A GESTURE-BASED TOOL FOR STERILE BROWSING OF RADIOLOGY IMAGES USING CNN AND OPEN CV

BATCH ID: PNT2022TMID12828

19L205- ANNAPOORNI D

19L206- ARIES R

19L225- MOHAMMED MASTHAN S

19L262- VEERAKARTHIK R D

1. INTRODUCTION:

1.1 PROJECT OVERVIEW

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.

1.2 PROJECT PURPOSE:

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

1.3 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.

2. LITERATURE SURVEY:

2.1. EXISTING PROBLEM

Computer information technology is increasingly penetrating into the hospital domain. 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. In the existing methods voice control is used to provide sterility, but the noise level in the operating room deems it problematic.

2.2 REFERENCES:

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.

To operate under sterilization condition in operating room help to reduce infection rate in patient. Therefore, surgeons cannot control computer by using mouse, or keyboard. So, the author proposed a method to controlling images in operating room by using a touchless sensor. This method consists of four steps. First, hand motion is sensed by a touchless sensor as Leap motion. Second, hand position is detected. Third, hand gesture is recognized by using movement direction. Finally, image is shown and

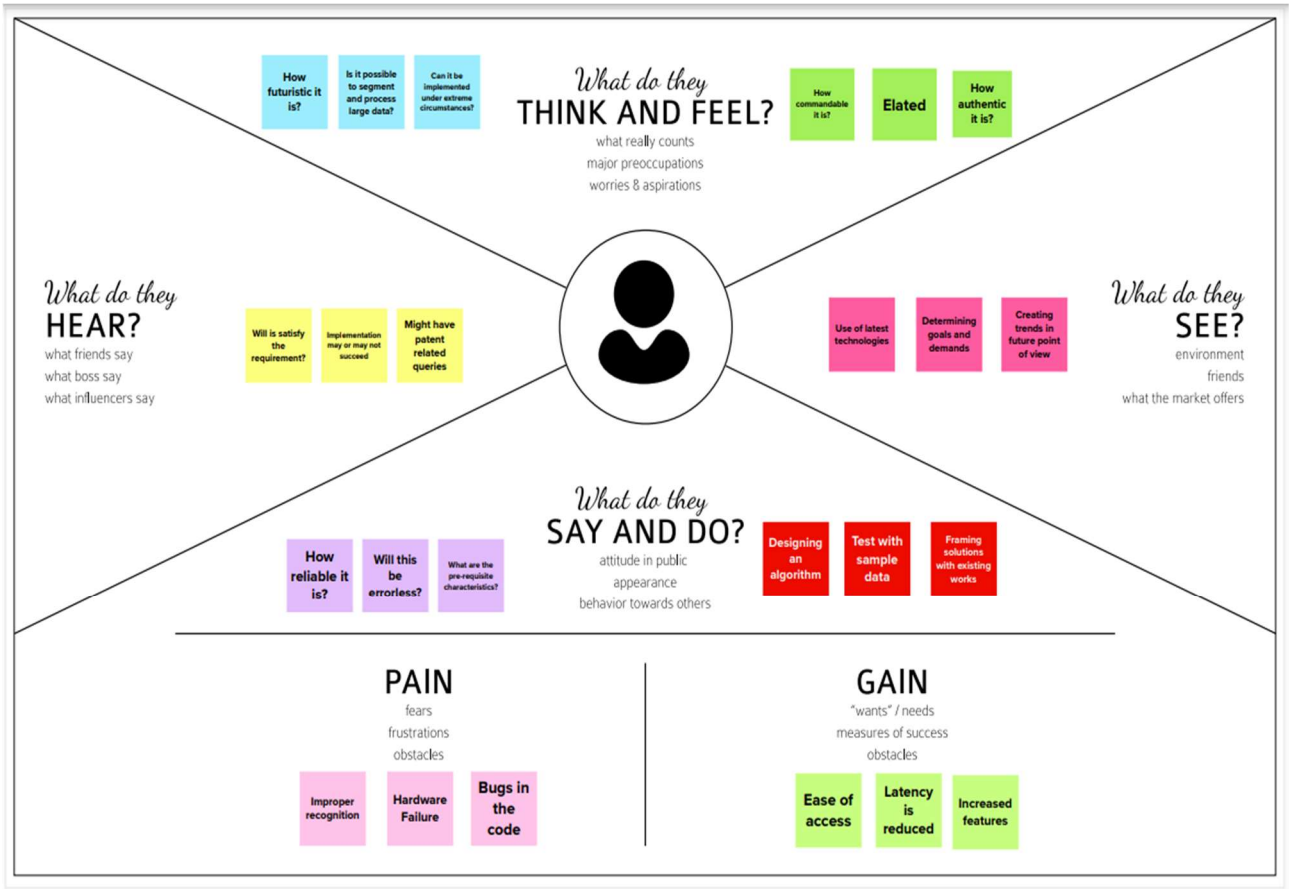
processed relating with hand gesture command. Accuracy rate of commands interpretation is 87.67%.

