

**A**  
**PROJECT REPORT**  
**ON**  
**Sign Language Recognition through Deep Learning**

**SUBMITTED IN PARTIAL FULFILMENT FOR THE AWARD  
OF DEGREE OF  
BACHELOR OF ENGINEERING  
IN  
INFORMATION TECHNOLOGY  
BY**

**ISHIKA GUPTA (160119737068)  
KRITIKA AGARWAL (160119737072)  
NISHANTH GANJI (160119737100)**

**UNDER THE GUIDANCE OF**

**MR. S. RAKESH,  
ASSISTANT PROFESSOR,  
DEPT. OF IT, CBIT.**



**DEPARTMENT OF INFORMATION TECHNOLOGY  
CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)**  
(Affiliated to Osmania University; Accredited by NBA and NAAC, ISO 9001:2015  
Certified Institution), GANDIPET, HYDERABAD – 500 075  
Website: [www.cbit.ac.in](http://www.cbit.ac.in)

**2022-2023**



**CHAITANYA BHARATHI  
INSTITUTE OF TECHNOLOGY (A)**

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. [www.cbit.ac.in](http://www.cbit.ac.in)



COMMITTED TO  
RESEARCH,  
INNOVATION AND  
EDUCATION

**44**  
years

## CERTIFICATE

This is to certify that the project work entitled “**SIGN LANGUAGE RECOGNITION THROUGH DEEP LEARNING**” submitted by **ISHIKA GUPTA (160119737068)**, **KRITIKA AGARWAL (160119737072)**, and **NISHANTH GANJI (160119737100)** in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF ENGINEERING** in **INFORMATION TECHNOLOGY** to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY(A)**, affiliated to **OSMANIA UNIVERSITY**, Hyderabad, is a record of bonafide work carried out by them under my supervision and guidance. The results embodied in this report have not been submitted to any other University for the award of any other Degree or Diploma.

### **Project Guide**

**Mr. S. Rakesh**

Asst. Professor, Dept. of IT,  
CBIT, Hyderabad.

### **Head of the Department**

**Dr. Rajanikanth Aluvalu**

Professor, Dept. of IT,  
CBIT, Hyderabad.

## **DECLARATION**

We declare that the project report entitled “**Sign Language Recognition Through Deep Learning**” is being submitted by us in the Department of Information Technology, Chaitanya Bharathi Institute of Technology (A), Osmania University.

This is record of bonafide work carried out by us under the guidance and supervision of Mr. S. Rakesh, Assistant Professor, Dept. of IT, C.B.I.T.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The reported are based on the project work done entirely by us and not copied from any other source.

**Ishika Gupta - 160119737068**

**Kritika Agarwal - 160119737072**

**Nishanth Ganji - 160119737100**

## **ACKNOWLEDGEMENTS**

We would like to express our heartfelt gratitude to Mr. S. Rakesh, our project guide and project coordinator, for his invaluable guidance and constant support, along with his capable instruction and persistent encouragement.

We are grateful to our Head of Department, Dr. Rajanikanth Aluvalu, for his steady support and for the provision of every resource required for the completion of this project.

We would like to take this opportunity to thank our Principal, Dr. P. Ravinder Reddy, as well as the Management of the Institute, for having designed an excellent learning atmosphere.

Our thanks are due to all members of the staff and our lab assistants for providing us with the help required to carry out the groundwork of this project.

## **ABSTRACT**

Most of us today do not realize how powerful words can be and how huge of an impact they can have on our lives. With this project, we aim to provide the power of communication to mute people. To express our emotions and thoughts through words and be understood is a huge privilege. This project intends to provide this privilege to mute people by converting American Sign Language, the most commonly and widely used sign language, into understandable terms. Since sign language is confined to people deprived of speech, this technology will bridge the gap between two sections of society through systems that convert sign language into text and speech. Additionally, we also aim to further minimize the communication gap by allowing the user to choose the language in which they desire the output.

Our proposed system uses the mediapipe hands model and keras sequential model to build a deep learning neural network to recognize the signs accurately. We extract the 42 hand landmarks of the sign being performed in front of the camera, opened using openCV, using the mediapipe hands model and store them in a csv file. We do this for 30 words, so our dataset consists of 30 unique classes with 43 columns and approximately 31,000 rows. Once the dataset is created we train it using the neural network built by the keras sequential model. The model is then trained and is stored and used for predicting the real time signs being performed in front of the camera. The python program is connected to a web interface using Flask, for easier access. We also provide the end users with an option to choose the language in which they would like to observe the output. This allows the application to be used by a wider range of audience. The predicted word text is also converted to speech, so the output can be observed as both text and speech, whichever is more convenient for the user. An accuracy of 99% was achieved.

Compared to other existing systems, this proposed system is expected to be more accurate with a larger vocabulary, simple, helpful, flexible, convenient, and easy to use. It provides realistic communication between the mute and non-mute people since it is able to recognize signs at the word level accurately.

## TABLE OF CONTENTS

	Page. No
Abstract	v
List of Figures	viii
List of Abbreviations	x
1. INTRODUCTION	1
1.1 Overview	1
1.2 Applications	2
1.3 Motivation	2
1.4 Problem Statement	3
1.5 Objectives	3
1.6 Organization of the Report	3
2. LITERATURE SURVEY	4
2.1 Vision Based Approach to Sign Language Recognition	4
2.2 One Model Is Not Enough: Ensembles for Isolated Sign Language Recognition	5
2.3 Hybrid Fist-Cnn Approach for Feature Extraction for Vision-Based Indian Sign Language Recognition	6
2.4 Implementation Smart Gloves for Deaf and Dumb Disabled	8
2.5 Sign Language Interpreter Using Image Processing and Machine Learning	10
2.6 Real Time Sign Language Detection	11
2.7 Arabic Sign Language Recognition Through Deep Neural Networks Fine-Tuning	11
2.8 Development of Smart Glove for Deaf-Mute People	12
2.9 Finger Spelling Recognition for Nepali Sign Language	13
2.10 Gesture Based Arabic Sign Language Recognition for Impaired People Based on Convolution Neural Network	13
2.11 Sign Language Recognition Using Mediapipe	16

2.12	American Sign Language Recognition for Alphabets Using Mediapipe and Lstm	16
2.13	An Integrated Mediapipe-Optimized Gru Model For Indian Sign Language Recognition	18
3.	SYSTEM REQUIREMENT SPECIFICATION	19
3.1	Functional Requirements	19
3.2	Non-Functional Requirements	19
3.3	Software Requirements	20
3.4	Hardware Requirements	21
4.	PROPOSED METHODOLOGY	22
4.1	Dataset	22
4.2	Keras Sequential Neural Network Model	23
4.3	Tuning	23
4.4	Activation Function	24
4.5	Loss function	25
4.6	Optimization	26
4.7	Language Translation	27
4.8	Text to Speech Conversion	28
4.9	Web Interface	28
5.	IMPLEMENTATION	29
6.	TESTING AND RESULTS	35
7.	CONCLUSION AND FUTURE SCOPE	46
6.1	Conclusion	46
6.2	Limitations	46
6.3	Future Scope	47
8.	VISIBLE OUTPUT	48
	BIBLIOGRAPHY	60
	APPENDIX	64

## LIST OF FIGURES

		Page no.
<b>Figure 2.1</b>	Image conversion	4
<b>Figure 2.2</b>	Different data modalities given as input	6
<b>Figure 2.3</b>	Architecture of FiST_CNN for ISL	7
<b>Figure 2.4</b>	Accuracy Comparison of FiST_CNN, CNN and SIFT_CNN	8
<b>Figure 2.5</b>	prototype of proposed solution	9
<b>Figure 2.6</b>	‘Hello’ being recognized as text	9
<b>Figure 2.7</b>	‘Thank you’ being recognized	10
<b>Figure 2.8</b>	‘Sorry’ being recognized	10
<b>Figure 2.9</b>	Architecture of Arabic Sign Language	14
<b>Figure 2.10</b>	Detection Rate of the system.	15
<b>Figure 2.11</b>	LSTM model	17
<b>Figure 2.12</b>	26 alphabets recognized using LSTM	17
<b>Figure 2.13</b>	Accuracy of various neural network models implemented	18
<b>Figure 4.1</b>	System Flow	22
<b>Figure 4.2</b>	Mediapipe hand model	23
<b>Figure 4.3</b>	Mathematical formula of Adam Optimizer	26
<b>Figure 4.4</b>	Bias correction	27
<b>Figure 5.1</b>	Dataset	29
<b>Figure 5.2</b>	List of Libraries used in the proposed system	30
<b>Figure 5.3</b>	Class Distribution for all the words	30
<b>Figure 5.4</b>	Building the model	31
<b>Figure 5.5</b>	Hyperband Tuner	31



<b>Figure 5.6</b>	Early Stopping	32
<b>Figure 5.7</b>	Training the data	32
<b>Figure 5.8</b>	Hyperparameter tuning best values	32
<b>Figure 5.9</b>	Training the model using tuner.	33
<b>Figure 5.10</b>	Language translation and text to speech conversion	33
<b>Figure 6.1</b>	Attained overall accuracy of 99.11%	35
<b>Figure 6.2</b>	Evaluation	35
<b>Figure 6.3</b>	Result Analysis	36
<b>Figure 6.4</b>	Accuracies achieved by different techniques	37
<b>Figure 6.5</b>	Number of words / alphabets recognized by different techniques	37
<b>Figure 6.6</b>	Overall Visual Result Analysis	38
<b>Figure 6.7</b>	Web App Interface	39
<b>Figure 6.8</b>	Language selection panel	39
<b>Figure 6.9</b>	Camera turns on	40
<b>Figure 6.10</b>	Output	40
<b>Figure 10.1</b>	Sigmoid Activation Function	72
<b>Figure 10.2</b>	Sigmoid activation function formula	72
<b>Figure 10.3</b>	Tanh activation function	73
<b>Figure 10.4</b>	Tanh activation function formula	73
<b>Figure 10.5</b>	ReLU activation function	74
<b>Figure 10.6</b>	SGD with momentum formula	74
<b>Figure 10.7</b>	Formula of NAG optimizer	75
<b>Figure 10.8</b>	formula of Adagrad optimizer	76

## LIST OF ABBREVIATIONS

- **ASL**: American Sign Language
- **SGD**: Stochastic Gradient Descent
- **CNN**: Convolutional Neural Network
- **API**: Application Programming Interface
- **ISL**: Indian Sign Language
- **SSD**: Single Shot Detector
- **GUI**: Graphical User Interface
- **LSTM**: Long Short-Term Memory
- **ReLU**: Rectified Linear Activation Unit
- **OpenCV**: Open-Source Computer Vision Library
- **OS**: Operating System
- **NumPy**: Numerical Python
- **GTTS**: Google Text to Speech
- **CPU**: Central Processing Unit
- **GPU**: Graphics Processing Unit
- **PC**: Personal Computer
- **CSV**: Comma Separated Values
- **ADAM**: Adaptive Moment Estimation
- **NAG**: Nesterov accelerated gradient
- **Adagrad**: Adaptive gradient

# **1. INTRODUCTION**

## **1.1 OVERVIEW**

Words can be a very powerful tool and have a huge impact on our lives. Effective communication helps us in expressing our thoughts, ideas and feelings. It helps us build lasting relationships in both personal and professional life. Unfortunately, not many people have the privilege to easily communicate with each other using speech. There are those who are unable to speak. They cannot use their voice to articulate words. Thus, sign language was created to enable such people to communicate. Sign language is a medium of communication using visual hand gestures called signs.

There are more than 70 million people in the world, who use sign language. But there is not a single universal sign language that people use to communicate. There are more than 300 sign languages that are used across the world. They can differ from country to country. Even within the same country, sign languages can have subtle differences. Sign languages are fully fledged natural languages, structurally distinct from the spoken languages. There is also an international sign language, which is used by deaf people in international meetings and informally when travelling and socializing. It is considered a pidgin form of sign language that is not as complex as natural sign languages and has a limited lexicon.

The benefits of learning sign language at an early age are numerous. American Sign Language (ASL) is one of the most widely used languages in the United States, and the most studied sign language at universities. At least 35 states have recognized ASL as a modern language for public schools, and hundreds of colleges and universities in the United States are offering ASL classes. ASL is primarily used by American and Canadians who are either deaf or hard of hearing. There are approximately 250,000 – 500,000 ASL users in the United States and Canada, most of whom use ASL as their primary language. In addition, ASL is used by: hearing children of deaf parents, hearing siblings and relatives of the deaf, hearing adults who are becoming deaf and are learning ASL from other deaf individuals, and a growing population of hearing, second-language students learning ASL in elementary, secondary, and post-secondary classrooms. Being proficient in ASL allows you to communicate with a wide range of

hearing, hard of hearing, and deaf individuals, including students in mainstream and deaf school or university programs and deaf or hard of hearing residents and business people in your community. In addition, ASL improves the quality of family communication for hearing people with deaf or hard of hearing family members.

