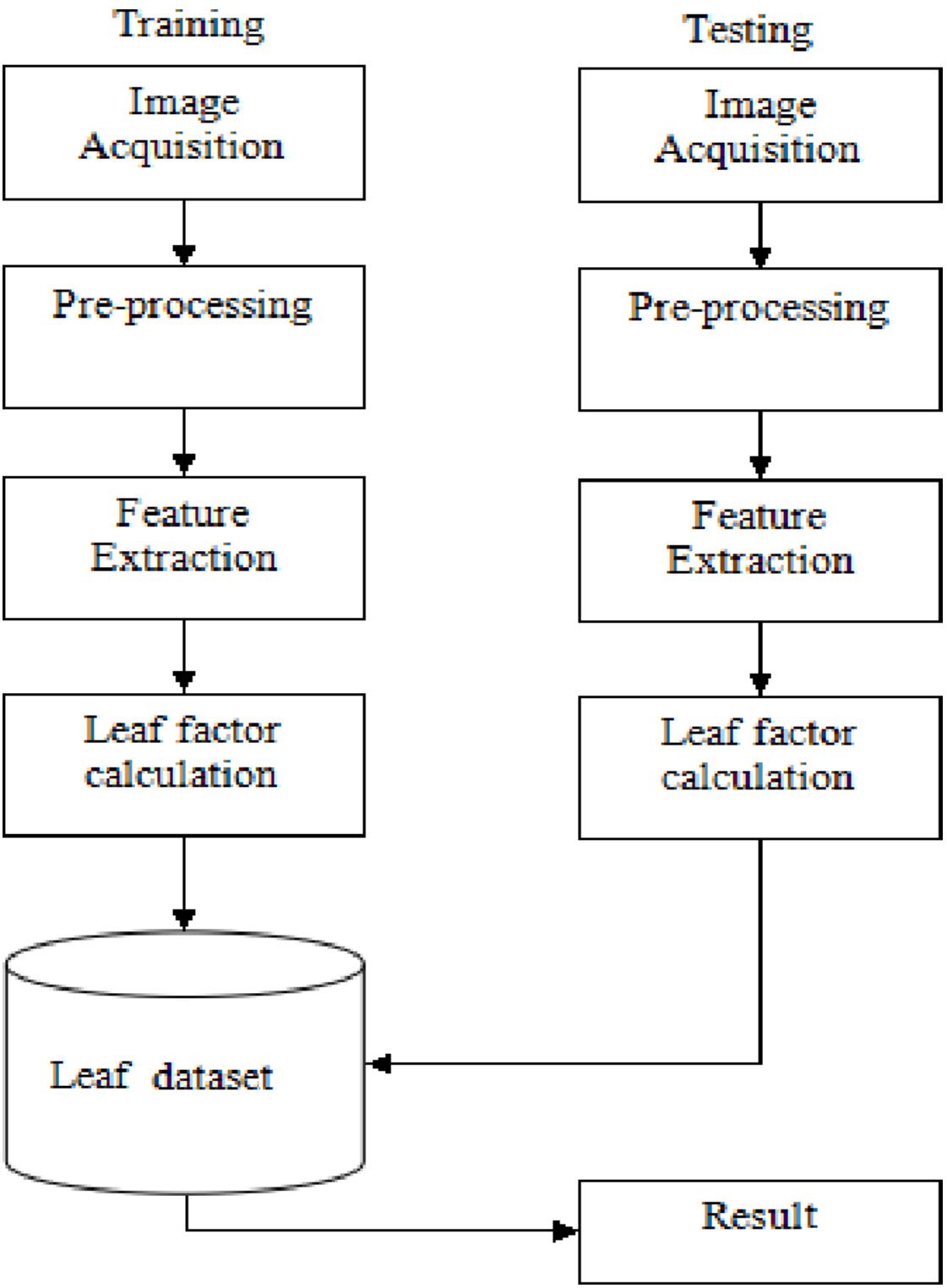
System design shows the overall design of system. In this section we discuss in detail the design aspects of the system:

1.1 System Diagram




IlitGE
PROC Es5IN G
FEATURE EXTRACTION
DATABASE IMAGES
TRAINING IMAGES
CNN CLASSIFICATION
ABNORMAL
NORMAL
DEFECT REGION CLASSIFIED

