Plant Disease Detection And Classification

A Project Report submitted
in partial fulfilment of the requirements for the degree of

Bachelor of Technology

By: Deepak Meena
(LCS2019059)

Under the supervision of:
Dr. Brijesh Kumar Chaurasia

to

Indian Institute of Information Technology, Lucknow

भारतीय सूचना प्रौद्योगिकी संस्थान, लखनऊ

Department of Computer Science

May 8, 2022

Declaration of Authorship

I, Deepak Meena, declare that this project titled, “Plant Disease Detection And Classification ” and the work presented in it are our own. I confirm that:

- This work was done wholly or mainly while in candidature for a B.Tech at this Institute.
- Where any part of this project has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this Project is entirely my own work.
- I have acknowledged all main sources of help.
- This project is based on the work done by me.

Signature: _____

Date: May 8, 2022
Place: Lucknow, India

Abstract

Agriculture is one of the main factor that decides the growth of any country. In India itself, around 65% of the population is based on agriculture. Due to various seasonal conditions the crops get infected by various kind of diseases. These diseases firstly affect the leaves of the plant and later the whole plant is infected. Therefore identification of these plant diseases is the key to prevent the losses in the yield and quantity of the agricultural product. As there are large number of plants in the farm, it becomes very difficult for the human eye to detect and classify the disease of each plant in the field. And it is very important to diagnose each plant because these diseases may spread. To make the identification of these plant diseases easy ,a lot of research work is done every year but this work never reaches to the needy farmers. That's why in this paper, I not only tried to develop an efficient ML algorithm to detect disease but also developed an open source software that can be used by many others to provide their solution to the end user and help farmers.

Key Functionalities And Features:

Plant disease detection model integrated in mobile application with other features such as weather forecast and useful news/info.

Open source mobile application with the ability to predict plant diseases using leaf images.

Back-end template that can be used to convert a model into REST API.

Acknowledgements

I would like to express my gratitude towards Dr. Brijesh Kumar Chaurasia for his valuable suggestions and encouragement in completion of our project. I would also extend my thanks to Dr. Deepsikha Agarwal, Dr. Dhananjoy Dey, Dr. Abhinesh Kaushik and Dr. Varun Shrama for their valuable feedback to make this project unique and scalable. I would also be thankful to our director Dr. Arun Mohan Sherry for providing all the required facilities in completion of this project.

Contents

List of Abbreviations

CNN	**C**onvolutional **N**eural **N**etwork
DL	**D**eep **L**earning
ML	**M**achine **L**earning
REST	**RE**presentational **S**tate **T**ransfer
API	**A**pplication **P**rogramming **I**nterface

For/Dedicated to/To my...

Chapter 1

Introduction

1.1 Background

India is a cultivated country and about 70% of the population depends on agriculture. Farmers have large range of diversity for selecting various suitable crops and finding the suitable pesticides for plant. Disease on plant leads to the significant reduction in both the quality and quantity of agricultural products. The studies of plant disease refer to the studies of visually observable patterns on the plants. Monitoring of health and disease on plant plays an important role in successful cultivation of crops in the farm. In early days, the monitoring and analysis of plant diseases were done manually by the expertise person in that field. This requires tremendous amount of work and also requires excessive processing time. The image processing techniques can be used in the plant disease detection. In most of the cases disease symptoms are seen on the leaves, stem and fruit. The plant leaf for the detection of disease is considered which shows the disease symptoms. This paper gives the introduction to image processing technique used for plant disease detection.

1.2 Motivation

In India agriculture sector contributes 15-18% towards Indian GDP and round 65-70% of the population is depends on agriculture. Due to various seasonal conditions the crops get infected by various kind of diseases. The attack of these numerous types of diseases on plants results in a huge loss in the yield performance in terms of quality as well as quantity. Plants affected by diseases add up to for about 20-30% of the entire crop deprivation. Due to this, heavy loss is suffered by the farmers of our country. Therefore identification of these plant diseases is the key to prevent the losses in the yield and quantity of the agricultural product. But there is still no easy way to identify these disease spicily in remote areas farmers have to travel far to consult an expert. And due to absence of a open source software work done by various researchers never reaches to the needy farmers. That's why my aim in this paper is not only try to develop a efficient ML algorithm to detect disease but also develops a open source software that can be used by many others to provide their solution to the end user and help farmers.

1.3 Objective

The main objective of my project is to develop a simple to use open source mobile application with the ability to predict plant diseases using leaf images.

I'm providing a

- Plant disease detection model integrated in mobile application with other features such as weather forecast and useful news/info.
- Back-end template that can be used to convert a model into REST API.
- Open source user friendly interface for farmers.

1.4 Contributions

Experts need years of training and knowledge to early disease detection with visual inspection but a plant disease detection model integrated in mobile application can be used by anybody who is not an expert. Any new users just have to take a clear image of plant leaf or can select from gallery to upload it and he will get immediately information about the plant, So that the user can take necessary action. Thus, preventive actions can be taken earlier.

The proposed open source interface can be incorporated with any other model to give valuable suggestions, remedies, disease management, and control strategies, thus ensuring better crop yields.

Chapter 2

Literature Review

Paper[1] "Detection of Potato Disease Using Image Segmentation and Machine Learning" by A. Iqbal and KH. Talukder give solution to the plant disease with image classification. In their approach they collect 500-600 images of different diseased plant leaves such as Bacterial Blight and more. There were total of 3 disease classes and one normal healthy leaf class. They used GoogleNet pre-trained deep convolutional neural network architecture based model. After that some feature extraction is done on the basis of which class is determined. They have achieved an accuracy of 97% but data-set that they have used is vary small and it trained only for one plant leaf.

