A GESTURE BASED TOOL FOR STERILE BROWSING OF RADIOLOGY IMAGES

Date	17-Nov-2022
TeamID	PNT2022TMID29238
ProjectName	A Gesture based tool for sterile browsing of radiology images

Submitted by:

TEAM LEADER:

E.SHARMILA-421619205041

TEAM MEMBERS:

T.BOOMIKA-421619205006

S.HARINIPRIYA -421619205013

T.S.MAHALAKSHMI-421619205024

M.ABIRAMI-421619205301

ABSTRACT:

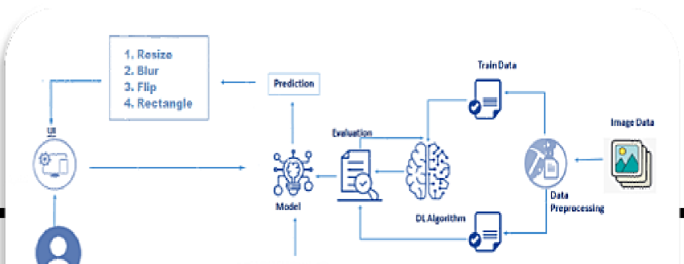
Humans can recognize body and sign language easily. This is possible due to the combination of vision and synaptic interactions that were formed along brain development. In order to replicate this skill in computers, some problems need to be solved: how to separate objects of interest in images and which 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 into (200,200), 2 - image is rotated by -45°, 3 - image is blurred, 4 - image is Resized into (400,400), 5 - image is converted into grayscale etc.

PROJECT OBJECTIVES:

- ➤ Show fundamental techniques and concepts of convolutional neural network (CNN)
- ➤ Gain a broad understanding of broad data
- ➤ Know how to build a web application using Flask framework.

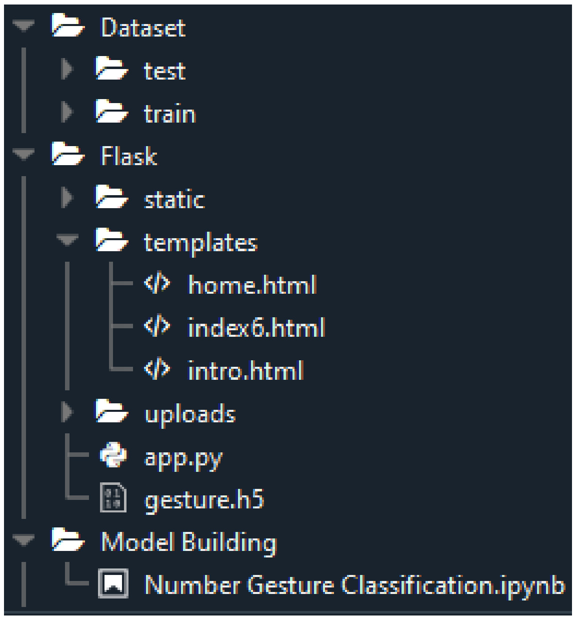
TECHNICAL ARCHITECTURE:



OVERVIEW:

1. Defining our classification categories
2. Collect training images
3. Train the model
4. Test our model

PROJECT STRUCTURE:



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

- ➤ Dataset folder contains the training and testing images for training our model.
- ➤ We are building a Flask Application which needs HTML pages stored in the templates folder and a python script app.py for server side scripting
- ➤ 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

PROJECT FLOW:

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

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

- **Data Collection**

  1.Collect the dataset or Create the dataset
- **Data Pre processing**
  1. Import the ImageDataGenerator library
  2. Configure ImageDataGenerator class
  3. Apply ImageDataGenerator functionality to Trainset and Testset
- **Model Building**
  1. Import the model building Libraries
  2. Initializing the model
  3. Adding Input Layer
  4. Adding Hidden Layer
  5. Adding Output Layer
  6. Configure the Learning Process
  7. Training and testing the model
  8. Save the Model
- **Application Building**
  1. Create an HTML file
  2. Build Python Code

Following software, concepts and packages are used in this project

- **Anaconda navigator**

- **Python packages:**
    1. Open anaconda prompt as administrator
    2. Type “pip install TensorFlow”
    3. Type “pip install opencv-python”
    4. Type “pip install flask”

DEEP LEARNING CONCEPTS:

CNN:

A convolutional 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 detection and object detection.

FLASK:

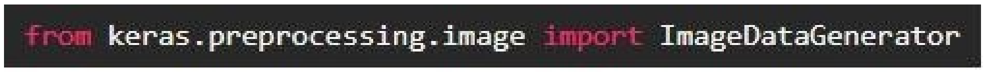
Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.

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 actions.

IMAGE PREPROCESSING:

In this step we improve the image data that suppresses unwilling distortions or enhances some image features important for further processing, although perform some geometric transformations of images like rotation, scaling, translation etc.



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

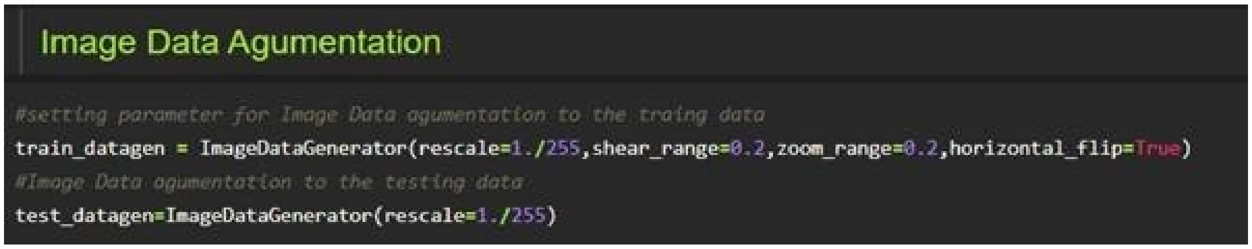
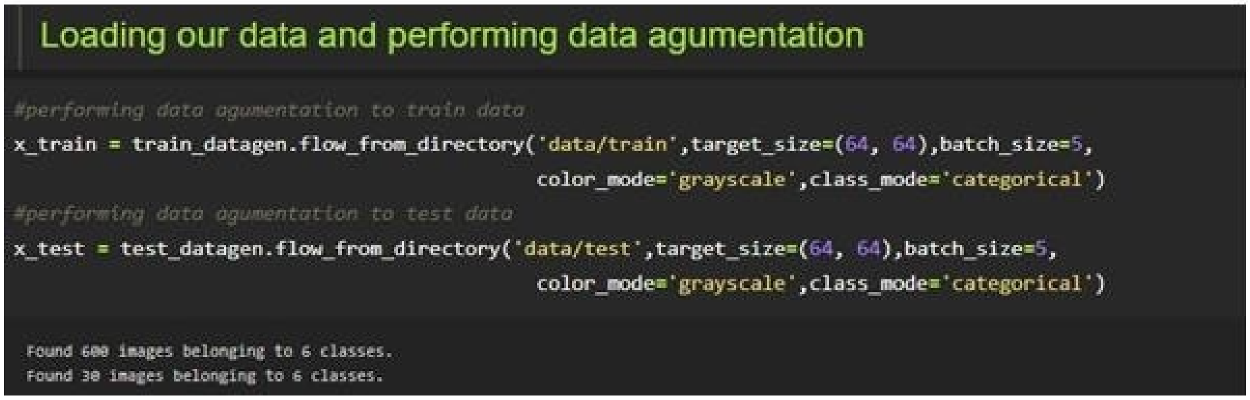