1.2 System Flowchart



Source Code:

```
import numpy as np

import pickle

import cv2

from os import listdir

from sklearn.preprocessing import LabelBinarizer

from keras.models import Sequential

from keras.layers.normalization import BatchNormalization

from keras.layers.convolutional import Conv2D

from keras.layers.convolutional import MaxPooling2D

from keras.layers.core import Activation, Flatten, Dropout, Dense

from keras import backend as K

from keras.preprocessing.image import ImageDataGenerator

from keras.optimizers import Adam

from keras.preprocessing import image

from keras.preprocessing.image import img_to_array

from sklearn.preprocessing import MultiLabelBinarizer

from sklearn.model_selection import train_test_split
```

```
import matplotlib.pyplot as plt

import tensorflow



EPOCHS = 25

INIT_LR = 1e-3

BS = 32

default_image_size = tuple((256, 256))

image_size = 0

directory_root = '../input/plantvillage/'

width=256

height=256

depth=3



def convert_image_to_array(image_dir):

    try:

        image = cv2.imread(image_dir)

        if image is not None :

            image = cv2.resize(image, default_image_size)

            return img_to_array(image)

        else :
```

```
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None

image_list, label_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
    for directory in root_dir :
        # remove .DS_Store from list
         if directory == ".DS_Store" :
            root_dir.remove(directory)

    for plant_folder in root_dir :
        plant_disease_folder_list = listdir(f"{directory_root}/{plant_folder}")

        for disease_folder in plant_disease_folder_list :
            # remove .DS_Store from list
            if disease_folder == ".DS_Store" :
```

```
            plant_disease_folder_list.remove(disease_folder)


        for plant_disease_folder in plant_disease_folder_list:
            print(f"[INFO] Processing {plant_disease_folder} ...")
            plant_disease_image_list = listdir(f"{directory_root}/{plant_folder}/{plant_disease_folder}/")

            for single_plant_disease_image in plant_disease_image_list :
                if single_plant_disease_image == ".DS_Store" :
                    plant_disease_image_list.remove(single_plant_disease_image)

            for image in plant_disease_image_list[:200]:
                image_directory = f"{directory_root}/{plant_folder}/{plant_disease_folder}/{image}"
                if image_directory.endswith(".jpg") == True or image_directory.endswith(".JPG") == True:
                    image_list.append(convert_image_to_array(image_directory))
                    label_list.append(plant_disease_folder)
    print("[INFO] Image loading completed")
except Exception as e:
```

```
    print(f"Error : {e}")


image_size = len(image_list)


label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)
pickle.dump(label_binarizer,open('label_transform.pkl', 'wb'))
n_classes = len(label_binarizer.classes_)


print(label_binarizer.classes_)


np_image_list = np.array(image_list, dtype=np.float16) / 225.0


print("[INFO] Spliting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_size=0.2,
random_state = 42)


aug = ImageDataGenerator(
    rotation_range=25, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2,
```

```
    zoom_range=0.2,horizontal_flip=True,
    fill_mode="nearest")


model = Sequential()
inputShape = (height, width, depth)
chanDim = -1
if K.image_data_format() == "channels_first":
    inputShape = (depth, height, width)
    chanDim = 1
model.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(128, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(1024))

model.add(Activation("relu"))

model.add(BatchNormalization())

model.add(Dropout(0.5))

model.add(Dense(n_classes))

model.add(Activation("softmax"))



opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)

# distribution
```

```
model.compile(loss="binary_crossentropy", optimizer=opt,metrics=["accuracy"])

# train the network

print("[INFO] training network...")



history = model.fit_generator(

    aug.flow(x_train, y_train, batch_size=BS),

    validation_data=(x_test, y_test),

    steps_per_epoch=len(x_train) // BS,

    epochs=EPOCHS, verbose=1

    )



acc = history.history['acc']

val_acc = history.history['val_acc']

loss = history.history['loss']

val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

#Train and validation accuracy

plt.plot(epochs, acc, 'b', label='Training accurarcy')

plt.plot(epochs, val_acc, 'r', label='Validation accurarcy')

plt.title('Training and Validation accurarcy')
```