Another paper[2] named “MDFC-ResNet: An Agricultural IoT System to Accurately Recognize Crop Diseases” clarifies that they have used MDFC-ResNet model for the leaf disease classification. In their methodology they have used a data-set of 60000 images divided into 50000 training and remaining 10000 testing. Total number of different plants used is not specified by the authors but they used around 50 classes of diseases. Three matrixes for R, G, B channels were used as input to MDFC-ResNet model. Average accuracy of around 80-85% was achieved. They used around 50 classes but achieved a less accuracy compared to others.

Paper[3] "Depthwise Separable Convolution Architectures for Plant Disease Classification" by Kc Kamal, Zhendong Yin, Mingyang Wu and Zhilu Wu, They have given a solution to plant disease detection using a Convolution Neural Network (CNN). In their methodology they have used a data-set of 72000 images divided into 60000 training and remaining 12000 testing. And they have achieved an accuracy of around 95% for 24 classes. Paper also describes various strategies for Extracting the nature of infected leaves and classifying plants disease using a Convolution Neural Network (CNN), Which consists of various levels that are used for forecasting.

Paper[3] and other papers [4], [5] are able to achieve an good accuracy but they are too much technical in order to understand and use by farmers they need a simple user interface to become useful for the farmers.

Chapter 3

Problem Formulation

Agriculture plays a very important role in the economic growth of any Country. It is the field which highly affect the GDP of the countries. Agriculture sector contributes around 16% of GDP of India. There are various factors that affects the quality and quantity of crops cultivated. Due to different weather and local conditions these plants are exposed to various diseases. And if these diseases remain undetected may cause some serious losses. The attack of these numerous types of diseases on plants results in a huge loss in the yield performance in terms of quality as well as quantity. Plants affected by diseases add up to for about 20-30% of the entire crop deprivation.

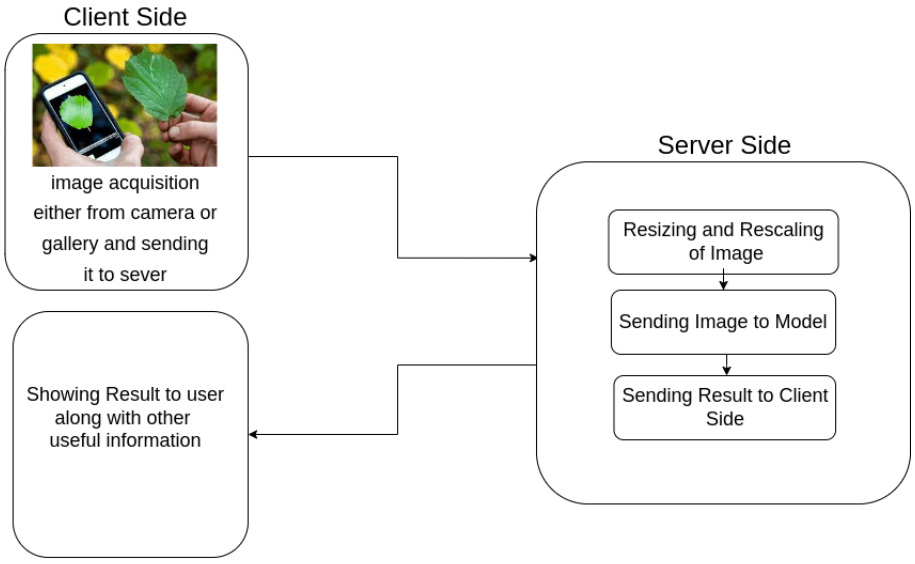
These disease-infected plants usually show obvious marks or lesions on leaves, stems, flowers, or fruits. Generally, each disease or pest condition presents a unique visible pattern that can be used to uniquely diagnose abnormalities. Usually, the leaves of plants are the primary source for identifying plant diseases, and most of the symptoms of diseases may begin to appear on the leaves.

In most cases, agricultural and forestry experts are used to identify on-site or farmers identify fruit tree diseases and pests based on experience.These experts need years of training and knowledge to early disease detection with visual inspection. But remote areas farmers don't have availability to these experts and this method is not only subjective, but also time-consuming, laborious, and inefficient. To counter these challenges, research into the use of image processing techniques for plant disease recognition has become a hot research topic.

Chapter 4

Proposed Methodology

FIGURE 4.1: The general flow of the proposed solution.



Client Side:- A client application is a process or program that sends a job request to a server via the communication network. Those jobs request the server to perform a particular task, such as looking up a record in a database or returning a portion or customized report. I have used **React Native** to build this client application and used **axios** to communicate between server and client.

Server Side:- The server is a collection of the programs, listens for client requests that are transmitted via the communication network. Servers perform actions such as database queries or reading files in my case it's work is to take input from client and process it through ML model and give result back to the client. I have used **flask** and **python** to create this server application.

Dataset:- Datasets are necessary for all research stages, both during the training phase and the phase evaluating the performance of recognition algorithms. The proposed system was trained and assessed on the open-source

data-set Plant Village.

