

THE FUSION AND COMPARISON OF IMAGE PROCESSING ALGORITHMS FOR  
PREDICTION OF LEAF DISEASE

A Thesis

by

HIMANSHU ANIL JOSHI

Submitted to the College of Graduate Studies  
Texas A&M University-Kingsville  
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2020

Major Subject: Computer Science

THE FUSION AND COMPARISON OF IMAGE PROCESSING ALGORITHMS FOR  
PREDICTION OF LEAF DISEASE

A Thesis

by

HIMANSHU JOSHI

Approved as to style and content by:

---

David Hicks, Ph.D.  
Chair of Committee

---

Ayush Goyal, Ph.D.  
Member of Committee

---

George Toscano, Ph.D.  
Member of Committee

---

Scott Smith, Ph.D.  
Chair of Department

---

George Allen Rasmussen, Ph.D.  
Vice President  
for Research and Graduate Studies

May 2020

## ABSTRACT

The Fusion and Comparison of Image Processing Algorithms for Prediction of Leaf Disease

May 2020

Himanshu Anil Joshi, B.E.

University of Pune

Chair of Advisory Committee: Dr. David Hicks

Leaf diseases are a significant threat to humans as consumers and breeders of the plants. Quality and productivity get affected by different types of diseases of the plant. Many of those are infectious and cause a total loss of crop yield. Many bacteria, fungi, and viruses damage a plethora of plants in the United States. More than sixty percent of U.S. crop losses are due to pathogens and fungi, adding up to an estimated expense of 137 billion dollars a year.

Timely and accurate detection of plant diseases is essential for crop quality and yield. Early detection can reduce plant disease; farmers rely on experts to identify most of the diseases. For a small farm measuring a few acres, the traditional method of a disease detection may be possible, but this method is not feasible and is costly for a large field. In many rural areas, farmers often lack precise knowledge or accessibility to an expert.

Using various image processing techniques such as the Convolutional Neural Network (CNN), the Support Vector Machine (SVM), and the k-Nearest Neighbor (k-NN) algorithm, plant diseases can be identified. A user-friendly and robust user interface is developed using web technologies. The fusion and comparison of different image processing algorithms result in a model that is used to predict leaf diseases. The optimal performance in this research is achieved by implementing the CNN+CNN fusion model.

This work's unique novel contribution is that it performs all of the machine learning with cloud computing on the AWS EC2 (Amazon Elastic Compute Cloud), whereas previous methods have only stored the data on the cloud; and, the proposed method of this work, fusion CNN + CNN has a 95% accuracy, higher than the other previous methods compared.

## ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Hicks, and my committee members, Dr. Goyal, and Dr. Toscano, for their guidance and support throughout this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University-Kingsville a great experience.

Finally, thanks to my family members for encouragement and my mother, father, and sister for their patience and love.

## TABLE OF CONTENTS

ABSTRACT.....	iii
ACKNOWLEDGEMENTS .....	v
TABLE OF CONTENTS.....	viii
LIST OF FIGURES .....	viii
LIST OF TABLES .....	viii
1 INTRODUCTION.....	1
2 LITERATURE SURVEY.....	3
3 METHODOLOGY .....	7
3.1 System Design.....	7
3.1.1 Unprocessed Data from Repository .....	7
3.1.2 Platform Module .....	8
3.1.3 Processing Module.....	8
3.1.4 Prediction Module.....	8
3.1.5 Interface Module.....	11
3.2 System Process.....	12
4 IMPLEMENTATION .....	13
4.1 Database Module.....	13
4.2 Cloud Implementation.....	13
4.2.1 AWS EC2.....	14
4.2.2 Putty .....	15
4.2.3 FileZilla.....	16
4.3 Dataset.....	17
4.4 Predictor Models .....	18
4.4.1 CNN .....	18

4.4.2	kNN.....	19
4.4.3	SVM.....	20
4.4.4	Fusion Model .....	20
4.5	Web Interface .....	22
4.5.1	Index page .....	22
4.5.2	Login page .....	23
4.5.3	Register .....	24
4.5.4	Connectivity .....	24
4.5.5	User Dashboard.....	24
5	RESULTS AND DISCUSSION.....	26
5.1	CNN Result .....	26
5.2	kNN Result.....	29
5.3	SVM Result .....	30
5.4	Fusion Model Results.....	31
5.4.1	CNN + CNN Fusion Model .....	31
5.4.2	CNN + SVM Fusion Model.....	32
5.5	Confusion Matrix .....	33
5.5.1	Precision.....	35
5.5.2	Recall .....	35
5.5.3	F1 Score .....	36
6	CONCLUSION .....	38
7	FUTURE WORK .....	39
	REFERENCES .....	41

## LIST OF FIGURES

FIGURE	Page
3.1 System Design .....	7
3.2 Support Vector Machine (SVM).....	9
3.3 General CNN .....	10
3.4 K - Nearest Neighbor .....	11
3.5 System Process.....	12
4.1 AWS Dashboard .....	14
4.2 Putty Dashboard.....	15
4.3 Putty SSH.....	16
4.4 FileZilla Dashboard .....	17
4.5 Dataset Directory Structure.....	17
4.4.1 CNN Model.....	18
4.4.2 CNN Compiler .....	19
4.4.3 KNN Model .....	19
4.4.4 SVM Model .....	20
4.4.5 CNN+CNN Fusion Model .....	21
4.4.6 CNN+SVM Fusion Model.....	21
4.5.1 Index Page.....	22
4.5.2 Login Page .....	23
4.5.3 Register Page .....	23



## LIST OF FIGURES

FIGURE	Page
4.5.4 User Dashboard.....	24
4.5.5 Result Page.....	25
5.1 CNN Result .....	27
5.2 Training Accuracy Vs Loss .....	28
5.3 Validation Accuracy Vs Loss .....	29
5.4 KNN Result.....	30
5.5 SVM Result.....	30
5.6 CNN+CNN Fusion Model Result.....	31
5.7 CNN+SVM Fusion Model Result.....	32
5.8 Confusion Matrix .....	33
5.9 Calculated Confusion matrix .....	34
5.9.1 Precision .....	35
5.9.2 Recall .....	36
5.9.3 F1 Score .....	37
8.1 Cat Vs Dog Dataset .....	40
8.2 Flower Dataset .....	40

## LIST OF TABLES

TABLE	Page
5.1 CNN Result.....	26
5.6 Accuracies.....	33

## 1 INTRODUCTION

Cultivation is an integral part of human survival. In densely populated countries like Bangladesh, Sri Lanka, Vietnam, or populous nations such as China and India, it is vital to improve the quality and fertility of vegetation. Quality indicates the nutritional values and fertility demonstrates the ability to produce; both are affected by different types of diseases within the plant. Many of those diseases are infectious and cause total loss of crops. Some diseases may live in the soil for a few years, which impacts the growing seasons, productivity, and fertility of the soil. Fungi, bacteria, and viruses cause most of the leaf diseases [1][2]. The leaves and stem of a plant are typically inspected for different visual symptoms to identify the leaf disease. Early detection and diagnosis can prevent the spread of the disease.

The existing universally accepted approach identifies the leaf disease by simple naked-eye inspection [3]. In this method, a team of experts are required to carry out continuous inspection and monitoring. For small farms, this approach may be possible, but this method is not feasible and is costly for large areas. Most of the time, farmers lack precise knowledge of the plant diseases, the handling techniques, and accessibility to the specialists. Expert advice or consulting is expensive and time-consuming. Leaf disease examination by sight is a tricky and challenging task as some diseases cannot be identified by sight.

In plants, some generic diseases include brownish and yellowish spots on leaves, scorch, and other fungal, viral, and bacterial epidemics [4]. These diseases can be observed and identified by automated systems; automatic detection is faster and more precise. It is possible to pinpoint the exact cause by examining a variety of tones of affected areas of the plant using an image processing technique. Experts can further analyze the result for corrective actions.

This approach of image processing brings significant value in the identification of leaf diseases since it provides the most reliable outcomes and decreases human efforts. The following proposed system for leaf disease detection is implemented on the cloud using different machine learning techniques. This system features a simple, robust, and user-friendly interface. Also, the system is secure, with access restricted to authorized users and experts. All the data is stored on an Amazon Web Services (AWS) relational cloud database. A dataset is used to train the model, which consists of more than 1000 images of leaves collected over the past several years. Different machine learning models are trained by using this dataset, which is further used for the prediction of leaf disease. This system is faster, easy to use, and inexpensive. Also, this system is highly scalable and accessible due to its cloud implementation. In this research, the performance of a CNN+CNN fusion model is optimal, and the same model is used to predict leaf disease.

The most relevant related research results are described in the literature survey. Chapter three in this report describes the methodology, which further provides an overview of the implemented cloud system as well as different machine learning algorithms. Throughout this research, the data is collected in many different scenarios. Those collected data are analyzed and discussed in the result section.

## 2 LITERATURE SURVEY

Recent research results have produced enhancements in technology for the examination and detection of plant and leaf disease. The following are some remarkable functional improvements that have made the systems for the detection of leaf disease less prone to errors.

The researcher Kunal Singh [1], provides a cloud-based platform accessible through mobile applications. The application allows users to upload photographs of various parts of a plant to diagnose the plant disease automatically [1]. Users can observe the disease frequency map for their community. A geolocation of the image, along with a timestamp, is used to show the appearance of the disease in a specific area. A Convolutional Neural Network (CNN) model is trained with an Amazon Web Services (AWS) S3 instance, which is used to store a dataset. This system is cost-efficient with a geolocation feature.

In the application discussed within [1], cloud techniques are used for highly concentrated work of preparing the neural network. An asynchronous deep CNN model is used by the classifier to analyze images and predict the disease. Whenever a new image of the leaf is added to the training dataset, it regularly regenerates the deep CNN model and classifies the data with more accurate predictions.

In [2], the identification of leaf diseases captured by a digital camera is made using a genetic algorithm. Various useful features were extracted by applying different image processing techniques for the analyses. First, an image acquisition module is implemented to extract the feature, followed by the processing of an input image. In the next step, a genetic algorithm is applied to obtain useful segments to classify leaf diseases. An evolutionary algorithm is similar to, a genetic algorithm, which produces solutions to the problems of optimization. The collection of

clarifications is called a population, with which the algorithm starts. Solutions are taken from one population, and is used to create a new population [2].

The system proposed in [3], applied soft computing techniques to identify leaf disease in pepper plants. Image acquisition is the first module used to capture the images in which pictures of leaves and stems are obtained from an equivalent distance [3]. All RGB pictures were transformed into HSV color scope to avoid the model's limitations, which is based on a Cartesian Coordinate operation. The segmented output image shows a disease-affected portion of the leaves. It then calculates, the total affected area by summing the number of pixels in the picture layout. The object analysis was conducted using MATLAB R2012A. The test was performed with a set of twenty leaves.

In [4], plant disease detection was achieved by a Pulse Coupled Neural Network with the Shuffle Frog Leaf Algorithm. For a fitness function, a weighted total of the cross-entropy and compactness of image segmentation was selected. It is used as a parameter to a shuffled frog leaf algorithm to increase its performance. The cross-entropy was utilized to estimate the diversity of information between probability patterns [4]. Further, it used to define the variation between the original image and the segmented image. Image segmentation compactness is utilized to estimate the integrity of image segmentation. It also ensures the integrity of the picture and helps to avoid fragmentation.

Learning Vector Quantization (LVQ), introduced by Kohonen, is a type of neural network that links the competitive learning with supervised learning [5]. It is a robust and heuristic computation for taking care of order issues. Because of its straightforward topology and versatile model, LVQ has remained broadly utilized in numerous applications. It characterizes the provided information in a determined quantity of classes. It comprises three layers, including an input, a

Kohonen, and an output layer [5]. Input layer neurons gather the estimation of information factors. The input and Kohonen layers are completely associated, while the Kohonen and output layers are in part associated [5]. The detailed results are transferred to the linear output layer. Classification is only limited to the tomato plants that are classified into four classes. The researcher uses 500 sizes of 512\*512 images of the tomato plant as a dataset, in which 400 were used for training purposes. In this proposed system, CNN is used for training the model and LVQ for the classification. Researchers in [5], use CNN and LVQ algorithm for the identification and distribution of the plant disease.

In [6], research is done on the identification of tomato plant diseases by utilizing a decision tree. The study is done by using 383 tomato plant images captured using a regular digital camera. The Otsu's segmentation is implemented on the dataset. The pictures were randomly accepted for training and experimentation. In this study, a supervised learning procedure is utilized for the training, and further distinguished tomato plant leaves into six groups: healthy along with five infected groups due to fungal, bacterial, and viral diseases [6].

