INDIAN SIGN LANGUAGE TO SPEECH

PROJECT REPORT

submitted by

Martin Joseph Juneie

MUT20CS086

to

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

of

Bachelor of Technology

In

Computer Science & Engineering

Department of Computer Science & Engineering

Muthoot Institute of Technology and Science

Varikoli PO, Puthencruz - 682308

JULY 2023

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Martin Joseph Juneie

MUT20CS086

Place: Muthoot Institute Of Technology and Science, Kochi
Date:26/06/2023

Muthoot
Institute of Technology & Science

CERTIFICATE

This is to certify that the report entitled **"INDIAN SIGN LANGUAGE TO SPEECH"**, *submitted by* **Martin Joseph Juneie** *to Muthoot Institute of Technology and Science, Varikoli for the award of the degree of Bachelor of Technology in Computer Science & Engineering is a bonafide record of the project work carried out by her, under our supervision and guidance. The content of the report, in full or parts have not been submitted to any other Institute or University for the award of any other degree or diploma.*

Mrs.Haritha H.	Mrs.Sheena K.V.	Dr. Anand Hareendran S.
Project Guide	Project Coordinator	Head of the Department

Place: Muthoot Institute Of Technology and Science, Kochi

Date: 26/06/2023

ACKNOWLEDGMENT

I am grateful to almighty who has blessed me with good health, committed and continuous interest throughout the project work.

I express my sincere thanks to my guide, **Mrs. Haritha H.**, Assistant Professor, Department of Computer Science And Engineering, Muthoot Institute of Technology and Science and **Dr. Anand Hareendran S.**, Professor, Head Of the Department, Muthoot Institute of Technology and Science for their guidance and support which were instrumental in all the stages of the project work and without whom the project could not have been accomplished.

In particular, I also wish to express my sincere appreciation to **Dr. Anand Hareendran S.**, Head Of the Department, Muthoot Institute of Technology and Science, who was willing to spend his precious time to give some ideas and suggestion towards this project.

I am grateful to my project coordinator **Mrs.Sheena K.V.** Assistant Professor, Department of Computer Science And Engineering, Muthoot Institute of Technology and Science, for her guidance and support.

I would like to thank **Dr. Neelakantan P.C.**, Principal, Muthoot Institute of Technology and Science, Varikoli for providing us all the necessary facilities.

The last but not the least, I extend my sincere thanks to the entire teaching and non-teaching staff of Computer Science And Engineering of Muthoot Institute of Technology and Science for their help and co-operation throughout our project work.

ABSTRACT

Humans interact with each other to convey their ideas, thoughts, and experiences to people using speech, but this is not possible for deaf and mute people. They use sign language for communication, but most people are not familiar with sign language. There are existing systems that convert American Sign Language to text and speech. However, there is no well-established software that converts Indian Sign Language to speech. Our aim is to build a pragmatic solution using Convolutional Neural Network (CNN) that enables people to understand Indian Sign Language, thereby reducing the communication gap between people. Compared to other gestures (arm, face, head, and body), hand gestures play an important role, as it expresses the user's views in less time. Ultimately, a successful ISL-to-speech conversion system requires a combination of technical solutions and human expertise.

METHODOLOGY

1. **Creating dataset for training and testing.**

   Capturing hand images. Applying hand landmarks. Defining a region of interest.

2. **Using Convolutional Neural Network (CNN) for building a training model.**

   Dataset is processed with Teachable Machine to output a trained model.

3. **Creating a GUI to convert signs into text.**

   UI shows the recognised sign as an overlay over the camera feed, at the same time outputting the symbol converted to voice.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Indian Sign Language (ISL) to Speech project is an innovative endeavor aimed at bridging the communication gap between the deaf and hearing communities in India. Indian Sign Language is a visual-gestural language used by the deaf and mute community to convey their thoughts, ideas, and emotions. However, due to limited familiarity with ISL among the hearing population, effective communication between these two communities becomes challenging.The goal of this project is to develop a robust and accurate system that can convert Indian Sign Language gestures into spoken language, thus enabling deaf individuals to communicate more effectively with the hearing world.

1.2 SCOPE AND MOTIVATION

Communication is the process of exchange of thoughts and messages in various ways such as speech, signals, behavior and visuals.

Deaf and mute people make use of their hands to express different gestures to express their ideas with other people.

Gestures are the nonverbally exchanged messages and these gestures are understood with vision. This nonverbal communication of deaf and mute people is called sign language.

In this project we are focusing on converting Indian Sign Language to speech.

CHAPTER 2

PROPOSED WORK

2.1 OBJECTIVES

1. Bridging the communication gap: The primary objective is to bridge the communication gap between the deaf and hearing communities in India. By converting ISL gestures into spoken language, the project aims to enable effective communication between deaf individuals and the hearing world.

2. Enhancing social inclusion: The project aims to promote the social inclusion of deaf individuals by providing them with a means to communicate with the larger hearing community. Effective communication can lead to better interactions, understanding, and inclusion in various social and professional settings.

3. Facilitating education and employment opportunities: By enabling deaf individuals to communicate more effectively, the project aims to enhance their access to education and employment opportunities. Clear communication can facilitate learning, participation in classroom discussions, and interaction with colleagues in the workplace.

4. Developing assistive technologies: The project aims to contribute to the development of assistive technologies for individuals with hearing impairments. The ISL to speech conversion system can serve as a foundation for creating educational resources, communication tools, and other assistive technologies to support the needs of deaf individuals.

5. Improving quality of life: By enabling deaf individuals to express themselves and be understood more easily, the project seeks to enhance their overall quality of life. Effective communication can lead to increased self-confidence, improved mental well-being, and a sense of belonging within the broader community.

6. Promoting research and innovation: The project aims to foster research and innovation in the field of sign language recognition and conversion. By exploring advanced technologies such as computer vision, machine learning, and natural language process-

ing, the project contributes to the advancement of these areas and promotes further research in the domain of ISL to speech conversion.

2.2 PROBLEM STATEMENT

To develop a communication approach for healthy people to understand sign language used by hearing impaired people with ease.

2.3 EXISTING SYSTEM AND PROPOSED SOLUTION

2.3.1 EXISTING SYSTEM

Currently there exists sign language to speech converters only for American Sign Language which is able to only solve a subset of the sign language conversion problem. There exists a plethora of sign languages prominent in various regions. Existing systems implements finger spelling translators, however, sign languages are also spoken in a contextual basis where each gesture could represent an object or verb.
Other issues include lack of detection accuracy because of the system not being trained with enough samples.

2.3.2 PROPOSED SYSTEM

1. **Creating dataset for training and testing.**
   Capturing hand images. Applying hand landmarks. Defining a region of interest.

