Plant Disease classification using Deep Learning
Team #8

Mahesh Mandava
Adhisekhar Raju Uppalapati
Sudheer Nimmagadda

1. Motivation:

We depend on edible plants just as we depend on oxygen. Without crops, there is no food, and without food, there is no life. It's no accident that human civilization began to thrive with the invention of agriculture.

Today, modern technology allows us to grow crops in quantities necessary for a steady food supply for billions of people. But diseases remain a major threat to this supply, and a large fraction of crops are lost each year to diseases. The situation is particularly dire for the 500 million smallholder farmers around the globe, whose livelihoods depend on their crops doing well. In Africa alone, 80% of the agricultural output comes from smallholder farmers.

With billions of smartphones around the globe, wouldn't it be great if the smartphone could be turned into a disease diagnostics tool, recognizing diseases from images it captures with its camera?

2. Project Idea:

Using CNN, Alexnet on a public data set by plant village for plant diseases classification.

First we collect data that is publicly available by plant village.

Second we do image pre processing is used to normalize the images of the dataset.

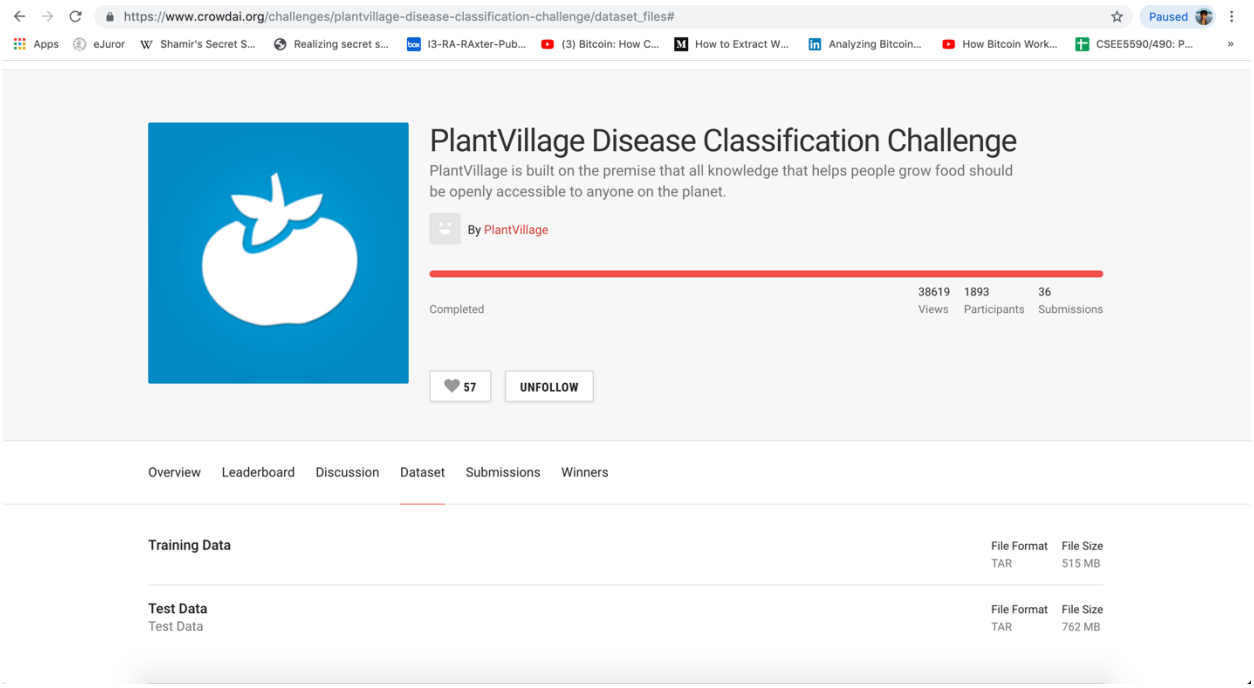
The most used techniques are image resizing.

Third, we train the data using CNN.

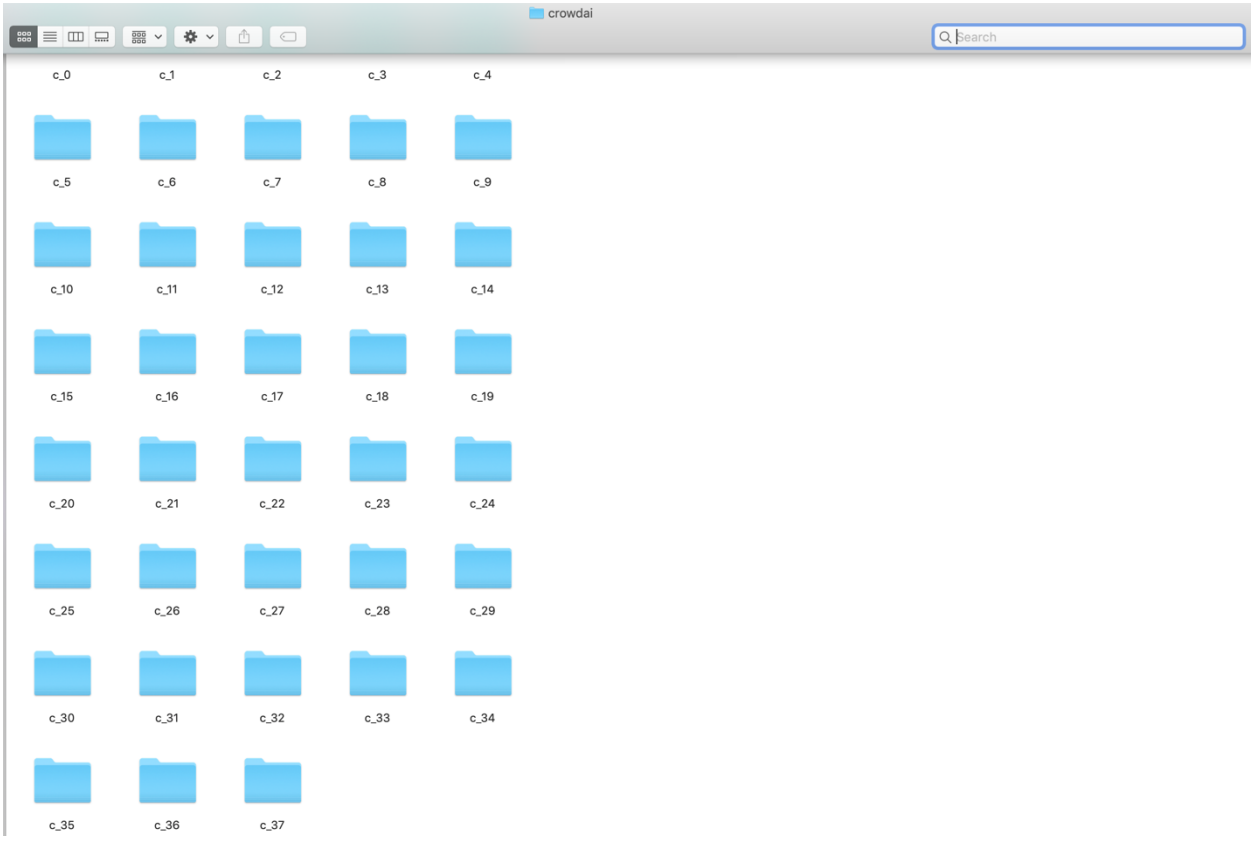
Fourth the model is deployed where the output of the network determines which diseases are present in the plant.

