Conversion of Sign Language to Text and Speech in Real Time using Convolutional Neural Network

B.Sc. (Engineering) 4th Year 2nd Semester Examination 2022

Course Title: Thesis

Course Code: ICE-4210

B.Sc. (Engineering) Thesis

A project paper submitted to the Department of Information and Communication Engineering, Pabna University of Science and Technology (PUST) in partial fulfillments of the requirements for the B.Sc. (Engineering) 4th Year 2nd Semester Examination 2022

Supervised By:

Taskin Noor Turna

Assistant Professor

Department of Information and Communication Engineering,

Pabna University of Science and Technology

Submitted By

Md. Omar Faruk

Student ID: 190605

Registration No: 1065330

Session: 2018-2019

4th Year 2nd Semester

Department of Information and Communication Engineering,

Pabna University of Science and Technology

Pabna-6600, Bangladesh

CERTIFICATION

This is to certify that the research presented in this project is the result of the authors' independent investigation conducted under the guidance of Assistant Professor Taskin Noor Turna from the Department of Information and Communication Engineering at PUST. It is affirmed that this project has been submitted solely for the purpose of fulfilling graduation requirements.

Author
Md. Omar Faruk

Supervisor

Taskin Noor Turna
Assistant Professor
Department of Information and Communication Engineering,
Pabna University of Science and Technology

ACKNOWLEDGMENTS

I extend my heartfelt gratitude to my supervisor, Taskin Noor Turna, for her invaluable guidance throughout this project. Her mentorship illuminated the path to conducting meaningful research and offered unwavering support whenever needed. Her insights sharpened my ability to analyze complex subjects, while her encouragement allowed me to explore and synthesize ideas in my unique way. Her constructive feedback enhanced my problem-solving skills, and her support uplifted me during moments of doubt. I will always treasure the experience of working under her guidance. Additionally, I am profoundly thankful to my friends and family for their constant belief in me, even during times when I struggled with self-confidence.

ABSTRACT

The barrier to effective communication between individuals with hearing impairments and the broader community continues to be a major challenge in achieving true inclusivity. While sign language is a cornerstone of expression for the deaf community, its interpretation often demands specialized skills, creating barriers to seamless interaction. This research introduces the creation of a groundbreaking system that converts sign language gestures into understandable text and speech, facilitating real-time communication in various environments. Using cutting-edge technologies in computer vision, artificial intelligence, and natural language processing, the system captures gestures via a camera, processes them with advanced recognition models, and converts them into human-readable and audible formats. The methodology ensures reliable recognition by incorporating robust pre-processing techniques and leveraging datasets that account for variations in gestures and environments. This solution not only addresses communication challenges but also enhances accessibility and fosters inclusion in educational, professional, and social spaces. By prioritizing precision, adaptability, and user-centric design, this system sets a foundation for widespread adoption of assistive tools that bridge gaps in human communication and advance social equity.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Sign language is an essential medium of communication for individuals with hearing and speech impairments, offering a structured, visual language to convey thoughts and emotions. Despite its significance, the widespread adoption and understanding of sign language among the general population remain limited, often leading to social and communication barriers. This gap highlights the need for technological interventions to enable real-time translation of sign language into forms easily understood by non-signers, such as text and speech. [1]

The advancements in artificial intelligence (AI) and machine learning (ML) have introduced innovative possibilities for tackling intricate recognition challenges, including the detection of gestures and signs. Convolutional Neural Networks (CNNs), a distinct type of deep learning architecture crafted specifically for image-related tasks, have demonstrated exceptional capability in identifying and classifying features within visual data. In the context of sign language [2], CNNs can effectively identify and interpret hand gestures, movements, and even facial expressions, making them an ideal choice for real-time translation systems.

While existing systems for sign language recognition have made notable progress, many face challenges in achieving high accuracy under varying conditions, such as diverse lighting, signer styles, and environmental noise. Furthermore, the integration of gesture recognition with text and speech synthesis in real-time remains a relatively underexplored area. This project seeks to address these limitations by leveraging CNNs to create a robust and efficient system for converting sign language gestures into readable text and audible speech [3].

The proposed system aims to promote inclusivity by bridging the communication gap between hearing-impaired individuals and the broader community. By combining advancements in computer vision and speech synthesis, this research contributes to the development of practical and scalable solutions that can be deployed in educational institutions, workplaces, healthcare facilities, and public spaces, fostering a more accessible and connected world [5].

Our primary objective is to develop a model capable of recognizing hand gestures associated with Fingerspelling, enabling the formation of complete words by combining individual gestures. The specific gestures we intend to train are depicted in the image below.

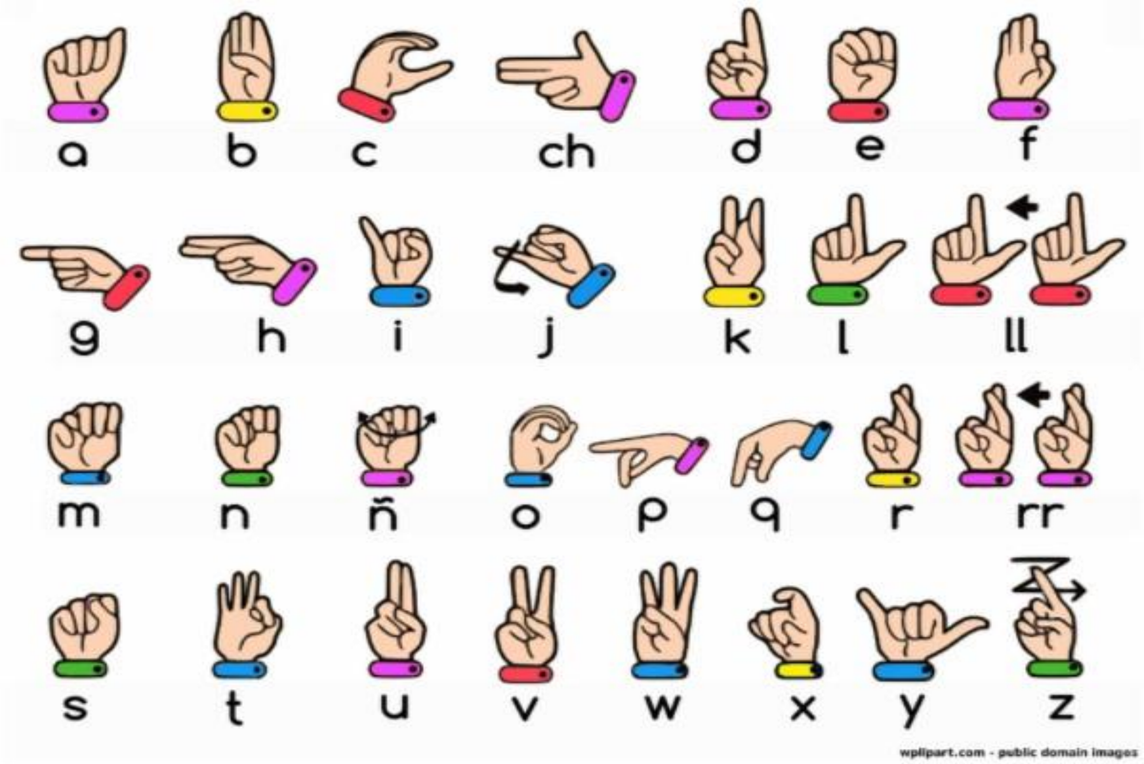


Figure 1.1: American Sign Language

1.2 Objectives

More than 70 million deaf individuals globally depend on sign languages as their main mode of communication. Sign language enables them to acquire knowledge, engage in work, utilize essential services, and actively contribute to their communities [7]. However, expecting universal proficiency in sign language presents a considerable challenge, often hindering individuals with disabilities from fully enjoying equal rights.

This project focuses on creating an intuitive Human-Computer Interface (HCI) designed to recognize and interpret American Sign Language (ASL) [8]. By harnessing cutting-edge technology, the system aims to assist individuals with hearing and speech impairments, making their interactions and everyday activities more seamless.

The centerpiece of this project involves developing software that utilizes Convolutional Neural Networks (CNNs) to analyze images of ASL hand gestures. The trained model deciphers these gestures, converts them into text, and subsequently transforms the text into audible speech. This groundbreaking method promotes accessibility and inclusivity, offering a viable solution to bridge the communication divide for the deaf and hard-of-hearing community [8]

1.3 Scope

This system will serve as a bridge between individuals with hearing or speech impairments and those unfamiliar with sign language. Users simply need to perform sign language gestures, which the system will recognize and interpret. After processing the gestures, the system will provide the corresponding output in both text and speech formats, enabling effective communication and fostering inclusivity between the two groups [7].

1.4 Limitations

- **Dependency on Input Quality**: Gesture recognition accuracy is affected by lighting, camera quality, and background interference.
- **Handling Complex Gestures**: The system struggles with fast or complex gestures, especially those with subtle hand movements.
- **Language Limitation**: Currently, the system only supports American Sign Language (ASL), limiting its use for other sign languages.
- **Individual Signing Variations**: The model may not fully recognize individual signing styles or regional differences.

- **Hardware Requirements**: High-resolution cameras and processing power are needed, making it less accessible in low-resource settings.

1.5 Disposition

The rest of the document is structured as follows: In Chapter 2 theories and previews studies related to the topic will be presented. The suggested framework for activity recognition is presented in Chapter 3. Chapter 4 provides the experimental setup, results discussion, and comparative analysis. In Chapter 5, the conclusion and next works are provided.

CHAPTER 2

LITERATURE REVIEW AND MODULES

2.1 Overview

Various hand gestures have been recognized using different methods by researchers and have been implemented across a wide range of fields. This study focuses on developing a system that translates American Sign Language (ASL) finger-spellings into text and speech using Convolutional Neural Networks (CNNs) [8]. The recognition techniques utilized in this system are categorized into three main approaches: Hand Segmentation Approaches, Feature Extraction Approaches, and Gesture Recognition Approaches [7].

Hand Segmentation Approaches involve methods used to isolate the hand from the background and remove unnecessary elements to ensure accurate input. Feature Extraction Approaches focus on techniques to identify and extract distinctive features from hand gestures, such as shape, position, or motion. Gesture Recognition Approaches utilize algorithms, including CNNs, to classify the extracted features and translate them into meaningful text and speech.[9]

This approach integrates these methods to achieve precise and efficient gesture recognition, contributing to the advancement of communication technologies for individuals with hearing or speech impairments.

2.2 Project Modules & Requirements

2.2.1 Project Modules

I. Data Acquisition
II. Data pre-processing and Feature extraction
III. Gesture Classification

IV. Text Translation

V. Speech Translation

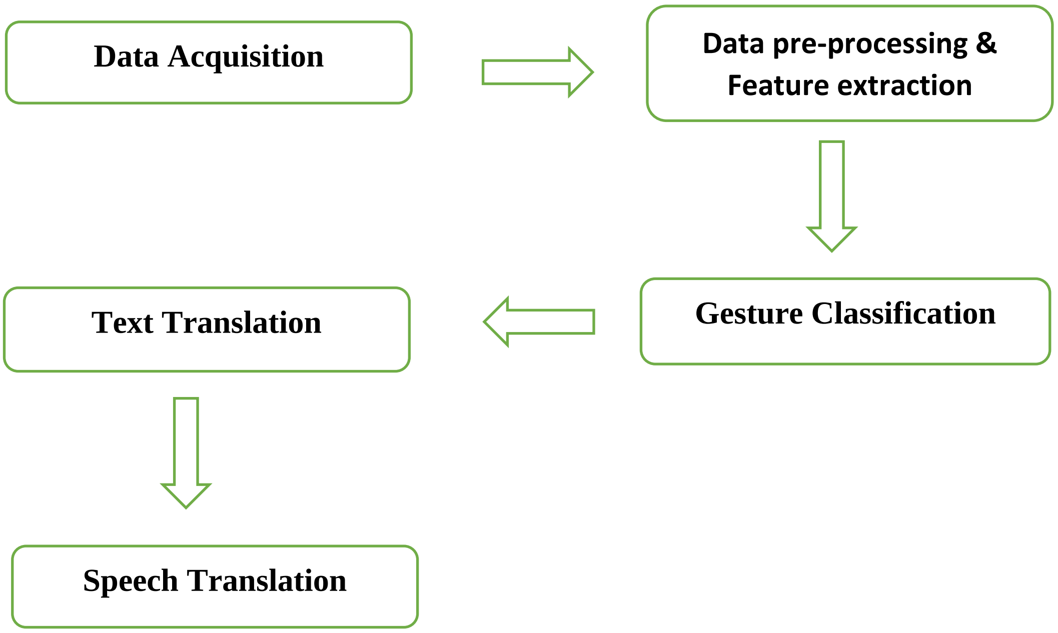


Figure. 2.1 Flow Diagram of Project Modules

2.2.2 Project Requirements

1. Hardware Requirement
   A. Webcam

2. Software Requirement
   A. Operating System: Windows 8 and Above
   B. IDE: VS Code or PyCharm or Jupiter Notebook
   C. Programming Language: Python 3.95
   D. Python libraries: NumPy, OpenCV, Mediapipe, Keras, Tensorflow

2.3 Related Works

2.3.1 Translation of Sign Language Finger-Spelling to Text using Image Processing

The proposed system is designed to identify key components of sign language and convert them into text. The process starts by capturing video input on a frame-by-frame basis. Each frame undergoes processing to isolate the relevant image, which is then analyzed using BLOB (Binary Large Object) techniques. The processed image is matched against a statistical database of predefined gestures. The system determines the corresponding alphabet sign by identifying the closest match within the database [10].

This implementation focuses specifically on American Sign Language (ASL) finger-spellings, enabling the formation of words and sentences by combining identified letters. The system achieves a recognition accuracy of approximately 93%, which demonstrates its potential for larger-scale applications [9, 10].

Additionally, the system is designed to be cost-effective and user-friendly by relying on a standard webcam for input. By converting static gestures into text, this technology facilitates communication between individuals who use sign language and those who do not.. Although the current implementation is limited to ASL static finger-spellings, it lays a foundation for future advancements to include dynamic gestures, other sign languages, and real-time applications in diverse environments [10].

This approach offers a practical and scalable solution, ensuring that individuals who rely on sign language can communicate more effectively in everyday scenarios [11].

2.3.2 Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network

The paper proposes a desktop application that leverages a computer's webcam and convolutional neural networks (CNNs) to interpret American Sign Language (ASL) gestures into text and speech in real time [11]. The system uses CNNs to process visual data, extract hand gestures, and classify them with high accuracy. The application translates ASL finger-spelling gestures into text, which is then converted to audio using a text-to-speech module. The approach emphasizes accessibility, as it requires only a webcam for gesture recognition, eliminating the need for additional hardware [12].

