Real-Time Sign Language Recognition and Conversion into Text Using Deep Learning

By

Mrinmoy Modak
ID: 1702060

Mithun Bhowmick
ID: 1702050

A thesis submitted in partial fulfillment of the requirements for the degree of

BACHELOR of SCIENCE in Electrical & Electronic Engineering

CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY

MAY 2023

Declaration

We hereby declare that the work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of our knowledge and belief, the Thesis contains no material previously published or written by another person except where due reference is cited. Furthermore, the Thesis complies with PLAGIARISM and ACADEMIC INTEGRITY regulation of CUET.

---------- ----------

Mrinmoy Modak **Mithun Bhowmick**

Student ID: 1702060 Student ID: 1702050

Department of Electrical & Electronic Engineering

Chittagong University of Engineering & Technology (CUET)

Dedication

To our loving parents and supportive supervisor Dr. Mohammad Rubaiyat Tanvir Hossain Sir

Approval/Declaration by the Supervisor(s)

This is to certify that Mrinmoy Modak and Mithun Bhowmick have carried out this research work under my supervision, and that they have fulfilled the relevant Academic Ordinance of the Chittagong University of Engineering and Technology so that they are qualified to submit the following Thesis in the application for the degree of BACHELOR of SCIENCE in the department of Electrical & Electronic Engineering. Furthermore, the Thesis complies with the PLAGIARISM and ACADEMIC INTEGRITY regulation of CUET.

Dr. Mohammad Rubaiyat Tanvir Hossain
Professor
Department of Electrical & Electronic Engineering
Chittagong University of Engineering & Technology

Acknowledgment

We want to extend our sincere gratitude and appreciation to everyone who helped finish this thesis. We want to start by expressing our gratitude to our supervisor, Dr. Mohammad Rubaiyat Tanvir Hossain for his encouragement, and priceless advice throughout the entire research process. His guidance and commitment have been essential in forming this work. We want to express our sincere gratitude to everyone who voluntarily shared their experiences and insights during the data collection process. Their assistance was crucial in achieving the goals of this study. We are appreciative of our family and friends' unwavering support, inspiration, and tolerance throughout the challenging process of finishing this thesis.

Abstract

For deaf and hard-of-hearing people, sign language is a fundamental form of communication. When trying to type using their normal way of communication, which is through hand gestures, they run into some challenges. Building a deep learning model that successfully translates hand gestures into text that contains an expression is our thesis's key focus. The realization that many of the available typing aid solutions do not adequately support this issue inspired the concept. The suggested system is trained using a large dataset that includes a wide variety of sign language gestures. We have detected the hand gesture using a mediapipe library. An LSTM-RNN-based algorithm then processed it and turned it into text. The model successfully translates hand gestures into text with a high degree of precision, making it a helpful tool for the deaf. Even though the results of the proposed system look good, more study is needed to find ways to make it better and deal with any possible problems. We also designed an app with two user-friendly interfaces to provide a complete solution to typing for blind and physically disabled people.

Table of Contents

List of Figures

Fig. No. **Figure Caption** **Page No**

List of Tables

Table No. **Table Caption** **Page No.**

Nomenclature

Acronyms and Abbreviations	Meaning
ASL	American Sign Language
Auslan	Australian Sign Language
BSL	British Sign Language
CSL	Chinese Sign Language
ISL	Irish Sign Language
JSL	Japanese Sign Language
SSL	Spanish Sign Language
CNN	Convolutional neural network
RNN	Recurrent neural network
LSTM	Long short-term memory
SLR	Sign Language Recognition
VR	Virtual Reality
ANN	Artificial Neural Network
RCNN	Regions with Convolutional Neural Networks
FFNN	Feed Forward Neural Networks
CGP	Cartesian Genetic Programming
SVM	Support Vector Machine
HMMs	Hidden Markov Models
SOM	Self-Organizing Map
GSM	Global System for Mobile

Acronyms and Abbreviations	Meaning
FOV	Field Of View
HOG	Histogram of Oriented Gradients
GUI	Graphical User Interface
JSON	JavaScript Object Notation
NLTK	Natural Language Toolkit
NLP	Natural Language Processing
GIF	Graphics Interchange Format
OS	Operating System
OpenCV	Open-Source Computer Vision Library
CUDA	Compute Unified Device Architecture
ARM	Advanced RISC Machine
CPU	Central Processing Unit
HPC	High-Performance Computing

Chapter 1: INTRODUCTION

1.1 BACKGROUND

Sign language is a method of communication that relies on the use of hand and arm gestures in addition to face emotions and body language. Deaf and hard-of-hearing people use sign language when spoken speech is either not feasible or inappropriate. There are many different sign languages in use, and they vary from one country to the next. There are presently more than 300 distinct sign languages used by people all over the globe. Several are solely used in a certain area, while others are used by millions of people worldwide. Each sign language has its own unique grammar and meaning. Once again, your body language matters. Hand gestures, face expressions, and body movements are only some of the expressive tools used in sign languages [1].

Popular Forms of Sign Language Used Around the World:

- American Sign Language (ASL)
- Auslan (Australian Sign Language)
- British Sign Language (BSL)
- Chinese Sign Language (CSL)
- Irish Sign Language (ISL)
- Japanese Sign Language (JSL)
- Spanish Sign Language (SSL)

Sign language ASL is the most extensively used (American Sign Language). ASL has the most speakers and is used in about 20 different nations. The alphabet is operated with one hand. There are numerous dialects of BSL (British Sign Language), which uses a two-handed alphabet and varies from region to region. On the other hand, Auslan (Australian Sign Language) has two main dialects,

each of which uses a distinct sign for a concept—such as an animal, color, or day of the week—while maintaining a consistent grammatical structure. Two hands are used to write in Auslan. Visual representations of written Chinese characters are used in CSL (Chinese Sign Language), which uses a one-handed alphabet. Of all the CSL dialects, Shanghai is the most prevalent. Japanese Sign Language (JSL) differs significantly from other sign languages in that it uses mouthing to distinguish between signals and alphabetic letters. With an estimated 100,000 signers, SSL (Spanish Sign Language), which is entirely distinct from ASL, is primarily utilized in Spain. Except for Catalonia, which utilizes Catalan Sign Language, and Valencia, which uses Valencian Sign Language, SSL is used throughout all of Spain[2].

People use sign language movements to communicate non-verbally, but the ordinary listener has a hard time understanding what they're saying. Sign language is a useful tool for communication, yet it is often overlooked. Hiring a translator full-time for the deaf and hard of hearing would be very costly. A person with a disability always has a hard time communicating verbally and writing down their thoughts[3].

Detecting sign language has been a topic of study in computer vision and machine learning for many years. In the 1960s and 1970s, the initial attempts to recognize sign language were made, however, the technology at the time placed limitations on these systems. The use of computer vision algorithms for sign language recognition was first investigated by researchers in the 1980s and 1990s. In these methods, hand gestures and movements were often identified by watching videos of sign language being used and utilizing image processing techniques[4].

Researchers began looking into the use of neural networks for sign language detection with the emergence of deep learning in the 2000s. Since it

enables the development of models that can learn to detect intricate hand motions and movements, this method has proven to be quite successful[5].

Sign language identification is a rapidly growing area of study because to its widespread potential uses in areas such as assistive technology, education, and communication. New and better algorithms for sign language detection are constantly being developed by researchers in order to provide more accurate, reliable, and user-friendly systems.

Today, techniques like Feed Forward Neural Networks (FFNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and others are employed for image processing and sign language recognition. Obtaining hand gestures, segmenting them, filtering them, representing them, and classifying them are the stages of a full vision-based system[6].

1.2 CONTEXT

The focus of this study is on developing an automated system that can interpret and display sign language in text format. persons who are deaf or hard of hearing sometimes find themselves at a communication disadvantage when interacting with hearing persons who do not know sign language. By developing a system to recognise sign language and convert it into text, we can eliminate this barrier to communication and open the door to more effective dialogue between the hearing and the deaf communities.

1.2.1 Statement of the Problem

The primary issue this work aims to address is the lack of precise and trustworthy systems for translating sign language to text. Despite some advancements in this area, the current methods have drawbacks like poor accuracy, a small vocabulary, and a challenge in identifying signs in various environments or illumination. As a result, there are now communication obstacles between deaf and hearing people, which can have a severe impact on

their social interactions, access to school and employment possibilities, and other areas. To efficiently translate sign language into text and show it in a text file, it is necessary to build more precise and trustworthy sign language recognition systems.

1.3 PURPOSE

This work aims to create a system for real-time, accurate recognition of sign language and its conversion to text. The useful conclusion of this research is the creation of a tool that can facilitate effective communication between hearing and deaf people.

Specific Aims and Objectives:

1. To create a real-time sign language recognition system that uses computer vision and convert the recognized sign language into text format.
2. Create a user-friendly app to improve typing accessibility for visually impaired and physically challenged people.

1.4 SIGNIFICANCE, SCOPE, AND DEFINITIONS

1.4.1 Importance

Several factors need the creation of real-time sign language recognition tools. To begin, it addresses the critical problem of the challenges faced by the deaf and hard of hearing in communicating with the hearing world. The current methods of communicating, such as lipreading or taking notes, can be time-consuming and inaccurate, leading to frustration and social isolation. Second, the use of computer vision technology in the real-time identification of sign language gestures has the potential to enhance the quality of life and effectiveness of communication for the deaf and hard of hearing. A major step forward in signal processing and computer vision, the development of such a system is inevitable.

1.4.2 Methodology

This study's methodology combines a literature review with data gathering, system design, and execution. The theoretical underpinnings for the design of the sign language recognition system will come from the literature review. The gathering of data entails creating a dataset of sign language motions and their matching text translations for the system's training. The system's design and implementation will make use of computer vision technologies to precisely identify sign language gestures and translate them into text in real time. Testing the system's precision, speed, and robustness will be part of assessing its performance.

1.4.3 Gap In Literature

Even though the considerable study has been done in the area of sign language recognition, there is still a large gap in the literature when it comes to computer vision technology-based real-time sign language recognition. As we've seen, a number of initiatives have been made to detect words letter by letter. However, there aren't many parts that can transform a gesture into a word or phrase. If we can enter a line or word with only one motion, we can reduce the amount of time we spend typing. To close this gap, a technology that enables them to enter their most often used word or line with a single gesture should be made available. Again, there is very little software available that can identify a gesture and store it as text right away. Again, the majority of the detection is performed by image classifiers built on the CNN method. Due to variations in lighting, perspectives, sizes, orientations, backgrounds, and occlusions, image classifiers struggle to generalise their results across samples belonging to the same category. Overfitting is another frequent problem, when the classifier gets

overly focused on the training set and is unable to generalise to new, untried images. The solution to this problem might lie in the RNN-LSTM technique.

Based on our study, we've found that there isn't a typing option that meets the needs of people who are deaf, blind, or have some other kind of physical disability. Even though there are now platforms for each of these groups, they often don't meet the needs of all three groups at the same time or make the users feel comfortable. Blind people have to use sounds or touch to help them type because they can't use visible hints. For blind people, Braille computers and special apps have been made. However, they may be hard to use and hard to find. People who can't see at all or who can only see a little bit can't use these ways. When making a typing system for the blind, it's important to put clarity, accessibility, and freedom at the top of the list. The interface could help people who can't see by letting them know when they press a key or type words. Speech recognition software can help people who are physically disabled and don't have much or any control over their muscles write much better. Voice recognition software can sometimes be used to type by voice, but people with disabilities might find it more helpful to have a single interface that can turn speech into text, store this text in a file, and provide listening services.

1.4.4 Scope and Delimitations

The purpose of this project is to develop a system that can accurately recognize sign language motions, translate them into text in real-time, and display the results in a text file. Technology based on computer vision will be utilized. A limited number of sign language gestures and text translations are used to train the system, and the research is limited to a certain region and camera setup.

1.4.5 Definitions

Sign language: Sign language is a way for deaf and hard-of-hearing people to interact through body movements, face expressions, and hand gestures.

Computer Vision (CV): Computer Vision is an area of study that focuses on making it possible for computers to read and understand pictures and videos from the real world.

Real-time: Real-time means that the sign language recognition system can recognise sign language movements and turn them into text quickly enough to be useful for conversation.

Dataset: A group of sign language movements and their written versions that are used to train a system that can understand sign language.

Sign Language Recognition (SLR): The technology and procedures used to recognise and understand sign language motions and movements and transform them into written or spoken language are referred to as sign language recognition.

Virtual Reality: The term "virtual reality" (VR) describes a simulated environment or experience that might be substantially distinct from or comparable to the actual world. Advanced computer technology is used to make it, and it is often displayed to the viewer using a head-mounted display or other immersive tools.

Neural Network: A computer model called a neural network—also referred to as an artificial neural network (ANN) or a neural network model—is modelled after the way in which the biological neural networks in the human brain are organised and operate. It is a machine learning algorithm that is able to discover patterns, recognise them, and make predictions in addition to resolving challenging issues.

Convolution Neural Network (CNN): It is a particular kind of deep learning algorithm made with processing and analysis of visual input, such as photos and videos, in mind. CNNs are commonly utilised for applications including image

classification, object recognition, and image segmentation. CNNs have revolutionised the area of computer vision.

Recurrent Neural Network (RNN): It is a kind of artificial neural network made to interpret temporal and sequential input. RNNs include feedback connections that enable information to survive and flow through the network in a recurrent way as opposed to feedforward neural networks, which analyse input data in a single pass.

Region-based Convolutional Neural Network (RCNN): Region-based Convolutional Neural Network is known as RCNN. It is a specific kind of neural network design that is frequently employed in computer vision applications such as object identification and recognition. Convolutional neural networks (CNNs) for feature extraction and region proposal techniques for precise object localization are combined in RCNN.

Feedforward Neural Network (FFNN): The multilayer perceptron (MLP), often known as a feedforward neural network (FFNN), is a kind of neural network. It is a sort of artificial neural network where information only flows in one way, from the input layer via one or more hidden layers to the output layer, without producing cycles or loops.

OpenCV: OpenCV (Open-Source Computer Vision Library) is a computer vision and machine learning library that is commonly used for creating computer vision applications**.**

Python: High-level programming language Python is renowned for its clarity, readability, and adaptability. Python is simpler to write and understand than other programming languages because it emphasizes code readability and has simple, expressive grammar. Programming styles supported by Python include functional, object-oriented, and procedural programming.

Cartesian Genetic Programming (CGP): A computational method known as Cartesian Genetic Programming (CGP) combines genetic programming and graph theory ideas. For automated programme synthesis and optimisation, it uses an evolutionary method.

Support Vector Machine (SVM): A supervised machine learning technique used for regression and classification problems, goes by the abbreviation SVM. Analysing data and seeing patterns both often utilize it.

Hidden Markov Models (HMMs) : HMMs, or Hidden Markov Models, are statistical models that are frequently employed in tasks involving sequential data analysis, pattern identification, speech recognition, and natural language processing.

Self-Organizing Map (SOM): An unsupervised artificial neural network approach called a Self-Organizing Map (SOM), sometimes referred to as a Kohonen network, maps high-dimensional input data onto a low-dimensional grid. It is frequently employed to visualise and cluster data while maintaining the topological links between samples.

TensorFlow: Google created TensorFlow, an open-source machine learning framework. It offers a complete collection of resources and libraries for creating and implementing machine learning models. Due to its flexibility, scalability, and efficiency, TensorFlow is a well-liked option for creating different kinds of deep learning models.

