LEXCON:AN ELECTRONIC SLATE FOR DEAF-MUTE

PROJECT REPORT

submitted by

AARCHA A S (Reg. No.PRN16EE001)
ANANDHU V (Reg. No.PRN16EE014)
ANCHANA KRISHNAN A (Reg. No.PRN16EE015)
SWATHY S G (Reg. No.PRN16EE057)

to

the APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology
In
Electrical and Electronics Engineering

Department of Electrical and Electronics Engineering

College of Engineering, Perumon

JULY 2020

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

COLLEGE OF ENGINEERING, PERUMON

CERTIFICATE

This is to certify that the Project report entitled " **LEXICON: AN ELECTRONIC SLATE FOR DEAF-MUTE "** submitted by **AARCHA A S, ANANDHU V, ANCHANA KRISHNAN A** and **SWATHY S G** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Electrical and Electronics Engineering is a bonafide record of the Project work carried out by him/her under my/our guidance and supervision during the period of 2019-2020.

Project Guide

Dr. Bindu S. J.
Associate professor
HoD, Dept. of Electrical & Electronics
College of Engineering Perumon

Project Coordinator

Edwina Rodrigues
Assistant Professor
Dept. of Electrical & Electronics
College of Engineering Perumon

Project Coordinator

Renjith G
Assistant Professor
Dept. of Electrical & Electronics
College of Engineering Perumon

Head of the Department

Dr. Bindu S. J.
Associate Professor
HoD, Dept. of Electrical & Electronics
College of Engineering Perumon

ACKNOWLEDGEMENT

First and foremost, We sincerely thank the almighty for his grace for the successful completion of our project on “LEXICON: AN ELECTRONIC SLATE FOR DEAF-MUTE”.

We hereby express sincere gratitude to our Project guide Mrs. Bindu S.J, Associate Professor & HoD, Department of Electrical and Electronics Engineering Department, College of Engineering Perumon for her valuable guidance and suggestions.

We also express our sincere thanks to Mrs. Edwina Rodrigues, Assistant Professor in Electrical and Electronics Engineering Dept. for her valuable assistance and necessary directions which aided us in the completion of this venture.

We also express our sincere thanks to Prof. Dr. Z. A Zoya, Principal, College of Engineering Perumon, for providing us with the best facilities and all the support. Finally, we extend our gratitude to the entire faculty of the department and to all our friends for their help in carrying out this work successfully.

AARCHA S

ANANDHU V

ANCHANA KRISHNAN A

SWATHY S G

ABSTRACT

Deaf and dumb conversation is hinged on sign language which renders words through hand and finger symbols. Normal people including many of us do not know what the sign language expresses. Hence, they find difficulty in communicating with normal people Main objective of the project is building a device that assist deaf and dumb people to convey their messages to normal people.

The system consists of two main parts. The first part is hand sign recognition and the second one is speech recognition. This ensures a bi directional communication avoiding the assistance of an interpreter.Main components of our system are webcam with in-built mic, Raspberry pi 4B ,a display device and a switch .Using Webcam with an in-built mic, either image is acquired from the environment or speech is taken as the input.Switch is provided for selecting the hand recognition mode or speech to text mode. If switch is in off state, the hand gesture mode is activated. If the switch is in on state, speech to text mode is activated. Hand recognition is done by using Convolutional Neural Networks. In the speech recognition part, speech to text is done by google API. A high performance and power efficient ARM Cortex A72 Processor recognition board is used. Sign language recognition has established its importance in many areas such as Human Computer Interactions.

CONTENTS

LIST OF FIGURES

ABBREVIATIONS

API	Application programming Interface
ASR	Automatic speech recognition
CNN	Convolutional neural network
CSI	Camera serial Interface
FLAC	Free lossless audio codec
FPS	Frame per second
GPIO	General purpose input output
GPU	Graphics Processing unit
gRPC	Google remote procedure call
HDMI	High definition multimedia Interface
HSV	Hue Saturation Value
LCD	Liquid crystal Display
MIPI	Mobile Industry Peripheral interface
REST	Representation state transfer
SGD	Stochastic gradient descent
SoC	System on chip
URI	Uniform resource Identifiers
WAV	Wave audio format

NOTATION

P	pulse motor
Hz	Hertz

CHAPTER 1
INTRODUCTION

1.1 GENERAL BACKGROUND

Sign languages or finger alphabets are used by those who are hear less or speechless. Also, it is used by people who can hear but cannot talk. Normal people including many of us do not know what the sign languages expresses. Hence, we find difficulty in communicating with such peoples. More than that, the problem faced by them due to lack of communication are many. The pre-eminent target of the system is to build an assistance for communication of deafened and speechless people. This waive the fence for communication between normal and disabled people. This eliminates the help of an interpreter for communication.

Many systems have been proposed so far. Approaches based on sensors are fast but expensive and accuracy of results cannot be assured. Some systems are light dependent so variation in light intensities interfere with the actual results. The amount of background noise also causes tracking and recognition difficulties. Our project aims to mitigate all these demerits.

The system proposed ensures a bidirectional communication between the disabled and the normal people. It is constituted by two main parts. A hand sign recognition module and a speech to text module. Main components of our system are webcam built with mic, raspberry pi 4B,a display device and a switch, Webcam built with mic is based on the mode selected, either , image is acquired from the environment or speech is taken as the input.Switch is provided for selecting the hand recognition mode or speech to text mode. If switch is in off state, the hand

gesture mode is activated. If the switch is in on state, speech to text mode is activated. The hand sign recognition part is used by deafened and speechless people to channel their ideas to normal people in visual form. The speech recognition part is used by listener to convey their reply to disabled people in also a visual form.

1.2 OBJECTIVES

1. The main objective of this project is to bridge the communication gap between speech and hearing-impaired people and normal people.
    - Two modes which are Hand gesture recognition and speech to text are provided for better communication between them.
    - Hand gesture recognition part converts hand gesture of normal people to text which is then used to make sentences.
    - Speech to text mode converts voice commands of normal people to text, so that disabled people can read it.

2. To design a fully functional real-world product that enables bidirectional communication between normal and disabled people so that it can be used in public spaces to overcome the lack of communication.

CHAPTER 2
LITERATURE SURVEY

1. ‘Hand Gesture Recognition and Voice Conversion System For Dumb People’. Vigneshwaran, ShifaFathima, VijaySagar , SNS College of Technology.

   2019 International Conference on Advanced Computing & Communication Systems (ICACCS).

   Here the flex sensor used for detecting the hand motion. Based upon the bending of the sensor their resistance is varied. Then their voltage also varied, according to the voltage value the command will be provided. It also incorporated a system to convert human speech to text.

   - From this paper we got the idea that bidirectional communication can be implemented to bridge the communication gap. But the sensor-based method created much inconvenience to user when the sensors are to be worn on hand and inappropriate bending of thee fingers led to inaccurate results.

2. ‘A belief-based sequential fusion approach for fusing manual signs and non manual signals’.