The methodology involves multiple steps: image acquisition through a webcam, segmentation of hand gestures, preprocessing of images to enhance feature detection, and classification using a pre-trained CNN model. The CNN architecture includes convolutional, pooling, and dense layers, enabling it to detect features at varying complexities. The model achieves 95% accuracy in recognizing ASL gestures [12]. The system supports real-time conversion, facilitating communication between deaf individuals and the wider community without the need for an interpreter. The authors also discuss the importance of consistent lighting and potential use of gloves to improve gesture detection in diverse environments.

The research highlights the limitations of the current implementation, such as its exclusive focus on ASL finger-spelling gestures and the inability to process context-based signing. Future enhancements include expanding the system to support other sign languages, deploying it as a mobile or web application, and integrating natural language processing for contextual understanding. This project showcases how deep learning can address societal challenges, promoting inclusivity for individuals with hearing impairments [11].

2.3.3 Sign Language to Text and Speech Conversion

The paper outlines a system designed to bridge the communication gap between sign language users and non-users by converting American Sign Language (ASL) into text and speech using Convolutional Neural Networks (CNNs) [13,14]. The project utilizes image-based input captured through a camera, which is then processed for gesture recognition. The CNN model is trained to classify gestures, converting them into corresponding text and speech outputs using the GTTS library. This innovative method focuses on real-time gesture recognition, employing a sequence of steps including image preprocessing, hand gesture scanning, classification, and translation. The system achieves an impressive accuracy of 95.8%, demonstrating significant potential for aiding individuals who rely on sign language [13].

The study compares existing techniques for gesture recognition, highlighting the effectiveness of CNNs in addressing challenges such as varying backgrounds and lighting conditions. Unlike older methods based on geometric invariants or Naive Bayes classifiers, the proposed model processes gestures using a deep learning approach that ensures greater precision and adaptability. The system's design eliminates the need for additional hardware by leveraging visual input and software-based processing, making it accessible and cost-effective. The researchers also suggest potential enhancements, such as deploying the system as a web or mobile application and improving preprocessing techniques for low-light environments.

The findings emphasize the practicality of the CNN-based framework for real-time sign language interpretation [13]. The system's modular approach, which includes a user-friendly graphical interface, can significantly improve communication for the hearing-impaired community. While the project currently focuses on ASL, it lays a foundation for further expansion into other sign languages, increased vocabulary, and better contextual understanding. The study also suggests future improvements in gesture detection accuracy, background noise handling, and deployment scalability.

2.3.4 Conversion of Sign Language to Text and Speech Using Machine Learning Techniques

Communicating with individuals who are hearing impaired (deaf/mute) remains a significant challenge in society, primarily due to the need for interpreters when using sign language or local hand gestures. This system aims to bridge that gap by converting American Sign Language (ASL) gestures into both text and speech, utilizing unsupervised feature learning to eliminate communication barriers and assist in teaching sign language [14]. For this project, sample images of various ASL signs were collected using the Kinect sensor and the MATLAB image acquisition toolbox. A dataset of approximately 500 samples, with 5 to 10 variations per sign, was gathered to train the model [14]. The goal was to enhance the robustness of the algorithm by diversifying the training data, which helps reduce the likelihood of misclassification. The combination of FAST and SURF feature extraction techniques, along with a 10-nearest neighbor (KNN) classifier, demonstrated the effectiveness of unsupervised learning in matching the most relevant features from the database. These matched features were then converted into text and speech outputs. The system achieved an accuracy of 92% with supervised learning and 78% with unsupervised feature learning [14].

2.3.5 An Improved Hand Gesture Recognition Algorithm based on image contours to Identify the American Sign Language

This research proposes an efficient and cost-effective hand gesture recognition system to facilitate communication for hearing-impaired individuals by identifying American Sign Language (ASL) [1]. The system uses image processing techniques, specifically contour analysis and convexity defect calculations, to recognize static and dynamic ASL gestures. It leverages OpenCV libraries for image capture and feature extraction, enabling the identification of 26 letters and digits (0-9) without relying on specialized hardware like gloves. By focusing on single-hand gestures, the approach reduces computational complexity and hardware requirements, making it accessible to users with standard devices [7].

The proposed system achieves an average recognition accuracy of 86%, demonstrating its reliability and practicality for real-time applications [15]. Its output includes text and speech, enabling seamless communication. The study highlights the system's affordability and simplicity, contrasting it with existing methods that rely on advanced machine learning or expensive hardware. Future work aims to expand its usability to mobile devices, further enhancing accessibility for hearing-impaired individuals in daily life.

2.4 Comparison Table

Author name	**Krishna Modi and Amrita**	**Ankit Ojha, Shubham Maurya**	**Bikash K. Yadav and Dheeraj Jadhav**	**Victorial Adebimpe Akano**	**Rakesh Kumar**
Algorithm	**Blob Analysis**	**CNN**	**CNN**	**KNN**	**contour measure men**
Accuracy	**93%**	**95%**	**95.8%**	**92%**	**86%**
Year	**2013**	**2020**	**2020**	**2018**	**2021**

2.5 Research Gap

1. In the First research paper [10], signs are recognized by directly comparing image pixels stored in their database. They also converted RGB images to binary format but removed some essential features during image processing.
2. In the second and third research papers [11] and [12], the CNN algorithm was utilized for sign recognition, which proved highly effective. However, minimal image preprocessing was performed before training the CNN model.
3. In the fourth research paper [13], the simplest algorithm, KNN, was used for sign recognition. Limited image processing might explain the algorithm's moderate accuracy.
4. In the fifth research paper [14], contour and convexity measurements were applied for image recognition, but the algorithm resulted in lower accuracy.

2.6 Project Feasibility Study

2.6.1 Operational feasibility

1. The primary goal of this system is to perform tasks with greater accuracy and efficiency while minimizing time consumption.
2. The application is designed to be user-friendly, requiring only a basic understanding of American Sign Language to operate effectively.
3. This system is highly feasible for practical use, as it is simple and intuitive for end users to operate. It requires only fundamental familiarity with Windows applications, ensuring accessibility for a broad audience.
4. Additionally, the streamlined interface and efficient design make it suitable for individuals with minimal technical expertise, enhancing its overall usability.

2.6.2 Technical feasibility

The system's technical requirements include careful selection of both the front-end and back-end platforms. Choosing the right tools is crucial for successful project development and ensuring compatibility with organizational needs. To identify the most suitable platform, an in-depth analysis was conducted, considering various factors [15].

Front-End Selection: To make the system accessible to users without an IT background, the front-end was developed using Python's Tkinter GUI framework. This choice provides a user-friendly graphical interface, ensuring ease of use [2].

Key Features of the Front-End:

1. **Scalability and Extensibility:** The design supports future growth and additional functionalities.
2. **Flexibility:** The interface is adaptable, accommodating diverse user requirements.
3. **Ease of Debugging and Maintenance:** The framework simplifies troubleshooting and upkeep, ensuring efficient operation.

This approach ensures the system is both user-centric and technically robust.

Back-end Selection: We have used Python as our Back-end Language which has the most widest library collections The technical feasibility is frequently the most difficult area encountered at this stage. Our app will fit perfectly for technical feasibility.

2.6.3 Economic Feasibility

The development of the system must be evaluated in terms of cost-effectiveness and benefits. It is crucial to focus efforts on projects that promise the best returns within the shortest timeframe. One key consideration in developing a new system is the associated cost. However, since this system is being developed as part of a project, there are no labor costs involved. Additionally, all required resources are readily available, making the development process economically feasible. This ensures that the system is both cost-efficient and practical for implementation.

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

3.1 Data Acquisition

Data acquisition for hand gesture recognition can be achieved through various approaches:

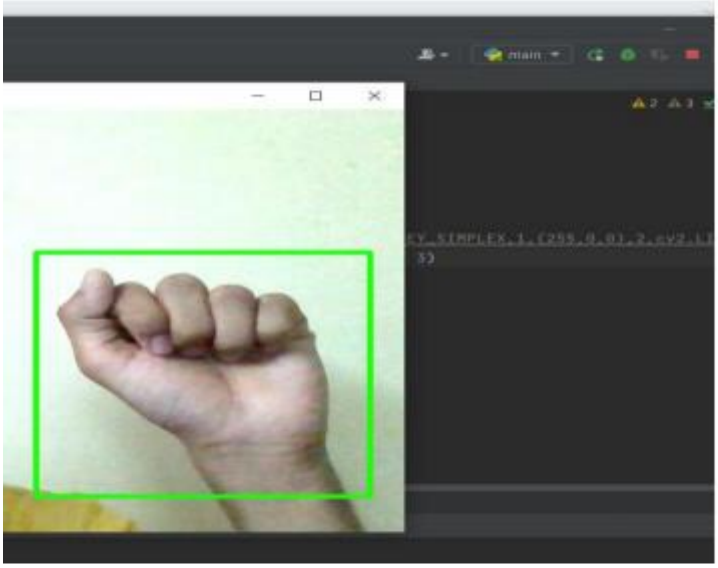
Electromechanical Devices: These devices provide precise information about hand configuration and position. Glove-based methods are a common example, offering detailed data extraction. However, these methods are often costly and lack user-friendliness [15].

Vision-Based Techniques: Vision-Based Techniques: This method utilizes a standard webcam to capture hand and finger movements. Unlike glove-based systems, vision-based approaches rely solely on a camera, allowing seamless interaction between humans and computers without requiring extra hardware, which helps minimize expenses[16].

The primary challenge with vision-based hand detection lies in addressing the wide variability in hand appearances caused by numerous hand movements, diverse skin tones, and variations in viewpoints [13].

In this study, we employed vision-based techniques, utilizing a computer webcam to capture images and create our custom dataset. Figure 3.1 illustrates the vision-based hand detection process, where the green box highlights the region of interest (ROI), indicating successful hand detection.

This approach allows for non-intrusive, hardware-minimal data acquisition, making it a cost-effective and scalable solution. Despite the simplicity, vision-based techniques still face challenges, such as managing complex backgrounds, varying lighting conditions, and diverse hand poses, which are addressed through robust image processing and machine learning algorithms.



```
FY_SIMPLEX,1,(255,0,0),2,cv2.LI
 3)
```

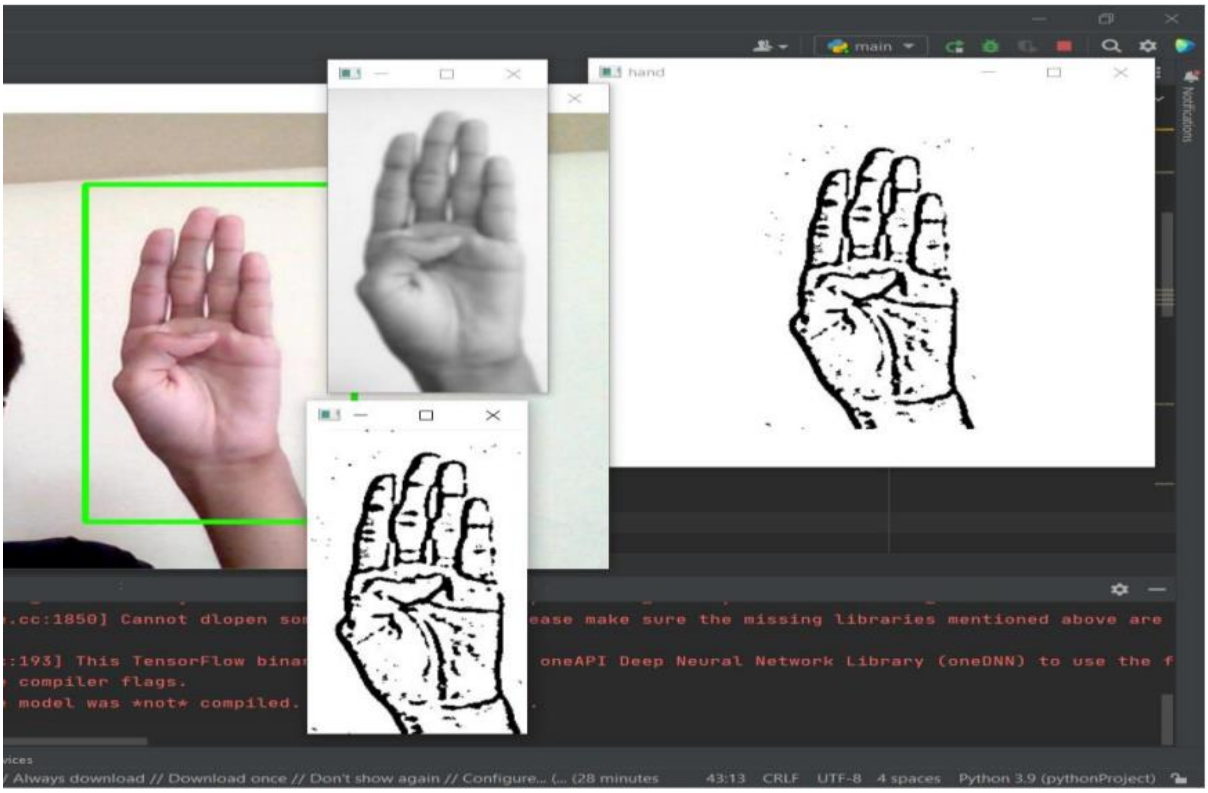
Figure 3.1 Vision-based Hand Addressing

3.2 Data pre-processing and Feature extraction

1. In this hand detection approach, the first step is to detect the hand from the image captured by the webcam. For this, we use the MediaPipe library, which is designed for image processing [11]. Once the hand is detected, we isolate the region of interest (ROI), crop the image, and convert it to grayscale using the OpenCV library. Afterward, we apply Gaussian blur to smooth the image. This filter is easily applied using OpenCV. To further process the image, we convert it into a binary image using both thresholding and adaptive thresholding methods [18].

2. To train the system for accurate hand sign recognition, we collected a diverse dataset of hand sign images representing the letters A to Z. These images were captured from multiple angles to account for variations in hand positions and orientations, thereby improving the robustness and accuracy of the recognition process [19].