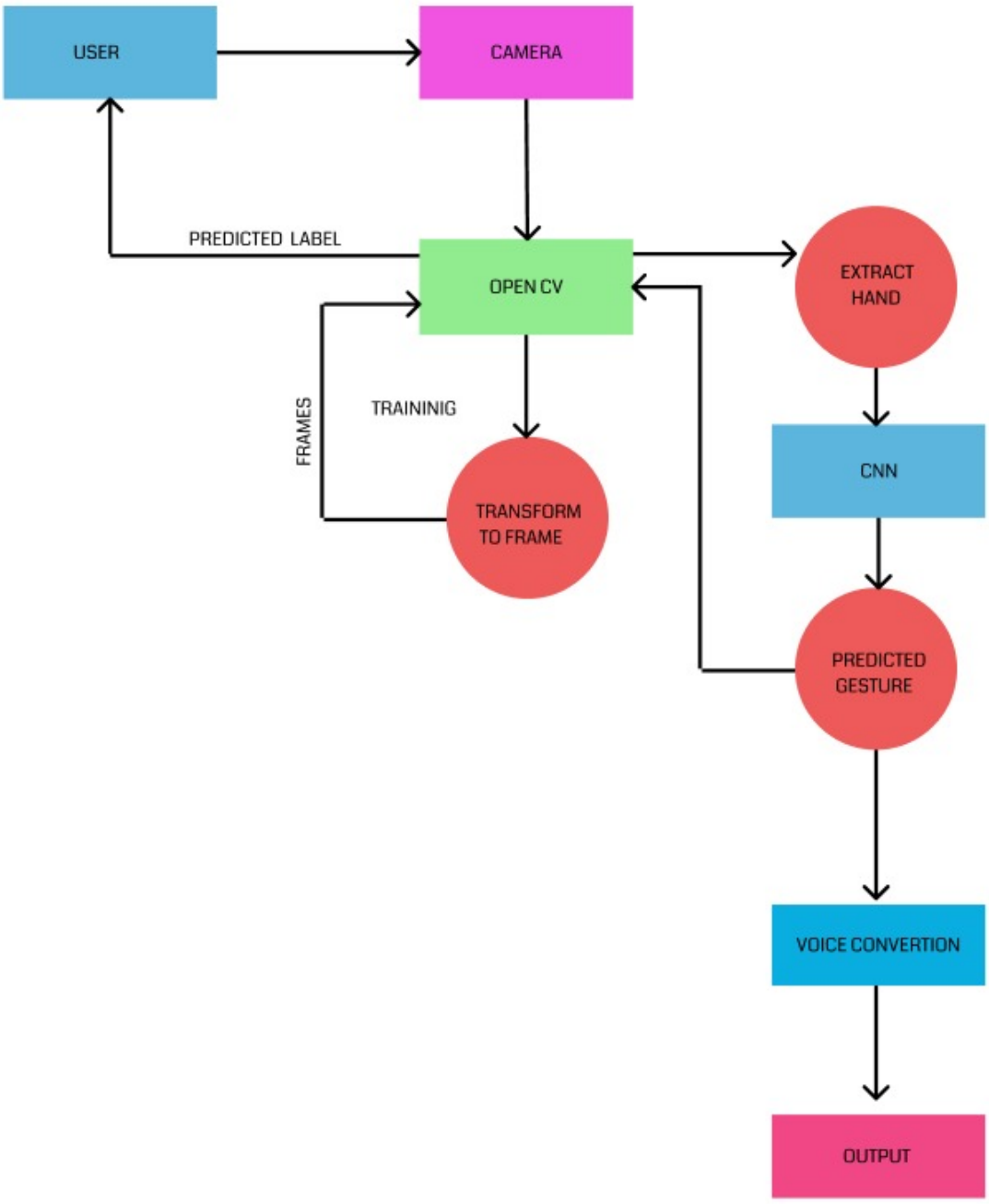
2. **Using Convolutional Neural Network (CNN) for building a training model.**
   Dataset is processed with Teachable Machine to output a trained model.

3. **Creating a GUI to convert signs into text.**
   UI shows the recognised sign as an overlay over the camera feed, at the same time outputting the symbol converted to voice.

CHAPTER 3

PROJECT DESIGN

3.1 SYSTEM ARCHITECTURE



3.2 MODULES

3.2.1 First Module

Image Acquisition

Install OpenCV, Import OpenCV

Initialize the Camera

Add hand landmarks : Add landmarks using Handtracking module.

Crop image : Crop the image with respect to the boundary.

Capture Frames:Capture image by clicking of a button.

Release the Camera.

3.2.2 Second Module

Letters detection

Live feed is obtained using cv2.

Hand is detected using Hand TrackingModule.

Hand landmarks are added.

Image is cropped and converted to greyscale.

Frames are run through a pretrained model and prediction is obtained.

Predicted word is displayed as overlay on hand.

3.2.3 Third Module

Words detection

Live feed is obtained using cv2..

Hand is detected using Hand TrackingModule.

Hand landmarks are added.

Image is cropped and converted to greyscale.

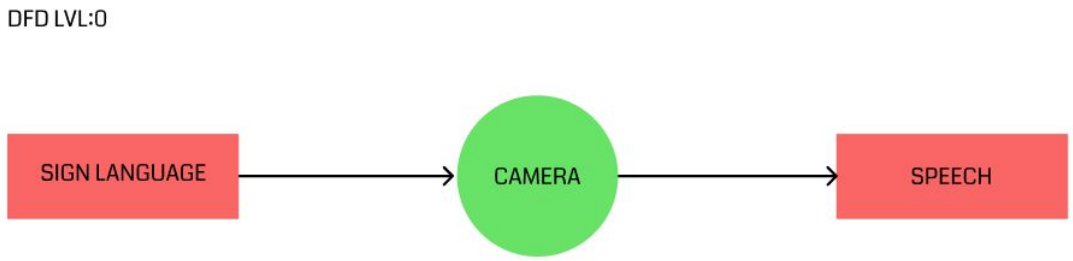
Frames are run through a pretrained model and prediction is obtained.

Predicted word is displayed as overlay on hand.

