DATS 6203 Final Project Report - Group 5
Cotton Plant Disease Prediction

Individual Final Report
6th Dec 2021
Pranay Bhakthula
G0796186

Introduction:

The goal of our project is to identify Diseased cotton plants when the photo of the plant is given.

Agriculture is one of the major industries and contributes to everyday life of people all around the world. Diseases on plantations is a major problem to all the farmers, which damages the crop and affects their yield, if unnoticed may even affect people who consume those products. Finding diseased plants in the acres of crop is a tedious task and requires a lot of manpower, time and mundane work and is prone to human error.

Our project aims to train a model on images of fresh, diseased cotton plants and develop an interface to identify plants containing disease or not.

Personal Contribution:

1. Model building:

   Built a Deep Learning model - Densenet121

2. GUI building:
   Built an application based on flask to display the results.

Deep Learning models:

For the purpose of the project we used Convolution Neural Networks to build a custom CNN model. We also used Pre-trained models like Resnet50, VGG16, Densenet. We trained the build model with 100 epochs and pre-trained models with 30 epochs and selected the model which gave the best accuracy and f1-score. We got the highest accuracy with the resnet50 Network.

Convolution Networks:

Convolution Neural Network -- [S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.]

A convolution network is a multilayer feedforward network that has two or three-dimensional inputs. It has weight functions that are not generally viewed as matrix multiplication operations.

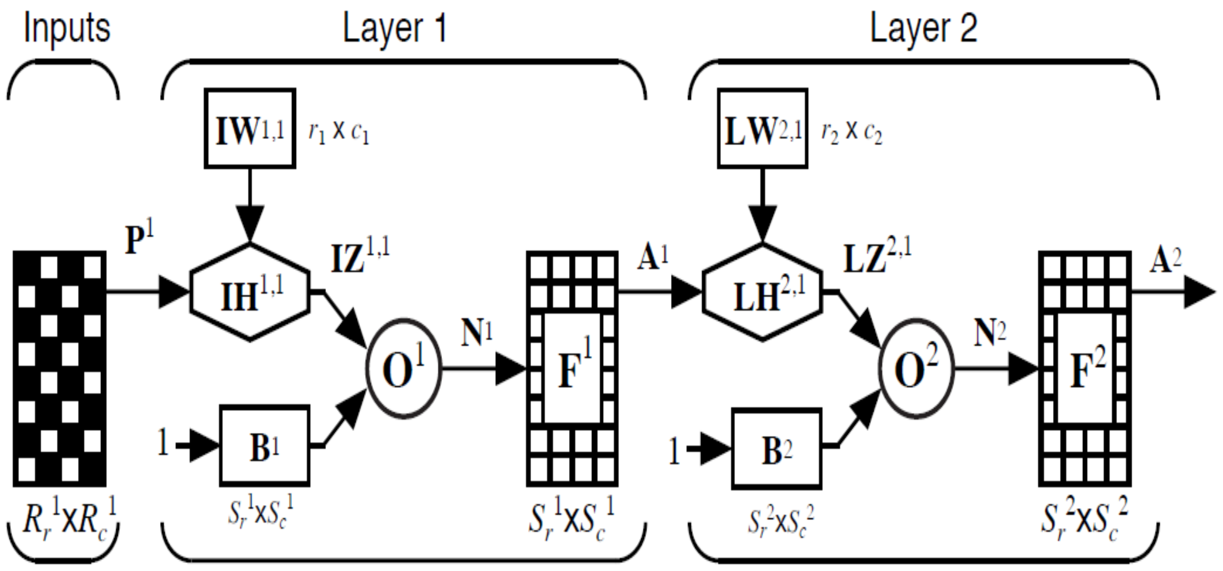


Figure1 : convolution network

Let the input image be represented by the Rr x Rc matrix V. The weight function for this layer performs a convolution kernel that is represented by the r x c matrix W.

$$z_{i,j} = \sum_{k=1}^{r} \sum_{l=1}^{c} w_{k,l} v_{i+k-1,j+l-1} \cdot$$

Figure2

In matrix form we write it as:

$$\mathbf{Z} = \mathbf{W} \circledast \mathbf{V}$$

Figure3

Densesnet:

Densenet -- [Source : **arXiv:1608.06993,** Authors: Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger]

A DenseNet is a type of convolutional neural network that utilises dense connections between layers, through Dense Blocks, where we connect *all layers* (with matching feature-map sizes) directly with each other. To preserve the feed-forward nature, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers.