Image Data Agumentation

```
#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

```
#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.
```

- Apply ImageDataGenerator Functionality To Trainset And Testset
- Let us apply ImageDataGenerator functionality to Trainset and Testset by using the following code
- For Trainingset using flow_from_directory function.
- This function will return 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}

Arguments:

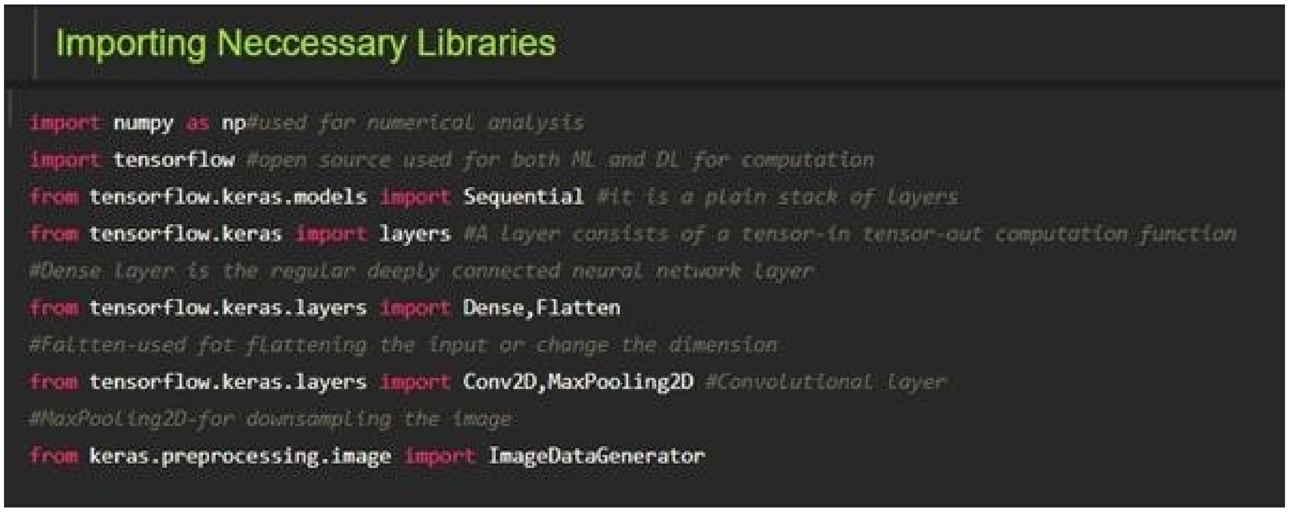
- directory: Directory where the data is located. If labels is "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- batch_size: Size of the batches of data. Default: 32.
- target_size: Size to resize images to after they are read from disk.
- class_mode:
- - ‘int': means that the labels are encoded as integers (e.g. for

sparse_categorical_crossentropy loss).

- - 'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical_crossentropy loss).
- - 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary_crossentropy).
- - None (no labels).

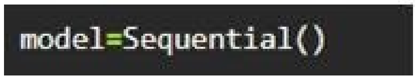
MODEL BUILDING:

In this step we build Convolutional Neural Networking which contains a input layer along with the convolution, maxpooling and finally a output layer.

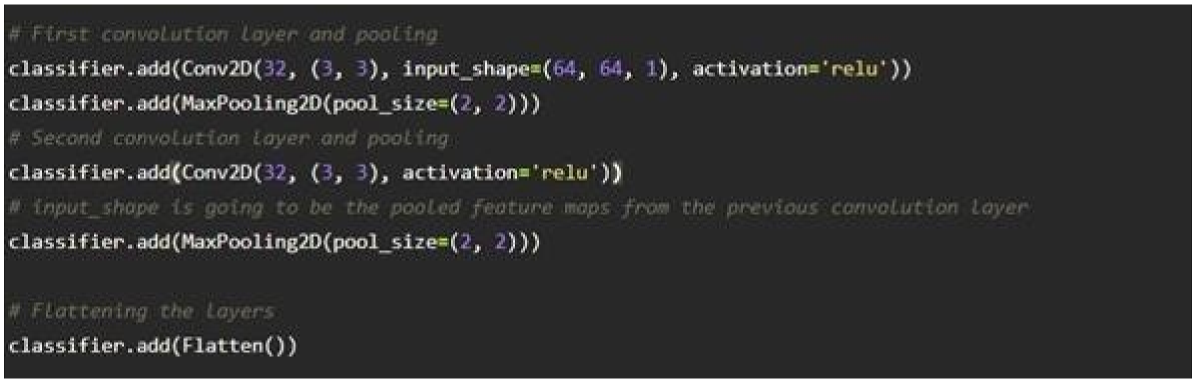


Importing Neccessary Libraries