The United Nations Convention on the Rights of Persons with Disabilities calls on states to accept, facilitate, and promote the use of sign languages with the goal to ensure that people with disabilities can enjoy their rights on an equal basis with others. But Human Rights Watch research around the world finds deaf people often struggle to access basic services. In India, Iran, and Russia, lack of sign language interpreters and information in accessible formats hampers access to public services and courts. In these and other countries, communication barriers also impede access to health care for deaf people. One of the solutions to this problem is to create an application that would interpret sign language and help the speech impaired people in communicating with the people who cannot understand sign language.

## **1.2 APPLICATIONS**

- Facilitates communication between mute people, who use sign language to communicate, and non-mute members of society, who do not understand sign language by themselves, eventually leading to stronger bonds and better understanding between people.
- Effectively recognizes signs on the word level and provides realistic communication.

## **1.3 MOTIVATION**

Our lives can be significantly impacted by the words we use. Our ability to express our thoughts, ideas, and feelings is aided by effective communication. In both our personal and professional lives, it aids in the development of enduring relationships. Unfortunately, not many people have the opportunity to speak to one another easily. There are those who are speechless. They are unable to articulate words using their

voice. Thus, sign language was developed to allow for communication between such individuals. The visual hand gestures used in sign language are referred to as signs.

#### **1.4 PROBLEM STATEMENT**

To develop a web application that will bridge the communication gap between speech-impaired people and others by recognizing sign language and giving the output in the form of text and speech.

#### **1.5 OBJECTIVES**

Since most of the existing models for helping the speech impaired people are glove based, which can be quite uncomfortable, we wanted to provide an easier and more comfortable way to help them. A web application is very convenient for anyone to use.

Many existing models are also only able to detect individual letters and not words or phrases. In reality, no one signs words by spelling them out. We want to create an application that would provide more realistic communication.

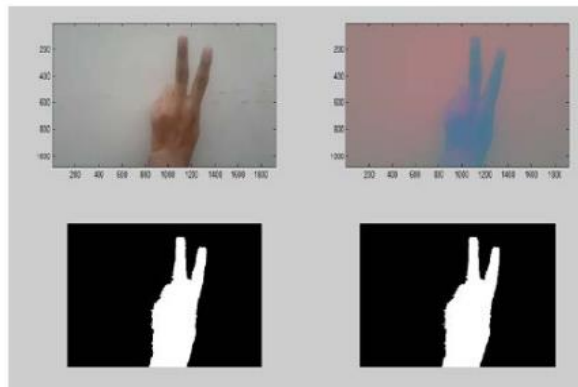
#### **1.6 ORGANIZATION OF THE REPORT**

The report first discusses the literature review done before implementing this project. It briefly covers the methodology used by the author, and discusses its merits and demerits. 13 papers have been included in the literature survey. Then, the methodology we used to implement sign language recognition is described. It is followed by the implementation details which includes code snippets as well. Finally, proof of testing and the results obtained, comparative analysis between our project and others' are showcased. Last but not the least, the conclusion and future scope is discussed. The visible output has also been added.

## 2. LITERATURE SURVEY

### 2.1 VISION BASED APPROACH TO SIGN LANGUAGE RECOGNITION

The authors of the paper have proposed an algorithm that uses YCbCr colour space, COG and template matching for segmenting the hand, detecting the fingers and then identifying the hand gesture from real imagery to help the deaf and mute people. According to the methodology proposed, the user will have to upload their video of performing a sign. The video is then divided into frames and each frame is used as an image. To segment the hand from the background, the RCB image is converted to YCbCr color space. YCbCr is a family of color spaces that allow brightness to be separated from color. RGB is a system to represent color used in computer systems. Different combinations of red, green and blue can give any color on the visible spectrum. The centroid of gravity (COG) is then calculated. The farthest distance from the centroid to the tip of the longest active finger in the particular gesture is taken as the radius to draw a circle. The circle consists of the gesture. Thus, COG was used to segment the hand. The unknown gesture is then compared with the preset template models of individual gestures to identify the gesture as shown in Fig 2.1. The major advantage was that their algorithm was able to detect the hand gesture from any background (robust to background) and was able to identify the hands of different skin tones, thereby giving a better result. But at the same time, it was computationally expensive and could not detect sign language in real time, since the user would have to provide a video of them.

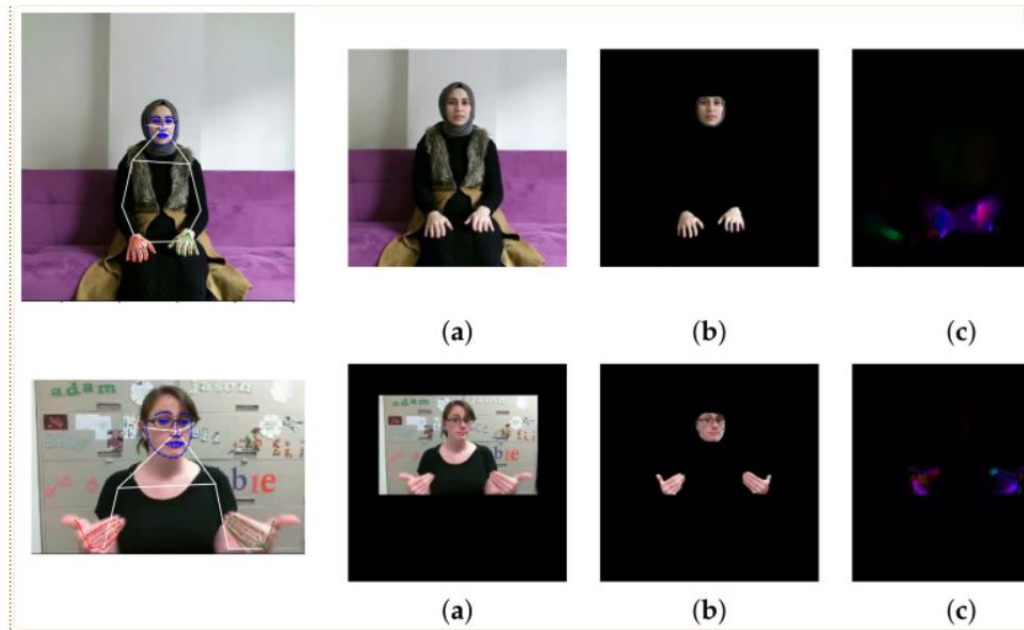


*Fig.2.1: Image conversion*

## **2.2 ONE MODEL IS NOT ENOUGH: ENSEMBLES FOR ISOLATED SIGN LANGUAGE RECOGNITION**

The authors work on two appearance-based approaches- I3D and TimeSformer, and one posed base approach- SPOTER on two publicly available datasets- AUTSL and WLASL300 to detect isolated signs. By using the CMA-ES optimization technique, they have improved the accuracy of using the WLASL300 dataset. They have also proposed an ensemble technique based on a Transformer model called Neural Ensembler. I3D (a family of convolutional neural network models for video classification) was implemented with 3 inputs: Crop and Resize, Masked, and OptFlow. 16 frames from each video were taken as the input as shown in Fig 2.2. Number of training epochs were 80 and 120 for AUTSL and WLASL300 respectively. For the parameter update, the SGD optimizer with starting learning rate  $lr=0.01$ , cosine learning rate schedule and batch size  $BS=10$  was used. All I3D models were pre trained using the Kinetics400 dataset. TimeSformer is a Transformer-based model which utilizes spatiotemporal features from sequences of frames. This was implemented with the same inputs as I3D. The authors followed the original Github implementation from facebooksearch. They made changes according to their hardware and software requirements. SPOTER (sign pose-based transformer) works on top of body-pose sequences, and was implemented using Pytorch by the authors. They followed a training set up by Boháček M., Hružík M. Sign Pose-Based Transformer for Word-Level Sign Language Recognition. They used the SGD optimizer with a learning rate of 0.001. Momentum and decay were set to 0. Model's initial weights were initialized from a uniform distribution of  $[0,1)$ . They trained SPOTER for 100 epochs on WLASL300. On the AUTSL dataset, they trained SPOTER with OpenPose and MMPose for 12 and 35 epochs, respectively, as each model variant was trained until it converged on the training split. CMA-ES (covariance matrix adaptation evolution strategy) was implemented to solve the problem of model weight optimization and since its original implementation was freely available. A neural ensemble was utilized by the authors that took the outputs from individual models and gave a final decision. It was implemented using PyTorch. The individual models implemented gave suboptimal results. Appearance based approaches were able to distinguish two similar signs, but it required a larger training dataset. Pose based approach required a smaller training dataset, but

was unable to distinguish between similar signs. By using ensemble techniques, better accuracies were achieved: 96.37 % in AUTSL dataset and 73.87 % in WLASL300 dataset. The neural ensemble method proposed by the authors has high potential for future research. But isolated sign language recognition still poses a problem in an uncontrolled environment.



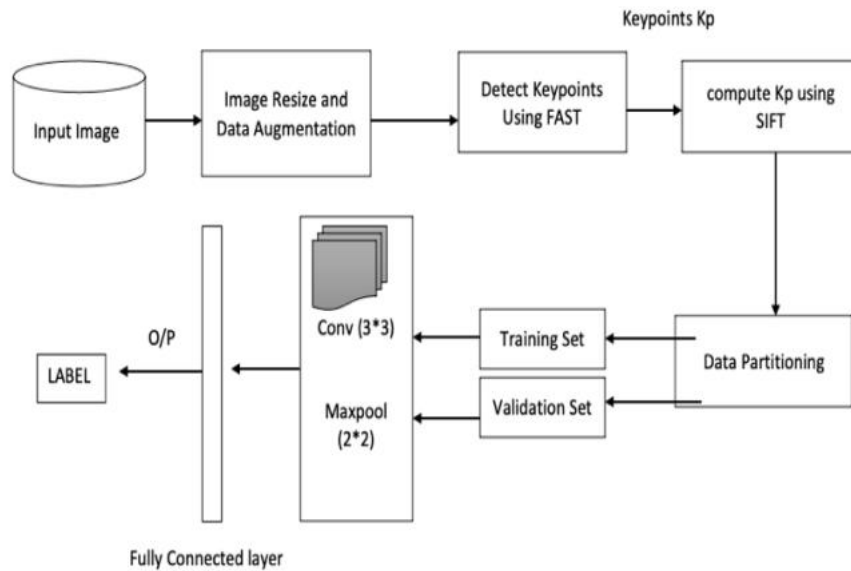
*Fig. 2.2: Different data modalities given as input*

### 2.3 HYBRID FIST-CNN APPROACH FOR FEATURE EXTRACTION FOR VISION-BASED INDIAN SIGN LANGUAGE RECOGNITION

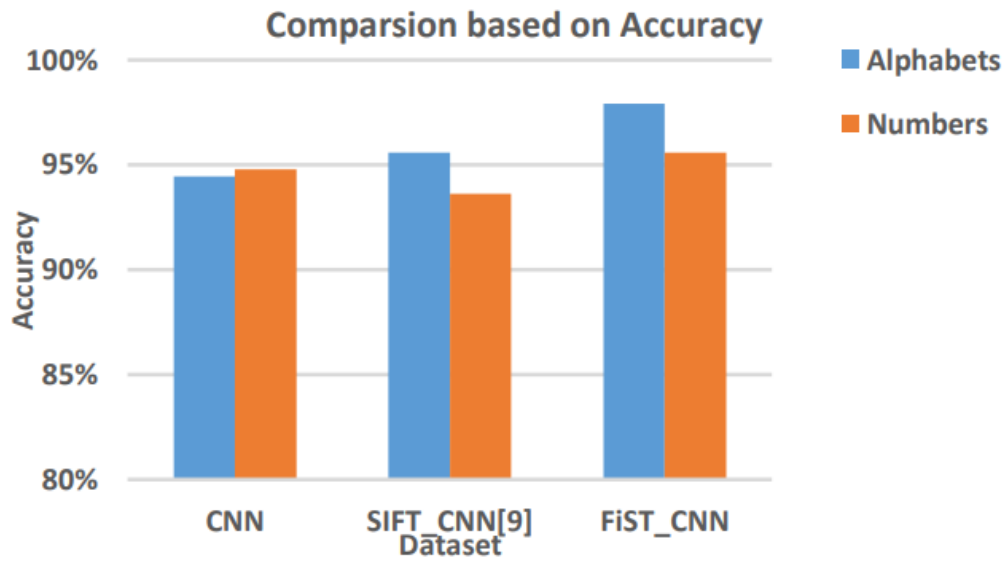
Techniques such as Features from Fast Accelerated Segment Test (FAST), Scale-Invariant Feature Transformation (SIFT), and Convolution Neural Networks (CNN) were used by the authors for effective feature extraction, necessary for automatic recognition of gestures. FAST with SIFT are used to detect and compute features, respectively from images. CNN was used for classification with the hybridization of FAST-SIFT features. Feature extraction is the process of extracting relevant information from the input hand gesture and converting into a compact vector form. FAST is a corner detection method used to extract feature points which is then



used for tracking and mapping objects in many computer vision tasks. SIFT is a computer vision algorithm to detect, describe, and match local features in images, invariant to image scale and rotation. Authors have proposed three major phases for sign recognition: data preprocessing, feature extraction, and training and testing of CNN. In the first phase, the stored static single-handed images are resized, and then data augmentation is done on those resized images. Next, key points are localized by FAST techniques. The value of these localized key points was computed using SIFT in the third phase. These values were passed to CNN then for training and then classification. The system was implemented and tested using the python-based library Keras. This is seen in fig 2.3. The results of the proposed techniques were tested on 34 gestures of Indian Sign Language (24 alphabets set and 10 digit sets), and on two publicly available datasets on Jochen Trisech Dataset (JTD) and NUS-II dataset. The performance of their proposed FiST\_CNN was compared with other models like CNN and SIFT\_CNN. FiST\_CNN gave a better performance, in terms of accuracy and computation time. The FiST\_CNN achieved an accuracy of 97.89%, 95.68%, 94.90% and 95.87% for ISL-alphabets, MNIST, JTD and NUS-II respectively. The disadvantage is that the model cannot detect word level signs. The comparison can be seen in fig 2.4.