Densenet Architecture:

In a DenseNet architecture, each layer is connected to every other layer, hence the name Densely Connected Convolutional Network. For L layers, there are L(L+1)/2 direct connections. For each layer, the feature maps of all the preceding layers are used as inputs, and its own feature maps are used as input for each subsequent layers.

This is really it, as simple as this may sound, DenseNets essentially connect every layer to every other layer. This is the main idea that is extremely powerful. The input of a layer inside DenseNet is the concatenation of feature maps from previous layers.

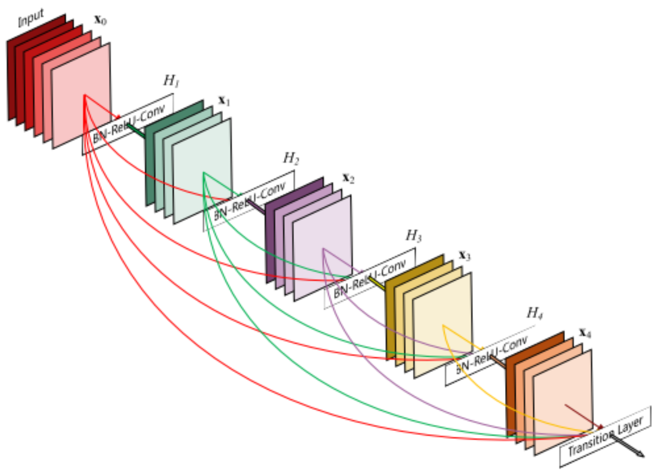


Figure4 : Densenet Architecture

<u>Model Metrics:</u>

We looked at the following metrics in deciding how effective our models were:

F1-score

F1-score weights precision and recall [blog.floydhub.com].

$$F1 - Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

Figure5

Accuracy

Accuracy is simply what percentage of observations were classified correctly [blog.floydhub.com].

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

Figure6

Personal contribution in detail:

Import libraries:

```
import tensorflow
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.densenet import DenseNet121
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import cv2
from skimage.transform import rotate
from skimage.util import random_noise
from skimage.exposure import adjust_gamma
from skimage.filters import gaussian
import numpy as np
import glob
from  sklearn.metrics  import  f1_score,  confusion_matrix,  accuracy_score,
classification_report

from tensorflow.python.framework.ops import tensor_id
```

Modeling:

```
IMAGE_SIZE = [224, 224]

train_path = '/train'
valid_path = '/val'
classes = glob.glob('train/*')

#Data Loader
train_datagen = ImageDataGenerator()
val_datagen = ImageDataGenerator()
training_set = train_datagen.flow_from_directory('train',
                                                 target_size = (224, 224),
                                                 batch_size = 64,
                                                 class_mode = 'categorical')
val_set = val_datagen.flow_from_directory('val',
                                            target_size = (224, 224),
                                            batch_size = 64,
                                            class_mode = 'categorical')



model = DenseNet121(input_shape=IMAGE_SIZE + [3], weights='imagenet',
include_top=False)
        for layer in model.layers:
            layer.trainable = False
        x = Flatten()(model.output)
        prediction = Dense(len(classes), activation='softmax')(x)
        model = Model(inputs=model.input, outputs=prediction)
        model.compile(
      loss='categorical_crossentropy',
      optimizer='Adam',
      metrics=['accuracy']
    )


check_point      =      tensorflow.keras.callbacks.ModelCheckpoint('model_densenet121.h5',
monitor='accuracy', save_best_only=True)

m = model.fit(
  training_set,
  validation_data=val_set,
  epochs=30,
  steps_per_epoch=len(training_set),
  validation_steps=len(val_set),
  callbacks=[check_point]
)

val_preds = final_model.predict(np.array([x for i in range(len(val_set)) for x in val_set[i][0]]))
val_preds = np.argmax(val_preds, axis=1)
val_actual = [x for i in range(len(val_set)) for x in val_set[i][1]]
val_actual = np.argmax(val_actual, axis=1)

print("Accuracy:" + str(accuracy_score(val_actual, val_preds)))
print("F1 Score:" + str(f1_score(val_actual, val_preds, average='micro')))
print("Confusion Matrix:\n" + str(confusion_matrix(val_actual, val_preds)))
print("Classification Report:\n" + str(classification_report(val_actual, val_preds)))


#Test Model
final_model = tensorflow.keras.models.load_model('model_resnet50.h5')

test_image = image.load_img('test/diseased cotton leaf/dis_leaf (322).jpg', target_size = (224,
224))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
preds = final_model.predict(test_image)
preds = np.argmax(preds, axis=1)

if preds==0:
  print("The leaf is diseased cotton leaf")
elif preds==1:
  print("The leaf is diseased cotton plant")
elif preds==2:
  print("The leaf is fresh cotton leaf")
else:
  print("The leaf is fresh cotton plant")
```

