ATHLONE INSTITUE OF TECHNOLOGY

Crop Disease Analyser

Final Project Report

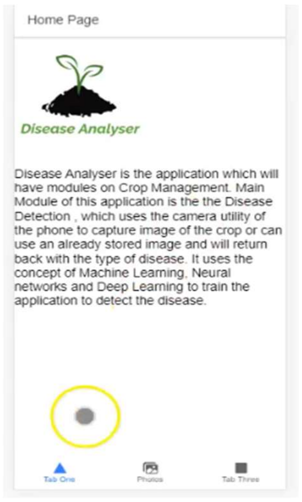
A00268808

Vyanktesh Chandurkar

Submitted in fulfilment of the

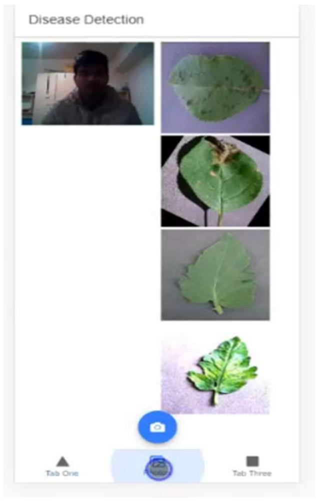
BEng (Hons) In Software Engineering

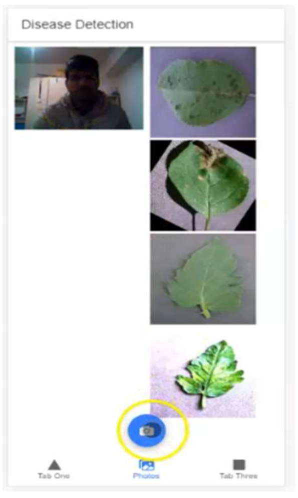
Crop Disease Analyser

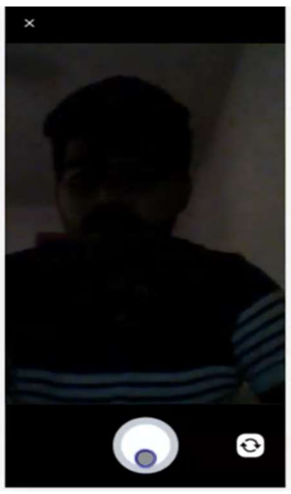

Home Page

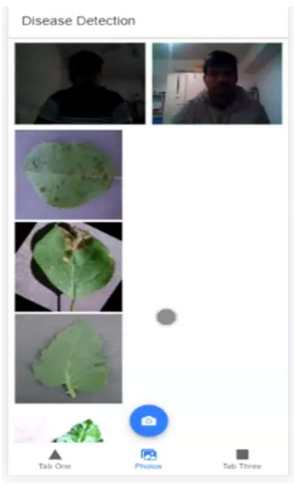
Disease Analyser

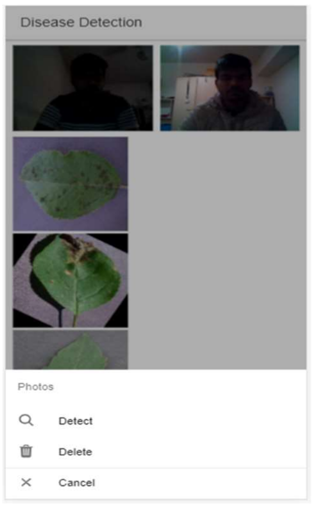
Disease Analyser is the application which will have modules on Crop Management. Main Module of this application is the the Disease Detection , which uses the camera utility of the phone to capture image of the crop or can use an already stored image and will return back with the type of disease. It uses the concept of Machine Learning, Neural networks and Deep Learning to train the application to detect the disease.











Author:

Vyanktesh Chandurkar

Supervisor:

Mr Amit Hirway

ABSTRACT

This report is to detail the building of cross platform software solution. The report will set out how and where the idea came from, and how it was developed and designed. This report will outline how relevant technologies where integrated to achieve the working solution.

This report will outline how the machine learning and tensorflow library was used to build convolutional neural network with the help which model was developed and trained to detect the crop disease when captured image of crop’s leaf or stored image of crop’s leaf was provided as an input. And also how the cross platform mobile app development platform i.e. Ionic was utilized in developing and designing of this mobile application.

ACKNOWLEDGEMENT

The accomplishment of success and final result of this project required supervision, and support from many people and I am very much fortunate to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank Dr. Declan Byrne, for providing me an opportunity to do the project work in Athlone Institute of Technology and giving me all support and guidance which made me complete the project duly. I am extremely thankful for questions asked in interim presentation which helped me to amend my mistake and plan precisely based upon provided inputs.

I owe my gratitude to the project supervisor Mr. Amit Hirway who took keen interest on my project work and guided me all along, till the completion of my project work by providing all the necessary information for better development of the project. I am thankful and fortunate enough to get constant encouragement, support and guidance which helped me in successfully completing my project work.

CONTENTS

1. INTRODUCTION

The purpose of this report is to show the implementation of Convolutional Neural Network (CNN) to build tensorflow model which will be loaded in mobile application to detect the disease of the captured or stored image of crop's leaf. The project explored the latest technologies,

- Tensorflow (Open source library to build and train ML models) :
  To build, train and deploy machine learning models, in this case the developing Convolutional Neural Network to manipulate the image and abstract its features.

- Ionic (Cross-Platform Mobile App Development) :
  To design and develop the mobile application which will use camera utility tool to capture the image of crop's leaf, also can use the already stored image (which can be received through social media), And will further process the image with script and ml model to get the output.

Designing the application went through the process of:

1. Identifying the technologies stack needed to develop the application.
2. Choosing from the possible languages available and analysing the best to complete the intended deliverable.
3. Using the various build tools and 3rd party application to model and develop designs.

Tensorflow model and python script which loads the trained model are necessary parts of this project as the project needs to detect the disease for the captured image of crop's leaf in order to provide intended functionality.

2. Aim and Objectives

This report outlines the development of an mobile application with the help of Ionic, Tensorflow and backend designed in python. The purpose of developing this project, was to identify how a software engineering solution could positively impact the agriculture sector. The provided solution would benefit the agriculture business and it would benefit from technology advances.

Learning the latest open source AI technologies in machine learning remains the prime of the project, and implementing and understanding how to integrate current developed technologies with the new developing emerging technologies becomes one of the major reason for undertaking the project as a major.

Aim of the project is to develop and complete a fully functional mobile application which provides the user with the functionality to detect the crop disease by clicking image of the crop’s leaf

Project Aim is:

- Build fully functioning mobile application.
    - Develop a neural network to manipulate and extract the features from an image.
    - Learn a new technology to build and train the model with the help of developed neural network.
    - Learn a new mobile application development technology, to build mobile application.
    - To understand and observe the applications of machine learning.

Objectives:

- Decide on what are the technologies to be utilized.
- To investigate and decide on what aspects of machine learning are needed to use.
- To learn how to build neural network.
- Decide on what is needed to build a neural network.
- To learn how to build and train the model with the help of neural network.
- Decide on which technology to be used to design mobile application.
- To establish a plan to implement design and integration of the build.
- Successfully import and integrate external code libraries needed to build the neural network.

3. Scope

Machine Learning being an emerging technology provides an infinite scope to test and implement new solutions.

The scope of project is defined to provide:

1. An mobile application that would be used to capture the image, further which will be manipulated for further process.
2. Retain the efficiency of the application while providing all intended functionality.
3. Machine Learning implementation to increase scalability.
4. Deployment of neural network model into the mobile device to get 100% uptime.

The scope of the project can be picturized as "*Providing the user a facility to detect the crop disease just by capturing the image of the crop's leaf to know about the condition of the leaf along with the disease name*".

<u>The scope of this project will encompass the main aim and objective want to achieve. The aim is to learn machine learning technologies and how this integrates and develops with the other technology.</u>

The scope does not define all that is there to learn about machine learning but completing this project will give the goof working knowledge of machine learning's aspect for neural network model with other existing technologies.

4. RESEARCH

RESEARCH INTRODUCTION

Machine Learning

Machine learning is the construction of algorithms for data analysis. It is based on the idea of development of computer programs that can access data, analyse it and learn from it. In simple terms, It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

Machine Learning techniques,

- Supervised Learning: In this method, we ask machines questions and compare machine's answer from actual answer and instruct machines to minimize the error which it makes. It is same as teaching your child some arithmetic operation and then when child does a mistake you correct that and instruct him/her to don't repeat that again and this is how he/she minimizes its mistake. Using this learning Machines can do things like:
    - Diagnostics
    - Can learn how to retain a customer
    - Classify Images
    - Detect Fraud
    - Weather Forecasting
    - Market Forecasting
    - Estimating Life Expectancy

- Unsupervised Learning: In this learning you give your machine huge chunks of data and instruct it to find some sort of pattern in that and based on those patterns your machines accomplish certain tasks. It works somewhat similar to human brain, how we learn behaviour patterns of people and then learn whom to trust and whom to not. So on basis of data which is life experience here we learn. Using this learning machines can do:
    - Customer Segmentation
    - Targeted Marketing
    - Recommendation (Example: YouTube home page)
    - Meaningful Compression of Data
    - Big Data Visualization

- Reinforcement Learning: In this learning Machine is left in an environment where some thing is happening and there is a reward if machine does what we want and there is a punishment if machine does what we don't want and based on it we instruct machine to maximize the reward and eventually machine learns how to do thing which we want it to do. Using this type of learning machines can:
    - Play Games (PUBG/GTA 5/Chess)
    - Acquire Skill
    - Learn Tasks
    - Navigation
    - Take Decisions(Real Time)

With the help of Machine Learning,

- Pattern recognition becomes very efficient and accurate, in many scientific and industrial projects
- Self-improvements are continuous in ML algorithms. As they learn more, they grow more and vice versa
- No human error, these algorithms are free from human psychological or perception-based shortcomings.

Hence, Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. It will be used to train the application to detect different crop diseases.

Deep Learning

Deep Learning is a subset of Machine Learning. Deep Learning has nodes that process information in the form of data and generate output. It is the statistical learning made by the means of many-layered artificial neural networks, also called deep neural networks. It works with large volumes of data (both structured and unstructured) and uses complex algorithms to train a model. Neural Networks help implement Deep Learning.

Neural Networks

Modelled in accordance with the human brain, a Neural Network was built to mimic the functionality of a human brain.

A Neural Network mostly consists of 3 layers (input, hidden and output) and each layer consists of neuron/nodes that perform numerical computations and other operations. Each node in a layer is interconnected to other nodes present in consecutive layers. There are weights assigned to each interconnection and a bias assigned to each layer. These weights and biases are called parameters of the network.

- The input layer in the network is responsible for receiving large volumes of data as inputs in different formats (text, csv files, images, etc). The hidden layer is where all the calculation, computation and feature extraction take place. The output layer is responsible for generating the desired output.

- The hidden layers use several activation functions like ReLU, Sigmoid, Step function, etc, to calculate the weighted sum of inputs plus a bias is added. The output of the computation will decide which nodes to fire. A Softmax function is usually applied to get the proper output.

- The predicted output is of the network is compared with the actual output and the difference in the output (i.e. the error) is backpropagated through the network and the weights are adjusted accordingly. By using a cost function, the error in the network is calculated. The same process follows again to get the desired output.

Some types of Neural networks,

- Recurrent Neural Network(RNN) – Long Short Term Memory:
  The Recurrent Neural Network works on the principle of saving the output of a layer and feeding this back to the input to help in predicting the outcome of the layer.