Data Augmentation:- The predictive accuracy of Supervised Deep Learning models depends largely on the amount and variability of data available during training. The relationship between in-depth learning models and the amount of training data required is similar to the relationship between rocket engines (in-depth learning models) and the large amount of fuel (large amount of data) required for a model to complete its function (successful in-depth learning model). DL models trained to achieve high performance in complex operations usually have a large number of hidden neurons. As the number of hidden neurons increases, the number of trained parameters also increases. The number of parameters in State-of-the-art Computer Vision models such as ResNet (60M) and Inception-V3 (24M) is a ten million program. For Indigenous Language Processing Models like BERT (340M), it is a 100 million project. These in-depth learning models are trained to perform complex tasks such as object acquisition or high language translation with a large number of flexible parameters. They need a large amount of data to learn the values of a large number of parameters during the training phase. In simple terms, Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, **resizing**, **re-scaling** padding, and horizontal flipping are commonly used to train large neural networks.

Model Creation and Training :- Model training is the phase of deep learning development where experts try to balance the best combination of weight and bias in the machine learning algorithm to reduce loss function over the prediction range. The purpose of the model training is to create the best mathematical representation of the relationship between the data elements and the target label (supervised reading) or between the features themselves (unregulated reading). Loss operations are an important part of model training as they explain how to develop machine learning algorithms. Depending on the purpose, data type and algorithm, the data science function uses a different type of loss functions. One of the most famous examples of loss operations is the Mean Square Error (MSE). In this i have used image-net with **TensorFlow** which is an open-source programming library for mathematical calculation utilizing information stream diagrams. Added layers using **keras** which are the basic building blocks of neural networks in Keras. A layer consists of a tensor-in tensor-out computation function (the layer's call method) and some state, held in TensorFlow variables (the layer's weights). After building model converted it into an rest api with the help of flask and python which can accept image as an input and provide it to the model for processing and give back result to client.

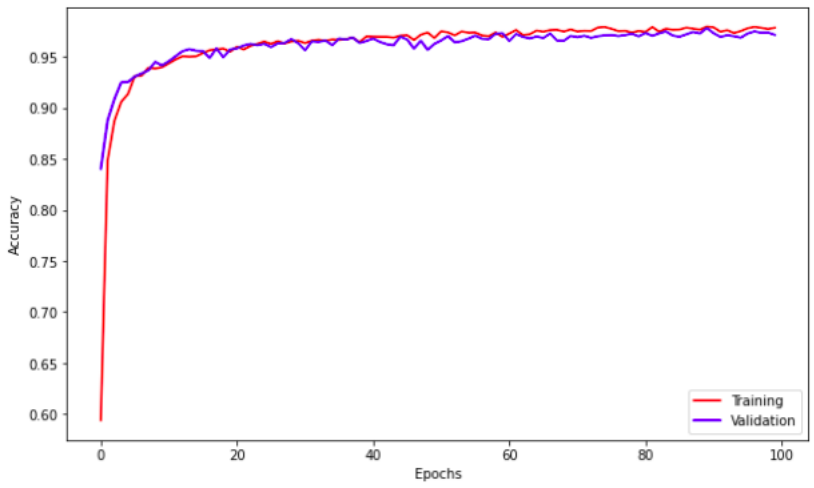
Chapter 5

Experiments and Results

The proposed CNN-based models were trained for approximately 150 epochs as the loss curve converged around the x-axis (Fig. 1). Accuracy and loss score of the network were recorded in every batch. An adam optimizer was exploited during the training step which the learning rate was initially set to 0.001 and momentum set to 0.9. The training parameters were shown in table. As number of epoch increases loss decrease significantly and accuracy increases accordingly.

Training Parameters	
Parameter	Value
Epochs	100
Batch size	64
image_size	224

FIGURE 5.1: Change in accuracy.



As a result created Plant disease detection model integrated in mobile application with around 97% of accuracy.

Chapter 6

Conclusion and Future Work

This paper presents a plant disease detection model integrated in mobile application for detecting and classifying different plant leaf disease presented makes use of convolutional neural network for classification purpose. The presented model used the data-set that consists of more than 76000 images with 38 total classes. The following model can be extended by using even more large data-set with more categories of diseases and the accuracy can also be improved by tuning the hyper-parameters. The remedies for the classified disease can also be included in the model. The proposed client application can be used to transform ML models into a simple mobile application.

Future Work

- Current model is trained for less number of crops due unavailability of large data-set but in future when data-set of larger size will be used to train the model, a better accuracy for large number of crops can be achieved.

- In order to make the application more user accessible it needs to support more local languages but it currently supports only Hindi and English.

- An application can be used for multiple features such as crop recommendation, crop management, weather and news these various features can be integrated into application to make it more useful for farmers.