GUI code:

```
from __future__ import division, print_function
# coding=utf-8
import sys
import os
import glob
import re
import numpy as np
import tensorflow as tf
import tensorflow as tf

from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

config = ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.2
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)
# Keras
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

# Flask utils
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
#from gevent.pywsgi import WSGIServer

# Define a flask app
app = Flask(__name__)

# Model saved with Keras model.save()
MODEL_PATH ='model_resnet50.h5'

# Load your trained model
model = load_model(MODEL_PATH)
```

```
def model_predict(img_path, model):
    print(img_path)
    img = image.load_img(img_path, target_size=(224, 224))

    # Preprocessing the image
    x = image.img_to_array(img)
    # x = np.true_divide(x, 255)
    ## Scaling
    #x=x/255
    x = np.expand_dims(x, axis=0)


    # Be careful how your trained model deals with the input
    # otherwise, it won't make correct prediction!
   # x = preprocess_input(x)

    preds = model.predict(x)
    preds=np.argmax(preds, axis=1)
    print(preds)
    if preds==0:
        preds="The leaf is diseased cotton leaf"
    elif preds==1:
        preds="The leaf is diseased cotton plant"
    elif preds==2:
        preds="The leaf is fresh cotton leaf"
    else:
        preds="The leaf is fresh cotton plant"

    return preds


@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')


@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)

        # Make prediction
        preds = model_predict(file_path, model)
        result=preds
        return result
    return None


if __name__ == '__main__':
    app.run(debug=True,port=9989,use_reloader=False,threaded=False)
```

Results:

The performance of the model was judged based on the accuracy and loss values for train, validation and test sets. The loss on train and validation sets for 30 epochs is given below for various models.

Densenet:

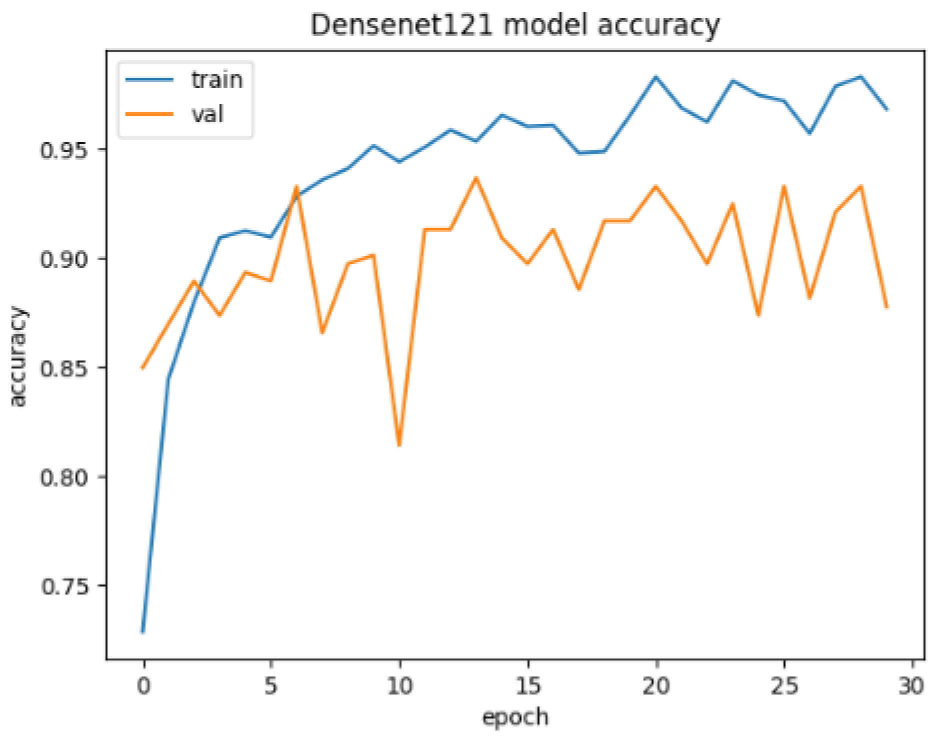


Figure7