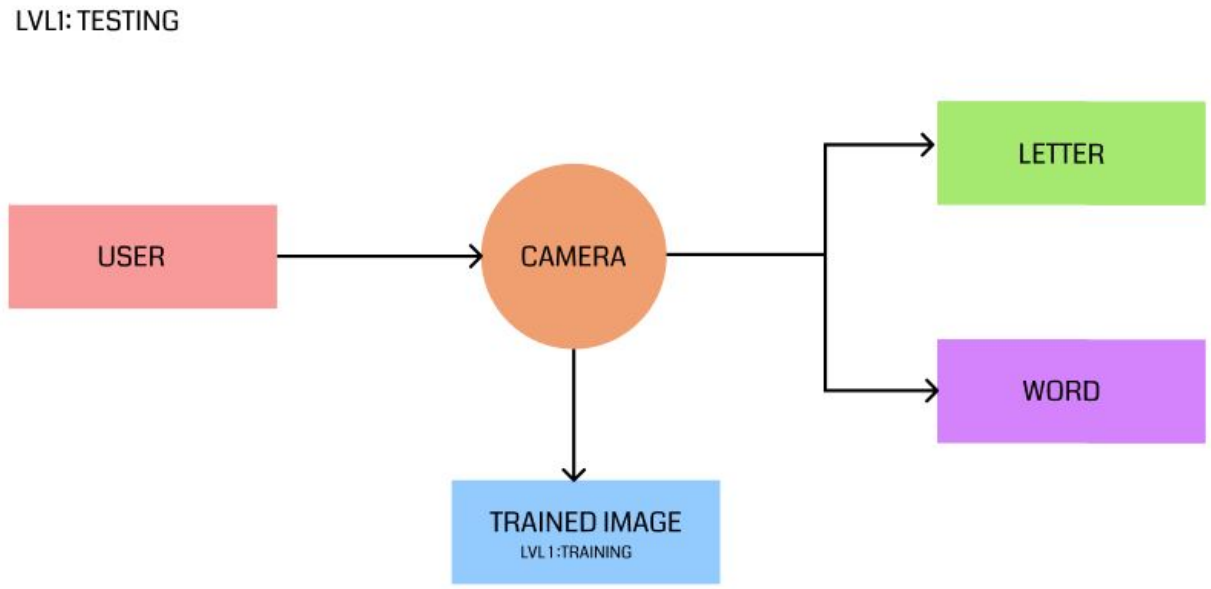
The prediction is outputted as audio using pyttsx3.