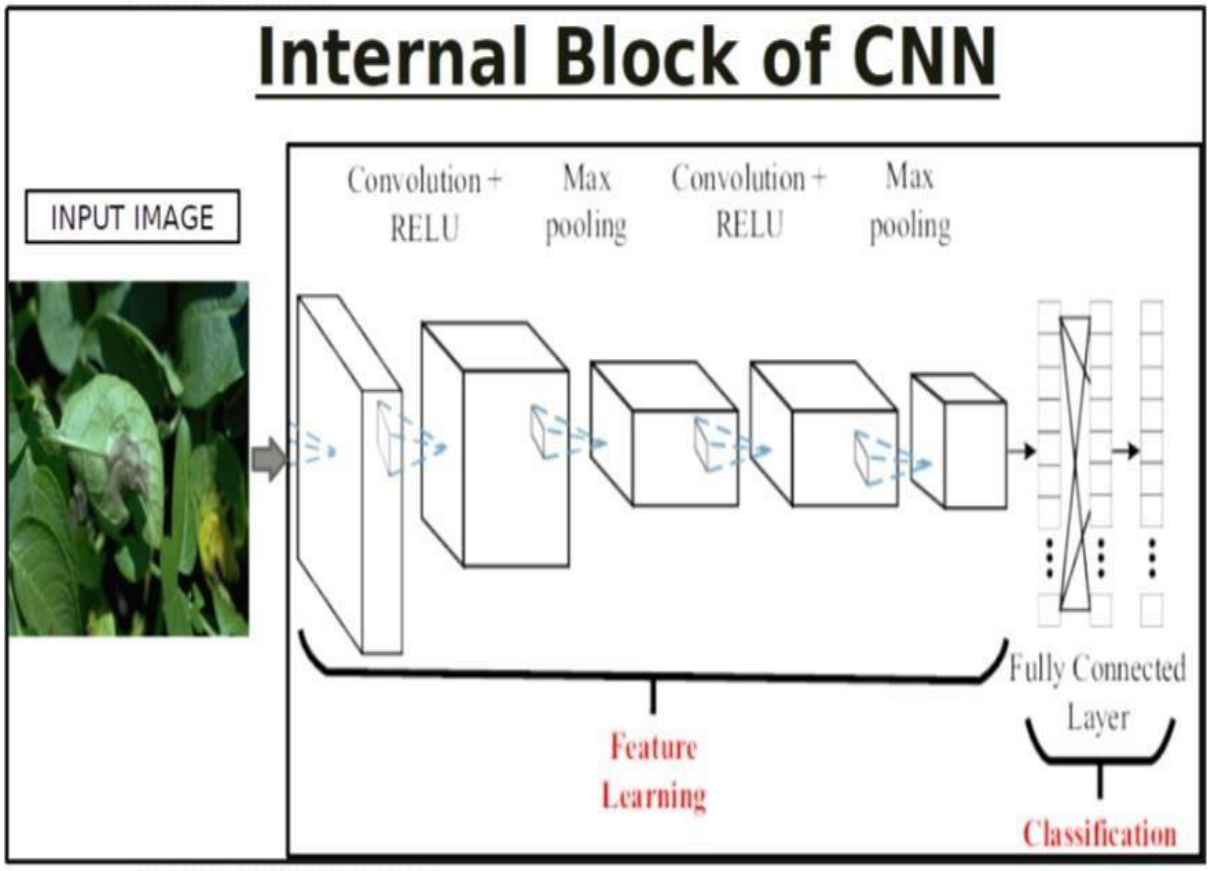
```
plt.legend()


plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()


print("[INFO] Calculating model accuracy")
scores = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")


# save the model to disk
print("[INFO] Saving model...")
pickle.dump(model,open('cnn_model.pkl', 'wb'))
```

2. Phases in Plant Disease Detection

2.1 Image pre-processing and labelling

Pre-processing images commonly involves removing low-frequency background noise, normalizing the intensity of the individual particle's images, removing reflections, and masking portions of images. Image preprocessing is the technique of enhancing data. Furthermore, procedure of image preprocessing involves cropping of all the images manually, making the square around the leaves, in order to highlight the region of interest (plant leaves). During the phase of collecting the images for the dataset, images with smaller resolution and dimension less than 500 pixels were not considered as valid images for the dataset. In addition, only the images where the region of interest was in higher resolution were marked as eligible candidates for the dataset. In that way, it was ensured that images contain all the needed information for feature learning. Many resources can be found by searching across the Internet, but their relevance is often unreliable. In the interest of confirming the accuracy of classes in the dataset, initially grouped by a keywords search, agricultural experts examined leaf images and labeled all the images with appropriate disease acronym. As it is known, it is important to use accurately classified images for the training and validation dataset. Only in that way may an appropriate and reliable detecting model be developed. In this stage, duplicated images that were left after the initial iteration of gathering and grouping images into classes were removed from the dataset.

2.2 Neural Network Training



Training the deep convolutional neural network for making an image classification model from a dataset is proposed. Tensor Flow 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 you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. Tensor Flow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well. In machine learning, a convolutional neural network is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons

is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli in a restricted region of space known as the receptive field. The receptive fields of different neurons partially overlap such that they tile the visual field. The response of an individual neuron to stimuli within its receptive field can be approximated mathematically by a convolution operation. Convolutional networks were inspired by biological processes and are variations of multilayer perceptron designed to use minimal amounts of pre-processing. They have wide applications in image and video recognition, recommender systems and natural language processing.