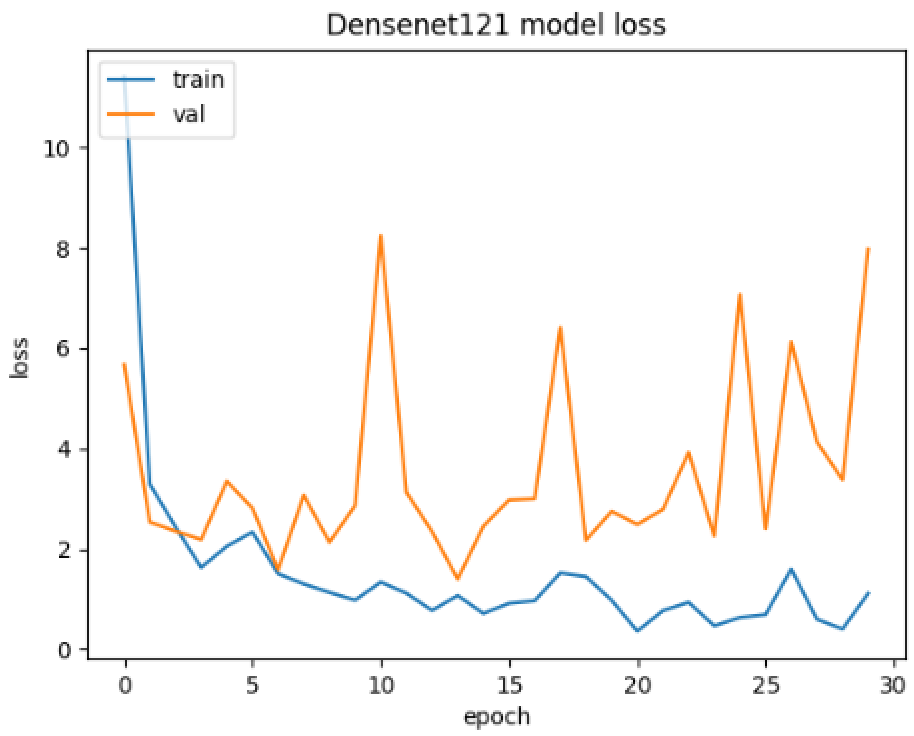
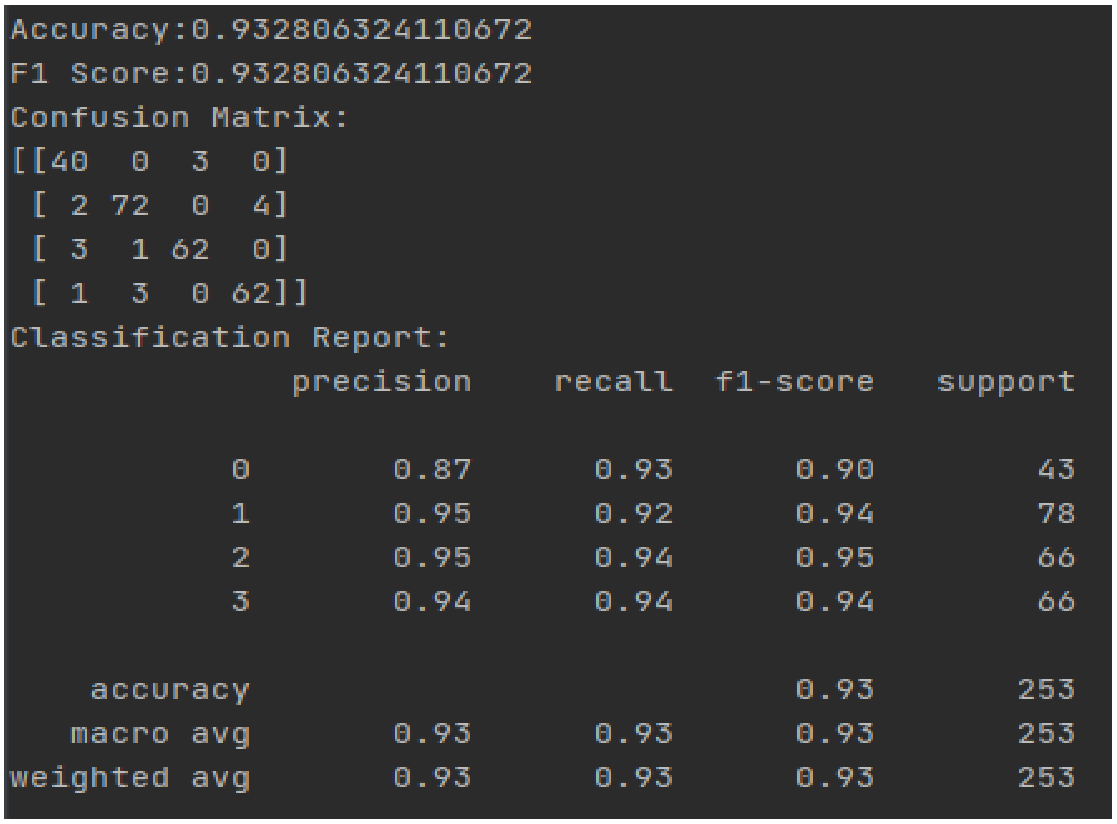