   Thad Starner, Joshua Weaver, and Alex Pentland.

   International journal of Pattern Recognition Letters, volume-32, number-4, pages: 572–577, pages 812-822 ,2009.

   In this paper **a** wearable camera is used where hand gestures are detected using HMM. Here 6000 gestures for common words are detected.

   - The method is unable to form sentences. Also, Speech of the normal people

is not converted to a form that can be understood by Deaf-Mute people.

3. ‘An Effective Approach to Communicate with the Deaf and Mute People by Recognizing Characters of One-hand Bangla Sign Language Using Convolutional Neural-Network’.

   Alfat Jahan Rony, Khairul Hossain Saikat, Mahdia Tanzeem and F.M.Rahat Hasan Robi.

   2018 4th International Conference on Electrical Engineering, Information & Communication Technology.

   In this paper Convolutional neural networks are used to to train and detect alphabets of Bangla one hand sign language and to make sentences with.

   - From this paper we got the idea to use convolutional neural networks which is quite accurate to classify alphabets of American sign language and some special characters.

4. Speech to text conversion system using google API

   Documentation in Google cloud

   - From this we got the technique of speech to text conversion.

.

CHAPTER 3
PROPOSED SYSTEM

3.1 BLOCK DIAGRAM

The system is constituted by two main parts. One is hand sign recognition part and the other is speech recognition part. The sign language recognition part is used by deaf and mute to channel their ideas to normal people in the form text thereby forming sentences. The speech recognition part is used by listener to convey their audio reply to disabled people also in the form of text. Both the modules are implemented in a Raspberry Pi 4B minicomputer. A switch is used for mode selection. A webcam with inbuilt mic is used to receive the voice commands and to capture the gesture. The converted text is seen on an LCD display. In this way the system made two-way communication possible. The block diagram of proposed system is shown in fig.3.1

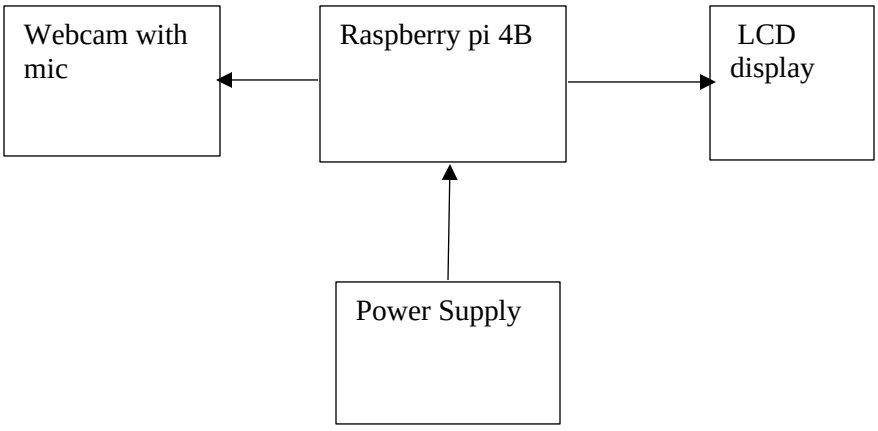

Fig.3.1 Proposed block diagram

3.2 COMPONENT DESCRIPTION

3.2.1 Raspberry Pi 4B

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation. Several generations of Raspberry Pi's have been released. All models feature a Broadcom system on a chip (SoC) with an integrated ARM-compatible central processing unit (CPU) and on-chip graphics processing unit (GPU). Raspberry Pi 4 Model B shown in figure 3.2 offers improved performance in terms of processor speed, multimedia performance, memory, and connectivity compared to the prior models. The key features are a high performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, and up to 4GB of RAM**.** The pin out diagram of it is shown in figure 3.3.

Fig.3.2 Raspberry pi 4B

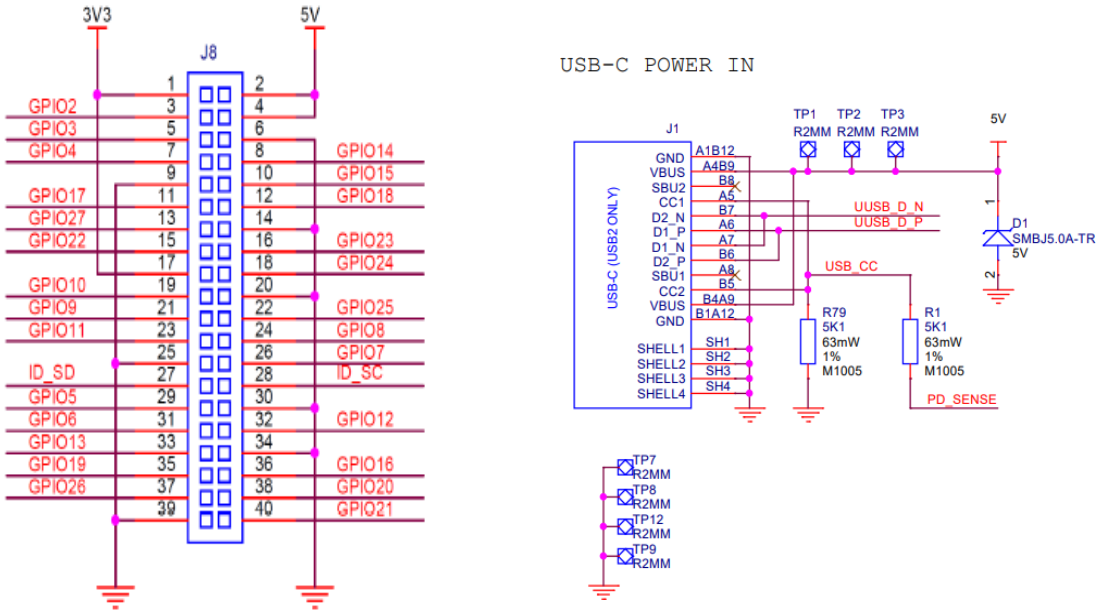

Fig.3.3 Pin out diagrams of Raspberry Pi 4B

3.2.1.2 Documentation

Power- The Pi4B requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.

Processor- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz.

Camera and Display Interfaces- The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2- lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.

Input and output- General purpose input/output pins can be configured to either read or write Inputs.

SD/SDIO interface- The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB

current is limited to approximately 1.1A in aggregate over the four sockets.

USB- The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current islimited to approximately 1.1A in aggregate over the four sockets.

HDMI- The Pi4B has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.

3.2.2 **Zebronics crystal pro web camera pi Camera**

Zeb – crystal pro is a USB powered web camera with built-in mi. Based on the mode selected, either , image is acquired from the environment or speech is taken as the input. It comes with 3P lens that produces that produces high resolution. The camera supports video resolution up to 640 x 480 [30 FPS]. The Webcam comes with a USB interface that makes it compatible with most of PCs.Fig 4.4 shows the camera.