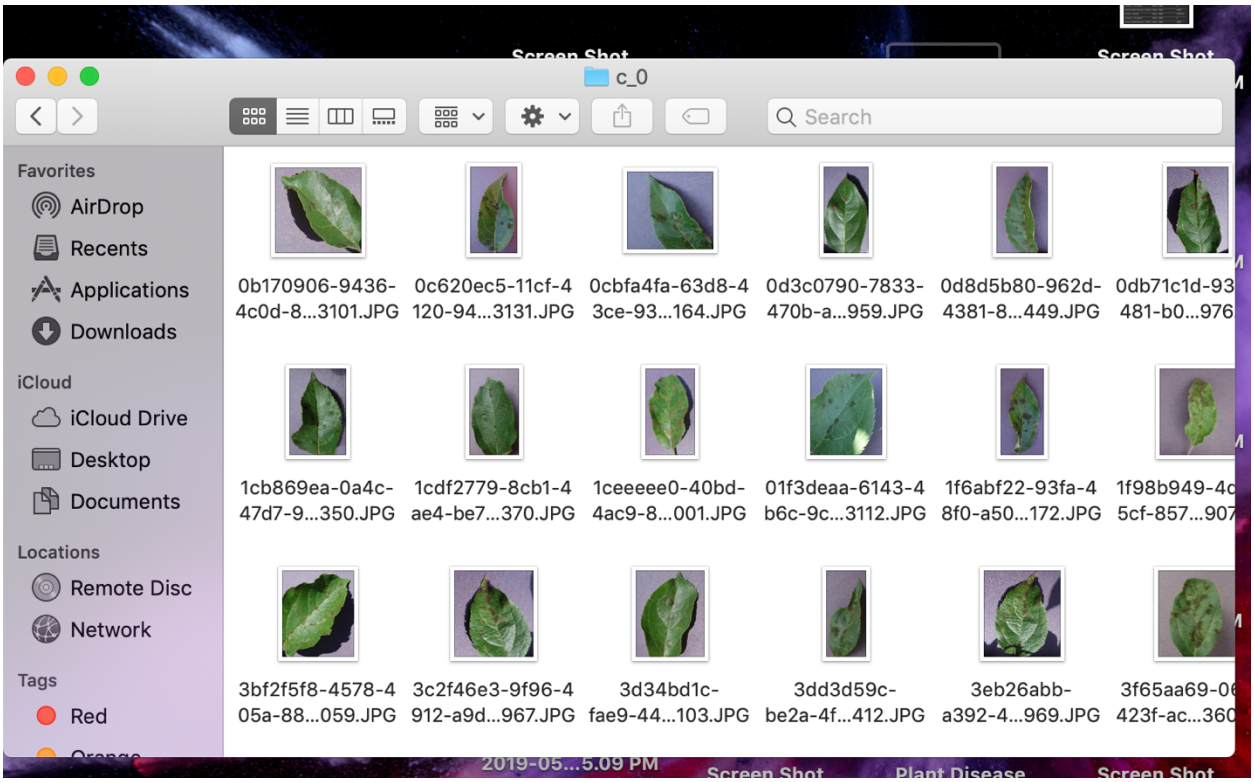
Below are 38 classes of crop disease pairs that the dataset is offering:

Dataset is downloaded from crowdai.org maintained by plant village.



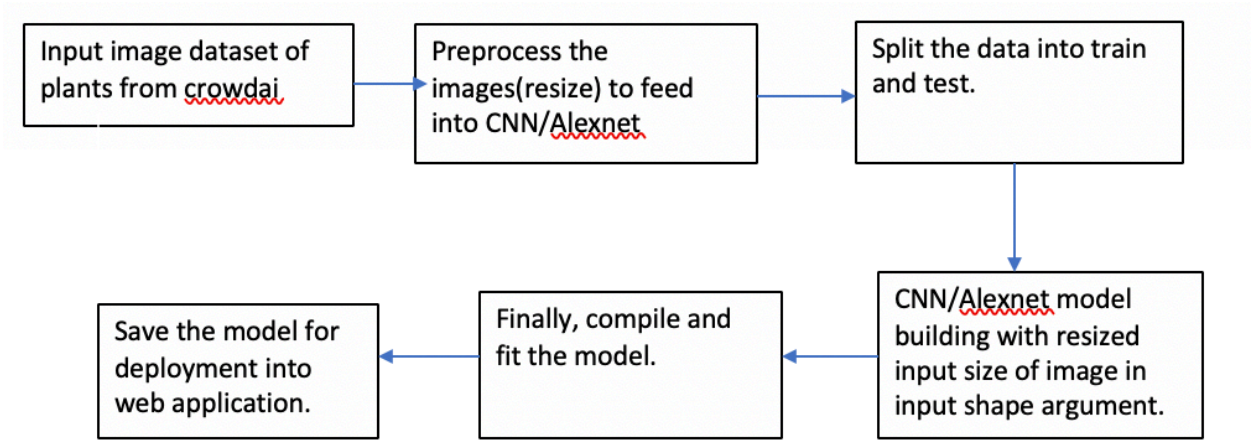
Below are the snapshots of our dataset.





3. System Architecture:

Basic Block Diagram:



CNN architecture in our project:

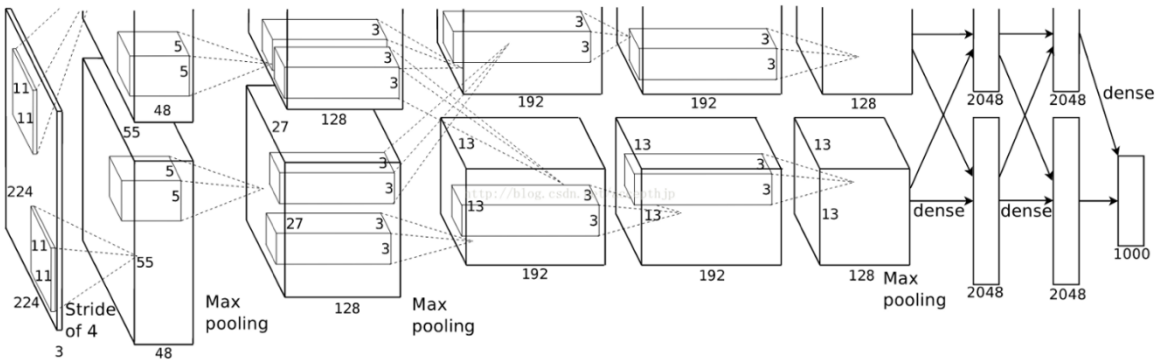
For CNN the image input shape is given as (28,28) and the below code represents the architecture of CNN.

```
model = Sequential()

model.add(Convolution2D(32,3,3,input_shape = (28,28,3)))
model.add(Activation('relu'))

model.add(Convolution2D(64,3,3))
model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Convolution2D(16,3,3))
model.add(Activation('relu'))

model.add(Flatten())

model.add(Dropout(0.2))
model.add(Dense(num_classes))

model.add(Activation('softmax'))

model.summary()

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Below is the summary of CNN:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        896
_________________________________________________________________
activation_1 (Activation)    (None, 26, 26, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 24, 24, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 16)        9232
_________________________________________________________________
activation_3 (Activation)    (None, 10, 10, 16)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1600)              0
_________________________________________________________________
dense_1 (Dense)              (None, 38)                60838
_________________________________________________________________
activation_4 (Activation)    (None, 38)                0
=================================================================
Total params: 89,462
Trainable params: 89,462
Non-trainable params: 0
_________________________________________________________________
```