Main Steps to build a CNN (or) Conv net:

Convolution Operation

ReLU Layer (Rectified Linear Unit)

Pooling Layer (Max Pooling)

Flattening

Fully Connected Layer

Start Writing Code.

1. Convolution is the first layer to extract features from the input image and it learns the relationship between features using kernel or filters with input images.

2. ReLU Layer: ReLU stands for the Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0,x)$. we use this because to introduce the non-linearity to CNN.

3. Pooling Layer: it is used to reduce the number of parameters by downsampling and retain only the valuable information to process further. There are types of Pooling:

Max Pooling (Choose this).

Average and Sum pooling.

4. Flattening: we flatten our entire matrix into a vector like a vertical one. so, that it will be passed to the input layer.

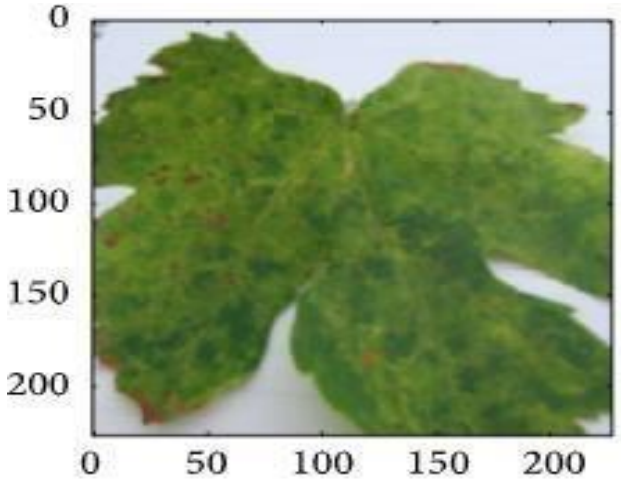
5. Fully Connected Layer: we pass our flatten vector into input Layer .we combined these features to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs.

Convolutional neural networks (CNNs) consist of multiple layers of receptive fields. These are small neuron collections which process portions of the input image. The outputs of these collections are then tiled so that their input regions overlap, to obtain a higher-resolution representation of the original image; this is repeated for every such layer. Tiling allows CNNs to tolerate translation of the input image. Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters. They also consist of various combinations of convolutional and fully connected layers, with point wise nonlinearity applied at the end of or after each layer. A convolution operation on small regions of input is introduced to reduce the number of free parameters and improve generalization .One major advantage of convolutional networks is the use of shared weight in convolutional layers, which means that the same filter (weights bank) is used for each pixel in the layer; this both reduces memory footprint and improves performance. The layer's parameters are comprised of a set of learnable kernels which possess a small receptive field but extend through the full depth of the input volume. Rectified Linear Units (Re LU) are used as substitute for saturating nonlinearities. This activation function adaptively learns the parameters of rectifiers and improves accuracy at negligible extra computational cost.

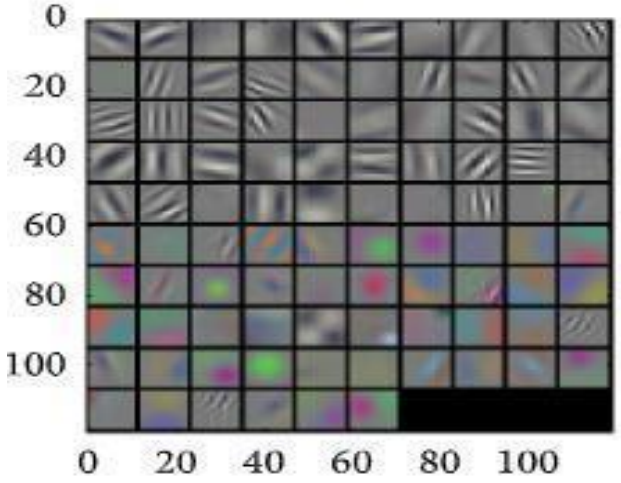
Deep CNN with Re LUs trains several times faster. This method is applied to the output of every convolutional and fully connected layer. In CNN, neurons within a hidden layer are

segmented into “feature maps.” The neurons within a feature map share the same weight and bias. The neurons within the feature map search for the same feature. These neurons are unique since they are connected to different neurons in the lower layer. So, for the first hidden layer, neurons within a feature map will be connected to different regions of the input image. The hidden layer is segmented into feature maps where each neuron in a feature map looks for the same feature but at different positions of the input image. Basically, the feature map is the result of applying convolution across an image. The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map. When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume because such network architecture does not take the spatial structure of the data into account. Convolutional networks exploit spatially local correlation by enforcing a local connectivity pattern between neurons of adjacent layers: each neuron is connected to only a small region of the input volume. The extent of this connectivity is a hyper parameter called the receptive field of the neuron. The connections are local in space (along width and height), but always extend along the entire depth of the input volume. Such architecture ensures that the learnt filters produce the strongest