Fig.3.4 Zebronics crystal pro web camera

3.2.3 LCD display

This 20*4 character LCD screen is built in with RW1603 control IC which are 4 lines SPI 6800 or I2C port choices .The WH2004G 20*4 LCD Display have the AA size and pin assignment as WH2004A AND WH2004B character LCD model but using smaller outline and VA size.They have 5V power supply. Negative voltage optional for is 3V power supply. The duty cycle is 1/16. LED can be driven by PIN1,PIN2,PIN15,PIN16 or A and Interfaces are WH2004G-6000,WH2004G1-SPI,WH2004G2-I2C.The LCD display is shown in fig 4.5 below.

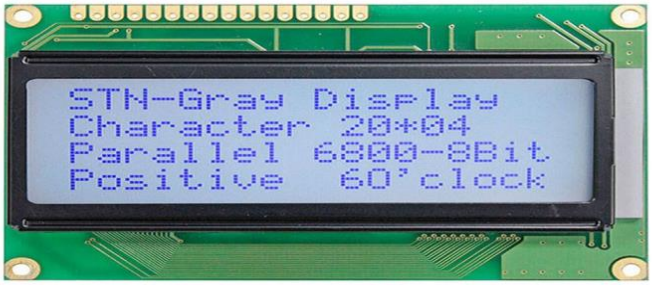

Fig.3.5 LCD display

3.3 CIRCUIT DIAGRAM OF PROPOSED SYSTEM

The switch used for mode selection is connected between GPIO 12 of Raspberry pi and the ground. LCD is interfaced to the microcontroller via GPIO 2 and GPIO 3. The display is powered with 5V by taking supply from Raspberry pi. The circuit diagram of the proposed system is shown in fig.4.6

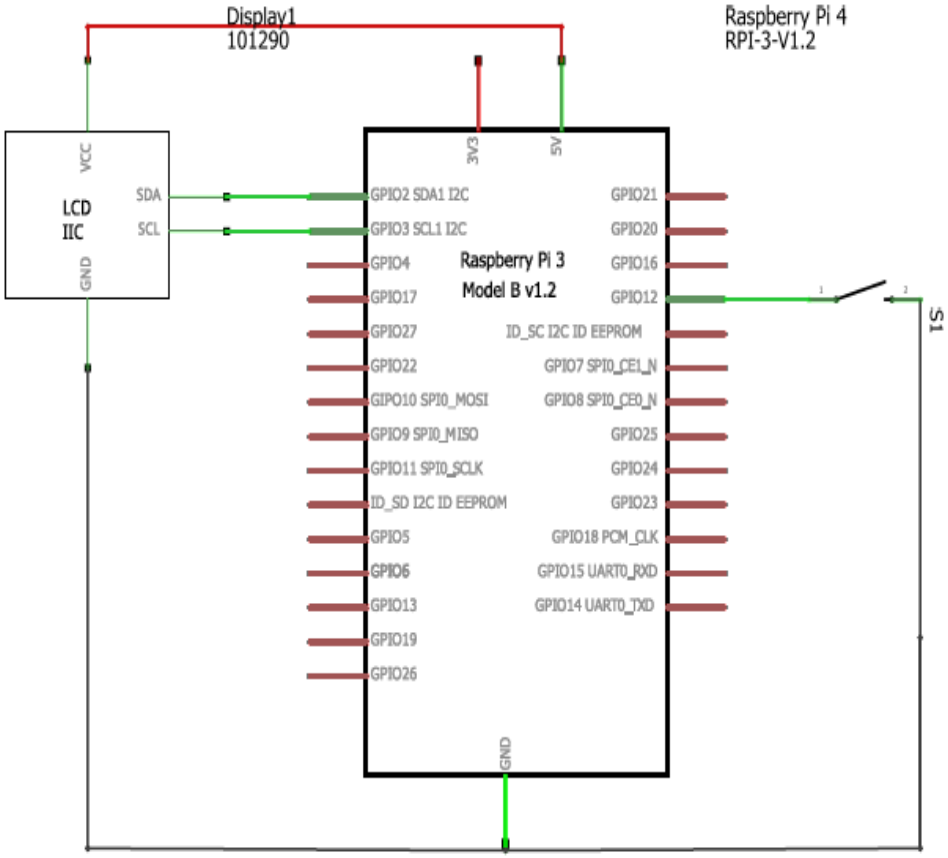

Fig.4.6 Circuit diagram of proposed system

3.4 SOFTWARE TOOLS

3.4.1 Open CV

Open CV (Open Source Computer Vision Library) is a library which mainly focuses at real-time computer vision. It supports programming languages such as python, c++ and java. The collection of more than 2500 algorithms make this tool suitable for computer vision and machine learning applications. These algorithms can be used to detect and recognize faces, identify objects, find similar images from database etc. For real rime image processing open CV can provide all the basic data structures. Python is used for programming in our project.

3.4.2 Tensorflow

Tensorflow is a free and open-source software library for machine learning system and developers can easily build ML powered applications. It has a comprehensive, flexible system of tools that lets the researchers put the state of art in ML. It can easily train and deploy models in the cloud, in the browser, or on device regardless of the language we use. It has a simple and flexible architecture to take new ideas from concept to code and especially in image processing applications.

3.4.3 Keras

Keras is a high-level neural networks API, written in python and capable of running on top of Tensorflow. It was developed with a focus of fast experimentation. Allows for easy and fast prototyping through user friendliness, modularity. Keras is a powerful module that allows us to avoid having to build neural networks from scratch. Keras also hides a lot of mathematical complexity inside of helpful packages, modules and methods.

CHAPTER 4
METHODOLOGY

4.1 THEORY

4.1.1 Neural networks

A deep neural network is a layered representation of data. The network learn to recognize the relationship between relevant features. In each layer our data is transformed in order to learn more about it. By performing these transformations , the model can better understand our data and therefore provide a better prediction.

A neural network can be decomposed as shown below in fig.4.1.