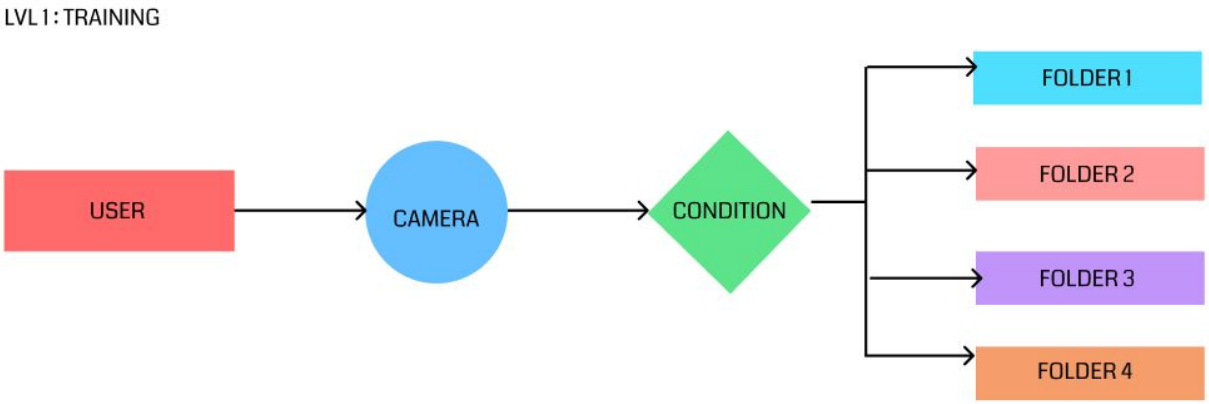
3.3 DATA FLOW DIAGRAM

3.3.1 DFD LEVEL 0

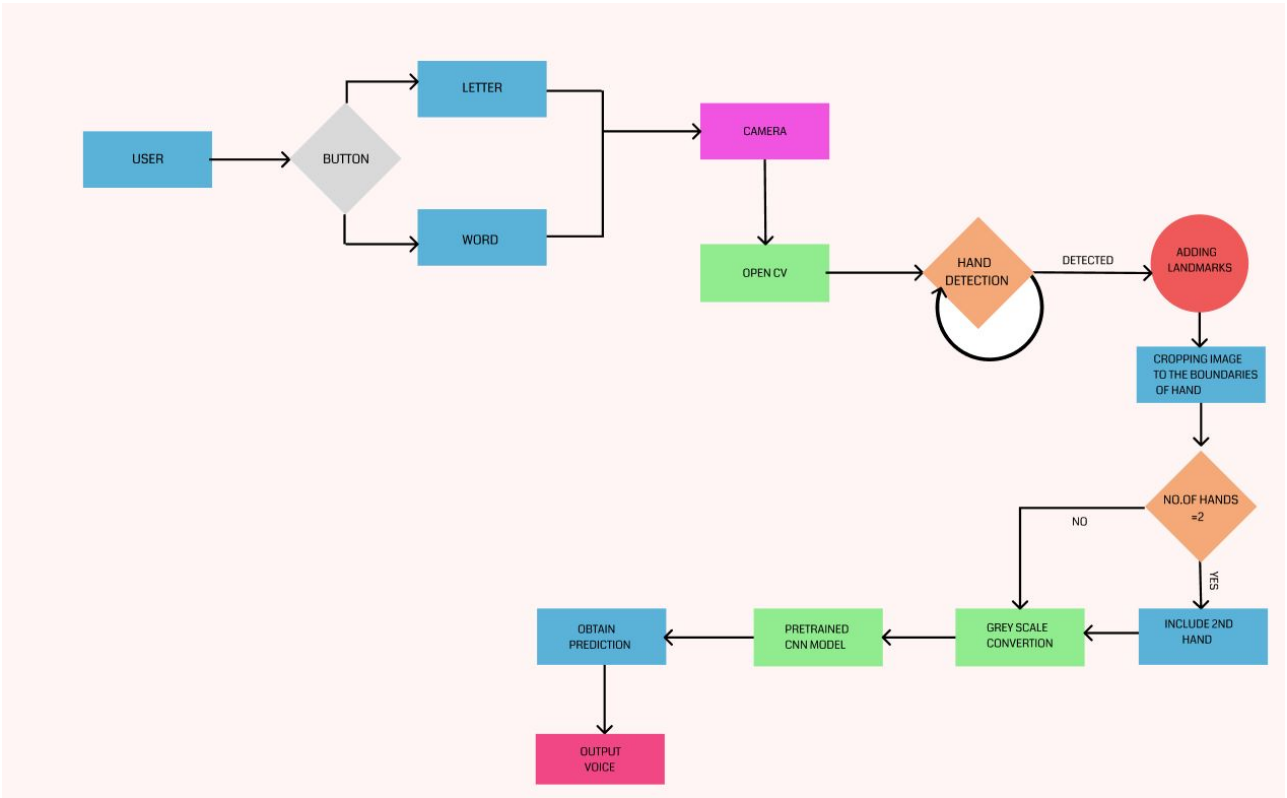


3.3.2 DFD LEVEL 1

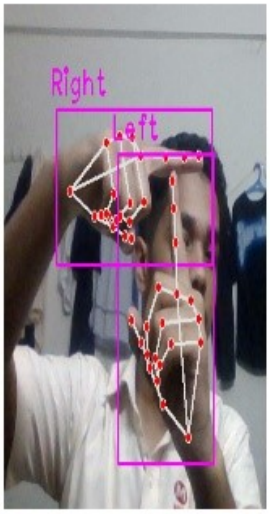


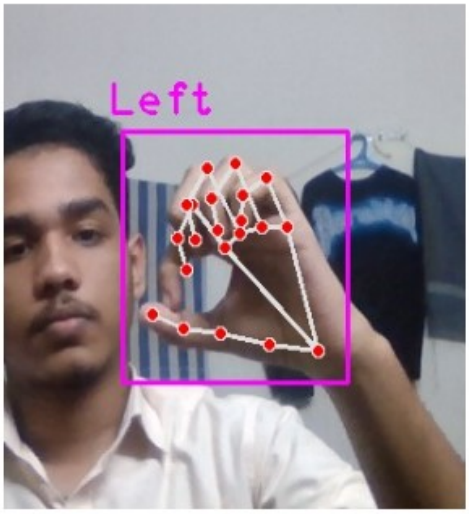


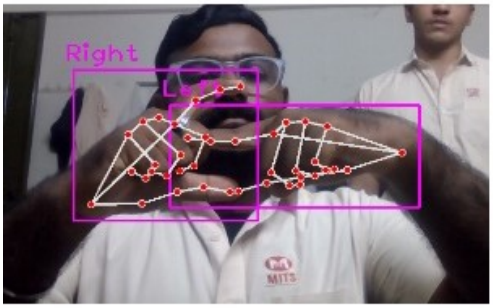
3.3.3 DFD LEVEL 2

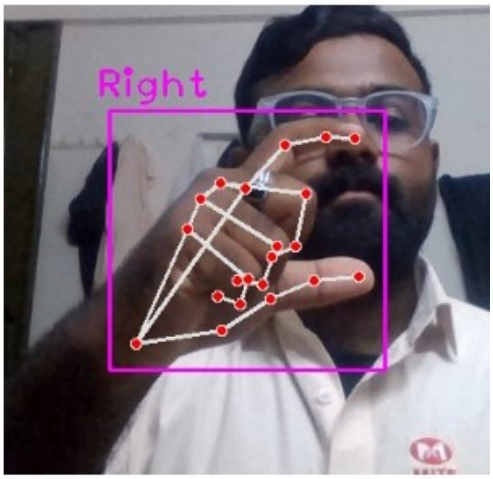


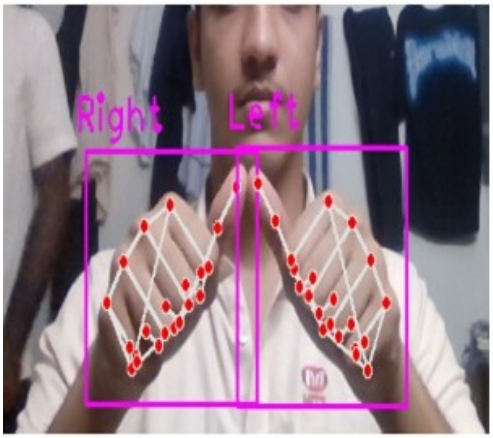
3.4 DATASET CREATION

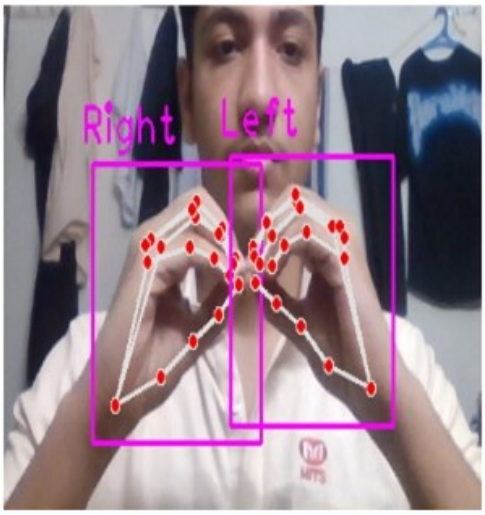


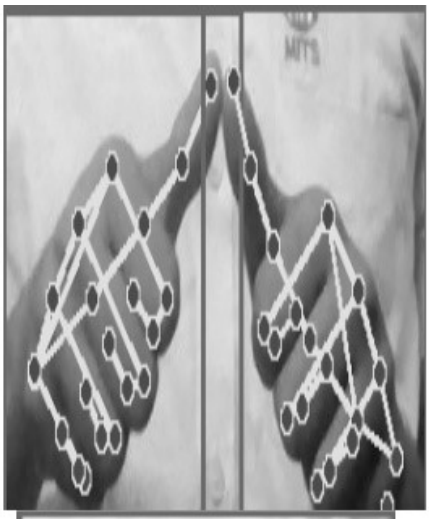


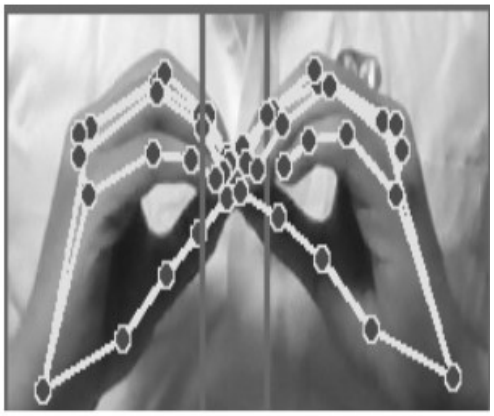




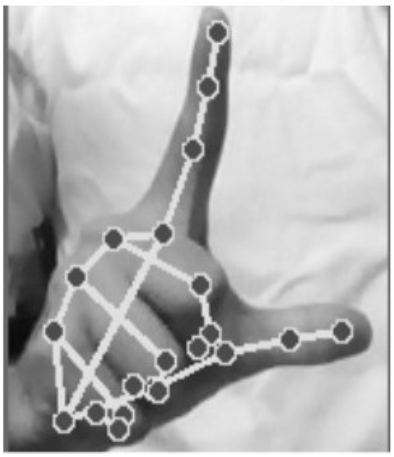


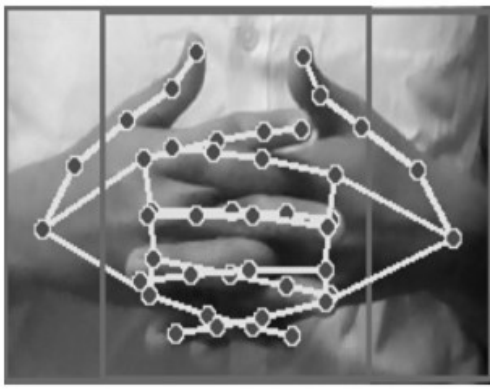


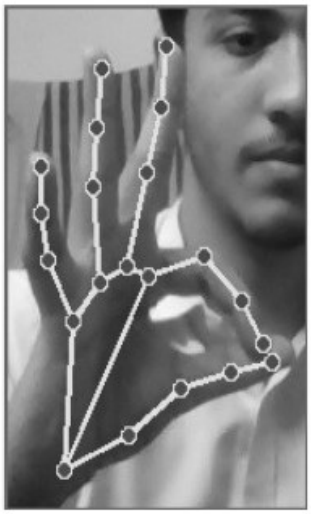


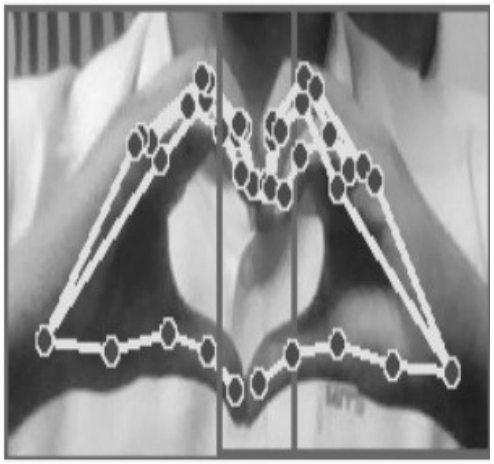


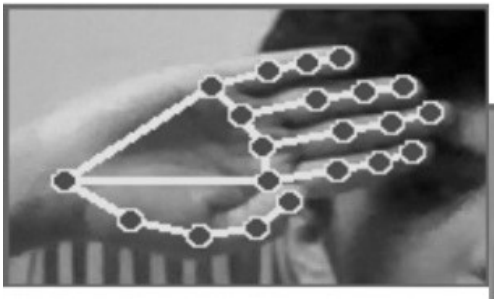

right

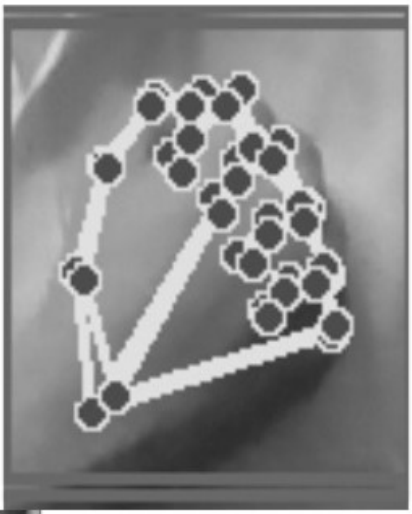












3.5 GUI DESIGN

The GUI makes use of HTML, CSS, JavaScript and Flask. Using HTML a webpage is created that defines the layout of HTML elements. There are three buttons- Letters, Words and Clear. The website is visually enhanced by the CSS which includes colors, fonts, margins etc. CSS design elements make the webpage more visually appealing. JavaScript specifies the functions to be performed when buttons are clicked. When a button is clicked, JavaScript routes us to the backend Python code using Flask microframework. On running the Python code, the camera pop up appears for detecting hand gestures.

3.6 TECHNOLOGY STACK

Convolutional Neural Networks (CNN): Convolutional Neural Networks are a class of deep learning models specifically designed for processing and analyzing visual data, such as images. In your sign language to speech conversion project, CNNs can be employed for various tasks, such as sign language recognition, hand gesture detection, or image classification.

1.Training a CNN Model: To recognize sign language gestures, you would typically need a large dataset of annotated sign language images. Using this dataset, you can train a CNN model. The CNN architecture typically consists of convolutional layers, pooling layers, and fully connected layers. The convolutional layers extract relevant features from the input images, while the fully connected layers perform classification based on these extracted features.

2.Preprocessing Images: Before inputting the sign language images into the CNN model, it's common to preprocess them. Preprocessing steps may include resizing the images to a consistent size, normalizing pixel values, or applying data augmentation techniques to increase the diversity of the training data.

3.Feature Extraction: CNNs can extract meaningful features from the sign language images. The early convolutional layers learn low-level features such as edges and textures, while deeper layers learn more complex and abstract features. By using multiple layers, CNNs can effectively capture the hierarchical representations of the sign lan-

guage gestures.