- Convolutional Neural Network:
  Convolutional neural networks are similar to feedforward neural networks, where the neurons have learn-able weights and biases. Its application has been in signal and image processing which takes over OpenCV in the field of computer vision.

- Modular Neural Network:
  Modular Neural Networks have a collection of different networks working independently and contributing towards the output. Each neural network has a set of inputs which are unique compared to other networks constructing and performing sub-tasks. These networks do not interact or signal each other in accomplishing the tasks. The advantage of a modular neural network is that it breakdowns a large computational process into smaller components decreasing the complexity.

Convolutional Neural Network

Requirement for the project idea is of Convolutional Neural Network (CNN). CNN basically can be defined through three components,

- Convolutional layer: weight matrix is defined which extracts certain features from the images
- Pooling layer: purpose is for reducing the spatial size of the image
- Output layer: output is a result as whether or not an image belongs to a particular class. The output layer has a loss function like categorical cross-entropy, to compute the error in prediction.

Need of CNN,

- A Deep Learning algorithm which can take in an input image.
- Assign importance (learnable weights and biases) to various aspects/objects in the image.
- Pre-processing required is much lower as compared to other classification algorithms.

TensorFlow

TensorFlow is a machine learning / deep learning / multilayer neural network library from Google. Libraries using data flow graphs can be used to describe complex networks in an easy-to-understand manner.

- Primary software tools used for deep learning developed by Google Brain team.
- Open source artificial intelligence library, using data flow graphs to build models.
- Mainly used for **Classification, Perception, Understanding, Discovering, Prediction and Creation** of images, sound, text or time series.

Challenges of TensorFlow

- First of all, because of its very high performance, hardware that uses TensorFlow is also required to have a matching high performance.
- In addition, although it can be said that distributed learning of data calculation can be said to be an advantage, it can be said that preparing an environment for doing that is a bit more difficult.

Usage:

- Image recognition: TensorFlow can analyse the information in the image. This image recognition is a technology that has great potential for this.
- Image search: By learning the original image, appropriate images can be searched against the features learned so far. Currently, this identification performance is said to be evolving at a speed.
- Speech recognition: Speech recognition rates are high, and most modern devices can also be operated with speech, which is Google's top-class performance.
- Language translation: TensorFlow has also had a major impact on the language translation environment. Use of the neural networks has made it possible to produce highly accurate results for speech as well as text.

Ionic

The Ionic framework is an open source library of mobile-optimized components in JavaScript, HTML, and CSS. It provides developers with tools for building robust and optimized apps that work on both, Android and iOS. Unlike a responsive framework, Ionic comes with very native-styled mobile UI elements and layouts that you'd get with a native SDK on iOS or Android but didn't really exist before on the web.

- Ionic is a tool for cross-platform mobile development.
- It enables you to develop mobile apps using web technologies and languages like HTML, CSS, JavaScript, Angular and TypeScript.

Some of the advantages of Ionic framework are here:

1. Cross-platform app development

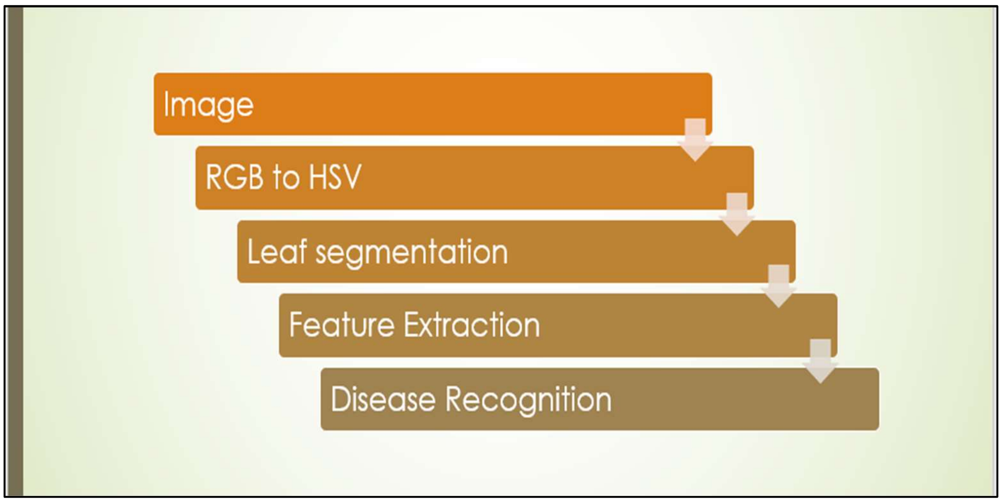
2. User Interface

3. Built on AngularJS

4. Performance

5. Cordova Plugins

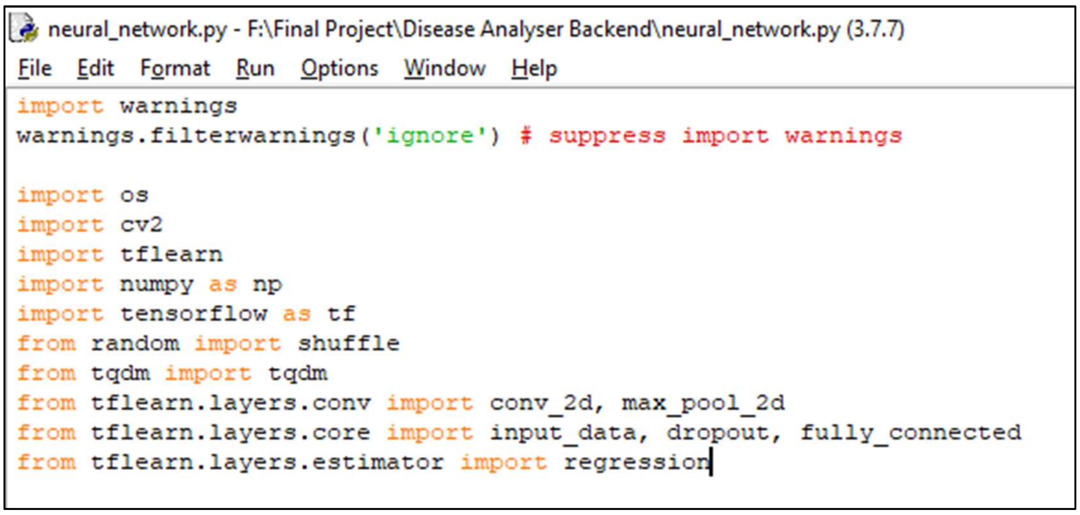
5. Flow Diagram of the Application



- Captured Image or stored image will be taken as an input.
- If required, image can be converted to grayscale for better processing.
- Leaf's image will be segmented i.e. manipulated with the help of cv library for example to read colour of the image or to resize it according t defined dimensions.
- Then feature extraction i.e. converting it to tensor will be done with the help of defined Neural Network consisting of different layers.
- Further passing it to the trained model for the prediction.

6. Implementation

Implementation of Tflearn and other required libraries.


neural_network.py - F:\Final Project\Disease Analyser Backend\neural_network.py (3.7.7)
File Edit Format Run Options Window Help

```
import warnings
warnings.filterwarnings('ignore') # suppress import warnings

import os
import cv2
import tflearn
import numpy as np
import tensorflow as tf
from random import shuffle
from tqdm import tqdm
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
```

➔ import tflearn
TFlearn is a modular and transparent deep learning library built on top of TensorFlow. It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it.

➔ Import os
The OS module in python provides functions for interacting with the operating system. OS, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.

➔ Import cv2
library of Python bindings designed to solve computer vision problems.
cv2.imread() method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

- ➔ import numpy
  Numpy is a library that used for computing scientific/mathematical data. NumPy supports large, multidimensional arrays and matrices. NumPy arrays are faster compared to Python lists.

- ➔ import tensorflow
  It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

- ➔ from random import shuffle
  Shuffle a list using random. shuffle() The random. shuffle() is used to shuffle the list in place. i.e., it randomizes the order of items in a list, we call it a randomizes the elements of a list in place

- ➔ from tqdm import tqdm
  TQDM is a progress bar library

- ➔ From tflearn.layers.conv import conv_2d, max_pool_2d
  Layers of convolutional neural network,
  Conv_2d is convolutional layer
  Max_pool_2d is pooling layer

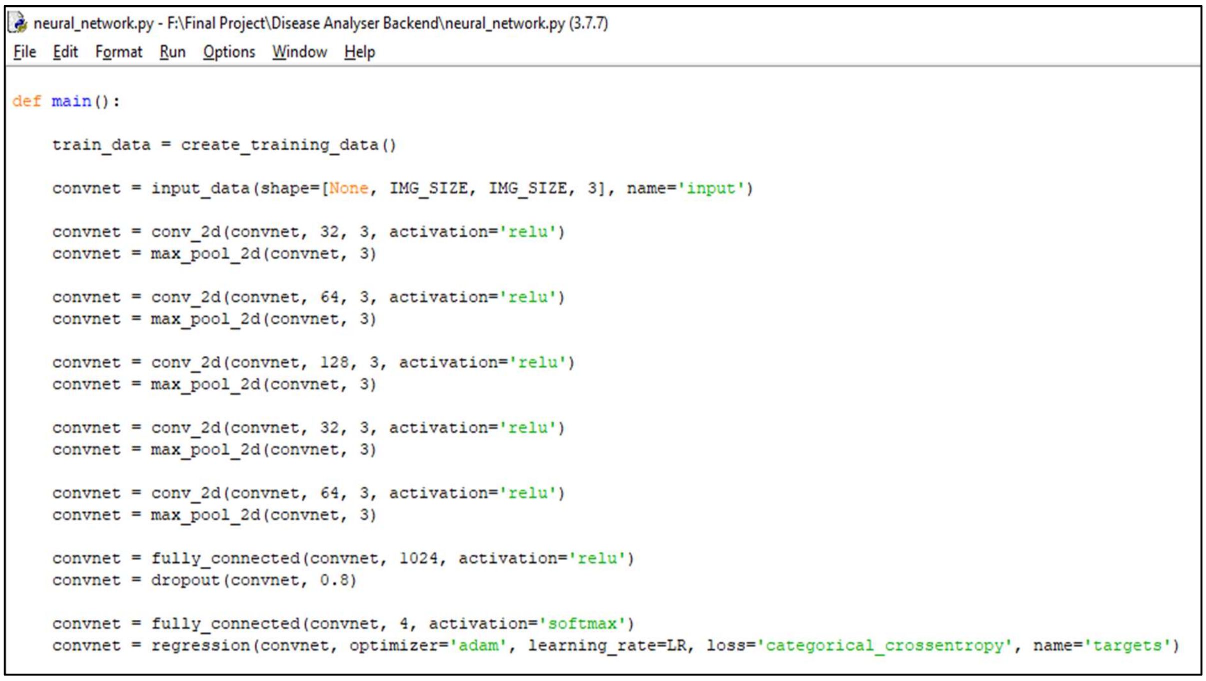
- ➔ From tflearn.layers.core import input_data, dropout, fully_connected
  Input_data layer is used for inputting (aka. feeding) data to a network.
  Dropout, fullyconnected layers connect every node in the first layer to every node in the second layer.

- ➔ From tflearn.layers.estimator import regression
  Regression layer subset of Output layer of CNN to apply a regression (linear or logistic) to the provided input

The simple workflow in TFLearn is as follows:

1. Create an input layer first.
2. Pass the input object to create further layers.
3. Add the output layer.
4. Create the net using an estimator layer such as regression.
5. Create a model from the net created in the previous step.
6. Train the model with the model.fit() method.
7. Use the trained model to predict or evaluate.