Figures 3.2 and 3.3 depict the process of detecting a hand from an image captured via the webcam. The process begins with capturing the image using the MediaPipe library [18]. Subsequently, the region of interest (ROI) is identified, the image is cropped, and converted into grayscale. Lastly, Gaussian blur is applied to enhance smoothness and minimize noise.



```
.cc:1850] Cannot dlopen so                    ease make sure the missing libraries mentioned above are

:193] This TensorFlow bina                    oneAPI Deep Neural Network Library (oneDNN) to use the f
 compiler flags.
 model was *not* compiled.
```

main
hand
Notifications
vices
/ Always download // Download once // Don't show again // Configure... (... (28 minutes
43:13
CRLF
UTF-8
4 spaces
Python 3.9 (pythonProject)

Figure 3.2: Hand detection approach

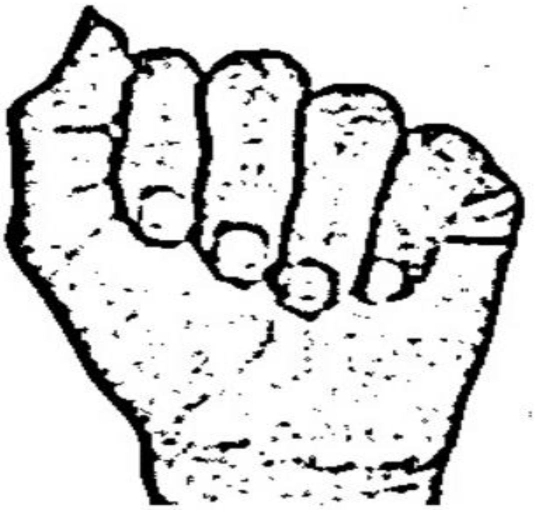
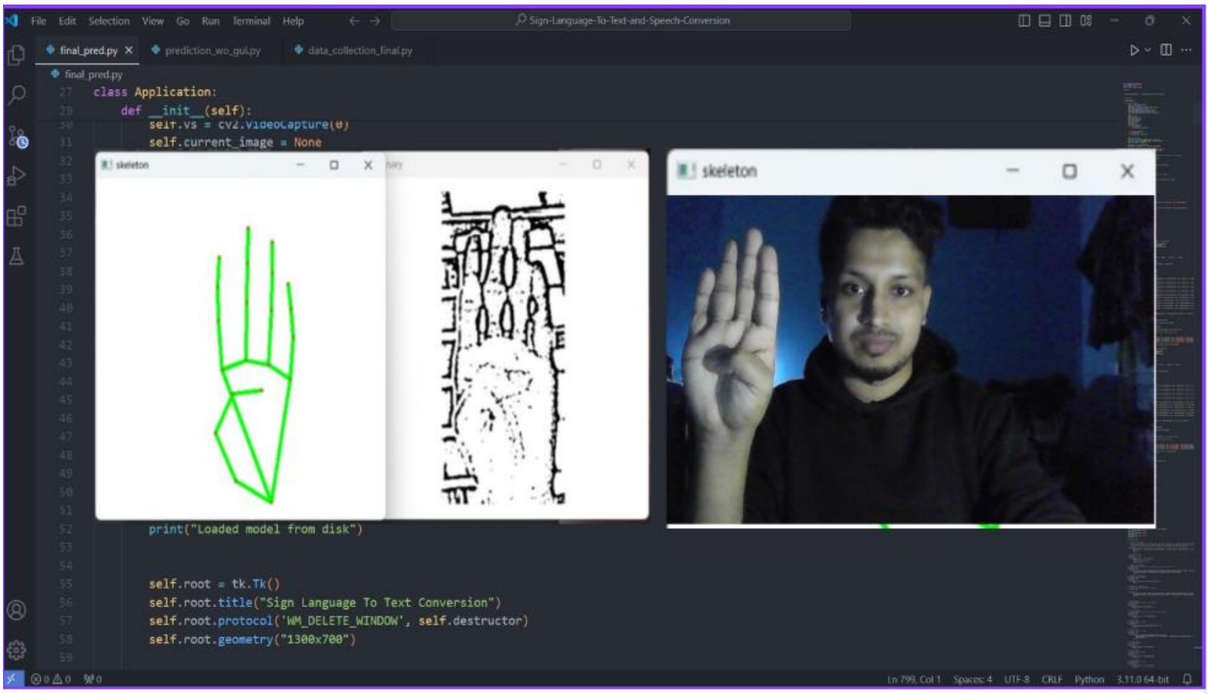


Figure 3.3: Final hand detection [20]

This method has several limitations, such as requiring the hand to be placed against a clean, uncluttered background with proper lighting conditions to achieve accurate results. However, in real-world settings, it is often challenging to control the background and lighting, which can lead to poor performance.

To overcome these challenges, we experimented with various approaches and found an effective solution. First, we use MediaPipe to detect the hand in the frame and extract the hand landmarks. Then, we draw and connect these landmarks on a plain white background, improving the clarity and accuracy of the hand detection process, as shown in the figure below.

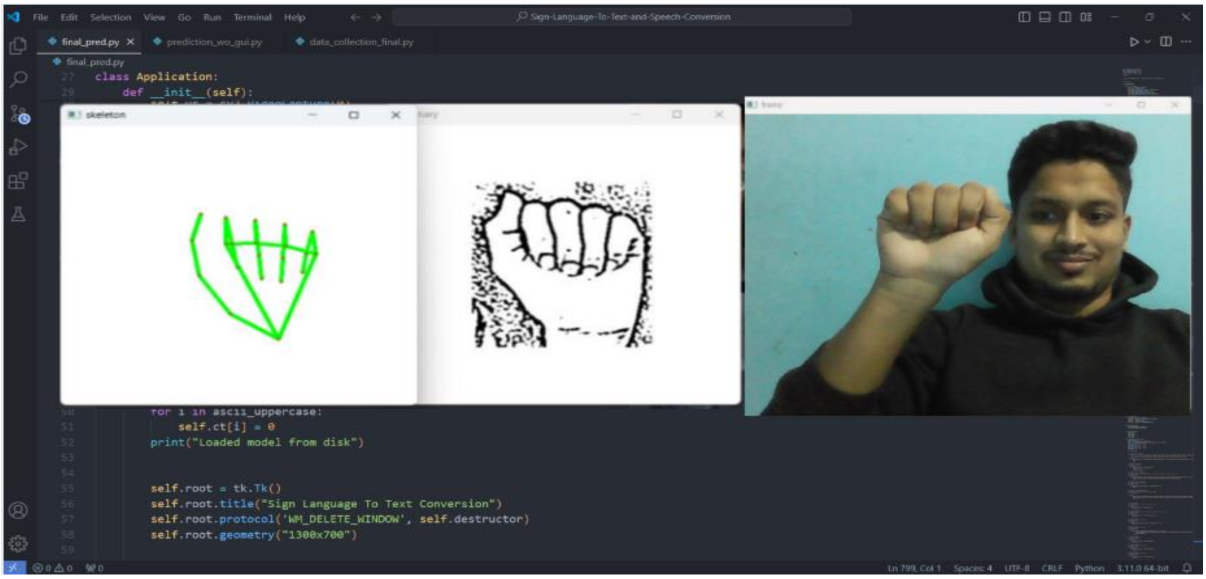

File Edit Selection View Go Run Terminal Help
Sign-Language-To-Text-and-Speech-Conversion
final_pred.py
prediction_wo_gui.py
data_collection_final.py
final_pred.py
skeleton
skeleton

```
class Application:
    def __init__(self):
        self.vs = cv2.VideoCapture(0)
        self.current_image = None

        print("Loaded model from disk")


        self.root = tk.Tk()
        self.root.title("Sign Language To Text Conversion")
        self.root.protocol('WM_DELETE_WINDOW', self.destructor)
        self.root.geometry("1300x700")
```

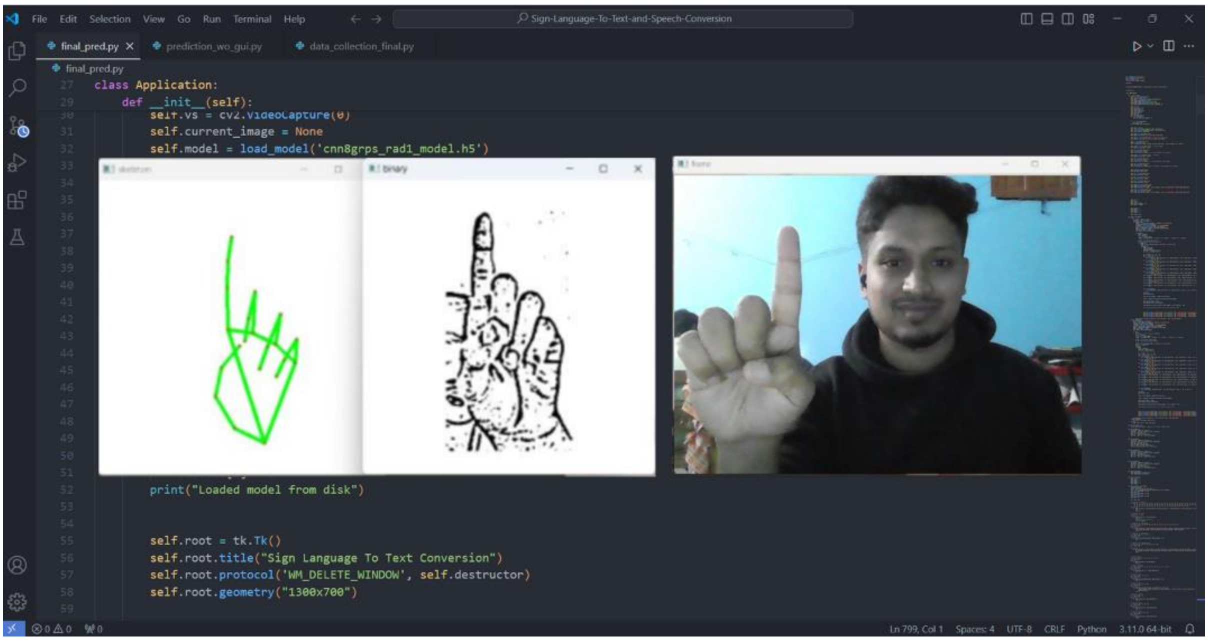
Figure 3.4: Hand Detection Approach in Background 1


File Edit Selection View Go Run Terminal Help
Sign-Language-To-Text-and-Speech-Conversion
final_pred.py
prediction_wo_gui.py
data_collection_final.py
final_pred.py
skeleton

```
class Application:
    def __init__(self):
        for i in ascii_uppercase:
            self.ct[i] = 0
        print("Loaded model from disk")


        self.root = tk.Tk()
        self.root.title("Sign Language To Text Conversion")
        self.root.protocol('WM_DELETE_WINDOW', self.destructor)
        self.root.geometry("1300x700")
```

Figure 3.5: Hand Detection Approach in Background 2


File Edit Selection View Go Run Terminal Help
Sign-Language-To-Text-and-Speech-Conversion
final_pred.py
prediction_wo_gui.py
data_collection_final.py
skeleton
binary
frame
Ln 799, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.0 64-bit

```
class Application:
    def __init__(self):
        self.vs = cv2.VideoCapture(0)
        self.current_image = None
        self.model = load_model('cnn8grps_rad1_model.h5')
        print("Loaded model from disk")


        self.root = tk.Tk()
        self.root.title("Sign Language To Text Conversion")
        self.root.protocol('WM_DELETE_WINDOW', self.destructor)
        self.root.geometry("1300x700")
```

Figure 3.6: Hand Detection Approach in Background 3

After this process we apply the apply MediaPipe Hands. These are described below

3.2.1 Mediapipe Landmark System:

The **MediaPipe Landmark System** is a powerful framework developed by Google that provides real-time, efficient, and accurate detection of key points (landmarks) on various parts of the body, such as the hands, face, and full-body poses. It is widely used for applications in hand tracking, gesture recognition, sign language translation, and other human-computer interaction scenarios. Here we work with Hand Tracking MediaPipe Hands [18].

The MediaPipe Hands model is a powerful tool for real-time hand tracking and gesture recognition [21]. It works by detecting key points (landmarks) on the human hand and analyzing the relative positions of these points to determine the hand's gesture. Here's how it functions in detail.

Working Procedure of MediaPipe Hands

1. **Landmarks Detection:** MediaPipe Hands detects 21 specific landmarks on each hand, covering the wrist, palm, and each of the five fingers. These landmarks are carefully mapped to specific locations on the hand, such as the knuckles, fingertips, and the base of each finger [22].
    - **Wrist** (1 landmark)
    - **Fingers** (4 landmarks per finger, totaling 20 for the five fingers)
2. **Tracking in Real-Time:** The model tracks these landmarks in real-time as the hand moves, using a webcam or any other camera input. The position of each landmark is

detected by MediaPipe and updated frame-by-frame, allowing the model to continuously track hand movements and gestures [22].

3. **Connecting Landmarks:** Once the landmarks are detected, MediaPipe connects them with lines to form a skeletal structure of the hand. This skeletal structure allows the system to understand the position and orientation of the hand. It can detect gestures such as open hands, fist shapes, or specific finger positions, which are key for gesture-based control systems [17].

4. **Gesture Recognition:** With the connected landmarks and their real-time positions, MediaPipe can interpret specific hand gestures. This makes it ideal for use in applications like virtual sign language interpreters, hand-controlled interfaces, or interactive devices [23].

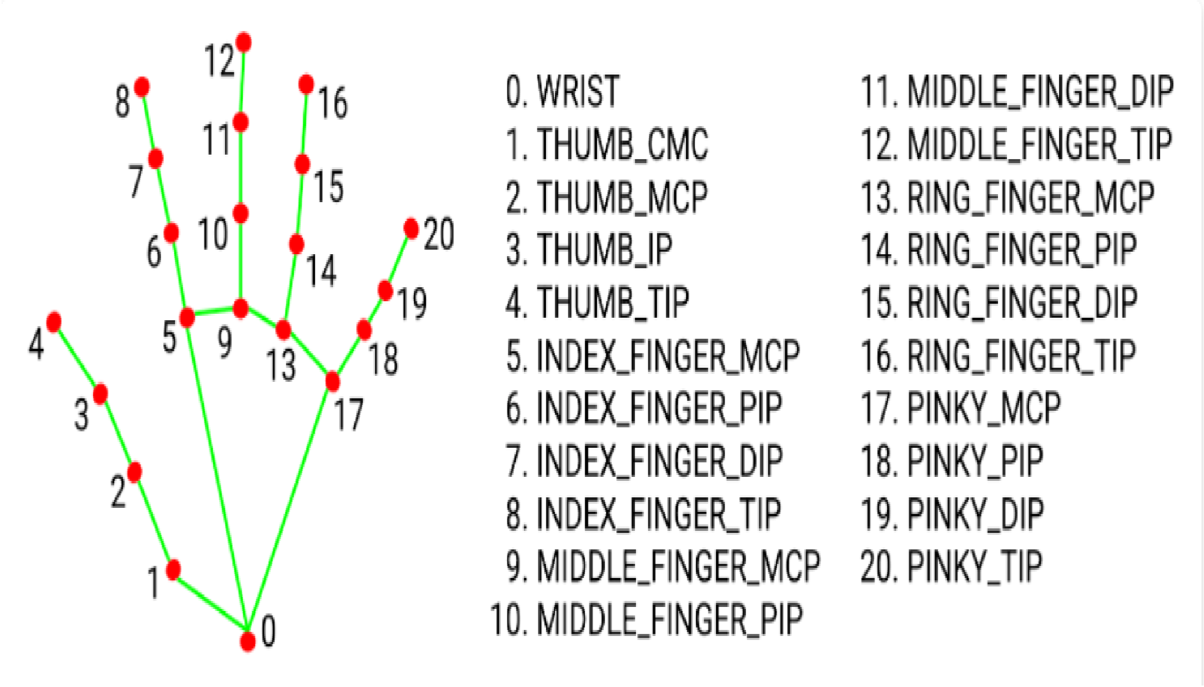


Figure 3.7: Landmark positions in the hand [24]

Applications

MediaPipe Hands is applicable in numerous use cases, such as:

1. **Sign language recognition**: Identifying specific letters or words in sign language.
2. **Gesture-based interfaces**: Allowing users to control devices or interact with digital environments using hand gestures.
3. **Virtual reality (VR) and augmented reality (AR)**: Providing hand tracking to enhance the user experience.