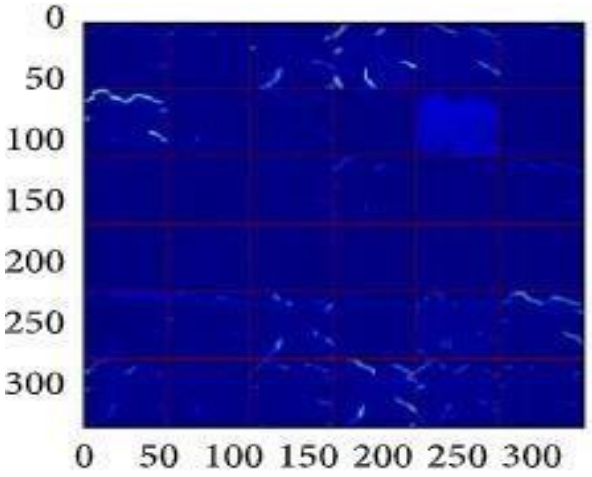
response to a spatially local input pattern. Three hyper parameters control the size of the output volume of the convolutional layer: the depth, stride and zero-padding.



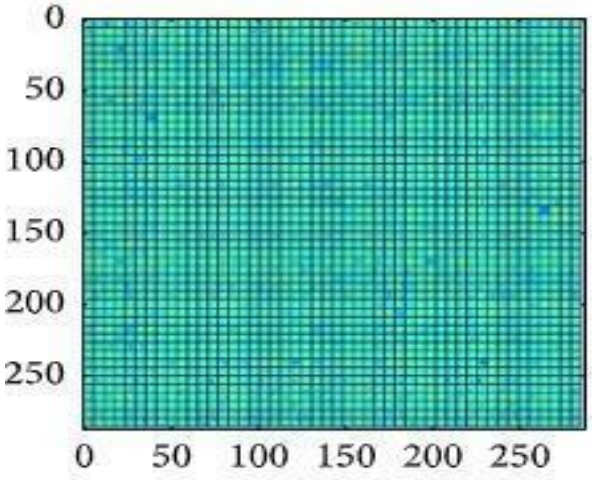
(a)



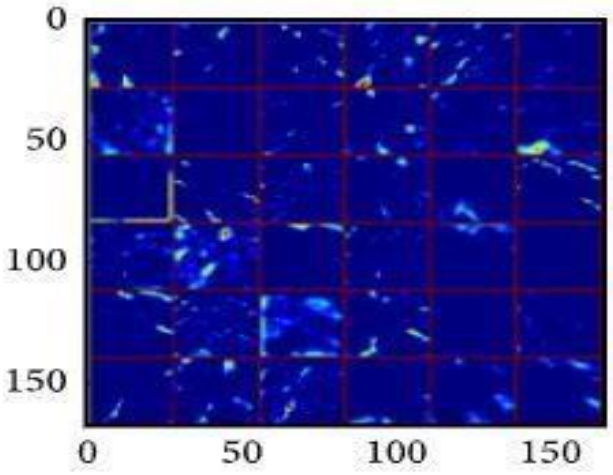
(b)


0
50
100
150
200
250
300
0 50 100 150 200 250 300

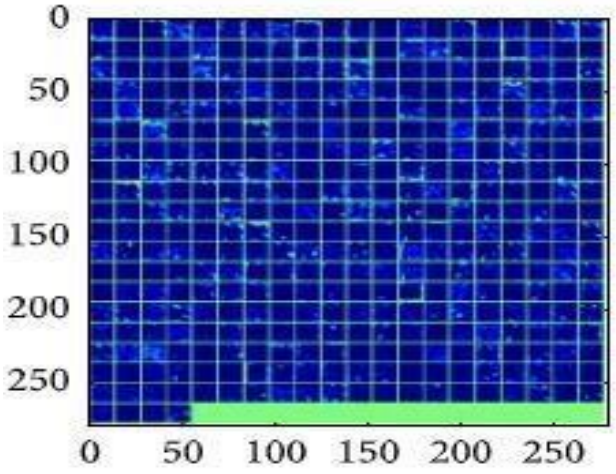
(c)