IMPLEMENTATION OF CONVOLUTIONAL NEURAL NETWORK


neural_network.py - F:\Final Project\Disease Analyser Backend\neural_network.py (3.7.7)
File Edit Format Run Options Window Help

```
def main():

    train_data = create_training_data()

    convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

    convnet = conv_2d(convnet, 32, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 64, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 128, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 32, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 64, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = fully_connected(convnet, 1024, activation='relu')
    convnet = dropout(convnet, 0.8)

    convnet = fully_connected(convnet, 4, activation='softmax')
    convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')
```

Convnet = input_data()

This layer is used for inputting (aka. feeding) data to a network. A TensorFlow placeholder will be used if it is supplied, otherwise a new placeholder will be created with the given shape.

Either a shape or placeholder must be provided, otherwise an exception will be raised.

tflearn.layers.core.input_data(shape=None,placeholder=None,dtype=tf.float32,data_preprocessing= None, data_augmentation=None, name='InputData')

Convnet = conv_2d()

Convolutional layer of the CNN where weight matrix is defined which extracts certain features from the images.

Input = 4-D Tensor [batch, height, width, in_channels].

Output = 4-D Tensor [batch, new height, new width, nb_filter].

tflearn.layers.conv.conv_2d (incoming, nb_filter, filter_size, strides=1, padding='same', activation='linear', bias=True, weights_init='uniform_scaling', bias_init='zeros', regularizer=None, weight_decay=0.001, trainable=True, restore=True, reuse=False, scope=None, name='Conv2D')

Convnet = max_pool_2d()

Pooling layer of the CNN and purpose is for reducing the spatial size of the image.

Input = 4-D Tensor [batch, height, width, in_channels].

Output = 4-D Tensor [batch, pooled height, pooled width, in_channels].

tflearn.layers.conv.max_pool_2d (incoming, kernel_size, strides=None, padding='same', name='MaxPool2D')

Convnet = fully_connected()

fullyconnected layers connect every node in the first layer to every node in the second layer.

Input = (2+)-D Tensor [samples, input dim]. If not 2D, input will be flatten.

Output = 2D Tensor [samples, n_units].

tflearn.layers.core.fully_connected (incoming, n_units, activation='linear', bias=True, weights_init='truncated_normal', bias_init='zeros', regularizer=None, weight_decay=0.001, trainable=True, restore=True, reuse=False, scope=None, name='FullyConnected')

Convnet = dropout()

Outputs the input element scaled up by 1 / keep_prob. The scaling is so that the expected sum is unchanged.

By default, each element is kept or dropped independently. If noise_shape is specified, it must be broadcastable to the shape of x, and only dimensions with noise_shape[i] == shape(x)[i] will make independent decisions. For example, if shape(x) = [k, l, m, n] and noise_shape = [k, 1, 1, n], each batch and channel component will be kept independently and each row and column will be kept or not kept together.

tflearn.layers.core.dropout (incoming, keep_prob, noise_shape=None, name='Dropout')

Convnet = regression()

Regression layer subset of Output layer of CNN to apply a regression (linear or logistic) to the provided input. It requires to specify a TensorFlow gradient descent optimizer 'optimizer' that will minimize the provided loss function 'loss' (which calculate the errors). A metric can also be provided, to evaluate the model performance.

tflearn.layers.estimator.regression (incoming, placeholder='default', optimizer='adam', loss='categorical_crossentropy', metric='default', learning_rate=0.001, dtype=tf.float32, batch_size=64, shuffle_batches=True, to_one_hot=False, n_classes=None, trainable_vars=None, restore=True, op_name=None, validation_monitors=None, validation_batch_size=None, name=None)

Built-in Operations

Besides layers concept, TFLearn also provides many different ops to be used when building a neural network. These ops are firstly mean to be part of the above 'layers' arguments, but they can also be used independently in any other Tensorflow graph for convenience. In practice, just providing the op name as argument is enough (such as activation='relu' or regularizer='L2' for conv_2d), but a function can also be provided for further customization.

relu:

Relu is used in the middle / hidden layers of the network to regularize the activation.

It is essentialy the function: max(0, x)

Activation should not be in negative, either it should be zero or more than that.

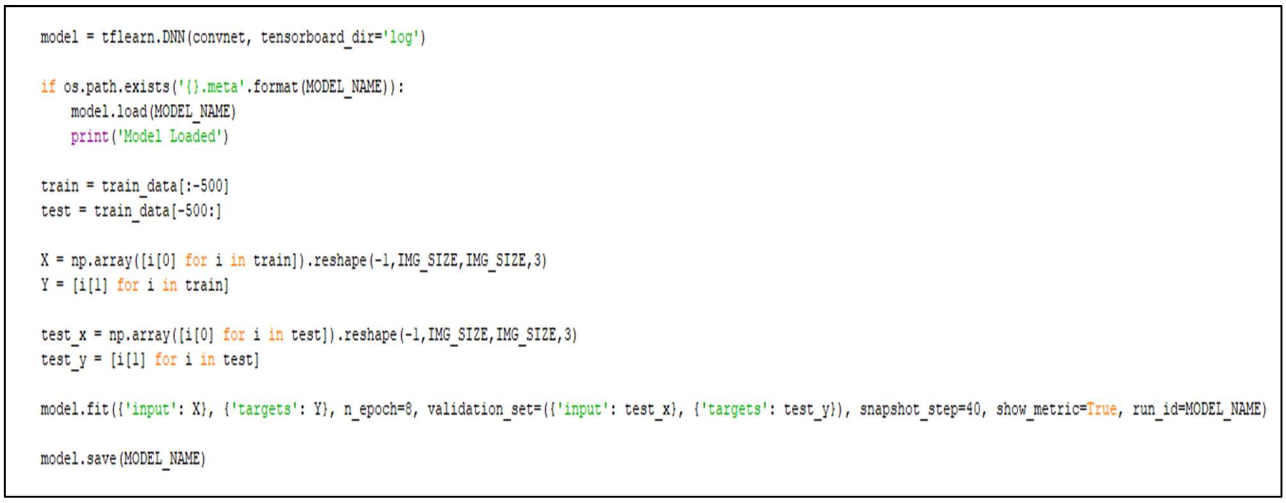
softmax:

Softmax is used for the output layer in multi class classification problems.

It is essentially the function: log(1 + e^x)

It outputs a vector of probabilities of each class.

IMPLEMENTATION OF DNN (DEEP NEURAL NETWORK) MODEL CLASS



```
model = tflearn.DNN(convnet, tensorboard_dir='log')

if os.path.exists('{}.meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print('Model Loaded')

train = train_data[:-500]
test = train_data[-500:]

X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
test_y = [i[1] for i in test]

model.fit({'input': X}, {'targets': Y}, n_epoch=8, validation_set=({'input': test_x}, {'targets': test_y}), snapshot_step=40, show_metric=True, run_id=MODEL_NAME)

model.save(MODEL_NAME)
```

Training functions are another core feature of TFLearn. In Tensorflow, there are no pre-built API to train a network, so TFLearn integrates a set of functions that can easily handle any neural network training, whatever the number of inputs, outputs and optimizers.

While using TFlearn layers, many parameters are already self managed, so it is very easy to train a model, using DNN model class.

This class is presented high-level API for neural networks. Model allows to set params for preprocessing input image. Model creates net from file with trained weights and config, sets preprocessing input and runs forward pass.

- ➔ Tflearn.DNN : creates net from file with trained weights and config, sets preprocessing input and runs forward pass.

- ➔ Model.load : helps in loading the model.

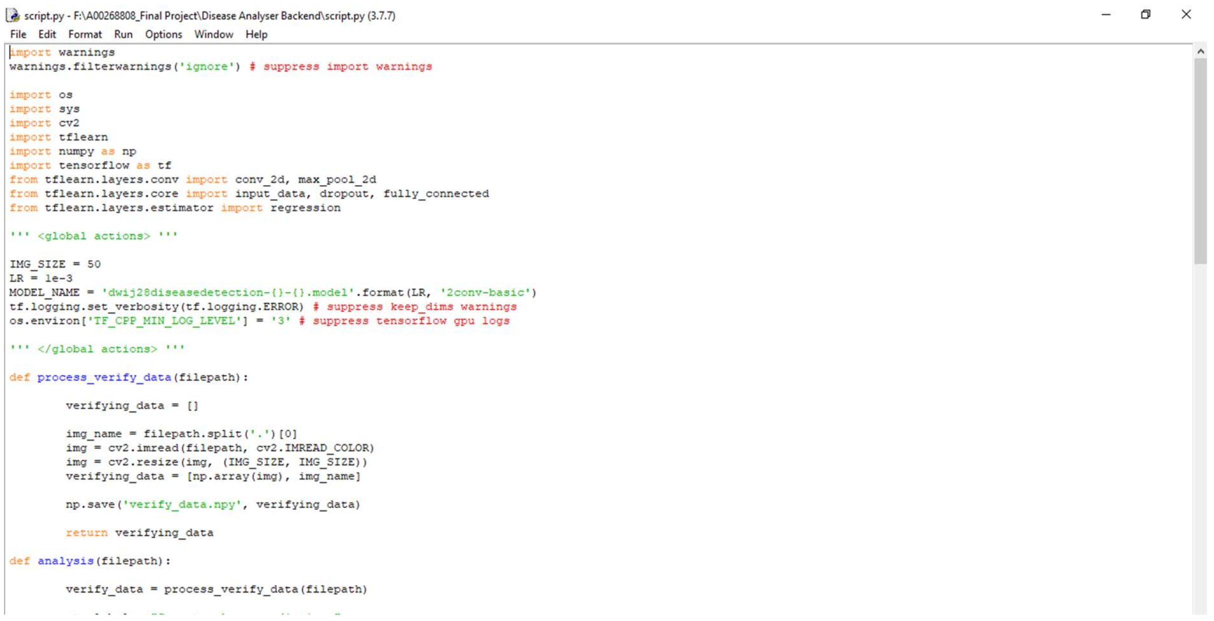
- ➔ Model.fit : used to train the model.

- ➔ Model.save : used to save the model after training it.