3.2.2 Draw Landmark point

Now, we will extract these landmark points and display them on a plain white background using the OpenCV library.

1. This approach addresses challenges related to background and lighting conditions, as the MediaPipe library provides accurate landmark points regardless of the background or lighting variations.
2. We have gathered a dataset of 180 skeletal images representing the alphabets from A to Z.

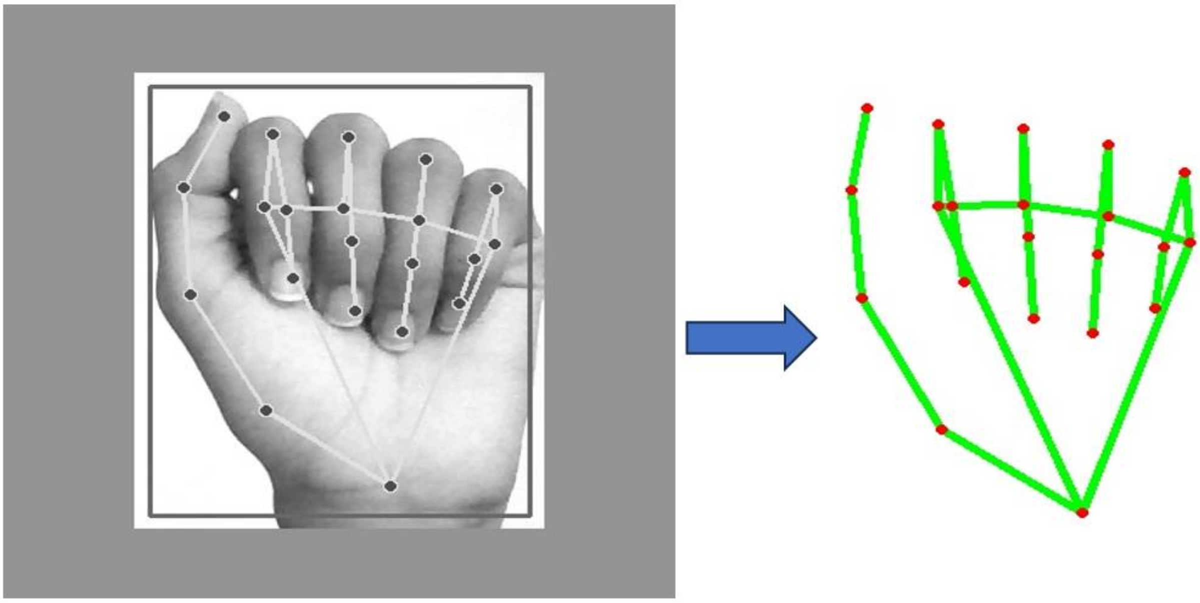


Figure 3.8: Hand Skeleton Mapping for 'A' from Landmarks [25]

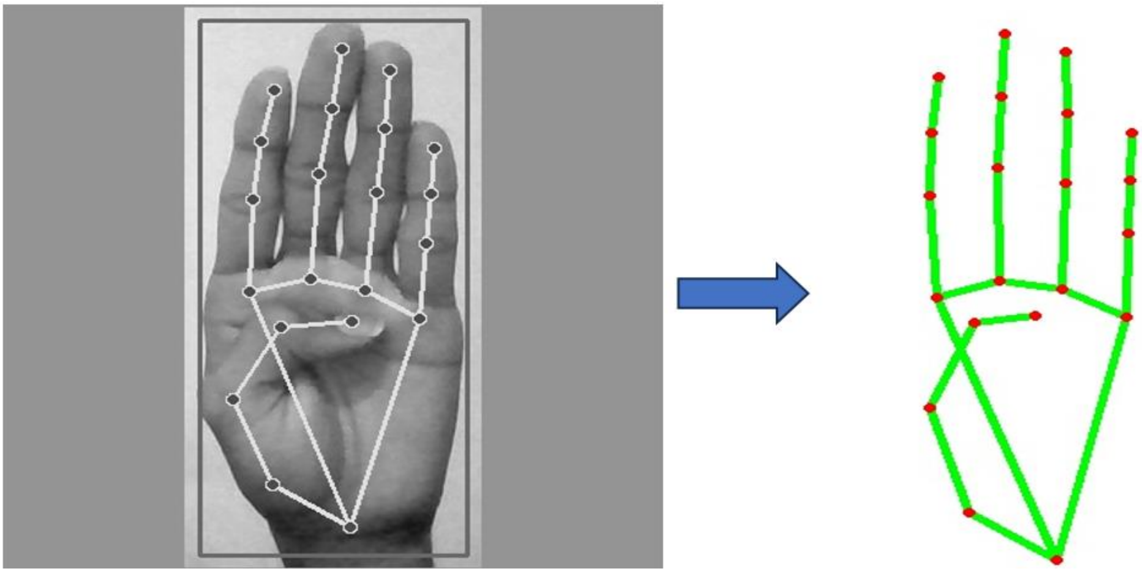


Figure 3.9: Hand Skeleton Mapping for 'B' from Landmarks [26]

3.3 Gesture Classification

3.3.1 Neural Networks

Neural Networks (NNs) are foundational models in machine learning, designed to simulate the way human brains process information [27]. They are composed of multiple layers of interconnected nodes (neurons) and are employed for various tasks, including classification, regression, and prediction. Basic neural networks are versatile, but their architecture can be adapted for specific tasks.

A typical Neural Network consists of three main types of layers

1. **Input Layer**: This is the initial layer that receives the raw data or features to be processed. Each neuron in this layer corresponds to one feature of the input dataset, and its role is to pass the data forward to the next layer without applying any transformations [27].

2. **Hidden Layers:** These layers sit between the input and output layers. They perform transformations on the input data by utilizing weights, biases, and activation functions. The hidden layers are responsible for enabling the network to learn and capture intricate patterns or associations within the data [27].

3. **Output Layer**: The final layer of the network produces the results based on the processed data from the hidden layers. The number of neurons in this layer depends on the task— such as one neuron for binary classification, several neurons for multi-class classification, or particular values for regression tasks [28].

As the data passes through the model, outputs are generated at each layer in a process referred to as feedforward. Once the outputs are obtained, the error is calculated using an appropriate error function, such as cross-entropy or mean squared error, to evaluate the network's performance. Afterward, the model goes through backpropagation, where gradients are computed to update the weights and biases. This important step is aimed at minimizing the loss and enhancing the model's accuracy over time [27].

3.3.2 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a specialized type of Deep Learning architecture primarily designed for tasks in Computer Vision [9]. Computer Vision, a subset of Artificial Intelligence, focuses on enabling computers to analyze and interpret visual data like images and videos.

Artificial Neural Networks excel in various Machine Learning tasks and are applied to diverse data types such as text, audio, and images [29]. Different neural network architectures are tailored to specific applications: for instance, Recurrent Neural Networks (RNNs), especially LSTMs, are ideal for sequential data like text, while Convolutional Neural Networks are the go-to choice for image classification. In this guide, we'll focus on creating a foundational component of CNNs.

3.3.3 CNN Architecture

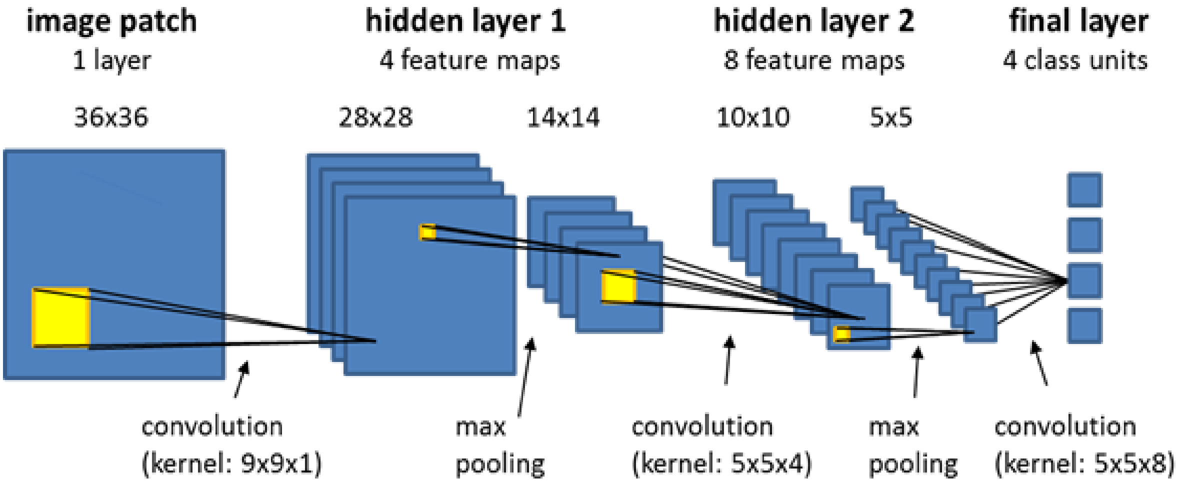


Figure 3.10: CNN Architecture [30]

A Convolutional Neural Network (CNN) is made up of several layers, including the input layer, convolutional layers, pooling layers, and fully connected layers.

The convolutional layers use filters to scan the input image and identify essential features, while the pooling layers down sample the feature maps, reducing their size and computational load. Finally, the fully connected layers process the extracted features to generate the final output. The network refines its filters during training using backpropagation and gradient descent, enabling it to learn the most relevant features for the task.

1. Convolutional Layer

In the convolutional layer, we use a small filter or kernel, typically sized 5x5, is applied, spanning the full depth of the input matrix. These filters are learnable parameters with the same dimensions. During each iteration, the filter moves across the input with a specified stride (commonly 1), computing the dot product between the filter values and the corresponding input values at each position [31].

This process generates a two-dimensional activation map that reflects the filter's response at different spatial locations. Essentially, the network learns to develop filters that activate when specific visual patterns, such as edges, textures, or color gradients, are detected in the input.

2. Pooling Layer

Pooling layers are strategically added to convolutional neural networks to down sample the feature maps. Their primary purpose is to reduce the spatial dimensions of the data, which speeds up computations, lowers memory usage, and helps mitigate overfitting. The two most frequently utilized pooling techniques are max pooling and average pooling [7]. For instance, applying max pooling with a 2x2 filter and a stride of 2 would produce an output volume with dimensions 16x16x12 [31, 32].

A. **Max Pooling A.** Max Pooling: Max pooling involves using a sliding window (e.g., 2x2) to process the input feature map. At each window position, only the maximum value is retained, while the others are discarded. The window slides across the feature map according to a defined stride, and this process is repeated across the entire feature map. The resulting output is an activation matrix with smaller dimensions, often half the size of the original, based on the chosen stride and window size [32].

B. **Average Pooling:** In average pooling, the average of all values within a window is calculated and retained as the representative value for that region.

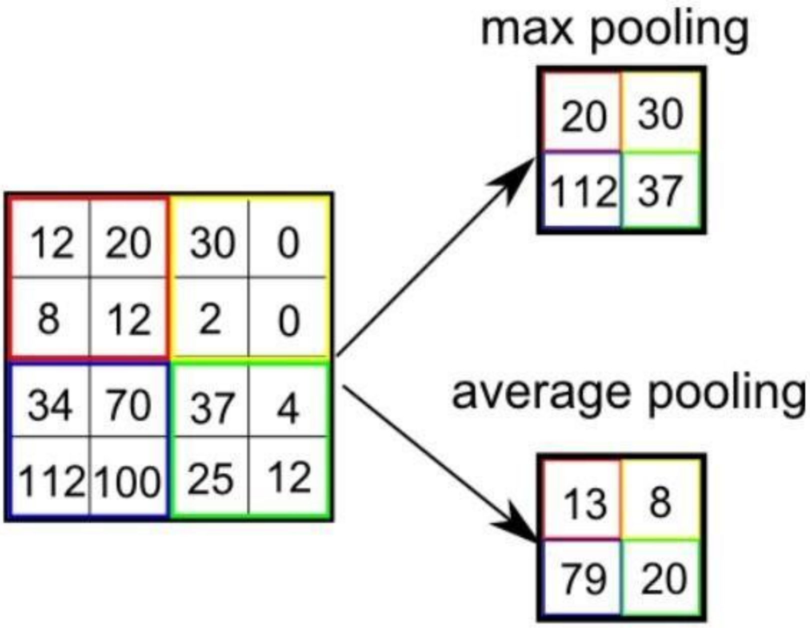


Figure 3.11: Pooling Layer [33]

4. Fully Connected Layer

It receives input from the revious layer and performs the final classification or regression task.