Appendix A

Code

A.1 Data Argumentation

```
train_datagen = keras.preprocessing.image.ImageDataGenerator(rescale = 1/255.0,
                                                              shear_range = 0.2,
                                                              zoom_range = 0.2,
                                                              width_shift_range = 0.2,
                                                              height_shift_range = 0.2,
                                                              fill_mode="nearest")

test_datagen = keras.preprocessing.image.ImageDataGenerator(rescale = 1/255.0)
```

Let's prepare our data. We will use `.flow_from_directory()` to generate batches of image data (and the
folders.

```
train_data = train_datagen.flow_from_directory(train_dir,
                                               target_size = (image_size, image_size),
                                               batch_size = batch_size,
                                               class_mode = "categorical")

test_data = test_datagen.flow_from_directory(test_dir,
                                             target_size = (image_size, image_size),
                                             batch_size = batch_size,
                                             class_mode = "categorical")
```

```
Found 70295 images belonging to 38 classes.
Found 17572 images belonging to 38 classes.
```

A.2 Model Training

```
inputs = keras.Input(shape = input_shape)

x = base_model(inputs, training = False)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dense(38,
                          activation="softmax")(x)

model = keras.Model(inputs = inputs,
                    outputs = x,
                    name="LeafDisease_MobileNet")
```

n our multiple experiments we found out Adam optimizer to work really well with it's default lea

```
optimizer = tf.keras.optimizers.Adam() #lr=0.05 --- Mention LR here, default - 0.01

model.compile(optimizer = optimizer,
              loss = tf.keras.losses.CategoricalCrossentropy(from_logits = True),
              metrics=[keras.metrics.CategoricalAccuracy(),
                       'accuracy'])
```

```
history = model.fit(train_data,
                    validation_data=test_data,
                    epochs=epochs,
                    steps_per_epoch=150,
                    validation_steps=100)
```

A.3 REST API python flask code

```python
def predict(file):
    pil_image = Image.open(io.BytesIO(file))
    my_image = img_to_array(pil_image)
    img = tf.keras.preprocessing.image.smart_resize(my_image, (256, 256))
    resize_image = tf.reshape(img, [1, 256, 256, 3])
    resize_image = resize_image / 255.0
    predict = model.predict(resize_image)
    index =(np.argmax(predict, axis = -1))[0]    # axis = -1 --> To compute the max ele
    disc=disease_map[index]

    return jsonify({
        'prediction': disease_map[index],
        'description': details_map[disc][0],
        'symptoms': details_map[disc][1],
        'source': details_map[disc][2],
    })


#API dummy request
@app.route("/ping", methods=['GET'])
def ping():
    return jsonify("Hello, I am alive")


#API requests are handled here
@app.route('/net/image/prediction/', methods=['POST'])
def api_predict():
    bytesOfImage = request.get_data()
    diseases = predict(bytesOfImage)
    return diseases
```

A.4 Code for Home Screen

```
return (
  <SafeAreaView style={styles.container}>
    <StatusBar barStyle='dark-content' backgroundColor='#fff' animated={true} />
    <View style={styles.header}>
      <View style={styles.header_title}>
        <Text style={styles.header_title_text}>Plant Test</Text>
      </View>
      <TouchableOpacity
        style={styles.lng_button}
        onPress={() => setVisible(true)}
        activeOpacity={0.5}>
        <Text style={styles.lng_text}>
          <AntDesign name='earth' size={15} /> {t('lng')}
        </Text>
      </TouchableOpacity>
    </View>
    <ScrollView showsVerticalScrollIndicator={false}>
      <WeatherCard />
      <News />
    </ScrollView>
    <BottomSheet setVisible={setVisible} isVisible={isVisible} i18n={i18n} />
    <TouchableOpacity
      style={styles.scan_btn}
      activeOpacity={0.85}
      onPress={() => navigation.navigate('Camera')}>
      <Text style={styles.scan_text}>{t('scan')}</Text>
    </TouchableOpacity>
  </SafeAreaView>
);
}
```

A.5 Code for Camera Screen

```
export default function CameraScreen({ navigation }) {
  let camera;

  const [CameraPermission, setCameraPermission] = useState(false);
  const [ImagePermission, setImagePermission] = useState(false);
  const [CameraType, setCameraType] = useState(Camera.Constants.Type.back);

  const askCameraPermission = async () => {
    const { granted } = await Camera.requestCameraPermissionsAsync();
    setCameraPermission(granted);
  };

  const askImagePermission = async () => {
    const { granted } = await ImagePicker.requestMediaLibraryPermissionsAsync();
    setImagePermission(granted);
  };

  const selectImage = async () => {
    if (!ImagePermission) askImagePermission();
    const result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.Images,
      allowsEditing: false,
      quality: 1,
    });

    if (!result.cancelled) {
      navigation.navigate('Result', { imgPath: result.uri });
    }
  };

  const snapImage = async () => {
    const result = await camera.takePictureAsync();
```