```
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
```



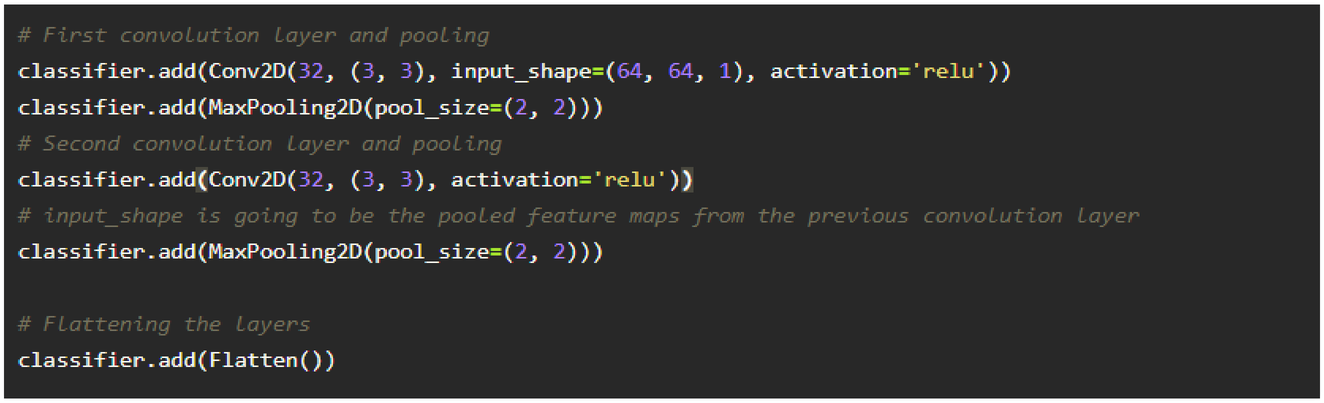
```
model=Sequential()
```



```
# 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 CNN layer:

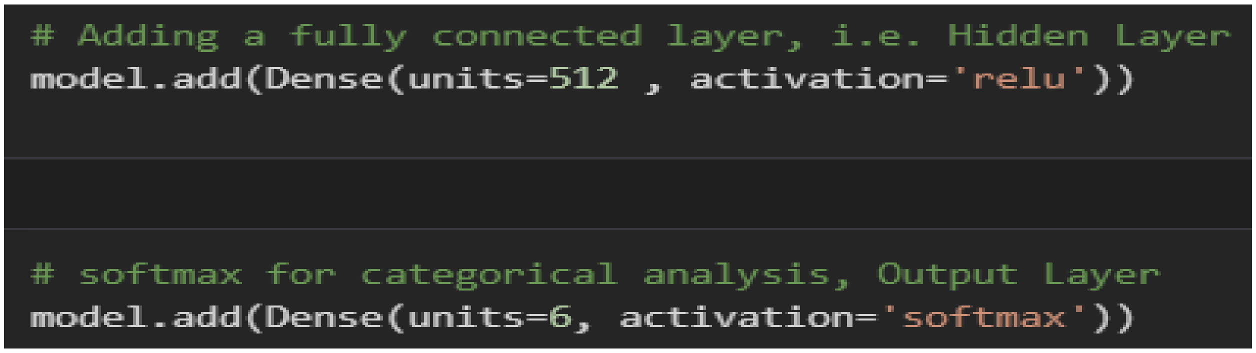
- We are adding a convolution layer with activation function as “relu” and with a small filter size (3,3) and number of filters (32) followed by a max pooling layer.
- Maxpool layer is used to downsample the input.
- Flatten layer flattens the input. Does not affect the batch size.



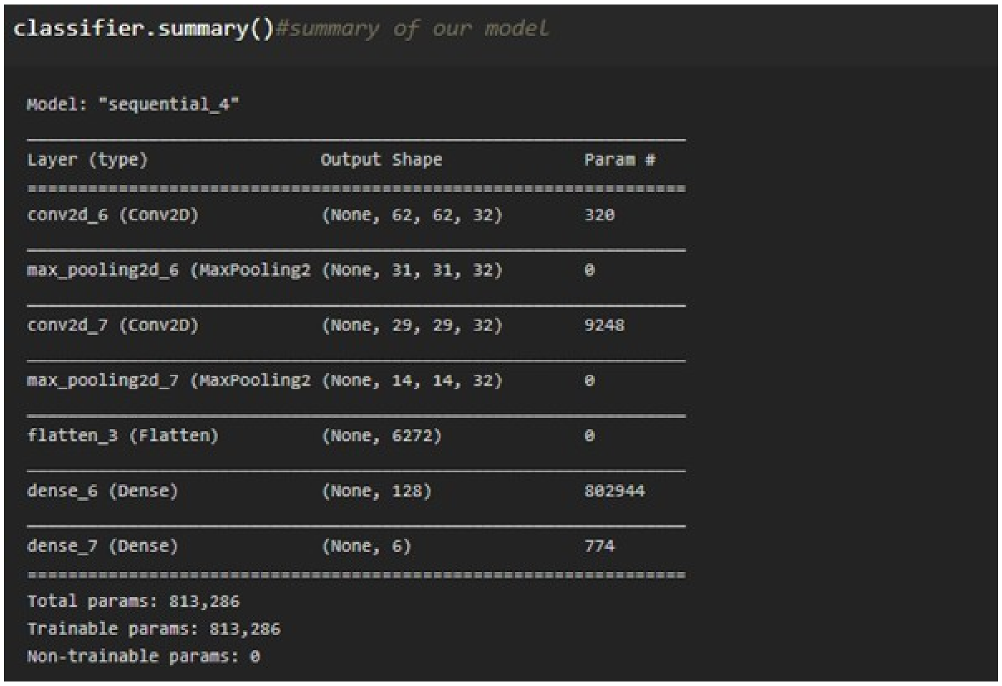
```
# First convolution layer and pooling
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 1), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Second convolution layer and pooling
classifier.add(Conv2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled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

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

Adding Dense Layer:

- 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 purposes. Keras provides a simple method, summary to get the full information about the model and its layers.



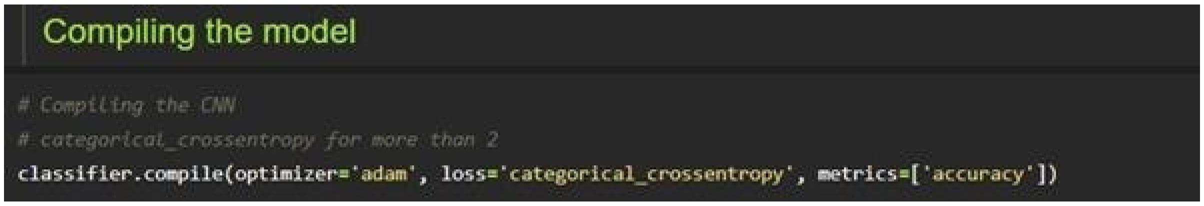
```
# Adding a fully connected layer, i.e. Hidden Layer
model.add(Dense(units=512 , activation='relu'))


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



```
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 compilation is the final step in creating a model. Once the compilation is done, we can move on to training phase. Loss function is used to find error or deviation in the learning process. Keras requires loss function during model compilation process.
- Optimization is an important process which optimize the input weights by comparing the prediction and the loss function. Here we are using Adam optimizer
- Metrics is used to evaluate the performance of your model. It is similar to loss function, 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'])
```