Figure8

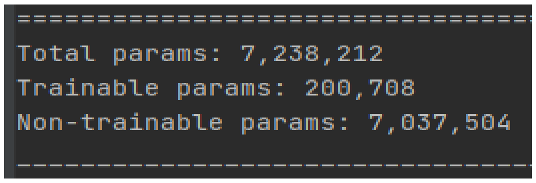


```
Accuracy:0.932806324110672
F1 Score:0.932806324110672
Confusion Matrix:
[[40  0  3  0]
 [ 2 72  0  4]
 [ 3  1 62  0]
 [ 1  3  0 62]]
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.93      0.90        43
           1       0.95      0.92      0.94        78
           2       0.95      0.94      0.95        66
           3       0.94      0.94      0.94        66

    accuracy                           0.93       253
   macro avg       0.93      0.93      0.93       253
weighted avg       0.93      0.93      0.93       253
```

Figure9



```
================================
Total params: 7,238,212
Trainable params: 200,708
Non-trainable params: 7,037,504
--------------------------------
```

Figure10

Application -

We created an app where you can upload photos of cotton plants and show whether the plant is diseased or Fresh.

This application predicts the values based on the best model of the 4 we created above. In this case it is the Custom CNN Model.

The results of the Application are shown below.

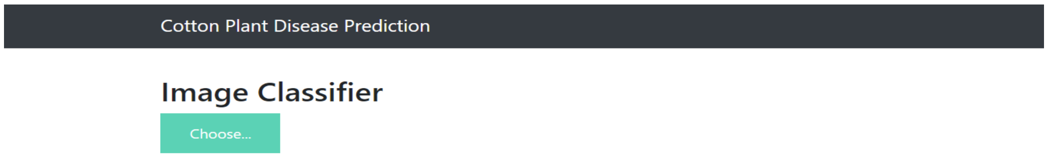


Figure11

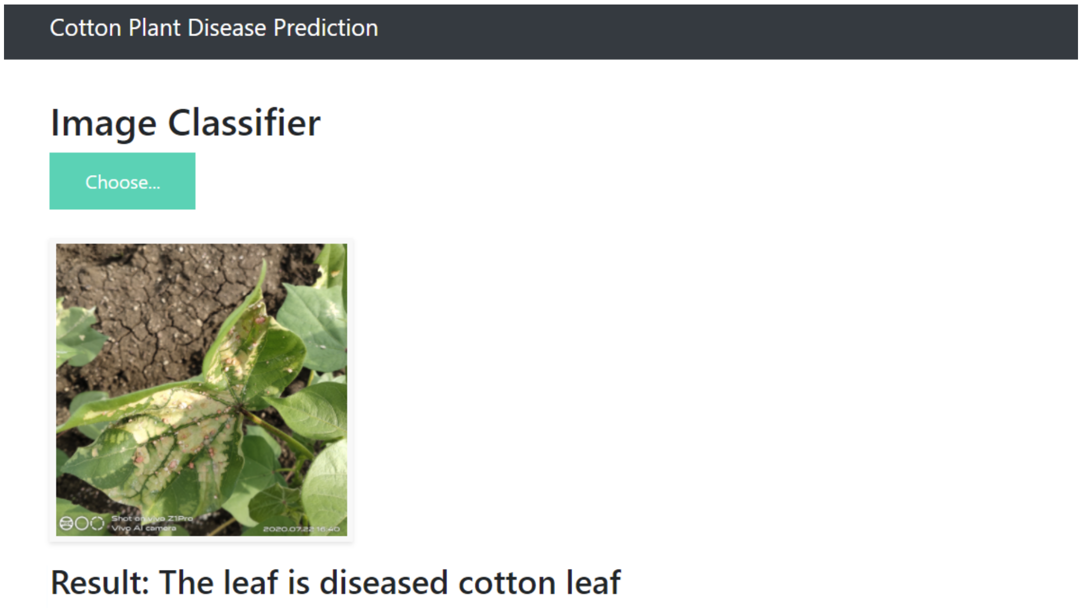

Cotton Plant Disease Prediction

Image Classifier

Result: The leaf is diseased cotton leaf

Figure12

Summary:

For Densenet50:

1. Accuracy of the model = 93.28%.

2. Overall f1-score of the model = 93.28%

3. From the classification report of models:

    1. F1 score for 0's = 0.90, f1 score for 1's = 0.94, f1 score for 2's = 0.95 and f1 score for 3's = 0.94.

    2. Precision for 0's = 0.87, Precision for 1's = 0.95, Precision for 2's = 0.95 and Precision for 3's = 0.94.

    3. Recall for 0's = 0.93, Recall for 1's = 0.92, Recall for 2's = 0.94 and Recall for 3's = 0.94.

4. From the confusion matrix we can observe which class is getting correctly predicted.

We can observe that f1-scores, precision and recall rate is high for 2's in this model.

This model has good accuracy, so we use this model for further predictions of the cotton plants images.

Conclusions:

The Densenet performed well with around 93% accuracy but it was the least accurate among all the models.

From the results we can see that the CNN model that was built performs better than the pretrained models. This can be seen by comparing the accuracy and loss functions of all the models. The manually built CNN model has 100 epochs where the closest model to that which is Resnet50 has almost the same accuracy with 30 epochs. The average training time for each of these models is approximately 1 hour.

Further work will be around interpretability of the model where one can see the features of the plant/leaf that the model thinks is important.

References:

1. https://github.com/krishnaik06/Cotton-Disease-Prediction-Deep-Learning
2. https://www.tensorflow.org/tutorials/images/data_augmentation
3. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
4. https://github.com/krishnaik06/Deployment-Deep-Learning-Model
5. https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet50/ResNet50
6. https://www.tensorflow.org/api_docs/python/tf/keras/applications/densenet/DenseNet121
7. https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg16/VGG16
8. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator