NAALAIYA THIRAN PROJECT -2022

Team ID	**PNT2022TMID52777**
Name	**A Gesture based tool for sterile browsing of Radiology images**

TABLEOFCONTENTS

LISTOFFIGURES

LISTOFTABLES

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

In this project we use gestures to browse images obtained during radiology. Gestures refer to non verbal form of communication made using hands. 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 principal method of human—computer interaction. However, the use of computer keyboards and mice 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 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 gray scale etc.

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. The Project Objectives are:

- To know fundamental concepts and techniques of Convolutional Neural Network (CNN).
- To gain a broad understanding of image data.
- To know how to pre-process/clean the data using different data pre- processing techniques.
- To know how to build a web application using Flask framework.

CHAPTER 2

LITERATURE SURVEY

This chapter gives an overview of research carried out related to the project work on “Gesture based tool for sterile browsing of radiology images”.

2.1 EXISTING SOLUTION

The existing solution that is illustrated in the paper is,

- The existing solution for gesture-based tools was to track Navigation and other gestures and translate to commands based on their temporal trajectories, through video capture.
- Computer vision algorithms were developed to extract intention and attention cues from the surgeon's behavior and combine them with sensory data from a commodity depth camera.

The following are inferred from the paper.

- A video-based hand gesture capture and recognition system used to manipulate MRI within a graphical user interface.
- A hand gesture vocabulary of commands are selected such as 1-Resize, 2-Flip, 3-Rotate, and 4-Rectangle.
- Each gesture is cognitively associated with the notion or command that is meant to represent it.
- Using this, doctors can analyze the image by having non-verbal communication.

2.2 REFERENCES

The Reference papers referred for the Project titled “Gesture based tool for Sterile browsing of radiology images are listed below.

- 2014 IEEE International Conference on Automation Science and Engineering (CASE) Taipei, Taiwan, August 18-22, 2014.
- 2020 IEEE Open Access journal on Electrical and Computer Engineering Dallas, Richardson, USA, December 18, 2020.
- 2018 IEEE Conference on Biomedical Engineering on “Classification of therapeutic hand poses using convolutional neural networks”, July 2018.

2.3 PROBLEM STATEMENT DEFINITION

Keyboards and pointing devices, such as a mouse, are today's principal methods of human-computer interaction. However, the use of computer keyboards and mice by doctors and nurses in intensive care units (ICUs) is a common method for spreading infections. Hence, as an alternative, the use of hand gestures to existing interface techniques offers the major advantage of sterility. Humans are able to recognize the body and sign language easily. This is possible due to the combination of vision and synaptic interactions that were formed during brain development. In order to replicate this skill in computers using image capture technology and classification techniques and to incorporate this feature in our project.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

The initial process is to choose the topic and team members. The team has been formed and confirmation mail has been received. After confirming the team members topic has been chosen and the topic is ” The Gesture-based tool for Sterile Browsing of Radiology Images”. The next step is registration to IBM cloud and requesting for accessing resourcing. The Rocket app is installed to chat with mentors. The GitHub account is created and collaborated with the project repository and project workspace.

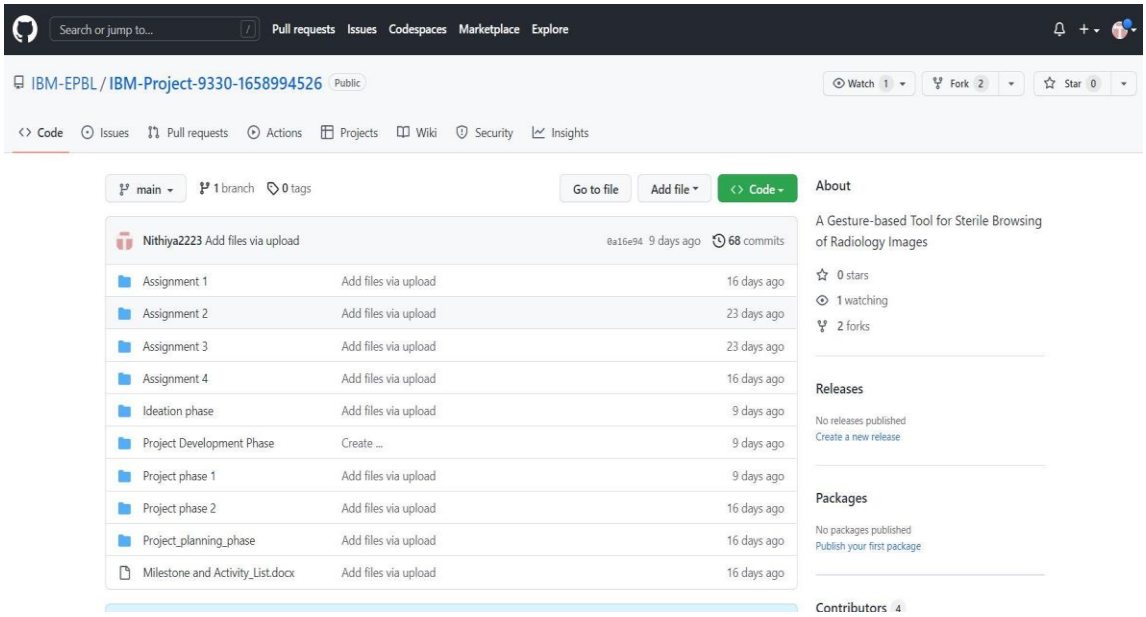


Fig 3.1 GitHub repository of IBM project

3.1 EMPATHY MAP CANVAS

An Empathy map allows one to sum up our learning from engagements with people in the field of design research. The map provides four major areas in which to focus our attenuation on, thus providing an overview of the person’s experience. An Empathy map consist of four

quadrants. The four quadrants reflects four key traits, which the user demonstrated during the observation stage. The four quadrants refer to what the user: Said, did, thought and felt. It's easy to determine what the user said and did. However, determining what they thought and felt should be based on careful observations and analysis as how they behaved and responded to certain activities, suggestions, conversations, etc.

The next step is preparing Empathy Map Canvas to capture the user's Pains & Gains

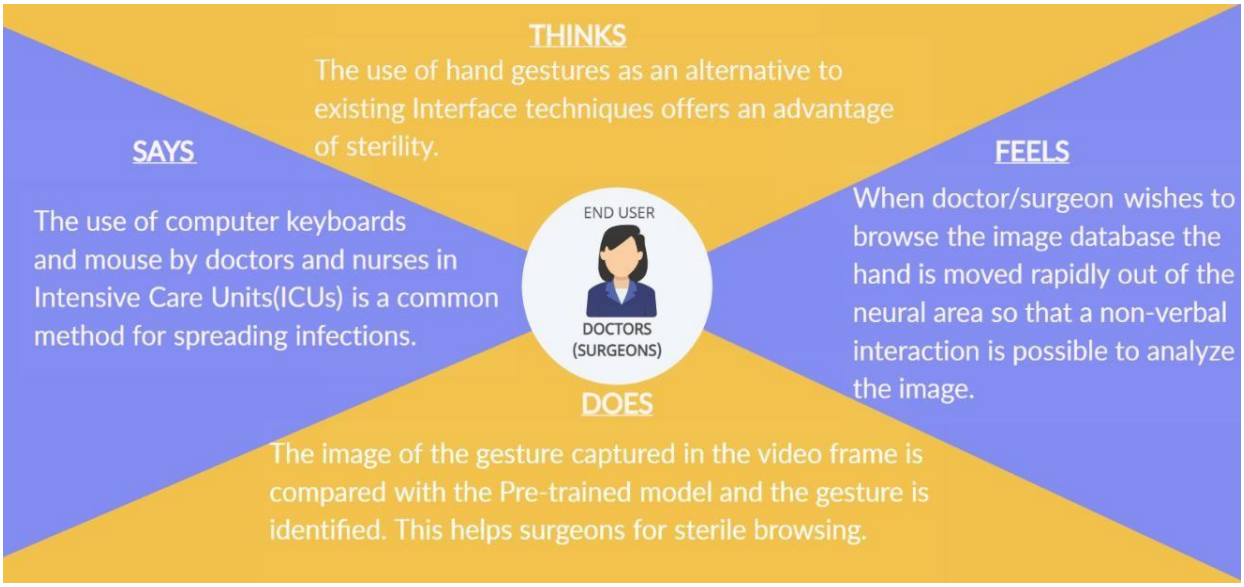


Fig 3.2 Empathy map

3.2 IDEATION AND BRAINSTORMING

In the ideation phase, As a first step the papers related to the topic is surveyed and the literature survey document is prepared. For content we referred to many papers that have content related to image processing, processing data sets, model building, training the model, testing the model,

creating a framework using a python library called flask. The existing solution is analyzed from the paper.

The next step is the Brainstorming session. Brainstorming is a method of generating ideas and sharing knowledge to solve a particular commercial or technical problem, in which participants are encouraged to think without interruption. Brainstorming is a group activity where each participant shares their ideas as soon as they come to mind. At the conclusion of the session, ideas are categorized and ranked for follow-on action. Here three ideas are listed as existing solutions, innovation, and solutions that can't be implemented but can imagine.

3.2.1 Existing solution:

- A video-based hand gesture capture and recognition system used to manipulate MRI within a graphical user interface.
- A hand gesture vocabulary of commands are selected such as 1-Resize, 2-Flip, 3-Rotate, 4-Rectangle.
- Each gesture is cognitively associated with the notion or command that is meant to represent it.
- Using this doctors can analyze the image by having non-verbal communication.

3.2.2 Our solution:

- By tracking the navigation and other gestures and translating to commands based on the temporal trajectories through video capture.

- To develop computer vision algorithms to extract intention and attention cues from the surgeons behavior and combine them with sensory data from a commodity depth camera.

3.2.3 Solution that can't be implemented:

- By using machine learning algorithms for image detection and recognition for sterile browsing of images in radiology.
- But the system becomes more complex and requires a large number of data sets for implementation.

IDEA 1	IDEA 2	IDEA 3
• A video-based hand gesture capture and recognition system used to manipulate MRI within a graphical user interface. • A hand gesture vocabulary of commands are selected such as 1-Resize, 2-Flip, 3-Rotate, 4-Rectangle. • Each gesture is cognitively associated with the notion or command that is meant to represent it. • Using this doctors can analyze the image by having non-verbal communication	• By tracking the navigation and other gestures and translate to commands based on the temporal trajectories through video capture. • To develop computer vision algorithms to extract intension and attention cues from the surgeons behavior and combine them with sensory data from a commodity depth camera.	• By using machine learning algorithms for image detection and recognition for sterile browsing of images in radiology. • But the system becomes more complex and requires large number of data set for implementation.

Fig 3.3 Brainstorming chart

3.3 PROPOSED SOLUTION

In the proposed solution we discussed the project's problem statement, solution description, the novelty in the idea, customer satisfaction, business model, and scalability of the solution.

3.3.1 Problem statement:

To replicate sterile browsing skill in computers using image capture technology and classification techniques.

3.3.2 Solution description:

A video-based hand gesture capture and recognition system used to manipulate MRI within a graphical user interface.

3.3.3 Novelty:

By tracking the navigation and other gestures and translating to commands based on the temporal trajectories through video capture.

3.3.4 Customer satisfaction:

Doctors can analyze the image by having non-verbal communication.

3.3.5 Business model:

The business model of this system extracts intention and attention cues from the surgeon's behaviour. Hence, it is useful for doctors and surgeons from any domain or region all over the world.