Alexnet architecture in our project:



Similar structure to LeNet, AlexNet has more filters per layer, deeper and stacked. There are 5 convolutional layers, 3 fully connected layers and with Relu applied after each of them, and dropout applied before the first and second fully connected layer.

Below is the code for alexnet architecture.

```
# Initializing the CNN
classifier = Sequential()

# Convolution Step 1
classifier.add(Convolution2D(96, 11, strides = (4, 4), padding = 'valid', input_shape=(224, 224,

# Max Pooling Step 1
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding = 'valid'))
classifier.add(BatchNormalization())

# Convolution Step 2
classifier.add(Convolution2D(256, 11, strides = (1, 1), padding='valid', activation = 'relu'))

# Max Pooling Step 2
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding='valid'))
classifier.add(BatchNormalization())

# Convolution Step 3
classifier.add(Convolution2D(384, 3, strides = (1, 1), padding='valid', activation = 'relu'))
classifier.add(BatchNormalization())

# Convolution Step 4
classifier.add(Convolution2D(384, 3, strides = (1, 1), padding='valid', activation = 'relu'))
classifier.add(BatchNormalization())

# Convolution Step 5
classifier.add(Convolution2D(256, 3, strides=(1,1), padding='valid', activation = 'relu'))

# Max Pooling Step 3
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding = 'valid'))
classifier.add(BatchNormalization())

# Flattening Step
classifier.add(Flatten())
```

```
# Full Connection Step
classifier.add(Dense(units = 4096, activation = 'relu'))
classifier.add(Dropout(0.4))
classifier.add(BatchNormalization())
classifier.add(Dense(units = 4096, activation = 'relu'))
classifier.add(Dropout(0.4))
classifier.add(BatchNormalization())
classifier.add(Dense(units = 1000, activation = 'relu'))
classifier.add(Dropout(0.2))
classifier.add(BatchNormalization())
classifier.add(Dense(units = 38, activation = 'softmax'))
classifier.summary()
```

we chose to train the top 2 conv blocks, i.e. we will freeze the first 8 layers and unfreeze the rest:

The code is as follows:

```
for i, layer in enumerate(classifier.layers[:20]):
    print(i, layer.name)
    layer.trainable = False
classifier.summary()

classifier.compile(optimizer=optimizers.SGD(lr=0.001, momentum=0.9, decay=0.005),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Below is the summary of Alexnet:

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 54, 54, 96)	34944
max_pooling2d_1 (MaxPooling2	(None, 27, 27, 96)	0
batch_normalization_1 (Batch	(None, 27, 27, 96)	384
conv2d_2 (Conv2D)	(None, 17, 17, 256)	2973952
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 256)	0
batch_normalization_2 (Batch	(None, 8, 8, 256)	1024
conv2d_3 (Conv2D)	(None, 6, 6, 384)	885120
batch_normalization_3 (Batch	(None, 6, 6, 384)	1536
conv2d_4 (Conv2D)	(None, 4, 4, 384)	1327488
batch_normalization_4 (Batch	(None, 4, 4, 384)	1536
conv2d_5 (Conv2D)	(None, 2, 2, 256)	884992
max_pooling2d_3 (MaxPooling2	(None, 1, 1, 256)	0
batch_normalization_5 (Batch	(None, 1, 1, 256)	1024
flatten_1 (Flatten)	(None, 256)	0
dense_1 (Dense)	(None, 4096)	1052672
dropout_1 (Dropout)	(None, 4096)	0
batch_normalization_6 (Batch	(None, 4096)	16384
dense_2 (Dense)	(None, 4096)	16781312
dropout_2 (Dropout)	(None, 4096)	0
batch_normalization_7 (Batch	(None, 4096)	16384

```
batch_normalization_7 (Batch (None, 4096)              16384
_________________________________________________________________
dense_3 (Dense)              (None, 1000)              4097000
_________________________________________________________________
dropout_3 (Dropout)          (None, 1000)              0
_________________________________________________________________
batch_normalization_8 (Batch (None, 1000)              4000
_________________________________________________________________
dense_4 (Dense)              (None, 38)                38038
=================================================================
Total params: 28,117,790
Trainable params: 4,137,038
Non-trainable params: 23,980,752
_________________________________________________________________
WARNING:tensorflow:From /Library/Frameworks/Python.framework/Version
Instructions for updating:
Use tf.cast instead.
Train on 13150 samples, validate on 8767 samples
Epoch 1/25
```

This is how our both CNN and alexnet models are built in our project.

4. Technical Stack:

We used keras for neural network model building and tensor board for viewing the accuracy and loss graphs. And we also used scientific packages like scikit-learn, numpy, matplotlib. And finally we deployed our neural network model in the web application developed using flask (a micro web framework written in python). And also we used Pycharm IDE.

5. Implementation:

CNN code snippets:

Below is the snippet for resizing the target image

```
import numpy as np
import keras
from keras.preprocessing import image
from keras.layers import MaxPooling2D,Convolution2D,Dropout, Flatten,Dense,Activation
from keras.models import Sequential,save_model
from keras.utils import np_utils
import os
import cv2
from sklearn.utils import shuffle

path = '/Users/adisekharrajuuppalapati/Downloads/crowdai'

leaf = os.listdir(path)
print(len(leaf),type(leaf))

print(leaf)

print(leaf[0][2:4])

x,y = [], []

for i in leaf:
    images = os.listdir(path+"/"+i)
    for j in images:
        img_path = path+"/"+i+"/"+j
        #Better method then cv2.imread
        img = image.load_img(img_path, target_size=(28,28))
        img = image.img_to_array(img)
        #print(img.shape)
        #img =img.flatten()
        #img = img.reshape(1,784)
#         print(img.shape)
#         img = img.reshape((28,28))
        img = img/255.0
        x.append(img)
        y.append(int(i[2:4]))
```

Below is the code snippet for converting to binary class matrix using to_categorical and also splitting data into train and test.

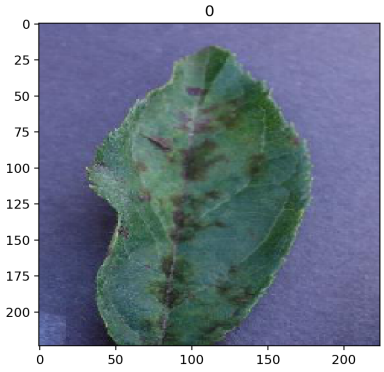
```
y_data = np_utils.to_categorical(y_data)
print(y_data.shape)
num_classes = y_data.shape[1]
print(num_classes)