**Fig. 2.3: Architecture of FiST\_CNN for ISL**

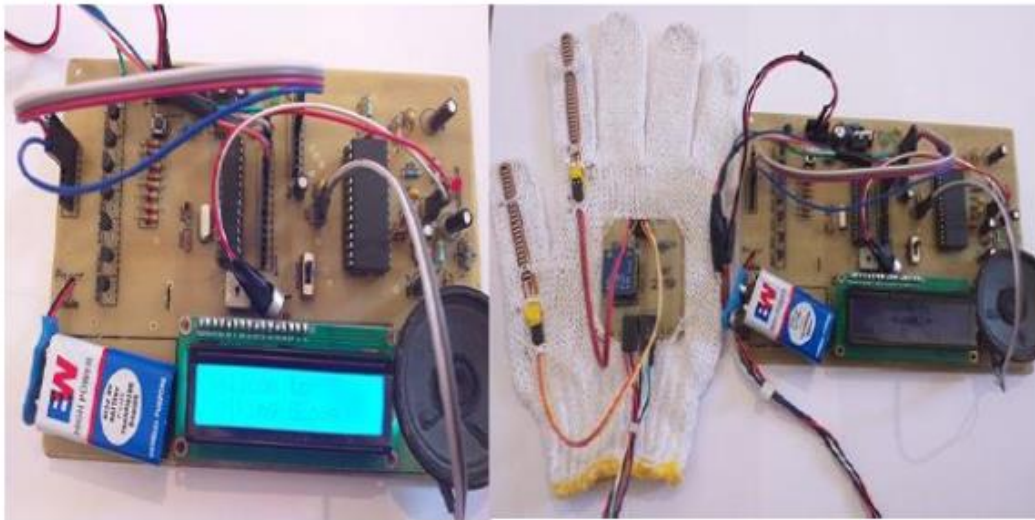


*Fig. 2.4: Accuracy Comparison of FiST\_CNN, CNN and SIFT\_CNN*

## 2.4 IMPLEMENTATION SMART GLOVES FOR DEAF AND DUMB DISABLED

It describes the working of a smart glove that is able to convert sign language to speech, said out loud with a speaker, and text which is displayed on a LCD screen. The different equipment required to implement this are: Data Glove, Arduino UNO, Flex sensors, VoiceRecord(APR33A3), LCD 16\*2, Speaker, Accelerometer (ADXL 335). The methodology proposed is as follows: On each finger of the glove, there is a flex sensor. The flex sensor is also called a bend sensor and it is placed on a surface. It measures how much the surface bends or deflects. The resistance of the flex sensor will change depending on how much the fingers are bent. Each bend of the finger is quantized into 10 levels. Every bend of the finger must belong to some level for it to be recognized. The binary data from the flex sensors will be sent to the microcontroller. A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. It includes a processor, memory, and input/output peripherals on a single chip. The accelerometer is used for measuring the tilt of the hand. This

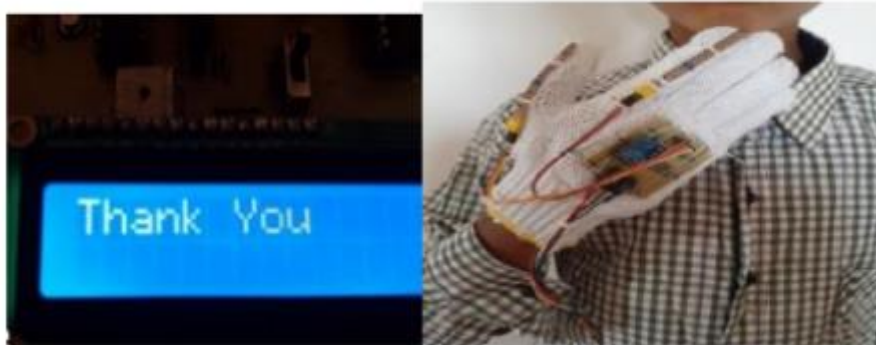
binary data is also sent to the microcontroller. The microcontroller will check each signal and compare it with previously stored values. Based on the comparison, the microcontroller identifies what hand gesture is being made, and thus what message is trying to be communicated. The message is sent to the LCD to display it and to the speaker to say it out loud in English as shown in fig 2.5. When the device built was tested, it worked well. But it could only identify 4 gestures. 8 The system could be enhanced by using 2 sensors, instead of just one. Different languages could also be used. A jacket could be used instead of a glove, to make it more reliable. The results can be seen in fig 2.6, 2.7, and 2.8.



*Fig. 2.5: prototype of proposed solution*



*Fig. 2.6: 'Hello' being recognized as text*



*Fig. 2.7: 'Thank you' being recognized*



*Fig. 2.8: 'Sorry' being recognized*

## **2.5 SIGN LANGUAGE INTERPRETER USING IMAGE PROCESSING AND MACHINE LEARNING**

A process is outlined where a still hand image frame was captured using a webcam. These frames were processed to get enhanced features. Then feature extraction and classification algorithms are used to translate the sign language into English text. This translation is converted to speech using text to speech API. The different steps described to detect signs are: first, the hand gestures are captured by video, they are then divided into frames. Image preprocessing is then done and then feature extraction is performed using HOG (Histogram of Oriented Gradients). SVM (Support vector machines) is used for classification and recognition. The text is given as the output. The google api is used

for converting text to speech to give audio as the output as well. The authors have created their own dataset of 26 English alphabets of ISL. Each gesture was performed by two different individuals 10 under different lighting conditions. 4800 images were used for training and 1200 images were used for testing. The overall accuracy of the model proposed was 88%. But this application is not portable. It could be made into a mobile application for convenience. Also, it cannot identify word level signs.

## **2.6 REAL TIME SIGN LANGUAGE DETECTION**

They used a user-defined dataset with around 2000 photos, approximately 400 for each of classes, to develop and implement a sign language recognition model based on a CNN. To train the dataset, they used a Pre-Trained SSD Mobile net V2 architecture. In this paper the three steps were recognised design the model- Obtaining footage of the user signing and take them as input, classify each frame in the video to a sign, and reconstruct and display the most likely Sign from classification scores to provide that as output. They made use of A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning system that can take an input image and assign importance (learnable weights and biases) to various elements and objects in the image, as well as distinguish between them. When compared to other classification systems, a ConvNet requires far less preprocessing. ConvNets can pick up these filters and properties with the right training, but basic approaches require filter engineering by hand. A dataset of 2000 images with 400 image divided among 5 classes was used to train this model. The proposed system gave accurate results der controlled light and intensity. As a result, by extending the dataset, the model may be simply extended on a wide scale. However, this approach has significant drawbacks, such as environmental conditions such as low light intensity and unmanaged backdrop that reduced detection accuracy.

## **2.7 ARABIC SIGN LANGUAGE RECOGNITION THROUGH DEEP NEURAL NETWORKS FINE-TUNING**

The authors used transfer learning and deep CNN fine tuning to recognise 32 Arabic sign language hand movements. The networks were fed regular 2D images of various

Arabic Sign Language data. The article employs transfer learning and fine-tuning deep convolutional neural networks to improve the precision of detecting 32 hand gestures from Arabic sign language (CNN). The suggested approach involves building models that are similar to the VGG16 and ResNet152 structures, loading pre-trained model weights into each network's layers, and adding our own soft-max classification layer as the last layer 11 following the last completely connected layer. An image is sent into a convolutional neural network (CNN), which then processes it through a series of layers that include convolutions, pooling, and other fully connected layers to produce an output that represents only one of the picture's potential classes. This paper introduces two well-known CNN architectures that were used in our experiment on the Arabic Sign Language dataset. The final model achieved a validation accuracy of 99.4% for the VGG 16 and 99.6% for the Resnet152. But the points of focus were that the model identified only static gestures and the dataset was small with only 32 hand gestures.

## **2.8 DEVELOPMENT OF SMART GLOVE FOR DEAF-MUTE PEOPLE**

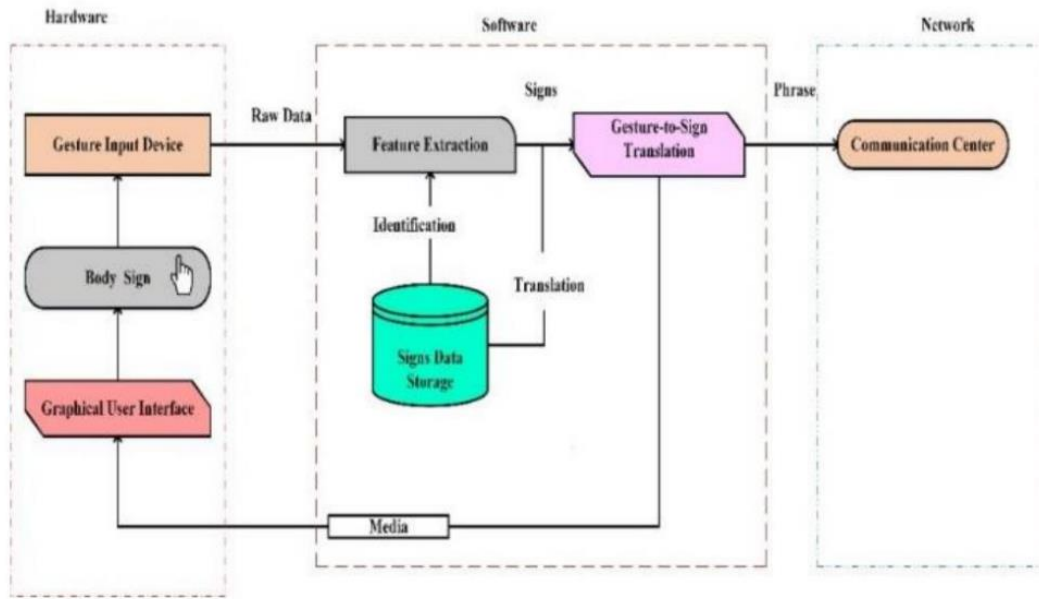
This paper uses flex sensor, arduino, and accelerometer as hardware components and Aurdino Software to build smart gloves to deaf and mute people. The process employed begins with identifying the issue, examining its primary cause, and outlining the series of connected acts that have an impact either directly or indirectly. There is hardware and software in this system. The hardware component consists of an accelerometer, an Arduino, and a flex sensor. The programming of the Arduino via gestures is part of the software. The three components of this system are the input from gestures, the processing of sensor data, and the output. Signals are transmitted to the microcontroller using the accelerometer and flex sensors. The microprocessor processes the accelerometer and flex sensor signals by converting them to a digital output. The processor's memory contains predetermined messages that correspond to the gestures. The text corresponding to the matching values is given as the output which is displayed on the laptop screen. The prototype had a good response to movement but had the following disadvantages: communication was possible only in english as a result of which it cannot be used by non-English speakers and it was not portable: cannot carry laptop everywhere to communicate.

## **2.9 FINGER SPELLING RECOGNITION FOR NEPALI SIGN LANGUAGE**

The paper for Nepali Hand Gesture Recognition uses Blob analysis to extract the hand gesture from the image considering that the largest blob is the hand. Then, the blob was resized, sampled and length encoding was done. The suggested method makes use of a skin colour model to recognise the hand in the image, and additional pre-processing is carried out to eliminate unnecessary noise and regions. Since the hand is the biggest blob in the 12 image, blob analysis is used to extract the hand gesture from the picture. Then, in order to remove the size variance limitation, the blob is shrunk to a standard size. Grid lines are crossed to sample the boundary line of the hand motion. The junction point between the grid line and the boundary is then extracted. The hand gesture's border is represented using the Freeman chain coding. In order to reduce the duration of encoding for chain code run length is completed. Its shape number is discovered by locating the first variation in the chain code. Each motion may be uniquely identified using a number. By converting the gesture to a shape number, the object descriptor used to convey boundary information made categorization simple and straightforward. After analysing a few NSL movements, this method for gesture detection is determined to be straightforward and satisfactory. It still has to be tested on additional gestures, though.

