A Mini Project Report On

SIGN LANGUAGE RECOGNITION

submitted

in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Submitted by

M. AAKANKSHA	**20B81A6701**
JAI KUMAR PAGARE	**20B81A6717**
VAISHNAVI VARAYOGI	**20B81A6753**

Under the esteemed guidance of

MR. S. BALAKRISHNA REDDY

Assistant Professor

Department of CSE(DS)

DEPARTMENT OF CSE (DATA SCIENCE)

CVR COLLEGE OF ENGINEERING

(***An Autonomous institution, NBA, NAAC Accredited and Affiliated to JNTUH, Hyderabad***)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M), Rangareddy (D), Telangana- 501 510

November 2023

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

DEPARTMENT OF CSE (DATA SCIENCE)

CERTIFICATE

This is to certify that the project report entitled **"Sign Language Recognition"** bonafide record of work carried out by **M. Aakanksha (20B81A6701), Jai Kumar Pagare (20B81A6717)** and **Vaishnavi Varayogi (20B81A6753)** submitted to **Mr. S. Balakrishna Reddy** for the requirement of the award of **Bachelor of Technology** in **Computer Science and Engineering (Data Science)** to the CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University, Hyderabad during the year 2023-2024.

Signature of the Project Guide,
Mr. S. Balakrishna Reddy
Assistant Professor
CSE (Data Science)

Project Coordinator
Dr. P. A. Abdul Saleem
Professor, CSE (Data Science)

Signature of the HOD
Dr. H. N. Lakshmi
Head of the Department
CSE (AI&ML, CS, DS)

External Examiner

DECLARATION

We hereby declare that the project report entitled **"Sign Language Recognition"** is an original work done and submitted to CSE(DS) Department, CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University Hyderabad in partial fulfilment for the requirement of the award of **Bachelor of Technology** in **Computer Science and Engineering (Data Science)** and it is a record of bonafide project work carried out by us under the guidance of **Mr. S. Balakrishna Reddy**, Assistant Professor, Department of Computer Science and Engineering.

We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma in this Institute or any other Institute or University.

Signature of the Student

M.AAKANKSHA

Signature of the Student

JAI KUMAR PAGARE

Signature of the Student

VAISHNAVI VARAYOGI

Date:

Place: Hyderabad

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project.

We sincerely thank our project guide **Mr. S. Balakrishna Reddy** under whom we have carried out the project work. Their incisive and objective guidance and timely encouraged us with constant flow of energy to continue the work.

It is a great pleasure to convey our profound sense of gratitude to **Dr. H. N. Lakshmi**, Head of CSE (AI&ML, CS, DS) Department, CVR College of Engineering, for having been kind enough to arrange the necessary facilities for executing the project in the college.

We thank our **Vice Principal Prof. L. C. Siva Reddy** for providing excellent computing facilities and a disciplined atmosphere for doing our work.

We wish a deep sense of gratitude and heartfelt thanks to our **Principal Dr. K. Ramamohan Reddy** and Management for providing excellent lab facilities and tools. Finally, we thank all those guidance helpful to us in this regard.

ABSTRACT

Sign language conversion to text and speech is a transformative technology that bridges communication gaps between the deaf and hearing communities. This innovation involves the real-time interpretation of sign language gestures into written text and spoken words, fostering seamless interaction and understanding. Through computer vision and machine learning algorithms, sign language recognition systems capture and interpret the intricate hand movements, facial expressions, and body language used in sign language communication.

One key benefit of this technology is its potential to enhance accessibility. Deaf individuals can express themselves more fluently in everyday situations, from ordering at a restaurant to participating in classroom discussions, as their signs are instantly translated into comprehensible text and speech. Additionally, it promotes inclusivity by enabling hearing individuals to engage in meaningful conversations with deaf individuals without the need for an interpreter. Moreover, this technology holds promise in education, as it can facilitate the learning of sign language and support the integration of deaf students into mainstream educational environments.

Sign language conversion to text and speech signifies a significant stride towards breaking down communication barriers and promoting equal opportunities for deaf individuals. As advancements in artificial intelligence and computer vision continue to evolve, the accuracy and accessibility of this technology are likely to improve, fostering a more inclusive and interconnected world for everyone.

TABLE OF CONTENTS

	Contents		Page No.
		List of Figures	v
		Abbreviations	vi
1		**Introduction**	
	1.1	Introduction	1
	1.2	Motivation	1
	1.3	Objectives	2
	1.4	Problem Statement	2
	1.5	Report Organization	3
2		**Literature Survey**	
	2.1	Existing Work	4
	2.2	Limitations	5
3		**System Requirements Specification**	
	3.1	Software requirements	6
	3.2	Hardware requirements	6
4		**Proposed System Design**	
	4.1	Use case Diagram	8
	4.2	Class Diagram	9
	4.3	Activity Diagram	10
	4.4	System Architecture Diagram	11
5		**Implementation and Results**	
	5.1	Implementation	12
	5.2	Testing	16
6		**Conclusion and Future Scope**	
	6.1	Conclusion	22
	6.2	Future Scope	22
7		**References**	23

List of Figures

Figure	Description	Pg No.
2.1.1	Existing Model	4
4.1.1	Use Case Diagram	8
4.2.1	Class Diagram	9
4.3.1	Activity Diagram	10
4.4.1	System Architecture	11
5.1.1.2.1	Data Preprocessing	13
5.1.1.2.2	Mediapipe Landmarks	13
5.1.1.3.1	CNN Working Methodolgy	14
5.1.1.3.2	Types of Pooling in CNN	15
5.2.6.1	Output of the Model	21

List of Abbreviations

ASL	American Sign Language
CNN	Convolutional Neural Network
UI	User Interface
NLP	Natural Language Processing
TTS	Text-to-Speech
GUI	Graphical User Interface
PY	Python

CHAPTER 1 - INTRODUCTION

1.1 INTRODUCTION

In the ever-evolving landscape of technology, the choice of the right tools can be the linchpin differentiating between effective solutions and persistent challenges. When it comes to the field of sign language recognition, the traditional methods often come with limitations that hinder accessibility and inclusivity. Recognizing these hurdles, our project embarks on a journey of innovation and empowerment by delving into the world of sign language recognition technology.

Traditional sign language recognition systems have been marred by complexities and constraints, including limited adaptability and accessibility. In response to these challenges, our project seeks to explore a novel solution: the development of advanced sign language recognition technology. This endeavour is rooted in the firm belief that technology should empower, not inhibit, the communication and interaction of deaf individuals with the world at large.

We aim to harness technology to make sign language more accessible and inclusive, ensuring that individuals who rely on this form of communication have the freedom to express themselves and participate fully in society. This project serves as a beacon of accessibility, empowerment, and progress in the field of sign language recognition.