3.3.6 Scalability:

By adding a few more gestures for manipulating the MRI images which are essential for sterile browsing by doctors.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To replicate sterile browsing skill in computers using image capture technology and classification techniques
2.	Idea / Solution description	A video based hand gesture capture and recognition system used to manipulate MRI within a graphical user interface
3.	Novelty / Uniqueness	By tracking the navigation and other gestures and translate to commands based on the temporal trajectories through video capture
4.	Social Impact / Customer Satisfaction	Doctors can analyse the image by having non verbal communication
5.	Business Model (Revenue Model)	The business model of this system extracts intension and attention cues from the surgeon's behaviour. Hence, it is useful for the doctors and surgeons from any domain or region all over the world
6.	Scalability of the Solution	By adding few more gestures for manipulating the MRI images which are most essential for sterile browsing by doctors

Table 3.1 Proposed solution table

3.4 PROBLEM-SOLUTION FIT

Problem-solution fit is a term used to describe the point validating that the base problem resulting in a business idea really exists and the proposed solution actually solves that problem.

The problem-solution fit is to -

- **Validate that the problem exists:** When you validate your problem hypothesis using real-world data and feedback. That is, you gather information from real users to determine whether or not they care about the pain point you're trying to solve.
- **Validate that your solution solves the problem:** When you validate that the target audience appreciates the value your solution delivers to them.

Here Project-solution fit chart is created and presented.