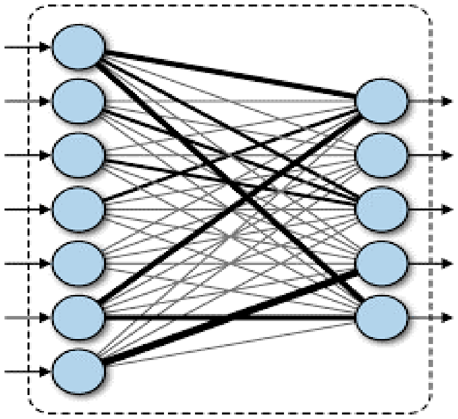


Figure 3.12: Fully Connected Layer [34]

The 180 preprocessed images for each alphabet will be input into the Keras CNN model. Due to low accuracy with 26 distinct classes, we grouped the 26 alphabets into 8 categories, each containing similar characters

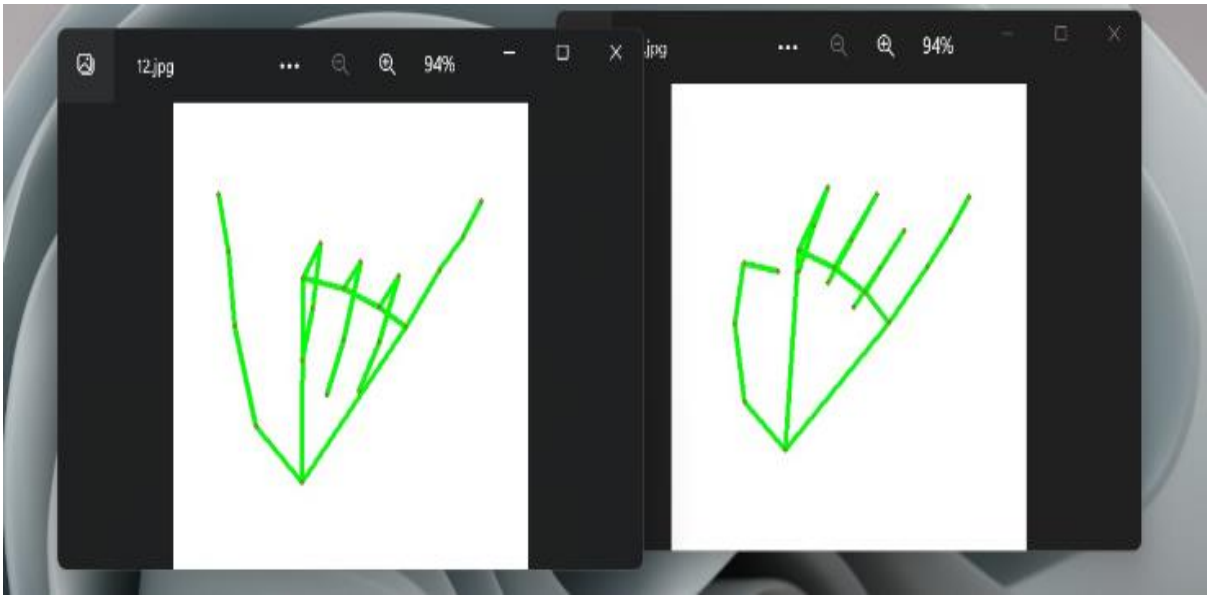


Figure 3.13: Class 1 for [Y, J]

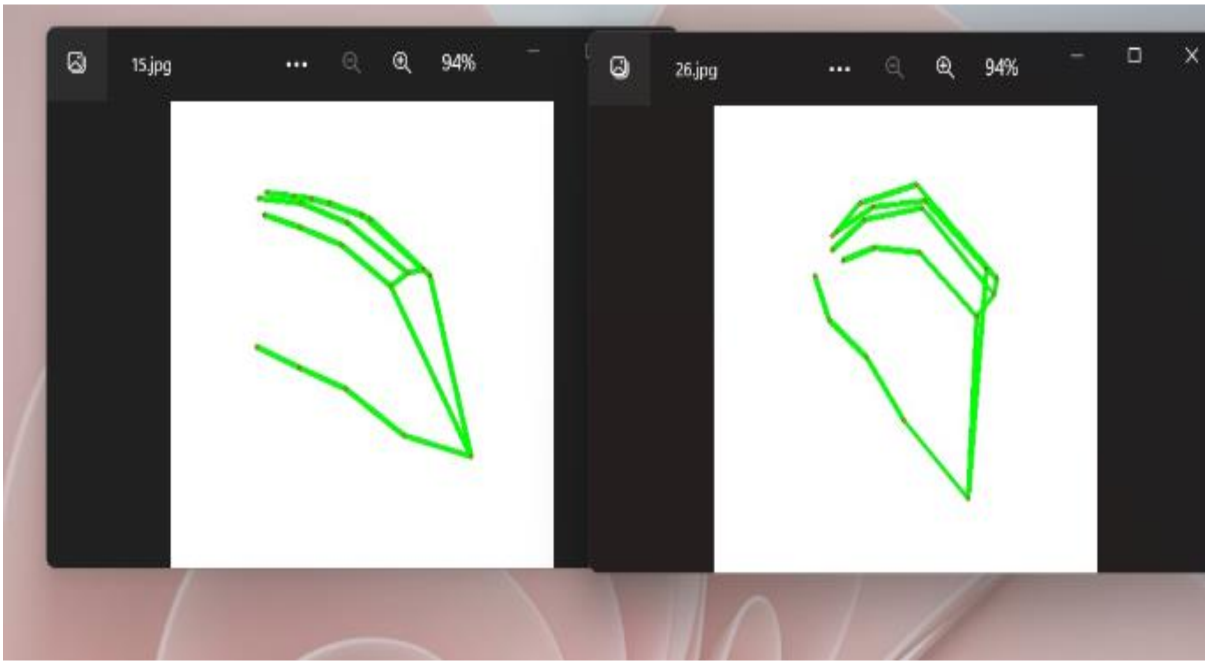


Figure 3.14: Class 2 for [C, O]

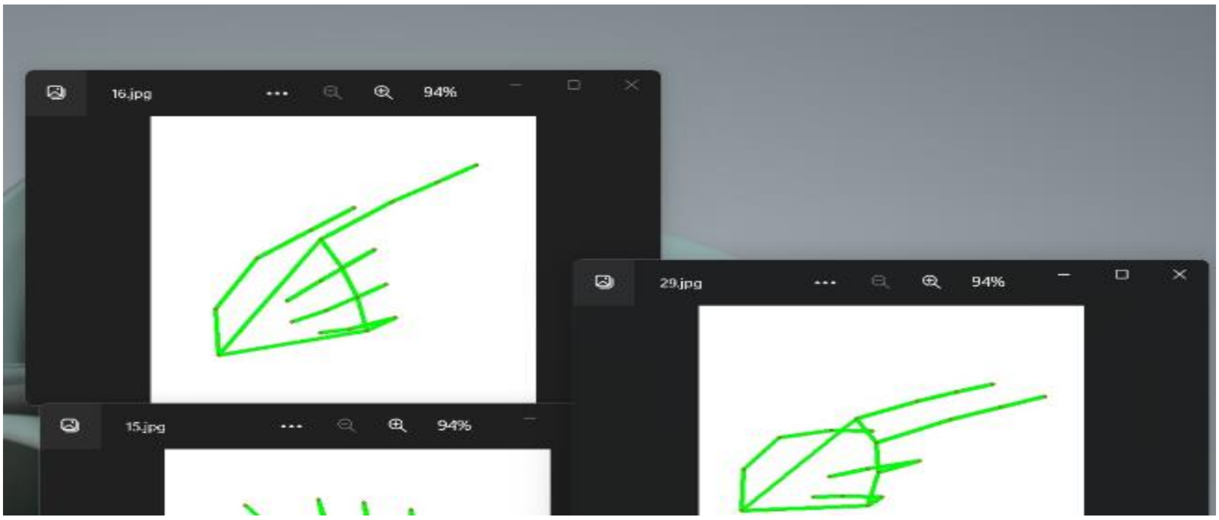


Figure 3.15: Class 4 for [G, H]

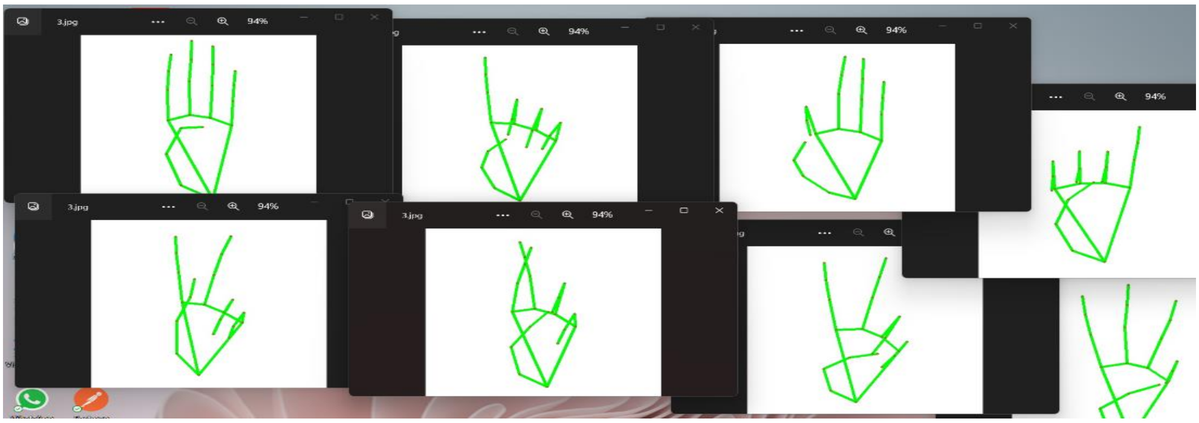


Figure 3.16: Class 4 for [B, H, F, L, U, V, K, R, W]

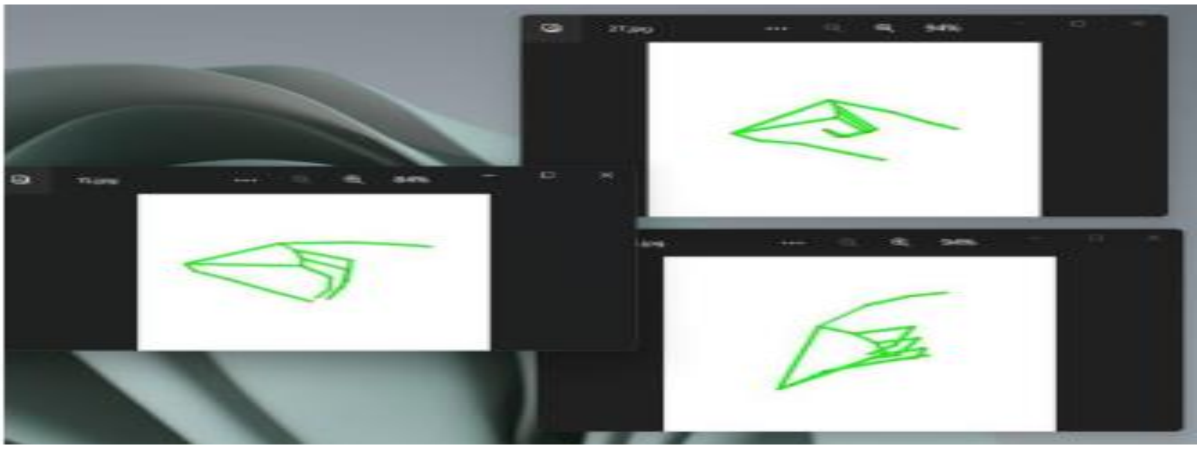


Figure 3.17: Class 5 for [P, Q, Z]

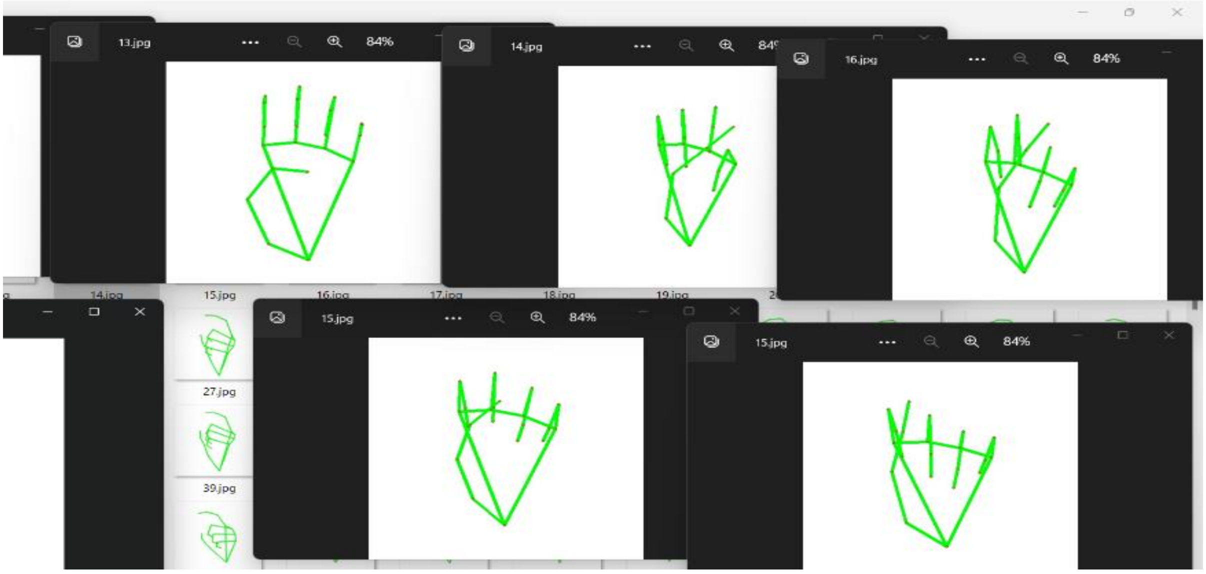


Figure 3.18: Class 6 for [A, E, M, N, S, T]

Each gesture label will be assigned a probability, and the label with the highest probability will be considered the predicted label. For example, when the model classifies [aemnst] as a single class based on mathematical operations applied to hand landmarks, we will further classify it into one of the individual alphabets: a, e, m, n, s, or t.