1.2 MOTIVATION

The primary motivation for embarking on this project is to address the evolving needs and challenges faced by developers in the realm of software development. Traditional Integrated Development Environments (IDEs) have long been powerful tools, but they come with their own set of obstacles, such as intricate installation procedures, compatibility issues, and the restriction to specific devices, which can hinder the efficiency and flexibility of developers in their work.

Ultimately, our project draws inspiration from the desire to equip developers with the knowledge and tools necessary to make informed decisions about their coding environments. We aim to contribute to a more efficient, inclusive, and adaptable software development landscape by providing insights into the advantages and considerations of web-based code editors. Our mission is to empower developers to work more efficiently and effectively while enjoying the freedom to code on their own terms.

1.3 PROBLEM STATEMENT

Sign languages can vary significantly across regions and communities. Recognizing diverse sign language gestures accurately can be challenging, as there may not be a universal standard for all signs. Addressing this variability and providing support for multiple sign languages can be a complex problem.

Achieving real-time processing of sign language gestures from video input is essential for effective communication. Delays or lag in gesture recognition can disrupt the flow of conversation and reduce usability. Sign language involves not only hand gestures but also facial expressions and body movements, which convey important information. Accurately capturing and interpreting these aspects can be computationally intensive and require advanced computer vision techniques.

Building robust recognition model necessitates a large and diverse dataset of sign language gestures. Collecting and annotating such data can be time-consuming and resource-intensive, and ensuring the accuracy of annotations is critical.

1.4 PROJECT OBJECTIVE

The central objective of the sign language recognition project is to develop a comprehensive and inclusive system that recognizes, interprets, and translates sign language gestures into written text or spoken language. This project aims to bridge the communication gap between the deaf and hearing communities, enhancing accessibility and inclusivity for the deaf population.

Key Goals and Features:

1.4.1 Sign Language Recognition: Implement a robust system capable of accurately recognizing a wide range of sign language gestures and expressions in real-time.

1.4.2 Translation and Interpretation: Provide the ability to translate recognized sign language gestures into written text and/or spoken language, ensuring effective communication between deaf individuals and those who do not understand sign language.

1.4.3 User-Friendly Interface: Create an intuitive and user-friendly interface that accommodates both sign language users and individuals unfamiliar with sign language, promoting ease of communication and collaboration.

1.4.4 Accessibility: Prioritize accessibility for all users, including those with disabilities, by ensuring compatibility with assistive technologies such as screen readers and promoting inclusive design principles.

1.4.5 Continuous Learning and Improvement: Establish a system that can adapt and learn from new sign language expressions, ensuring that it remains up-to-date with evolving sign languages and dialects.

1.4.6 Security and Privacy: Implement strong security measures to protect user data, as sign language recognition may involve personal or sensitive information, and ensure that user privacy is maintained.

1.4.7 Scalability: Design the system to be scalable to accommodate potential future enhancements and a growing user base, allowing for widespread adoption and accessibility.

1.4.8 Cross-Platform Compatibility: Ensure that the sign language recognition system can be accessed and utilized across various devices and platforms, further increasing its reach and utility in diverse settings.

1.5 PROJECT REPORT ORGANIZATION

This book contains six chapters. The first chapter contains motivation, problem statement and objectives of the project. The second chapter includes the survey done regarding the project and the limitations of the project. The third chapter includes requirements and specifications. The fourth chapter includes UML diagrams and Technology description. The fifth chapter includes Implementation which contains the technologies used for developing the applications and code snippets. The fifth chapter also contains test cases and screenshots of the applications. The sixth chapter looks into the references and the conclusion of the project.

CHAPTER 2 - LITERATURE SURVEY

2.1 EXISTING WORK

The project undertaken by Luv (IIT2016085), Nitin Raj Singh (IIT2016132), Ravi Chandra (IIT2016141), Sheldon Tauro (IIT2016137), and Siddhant Gautam (IIT2016069, students of IIT in 2019. Their primary objective is to develop a computer application that can recognize and interpret real-time hand gestures in American Sign Language (ASL) from video feeds. This application will employ advanced machine learning models to process and understand the intricate movements of the hand gestures, subsequently providing the corresponding output in text format on the screen.

This project represents a significant leap forward in accessibility and communication for individuals who use ASL as their primary means of interaction. By bridging the gap between sign language and text, the team aims to empower deaf and hard-of-hearing individuals, enabling them to seamlessly communicate in various settings, including educational, professional, and everyday social contexts. Furthermore, the development of this application has the potential to benefit a wide range of applications, such as accessibility in technology, education, and communication, making it a promising contribution to the field of assistive technology. The combined efforts of these dedicated individuals will undoubtedly result in a valuable and impactful solution for the ASL community and beyond.

Fig. No: 2.1.1 Existing Model

2.2 LIMITATIONS OF EXISTING WORK

The project aimed at creating a computer application for real-time American Sign Language (ASL) recognition has shown promise, but it does come with certain limitations. Here are some of the key limitations of the ASL recognition model:

2.2.1 Lighting Sensitivity: The model's performance is heavily dependent on good lighting conditions. Poor lighting can significantly affect its accuracy, limiting its usability in various environments.

2.2.2 Plain Background Requirement: For accurate gesture detection, a plain background is necessary. This limitation can be impractical in real-world scenarios where backgrounds are often complex and dynamic.

2.2.3 Text-to-Speech Conversion Absence: The system lacks the capability to convert the recognized sign language gestures into audible speech. This restricts its usability for individuals who may rely on spoken language as their primary mode of communication.

2.2.4 Word Recommendation Feature: The absence of a word recommendation feature means that the model does not offer assistance or suggestions for sign language users, which could be particularly useful for individuals who are less proficient in ASL or for enhancing communication efficiency.

CHAPTER 3 - SOFTWARE & HARDWARE SPECIFICATIONS

3.1 SOFTWARE REQUIREMENTS

3.1.1 Gesture Recognition Software: This software utilizes computer vision algorithms and machine learning models to recognize and interpret sign language gestures. Popular libraries and frameworks for this purpose include OpenCV, TensorFlow, and PyTorch.

3.1.2 Natural Language Processing (NLP) Software: NLP software is used to convert recognized signs into written text. It may involve tokenization, part-of-speech tagging, and language modeling. Libraries like NLTK, spaCy, and Transformers (Hugging Face) are often employed.

3.1.3 Speech Synthesis Software: Text-to-speech (TTS) software generates spoken language from the converted text. TTS engines like Google Text-to-Speech, Amazon Polly, or custom TTS models built using Tacotron and WaveNet are commonly used.

3.1.4 Real-Time Communication Software: For real-time applications, software for live streaming and communication, such as WebRTC or specialized video conferencing platforms, is crucial to enable immediate interaction.

3.1.5 User Interfaces (UI): User-friendly interfaces are developed using frontend technologies like HTML, CSS, and JavaScript for web applications or platform-specific frameworks for mobile apps.