The above papers implemented and used several different algorithms and techniques to detect the leaf diseases from a dataset. However, none of the systems are entirely implemented on the cloud. Also, most of the previous work has been implemented on a single machine-learning algorithm. The approach presented in this thesis is designed as a cloud-based system and utilizes multiple machine learning algorithms.

Using various image processing techniques such as the Convolutional Neural Network (CNN), the Support Vector Machine (SVM), and the k-Nearest Neighbor (k-NN) algorithm, plant diseases are identified. Models based on these different algorithms are tested for better-enhanced accuracy and to reduce loss. For training purposes, a dataset is used which contains multiple

different images of several plant leaves. By comparing and combining, different models were analyzed carefully to improve accuracy. The presented model is designed to self-evaluate by training itself after fixed intervals. This system predicts leaf diseases with high precision, and the result is displayed on the web interface.



### 3 METHODOLOGY

This section starts with a description of the system design. This is followed by a description of the system process in which five different processing modules and the significance of every process are explained in detail to understand the working flow of the system.

#### 3.1 System Design

System design explains five different working modules of the implemented system in detail. Figure 3.1 illustrates the different processing modules of the implemented system. The architecture of the system is defined by five different modules explained below.

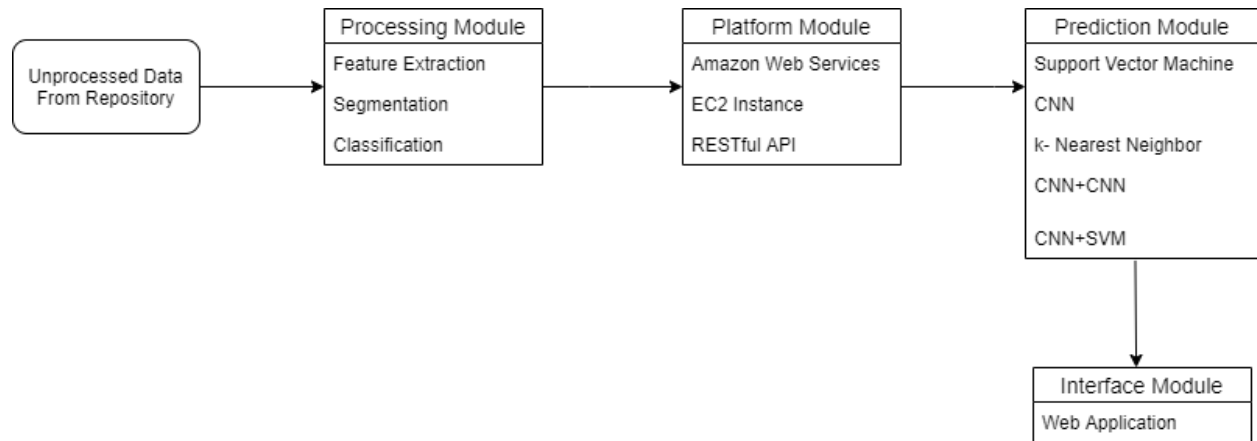


Figure 3.1 System Design

##### 3.1.1 Unprocessed Data from Repository

This system uses a leaf disease dataset, which contains the images of various leaves collected and maintained over the past several years. The following dataset is divided into two sets: test data and training data. Each set contains images of leaf diseases further divided into three types. Those images from the dataset are colored (RGB) with different resolutions. The raw images from the

dataset may be incomplete or inconsistent and may also contain noise. These issues can be resolved by using image processing techniques further explained in the processing module.

### 3.1.2 Platform Module

This module is used to store large datasets into the cloud-based storage. Relational Database Service (RDS) of Amazon Web Services (AWS) is a solution to store data in a structured format. This stored data is further accessed by database engines such as MySQL. AWS creates multiple instances of a database for security and scalability purposes in different applications. Further, this dataset is accessed by the processing model to apply different functions for better performance.

### 3.1.3 Processing Module

Images from the dataset may be noisy, incomplete, and may also contain errors that directly affect the accuracy of the prediction of an input image. For better performance and accuracy, the image is required to be processed by using different filters and techniques. Dataset has images of different sizes and colors. It is required to provide images with the same resolutions to train the model for better accuracy and reduce the loss. This can be achieved by applying a rescale, rotation, zoom, or horizontal/vertical flip function on the training and validation dataset.

### 3.1.4 Prediction Module

Different image processing algorithms are used to create prediction models. In this System, the Support Vector Machine (SVM), the k-Nearest Neighbor (kNN), and the Convolutional Neural Network are used. The performance of the model is determined by accuracy and fault on test data. These different models were analyzed carefully to improve accuracy. These three models are further explained below.

#### 3.1.4.1 Support Vector Machine

Figure 3.2 illustrates the general architecture of a Support Vector Machine. Support Vector Machines are supervised learning classifiers with associated learning algorithms that used to analyze the data used for classification and regression analysis.

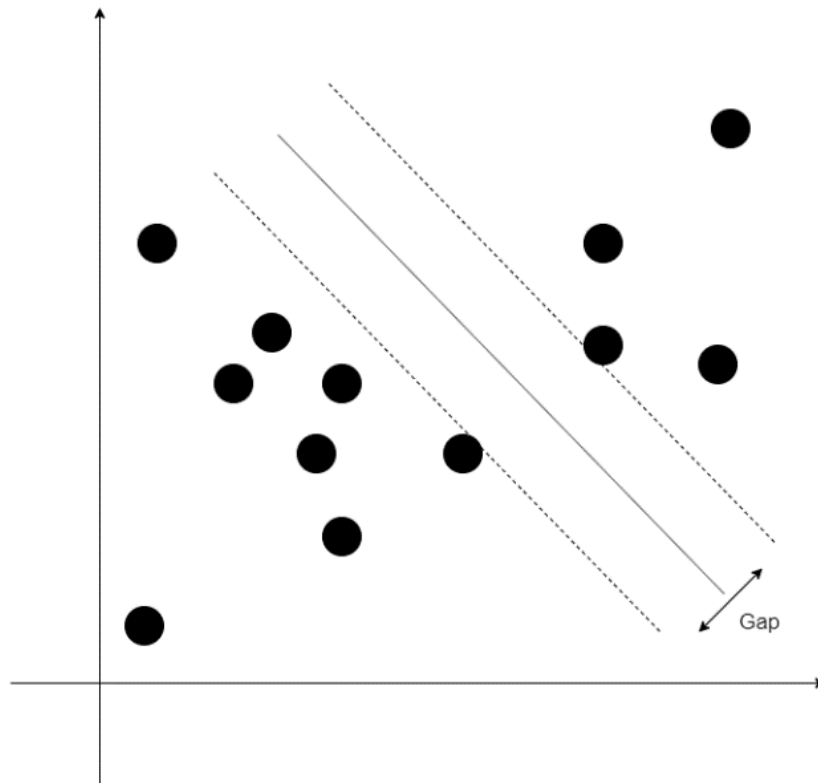


Figure 3.2 Support Vector Machine (SVM)

Support Vector Machines are linear hyperplanes that classify data into various subsets. The observations on edge and within the soft margin are known as support vectors. When the data is in two dimensions, the support vector is a line. Three-dimensional support vectors form a plane.

The final model of the SVM is trained by using ‘Linear’ and ‘Radial Basis Function’ kernels as accuracies collected from both kernels are compared. Also, in a combined model of CNN and SVM linear kernel is used. Both SVM models classify images into three types of classes.

### 3.1.4.2 Convolutional Neural Network

Figure 3.3 illustrates the general architecture of the Convolutional Neural Network. The very first and last layer is known as the input and output layer. The middle layer is known as the convolution and max pooling.

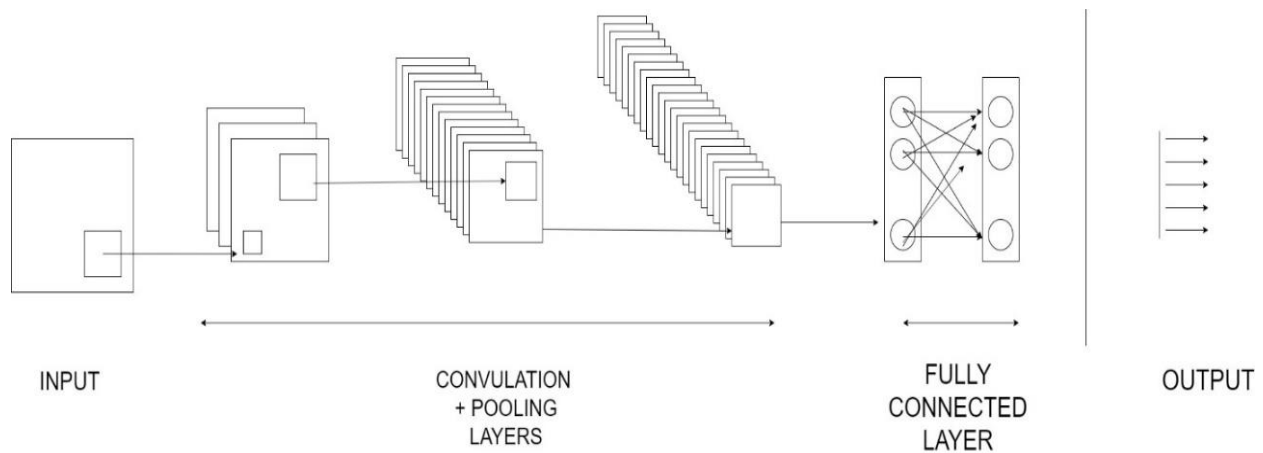


Figure 3.3 General CNN Architecture (CNN)

A Convolutional Neural Network is implemented in the system. The images from the training dataset are used in the input layer. All layers in the neural network are densely connected. Convolution and pooling techniques are applied to the hidden layer. The final output is based on the three classes. These three classes correspond to the diseases that are based on the used dataset.

### 3.1.4.3 k-Nearest Neighbor

k-NN is a classification algorithm that checks for the most appropriate match in the test data for the expected scope. k-NN requires three things: saved record from sets, distance matrix of a training data, and the value of k. Distance is calculated between the training records. The idea is to search for the ideal match of the test data in the given space.

K is a class in the dataset. In this research work, the value of k is three because three different classes are present in the dataset. They are classified as Bacterial Leaf Blight, Brown Spot, and Leaf Smut. The Euclidean distance formula is used to estimate the distance within two data points. Figure 3.4 describes the general architecture of the k-Nearest Neighbor.

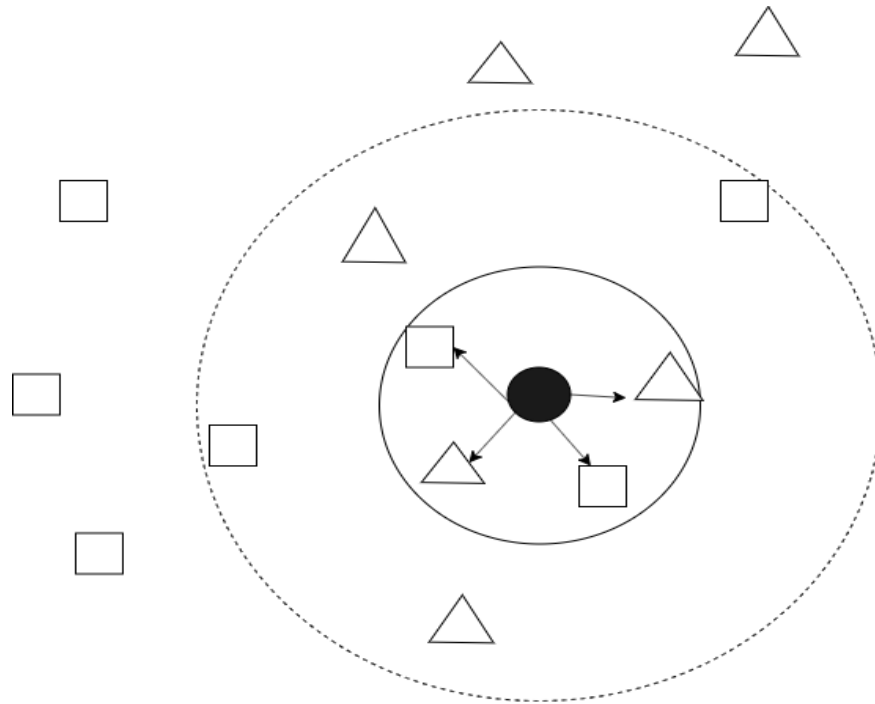


Figure 3.4 k-Nearest Neighbor

### 3.1.5 Interface Module

The prediction system consists of a robust, user-friendly interface with secure and authorized access to the users and experts developed in the Flask framework. Users can upload images of a leaf to the cloud-based machine learning algorithm. Then this system is responsible for predicting the disease using a pre-trained model stored on a cloud database. The prediction results are displayed back on the user interface, which is accessible to the users as well as experts in this field.

