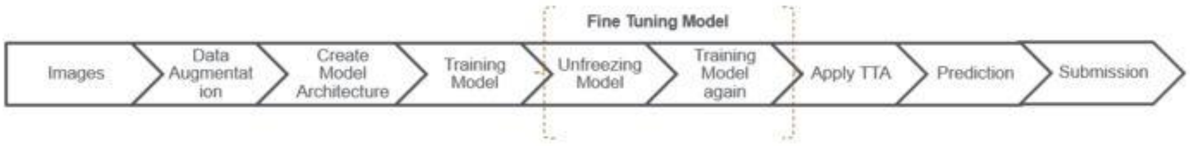
Identification of Infected Crop

Project Overview

The objective of our project is to make a real-world application for identification of infected crops by the images of their leaves. Our model uses Deep Learning algorithm to identify the infected crops. *It has tremendous promise for improving crop monitoring at scale. We present our learnings from building such models for detecting infected crops.*

Methodology

- The state-of-the-art approach for determining a plant's health based on pictures is a type of deep learning called Convolutional Neural Network (CNN).
- The overall modelling process required several steps for effectively preparing the data for the CNN model to yield a good result.



- Data Augmentation particularly played a significant role in increasing the model performance. We used multiple data augmentation techniques -illustrated below: -
    1. Vertical and Horizontal flip
    2. Light Standardization
    3. Zoom and Crop
- We tuned the model, along with different data augmentation methods iteratively before we achieved our final performance

Software Bundles:

- ✓ Google Collab (IDE)
- ✓ Jupyter Notebook
- ✓ GitHub
- ✓ VS Code
- ✓ Python Language
- ✓ Packages used-
    - o 1.Keras
    - o 2.Tenserflow
    - o 3.Matplotlib
- ✓ Streamlit

Implementation:

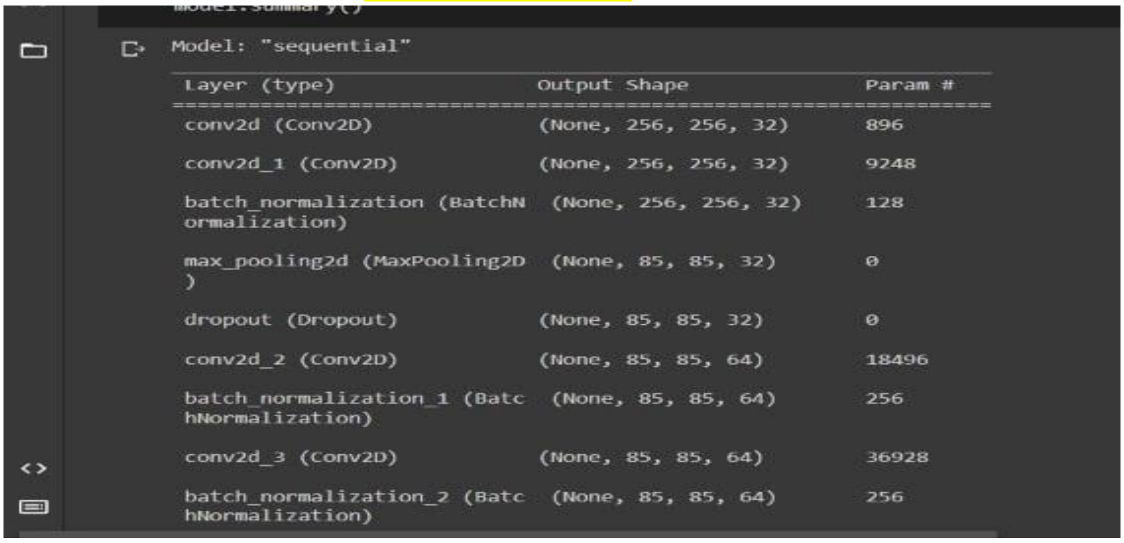
- Creating virtual environment in google collab
- Installed all required libraries in the virtual environment.
- Imported required libraries in google collab.

Model creation:

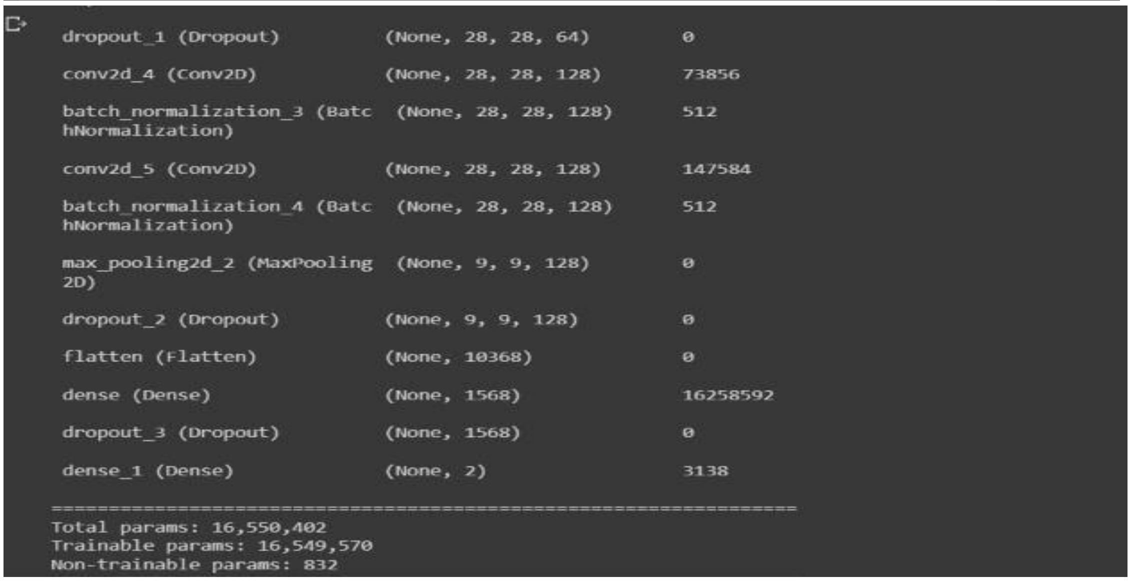
- We used Keras to create our sequential model of Convolutional Neural Network (CNN).
- Used many layers of convolution, pooling, batch normalization, flatten and dropout. Further proceeded by dense layers which will produce output.
- The input size of image is 256 X 256 X 3.
- To decrease the number of training epochs required to train the deep neural networks,
- we used ‘batch normalization’ technique that normalizes the contributions to a layer for every mini batch.
- For transforming the summed weighted input from the node into the activation of the node or output for that input we use ReLU activation function.

- ‘Same’ padding is used to the input image so that the input image gets fully covered by the filter and specified stride.
- ‘SoftMax’ activation function is used normalize the outputs, converting them from weighted sum values into probabilities that sum to one.
- Each value in the output of the SoftMax function is interpreted as the probability of membership for each class.
- We have 70295 images for training set data and 17572 images for cross validation.

Model Summary



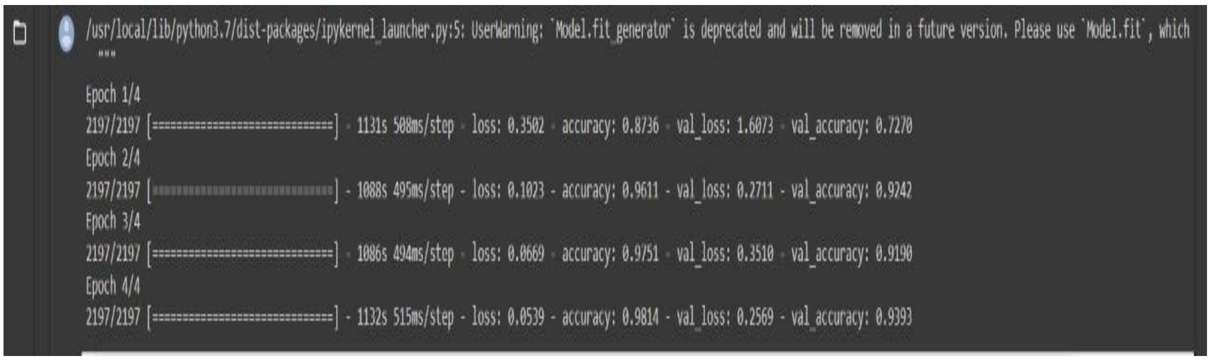
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 256, 256, 32)      896

 conv2d_1 (Conv2D)           (None, 256, 256, 32)      9248

 batch_normalization (BatchN  (None, 256, 256, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 85, 85, 32)       0
 )

 dropout (Dropout)           (None, 85, 85, 32)        0

 conv2d_2 (Conv2D)           (None, 85, 85, 64)        18496

 batch_normalization_1 (Batc  (None, 85, 85, 64)       256
 hNormalization)

 conv2d_3 (Conv2D)           (None, 85, 85, 64)        36928

 batch_normalization_2 (Batc  (None, 85, 85, 64)       256
 hNormalization)
```



```
 dropout_1 (Dropout)         (None, 28, 28, 64)        0

 conv2d_4 (Conv2D)           (None, 28, 28, 128)       73856

 batch_normalization_3 (Batc  (None, 28, 28, 128)      512
 hNormalization)

 conv2d_5 (Conv2D)           (None, 28, 28, 128)       147584

 batch_normalization_4 (Batc  (None, 28, 28, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 9, 9, 128)        0
 2D)