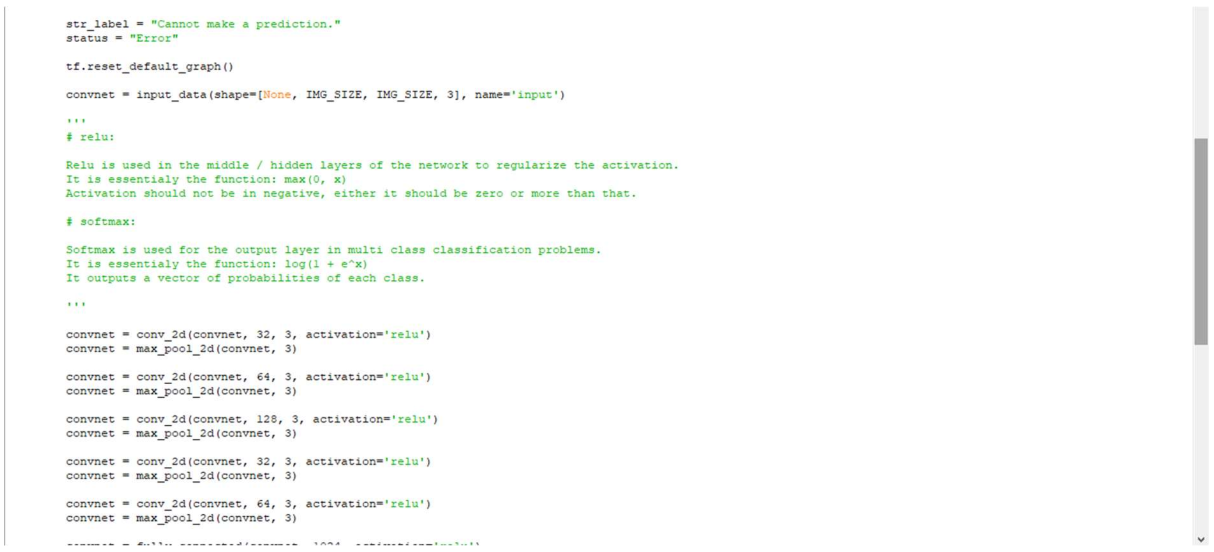
7. Code Development

Backend

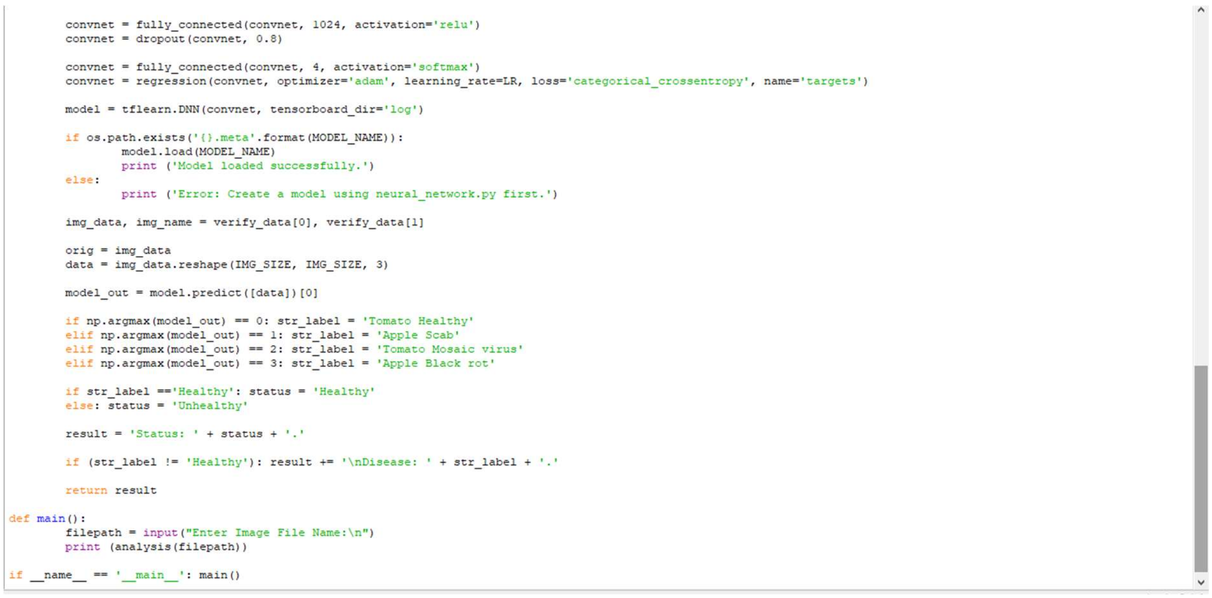
Main file i.e. script designed in python which takes the input manipulates it, process it further with defined convolutional neural network and the result is passed on to the trained model for the prediction.


script.py - F:\A00268808_Final Project\Disease Analyser Backend\script.py (3.7.7)
File Edit Format Run Options Window Help

```python
import warnings
warnings.filterwarnings('ignore') # suppress import warnings

import os
import sys
import cv2
import tflearn
import numpy as np
import tensorflow as tf
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

''' <global actions> '''

IMG_SIZE = 50
LR = 1e-3
MODEL_NAME = 'dwij28diseasedetection-{}-{}.model'.format(LR, '2conv-basic')
tf.logging.set_verbosity(tf.logging.ERROR) # suppress keep_dims warnings
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' # suppress tensorflow gpu logs

''' </global actions> '''

def process_verify_data(filepath):

        verifying_data = []

        img_name = filepath.split('.')[0]
        img = cv2.imread(filepath, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        verifying_data = [np.array(img), img_name]

        np.save('verify_data.npy', verifying_data)

        return verifying_data

def analysis(filepath):

        verify_data = process_verify_data(filepath)
```

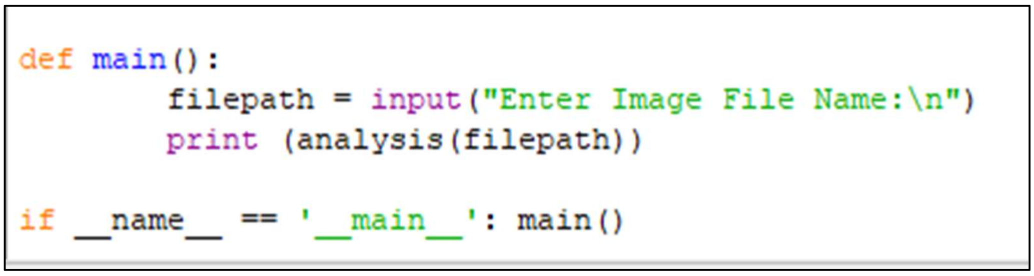


```python
        str_label = "Cannot make a prediction."
        status = "Error"

        tf.reset_default_graph()

        convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

        '''
        # relu:

        Relu is used in the middle / hidden layers of the network to regularize the activation.
        It is essentialy the function: max(0, x)
        Activation should not be in negative, either it should be zero or more than that.

        # softmax:

        Softmax is used for the output layer in multi class classification problems.
        It is essentialy the function: log(1 + e^x)
        It outputs a vector of probabilities of each class.

        '''

        convnet = conv_2d(convnet, 32, 3, activation='relu')
        convnet = max_pool_2d(convnet, 3)

        convnet = conv_2d(convnet, 64, 3, activation='relu')
        convnet = max_pool_2d(convnet, 3)

        convnet = conv_2d(convnet, 128, 3, activation='relu')
        convnet = max_pool_2d(convnet, 3)

        convnet = conv_2d(convnet, 32, 3, activation='relu')
        convnet = max_pool_2d(convnet, 3)

        convnet = conv_2d(convnet, 64, 3, activation='relu')
        convnet = max_pool_2d(convnet, 3)
```



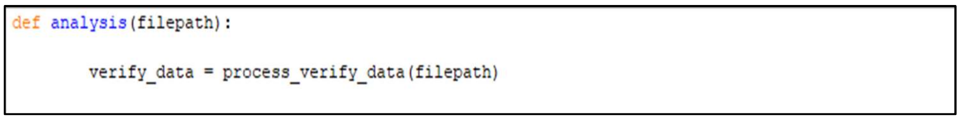
```python
	convnet = fully_connected(convnet, 1024, activation='relu')
	convnet = dropout(convnet, 0.8)

	convnet = fully_connected(convnet, 4, activation='softmax')
	convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

	model = tflearn.DNN(convnet, tensorboard_dir='log')

	if os.path.exists('{}.meta'.format(MODEL_NAME)):
		model.load(MODEL_NAME)
		print ('Model loaded successfully.')
	else:
		print ('Error: Create a model using neural_network.py first.')

	img_data, img_name = verify_data[0], verify_data[1]

	orig = img_data
	data = img_data.reshape(IMG_SIZE, IMG_SIZE, 3)

	model_out = model.predict([data])[0]

	if np.argmax(model_out) == 0: str_label = 'Tomato Healthy'
	elif np.argmax(model_out) == 1: str_label = 'Apple Scab'
	elif np.argmax(model_out) == 2: str_label = 'Tomato Mosaic virus'
	elif np.argmax(model_out) == 3: str_label = 'Apple Black rot'

	if str_label =='Healthy': status = 'Healthy'
	else: status = 'Unhealthy'

	result = 'Status: ' + status + '.'

	if (str_label != 'Healthy'): result += '\nDisease: ' + str_label + '.'

	return result

def main():
	filepath = input("Enter Image File Name:\n")
	print (analysis(filepath))

if __name__ == '__main__': main()
```

BACKEND - CODE EXPLANATION



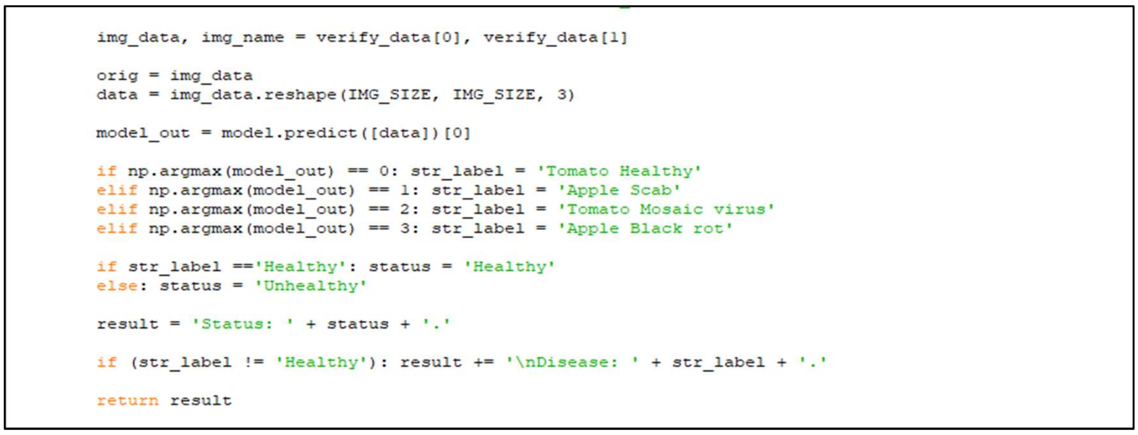
```python
def main():
	filepath = input("Enter Image File Name:\n")
	print (analysis(filepath))

if __name__ == '__main__': main()
```

- Main function to get the file path
- File path is obtained from the ionic application with the help of the API
- Further this function will pass the file path obtained to the another function defined as analysis and which will return back with the result, displayed with print().