A.6 Code for Result Screen

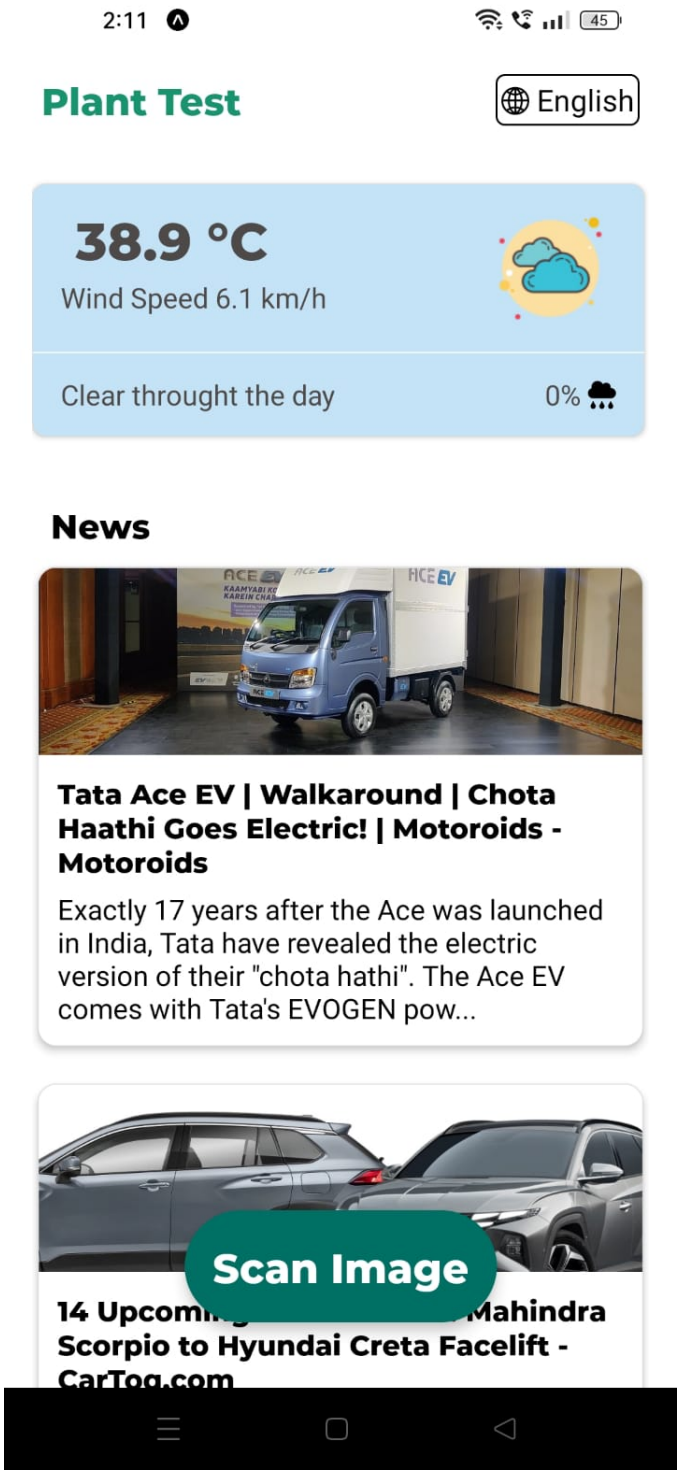
```
return (
  <SafeAreaView style={styles.container}>
    <ScrollView showsVerticalScrollIndicator={false}>
      <Image source={{ uri: imgPath }} style={styles.img} />

      {loading ? (
        <ActivityIndicator style={styles.load} color='green' size={60} />
      ) : (
        <>
          <Text style={styles.title}>
            {t('result')} : {Data?.prediction}
          </Text>
          <Text style={styles.title2}>{t('disc')} : </Text>
          <Text style={styles.disc}>{Data?.description}</Text>
          <Text style={styles.title2}> {t('identify')} : </Text>
          <Text style={styles.disc}>{Data?.symptoms}</Text>
          <Text style={styles.title2}> {t('treatment')} : </Text>
          <TouchableOpacity
            onPress={() => {
              Linking.openURL(Data?.source);
            }}>
            <Text style={styles.disc}>{Data?.source}</Text>
          </TouchableOpacity>
        </>
      )}
    </ScrollView>
  </SafeAreaView>
);
}
```

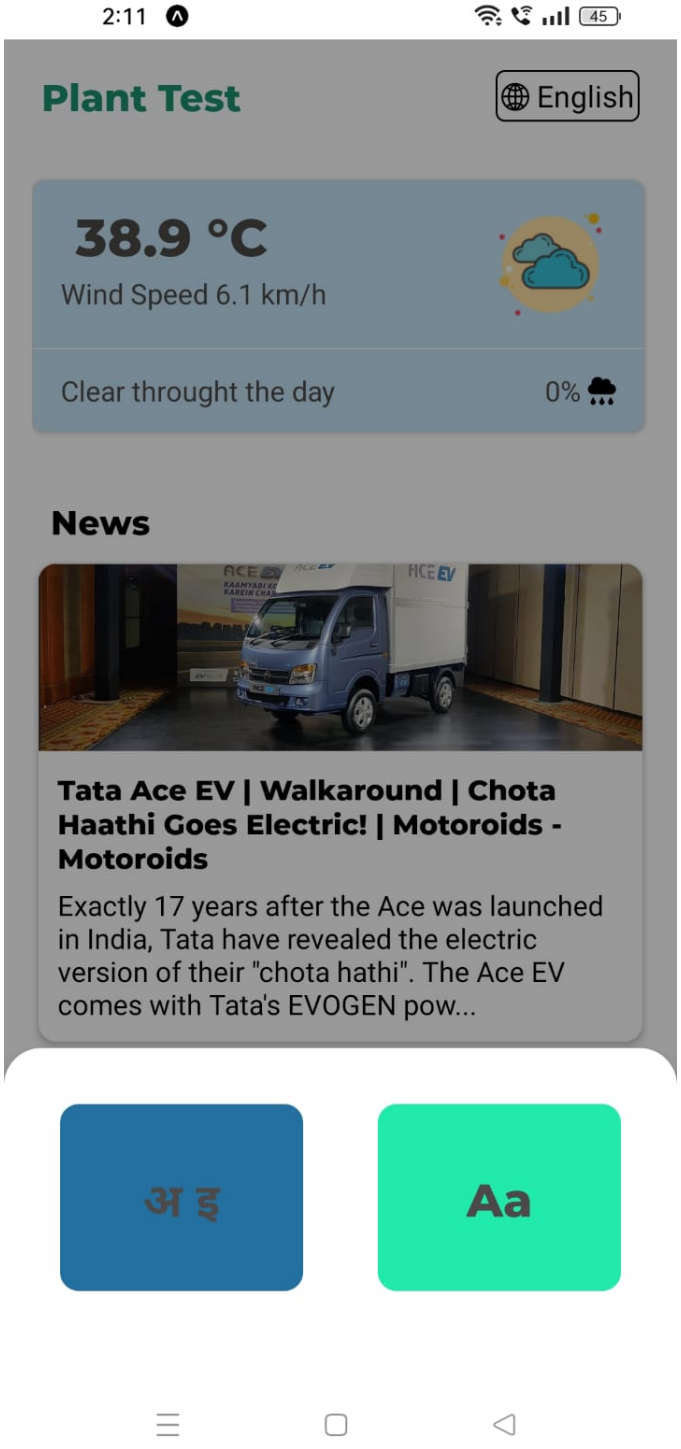
Appendix B

Glimpse of Mobile Application

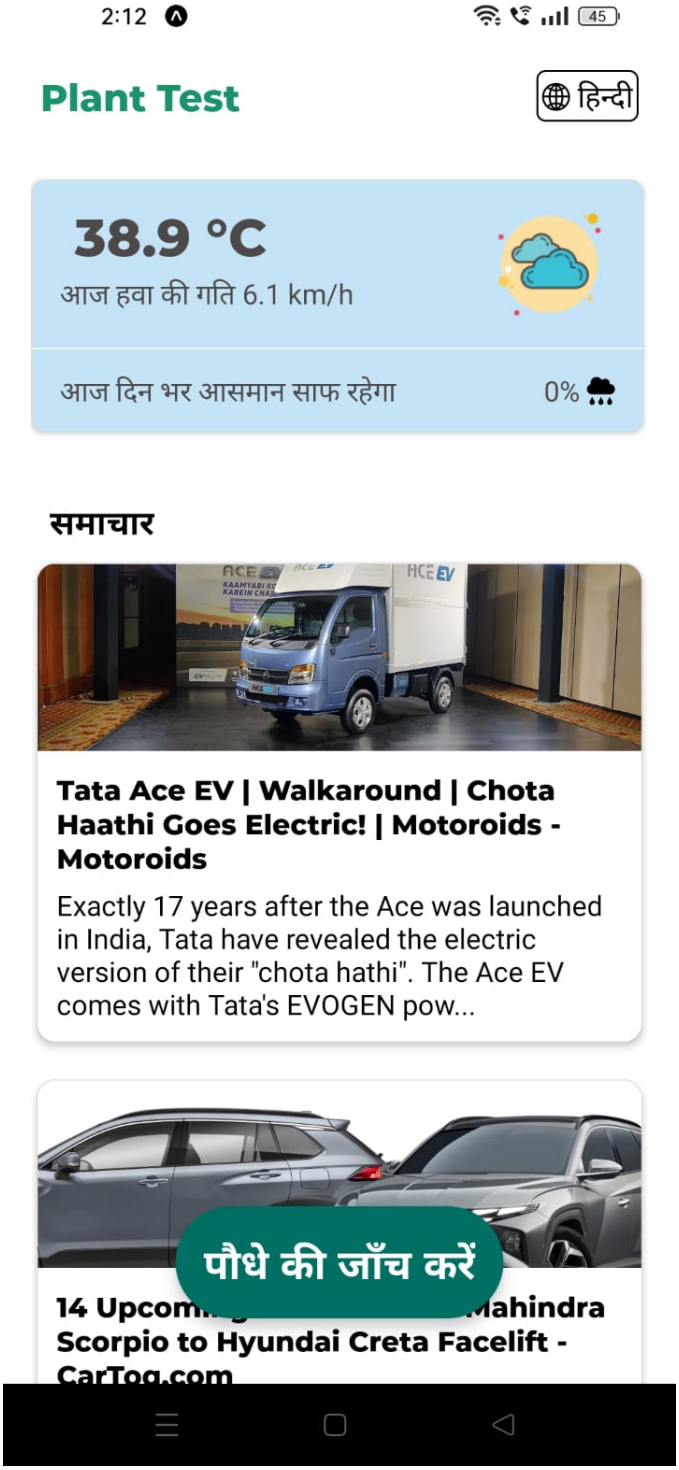
B.1 First Screen of Mobile Application



B.2 User can change language

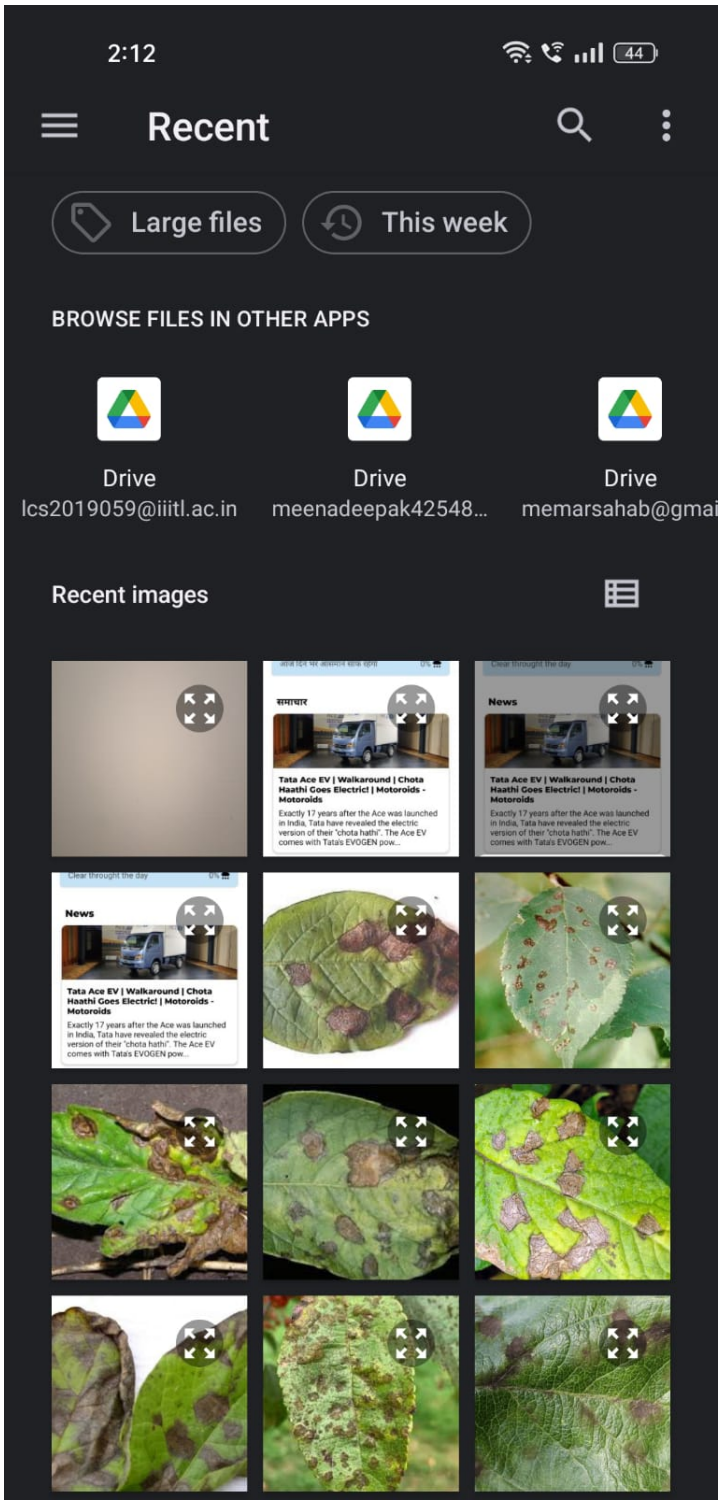


B.3 Home Screen after language change

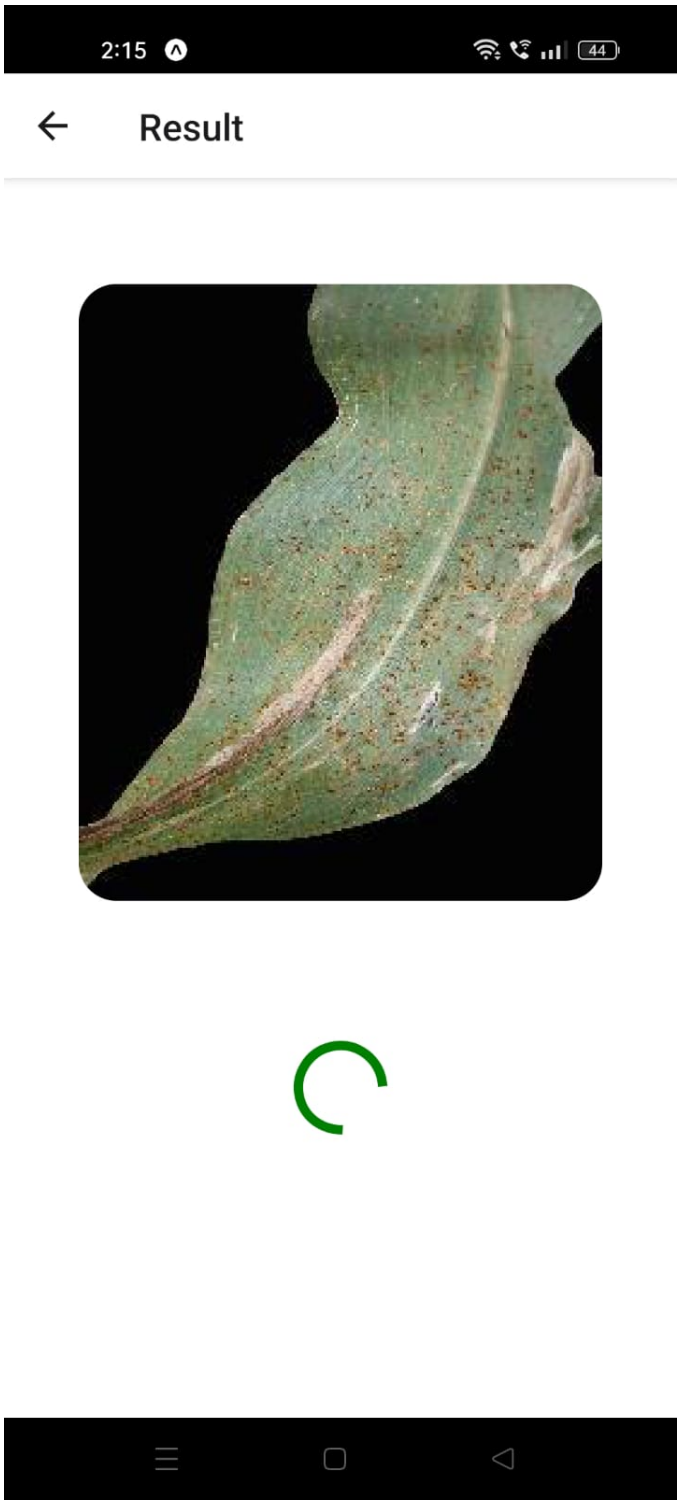


B.4 User can click picture of plant leaf

B.5 User can select image form gallery

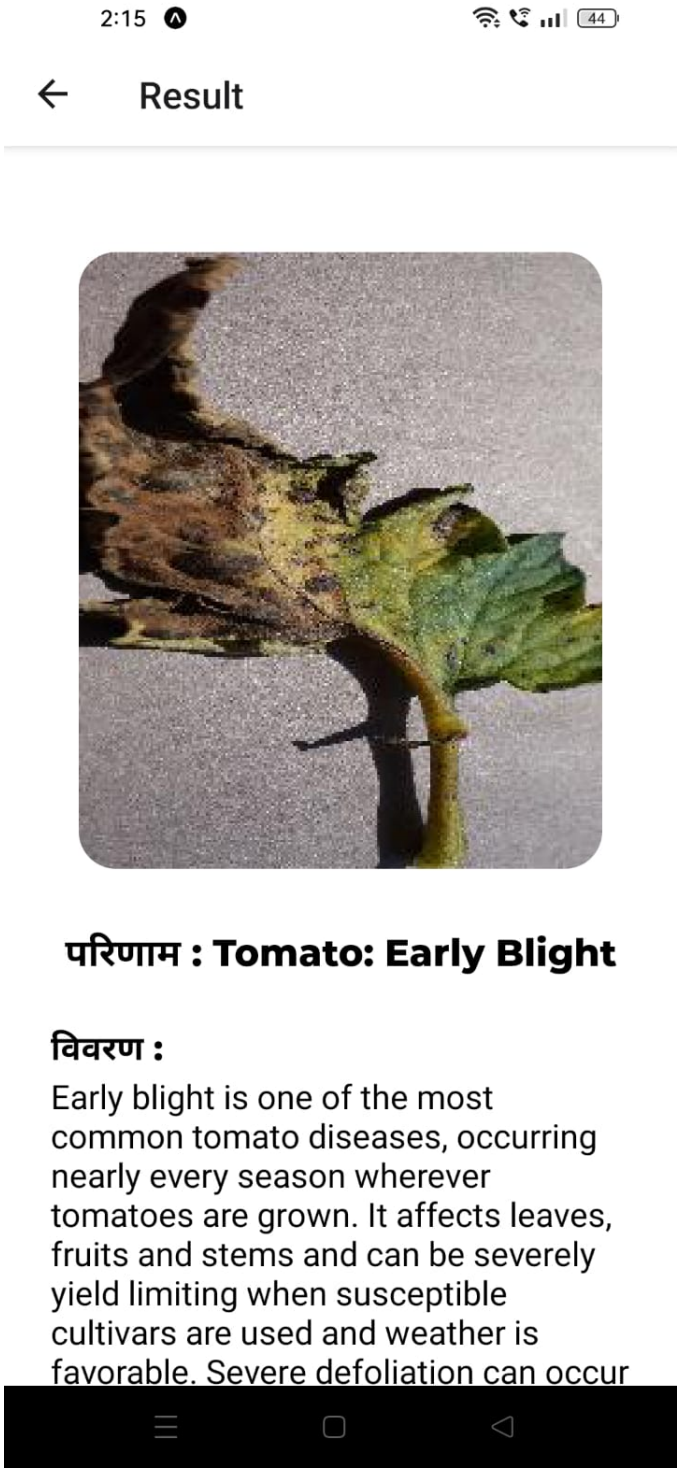


B.6 Waiting for result from server



B.7 Result from server

B.8 Result from server with Hindi



Bibliography