## **2.10 GESTURE BASED ARABIC SIGN LANGUAGE RECOGNITION FOR IMPAIRED PEOPLE BASED ON CONVOLUTION NEURAL NETWORK**

The Convolution Neural Network has been incorporated in the proposed system, which is based on machine learning. It can recognise 30 hand sign letters using hand movements collected by DG5-Z hand gloves with wearable sensors. Based on the technique described, the Architecture for Arabic Sign Language Communication System is a proposed system for translating various languages at a minimal cost. A sign language-based communications network that is both precise and cost-effective.

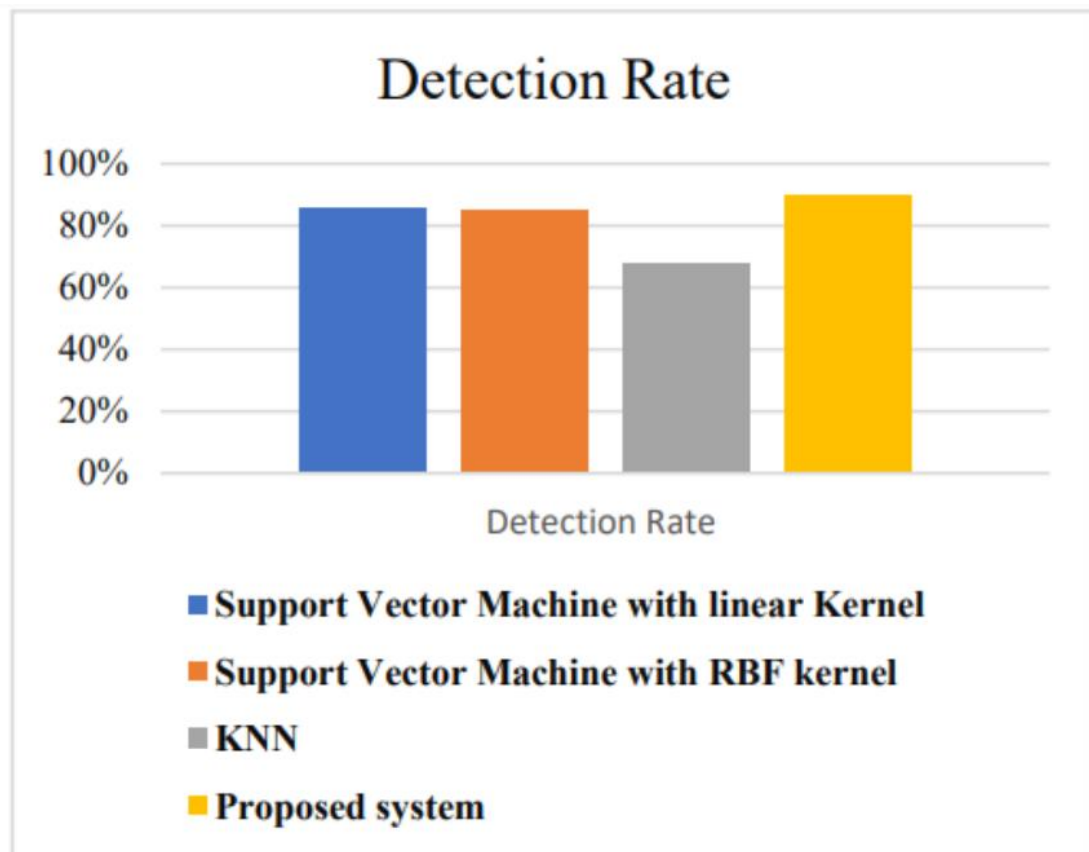


***Fig. 2.9: Architecture of Arabic Sign Language***

The hardware (components of the system supply the transmitter and receiver channels through which the user controls the system.), software (The program's software is in charge of extracting features and movement translation, as well as providing a simple Graphical User Interface (GUI). The Sign Identification Center converts raw data into a set of predefined signs based on the existing and established lexicon. To get information about signs, the Sign language datastore is employed, as is the network. The three pieces that comprise the system architecture (Fig 2.9) are the Signs media player, which sends media to the communication centre, which then transmits this via the system to its intended destination. Computer vision architecture was developed to distinguish fingers and hands. It isolates and monitors them inside its field of view. The architecture gathers movement monitoring data in the form of pixels. The monitored data frames hold the measured locations, sign orientations, and other information about every object identified in the current frames. To depict the identifiable fingers and hands, each pixel is unique. Initially, a deep Convolutional network is built to extract features from data collected by sensing devices. These sensors can distinguish the 30 hand sign letters used in Arabic sign language. The dataset's hand movements were recorded using DG5-V hand gloves equipped with wearable sensors. The CNN



approach is used for categorisation. The advantage is that the proposed algorithm the overall success of the system was 85-89%, and they didn't use any complicated technology. (Fig 2.10).



*Fig. 2.10: Detection Rate of the system.*

This system used several classification types and the outputs of each, and Arabic Sign Language was used to analyse the results. In terms of accuracy, the results of the research were produced by CNN classifier. Using Convolution Neural Network Systems, gesture-based Arabic Sign Language Recognition is implemented for the needs of visually impaired people. Moreover, future study projects could increase the size of data collection further.

## **2.11 SIGN LANGUAGE RECOGNITION USING MEDIAPIPE**

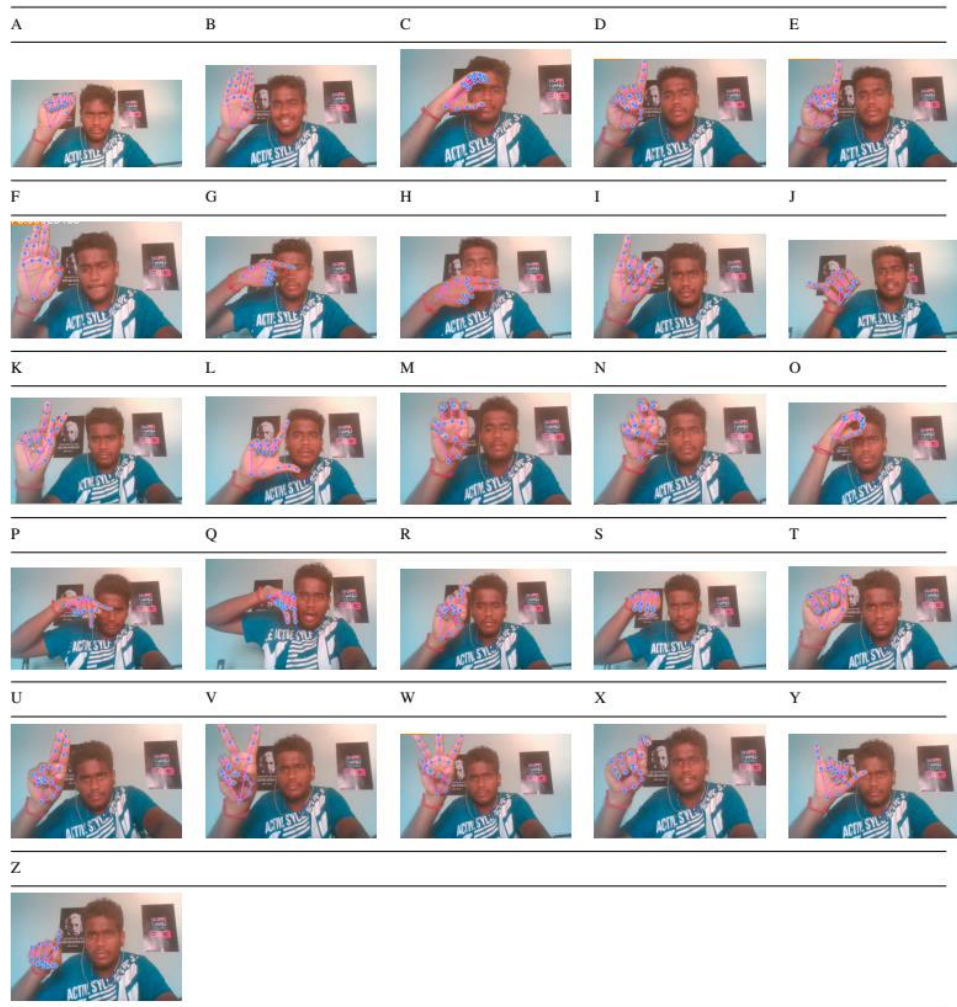
The objective of this paper was to recognize American Sign Language (ASL) alphabets using the mediapipe hands model and the K nearest neighbour (KNN) machine learning model. It has an average accuracy of 86 to 91%. The advantage we have over this paper is that we use a deep learning neural network, instead of a machine learning model to predict the signs and thus we have a higher accuracy (99%). We are also able to recognize hands on the word level to provide a much more realistic communication.

## **2.12 AMERICAN SIGN LANGUAGE RECOGNITION FOR ALPHABETS USING MEDIAPIPE AND LSTM**

The authors of this paper used the mediapipe hands model and a recurrent neural network LSTM (Long Short Term Memory) to recognize 26 alphabets of ASL. In their proposed system they first extract the key points from the hand using the mediapipe hands model, create the dataset, use this dataset to create the LSTM model, store the model, use this stored model to recognize the hand gestures and detect the alphabet being signed. The dataset was created by four different people: two males, one female and one kid. It contains 93,600 values. The algorithm was made to run on 2000 epochs. The neural network consists of 6 layers: 3 LSTM layers and 3 dense layers. This can be seen in fig 2.11. The RELU activation function was used along with the Adam optimizer. An accuracy of 99% was achieved. The recognized alphabets can be seen in fig 2.12. The disadvantage of this project is that it is only able to understand alphabets and not words, and thus does not provide a realistic communication.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 30, 64)	32768
lstm_1 (LSTM)	(None, 30, 128)	98816
lstm_2 (LSTM)	(None, 64)	49408
dense (Dense)	(None, 64)	4160
dense_1 (Dense)	(None, 32)	2080
dense_2 (Dense)	(None, 26)	858
Total params: 188,090		
Trainable params: 188,090		
Non-trainable params: 0		

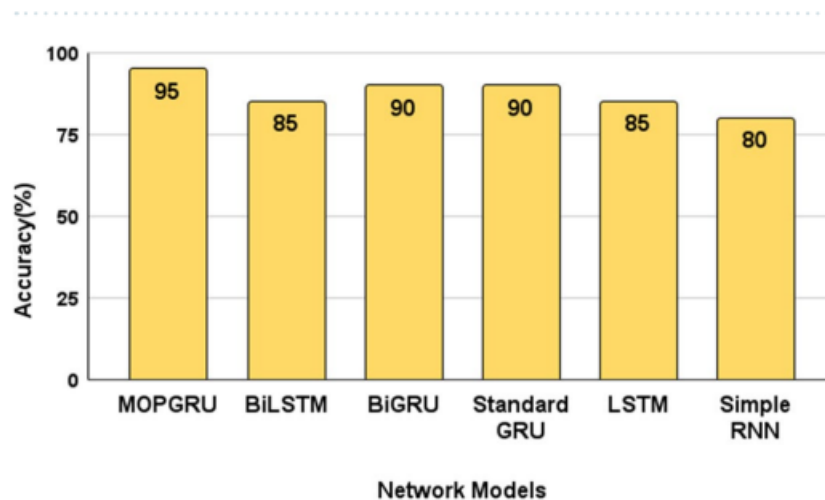
*Fig. 2.11: LSTM model*



*Fig 2.12: 26 alphabets recognized using LSTM*

### 2.13 AN INTEGRATED MEDIAPIPE-OPTIMIZED GRU MODEL FOR INDIAN SIGN LANGUAGE RECOGNITION

This paper makes use of an integrated mediapipe optimized gated recurrent unit (MOPGRU) model to recognise Indian Sign Language (ISL). The mediapipe holistic model is used to incorporate not just the hands as the input features but also the arms and facial expressions, to help identify the signs. The exponential linear unit (ELU) activation function is used for avoiding vanishing and exploding gradient problems, and the softsign activation function is used in the output layer. There are three stages in their proposed methodology. In the first stage, the mediapipe framework extracts the hands, face, and landmark features which are then pre-processed. In stage 2, the features extracted are saved in a file. The null entries are removed the file and then the records are labelled. In stage 3, the cleaned and labelled data is trained and classified by the MOPGRU model for recognizing ISL. There are 9 layers in the neural network: 3 GRU layers, a batch normalization layer, 2 dropout layers, and 3 dense layers. It achieved an accuracy of 95% as shown in fig 2.13. The disadvantage of this paper is that it has a limited vocabulary. It is only able to recognize 13 words. It also did not make use of hyperparameter tuning, which we used in our approach, and thus we were able to achieve a higher accuracy than them with a larger vocabulary.