0
50
100
150
200
250
0 50 100 150 200 250

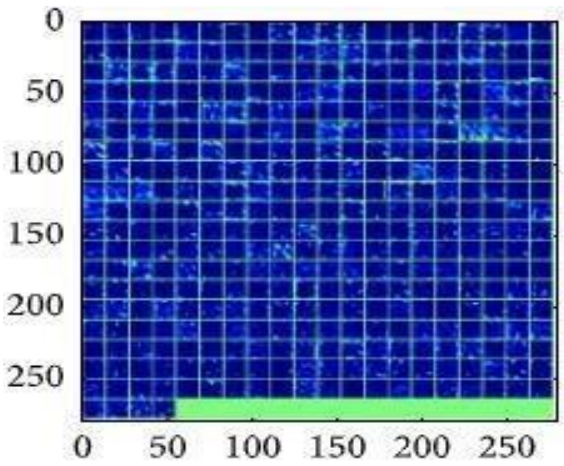
(d)



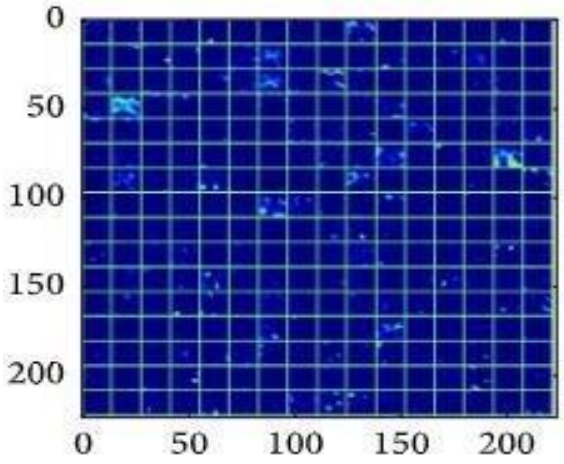
(e)



(f)

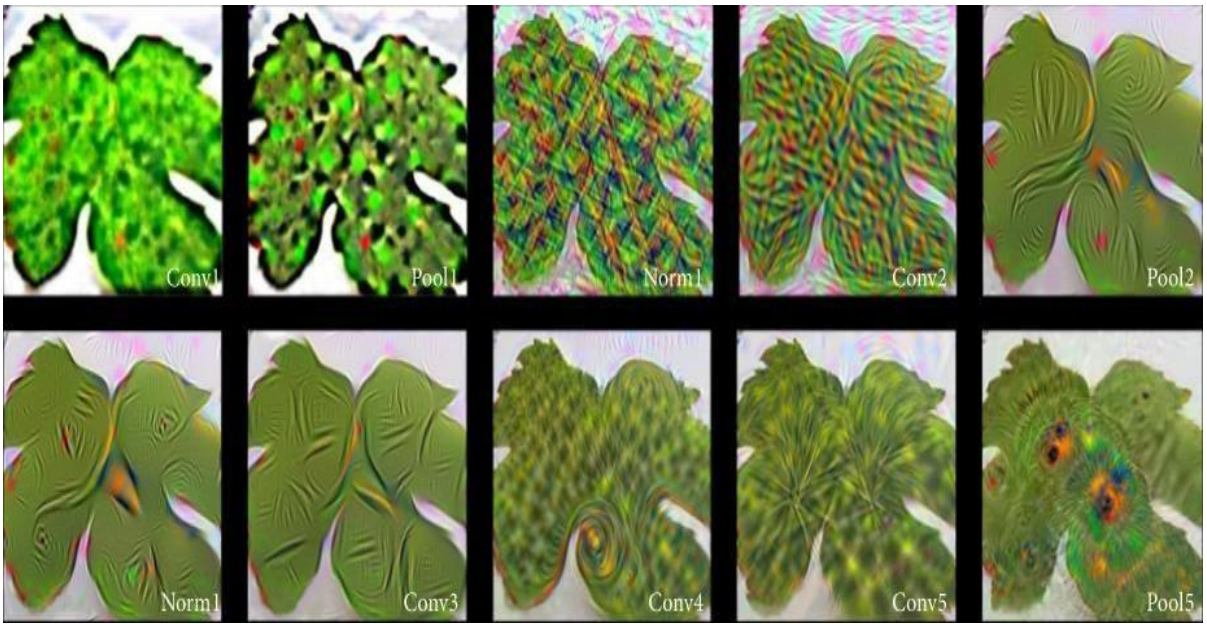


(g)