3.1.6 Backend and Server Components: To process data, manage user accounts, and ensure system reliability, backend software is needed. Common technologies include Node.js, Django, Flask, or serverless solutions like AWS Lambda.

3.2 HARDWARE SPECIFICATIONS

3.2.1 Cameras or Depth Sensors: High-resolution cameras or depth sensors like Microsoft Kinect are used for capturing sign language gestures accurately. Multiple cameras may be required for capturing different angles.

3.2.2 Sensors and Wearables: In some cases, specialized gloves equipped with sensors or other wearable devices are used to capture hand and body movements more precisely.

3.2.3 Computer/Server: A powerful computer or server is necessary to process the data in real-time, especially for complex gesture recognition models and speech synthesis.

3.2.4 Microphones and Speakers: For speech synthesis and audio communication, microphones and speakers are essential components of the hardware setup.

3.2.5 Internet Connectivity: Reliable internet connectivity is crucial for real-time communication applications, ensuring low latency and smooth interactions.

3.2.6 Display Devices: For displaying converted text or video feeds, monitors, screens, or mobile devices are used.

3.2.7 Audio and Video Interfaces: Adequate audio and video input/output interfaces are needed to connect cameras, microphones, and speakers to the computer or server.

3.2.8 Power Supply: Ensure an uninterrupted power supply to prevent system downtime

CHAPTER 4 – PROPOSED SYSTEM DESIGN

4.1 USE CASE DIAGRAM

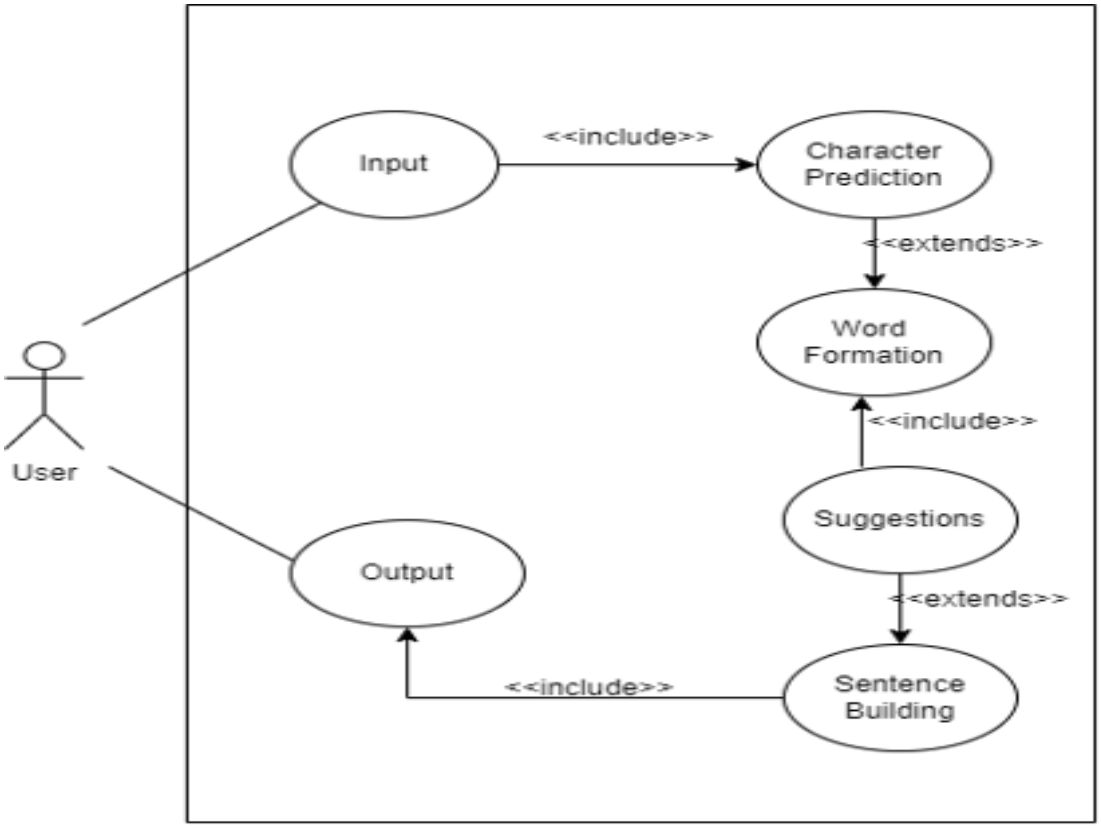

Fig. No: 4.1.1 Use Case Diagram

The use case diagram demonstrates the flow of the process for real-time American Sign Language (ASL) recognition and translation into text format.The User initiates the process by providing input through ASL gestures. The Character Prediction component predicts individual characters or signs based on the input. Word Formation assembles these characters into words. Suggestions may offer recommendations or corrections during word formation. Sentence Building constructs coherent sentences from recognized words. The final Output displays the text format of the ASL signs.

4.2 CLASS DIAGRAM

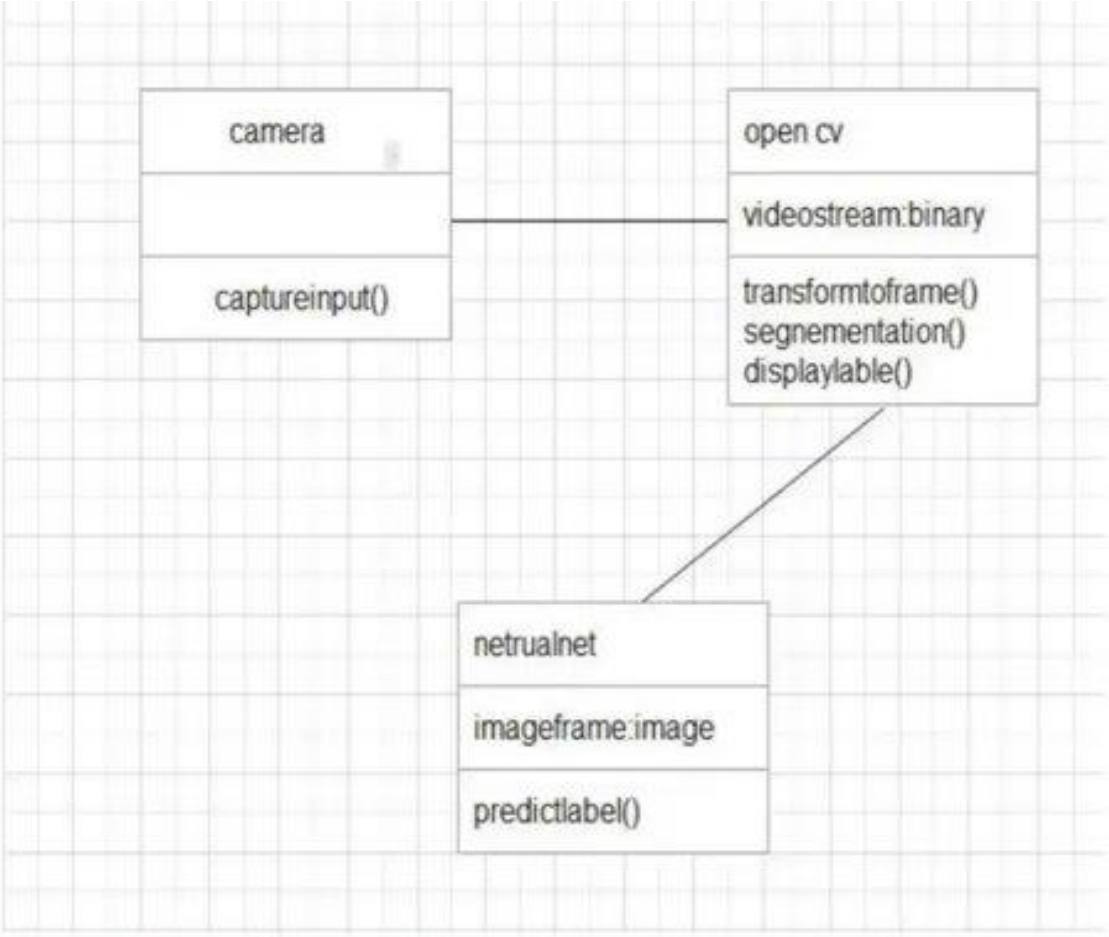

Fig. No: 4.2.1 Class Diagram

This class diagram illustrates three classes: "Camera", "open cv," and "Neuralnet". The 'Camera" class represents the hardware or software component responsible for capturing visual data, such as images or video streams. It is a crucial element for input in various computer vision applications. The "OpenCV" class signifies the software library or framework utilized for image processing and computer vision tasks. This class offers a wide range of functions and tools for tasks like image manipulation, feature detection, and more. The "Neural Net" class indicates the neural network model or architecture used for deep learning and artificial intelligence applications. It plays a fundamental role in tasks like object recognition and image analysis.

4.3 ACTIVITY DIAGRAM

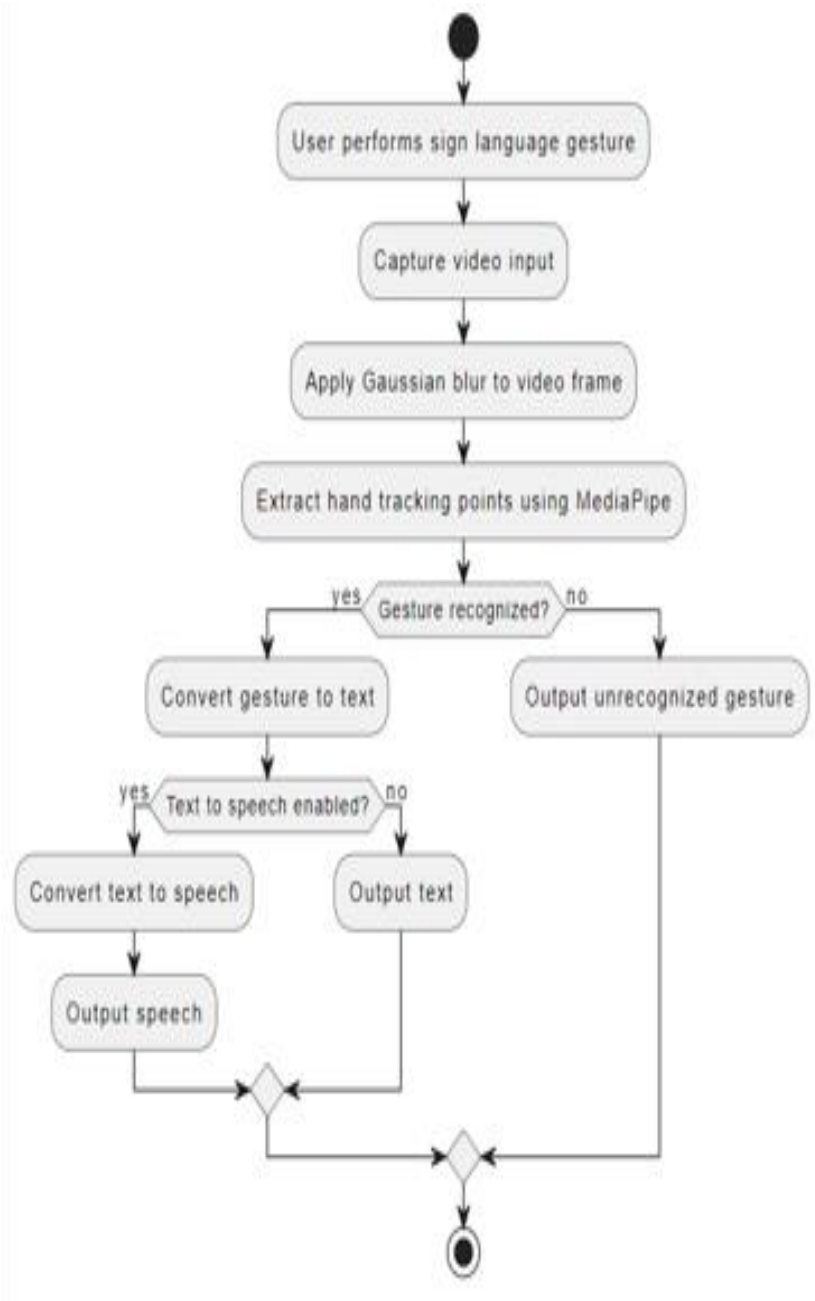

Fig. No: 4.3.1 Activity Diagram

The activity diagram commences with the "start" node, followed by the action to "Extracting hand tracking points using MediaPipe." A decision point, represented by the "if" statement, determines the need for the gesture to be classified. In the "yes" branch, the gesture is converted to the text. If the answer is "no," we continue using the user interface. The diagram concludes with obtaining the classified gesture into its respective text.

4.4 SYSTEM ARCHITECTURE

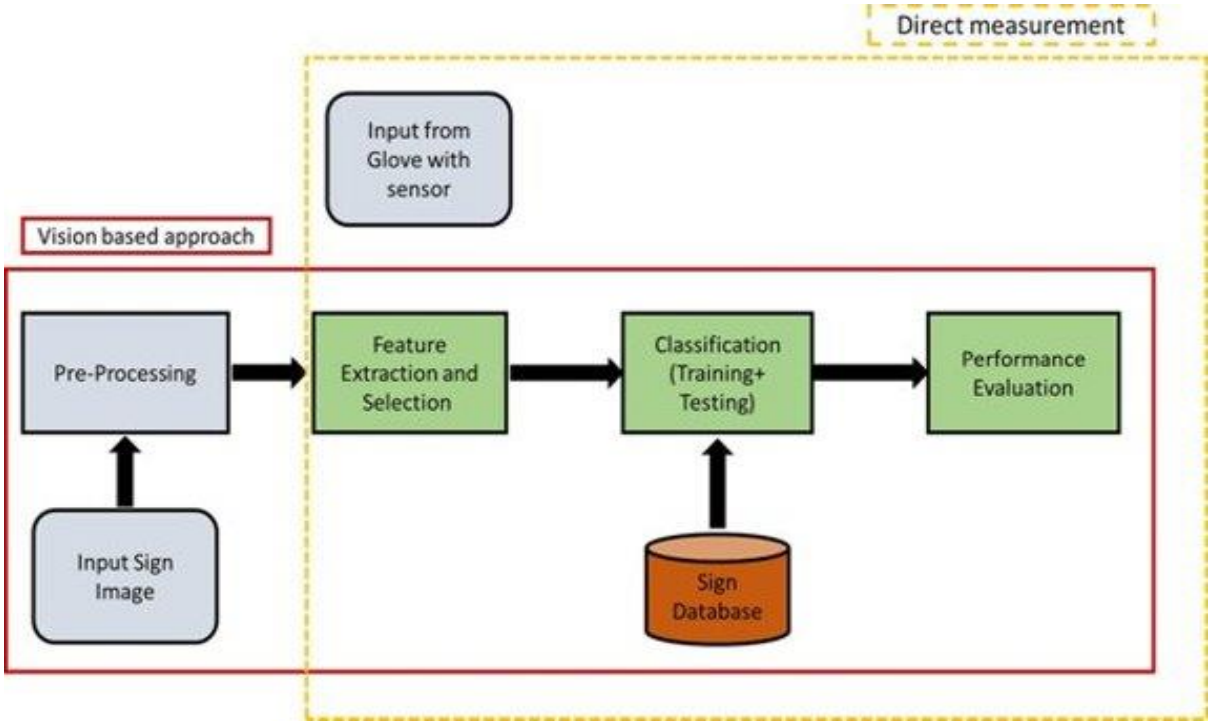

Fig. No: 4.4.1 System Architecture

The system architecture for sign language recognition is a comprehensive framework that seamlessly integrates multiple components to provide accurate and efficient sign language recognition. Input from Sensor is responsible for capturing sign language gestures through sensors embedded in a glove or wearable device. The vision-based approach encompasses several key components that process visual sign language cues. Input Sign Image captures the visual sign language cues in the form of images or video streams. The preprocessing component cleans and enhances the input sign images to improve the quality of the data. Feature Extraction and Selection extracts relevant features from the preprocessed images and selects the most discriminative ones for classification. The classification module employs machine learning techniques to recognize sign language gestures against real-time or database-based sign language inputs. The performance evaluation component assesses the accuracy and efficiency of the sign language recognition system. It may involve metrics, such as recognition accuracy, processing time, and error rates.

CHAPTER 5 - IMPLEMENTATION AND RESULTS

5.1 IMPLEMENTATION

5.1.1 Functional requirements

1. Data Acquisition :

Data acquisition is a critical phase in a sign language recognition project. It involves the collection of a diverse dataset of hand gestures, encompassing variations in skin tones and capturing scenarios with different viewpoints, scales, and camera speeds. This diversity ensures that the machine learning model trained on this dataset can effectively recognize and generalize hand gestures under various real-world conditions. The dataset's comprehensiveness is pivotal for the success of the project, as it allows the model to adapt to the diverse ways sign language can be expressed, ultimately making the system more inclusive and accurate for a wide range of users.

2. Data pre-processing and Feature extraction:

In the critical stage of data pre-processing and feature extraction for a sign language recognition project, the primary aim is to refine and prepare the input data for effective analysis. It commences with the identification and isolation of the region of interest (ROI) within the captured image, typically encompassing the hand gestures being performed. Leveraging the powerful OpenCV library, this selected area is meticulously cropped, enabling a more focused and precise analysis of the hand's movements. Subsequently, the grayscale transformation is applied to the cropped image, a fundamental step that simplifies the data while retaining crucial visual information necessary for in-depth analysis. To enhance the image quality, a Gaussian blur is introduced, effectively reducing noise and ensuring a smoother, more uniform appearance.

Following grayscale conversion, a thresholding method is employed to convert the image into a binary representation, effectively segregating the hand from the background. This binary image serves as the canvas for the identification and extraction of hand landmarks, which encompass vital points on the fingers and palm. These landmarks are then visually represented by drawing and connecting them on a blank white canvas, providing a clear and standardized depiction of the hand's position and shape. This comprehensive and standardized visual representation of the hand is pivotal

for subsequent feature extraction and analysis, ultimately enabling the system to consistently and accurately recognize and interpret a broad spectrum of sign language gestures with precision and reliability.