*Fig 2.13: Accuracy of various network models implemented*

### **3. SYSTEM REQUIREMENT SPECIFICATION**

#### **3.1 FUNCTIONAL REQUIREMENTS**

- Application should use the webcam (in case of a desktop) or camera (in case of a mobile phone) in order to capture the video of the person signing language.
- Application should be able to extract appropriate images from the video.
- Mediapipe should be able to capture the appropriate hand landmarks when word is signed in front of camera.
- The captured image should detect the sign appropriately.
- The user must have the provision to choose their own default language.
- The identified word should be translated into the default language of the user.
- The output should be in the native language of the user either in the voice format.

#### **3.2 NON-FUNCTIONAL REQUIREMENTS**

- Accuracy: Since we will give the priority to the accuracy of the software, the performance of the language detector and translator will be based on its accuracy.
- Failure handling: System components may fail independently of others. Therefore, system components must be built so they can handle failure of other components they depend on.
- Openness: The system should be extensible to guarantee that it is useful for a reasonable period.
- Security: Sensitive information should be kept in safe. User profile information will be used, so data security is one of the most important concerns of the system.
- Reliability: The system should have accurate results and fast responses to user's changing emotions.

### 3.3 SOFTWARE REQUIREMENTS

- Python: Since it's relatively easy to learn and implement AI/ML programs on python, we will be adopting it for implementing the functionality. It's a programming language that supports machine learning and deep learning through its libraries, and we can easily create a web interface for the project using python.
- OpenCV: It is used in this project to open the camera, in front of which the user can sign the word, and on which they can see the predicted word displayed as text.
- Os: this module was used to save the checkpoints during training so that they can be accessed and used for resuming training at a later point of time.
- Pandas: It was used to load the dataset and perform initial data exploration.
- Mediapipe: It was used to identify the hand landmarks when the word is signed in front of the camera. The dataset is numerical in nature and comprises the position of the hand landmarks captured by mediapipe.
- Matplotlib, Seaborn: These were used to visualize the class distribution in the dataset and the confusion matrix for accuracy.
- TensorFlow and Keras: Used to create the sequential neural network model that performs multiclass classification and ultimately predicts the word for the sign being performed in front of the camera.
- Keras Tuner: It was used to perform hyperparameter tuning so that we get the best values for the hyperparameters to get the most optimal solution.
- Jupyter notebook: We are using Jupyter notebook with CPU for building and training the neural network model.
- Visual Studio 2022: We used this to write the python program that includes flask. The program opens the camera, gets the input hand landmarks, provides it to the trained model, displays the predicted text as the output, performs language translation and converts text to speech. The html and css code for the front end of the website was also written using Visual Studio 2022.
- Googletrans: This is used to execute the language translation feature of our project.

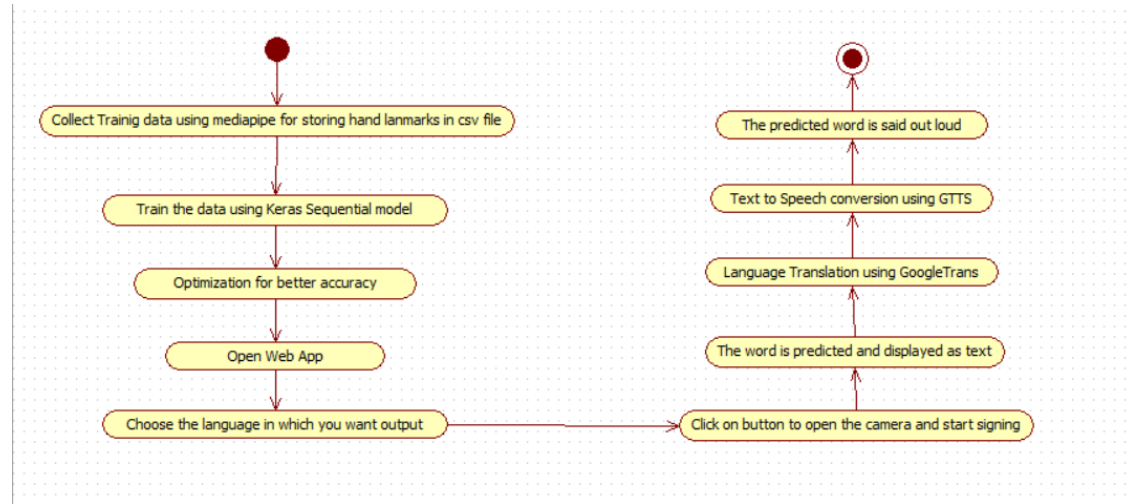
- Gtts: it stands for Google text to speech. It interfaces with Google Translate's text-to-speech API. It converts the text to speech and stores it in an mp3 file, which can be automatically played. This was used for the text to speech feature of the project.
- Flask: It is a web framework written in python. It provides us with tools and libraries which allow us to create web applications.

### **3.4 HARDWARE REQUIREMENTS**

When it comes to hardware needed, it is very simple. To create this application, we needed a computer with CPU or GPU. This application uses a lot of images and videos to properly train data so to store and process them GPU is used. It is easy to create this by using Windows PC and Windows 7 as these operating systems are stable and reliable. Since the application can run on a mobile phone too, a mobile with suitable OS can also be used. Another consideration for hardware is some sort of backup hard drive or removable USB memory stick(s) as it is important that we back up all of our website data.

## 4. PROPOSED METHODOLOGY

To facilitate a more realistic communication between the mute and non-mute people, the sign language words and phrases which are being signed are recognized and are converted to speech by our model. The system design around which our model was built is shown in Fig. 4.1.



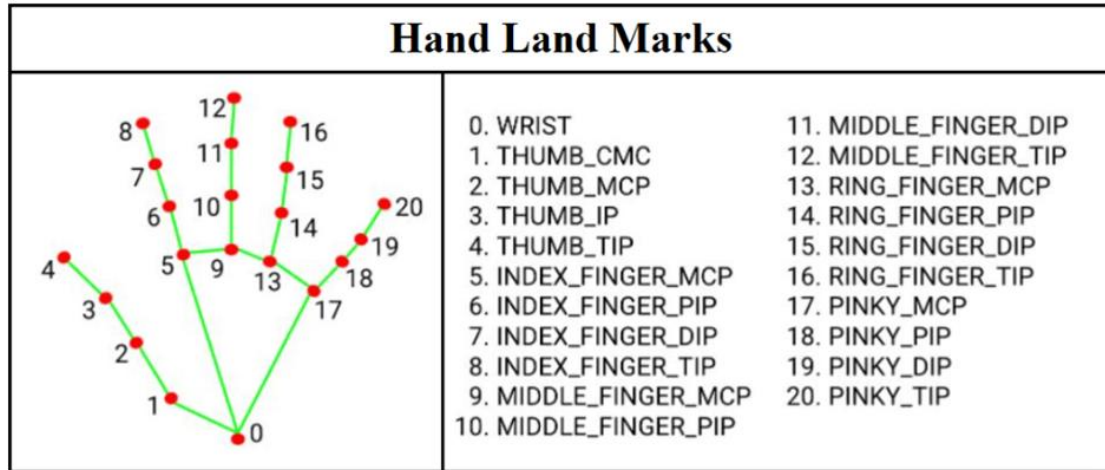
*Fig. 4.1: System Flow*

### 4.1 DATASET

The two main technologies that we use to create our dataset are Mediapipe and Opencv. OpenCV was used for opening the web camera so that we can start signing. Mediapipe was used to capture the hand landmarks when the word is signed in front of the camera. There are 21 points being identified for one hand which are shown in Fig.4.2. So, for 2 hands, 42 landmarks will be identified. These 42 landmarks will make up 42 columns in the csv file. An additional column will be for the unique ID of the word. For each gesture, multiple rows of data will be collected so that there is enough data associated with a word, for the model to learn about it. Thus, our dataset consists of 43 columns. The first column is the ID of the word, and the rest 42 columns are the data points regarding the hand landmarks. As we sign, the hand landmarks get stored in the csv file



which is then used for training. There are approximately 31000 rows in our dataset and 30 unique classes. Our dataset consists of a larger vocabulary when compared to other projects and it is able to recognize signs at the word level and not just alphabets. This provides a more realistic communication, since nobody spells out the word they want to communicate, using alphabets.



*Fig. 4.2: Mediapipe hand model*

## 4.2 KERAS SEQUENTIAL NEURAL NETWORK MODEL

We used a neural network to perform the classification on the dataset that we created. The model we are using is the sequential model of Keras. Our input data is the 42 hand landmarks. Thus, the input layer consists of 42 neurons. The output layer consists of 30 neurons since we have 30 unique classes. We used Keras tuner to determine the appropriate number of hidden layers and the number of neurons in each layer.

## 4.3 TUNING

Using a tuner to figure out the number of hidden layers, the number of neurons present in each hidden layer, and learning rate is much faster and efficient when compared to manually changing the values, and retraining the model every time. Using

hyperparameter tuning, we can achieve much higher accuracies. Keras tuner offers 4 tuners to perform hyperparameter tuning: RandomSearch, Hyperband, BayesianOptimization, and Sklearn. We chose Hyperband for our model.

The Hyperband tuning algorithm uses adaptive resource allocation and early-stopping to quickly converge on a high-performing model. This is done using a sports championship style bracket. The algorithm trains many models for a few epochs and carries forward only the top-performing half of models to the next round. Hyperband determines the number of models to train in a bracket by computing  $1 + \log_{\text{factor}}(\text{max\_epochs})$  and rounding it up to the nearest integer.

We did not make use of RandomSearch tuner because it creates multiple models by choosing a random combination of hyperparameters. Thus, the final model created may not contain the optimal set of hyperparameters which might give us a low accuracy. The BayesianOptimizer initially creates some models by choosing a random combination of hyperparameter values. The subsequent models created or hyperparameter values chosen will depend on the previous ones. This can take a very long time to train and thus we did not choose this. We did not make use of the sklearn tuner since it is used for sklearn models, and ours is a keras model.

The objective of the tuner is to increase the accuracy. The maximum number of epochs to train the model is 100. Factor is the number of models for each bracket. Its default value is 3.

We gave the range for the number of hidden layers to be from 2-5. We gave the minimum number to be 2 because having only 1 hidden layer for a large dataset such as ours would not give us a good accuracy. We did not want more than 5 hidden layers because then the model would become too complex which might lead to overfitting. We gave the range for the number of neurons in each hidden layer from 10 to 41 neurons.

#### **4.4 ACTIVATION FUNCTION**

For all hidden layers, we have chosen ReLU (Rectified Linear Unit) as the activation function. The linear activation function was not used, even if it is easier to train, is because it fails to understand complex structures. The sigmoid (gives the output between 0 and 1) and tanh (gives the output between -1 and 1) non-linear activation functions were not used due to the gradient vanishing problem. In this problem, during back propagation, the gradient by which the weights should be updated decreases, such that by the time it reaches the first layer, it is barely even significant. This makes it harder to change the values of the weights during back propagation and thus improve the performance of the model. Thus, the ReLU activation function was used since it looks like a linear function, but it is a non-linear function which is able to understand the complex relationships between the data. Softmax is used as the activation function in the output layer since it is needed to predict the probability of a particular word being the output.

#### **4.5 LOSS FUNCTION**

We have utilised the sparse categorical crossentropy loss function. This loss function is used when multiclass classification is performed, which is what we are doing in this project (there are 30 unique classes which are the 30 unique words that the model can recognize). This loss function is used when the class labels are integers. In our dataset, the first column is the class label column, which contains the unique id associated with each unique word. The class labels in our dataset is from 0 to 29.

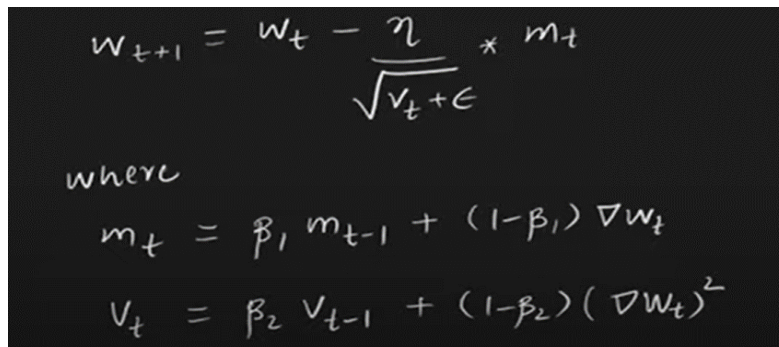
There are other loss functions for multiclass classification as well such as categorical crossentropy, the poisson loss, and Kullback-leibler divergence loss. We did not make use of the categorical crossentropy loss function since it is used when the labels are one-hot encoded, which they are not in our dataset. The poisson loss was not used, since it should be used when the dataset comes from a poisson distribution, which it does not in our case. The KL divergence loss was not used since it is used for measuring the difference between two probability distributions, which is not required in our project.

Keras offers other loss functions for other purposes as well such as the focal loss and generalised intersection over union for object detection; mean squared error, mean absolute percentage error, mean squared logarithmic error, cosine similarity loss, LogCosh loss, Huber loss for regression problems; and Triplet loss for learning embedding. We did not make use of any of these loss functions since they were out of scope of our project.