2.3. PROBLEM STATEMENT DEFINITION:

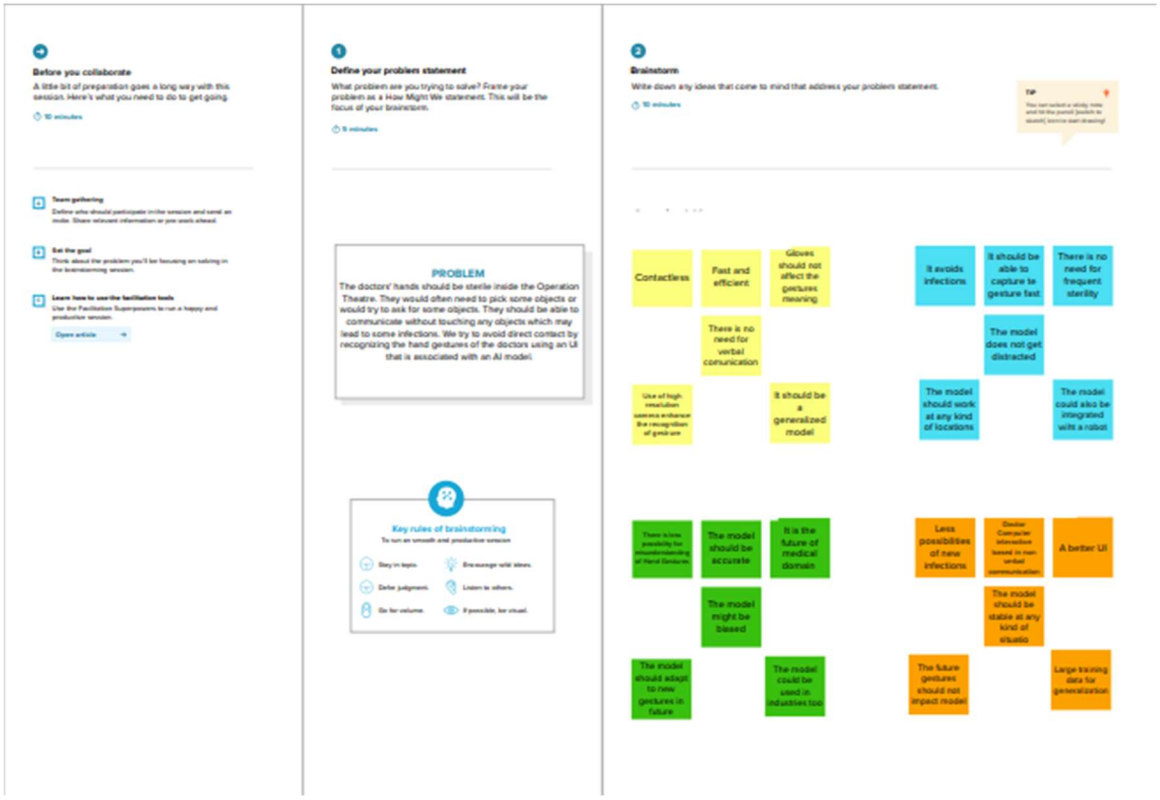
Hand gesture recognition was imported for human computer interaction. Hand gesture recognition for radiology images used to detect gesture using webcam and to perform exacting action for the gesture. The gesture cannot be recognized in long distances. The image resolution is low due to quality of webcam. The hand gesture is restricted to lighting backgrounds and image noise. The gesture should be recognized as pattern rather than images. The gesture interaction should be faster and more convenient. The gesture movements of some fingers are limited. The gesture should be detected and classified properly to produce exact action of the gesture given as input. The difference between intentional and unintentional gesture should be identified.

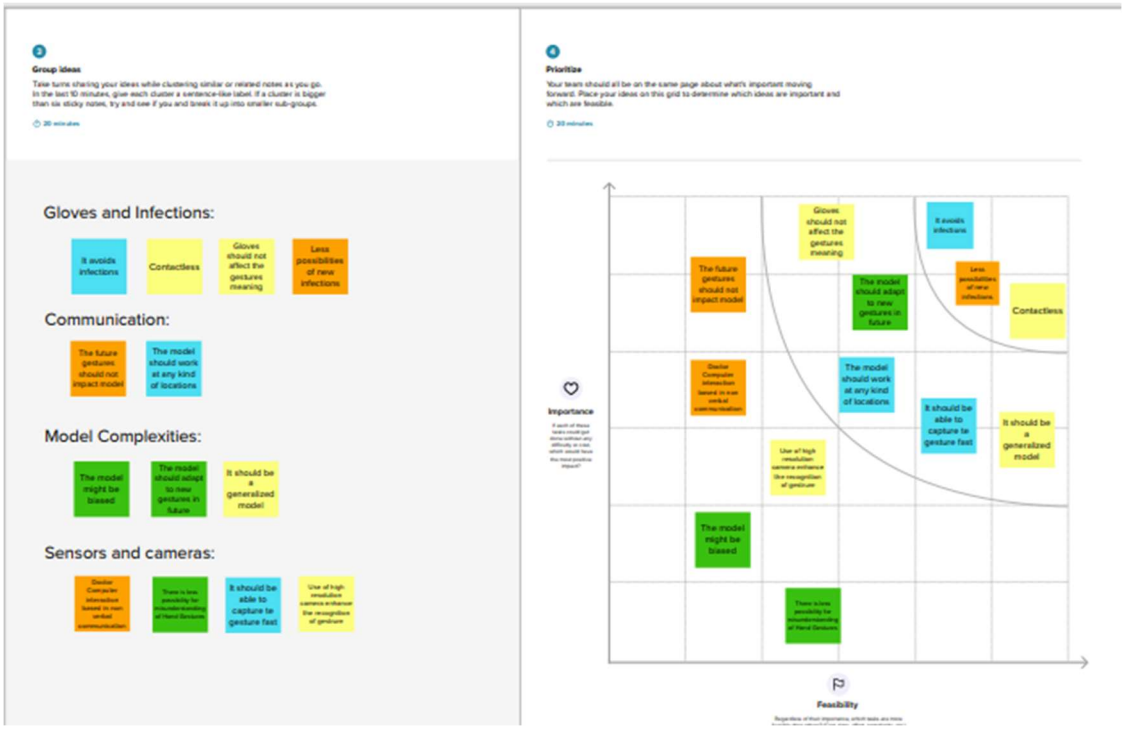
3. IDEATION AND PROPOSED SOLUTION:

3.1. EMPATHY MAP CANVAS:



3.2. IDEATION & BRAINSTORMING:

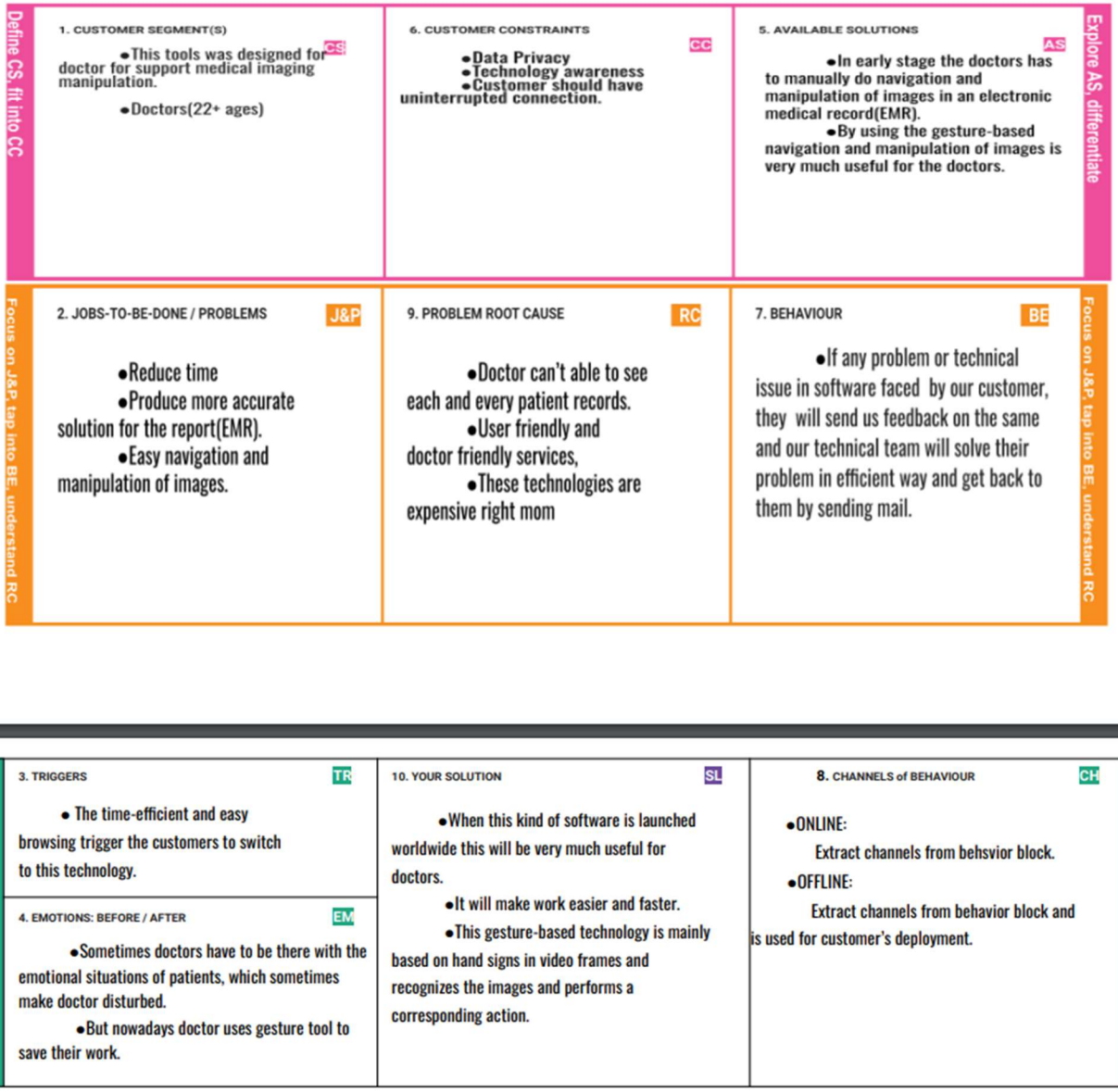




3.3 PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	A major challenge involved is to provide Doctors with efficient, intuitive, accurate and safe means of interaction without affecting the quality of their work. However,the use of computer keyboards and mouse by doctors in intensive care unit(ICU) is a common mean for spreading infections.We suggest the use of hand gestures in medical field as an alternative to the existing interface techniques by offering maximum level of sterility.
2.	Idea / Solution description	Doctor can make use of hand gesture to move or control the images in order to maintain sterility.
3.	Novelty / Uniqueness	In this method unlike other methods of non-verbal communication, gesture do not cause loss of concentration in operation theatre. It performs better in detecting pattem in images.
4.	Social Impact / Customer Satisfaction	It co-operates social responsibility by providing better solution to patient's health and it also helps professionals to have the control over images without having direct contact with system which avoids the harmful rays and infections.
5.	Business Model (Revenue Model)	Cost efficient to deploy this Software for health care department as well as in hospitals and can collaborate with government for health awareness camps.
6.	Scalability of the Solution	Better execution in accurate results, sensitivity, system architecture design and transperancy and flexibility of the software.

3.4 PROBLEM SOLUTION FIT:



4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Launching the model	Launch the trained CNN model from the cloud
FR-2	Capturing the images	After capturing the images in camera we have to upload the images in the system
FR-3	Performing gestures	After classifying, identify the correct image by the gesture and it should perform the operation
FR-4	Model rendering	After capturing the image the algorithm will start its processing task
FR-5	Sterile browsing	The sterile browsing can be performed after identifying the gestures
FR-6	Visibility of images	After completing all the processes,a user can be able to see the images

4.2 NON-FUNCTIONAL REQUIREMENTS:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	This system helps to have the control over images without having direct contact with system which avoids the harmful rays and is ease of use
NFR-2	**Security**	This system is protected and only authorized users can access it
NFR-3	**Reliability**	After installing the application,the system will predict the gesture and performs sterile browsing
NFR-4		The system responds to a user in seconds and the hardware and software works well
	Performanc	It is accessible by authorised user from anywhere at any time whenever there is an emergency
e NFR-5	**Availability**	This system allows more number of users at a time and there is no loss can be identified