[1] A. Iqbal and KH. Talukder. “Detection of Potato Disease Using Image Segmentation and Machine Learning”, IEEE international conference. pp. 43–47, 2020.

[2] Hu Wei Jian, Jie Fan, Yong Xing Du, Bao Shan Li, Naixue Xiong and Ernst Bekkering. “MDFC-ResNet: An Agricultural IoT System to Accurately Recognize Crop Diseases”, IEEE Access. Vol. 8, pp. 115287–115298, 2020.

[3] Kc Kamal, Zhendong Yin, Mingyang Wu and Zhilu Wu. “Depthwise Separable Convolution Architectures for Plant Disease Classification”, Computers and Electronics in Agriculture. Vol. 165, pp.1-6, 2019.

[4] A. Lakshmana rao, M. Raja Babu, T. Srinivasa Ravi Kiran, "Plant Disease Prediction and classification using Deep Learning ConvNets" IEEE Access. Vol. 8, pp. 21457233, 2021.

[5] K.Singh,S.Kumar, andP.Kaur, "Support vector machine classifier based detection of fungal rust disease in Pea Plant (Pisamsativam)," International Journal of Information Technology, vol. 11, pp. 485-492, 2019.

[6] Plant Village DataSet, David. P. Hughes, Marcel Salathe An open access repository of images on plant health to enable the development of mobile disease diagnostics, eprint 2015.