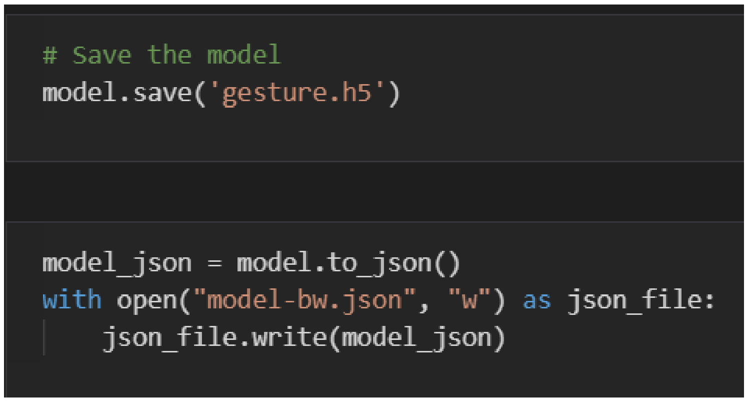
<u>**TRAIN THE MODEL:**</u>

Train the model with our image dataset.

fit_generator functions 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 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.
- validation_steps :only if the validation_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 validation data points in your dataset divided by the validation batch size.



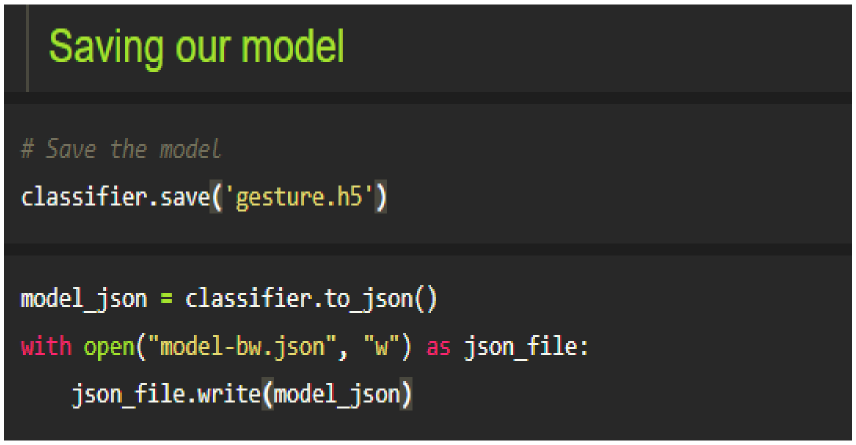
```
# Save the model
model.save('gesture.h5')
```

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

SAVING THE MODEL:

The model is saved with .h5 extension as follows

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.



Saving our model

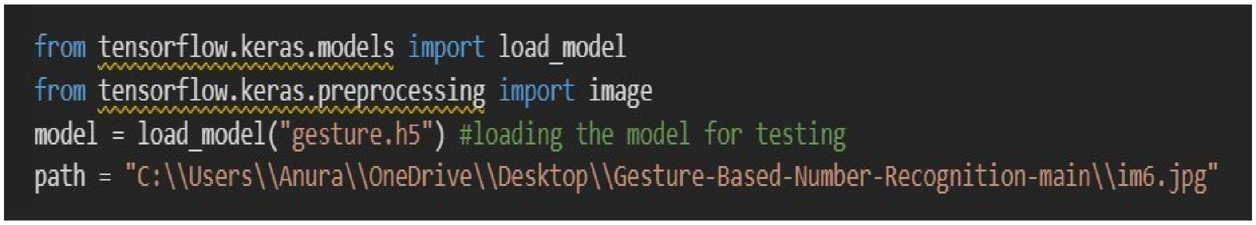
```
# Save the model
classifier.save('gesture.h5')
```

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

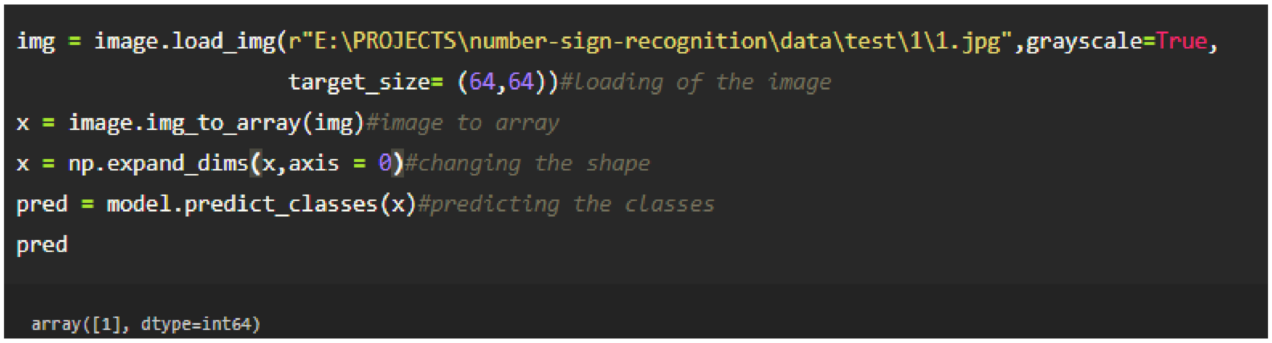
TEST THE MODEL:

Evaluation 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

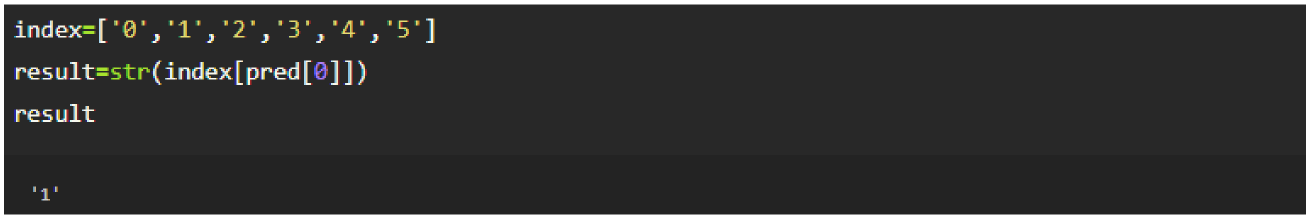


```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model = load_model("gesture.h5") #loading the model for testing
path = "C:\\Users\\Anura\\OneDrive\\Desktop\\Gesture-Based-Number-Recognition-main\\im6.jpg"
```



```
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
```

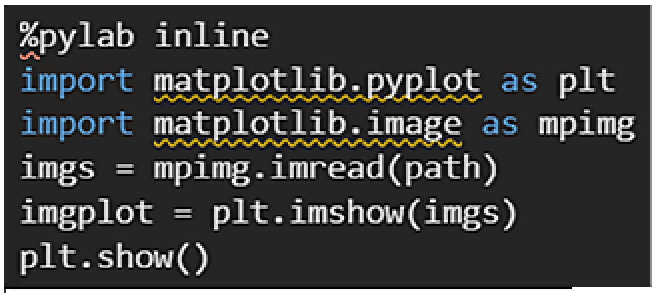
```
array([1], dtype=int64)
```



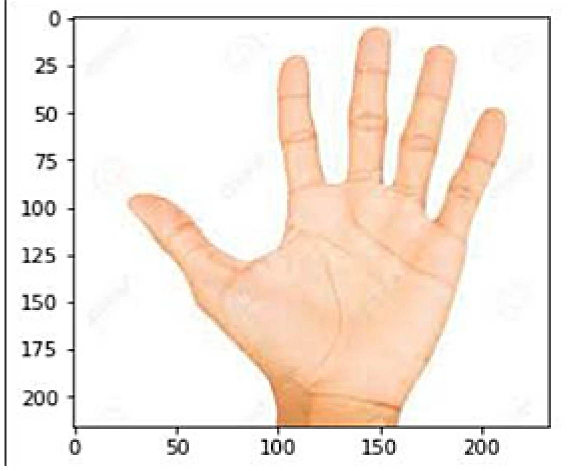
```
index=['0','1','2','3','4','5']
result=str(index[pred[0]])
result
```

```
'1'
```

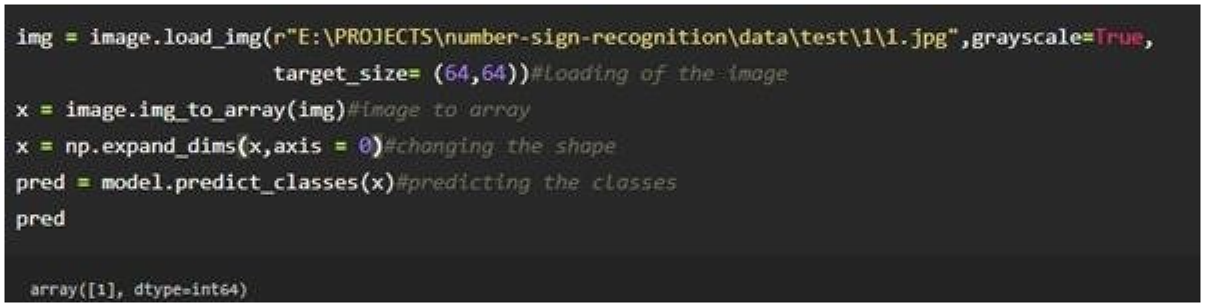
PLOTTING IMAGES:



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

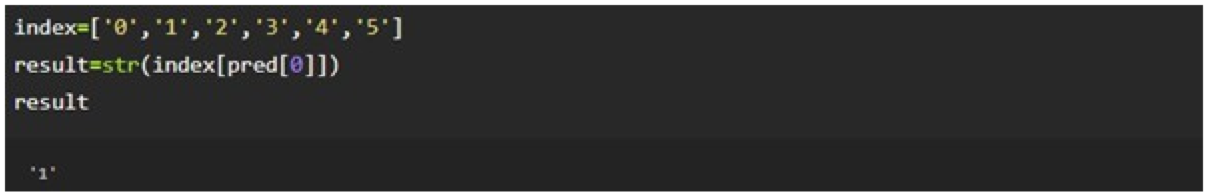


Taking an image as input and checking the results:

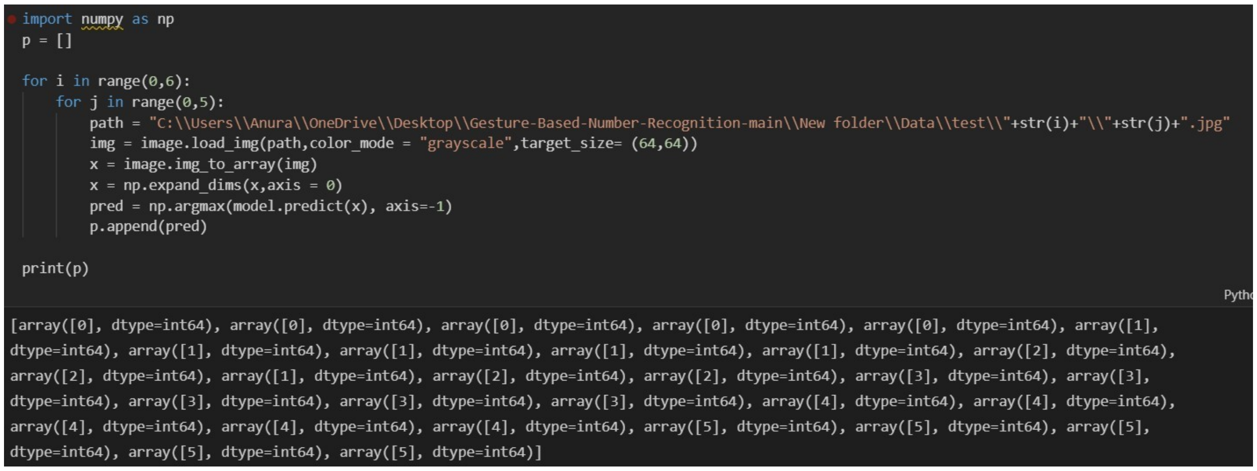


```
img = image.load_img(r"E:\PROJECTS\number-sign-recognition\data\test\1\1.jpg",grayscale=True,
                     target_size= (64,64))#Loading of the image
x = image.img_to_array(img)#image to array
x = np.expand_dims(x,axis = 0)#changing the shape
pred = model.predict_classes(x)#predicting the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

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

By using the model we are predicting the output for the given input image:



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

'1'
```



```python
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)
```

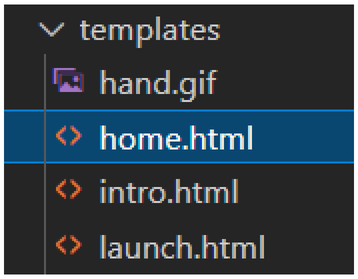
```
[array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([1],
dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([2], dtype=int64),
array([2], dtype=int64), 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:

After the model is trained in this particular step, we will be building our flask application 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 index6.html
- ➤ home.html displays home page.
- ➤ Intro.html displays introduction about the hand gesture recognition
- ➤ index6.html accepts input from the user and predicts the values.
- ➤ We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.

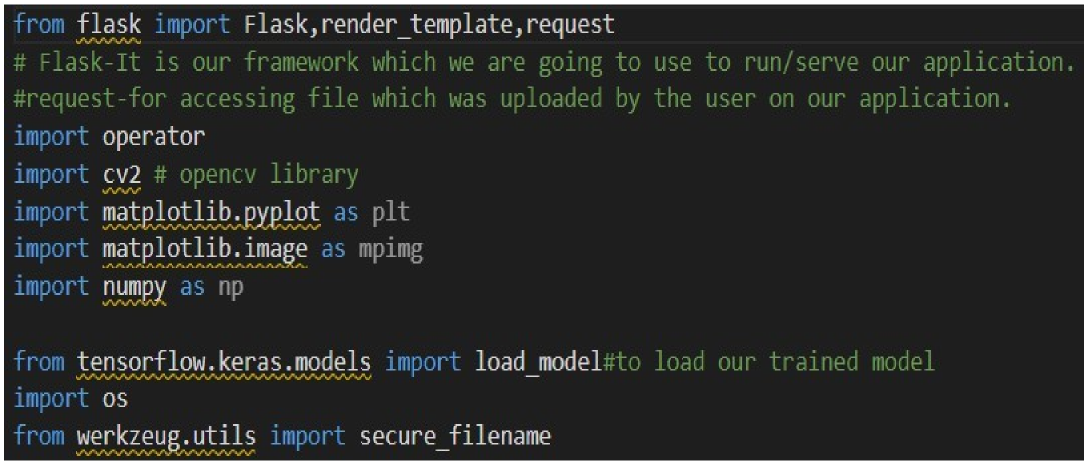


```
templates
    hand.gif
    home.html
    intro.html
    launch.html
```

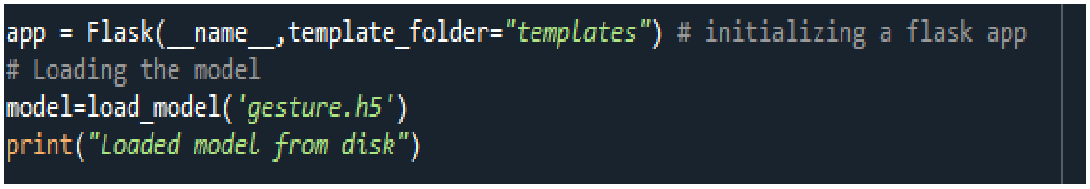
BUILD PYTHON CODE:

- Build flask file ‘app.py’ which is a web framework written in python for server-side scripting.
- 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.

- **Importing Libraries:**

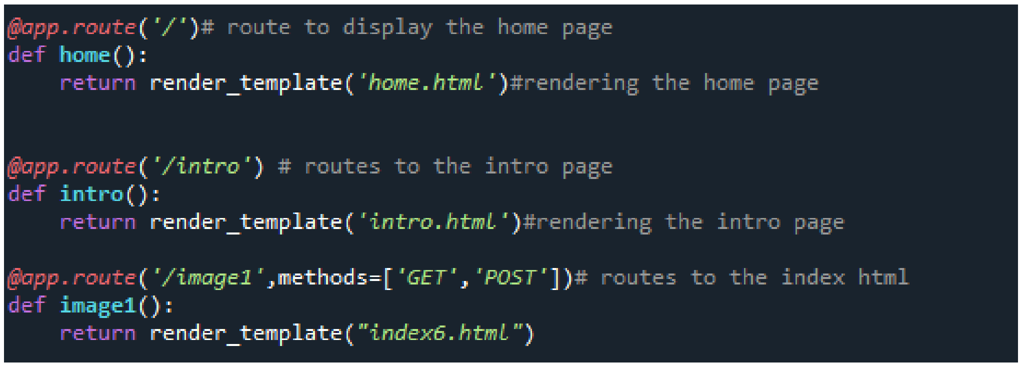


```
from flask import Flask,render_template,request
# Flask-It is our framework which we are going to use to run/serve our application.
#request-for accessing file which was uploaded by the user on our application.
import operator
import cv2 # opencv library
import matplotlib.pyplot as plt
import matplotlib.image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

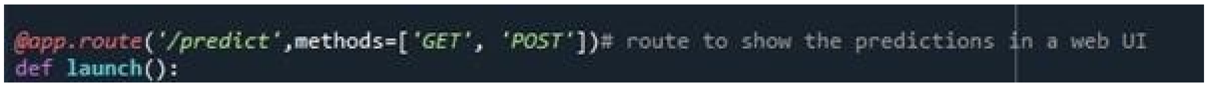


```
app = Flask(__name__,template_folder="templates") # initializing a flask app
# Loading the model
model=load_model('gesture.h5')
print("Loaded model from disk")
```

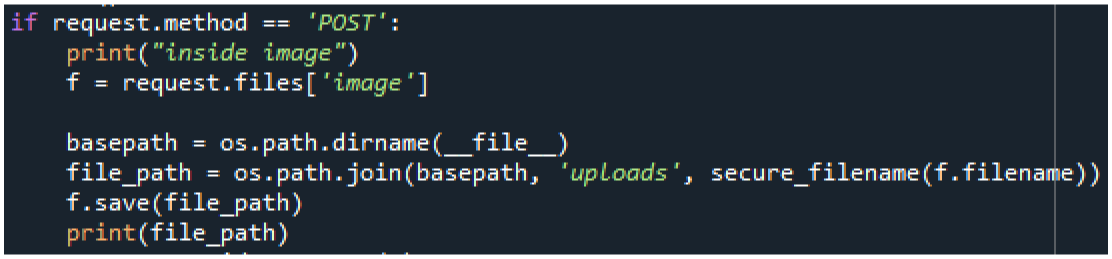
- **Routing to the html Page:**



```
@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")
```



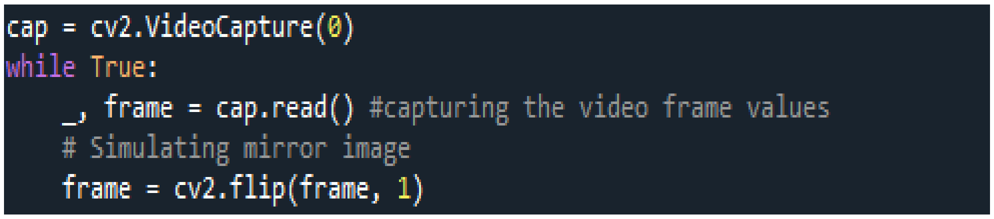
```
@app.route('/predict',methods=['GET', 'POST'])# route to show the predictions in a web UI
def launch():
```



```
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)
```

<u>Grab the frames from the webcam:</u>

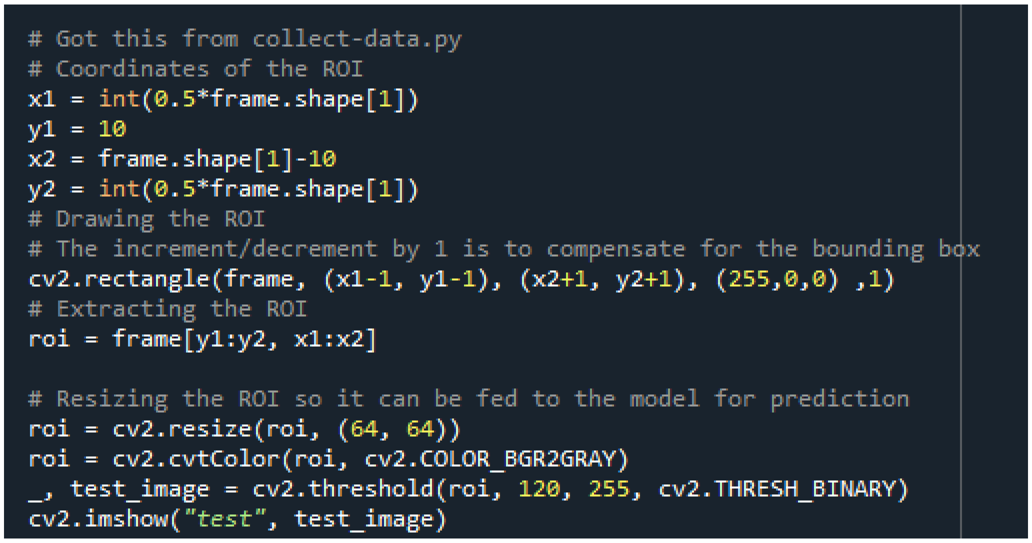
When we run the code a web cam will be opening to take the gesture input so we will be capturing the frames of the gesture for predicting our results.



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

<u>Creating ROI:</u>

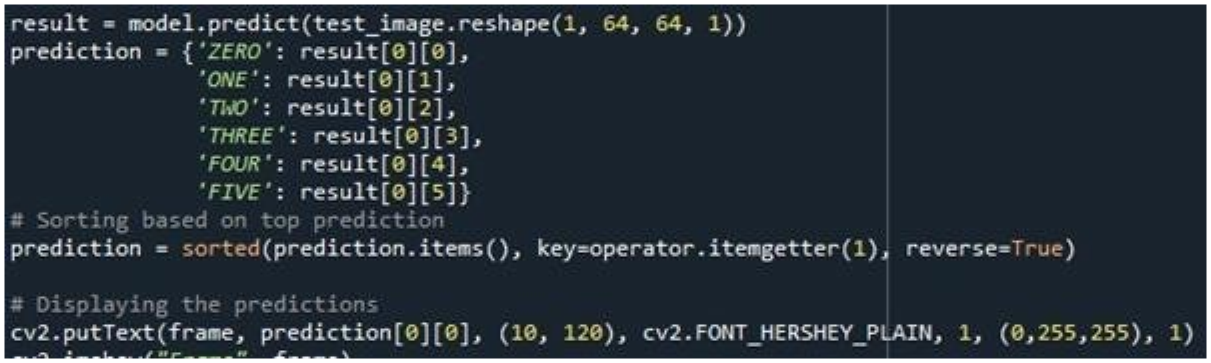
A region of interest (ROI) is a portion of an image that you want to filter or operate on in some way. The toolbox supports a set of ROI objects that you can use to create ROIs of many shapes, such circles, ellipses, polygons, rectangles, and hand-drawn shapes. A common use of an ROI is to create a binary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



```
# 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 THE RESULTS:

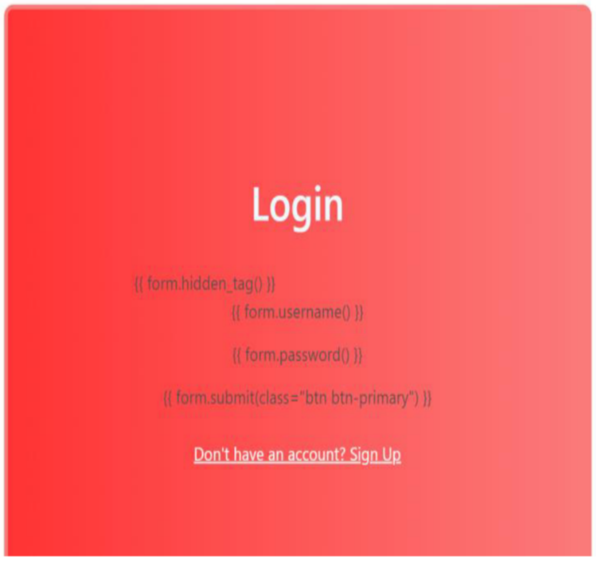
After placing the ROI and getting the frames from the web cam now its time to predict the gesture result using the model which we trained and stored it into a variable for the further operations.



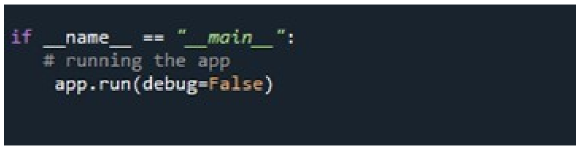
```
result = model.predict(test_image.reshape(1, 64, 64, 1))
prediction = {'ZERO': result[0][0],
              'ONE': result[0][1],
              'TWO': result[0][2],
              'THREE': result[0][3],
              'FOUR': result[0][4],
              'FIVE': result[0][5]}
# Sorting based on top prediction
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 operations like resize, blur , rotate etc.

RUN THE APPLICATION:


Login
{{ form.hidden_tag() }}
{{ form.username() }}
{{ form.password() }}
{{ form.submit(class="btn btn-primary") }}
Don't have an account? Sign Up

At last, we will run our flask application

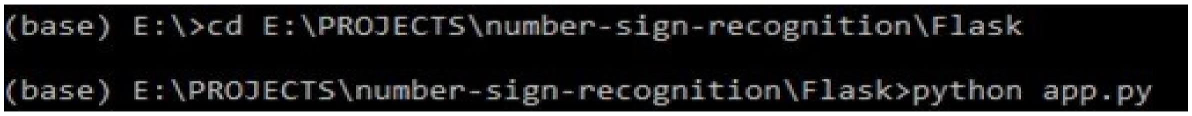


```
if __name__ == "__main__":
    # running the app
    app.run(debug=False)
```

Run The app in local browser

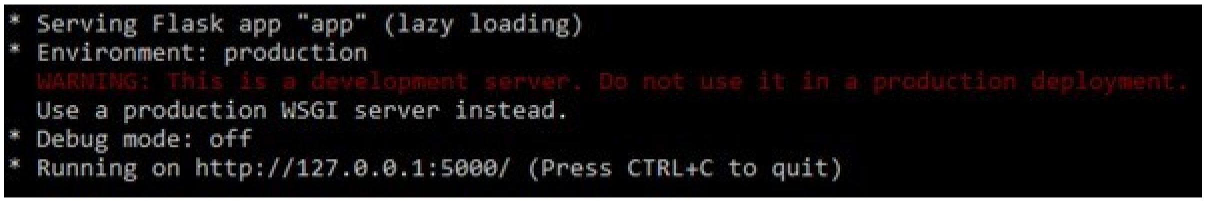
- ➤ Open anaconda prompt from the start menu
- ➤ Navigate to the folder where your python script is.
- ➤ Now type "python app.py" command

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 (http://127.0.0.1:5000/)where you can view your web page.

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

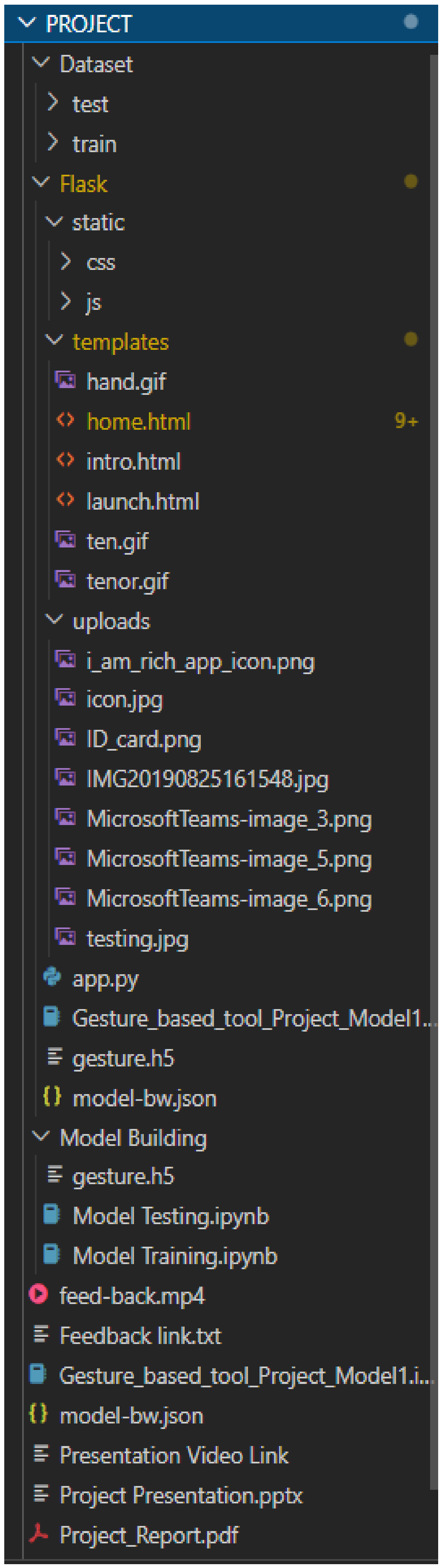
When “Info” button is clicked, localhost redirects to “intro.html”
Upload the image and click on choose and then predict button to view the result

APPLICATIONS:

- This hand based gesture tool developed can be mainly used in the medical industry to browse images without compromising the sterility.
- However it can also be used in different industries while presenting certain ideas, during meetings, and can be used by teachers while teaching.

RESULT:

- Final Findings(Output) of the project along with screenshots.
- Through this project we found that we can maintain the sterility of an operation theater, Etc by using hand based gesture tools to browse the images obtained.



```
PROJECT
  Dataset
    test
    train
  Flask
    static
      css
      js
    templates
      hand.gif
      home.html                      9+
      intro.html
      launch.html
      ten.gif
      tenor.gif
    uploads
      i_am_rich_app_icon.png
      icon.jpg
      ID_card.png
      IMG20190825161548.jpg
      MicrosoftTeams-image_3.png
      MicrosoftTeams-image_5.png
      MicrosoftTeams-image_6.png
      testing.jpg
    app.py
    Gesture_based_tool_Project_Model1...
    gesture.h5
    model-bw.json
  Model Building
    gesture.h5
    Model Testing.ipynb
    Model Training.ipynb
  feed-back.mp4
  Feedback link.txt
  Gesture_based_tool_Project_Model1.i...
  model-bw.json
  Presentation Video Link
  Project Presentation.pptx
  Project_Report.pdf
```

RESULT:

- Final Findings(Output) of the project along with screenshots.
- Through this project we found that we can maintain the sterility of an operation theater, Etc by using hand based gesture tools to browse the images obtained.

ADVANTAGES:

- Major advantange of this tool is that it helps to maintain the sterility of the environment.
- It is also easy to use and is quicker than the existing methods to browse images.
- It can also be performed even if the surgeon is a bit far away from the system, this Helps to save time.
- The tool does not need the person using it to have an apparatus or any devices on Them to use it.
- They can simply move their hands to browse through the images.

DISADVANTAGES:

- The tool can be quite experience as it requires cameras and other expensive devices to capture images and process it.

CONCLUSION:

- In this project we developed a tool which recognises hand gestures and enables doctors
- To browse through radiology images using these gestures. This enables doctors and Surgeons to maintain the sterility as they would not have to touch any mouser or Keyboard to go through the images.
- This tool is also easy to use and is quicker than the regular method of using Mouse/keyboard.
- It also does not require the user location since they don’t have to be in contact with any Device.
- It also does not require the user to have any device on them to use it.

Further this technology can be extended to other industries .