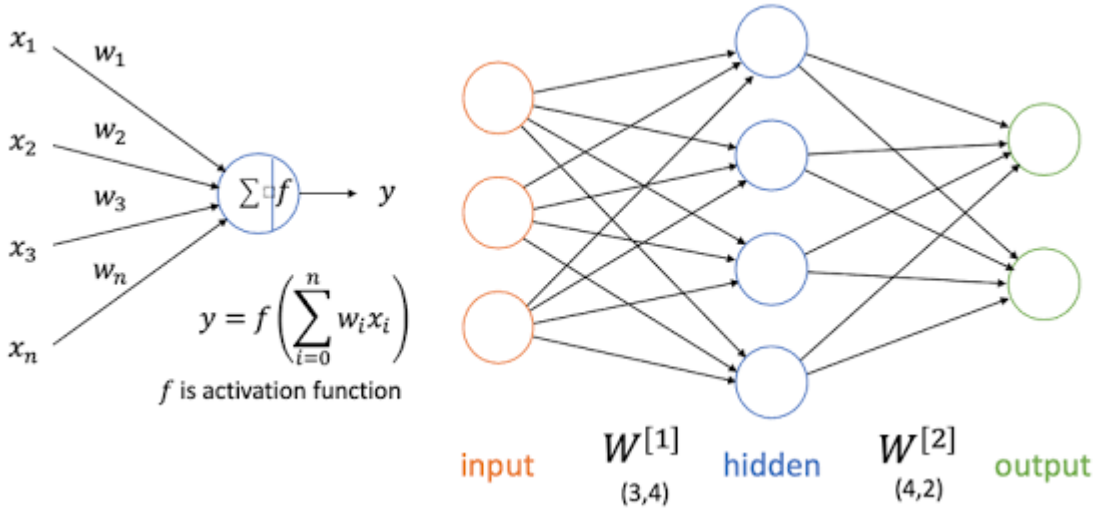

Fig.4.1 Example of a neural network

Data

The type of data that a neural network handle varies depending on the

applications. It may sometimes be necessary to modify the datasets that it can be passed to the neural network. Common type of data used is Image which is the case here also.

Input layer

The input layer is the layer that our initial data is passed to. It is the first layer in our neural network.

Output layer

The output layer is the layer that we will retrieve our results from. Once the data has passed through all other layers it will arrive here.

Hidden layers- This is because they are hidden to us, we cannot observe them. Most neural networks consist of at least one hidden layer but can have an unlimited amount. The more number of hidden layers, the more complex the network will be.

Neurons- Each layer is made up of what are called neurons. The important aspect to understand now is that each neuron is responsible for generating or holding one numeric value.

Connected layers- Each neuron in one layer is connected to every neuron in the next layer. This is called a dense layer.

Weights - Weights are associated with each connection in our neural network. Every pair of connected nodes will have one weight that denotes the strength of the connection between them. These are vital to the inner workings of a neural network and will be tweaked as the neural network is trained. The model will try to determine what these weights should be to achieve the best result.

Biases

Biases are another important part of neural networks and will also be tweaked as the model is trained. A bias is simply a constant value associated with each layer.

Activation function

Activation functions are simply a function that is applied to the weighed sum of a neuron. They can be anything we want but are typically higher order/degree functions that aim to add a higher dimension to our data. A common example is shown in fig.4.2

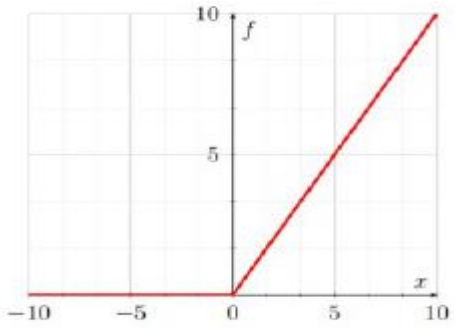

Fig.4.2 Rectified linear unit function

Back propagation

Backpropagation is the fundamental algorithm behind training neural networks. It is what changes the weights and biases of our network.

Loss /Cost function

This function is responsible for determining how well the network did. We pass it the output and the expected output, and it returns to us some value representing the cost/loss of the network. This effectively makes the networks job to optimize this cost function, trying to make it as low as possible.

Gradient descent

Gradient descent is the algorithm used to find the optimal parameters (weights and biases) for our network.

4.1.2 Convolutional neural networks

It is a neural network in which at least one layer is a convolutional layer. The goal of the network is to find patterns from within images that can be used to classify the image. For this, we use image data as features and a label for those features as label or output.

A Convolutional neural network consists of some combinations of the following layers.

- Convolutional layers
- Pooling layers
- Dense layers

The fundamental difference between a neural network and a CNN is that the former detects patterns globally while CNN detect patterns locally. While the neural networks output numeric values, the output of a CNN will be feature maps.

A convolutional layer is particularly used for image processing applications. An image data is usually made up of three dimensions which are height, width, color. The number of color channels coorelates to the colors used in it. Each pixel is associated with a numeric value between 0-255.

A convolutional layer is composed of Multiple convolutional layers. In our models we use three convolutional layers. These layers work together by Increasing complexity and abstraction at each subsequent layer.

A prime difference between neural network and a convolutional network is that a neural network outputs a bunch of numeric values while a CNN outputs Feature maps. The convolutional layer takes Feature maps as input and return a new feature map that represents the presence of specific filters from the previous feature maps. Fig 4.3 shows the concept of a feature map.

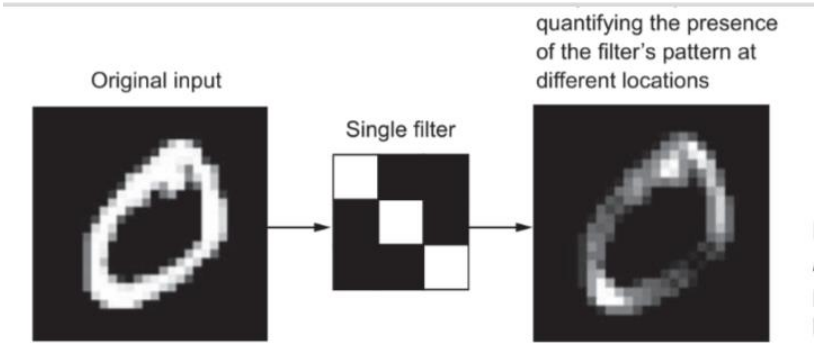

Fig.4.3 The concept of a feature map

A filter is a m x n pattern of pixels that we are looking in an image. The number of filters represent how many patterns each layer is looking for and the depth of response map.

Each convolutional layer is going to examine n×m pixels in each image. It is the sample size of the filter. The layers work by sliding the filters of n×m pixels over every possible position in the image and producing a new feature map.

The convolutional layers take feature maps as inputs and new feature maps as outputs that represents the presence of specific filters from the previous feature map. As there are

about 32 or 64 filters in each convolutional layer, it returns a large number of feature maps which make the system more complex. To make the system simple and easier to use an operation called pooling is performed.

The pooling layer is to down sample the feature maps and reduce their dimensions. They work in a way similar to convolutional layers where they extract windows from the feature map and return a response map of the max, min or average values of each channel. Pooling is usually done using a windows of size 2×2 and a stride of 2.

A common architecture for a CNN is a stack of Conv2D and Maxpooling2D layers followed by a few densely connected layers. The figure of CNN architecture implemented is shown in fig.4.4.