4.Training and Optimization: During training, the CNN model is presented with batches of sign language images, and the model's parameters are adjusted iteratively to minimize the difference between the predicted output and the ground truth labels. This process is typically done using optimization algorithms, such as stochastic gradient descent (SGD) or Adam, and the training is performed over multiple epochs.

HTML (HyperText Markup Language): HTML is the standard markup language for creating the structure and content of web pages. It provides a set of tags or elements that define the structure and presentation of the web page. In your project, HTML is used for creating the front-end user interface. It allows you to define the layout, structure, and content of the web pages that users interact with.

CSS (Cascading Style Sheets): CSS is a style sheet language used for describing the visual presentation of a document written in HTML. It is responsible for the styling and layout of HTML elements on the web page. CSS is used to enhance the visual appearance of the user interface created with HTML.

CSS works by selecting HTML elements and applying styles to them. Styles can define properties like colors, fonts, margins, paddings, positioning, and more. By using CSS, you can control the layout, colors, typography, and other visual aspects of the web page. CSS rules consist of a selector and a declaration block. The selector specifies which HTML elements the styles should be applied to, and the declaration block contains the actual styles.

JavaScript (JS): JavaScript is a scripting language that enables interactivity and dynamic behavior in web pages. In your project, JavaScript plays a vital role in enhancing the functionality and user experience of the sign language to speech conversion application.

With JavaScript,manipulate the HTML structure, interact with the user, and perform client-side validation. It allows you to respond to user actions, such as button clicks or form submissions, and update the web page dynamically without reloading the entire page.

Flask: Flask is a micro web framework written in Python. It provides a simple and efficient way to build web applications. Flask is used for connecting the front-end

HTML/CSS user interface with the back-end Python code that handles the business logic and data processing.

Flask acts as a bridge between the front-end and the back-end. It receives requests from the user interface, processes them, and returns responses accordingly. Flask provides routing capabilities, allowing you to define URL routes and associate them with specific Python functions, known as view functions. These view functions handle the logic for processing requests, interacting with the database, and returning appropriate responses.

Flask also supports the use of templates, such as Jinja2 templates, which allow you to dynamically generate HTML content by embedding Python code within the HTML files. This enables you to generate dynamic web pages based on data retrieved from the back-end.

Flask can handle form submissions, manage user sessions, and interact with databases, among other functionalities. It provides a lightweight and flexible framework for building web applications in Python.