Global System for Mobile (GSM): It is a specification for digital cellular networks, which are commonly employed for mobile communication around the world. In order to provide dependable voice communication, GSM technology was created. Later, it was enhanced to accommodate data services like SMS (Short Message Service) and mobile internet access.

Field of View (FOV): It describes the size of the world that can be observed or the region that a camera, sensor, or human eye can see. The FOV in computer vision and imaging specifies the range and angle of the scene that may be viewed or recorded.

Microsoft Kinect V2: The second iteration of the Microsoft-developed Kinect sensor is referred to as Kinect V2. It is a motion-sensing input device that integrates depth detection, RGB camera imaging, and microphone array technologies.

HOG (Histogram of Oriented Gradients): The HOG (Histogram of Oriented Gradients) feature descriptor technique is often employed in computer vision and image processing tasks, notably in object recognition. By examining the distribution of gradient orientations, it seeks to extract local shape and texture information of a picture.

GUI (Graphical User Interface): Graphical User Interface is what it stands for. It describes a visual user interface that enables users to communicate with a computer or software programme using graphical components like buttons, menus, windows, and icons.

MATLAB: An environment and high-level programming language for numerical computation, data analysis, and visualisation are called MATLAB. MATLAB, which was created by MathWorks, offers a complete range of tools and functions that enable users to carry out a variety of activities in the scientific and engineering areas.

Capsule Network: A particular design for neural networks is called a capsule network, or CapsNet. They are intended to overcome a few of the drawbacks of conventional convolutional neural networks (CNNs) in applications like object and image detection.

JSON (JavaScript Object Notation): JSON (JavaScript Object Notation) is a simple standard for exchanging data between servers and online applications as well as across various platforms. It offers a format for describing data items that are organized and simple to read.

NLTK (Natural Language Toolkit): A popular Python package for natural language processing (NLP) activities is NLTK (Natural Language Toolkit). Developers and researchers may quickly carry out a variety of NLP operations because to the tools and resources it offers for dealing with human language data.

TELite: The Communications-Electronics Research, Development, and Engineering Centre (CERDEC) of the United States Army created the software development platform known as TELite. The acronym "TELite" refers for "Tactical Edge (TE) Lite," and it was created to offer a compact and adaptable tactical communication system for use in military operations.

Natural Language Processing (NLP): The interplay of computers and human language is the main topic of this branch of artificial intelligence and computational linguistics.

Google API: A group of application programming interfaces (APIs) established by Google that enable developers to incorporate Google features and services into their own applications are collectively referred to as Google API. Google Maps, Google Drive, Google Cloud Platform, Google Translate, Google Calendar, and many more services are accessible using these APIs.

Graphics Interchange Format (GIF): It is a typical file format for presenting animated graphics or short video snippets online. GIF files may include both static and moving pictures, and since they employ a lossless compression method, they preserve image quality with little data loss.

1.5 MOTIVATION

According to the World Health Organization, about 285 million people in the world are blind, 300 million are deaf and 1 million are dumb[7].About, Four thousand deaf children are born each year. Over 1.5 billion people globally live with hearing loss; this number could rise to over 2.5 million by 2050. According to the survey, in 2022 about 11.5 million Americans have varying degrees of hearing differences which is 3.5% of the population. The 2011 American Community Survey estimates 3.6% of the U.S. population is deaf or hard of hearing. About 9.6% of the population of Bangladesh (about 13.7 million people) is deaf or hard of hearing (having a loss of 40 dB or more) [8].The goal of our work is to develop a sign language typing system using real time computer vision. In recent years, research about sign language has been used in many fields specially robotics, virtual reality, computer vision,neural networks, etc [6].

This project aims to improve the quality of life and access to information for people with disabilities via increased empowerment and easier communication. You want to eliminate barriers to communication for the deaf, the blind, and those with other physical impairments by creating a machine-learning model and an app tailored to their requirements. Deaf people may benefit greatly from being able to communicate via the use of a system that interprets hand gestures into text. If they have trouble communicating in more conventional ways, you may help by building a system that can reliably understand their hand gestures and transform them into text using machine learning techniques like the RNN LSTM algorithm. This not only improves their communication skills but also gives children greater confidence to speak out in a variety of social situations. In addition, the difficulties blind people encounter while trying to type and communicate have been addressed by creating a typing interface tailored to them, which uses just three buttons. You provide customers with a quick and easy way to enter frequently used phrases or lines by providing a streamlined and understandable interface. This allows the visually impaired to

communicate with more freedom and independence, which in turn increases their productivity and participation in a wide range of activities. Those who are unable to use their hands to interact with the software may still profit from it thanks to the addition of voice recognition. The ability to translate speech to text removes the barrier of typing and makes voice communication a viable option. People with mobility impairments may now express themselves, engage in social interactions, and complete a variety of activities with the help of this tool. This project's motivation stems from the conviction that all people should be afforded the same freedom of speech and communication regardless of their physical or mental capacities. We want to provide people with disabilities the tools they need to interact successfully and freely, so you're using machine learning to create interfaces that are easy to use. With this initiative, you want to improve their standard of living, spread acceptance, and create a more accessible and accepting society.

Chapter 2: LITERATURE REVIEW

2.1 HISTORICAL OVERVIEW

Deaf and dumb people mostly utilize sign language to communicate with others and in their society by using hand and body gestures. Teaching sign language to the deaf and dumb would be difficult because there are so many different sign languages. SLR (Sign Language Recognition) focuses on identifying these hand signals and translating them into the appropriate text or voice[9].

There is a ton of advice, tips, and recommendations on how tremendous aid folks with complex and stupid communication issues. Automatic sign language recognition and the accompanying translation procedure have benefited from research contributions and development[10].

To identify hand and body movements, numerous systems have been proposed. The following methods are used to recognize hand gestures: Polygon Approximation, hand gesture recognition based on shape parameters, computer-aided interpreter using voice-to-gesture and Gesture-to-voice conversions, a straightforward shape-based approach, an optimized wavelet hand pose, Cartesian Genetic Programming, Gradient slope magnitude edge images, Support Vector Machine, hand gesture recognition using Self-Organizing Map for human-computer interaction, and a neural network-based method. However, many systems for recognizing hand gestures require gloves or other data-gathering tools to capture the hand object[11].

The image or video is captured using image processing techniques, and it is afterward recognized by a sign language recognition system. Sensitive devices

record images. Suitable definite or statistical mathematical models are used to adjust the brightness values of the pixels and associated geometrical imperfections. By altering the pixel brightness levels, we can enhance the visual impact of the image. A variety of approaches are applied to enhance an image's visual appeal or transform it into a format that is better suitable for either human or computer interpretation[12],[13].

Based on Hidden Markov Models (HMMs), the authors provide a system that can recognize hand gestures from continuous hand motion for English digits from 0 to 9 in real-time (HMMs). The two types of gestures are link gestures and key gestures. The key gestures are distinguished from additional hand motion trajectories, or "spotting," by the link gestures. This kind of gesture spotting uses a heuristic-based method to pinpoint the beginning and ending points of the important gestures before passing the gesture's path to HMMs for classification. This model addresses how gestures change throughout time. By following the skin-color blobs that correspond to the hand into a body-facial space that is focused on the user's face, gestures can be recovered from a series of video shots. The objective is to distinguish between the deictic and symbolic kinds of gestures. A quick look-up indexing table is used to filter the image. Blobs of skin color pixels are formed after filtering[14].

Another recognition technique that makes use of computer vision based on pattern recognition and convolutional neural networks is finger spelling (a Deep Learning algorithm). There is no longer a requirement for an interpreter in this procedure[15].

Another gesture recognition system based on 3D data is described. The technology makes use of an innovative 3D sensor to provide a dense-range image of the scene. Without relying on color information, it can recognize complicated hand positions like those seen in sign language alphabets. The entire processing chain is covered by some unique 3D image analysis techniques. The suggested

system is tested in an application scenario including the identification of postures used in sign language[16].

Another approach is suggested in which Connected Component Labeling is used to segment the hand from the body. Based on the related pixels, the scene is labeled. The same name is given to pixels that fall within the same depth range. Since the hands are typically the nearest item to the camera, the closest set of pixels is chosen as prospective candidates for the hand [17].

A technique based on advanced convolutional neural networks and employing TensorFlow was proposed in [18] to identify plants. This method is considered to be a cutting-edge method for deep learning from the perspective of computers. By employing a complete image of a plant or certain parts of it, CNN tackles the most difficult task of plant identification.

Real-time speech synthesis from sign language is accomplished using an Android phone with a GSM (Global System for Mobile) module. The wearable gloves are equipped with flex sensors, and Bluetooth is used to transmit the corresponding messages to the android application. The layered feed is used, and the prediction is done using an ANN (Artificial Neural Network) method. Users of sign language may easily and hassle-free converse thanks to the Android app[19].

With Microsoft's Kinect V2, a sign language interpreter is created. This system utilizes two different communication channels. It transforms both audio and sign language into hand gestures. This model performs the roles of an interpreter and translator. Using hand gestures in front of the Kinect's field of view, one can translate sign language into voice (FOV). Sensors are used to capture the hand motion, which is then compared to the trained sign language in the database. An audio file is translated to text after a match is detected and mapped to the term it corresponds with. For voice-to-sign conversion to go to a place, one person must speak the Kinect FOV sensor in native English. Using API, sign language can be translated from audio[20].

American Sign Language (ASL) real-time translation to text and audio was proposed using an android application and a Support Vector Machine (SVM). ASL's 26 alphabets and a specially created symbol for "space" comprise the dataset, which is incredibly small. The small data set is handled using a Histogram of Gradient (HOG) by Support Vector Machine, which is important in determining an object's shape and edge direction. The limitation of this is a smaller dataset and accuracy. The alphabet, backspace, clear, and audio additions were the app's key features[21].

A webcam may be used to recognize American sign language in real-time and translate it to text in English. The sign is captured using a GUI (Graphical User Interface), and the image is pre-processed by using edge detection, morphological filtering, and morphing. Image cross-correlation identifies and matches features when a person uses sign language. This machine could read and translate alphabets as well as a few English words[22].

In [23], the authors have proposed a system that used a minimum eigenvalue algorithm. This algorithm is used for training double-handed sign language. For image acquisition and processing, they used a Logitech web camera and MATLAB. Each pixel in a picture is evaluated by using the value of two eigenvectors calculated. After the score reaches a particular point, a pixel is marked as a corner. Following the extraction of the text output, it is examined and looked up in a lexicon of letters and sounds. Upon finding a match, it is converted to speech.

Another approach uses a hardware-based controller gesture recognition system that utilizes a Flex sensor to capture hand gestures from gloved hands. The tactile sensor and sensor glove design serve to enhance accuracy by decreasing ambiguity in motions. The speech synthesizer receives the recognized text as input from the microcontroller as its output.

Each specific gesture is processed by the Arduino microcontroller. For each letter, the system is trained for a distinct voltage value. This approach, however, is not very cost-effective and needs high maintenance[24].

Capsule Network was used to implement character recognition for American Sign Language. Hand gesture photos have undergone training. Each letter has a different number of training images: the letter 'a' has 1126, whereas the letter 'b' has 1010. Data are not distributed evenly. The convolution layer's learned features are merged by the capsule network. Calculate the loss function for each letter, then predict the next letter using the loss. The accuracy of the capsule network was 95%[25].

For Sri Lankan sign language translation, region-based convolution neural networks (RCNN) and convolution neural networks (CNN) was suggested. Each letter has three pictures collected. Faster RCNN is built using TensorFlow. For the discovered classes, labels are created. The image size is decreased to 800600 pixels, allowing for quicker training and storage. Labeling and XML obtaining are done using the Pascal VOC tool. It has been trained using Faster RCNN. The accuracy is 95%. The word is evaluated using NLTK (Natural Language Toolkit) in Natural Language Processing to translate it into speech by breaking up the sentence into words. When a match is made with a word in the database, the corresponding signs have been correctly organized and will produce GIF images[26].

Fleksy is a famous keyboard app that has tools to make it easier for people with disabilities to use, such as support for blind users[27]. It has a unique way to type with gestures. Users can make words by swiping their fingers across the screen. The app also helps blind people type better by giving them spoken feedback, audio cues, and adjustable keyboard setups. Fleksy Keyboard's benefits make it popular. It offers rapid, accurate typing. Gesture typing allows one-handed typing. Fleksy Keyboard also offers several themes and colors for customization. It's multilingual, too. There are certain restrictions. Fleksy

Keyboard's layout and gesture typing may take some getting used to. While the fundamental capabilities are free, in-app purchases offer premium features and themes. Device and platform compatibility may vary. Fleksy Keyboard is a good keyboard; however, users should examine their needs and preferences before using it[28].

BrailleBack is a virtual keyboard app for Braille-using visually challenged people[29]. Its benefits are numerous. First, BrailleBack seamlessly integrates Braille devices with Android devices, allowing visually challenged people to utilize their smartphones and tablets. It allows users to choose between contracted and uncontracted Braille. Visually challenged users may read and write text messages, emails, and other digital information independently using BrailleBack's accurate and real-time Braille translation. Additionally, it supports many languages, making it global. There are certain restrictions, though. BrailleBack requires suitable Braille devices, which might be expensive and need setup. BrailleBack's functionality may also vary per Android device and OS version, causing compatibility concerns. Furthermore, some programs and platforms may not support the BrailleBack keyboard, restricting its use.

Google Docs and Google Slides' speaker notes both provide voice typing, which is a useful accessibility tool. Only Chrome browsers support this feature, and some of its functionality (such as voice commands for editing, formatting, and punctuation) is language-specific. Google Docs Voice Typing lets users speak rather than type. Its benefits improve user experience and efficiency. Google Docs Voice Typing allows users to talk usually and have their words automatically typed. This removes manual typing and is especially helpful for folks with low typing skills or prefer a hands-free approach. Voice recognition accuracy is another benefit. Google's speech-to-text technology is accurate. This lets users dictate information without mistakes, saving time on proofreading and editing. Voice Typing supports several languages, making it accessible worldwide. Integrating Google Docs functionality is another benefit. Voice

Typing integrates smoothly with other collaborative capabilities in Google Docs, allowing users to dictate and modify texts in real-time. This tool improves cooperation and productivity for distant teams and those with mobility issues. There are certain restrictions. Background noise, accents, and speech patterns can influence Voice Typing accuracy. For best results, users should talk clearly and keep it quiet. Since Voice Typing relies on broad voice recognition, it may struggle with sophisticated formatting directives or industry terms[30], [31]. Dragon Anywhere, a Nuance Communications smartphone app, provides excellent speech recognition and dictation. For voice-based productivity and efficiency, the app has several benefits. Dragon Anywhere's precision is a major benefit. The app's accurate transcriptions come from Nuance's deep learning technology, which adjusts to the user's voice and speech patterns. This lets users dictate papers, emails, notes, and other text-based information with few mistakes, saving time on typing and reviewing. Another benefit is the app's mobility. Users may dictate and transcribe text on cellphones and tablets with Dragon Anywhere. It's handy for professionals who are often on the go or need to record their thoughts and ideas away from their laptops. Dragon Anywhere's customizing options are extensive. Voice commands and macros allow users to automate tedious activities or enter commonly used phrases and patterns. Customization streamlines dictation and boosts productivity. Dragon Anywhere also integrates with Dropbox and Evernote. Users may seamlessly sync their dictated papers across devices and view them from other platforms, improving collaboration and data availability. However, there are some constraints. First, Dragon Anywhere needs a membership, which may alienate customers who prefer free or cheaper options. The program also requires a reliable internet connection for voice recognition processing. Another drawback is the app's learning curve for recognizing certain speech patterns and terminology. It may take some time to teach Dragon Anywhere to recognize a user's speech and pronunciation[32], [33].

