# A Gesture-Based Tool for Sterile Browsing of Radiology Images

# CNN and Open CV

**Introduction:**

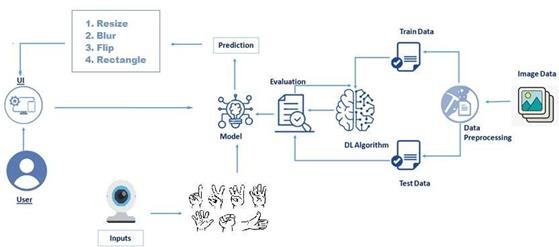
Humans can recognize body and sign language easily. This is possible due to the combination of vision and synaptic interactions that were formed along brain development. In order to replicate this skill in computers, some problems need to be solved: how to separate objects of interest in images and which 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

* Know fundamental concepts and techniques of Convolutional Neural Network (CNN).
* Gain a broad understanding of image data.
* Know how to pre-process/clean the data using different data pre- processing techniques.
* 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 mode

# 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.
  + Collect the dataset or Create the dataset
* Data Pre processing
  + Import the ImageDataGenerator library
  + Configure ImageDataGenerator class
  + Apply ImageDataGenerator functionality to Trainset and Testset
* 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

Following software, 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 opencv-python”
  + 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

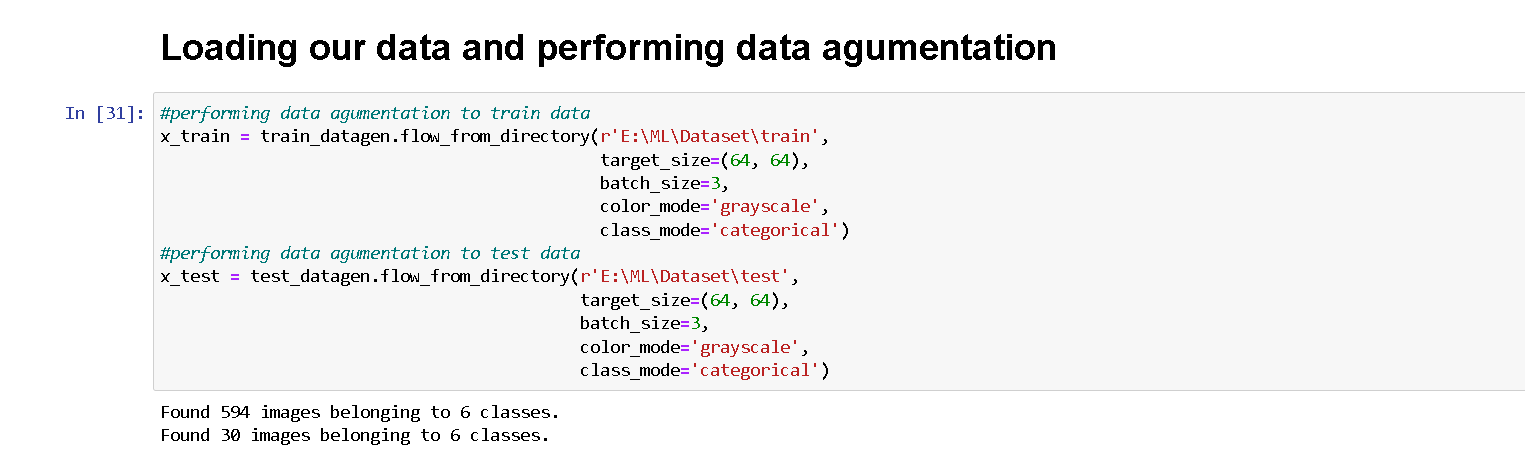
ML depends heavily on data, without data, it is impossible for a machine to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

# Image Preprocessing

In this step we improve the image data that suppresses unwilling distortions or enhances some image features important for further processing, although

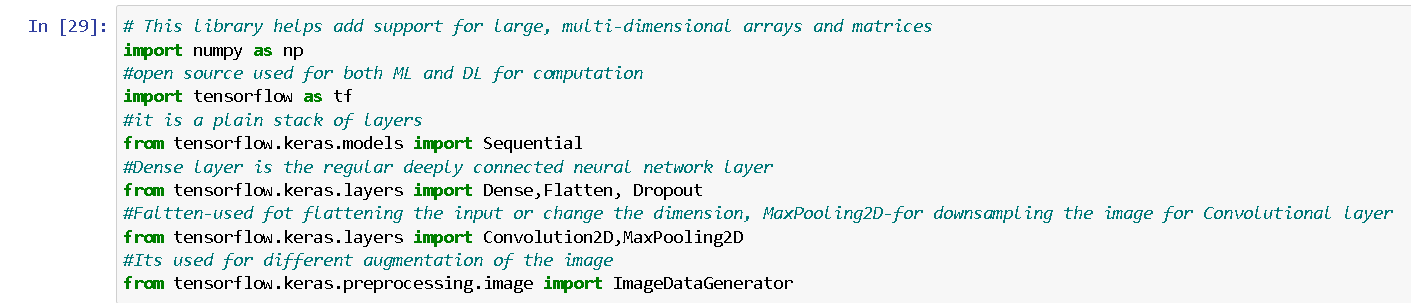
perform some geometric transformations of images like rotation, scaling, translation etc.



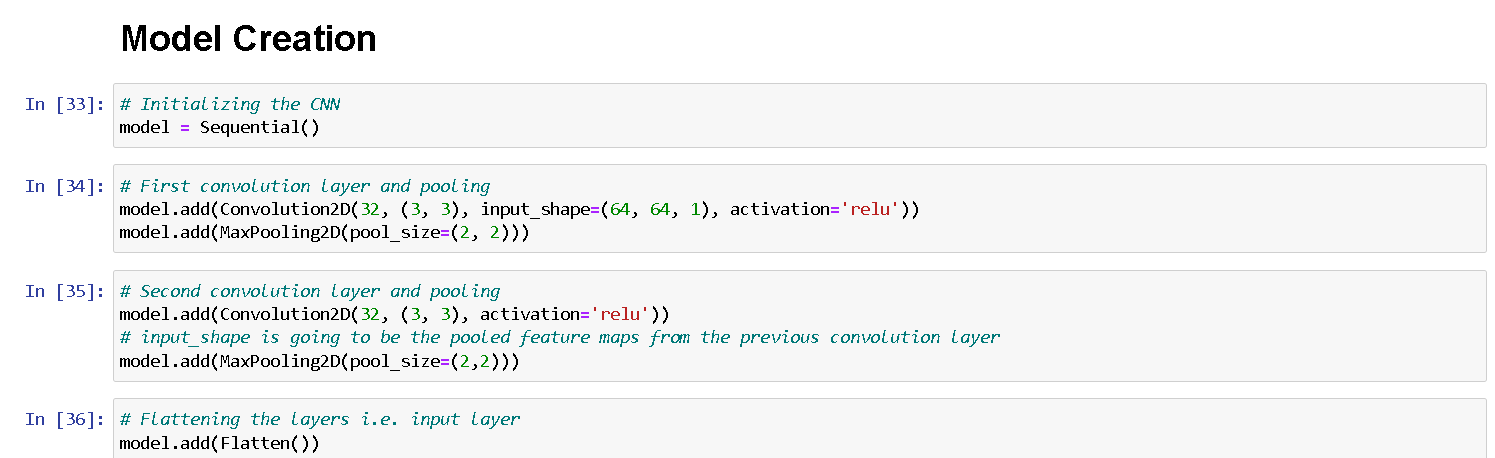


# Model Building

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



# Adding CNN Layers

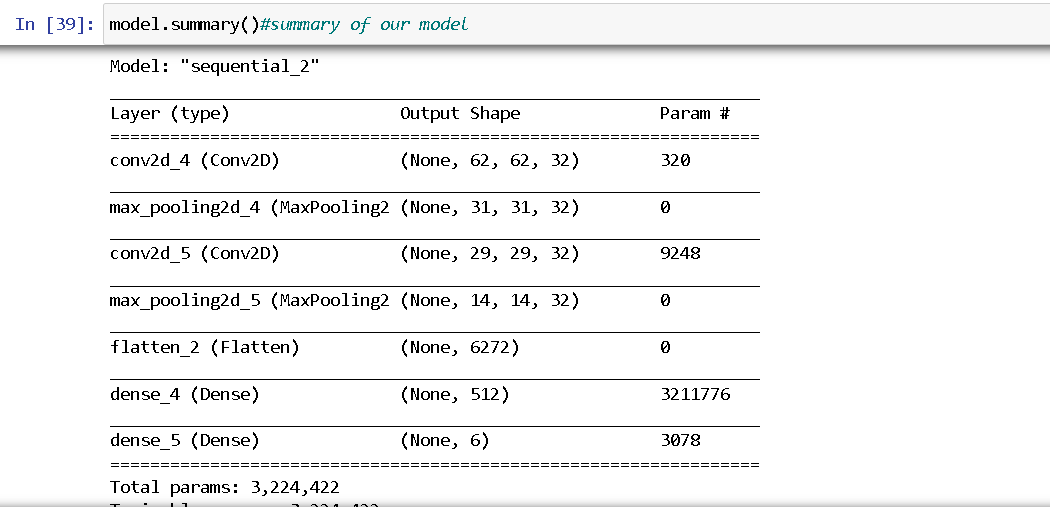


**Adding Dense Layers**

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



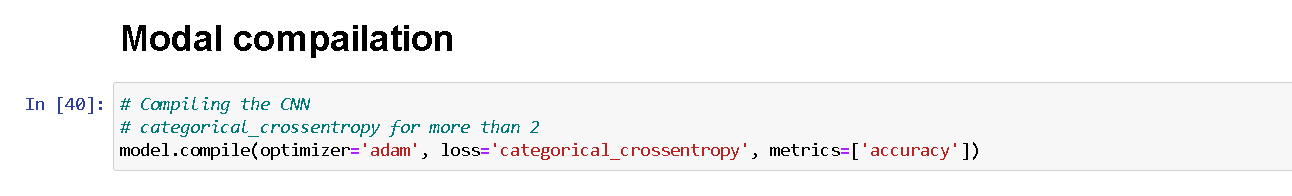
Understanding the model is very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.



# 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



# Train The Model

Train the model with our image dataset.

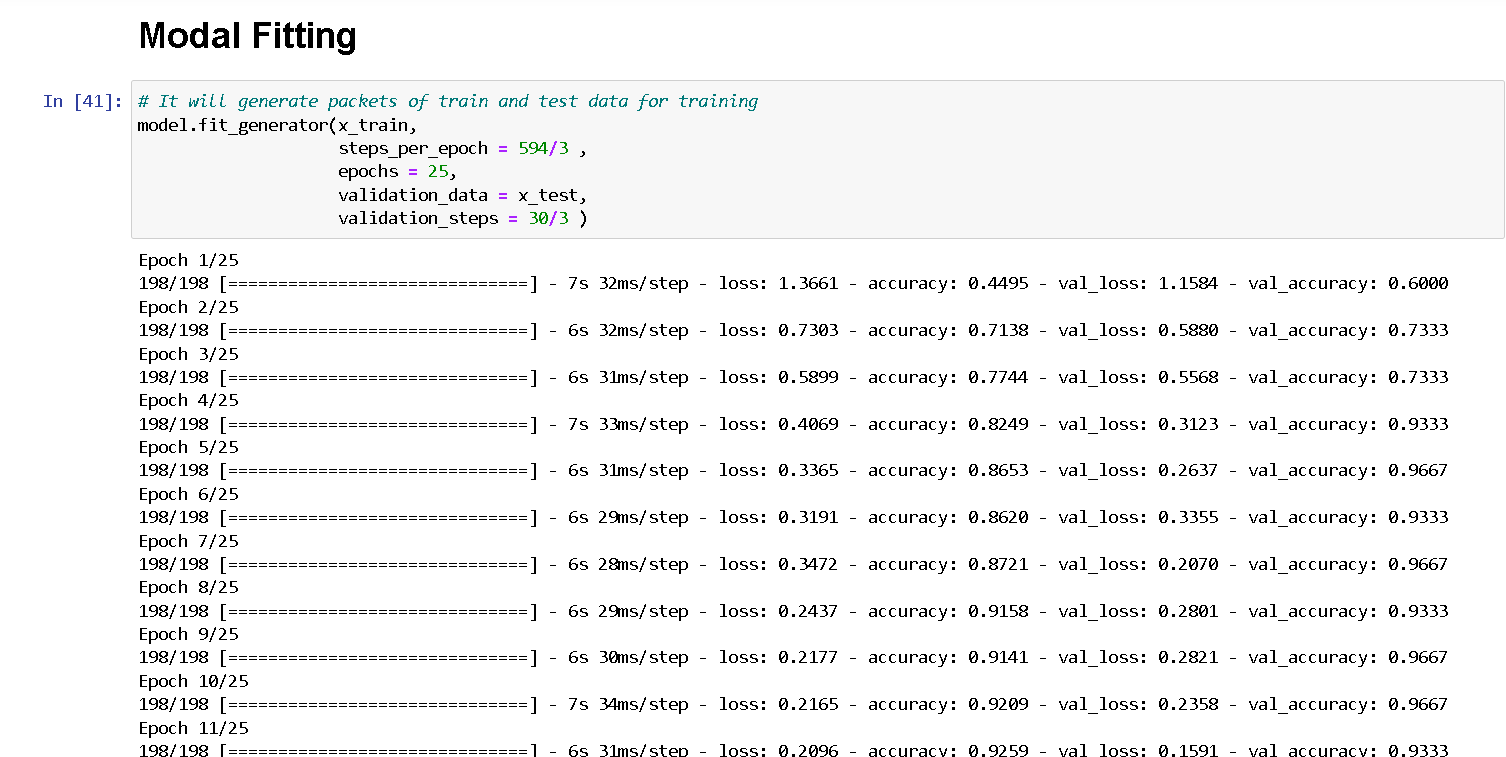
**fit\_generator** functions used to train a deep learning neural network

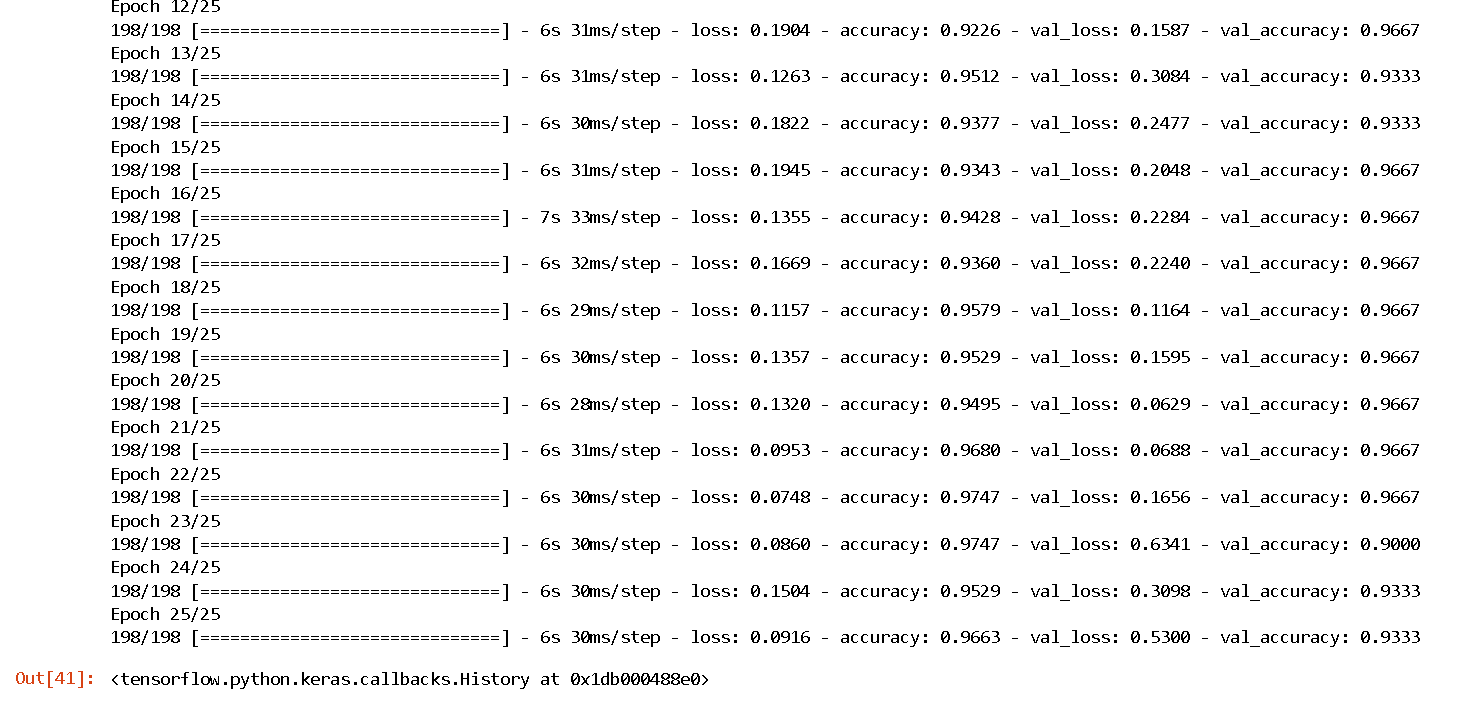
# Arguments:

* 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:

1. an inputs and targets list
2. a generator
3. 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.



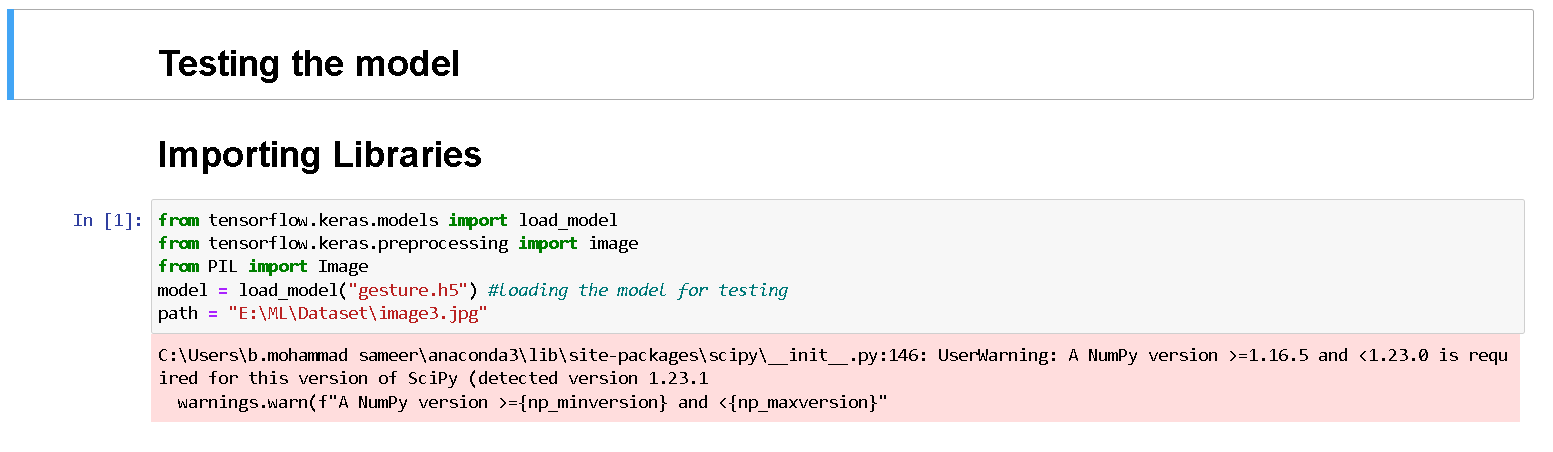


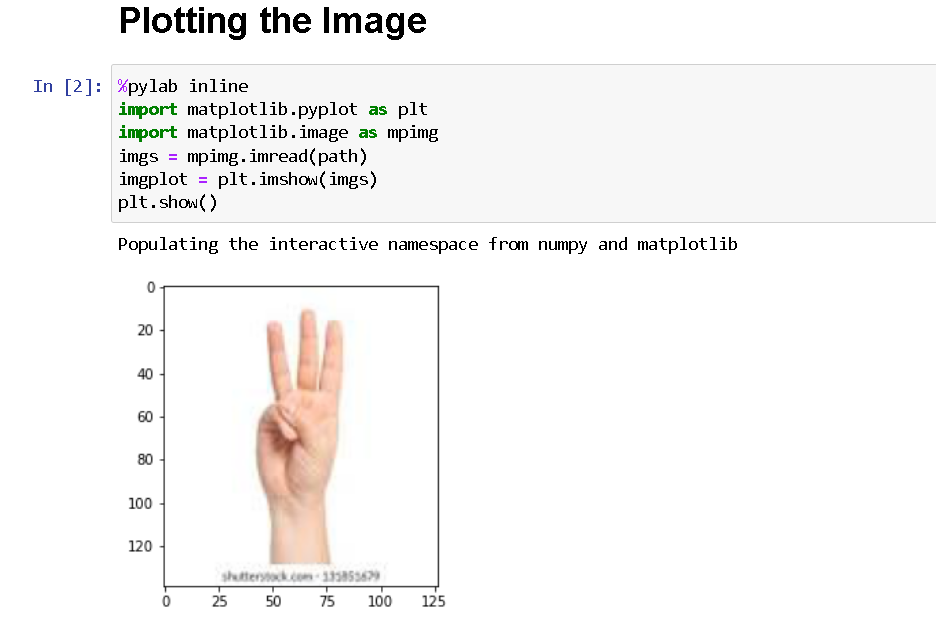


# Test The Model

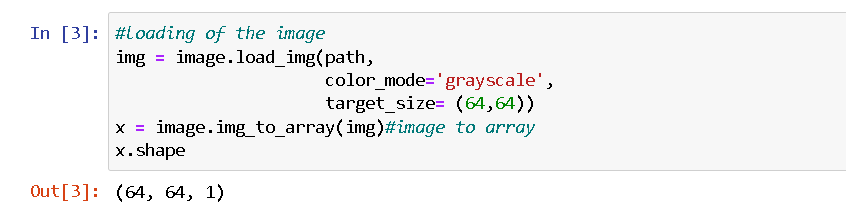
Evaluation is a process during development of the model to check whether the model is best fit for the given problem and corresponding data.

Load the saved model using load\_model

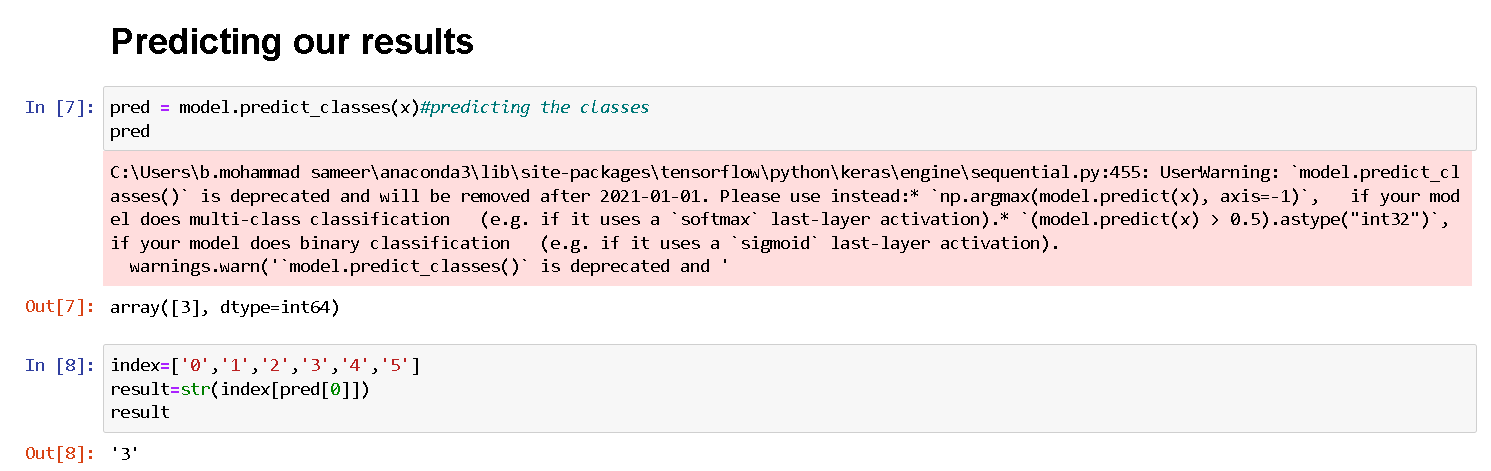




Taking an image as input and checking the results



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



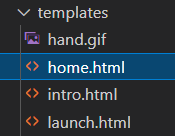
The predicted class index name will be printed here.

# 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.

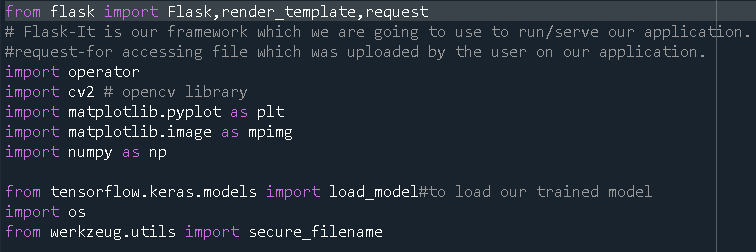
# Create HTML Pages

* 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.

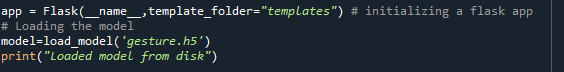


# Build Python Code

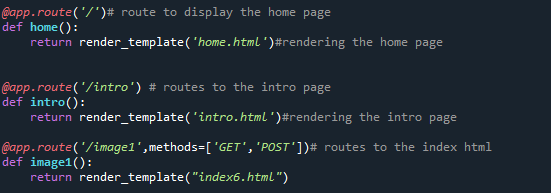
* Build flask file ‘app.py’ which is a web framework written in python for server-side scripting.
* App starts running when “ name ” constructor is called in main.
* render\_template is used to return html file.
* “GET” method is used to take input from the user.
* “POST” method is used to display the output to the user.
* Importing Libraries



* Creating our flask application and loading our model



# Routing to the html Page



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



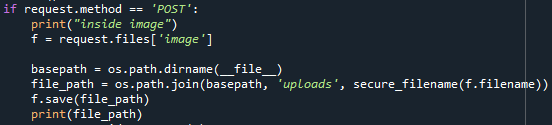
And the predict route is used for prediction and it contains all the codes which are used for predicting our results.

Firstly, inside launch function we are having the following things:

* Getting our input and storing it
* Grab the frames from the web cam.
* Creating ROI
* Predicting our results
* Showcase the results with the help of opencv
* Finally run the application

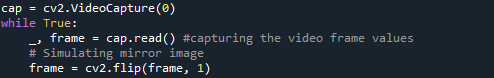
# Getting our input and storing it

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 os function we will be storing the image in the uploads folder in our local system.



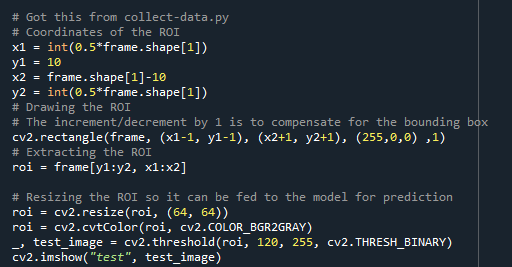
# Grab the frames from the web cam

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 predicting our results.



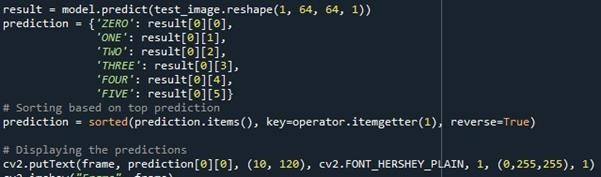
# Creating ROI

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.

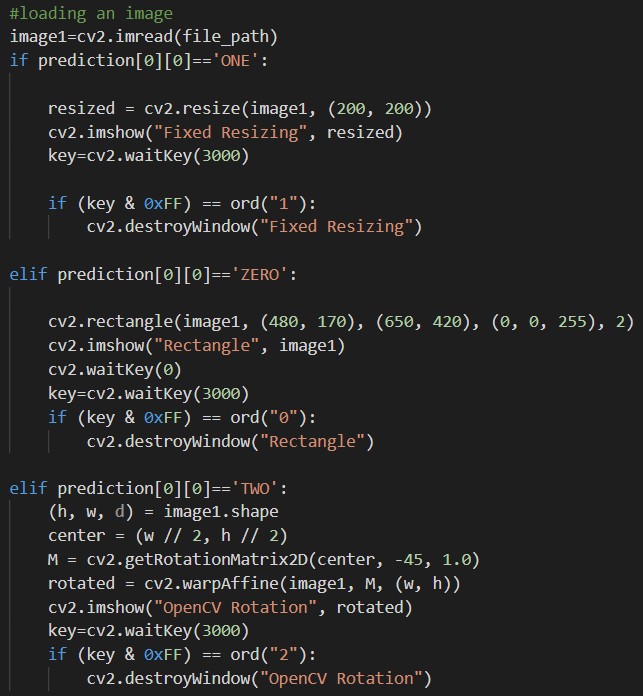


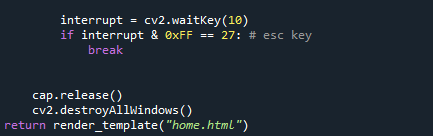
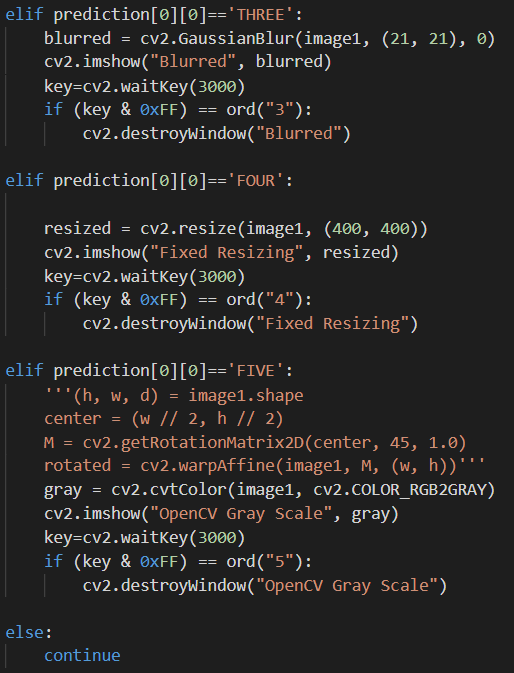
# Predicting our results

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.



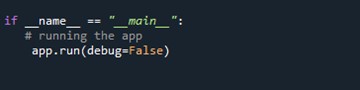
Finally according to the result predicted with our model we will be performing certain operations like resize, blur , rotate etc.





# Run The Application

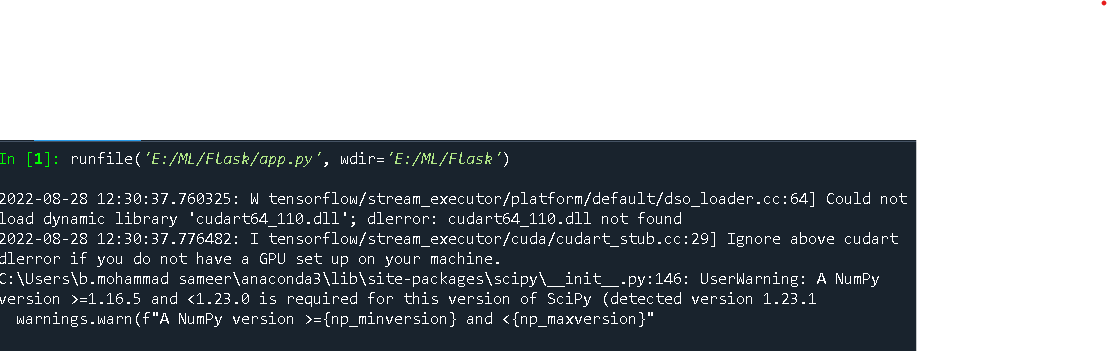
At last, we will run our flask application