3.7 SYSTEM REQUIREMENTS

3.7.1 Hardware Requirements

The system can run only on computers, as mobile devices have very limited computational power,and required packeges are not available for mobile platforms.
To recognize signs the system requires a camera interface or image sensor available on the host device.
A minimum of 8 GB ram will be required for all the computation to be performed.

3.7.2 Software Requirements

Python-3.7

OpenCV -4.5.4.60

CVZone -1.5.6

MediaPipe-0.8.10.1

TensorFlow-2.9.1

NumPy-1.21.6

Flask2.2.5

pyttsx3-2.90

gTTS2.3.2

CHAPTER 4

IMPLEMENTATION

4.1 CODE SNIPPETS

4.1.1 dataAcquisition.py

```
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
import time

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=2)

offset = 20
imgSize = 300

folder="DataSet/DataLetters/A"
counter=0

while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
    if hands:
        hand1 = hands[0]
        x1, y1, w1, h1 = hand1['bbox']
        h=h1
        w=w1
        imgCrop = img[y1 - offset:y1 + h1 + offset, x1 - offset:x1 + w1 + offset]

        if len(hands) == 2:
            hand2 = hands[1]
            x2, y2, w2, h2 = hand2['bbox']

            if y1 > y2:
                yb = y2
```

```
        yt = y1
    else:
        yb = y1
        yt = y2

    if h1 > h2:
        h = h1
    else:
        h = h2

    firstHandType = hand2["type"]

    if firstHandType == "Left":
        xl = x1
        xr = x2
        w = w2
    else:
        xl = x2
        xr = x1
        w = w1

    imgCrop = img[yb - offset: yt + h + offset, xl - offset: xr + w + offset]

imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
impgCropShape =imgCrop.shape

aspectRatio = h/w

if aspectRatio > 1:
    k = imgSize / h
    wCal = math.ceil(k * w)
```

```
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap: wCal + wGap] = imgResize

        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, : ] = imgResize

        cv2.imshow("ImageWhite", imgWhite)
        grayscale = cv2.cvtColor(imgWhite, cv2.COLOR_BGR2GRAY)
        cv2.imshow("Gray", grayscale)

    cv2.imshow("Image", img)
    key= cv2.waitKey(1)
    if key == ord("s") or key == ord("S"):
        counter += 1
        cv2.imwrite(f'{folder}/Image_{time.time()}.jpg',grayscale)
        print (counter)
```

4.1.2 letters.py

```
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import numpy as np
import math

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=2)
classifier = Classifier("ModelLetter/keras_model.h5", "ModelLetter/labels.txt")

offset = 20
imgSize = 128

labels = ["B", "C", "L", "O", "T", "V", "W", "A"]

while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
    cv2.waitKey(1)

    if hands:
        hand1 = hands[0]
        x1, y1, w1, h1 = hand1['bbox']
        h = h1
        w = w1
        imgCrop = img[y1 - offset:y1 + h1 + offset, x1 - offset:x1 + w1 + offset]

        if len(hands) == 2:
            hand2 = hands[1]
            x2, y2, w2, h2 = hand2['bbox']
```

```
    if y1 > y2:
        yb = y2
        yt = y1
    else:
        yb = y1
        yt = y2

    if h1 > h2:
        h = h1
    else:
        h = h2

    firstHandType = hand2["type"]

    if firstHandType == "Left":
        xl = x1
        xr = x2
        w = w2
    else:
        xl = x2
        xr = x1
        w = w1

    imgCrop = img[yb - offset: yt + h + offset, xl - offset: xr + w + offset]

imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
impgCropShape = imgCrop.shape

aspectRatio = h / w

if aspectRatio > 1:
```

```
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap: wCal + wGap] = imgResize
            grayscale = cv2.cvtColor(imgWhite, cv2.COLOR_BGR2GRAY)
            imgColor = cv2.cvtColor(grayscale, cv2.COLOR_GRAY2BGR)  # Convert grayscale to color
        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize
            grayscale = cv2.cvtColor(imgWhite, cv2.COLOR_BGR2GRAY)
            imgColor = cv2.cvtColor(grayscale, cv2.COLOR_GRAY2BGR)  # Convert grayscale to color

        cv2.imshow("Gray", imgColor)
        prediction, index = classifier.getPrediction(imgColor, draw=False)  # Use the color image for
prediction
        cv2.putText(img, labels[index], (100, 100), cv2.FONT_HERSHEY_COMPLEX, 2, (255, 0, 255), 2)

        cv2.imshow("Image", img)
        cv2.waitKey(1)

        print(prediction, labels[index])
```

4.1.3 words.py

```
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import numpy as np
import math
import pyttsx3

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=2)
classifier = Classifier("ModelWord/keras_model.h5", "ModelWord/labels.txt")
engine = pyttsx3.init()

offset = 20
imgSize = 128

labels = ["Bird", "Flower", "Good", "Love", "Salute", "Sorry", "Thank You"]

while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
    cv2.waitKey(1)

    if hands:
        hand1 = hands[0]
        x1, y1, w1, h1 = hand1['bbox']
        h = h1
        w = w1
        imgCrop = img[y1 - offset:y1 + h1 + offset, x1 - offset:x1 + w1 + offset]

        if len(hands) == 2:
            hand2 = hands[1]
```

```
    x2, y2, w2, h2 = hand2['bbox']

    if y1 > y2:
        yb = y2
        yt = y1
    else:
        yb = y1
        yt = y2

    if h1 > h2:
        h = h1
    else:
        h = h2

    firstHandType = hand2["type"]

    if firstHandType == "Left":
        xl = x1
        xr = x2
        w = w2
    else:
        xl = x2
        xr = x1
        w = w1

    imgCrop = img[yb - offset: yt + h + offset, xl - offset: xr + w + offset]

imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
impgCropShape = imgCrop.shape

aspectRatio = h / w
```

```
if aspectRatio > 1:
    k = imgSize / h
    wCal = math.ceil(k * w)
    imgResize = cv2.resize(imgCrop, (wCal, imgSize))
    imgResizeShape = imgResize.shape
    wGap = math.ceil((imgSize - wCal) / 2)
    imgWhite[:, wGap: wCal + wGap] = imgResize
    grayscale = cv2.cvtColor(imgWhite, cv2.COLOR_BGR2GRAY)
    imgColor = cv2.cvtColor(grayscale, cv2.COLOR_GRAY2BGR)  # Convert grayscale to color
else:
    k = imgSize / w
    hCal = math.ceil(k * h)
    imgResize = cv2.resize(imgCrop, (imgSize, hCal))
    imgResizeShape = imgResize.shape
    hGap = math.ceil((imgSize - hCal) / 2)
    imgWhite[hGap:hCal + hGap, :] = imgResize
    grayscale = cv2.cvtColor(imgWhite, cv2.COLOR_BGR2GRAY)
    imgColor = cv2.cvtColor(grayscale, cv2.COLOR_GRAY2BGR)

cv2.imshow("Gray", imgColor)
prediction, index = classifier.getPrediction(imgColor, draw=False)
cv2.putText(img, labels[index], (100, 100), cv2.FONT_HERSHEY_COMPLEX, 2, (255, 0, 255), 2)

cv2.imshow("Image", img)
cv2.waitKey(1)

spoken_word = labels[index]
print(prediction, spoken_word)

engine.say(spoken_word)
```

```
engine.runAndWait()


print(prediction, labels[index])
```