## 4.6 OPTIMIZATION

We used the Adam optimizer to minimize the loss function and get ideal weight values to improve the accuracy as best as possible. Adam optimizer is one of the best and common optimizers to be used in neural networks, and thus we made use of it.

Adam stands for Adaptive Moment Estimation. It is one of the most famous and widely used optimizers to optimize neural networks. It is very powerful. Adam makes use of the concept of both momentum and learning rate decay to give the best possible accuracy in the least possible time. The mathematic formulation is as shown in Fig 4.3.



$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \epsilon}} * m_t$$

where

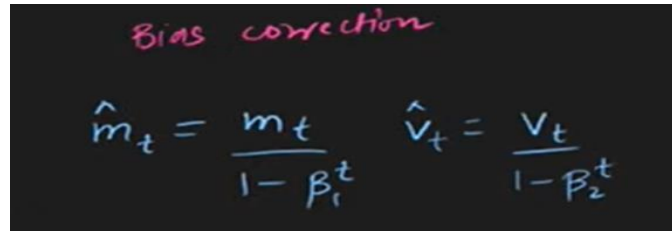
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla w_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) (\nabla w_t)^2$$

**Fig. 4.3: Mathematical formula of Adam Optimizer**

$m_t$  is related to momentum and  $v_t$  is related to learning rate decay.

After calculating  $m_t$  and  $v_t$ , bias correction needs to be performed as shown in figure 4.4. Bias correction is performed since the initial values of  $m_t$  and  $v_t$  are 0. To avoid starting from 0, we do bias correction.



Bias correction

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

**Fig. 4.4: Bias correction**

$t$ , here is the epoch number. Usually, beta 1 value is 0.9 and beta 2 values is 0.99. these can be configured in Keras.

There are other optimizers as well such as stochastic gradient descent (SGD) with momentum, NAG (Nesterov accelerated gradient), Adagrad (Adaptive gradient), and RMSProp (Root mean square propagation). We did not choose SGD with momentum, even though it is able to overcome the local minima problem since it oscillates (due too the momentum gained) before it reaches the global minima. Thus, it takes longer to converge. NAG was introduced to overcome this problem. By dampening the oscillations, it reduces the amount of time it takes for the algorithm to converge. But because the momentum decreases, there is a chance the algorithm may get stuck in the local minima and never converge at the global minima. Thus, we did not make use of NAG. Adagrad is more suitable for those datasets which have sparse input features (features with numerous zero values) or input features that need to be scaled. Since our dataset did not have sparse features, nor did it need to be scaled, we did not make use of Adagrad. Also, one of the biggest disadvantages of adagrad is that it never converges at the global minima. It just converges somewhere near it.

We then get the best hyperparameter values after performing hyperparameter tuning. There are 3 hidden layers with 41, 30, and 25 neurons respectively. The learning rate is 0.001. We train the model using the best values obtained after tuning and find out the epoch till which we get the highest accuracy. Once we have the best epoch, we train the model till this epoch. The optimal number of epochs obtained to get the highest accuracy was 149.

#### **4.7 LANGUAGE TRANSLATION**

To provide the output in the language preference of the user, we used the googletrans python library. This language translation feature allows us the project to be used by a wider range of audience.

#### **4.8 TEXT TO SPEECH CONVERSION**

We were also able to convert the predicted word text to speech, for the word to be said out loud, by using the gtts (google text to speech) library. It converts the text to speech and stores it an mp3 audio file which is played automatically. This feature allows the user to observe the output in whatever form they find convenient.

#### **4.9 WEB INTERFACE**

To provide a proper interface for our project, we used flask to connect our python program to a web interface. Flask is a framework which allows us to build websites using python. With this web interface, users will be easily able to access it.

## 5. IMPLEMENTATION

The first step is to collect the data for training. Since we could not find any proper dataset that we could process to get our desired result, we had to create our own dataset. We open the camera using opencv and start performing the sign for that word. Simultaneously we press the id of the word. This word id will act as the class label which needs to be predicted. The data about the hand landmarks are identified by mediapipe and are written into the csv file. Mediapipe hand model was used to collect the data and export it to a csv file for training. The dataset as seen in figure 5.1 consisted of 30 words with the total 31818 rows. It is a numerical dataset. Another csv file contains the list of words that the model can predict. The id predicted will be mapped to the word so that it can be displayed on the screen, be converted to speech, and be said out loud.

	0	1	2	3	4	5	6	7	8	9	...	33	34	35	36	37	38
0	0	0.0	0.0	0.200787	-0.051181	0.366142	-0.181102	0.484252	-0.307087	0.594488	...	0.039370	-0.976378	-0.074803	-0.429134	-0.145669	-0.57874
1	0	0.0	0.0	0.206349	-0.043651	0.376984	-0.162698	0.507937	-0.273810	0.615079	...	0.047619	-0.980159	-0.079365	-0.428571	-0.142857	-0.58333
2	0	0.0	0.0	0.202381	-0.043651	0.373016	-0.162698	0.500000	-0.277778	0.607143	...	0.039683	-0.980159	-0.083333	-0.428571	-0.146825	-0.57936
3	0	0.0	0.0	0.207171	-0.039841	0.382470	-0.159363	0.509960	-0.274900	0.617530	...	0.051793	-0.980080	-0.079681	-0.426295	-0.143426	-0.58167
4	0	0.0	0.0	0.206349	-0.039683	0.380952	-0.162698	0.507937	-0.281746	0.615079	...	0.051587	-0.980159	-0.079365	-0.428571	-0.142857	-0.58333
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
31813	29	0.0	0.0	-0.115385	-0.185897	-0.153846	-0.403846	-0.096154	-0.564103	-0.019231	...	0.108974	-0.474359	0.128205	-0.365385	0.166667	-0.58974
31814	29	0.0	0.0	-0.116129	-0.180645	-0.154839	-0.387097	-0.103226	-0.535484	-0.012903	...	0.109677	-0.477419	0.135484	-0.367742	0.174194	-0.60000
31815	29	0.0	0.0	-0.123377	-0.181818	-0.162338	-0.396104	-0.110390	-0.551948	-0.025974	...	0.103896	-0.474026	0.123377	-0.376623	0.162338	-0.60385
31816	29	0.0	0.0	-0.113208	-0.176101	-0.150943	-0.396226	-0.094340	-0.559748	-0.018868	...	0.113208	-0.471698	0.125786	-0.364780	0.169811	-0.59115
31817	29	0.0	0.0	-0.107595	-0.164557	-0.164557	-0.379747	-0.113924	-0.537975	-0.018987	...	0.126582	-0.487342	0.126582	-0.373418	0.170886	-0.59493

31818 rows × 43 columns

**Fig. 5.1: Dataset**

Then we imported the required libraries to implement our functionality. List of Libraries used in the proposed system can be seen in figure 5.2. Apart from these, some other libraries such as os, mediapipe, matplotlib, seaborn, flask, gttts and googletrans were also used.

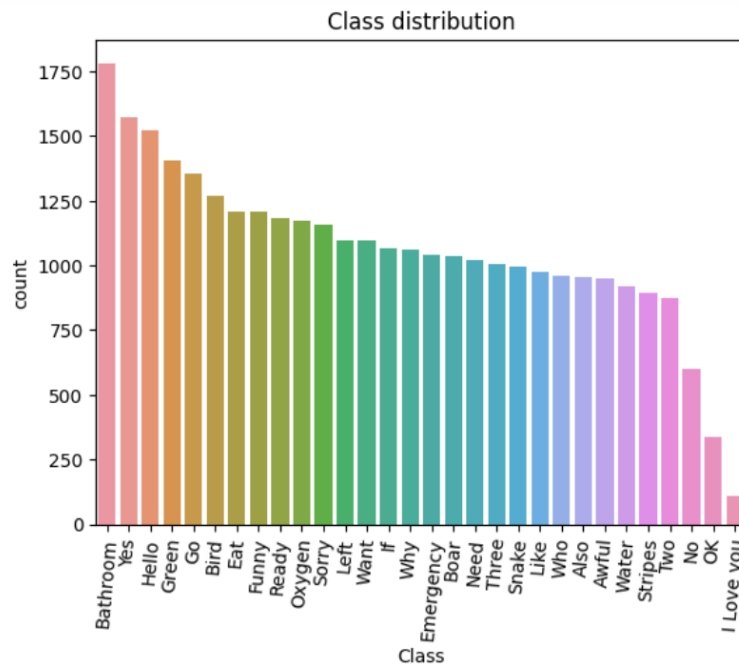
```
In [1]: import csv

import numpy as np
import pandas as pd
import tensorflow as tf
import keras_tuner as kt
from tensorflow import keras
from keras import regularizers
from sklearn.model_selection import train_test_split

RANDOM_SEED = 42
```

**Fig. 5.2: List of Libraries used in the proposed system**

The class distribution of various classes in the dataset can be seen in figure 5.3. The graph was drawn using matplotlib and seaborn. The x axis represents the word and the y axis represents the number of records associated with each word in the dataset.



**Fig. 5.3: Class Distribution for all the words**

The next step is to build the model, once all the training data has been collected. This can be seen in fig 5.4. The input layer consists of 42 neurons since 42 hand landmarks are recognized by the mediapipe hands model when a word is signed in front of the camera. The keras hyperband tuner was used to determine the number of hidden layers, the number of neurons in the hidden layer and the learning rate for adam optimizer. The



range given for the number of hidden layers is 2-5 and the range given for the number of neurons in each hidden layer is 10-41. We then set the parameters like activation function. We used Relu activation function and softmax activation function as seen in figure 5.4. The sparse categorical crossentropy loss function was used. The number of neurons in the output layer is 30 since we have 30 unique classes in our dataset. To improve the accuracy, we made use of the Adam optimizer.

```
def build_model(hp):                #hp means hyper parameters
    model=keras.Sequential()
    model.add(keras.layers.Input((21 * 2, )))
    #providing the range for hidden layers
    for i in range(hp.Int('num_of_layers',2,6)):
        #providing range for number of neurons in hidden layers
        model.add(keras.layers.Dense(units=hp.Int('num_of_neurons'+ str(i),min_value=10,max_value=41),
                                         activation='relu'))
    model.add(keras.layers.Dense(30,activation='softmax')) #output layer
    #compiling the model
    model.compile(optimizer=keras.optimizers.Adam(hp.Choice('learning_rate',values=[1e-2, 1e-3, 1e-4])), #tuning learning rate
                  loss='sparse_categorical_crossentropy',metrics=['accuracy'])
    return model
```

**Fig. 5.4: Building the model**

We chose Hyperband as the tuner and implemented it as shown in figure 5.5. The objective of the tuner is to maximize the accuracy. The maximum number of epochs for which it can run is 100. The number of models that can be created in a bracket is 3. The name of the directory is specified so that all checkpoints of the model training can be saved. This can be used to resume the training.

```
tuner = kt.Hyperband(build_model,
                     objective='val_accuracy',
                     max_epochs=100,
                     factor=3,
                     directory = 'my_dir5')
```

**Fig. 5.5: Hyperband Tuner**

We also used early stopping to help identify words faster and accurately as shown in figure 5.6. Early stopping is a technique which is used to avoid overfitting. Overfitting takes place when the model is too complex, and thus instead of learning, it memorizes the training data, even the noise. So it works very well with training data, but since it is unable to generalise to previously unseen data, it performs poorly with testing or

validation data. People assume that if the model is trained for larger number of epochs, the accuracy will improve. That is not always the case. Training the model for a large number of epochs can lead to overfitting as well. Thus, early stopping allows us to set a large number of epochs for training, but it stops the training once the model performance stops improving. So it is not necessary that the model will run for all the epochs mentioned. We are monitoring the loss here. The patience is set to 20. Patience is the number of epochs with no improvement after which the training will be stopped. So, here we stop the training if the loss does not decrease even after 20 epochs.

```
stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=20)
```

***Fig. 5.6: Early Stopping***

Then we find the best values for the hyperparameters using the hyperband tuner as seen in figure 5.7

```
tuner.search(X_train, y_train, epochs=200, validation_split=0.2, callbacks=[stop_early])  
  
# Get the optimal hyperparameters  
best_hps=tuner.get_best_hyperparameters(num_trials=2)[0]
```

***Fig. 5.7: Training the data***

Figure 5.8 shows the best values obtained for the hyperparameters after performing hyperparameter tuning. The number of hidden layers in our model should be 3 with 41, 30 and 25 neurons in each of them respectively. The learning rate should be 0.001.

```
In [23]: best_hps.values

Out[23]: {'num_of_layers': 3,
          'num_of_neurons0': 41,
          'num_of_neurons1': 30,
          'learning_rate': 0.001,
          'num_of_neurons2': 25,
          'num_of_neurons3': 38,
          'num_of_neurons4': 40,
          'num_of_neurons5': 22,
          'tuner/epochs': 100,
          'tuner/initial_epoch': 34,
          'tuner/bracket': 3,
          'tuner/round': 3,
          'tuner/trial_id': '0203'}
```