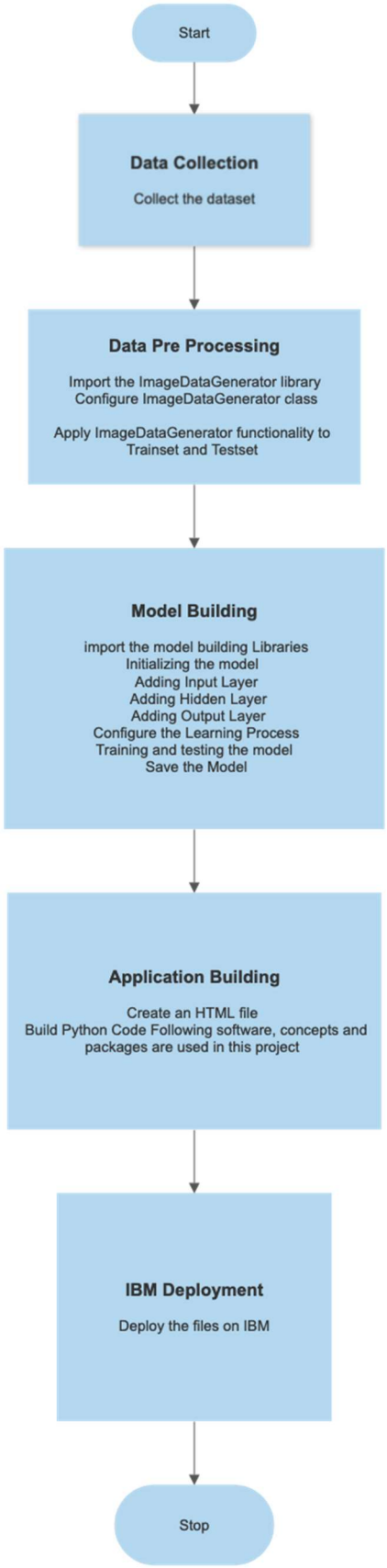
5. PROJECT DESIGN:

5.1 DATA FLOW DIAGRAMS:

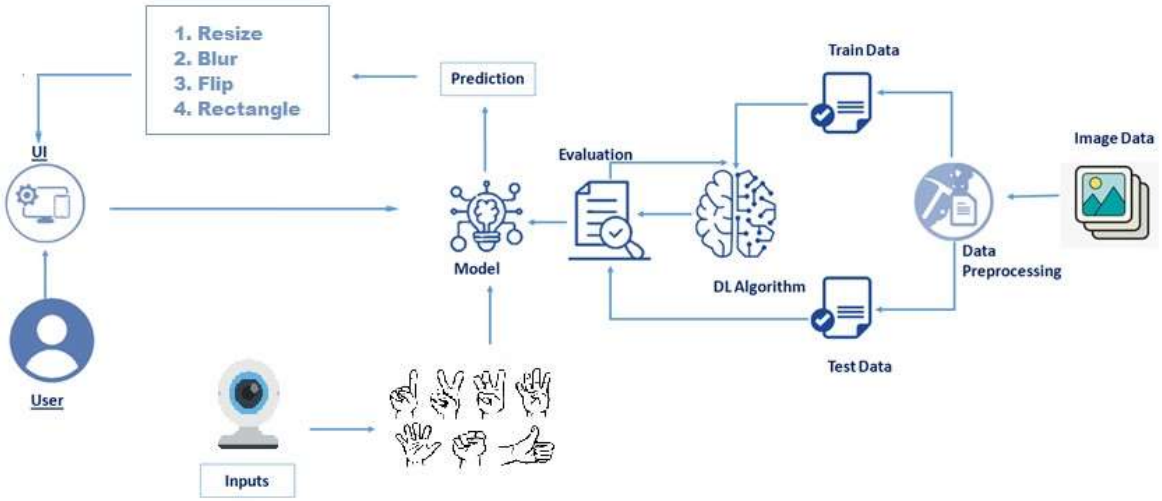
- User interacts with the UI (User Interface) to upload the image as input.
- Depending on the different gesture inputs different operations are applied to the input image.
- Once model analyses the gesture, the prediction with operation applied on image is showcased on the

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

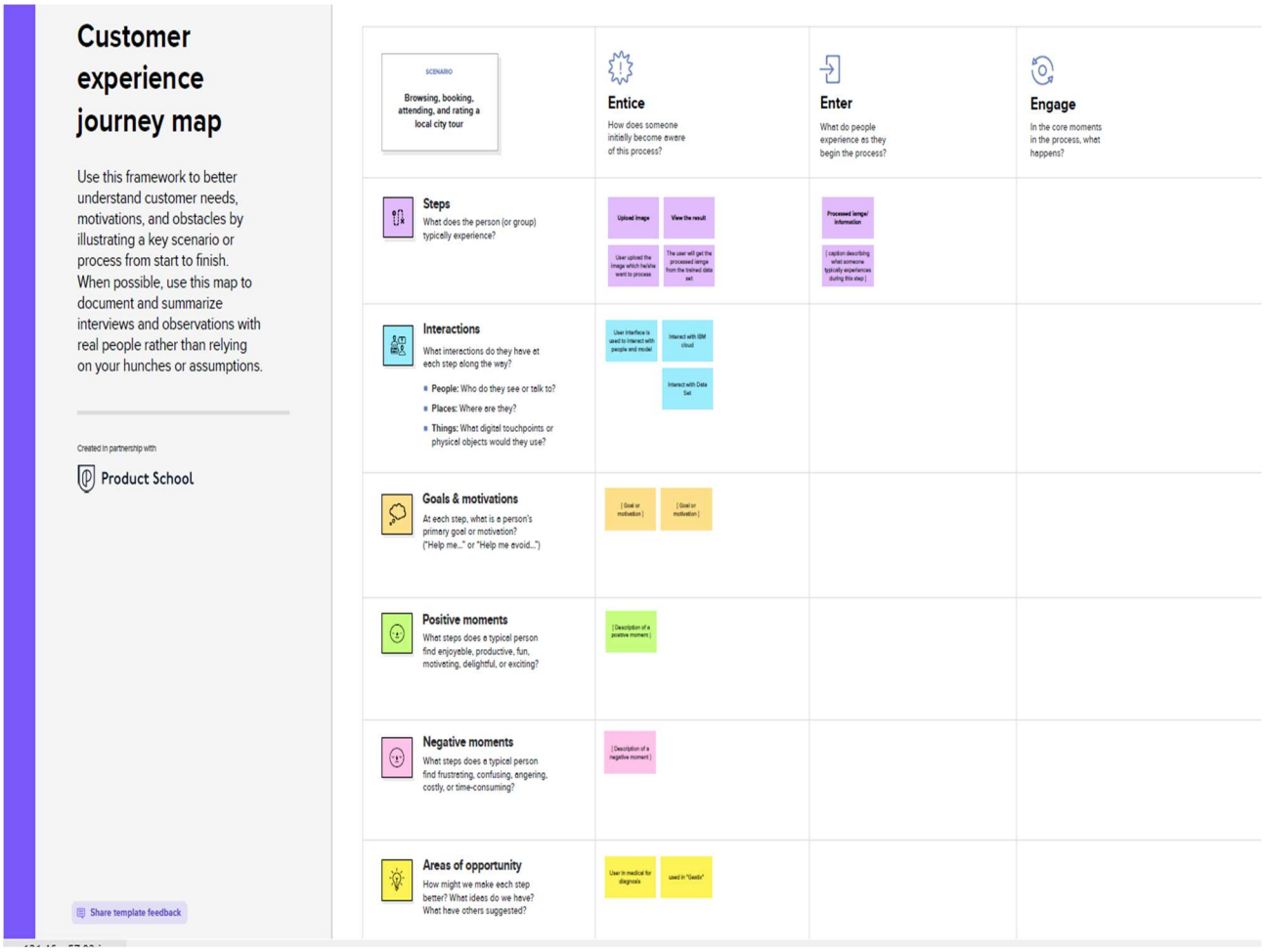
- Data Collection.
  - Collect the dataset or Create the dataset
- Data Pre processing
  - Import the ImageDataGenerator library
  - Configure ImageDataGenerator class
  - Apply ImageDataGenerator functionality to Trainset and Testset
- Model Building
  - Import the model building Libraries
    - Initializing 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”