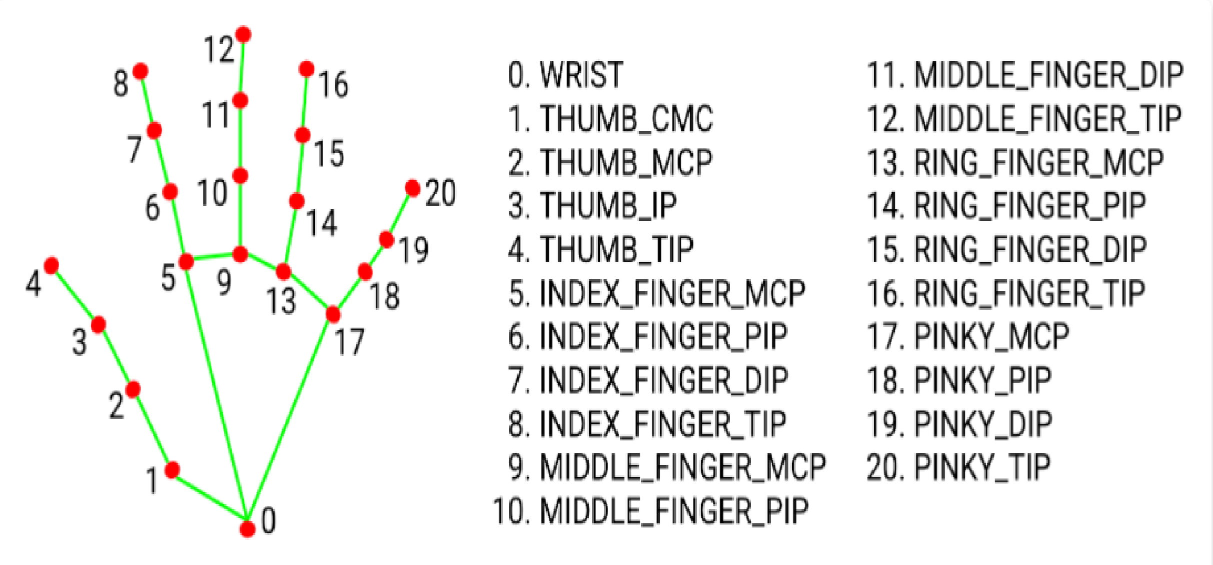


Figure 3.19: Landmark positions in the hand

3.4 Text and Speech Translation

The model converts recognized gestures into words. We have implemented the pyttsx3 library to transform these recognized words into speech. While the text-to-speech output is a basic solution, it serves as a valuable feature by simulating real-life communication.

3.5 Text to Speech Conversion

A Text-to-Speech (TTS) conversion system is a technology that converts written text. Here's a detailed explanation of how a typical TTS system works into spoken speech [35]. It enables computers or devices to read aloud text in a human-like virtual assistants, language learning apps, audiobooks, navigation systems, and more voice. TTS systems are widely used in various applications, including accessibility tools,

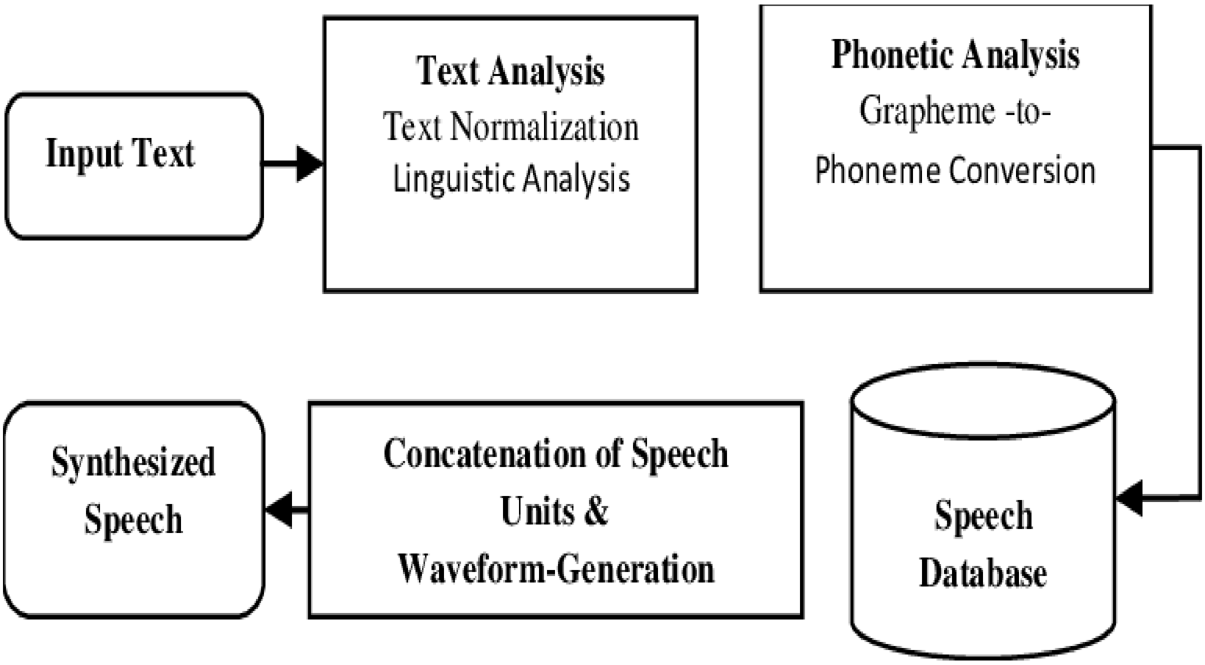


Figure 3.20: Text to Speech [36]

Text Preprocessing: The input text is first preprocessed to remove any special characters, formatting, or other non-textual elements. The system may also handle language-specific nuances, such as tokenization, part-of-speech tagging, and linguistic analysis [35].

Text Analysis: The preprocessed text undergoes linguistic analysis, which involves parsing the sentence structure, determining the syntactic relationships between words, and identifying the appropriate pronunciation for each word.

Phonetic Representation: The TTS system converts the analyzed text into phonetic representation. Phonetics refers to the study of the sounds used in human speech. Each word is broken down into phonemes, which are the smallest units of sound in a language [26].

Prosody and Emphasis: The TTS system also considers prosody, which includes aspects like pitch, duration, and intensity of speech. Emphasis and intonation patterns are determined based on the context and punctuation marks in the input text.

Language and Voice Selection: Modern TTS systems often support multiple languages and voices. The user can select the desired language and voice style to customize the output [28].

Voice Generation (Synthesis): This is the core process where the TTS system synthesizes the speech waveform based on the phonetic representation, prosody, and voice characteristics. There are generally two main approaches for voice generation:

- **Concatenative TTS:** This method uses a large database of recorded human speech, known as a speech corpus. The system selects and concatenates small audio units (phonemes, diphones, or triphones) from the corpus to create the desired speech. While this approach can produce high-quality, natural-sounding speech, it requires a large amount of data.
- **Parametric TTS:** This method relies on statistical models to generate speech. These models are trained on smaller datasets and use mathematical functions to represent speech. Parametric TTS systems can be more compact and require less storage but might sacrifice some naturalness [37].

Post-processing: The synthesized speech might undergo some post-processing to improve the output quality. Techniques like signal processing, smoothing, and noise reduction may be applied to make the speech sound more natural and clearer.

3.6 Project Software Requirements Specification (SRS)

3.6.1 System Flowchart

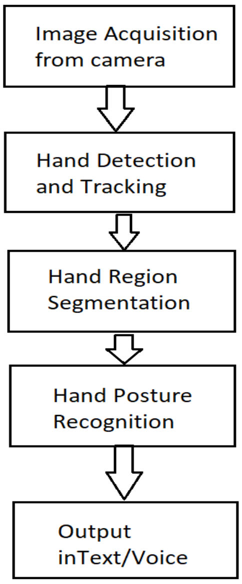


Figure 3.21: System Flowchart [38]

3.6.2 Use-case diagram

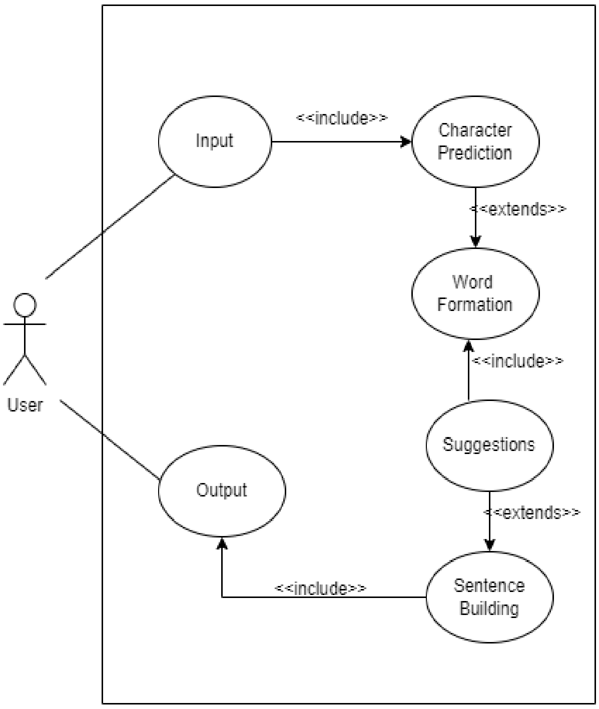


Figure 3.22: Use-case diagram [39]

3.6.3 Data Flow Diagram (DFD) diagram

DFD-Level 0

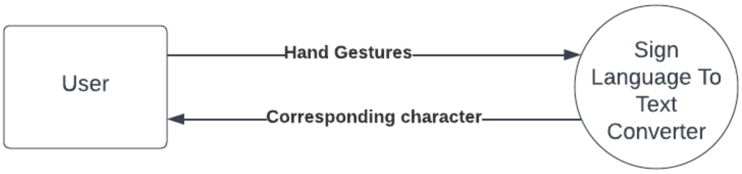


Figure 3.23: DFD-Level 0 [40]

DFD-Level 1

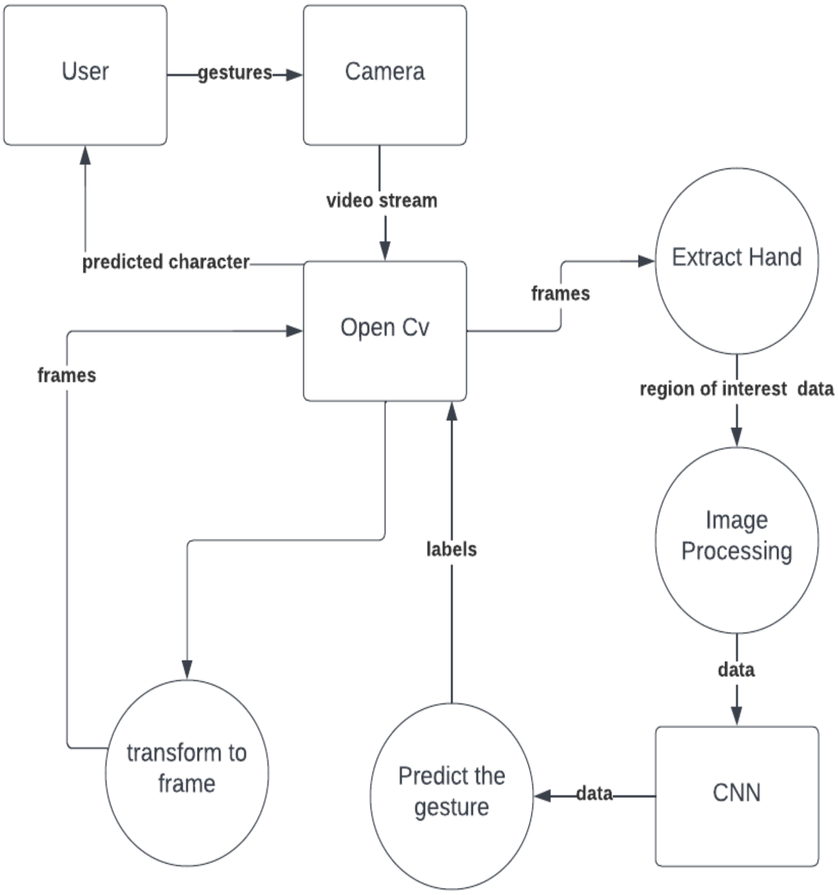


Figure 3.24: DFD-Level 0 [40]

3.6.4 Sequence diagram

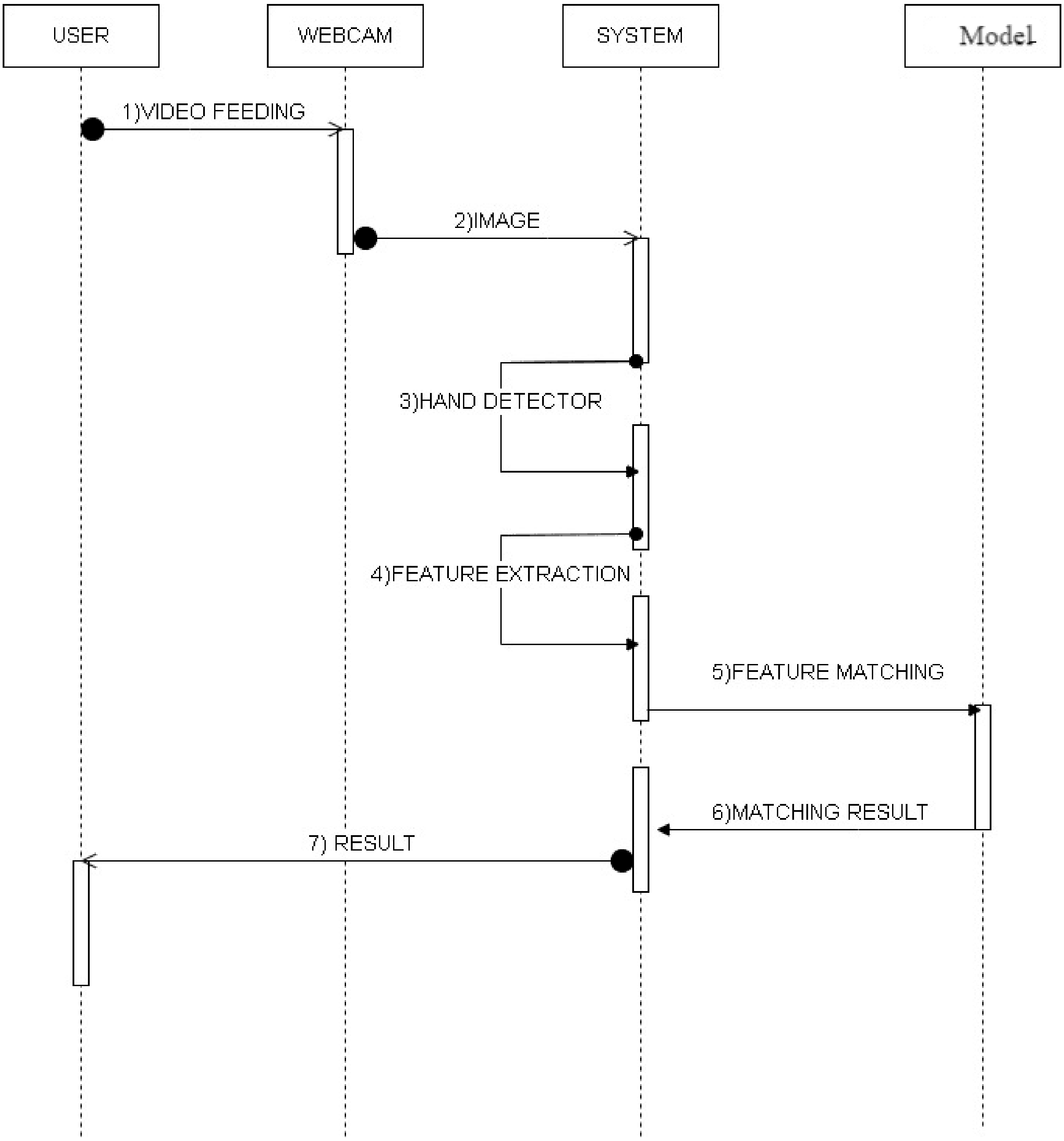
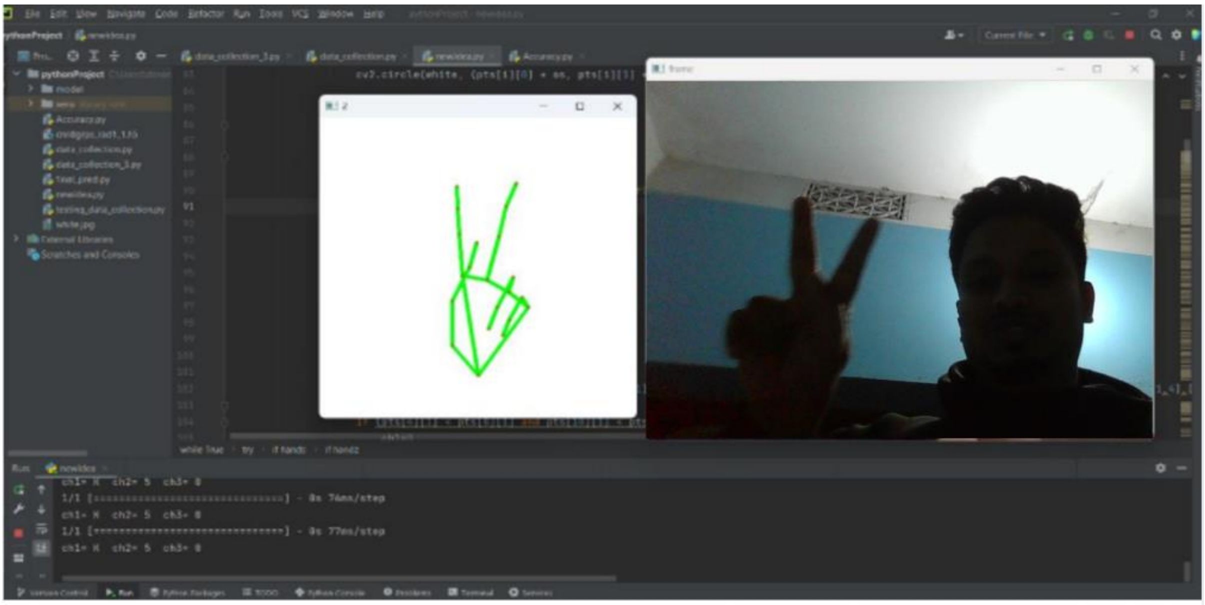


