THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

# Cotton Plant Disease Prediction


Group 10

Rohan Thekanath
Nigel Martis
Ashwin Dixit


Cloud Computing

April 7th, 2022

# Motivation

- Cotton is one of the economically significant agricultural products.
- But it is exposed to different constraints in the leaf area.
- Mostly, these constraints are identified as diseases and pests that are sometimes difficult to detect.
- This study is focused to develop a model to boost the detection of cotton leaf disease and pests using the deep learning technique, CNN.

# Introduction

Diseased leaf

Diseased plant

Fresh leaf

Fresh plant

- Our project aims to identify 4 classes of labels in a cotton plant/leaf:
  1. Diseased cotton leaf
  2. Diseased cotton plant
  3. Fresh cotton leaf
  4. Fresh cotton plant

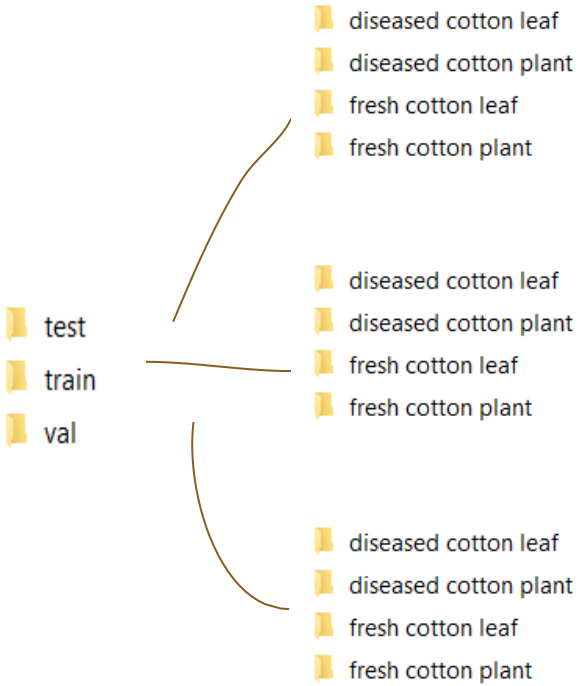
- Thus, we can efficiently recognize whether a cotton plant/leaf is infected or not and it could be treated accordingly.

# Dataset


- For our project we used Cotton Plant Disease dataset from Kaggle

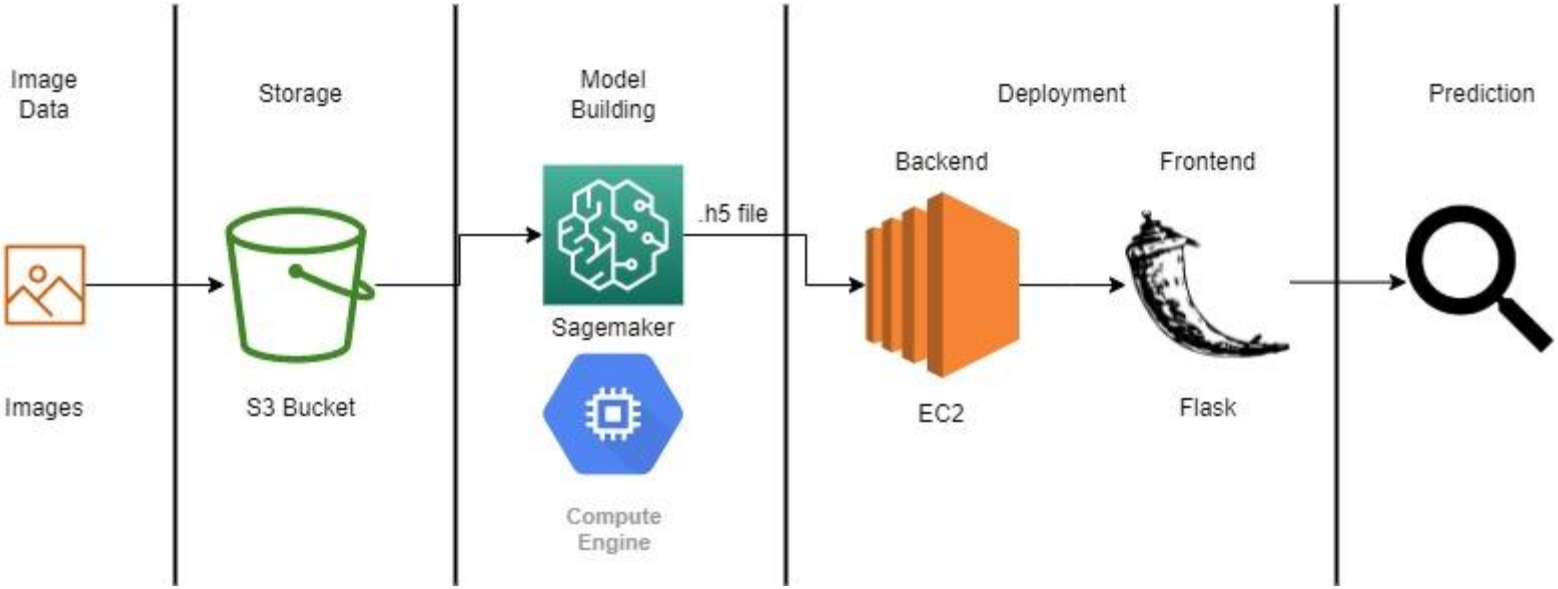
- This dataset contains hand taken pictures of cotton plants/leaves from a farm.

