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

Rohan Paul Thekanath
The George Washington University
6th December 2021

## 1. 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.

## 2. 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.

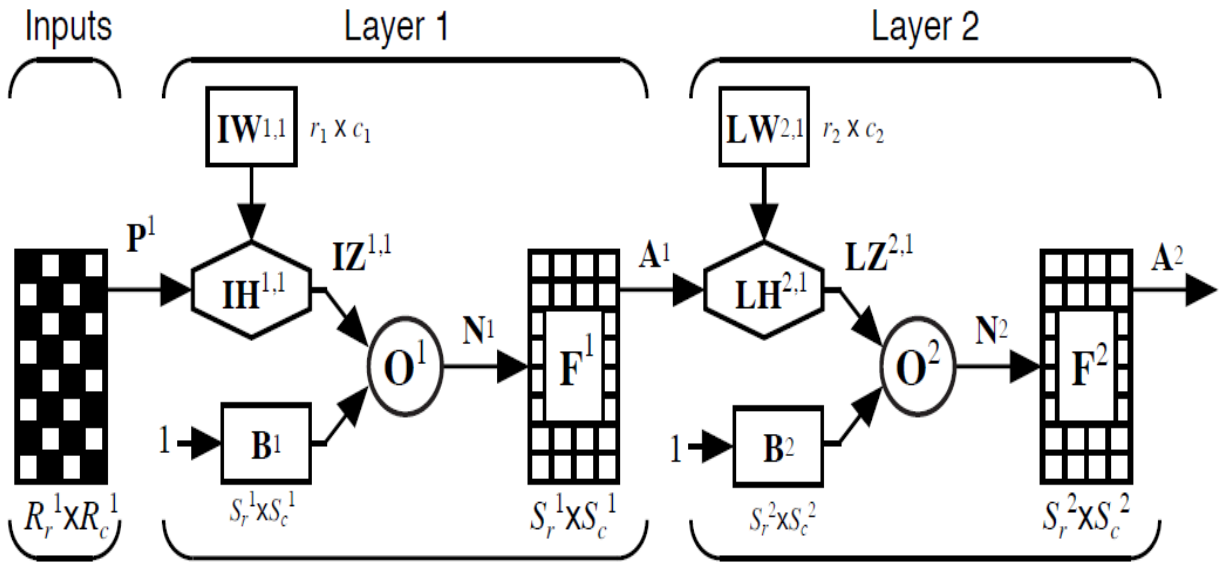



Figure7 : convolution network

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.

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

Figure8

In matrix form we write it as:

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

**Figure9**

My Individual work consisted of creating a manual CNN model and finding parameters that best fit the data. I also helped in understanding and building the GUI/deployment part for this project.

