

PROJECT REPORT ON PLANT DISEASE DETECTION USING CNN

Abstract

Plants have become an important source of energy, and are a fundamental piece in the puzzle to solve the problem of global warming. However, plant diseases are threatening the livelihood of this important source and as agriculture is one of the major sectors in our society and from the medieval times, it is the sector that humans have dwelled upon. About 60-70% of the Indian population depends on the agricultural industry. Across the world, loss faced by the crop due to numerous factors like weeds, diseases and arthropods have increased to an alarming rate of about 34.9% in 1965 to about 42.1% in the late 1990s. The recent revolution in smartphone penetration and computer vision models has created an opportunity for image classification in agriculture. Convolutional Neural Networks (CNNs) are considered state-of-the-art in image recognition and offer the ability to provide a prompt and definite diagnosis. A dataset containing leaf images is established for training and validating the model. In this demonstration to design such a system which can detect crop disease through leaf images and to provide remedy for the disease that is detected. This demonstrates the technical feasibility of CNNs in classifying plant diseases and presents a path towards AI solutions for farmers.

INTRODUCTION

Agriculture has become much more than simply a means to feed ever growing populations. However, plant diseases are threatening the livelihood of this important source. Plant diseases cause major production and economic losses in agriculture and forestry. There are several ways to detect plant pathologies. Some diseases do not have any visible symptoms associated, or those appear only when it is too late to act. In these cases, it is necessary to perform sophisticated analysis, usually by means of powerful microscopes. In some cases, the signs can only be detected in parts of the electromagnetic spectrum that are not visible to humans. Most diseases, however, generate some kind of manifestation in the visible spectrum. The diseases may exhibit symptoms on different parts of the plant, i.e., leaves, stems, fruits/seeds etc.

In most cases, the diagnosis, or at least a first guess about the disease, is performed visually by humans. Trained experts may be efficient in recognizing the disease. Unfortunately, most of the time there are no experts in the area to give a data-based analysis and advice to the farmers. Therefore; looking for a fast, automatic, less expensive and accurate method to detect plant diseases is of great importance.

Machine learning methods, such as artificial neural networks (ANNs), Decision Trees, K-means, k nearest neighbors, and Support Vector Machines (SVMs) have been applied in agricultural research. The traditional approach for image classification tasks has been based on hand-engineered features such as SIFT, HoG, SURF etc., and then to use some form of learning algorithm in these feature spaces. This led to the performance of all these approaches depending heavily on the underlying predefined features. However, a recent trend in machine learning has demonstrated that learned representations are more effective and efficient. The main advantage of representation learning is that algorithms automatically analyze large collections of images and identify features that can categorize images with minimum error. One of the most popular techniques is convolutional neural networks (CNN) and it has been used for object recognition and image classification. A convolutional neural network is a type of deep neural network (DNN) inspired by the human visual system, used for processing images.

LITERATURE REVIEW

In this section the recent trends in using CNN and deep learning architectures in agricultural application are discussed. Prior to the advent of deep learning, image processing and machine learning techniques have been used to classify different plant diseases

Generally, most of these systems follow the following steps: First digital images are acquired using digital cameras. Then image processing techniques, such as image enhancement, segmentation, color space conversion and filtering, are applied to make the images suitable for the next steps. Then important features are extracted from the image and used as an input for the classifier.

The overall classification accuracy is therefore dependent on the type of image processing and feature extraction techniques used. However, latest studies have shown that state of the art performance can be achieved with networks trained using generic data. CNNs are multi-layer supervised networks which can learn features automatically from datasets. For the last few years, CNNs have achieved state-of-the-art performance in almost all-important classification tasks. It can perform both feature extraction and classification under the same architecture.

A CNN is a special kind of neural network that has been widely applied to a variety of pattern recognition problems, such as computer vision, speech recognition, etc. The CNN is based on the human visual system; first inspired by LeCun et al and continually

implemented by many researchers. CNNs combine three architectural ideas to ensure some degree of shift, scale, and distortion invariance: local receptive fields, shared weights and spatial or temporal sub-sampling .