# Data distribution and Train-Test Split




- 160 MB of image Data
- Of the total ~2000 images used for training Custom CNN model,
  - 288 images of **diseased cotton leaves**
  - 815 images of **diseased cotton plants**
  - 427 images of **fresh cotton leaves**
  - 421 images of **fresh cotton plants**
- For the pretrained models, the image augmentations has given us nearly 12,000 images to train.

# Cloud Architecture



# Models trained:

Pre-trained networks:

1. Densenet121- 120 Convolutions and 4 AvgPool
2. VGG16- 16 layers
3. Resnet50- 50 layers

Manual Built network:

1. CNN model

```
#Building cnn model
mannual_cnn_model = keras.models.Sequential([
                                    keras.layers.Conv2D(32, 3, input_shape=[300, 300, 3]),
                                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                                    keras.layers.Conv2D(64, 3),
                                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                                    keras.layers.Conv2D(128, 3),
                                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                                    keras.layers.Conv2D(128, 3),
                                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                                    keras.layers.Conv2D(128, 3),
                                    keras.layers.MaxPooling2D(pool_size=(2,2)),

                                    keras.layers.Dropout(0.3),
                                    keras.layers.Flatten(),
                                    keras.layers.Dense(units=128, activation='relu'),
                                    keras.layers.Dropout(0.1),
                                    keras.layers.Dense(units=256, activation='relu'),
                                    keras.layers.Dropout(0.2),
                                    keras.layers.Dense(units=4, activation='softmax')
])
```

# Deep Learning Network & Fine Tuning

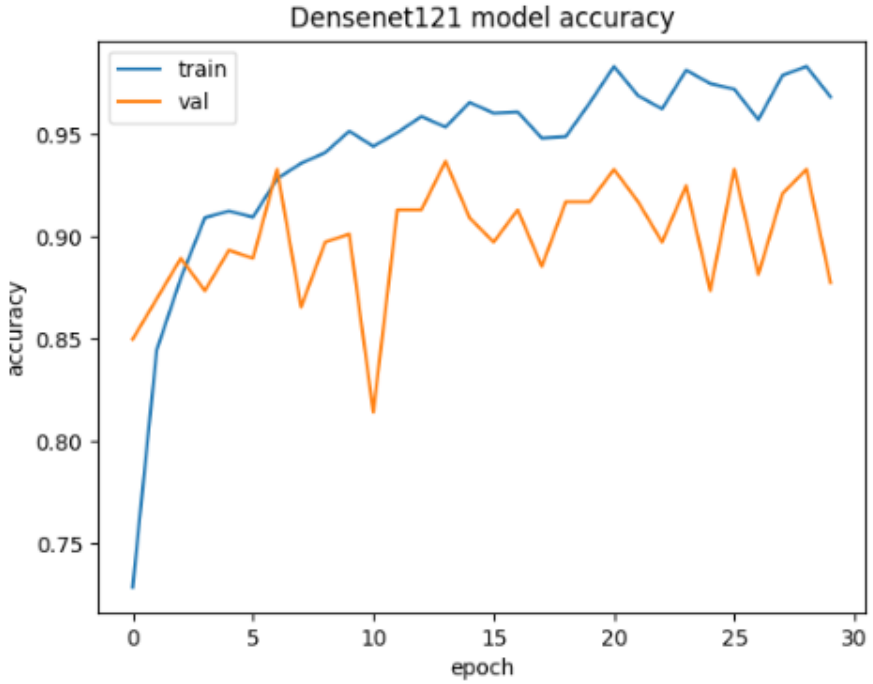
- A manual CNN model worked best for the dataset and accuracy of ~ 97% was achieved. This was compared with pretrained models like Resnet50,VGG16 and Densenet121.
- The data consisted of 3 directories: train, test and validation of which train and validation was used to create and test the model and validation was used as unseen data to check how well the model performs.