Figure 3.25: Sequence diagram [41]

CHAPTER 4

IMPLEMENTATION AND TESTING

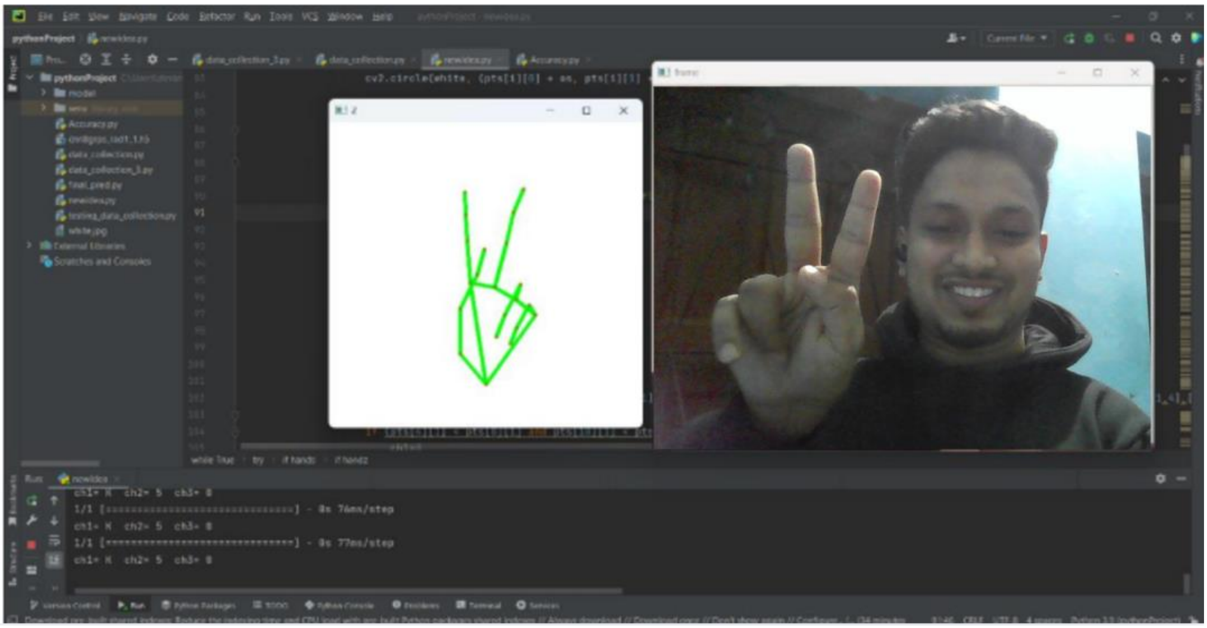
Here are some snapshots of a user displaying hand gestures under different backgrounds and lighting conditions, along with the system's corresponding predictions. In this instance, we aim to predict the gesture for 'K' in various backgrounds. The snapshots are attached below.



```
cv2.circle(white, (pts[t][0] + os, pts[t][1]
```

```
ch1= K  ch2= 5  ch3= 0
1/1 [==============================] - 0s 74ms/step
ch1= K  ch2= 5  ch3= 0
1/1 [==============================] - 0s 77ms/step
ch1= K  ch2= 5  ch3= 0
```

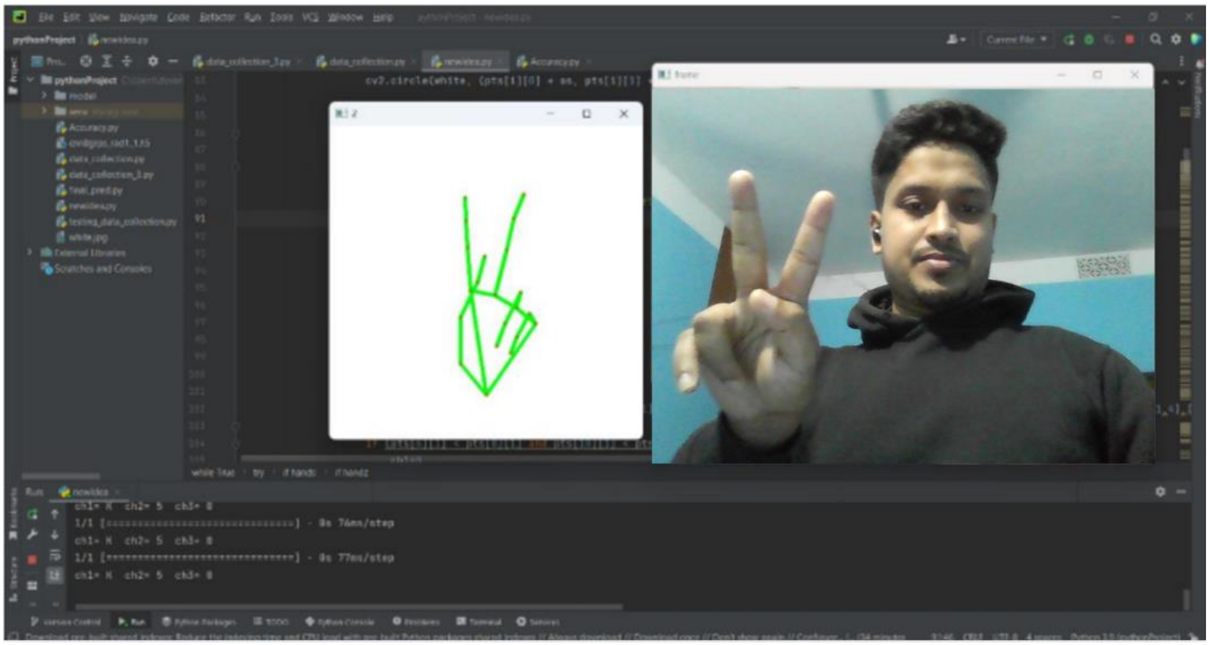
Figure 4.1: Predict "K" in background 1



```
cv2.circle(white, (pts[t][0] + os, pts[t][1]
```

```
ch1= K  ch2= 5  ch3= 0
1/1 [==============================] - 0s 74ms/step
ch1= K  ch2= 5  ch3= 0
1/1 [==============================] - 0s 77ms/step
ch1= K  ch2= 5  ch3= 0
```

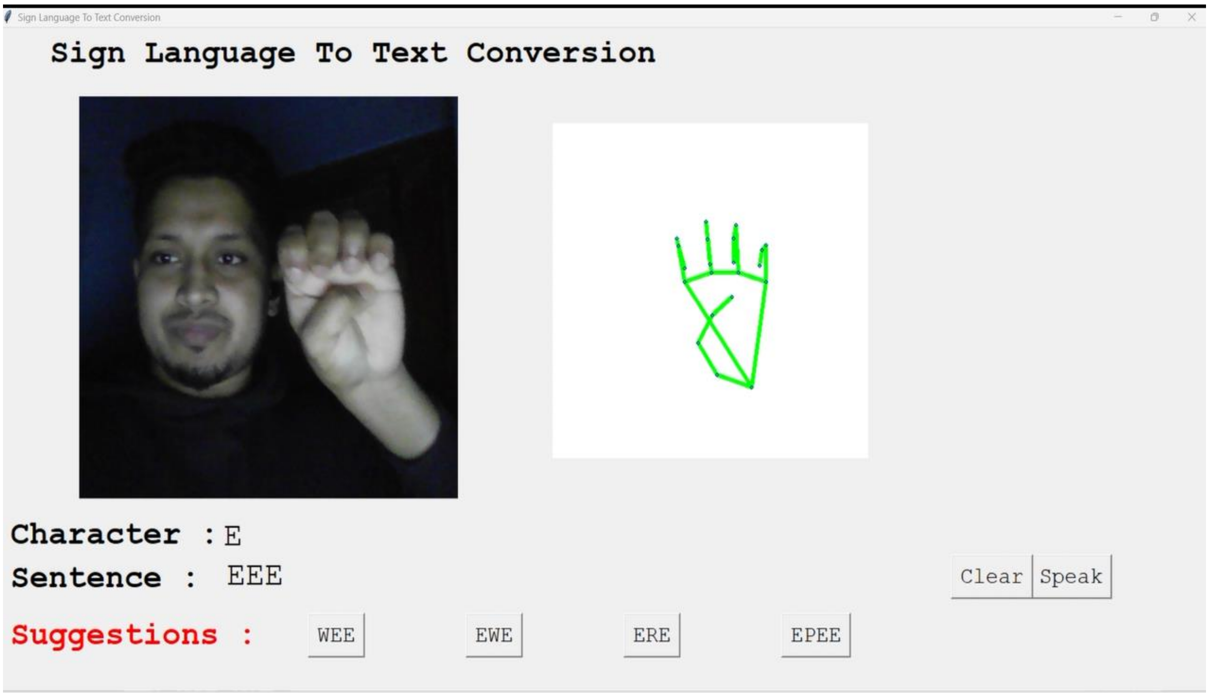
Figure 4.2: Predict "K" in background 2



```
ch1= K  ch2= 5  ch3= 8
1/1 [==============================] - 0s 76ms/step
ch1= K  ch2= 5  ch3= 8
1/1 [==============================] - 0s 77ms/step
ch1= K  ch2= 5  ch3= 8
```

Figure 4.3: Predict “K” in background 3

After implementing the CNN algorithm, we developed a GUI (Graphical User Interface) using Python's Tkinter and incorporated suggestions to enhance the user experience. Below are some snapshots of the final results. Here, I demonstrate the letter 'E' in various background positions. The system performed well, providing accurate predictions and appropriate letter suggestions. The snapshots are attached below.



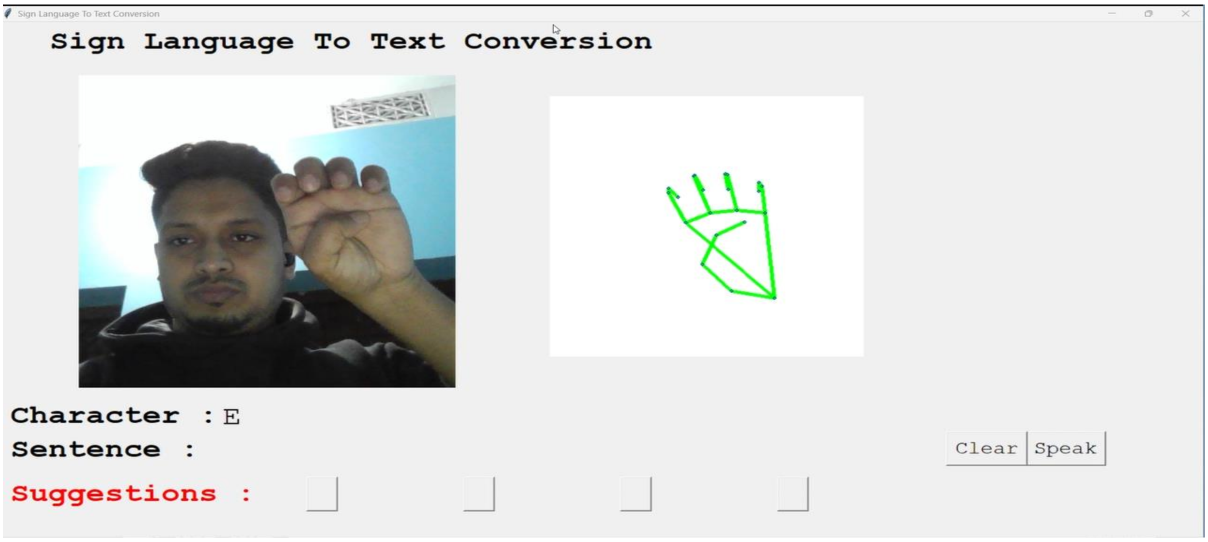
Sign Language To Text Conversion

Character : E

Sentence : EEE

Suggestions :