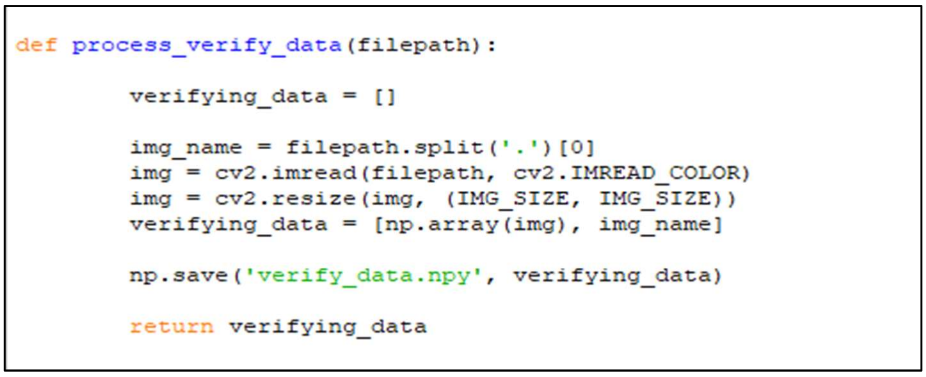
```python
def analysis(filepath):

	verify_data = process_verify_data(filepath)
```

- In the analysis function the file is further passed to another function 'process_verify_data()'.
- Verify_data will store the data returned by the process_verify_data function.



```
        img_data, img_name = verify_data[0], verify_data[1]

        orig = img_data
        data = img_data.reshape(IMG_SIZE, IMG_SIZE, 3)

        model_out = model.predict([data])[0]

        if np.argmax(model_out) == 0: str_label = 'Tomato Healthy'
        elif np.argmax(model_out) == 1: str_label = 'Apple Scab'
        elif np.argmax(model_out) == 2: str_label = 'Tomato Mosaic virus'
        elif np.argmax(model_out) == 3: str_label = 'Apple Black rot'

        if str_label =='Healthy': status = 'Healthy'
        else: status = 'Unhealthy'

        result = 'Status: ' + status + '.'

        if (str_label != 'Healthy'): result += '\nDisease: ' + str_label + '.'

        return result
```

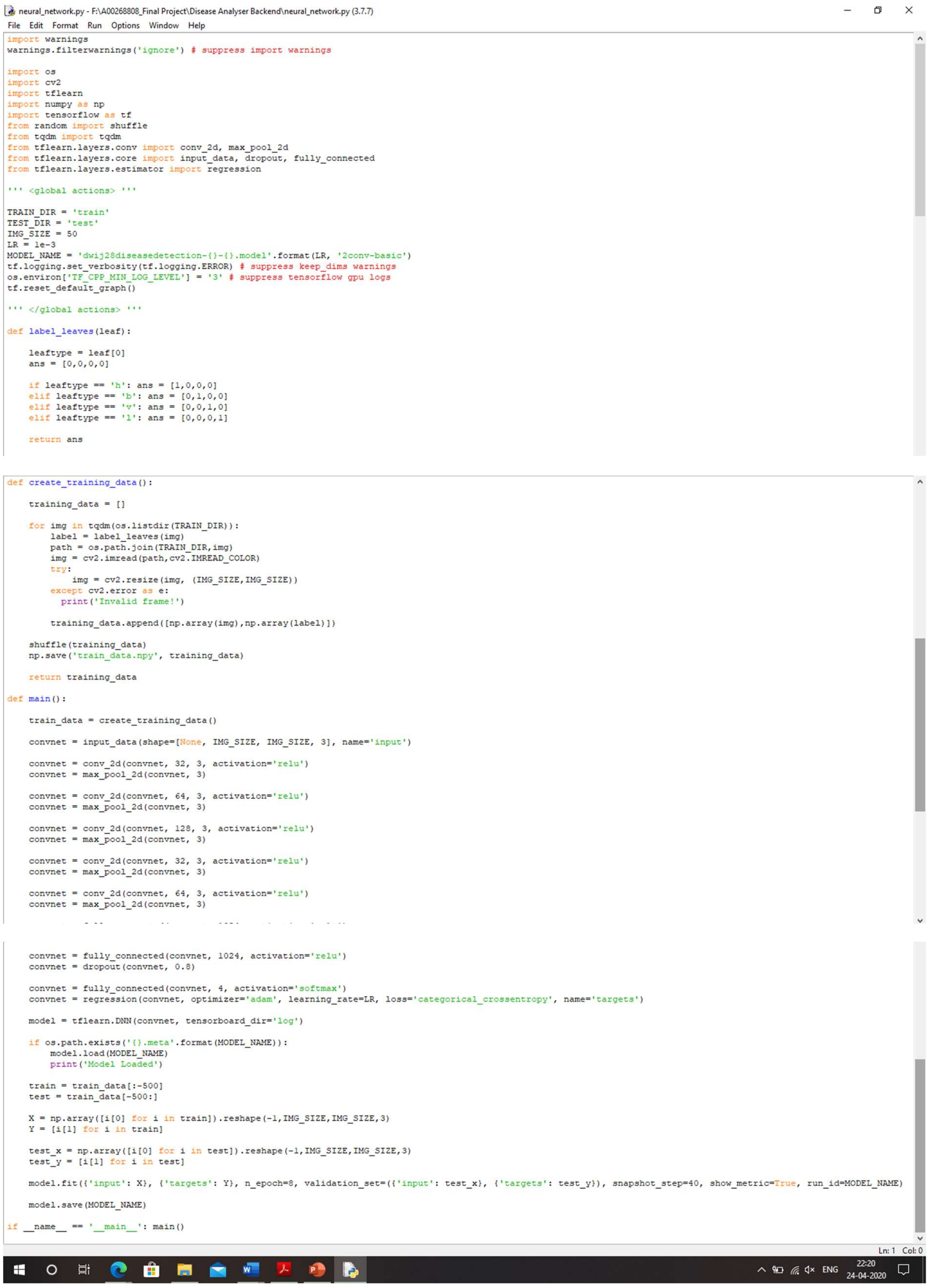
- Further the Analysis function uses the return of the process_verify_data function stored under numpy array i.e. verify_data
- Array data is passed to the loaded model which will carry out the prediction process.
- Result is returned on the basis of the model prediction.



```
def process_verify_data(filepath):

        verifying_data = []

        img_name = filepath.split('.')[0]
        img = cv2.imread(filepath, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        verifying_data = [np.array(img), img_name]

        np.save('verify_data.npy', verifying_data)

        return verifying_data
```

- This function uses the cv2 library to get and store the image details.
- Cv2.imread – reads the image colour.
- Cv2.resize – resizes the image with the defined image size under global actions.
- Results are stored in numpy array i.e. verifying_data.

Neural Network File where CNN was defined and with the help of DNN model class, a tensorflow model was build and was trained with available datasets.

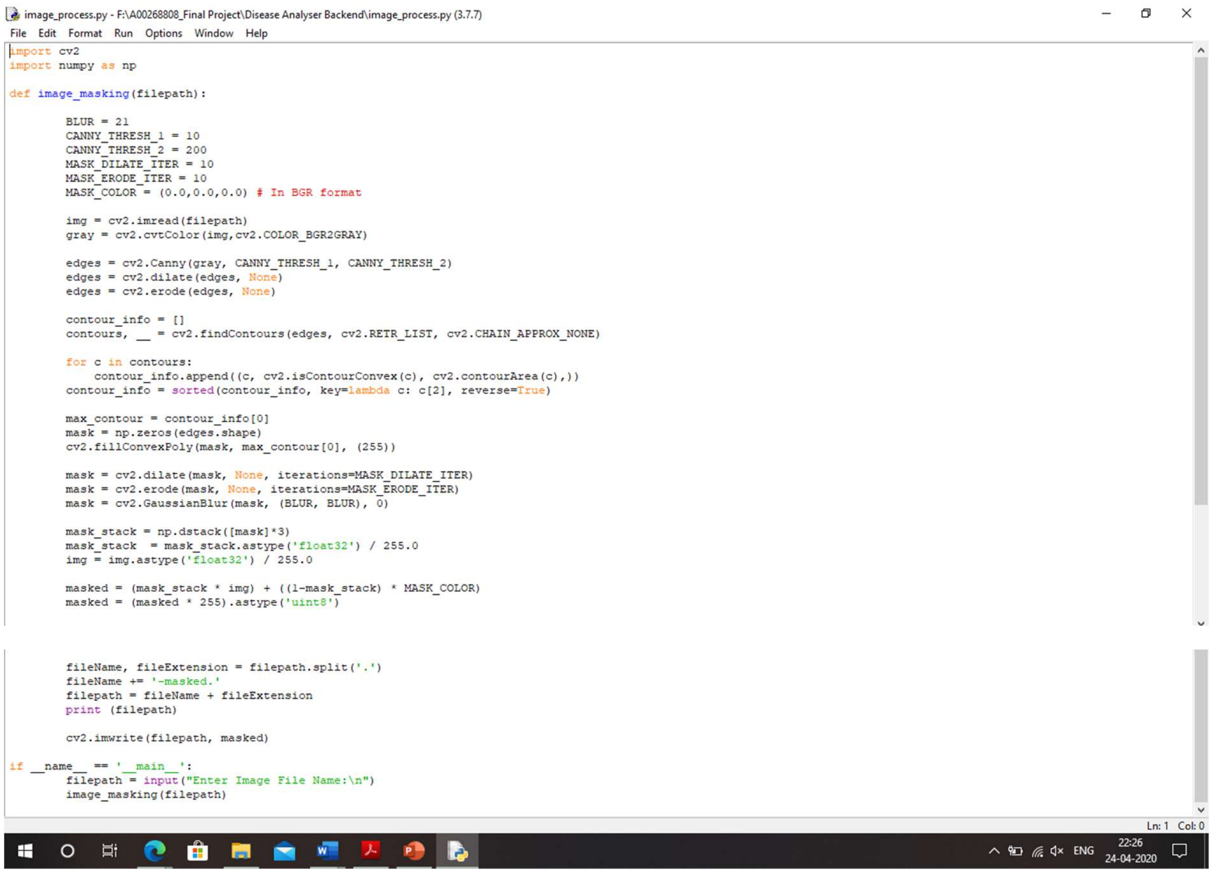

neural_network.py - F:\A00268808_Final Project\Disease Analyser Backend\neural_network.py (3.7.7)
File Edit Format Run Options Window Help
Ln: 1 Col: 0
ENG
22:20
24-04-2020

```python
import warnings
warnings.filterwarnings('ignore') # suppress import warnings

import os
import cv2
import tflearn
import numpy as np
import tensorflow as tf
from random import shuffle
from tqdm import tqdm
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

''' <global actions> '''

TRAIN_DIR = 'train'
TEST_DIR = 'test'
IMG_SIZE = 50
LR = 1e-3
MODEL_NAME = 'dwij28diseasedetection-{}-{}.model'.format(LR, '2conv-basic')
tf.logging.set_verbosity(tf.logging.ERROR) # suppress keep_dims warnings
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' # suppress tensorflow gpu logs
tf.reset_default_graph()

''' </global actions> '''

def label_leaves(leaf):

    leaftype = leaf[0]
    ans = [0,0,0,0]

    if leaftype == 'h': ans = [1,0,0,0]
    elif leaftype == 'b': ans = [0,1,0,0]
    elif leaftype == 'v': ans = [0,0,1,0]
    elif leaftype == 'l': ans = [0,0,0,1]

    return ans
```

```python
def create_training_data():

    training_data = []

    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_leaves(img)
        path = os.path.join(TRAIN_DIR,img)
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        try:
            img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        except cv2.error as e:
          print('Invalid frame!')

        training_data.append([np.array(img),np.array(label)])

    shuffle(training_data)
    np.save('train_data.npy', training_data)

    return training_data

def main():

    train_data = create_training_data()

    convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

    convnet = conv_2d(convnet, 32, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 64, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 128, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 32, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 64, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)
```

```python
    convnet = fully_connected(convnet, 1024, activation='relu')
    convnet = dropout(convnet, 0.8)

    convnet = fully_connected(convnet, 4, activation='softmax')
    convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

    model = tflearn.DNN(convnet, tensorboard_dir='log')

    if os.path.exists('{}.meta'.format(MODEL_NAME)):
        model.load(MODEL_NAME)
        print('Model Loaded')

    train = train_data[:-500]
    test = train_data[-500:]

    X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
    Y = [i[1] for i in train]

    test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
    test_y = [i[1] for i in test]

    model.fit({'input': X}, {'targets': Y}, n_epoch=8, validation_set=({'input': test_x}, {'targets': test_y}), snapshot_step=40, show_metric=True, run_id=MODEL_NAME)

    model.save(MODEL_NAME)

if __name__ == '__main__': main()
```

Image Processing File was developed in order to masking the image provided as an input which was not essential as the project progressed further and the concepts got much clearer.