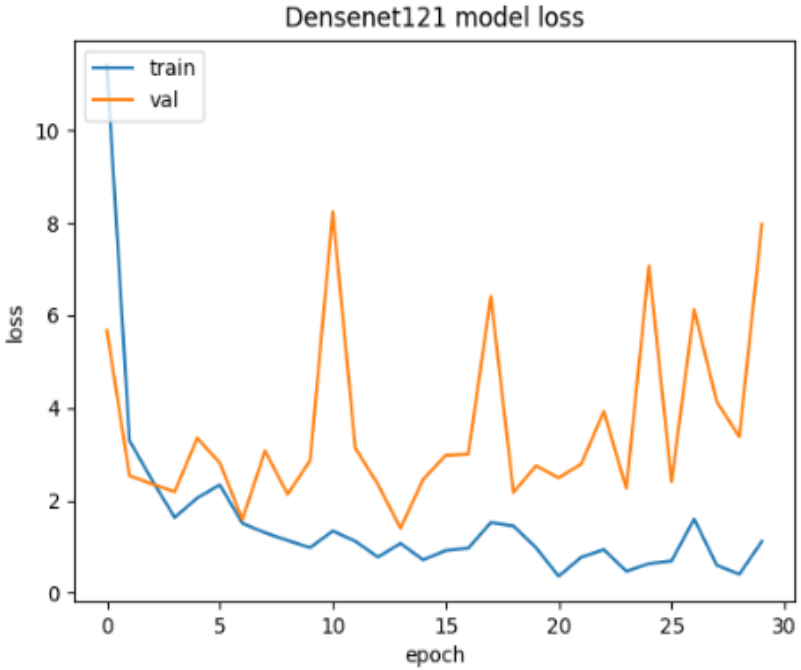
- **Transformations**
- Resize
- Horizontal Flip
- zoom
- Distortion
- Height and width movement

- **Hyperparameters**
- Batch Size - 32
- Learning Rate - 0.0001
- Epochs - 100
- Optimizer - Adam

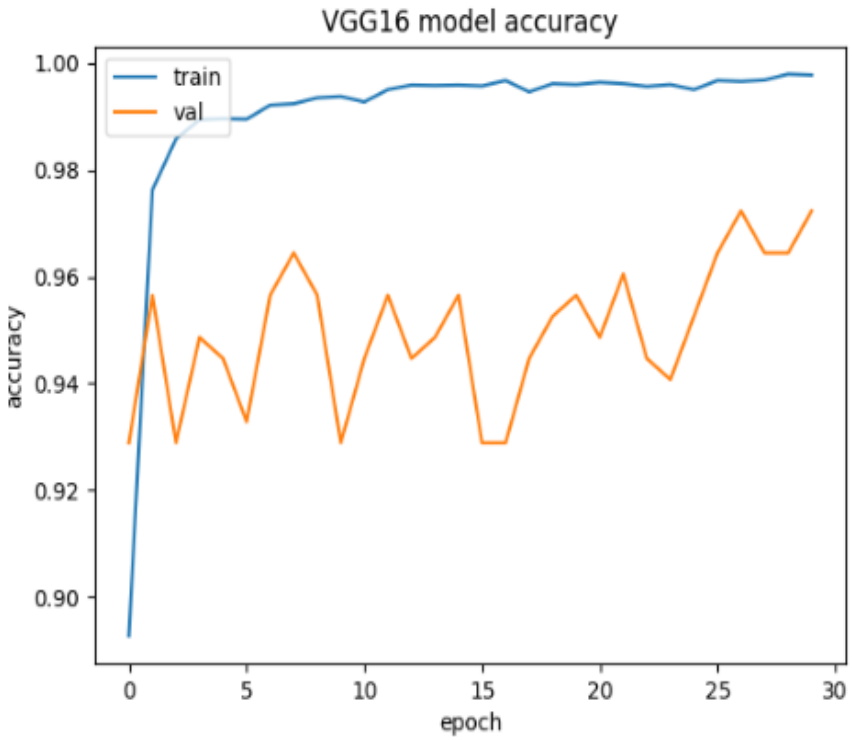
- **Regularization**
- Dropout Rate - 0.25,0,5,0.1

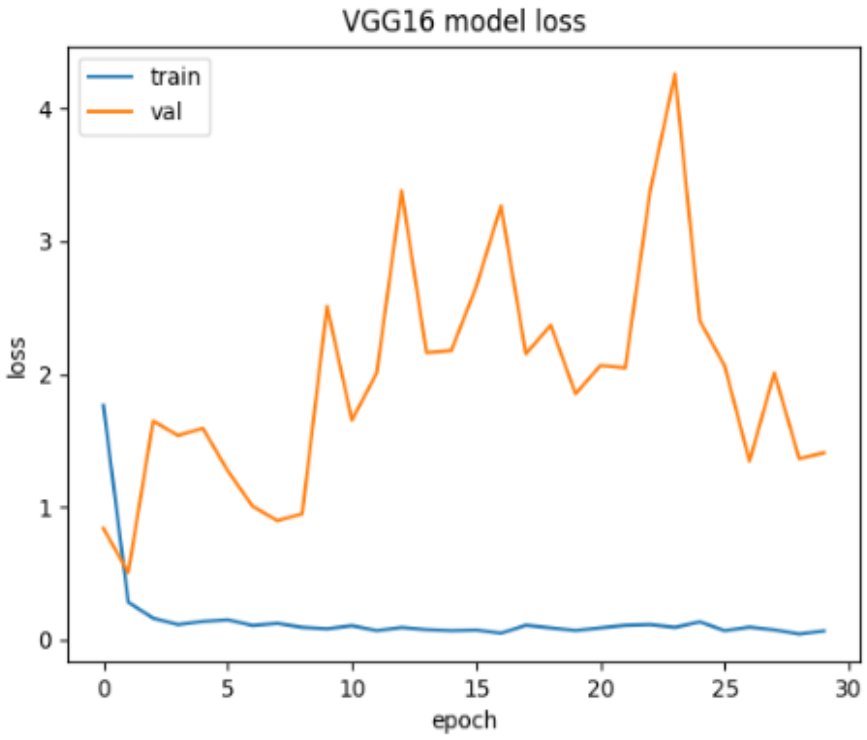
# Results: Inference for Classification



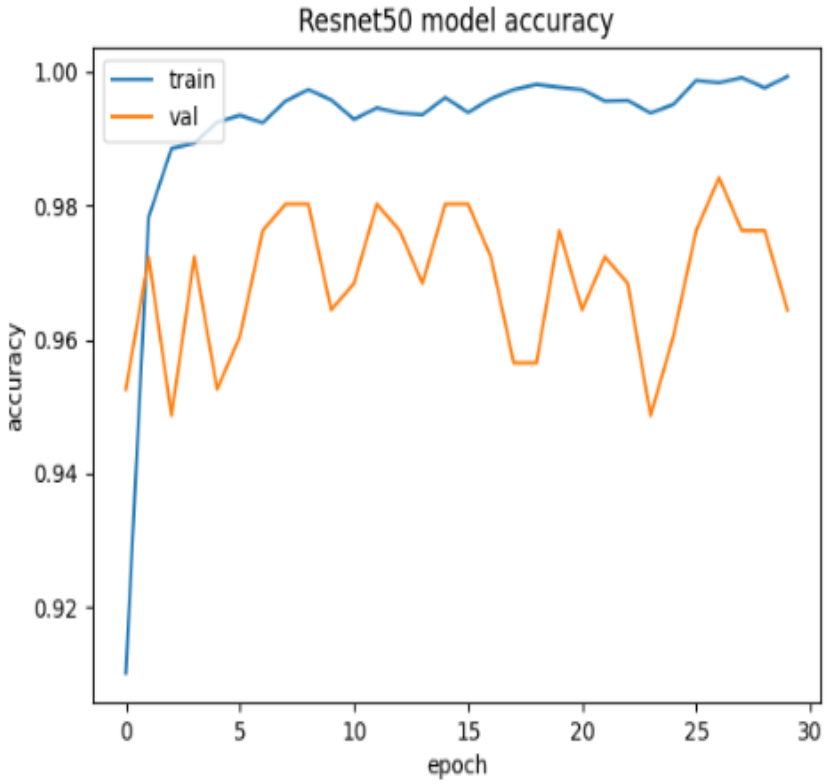



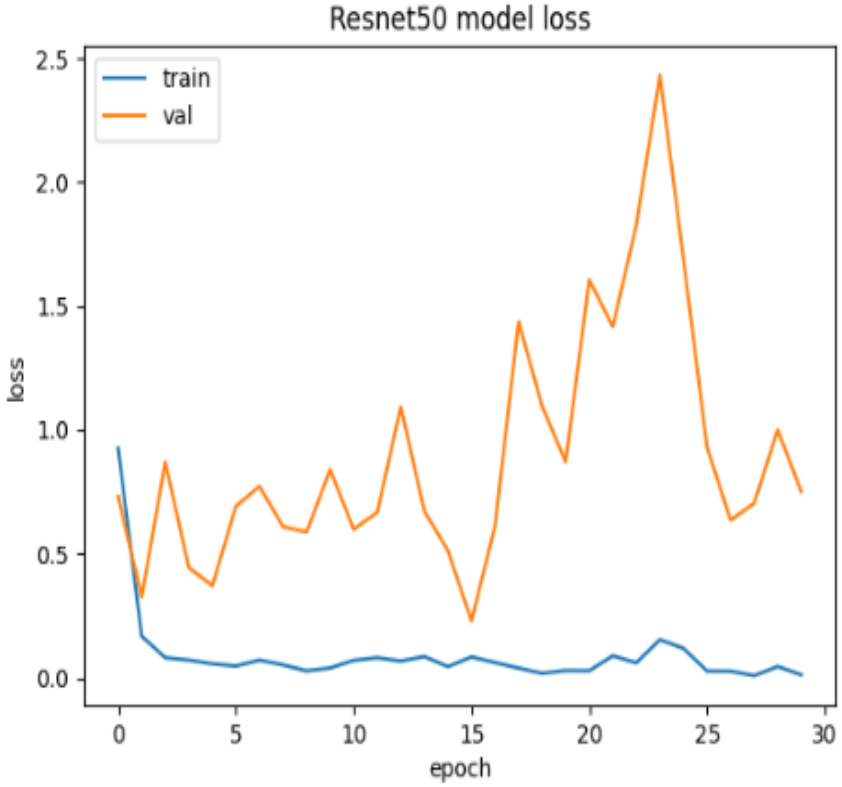
# Results: Inference for Classification



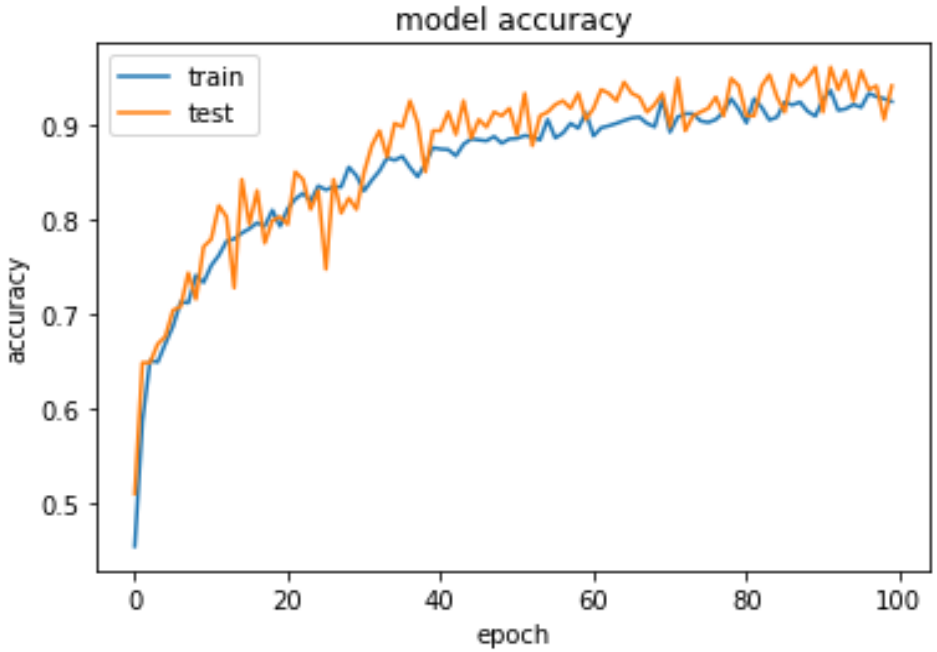



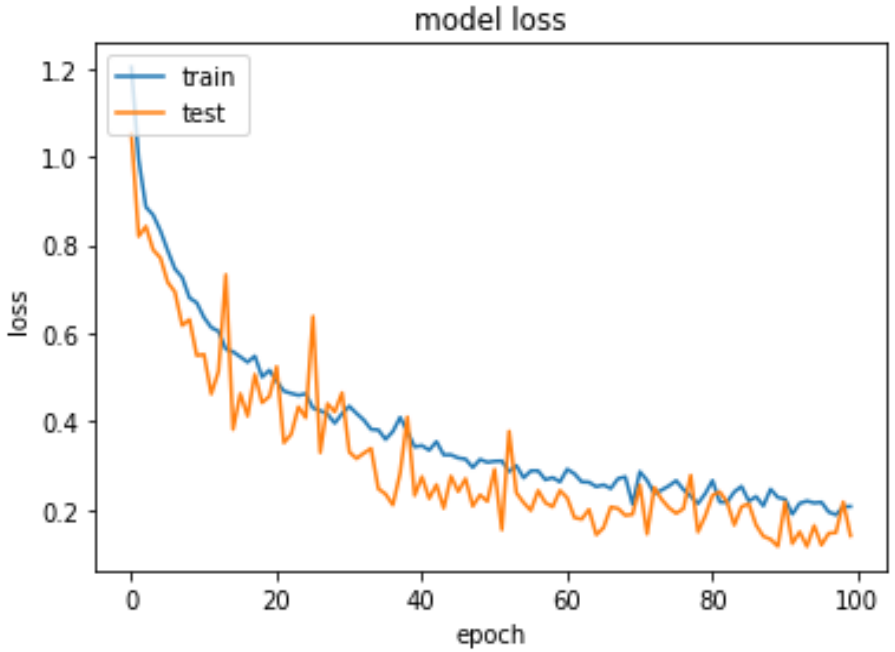
# Results: Inference for Classification



# Results: Inference for Classification



# Results: Inference for Classification

Confusion matrix along with other F1 scores of each class of the best model( Manual Cnn)

```
Accuracy:0.9723320158102767
F1 Score:0.9723320158102767
Confusion Matrix:
[[41  1  1  0]
 [ 0 77  1  0]
 [ 0  0 66  0]
 [ 0  4  0 62]]
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      0.95      0.98        43
         1.0       0.94      0.99      0.96        78
         2.0       0.97      1.00      0.99        66
         3.0       1.00      0.94      0.97        66

    accuracy                           0.97       253
   macro avg       0.98      0.97      0.97       253
weighted avg       0.97      0.97      0.97       253
```

# Model Deployment

Cotton Plant Disease Prediction

Image Classifier

Choose...

Cotton Plant Disease Prediction

Image Classifier

Choose...

Result: The leaf is diseased cotton leaf

# Summary & Conclusion

- A cotton disease detector model was developed to detect if a plant/leaf falls under categories like diseased leaf, diseased plant, fresh leaf or fresh cotton plant.

- Pre-trained models like Resnet50,VGG16, Densenet121 was compared with a manually trained CNN model.

- Accuracy achieved was about ~97% and the model also performs very well on unseen plant/leaf images.

- This model has also been deployed where the user can choose an image and see its result.

# Future Work

- Future work would consist of adding model interpretability into the project where one can visually see how the model classifies the images.

- A mobile app could also be made where the farmer can take pictures of the plant/leaf to identify if it is diseased on not. The app would also display the solution including how and which fertilizers to use.

# THANK YOU

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
9.https://www.kaggle.com/anuragupadhyay6212/cotton-disease-prediction-using-vgg16-and-renet50