x_data, y_data = shuffle(x_data,y_data, random_state=0)

split = int(0.6*(x_data.shape[0]))

x_train = x_data[:split]
x_test = x_data[split:]
y_train = y_data[:split]
y_test = y_data[split:]
```

CNN architecture code snippets are in the 3.System architecture part.

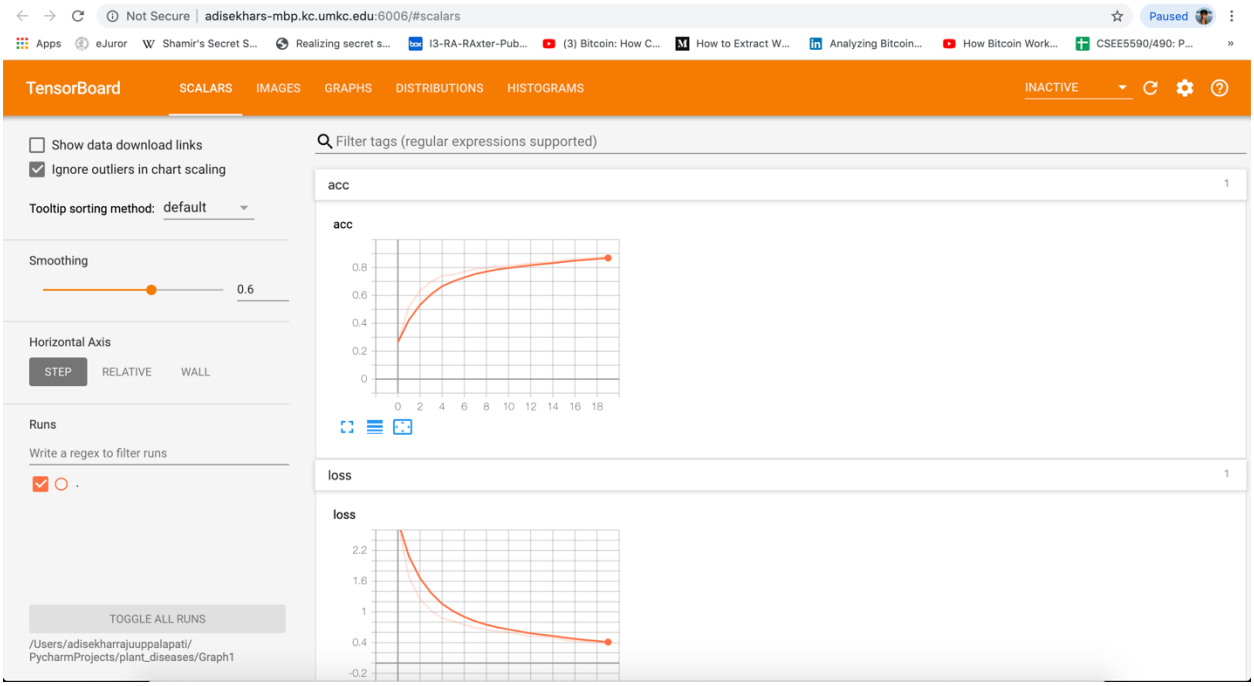
Below is the code snippet of predicting the disease label using model.predict from keras.

```
import numpy as np
from keras.models import load_model
from keras.preprocessing import image
import matplotlib.pyplot as plt

model = load_model('Disease_detector_model.h5')

path = '/Users/adisekharrajuuppalapati/PycharmProjects/plant_diseases/static/4e770559-e332-470e-aae3-c7ad0399dd20___FREC_Scab 3513.JPG'

img = plt.imread(path)
img = np.array(img)
img = img/255.0
plt.imshow(img)
plt.show()
img = image.load_img(path, target_size=(28,28))
img = image.img_to_array(img)
img = img/255.0
img = np.array(img)
plt.imshow(img)

img = img.reshape((1,28,28,3))

print(img.shape)

ans = model.predict(img).argmax()

print(ans)
plt.title(ans)
plt.show()
```

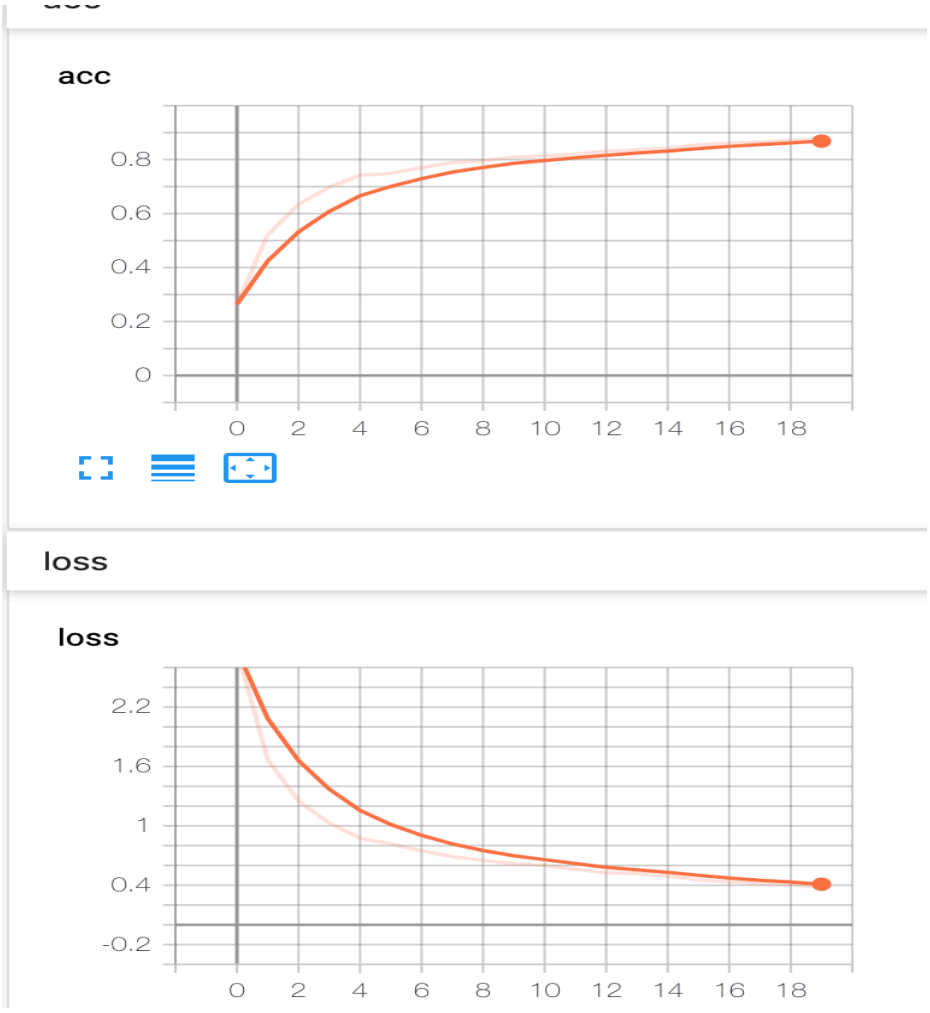
Below is the sample output for the predicted disease label ‘0’.



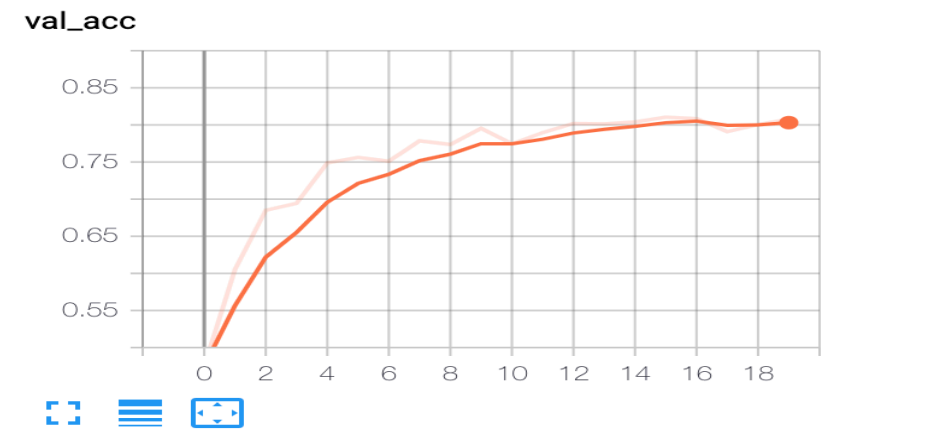
Below is the Tensorboard loss and accuracy graphs for CNN.



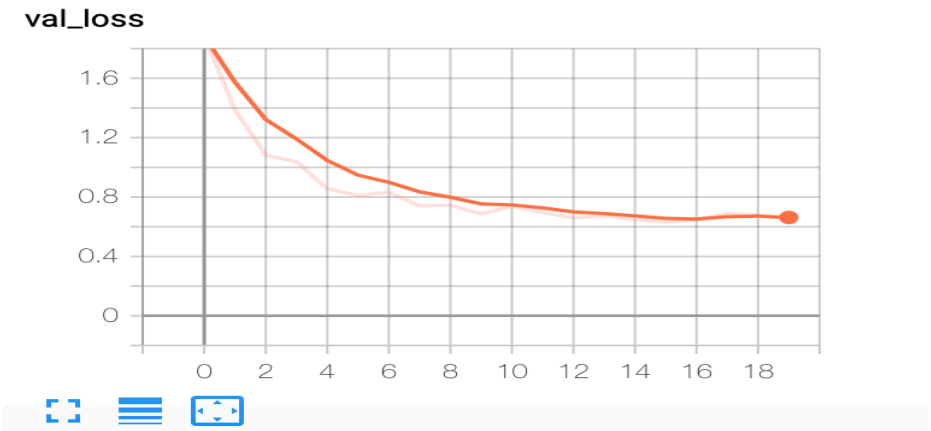
Below is the graph for training accuracy and loss for CNN.



Below is the graph for validation accuracy and loss for CNN.



val_loss

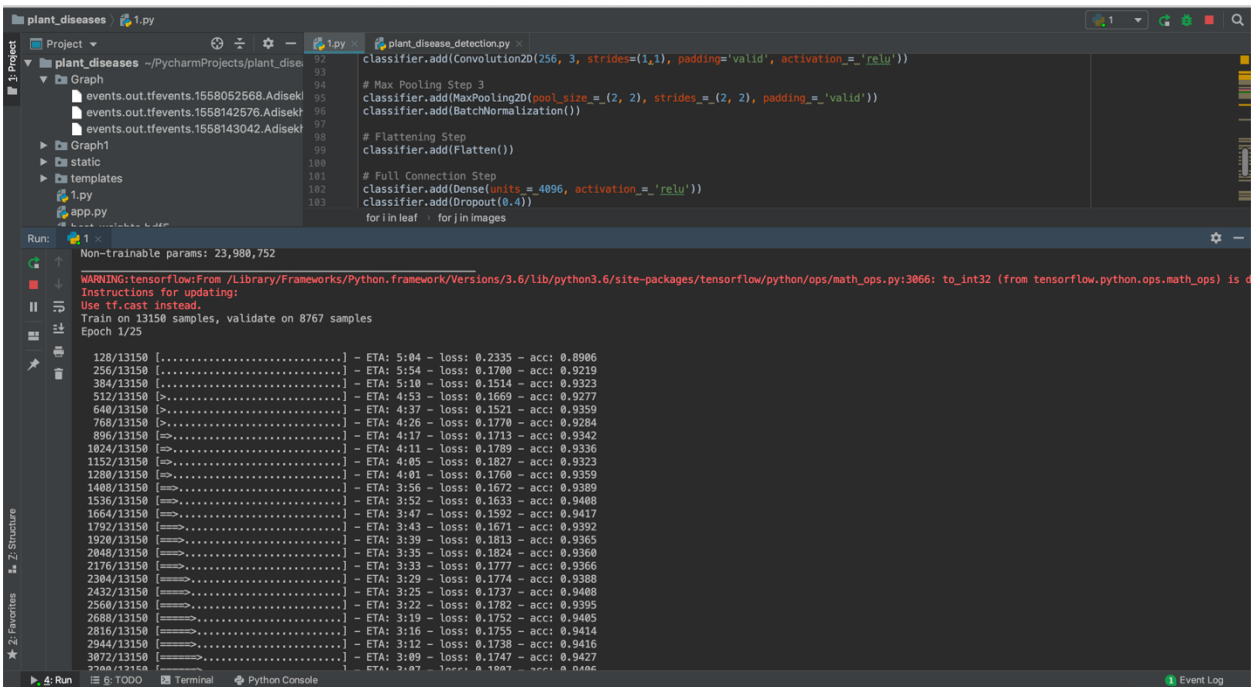


Alexnet code snippets:

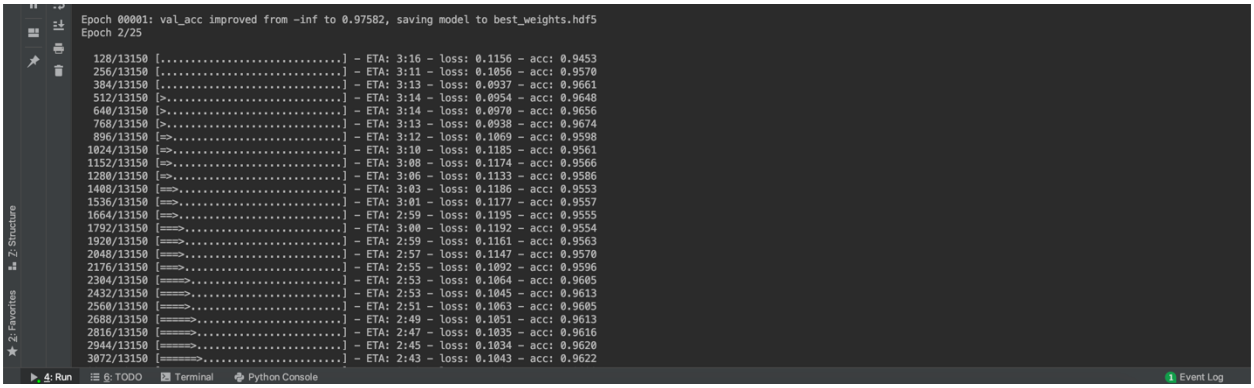
The image preprocessing steps are same for CNN and alexnet. Difference is in the architecture.

Those snippets for alexnet are in the 3.System architecture part.

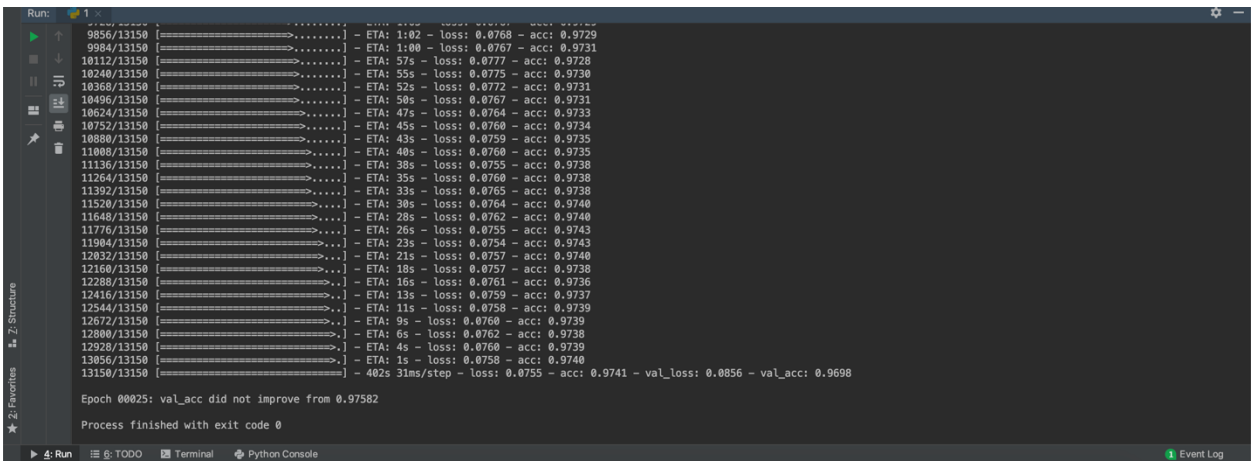
Below are the code snippets while running alexnet architecture in pycharm.



As you can see from below snippet the validation accuracy improved from -inf to 0.97582.



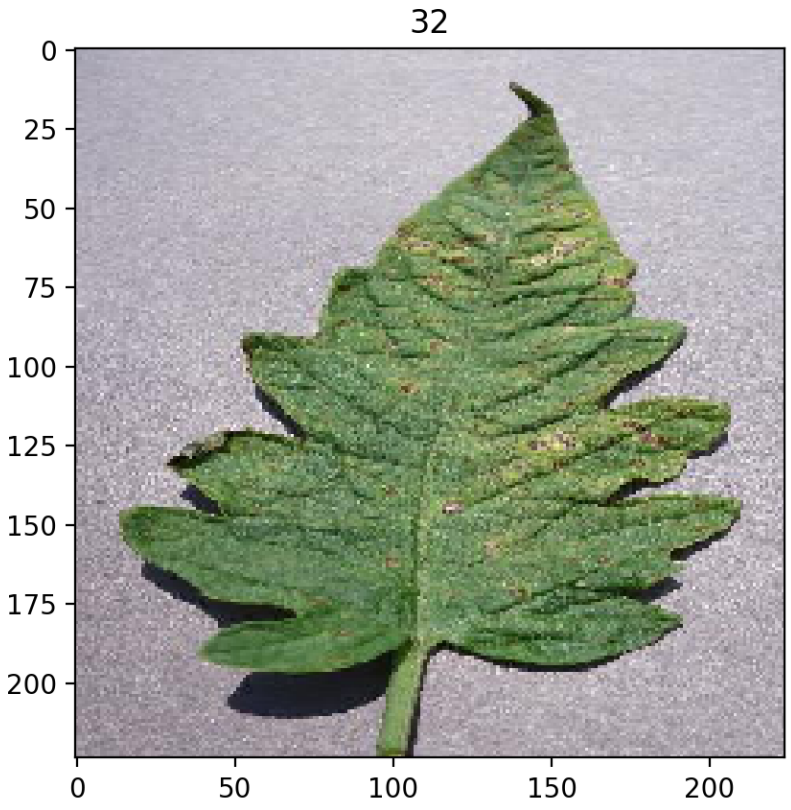
Below snippet is the final training accuracy, loss and validation loss, accuracy for alexnet.



Below is the code snippet for predicting the disease label using alexnet.

```
import matplotlib.pyplot as plt
import matplotlib.image as im

model = load_model('disease_alexnet_detector.h5')

path = '/Users/adisekharrajuuppalapati/PycharmProjects/plant_diseases/static/9bd4db81-94ef-4a9e-a44a-e335fb372691___Matt.S_CG 7786.JPG'


img = plt.imread(path)
img = np.array(img)
img = img / 255.0
plt.imshow(img)
plt.show()


img = image.load_img(path, target_size=(224,224))
img = image.img_to_array(img)
img = img / 255.0
img = np.array(img)
plt.imshow(img)


img = img.reshape((1,224,224,3))

ans = model.predict(img).argmax()
print(ans)

plt.title(ans)
plt.show()
```

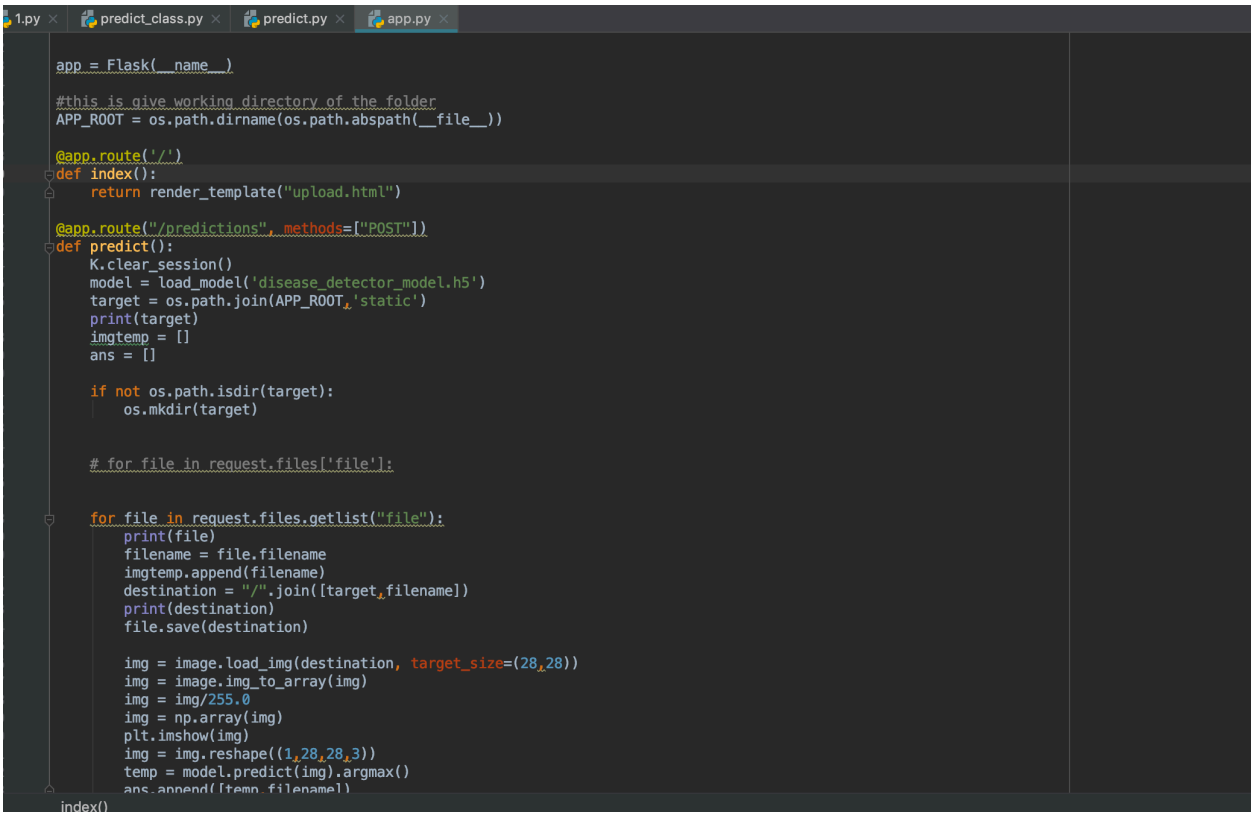
Below is the output for predicted disease label ‘32’.



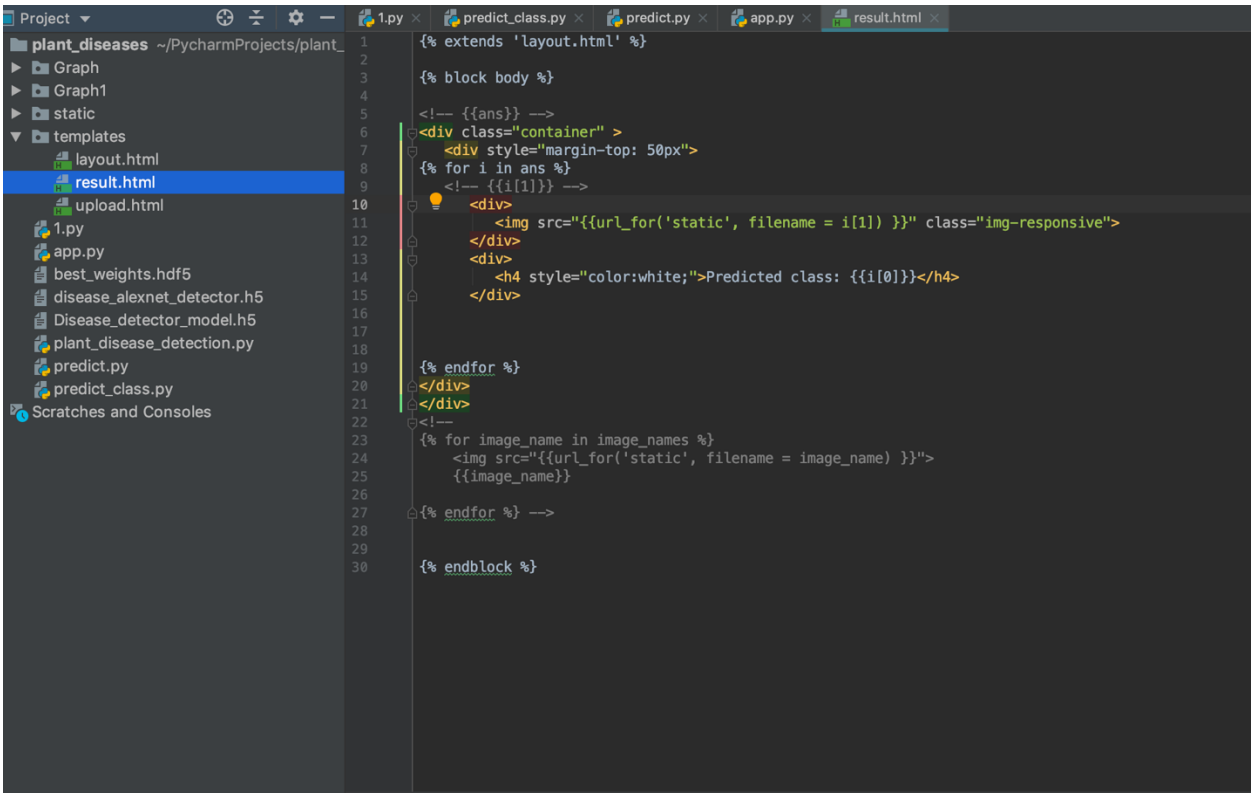
Finally we develop our web application using flask with the saved model.

The Flask Framework looks for HTML files in a folder called **templates.** You **need to create a templates** folder and put all your HTML files in there.

Next is the creation .py file where we import flask module and creating a flask web server from the flask module. Below in the code snippet you can see **__name__ means this current file**. In this case, it will be app.py. This current file will represent my web application.

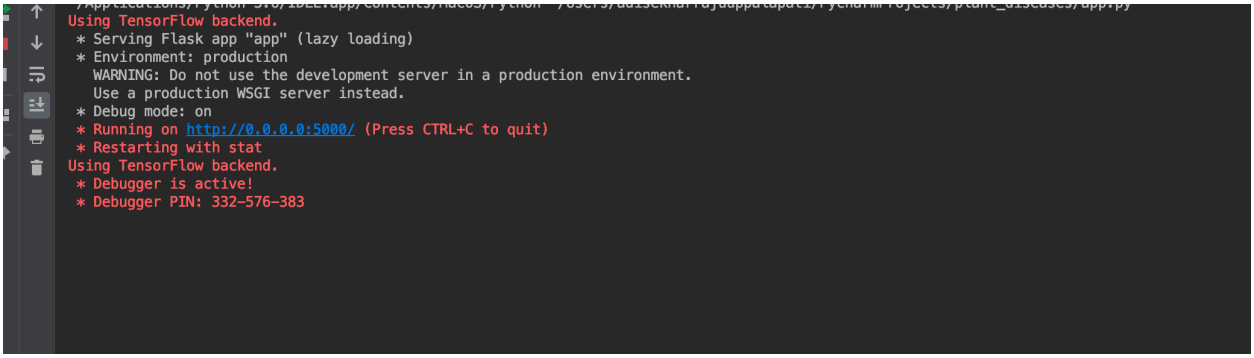


Below snippet is the templates folder.

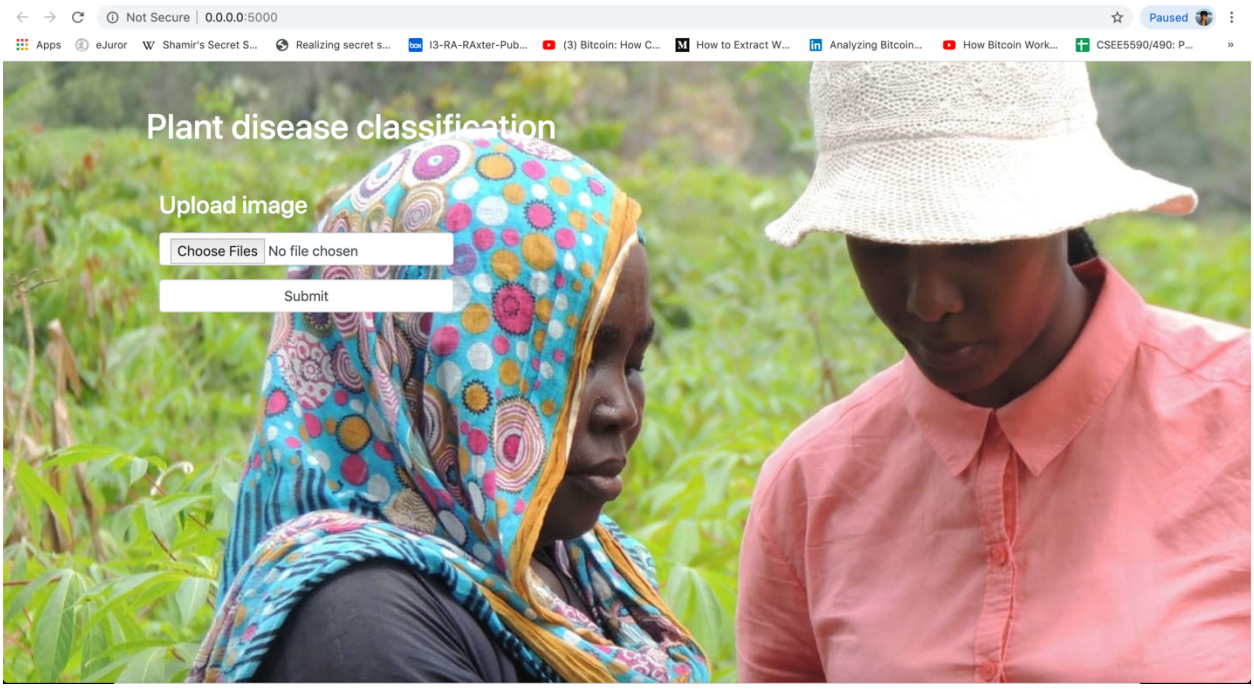


And next we run our flask web application by running app.py file.

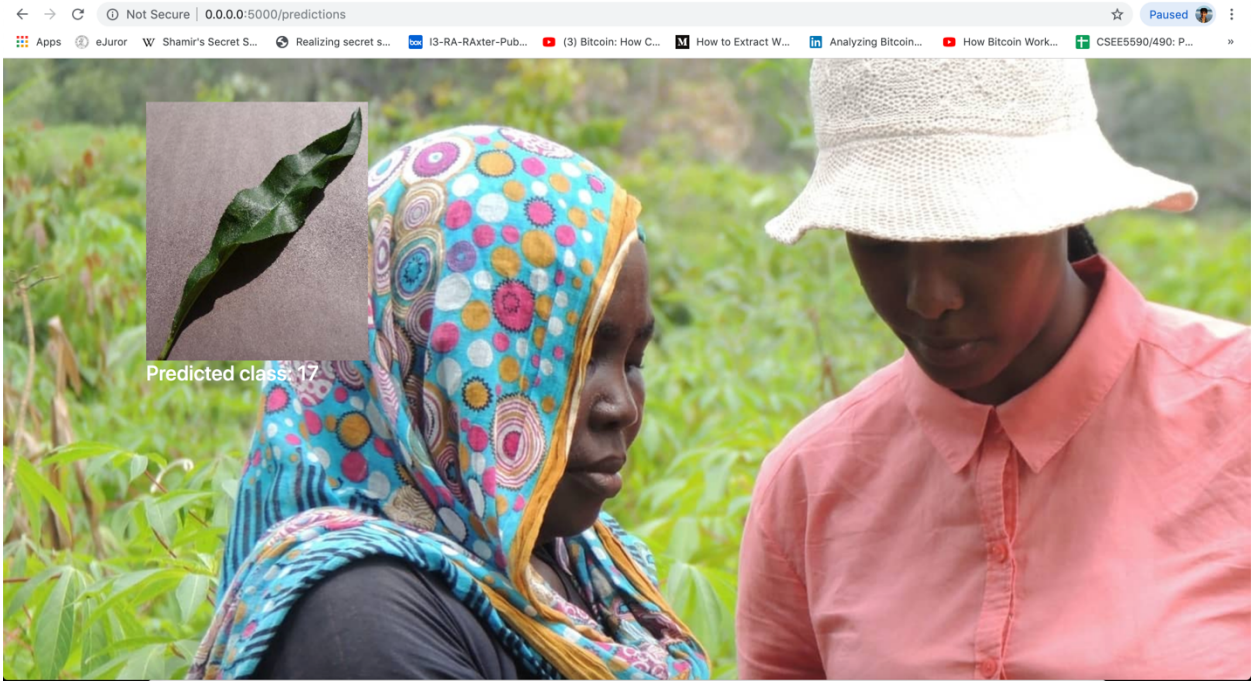
Below is the snippet,



Below is the web site running on 0.0.0.0 with port number 5000,



Below is the predicted disease label ‘17’.



Below is the Github link available for the entire model building and web application:

https://github.com/sekhar1926/CSEE_5590_PYTHON_ICP/tree/master/team_8_project

6. Contribution:

Adhisekhar contribution is on CNN model building and model predictions.

Sudheer contribution is on Alexnet model building, computing best weights, and model predictions.

Mahesh contribution is on deploying the model to the web application using flask.

7. References:

1. https://medium.com/coinmonks/understand-alexnet-in-just-3-minutes-with-hands-on-code-using-tensorflow-925d1e2e2f82

2. https://www.crowdai.org/challenges/plantvillage-disease-classification-challenge

3. https://medium.freecodecamp.org/how-to-build-a-web-application-using-flask-and-deploy-it-to-the-cloud-3551c985e492

4. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

5. https://www.mydatahack.com/building-alexnet-with-keras/