## 3.2 System Process

Figure 3.5 illustrates the different components of the implemented system. For a better understanding, the recommended system is divided into five distinct processes.

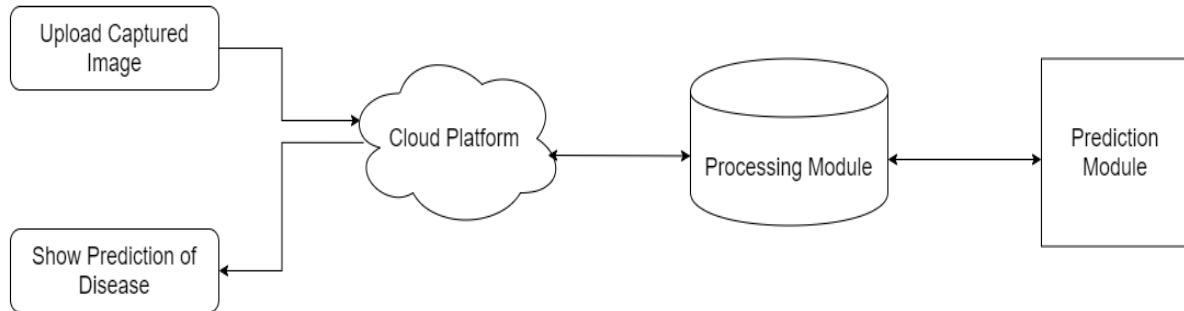


Figure 3.5 System Process

The first process allows users to upload an image using a web application. This web application is developed by using a Flask web interface and a Python programming language. This application has a robust, secure, and user-friendly interface.

In the second process, the uploaded image is stored in cloud-based storage. An Amazon RDS is used to store and retrieve data. During the third process, the uploaded image is provided to the processing module. The processing module applies filters to improve the standard of the image and resize it to the desired format.

After the third process, the improved image is transferred to pre-trained image processing models to calculate accuracy. Different image processing algorithms are used to calculate the accuracy of the image. After prediction, the model is retrained with new data after a fixed interval. The last process is responsible for displaying the prediction result back on the result page.

## 4 IMPLEMENTATION

The implementation of the system is organized into five modules: database module, cloud implementation, dataset, predictor model, and web interface. The implementation steps offer a clear picture of the research objective and the particular method required to achieve the research objective.

### 4.1 Database Module

Amazon RDS is used to store the data. A database called User is designed using a MySQL engine. A table called User\_data is created in the User database, which contains the following attributes:

- Username (Varchar 250): The username attribute is type varchar of length 250. This attribute is used to store a unique username of a particular user.
- User ID (Email): The user-ID attribute stores the email of each user from the User database. This email is used for authentication purposes and to set the session.
- Password (Password): This attribute is used to store the password of a user in an encrypted format.
- Phone Number (Int): The ten-digit phone number is used to validate the user for first time registration.
- Images (Blob): All upload images by the user are saved in this attribute. Users can upload multiple images and save them for future reference.

### 4.2 Cloud Implementation

An Amazon EC2 instance is used along with Putty and FileZilla to transfer files to the AWS database. All three different technologies used in this system are explained below.

## 4.2.1 AWS EC2

An Amazon EC2, known as Amazon Elastic Compute Cloud. AWS EC2 is a web-based cloud service that offers reliable, re-sizable cloud infrastructure capability. This cloud service is designed to make web-based cloud computing easier for developers. An Amazon EC2 helps developers to acquire and customize the minimal power required and to provide a simple web-based interface. This interface gives full control of your resources and allows the user to run on the verified computing infrastructure of Amazon. Figure 4.1 illustrates the dashboard of an AWS.

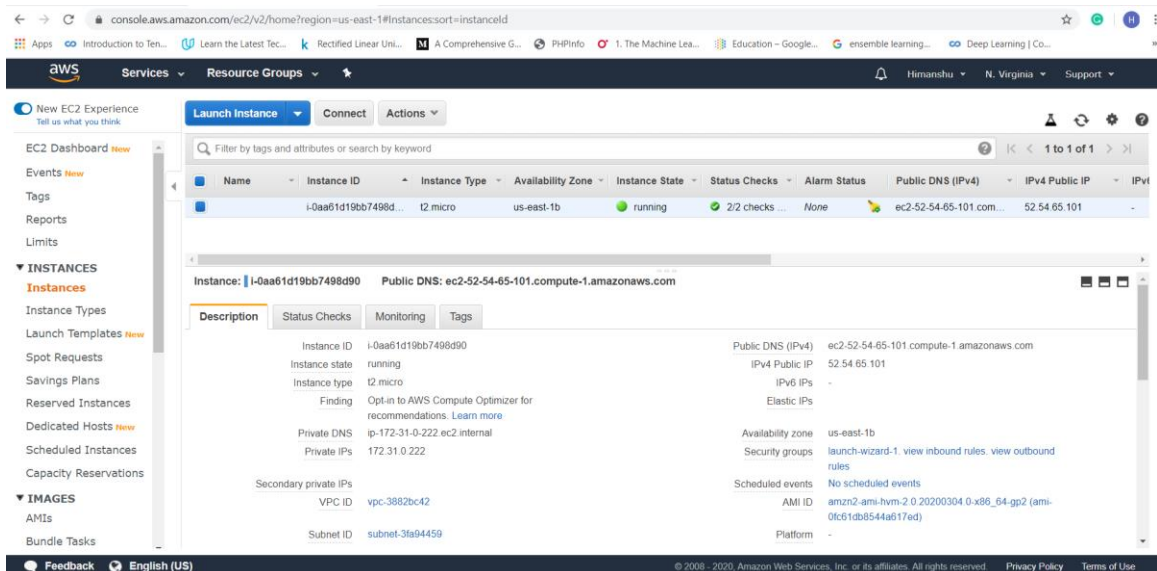


Figure 4.1 AWS Dashboard

Amazon EC2 provides multiple features. Some of the relevant ones are highlighted below.

- AWS determines if the users want to run the instance in multiple locations, to use static IP endpoints, or add persistent storage of blocks to the instances.
- Amazon EC2 pre-configured templates also called Amazon Machine Images (AMIs). This template includes the operating system (OS) as well as an additional software. Users can select the required OS and packages based on a particular use.



- Multiple memory configurations, storage, and networking capacity options are available for the selected instance.

#### 4.2.2 Putty

Amazon Linux does not provide a GUI to the user. Putty is working as a Secure Service Client (SSH) between the user and Amazon Linux. After the successful installation of putty, Secure Service Client is configured to initiate an EC2 Instance. Configured SSH opens the EC2 command line, which is illustrated in Figures 4.2 and 4.3 below.

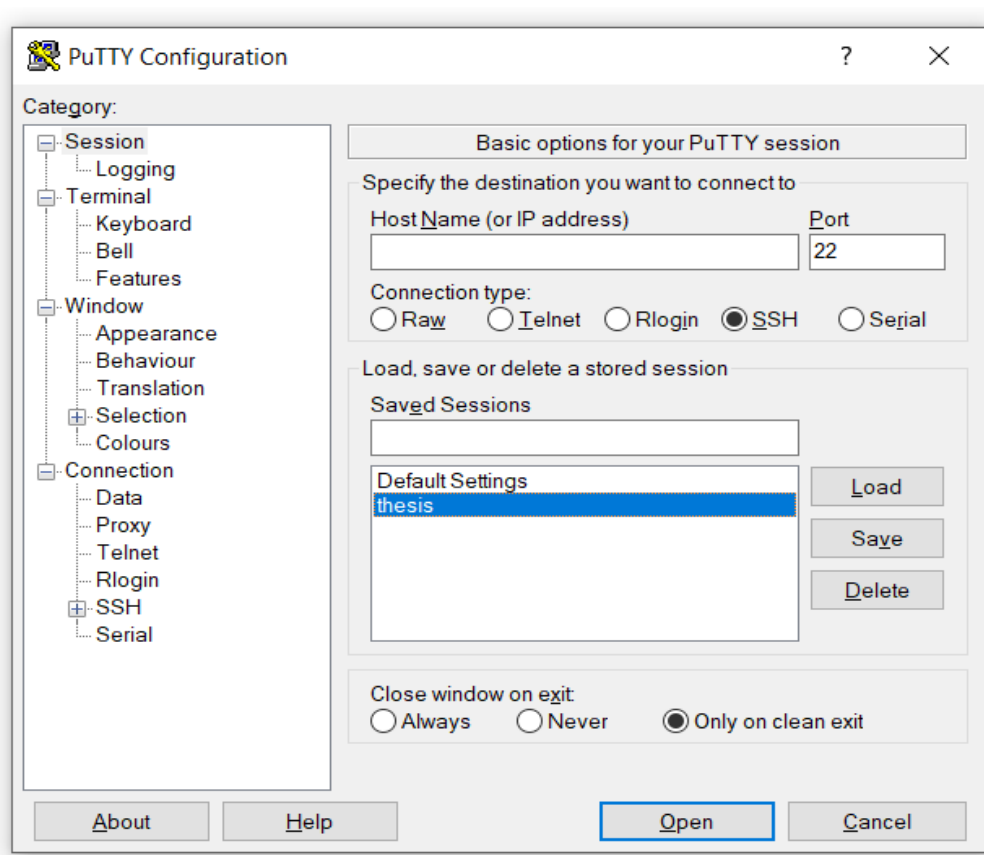
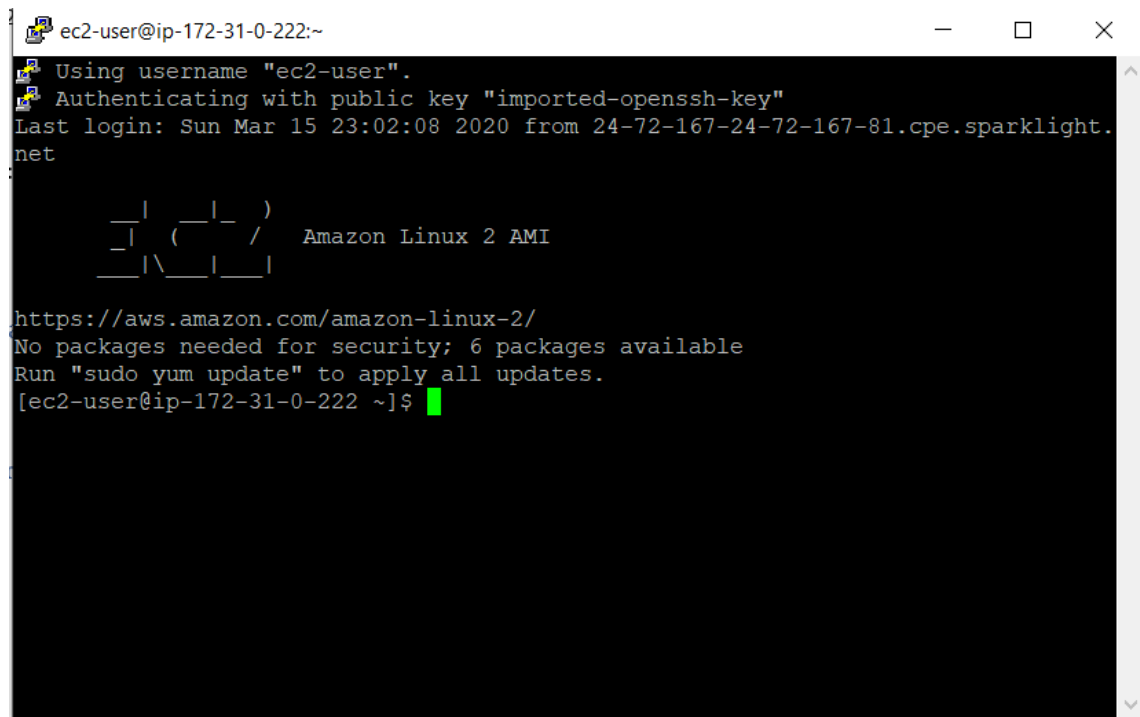


Figure 4.2 Putty Dashboard



```
ec2-user@ip-172-31-0-222:~  
Using username "ec2-user".  
Authenticating with public key "imported-openssh-key"  
Last login: Sun Mar 15 23:02:08 2020 from 24-72-167-24-72-167-81.cpe.sparklight.net  
  
  _ |  _ |  _ )  
  _ | (  _ | /  Amazon Linux 2 AMI  
  _ | \ _ | _ |  
  
https://aws.amazon.com/amazon-linux-2/  
No packages needed for security; 6 packages available  
Run "sudo yum update" to apply all updates.  
[ec2-user@ip-172-31-0-222 ~]$
```

Figure 4.3 Putty SSH

#### 4.2.3 FileZilla

All the required files and saved models are stored on AWS storage using the FTP client FileZilla. FileZilla provides a user interface in which a user can simply drag the required files and drop to the AWS storage. FileZilla connects to AWS using the PPK file, which is a key used to authenticate an AWS. This private key is generated while creating an EC2 instance. The private keys are unique for each instance. Figure 4.4 illustrates the FileZilla dashboard.