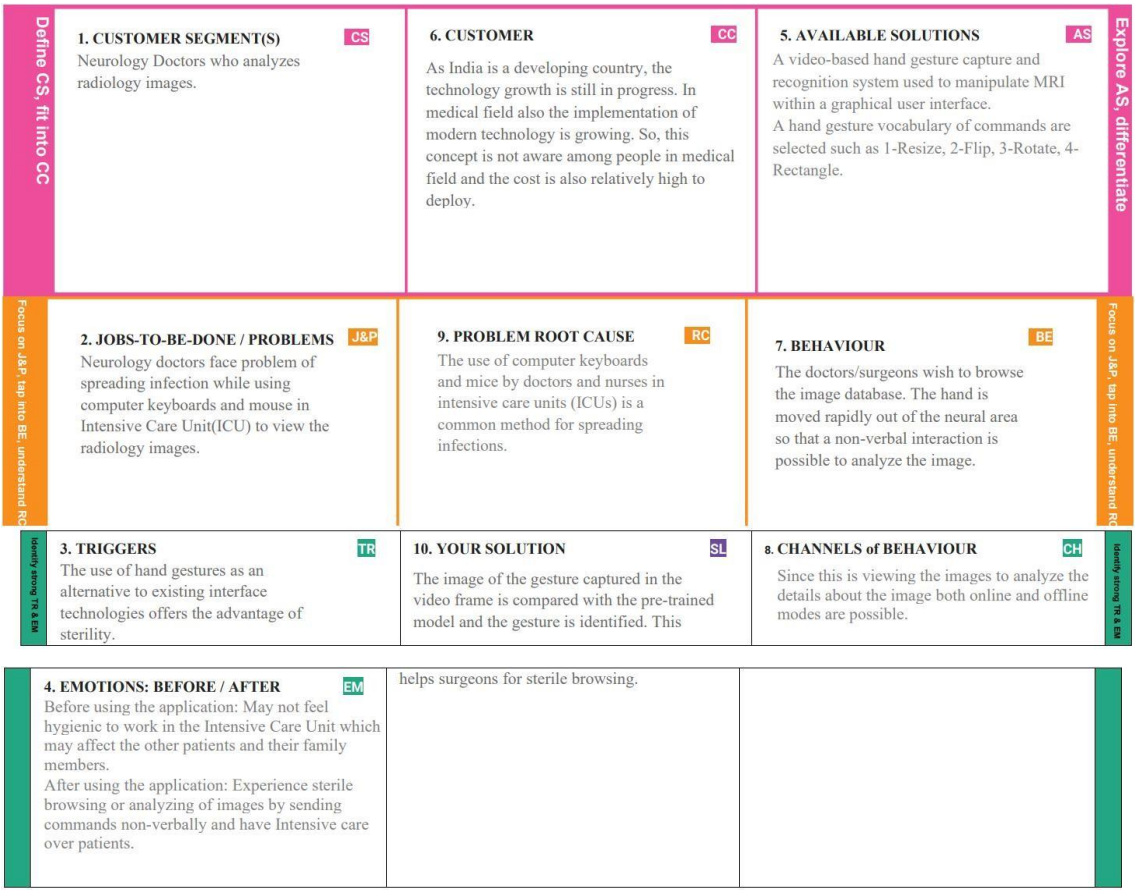


Fig 3.4 Problem-solution fit

CHAPTER 4

REQUIREMENT ANALYSIS

Solution requirements are identified before the technical solution is selected or designed. These requirements can be defined when the two categories mentioned before are understood. They describe the characteristics of a solution (functional and non-functional) that meet Business Requirements and Stakeholder Requirements. These requirements will drive either the selection or the design of the technical solution and its implementation.

4.1 FUNCTIONAL REQUIREMENTS

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected.

The below table lists all the functional requirements of the proposed solution.

FR No.	**Functional Requirement (Epic)**	**Sub Requirement (Story / Sub-Task)**
FR-1	User Registration	Registration through Gmail or other login means for registering the user
FR-2	User Confirmation	Confirmation via Email
FR-3	User Sign up	Sign up using Gmail,user ID and password

FR-4	User Login	Login using user ID and password
FR-5	User input	Setting unique hand gestures for easy and non-verbal communication
FR-6	User application	Input during usage of application to analyse the image via gestures

Table 4.1 Functional Requirement

4.2 NON – FUNCTIONAL REQUIREMENTS

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other.

The Below table lists all the non- functional requirements of the proposed solution.

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	User friendly as the instructions are displayed to the user when they enter the home page. The page will load in a short duration.
NFR-2	**Security**	The user can only login with their user ID and password. The data will be protected from the

		unauthorized user.
NFR-3	**Reliability**	The application will perform up to 80% without failure (in predicting the gesture).
NFR-4	**Performance**	The application will respond within short duration provided the reasonable network speed.
NFR-5	**Availability**	The application will be available as a web page. Like other websites this can be accessed with the domain name. This is available as free service.
NFR-6	**Scalability**	The application can be able to support the workload provided by the user to resize the image to their convenience. By adding few more gesturing for manipulation the MRI images which are most essential for sterile browsing by doctors.

Table 4.2 Non-Functional Requirement

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

There are four main elements of a DFD — external entity, process, data store, and data flow.

5.1.1 External entity

An external entity, which are also known as terminators, sources, sinks, or actors, are an outside system or process that sends or receives data to and from the diagrammed system. They’re either the sources or destinations of

information, so they're usually placed on the diagram's edges. External entity symbols are similar across models except for Unified, which uses a stick-figure drawing instead of a rectangle, circle, or square.

5.1.2 Process

Process is a procedure that manipulates the data and its flow by taking incoming data, changing it, and producing an output with it. A process can do this by performing computations and using logic to sort the data, or change its flow of direction. Processes usually start from the top left of the DFD and finish on the bottom right of the diagram.

5.1.3 Data store

Data stores hold information for later use, like a file of documents that's waiting to be processed. Data inputs flow through a process and then through a data store while data outputs flow out of a data store and then through a process.

5.1.4 Data flow

Data flow is the path the system's information takes from external entities through processes and data stores. With arrows and succinct labels, the DFD can show the direction of the data flow.

The data flow diagram for Gesture based tool for sterile browsing of radiology images is shown in following figure 5.1