**Fig. 5.8: hyperparameter tuning best values**

We train the model with the best hyperparameter values and find out the epoch till which we get the highest accuracy. The best epoch found was 149. We retrain the model for 149 epochs. This can be seen in fig 5.9.

```
In [24]: model = tuner.hypermodel.build(best_hps)
history = model.fit(X_train, y_train, epochs=150, validation_split=0.2)
val_acc_per_epoch = history.history['val_accuracy']
best_epoch = val_acc_per_epoch.index(max(val_acc_per_epoch)) + 1
print('Best epoch: %d' % (best_epoch,))

Epoch 145/150
597/597 [=====] - 2s 3ms/step - loss: 0.0101 - accuracy: 0.9969 - val_loss: 0.0395 - val_accuracy:
0.9922
Epoch 146/150
597/597 [=====] - 2s 3ms/step - loss: 0.0155 - accuracy: 0.9953 - val_loss: 0.0407 - val_accuracy:
0.9918
Epoch 147/150
597/597 [=====] - 2s 3ms/step - loss: 0.0130 - accuracy: 0.9956 - val_loss: 0.0425 - val_accuracy:
0.9916
Epoch 148/150
597/597 [=====] - 2s 3ms/step - loss: 0.0080 - accuracy: 0.9978 - val_loss: 0.0438 - val_accuracy:
0.9935
Epoch 149/150
597/597 [=====] - 2s 3ms/step - loss: 0.0100 - accuracy: 0.9963 - val_loss: 0.0391 - val_accuracy:
0.9941
Epoch 150/150
597/597 [=====] - 2s 3ms/step - loss: 0.0128 - accuracy: 0.9958 - val_loss: 0.0439 - val_accuracy:
0.9912
Best epoch: 149

In [25]: hypermodel = tuner.hypermodel.build(best_hps)
# Retrain the model
hypermodel.fit(X_train, y_train, epochs=best_epoch, validation_split=0.2)
```

**Fig. 5.9: Training the model using tuner.**

After the model was trained, we included a language translation feature using the googletrans library. The text is converted to speech to get the final output, by using the gtts python library. This can be seen in figure 5.10.

```

text_to_translate = translator.translate(hand_sign_text,src= from_lang,dest= language)
text = text_to_translate.text
#language = 'en'

# Passing the text and language to the engine,
# here we have marked slow=False. Which tells
# the module that the converted audio should
# have a high speed
#myobj = gTTS(text=hand_sign_text, lang=language, slow=False)
myobj = gTTS(text=text, lang=language, slow=False)

# Saving the converted audio in a mp3 file named
# after the word signed
filename = hand_sign_text + "-" + language + ".mp3"
file_exists = exists(filename)
if file_exists==False:
    myobj.save(filename)
else:
    # Playing the converted file
    os.system(filename)

```

***Fig. 5.10: Language translation and text to speech conversion***

Once the model was created, trained and tested, it was stored and used in a python file which contains flask code as well. This python file will allow the user to choose the language, open the web camera, and recognise the sign being performed by making use of the stored model, display the predicted word as text on the screen, translate the language, and convert the text to speech, so that the output is said out loud as well.

## 6. TESTING AND RESULTS

After successful building of model, the result was pretty accurate with great accuracy as shown in figure 6.1 and 6.2

```
eval_result = hypermodel.evaluate(X_test, y_test)
print("[test loss, test accuracy]:", eval_result)
```

```
249/249 [=====] - 1s 2ms/step - loss: 0.0682 - accuracy: 0.9911
[test loss, test accuracy]: [0.06824685633182526, 0.991074800491333]
```

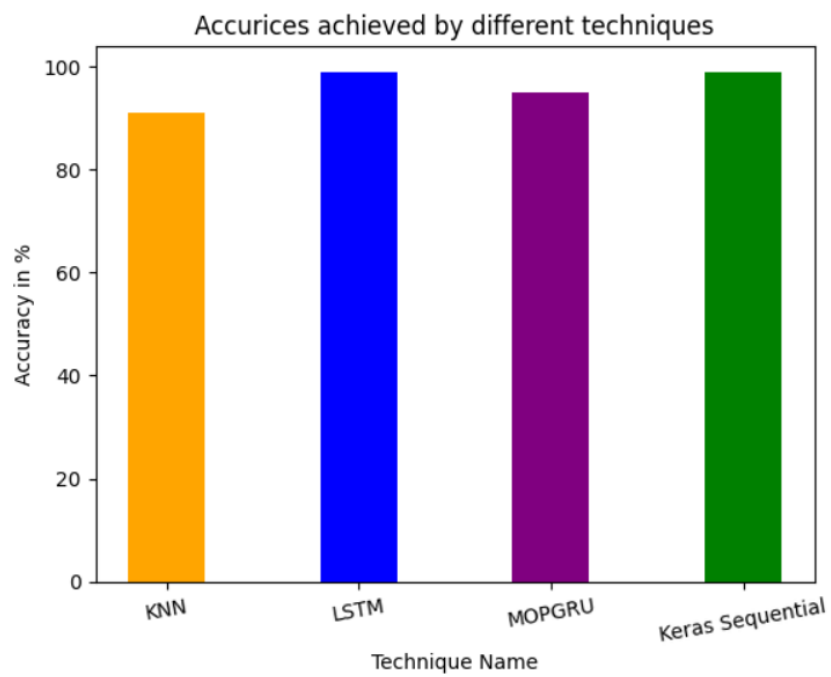
**Fig 6.1: Attained overall accuracy of 99.11%**

Classification Report				
	precision	recall	f1-score	support
0	0.96	0.99	0.98	368
1	0.96	0.98	0.97	383
2	0.98	0.99	0.98	356
3	1.00	1.00	1.00	80
4	0.96	1.00	0.98	25
5	0.99	0.97	0.98	154
6	0.97	0.97	0.97	279
7	1.00	1.00	1.00	222
8	1.00	1.00	1.00	317
9	0.98	0.94	0.96	270
10	1.00	0.99	1.00	428
11	0.98	0.95	0.97	256
12	1.00	1.00	1.00	245
13	1.00	1.00	1.00	300
14	1.00	1.00	1.00	317
15	1.00	1.00	1.00	288
16	0.97	0.99	0.98	227
17	1.00	1.00	1.00	238
18	1.00	1.00	1.00	242
19	0.99	1.00	1.00	205
20	1.00	1.00	1.00	251
21	0.99	0.99	0.99	260
22	1.00	1.00	1.00	224
23	1.00	1.00	1.00	283
24	0.99	1.00	1.00	366
25	1.00	1.00	1.00	272
26	1.00	1.00	1.00	297
27	0.99	1.00	0.99	315
28	1.00	0.99	0.99	247
29	1.00	1.00	1.00	240
accuracy			0.99	7955
macro avg	0.99	0.99	0.99	7955
weighted avg	0.99	0.99	0.99	7955

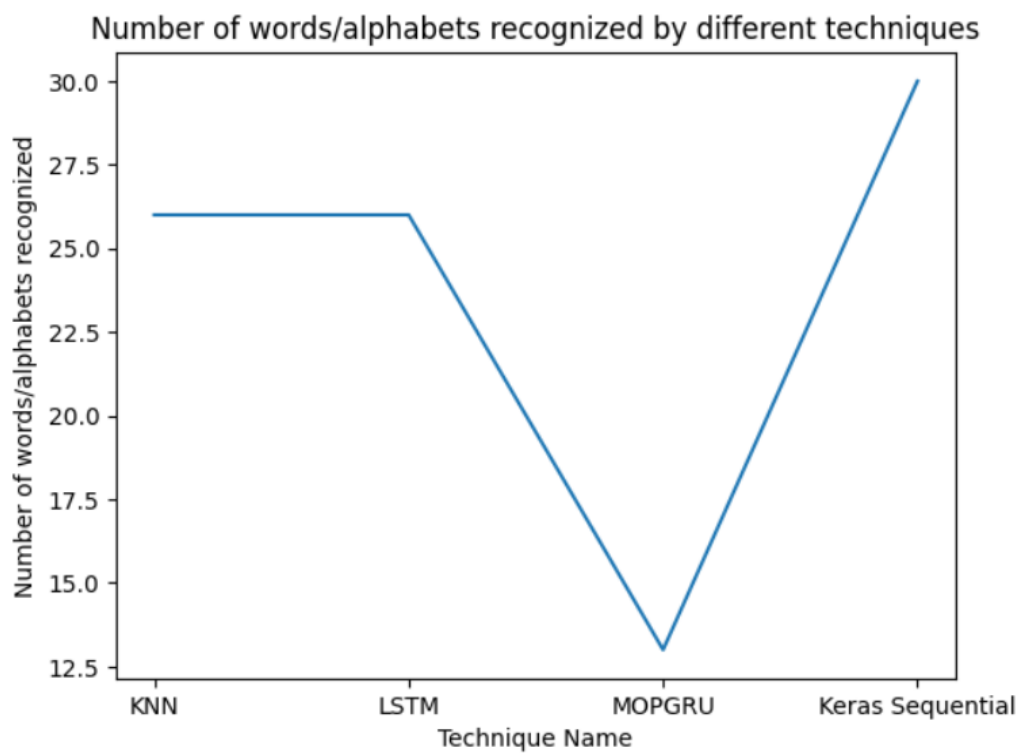
**Fig 6.2: Evaluation**

<b>Paper</b>	<b>Algorithm/Model used</b>	<b>Number of words / alphabets recognized</b>	<b>Accuracy Achieved</b>
Ketan Gomase, Akshata Dhanawade, Prasad Gurav, Sandesh Lokare: Sign Language Recognition using Mediapipe: International Research Journal of Engineering and Technology (IRJET) Volume: 09 Issue: 01   Jan 2022	Mediapipe with K nearest neighbor (KNN) machine learning model	26 English Alphabets in ASL	86-91%
Sundar B, Bagyammal T: American Sign Language Recognition for Alphabets Using MediaPipe and LSTM: Elsevier, Science Direct, Procedia Computer Science215 (2022) 642-651	Mediapipe with Recurrent neural network LSTM (Long Short Term Memory) model	26 English Alphabets in ASL	99%
Subramanian, B., Olimov, B., Naik, S.M. et al. An integrated mediapipe-optimized GRU model for Indian sign language recognition. Sci Rep 12, 11964 (2022).	Integrated mediapipe optimized gated recurrent unit (MOPGRU) model	13 words in ISL	95%
Proposed Methodology	Mediapipe with Keras Sequential Neural Network Model	30 words in ASL	99%