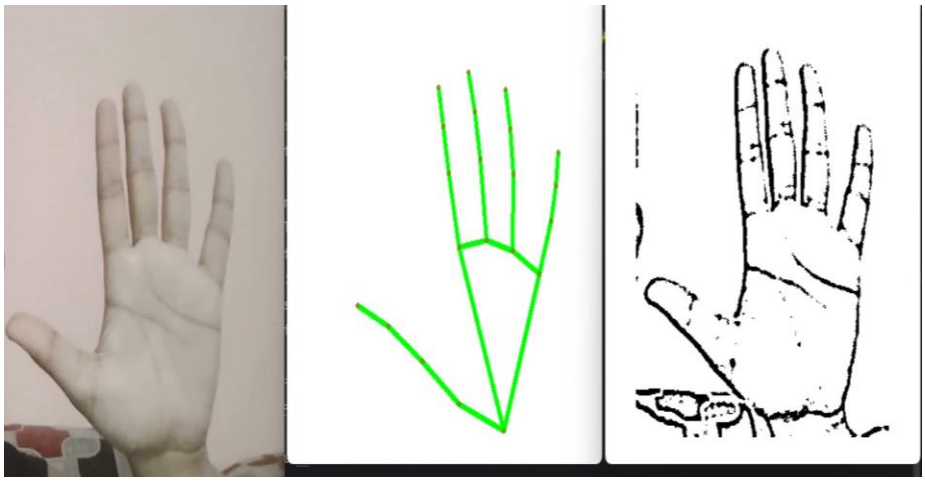

Fig. No: 5.1.1.2.1 Data Preprocessing

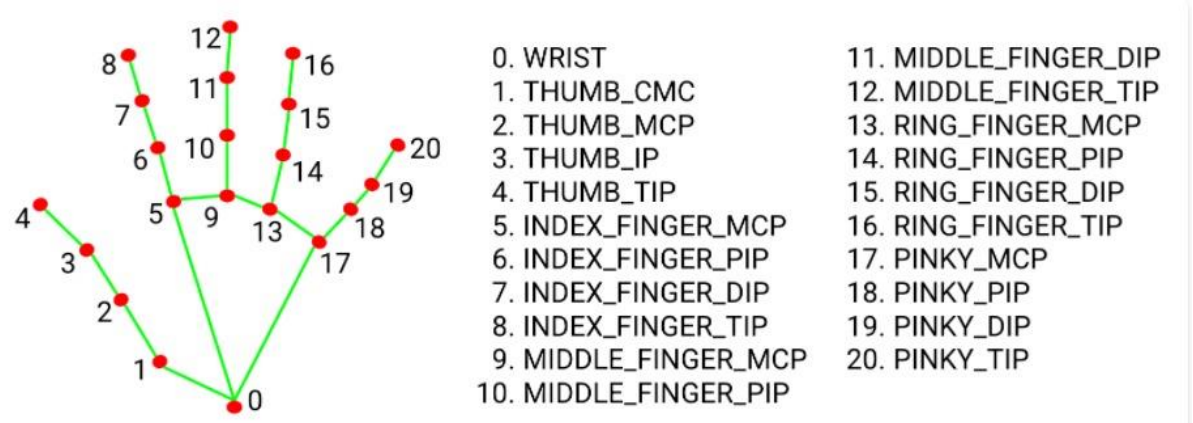

Fig. No: 5.1.1.2.2 Mediapipe Landmarks

3. Gesture Classification and Convolutional Neural Network (CNN) :

CNN is a class of neural networks that are highly useful in solving computer vision problems. They found inspiration from the actual perception of vision that takes place in the visual cortex of our brain. They make use of a filter/kernel to scan through the entire pixel values of the image and make computations by setting appropriate weights to enable detection of a specific feature. CNN is equipped with layers like convolution layer, max pooling layer, flatten layer, dense layer, dropout layer and a fully connected neural network layer. These layers together make a very powerful tool that can identify

features in an image. The starting layers detect low level features that gradually begin to detect more complex higher-level features.

Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth.The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner.Moreover, the final output layer would have dimensions(number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores.

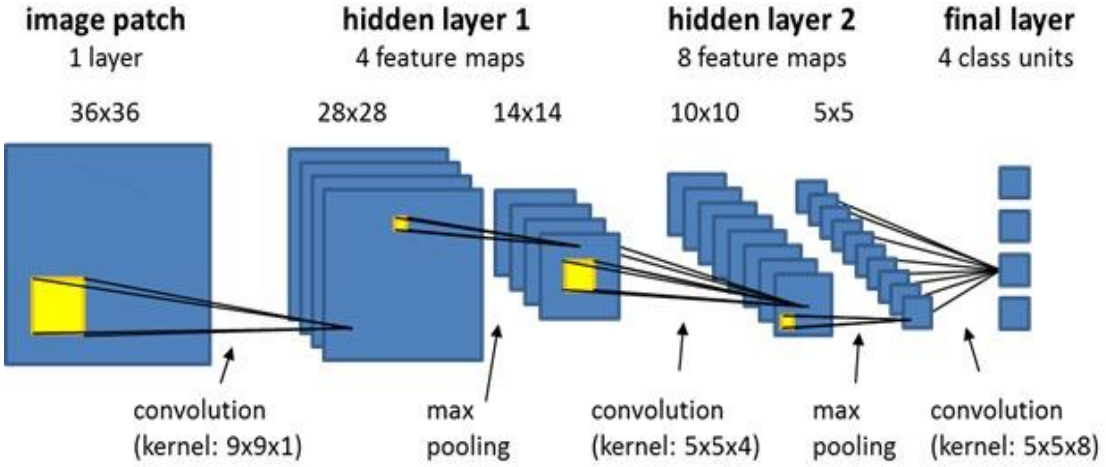

Fig. No: 5.1.1.3.1 CNN Working Methodology

In convolution layer we have taken a small window size [typically of length 5*5] that extends to the depth of the input matrix.The layer consists of learnable filters of window size. During every iteration we slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position.As we continue this process well create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position.

Pooling Layer:

We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters.

There are two types of pooling:

a. **Max Pooling** :In max pooling we take a window size [for example window of size 2*2], and only taken the maximum of 4 values.

Well lid this window and continue this process, so well finally get an activation matrix half of its original Size.

b. **Average Pooling**: In average pooling we take average of all Values in a window.

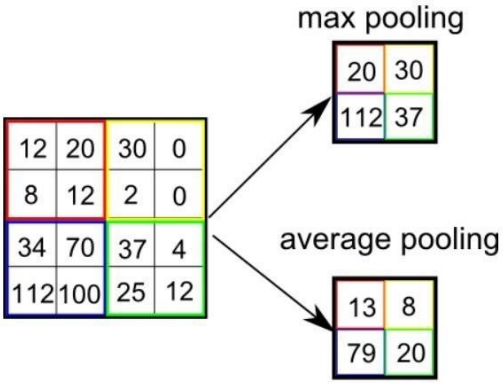

Fig. No: 5.1.1.3.2 Types of pooling in CNN

We divided whole 26 different alphabets into 8 classes in which every class contains similar alphabets:

[y,j

[c,o]

[g,h]

[b,d,f,I,u,v,k,r,w]

[p,q,z]

[a,e,m,n,s,t]

All the gesture labels will be assigned with a probability. The label with the highest probability will treated to be the predicted label. So when model will classify [aemnst] in one single class using mathematical operation on hand landmarks we will classify further into single alphabet a or e or m or n or s or t.

Finally, we got better accuracy (with and without clean background and proper lightning conditions) through our method. And if the background is clear and there is good lightning condition then we got even high accurate results.

4.Text To Speech Translation:

The system's functionality includes the translation of recognized gestures into spoken words. To achieve this, we've integrated the pyttsx3 library, which effectively converts the recognized words into audible speech. While the text-to-speech output may be a relatively straightforward method, it serves as a valuable feature, as it closely mimics natural, real-life conversations. This capability enhances the user experience and fosters

effective communication, making the sign language recognition system more accessible and user-friendly, enabling a seamless interaction between users and the system.