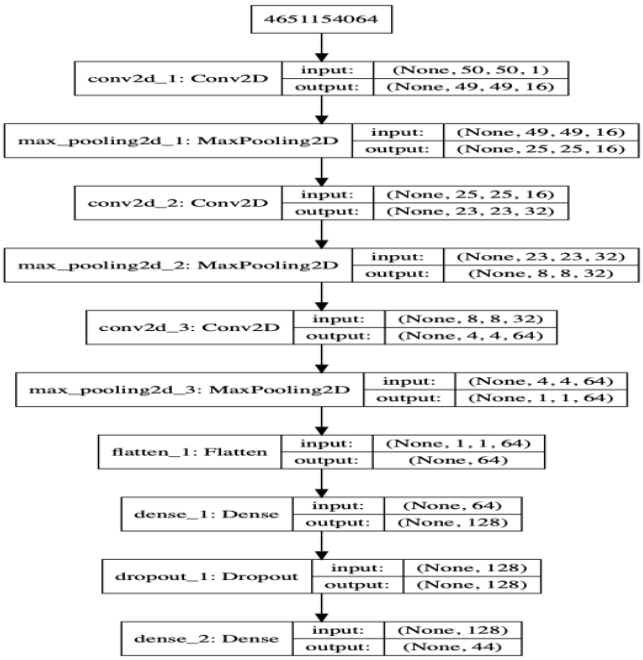

Fig.4.4 Implemented CNN architecture

4.2 METHODOLOGY FOR HAND GESTURE RECOGNITION

The system was implemented through a Raspberry pi 4B with a 3p lens web camera with mic. The camera captured the images of the hands that will be fed in the system. When the camera has already captured the gesture from the user, the system classifies the test sample and compared it in the stored gestures in a dictionary, and the corresponding output was displayed on the screen for the user. By converting each American sign language alphabets to letters sentences could be formed.

The first step was Gathering of Training Data, Image Augmentation, and Cropping Procedures. The Kaggle dataset of American sign language , 0-9 digits and eight special characters are used for training. For each gesture, there were 2000 images in the dataset and were being converted to 50 x 50 pixels black and white samples. The Kaggle dataset is shown in fig.4.5

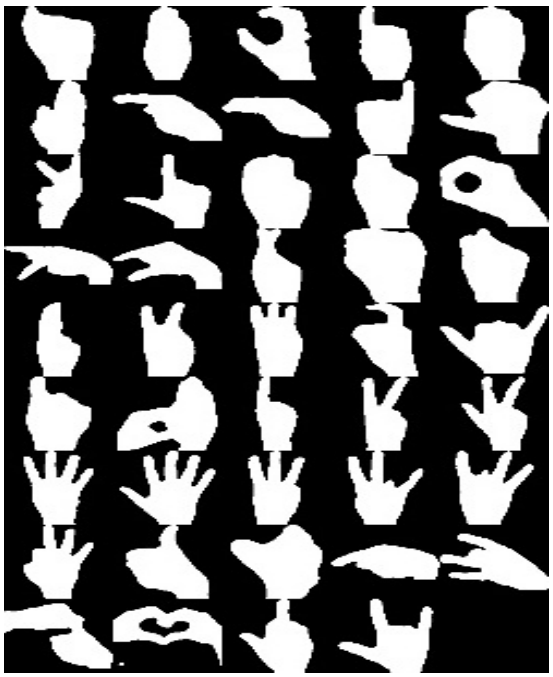

Fig.4.5 Kaggle dataset

The next step was hand Skin Color Detection using Image Processing. For improved skin colour recognition, the signer was advised to have a clear background for the hands, which will make it easier for the system to detect the skin colours. Skin detection took place by using cv2.cvtColor. Images were converted from RGB to HSV. Through the cv2.inRange function, the HSV frame was supplied, with the lower and upper ranges as the arguments. The mask was the output from the cv2.inRange function. White pixels in the mask produced were the region of the frame weighed as the skin. Lastly resulting masks were smoothened by gaussian blur.

The goal of this study was to design a network that can effectively classify an image of a static sign language gesture to its equivalent text by a CNN. The Pre-processed images were then given to CNN layers. To attain specific results, we used Keras and CNN architecture containing a set of different layers for processing of training of data. The CNN architecture is composed of conv layer of 16 filters, then a pooling layer, then a conv layer of 32 layers followed by pooling layer, again a conv layer of 64 filters followed by pooling layer.

The model was then being flattened or converted into a vector; then, two dense layers were added. The network's output was then a 25-dimension vector similar to each sign language alphabets, a 10-dimension vector similar to 0-9 digits, an 8-dimension vector similar to other special characters. Then, for a 44-dimension vector corresponding to static sign language gestures, each class was trained through the network individually.

The training for the gestures and static sign language recognition was done separately. Each dataset was divided into two: Training and testing. The network was

implemented and trained through Keras and TensorFlow. The total number of epochs used to train the network is 50 epochs with a batch size of 500. The images were resized to (50, 50, 1) for training and testing. We use a stochastic gradient descent optimizer, also known as the incremental gradient descent, to minimize the batch size of large datasets. SGD helps to eliminate redundancy.

The batch gradient descent performs redundant computations for large datasets as gradients are recalculated before each parameter update for similar examples. By performing one update at a time, SGD eliminates this redundancy. It is typically much faster and can also be used for online learning.

4.3 SPEECH RECOGNITION USING GOOGLE API

4.3.1 Synchronous Recognition

A Speech-to-Text API synchronous recognition request is the simplest method for performing recognition on speech audio data. Speech-to-Text can process up to 1 minute of speech audio data sent in a synchronous request. After Speech-to-Text processes and recognizes all of the audio, it returns a response.

A synchronous request is blocking, meaning that Speech-to-Text must return a response before processing the next request. Speech-to-Text typically processes audio faster than real time, processing 30 seconds of audio in 15 seconds on average. Speech-to-Text has both REST and gRPC methods for calling Speech-to-Text API synchronous.

4.3.2 Synchronous Speech Recognition Requests

A synchronous Speech-to-Text API request consists of a speech recognition configuration, and audio data. A Recognition configuration contains the sub-fields such as encoding, sample rate, language code, max alternatives, profanity filter, speech context.

- Encoding - (required) specifies the encoding scheme of the supplied audio. If you have a choice in codec, prefer a lossless encoding such as FLAC or LINEAR16 for best performance.
- Sample Rate Hertz - (required) specifies the sample rate (in Hertz) of the supplied audio. The sample Rate Hertz field is optional for FLAC and WAV files where the sample rate is included in the file header.

- Language Code - (required) contains the language + region/locale to use for speech recognition of the supplied audio. The language code must be a BCP-47 identifier. The language codes typically consist of primary language tags and secondary region sub tags to indicate dialects.
- Max Alternatives - (optional, defaults to 1) indicates the number of alternative transcriptions to provide in the response. By default, the Speech-to-Text API provides one primary transcription. If you wish to evaluate different alternatives, set max alternatives to a higher value.

- Profanity Filter - (optional) indicates whether to filter out profane words or phrases. Words filtered out will contain their first letter and asterisks for the remaining characters. The profanity filter operates on single words, it does not detect abusive or offensive speech that is a phrase or a combination of words.
- Speech Context - (optional) contains additional contextual information for processing this audio. A context contains the following sub-field:

  Phrases - contains a list of words and phrases that provide hints to the speech recognition task.