The Microsoft Office add-on Microsoft Dictate lets users dictate text by speaking into a microphone. It uses Microsoft's voice recognition technology to transcribe verbal input into text and provides a number of benefits to its users. An important benefit is the ease and quickness with which content may be produced, which allows for more organic and spontaneous writing. It integrates seamlessly into the established Office workflow and supports a wide variety of languages. However, the software's identification accuracy may vary from one user to the next and you'll need a working internet connection to use it[34].

2.2 SUMMARY AND IMPLICATIONS

There are many guidance and suggestions about techniques and methods for helping people with deep and dumb problems to help them in communication[35]. Researchers have contributed to and developed the field of automatic sign language recognition and the corresponding translation process[9]. Many schemes have been proposed to recognize hand and body gestures[36]. Image processing techniques are used to capture the image/video and then recognize it using a sign language recognition system[13]. As images can be affected by many facts, there are many algorithms for image processing[12]. Nowadays, Feed Forward Neural Networks (FFNN)[37], Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), etc are techniques used for image processing and sign language recognition. In recent years, research about sign language has been used in many fields specially robotics, virtual reality, computer vision, neural networks, etc[6].

The sign language dataset differs depending on the chosen sign language and what has to be identified because there is no single sign language. There are different types of translation processes: sensor-based translation, machine learning, deep learning-based translation, etc.

Convolution neural networks have been successfully applied for a long time in the fields of speech recognition and digital image processing. Both speech recognition and picture processing were previously handled by conventional machine learning algorithms until the convolutional neural network was invented. Conventional algorithms delivered excellent outcomes, but it was difficult to accomplish additional advances. So, CNN was created. Speech recognition and image processing using CNN are currently in a relatively advanced stage. The success of the theoretical study and commercial application has aided CNN's rapid development. The success of CNN's image processing and speech recognition has sparked a research craze in natural language processing. Although there have been some advancements, the present effect of using CNN to handle natural language is not very good. CNN overcomes most of the conventional issues, making it superior to other deep learning techniques in applications involving computer vision and natural language processing.

One of the major effects of sign language recognition technology is its ability to increase the accessibility of communication for the deaf and hard-of-hearing. Sign language recognition technology can eliminate communication barriers and advance greater inclusion in society by allowing sign language to be converted into text or voice in real-time.

The development of communication for those who are deaf or hard of hearing is seen in the history of sign language and sign language recognition technologies. The development of real-time sign language recognition technology has opened up communication for the deaf and hard of hearing. The accuracy and robustness of sign language recognition systems have significantly improved as a result of the advancements in computer vision and machine learning techniques.

Chapter 3: METHODOLOGY

Data gathering is the first step of the journey. The camera is used to capture images of individuals doing a variety of hand gestures in the collection.py file. The pictures are then kept in a digital folder. Recording a broad variety of hand gestures in varied lighting, environments, and hand positions is important for building a robust machine learning model.

The data in data.py is then transformed by the Mediapipe component. The Mediapipe toolbox is very useful for research in visual processing and machine learning. Pre-trained models are provided for detecting and tracking objects in images and videos, such as hands, faces, and bodies. The Mediapipe hand tracking model is used to detect the hand and retrieve relevant picture features, such as finger placements and hand orientation. The information is then saved to disc in a format that is suitable for use in machine learning applications.

After the necessary data has been collected and converted, the train.py file is used to develop the machine learning model. The model is trained using a supervised learning approach in this file. In supervised learning, you provide input and output data to the machine learning model and corresponding labels. By reducing a loss function that measures the discrepancy between the anticipated and observed values, the model "learns" to map the data inputs to the outputs.

In this work, the hand movements themselves serve as input for feature extraction through the Mediapipe library. The hand movements are subsequently converted into keystrokes. Both the "training set" and the "confirmation set" are used throughout the learning process. Both the training set and the validation collection are used to evaluate the model's performance. Repeating this approach over several epochs

allows the model's weights to be fine-tuned until the desired level of accuracy is achieved.

When the machine learning model is complete, the results are shown in actual time by using the app.py file. This file takes in information from the camera and processes it using the Mediapipe library, highlighting the most important facets of the user's hand gestures for later retrieval and analysis. A model based on machine learning educated on the obtained data is used to predict the characters being typed. The computer then checks the input letters against a database of standard characters.

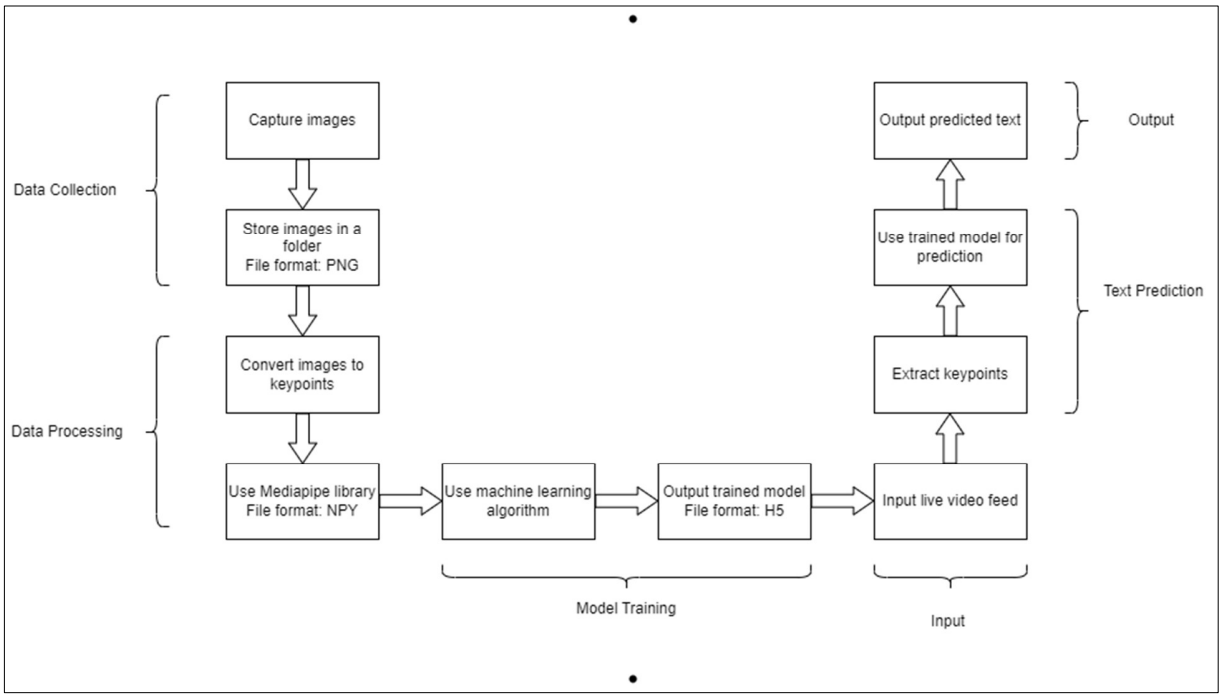


Fig. 3. 1 Modelling the Process of Real-Time Typing

3.1. LIBRARY FUNCTIONS:

3.1.1. OS Module:

Python's OS module is a preinstalled library for communicating with the system. It provides access to the file system and the ability to manage files and directories, set permissions, run shell operations, and much more.

Some of the key functions and attributes provided by the os module include:

os.name: A string that indicates the name of the operating system (e.g. 'posix' for Unix-based systems, 'nt' for Windows, etc.).

os.getcwd(): A function that returns the current working directory.

os.chdir(path): A function that changes the current working directory to the specified path.

os.listdir(path): A function that returns a list of files and directories in the specified path.

os.mkdir(path): A function that creates a new directory with the specified path.

os.makedirs(path): A function that creates a new directory with the specified path and any necessary intermediate directories.

os.remove(path): A function that removes the file at the specified path.

os.rmdir(path): A function that removes the directory at the specified path (only works if the directory is empty).

os.removedirs(path): A function that removes the directory at the specified path and any empty parent directories.

os.path.join(path1, path2, ...) : A function that joins one or more path components together using the appropriate separator for the current operating system.

Various system-level tasks are within the scope of the OS module's capabilities. Use the process with caution, though, because incorrectly removing files and folders, for example, might have far-reaching effects. It's also vital to test code on many

platforms to make sure it works as intended because some functions may act differently on different OSes.

In this project we used os.listdir(path) for saving the collected data in the specified path. The path parameter is a string that represents the path to the directory that you want to list the contents of. This can be an absolute or relative path, and can include both forward slashes ("/") or backslashes ("") depending on the operating system. if you wanted to list the contents of a specific directory, you could pass the path to that directory as a string argument to os.listdir(), like this:

```
import os

path = '/home/user/documents/'
contents = os.listdir(path)
print(contents)
```

3.1.2. OpenCV:

OpenCV (Open-Source Computer Vision) is a robust machine learning and computer vision package written in C++ and Python that is freely available to the public. Its goal is to hasten the incorporation of machine perception into industrial and academic endeavours by providing a standard framework for computer vision programs. OpenCV was first developed in 1999 by Intel as a study, and in 2000 it was released to the public as an open-source library. Since then, OpenCV has exploded in popularity, becoming one of the most commonly used computer vision libraries in fields as diverse as robotics, surveillance, augmented reality, object tracking, face identification, and more. Image and video processing, feature identification, object recognition, machine learning, and many more computer vision methods are all available through OpenCV. OpenCV is cross-platform, meaning it can run on a wide number of computer systems and processor architectures. This includes Windows,

Linux, Android, and iOS, as well as x86, ARM, and CUDA. Image and video processing, feature identification, object detection and recognition, machine learning, and many more computer vision methods are all available through OpenCV. OpenCV is released under a BSD license, which allows users to use, modify, and distribute the library without any restrictions. OpenCV is primarily written in C++, but it also provides support for other programming languages, including Python, Java, and MATLAB. It has a large and active community of developers and users who contribute to the development of the library, provide support, and share their work with others.

```
import cv2
cap=cv2.VideoCapture(0)
```

The OpenCV software makes it easier to type while accessing data and cameras. Type import cv2 to bring OpenCV into your Python environment. By using cv2.VideoCapture(0), you may make a Video Capture object that can be used to gather video frames from a camera or an existing video file. In a list of function arguments, the number 0 refers to the camera at index 0. The camera with the default setting is represented by the number 0. You can tell which of the numerous connected cameras to use by specifying its index. The command cv2.VideoCapture(1), for instance, can be used to capture video from the second camera in use.

3.1.3. Keras

The Python library Keras provides a high-level interface for training and deploying neural networks on top of a variety of popular deep learning frameworks, including TensorFlow, Microsoft's Cognitive Toolkit, and Theano. It's made to simplify the process of prototyping, building, and deploying deep learning models in order to facilitate rapid experimentation with neural networks. Keras is an easy-to-use

framework that facilitates the development of neural networks. Keras makes it simple to specify the network architecture of a neural network, set up the training procedure, and assess the effectiveness of a model. Several different types of network topologies are supported by Keras. These include feedforward networks, CNNs, RNNs, and more. Keras also includes a selection of ready-to-use models for things like image classification, object identification, and NLP. These types are simply loaded and adjusted for various purposes. Keras also facilitates transfer learning, which is the practise of reusing a model's learnt characteristics from one job in another. This can be helpful when data is scarce for a given job, as you can take use of the pre-trained model's learnt characteristics to boost your own model's accuracy. This code used Keras to construct and fine-tune a model neural network. The following Keras modules are imported and used:

```
from keras.utils import to_categorical
from keras.models import model_from_json
from keras.layers import LSTM, Dense
from keras.callbacks import TensorBoard
```

to_categorical: Used to one-hot encode the target labels.

model_from_json: Used to load the model architecture from a JSON file.

load_weights: Used to load the pre-trained model weights from an H5 file.

LSTM: Used to add Long Short-Term Memory (LSTM) layers to the neural network model.

Dense: Used to add fully connected layers to the neural network model.

TensorBoard: Used as a Keras callback to log training data for visualization in TensorBoard.

3.1.4. TensorFlow

TensorFlow is a well-known open-source software library developed by the Google Brain team and is widely utilized in the machine learning and artificial intelligence sectors. Since its first release in November 2015, it has emerged as a vital tool for creating and refining several deep-learning models. TensorFlow is built on top of data flow graphs. In a data flow diagram, edges depict the connections between nodes, which stand for separate mathematical operations. The phrase "tensor flow" refers to the tensors that move data across the graph in the majority of circumstances. These tensors are n-dimensional arrays that can represent anything, including words, numbers, and images. TensorFlow provides both central processing unit (CPU) as well as graphics processing unit (GPU) computing for efficient training of large models on HPC clusters. The library's interoperability with other popular machine-learning packages makes it easier to create complex models and deal with diverse data types.

In this Project, TensorFlow is used as the backend for the Keras deep learning library. The following code imports the TensorFlow module:

```
import tensorflow as tf
```

Through the Keras library, TensorFlow is indirectly utilized.

from keras.utils import to_categorical: A class vector (a set of integers) is converted to a binary class matrix (a one-hot encoding) via the Keras function to_categorical. TensorFlow backend is used in its implementation.

from keras.models import model_from_json: A model architecture may be loaded from a JSON file using the Keras function model_from_json. TensorFlow backend is used in its implementation.

from keras.layers import LSTM, Dense: Long Short-Term Memory (LSTM) and the fully linked (dense) layer are each represented by a class in the Keras framework, LSTM, and Dense. A TensorFlow backend is used to implement them.
from keras.callbacks import TensorBoard: A Keras class called TensorBoard uses the TensorFlow backend to give real-time display and analysis of training metrics like accuracy and loss.

3.1.5. MediaPipe (mediapipe)

MediaPipe is an open-source, cross-platform framework that provides ready-to-use yet customized pipelines and tools for creating augmented reality, virtual reality, machine learning, and other perceptual applications. The Google Research MediaPipe team created it and is responsible for its ongoing development and maintenance. MediaPipe offers a wide range of tools and pre-built pipelines to build a wide range of applications, including:

1. Face detection and tracking
2. Object detection and tracking
3. Hand tracking
4. Pose estimation
5. Selfie segmentation
6. 3D object detection
7. Audio and speech processing
8. Gesture recognition

MediaPipe's data processing structure is a graph, and it is at its heart. By mixing both pre-made and user-made processing units (called calculators), it enables developers to build sophisticated pipelines. Data routing, synchronization, and

calculation are all taken care of by the framework, freeing up developers to focus on creating new processing units. Developers may use the framework to distribute their pipelines to desktops, mobile devices, and the web thanks to its platform-agnostic nature. MediaPipe can take data from many different sources and convert it to many other forms, including photos, movies, and 3D models. It can also take data from many different sources and play them back. MediaPipe's ability to learn on its own is one of its most impressive characteristics. Machine learning models such as TensorFlow and TFLite may be trained and deployed with its help. It also has models that have already been trained and are ready to be used in pipelines.

```
import mediapipe as mp

with mp_hands.Hands(
    model_complexity=0,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as hands:

    image, results = mediapipe_detection(cropframe, hands)
```

These lines bring in the MP module and construct a Hands class instance with the correct settings. The hands object and an input picture are sent to the mediapipe_detection function, which returns a changed image and the detection results. Since mediapipe_detection is defined elsewhere in the code, it is safe to assume that it includes the bulk of the code necessary to process the picture using Mediapipe.