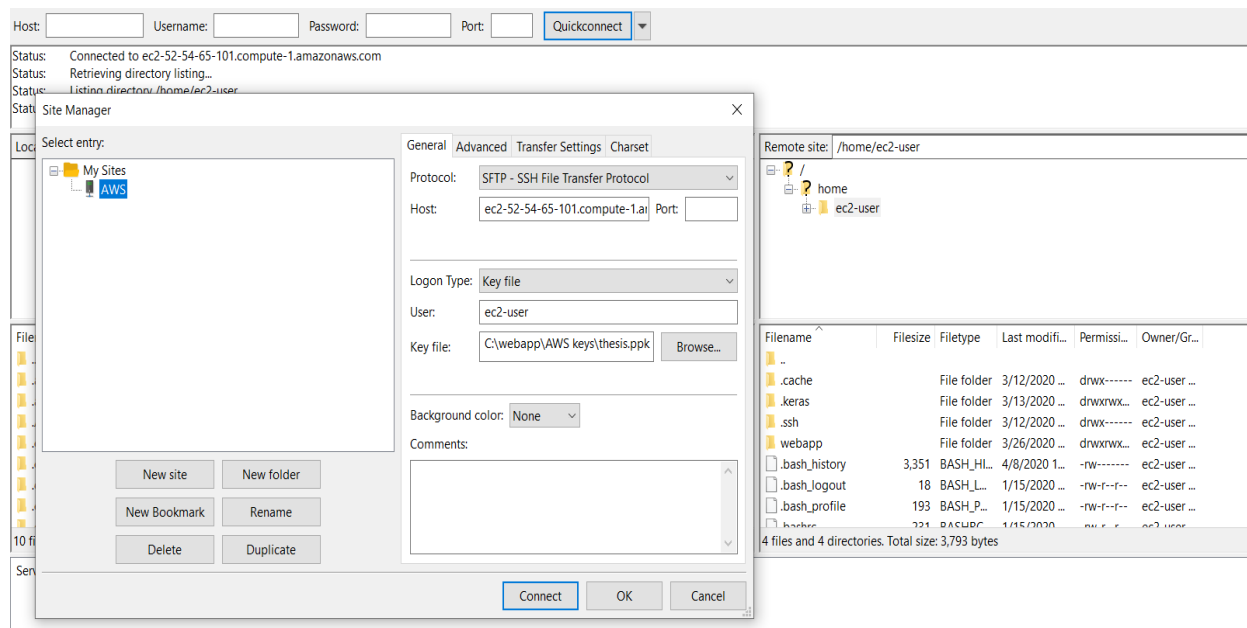


Figure 4.4 FileZilla Dashboard

### 4.3 Dataset

Figure 4.5 illustrates the structure of the dataset. The primary dataset is stored on Google Drive. Users can access datasets from anywhere by logging on to Google Drive. The top-level directory structure of the dataset is as follows.

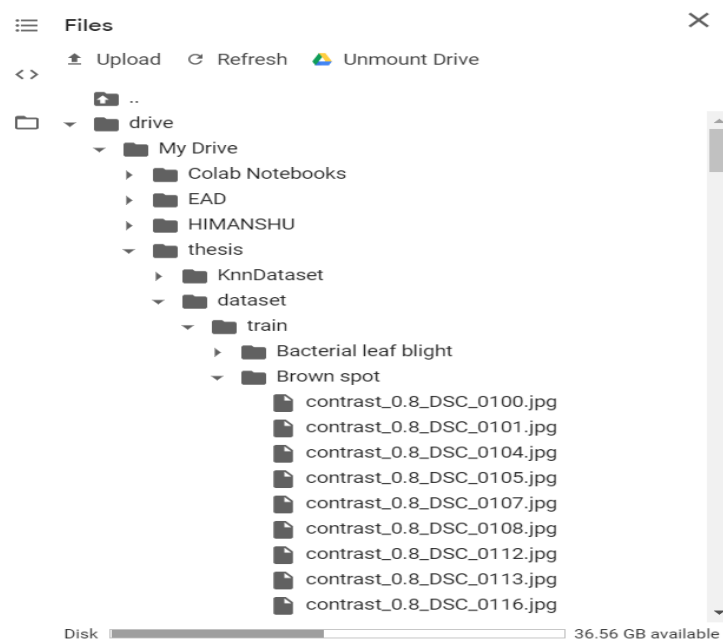


Figure 4.5 Dataset directory structure

## 4.4 Predictor Models

Predictor models are the different models implemented using different image processing algorithms. In this research work, a CNN, an SVM, a kNN, along with a fusion of two CNNs and a fusion of a CNN and an SVM are examined. Each implemented model is explained below.

### 4.4.1 CNN

Figure 4.4.1 illustrates the code segment required to establish the structure of the CNN model used in this system. Three convolutions and three max-pooling layers of different sizes are implemented. The final output layer has three classes for three different types of diseases. The activation function SoftMax is used to display the probability distributions.

```
model = Sequential()

model.add(Conv2D(16, 3, padding='same', activation='relu', input_shape=(IMG_SHAPE, IMG_SHAPE, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, 3, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, 3, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))

model.add(Dropout(0.2))
model.add(Dense(3, activation='softmax'))
```

Figure 4.4.1 CNN Model

The model is compiled using an adam optimizer and a sparse\_categorical\_crossentropy as loss function to calculate the loss in test and train dataset. After the compilation, this model is used to predict the output class. The code segment required to compile the CNN model is illustrated in Figure 4.4.2

```
model.compile(optimizer='adam',  
loss='sparse_categorical_crossentropy',  
metrics=['accuracy'])
```

Figure 4.4.2 CNN Compiler

#### 4.4.2 kNN

The following figure 4.4.3 explains the code segment required to establish the kNN model in detail. In this model, the model is trained with multiple different values of k. The value of k is an odd integer in the range from 1 to 30. Accuracy of every k value is displayed to analyze the result.

```
kVals = range(1, 30, 2)  
  
for k in range(1, 30, 2):  
    knn = KNeighborsClassifier(n_neighbors = k)  
    knn.fit(X_train,y_train)  
  
    score = knn.score(X_test, y_test)  
    print("k=%d, accuracy=%.2f%%" % (k, score * 100))  
    accuracies.append(score)
```

Figure 4.4.3 KNN Model

#### 4.4.3 SVM

Figure 4.4.4 below illustrates the code segment required to establish the SVM model. In this SVM model, both linear and RBF kernels were used. A training image dataset of 70% images is used to train the model, and the remaining 30% is used to test the dataset.

```
param_grid = [  
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']},  
    {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']}]  
  
svc = svm.SVC()  
clf = GridSearchCV(svc, param_grid)  
clf.fit(X_train, y_train)
```

Figure 4.4.4 SVM Model

#### 4.4.4 Fusion Model

A combination of two models is implemented to make a fusion model. In this section, the two different fusion models created and explained below.

##### 4.4.4.1 CNN + CNN Fusion Model

The combination of two CNNs are implemented. After multiple convolutions and maximum pooling layers in the CNN, one fully connected layer is implemented, and the result of the final layer is provided as input to the second CNN model.

The final class prediction is based on the CNN classifier. In the second CNN model, the last layer is a dense layer in which the 'SoftMax' function is activated. The SoftMax function provides accuracy based on probability distribution. Figure 4.4.5 illustrates the graphical representation of the CNN+CNN fusion model.

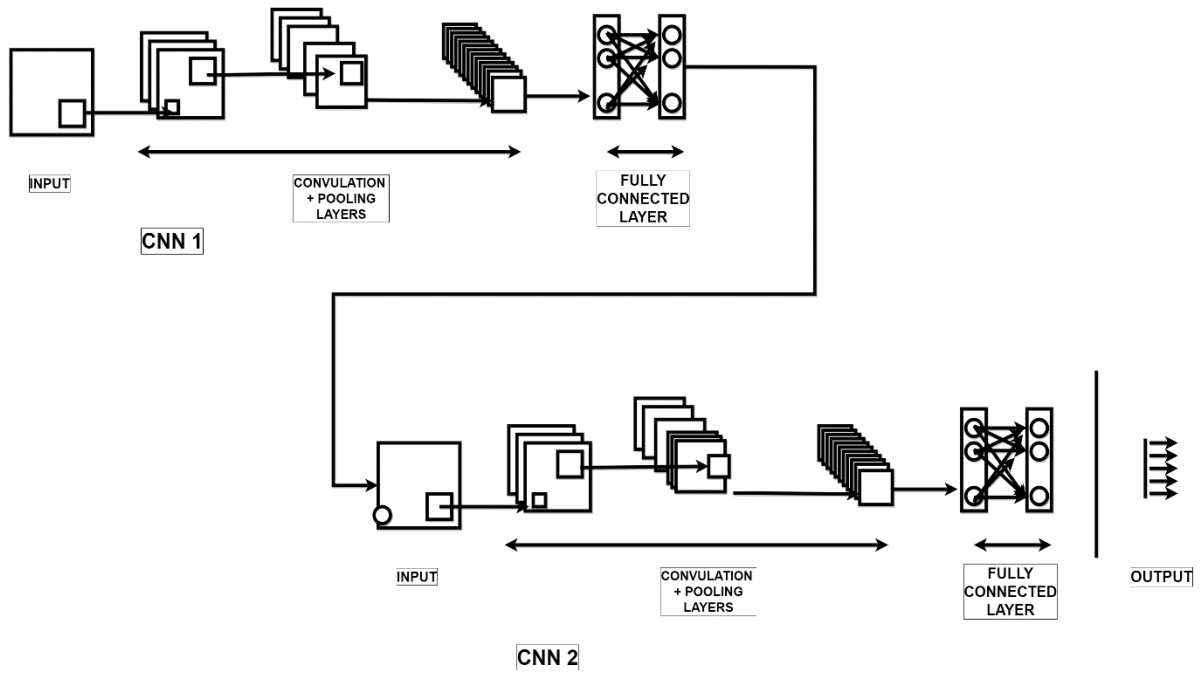


Figure 4.4.5 CNN Fusion Model

#### 4.4.4.2 CNN + SVM Fusion Model

The combination of a CNN and an SVM is implemented. After multiple convolutions and pooling layers in the CNN model, one fully connected layer is implemented, and the result of the final layer is provided as input to the SVM model, which is based on a 'Linear' kernel. The final class prediction is based on the SVM classifier. Figure 4.4.6 illustrates the graphical representation of the CNN+SVM fusion model.

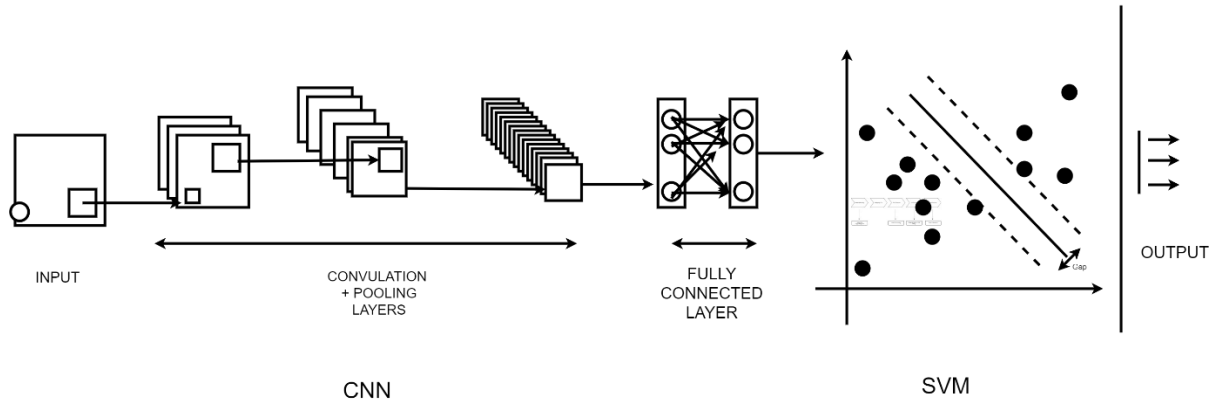


Figure 4.4.6 SVM Fusion Model

## 4.5 Web Interface

The web application was developed using the Flask framework and the Python programming language. All the template pages are developed in HTML5. These different webpages are explained below.