5.2 TESTING

1.Data Collection and Preprocessing:

In this hand detection approach, the process starts with the detection of a hand in an image captured by a webcam, utilizing the MediaPipe library for image processing. Once the hand is detected, a region of interest (ROI) is identified, and the image is cropped. Then, the image is converted to grayscale using the OpenCV library and subjected to Gaussian blur. The grayscale image is further transformed into a binary image using threshold and Adaptive threshold methods.

```
while True:
    try:
        _, frame = capture.read()
        frame = cv2.flip(frame, 1)
        hands= hd.findHands(frame, draw=False, flipType=True)
        img_final=img_final1=img_final2=0

        if hands:
            hand = hands[0]
            x, y, w, h = hand['bbox']
            image = frame[y - offset:y + h + offset, x - offset:x + w +
offset]
            roi = image
            gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
            blur = cv2.GaussianBlur(gray, (1, 1), 2)
            img_final = np.ones((400, 400), np.uint8) * 255
            h = test_image.shape[0]
            w = test_image.shape[1]
            img_final[((400 - h) // 2):((400 - h) // 2) + h, ((400 - w) //
            2):((400 - w) // 2) + w] = test_image
            hands = hd.findHands(white, draw=False, flipType=True)
            if hands:
            hand = hands[0]
            x, y, w, h = hand['bbox']
            cv2.rectangle(white, (x - offset, y - offset), (x + w, y + h),
(3, 255,  25),  3)

cv2.imshow("binary", img_final)
cv2.imshow("skeleton",imgwhite)
```

2. Natural Language Processing (NLP) Software:

NLP (Natural Language Processing) software is a critical component in the sign language recognition system, tasked with transforming recognized signs into coherent written text. This intricate process entails several steps, including tokenization, part-of-speech tagging, and language modelling. To ensure the system provides an accurate and coherent interpretation of sign language, we've integrated the versatile enchant module. Enchant provides a valuable dictionary and suggestion framework, which aids in improving the accuracy and quality of the text generated from the sign language gestures. It not only enhances the precision of tokenization and language modelling but also offers helpful suggestions, enabling users to correct any potential errors and ensure the final text output is both meaningful and linguistically accurate. This integration of the enchant module reflects a commitment to user-friendliness and communication efficacy, ensuring that the system not only translates sign language accurately but also presents the results in a format that is readily understood by all.

```
import enchant
ddd=enchant.Dict("en-US")

self.T4 = tk.Label(self.root)
self.T4.place(x=10, y=700)
self.T4.config(text="Suggestions :", fg="red", font=("Courier", 30,
"bold"))

if len(word.strip())!=0:
    ddd.check(word)
    lenn = len(ddd.suggest(word))
    if lenn >= 4:
        self.word4 = ddd.suggest(word)[3]

    if lenn >= 3:
        self.word3 = ddd.suggest(word)[2]

    if lenn >= 2:
        self.word2 = ddd.suggest(word)[1]

    if lenn >= 1:
        self.word1 = ddd.suggest(word)[0]
else:
    self.word1 = " "
    self.word2 = " "
    self.word3 = " "
    self.word4 = " "
```

3.Speech Synthesis Software:

Speech Synthesis Software is a pivotal component in our sign language recognition system, responsible for the transformation of converted text into spoken language. To fulfill this role, we've integrated the versatile "pyttsx3" library, which serves as a reliable and efficient tool for text-to-speech conversion. With pyttsx3, the system can generate natural and intelligible speech from the translated text, providing a seamless and inclusive experience for users. This text-to-speech capability is invaluable as it bridges the gap between the system and its users, simulating real-life dialogue and enabling effective communication. This comprehensive approach underlines the commitment to providing a versatile and user-friendly sign language recognition system.

```
import pyttsx3

def __init__(self):
    self.vs = cv2.VideoCapture(0)
    self.current_image = None
    self.model = load_model('cnn8grps_rad1_model.h5')
    self.speak_engine=pyttsx3.init()
    self.speak_engine.setProperty("rate",100)
    voices=self.speak_engine.getProperty("voices")
    self.speak_engine.setProperty("voice",voices[0].id)
```

4. User Interface (UI):

User Interface (UI) play a fundamental role in ensuring the accessibility and usability of any software application, including our sign language recognition system. To create intuitive and user-friendly interfaces, we have adopted the widely-used Tkinter library, which is a Python library for developing graphical user interfaces (GUIs). Tkinter offers a versatile and efficient toolkit for building interactive and visually appealing interfaces, making it an excellent choice for our project.

With Tkinter, we have crafted interfaces that facilitate seamless interactions between users and the sign language recognition system. The library provides a range of widgets such as buttons, labels, text boxes, and more, allowing us to design a user-friendly environment that simplifies the user's experience. Tkinter's capability to create platform-independent and responsive interfaces aligns

```
import tkinter as tk

self.root = tk.Tk()
self.root.title("Sign Language Recognition")
self.root.protocol('WM_DELETE_WINDOW', self.destructor)
self.root.geometry("1300x700")
self.panel = tk.Label(self.root)
self.panel.place(x=100, y=3, width=480, height=640)
self.panel2 = tk.Label(self.root)  # initialize image panel
self.panel2.place(x=700, y=115, width=400, height=400)
self.T = tk.Label(self.root)
self.T.place(x=60, y=5)
self.T.config(text="Sign Language To Text Conversion",fg="blue",
font=("Times", 35, "bold italic"))

self.panel3 = tk.Label(self.root)
self.panel3.place(x=280, y=585)

self.T1 = tk.Label(self.root)
self.T1.place(x=10, y=580)
self.T1.config(text="Character :", font=("Courier", 30, "bold"))

self.panel5 = tk.Label(self.root)  # Sentence
self.panel5.place(x=260, y=632)

self.T3 = tk.Label(self.root)
self.T3.place(x=10, y=632)
self.T3.config(text="Sentence :", font=("Courier", 30, "bold"))

self.T4 = tk.Label(self.root)
self.T4.place(x=10, y=700)
self.T4.config(text="Suggestions :", fg="red", font=("Courier", 30,
"bold"))


self.b1=tk.Button(self.root)
self.b1.place(x=390,y=700)

self.b2 = tk.Button(self.root)
self.b2.place(x=590, y=700)

self.b3 = tk.Button(self.root)
self.b3.place(x=790, y=700)

self.b4 = tk.Button(self.root)
self.b4.place(x=990, y=700)

self.speak = tk.Button(self.root)
self.speak.place(x=1305, y=630)
self.speak.config(text="Speak", font=("Courier", 20), wraplength=100,
command=self.speak_fun)

self.clear = tk.Button(self.root)
self.clear.place(x=1205, y=630)
self.clear.config(text="Clear", font=("Courier", 20), wraplength=100,
command=self.clear_fun)
```

