A Gesture-based Tool for Sterile Browsing of Radiology Images

[Gesture Based Automation Tool]

Industry Mentor Name   PRADEEPTHI

Faculty Mentor(s) Name: SIVASANKARI K

Project Group Name:
AGesturebasedToolforSterileBrowsingof RadiologyImages41GP

Team id: **PNT2022TMID20976**

Team Members:

PRAKASH KUMAR S	-- 412419106051
NITHISH N	-- 412419106048
NAVEENVISHAL M	-- 412419106046
RISHI RITVIK RAJ A	-- 412419106062
PRASANNA S	-- 412419106046

INTRODUCTION

- Naturally Humans can recognize body and sign language easily. This can be done because of the combination of vision and synaptic interactions that were formed along brain development activities.

- In this project, the model is first trained on the images of different gestures, such as showing numbers using our fingers as in the order of 1,2,3,4.

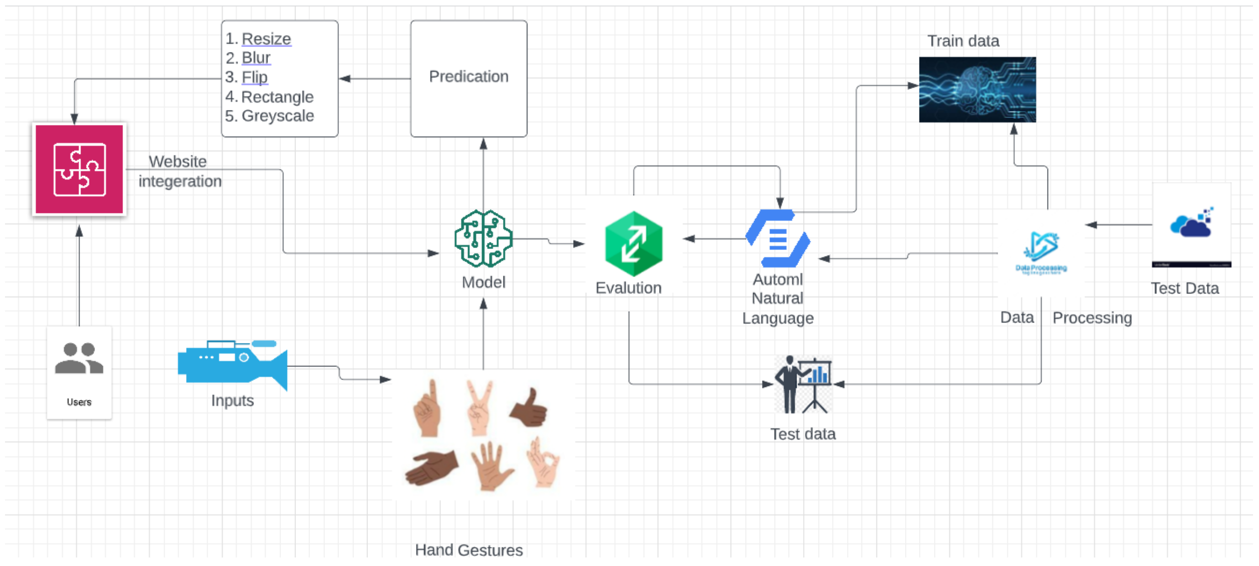
- This model uses the integrated camera that is webcam to capture the video frame. The image captured in the video frame is compared with the pre trained model that was captured before, and the gesture is identified.

- If the gesture identifies or predicts that the number is 1 then image gets blurred; If the number is 2 then the image is resized; If the number is 3 then the image gets rotated etc.,

PROJECT OBJECTIVES

- Know how to build a web application platform using Flask Framework.
- The main objective of this project is to know the fundamental and technical concepts of Convolutional Neural Network [CNN].
- Know how to clean the data using different data preprocessing techniques.
- Elaborate understanding of image data is obtained.

PROJECT ARCHITECTURE

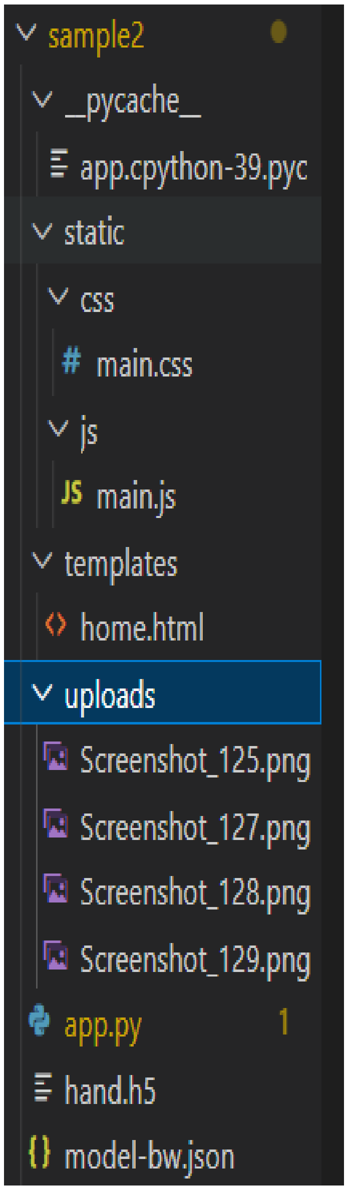


PROJECT FLOW

To accomplish this, we have to complete all the activities and tasks listed below:

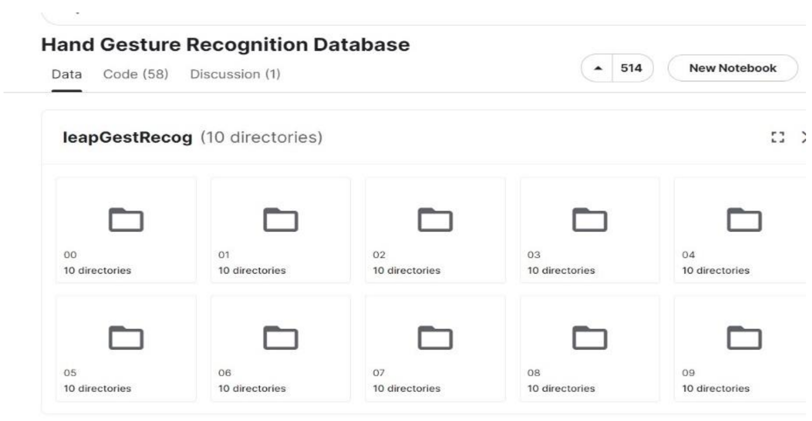
- ***Data Collection.***
    - ***Collect the dataset or create the dataset***
- ***Data Preprocessing***
    - ***Import the ImageDataGenerator library***
    - ***Configure ImageDataGenerator class***
    - ***Apply ImageDataGenerator functionality to Trainset and Test set***
- ***Model Building***
    - ***Import the model building Libraries***
    - ***Initializing the model***
    - ***Adding Input Layer***
    - ***Adding Hidden Layer***
    - ***Adding Output Layer***
    - ***Configure the Learning Process***
    - ***Training and testing the model***
    - ***Save the Model***
- ***Application Building***
    - ***Create an HTML file***
    - ***Build Python Code***

PROJECT STRUCTURE



```
sample2
  _pycache_
    app.cpython-39.pyc
  static
    css
      main.css
    js
      main.js
  templates
    home.html
  uploads
    Screenshot_125.png
    Screenshot_127.png
    Screenshot_128.png
    Screenshot_129.png
  app.py                1
  hand.h5
  model-bw.json
```

DOWNLOAD DATA-SET AND IMAGE PRE-PROCESSING



- A Dataset is a set or collection of data. This set is normally presented in a tabular pattern. Every column describes a particular variable. And each row corresponds to a given member of the data set, as per the given question. This is a part of data management.
- In this project, we are willing to improve the data that clears the unwanted distortions.

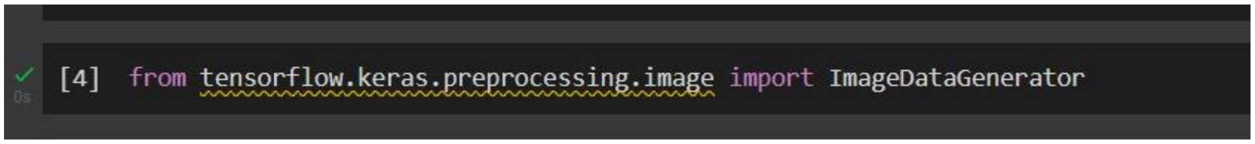
- But in other side, it enhances some important features for further processing purpose.

- Although, it performs some geometric transformations of images like scaling, translation, and rotation.

IMPORT THE IMAGE DATA GENERATOR LIBRARY

- Image data augmentation is a technique that can be used artificially to expand the size of a dataset by creating modified versions of images present in the dataset.

- The Kera's deep learning neural network contains a library provides the capability to fit models using image data augmentation through the ImageDataGenerator Class.



```
[4]  from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

CONFIGURE IMAGE DATA GENERATOR CLASS

- The ImageDataGenerator Class is developed and the configuration for the types of data augmentation.

- There are five main types of data augmentation Techniques. They are

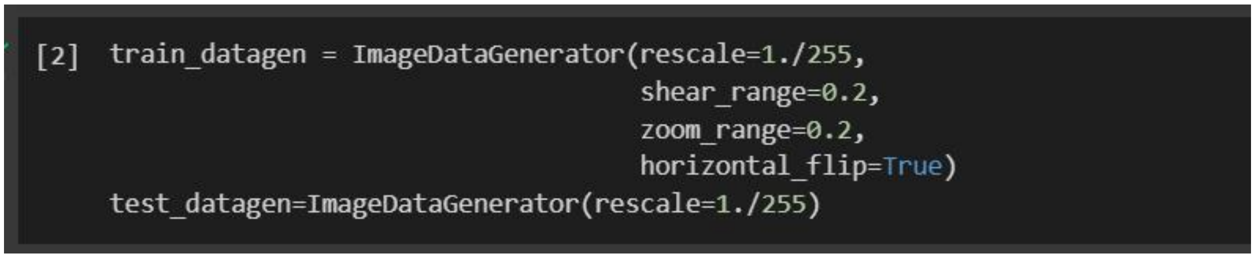
1) Image shifts via the width_shift_range and height_shift_range arguments.

2) Image flips via the horizontal_flip and vertical_flip arguments.

3) Image rotations via the rotation_range arguments.

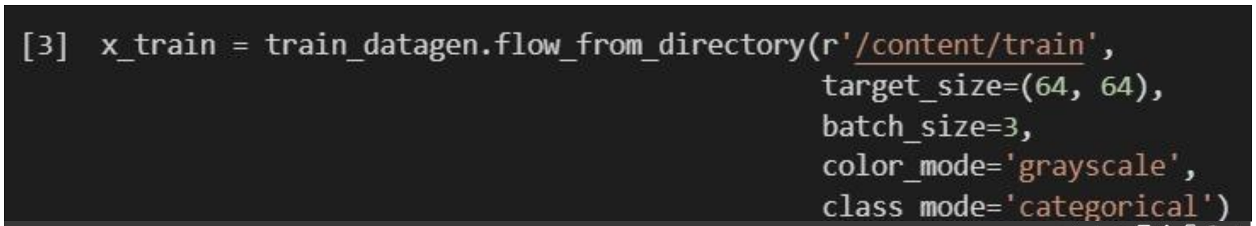
4) Image brightness via the brightness_range arguments.

5) Image zooms via the zoom_range arguments.

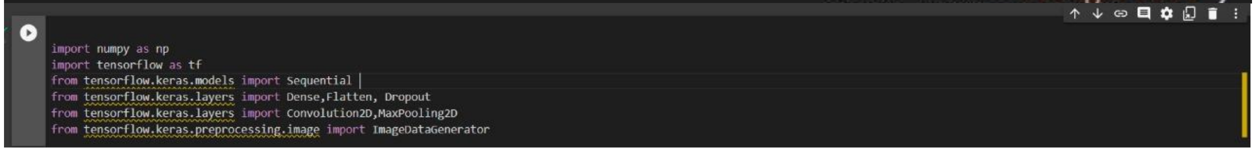


```
[2]  train_datagen = ImageDataGenerator(rescale=1./255,
                                        shear_range=0.2,
                                        zoom_range=0.2,
                                        horizontal_flip=True)
     test_datagen=ImageDataGenerator(rescale=1./255)
```

APPLYING IMAGE DATA GENERATOR FUNCTIONALITY TO TRAINSET AND TO TESTSET



```
[3]  x_train = train_datagen.flow_from_directory(r'/content/train',
                                                  target_size=(64, 64),
                                                  batch_size=3,
                                                  color_mode='grayscale',
                                                  class_mode='categorical')
```



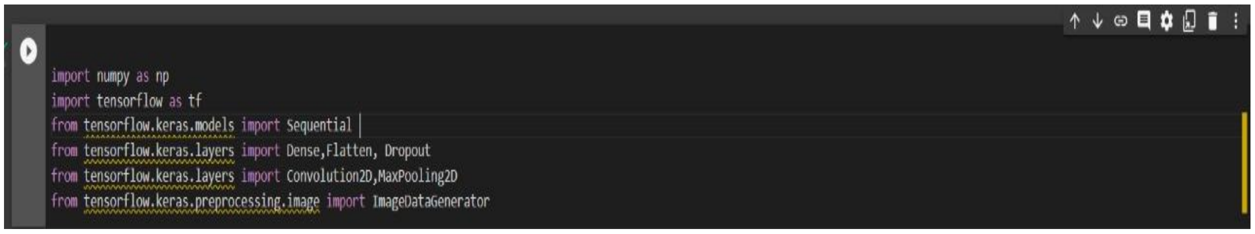
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten, Dropout
from tensorflow.keras.layers import Convolution2D,MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

- This is the function which returns the batches of images from the subdirectories 0,1,2,3,4,5 together with labels 0 to 5 {‘0’: 0, ‘1’ : 1, ‘2’ : 2, ‘3’ : 3, ‘4’ : 4, ‘5’ : 5 }.

MODEL BUILDING

- Here, the building of our Convolutional Neural Networking is done which contains a input layer along with convolution, maxpooling and finally an output layer part.

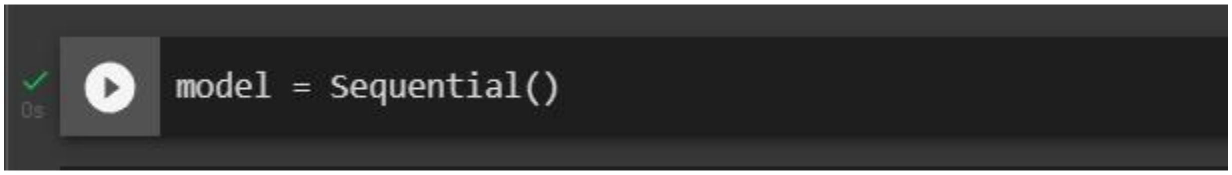
IMPORTING THE MODEL BUILDING LIBRARIES



```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten, Dropout
from tensorflow.keras.layers import Convolution2D,MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

- Importing the necessary modules needed for the project.

INITIALIZING THE MODEL



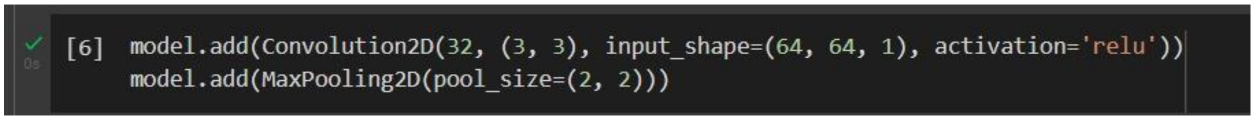
```
model = Sequential()
```

- The above figure represents sequential model. It is a linear stack of layers.

- We can create a sequential model by passing a list of layer instances to the constructor from keras models import Sequential from keras.

ADDING CNN LAYERS

- We are adding a convolution layer with activation function as “relu” and with a small filter size (3,3) and number of filters (32) followed by a max pooling layer.

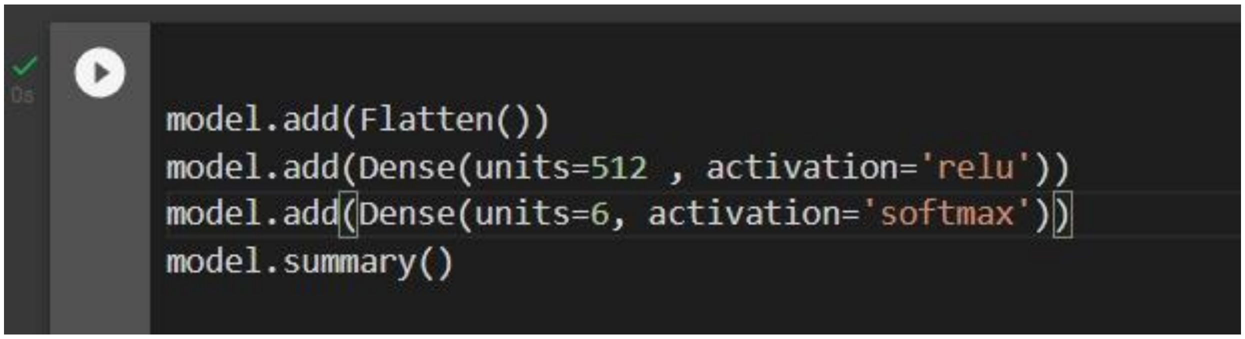


```
[6] model.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
```

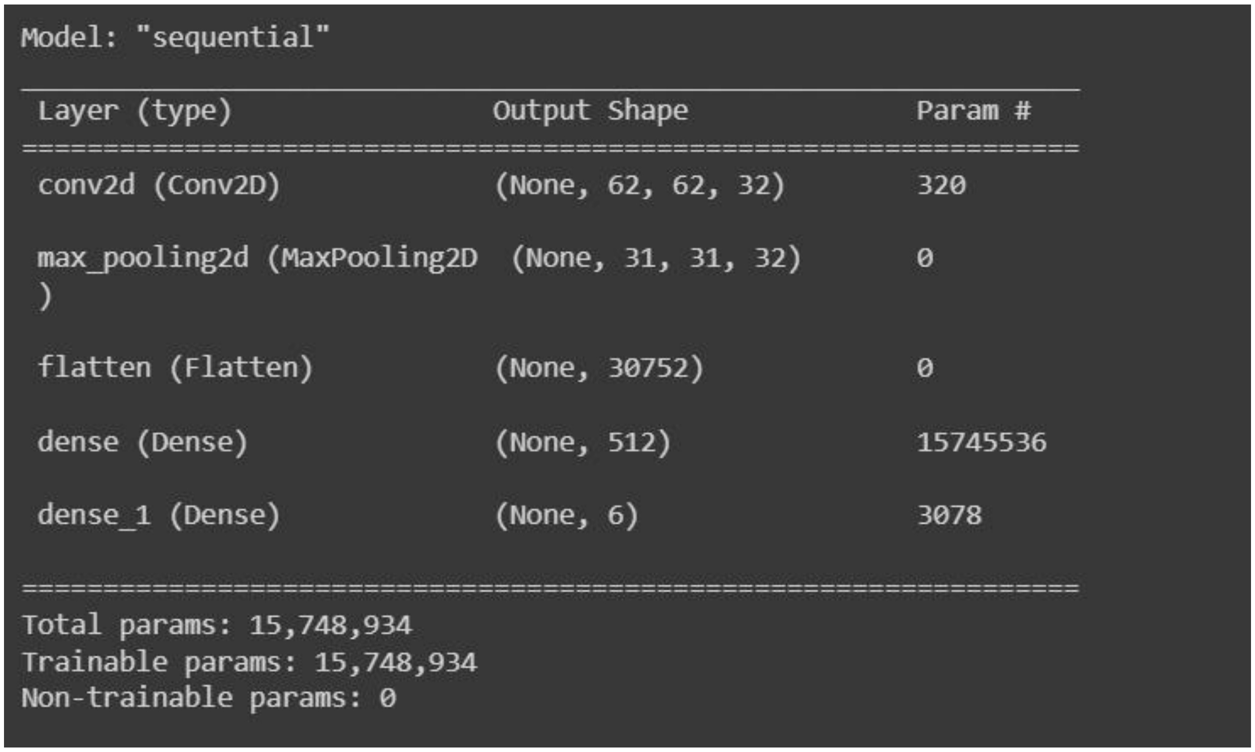
- There are two types of CNN Layers. They are

1) Maxpool Layer - Used to down sample the input.
2) Flatten Layer - Used to flattens the output.

ADDING DENSE LAYERS



```
model.add(Flatten())
model.add(Dense(units=512 , activation='relu'))
model.add(Dense(units=6, activation='softmax'))
model.summary()
```



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 flatten (Flatten)           (None, 30752)             0

 dense (Dense)               (None, 512)               15745536

 dense_1 (Dense)             (None, 6)                 3078

=================================================================
Total params: 15,748,934
Trainable params: 15,748,934
Non-trainable params: 0
```

- First, Dense layer is deeply connected neural network layer. It is most common and frequently used layer.

- Understanding the model is very important phase to properly use it for training and prediction related purpose.

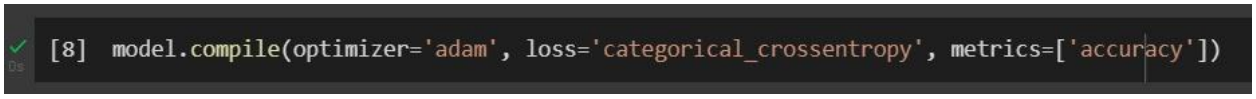
- That’s why keras platform is used. It provides a simple method, summary to get the full information about the models and its layers.

CONFIGURE THE LEARNING PROCESS

- The compilation is the final step of learning process in creating the model. Once the compilation is done, we can move on to the training process.

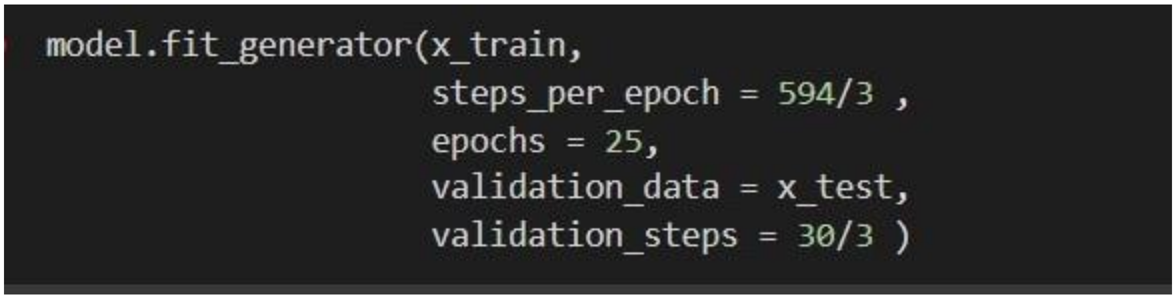
- Optimization is an important process which optimizes the input weights by comparing the prediction and the loss function.

- Metrics is used to evaluate the performance of the model. It is like loss function, but not used in training process.



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

TRAIN THE MODEL



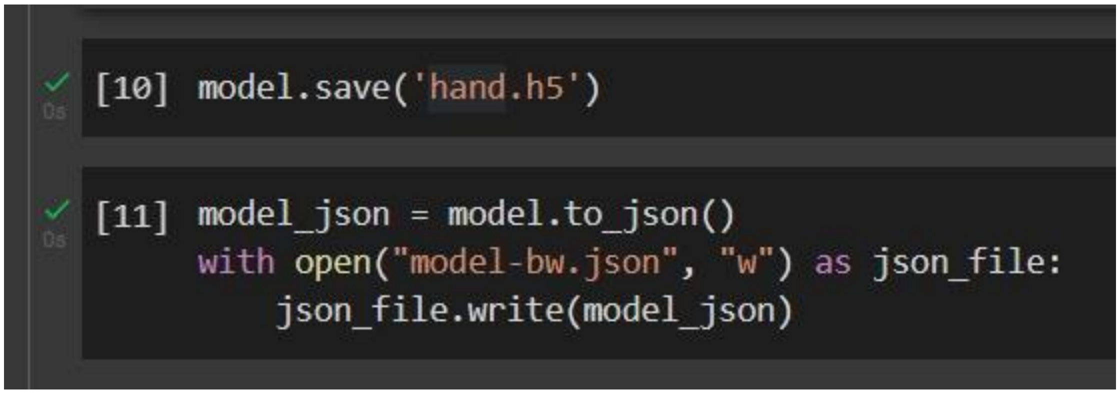
```
model.fit_generator(x_train,
                    steps_per_epoch = 594/3 ,
                    epochs = 25,
                    validation_data = x_test,
                    validation_steps = 30/3 )
```

- Fit_generator functions are used to train a deep learning neural network

- Steps_per_epoch: It specifies the total number of steps taken from the generator as soon as one epoch is finished and next epoch has started. Epochs: an integer and number of epochs we want to train our model for Validation_data can be either •An inputs and targets list •A generator •An inputs, targets and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.

SAVE THE MODEL

- Saving the model is an important part.
- The model is saved with .h5 extension.
- An H5 file is nothing but a data file saved in the Hierarchical Data Format (HDF).
- It contains multidimensional arrays of scientific data.

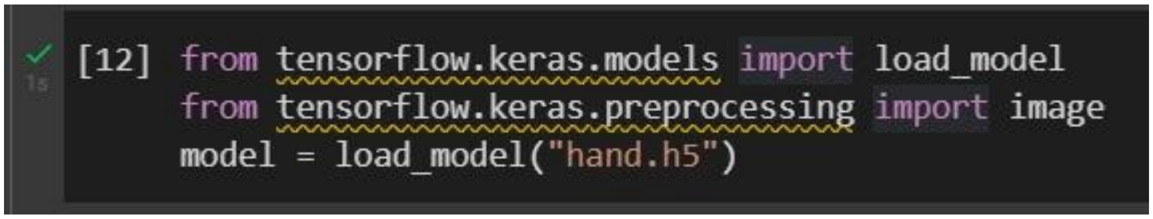


```
[10]  model.save('hand.h5')
```

```
[11]  model_json = model.to_json()
      with open("model-bw.json", "w") as json_file:
          json_file.write(model_json)
```

TEST THE MODEL

- Test part is also an important part after saving the model.
- Evaluation or Testing is a process during development of the model to check whether the model is best fit or not for the given problem and corresponding data.

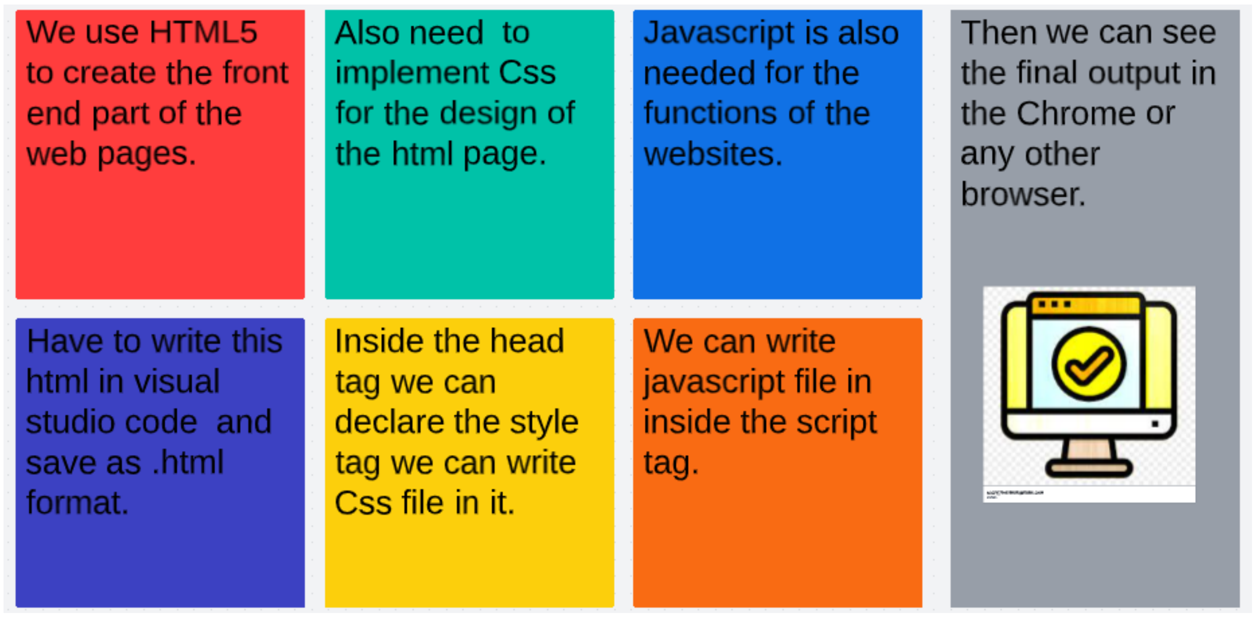


```
[12] from tensorflow.keras.models import load_model
     from tensorflow.keras.preprocessing import image
     model = load_model("hand.h5")
```

APPLICATION BUILDING

- After the model is trained in this area, we will be able to build our flask application which will be running in our local browser with a user interface.

CREATE HTML PAGES



We use HTML5 to create the front end part of the web pages.

Also need to implement Css for the design of the html page.

Javascript is also needed for the functions of the websites.

Then we can see the final output in the Chrome or any other browser.

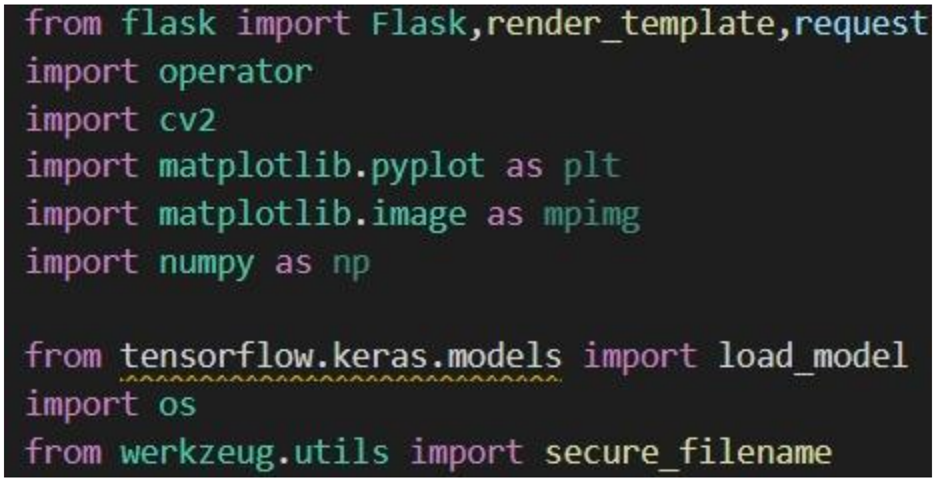
Have to write this html in visual studio code and save as .html format.

Inside the head tag we can declare the style tag we can write Css file in it.

We can write javascript file in inside the script tag.

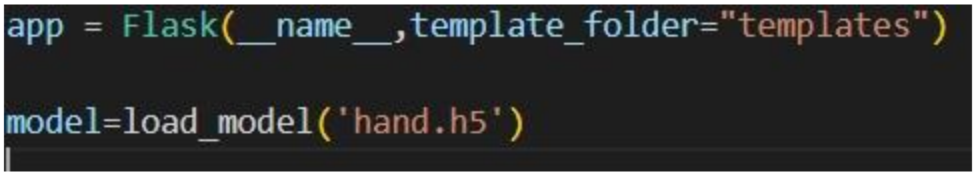
BUILD PYTHON CODE

Importing Libraries:



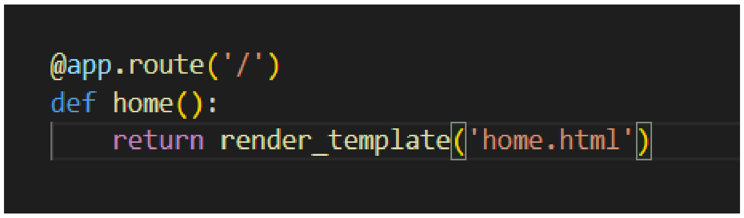
```python
from flask import Flask,render_template,request
import operator
import cv2
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np

from tensorflow.keras.models import load_model
import os
from werkzeug.utils import secure_filename
```

Creating our flask applications and loading our model in this file:



```python
app = Flask(__name__,template_folder="templates")

model=load_model('hand.h5')
```

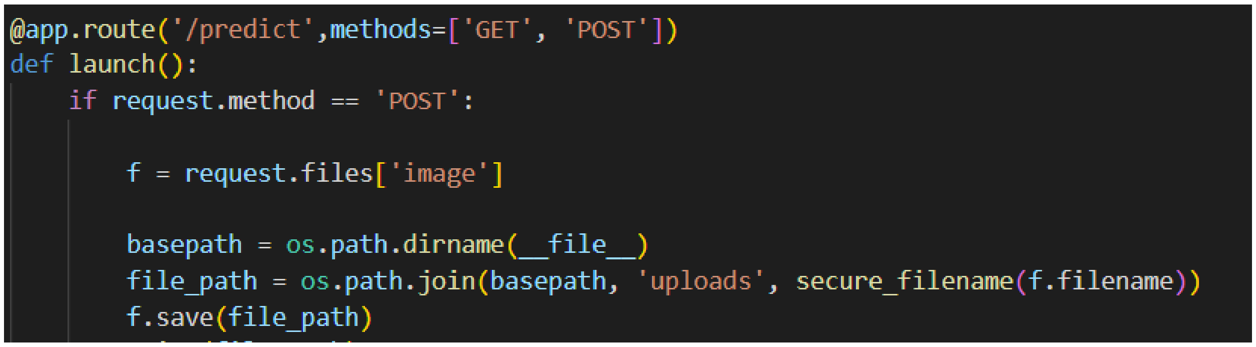
Routing to the html page



```python
@app.route('/')
def home():
    return render_template('home.html')
```

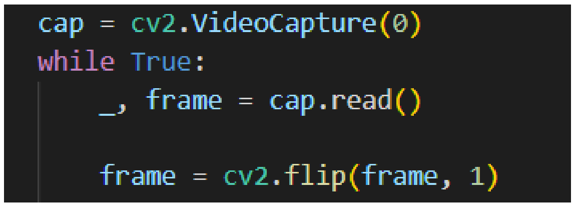
CODE

Get the input and store it:



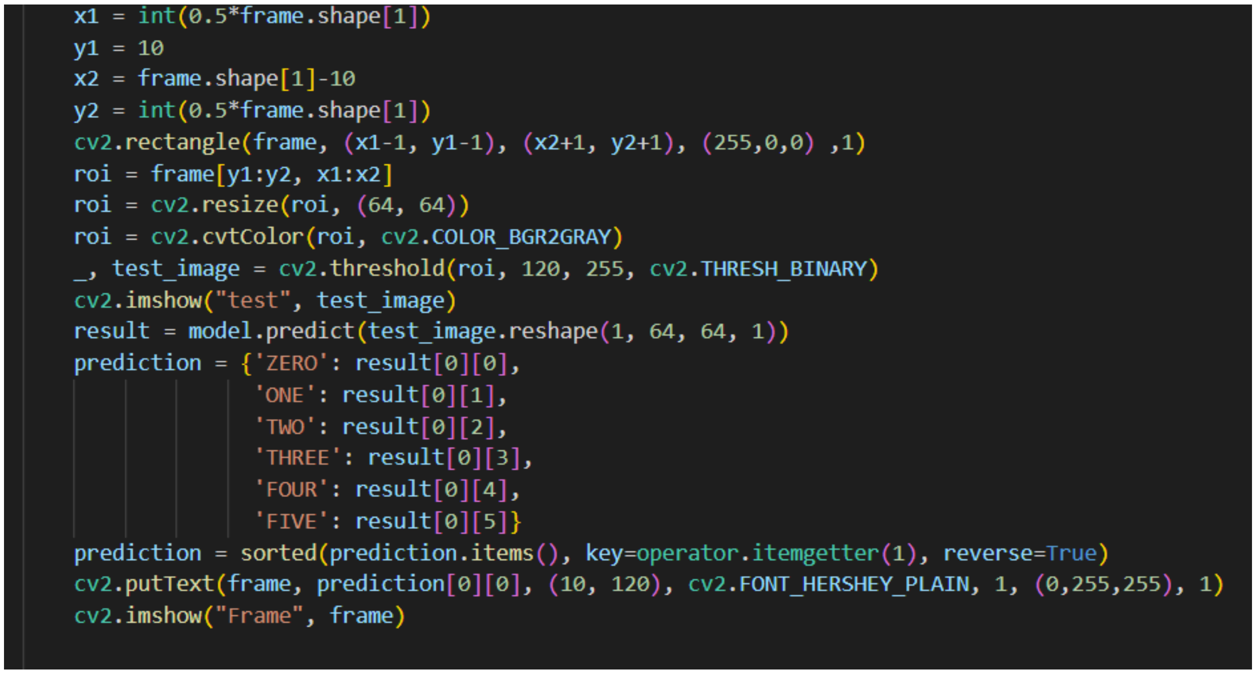
```python
@app.route('/predict',methods=['GET', 'POST'])
def launch():
    if request.method == 'POST':

        f = request.files['image']

        basepath = os.path.dirname(__file__)
        file_path = os.path.join(basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
```

Adjust the frames from camera stream:



```python
cap = cv2.VideoCapture(0)
while True:
    _, frame = cap.read()

    frame = cv2.flip(frame, 1)
```

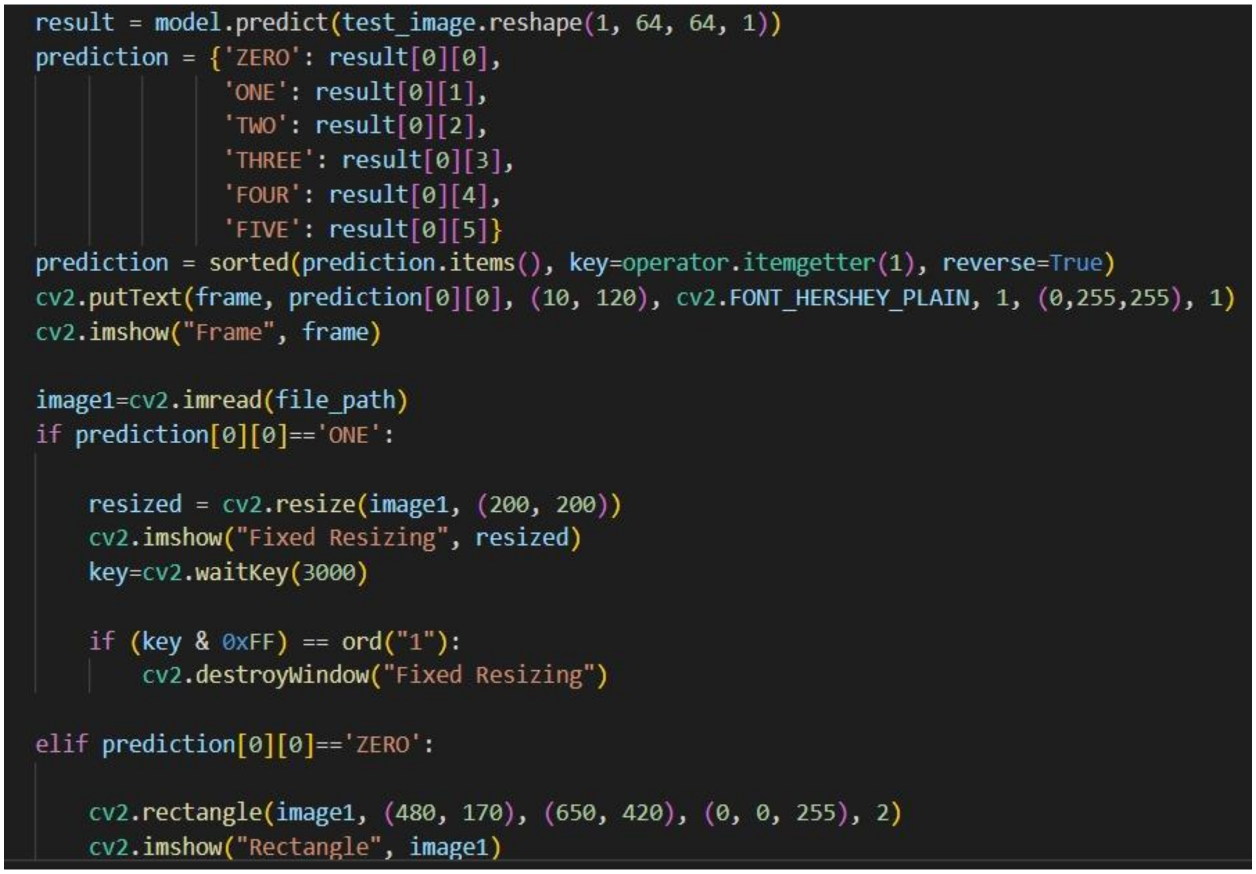
Create ROI:



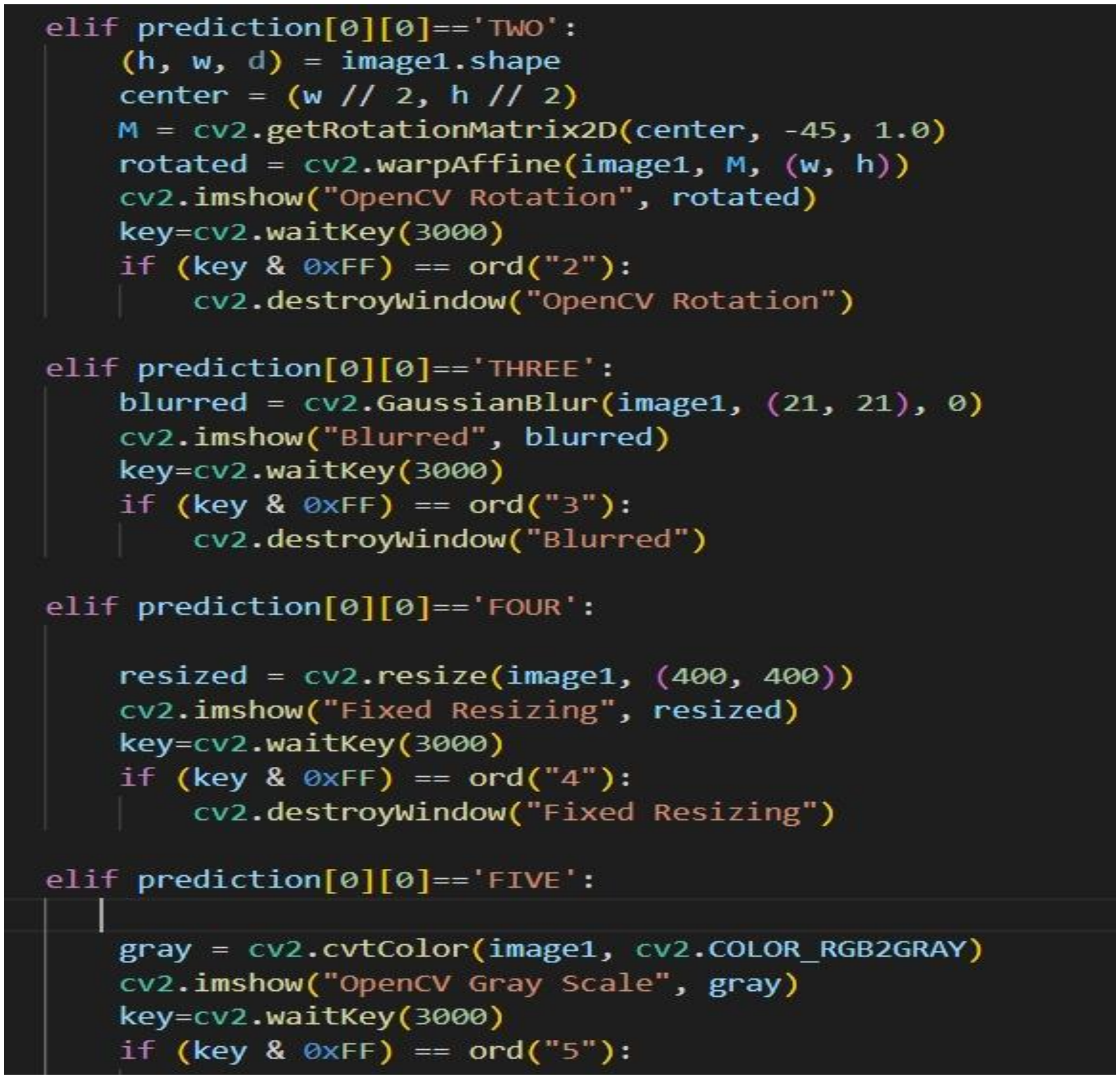
```python
    x1 = int(0.5*frame.shape[1])
    y1 = 10
    x2 = frame.shape[1]-10
    y2 = int(0.5*frame.shape[1])
    cv2.rectangle(frame, (x1-1, y1-1), (x2+1, y2+1), (255,0,0) ,1)
    roi = frame[y1:y2, x1:x2]
    roi = cv2.resize(roi, (64, 64))
    roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    _, test_image = cv2.threshold(roi, 120, 255, cv2.THRESH_BINARY)
    cv2.imshow("test", test_image)
    result = model.predict(test_image.reshape(1, 64, 64, 1))
    prediction = {'ZERO': result[0][0],
                  'ONE': result[0][1],
                  'TWO': result[0][2],
                  'THREE': result[0][3],
                  'FOUR': result[0][4],
                  'FIVE': result[0][5]}
    prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
    cv2.putText(frame, prediction[0][0], (10, 120), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    cv2.imshow("Frame", frame)
```

CODE

Predict the result:



```python
result = model.predict(test_image.reshape(1, 64, 64, 1))
prediction = {'ZERO': result[0][0],
              'ONE': result[0][1],
              'TWO': result[0][2],
              'THREE': result[0][3],
              'FOUR': result[0][4],
              'FIVE': result[0][5]}
prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
cv2.putText(frame, prediction[0][0], (10, 120), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
cv2.imshow("Frame", frame)

image1=cv2.imread(file_path)
if prediction[0][0]=='ONE':

    resized = cv2.resize(image1, (200, 200))
    cv2.imshow("Fixed Resizing", resized)
    key=cv2.waitKey(3000)

    if (key & 0xFF) == ord("1"):
        cv2.destroyWindow("Fixed Resizing")

elif prediction[0][0]=='ZERO':

    cv2.rectangle(image1, (480, 170), (650, 420), (0, 0, 255), 2)
    cv2.imshow("Rectangle", image1)
```

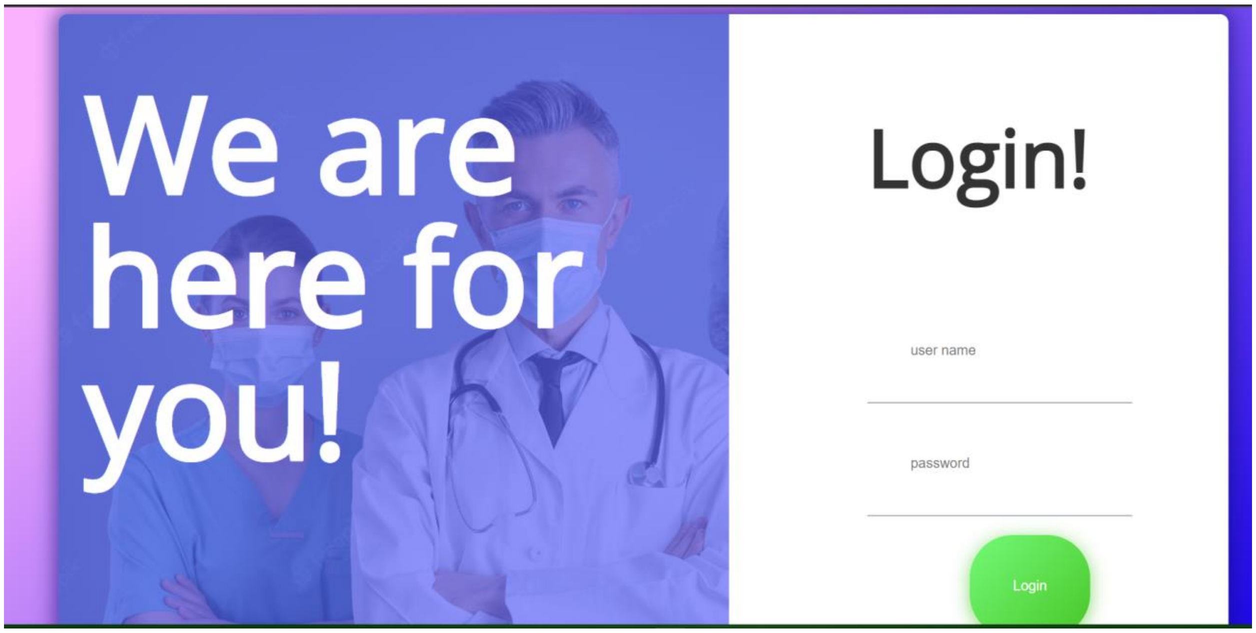


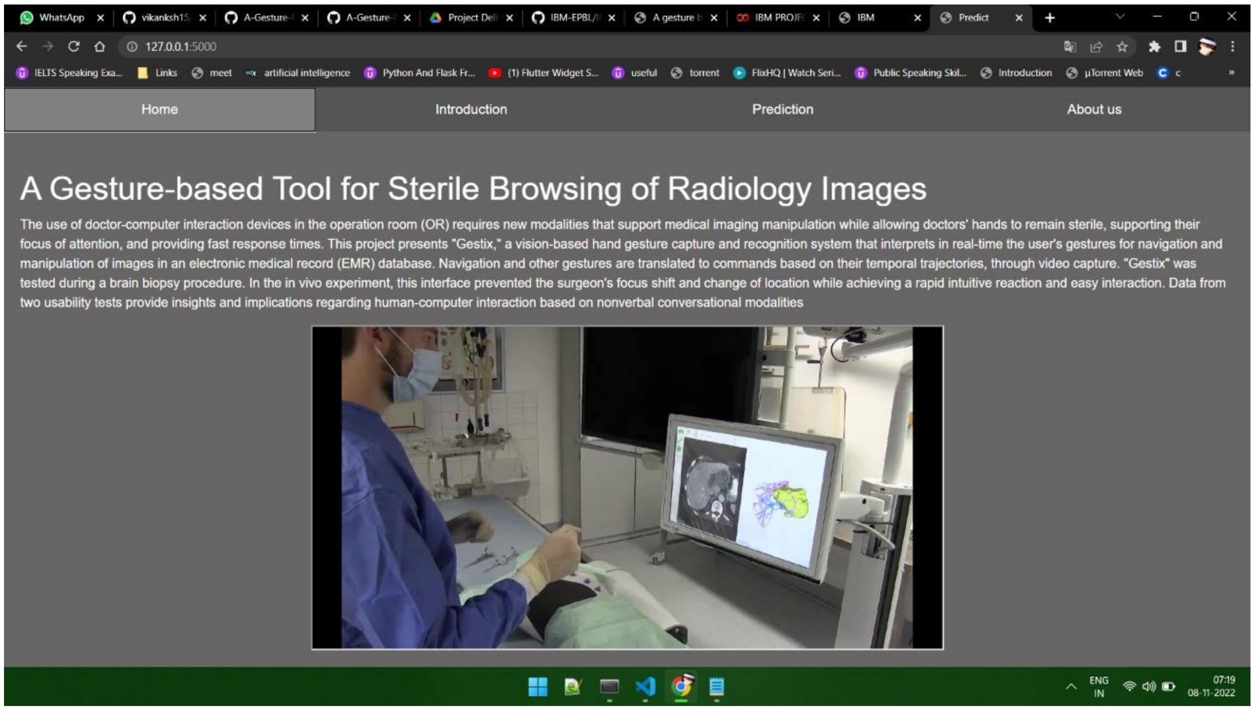
```
elif prediction[0][0]=='TWO':
    (h, w, d) = image1.shape
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, -45, 1.0)
    rotated = cv2.warpAffine(image1, M, (w, h))
    cv2.imshow("OpenCV Rotation", rotated)
    key=cv2.waitKey(3000)
    if (key & 0xFF) == ord("2"):
        cv2.destroyWindow("OpenCV Rotation")

elif prediction[0][0]=='THREE':
    blurred = cv2.GaussianBlur(image1, (21, 21), 0)
    cv2.imshow("Blurred", blurred)
    key=cv2.waitKey(3000)
    if (key & 0xFF) == ord("3"):
        cv2.destroyWindow("Blurred")

elif prediction[0][0]=='FOUR':

    resized = cv2.resize(image1, (400, 400))
    cv2.imshow("Fixed Resizing", resized)
    key=cv2.waitKey(3000)
    if (key & 0xFF) == ord("4"):
        cv2.destroyWindow("Fixed Resizing")

elif prediction[0][0]=='FIVE':

    gray = cv2.cvtColor(image1, cv2.COLOR_RGB2GRAY)
    cv2.imshow("OpenCV Gray Scale", gray)
    key=cv2.waitKey(3000)
    if (key & 0xFF) == ord("5"):
```

RUNNING THE APPLICATION

LOGIN AND HOME PAGE

Description about the project:



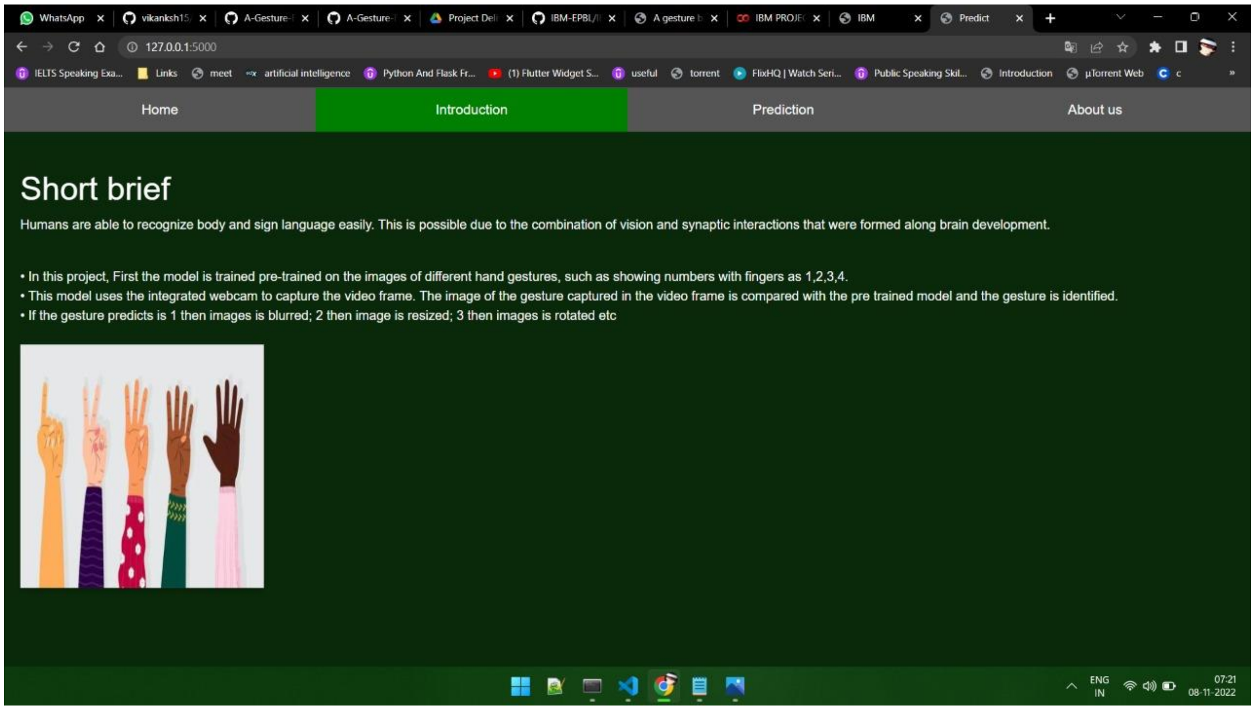


A Gesture-based Tool for Sterile Browsing of Radiology Images

The use of doctor-computer interaction devices in the operation room (OR) requires new modalities that support medical imaging manipulation while allowing doctors' hands to remain sterile, supporting their focus of attention, and providing fast response times. This project presents "Gestix," a vision-based hand gesture capture and recognition system that interprets in real-time the user's gestures for navigation and manipulation of images in an electronic medical record (EMR) database. Navigation and other gestures are translated to commands based on their temporal trajectories, through video capture. "Gestix" was tested during a brain biopsy procedure. In the in vivo experiment, this interface prevented the surgeon's focus shift and change of location while achieving a rapid intuitive reaction and easy interaction. Data from two usability tests provide insights and implications regarding human-computer interaction based on nonverbal conversational modalities

INTRODUCTION PAGE

Project Works:



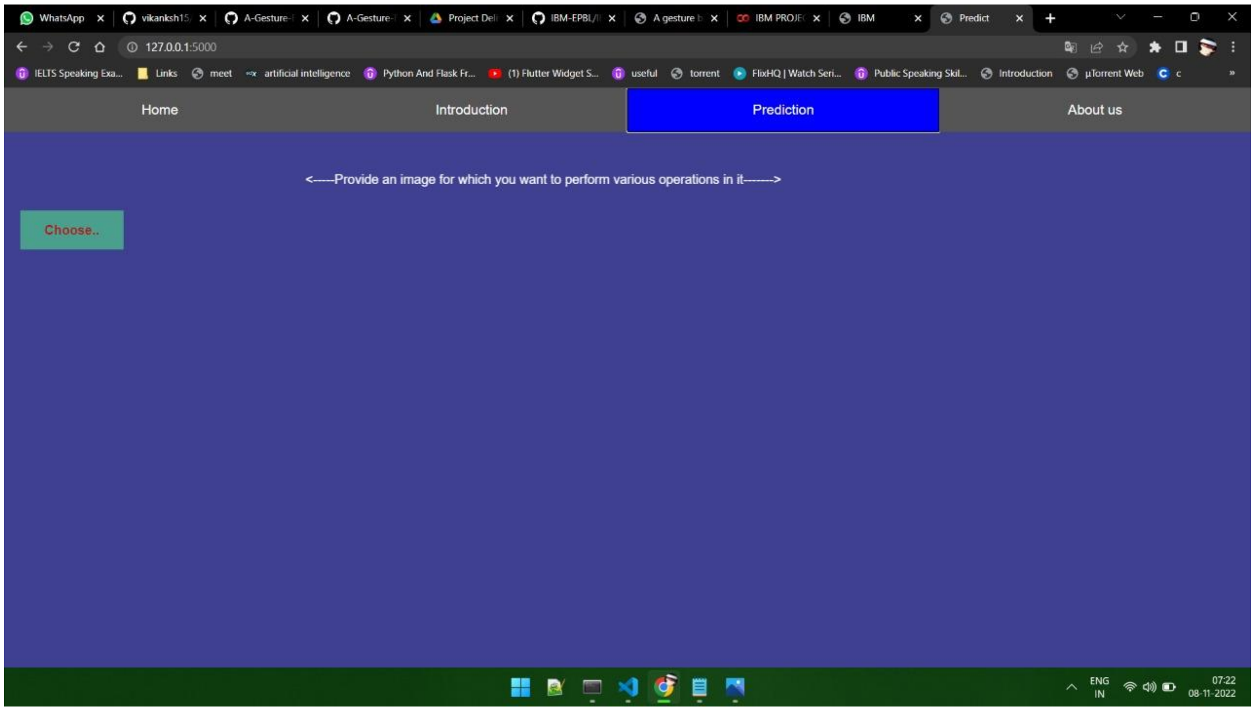
Short brief

Humans are able to recognize body and sign language easily. This is possible due to the combination of vision and synaptic interactions that were formed along brain development.

• In this project, First the model is trained pre-trained on the images of different hand gestures, such as showing numbers with fingers as 1,2,3,4.
• This model uses the integrated webcam to capture the video frame. The image of the gesture captured in the video frame is compared with the pre trained model and the gesture is identified.
• If the gesture predicts is 1 then images is blurred; 2 then image is resized; 3 then images is rotated etc

PREDICTION PAGE

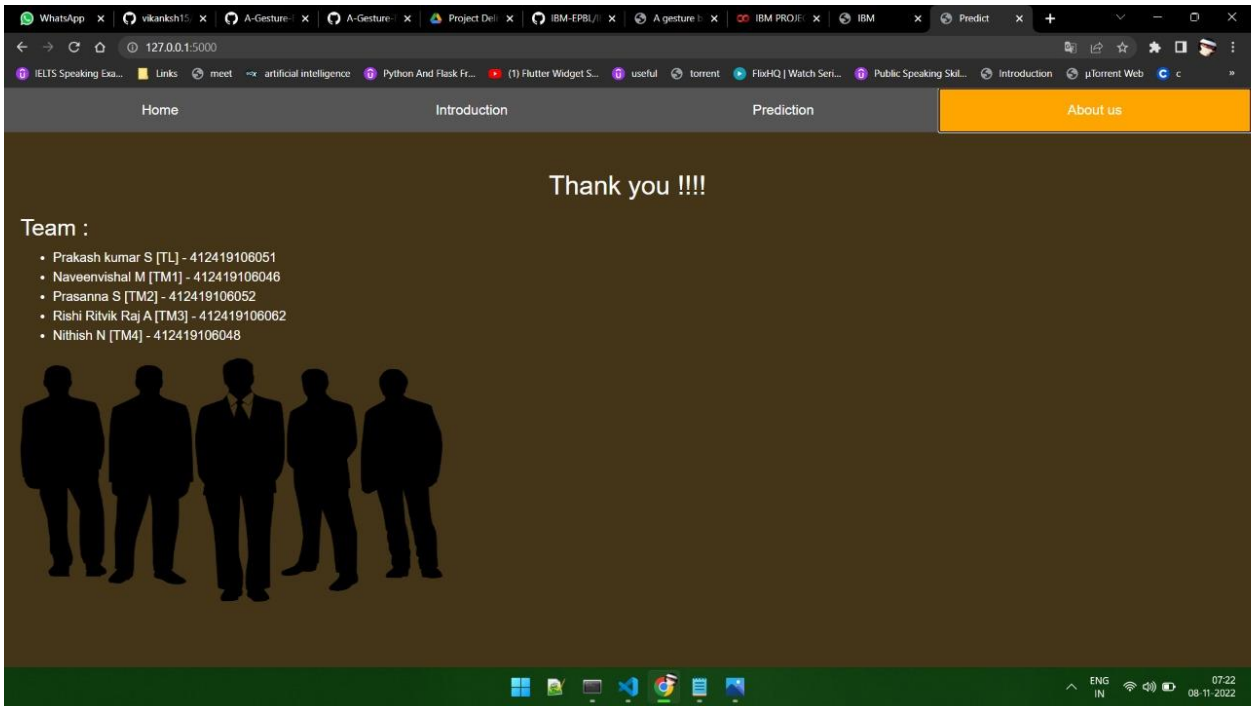
Output can be done by giving the corresponding input:



<-----Provide an image for which you want to perform various operations in it------->

ABOUT US

Team details:



Thank you !!!!

Team :

- Prakash kumar S [TL] - 412419106051
- Naveenvishal M [TM1] - 412419106046
- Prasanna S [TM2] - 412419106052
- Rishi Ritvik Raj A [TM3] - 412419106062
- Nithish N [TM4] - 412419106048