Various CNN architectures were proposed to be used for object recognition e.g., LeNet, AlexNet, GoogLeNet etc. The LeNet architecture is the first CNN introduced by LeCun et al. to recognize handwritten digits.

It consists of two convolutional layers and two sub-sampling layers followed by a fully connected MLP. Few researchers proposed the use of CNN for leaf recognition and plant disease classification. Atabay designed a convolutional neural network architecture to identify plants based on leaf images. The proposed architecture consists of five layers. After each convolutional layer a rectified Linear Unit (ReLU) or Exponential Linear Unit (ELU) activation function is used and for each pooling layer, MaxPooling approach is applied. The proposed system is applied on Flavia and Swedish leaf datasets containing 32 plant species with 1907 samples and 15 species with 1125 samples respectively. The images in the dataset are pictures of a single leaf taken at uniform background. All the input images are 256x256 pixel grayscale images. The model achieved a classification accuracy of 97.24% and 99.11% accuracy for each dataset. The results showed that the proposed architecture for CNN based leaf classification is closely competing with the latest extensive approaches on devising leaf features and classifiers. Angie K. Reyes et al., used a deep learning approach in which the complete system was learned without hand- engineered components. The designed system has 5 Conv layers followed by 2 fully connected layers. The CNN is trained using 1.8 million images from ILSVRC 2012 dataset 1 and uses a fine-tuning strategy to transfer learned recognition capabilities from general domains to the specific challenge of the Plant Identification task. The dataset is a combination of images of a plant or part of a plant taken both under a controlled environment as well as in the natural environment. They obtained an average precision of 0.486. Sharada P. Mohanty et al, used the existing deep CNN architectures, i.e. AlexNet and GoogLeNet to classify plant diseases. Using a public dataset of 54,306 images of diseased and healthy plant leaves collected under controlled conditions, the CNN was trained to identify 14 crop species and 26 diseases (or absence thereof). The models achieved 99.35% accuracy. When tested on a set of images taken at a different environment than the images used for the training, however, the model's accuracy dropped to 31.4%. Overall, the result demonstrates the feasibility of deep CNN for plant disease classification.

METHODOLOGY

Creating an AI web application that detects diseases in plants using CNN model which built on the top of Google's deep learning platform: TensorFlow. According to the Food and Agriculture Organization of the United Nations (UN), transboundary plant pests and diseases affect food crops, causing significant losses to farmers and threatening food security.

Dataset used: The team used the "Plant Village" dataset by S.P. Mohanty. This dataset contains an open access repository of images on plant health to enable the development of mobile disease diagnostics. The dataset contains 54,309 images. The images span 14 crop species: Apple, Blueberry, Cherry, Grape, Orange, Peach, Bell Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, and Tomato. It contains images of 17 fungal diseases, 4 bacterial diseases, 2 molds (oomycete) diseases, 2 viral diseases, and 1 disease caused by a mite. 12 crop species also have images of healthy leaves that are not visibly affected by a disease.

Dataset consists of 38 disease classes from Plant Village dataset. 80% of the dataset is used for training and 10% for validation and 10% for testing. The code: We can use the pre-trained resnet50 model or we will train the model.

Import all the import libraries:

```
import tensorflow as tf
from tensorflow import keras
from keras import models, layers
import matplotlib.pyplot as plt
from keras.applications.resnet import ResNet50
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
```

Defining all the necessary constants:

```
IMAGE_SIZE=256
BATCH_SIZE=32
CHANNELS=3
EPOCHS=50
```

Adding path of the dataset (from google drive):

```
dataset=tf.keras.preprocessing.image_dataset_from_directory(
    "/content/drive/MyDrive/ML/PlantVillage/segmented",
    shuffle=True,
    image_size= (IMAGE_SIZE, IMAGE_SIZE),
    batch_size= BATCH_SIZE
)
```

directory:: Directory where the data is located.

Shuffle: Whether to shuffle the data to get random images from dataset.