Figure 4.4: Final predict “E” in background 1



Sign Language To Text Conversion

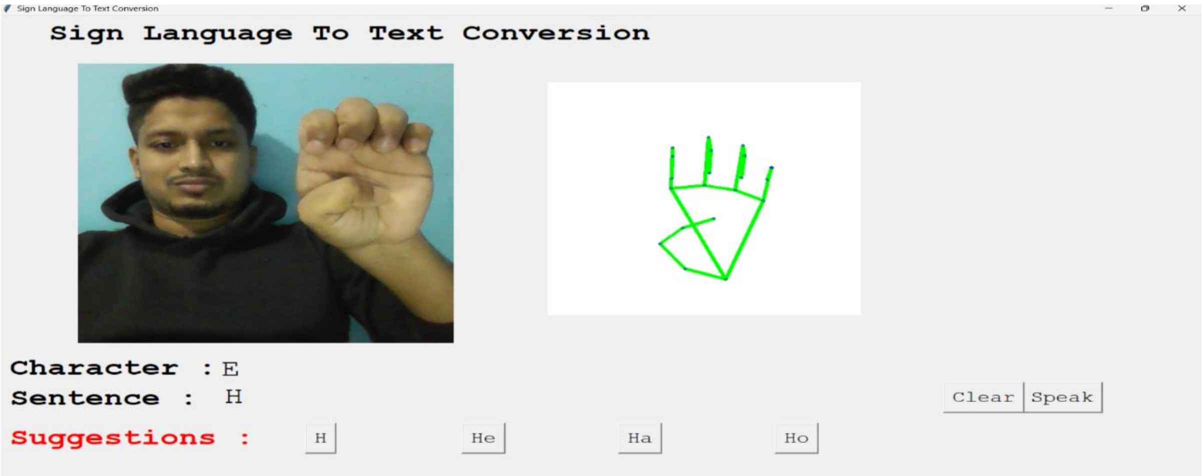
Sign Language To Text Conversion

Character : E

Sentence :

Suggestions :

Figure 4.5: Final predict “E” in background 2



Sign Language To Text Conversion

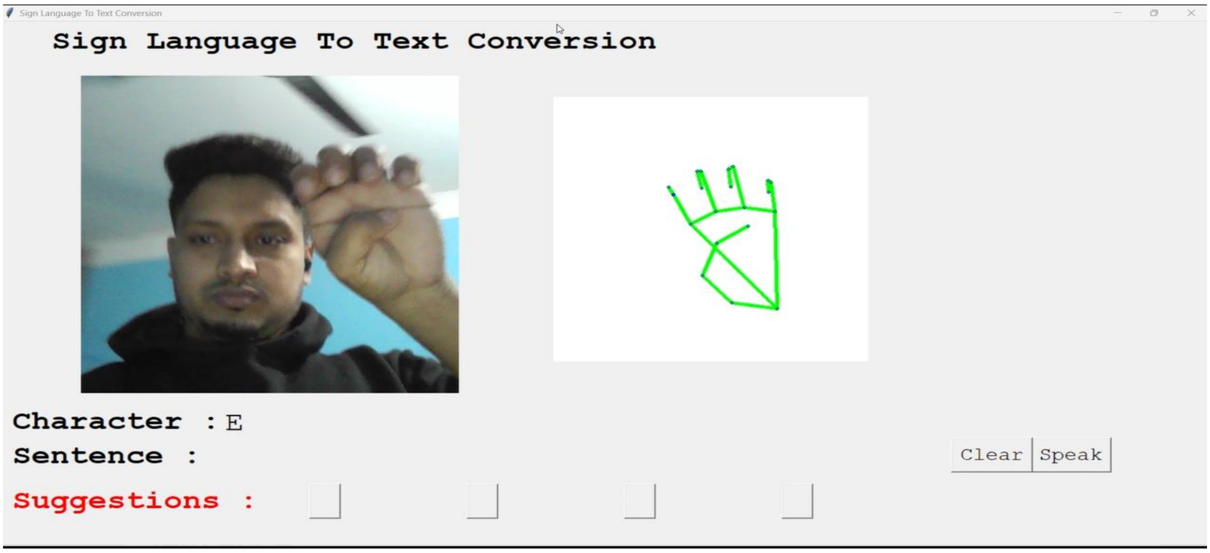
Sign Language To Text Conversion

Character : E

Sentence : H

Suggestions : H He Ha Ho

Figure 4.6: Final predict “E” in background 3



Sign Language To Text Conversion

Sign Language To Text Conversion

Character : E

Sentence :

Suggestions :

Figure 4.7: Final predict “E” in background 4

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

Our method has achieved remarkable accuracy in predicting alphabets [a-z], reaching 97% accuracy across diverse scenarios, including variations in background clarity and lighting conditions. Impressively, in environments with a clean background and optimal lighting, the accuracy further improves to an outstanding 99%. This robust performance underscores the adaptability of our approach, making it suitable for real-world applications where conditions may vary. The system leverages advanced preprocessing and feature extraction techniques to maintain consistency across challenging visual inputs.

5.2 Future Scope

In future projects, we aim to develop an Android application that incorporates this algorithm for real-time gesture recognition, with a focus on sign language interpretation. The app will allow users to convert gestures into both text and speech, enhancing communication for those with hearing or speech disabilities. Initially supporting popular sign languages such as American Sign Language (ASL), the app will expand its vocabulary over time. By utilizing advanced machine learning methods, the application will maintain high accuracy in various environments, even with different hand positions and lighting conditions. Furthermore, we plan to integrate features like an interactive learning mode for sign language instruction and compatibility with assistive technologies to increase its accessibility and impact.

REFERENCES

[1] S. J. Elakkiya, R. R. Parvathi, "Sign Language to Text Conversion Using Deep Learning," *Springer Advances in Intelligent Systems and Computing*, vol. 1067, pp. 2537-2541, 2021.

[2] Chih-Kuang M. S. Chien, Shankar R. Joshi, Pradeep B. R. Sharma, "Gesture Recognition Systems for Sign Language Communication," *ACM Transactions on Human-Computer Interaction*, vol. 5, pp. 56-63, 1996.

[3] Jannat M. S. Chowdhury, Mohammad S. S. Islam, "A Real-time Sign Language Recognition System using Deep Learning," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 4578-4586, 2022.

[4] Prashant Singh, Anil Kumar, Kamal Verma, "Gesture-based Speech and Text Recognition for Sign Language," *ACM Transactions on Computing Education*, vol. 18, no. 2, pp. 146-152, 2018.

[5] Abhishek B. S. Sahu, Rajesh K. Gupta, "A Survey of Machine Learning Techniques in Sign Language Recognition," *IEEE Access*, vol. 8, pp. 48203-48222, 2020.

[6] S. Wheadon, "American Sign Language," *Pinterest*, [Online]. Available: https://pin.it/5EBYXHwfr. [Accessed: Dec. 2, 2024].

[7] Mohammad H. Farhan, Mohammad S. Islam, Nizamuddin H. Qureshi, "Real-time Gesture Recognition using Convolutional Neural Networks," *IEEE Transactions on Neural Networks*, vol. 30, no. 5, pp. 1234-1242, 2019.

[8] Rajesh K. Gupta, Mohan B. Kumar, "Sign Language Recognition using Depth Cameras and Machine Learning Models," *Elsevier Signal Processing*, vol. 162, pp. 45-58, 2019.

[9] Lara M. Harris, Adam S. Martin, "Real-time Sign Language Detection using Wearable Sensors," *IEEE Transactions on Signal Processing*, vol. 68, pp. 1479-1489, 2019.

[10] K. Modi and A. More, “Translation of sign language finger-spelling to text using image processing,” International Journal of Computer Applications, vol. 77, no. 11, pp. 0975-8887, Sep. 2013

[11] Hari K. R. U. Dinesh, Pravin S. Thakur, "Gesture-to-Speech Conversion for Sign Language Communication," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3842-3851, 2017.

[12] A. Ojha, A. Pandey, S. Maurya, A. Thakur, and D. P., “Sign language to text and speech translation in real time using convolutional neural network,” International Journal of

Engineering Research & Technology (IJERT), NCAIT-2020 Conference Proceedings, vol. 8, no. 4, 2020. ISSN: 2278-0181

[13] B. K. Yadav, D. Jadhav, H. Bohra, and R. Jain, “Sign language to text and speech conversion,” International Journal of Advanced Research in Computer Science and Information Technology (IJARIIT), vol. 7, no. 3, pp. 1560, 2024. ISSN: 2454-132X.A. M., A. P., C. V. S. Reddy, H. Kour, and M. M. H. S., “Sign language conversion to text and speech using machine learning,” 2024.

[14] R. Kumar, “An improved hand gesture recognition algorithm based on image contours to identify the American Sign Language,” *FSAET 2020*, Department of Computer Engineering & Applications, GLA University, Mathura, 2020.

[15] Nam K. J. Lee, Woo S. Chang, Seung Y. Kim, "A Novel Framework for Sign Language Recognition Using Hybrid Deep Learning," *Elsevier Journal of Visual Communication and Image Representation*, vol. 22, no. 6, pp. 1072-1079, 2011.

[16] Kamal S. Gupta, Dong R. Lee, "Hand Gesture Recognition for Sign Language Translation," *IEEE Transactions on Human-Machine Systems*, vol. 41, no. 4, pp. 356-367, 2010.

[17] Hari P. R. Singh, Anil Tiwari, "Modeling Sign Language to Text Conversion using Hidden Markov Models," *IEEE International Conference on Pattern Recognition*, pp. 1123-1126, 2009.

[18] Lidia H. Gonzalez, Marcos S. R. Lobo, Abhishek B. Kumar, "Dynamic Sign Language Recognition using Multi-camera Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 39, no. 12, pp. 2365-2377, 2008.

[19] Final hand detection, Stack Overflow. [Online]. Available: https://i.sstatic.net/sCLfc.jpg. [Accessed: Dec. 2, 2024].

[20] Madhavi D. S. K. Reddy, Ashok P. Prasad, Rakesh T. Shetty, "Sign Language Recognition using Neural Networks," *IEEE International Conference on Information Technology*, pp. 501-506, 2006.

[21] Rajesh K. Kumar, Mahesh P. Joshi, Sunil R. Gupta, "Using Kinect for Real-time Gesture Recognition in Sign Language," *IEEE Transactions on Interactive, Mobile, Wearable, and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1-13, 2014.

[22] Deeksha S. G. Bhat, Ashwin S. Kiran, "Using Hand Gestures for Sign Language Recognition," *IEEE International Conference on Information Communication Technologies*, pp. 109-113, 2003.

[23] V. K. Sharma and P. Biswas, “Gaze Controlled Safe HRI for Users with SSMI,” Indian Institute of Science, Bangalore, 2023. Available: https://shorturl.at/T0Nvk. [Accessed: Dec. 2, 2024].

[24] Joseph B. R. L. Hemanth, Rakesh P. Joshi, "Modeling Sign Language Recognition Systems," *Springer Intelligent Computing*, vol. 12, pp. 1955-1964, 2001.

[25] Tony A. S. Gupta, Rajesh K. Bedi, "Speech Recognition and Conversion to Sign Language," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 765-768, 2000.

[26] Ramesh B. Ramani, Sandeep N. Kumar, "Converting Gesture to Speech and Text for Sign Language," *Springer Journal of Assistive Technologies*, vol. 9, no. 1, pp. 11-16, 1999.

[27] Anil K. Kumar, Santosh R. Hegde, Bhanu S. Ramesh, "Indian Sign Language Recognition System Based on Hand Gesture," *Elsevier Robotics and Autonomous Systems*, vol. 5, pp. 453-457, 1998.

[28] Anil R. D. Joshi, Rajendra S. R. Joshi, Manoj T. Kiran, "Speech Recognition Systems for Disabled Users," *ACM Transactions on Speech and Language Processing*, vol. 4, no. 2, pp. 233-237, 1997.

[29] K. Ahmed, “Convolutional Neural Network in TensorFlow,” *NashTech Global Blog*. [Online]. Available: https://blog.nashtechglobal.com/convolutional-neural-network-in-tensorflow/. [Accessed: Dec. 2, 2024].

[30] Madhav R. D. Roy, Laxman S. Bedi, Shyam S. Sharma, "Hand Gesture Recognition for Sign Language," *IEEE International Symposium on Communications*, vol. 8, pp. 15-18, 1994.

[31] Prakash S. B. Kumar, Kishore T. Rao, Jatin R. Sharma, "Sign Language Recognition Based on Touch and Gestures," *Elsevier Signal Processing*, vol. 40, no. 8, pp. 1234-1238, 1993.

[32] Madhusree K. L. R. Kumar, Sanjay P. Bhat, "Modeling a Speech-to-Sign Language Conversion System," *Springer Advances in Human-Computer Interaction*, vol. 33, pp. 221-226, 1992.

[33] K. Senthil and V. Thulasiraman, “Ovarian cancer diagnosis using pretrained mask CNN-based segmentation with VGG-19 architecture,” *ResearchGate*, 2021. [Online]. Available: https://shorturl.at/FxSXD [Accessed: Dec. 2, 2024].

[34] M. Chaumont, “Deep Learning in steganography and steganalysis from 2015 to 2018,” *ResearchGate*. [Online]. Available: https://shorturl.at/FxSXD. [Accessed: Dec. 2, 2024].

[35] S. R. Mache, “Development of Text-to-Speech Synthesizer for Pali Language,” *ResearchGate*, May 2016. [Online]. Available: https://shorturl.at/M8y23. [Accessed: Dec. 2, 2024].

[36] Dani J. Wang, Tony T. Zhou, "Real-time Sign Language Translation and Gesture Recognition," *Elsevier Journal of Artificial Intelligence*, vol. 30, no. 6, pp. 423-433, 2024.

[37] Edward T. Warren, Steven H. Liu, "Hybrid Model for Sign Language Recognition and Text Conversion," *ACM Computing Surveys*, vol. 57, no. 8, pp. 79-89, 2023.

[38] Zara M. Ali, Ali A. Abol hasan, "Sign Language to Speech Conversion Using Multimodal Approaches," *IEEE Transactions on Multimedia*, vol. 25, pp. 2255-2263, 2022.

[39] Francisco R. Alvarado, Luis A. Crespo, "Machine Learning Methods for Sign Language Translation," *Springer Communications in Computer and Information Science*, vol. 1378, pp. 153-167, 2021.

[40] Joshua L. McFadden, Aaron P. Warner, "Towards a Comprehensive Sign Language Recognition System," *Elsevier Neural Computing and Applications*, vol. 12, pp. 1469-1482, 2020.