Audio is supplied to Speech-to-Text through the audio parameter of type recognition audio. The audio field contains either of the following sub-fields:

- Content- contains the audio to evaluate, embedded within the request. Audio passed directly within this field is limited to 1 minute in duration.
- Uri- contains a URI pointing to the audio content. It is use to address the supplied audio resources. The file must not be compressed.

4.3.3 Selecting models in speech processing systems

Speech-to-Text can use one of several machine learning models to transcribe audio file. Google has trained these speech recognition models for specific audio types and sources. When one sends an audio transcription request to Speech-to-Text, one can improve the results that you receive by specifying the source of the original audio. This allows the Speech-to-Text API to process your audio files using a machine learning model trained to recognize speech audio from that particular type of source.

Speech-to-Text can use the following types of machine learning models for transcribing the audio files.

- Video- Use this model for transcribing audio in video clips or that includes multiple speakers. For best results, provide audio recorded at 16,000Hz or greater sampling rate.
- Phone calls- Use this model for transcribing audio from a phone call. Typically, phone audio is recorded at 8,000Hz sampling rate.
- ASR: Command and search - Use this model for transcribing shorter audio clips. Some examples include voice commands or voice search.
- ASR Default- Use this model if your audio does not fit one of the previously described models. For example, you can use this for long-form audio recordings that feature a single speaker only. Ideally, the audio is high-fidelity, recorded at 16,000Hz or greater sampling rate.

4.3.4 Speech to text API responses

Synchronous Speech-to-Text API response may take some time to return results, proportional to the length of the supplied audio. It contains several fields such results, alternatives, transcriptions, confidence value.

- Results- contains the list of results where each result corresponds to a segment of audio (segments of audio are separated by pauses).
- Alternatives- contains a list of possible transcriptions, of type speech recognition alternatives. Whether more than one alternative appears depends both on whether you requested more than one alternative (by setting max Alternatives to a value greater than 1) and on whether Speech-to-Text produced alternatives of high enough quality. Setting max Alternatives to a higher value than 1 does not imply or guarantee that multiple alternatives will be returned.
- Transcriptions- contains a list of possible transcriptions, of type speech recognition alternatives. Whether more than one alternative appears depends both on whether you requested more than one alternative (by setting max Alternatives to a value greater than 1) and on whether Speech-to-Text produced alternatives of high enough quality. Setting max Alternatives to a higher value than 1 does not imply or guarantee that multiple alternatives will be returned.
- Confidence- The confidence value is an estimate between 0.0 and 1.0. It's calculated by aggregating the "likelihood" values assigned to each word in the audio. A higher number indicates an estimated greater likelihood that the individual words were recognized correctly. This field is typically provided only for the top hypothesis, and only for results where is final = true. If Speech-to-Text determines that an alternative has a sufficient confidence value, then that

alternative is included in the response. The first alternative in the response is always the best (most likely) alternative.

4.4 ALGORITHM FOR GESTURE RECOGNITION & SPEECH TO TEXT CONVERSION

Step 1: select mode using switch. If switch on, go to 8

Step 2: Capture gesture

Step 3: Pre-processing of image

Step 4: Analyse gesture using CNN

Step 5: Compare the given gesture with trained gesture stored in database.

Step 6: Recognizing gesture.

Step 7: Converting gesture to text. Go to step 14.

Step 8: Receive voice commands using webcam with microphone

Step 9: Audio given to google through raspberry pi

Step 10: Analysing audio using Google Voice API

Step 11: Recognizing the audio

Step 12: Converting audio to text

Step 13: Converted text is given to Raspberry Pi

Step 14: Text is displayed

CHAPTER 5

IMPLEMENTATION

The system was implemented within a case made up of PVC foam sheet. A switch was provided on the frame of the case for mode selection by the user. The implemented system is as shown in fig.5.1

Fig.5.1 Implemented hardware

It was recommended in the testing stage that the light is not directly hitting the hand. The system was sensitive to light; thus, determining the proper place of the lamp should be considered. If the edges of the hand in the masking were detected clearly, the user may begin to use the translator. For the signs to be recognized, the hand should in front of the camera. The detection could only be done if the hand is inside the box that can be seen on the screen of a computer's monitor. Since the size of the hand of each

individual was different, a user may move his/her hand back and forth to fit inside the virtual box. The user should then wait for the system to generate the desired equivalent of the signs in textual form. It also recommended that the user's hand does not make any movement until the system generates the output.

In the testing stage 44 gestures were used to test the system. The system recognized 42 of them correctly. The detection accuracy was 95.45%. Computational time of the system for train the data set is 51.14 minutes. It includes loading the images from the folder 5.42 minutes, creating jpg.txt file (creating the train file). It took 3s to detect the hand 4.2s to recognize hand sign and 2.4s to convert the sign to text.

The system didn't work well in low light. Speech to text module takes 8s to convert the speech to text. To start the system, it took around 35s. It was also mandatory to keep the background white as because the background of the trained images was white.

CHAPTER 6
CONCLUSION

Compared with any methods, the proposed system is slow, accurate and dependent on data. It is believed that improvement of computer operating speed can improve the real time performance better. While conducting the experiments, different illumination conditions and varied distances between hand sign and camera often occurred. In such cases, confirmed the result is very effective when the proposed method is used. The proposed system is not very expensive. It will be convenient for the user. Hand gesture and speech recognition makes two-way communication possible.

CHAPTER 7
FUTURE SCOPE

The system can be used a sign language learning aid in future so that normal people could understand what the speech and hearing- impaired expresses. The system we implemented is at present powered from wall supply. It can be made portable by including a battery and necessary converter in the design. The system can be installed in public spaces in order to communicate with the disabled people effortlessly. Since the system accurately recognizes hand gestures, in the future the system can be modified to control gesture controlled devices.

REFERENCES

1. Hand Gesture Recognition and Voice Conversion System for Dumb People.

   Vigneshwaran, ShifaFathima, VijaySagar, SNS College of Technology. 2019 International Conference on Advanced Computing & Communication Systems (ICACCS).

2. A belief-based sequential fusion approach for fusing manual signs and non-manual signals.

   Thad Starner, Joshua Weaver, and Alex Pentland. international journal of Pattern Recognition Letters, volume-32, number-4, pages: -572–577,pages 812-822 ,2009.

3. An Effective Approach to Communicate with the Deaf and Mute People by Recognizing Characters of One-hand Bangla Sign Language Using Convolutional Neural-Network.

   Alfat Jahan Rony, Khairul Hossain Saikat, Mahdia Tanzeem and F.M.Rahat Hasan Robi. Department of Computer Science and Engineering Bangabandhu Sheikh Mujibur Rahman Science and Technology University, Bangladesh.

4. Speech-To-Text Conversion (STT) System Using Google API
   Documentation in Google cloud.

APPENDIX-A

A.1 FINAL PROGRAM FOR THE IMPLEMENTED SYSTEM

```
import RPi.GPIO as GPIO
import lcddriver
from time import *
lcd = lcddriver.lcd()
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
lcd.lcd_display_string("      LEXICON", 1)
lcd.lcd_display_string("An Electronic Slate", 2)
lcd.lcd_display_string("   For Deaf-Mute", 3)
sleep(3)
lcd.lcd_clear()
while True:
    button_state=GPIO.input(10)
    if(button_state==0):
        lcd.lcd_clear()
        lcd.lcd_display_string("   Gesture mode", 2)
        lcd.lcd_display_string("     Activated", 3)
        sleep(2)
        lcd.lcd_clear()
        import fun_util
        fun_util.recognize()
    else:
        lcd.lcd_clear()
        lcd.lcd_display_string("    Speech mode", 2)
        lcd.lcd_display_string("     Activated", 3)
        sleep(2)
        lcd.lcd_clear()
        import speech
        speech.rec()
```

A.2 PROGRAM FOR HAND GESTURE RECOGNITION MODULE.

```
import cv2, pickle
import numpy as np
import tensorflow as tf
from cnn_tf import cnn_model_fn
import os
import sqlite3, pyttsx3
from keras.models import load_model
from threading import Thread
engine = pyttsx3.init()
engine.setProperty('rate', 150)
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
model = load_model('cnn_model_keras2.h5')
def get_hand_hist():
	with open("hist", "rb") as f:
		hist = pickle.load(f)
	return hist
def get_image_size():
	img = cv2.imread('gestures/0/100.jpg', 0)
	return img.shape
image_x, image_y = get_image_size()
def keras_process_image(img):
	img = cv2.resize(img, (image_x, image_y))
	img = np.array(img, dtype=np.float32)
	img = np.reshape(img, (1, image_x, image_y, 1))
	return img
def keras_predict(model, image):
	processed = keras_process_image(image)
	pred_probab = model.predict(processed)[0]
	pred_class = list(pred_probab).index(max(pred_probab))
	return max(pred_probab), pred_class
def get_pred_text_from_db(pred_class):
	conn = sqlite3.connect("gesture_db.db")
	cmd = "SELECT g_name FROM gesture WHERE g_id="+str(pred_class)
	cursor = conn.execute(cmd)
	for row in cursor:
		return row[0]
def get_pred_from_contour(contour, thresh):
	x1, y1, w1, h1 = cv2.boundingRect(contour)
	save_img = thresh[y1:y1+h1, x1:x1+w1]
	text = ""
	if w1 > h1:
		save_img = cv2.copyMakeBorder(save_img, int((w1-h1)/2) ,
int((w1-h1)/2) , 0, 0, cv2.BORDER_CONSTANT, (0, 0, 0))
	elif h1 > w1:
		save_img = cv2.copyMakeBorder(save_img, 0, 0, int((h1-w1)/2)
, int((h1-w1)/2) , cv2.BORDER_CONSTANT, (0, 0, 0))
	pred_probab, pred_class = keras_predict(model, save_img)
	if pred_probab*100 > 70:
		text = get_pred_text_from_db(pred_class)
	return text
def get_operator(pred_text):
	try:
		pred_text = int(pred_text)
```

```
        except:
                return ""
        operator = ""
        if pred_text == 1:
                operator = "+"
        elif pred_text == 2:
                operator = "-"
        elif pred_text == 3:
                operator = "*"
        elif pred_text == 4:
                operator = "/"
        elif pred_text == 5:
                operator = "%"
        elif pred_text == 6:
                operator = "**"
        elif pred_text == 7:
                operator = ">>"
        elif pred_text == 8:
                operator = "<<"
        elif pred_text == 9:
                operator = "&"
        elif pred_text == 0:
                operator = "|"
        return operator
hist = get_hand_hist()
x, y, w, h = 300, 100, 300, 300
is_voice_on = True
def get_img_contour_thresh(img):
        img = cv2.flip(img, 1)
        imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
        dst = cv2.calcBackProject([imgHSV], [0, 1], hist, [0, 180, 0, 256],
1)
        disc = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(10,10))
        cv2.filter2D(dst,-1,disc,dst)
        blur = cv2.GaussianBlur(dst, (11,11), 0)
        blur = cv2.medianBlur(blur, 15)
        thresh =
cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
        thresh = cv2.merge((thresh,thresh,thresh))
        thresh = cv2.cvtColor(thresh, cv2.COLOR_BGR2GRAY)
        thresh = thresh[y:y+h, x:x+w]
        contours = cv2.findContours(thresh.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)[1]
        return img, contours, thresh
def say_text(text):
        if not is_voice_on:
                return
        while engine._inLoop:
                pass
        engine.say(text)
        engine.runAndWait()
def calculator_mode(cam):
        global is_voice_on
        flag = {"first": False, "operator": False, "second": False, "clear":
False}
        count_same_frames = 0
```

```
        first, operator, second = "", "", ""
        pred_text = ""
        calc_text = ""
        info = "Enter first number"
        Thread(target=say_text, args=(info,)).start()
        count_clear_frames = 0
        while True:
                img = cam.read()[1]
                img = cv2.resize(img, (640, 480))
                img, contours, thresh = get_img_contour_thresh(img)
                old_pred_text = pred_text
                if len(contours) > 0:
                        contour = max(contours, key = cv2.contourArea)
                        if cv2.contourArea(contour) > 10000:
                                pred_text = get_pred_from_contour(contour,
thresh)
                                if old_pred_text == pred_text:
                                        count_same_frames += 1
                                else:
                                        count_same_frames = 0
                                if pred_text == "C":
                                        if count_same_frames > 5:
                                                count_same_frames = 0
                                                first, second, operator,
pred_text, calc_text = '', '', '', '', ''
                                                flag['first'], flag['operator'],
flag['second'], flag['clear'] = False, False, False, False
                                                info = "Enter first number"
                                                Thread(target=say_text,
args=(info,)).start()
                                elif pred_text == "Best of Luck " and
count_same_frames > 15:
                                        count_same_frames = 0
                                        if flag['clear']:
                                                first, second, operator,
pred_text, calc_text = '', '', '', '', ''
                                                flag['first'], flag['operator'],
flag['second'], flag['clear'] = False, False, False, False
                                                info = "Enter first number"
                                                Thread(target=say_text,
args=(info,)).start()
                                        elif second != '':
                                                flag['second'] = True
                                                info = "Clear screen"
                                                #Thread(target=say_text,
args=(info,)).start()
                                                second = ''
                                                flag['clear'] = True
                                                try:
                                                        calc_text += "=
"+str(eval(calc_text))
                                                except:
                                                        calc_text = "Invalid
operation"
                                                if is_voice_on:
```

```
                                                speech = calc_text
                                                speech =
speech.replace('-', ' minus ')
                                                speech =
speech.replace('/', ' divided by ')
                                                speech =
speech.replace('**', ' raised to the power ')
                                                speech =
speech.replace('*', ' multiplied by ')
                                                speech =
speech.replace('%', ' mod ')
                                                speech =
speech.replace('>>', ' bitwise right shift ')
                                                speech =
speech.replace('<<', ' bitwise leftt shift ')
                                                speech =
speech.replace('&', ' bitwise and ')
                                                speech =
speech.replace('|', ' bitwise or ')