Start
Data Collection
Collect 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
IBM Deployment
Deploy the files on IBM
Stop

5.2 SOLUTION & TECHNICAL ARCHITECTURE:



5.3 USER STORIES:



6. PROJECT PLANNING & SCHEDULING:

6.1 SPRINT PLANNING & ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	To download the data and apply image preprocessing on the data. Import the library, ImageDataGenerator, and configure that class	2	High	Aries R Annapoorni D Mohammed Masthan S Veerakarthik R D
Sprint-2	Model Building	USN-2	Importing the necessary Model Building Libraries and initialize the model	1	High	Aries R Annapoorni D Mohammed Masthan S Veerakarthik R D
Sprint-3	Application Building	USN-3	Create HTML Pages and build a python code and then run the application	2	Medium	Aries R Annapoorni D Mohammed Masthan S Veerakarthik R D

6.2. SPRINT DELIVERY SCHEDULE:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	15	06 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	10	14 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	5	20 Nov 2022

DEEP LEARNING CONCEPTS

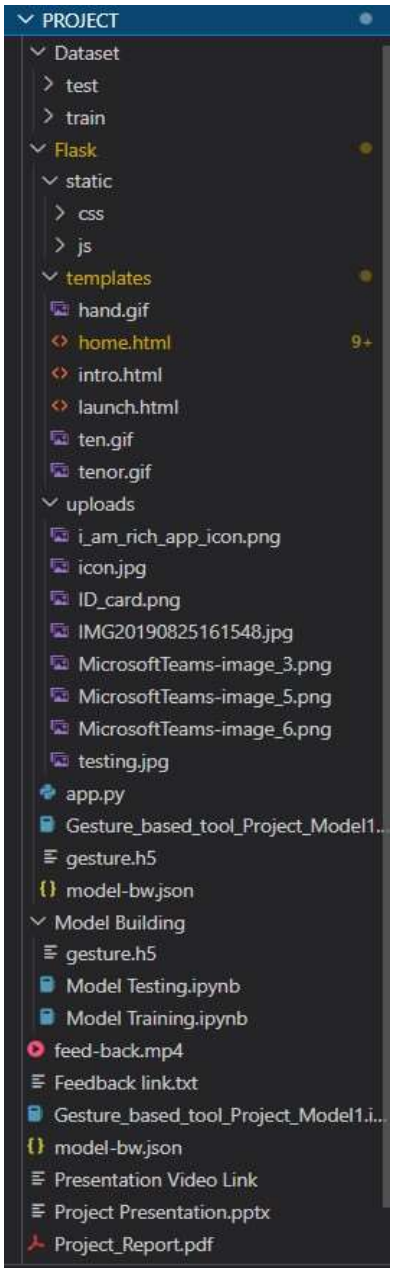
CNN: A convolutional neural network is a class of deep neural networks, most commonly applied to analysing visual imagery.

Open-CV: 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.

PROJECT STRUCTURE

- 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.



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.
```

```
#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 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()
```

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 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 prediction purposes. Keras provides a simple method, summary to get the full information 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
```

CONFIGURING 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 function is 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.

```
# 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
[illegible]
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>
```

```
# 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)
```

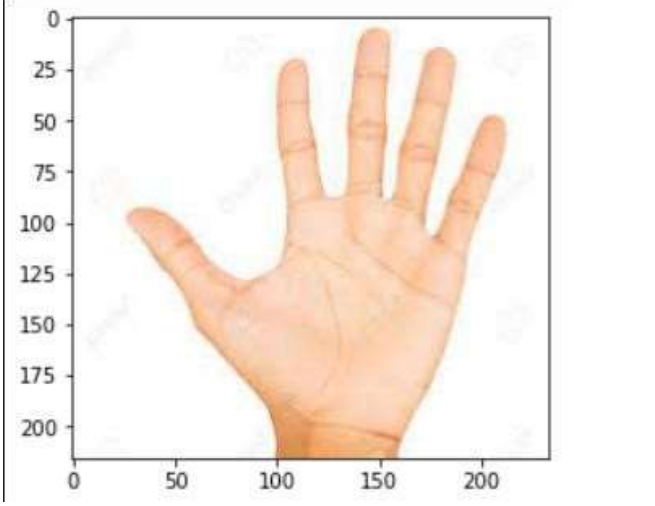
TESTING THE MODEL

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"
```

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'
```

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)
```

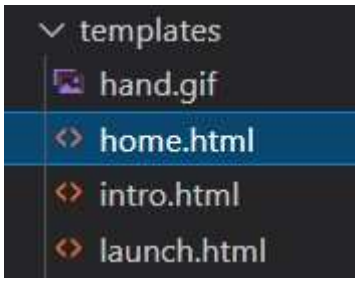
```
[array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([1],
dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([2], dtype=int64),
array([2], dtype=int64),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

CREATING 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.