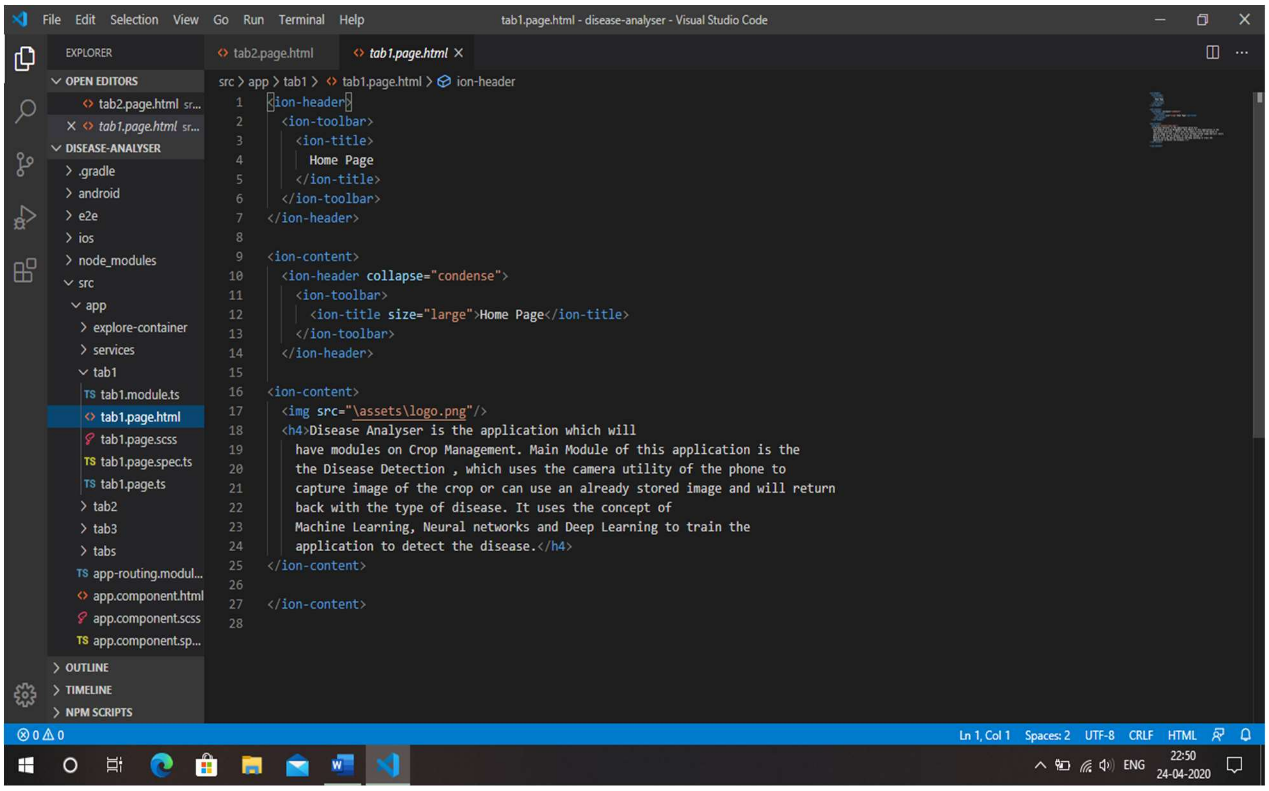

image_process.py - F:\A00268808_Final Project\Disease Analyser Backend\image_process.py (3.7.7)
File Edit Format Run Options Window Help
Ln: 1 Col: 0
ENG
22:26
24-04-2020

```python
import cv2
import numpy as np

def image_masking(filepath):

        BLUR = 21
        CANNY_THRESH_1 = 10
        CANNY_THRESH_2 = 200
        MASK_DILATE_ITER = 10
        MASK_ERODE_ITER = 10
        MASK_COLOR = (0.0,0.0,0.0) # In BGR format

        img = cv2.imread(filepath)
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

        edges = cv2.Canny(gray, CANNY_THRESH_1, CANNY_THRESH_2)
        edges = cv2.dilate(edges, None)
        edges = cv2.erode(edges, None)

        contour_info = []
        contours, __ = cv2.findContours(edges, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)

        for c in contours:
            contour_info.append((c, cv2.isContourConvex(c), cv2.contourArea(c),))
        contour_info = sorted(contour_info, key=lambda c: c[2], reverse=True)

        max_contour = contour_info[0]
        mask = np.zeros(edges.shape)
        cv2.fillConvexPoly(mask, max_contour[0], (255))

        mask = cv2.dilate(mask, None, iterations=MASK_DILATE_ITER)
        mask = cv2.erode(mask, None, iterations=MASK_ERODE_ITER)
        mask = cv2.GaussianBlur(mask, (BLUR, BLUR), 0)

        mask_stack = np.dstack([mask]*3)
        mask_stack  = mask_stack.astype('float32') / 255.0
        img = img.astype('float32') / 255.0

        masked = (mask_stack * img) + ((1-mask_stack) * MASK_COLOR)
        masked = (masked * 255).astype('uint8')


        fileName, fileExtension = filepath.split('.')
        fileName += '-masked.'
        filepath = fileName + fileExtension
        print (filepath)

        cv2.imwrite(filepath, masked)

if __name__ == '__main__':
        filepath = input("Enter Image File Name:\n")
        image_masking(filepath)
```

FRONTEND

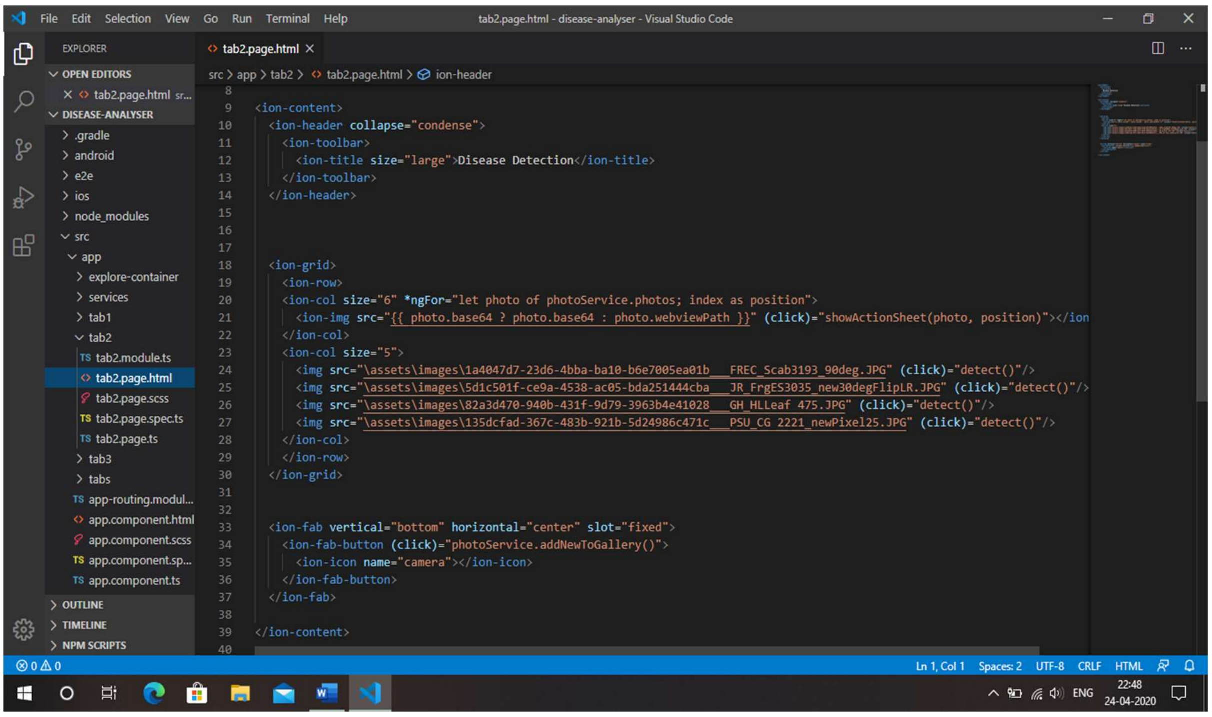
Main page of the mobile application where the camera utility is used to capture image and with the help of services declared image is stored and can be viewed in the gallery where If image is clicked, detect, delete and cancel options pop-up. If detect option is clicked then the image url will be passed to the detect function defined which generally uses API to provide it as an input to the python script.

HTML file of tab1 page : Introduction about the application



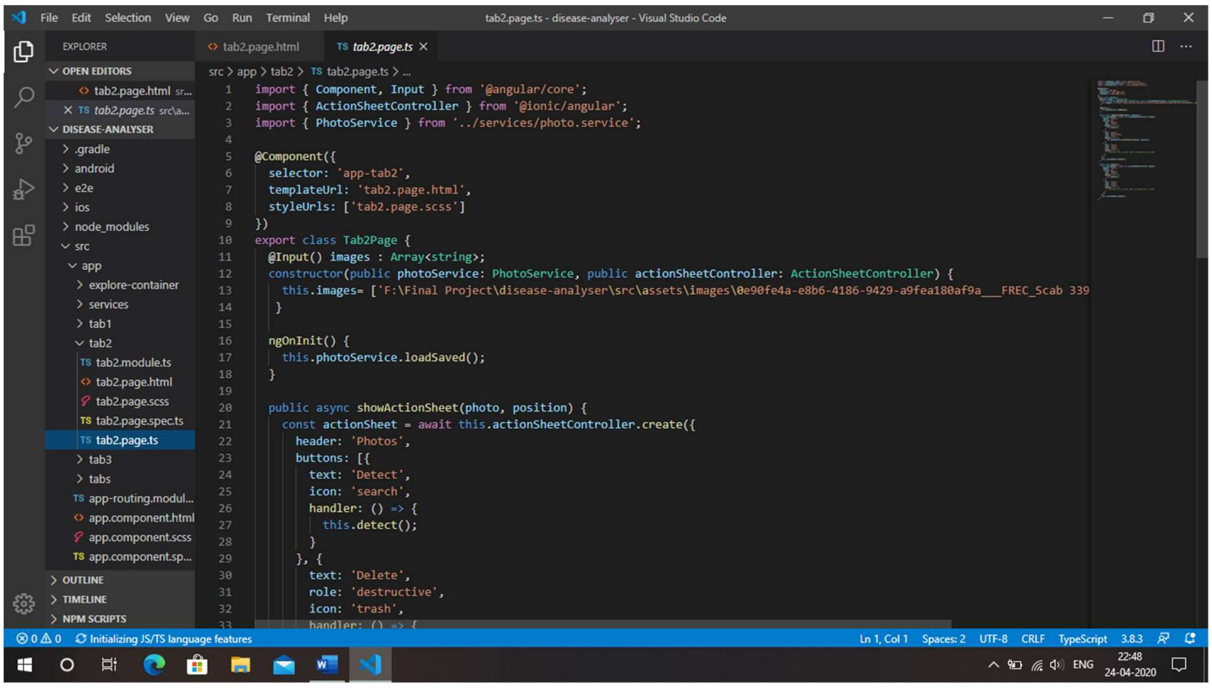
```html
<ion-header>
  <ion-toolbar>
    <ion-title>
      Home Page
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-header collapse="condense">
    <ion-toolbar>
      <ion-title size="large">Home Page</ion-title>
    </ion-toolbar>
  </ion-header>

<ion-content>
  <img src="\assets\logo.png"/>
  <h4>Disease Analyser is the application which will
    have modules on Crop Management. Main Module of this application is the
    the Disease Detection , which uses the camera utility of the phone to
    capture image of the crop or can use an already stored image and will return
    back with the type of disease. It uses the concept of
    Machine Learning, Neural networks and Deep Learning to train the
    application to detect the disease.</h4>
</ion-content>

</ion-content>
```

HTML file of tab2 page : Stored and captured image are displayed with options defined.