                                                Thread(target=say_text,
args=(speech,)).start()
                                    elif first != '':
                                            flag['first'] = True
                                            info = "Enter operator"
                                            Thread(target=say_text,
args=(info,)).start()
                                            first = ''
                                elif pred_text != "Best of Luck " and
pred_text.isnumeric():
                                    if flag['first'] == False:
                                            if count_same_frames > 15:
                                                    count_same_frames = 0
                                                    Thread(target=say_text,
args=(pred_text,)).start()
                                                    first += pred_text
                                                    calc_text += pred_text
                                    elif flag['operator'] == False:
                                            operator =
get_operator(pred_text)
                                            if count_same_frames > 15:
                                                    count_same_frames = 0
                                                    flag['operator'] = True
                                                    calc_text += operator
                                                    info = "Enter second
number"
                                                    Thread(target=say_text,
args=(info,)).start()
                                                    operator = ''
                                    elif flag['second'] == False:
                                            if count_same_frames > 15:
                                                    Thread(target=say_text,
args=(pred_text,)).start()
                                                    second += pred_text
                                                    calc_text += pred_text
                                                    count_same_frames = 0
            if count_clear_frames == 30:
```

```
                        first, second, operator, pred_text, calc_text = '',
'', '', '', ''
                        flag['first'], flag['operator'], flag['second'],
flag['clear'] = False, False, False, False
                        info = "Enter first number"
                        Thread(target=say_text, args=(info,)).start()
                        count_clear_frames = 0
                blackboard = np.zeros((480, 640, 3), dtype=np.uint8)
                cv2.putText(blackboard, "Calculator Mode", (100, 50),
cv2.FONT_HERSHEY_TRIPLEX, 1.5, (255, 0,0))
                cv2.putText(blackboard, "Predicted text- " + pred_text, (30,
100), cv2.FONT_HERSHEY_TRIPLEX, 1, (255, 255, 0))
                cv2.putText(blackboard, "Operator " + operator, (30, 140),
cv2.FONT_HERSHEY_TRIPLEX, 1, (255, 255, 127))
                cv2.putText(blackboard, calc_text, (30, 240),
cv2.FONT_HERSHEY_TRIPLEX, 2, (255, 255, 255))
                cv2.putText(blackboard, info, (30, 440),
cv2.FONT_HERSHEY_TRIPLEX, 1, (0, 255, 255) )
                if is_voice_on:
                        cv2.putText(blackboard, " ", (450, 440),
cv2.FONT_HERSHEY_TRIPLEX, 1, (255, 127, 0))
                else:
                        cv2.putText(blackboard, " ", (450, 440),
cv2.FONT_HERSHEY_TRIPLEX, 1, (255, 127, 0))
                cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)
                res = np.hstack((img, blackboard))
                cv2.imshow("Recognizing gesture", res)
                cv2.imshow("thresh", thresh)
                keypress = cv2.waitKey(1)
                if keypress == ord('q') or keypress == ord('t'):
                        break
                if keypress == ord('v') and is_voice_on:
                        is_voice_on = False
                elif keypress == ord('v') and not is_voice_on:
                        is_voice_on = True
        if keypress == ord('t'):
                return 1
        else:
                return 0
def text_mode(cam):
        global is_voice_on
        text = ""
        word = ""
        count_same_frame = 0
        while True:
                img = cam.read()[1]
                img = cv2.resize(img, (640, 480))
                img, contours, thresh = get_img_contour_thresh(img)
                old_text = text
                if len(contours) > 0:
                        contour = max(contours, key = cv2.contourArea)
                        if cv2.contourArea(contour) > 10000:
                                text = get_pred_from_contour(contour, thresh)
                                if old_text == text:
                                        count_same_frame += 1
```

```
                                else:
                                        count_same_frame = 0
                                if count_same_frame > 20:
                                        if len(text) == 1:
                                                Thread(target=say_text,
args=(text, )).start()
                                        word = word + text
                                        if word.startswith('I/Me '):
                                                word = word.replace('I/Me ', 'I
')
                                        elif word.endswith('I/Me '):
                                                word = word.replace('I/Me ', 'me
')
                                        count_same_frame = 0
                        elif cv2.contourArea(contour) < 1000:
                                if word != '':
                                        #print('yolo')
                                        #say_text(text)
                                        Thread(target=say_text, args=(word,
)).start()
                                text = ""
                                word = ""
                else:
                        if word != '':
                                #print('yolo1')
                                #say_text(text)
                                Thread(target=say_text, args=(word, )).start()
                        text = ""
                        word = ""
                blackboard = np.zeros((480, 640, 3), dtype=np.uint8)
                cv2.putText(blackboard, " ", (180, 50),
cv2.FONT_HERSHEY_TRIPLEX, 1.5, (255, 0,0))
                cv2.putText(blackboard, "Predicted text- " + text, (30,
100), cv2.FONT_HERSHEY_TRIPLEX, 1, (255, 255, 0))
                cv2.putText(blackboard, word, (30, 240),
cv2.FONT_HERSHEY_TRIPLEX, 2, (255, 255, 255))
                if is_voice_on:
                        cv2.putText(blackboard, " ", (450, 440),
cv2.FONT_HERSHEY_TRIPLEX, 1, (255, 127, 0))
                else:
                        cv2.putText(blackboard, " ", (450, 440),
cv2.FONT_HERSHEY_TRIPLEX, 1, (255, 127, 0))
                cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)
                res = np.hstack((img, blackboard))
                cv2.imshow("Recognizing gesture", res)
                cv2.imshow("thresh", thresh)
                keypress = cv2.waitKey(1)
                if keypress == ord('q') or keypress == ord('c'):
                        break
                if keypress == ord('v') and is_voice_on:
                        is_voice_on = False
                elif keypress == ord('v') and not is_voice_on:
                        is_voice_on = True
        if keypress == ord('c'):
                return 2
        else:
```

```
                return 0
def recognize():
        cam = cv2.VideoCapture(1)
        if cam.read()[0]==False:
                cam = cv2.VideoCapture(0)
        text = ""
        word = ""
        count_same_frame = 0
        keypress = 1
        while True:
                if keypress == 1:
                        keypress = text_mode(cam)
                elif keypress == 2:
                        keypress = calculator_mode(cam)
                else:
                        break
keras_predict(model, np.zeros((50, 50), dtype = np.uint8))
recognize()
```

A.3 PROGRAM FOR SPEECH TO TEXT CONVERSION

```
import speech_recognition as sr
import lcddriver
from time import *
lcd = lcddriver.lcd()

def rec():
    try:
        r = sr.Recognizer()
        with sr.Microphone() as source:
            audio_data = r.listen(source)
            text = r.recognize_google(audio_data)
            lcd.lcd_display_string(str(text), 2)
    except:
        pass
```