4.1.4 app.py

```
from flask import Flask, render_template
import subprocess
import numpy as np
import sys
import gtts
from playsound import playsound


app = Flask(__name__)

letter_process = None
words_process = None


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/run_letter')
def run_letter():
    t1 = gtts.gTTS("Place your hand towards the camera module.")
    soundName=str(np.random.randint(0,100))+".mp3"
    t1.save(soundName)
    playsound(soundName)
    global letter_process
    if letter_process is None or letter_process.poll() is not None:
        letter_process = subprocess.Popen([sys.executable, 'letters.py'])
        return 'Letter.py is running'
    else:
        return 'Letter.py is already running'
```

```
@app.route('/run_words')
def run_words():
    global words_process
    if words_process is None or words_process.poll() is not None:
        words_process = subprocess.Popen([sys.executable, 'words.py'])
        return 'Words.py is running'
    else:
        return 'Words.py is already running'


@app.route('/clear_program')
def clear_program():
    global letter_process, words_process
    if letter_process is not None:
        letter_process.kill()
        letter_process = None
    if words_process is not None:
        words_process.kill()
        words_process = None
    return 'Program terminated'


if __name__ == '__main__':
    app.run()
```

4.1.5 index.py

```
<html>
<head>

        <title>ISL TO SPEECH</title>

        <style>
    *{
        margin:0;
        padding :0;
        font-family:sans-serif;
}
.banner{
        width:100%;
        height:100vh;
        background-image: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)),
url("https://e1.pxfuel.com/desktop-wallpaper/50/882/desktop-wallpaper-signal-black-background-
wave-neon-for-signal.jpg");
        background-size:cover;
        background-postion:center;

}
.content{
        width:100%;
        position:absolute;
        top:50%;
        transform:translateY(-50%);
        text-align: center;
        color:A9A9A9;
}
.content h1{
        font-size: 70px;
```

```
        margin-top: 80px;

}
.content p{
        margin: 20px auto;
        font-weight: 100;
        line-height: 25px;
}
button{
        width:200px;
        padding:15px 0;
        text-align: center;
        margin:20px 10px;
        border-radius:25px;
        font-weight: bold;
        border:2px solid #009688;
        background: transparent;
        color:#fff;
        cursor:pointer;
        position:relative;
        overflow:hidden;
}
span{
        background:#009688;
        height:100%;
        width:0;
        border-radius:25px;
        position:absolute;
        left:0;
        bottom:0;
        z-index:-1;
```

```
        transiton:0.5s;
}
button:hover span{
        width:100%;



}
button:hover{
        border:none;
}
  </style>



</head>



<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<body>
        <div class="banner">
                <div class="content">
                        <h1>ISL TO SPEECH</h1>
                        <div>
                                <button type="button" onClick =
"runLetter()"><span></span>LETTERS</button>
                                <button type="button" onClick =
"runWords()"><span></span>WORDS</button>
                                <button type="button" onClick = "clearProgram()"
><span></span>CLEAR</button>
        </div>
 <script>
     function runLetter() {
```

```
        fetch('/run_letter');
    }

    function runWords() {
        fetch('/run_words');
    }
    function clearProgram() {
        fetch('/clear_program');
    }
</script>
</body>
</html>
```

4.2 SCREENSHOTS

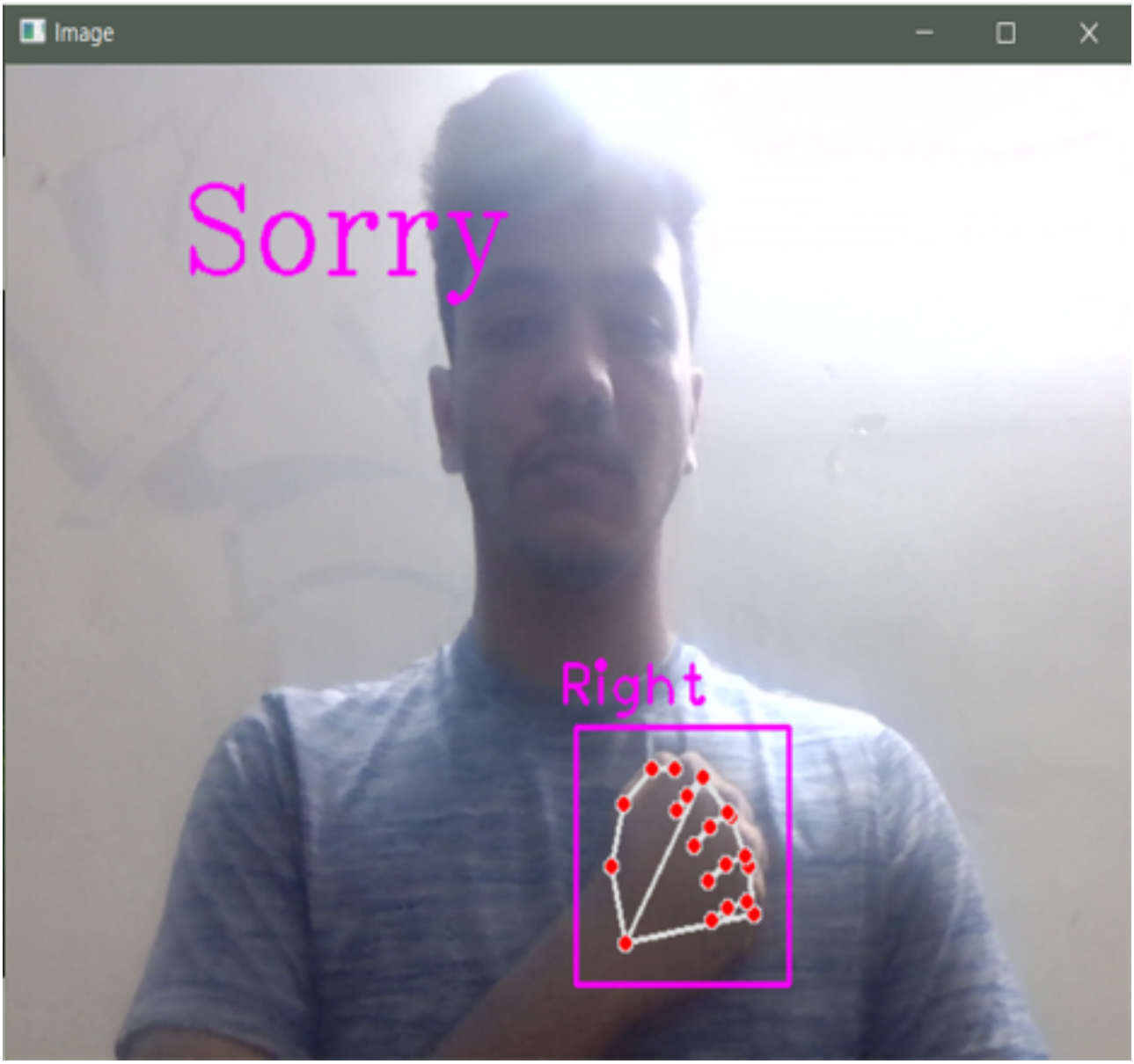
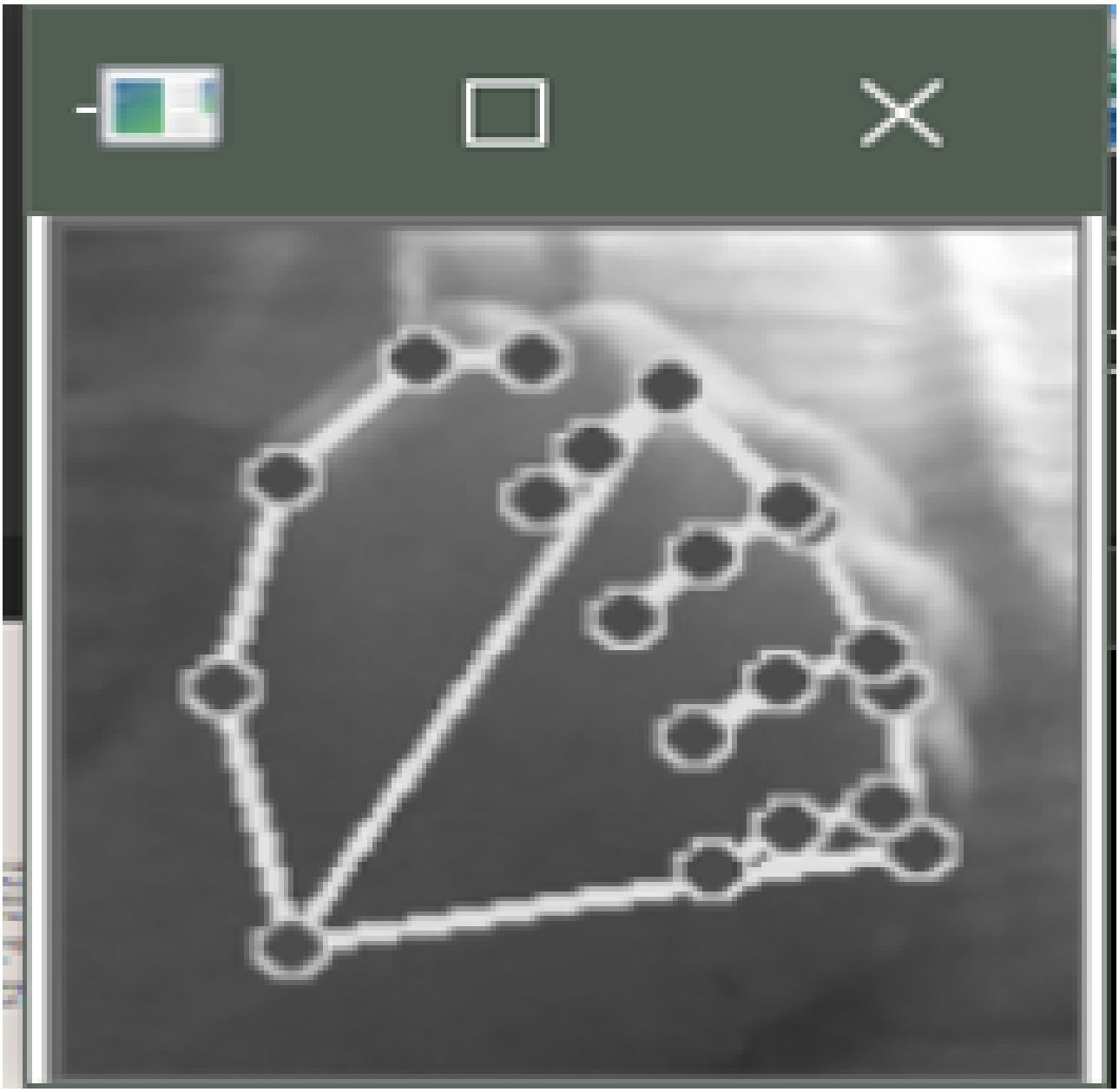


Image
Sorry
Right



CHAPTER 5

CONCLUSION

In conclusion, the development of an Indian Sign Language (ISL) to voice converter using Convolutional Neural Networks (CNNs) offers a promising solution for enhancing communication accessibility for the hearing-impaired community. By leveraging the power of CNNs, the system can accurately recognize and interpret sign language gestures. The trained CNN model extracts meaningful features from input images, enabling precise classification and mapping to corresponding speech output. This technology bridges the gap between sign language and spoken language, providing real-time conversion and facilitating effective communication for individuals who use ISL. The ISL to voice converter using CNNs has the potential to significantly improve inclusivity and quality of life for the hearing-impaired population in India.

5.1 REFERENCES

[1]Truong, Vi NT, Chuan-Kai Yang, and Quoc-Viet Tran. "A translator for American sign language to text and speech." 2016 IEEE 5th Global Conference on Consumer Electronics. IEEE, 2016

[2]Dutta, Kusumika Krori, and Anil Kumar GS. "Double handed Indian Sign Language to speech and text." 2015 Third International Conference on Image Information Processing (ICIIP). IEEE, 2015.

[3]Ahmed, Mateen, et al. "Deaf talk using 3D animated sign language: A sign language interpreter using Microsoft's kinect v2." 2016 SAI Computing Conference (SAI). IEEE, 2016.

[4]Song, Nan, Hongwu Yang, and Tingting Zhang. "A DNN-Based Framework for Converting Sign Language to Mandarin-Tibetan Cross-Lingual Emotional Speech." 2018 International Conference on Asian Language Processing (IALP). IEEE, 2018.

[5]Song, Nan, Hongwu Yang, and Pengpeng Zhi. "Towards realizing sign language to emotional speech conversion by deep learning." Data Science: 4th International Conference of Pioneering Computer Scientists, Engineers and Educators, ICPCSEE 2018, Zhengzhou, China, September 21-23, 2018, Proceedings, Part II. Springer Singapore, 2018.

[6]Fernandes, Lance, et al. "Convolutional neural network based bidirectional sign language translation system." 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE, 2020.