BUILDING 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
```

- Creating our flask application and loading our model

```
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")
```

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

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

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 open-cv
- 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.

```
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 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)
```

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.

```
# 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

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.

```
#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")
```

```
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
```

```
        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 application

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

Run The app in local browser

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

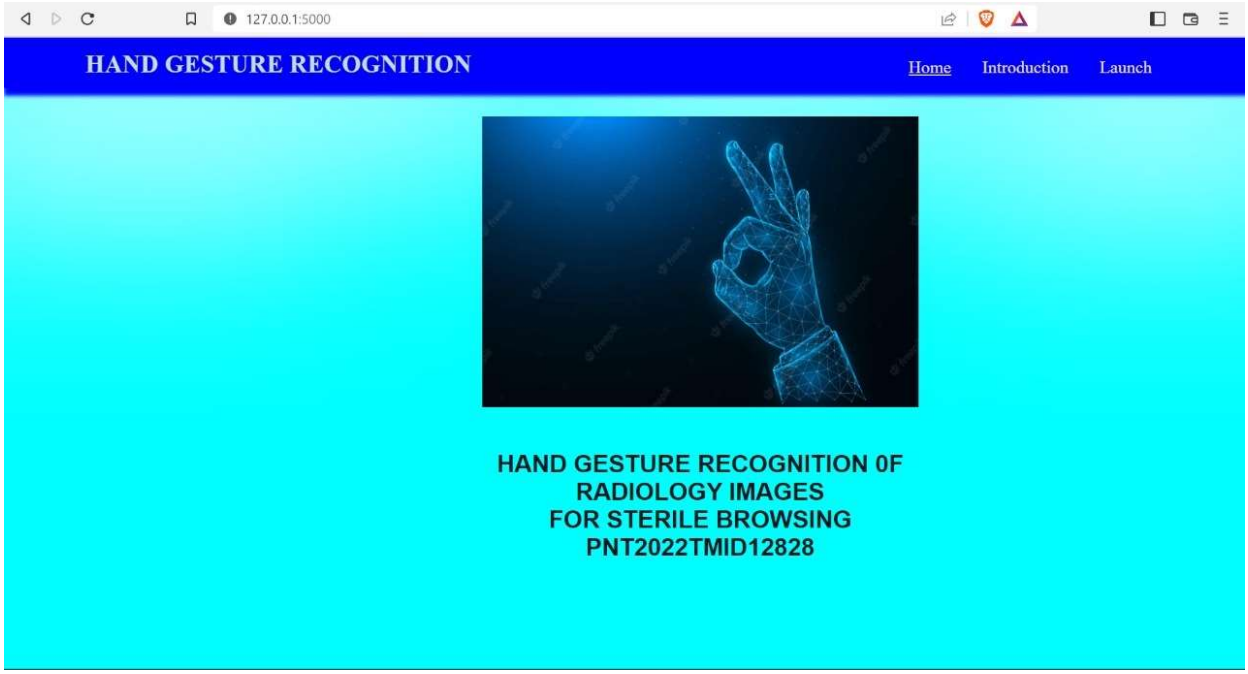
Navigate to the localhost where you can view your web page

Then it will run on localhost:5000

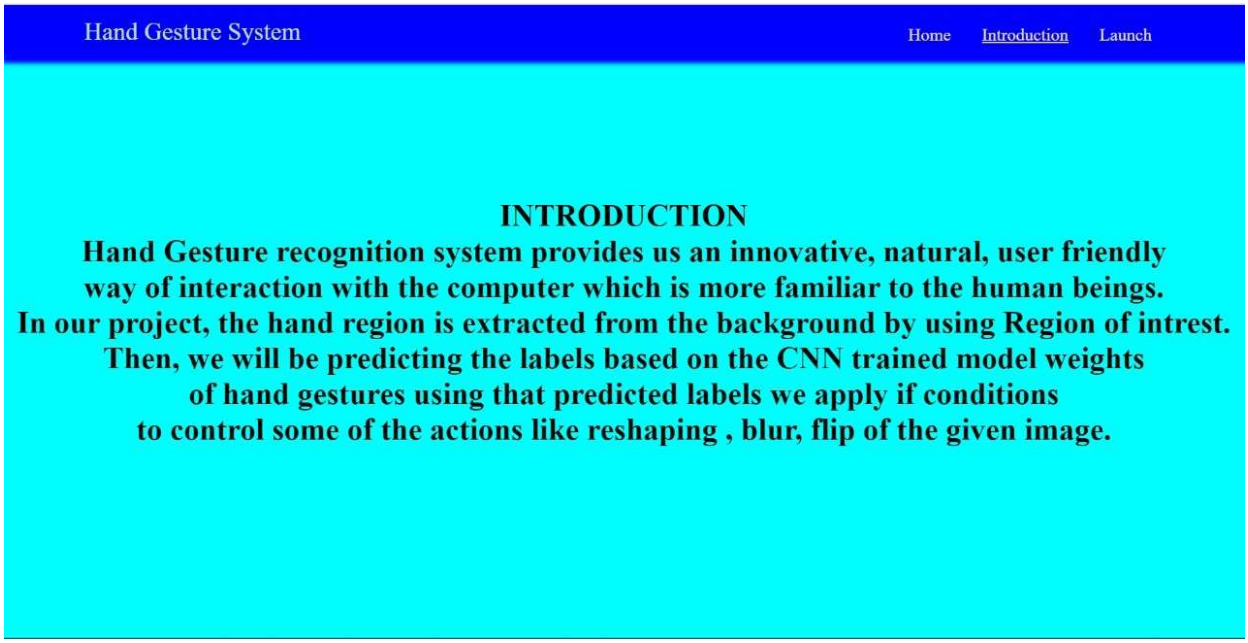
```
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