 dropout_2 (Dropout)         (None, 9, 9, 128)         0

 flatten (Flatten)           (None, 10368)             0

 dense (Dense)               (None, 1568)              16258592

 dropout_3 (Dropout)         (None, 1568)              0

 dense_1 (Dense)             (None, 2)                 3138

=================================================================
Total params: 16,550,402
Trainable params: 16,549,570
Non-trainable params: 832
```

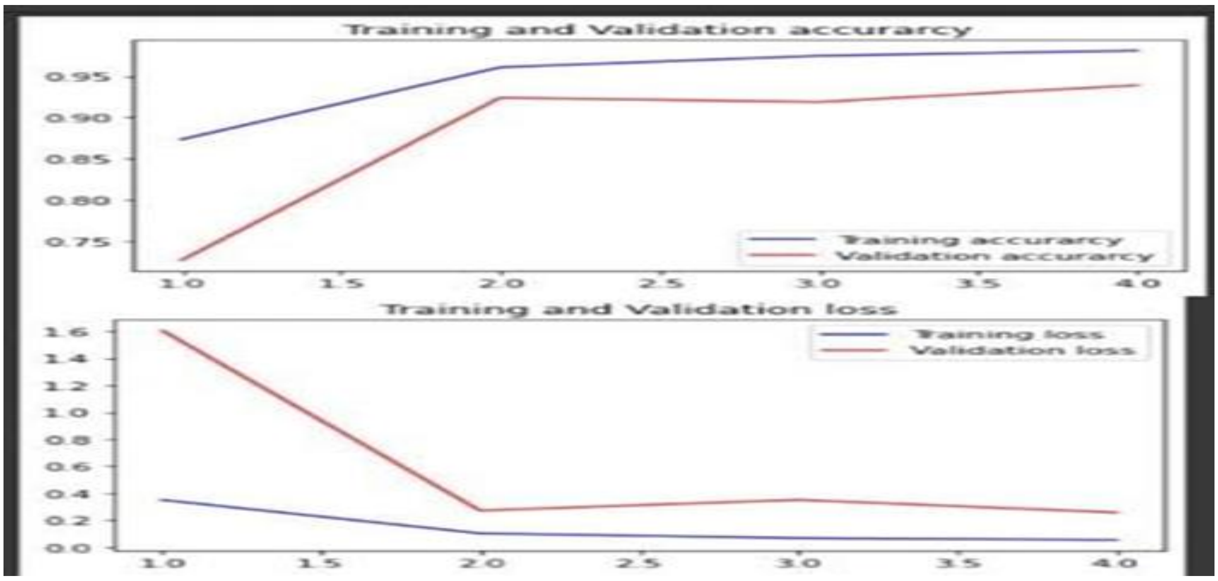
Result:

- Total Time taken for fitting the model for 4 iterations: 1131+1088+1086+1132 = 73.95 minutes



```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which
  """
Epoch 1/4
2197/2197 [==============================] - 1131s 508ms/step - loss: 0.3502 - accuracy: 0.8736 - val_loss: 1.6073 - val_accuracy: 0.7270
Epoch 2/4
2197/2197 [==============================] - 1088s 495ms/step - loss: 0.1023 - accuracy: 0.9611 - val_loss: 0.2711 - val_accuracy: 0.9242
Epoch 3/4
2197/2197 [==============================] - 1086s 494ms/step - loss: 0.0669 - accuracy: 0.9751 - val_loss: 0.3510 - val_accuracy: 0.9190
Epoch 4/4
2197/2197 [==============================] - 1132s 515ms/step - loss: 0.0539 - accuracy: 0.9814 - val_loss: 0.2569 - val_accuracy: 0.9393
```

Graph for accuracy and loss for training and validation



- Training Accuracy: 98.14%
- Validation Accuracy: 93.93%

Timeline:

- Week-1(20/05/2022): While commencement, we only have intuition of the basic ml algorithm. After discussion with our mentors, we started to focus on python syntax and started experimenting with google collab.

- Week-2(27/05/2022): After getting grip in language, in this timeframe we had covered neural networks, CNN algorithm & little bit about ANN. We had finalized our data set for our model.
- Week-3(05/06/2022): In this session, we started to work for implementation of CNN. We travel over various python library and come to know about Keras library for building the model.
- Week-4&5(12/06/2022): After setting up the model, we had faced difficulties like overfitting and class imbalance error. After going through various articles, video tutorial we had covered those errors, by increasing overall accuracy.
- Week-6&7(26/07/2022): End Sem preparation
- Week-8(02/07/2022): Since the final objective was to make a real-world application. We decided to make the web-app by using Streamlit python library.

Web App link:

- ✓

REFERENCES:

- https://www.kaggle.com/code/emmarex/plant-disease-detection-using-keras
- https://www.kdnuggets.com/2020/06/crop-disease-detection-computer-vision.html
- https://github.com/farhad324/Auto-Chloro-A-Crop-Disease-Classifier-and-Remedies-Provider-In-Bangla