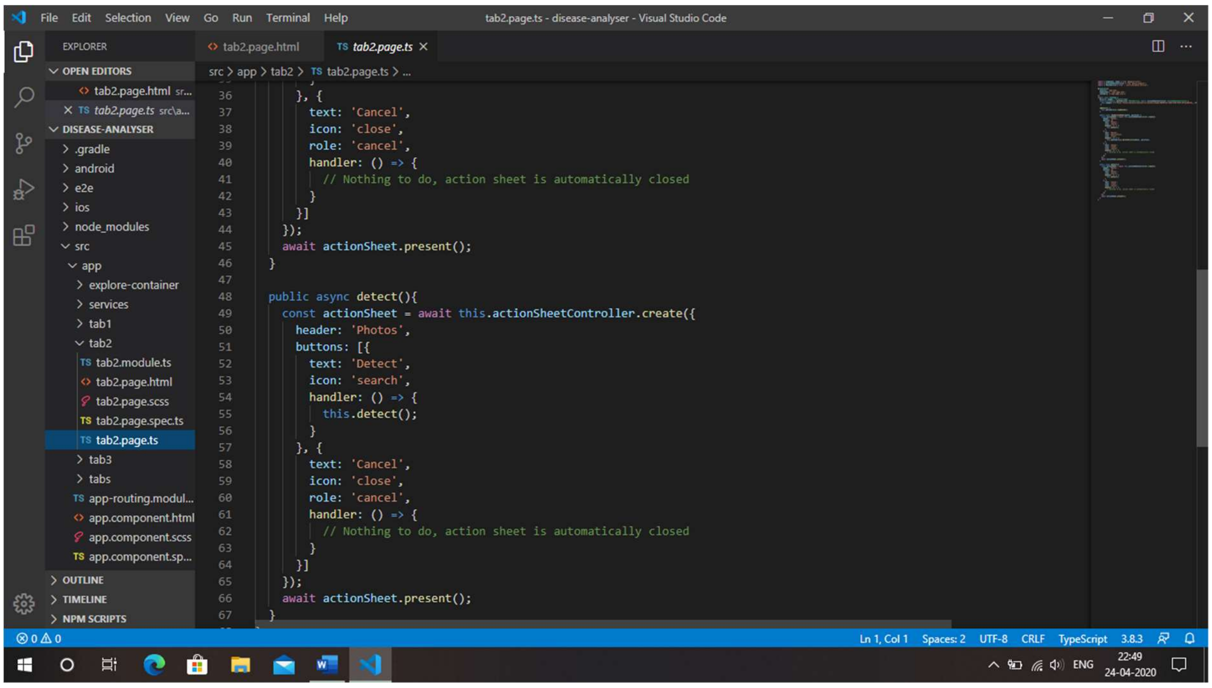


```html

<ion-content>
  <ion-header collapse="condense">
    <ion-toolbar>
      <ion-title size="large">Disease Detection</ion-title>
    </ion-toolbar>
  </ion-header>



  <ion-grid>
    <ion-row>
    <ion-col size="6" *ngFor="let photo of photoService.photos; index as position">
      <ion-img src="{{ photo.base64 ? photo.base64 : photo.webviewPath }}" (click)="showActionSheet(photo, position)"></ion
    </ion-col>
    <ion-col size="5">
      <img src="\assets\images\1a4047d7-23d6-4bba-ba10-b6e7005ea01b___FREC_Scab3193_90deg.JPG" (click)="detect()"/>
      <img src="\assets\images\5d1c501f-ce9a-4538-ac05-bda251444cba___JR_FrgES3035_new30degFlipLR.JPG" (click)="detect()"/>
      <img src="\assets\images\82a3d470-940b-431f-9d79-3963b4e41028___GH_HLLeaf 475.JPG" (click)="detect()"/>
      <img src="\assets\images\135dcfad-367c-483b-921b-5d24986c471c___PSU_CG 2221_newPixel25.JPG" (click)="detect()"/>
    </ion-col>
    </ion-row>
  </ion-grid>


  <ion-fab vertical="bottom" horizontal="center" slot="fixed">
    <ion-fab-button (click)="photoService.addNewToGallery()">
      <ion-icon name="camera"></ion-icon>
    </ion-fab-button>
  </ion-fab>

</ion-content>

```

Typescript file of Tab2 page: functions like detect, delete and cancel are defined.

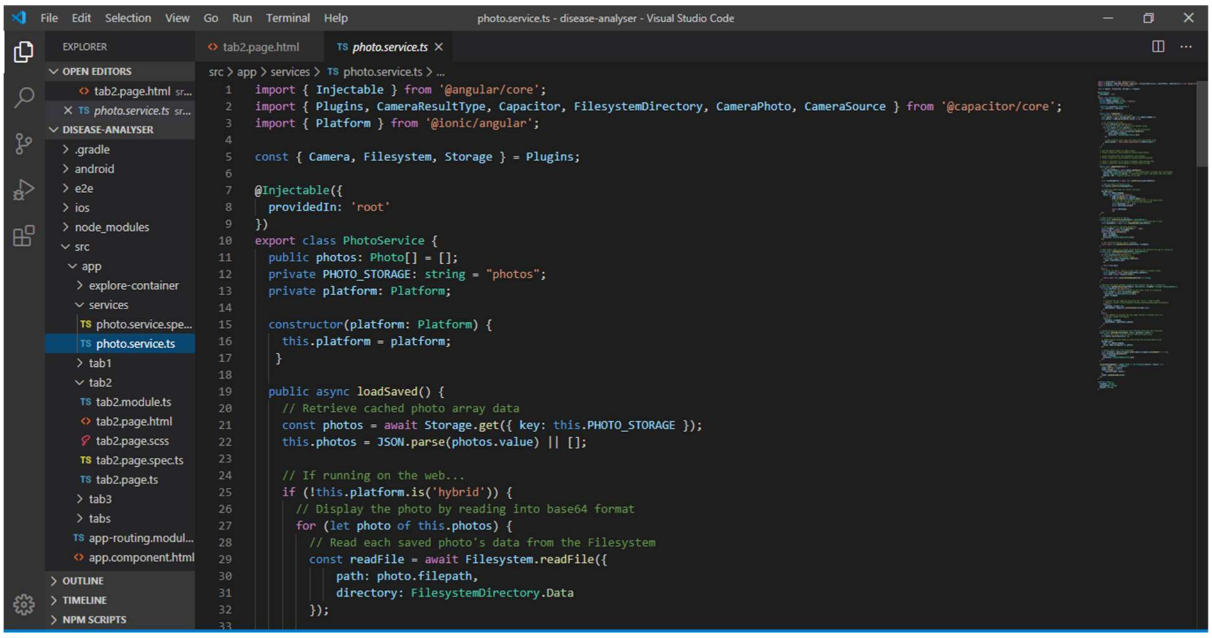


```
import { Component, Input } from '@angular/core';
import { ActionSheetController } from '@ionic/angular';
import { PhotoService } from '../services/photo.service';

@Component({
  selector: 'app-tab2',
  templateUrl: 'tab2.page.html',
  styleUrls: ['tab2.page.scss']
})
export class Tab2Page {
  @Input() images : Array<string>;
  constructor(public photoService: PhotoService, public actionSheetController: ActionSheetController) {
    this.images= ['F:\Final Project\disease-analyser\src\assets\images\0e90fe4a-e8b6-4186-9429-a9fea180af9a___FREC_Scab 339
   }

  ngOnInit() {
    this.photoService.loadSaved();
  }

  public async showActionSheet(photo, position) {
    const actionSheet = await this.actionSheetController.create({
      header: 'Photos',
      buttons: [{
        text: 'Detect',
        icon: 'search',
        handler: () => {
          this.detect();
        }
      }, {
        text: 'Delete',
        role: 'destructive',
        icon: 'trash',
        handler: () => {
```

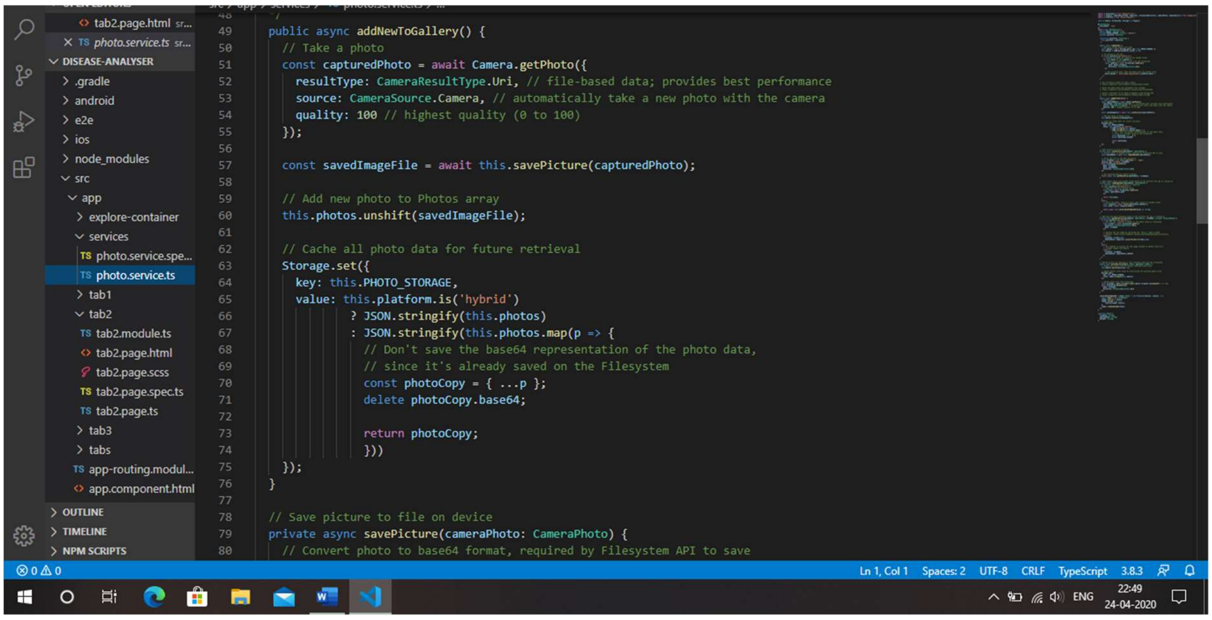


```
      }, {
        text: 'Cancel',
        icon: 'close',
        role: 'cancel',
        handler: () => {
          // Nothing to do, action sheet is automatically closed
        }
      }]
    });
    await actionSheet.present();
  }

  public async detect(){
    const actionSheet = await this.actionSheetController.create({
      header: 'Photos',
      buttons: [{
        text: 'Detect',
        icon: 'search',
        handler: () => {
          this.detect();
        }
      }, {
        text: 'Cancel',
        icon: 'close',
        role: 'cancel',
        handler: () => {
          // Nothing to do, action sheet is automatically closed
        }
      }]
    });
    await actionSheet.present();
  }
```

Typescript file of photo.service : services are defined to use the camera to capture the image, store it and display it the photo gallery.