### 4.5.1 Index page

The simple home page is created using HTML5/CSS and the Flask interface. The home page contains four different tabs of user login, expert login, registration, and information. All required procedures are written in the app.py file, which is used to launch the web application. Figure 4.5.1 illustrates the index page.



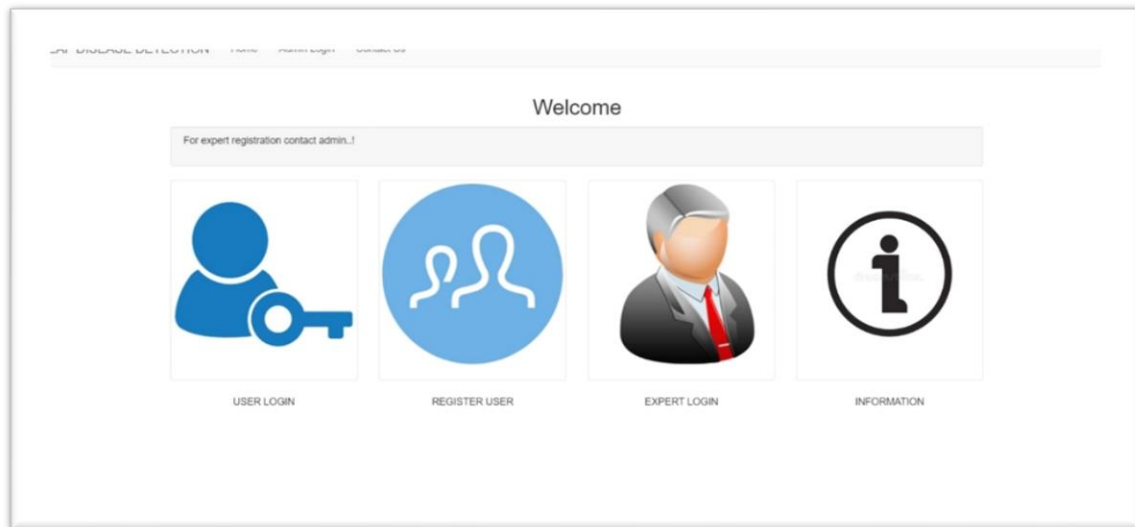


Figure 4.5.1 Index Page

#### 4.5.2 Login page

The simple login page is designed to authenticate the user. This simple form is based on HTML5. This page takes the username as an email and password from the user. Furthermore, these details are compared with values in the database and redirect it to the user dashboard. Figure 4.5.2 illustrates the login page.

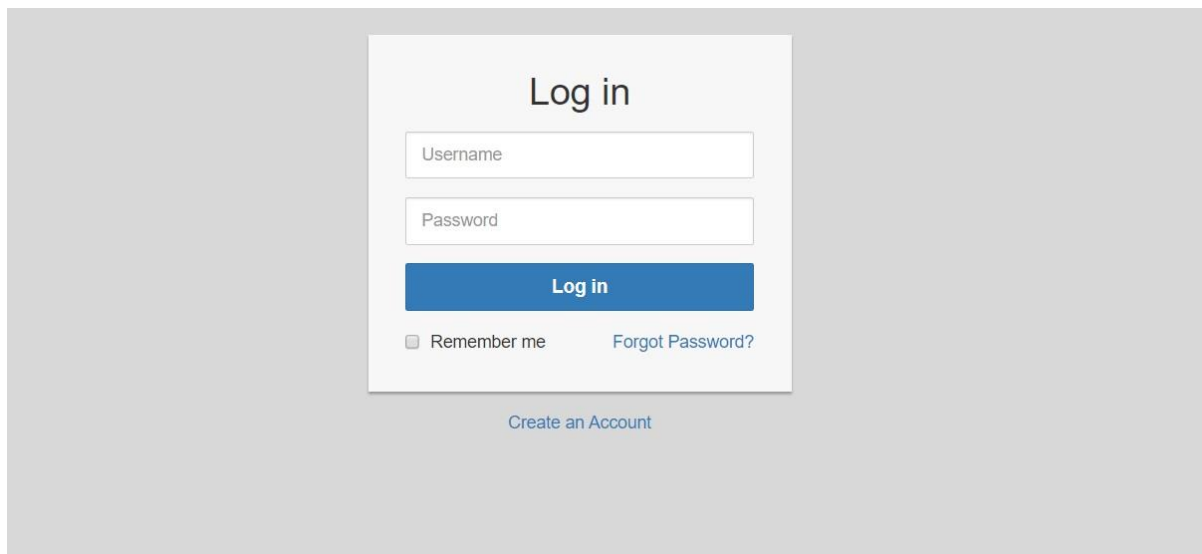


Figure 4.5.2 Login Page

### 4.5.3 Register

Figure 4.5.3 illustrates a simple HTML form designed along with a style sheet. All details are collected in the container and then sent to the database as an insert query. After successful validation, the user is redirected to the login page for authentication.

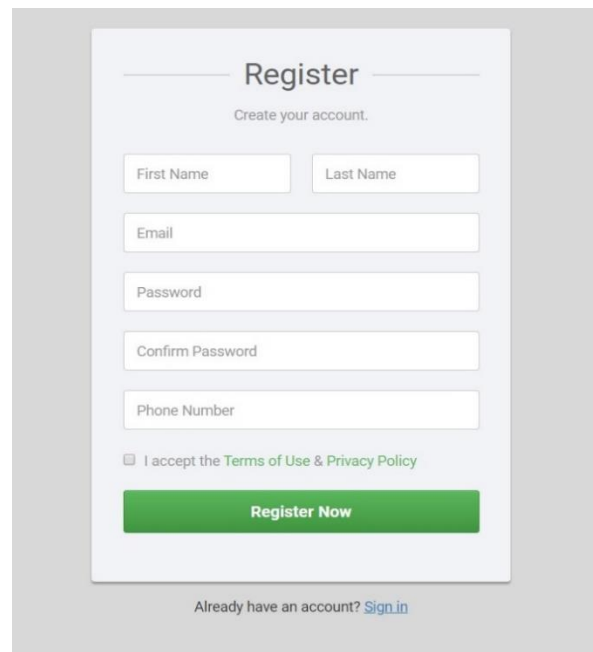


Figure 4.5.3 Register page

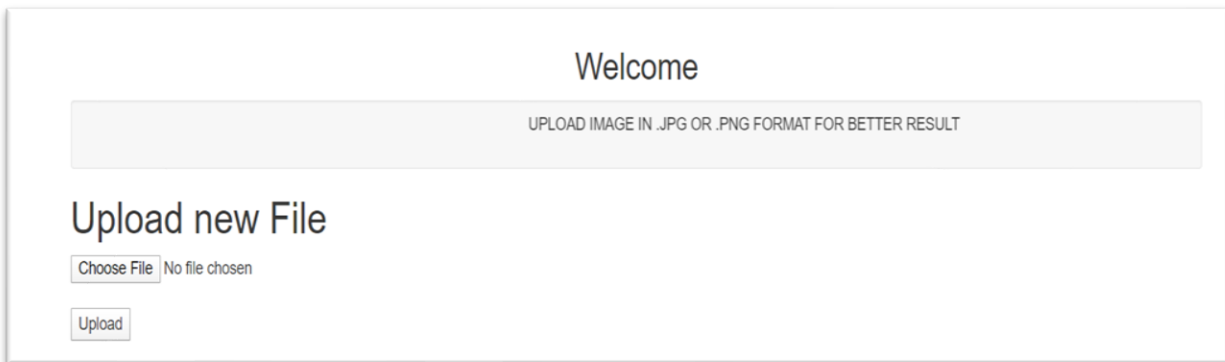
### 4.5.4 Connectivity

All the templates and web pages are connected using the Flask interface. App.py is used to launch the web application. This file is responsible for collecting, holding, and redirecting between different pages of the system.

### 4.5.5 User Dashboard

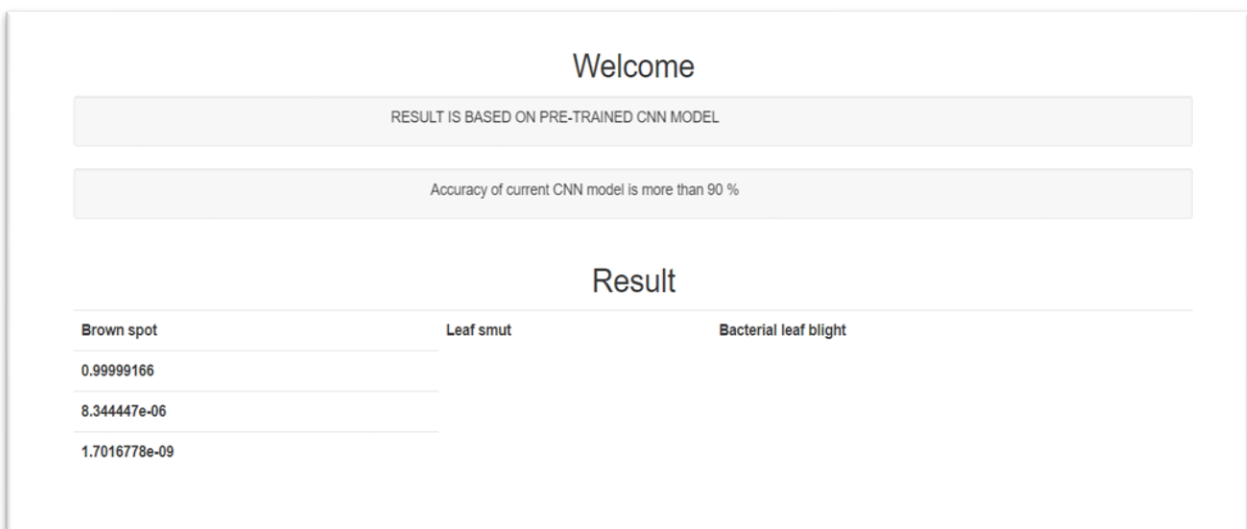
In the following figure 4.5.4, the user can upload an image using the upload tab. PNG and JPG formats are suggested for better accuracy. Uploaded image is provided to the machine learning model to calculate accuracy, which is stored on the AWS storage. All results, including accuracy, are displayed back on the result page.

In this system, the calculated result is based on a pre-trained CNN+CNN fusion model with 100 epochs. The epoch is one forward and one backward propagation of input data. The current uploaded model has training accuracy of more than 95%. Also, the model shows the accuracy concerning all three different classes, which indicate three different diseases of the leaf.



The User Dashboard interface features a 'Welcome' header. Below it is a light gray box with the text 'UPLOAD IMAGE IN .JPG OR .PNG FORMAT FOR BETTER RESULT'. The main section is titled 'Upload new File' and contains a file upload area with a 'Choose File' button and the text 'No file chosen'. Below this is an 'Upload' button.

Figure 4.5.4 User Dashboard



The Result page interface features a 'Welcome' header. Below it are two light gray boxes: the first contains 'RESULT IS BASED ON PRE-TRAINED CNN MODEL' and the second contains 'Accuracy of current CNN model is more than 90 %'. The main section is titled 'Result' and contains a table with three columns: 'Brown spot', 'Leaf smut', and 'Bacterial leaf blight'. The 'Brown spot' column has three rows of values: '0.99999166', '8.344447e-06', and '1.7016778e-09'.

Brown spot	Leaf smut	Bacterial leaf blight
0.99999166		
8.344447e-06		
1.7016778e-09		

Figure 4.5.5 Result page

## 5 RESULTS AND DISCUSSION

This section discusses multiple aspects of the results based on data and graphs collected in the different scenarios. The result of each model is discussed in detail.

### 5.1 CNN Result

Table 5.1 below illustrates the accuracy and the loss of the training and validation data collected in different epochs. The lowest accuracy is observed at the 40<sup>th</sup> epoch, which is around 56%, and the highest accuracy achieved at the 120<sup>th</sup> epoch is around 92 %.

Table 5.1 CNN Result

<b>Epochs</b>	<b>Training Accuracy</b>	<b>Training Loss</b>	<b>Validation Accuracy</b>	<b>Validation Loss</b>
<b>20</b>	82 %	50 %	70 %	90 %
<b>40</b>	56 %	45 %	80 %	90 %
<b>100</b>	85 %	30 %	82 %	40 %
<b>150</b>	92 %	25 %	90 %	26 %

Figure 5.1 compares the training and validation accuracies with the loss in different epochs. By analyzing the following graphs, we can find if the model is overfitting or underfitting of the input data.

In this research, the model is trained with 20, 40, 100,150 epochs as the model is under fitted for 20 epochs and overfitted for 150 epochs. It is possible to identify the overfitting and the underfitting by noticing a sudden change or increase and decrease in the graph at a certain point. The model is trained for 120 epochs because, after 120 epochs, the model starts to overfit.

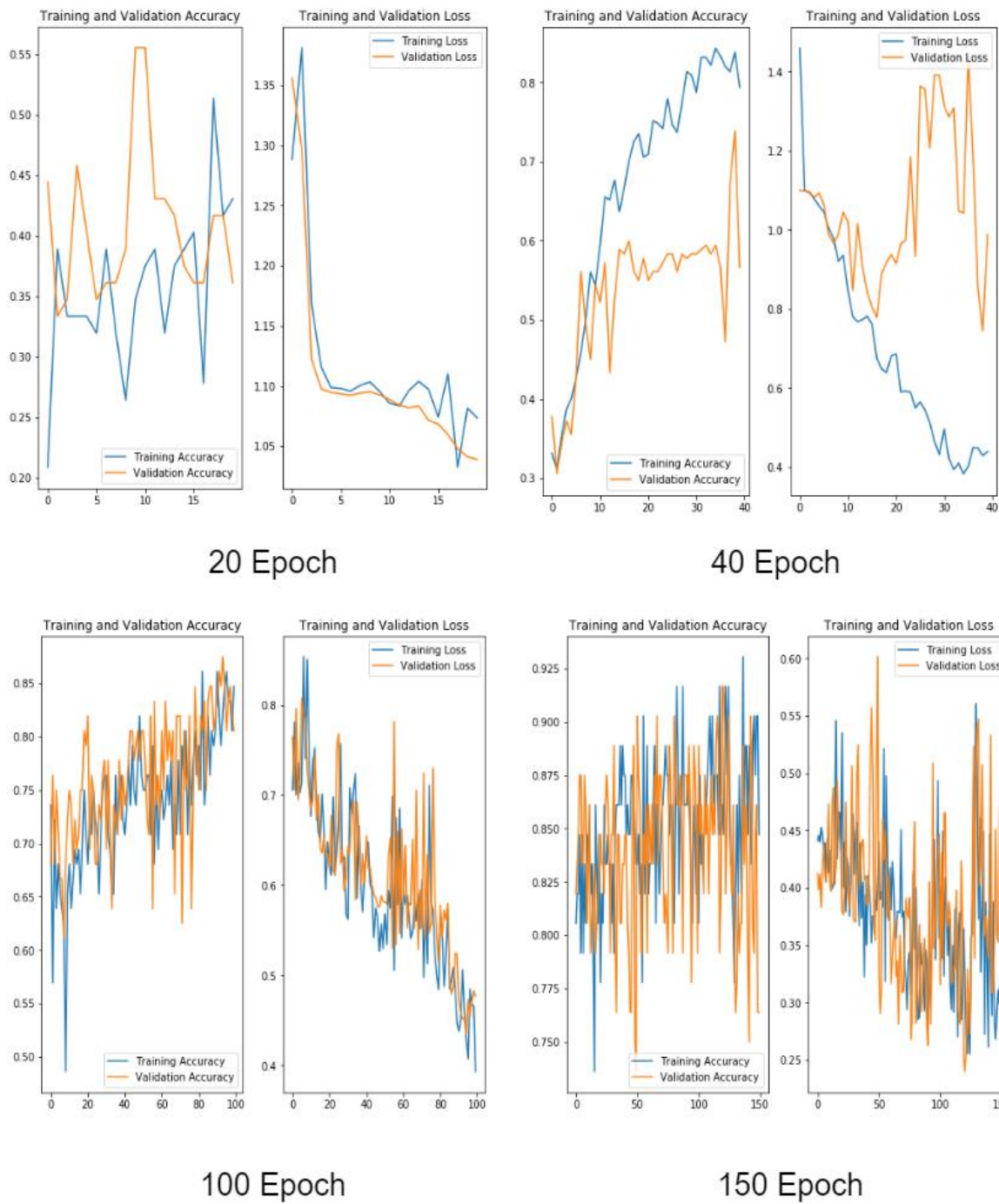


Figure 5.1 CNN Result

In Figure 5.2, the result of training accuracy and the loss are presented in the graphical format. The training accuracy of 82% and the highest loss of 50% is achieved after the 20<sup>th</sup> epoch. The model is under-fitted for the 20<sup>th</sup> epoch. After 40 epochs, accuracy is decreased to 56 %. More than 86% of the accuracy is recorded, and a loss of 30% during the 100<sup>th</sup> epoch. The highest accuracy of 92% is achieved between 100 and 150 epochs, with a minimum loss of 25%.

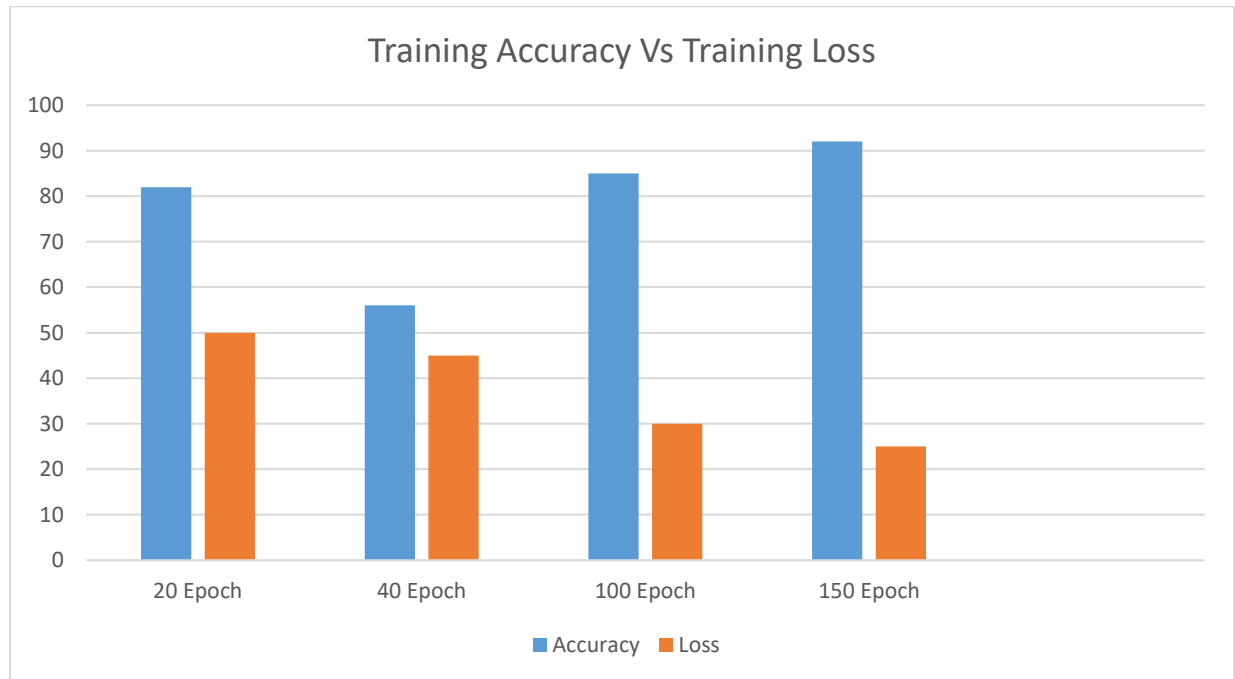


Figure 5.2 Training Accuracy Vs. Loss

Figure 5.3 illustrates the result of a validation accuracy and the loss in the graphical format. A validation accuracy of 70% is achieved after the 20<sup>th</sup> epoch. More than 82% of the accuracy is recorded and a loss of 40% during the 100<sup>th</sup> epoch. The highest validation accuracy of 90% and the lowest loss of 26 % is achieved between 100 and 150 epochs. Based on this result, the final model is trained for 125 epochs.

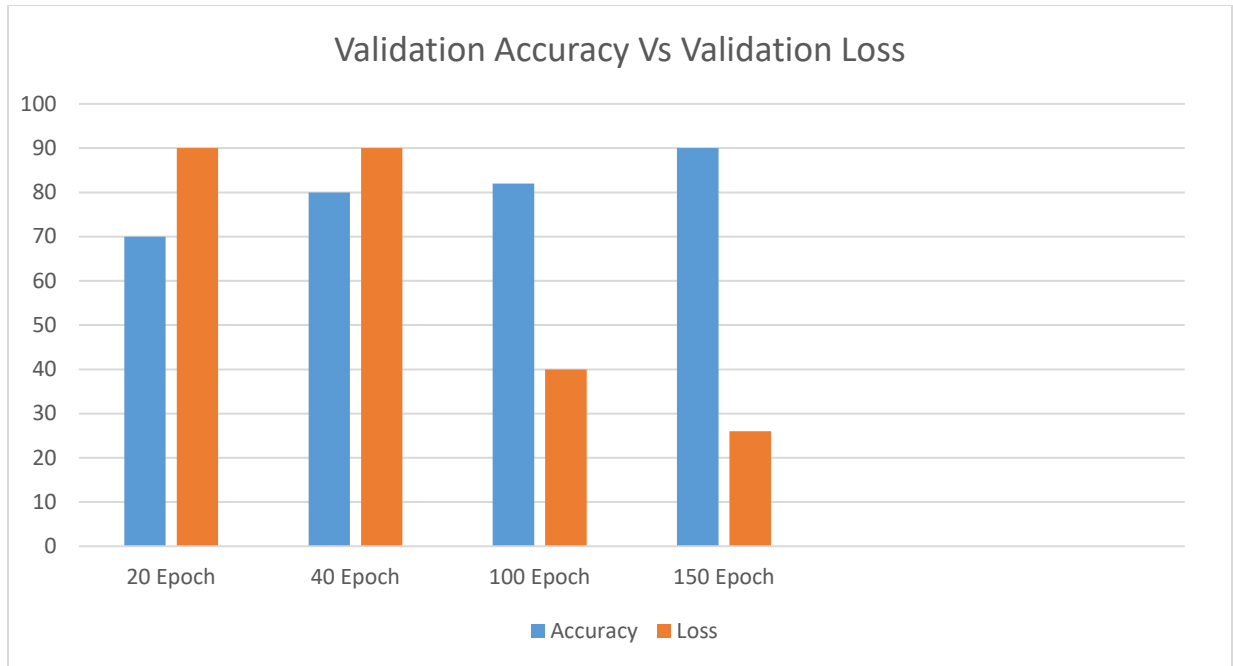


Figure 5.3 Validation Accuracy Vs. Loss

## 5.2 kNN Result

Figure 5.4 illustrates the result of the kNN model in detail. The model is trained with multiple different values of  $k$ . The value of  $k$  is an odd integer in the range from 1 to 30. The accuracy of every  $k$  value is displayed to analyze the result.

Highest achieved accuracy is approximately 70% because this kNN model is not supported for the unsupervised dataset. The aim is to achieve the highest accuracy; therefore, the model is trained with multiple values of  $k$ .