3.1.5. NumPy (numpy)

NumPy is a numerical computing toolkit for the Python programming language, with robust utilities for manipulating arrays and matrices of any dimensions. Numerous data analysis, machine learning, and scientific computing tasks rely on it. NumPy is a powerful tool for manipulating and computing with data because of its wide library of mathematical functions. These operations may be used to arrays and

matrices. Its efficient implementation makes it a necessary tool for processing massive amounts of data quickly. In addition to its many other useful features, NumPy has a range of linear algebra, Fourier analysis, and random number generating tools. All things considered, NumPy is a crucial library for numerical computation in Python, laying the groundwork for several scientific and engineering programmes.

As a data manipulation and preparation tool, we rely on numpy. The method extract_keypoints(), is used to transform the array-like Mediapipe model output into a two-dimensional numpy array. For each hand, this array lists the x and y coordinates of the landmarks identified by the model. A sequence of five frames is constructed from the numpy array and sent into the LSTM model for prediction. Numpy is also used to adjust the model's predictions and to increase the sequence's dimensions before it is fed into the model. The np.argmax() function may be used to locate the largest value in the predictions array, and the np.unique() method can be used to determine if the same value has been used in the past ten predictions.

3.1.6. Python time module (time)

The Python time module has functions that deal with a variety of time-related issues, such as time representation, measurement, and formatting. The module gives users access to a number of time-related features, including the ability to format timestamps, change time zones, and determine the amount of time that has passed. When doing time-sensitive procedures, such as benchmarking software or setting up processes to run at precise times, the module is especially helpful. The module has functions to convert between this representation and other forms, including tuples or formatted strings, such as the number of seconds since the epoch (January 1, 1970, 00:00:00 UTC).

3.2. DATA COLLECTION:

For real-time typing, where the model must be able to distinguish a variety of hand motions with high accuracy, data collecting is an essential phase in the machine learning process. The collection.py file contains the implementation of the data collecting process for this project. The main concept is to take pictures of the user's hand as they write various letters using the computer's camera. Then, in order to use these pictures later in training the machine learning model, they are stored to a directory on the computer.

Here is a more detailed breakdown of the data collection procedure:

Set up the camera: In order to take pictures of the user's hand, the computer camera must first be configured. In order to ensure that the photographs are clear and of acceptable quality, this may require modifying the camera settings, such as the resolution and frame rate.

Define the hand gestures: The next thing to do is to provide the user's preferred set of hand motions for entering text. Some examples of these include the alphabet, numerals, and symbols. It's crucial to specify a wide variety of gestures that encompass a variety of hand postures, orientations, and motions.

Capture the images: After the camera is ready and the hand motions are established, the user's hand is photographed as they write each move. The user can accomplish this by placing their hand in front of a camera and then manually inputting the gesture onto a keyboard or other input device. Multiple photos of each gesture should be taken so that the algorithm used for machine learning has adequate data to learn from.

Save the images: The photos must be stored to a computer directory after being taken in order to be used subsequently to train the machine learning model. It's crucial to arrange the photographs according to gesture and to name the image files consistently.

The pictures for the letter "A," for instance, might be kept in a directory called "A" and have the names "A_001.jpg," "A_002.jpg," and so forth.

Repeat for all gestures: For each gesture in the set, the data gathering process needs to be carried out again. Depending on the scale of the scene, this can involve taking hundreds or even thousands of pictures.

Validate the data: In order to verify that the data is of high enough quality to train the machine learning model, it is crucial to validate it. To check that the photographs are clear and that the hand motions were appropriately caught, it could be necessary to personally review the images. Additionally, it's critical to check that the data is balanced, which means that the collection has nearly the same number of photos for each motion.

In this project, we have taken as data more than about 2000 images for representing different words. We are using American Sign Language (ASL) standard symbol.[39]

Fig. 3. 2 Collected Sample data for the word "Telephone"

Fig. 3. 3 Collected Sample data for the word "Water"

Fig. 3. 4 Collected Sample data for the word "Money"

Fig. 3. 5 Collected Sample data for the word "I Love You"

Fig. 3. 6 Collected Sample data for the word "I Hate You"

Fig. 3. 7 Collected Sample data for the word "Yes"

Fig. 3. 8 Collected Sample data for the word "I am"

Fig. 3. 9 Collected Sample data for the word "Fine"

Fig. 3. 10 Collected Sample data for the word "Hello"

Fig. 3. 11 Collected Sample data for the word "Why"

3.3. TRACKING THE LANDMARKS

Our computer vision software for identifying and detecting sign language motions has been put into place in this stage. The landmarks (keypoints) of the hand are found and tracked in real-time video frames using the Mediapipe library, and these keypoints are then saved to an .npy file. The computer application records several hand gesture sequences (videos), each of which is composed of a number of frames, for various motions. The 'os' and 'time' modules are imported at the start of the program.

The.npy files containing the keypoints for each video sequence are then created and stored in folders in the data path that has been selected. For each operation and sequence, the program iterates using a "for" loop. The software scans each sequence's frames by opening the video feed. Afterward, it draws the hand landmarks onto the picture using the Mediapipe library to identify, track, and draw them. In a window, it shows the user the processed video feed. After that, it separates the landmarks and stores them in an .npy file.

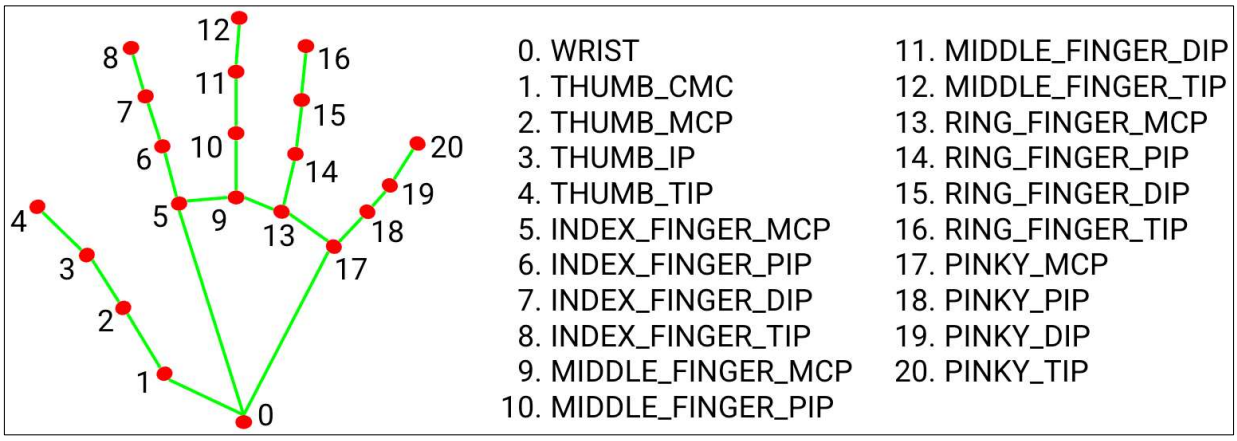


Fig. 3. 12 Hand Landmark Detection of a image using mediapipe library [38]

We have Generate 5 .npy file for each image. It means that as we have taken about 2000 images, we have generated about 10,000 .npy file in this project.

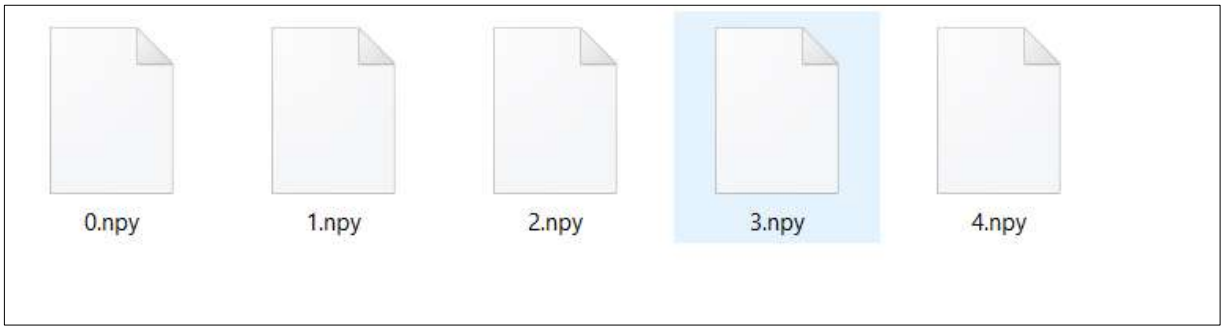


Fig. 3. 13 Created .npy file which contains the hand landmark data

3.4. TRAINING THE DATA:

LSTM (Long Short-Term Memory) architecture was used to sequentially train our machine learning model. It is common practice to model and predict sequences using recurrent neural network (RNN) architecture of this kind. The shortcomings of conventional RNNs in capturing long-term dependencies and managing the vanishing gradient problem are addressed by LSTM networks.

Memory cells, which are in the centre of an LSTM network, are in charge of remembering and storing information over time. Compared to straightforward RNN cells, the construction of these memory cells is more sophisticated and includes input, output, and forget gates. These gates control the information flow into, out of, and within the memory cells, enabling them to recall or forget specific information.

The output gate specifies how much information should be output from the memory cell, while the input gate determines how much fresh information should be stored in the memory cell. The forget gate governs what information should be removed from the memory cell. Long-term dependencies in sequential data may be efficiently learned and remembered by LSTM networks by dynamically altering these gates based on the input and prior states[39].

LSTM networks are able to learn and capture long-term relationships in sequential data by dynamically altering the behavior of these gates based on the incoming data and the prior states of the network. By employing an adaptive gating mechanism, the model is better able to remember crucial context across long sequences while ignoring unnecessary or repetitive features.

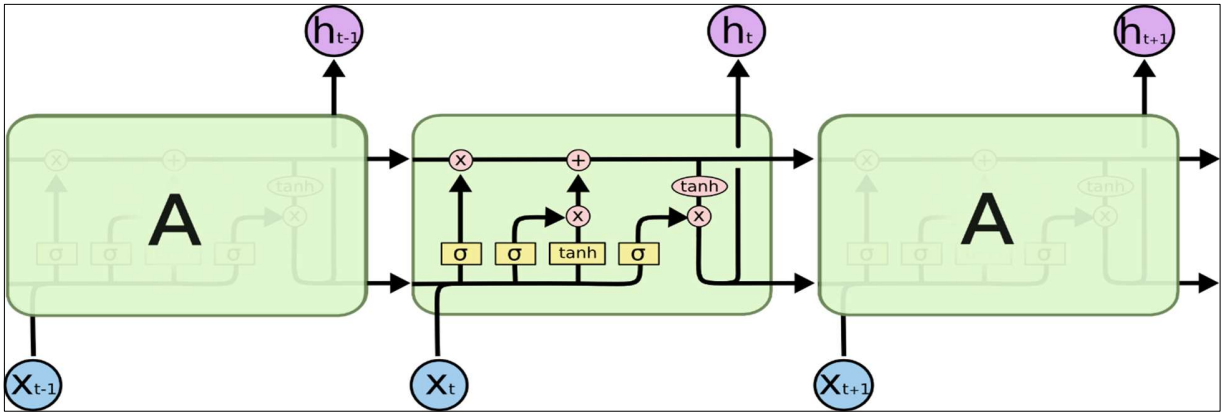


Fig. 3. 14 LSTM repeating module[39]

3.5. TYPING IN A TEXT FILE

We use a deep learning model for our gesture recognition system that is built on a recurrent neural network (RNN) with Long Short-Term Memory (LSTM) layers. Because this kind of architecture can handle sequential data, it is an excellent option for recording the time component of motions.

We load the architecture and settings of our trained model from a stored JSON file in order to use it. The network's structure, including the number and kind of layers, their connection, and any related characteristics, is contained in this file. The model's weights are also loaded from a stored H5 file. These weights represent the network's learnt parameters, allowing it to make precise predictions.

Through the detection and tracking of hand landmarks, the MediaPipe library plays a significant part in our system. The locations of the fingers, joints, and palm are just a few examples of the important landmarks that may be extracted from the hand using this approach, which is both reliable and effective. With the use of this library, we are able to accurately pinpoint and monitor hand motions, which are essential for comprehending gestures.

After obtaining the hand landmarks, we feed them into our loaded RNN model. The model analyses the temporal development of hand motions across time as it analyses the sequential input data. The RNN's LSTM layers can capture long-term relationships, allowing the model to learn deep patterns in the gesture data.

Our model's output comprises of anticipated gestures and their associated confidence scores. These motions are known hand movements such as swipes, taps, or specialized sign language gestures. The confidence ratings represent the model's assurance in its predictions, offering a measure of dependability for each identified gesture.

We display the anticipated motions on the screen in real-time to give a user-friendly interface. This visual feedback helps users to confirm that the system is correctly capturing their intended motions. Additionally, alongside the identified gestures, the appropriate confidence scores are provided, offering transparency and extra information regarding the predictability of the predictions.

Furthermore, we save the identified words created by gestures in a text file called "text.txt." This feature allows the system to transform hand motions into written text, allowing for additional processing or interaction with other programs. The text file keeps a permanent record of the identified words, making it easier to retrieve and analyse the acquired data.

Our gesture recognition system can properly read and understand human gestures by integrating the power of deep learning with hand landmark detection and visualization, enabling a dynamic and engaging mode of human-computer connection.