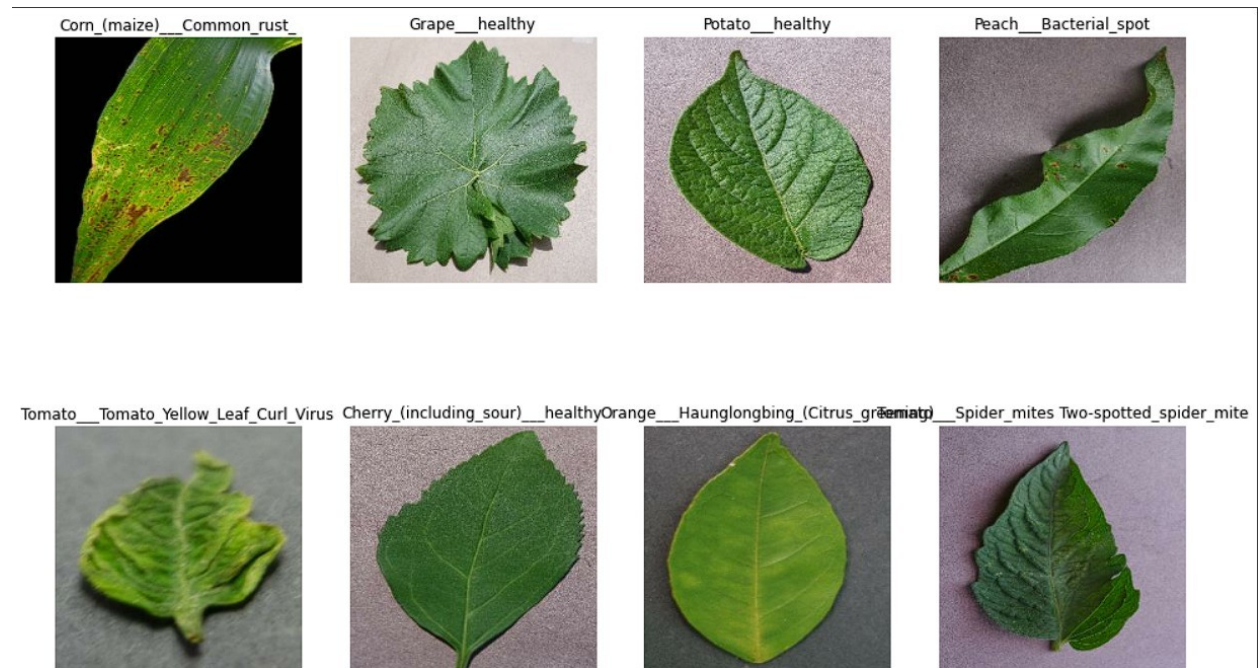
image_size: Size to resize images to after they are read from disk.

batch_size: Size of the batches of data. Default: 32.

Plotting images:

To look at a random sample of images, we can use matplotlib and for loop to iterate over the dataset.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```



Print all the data classes present in the data. In total, we have images in 39 classes as mentioned above

```
class_names=dataset.class_names
class_names
```

```
['Apple___Apple_scab', 'Apple___Black_rot', 'Apple___Cedar_apple_rust', 'Apple___healthy',
'Blueberry___healthy', 'Cherry_(including_sour)___Powdery_mildew',
'Cherry_(including_sour)___healthy', 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
'Corn_(maize)___Common_rust_', 'Corn_(maize)___Northern_Leaf_Blight',
'Corn_(maize)___healthy', 'Grape___Black_rot', 'Grape___Esca_(Black_Measles)',
'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)', 'Grape___healthy',
'Orange___Haunglongbing_(Citrus_greening)', 'Peach___Bacterial_spot', 'Peach___healthy',
'Pepper_bell___Bacterial_spot', 'Pepper_bell___healthy', 'Potato___Early_blight',
'Potato___Late_blight', 'Potato___healthy', 'Raspberry___healthy', 'Soybean___healthy',
'Squash___Powdery_mildew', 'Strawberry___Leaf_scorch', 'Strawberry___healthy',
'Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___Late_blight',
'Tomato___Leaf_Mold', 'Tomato___Septoria_leaf_spot', 'Tomato___Spider_mites Two-spotted_spider_mite',
'Tomato___Target_Spot', 'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
'Tomato___Tomato_mosaic_virus', 'Tomato___healthy']
```

Split the dataset in train test and validation data creating a function.

```
def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0.1, shuffle=True, shuffle_size=10000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds
train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)
```

DO some resizing and rescaling along with data augmentation:

```
resize_and_rescale= tf.keras.Sequential([layers.Resizing(IMAGE_SIZE, IMAGE_SIZE),
```

```
layers.Rescaling(1.0/255)])
```

Resize and Rescale: to make images of same size before feeding it to the model.

```
data_augmentation= tf.keras.Sequential([
    layers.RandomFlip("horizontal_and_vertical"),
    layers.RandomRotation(0.2),
])
```

Data Augmentation: To make the model robust so that if the image is of different contrast or in different orientation (ex: vertical taken) the model should be able to perform accurately.

Building The Model:

```
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3

model=models.Sequential([
    resize_and_rescale,
    data_augmentation,
    layers.Conv2D(32, (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2))
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])
model.build(input_shape=input_shape)
```

The proposed model consists of 7 convolutional layers each followed by a maxpooling layer. The final layer is fully connected dense layer with. ReLu activation function is applied to the output of every convolutional layer and fully connected layer.

The first convolutional layer filters the input image with 32 kernels of size 3x3. After maxpooling is applied, the output is given as an input for the second convolutional layer with 64 kernels of size 3x3. The last convolutional layer has 64 kernels of size 3x3 followed by a fully connected layer flatten layer of 256 neurons followed by a dense layer. The output of this layer is given to softmax function which produces a probability distribution of the three output classes. The architecture of the proposed model is

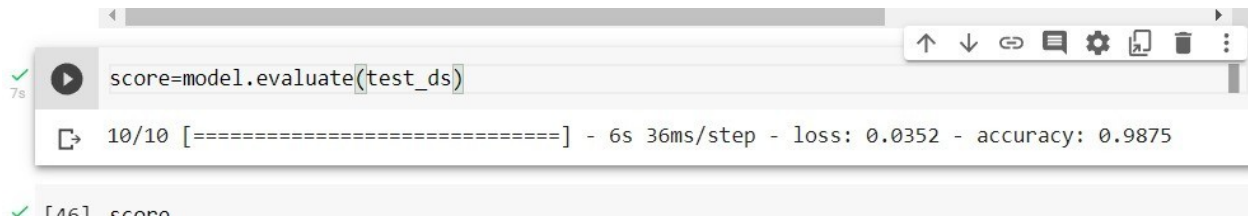
Compiling the model:

```
model.compile(  
    optimizer='adam',  
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),  
    metrics=['accuracy']  
)
```

The model is trained using adaptive moment estimation (Adam) with batch size of 32 for 10 epochs.

Traning the Model:

```
history=model.fit(  
    train_ds,  
    epochs=EPOCHS,  
    batch_size=BATCH_SIZE,  
    verbose=1,  
    validation_data=val_ds  
)
```

```
score=model.evaluate(test_ds)
```

10/10 [=====] - 6s 36ms/step - loss: 0.0352 - accuracy: 0.9875

As we can see above by just running ten epochs with the default setting our accuracy for this fine-grained classification task is around ~98.75%.

Now save the model

We append the model to the list of models as a new version

```
import os
model_version=max([int(i) for i in
os.listdir("../models") + [0]])+1
model.save(f"../models/{model_version}")
model.save("../model.h5")
```

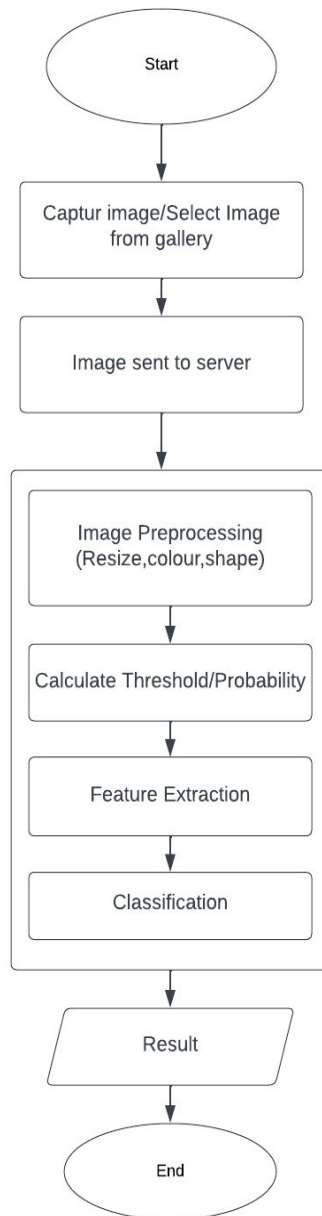
The saved and exported model will be used for app and the website.

For website will be using react js and fast api server and

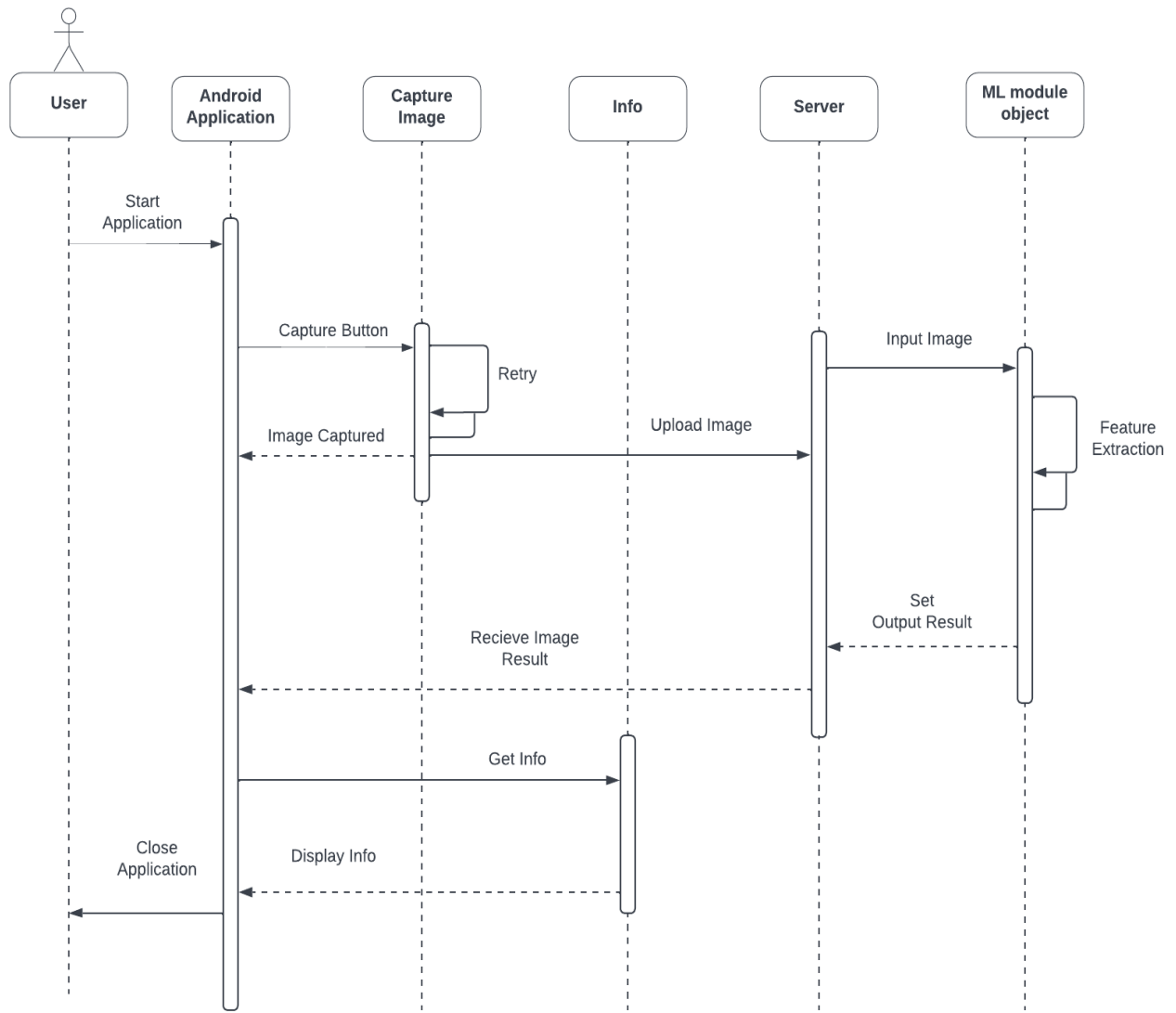
For our app will be using react native along with google cloud platform

UML DIAGRAM

Flow Diagram of a System

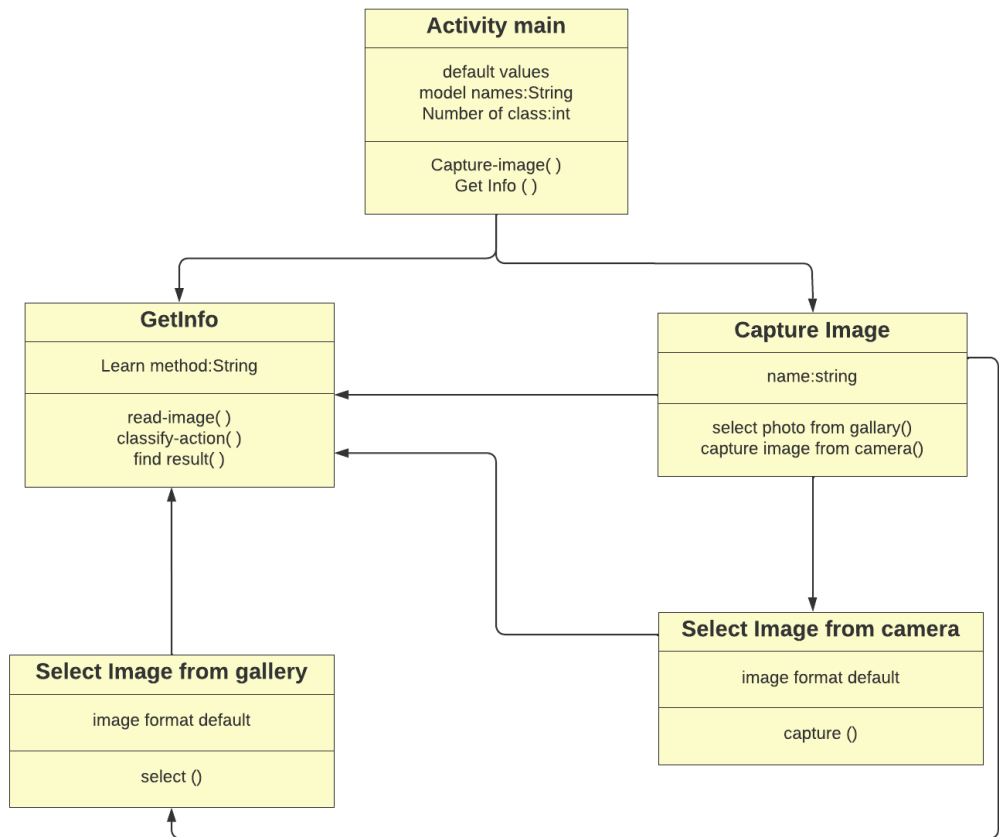


Sequence diagram



Sequence Diagram

Class Diagram



Class Diagram

Module Split-up

1.Capturing and uploading of image:

Using Mobile Phone's camera capture the image of the leaf plant and by mobile application upload the image to the server.

2. Pre-processing of image:

The main objective of pre-processing any image is to improve the quality of the image. Pre-processing of images also helps to better extract features from images and reduce the noise present in it.

3. Feature extraction from image:

Feature extraction plays a significant part in image classification. Feature extraction can be utilized in numerous applications. In-plant disease classification uses many features like color, texture, morphology, and edges, etc. This approach has tracked down that morphological outcomes give preferable outcomes over different features. It tends to be identifying the infected plant leaf of the classification plant image. We can likewise extricate unique plants and unique diseases from the dataset.

4. Classification of diseases present in image:

Dataset is divided into training and testing dataset in some fixed ratio (Here 80% and 20% respectively). Images present in the training dataset are fed to CNN. Features extracted from each image are fed to each neuron present in Input layer. Output of the input layer is input for the next layer. Here the ReLU activation function is used which works as follows, if the feature value is between certain ranges for that particular neuron, then it outputs 1 otherwise 0. Adam optimizer is used for weight initialization for each edge and it also adaptively, changes the value of weight iteratively in each Epoch. Model is trained for a certain number of Epochs so to get optimum accuracy.

5 Report to user:

Report displays the detailed description about the disease caused. It also contains the necessary steps and pesticides that should be taken to further prevent the disease.

Proposed System

1.Capturing and uploading of image:

Using Mobile Phone's camera capture the image of the leaf plant and by mobile application upload the image to the server.

2. Pre-processing of image:

The main objective of pre-processing any image is to improve the quality of the image. Pre-processing of images also helps to better extract features from images and reduce the noise present in it.

3. Feature extraction from image:

Feature extraction plays a significant part in image classification. Feature extraction can be utilized in numerous applications. In-plant disease classification uses many features like color, texture, morphology, and edges, etc. This approach has tracked down that morphological outcomes give preferable outcomes over different features. It tends to be identifying the infected plant leaf of the classification plant image. We can likewise extricate unique plants and unique diseases from the dataset.

4. Classification of diseases present in image:

Dataset is divided into training and testing dataset in some fixed ratio (Here 80% and 20% respectively). Images present in the training dataset are fed to CNN. Features extracted from each image are fed to each neuron present in Input layer. Output of the input layer is input for the next layer. Here the ReLU activation function is used which works as follows, if the feature value is between certain ranges for that particular neuron, then it outputs 1 otherwise 0. Adam optimizer is used for weight initialization for each edge and it also adaptively, changes the value of weight iteratively in each Epoch. Model is trained for a certain number of Epochs so to get optimum accuracy.

5 Report to user:

Report displays the detailed description about the disease caused. It also contains the necessary steps and pesticides that should be taken to further prevent the disease.

Software Tools Used

There is various presence of designing tools to create figures and diagrams like use case diagram, sequence diagram and another desired diagram. In this project, Lucid flow (i.e., flowchart maker and Online Diagram Software) was used for diagrammatic design of the proposed system.

Client Side

□ Front end

The following tools are used for Front-End design:

React Native - React Native (also known as RN) is a popular JavaScript-based mobile app framework that allows you to build natively-rendered mobile apps for iOS and Android. React Native development is used to power some of the world's leading mobile apps, including Instagram, Facebook, and Skype.

React JS - ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application

□ Back end

We have used the following tools are used for Back-End:

□ **Python:** interpreted high-level programming language. Both train and testing data is done.

□ **Matplotlib:** a plotting library for Python programming language and for numerical mathematics extension.

□ **TensorFlow:**

□ **Flask API:** Flask API provides an implementation of browsable APIs (Application Platform

Interface). API is a set of functions and procedures allowing the creation of applications that

access the features or data of an operating system, application, or other service. Flask API

gives proper content negotiated-responses and smart request parsing. It is designed to start quickly and easily, with the ability to scale up to complex application.

Proposed outcomes

Build and deploy a Model on Fast Server to be used by website and GCP which will be used by the Application used for detecting various plant disease.