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



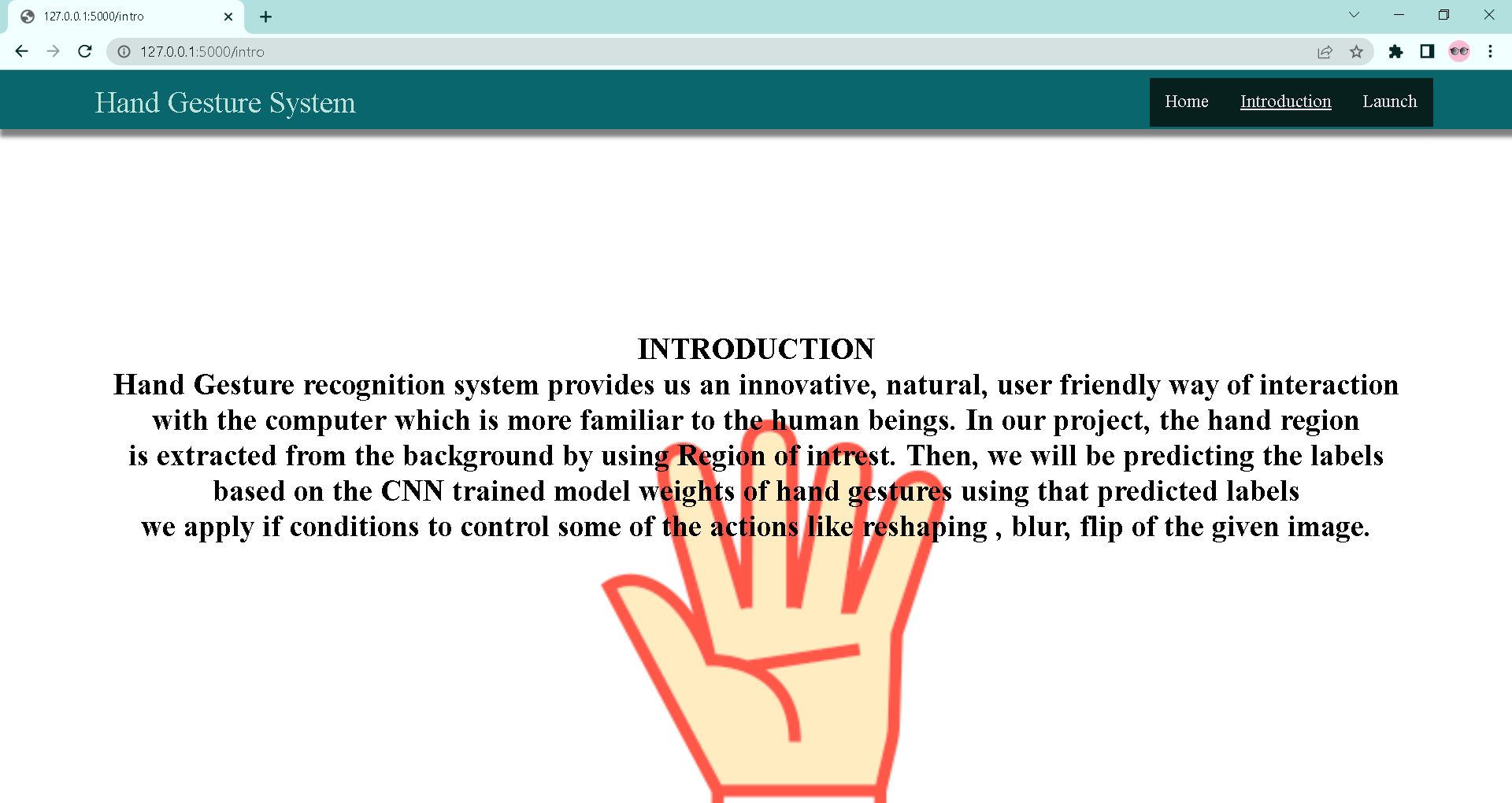
Then it will run on localhost:5000

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

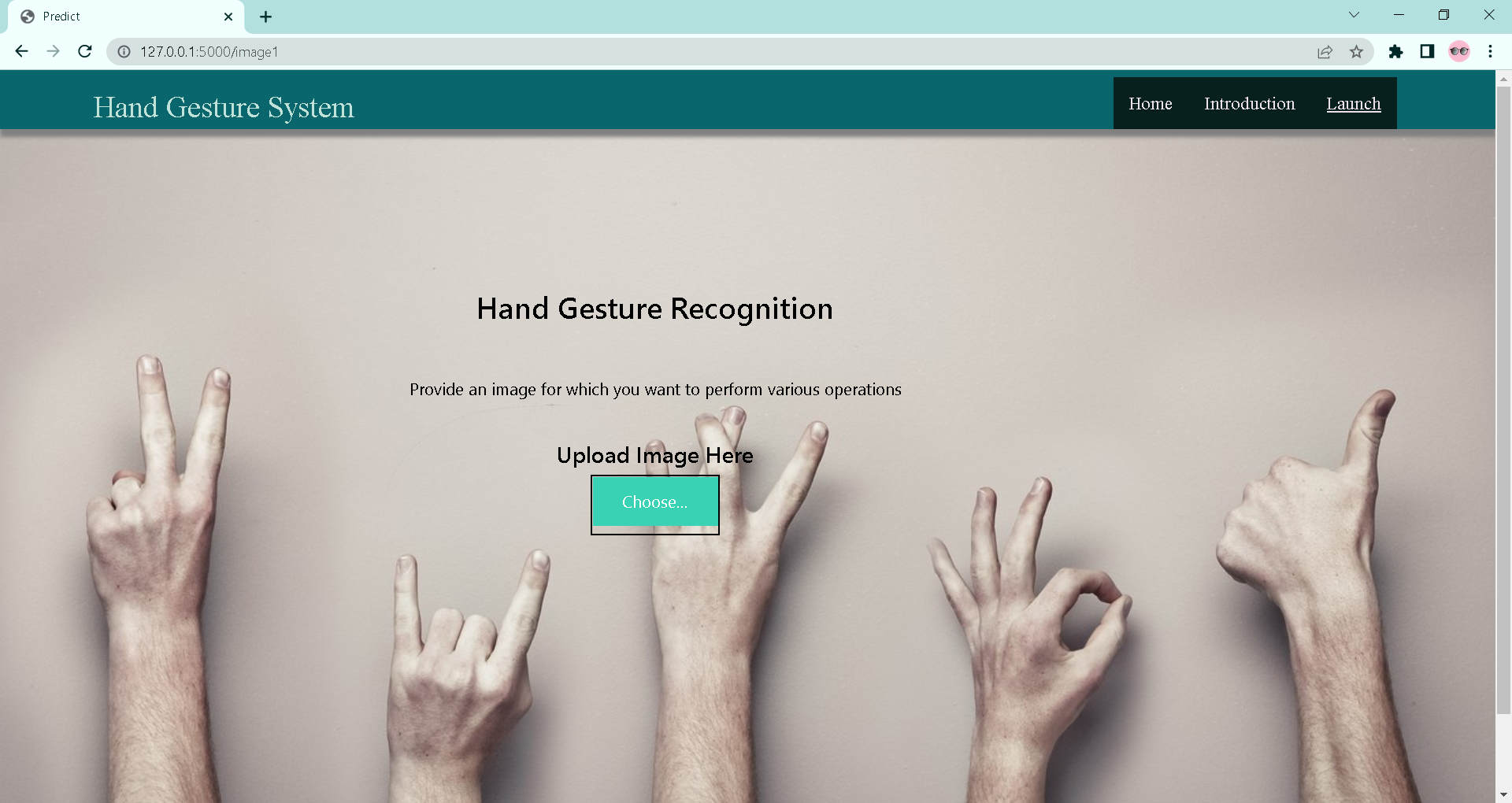
Let’s see how our home.html page looks like:



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



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



**Project Made by GroupA14 :**

**B. Mohammed Sameer - 19481A0511**

**K. AKHILANDESHWARI - 19481A502**

**C. JYOTHIKA - 19481A0540**

**G. YUMESH GUPTA - 19481A0561**