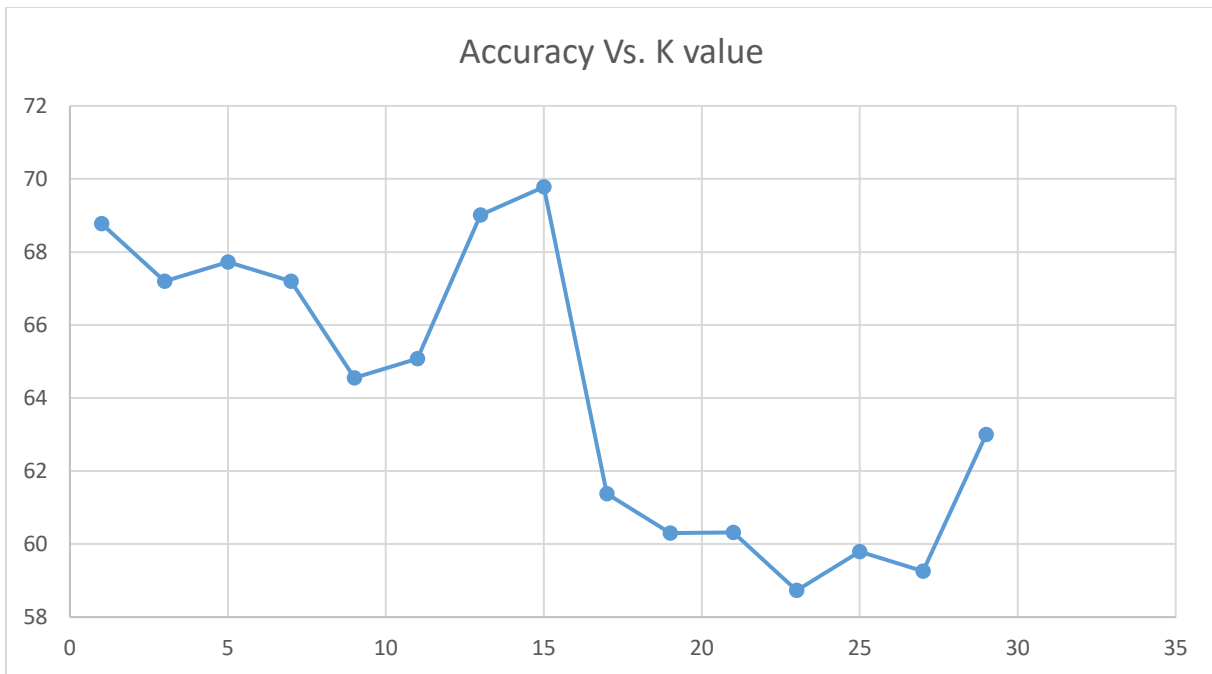


Figure 5.4 KNN Result

### 5.3 SVM Result

Figure 5.5 illustrates the code segment for a prediction of an SVM model on the testing dataset. This SVM model is trained using linear as well as a Radial Basis Function kernel. An approximately 82 % accuracy is achieved on training data in the SVM model. Although accuracy on the validation data is approximately 79%, which is comparatively lower than the CNN model.

```
y_pred = clf.predict(X_test)
print(y_pred)
print()
[2 0 1 1 1 0 1 2 2 0 2 2 1 0 2 2 2 2 1 1 0 1 1 0 1 1 0 2 1 2 2 1]
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
Accuracy: 0.8517575757575758
```

Figure 5.5 SVM Result



## 5.4 Fusion Model Result

In this section, the results of the fusion models are discussed. The results of both models are collected by training each model for 100 epochs. The CNN + CNN fusion model gives the highest accuracy of 95% in just 100 epochs, which is the highest achieved accuracy in this research work.

### 5.4.1 CNN + CNN Fusion Model

Figure 5.6 illustrates the result of the training versus validation accuracy with the loss in the graphical format. After the 100<sup>th</sup> epoch, the training accuracy of more than 95% is achieved, and the training loss is recorded lowest of 1.8%. The highest accuracy is achieved in this model. This model is used in the implemented cloud-based system, as this provided the highest accuracy with a minimum loss.

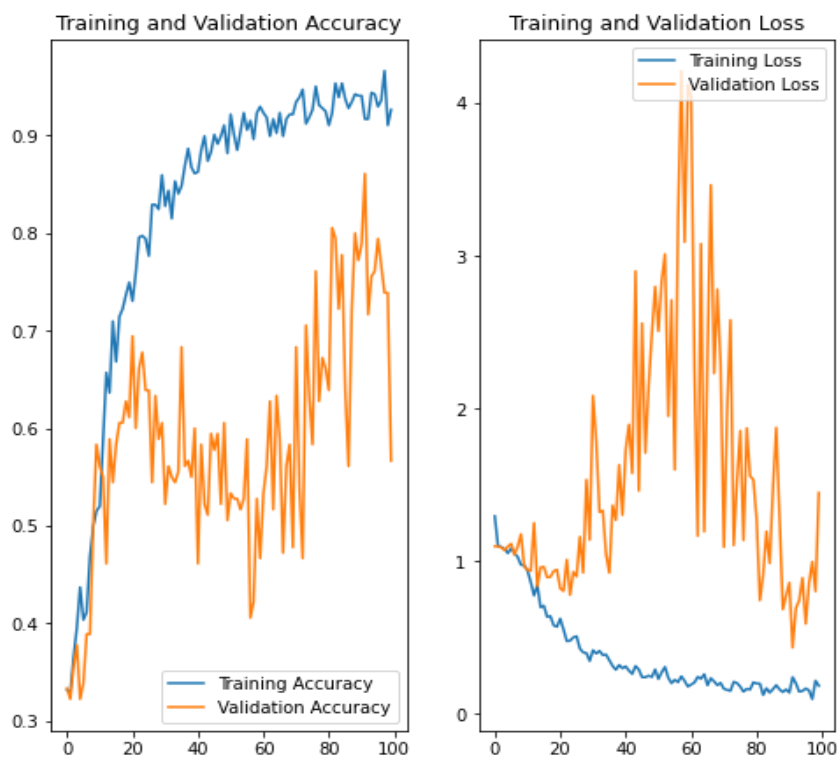


Figure 5.6 CNN+CNN Fusion Model Result

### 5.4.2 CNN + SVM Fusion Model

Figure 5.7 illustrates the CNN + SVM fusion model. The output of the CNN model is provided as an input to the SVM model. This fusion model is trained for 100 epochs. An approximately 49% accuracy is achieved on a training dataset in the CNN + SVM fusion model. Although the accuracy of the CNN + CNN fusion model is approximately 95%, which is way better than the CNN+SVM fusion model.



Figure 5.7 CNN+SVM Fusion Model Result

Table 5.2 shows the accuracies of different image processing algorithms in the leaf dataset. These values are collected throughout the research in different situations. For this dataset, the CNN+CNN fusion model gives the highest accuracy of approximately 95%, whereas the CNN+SVM fusion model provides the lowest accuracy of 73%.

Table 5.2 Accuracies

MODEL	ACCURACY
<b>FUSION CNN+CNN</b>	95%
<b>CNN</b>	92%
<b>SVM</b>	82%
<b>KNN</b>	76%
<b>FUSION CNN+SVM</b>	49%

## 5.5 Confusion Matrix

A confusion matrix illustrated in figure 5.8 is a matrix with four or more combinations of anticipated and real values. Calculating a confusion matrix provides a unique understanding of the errors and results of the algorithms.

		Actual/Values	
		POSITIVE	NEGATIVE
Prediction values	POSITIVE	TP	FP
	NEGATIVE	FN	TN

Figure 5.8 Confusion Matrix

In binary classification, the output has two classes, often known as positive and negative classes. If the image is classified in a correct class, the image fits into the positive class, also known as True Positive; otherwise, the image fits into the negative class (True Negative). A binary classifier always makes two types of mistakes: false negative, and false positive.

Figure 5.9 illustrates the confusion matrix of five different algorithms analyzed in this research work. A leaf disease is classified into three types of diseases; therefore, the confusion matrix is generated of the size of 3 \* 3.

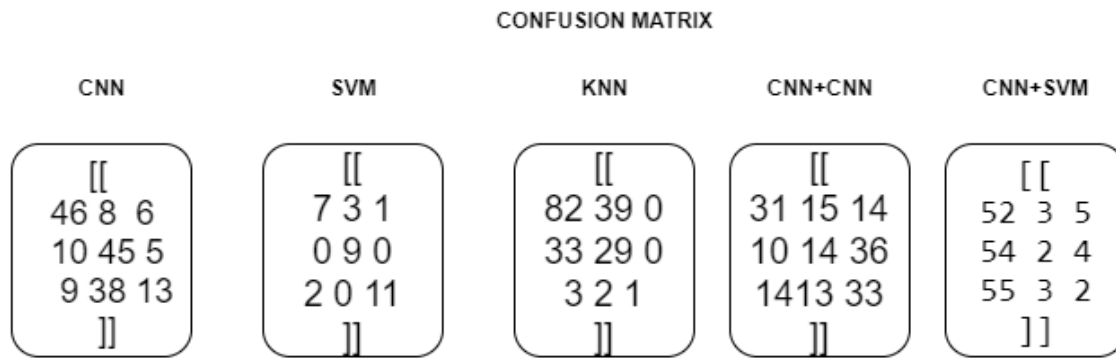


Figure 5.9 Calculated Confusion Matrix

Based on the result of the confusion matrix, the classification report is generated. The classification report is used to measure the quality of predictions from the different classification algorithms. This report ensures how many predictions belong to the predicted class. The following values are calculated to generate a classification report.

- Precision – Precision shows that the percentage of the model's predictions were correct.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

- Recall – The percentage of the positive values did model catch.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

- F1 score – F1 score defines the percentage of positive predictions that were corrected by the model.

$$\text{F1 score} = 2 \times (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$$

### 5.5.1 Precision

Figure 5.9.1 illustrates the comparison of the precision of various models generated during this research. This report is generated using the confusion matrix, and formulas explained above. In the following graph, the CNN+CNN fusion model shows a better result. The CNN+CNN fusion model predicts the output classes correctly, and the images are equally distributed. In terms of precision, the kNN model performs worst compared with other models.

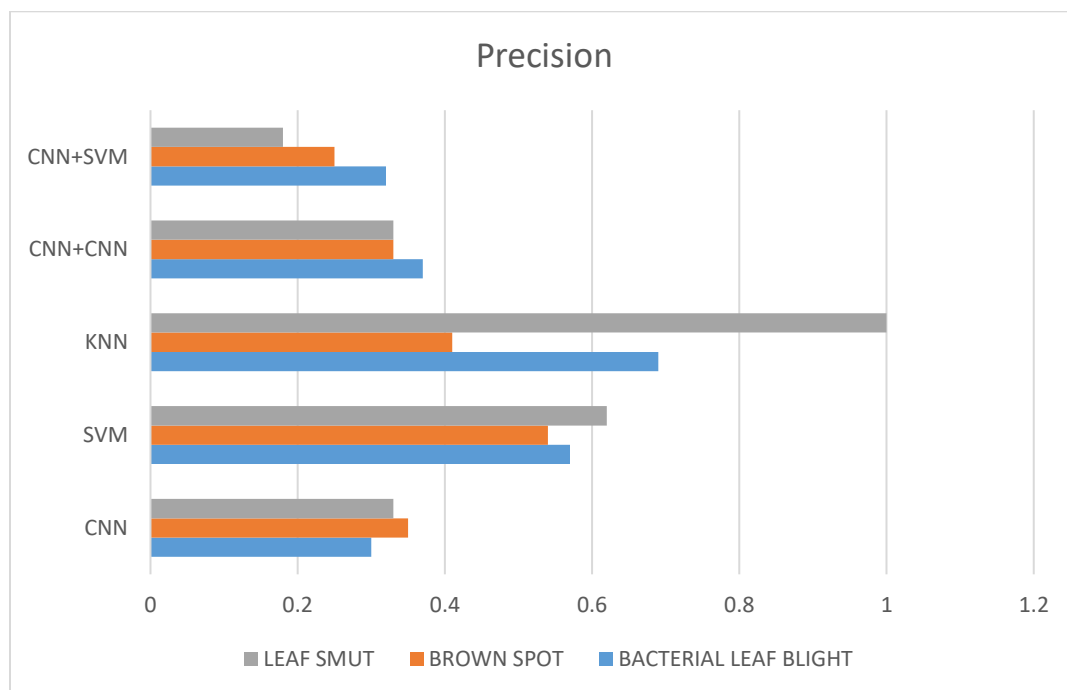


Figure 5.9.1 Precision

### 5.5.2 Recall

Figure 5.9.2 illustrates the comparison of the recall of various models generated during this research. In a recall, the result of the CNN+SVM fusion model performs very poorly as maximum

images were classified in the third class. CNN+CNN fusion model performs better to compare with other models.

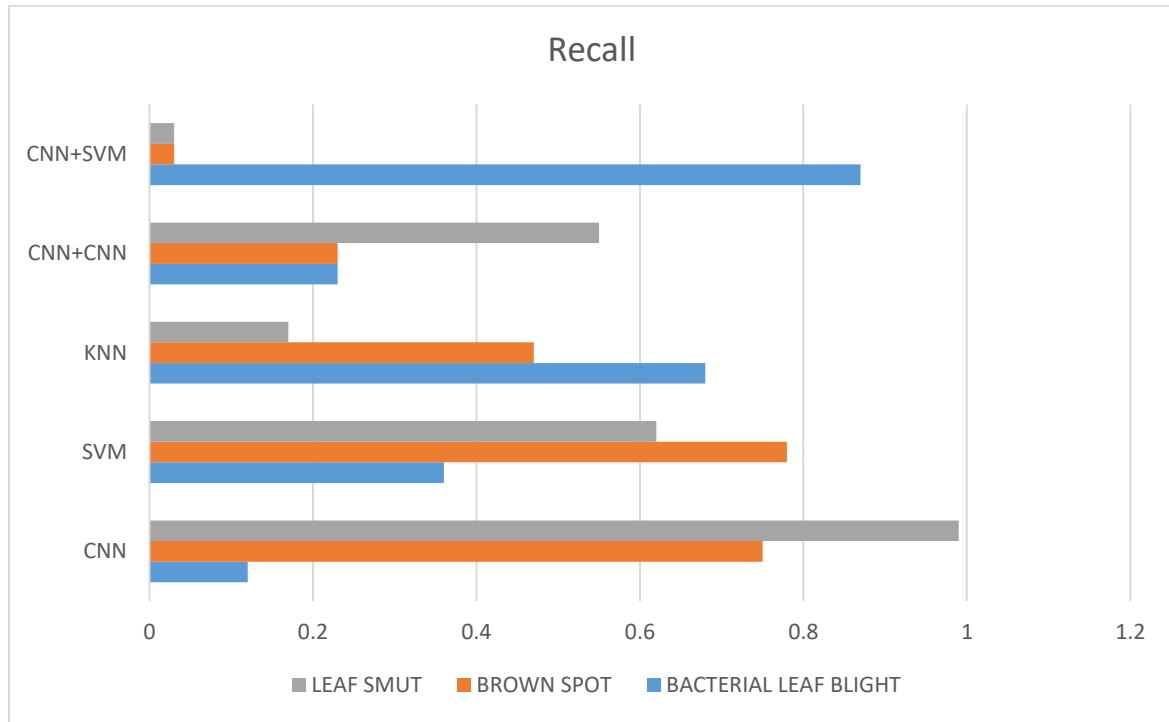


Figure 5.9.2 Recall

### 5.5.3 F1 Score

Figure 5.9.3 illustrates the comparison of the F1 score of various models generated during this research. This report is generated using the confusion matrix, and formulas explained above. The optimal performance in this research is achieved by implementing the CNN+CNN fusion model. The calculations of a CNN+SVM fusion model and a kNN model are worst in terms of the F1 score.

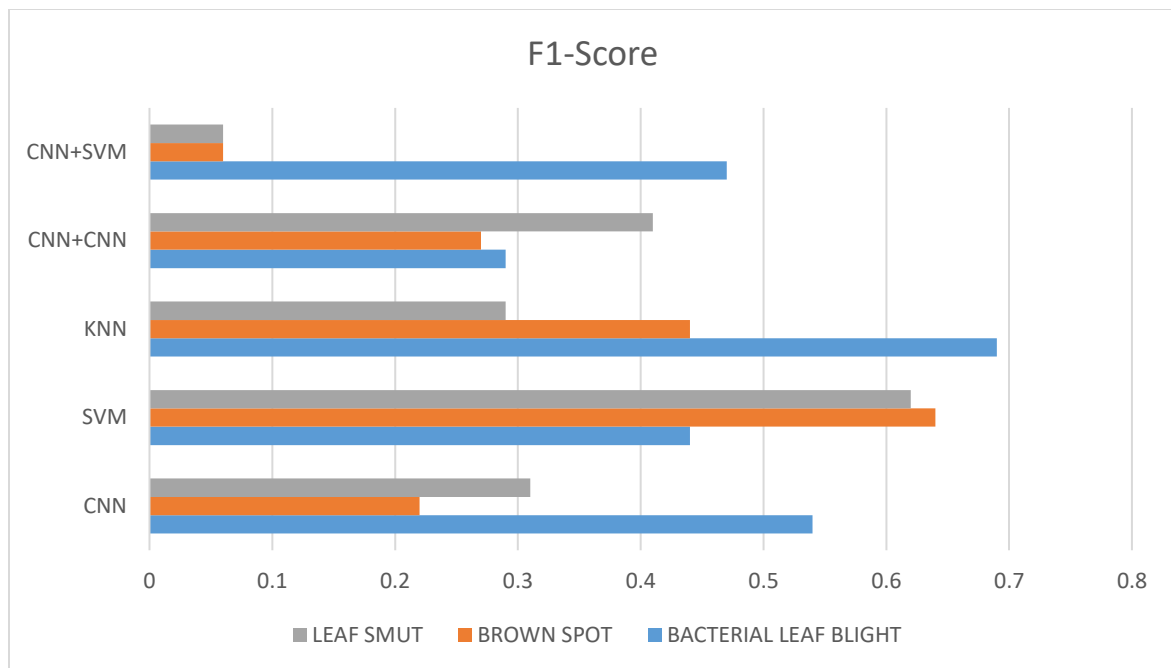


Figure 5.9.3 F1-Score

## 6 CONCLUSION

The system is developed and implemented within the set timeline, which contains a cloud-based web user interface. By executing a fusion and comparison of different image processing algorithms, models are created to predict the leaf disease. The proposed system is inexpensive and user-friendly. Also, the end to end solution is provided to the user with better accuracy.

The earlier studies showed the accuracy of the leaf disease using a single algorithm, but this system is implemented by comparing three algorithms. The previous researchers did not provide an adaptive user interface. The adaptive user interface is developed in this system to provide a more satisfying experience.

This cloud-based web application is available to access from anywhere at any moment. In this research, the CNN+CNN fusion model gives the highest accuracy of approximately 95 %, whereas the CNN+SVM fusion model provides the lowest accuracy of 73%.

This work's unique novel contribution is that it performs all of the machine learning with cloud computing on the AWS EC2 (Amazon Elastic Compute Cloud), whereas previous methods have only stored the data on the cloud; and, the proposed method of this work, fusion CNN + CNN has a 95% accuracy, higher than the other previous methods compared.



## 7 FUTURE WORK

This leaf disease system is implemented using the cloud-based web application. But future work consists of an android application with advanced features like location-specific crop selection and multiple crop-specific disease analysis. Also, the latest technologies like virtual reality or augmented reality will make this application user friendly.

Also, in the near future, the application of transfer learning will be used to make the prediction model more precise. It is a process that reuses the machine learning model implemented by experts, and that model was already trained on a dataset.

A neural network model is trained on a dataset and applies its knowledge to a dataset it has never seen before is known as transfer learning. As of now, the leaf disease dataset is not trained by any of the transfer learning models. For the same reason, it is not possible to use transfer learning on the current dataset.

Two different datasets are used to showcase the advantages of transfer learning. The first dataset is a Cat vs. Dogs dataset followed by the Flower dataset. A Cat vs. Dog dataset is the property of Microsoft, and only 20,000 images are allowed to be used.

Figure 7.1 below illustrates the result of a CNN model. This CNN model is enabled with the transfer learning and trained for 40 epochs. As this dataset is pre-trained by the Mobile-net transfer learning model, just the last output class is changed. More than 92 % accuracy is achieved in just 6 to 8 epochs. For the standard CNN model, it takes 150 epochs to achieve 86% accuracy.



Figure 7.1 Cat vs. Dog dataset

Figure 7.2 illustrates the accuracy of a CNN model on the flower dataset. It is possible to achieve an accuracy of more than 95 % in just 8-10 using transfer learning, which is way faster than the standard CNN model. In the near future, this system will be able to provide a prediction based on the transfer learning model on the leaf dataset.

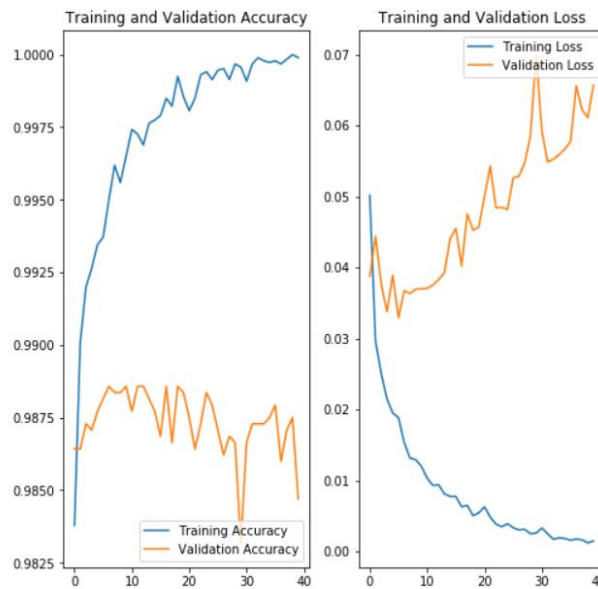


Figure 7.2 Flower dataset

## REFERENCES

1. Singh, Kaushik Kunal. "An Artificial Intelligence and Cloud Based Collaborative Platform for Plant Disease Identification, Tracking and Forecasting for Farmers." In 2018 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), pp. 49-56. IEEE, 2018.
2. Singh, Vijai, and A. K. Misra. "Detection of unhealthy region of plant leaves using image processing and genetic algorithm." In 2015 International Conference on Advances in Computer Engineering and Applications, pp. 1028-1032. IEEE, 2015.
3. Francis, Jobin, and B. K. Anoop. "Identification of leaf diseases in pepper plants using soft computing techniques." In 2016 conference on emerging devices and smart systems (ICEDSS), pp. 168-173. IEEE, 2016.
4. Guo, Xiaoyan, Ming Zhang, and Yongqiang Dai. "Image of Plant Disease Segmentation Model Based on Pulse Coupled Neural Network with Shuffle Frog Leap Algorithm." In 2018 14th International Conference on Computational Intelligence and Security (CIS), pp. 169-173. IEEE, 2018.
5. Sardogan, Melike, Adem Tuncer, and Yunus Ozen. "Plant leaf disease detection and classification based on CNN with LVQ algorithm." In 2018 3rd International Conference on Computer Science and Engineering (UBMK), pp. 382-385. IEEE, 2018.
6. Sabrol, H., and K. Satish. "Tomato plant disease classification in digital images using classification tree." In 2016 International Conference on Communication and Signal Processing (ICCSP), pp. 1242-1246. IEEE, 2016.
7. Niu, Xiaoxiao. "Fusions of CNN and SVM Classifiers for Recognizing Handwritten Characters." Ph.D. diss., Concordia University, 2011.

8. Akram, Tallha, Syed Rameez Naqvi, Sajjad Ali Haider, and Muhammad Kamran. "Towards real-time crops surveillance for disease classification: exploiting parallelism in computer vision." *Computers & Electrical Engineering* 59 (2017): 15-26.
9. Rathod, Arti N., Bhavesh Tanawal, and Vatsal Shah. "Image processing techniques for detection of leaf disease." *International Journal of Advanced Research in Computer Science and Software Engineering* 3, no. 11, (2013).
10. George, Jeena Elsa, J. Aravinth, and S. Veni. "Detection of pollution content in an urban area using landsat 8 data." In *2017 International Conference on Advances in Computing, Communications, and Informatics (ICACCI)*, pp. 184-190. IEEE, 2017.
11. Khitthuk, Chaowalit, Arthit Srikaew, Kitti Attakitmongkol, and Prayoth Kumsawat. "Plant Leaf Disease Diagnosis from Color Imagery Using Co-Occurrence Matrix and Artificial Intelligence System." In *2018 International Electrical Engineering Congress (iEECON)*, pp. 1-4. IEEE, 2018.
12. Prashar, Kapil, Rajneesh Talwar, and Chander Kant. "CNN based on Overlapping Pooling Method and Multi-layered Learning with SVM & k-NN for American Cotton Leaf Disease Recognition." In *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, pp. 330-333. IEEE, 2019.
13. Abdu, Aliyu Muhammad, Musa Mohd Mokji, Usman Ullah Sheikh, and Kamal Khalil. "Automatic Disease Symptoms Segmentation Optimized for Dissimilarity Feature Extraction in Digital Photographs of Plant Leaves." In *2019 IEEE 15th International Colloquium on Signal Processing & Its Applications (CSPA)*, pp. 60-64. IEEE, 2019.
14. Sahithya, Velamakanni, Brahmadevara Saivihari, Vellanki Krishna Vamsi, Parvathreddy Sandeep Reddy, and Karthigha Balamurugan. "GUI based Detection of Unhealthy Leaves

- using Image Processing Techniques." In 2019 International Conference on Communication and Signal Processing (ICCSP), pp. 0818-0822. IEEE, 2019.
15. Ahmed, Md Humayan, Tajul Islam, and Romana Rahman Ema. "A New Hybrid Intelligent GAACO Algorithm for Automatic Image Segmentation and Plant Leaf or Fruit Diseases Identification Using TSVM Classifier." In 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), pp. 1-6. IEEE, 2019.
  16. Chanda, Moumita, and Mantosh Biswas. "Plant disease identification and classification using Back-Propagation Neural Network with Particle Swarm Optimization." In 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), pp. 1029-1036. IEEE, 2019.
  17. Adedoja, Adedamola, Pius Adewale Owolawi, and Temitope Mapayi. "Deep Learning Based on NASNet for Plant Disease Recognition Using Leave Images." In 2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD), pp. 1-5. IEEE, 2019.
  18. Chouhan, Siddharth Singh, Ajay Kaul, Uday Pratap Singh, and Sanjeev Jain. "Bacterial foraging optimization based radial basis function neural network (BRBFNN) for identification and classification of plant leaf diseases: An automatic approach towards plant pathology." IEEE Access 6 (2018): 8852-8863.