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

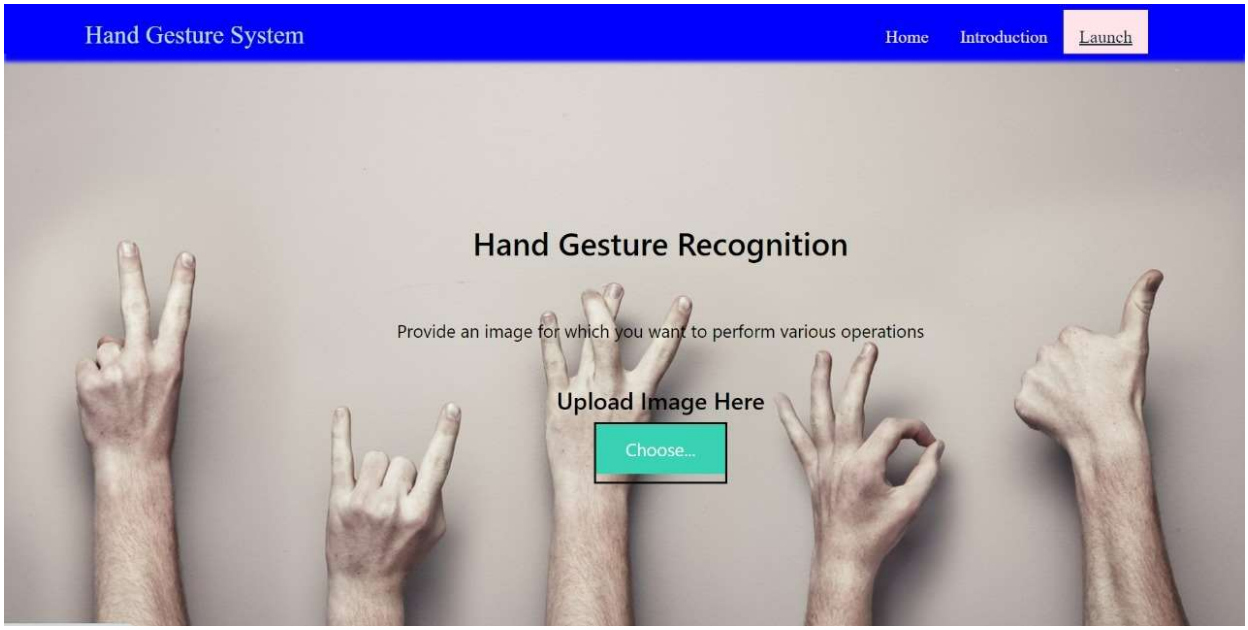
Let’s see how our home.html page looks like:

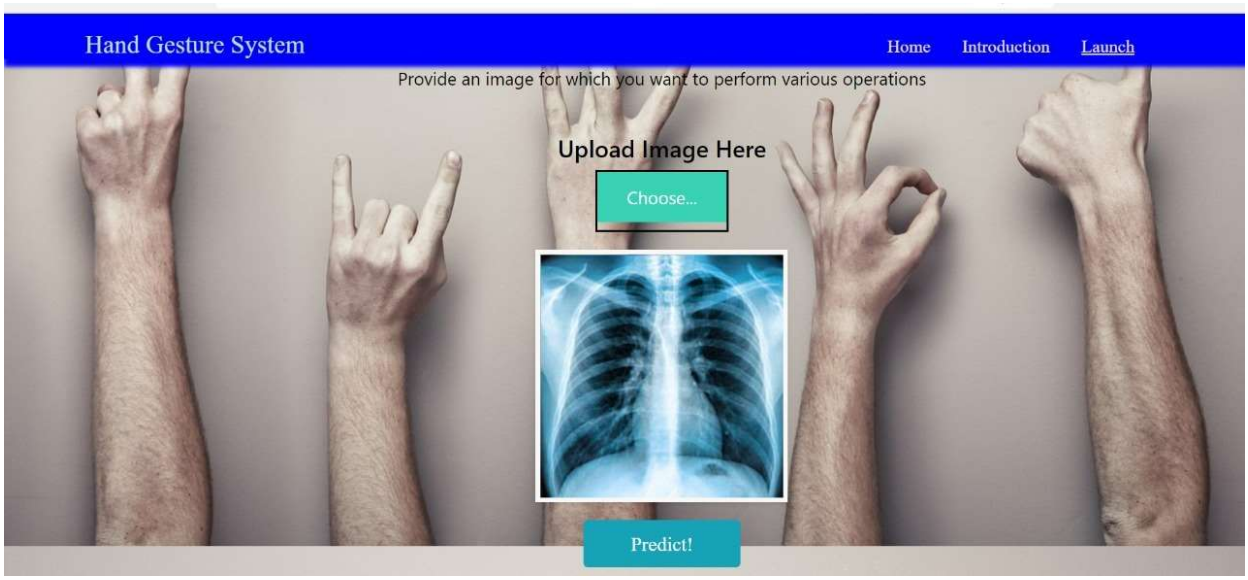


When “Introduction” button is clicked, localhost redirects to “intro.html”

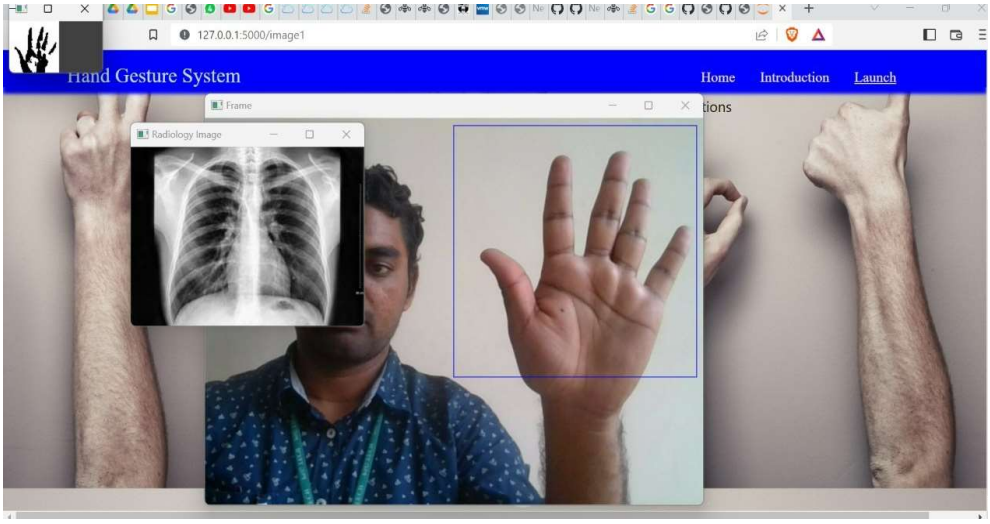


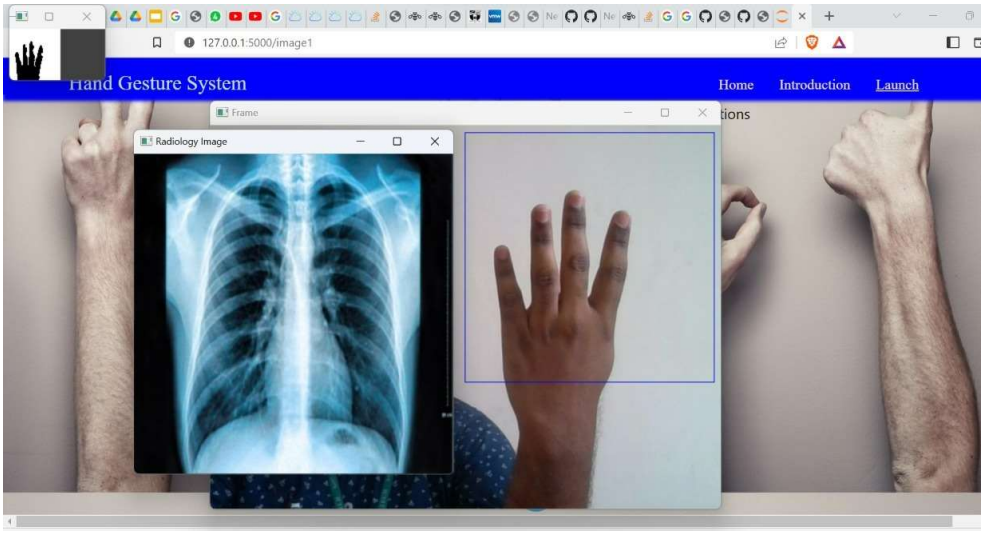
Upload the image and click on Predict button to view the result

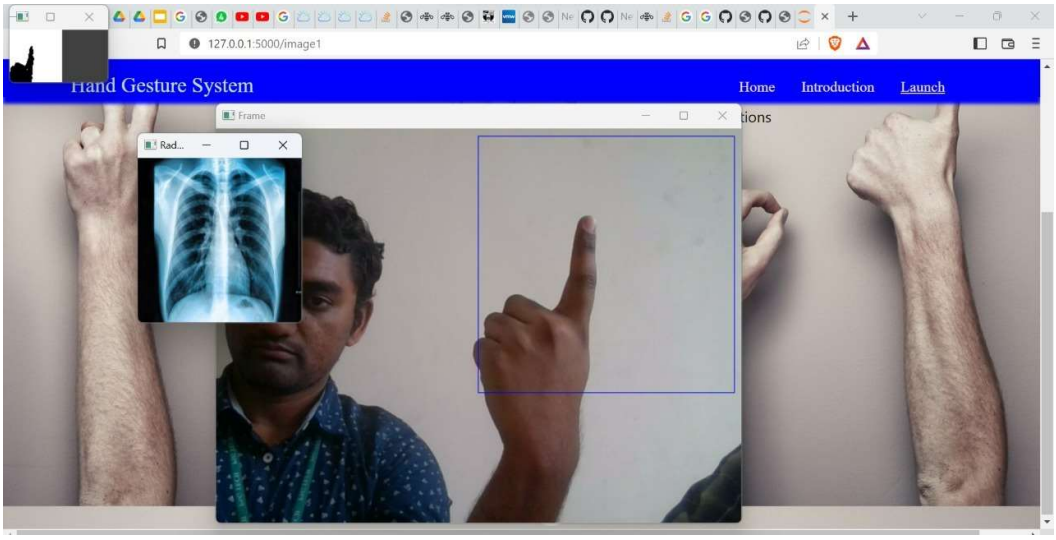



Hand Gesture System
Home
Introduction
Launch
Provide an image for which you want to perform various operations
Upload Image Here
Choose...
Predict!

RESULTS


127.0.0.1:5000/image1
Hand Gesture System
Home
Introduction
Launch
Frame
Radiology Image


127.0.0.1:5000/image1
Hand Gesture System
Home
Introduction
Launch
Frame
Radiology Image



10. ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

- Major advantage 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 expensive as it requires cameras and other expensive devices to capture images and process it.

11. 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 mouse 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 can be used regardless of the users 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 like it can be used by presenters, by teachers for show images in the classroom, etc.

12. FUTURE SCOPE :

The tool can be made quicker by increasing the recognition speed. More number of gestures can be added thereby increasing this tool’s functionality and useability for different purposes. Tracking of both hands can be added to increase the set of commands. Voice commands can also be added to further increase the functionality.