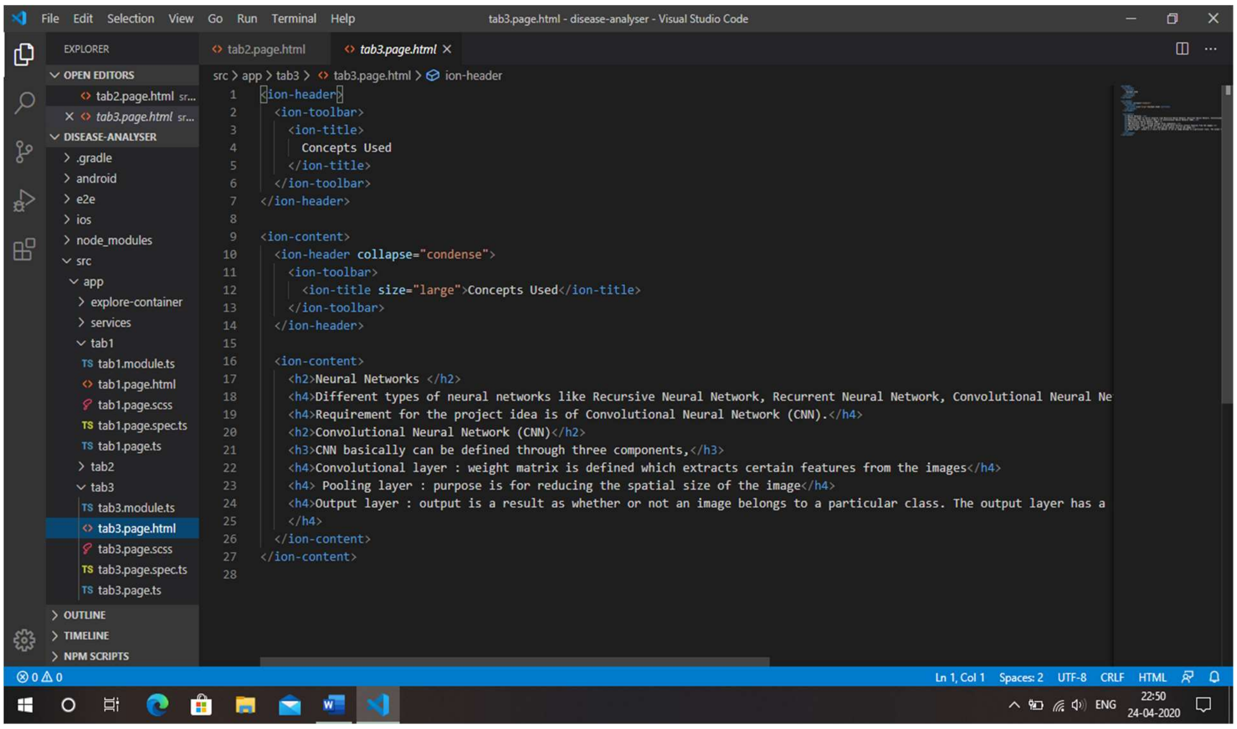


```
import { Injectable } from '@angular/core';
import { Plugins, CameraResultType, Capacitor, FilesystemDirectory, CameraPhoto, CameraSource } from '@capacitor/core';
import { Platform } from '@ionic/angular';

const { Camera, Filesystem, Storage } = Plugins;

@Injectable({
  providedIn: 'root'
})
export class PhotoService {
  public photos: Photo[] = [];
  private PHOTO_STORAGE: string = "photos";
  private platform: Platform;

  constructor(platform: Platform) {
    this.platform = platform;
  }

  public async loadSaved() {
    // Retrieve cached photo array data
    const photos = await Storage.get({ key: this.PHOTO_STORAGE });
    this.photos = JSON.parse(photos.value) || [];

    // If running on the web...
    if (!this.platform.is('hybrid')) {
      // Display the photo by reading into base64 format
      for (let photo of this.photos) {
        // Read each saved photo's data from the Filesystem
        const readFile = await Filesystem.readFile({
            path: photo.filepath,
            directory: FilesystemDirectory.Data
        });
```



```
  public async addNewToGallery() {
    // Take a photo
    const capturedPhoto = await Camera.getPhoto({
      resultType: CameraResultType.Uri, // file-based data; provides best performance
      source: CameraSource.Camera, // automatically take a new photo with the camera
      quality: 100 // highest quality (0 to 100)
    });

    const savedImageFile = await this.savePicture(capturedPhoto);

    // Add new photo to Photos array
    this.photos.unshift(savedImageFile);

    // Cache all photo data for future retrieval
    Storage.set({
      key: this.PHOTO_STORAGE,
      value: this.platform.is('hybrid')
              ? JSON.stringify(this.photos)
              : JSON.stringify(this.photos.map(p => {
                // Don't save the base64 representation of the photo data,
                // since it's already saved on the Filesystem
                const photoCopy = { ...p };
                delete photoCopy.base64;

                return photoCopy;
                }))
    });
  }

  // Save picture to file on device
  private async savePicture(cameraPhoto: CameraPhoto) {
    // Convert photo to base64 format, required by Filesystem API to save
```

HTML file of tab3 page : Short description of Concepts used while building the application.



```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Concepts Used
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-header collapse="condense">
    <ion-toolbar>
      <ion-title size="large">Concepts Used</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content>
    <h2>Neural Networks </h2>
    <h4>Different types of neural networks like Recursive Neural Network, Recurrent Neural Network, Convolutional Neural Ne
    <h4>Requirement for the project idea is of Convolutional Neural Network (CNN).</h4>
    <h2>Convolutional Neural Network (CNN)</h2>
    <h3>CNN basically can be defined through three components,</h3>
    <h4>Convolutional layer : weight matrix is defined which extracts certain features from the images</h4>
    <h4> Pooling layer : purpose is for reducing the spatial size of the image</h4>
    <h4>Output layer : output is a result as whether or not an image belongs to a particular class. The output layer has a
    </h4>
  </ion-content>
</ion-content>
```

8. Testing And Evaluation

As the application was developed into individual parts, frontend was designed with the help of Ionic and backend was designed with the help of general purpose, versatile programming language i.e. python. At First, there were various difficulties due to my little experience with ionic and python which made linking the frontend with the backend a relatively difficult task. The approach used to build the application was to load the python script in the application with the help of plugin or any other medium defined and further which will load the tensorflow model. Satisfaction with the individually working and giving in the required output, integration phase started to integrate the individual modules.

Integrating the ionic application with python script turned out to be hardest part of the project as ionic does not have any plugin to load the main script which was the essential link between frontend and backend. Struggling to load the script come to an end when a solution was achieved with the help of college teachings and online sources. Solution was to develop an API which will pass the image data to the script and it was designed with the help of Python and the Flask web framework.

Testing the application for acceptance but images with different colour contrast was tough part. To overcome this, cnn model was trained with these kinds of images in larger batches for better efficiency.

At this point working mobile application completed, to implement convolutional neural network which was a major part. After the system and acceptance testing for the end product the application meets the requirements and specifications that were set out in the research and design phase of the project build. The application is ready to function as a hybrid application.

While functional requirement was straight "To detect the crops disease with the help of leaf's image" building this application took many sharp turns around non-functional requirements which included setting up TensorFLow 2 and then migrating to TensorFlow 1. There were certain miscalculations which were either due to the inexperience with technologies or inaccurate planning for non-functional need of project which resulted in extra time and for some instances redoing the work. This was all part of project but gave a better understanding and an avid learning for future project that will be pursing in longer run. All functional and non-functional requirements were met and satisfied to all levels but would have been done more efficiently if had enough experience with some interim technologies and planning phases that were part of the project.

Design and technology decisions made for the development of the project were precise and fulfil the demands while complying with best technologies available for the intended application. While design solely depends upon the customer needs in accordance to the look and feel of the application the basic architecture running behind the application would not change.

LESSONS LEARNT

➔ Migrated from TensorFlow 2 to TensorFlow 1 because of the high-level API i.e. tflearn TfLearn syntax seemed a little cleaner and was better to understand. Some benefits of tflearn are

  - Can save the models as checkpoint, index, and meta files, which can used to create a frozen version of the model quite easily (frozen models are very important if want to be able to use them in your Android apps).
  - Easy-to-use and understand high-level API for implementing deep neural networks, with tutorial and examples.
  - Full transparency over Tensorflow. All functions are built over tensors and can be used independently of TFLearn.
  - Powerful helper functions to train any TensorFlow model, with support of multiple inputs, outputs and optimizers.
  - Its API is closer to that of TensorFlow. At any point, one can quickly switch from writing TFLearn code to TensorFlow code, and won't face any problems.
  - It seems to offer slightly better performance than Keras.
  - Used to train with large datasets.

➔ Connecting the backend with the frontend,

Struggled to load the python script to the ionic application as it does not have any plugin related to this raised issue.

Solution was to develop an API which will pass the image data to the script and it was designed with the help of Python and the Flask web framework.

FUTURE PLANS

- ➔ For better user experience and future scenario, I will host the python script and tensorflow model on cloud and with the help of API I will pass the image to the script which will respond back with the disease. It will be good for the future scenario development of model with large datasets to detect large number of crop diseases.

- ➔ For future development of app, some functionalities will be,

  - Crop disease will be detected more accurately with the help of RCNN which is Regional Convolutional Neural Network which will further help in manipulating the captured images

  - Once the crop disease is detected the App will provide further functionality which will be the 'solutions to this disease and all details regarding it'

CONCLUSION

At the outset of this project I had a set of goals that I felt to be needed to achieve in order for this project to be a success. Firstly, I decide that the aim of my project was to apply a software engineering solution to the agriculture industry. Choosing Crop Disease Analyser as I felt that this could benefit the farmers and could be used anywhere irrelevant of agriculture sector. I decided that this application could provide a easier way to detect the crop's disease.

Based on the project idea, I identified that how I will approach the design. Planned on developing this application with the help of Cross-Platform Mobile App development incorporating python script which will load the CNN model into the application. I felt that this project would advance my knowledge and skill as regards design, development and further understanding the complete software project lifecycle. Being new to this technology stack I had very little experience in python and angularjs, so I felt this project would add to my honours degree.

The little knowledge of python and angularjs, development proved to be redundant as the level needed to develop the system was far beyond the development I had previously completed. I enjoyed learning a lot of the new principals of python development and was amazed at the possibilities of what can be achieved with machine learning. I started with learning the basics and before long I was developing with advanced concepts of machine learning.

At the outset of this project I had no working technical knowledge of machine learning. The reason I decided to incorporate machine learning into project was that I felt it is a very relevant technology with huge possibility for advancement in the future. It also gives application ease of scalability which is vital for any user driven application in the modern technology age.

REFRENCES

- http://neuralnetworksanddeeplearning.com/chap1.html
- https://towardsai.net/p/machine-learning/what-is-machine-learning-how-does-it-work-and-why-is-it-important/robiriondo/2019/04/30/
- https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-f7d02aa9d477
- https://www.mathworks.com/discovery/deep-learning.html
- https://pathmind.com/wiki/neural-network
- https://towardsdatascience.com/introducing-convolutional-neural-networks-in-deep-learning-400f9c3ad5e9
- https://www.tensorflow.org/
- https://ionicframework.com/
- https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask