Gesture Based Tool for Sterile Browsing of Radiology Images

Team ID: PNT2022TMID47848

TEAM MEMBERS:

AJEETH KUMAR R:

MOHAMMED ISMAIL K

NAVEEN K

POOVIZHI KABILAN M

1.INTRODUCTION

1.1 Overview

In this project we use gestures to browse radiology images. Gestures refer to non-verbal form of communication.
A major challenge involved in this process is to provide doctors with efficient, intuitive, accurate and safe means of interaction without affecting the quality of their work. Keyboards and pointing devices, such as a mouse, are today's common method of human—computer interaction. However, the use of computer keyboards and mouse by doctors and nurses in intensive care units (ICUs) is a common method for spreading infections.

Humans can 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 order to replicate this skill in computers, some problems need to be solved: how to separate objects of interest in images and which image capture technology and classification technique are more appropriate, among others. In this project Gesture based Desktop automation, First the model is trained pre trained on the images of different hand gestures, such as a 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 0 - then images is converted into rectangle, 1 - image is Resized , 2 - image is rotated, 3 - image is blurred.

1.2 PURPOSE

It is used to browse through the images obtained using radiology using hand gestures rather than using mouse,keyboard,etc thereby maintaining sterility.

2.LITERATURE SURVEY

2.1 A Gesture-based Tool for Sterile Browsing of Radiology Images - research paper by national library of medicine
The hand gesture control system "*Gestix*" developed by the authors helped the doctor to remain in place during the entire operation, without any need to move to the main control wall since all the commands were performed using hand gestures.The sterile gesture interface consists of a Canon VC-C4 camera, whose pan/tilt/zoom can be initially set using an infrared (IR) remote.
This camera is placed just over a large flat screen monitor .

Additionally, an Intel Pentium IV, (600MHz, OS: Windows XP) with a Matrox Standard II video-capturing device is used.

The "*Gibson*" image browser is a 3D visualization medical tool that enables examination of images, such as: MRIs, CT scans and X-rays. The images are arranged over a multiple layer 3D cylinder. The image of interest is found through rotating the cylinder in the four cardinal directions. To interface the gesture recognition routines with the "*Gibson*" system, information such as the centroid of the hand, its size, and orientation are used to enable screen operations in the "*Gibson*" graphical user interface.

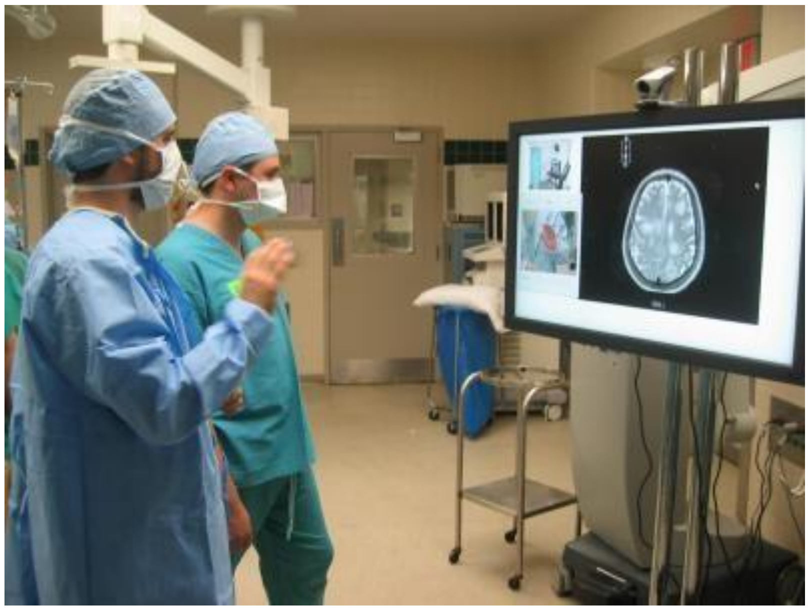


Fig 2. Radiology image browsing using hand gesture in hospital

3.THEORITICAL ANALYSIS

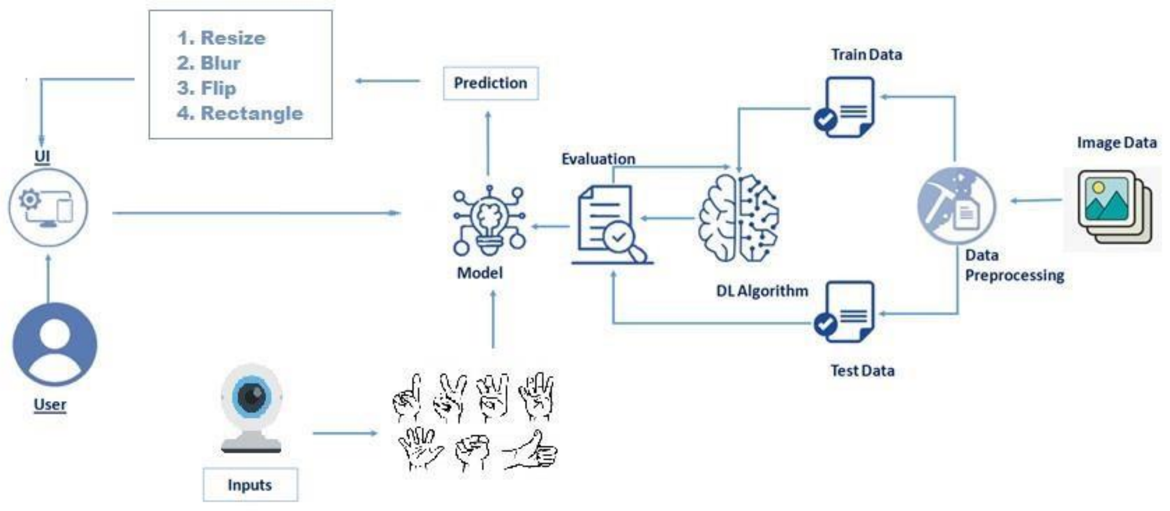


Fig 3. Architecture of Gesture Based Tool for Sterile Browsing of Radiology Images

4.EXPERIMENTAL INVESTIGATIONS

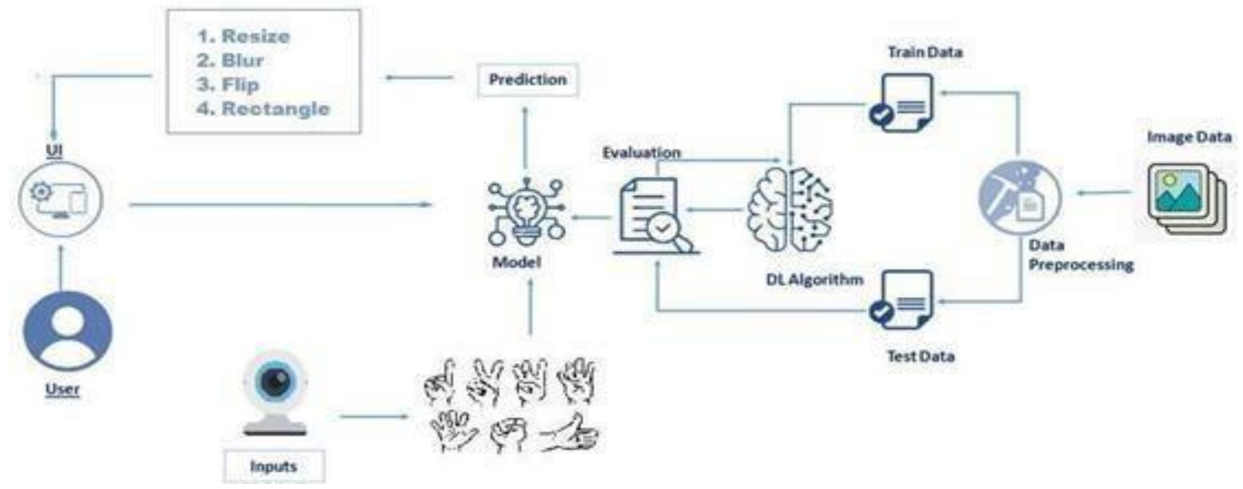
We found that many hospitals rely on mouse and keyboard to browse the images that are obtained during different surgeries, scans, etc. This can contaminate the environment with various infections thus compromising the sterility.

Various technologies have been developed to overcome this issue and one such technology was called ‘Gestix’.

This hand gesture system for MRI manipulation in an EMR image database called “*Gestix*” was tested during a brain biopsy surgery. This system is a real-time hand-tracking recognition technique based on color and motion fusion. In an in vivo experiment, this type of interface prevented the surgeon's focus shift and change of location while achieving rapid intuitive interaction with an EMR image database. In addition to allowing sterile interaction with EMRs, the “*Gestix*” hand gesture interface provides:

1. ease of use—the system allows the surgeon to use his/her hands, their natural work tool;
2. rapid reaction—nonverbal instructions by hand gesture commands are intuitive and fast
3. an unencumbered interface—the proposed system does not require the surgeon to attach a microphone, use head-mounted (body-contact) sensing devices or to use foot pedals
4. distance control—the hand gestures can be performed up to 5 meters from the camera and still be recognized accurately.

Technical Architecture:



Overview:

1. Defining our classification categories

2. Collect training images

3. Train the model

4. Test our model

Project Flow:

- User interacts with the UI (User Interface) to upload the image as input
- Depending on the different gesture inputs different opera ons are applied to the input image.
- Once model analyses the gesture, the predic on with opera on applied on image is showcased on the UI.

To accomplish this, we have to complete all the ac vi es and tasks listed below:

- Data Collection.
  - Collect the dataset or Create the dataset
  - Data Pre-processing
    - Import the ImageDataGenerator library
    - Configure ImageDataGenerator class
    - Apply ImageDataGenerator func onality to Train set and Test set
- Model Building

- Import the model building Libraries
- Ini alizing the model
- Adding Input Layer
- Adding Hidden Layer ○ Adding Output Layer
- Configure the Learning Process
- Training and tes ng the model
- Save the Model

- Application Building
    - Create an HTML file
    - Build Python Code

Following so ware, concepts and packages are used in this project

- Anaconda navigator
- Python packages:
    - Open anaconda prompt as administrator
    - Type “pip install TensorFlow” (make sure you are working on python 64 bit)
    - Type “pip install opencast-python”
    - Type “pip install flask”

Deep Learning Concepts:

CNN: a convolu on neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery.

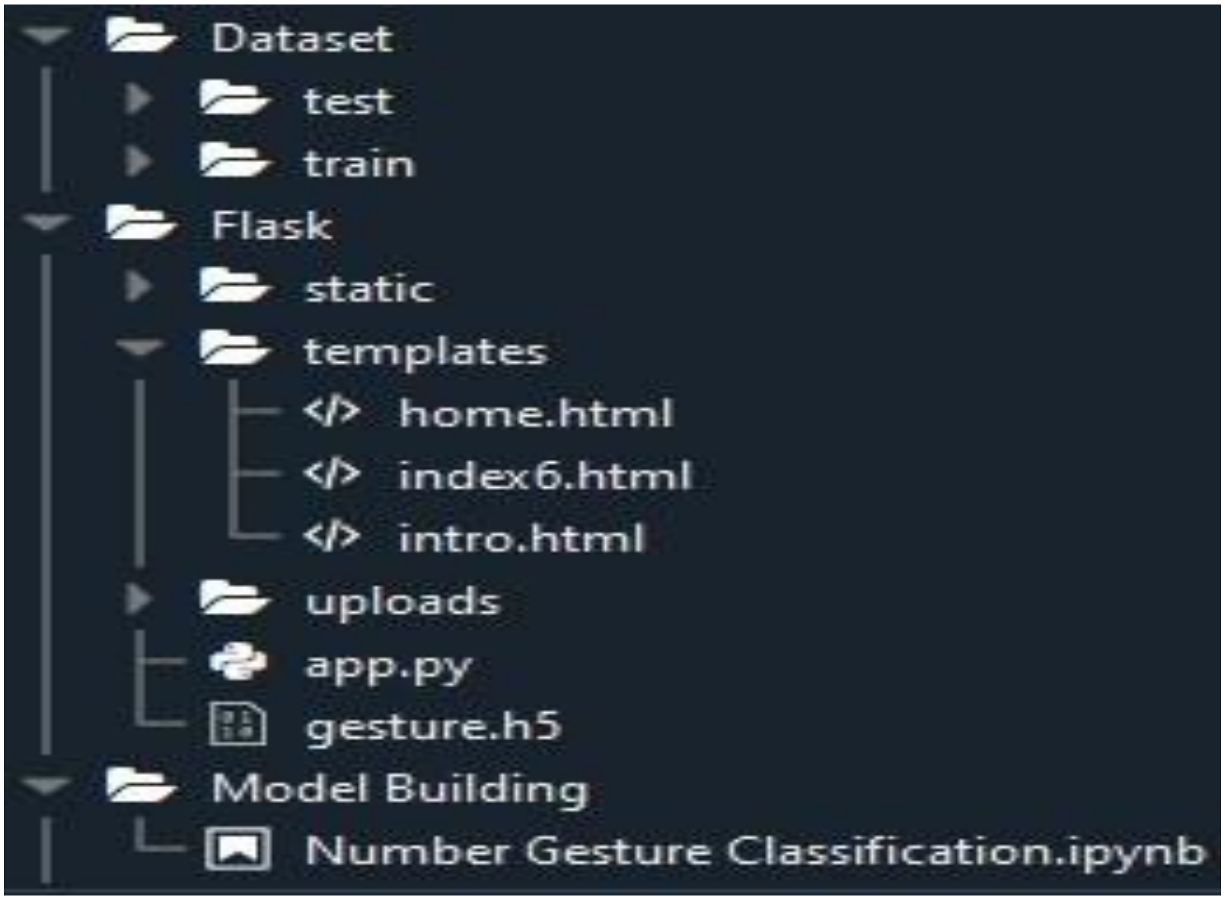
OpenCV: It is an Open Source Computer Vision Library which are mainly used for image processing, video capture and analysis including features like face detec on and object detec on.

Flask: Flask is a popular Python web framework, meaning it is a third-party

Python library used for developing web applica ons.

Project Structure:

- Dataset folder contains the training and tes ng images for training our model.
- We are building a Flask Applica on which needs HTML pages stored in the templates folder and a python script app.py for server side scrip ng ● we need the model which is saved and the saved model in this content is gesture.h5
- The static folder will contain js and css files
- Whenever we upload a image to predict, that images is saved in uploads folder



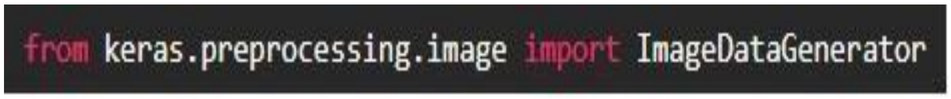
```
Dataset
    test
    train
Flask
    static
    templates
        home.html
        index6.html
        intro.html
    uploads
    app.py
    gesture.h5
Model Building
    Number Gesture Classification.ipynb
```

Data Collection:

ML depends heavily on data, without data, it is impossible for a machine to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various ac ons.

Image Pre-processing:

In this step we improve the image data that suppresses unwilling distor ons or enhances some image features important for further processing, although perform some geometric transforma ons of images like rota on, scaling, transla on etc.



```python
from keras.preprocessing.image import ImageDataGenerator
```

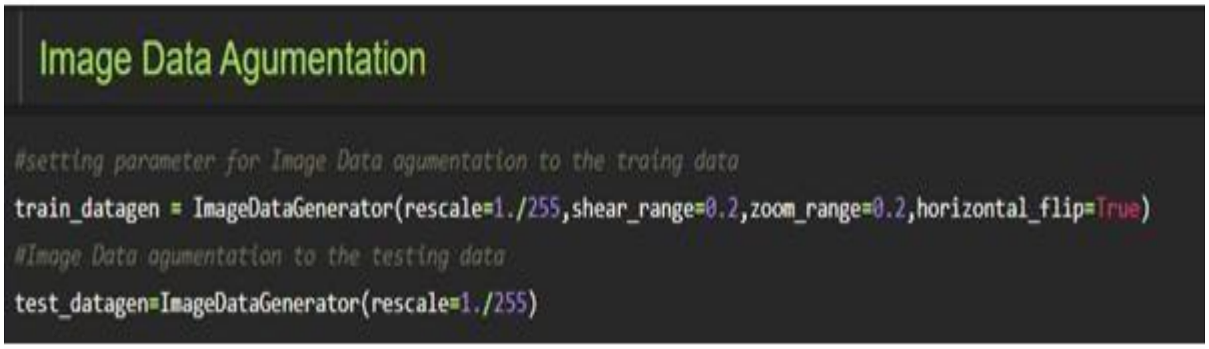
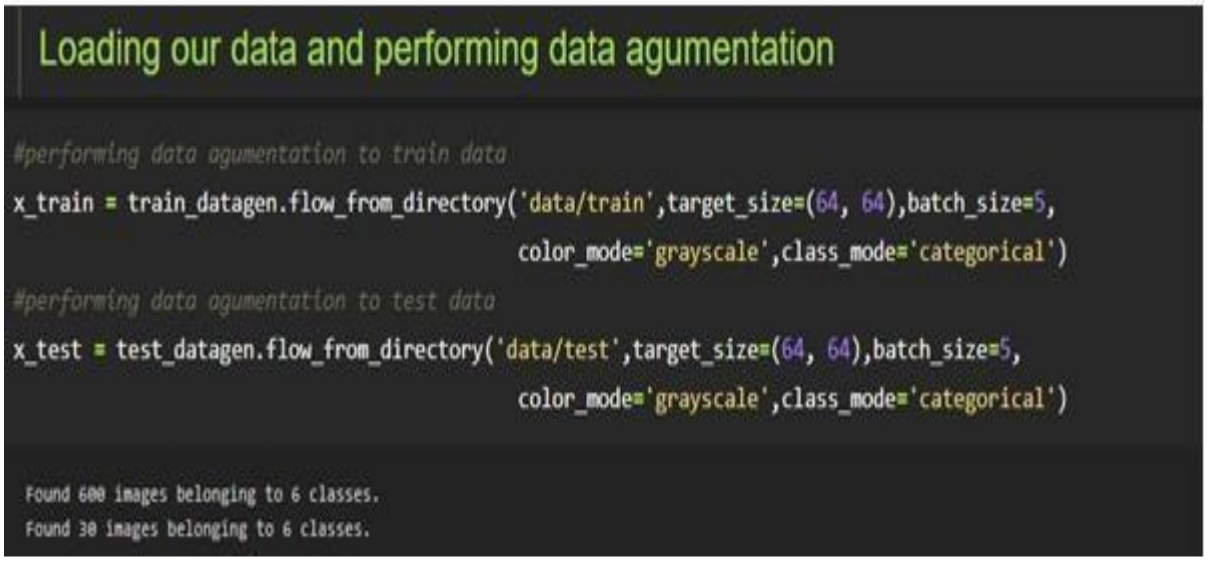


Image Data Agumentation

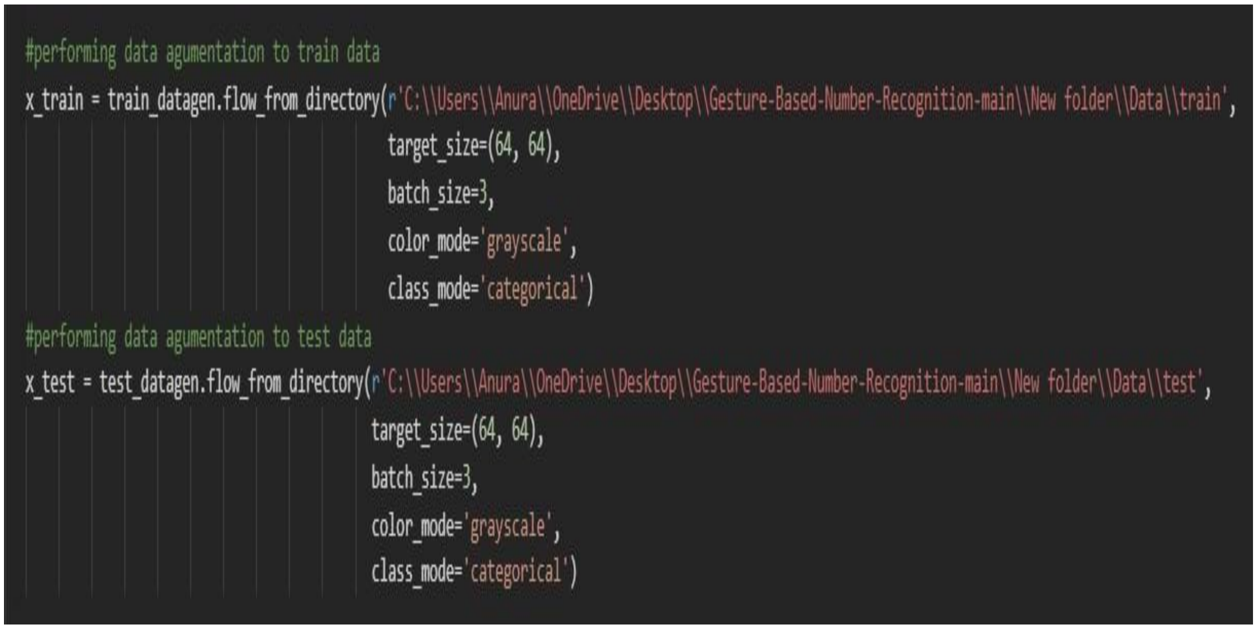
```python
#setting parameter for Image Data agumentation to the traing data
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
#Image Data agumentation to the testing data
test_datagen=ImageDataGenerator(rescale=1./255)
```



Loading our data and performing data agumentation

```python
#performing data agumentation to train data
x_train = train_datagen.flow_from_directory('data/train',target_size=(64, 64),batch_size=5,
                                            color_mode='grayscale',class_mode='categorical')
#performing data agumentation to test data
x_test = test_datagen.flow_from_directory('data/test',target_size=(64, 64),batch_size=5,
                                          color_mode='grayscale',class_mode='categorical')
```

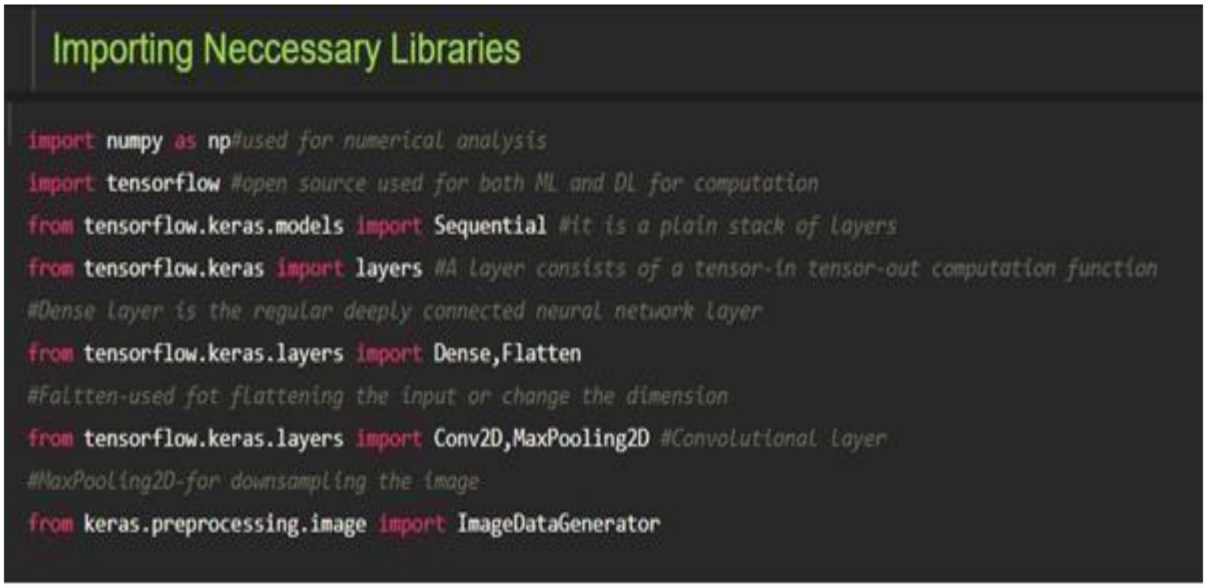
```
Found 600 images belonging to 6 classes.
Found 30 images belonging to 6 classes.
```



```
#performing data agumentation to train data
x_train = train_datagen.flow_from_directory(r'C:\\Users\\Anura\\OneDrive\\Desktop\\Gesture-Based-Number-Recognition-main\\New folder\\Data\\train',
                                            target_size=(64, 64),
                                            batch_size=3,
                                            color_mode='grayscale',
                                            class_mode='categorical')
#performing data agumentation to test data
x_test = test_datagen.flow_from_directory(r'C:\\Users\\Anura\\OneDrive\\Desktop\\Gesture-Based-Number-Recognition-main\\New folder\\Data\\test',
                                          target_size=(64, 64),
                                          batch_size=3,
                                          color_mode='grayscale',
                                          class_mode='categorical')
```

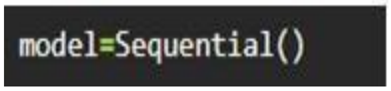
Model Building:

In this step we build Convolu on Neural Networking which contains a input layer along with the convolu on, max-pooling and finally a output layer.



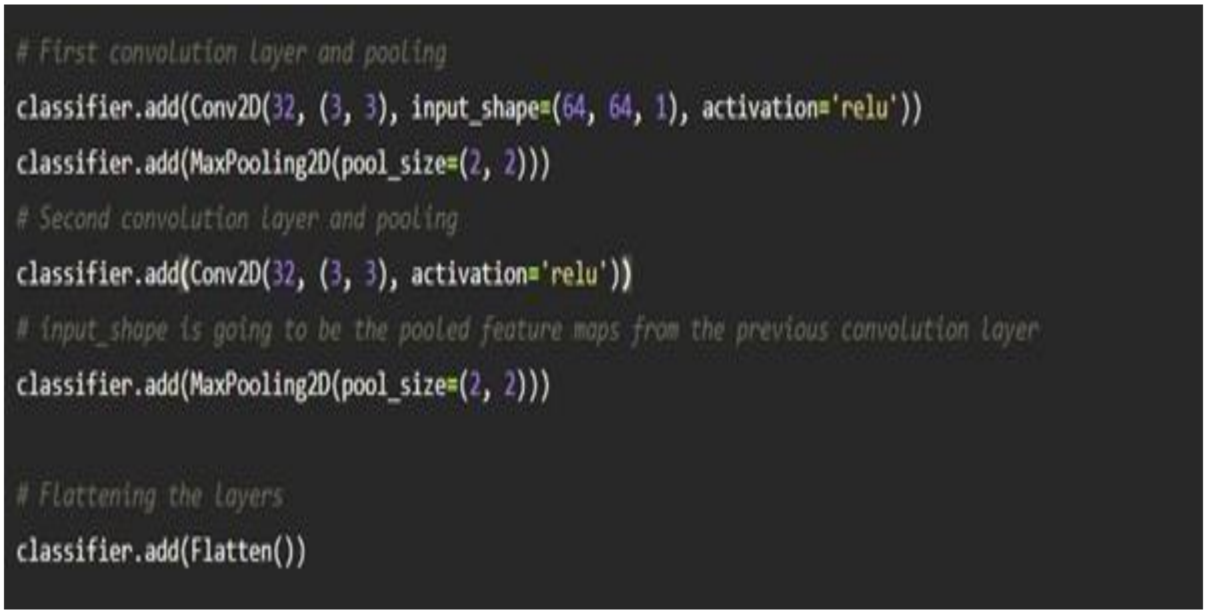
Importing Neccessary Libraries

```
import numpy as np#used for numerical analysis
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.models import Sequential #it is a plain stack of layers
from tensorflow.keras import layers #A layer consists of a tensor-in tensor-out computation function
#Dense layer is the regular deeply connected neural network layer
from tensorflow.keras.layers import Dense,Flatten
#Faltten-used fot flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D,MaxPooling2D #Convolutional layer
#MaxPooling2D-for downsampling the image
from keras.preprocessing.image import ImageDataGenerator
```



```
model=Sequential()
```

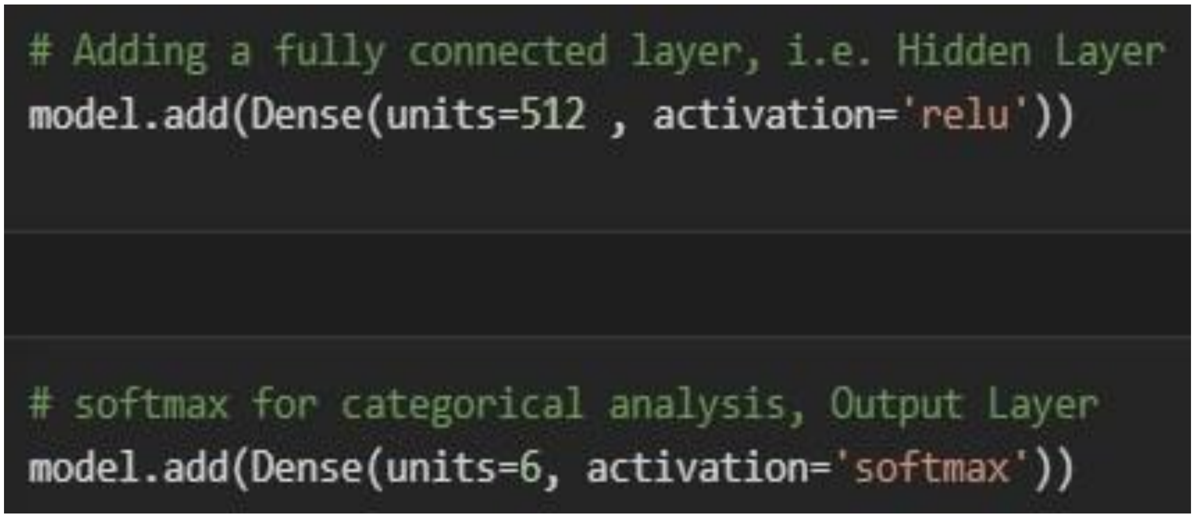
Adding CNN Layers:



```
# First convolution layer and pooling
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 1), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Second convolution layer and pooling
classifier.add(Conv2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening the layers
classifier.add(Flatten())
```

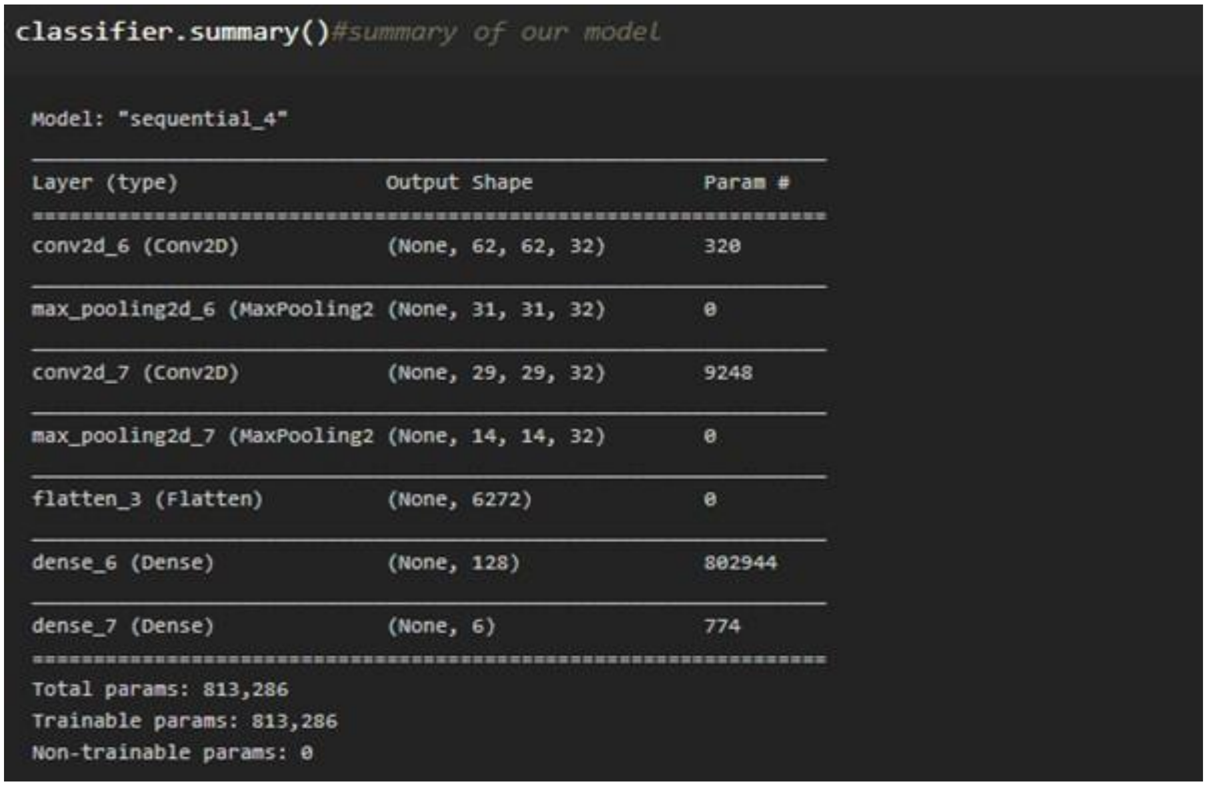
Adding Dense Layers:

Dense layer is deeply connected to neural network layer. It is most common and frequently used layer.



```
# Adding a fully connected layer, i.e. Hidden Layer
model.add(Dense(units=512 , activation='relu'))


# softmax for categorical analysis, Output Layer
model.add(Dense(units=6, activation='softmax'))
```

Understanding the model is very important phase to properly use it for training and predic on purposes. Kera-s provides a simple method, summary to get the full informa on about the model and its layers.

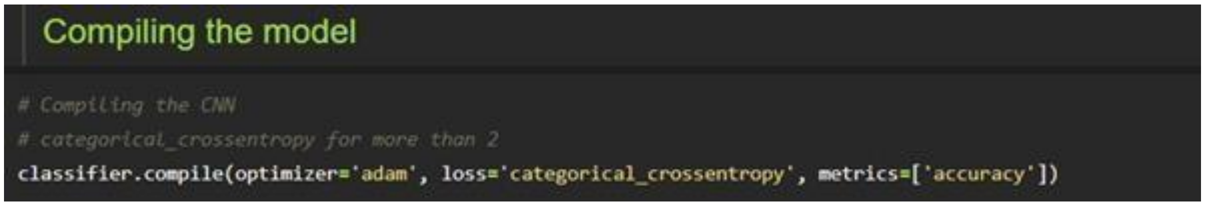


```
classifier.summary()#summary of our model

Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 62, 62, 32)        320
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               802944
_________________________________________________________________
dense_7 (Dense)              (None, 6)                 774
=================================================================
Total params: 813,286
Trainable params: 813,286
Non-trainable params: 0
```

Configure The Learning Process:

- The compila on is the final step in crea ng a model. Once the compila on is done, we can move on to training phase. Loss func on is used to find error or devia on in the learning process. Kera-s requires loss func on during model compila on process.
- Op miza on is an important process which op mize the input weights by comparing the predic on and the loss func on. Here we are using Adam op mizer

Metrics is used to evaluate the performance of your model. It is similar to loss func on, but not used in training process



Compiling the model

```
# Compiling the CNN
# categorical_crossentropy for more than 2
classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

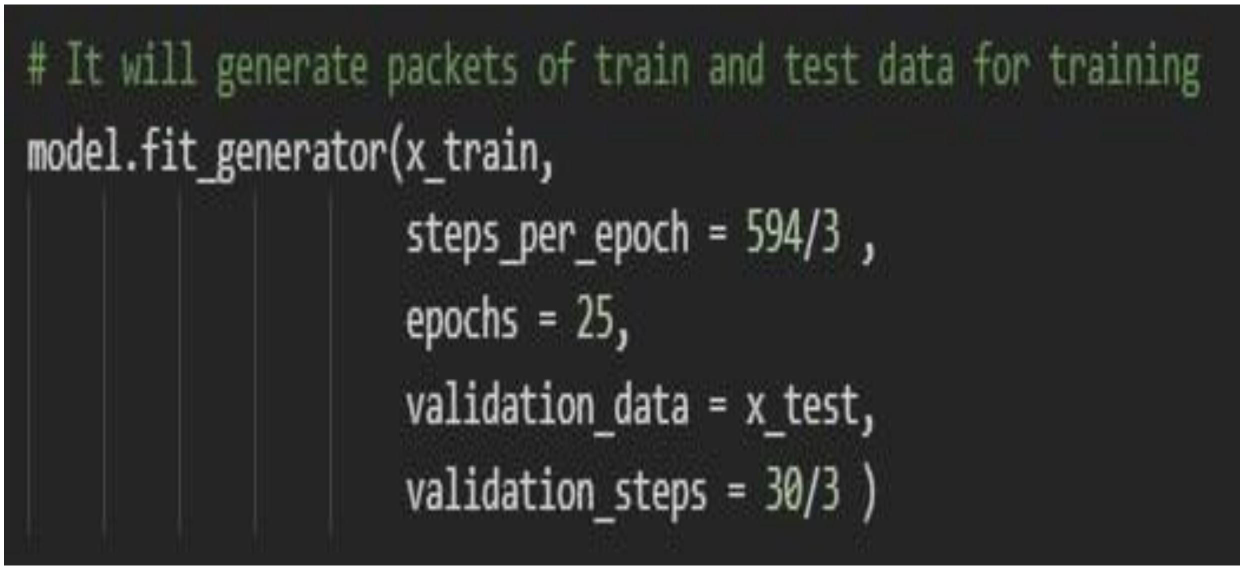
Train The Model:

Train the model with our image dataset. **fit_generator** func ons used to train a deep learning neural network

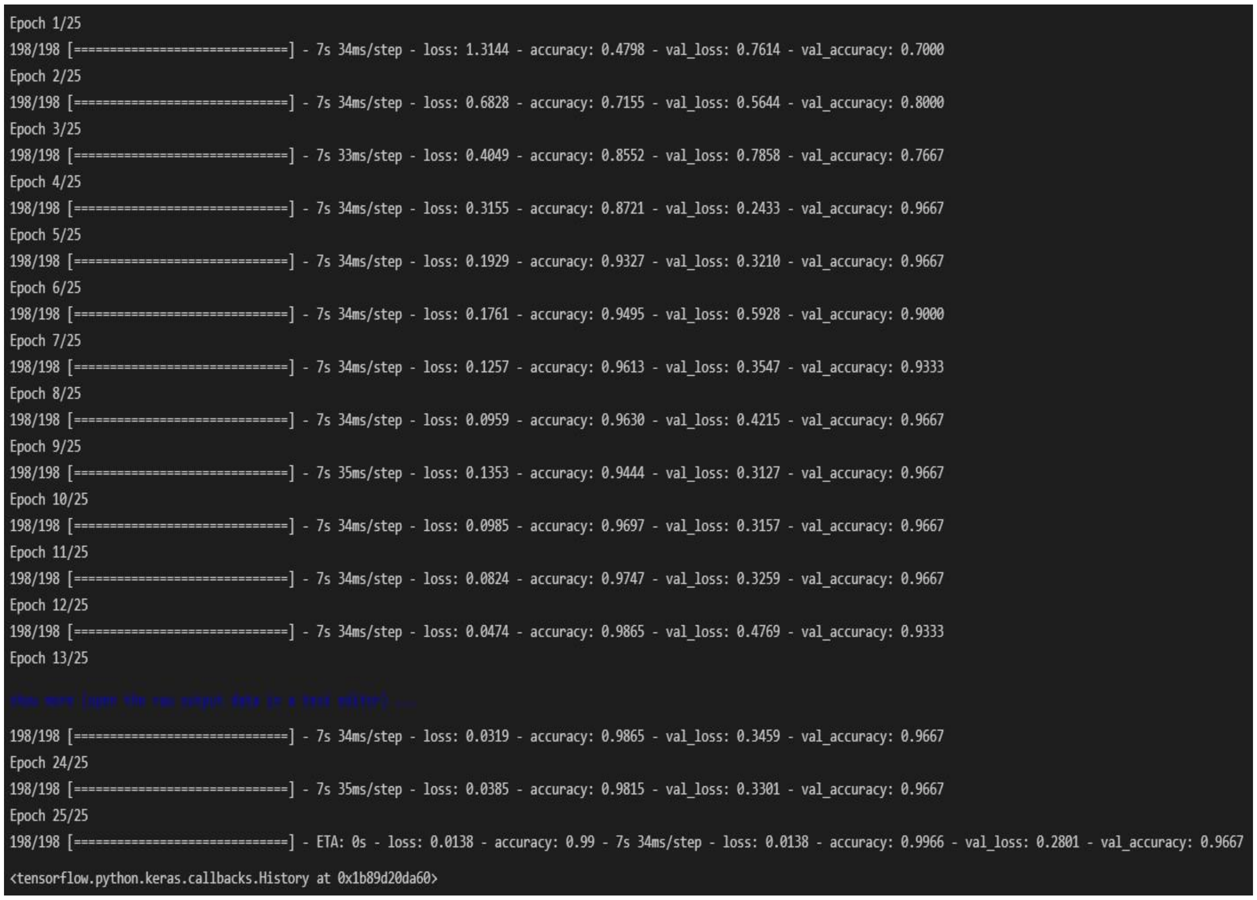
Arguments:

- 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. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.
- Epochs : an integer and number of epochs we want to train our model.
- valida on_data can be either:

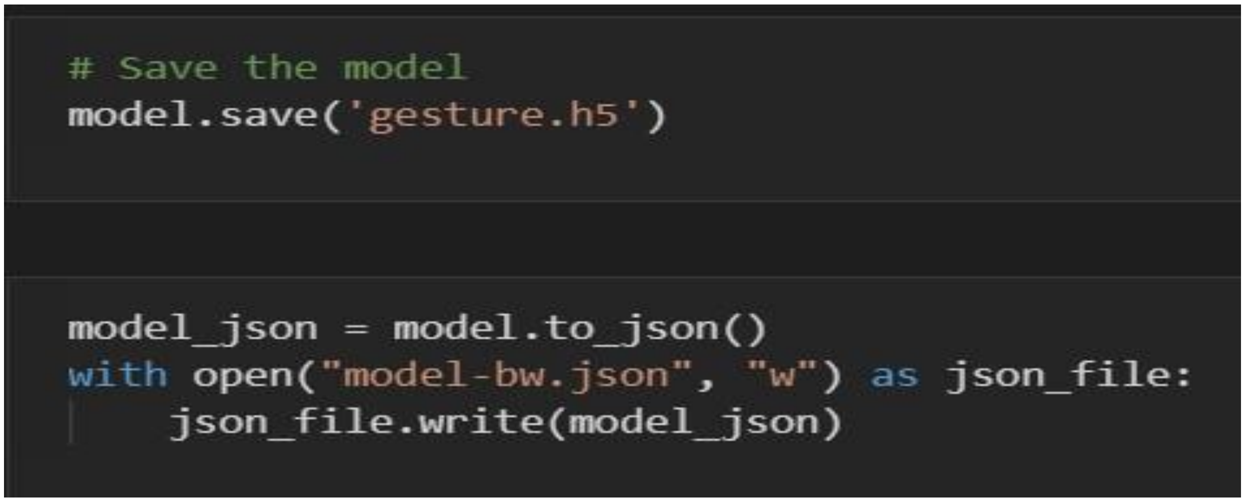
1. an inputs and targets list
2. a generator
3. an inputs, targets, and sample_weights list which can be used. ● valida on_steps :only if the valida on_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of valida on data points in your dataset divided by the valida on batch size.



```
# It will generate packets of train and test data for training
model.fit_generator(x_train,
                    steps_per_epoch = 594/3 ,
                    epochs = 25,
                    validation_data = x_test,
                    validation_steps = 30/3 )
```



```
Epoch 1/25
198/198 [==============================] - 7s 34ms/step - loss: 1.3144 - accuracy: 0.4798 - val_loss: 0.7614 - val_accuracy: 0.7000
Epoch 2/25
198/198 [==============================] - 7s 34ms/step - loss: 0.6828 - accuracy: 0.7155 - val_loss: 0.5644 - val_accuracy: 0.8000
Epoch 3/25
198/198 [==============================] - 7s 33ms/step - loss: 0.4049 - accuracy: 0.8552 - val_loss: 0.7858 - val_accuracy: 0.7667
Epoch 4/25
198/198 [==============================] - 7s 34ms/step - loss: 0.3155 - accuracy: 0.8721 - val_loss: 0.2433 - val_accuracy: 0.9667
Epoch 5/25
198/198 [==============================] - 7s 34ms/step - loss: 0.1929 - accuracy: 0.9327 - val_loss: 0.3210 - val_accuracy: 0.9667
Epoch 6/25
198/198 [==============================] - 7s 34ms/step - loss: 0.1761 - accuracy: 0.9495 - val_loss: 0.5928 - val_accuracy: 0.9000
Epoch 7/25
198/198 [==============================] - 7s 34ms/step - loss: 0.1257 - accuracy: 0.9613 - val_loss: 0.3547 - val_accuracy: 0.9333
Epoch 8/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0959 - accuracy: 0.9630 - val_loss: 0.4215 - val_accuracy: 0.9667
Epoch 9/25
198/198 [==============================] - 7s 35ms/step - loss: 0.1353 - accuracy: 0.9444 - val_loss: 0.3127 - val_accuracy: 0.9667
Epoch 10/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0985 - accuracy: 0.9697 - val_loss: 0.3157 - val_accuracy: 0.9667
Epoch 11/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0824 - accuracy: 0.9747 - val_loss: 0.3259 - val_accuracy: 0.9667
Epoch 12/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0474 - accuracy: 0.9865 - val_loss: 0.4769 - val_accuracy: 0.9333
Epoch 13/25

Show more (open the raw output data in a text editor) ...

198/198 [==============================] - 7s 34ms/step - loss: 0.0319 - accuracy: 0.9865 - val_loss: 0.3459 - val_accuracy: 0.9667
Epoch 24/25
198/198 [==============================] - 7s 35ms/step - loss: 0.0385 - accuracy: 0.9815 - val_loss: 0.3301 - val_accuracy: 0.9667
Epoch 25/25
198/198 [==============================] - ETA: 0s - loss: 0.0138 - accuracy: 0.99 - 7s 34ms/step - loss: 0.0138 - accuracy: 0.9966 - val_loss: 0.2801 - val_accuracy: 0.9667

<tensorflow.python.keras.callbacks.History at 0x1b89d20da60>
```



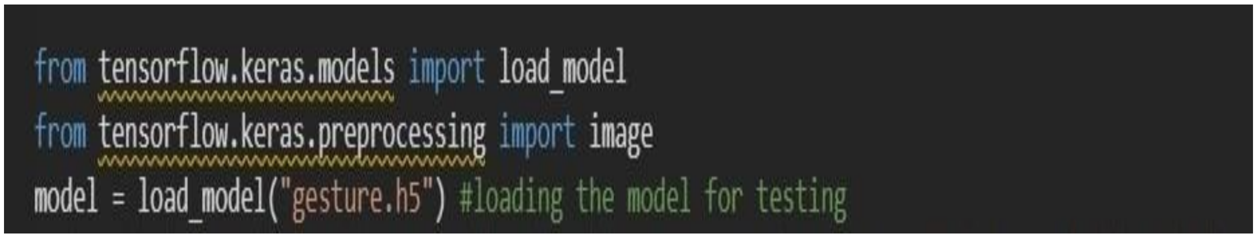
```python
# Save the model
model.save('gesture.h5')
```

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

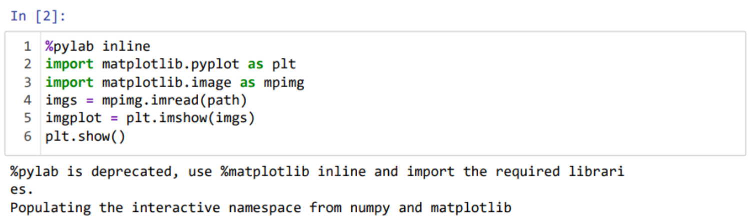
Test The Model:

Evalua on is a process during development of the model to check whether the model is best fit for the given problem and corresponding data.

Load the saved model using load_model

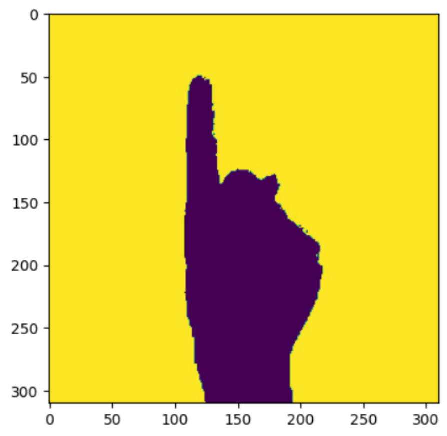


```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model = load_model("gesture.h5") #loading the model for testing
```



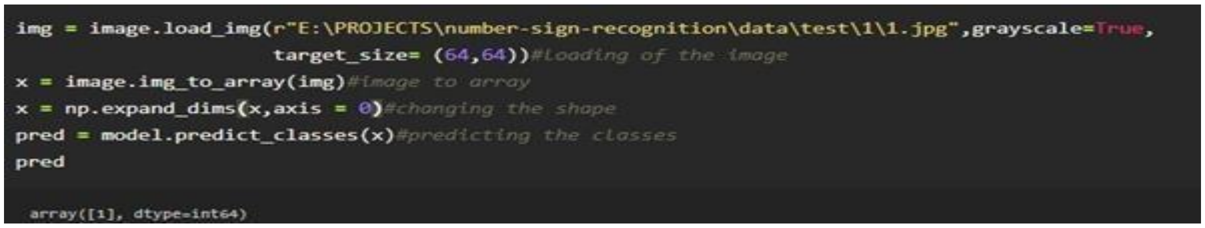
```
In [2]:
%pylab inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
imgs = mpimg.imread(path)
imgplot = plt.imshow(imgs)
plt.show()
```

```
%pylab is deprecated, use %matplotlib inline and import the required librari
es.
Populating the interactive namespace from numpy and matplotlib
```



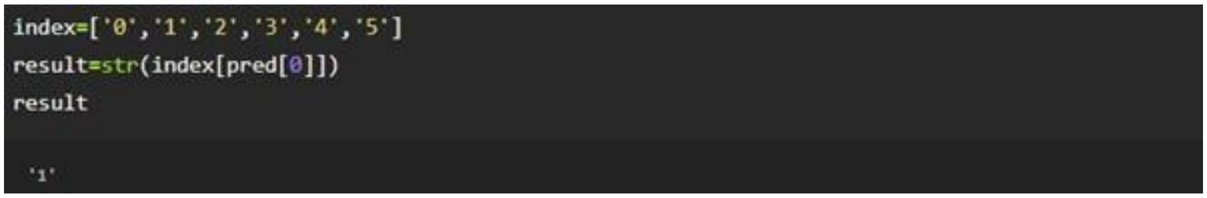
Plotting images:

Taking an image as input and checking the results



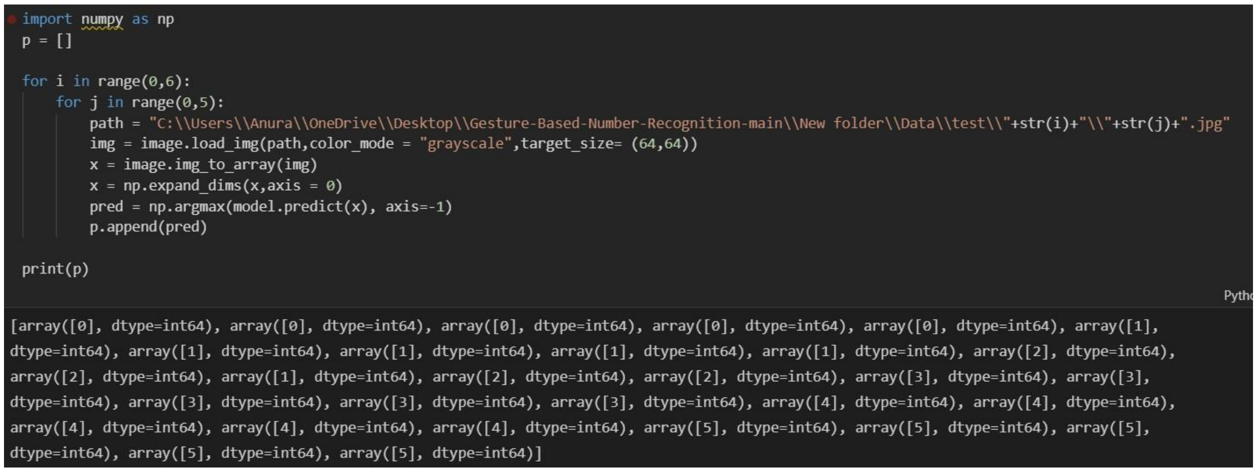
```
img = image.load_img(r"E:\PROJECTS\number-sign-recognition\data\test\1\1.jpg",grayscale=True,
                     target_size= (64,64))#Loading of the image
x = image.img_to_array(img)#image to array
x = np.expand_dims(x,axis = 0)#changing the shape
pred = model.predict_classes(x)#predicting the classes
pred

 array([1], dtype=int64)
```

By using the model we are predic ng the output for the given input image



```
index=['0','1','2','3','4','5']
result=str(index[pred[0]])
result

 '1'
```

The predicted class index name will be printed here.



```
import numpy as np
p = []

for i in range(0,6):
    for j in range(0,5):
        path = "C:\\Users\\Anura\\OneDrive\\Desktop\\Gesture-Based-Number-Recognition-main\\New folder\\Data\\test\\"+str(i)+"\\"+str(j)+".jpg"
        img = image.load_img(path,color_mode = "grayscale",target_size= (64,64))
        x = image.img_to_array(img)
        x = np.expand_dims(x,axis = 0)
        pred = np.argmax(model.predict(x), axis=-1)
        p.append(pred)

print(p)
```

```
[array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([1],
dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([2], dtype=int64),
array([2], dtype=int64), array([1], dtype=int64), array([2], dtype=int64), array([2], dtype=int64), array([3], dtype=int64), array([3],
dtype=int64), array([3], dtype=int64), array([3], dtype=int64), array([3], dtype=int64), array([4], dtype=int64), array([4], dtype=int64),
array([4], dtype=int64), array([4], dtype=int64), array([4], dtype=int64), array([5], dtype=int64), array([5], dtype=int64), array([5],
dtype=int64), array([5], dtype=int64), array([5], dtype=int64)]
```

Application Building:

A er the model is trained in this par cular step, we will be building our flask-applica on which will be running in our local browser with a user interface.

Create HTML Pages:

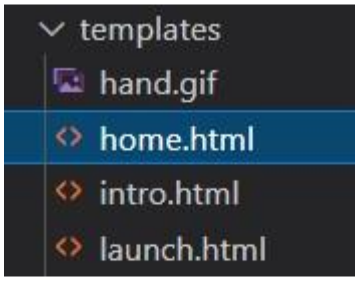
● We use HTML to create the front end part of the web page.

● Here, we created 3 html pages- home.html, intro.html and launch.html

● home.html displays home page.

● Intro.html displays introduc on about the hand gesture recogni on

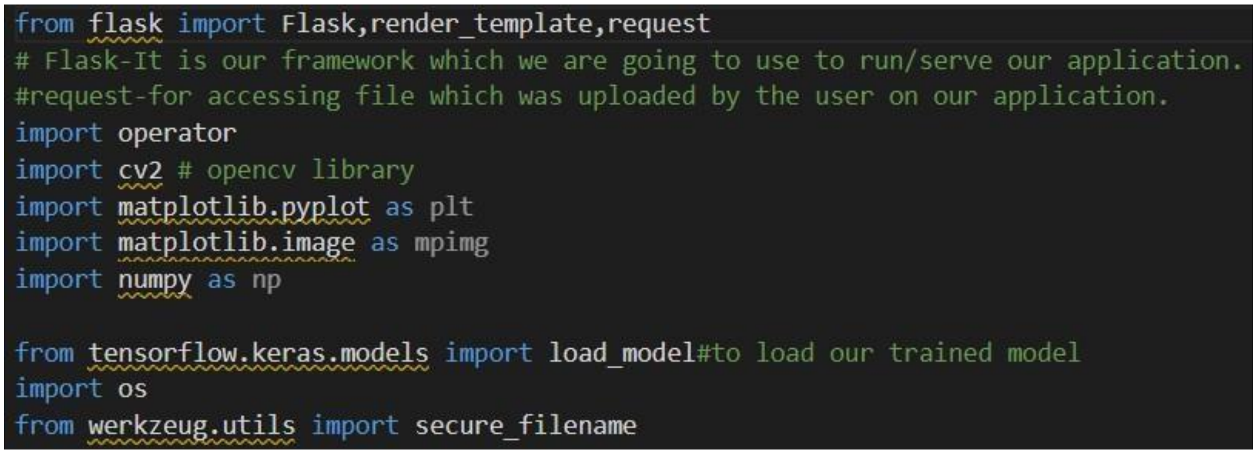
● launch.html accepts input from the user and predicts the values. ● We also use JavaScript-main.js and CSS-main.css to enhance our func onality and view of HTML pages.



Build Python Code:

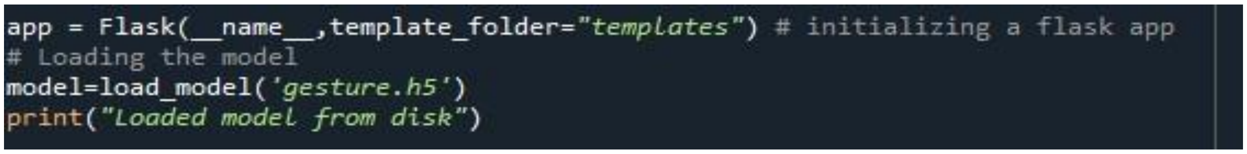
● Build flask file ‘app.py’ which is a web framework wri en in python for server-side scrip ng.

- App starts running when “ name ” constructor is called in main.
- render_template is used to return html file.
- “GET” method is used to take input from the user.
- “POST” method is used to display the output to the user.
- Impor ng Libraries



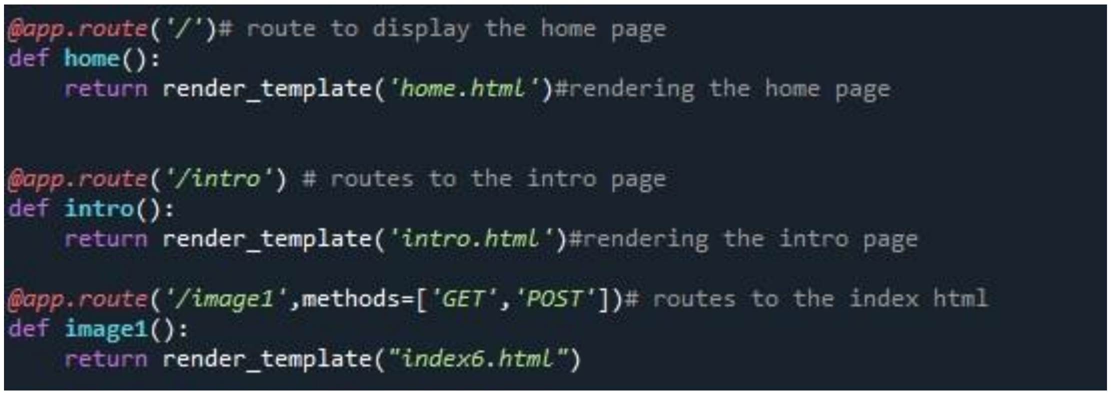
```python
from flask import Flask,render_template,request
# Flask-It is our framework which we are going to use to run/serve our application.
#request-for accessing file which was uploaded by the user on our application.
import operator
import cv2 # opencv library
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np

from tensorflow.keras.models import load_model#to load our trained model
import os
from werkzeug.utils import secure_filename
```

- Crea ng our flask applica on and loading our model



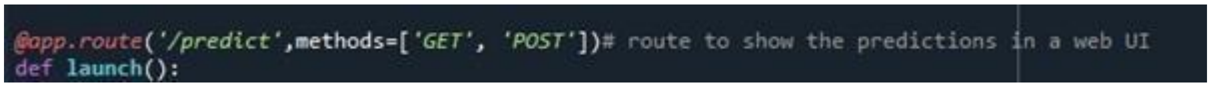
```python
app = Flask(__name__,template_folder="templates") # initializing a flask app
# Loading the model
model=load_model('gesture.h5')
print("Loaded model from disk")
```

Routing to the html Page



```python
@app.route('/')# route to display the home page
def home():
    return render_template('home.html')#rendering the home page


@app.route('/intro') # routes to the intro page
def intro():
    return render_template('intro.html')#rendering the intro page

@app.route('/image1',methods=['GET','POST'])# routes to the index html
def image1():
    return render_template("index6.html")
```

The above three route are used to render the home, introduc on and the index html pages.



```
@app.route('/predict',methods=['GET', 'POST'])# route to show the predictions in a web UI
def launch():
```

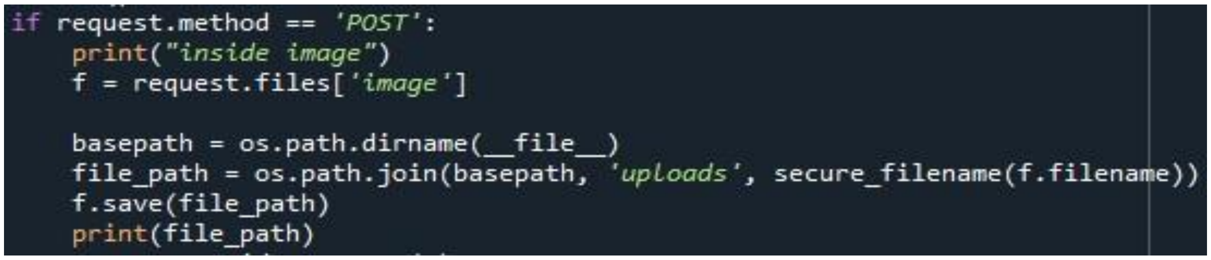
And the predict route is used for predic on and it contains all the codes which are used for predic ng our results.

Firstly, inside launch func on we are having the following things:

- Getting our input and storing it
- Grab the frames from the web cam.
- Creating ROI
- Predi ting our results
- Showcase the results with the help of opencast
- Finally run the application

Getting our input and storing it:

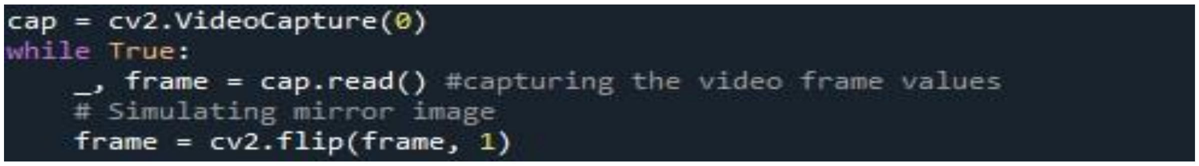
Once the predict route is called, we will check whether the method is POST or not if is POST then we will request the image files and with the help of is func on we will be storing the image in the uploads folder in our local system.



```
if request.method == 'POST':
    print("inside image")
    f = request.files['image']

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

Grab the frames from the web cam:

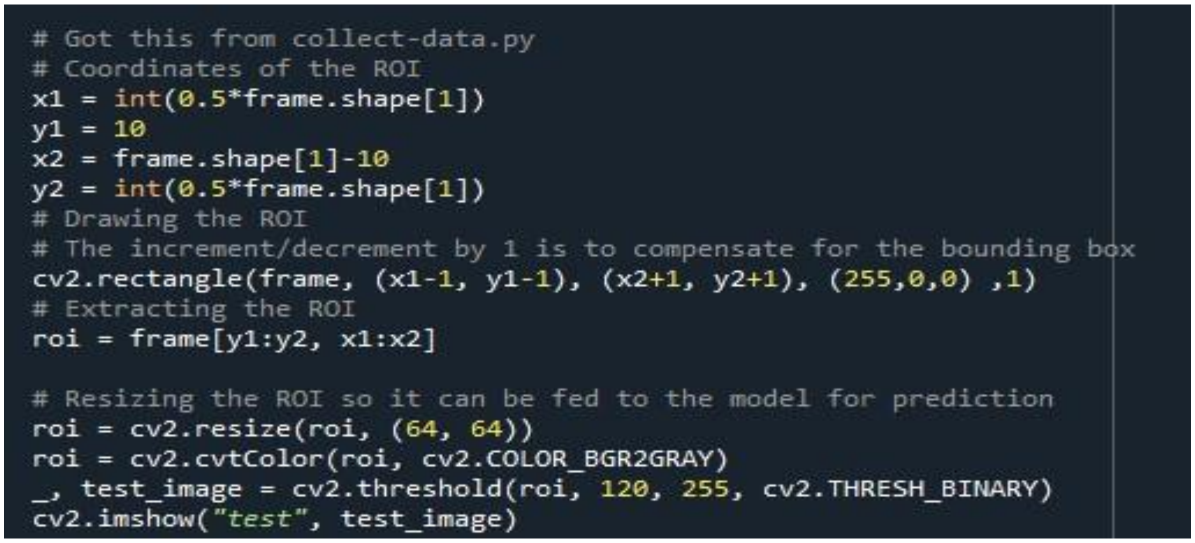
when we run the code a web cam will be opening to take the gesture input so we will be capturing the frames of the gesture for predic ng our results.



```
cap = cv2.VideoCapture(0)
while True:
    _, frame = cap.read() #capturing the video frame values
    # Simulating mirror image
    frame = cv2.flip(frame, 1)
```

Creating ROI:

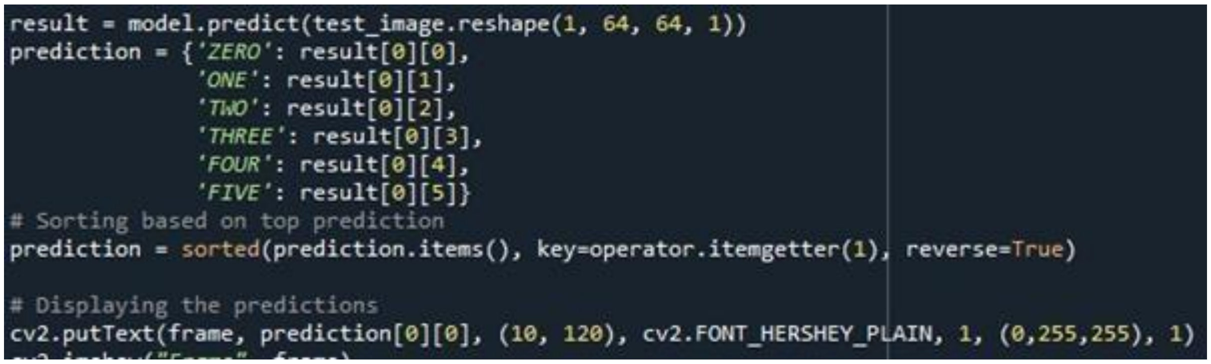
A region of interest (ROI) is a por on of an image that you want to filter or operate on in some way. The toolbox supports a set of ROI objects that you can use to create ROIs of many shapes, such circles, ellipses, polygons, rectangles, and hand-drawn shapes. A common use of an ROI is to create a binary mask image.



```python
# Got this from collect-data.py
# Coordinates of the ROI
x1 = int(0.5*frame.shape[1])
y1 = 10
x2 = frame.shape[1]-10
y2 = int(0.5*frame.shape[1])
# Drawing the ROI
# The increment/decrement by 1 is to compensate for the bounding box
cv2.rectangle(frame, (x1-1, y1-1), (x2+1, y2+1), (255,0,0) ,1)
# Extracting the ROI
roi = frame[y1:y2, x1:x2]

# Resizing the ROI so it can be fed to the model for prediction
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)
```

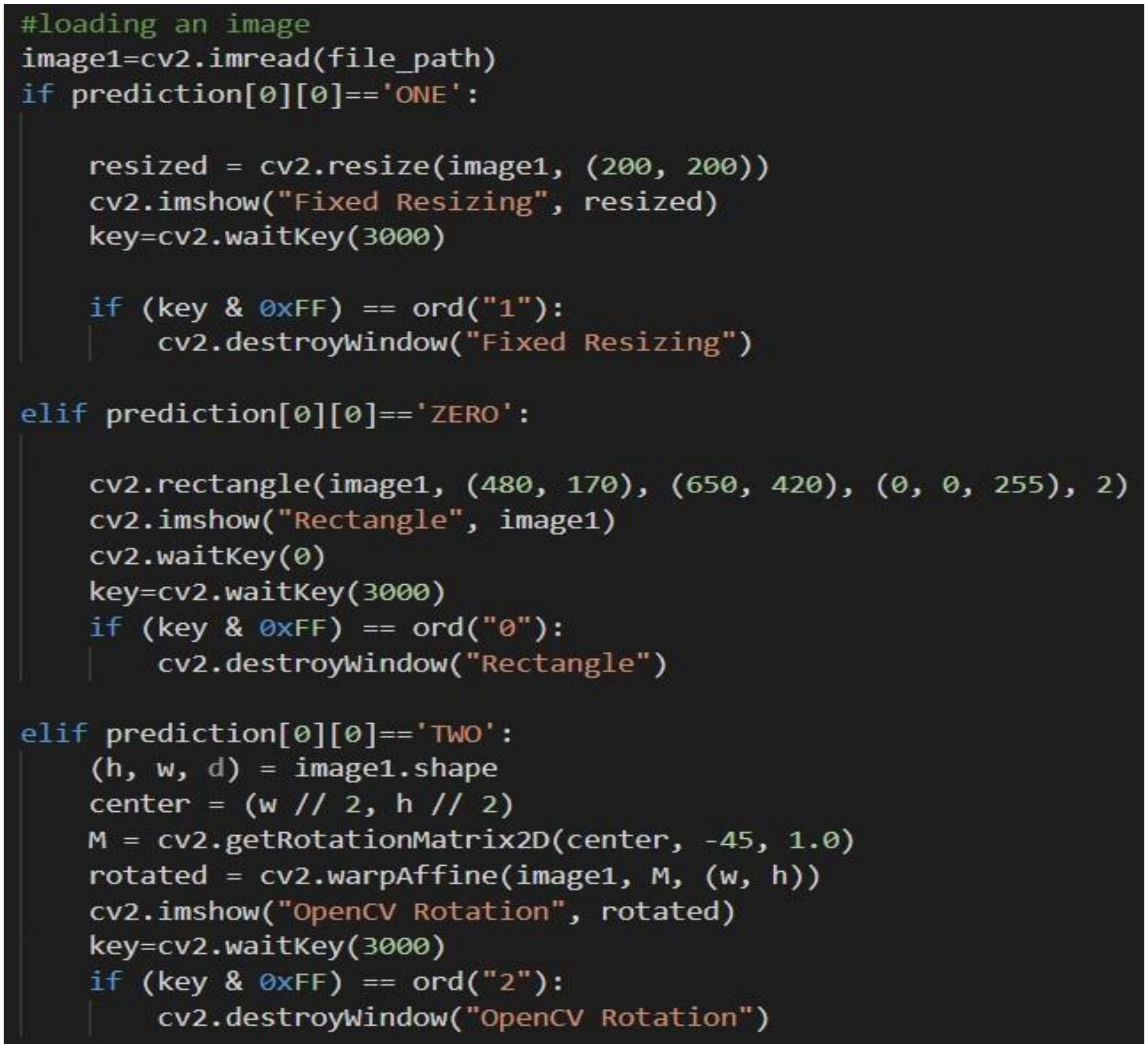
Predicting our results:

A er placing the ROI and ge ng the frames from the web cam now its me to predict the gesture result using the model which we trained and stored it into a variable for the further opera ons.

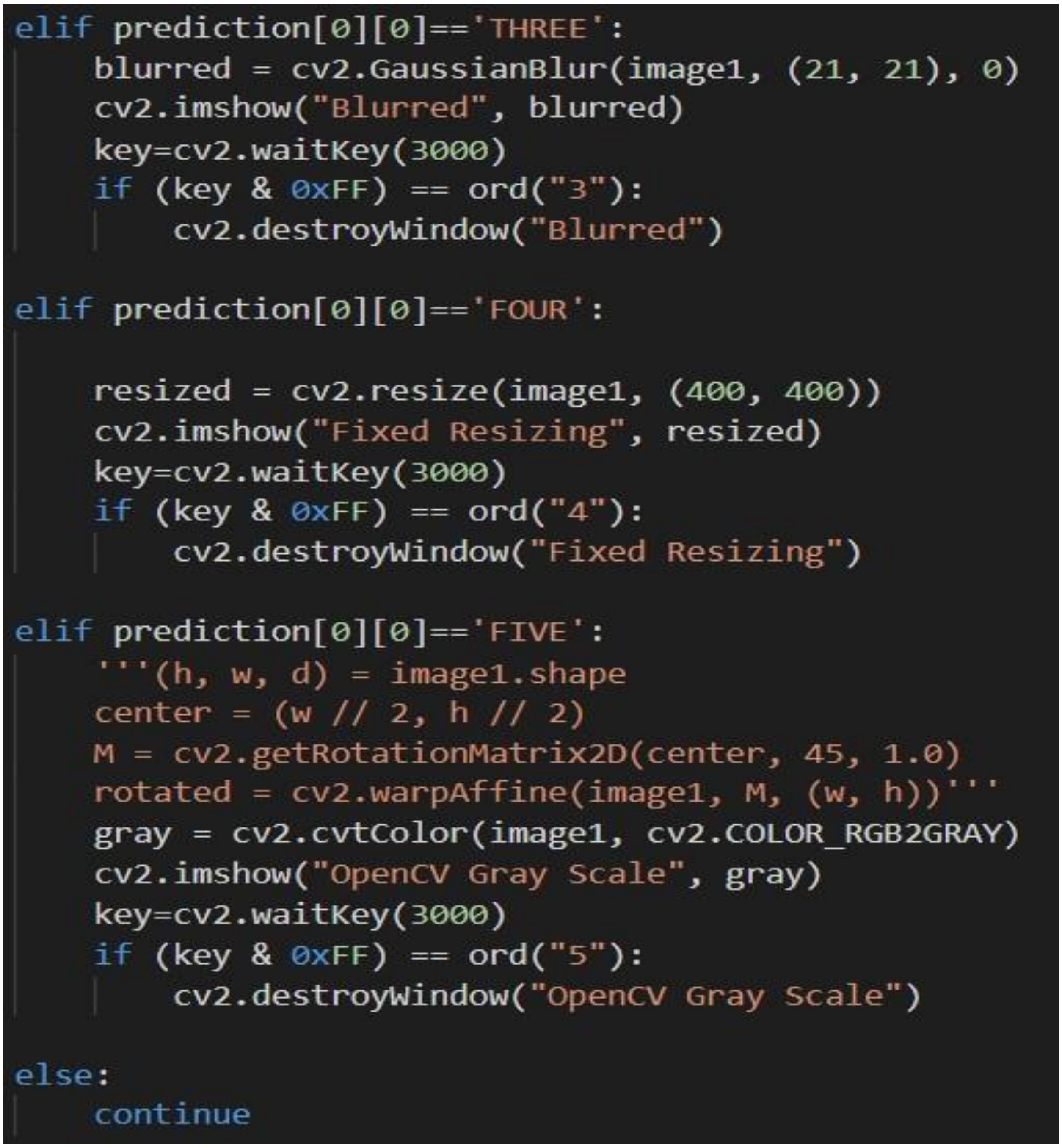


```python
result = model.predict(test_image.reshape(1, 64, 64, 1))
prediction = {'ZERO': result[0][0],
              'ONE': result[0][1],
              'TWO': result[0][2],
              'THREE': result[0][3],
              'FOUR': result[0][4],
              'FIVE': result[0][5]}
# Sorting based on top prediction
prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)

# Displaying the predictions
cv2.putText(frame, prediction[0][0], (10, 120), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
```

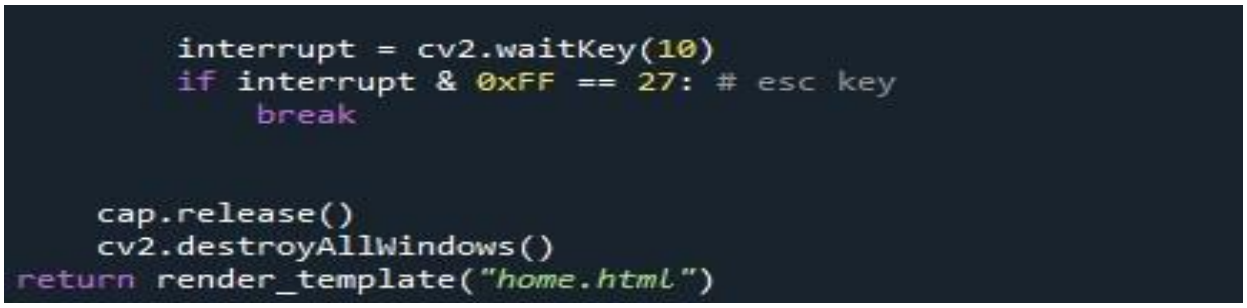
Finally according to the result predicted with our model we will be performing certain opera ons like resize, blur , rotate etc.



```python
#loading an image
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)
    cv2.waitKey(0)
    key=cv2.waitKey(3000)
    if (key & 0xFF) == ord("0"):
        cv2.destroyWindow("Rectangle")

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")
```



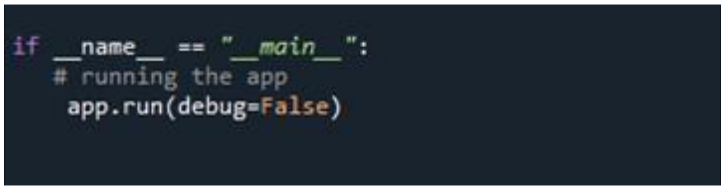
```python
elif prediction[0][0]=='THREE':
    blurred = cv2.GaussianBlur(image1, (21, 21), 0)
    cv2.imshow("Blurred", blurred)
    key=cv2.waitKey(3000)
    if (key & 0xFF) == ord("3"):
        cv2.destroyWindow("Blurred")

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':
    '''(h, w, d) = image1.shape
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, 45, 1.0)
    rotated = cv2.warpAffine(image1, M, (w, h))'''
    gray = cv2.cvtColor(image1, cv2.COLOR_RGB2GRAY)
    cv2.imshow("OpenCV Gray Scale", gray)
    key=cv2.waitKey(3000)
    if (key & 0xFF) == ord("5"):
        cv2.destroyWindow("OpenCV Gray Scale")

else:
    continue
```



```python
        interrupt = cv2.waitKey(10)
        if interrupt & 0xFF == 27: # esc key
            break


    cap.release()
    cv2.destroyAllWindows()
return render_template("home.html")
```

Run The Application:

At last, we will run our flask applica on

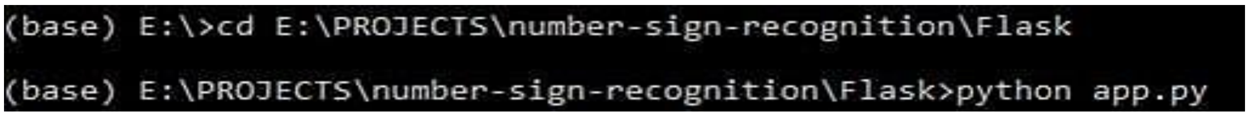


```
if __name__ == "__main__":
   # running the app
    app.run(debug=False)
```

Run The app in local browser

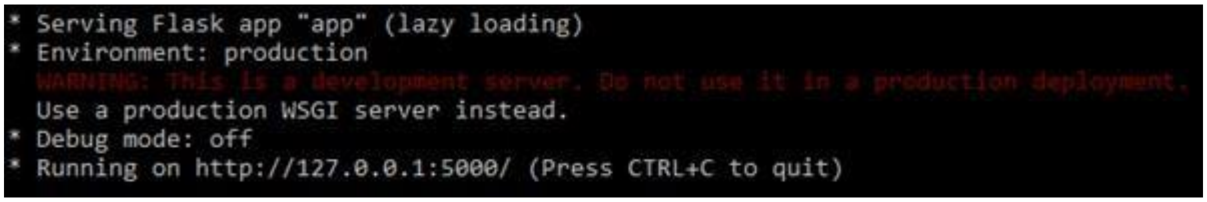
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command

Navigate to the localhost where you can view your web page



```
(base) E:\>cd E:\PROJECTS\number-sign-recognition\Flask

(base) E:\PROJECTS\number-sign-recognition\Flask>python app.py
```

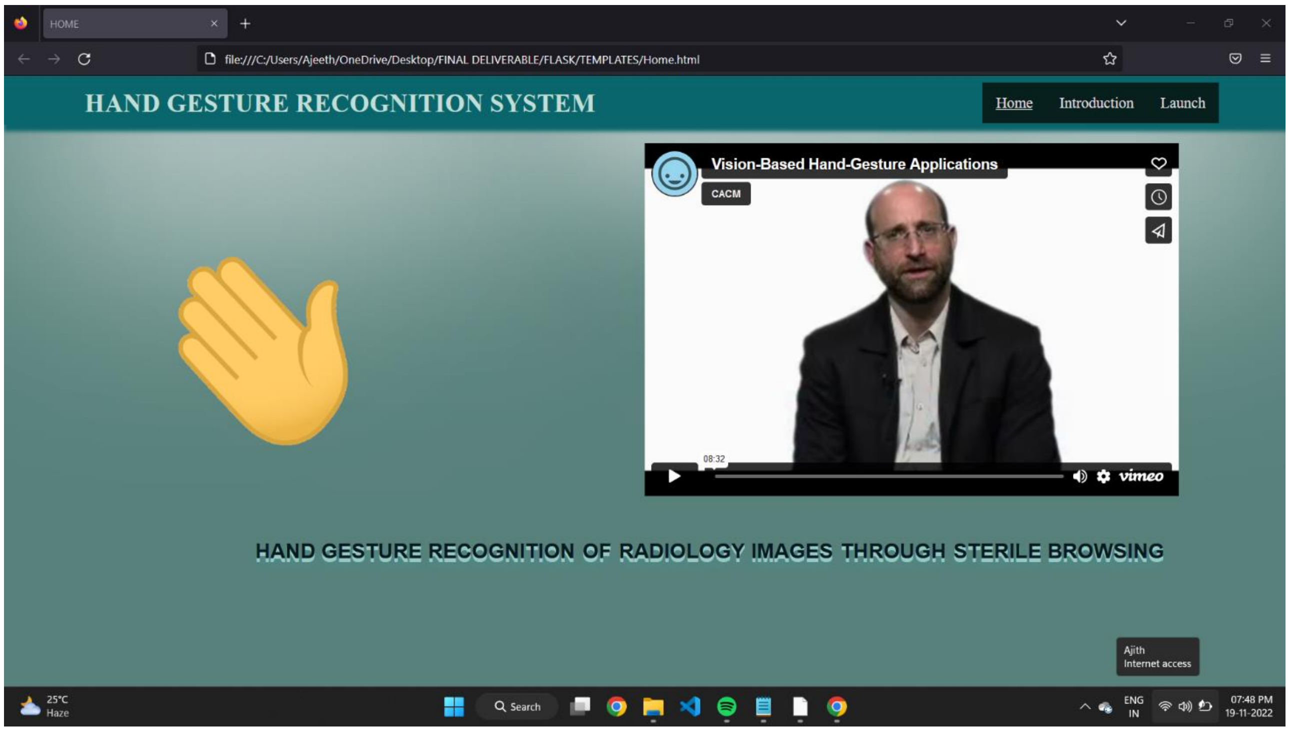
Then it will run on localhost:5000



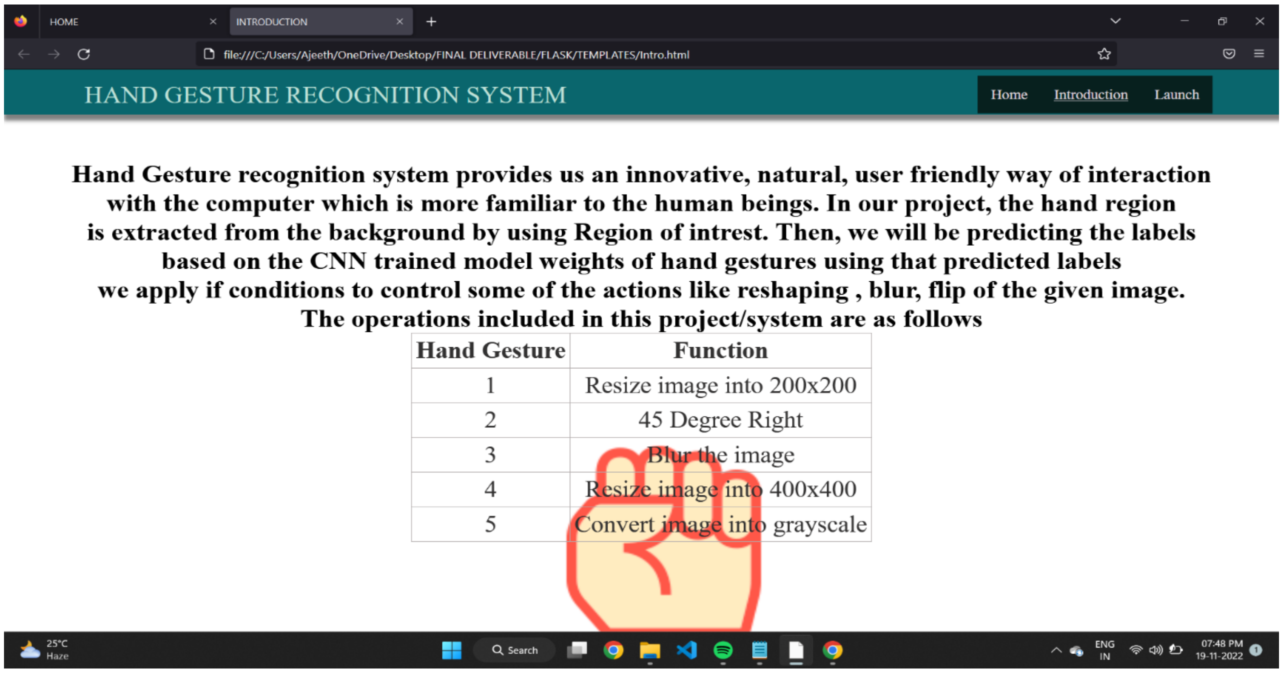
```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Navigate to the localhost (h p://127.0.0.1:5000/)where you can view your web page.

Let’s see how our home.html page looks like:


HOME
file:///C:/Users/Ajeeth/OneDrive/Desktop/FINAL DELIVERABLE/FLASK/TEMPLATES/Home.html
HAND GESTURE RECOGNITION SYSTEM
Home
Introduction
Launch
Vision-Based Hand-Gesture Applications
CACM
vimeo
HAND GESTURE RECOGNITION OF RADIOLOGY IMAGES THROUGH STERILE BROWSING

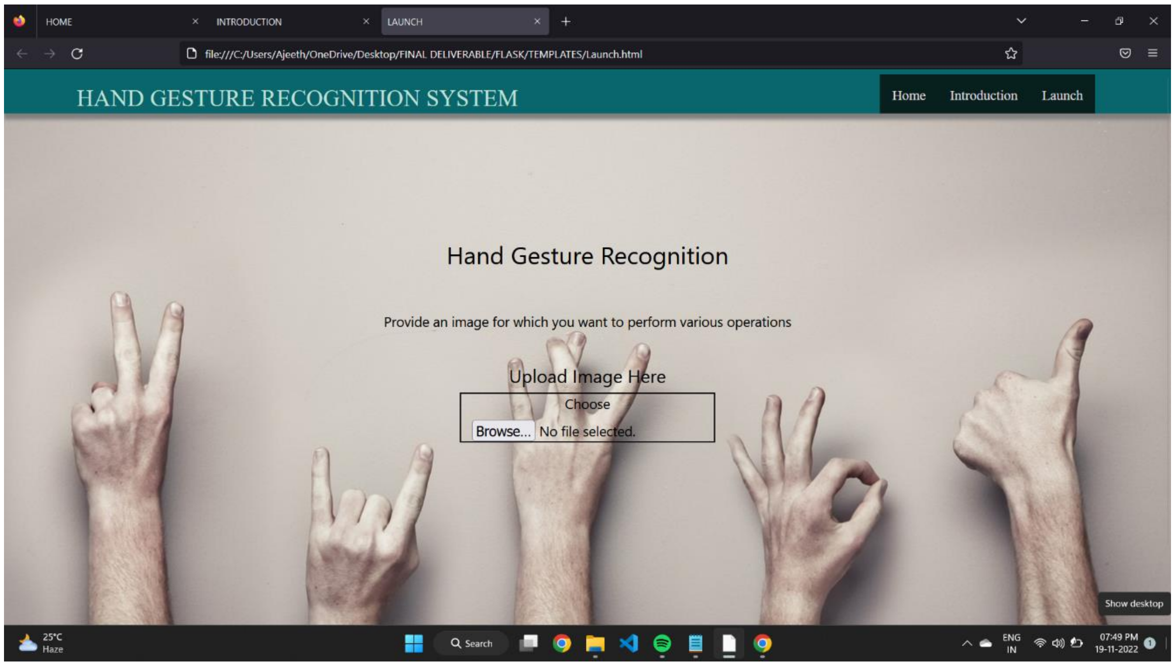
When “Info” bu on is clicked, localhost redirects to “intro.html”


HOME
INTRODUCTION
file:///C:/Users/Ajeeth/OneDrive/Desktop/FINAL DELIVERABLE/FLASK/TEMPLATES/Intro.html
HAND GESTURE RECOGNITION SYSTEM
Home
Introduction
Launch

Hand Gesture recognition system provides us an innovative, natural, user friendly way of interaction with the computer which is more familiar to the human beings. In our project, the hand region is extracted from the background by using Region of intrest. Then, we will be predicting the labels based on the CNN trained model weights of hand gestures using that predicted labels we apply if conditions to control some of the actions like reshaping , blur, flip of the given image. The operations included in this project/system are as follows

Hand Gesture	Function
1	Resize image into 200x200
2	45 Degree Right
3	Blur the image
4	Resize image into 400x400
5	Convert image into grayscale

Upload the image and click on Predict bu on to view the result


HOME
INTRODUCTION
LAUNCH
file:///C:/Users/Ajeeth/OneDrive/Desktop/FINAL DELIVERABLE/FLASK/TEMPLATES/Launch.html
HAND GESTURE RECOGNITION SYSTEM
Home
Introduction
Launch
Hand Gesture Recognition
Provide an image for which you want to perform various operations
Upload Image Here
Choose
Browse... No file selected.
Show desktop
07:49 PM
19-11-2022

OVERALL PROJECT SUMMARY

PHASES COMPLETED:

- **IDEATION PHASE:**
  - **Brainstorming**
  - **Problem Statement**
  - **Empathy Map Canvas**
  - **Idea on Of Project ideas**
  - **Literature Survey**
- **PROJECT DESIGN PHASE-1:**
  - **Problem Solu on Fit**
  - **Proposed Solu on**
  - **Solu on Architecture**
- **PROJECT DESIGN PHASE-2:**

- Customer Journey Map
- Data Flow Diagrams
- User Stories
- Technology Stack
- Func onal Requirements

- **PROJECT PLANNING PHASE:**
  - Prepare Milestone & Activity List
  - Sprint Delivery Plan
- **PROJECT DEVELOPMENT PHASE-1:**
- **SPRINT-1:**
  - Data collection
  - Image Pre-processing
- **SPRINT-2:**
  - Model Building
- **SPRINT-3:**
  - Static
- **SPRINT-4:**
  - uploads
  - App.py
- **PROJECT ASSIGNMENTS:**
  - Assignment-1
  - Assignment-2
  - Assignment-3
  - Assignment-4

- **PROJECT QUIZ:**
    - **Quiz-1**
    - **Quiz-2**
    - **Quiz-3**
    - **Quiz-4**