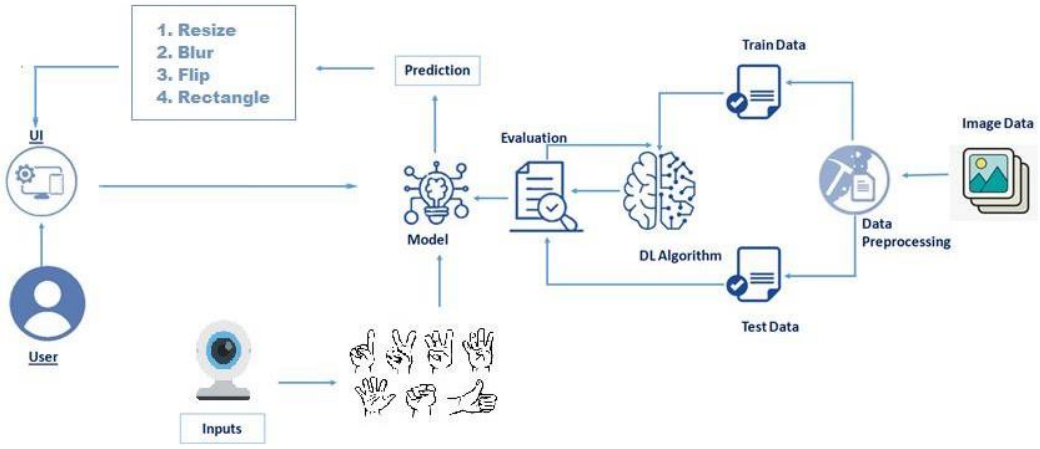


Fig 5.1 Data flow diagram

5.2 SOLUTION & TECHNICAL ARCHITECTURE

Solution architecture provides the ground for software development projects by tailoring IT solutions to specific business needs and defining their functional requirements and stages of implementation. It is comprised of many subprocesses that draw guidance from various enterprise architecture viewpoints.

This table tabulates the components and technologies and the respective descriptions.

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI..	HTML, CSS, JavaScript.
2.	Application Logic-1	Upload image in an application	Python
3.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
4.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
5.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.
6.	Convolutional Neural Network	Initialize the model	CNN Layer

Table 5.1 Components and technologies and the respective descriptions

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Tensor flow,Theano, RNN, pyTorch, Flask
2.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Firewall and other security related softwares
3.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Data, models, operate at size, speed, consistency and complexity
4.	Performance	The system responds to the user in a second and the hardware and software works well	Image and facial recognition, speech recognition and real time captioning

Table 5.2 Application characteristics and the respective descriptions

5.3 USER STORIES

Here are some of the user stories, criteria and priority based on different functional requirements.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Doctor)	Medical image manipulation	USN-1	As a user, I can make use of medical image manipulation and providing faster reponses at critical times	I can access the image manipulation data faster as before	High	Sprint-1
		USN-2	As a user, This model has best ease of use—the system allows me to use just my hands as a natural work tool	I can achieve the set target in short span of time with ease of use	High	Sprint-1
Customer (Surgeon)	Gesture commands operation in real-time	USN-1	As a user, this prevents my focus shift and change of location while achieving a rapid intuitive reaction and easy interaction.	I can use the browsing of data with sterile postures	High	Sprint-1
		USN-2	As a user, this model responds to the surgeon's gesture commands in real-time (intuitive and fast)	I can access the manipulated images very fast and intuitive	High	Sprint-1

Table 5.3 User story table

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team.

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	Download the Dataset	10	High	Anitha B Indhumeena N M
Sprint-1		USN-2	Image Pre-processing	10	High	Indhumathi K Nithiya Shri D
Sprint-1		USN-3	Import and Configure the Image Data Generator Library and Class	10	High	Anitha B Indhumeena N M
Sprint-1		USN-4	Apply Image Data Generator Functionality to Train-Set and Test-Set	10	High	Anitha B Indhumeena N M
Sprint-2	Model Building	USN-5	Import the Model Building Libraries and Initializing the Model	10	High	Indhumathi K Nithiya Shri D

Table 6.1 Sprint planning and estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2		USN-6	Adding CNN Layers and Dense Layers	10	High	Anitha B Indhumathi K
Sprint-2		USN-7	Configure the Learning Process	10	High	Nithiya Shri D Indhumeena N M
Sprint-2		USN-8	Train the Model, Save the Model and Test the Model	10	High	Indhumeena N M Nithiya Shri D
Sprint-3	Application Building	USN-9	Create Web Application using HTML, CSS, JavaScript	10	High	Anitha B Indhumeena N M
Sprint-3		USN-10	Build Python code	10	High	Anitha B Indhumeena N M
Sprint-4	Train The Model on IBM	USN-11	Register for IBM Cloud	10	High	Indhumathi K Nithiya Shri D
Sprint-4		USN-12	Train the Model and Test the Model and its Overall Performance	10	High	Nithiya Shri D Anitha B

Table 6.2 Sprint delivery plan

CHAPTER 7

CODING AND SOLUTIONING

7.1 FEATURE

7.1.1 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.