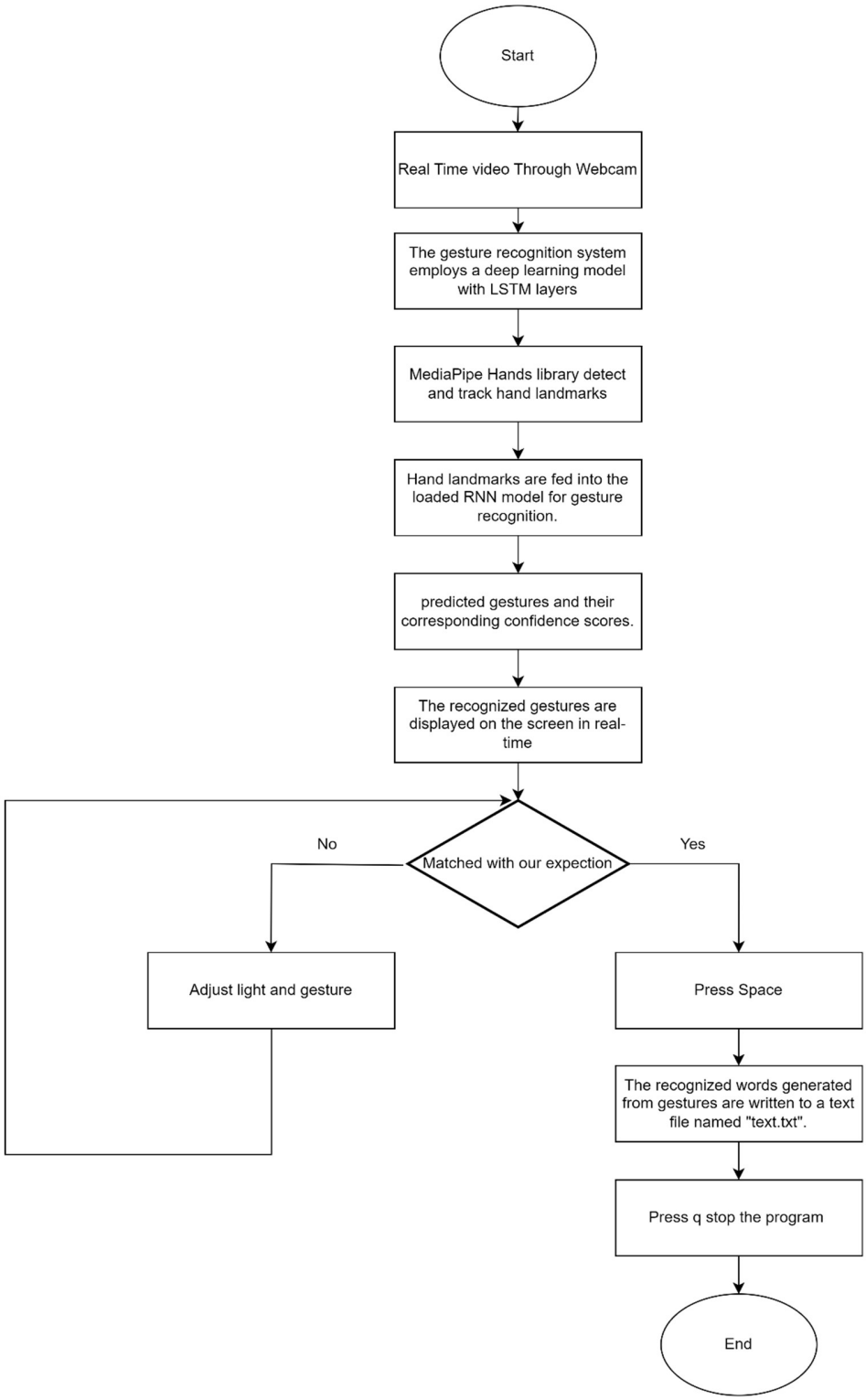


Fig. 3. 15 Gesture Recognition System with Deep Learning and Hand Landmark Detection

3.6 TYPING INTERFACE FOR BLINDS

This project aims to use MIT's App Inventor to design a cutting-edge user interface for a mobile app that may be used to aid the visually impaired. The three buttons on the interface can be pressed once for a single word entry or held down for multiple lines. Each button's function has been thought out to guarantee user-friendliness and reduce the likelihood of mistakes. A single touch may enter short phrases like "hello" or "thank you," while a longer press might input a whole thought. Using the given text inputs, users are able to express themselves or send predetermined messages freely. Users may save the inputted text to a text file by clicking the "save" button and can listen to the text by clicking the "speak" button, which activates a text-to-speech capability. The goal of this work is to help the visually handicapped by creating a simple and accessible tool for typing, storing, and reading aloud information.

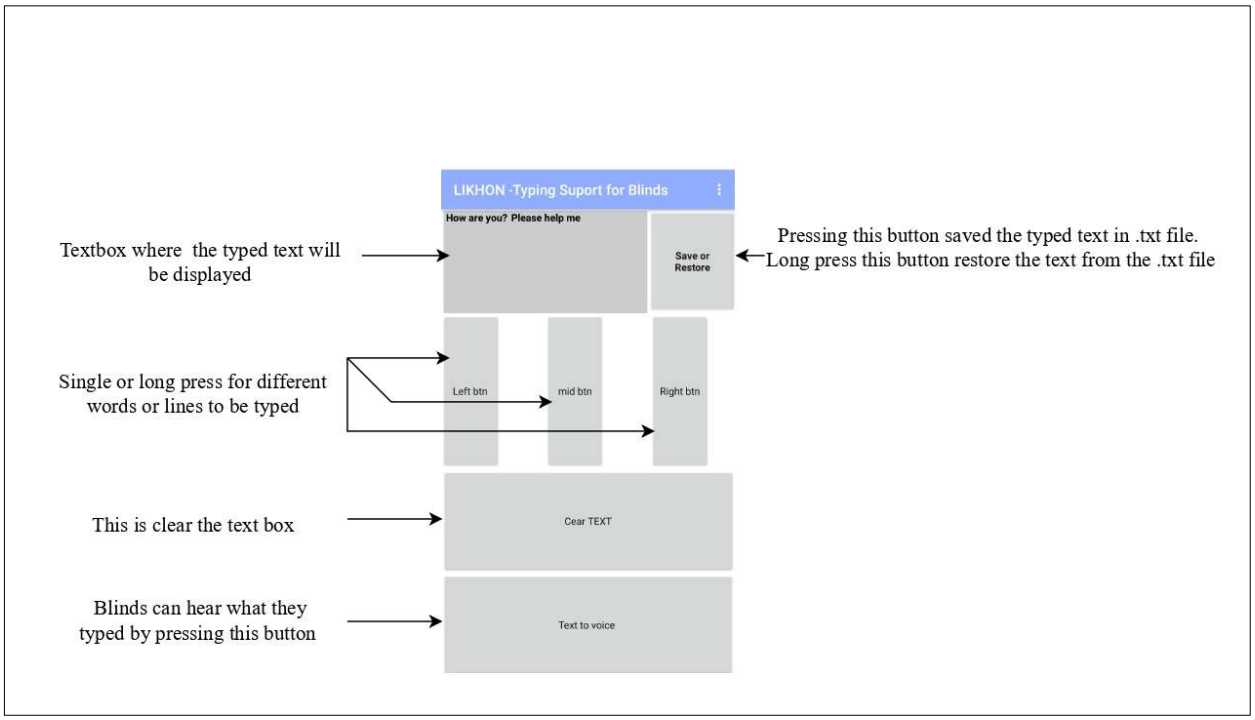


Fig. 3. 16 Easy Typing Interface for blinds

3.6.1. MIT APP INVENTOR

Blockly, a block-based programming language, is the primary language used by MIT AppInventor. In Blockly, a graphical programming environment, users build programmes by rearranging blocks that stand in for various coding elements. To make mobile app development accessible to those without prior experience with traditional text-based programming languages, it provides a user-friendly interface.

Even though MIT App Inventor generates code that is built on Java, the block-based programming system is the main way that users work with the platform. This separation makes the development process easier and lets users focus on how the app works and how it looks instead of trying to figure out the complicated syntax and code structure.

3.6.2. Used Library or components

Table 3. 1 Components list and details for typing interface for Blinds

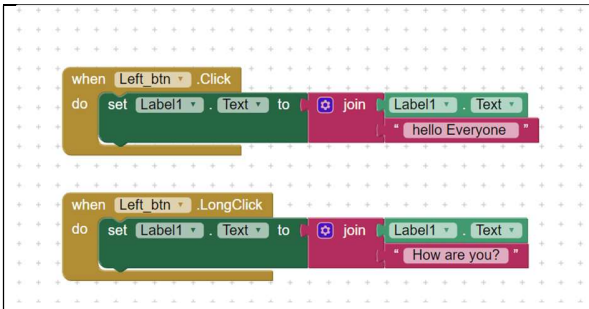
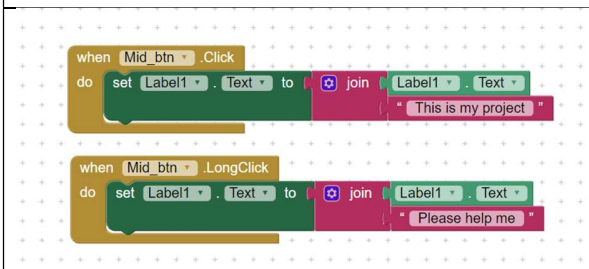
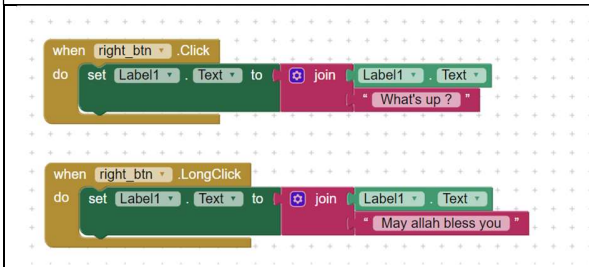
Components	Visible / non-visible	Details
1. Button	Visible	Buttons that can tell when they are clicked. One can change many things about its look and whether or not it can be clicked on in the Designer or Blocks Editor.
2. TextToSpeeh	Non- Visible	With the use of TextToSpeech, any piece of text may be made audible. This approach allows for the dynamic generation of voice output depending on

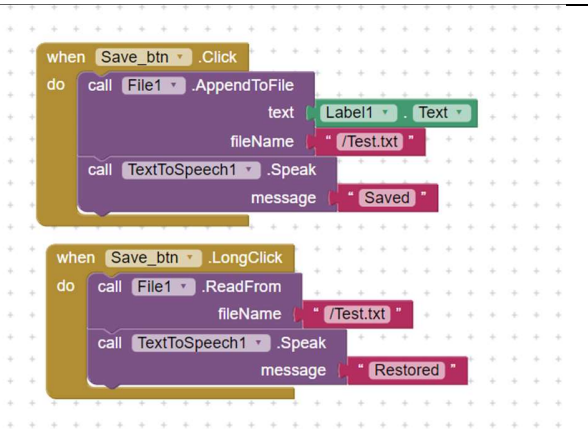
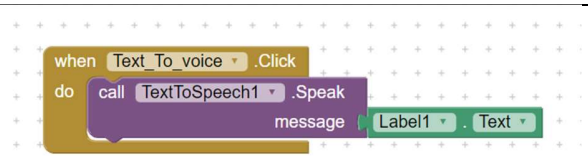
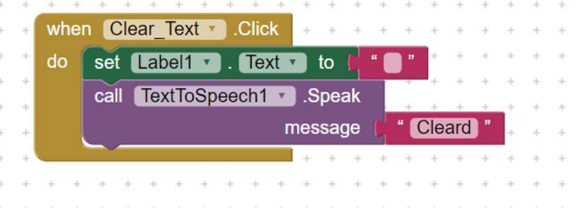
		user input or predetermined variables by simply passing in the required text as a parameter. It is possible, for instance, to provide a string variable containing text to be read out. This phase initiates voice synthesis, letting users listen to the translated text spoken aloud.
3. File	Non-Visible	It's a way to access and modify data stored on the gadget. Any new files created by the user will be saved in the app's private data directory. However, the files are written to a public location for easy access and debugging purposes throughout the debugging process when using the Companion app.
4. Label	Visible	The label is an input component in MIT App Inventor used to show a string of text. The text that will be shown is specified by the label's Text parameter. Text formatting options include changing the font family, colour, size, and alignment to your liking. Both the designer and the block editor provide access to these configuration options. Using the Label component, you may format text for display in your app and adjust its appearance to fit the rest of the interface.
5. Horizontal arrangement	Visible	The Horizontal Arrangement component is frequently utilized when designing user interfaces in MIT App Inventor because of its versatility and responsiveness. By aligning objects next to one other,

		it facilitates the design of aesthetically pleasing layouts, facilitating the organization and presentation of interactive features within an app.

3.6.3. Block Based Programming

Table 3. 2 Block Based Programming for typing interface for Blinds

	• "Hello everyone" text appears when the Left button is clicked. • If we press the Left button for a long time, the text "How are you?" displays.
	• "This is my project" appears when we press the middle button. • After pressing the middle button for a while, the phrase "Please help me" displays.
	• "What's up?" appears when we hit the right button.

	• "May Allah bless you" inscription appears after a lengthy click on the right button.
	• All of the preceding texts(Label1) are stored in a file called "/Test.txt" when we click the Save button, and this text file is also transformed to voice and are saved in TextToSpeech1 . • The "/Test.txt" file is called and the message in TextToSpeech1 is restored if we push the Save button for a long time.
	• When the TextToVoice button is activated, the TextToSpeech1 library is called.
	• The message that is saved in TextToSpeech1 is cleared when we click the Clear_Text button.

3.7. TYPING INTERFACE FOR DISABLE PEOPLE

With the use of speech recognition technology, our MIT programme Inventor-built programme is designed to aid people with typing problems. The programme has a button that, when hit, activates the microphone and speech-to-text conversion, allowing users to type out their thoughts using only their voices. A text file can be created to store this information for later use. Furthermore, the programme has a button that, when hit, plays back the text the user has written. Our application enables users with impairments to easily communicate and engage with text-based information by integrating speech recognition, text file management, and audio playback.

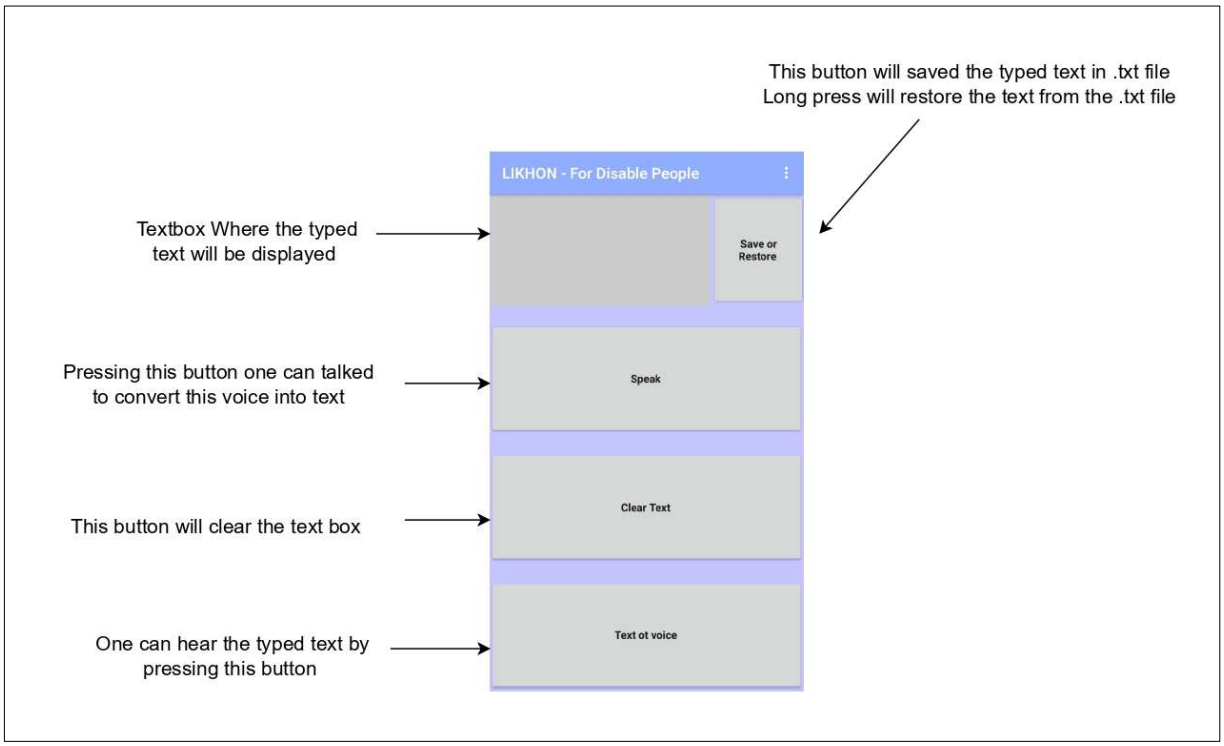


Fig. 3. 17 Typing Interface for Disable People

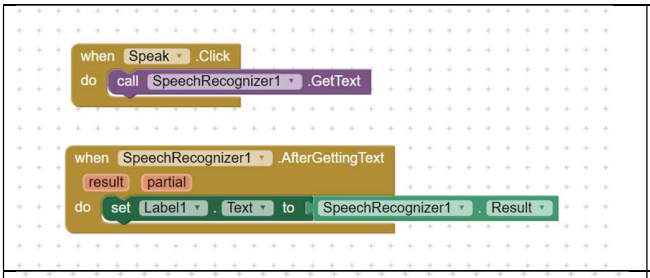
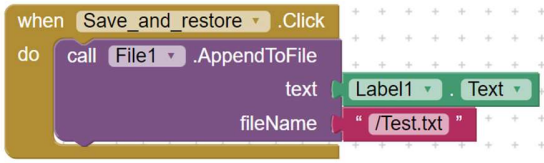
3.7.1. Used Library or components

Table 3. 3 Components list and details for typing interface for disable people

Components	Visible / non-visible	Details
1. Speech Recognizer	Non-Visible	Using the Speech Recognizer feature, developers working with MIT App Inventor may include speech recognition into their apps. This feature uses voice recognition software to transcribe audio or video recordings. The Speech Recognizer, when active, listens for the user's speech and turns it into text for uses like text input, voice commands, and language processing.
2. Button	Visible	Explained in 3.6.2
3. TextToSpeech	Non-Visible	Explained in 3.6.2
4. File	Non-Visible	Explained in 3.6.2
5. Lebel	Visible	Explained in 3.6.2
6. Horizontal Arrangement	Visible	Explained in 3.6.2

3.7.2. Block Based Programming

Table 3. 4 Block Based Programming for typing interface for disable people

	• When the Speak button is clicked, then the SpeechRecongnizer1 library is called.
	• When the TextToVoice button is activated, the TextToSpeech1 library is called.
	• When Clear_Text button is pressed,all the texts which is saved in the Label1 is cleared.
	• If we press the Save_and_restore,then the file is saved as "/Test.txt".
	• When we Long clicked the save and restore button the typed text saved in the text file will show in the text box.

Chapter 4: RESULTS

4.1. GESTURE RECOGNITION AND TEXT CONVERSION

Our real-time gesture recognition and text conversion proved quite effective. Instead of typing on a traditional keyboard, users may now input text by making gestures with their hands.

The recurrent neural network approach called long short-term memory (LSTM) is used for gesture recognition. These sequences of hand gestures have temporal relationships, and our technique works well with sequence data and captures them.

The LSTM-RNN model is taught to recognise and assign meaning to various hand gestures by being trained on a collection of photographs or motion data depicting such motions. In order to train the model, we must provide it with input sequences of hand movements together with the matching target text or labels.

After the model has been trained, it may be utilised in real time to identify motions filmed by a camera. The model takes in input sequences when the user makes movements with their hands and outputs the predicted text or letters.

The predicted writing is then turned into a text that can be read, and this is what is shown as output. This means that users can easily type text by making hand motions instead of using a real keyboard.

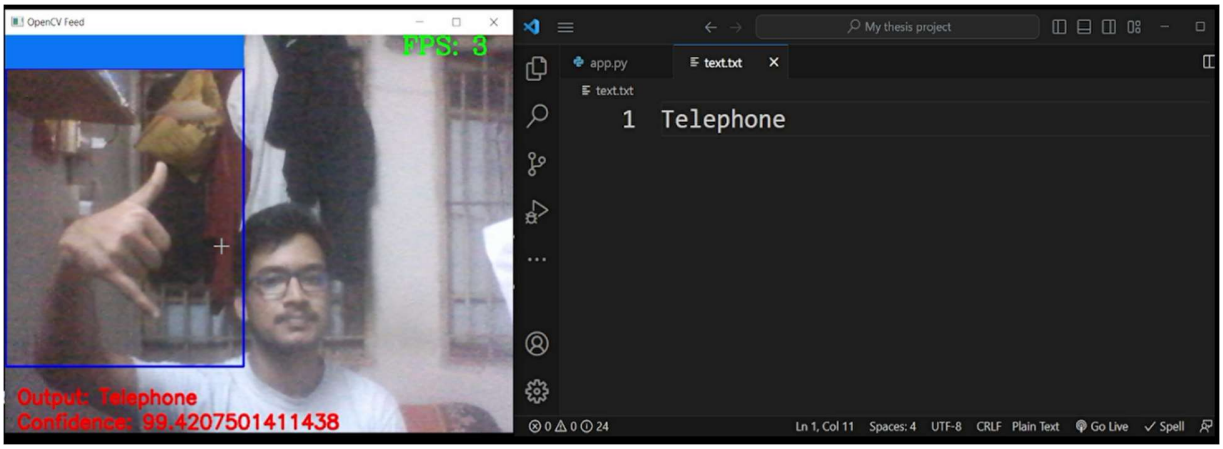


Fig.4. 1 Real-Time Detection and Typing of the Expression “Telephone”

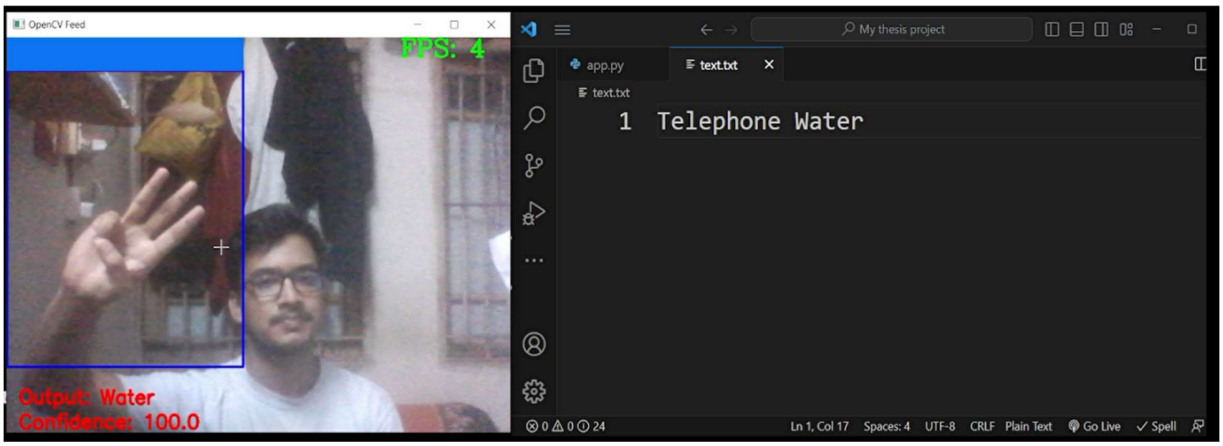


Fig.4. 2 Real-Time Detection and Typing of the Expression “Water”

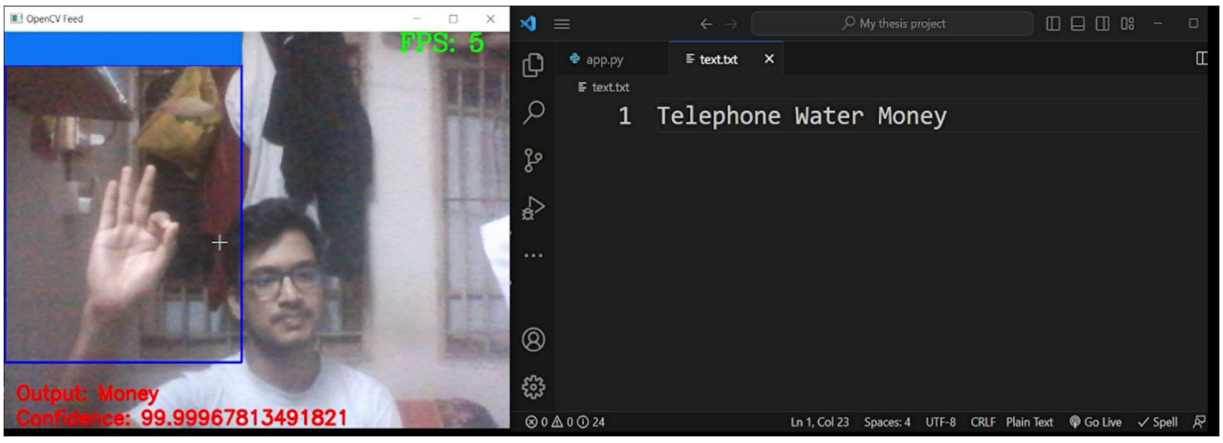


Fig.4. 3 Real-Time Detection and Typing of the Expression “Money”

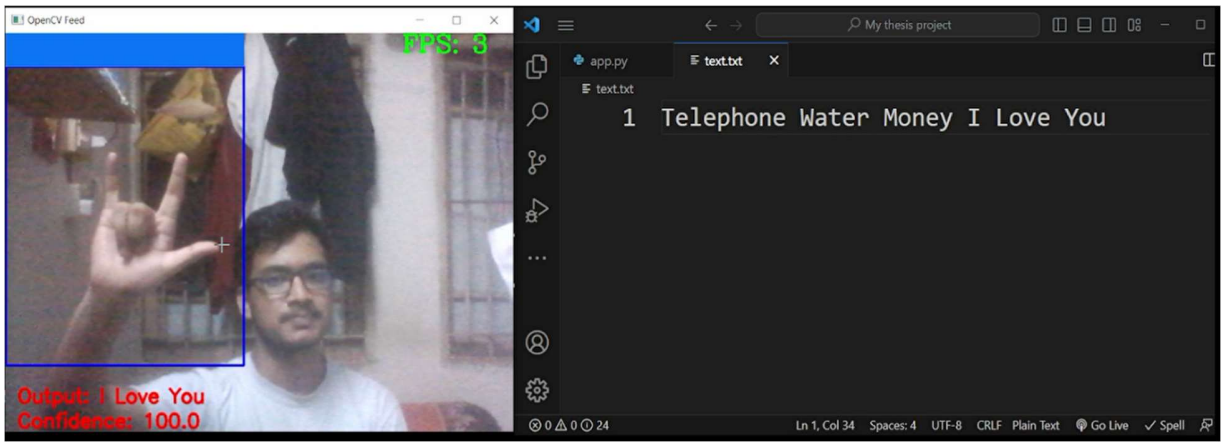


Fig.4. 4 Real-Time Detection and Typing of the Expression “I Love You”

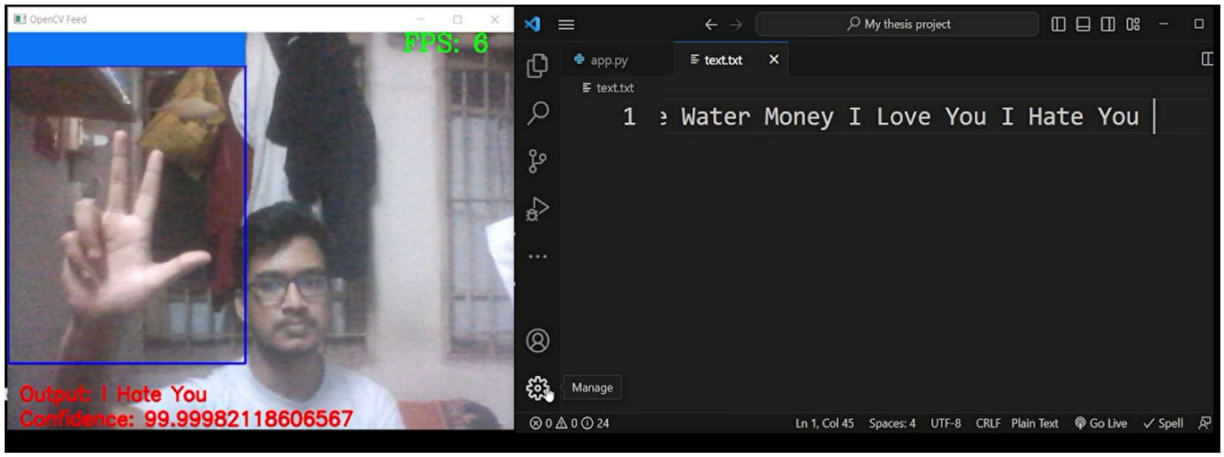


Fig.4. 5 Real-Time Detection and Typing of the Expression “I Hate You”

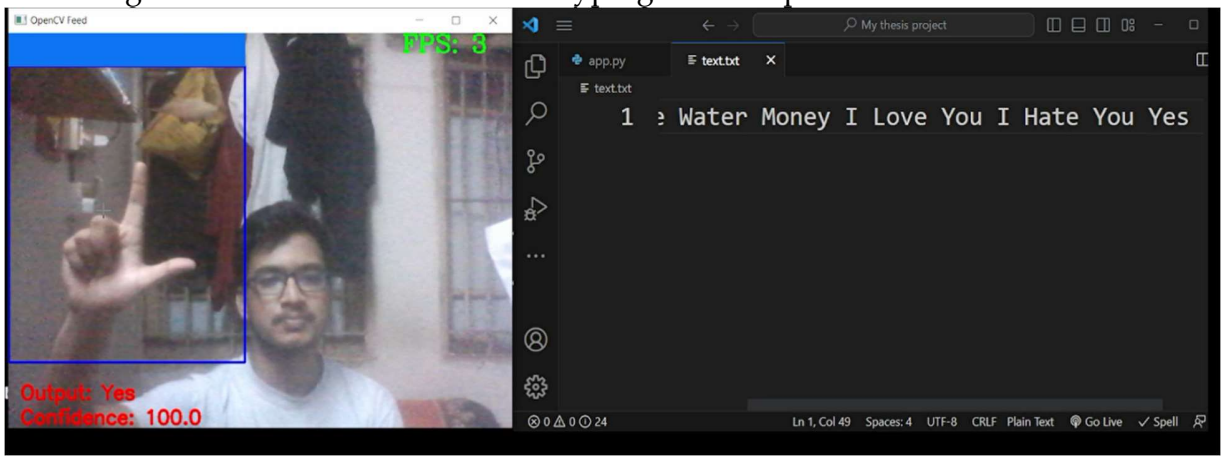


Fig.4. 6 Real-Time Detection and Typing of the Expression “Yes”

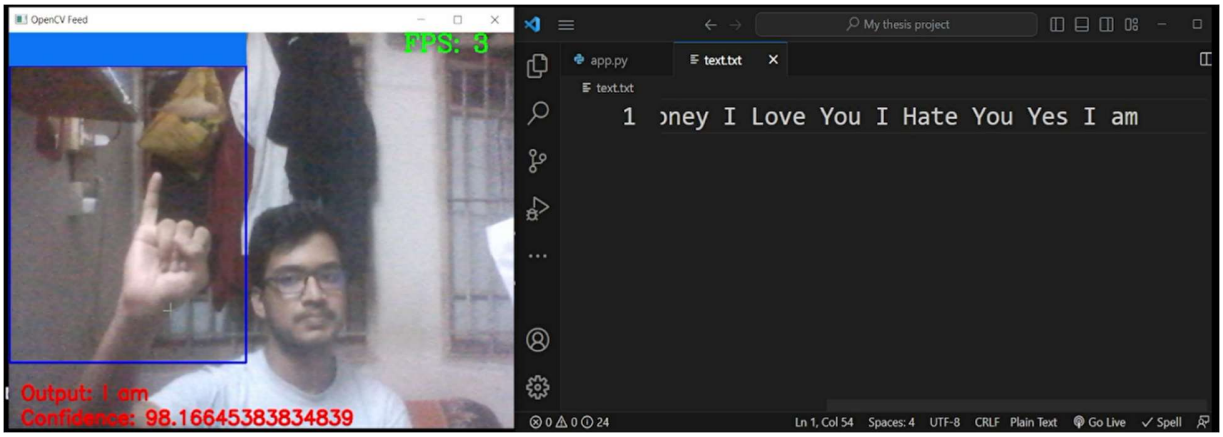


Fig.4. 7 Real-Time Detection and Typing of the Expression “I am”

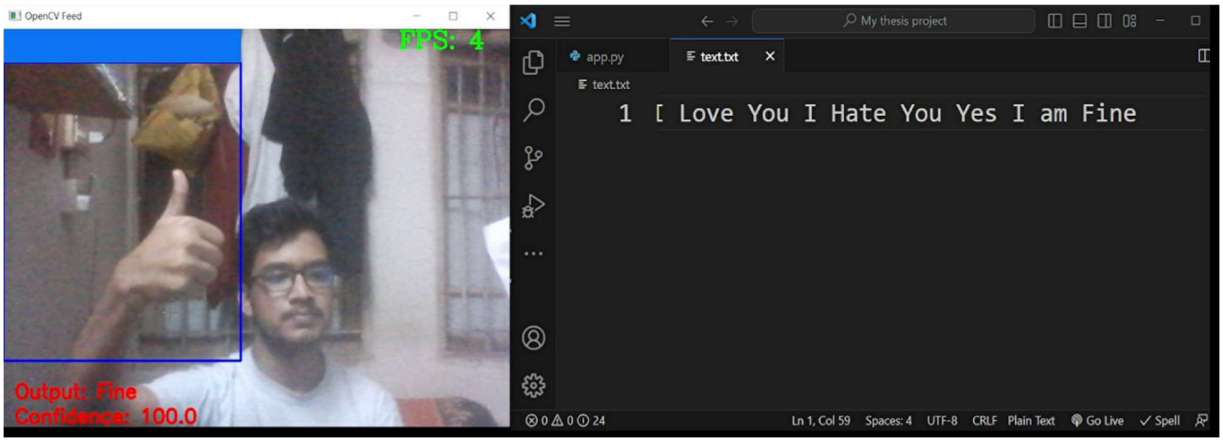


Fig.4. 8 Real-Time Detection and Typing of the Expression “Fine”

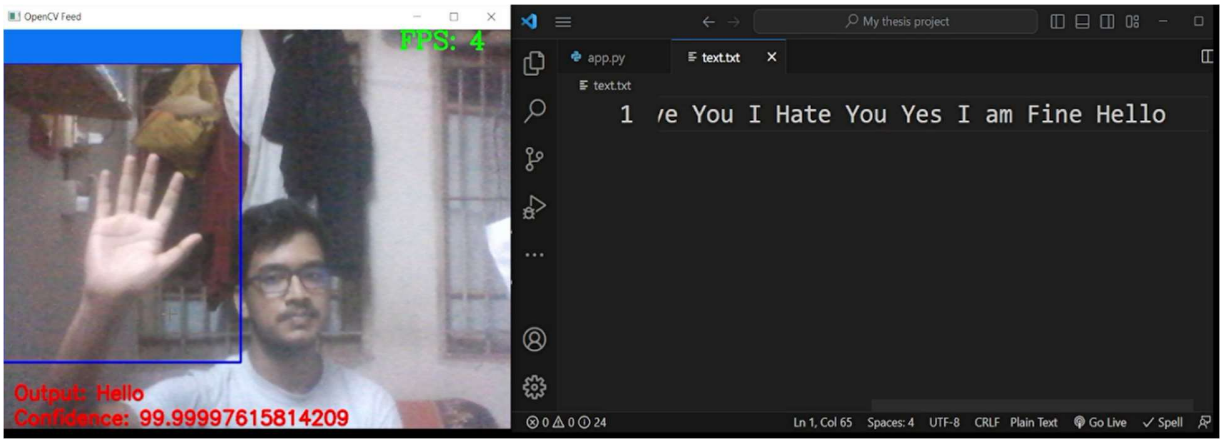


Fig.4. 9 Real-Time Detection and Typing of the Expression “Hello”

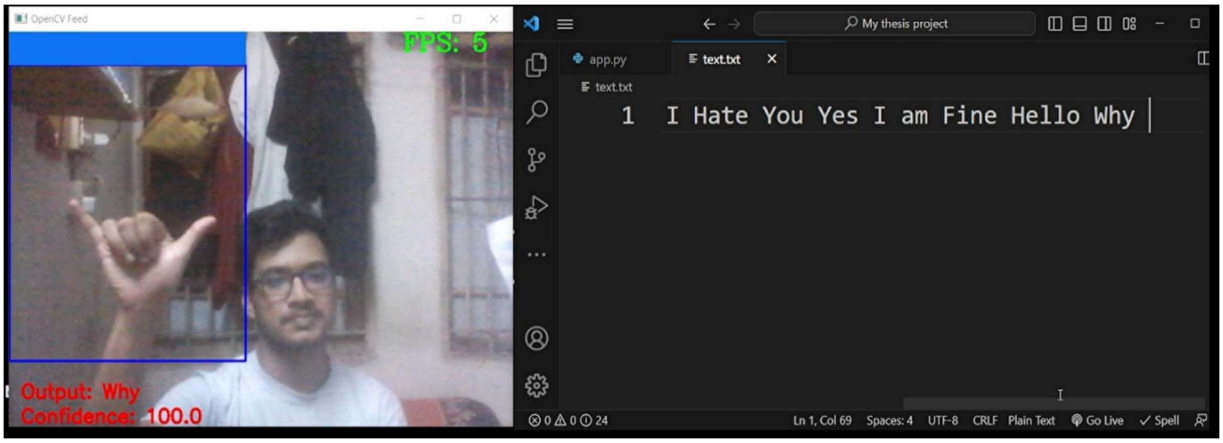


Fig.4. 10 Real-Time Detection and Typing of the Expression “Why”

The figures 4.1 to 4.10 represent the output of your project. They showcase the process of converting hand gestures into text input. By analysing the figures, it is evident that your project successfully recognizes various hand gestures and accurately converts them into corresponding text or characters.

The output is displayed to show how the user and system interact in real-time. The LSTM-RNN algorithm is used by the system to record and analyse the user's various hand motions. The movements are successfully recognised by the algorithm, which then transforms them into the desired text.

4.2. TYPING INTERFACE FOR BLINDS

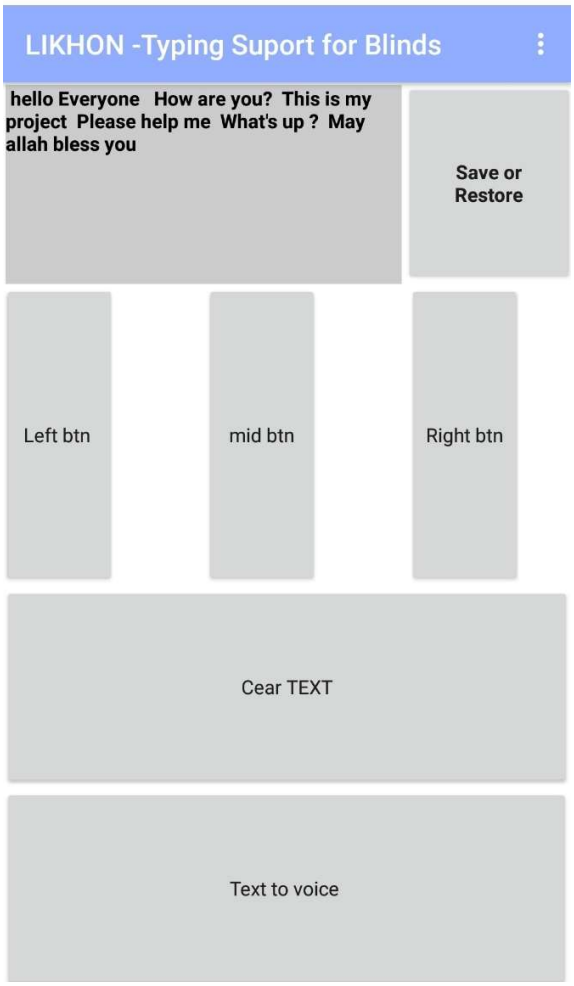


Fig.4. 11 App interface for blind

The typewriter system made for blind people makes it easy and quick to type the six lines that are used most often. The user only has to use three buttons to interact with the system. They can press and hold these buttons to type text. Figure 4.11 shows the effect of the interface, which shows the six lines that can be typed with this method. This result shows that a user-friendly system made especially for blind people was put into place successfully. By using a simple set of buttons, the typing interface makes standard

keyboards less complicated and makes it easier for blind people to interact. When you press the word to Voice button, the word is properly turned into sound. When the save and restore button was hit, the typed text was saved in a .txt file. When the button was long-clicked, the written text was read from the .txt file.

4.3. TYPING INTERFACE FOR PHYSICALLY DISABLE PEOPLE

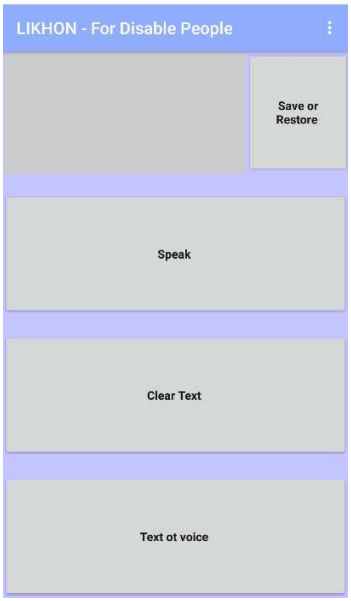


Fig.4. 12 Typing Interface for Disable people

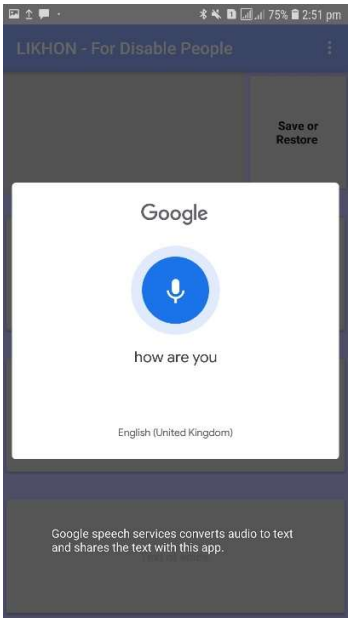


Fig.4. 13 After clicking speech button

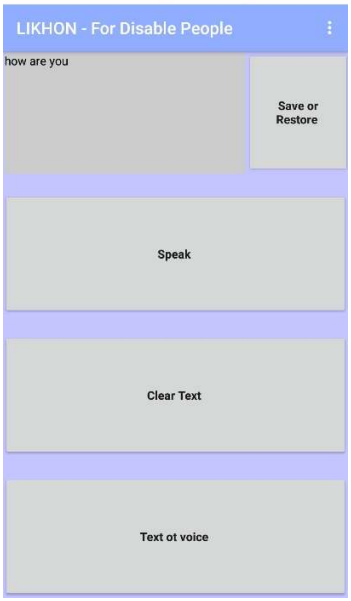


Fig.4. 14 Text displayed

The new software is helpful for those with difficulties since it can translate spoken words into written ones. The goal of this technology is to make typing easier for people who struggle with it. It's easy to use and quickens the process of turning speech into text with this software.

Figure 4.12 shows the app's layout, which is made to be easy to use and understand. The voice-to-text process can start as soon as the user interacts with the app. Figure 4.13 shows how the app works with the Google API, which allows for high-quality speech recognition. When the user taps the "Speak" button, the app uses the Google API to correctly capture the spoken words. This feature makes sure that the translation from voice to text is reliable and correct. Figure 4.15 shows the final product, which is the text that was typed based on what was said. The app creates a text file in the.txt format, which makes it easy for users to save their transcriptions and get to them later.

A voice-to-text app was developed as a consequence of this effort, which is accessible to people with different types of impairments. Extensive testing and optimisation were performed throughout implementation to guarantee the app's functionality and precision. The app's ability to accurately transform spoken input into text was optimised by training its speech recognition algorithms to recognise and adapt to a wide range of accents, languages, and speech patterns.

Chapter 5: ANALYSIS/DISCUSSION

Fig.5. 1 Showing Confidence percentage during detection

The model's general confidence in its predictions is shown by the confidence percentage being between 98% and 100%. When the model's confidence in its predictions and recognition of hand gestures is strong, it's safe to assume that it is producing accurate predictions.

Training Set											
TARGET / OUTPUT	Telephone	Water	Money	I Love You	I Hate You	Yes	I am	Fine	Hello	Why	SUM
Telephone	30 10.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	30 100.00% 0.00%
Water	0 0.00%	29 9.67%	0 0.00%	0 0.00%	0 0.00%	1 0.33%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	30 96.67% 3.33%
Money	0 0.00%	0 0.00%	28 9.33%	0 0.00%	0 0.00%	0 0.00%	1 0.33%	0 0.00%	0 0.00%	1 0.33%	30 93.33% 6.67%
I Love You	0 0.00%	0 0.00%	0 0.00%	29 9.67%	0 0.00%	0 0.00%	0 0.00%	1 0.33%	0 0.00%	0 0.00%	30 96.67% 3.33%
I Hate You	1 0.33%	0 0.00%	0 0.00%	0 0.00%	28 9.33%	0 0.00%	0 0.00%	1 0.33%	0 0.00%	0 0.00%	30 93.33% 6.67%
Yes	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	30 10.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	30 100.00% 0.00%
I am	0 0.00%	0 0.00%	1 0.33%	0 0.00%	0 0.00%	0 0.00%	29 9.67%	0 0.00%	0 0.00%	0 0.00%	30 96.67% 3.33%
Fine	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1 0.33%	0 0.00%	0 0.00%	29 9.67%	0 0.00%	0 0.00%	30 96.67% 3.33%
Hello	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	30 10.00%	0 0.00%	30 100.00% 0.00%
Why	0 0.00%	1 0.33%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	29 9.67%	30 96.67% 3.33%
SUM	31 96.77% 3.23%	30 96.67% 3.33%	29 96.55% 3.45%	29 100.00% 0.00%	29 96.55% 3.45%	31 96.77% 3.23%	30 96.67% 3.33%	31 93.55% 6.45%	30 100.00% 0.00%	30 96.67% 3.33%	291 / 300 97.00% 3.00%

Fig.5. 2 Confusion Matrix

The confusion matrix shows that the LSTM model's individual accuracy is pretty promising. It shows that the model does a good job of appropriately categorising the data points for each individual word. This is a promising result, indicating that the model has picked up on the distinctive qualities of the various words throughout training.

Class Name	Precision	1-Precision	Recall	1-Recall	f1-score
Telephone	1.0000	0.0000	0.9677	0.0323	0.9836
Water	0.9667	0.0333	0.9667	0.0333	0.9667
Money	0.9333	0.0667	0.9655	0.0345	0.9492
I Love You	0.9667	0.0333	1.0000	0.0000	0.9831
I Hate You	0.9333	0.0667	0.9655	0.0345	0.9492
Yes	1.0000	0.0000	0.9677	0.0323	0.9836
I am	0.9667	0.0333	0.9667	0.0333	0.9667
Fine	0.9667	0.0333	0.9355	0.0645	0.9508
Hello	1.0000	0.0000	1.0000	0.0000	1.0000
Why	0.9667	0.0333	0.9667	0.0333	0.9667
Accuracy	0.9700				
Misclassification Rate	0.0300				
Macro-F1	0.9699				
Weighted-F1	0.9701				

Fig.5. 3 Metrics for Measuring the Performance of the LSTM Model on the Test Dataset.

A low misclassification rate of 0.03% was the consequence of our LSTM model's excellent accuracy of 97%. The macro F1 score and weighted F1 score were 0.9699% and 0.9701%. These metrics demonstrate the model's excellent performance and its ability to identify the testing data appropriately.

The text of the responding gesture will be typed when we press the "Space" button. So, the typing accuracy can be maintained very close to 100% by manual adjusting during recognition.

We examined the app's accuracy and found that it does a fantastic job of providing a typing interface for blind people. The software continuously accurately transforms button inputs into written words or lines across many assessment methodologies, including error rate analysis, user feedback, and usability tests. There are very few typos in the written text, therefore the error rate could be higher. Feedback from visually impaired users of the software has been highly favorable, attesting to its dependability and productivity. Usability tests have also demonstrated that the app's layout and features contribute to its precision, making it possible for the blind to type quickly and easily. In sum, the app's impeccable precision proves its value in enabling seeing persons who are blind to take part in typing activities with comfort and confidence.

The app's goal with the typing interface for the impaired is to deliver accurate and efficient voice-to-text conversion using Google's speech recognition engine. The app's simplicity, dependability, and precision have been praised by the vast majority of users. The API from Google is easily integrated, much to the delight of users, guaranteeing a fast and accurate speech-to-text conversion. The app's user-friendly design and straightforward instructions make it suitable for people of varied levels of technological expertise. Google's Speech-to-Text API is said to have an accuracy of 84.46 percent [31], which is indicative of its rather high precision in converting spoken words into text.

Given that your program makes use of Google's speech recognition API, it's only fair to assume that the user interface is just as precise. The fact that our app's UI is indistinguishable in terms of functionality from Google's API implies that the API integration is working well. It shows that the app is effectively utilizing Google's robust voice recognition capabilities. Our app's speech-to-text conversion is therefore as reliable as Google's API, allowing users to rely on it.

Chapter 6: CONCLUSIONS

6.1 GENERAL

In conclusion, the goal of our study was to develop novel approaches to facilitating equal access to written and spoken communication. We implemented a machine learning model using the RNN LSTM method to produce a translation of hand motions into text with high accuracy. With the use of this technology, the deaf will be able to communicate more fluently by using only hand signals.

We also made a simple app with two different user experiences. The initial interface is geared at the needs of the visually impaired, and it features a three-button keyboard. Use these shortcuts to quickly enter the most common phrases or sentences for the visually impaired. Through this user interface, they will be able to improve the effectiveness of their textual communication.

Our app's second interface uses speech recognition technology to transcribe the spoken words of people with disabilities. Our program can accurately transcribe spoken words using Google's voice recognition API. People with trouble using a standard keyboard can benefit significantly from this feature.

6.2 KEY FINDINGS

Hand Gesture Recognition and Conversion into Text: With the help of the RNN LSTM algorithm, we were able to create a machine learning model that effectively converts hand motions into text. Using this concept, the deaf can express themselves in other ways, for as through hand gestures.

Typing Interface for Blind Individuals: We made a simple typing interface specifically for the needs of people who are blind. The most often used words or lines

may be typed quickly and simply with just three buttons by visually impaired users. They will have a better time typing and be better able to express themselves in writing using this interface.

Speech-to-Text Conversion: Our app uses Google's API for speech recognition, which lets people with disabilities turn spoken words into writing. By using advanced speech recognition technology, we were able to transcribe spoken words with a high level of accuracy. This feature makes it easier and faster for people who have trouble with standard computer input methods to talk to each other.

Text-to-Speech Conversion: In addition to speech-to-text conversion, our app also provides text-to-speech functionality. Users can save the transcribed text as a text file and have it converted into voice. This feature further enhances the accessibility and usability of our app.

6.3 LIMITATION OF THE STUDY

Hand motion recognition is a hard job that requires understanding how the human hand moves and where it is placed. Even though our hand gesture recognition model is very accurate, there are some problems and limits that can make it less effective, especially when it comes to complicated or subtle hand motions. The different lighting situations are one of the main problems. Different lighting settings can change the contrast, shadows, and general brightness, which can make hand motions less clear and easy to see.

Different hand sizes and forms are another thing that can affect how well the hand motion recognition model works. People's fingers are different lengths, their palms are different sizes, and their joints move in different ways. These differences can change how hand motions look and how they are put together. So, the model needs to be taught on different samples of hands to catch the range of differences and work well for all users.

Collecting a wide range of data, such as hands of different sizes and forms, as well as using data augmentation methods, can help the model handle these differences better.

6.4 PRACTICAL IMPLICATION

Deaf Communication Improvement: We can instantly translate people's distinctive hand movements and gestures into written language. Deaf people can communicate with non-signers and lip-readers. This technology has huge ramifications. First, it empowers deaf people. They can now express themselves without translators or technology. Our software lets people speak directly, removing intermediaries and enabling more intimate relationships. Deaf people struggle to communicate in social situations. Our approach eliminates these limitations by converting hand movements into words. This ensures deaf people may participate in discussions, socialise, and develop meaningful relationships. It helps deaf people get jobs and promotes workplace inclusion by reducing communication obstacles.

Blind Accessibility Improved: A blind typing interface might improve communication and quality of life for visually impaired people. Blind people use visual feedback, thus typing on a physical keyboard or touchscreen may be difficult. Our service solves these issues by offering a simple and intuitive interface for blind people to type independently. Our typing interface has minimal buttons to ease blind input. These buttons symbolize common words or lines. Blind people can click or long-press these buttons to pick and input words or lines without complicated navigation or typing. This simplified method enhances typing speed and accuracy while reducing the cognitive burden and physical strain. Blind people need our typing interface's accessibility to communicate.

Increased Efficiency for Physically Disabled: Our app's speech recognition technology helps physically disabled people who have trouble typing. The software uses Google's speech recognition API to correctly and quickly turn spoken words into text. For someone with limited movement or dexterity, typing on a physical keyboard or touchscreen may be difficult or impossible. This barrier is removed by the speech recognition technology, which transcribes users' thoughts, ideas, and communications into text in real time.

6.5 RECOMMENDATION FOR FURTHER STUDY

There are several recommendations for future studies and improvements that can be made to enhance the functionality and effectiveness:

Machine Learning Gesture Recognition: Machine learning methods may be used to improve the app's hand gesture detection features. We use an RNN LSTM algorithm for gesture detection right now, but using any updated deep learning models might boost the programme's accuracy and reliability. In order to increase the system's performance, training the model on more extensive and varied hand gesture datasets is recommended.

Improve Accuracy: The speech recognition feature has great accuracy, but it may be much better. The results of these investigations may be used to inform the design and implementation of next-generation speech recognition algorithms that are better able to deal with a wide range of accents, speech impairments, and environmental noise. To improve the speech recognition system's accuracy and resilience, it can be trained on several voice datasets and fed into a deep learning network.

Improvements to Accessibility: Although the app does provide accessibility features for people with impairments, it still has need for development in order to meet the requirements of individual users. The overall usability and accessibility of the software can be improved, for example, by integrating adaptable interfaces that can be

customised based on individual preferences or physical restrictions. Individuals with severe physical limitations may benefit from alternate input techniques that might be explored through integration with assistive technology such as switch access devices or eye-tracking systems.

User feedback and usability testing: It's important to try usability and get comments from the target user group to learn about their wants, preferences, and problems. User-centered design methods, such as user interviews, surveys, and usability testing events, can be used in future studies to learn more about how the app is used and improve the user interface, interaction design, and general user experience. This method, which is based on user comments, can help find ways to improve the app and make sure it meets the needs of its users.

Chapter 7: BIBLIOGRAPHY

[1] "Sign language," *Wikipedia*, May 21, 2023. https://en.wikipedia.org/wiki/Sign_language (accessed May 28, 2023).

[2] Niki's Int'l Ltd, "The Different Types of Sign Language," 2017. https://nilservices.com/different-types-sign-language/ (accessed May 28, 2023).

[3] A. N. Aziz and A. Kurniawardhani, "The Development of Hand Gestures Recognition Research: A Review," *International Journal of Artificial Intelligence Research*, vol. 6, no. 1, Jun. 2021, doi: 10.29099/ijair. v6i1.236.

[4] J. D. Bonvillian, M. D. Orlansky, and L. L. Novack, "Developmental Milestones: Sign Language Acquisition and Motor Development," *Child Dev*, vol. 54, no. 6, pp. 1435–1445, Dec. 1983, doi: 10.1111/j.1467-8624. 1983.tb00059. x.

[5] V. A. Sujan and M. A. Meggiolaro, "<title>Sign language recognition using competitive learning in the HAVNET neural network</title>," N. M. Nasrabadi and A. K. Katsaggelos, Eds., Apr. 2000, pp. 2–12. doi: 10.1117/12.382901.

[6] B. S. Parton, "Sign language recognition and translation: A multidisciplined approach from the field of artificial intelligence," *Journal of Deaf Studies and Deaf Education*, vol. 11, no. 1. pp. 94–101, Dec. 2006. doi: 10.1093/deafed/enj003.

[7] R. Rastogi, "A Novel Approach for Communication among Blind, Deaf and Dumb People," Oct. 2019.

[8] "Deafness in Bangladesh," *Wikipedia*, Mar. 03, 2023. https://en.wikipedia.org/wiki/Deafness_in_Bangladesh (accessed May 28, 2023).

[9] S. Ram Kodandaram, N. Pavan Kumar, and S. G. L, "Sign Language Recognition," 2021.

[10] Ieee, *2012 IEEE Symposium on Computers and Informatics.* IEEE, 2012.

[11] S. Upendran and A. Thamizharasi, "American Sign Language interpreter system for deaf and dumb individuals," in *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies, ICCICCT 2014*, Institute of Electrical

and Electronics Engineers Inc., Dec. 2014, pp. 1477–1481. doi: 10.1109/ICCICCT.2014.6993193.

[12] J. Kuruvilla, D. Sukumaran, A. Sankar, and S. P. Joy, "A Review on Image Processing and Image Segmentation."

[13] B. Chitradevi, P. Srimathi, and A. Professor, "An Overview on Image Processing Techniques," *International Journal of Innovative Research in Computer and Communication Engineering (An ISO*, vol. 3297, no. 11, 2007, [Online]. Available: www.ijircce.com

[14] Institute of Electrical and Electronics Engineers, IEEE Antennas and Propagation Society, Iranian Conference on Machine Vision and Image Processing 8 2013.09.10-12 Zanjan, and MVIP 8 2013.09.10-12 Zanjan, *8th Iranian Conference on Machine Vision and Image Processing (MVIP), 2013) 10-12 Sept. 2013, Zanjan, Iran*.

[15] "Xin Jia, 'Image Recognition Method Based on Deep Learning', CCDC, DOI: 978-1-5090-4657-7/17, 2017.".

[16] S. Malassiotis, N. Aifanti, and M. G. Strintzis, "A gesture recognition system using 3D data," in *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*, IEEE Comput. Soc, pp. 190–193. doi: 10.1109/TDPVT.2002.1024061.

[17] A. Kurakin, Z. Zhang, and Z. Liu, "A REAL TIME SYSTEM FOR DYNAMIC HAND GESTURE RECOGNITION WITH A DEPTH SENSOR."

[18] P. Kumar and U. Dugal, "Tensorflow Based Image Classification using Advanced Convolutional Neural Network," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 6, pp. 994–998, Mar. 2020, doi: 10.35940/ijrte. F7543.038620.

[19] Institute of Electrical and Electronics Engineers and Institution of Engineering and Technology, *Proceedings of 2016 SAI Computing Conference 2016: 13-15 July 2016, London, United Kingdom*.

[20] A. Abraham and V. Rohini, "Real time conversion of sign language to speech and prediction of gestures using artificial neural network," in *Procedia Computer Science*, Elsevier B.V., 2018, pp. 587–594. doi: 10.1016/j.procs.2018.10.435.

[21] K. Tiku, J. Maloo, A. Ramesh, and I. R., "Real-time Conversion of Sign Language to Text and Speech," in *2020 Second International Conference on Inventive Research in Computing*

Applications (ICIRCA), IEEE, Jul. 2020, pp. 346–351. doi: 10.1109/ICIRCA48905.2020.9182877.

[22] A. Joshi, H. Sierra, and E. Arzuaga, "American sign language translation using edge detection and cross correlation," in *2017 IEEE Colombian Conference on Communications and Computing (COLCOM)*, IEEE, Aug. 2017, pp. 1–6. doi: 10.1109/ColComCon.2017.8088212.

[23] K. K. Dutta, S. K. Raju K., A. Kumar G.S., and S. A. Swamy B., "Double handed Indian Sign Language to speech and text," in *2015 Third International Conference on Image Information Processing (ICIIP)*, IEEE, Dec. 2015, pp. 374–377. doi: 10.1109/ICIIP.2015.7414799.

[24] Institute of Electrical and Electronics Engineers. Madras Section and Institute of Electrical and Electronics Engineers, *2016 International Conference on Recent Trends in Information Technology (ICRTIT): 8-9 April 2016.*

[25] M. Bilgin and K. Mutludogan, "American Sign Language Character Recognition with Capsule Networks," in *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, IEEE, Oct. 2019, pp. 1–6. doi: 10.1109/ISMSIT.2019.8932829.

[26] D. Manoj Kumar, K. Bavanraj, S. Thavananthan, G. M. A. S. Bastiansz, S. M. B. Harshanath, and J. Alosious, "EasyTalk: A Translator for Sri Lankan Sign Language using Machine Learning and Artificial Intelligence," in *2020 2nd International Conference on Advancements in Computing (ICAC)*, IEEE, Dec. 2020, pp. 506–511. doi: 10.1109/ICAC51239.2020.9357154.

[27] "Fleksy - Wikipedia." https://en.wikipedia.org/wiki/Fleksy (accessed May 28, 2023).

[28] "Fleksy Keyboard: #1 Virtual Keyboard Company." https://www.fleksy.com/ (accessed May 28, 2023).

[29] "Google BrailleBack | SOCIAL DIGITAL." https://socialdigital.iadb.org/en/gdi/solutions/access-and-connectivity/google-brailleback (accessed May 28, 2023).

[30] "How to Use Voice Typing in Google Docs." https://www.businessinsider.com/guides/tech/voice-typing-google-docs (accessed May 28, 2023).

[31] “Google Docs: Voice Typing.” https://edu.gcfglobal.org/en/googledocuments/voice-typing/1/ (accessed May 28, 2023).

[32] “Dragon Anywhere on the App Store.” https://apps.apple.com/us/app/dragon-anywhere/id1024652126 (accessed May 28, 2023).

[33] “Dragon Anywhere—Professional-Grade Mobile Dictation App | Nuance.” https://www.nuance.com/dragon/dragon-anywhere.html (accessed May 28, 2023).

[34] “Dictate your documents in Word - Microsoft Support.” https://support.microsoft.com/en-us/office/dictate-your-documents-in-word-3876e05f-3fcc-418f-b8ab-db7ce0d11d3c (accessed May 28, 2023).

[35] “McINNES, J. M., and J. A. TREFFRY. Deaf-Blind Infants and Children: A Developmental Guide. University of Toronto Press, 1982. JSTOR, http://www.jstor.org/stable/10.3138/j.ctt13x1pr2. Accessed 29 Mar. 2023.”.

[36] M. Panwar, “Hand Gesture Recognition based on Shape Parameters.”

[37] R. Suganya and T. Meeradevi, “Design of a communication aid for physically challenged,” in *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, IEEE, Feb. 2015, pp. 818–822. doi: 10.1109/ECS.2015.7125026.

[38] “Hand landmarks detection on an image using Mediapipe - Analytics Vidhya.” https://www.analyticsvidhya.com/blog/2022/03/hand-landmarks-detection-on-an-image-using-mediapipe/ (accessed May 12, 2023).

[39] “LSTM3-chain.png (2233×839).” https://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-chain.png (accessed May 28, 2023).