5.3 RESULTS

5. Model Training

The process of training a Convolutional Neural Network (CNN) model for sign language recognition involves several essential steps. It begins with initializing the model with random weights and feeding it a dataset of sign language images. During training, the model computes the loss, which quantifies the disparity between its predicted outputs and the actual labels. Concurrently, validation is crucial, as the model's performance on a separate validation dataset is periodically assessed. Adjustments to the model are made based on validation results to prevent overfitting. Fine-tuning and hyperparameter optimization may follow, exploring various parameters and model architectures to enhance performance. The trained CNN model is then deployed in a sign language recognition application for real-time gesture interpretation, with continuous monitoring and updates to adapt to evolving sign language gestures or to improve overall accuracy.

```
import cv2, os
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from tensorflow.python.keras.layers.convolutional import Conv2D
from tensorflow.python.keras.layers.convolutional import MaxPooling2D

def cnn_model():
    num_of_classes = get_num_of_classes()
    model = Sequential()
    model.add(Conv2D(16, (2,2), input_shape=(image_x, image_y, 1),
activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2),
padding='same'))
    model.add(Conv2D(32, (3,3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(3, 3), strides=(3, 3),
padding='same'))
    model.add(Conv2D(64, (5,5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(5, 5), strides=(5, 5),
padding='same'))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(num_of_classes, activation='softmax'))
    sgd = optimizers.SGD(lr=1e-2)
    model.compile(loss='categorical_crossentropy', optimizer=sgd,
metrics=['accuracy'])
    filepath="cnn8grp_rad_model.h5"
    checkpoint1 = ModelCheckpoint(filepath, monitor='val_acc',
verbose=1, save_best_only=True, mode='max')
    callbacks_list = [checkpoint1]
    return model, callbacks_list
```

6. Output of the Model

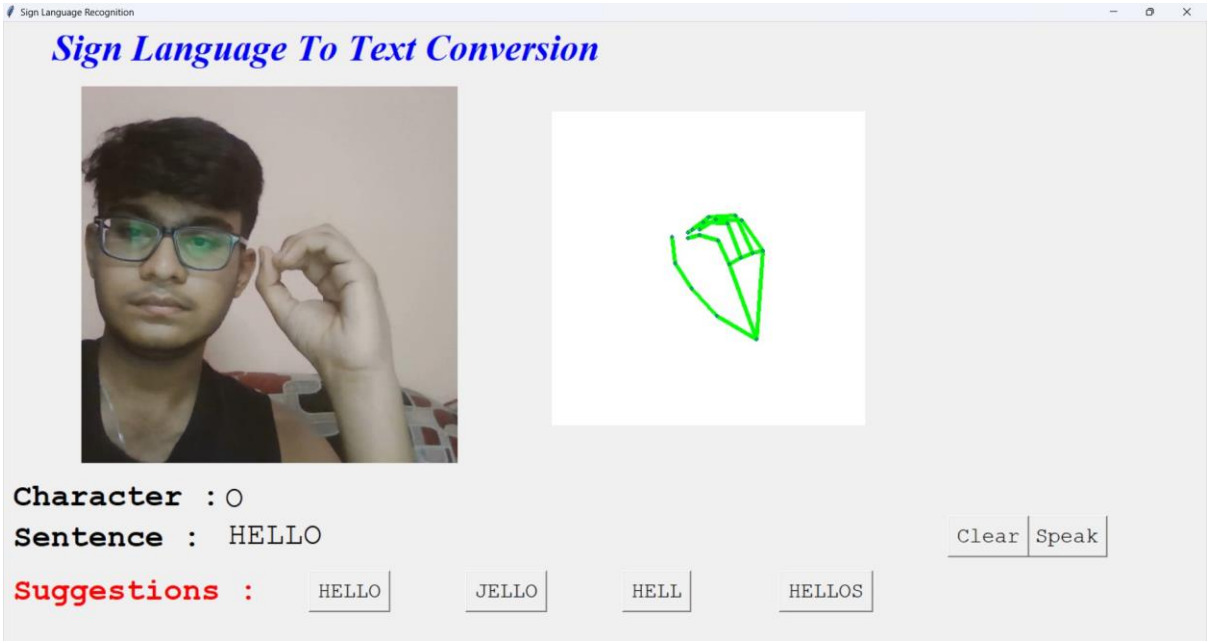

Fig. No: 5.2.6.1 Output of the Model

CHAPTER 6 - CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

In conclusion, the conversion of sign language into text and speech represents a significant advancement in technology and accessibility. This innovation has the potential to bridge communication gaps between individuals who use sign language and those who do not, thereby promoting inclusion and fostering better understanding in society. By harnessing the power of machine learning and computer vision, sign language conversion systems have made strides in real-time translation, making it easier for deaf and hard-of-hearing individuals to interact with the broader community.

In the future, continued research and development in this field will likely lead to more sophisticated and accurate sign language conversion systems. These advancements will further enhance communication options for individuals who use sign language, breaking down barriers and promoting inclusivity in our increasingly interconnected world. Ultimately, the conversion of sign language into text and speech is a promising endeavor that holds the potential to enrich the lives of many individuals and contribute to a more inclusive and understanding society.

6.2 FUTURE SCOPE

In the future, the scope for sign language conversion into text and speech extends beyond just accurate recognition. We can anticipate the development of immersive augmented reality (AR) experiences, where individuals can use AR glasses or similar devices to receive real-time sign language translations overlaid on their field of view. This technology would allow for seamless communication between deaf or hard-of-hearing individuals and hearing individuals, fostering greater inclusivity in various settings, from classrooms and workplaces to social interactions.

Furthermore, the future of sign language conversion holds immense potential in education. Sign language translation systems can play a pivotal role in making education more accessible to deaf and hard-of-hearing students. These systems can provide real-time sign language interpretations of classroom lectures, enabling students to engage fully with the curriculum. creation of accessible educational content, including sign language captions for online videos and interactive sign language tutorials. The continued development of sign language conversion technology will undoubtedly lead to greater inclusivity, improved education, and enhanced communication for the deaf and hard-of-hearing communities.

REFERENCES

[**1**] **Beniz, D., & Espindola, A.** (2016). Using Tkinter of python to create graphical user interface (GUI) for scripts in LNLS. *WEPOPRPO25*, *9*, 25-28.

[**2**] **Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C. L., & Grundmann, M.** (2020). Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*.

[**3**] **Amrutha K., & Prabu, P.** (2021, February). ML based sign language recognition system. In *2021 International Conference on Innovative Trends in Information Technology (ICITIIT)* (pp. 1-6). IEEE.

[**4**] **Yu, L., Li, B., & Jiao, B.** (2019, April). Research and Implementation of CNN based on TensorFlow. In *IOP Conference Series: Materials Science and Engineering* (Vol. 490, No. 4, p. 042022). IOP Publishing.

[**5**] **Kavitha, M., Chatterjee, A., Shrivastava, S., & Sarkar, G.** (2022, July). Formation of Text from Indian Sign Language using Convolutional Neural Networks. In *2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)* (pp. 1-5). IEEE.

[**6**] **Seetala, K., Birdsong, W., & Reddy, Y. B.** (2019). Image classification using tensorflow. In *16th International Conference on Information Technology-New Generations (ITNG 2019)* (pp. 485-488). Springer International Publishing.

[7] Porwal, R., Tomar, U., Dubey, V., Mishra, A., & Mandloi, G. (2021). Voice Assistant.