7.1.2 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')
```

7.1.3 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()
```

7.1.4 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())
```

7.1.5 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
```

7.1.6 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

7.1.7 Train The Model

Train the model with our image dataset.

fit_generator functions used to train a deep learning neural network

7.1.8 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
...
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)
```

7.1.9 Test The Model

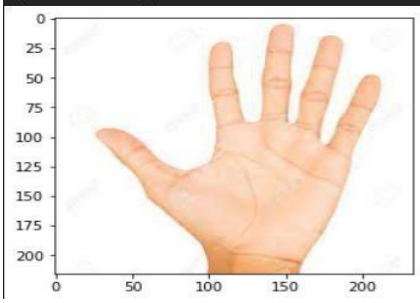
Evaluation is a process during development of the model to check whether the model is best fit for the given problem and corresponding data.

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"
```

7.1.10 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)
```

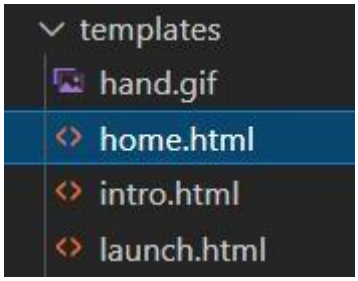
```
[array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([1],
dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([2], dtype=int64),
array([2], dtype=int64),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

7.1.11 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.

7.1.12 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.



7.1.13 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
```

- 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():
```

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

7.1.14 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 uploads folder in the image 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)
```

7.1.15 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)
```

7.1.16 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)
```

7.1.17 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")
```

CHAPTER 8

TESTING

8.1 TEST CASES

8.1.1 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 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.