(h)

Visualization of features in trained classification model: (a) original image; (b) the first layer filters (c) the first layer output (d) the second layer filters; (e) the second layer output (f) the third layer output (g) the fourth layer output; (h) the fifth layer output



Output layer images

CHAPTER 6

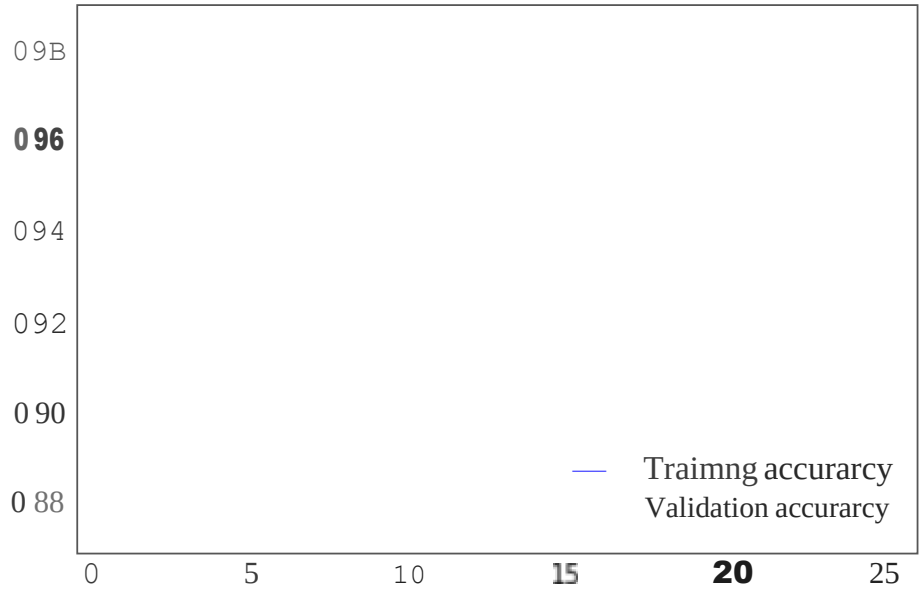
RESULT

The results presented in this section are related to training with the whole database containing both original and augmented images. As it is known that convolutional networks are able to learn features when trained on larger datasets, results achieved when trained with only original images will not be explored. After fine-tuning the parameters of the network, an overall accuracy of 96.77% was achieved. Furthermore, the trained model was tested on each class individually. Test was performed on every image from the validation set. As suggested by good practice principles, achieved results should be compared with some other results. In addition, there are still no commercial solutions on the market, except those dealing with plant species recognition based on the leaf's images.

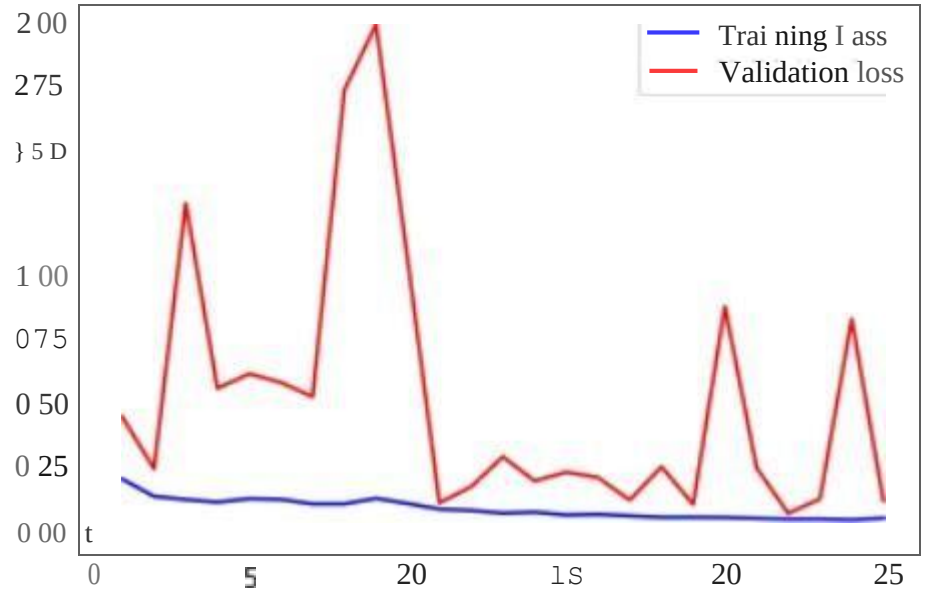
Crop diseases -->Tomato Leaf .