The below CNN architecture is what was finalized

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 298, 298, 32)      896
```

```
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 149, 149, 32)      0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 147, 147, 64)      18496
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 73, 73, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 71, 71, 128)       73856
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 35, 35, 128)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 33, 33, 128)       147584
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 128)       147584
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
dropout_6 (Dropout)          (None, 7, 7, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               802944
_________________________________________________________________
dropout_7 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 256)               33024
_________________________________________________________________
dropout_8 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 4)                 1028
=================================================================
Total params: 1,225,412
Trainable params: 1,225,412
Non-trainable params: 0
```

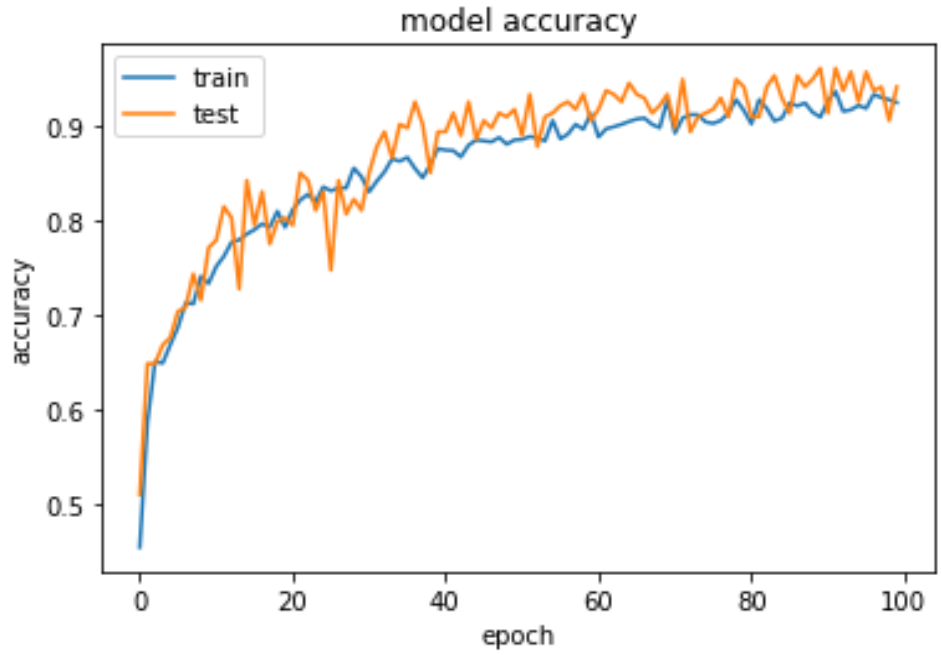
**3.** The code that was used to create this is attached below

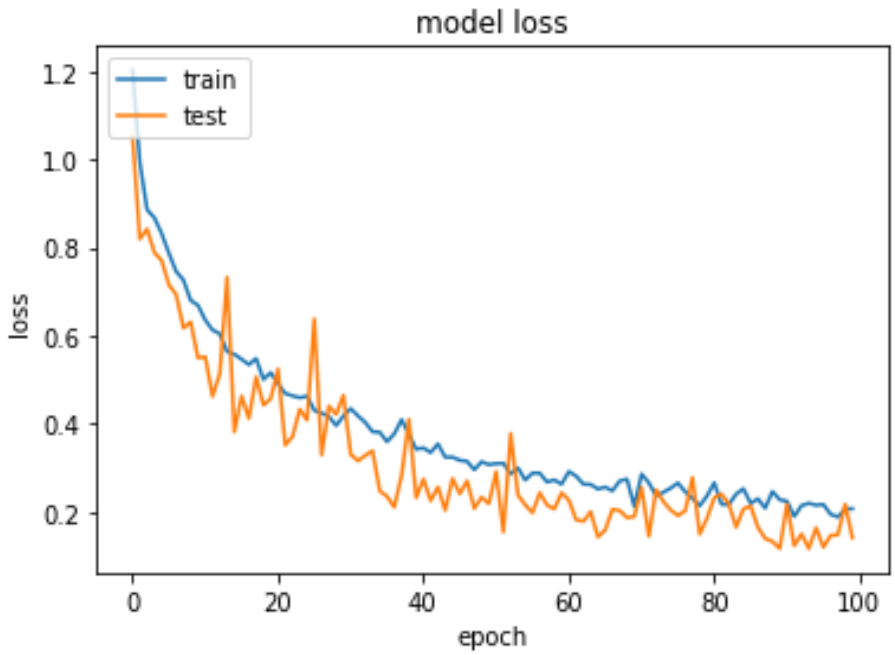
mannual_cnn_code.py

**4. Results**

**For Custom CNN model**:

- Accuracy of the model = ~97%.
- Overall f1-score of the model = ~97%
- From the classification report of models:
    - F1 score for 0's = 0.95, f1 score for 1's = 0.97, f1 score for 2's = 0.97 and f1 score for 3's = 0.97.
    - Precision for 0's = 0.93, Precision for 1's = 0.99, Precision for 2's = 0.97 and Precision for 3's = 0.97.
- Recall for 0's = 0.98, Recall for 1's = 0.96, Recall for 2's = 0.97 and Recall for 3's = 0.97.
- From the confusion matrix we can observe which class is getting correctly predicted.






The accuracy of this model increases in both train and test as the epochs increases which shows that there is no overfitting. The graph which shows the model loss also shows that the loss is decreasing in both train and test in a steady manner.

**5.** The manually created CNN model performed the best as compared to the pretrained models. To achieve this, we had to increase the epochs to 100 as compared to 30 epochs used in the pretrained models. The F1 score and accuracy is ~97% for this model. This model could be further optimized by using a grid search for finding optimal parameters.

**6.** 20% of the code was used as reference from the internet

**7.References:**

https://github.com/krishnaik06/Cotton-Disease-Prediction-Deep-Learning

https://www.tensorflow.org/tutorials/images/data_augmentation

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

https://github.com/krishnaik06/Deployment-Deep-Learning-Model