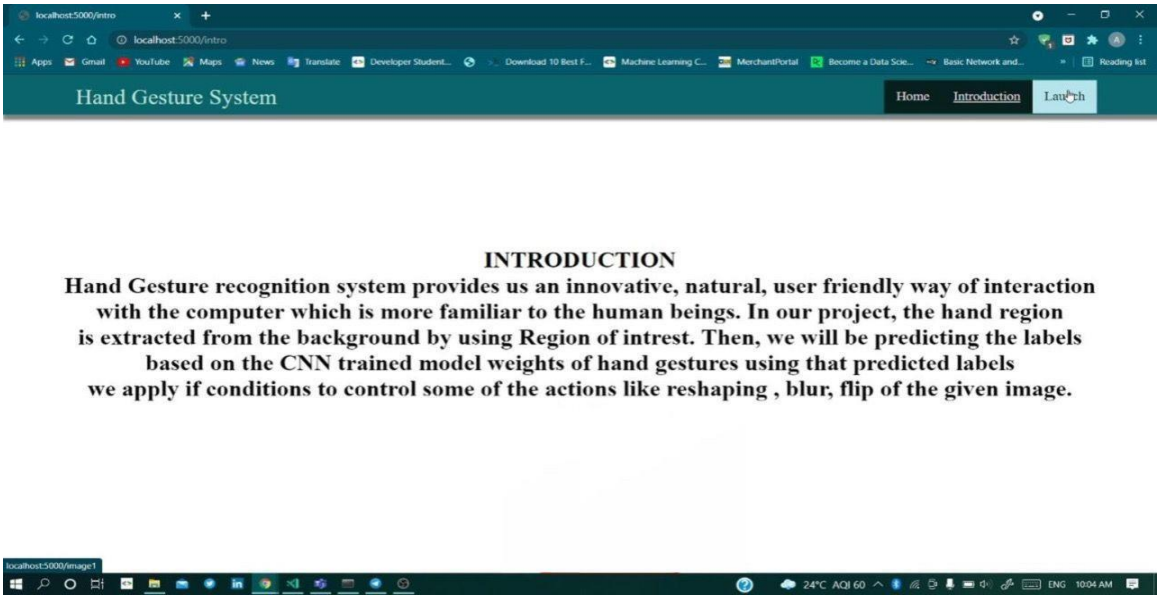
CHAPTER 9

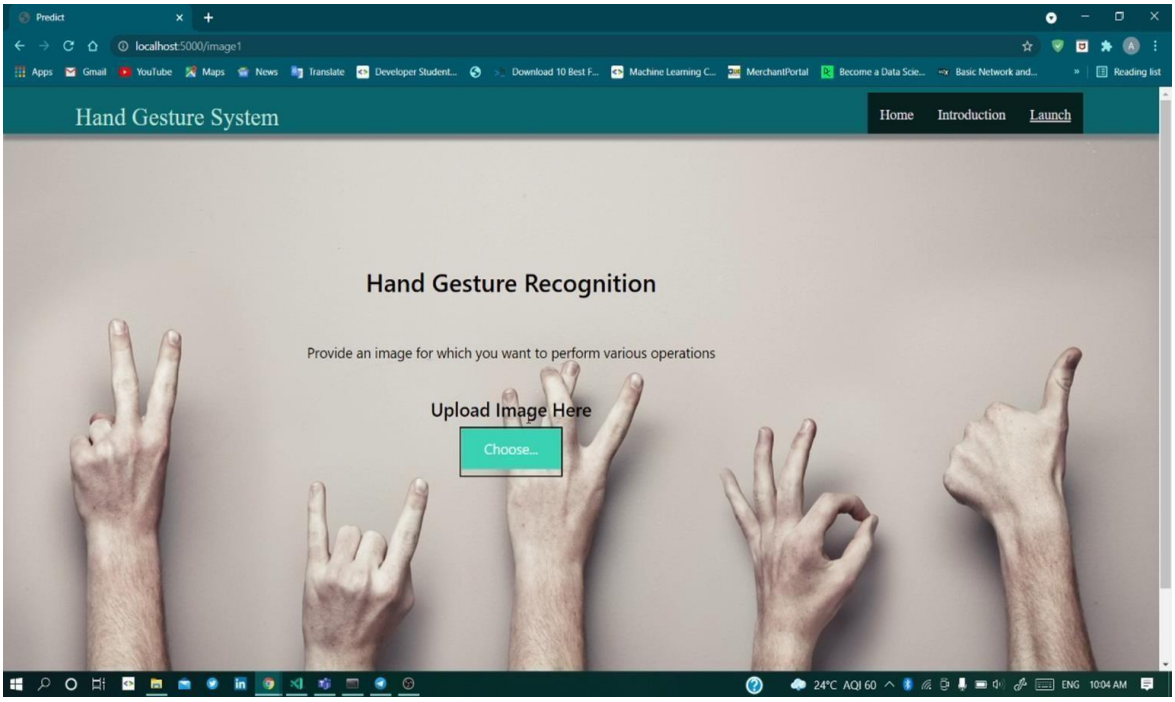
RESULTS

9.1 PERFORMANCE METRICS

The final html webpage for hand gesture recognition is shown below:

When the Introduction tab is selected, local host redirects to “intro.html”. The hand gesture system is explained in the Introduction tab.Upload the image and click on Predict button to view the result.




Predict
localhost:5000/image1
Hand Gesture System
Home
Introduction
Launch
Hand Gesture Recognition
Provide an image for which you want to perform various operations
Upload Image Here
Choose...

CHAPTER 10

ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- The 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 im ages.
- It can also be performed even if the surgeon is a bit far away from thesyste m, this helps to save time.
- The tool does not need the person using it to have an apparatus or any devi ces 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.

CHAPTER 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.

CHAPTER 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 usability 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.

APPENDIX

SOURCE CODE

```
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")



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

    # 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)
        # Batch of 1
        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)
        cv2.imshow("Frame", frame)


        #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")

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

GITHUB LINK

https://github.com/smartinternz02/Gesture-based-Tool-for-Sterile-Browsing-of-Radiology-Images-Using-IBM-Watson