Bacterial Leafspot	**Late BLight**	EarLy BLight

Training and Validation accurarcy



Training and Validation loss



CHAPTER 7

CONCLUSION

Plant Diseases are major food threats that should have to overcome before it leads to further loss of the entire field. But, often framers unable to distinguish between similar symptoms but ace different diseases. This will mislead to wrong or overdosage offertilizers.

In this project, an approach of using deep learning method was explored in order to automatically classify and detect plant diseases from leaf images.

The complete procedure consists of collecting the images used for training and validation, image pre-processing and augmentation and finally the procedure of training the deep CNN and fine-tuning. Different tests were performed in order to check the performance of newly created model.

CHAPTER 8

References

[1] Aakanksha Rastogi, Ritika Arora, Shanu Sharma, “Leaf Disease Detection and Grading using Computer Vision Technology &Fuzzy Logic,” presented at the 2nd International Conference on Signal Processing and Integrated Networks (SPIN), IEEE, 2015, pp. 500–505.

[2] Garima Tripathi, Jagruti Save, “AN IMAGE PROCESSING AND NEURAL NETWORK BASED APPROACH FOR DETECTION AND CLASSIFICATION OF PLANT LEAF DISEASES,” Int. J. Comput. Eng. Technol. IJCET, vol. 6, no. 4, pp. 14–20, Apr. 2015.

[3] S. Arivazhagan, R. Newlin Shebiah, S. Ananthi, S. Vishnu Varthini, “Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features,” Agric Eng Int CIGR J., vol. 15, no. 1, pp. 211–217, Mar. 2013.

[4] Prof. Sanjay B. Dhaygude, Mr.Nitin P. Kumbhar, “Agricultural plant Leaf Disease Detection Using Image Processing ” IJAREEIE, vol. 2(1),pp. 599-602, January 2013.

[5] K. Muthukannan, P. Latha, R. PonSelvi and P. Nisha, “CLASSIFICATION OF DISEASED PLANT LEAVES USING NEURAL NETWORK ALGORITHMS,” ARPN J. Eng. Appl. Sci., vol. 10, no. 4, pp. 1913–1918, Mar. 2015.

[6] K. A. Garrett, S. P. Dendy, E. E. Frank, M. N. Rouse, and S. E. Travers, “Climate change effects on plant disease: genomes to ecosystems,” Annual Review of Phytopathology, vol. 44, pp. 489–509, 2006.View at: Publisher Site | Google Scholar

[7] S. M. Coakley, H. Scherm, and S. Chakraborty, “Climate change and plant disease management,” Annual Review of Phytopathology, vol. 37, no. 1, pp. 399–426, 1999.View at: Publisher Site | Google Scholar

[8] S. Chakraborty, A. V. Tiedemann, and P. S. Teng, “Climate change: potential impact on plant diseases,” Environmental Pollution, vol. 108, no. 3, pp. 317–326, 2000.View at: Publisher Site | Google Scholar

[9] A. J. Tatem, D. J. Rogers, and S. I. Hay, “Global transport networks and infectious disease spread,” Advances in Parasitology, vol. 62, pp. 293–343, 2006.View at: Publisher Site | Google Scholar

[10] J. R. Rohr, T. R. Raffel, J. M. Romansic, H. McCallum, and P. J. Hudson, “Evaluating the links between climate, disease spread, and amphibian declines,” Proceedings of the National Academy of Sciences of the United States of America, vol. 105, no. 45, pp. 17436–17441, 2008.View at: Publisher Site | Google Scholar

[11] T. Van der Zwet, “Present worldwide distribution of fire blight,” in Proceedings of the 9th International Workshop on Fire Blight, vol. 590, Napier, New Zealand, October 2001.View at: Google Scholar

[12] S. A. Miller, F. D. Beed, and C. L. Harmon, “Plant disease diagnostic capabilities and networks,” Annual Review of Phytopathology, vol. 47, pp. 15–38, 2009.View at: Publisher Site | Google Scholar

[13] M. B. Riley, M. R. Williamson, and O. Maloy, “Plant disease diagnosis. The Plant Health Instructor,” 2002.View at: Google Scholar

[14] J. G. Arnal Barbedo, “Digital image processing techniques for detecting, quantifying and classifying plant diseases,” SpringerPlus, vol. 2, article 660, pp. 1–12, 2013.View at: Publisher Site | Google Scholar

[15] H. Cartwright, Ed., Artificial Neural Networks, Humana Press, 2015.