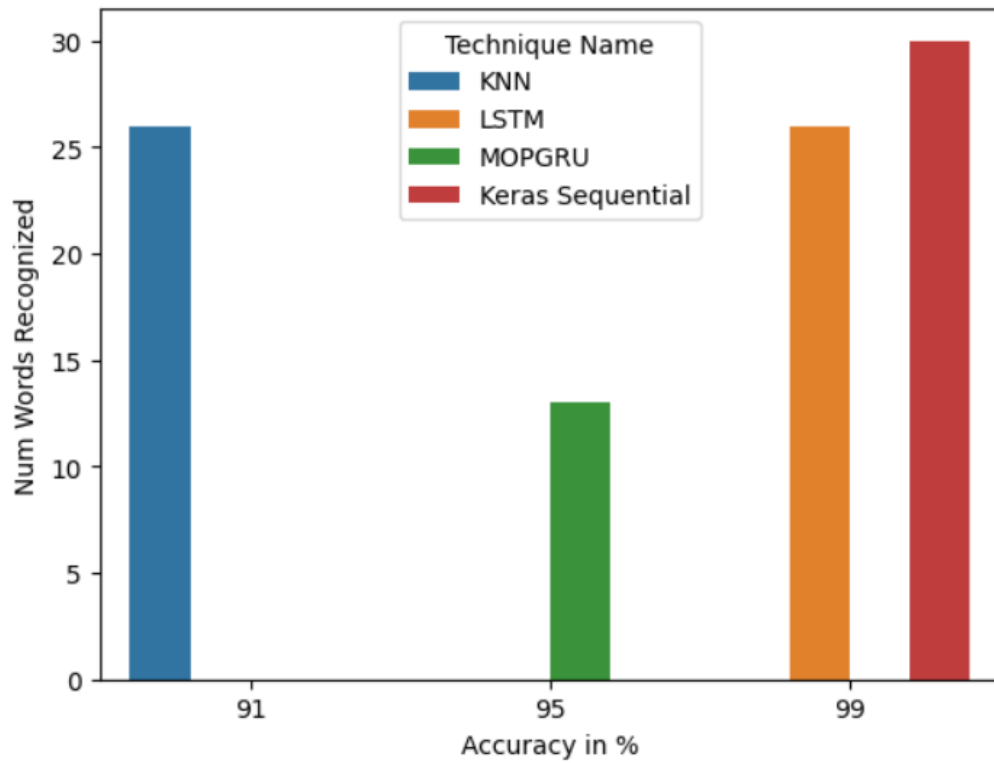
***Fig 6.3: Result Analysis***



***Fig 6.4: Accuracies achieved by different techniques***



***Fig 6.5: Number of words / alphabets recognized by different techniques***

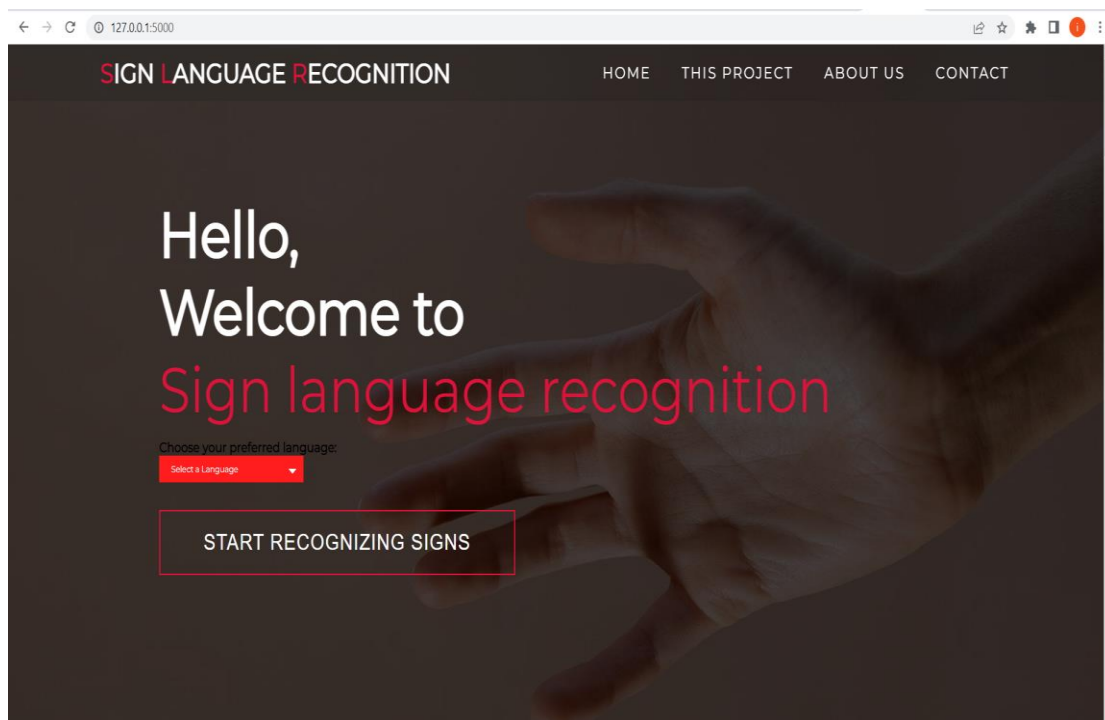


***Fig 6.6: Overall Visual Result Analysis***

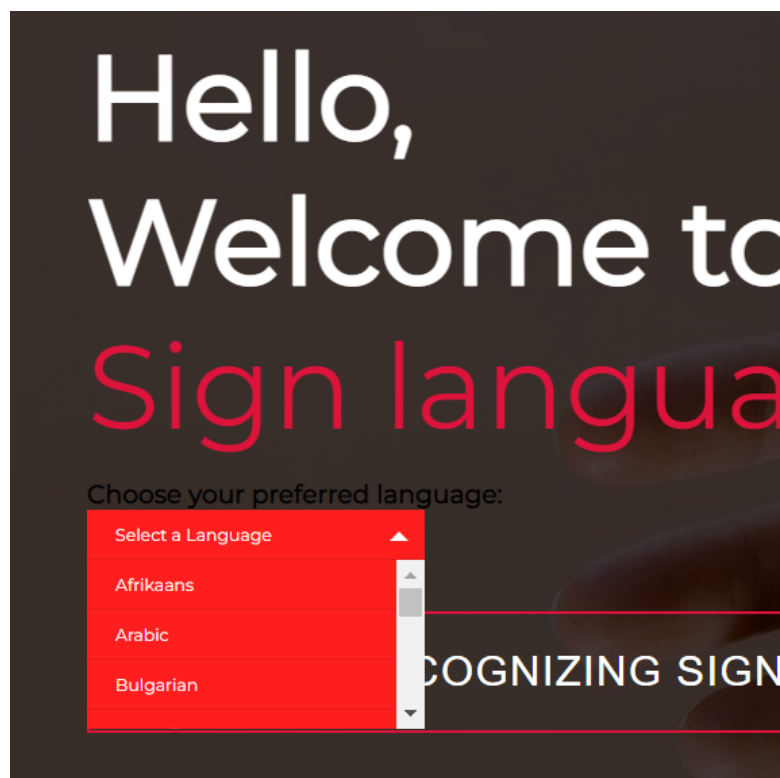
Fig 6.3 shows the comparison between our project and three other papers in terms of the algorithm or model used, the number of words or alphabets recognized and accuracy achieved. It can be seen from the graphs displayed in fig 6.4, 6.5, and 6.6 that our project (that used the technique ‘Keras Sequential’) gave the highest accuracy with the largest vocabulary, thereby outperforming the other projects.

After connecting the python program to the web interface, figures 6.7 and 6.8 demonstrate how our web interface appeared like and the language selection feature of our web interface.





*Fig. 6.7: Web App Interface*



*Fig. 6.8: Language selection panel*

Figure 6.9 shows what happens when we choose our language and then click on “start recognizing signs” button of the web interface.



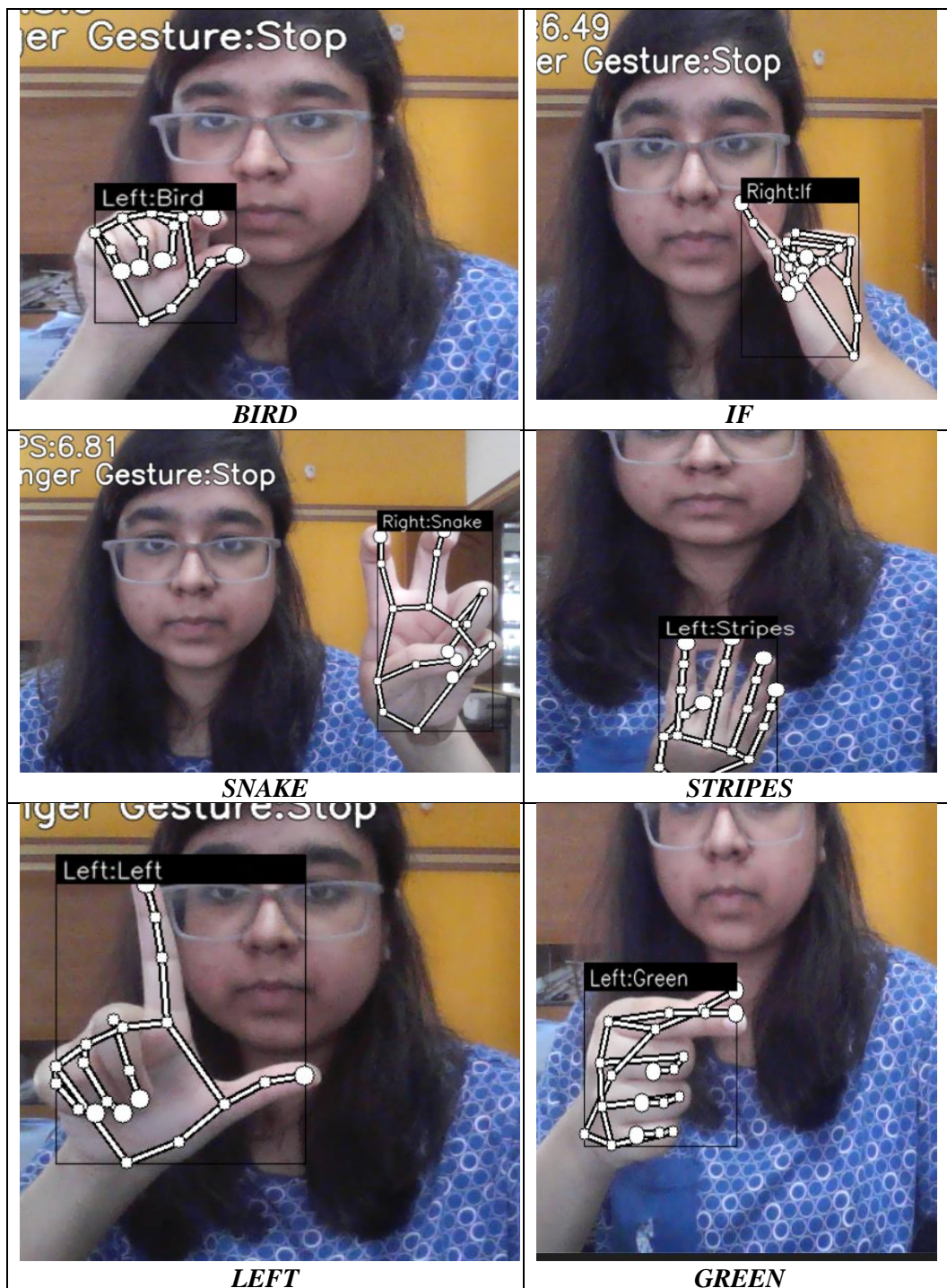
***Fig. 6.9: Camera turns on***

As shown in figure 6.10, our application recognizes signs and output it using computer speaker in the preferred language.

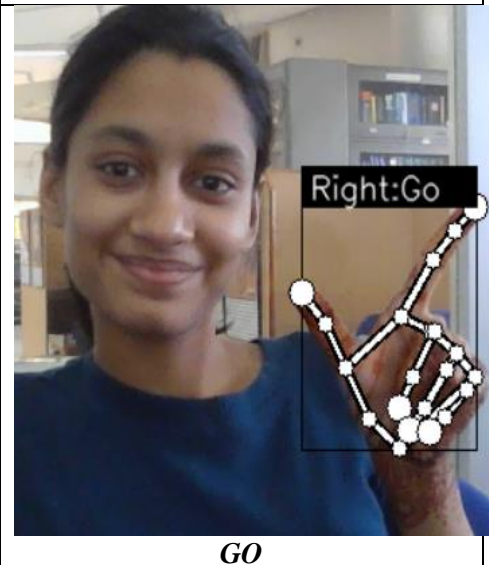
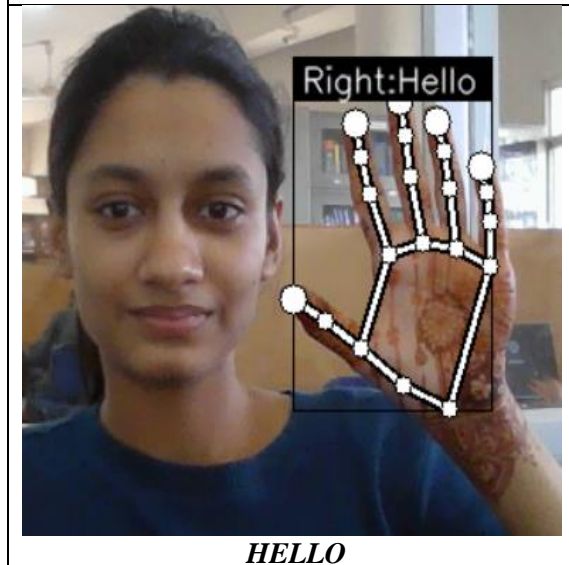
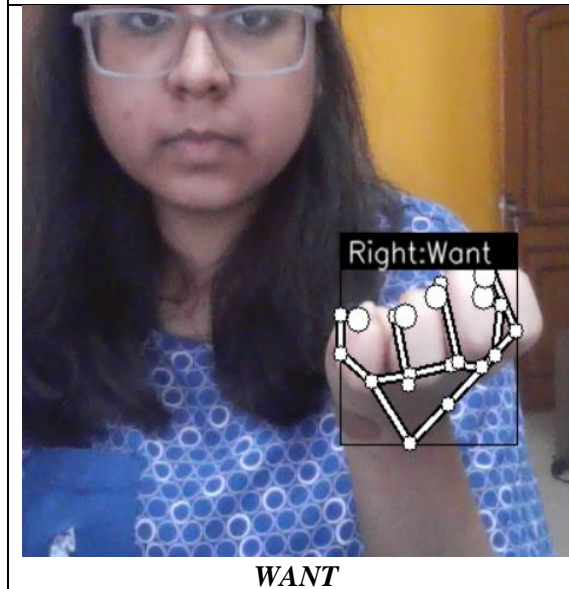


***Fig. 6.10: Output***

Our application recognizes 30 signs accurately which can be seen in the following table. It can also be seen from the table that the sign is being recognized even with mahendi on hand, and is unaffected by background or lighting conditions. In addition, even though the dataset was created with signs performed by a single person, our model is able to generalise well and is able to accurately recognize signs performed by multiple different people.

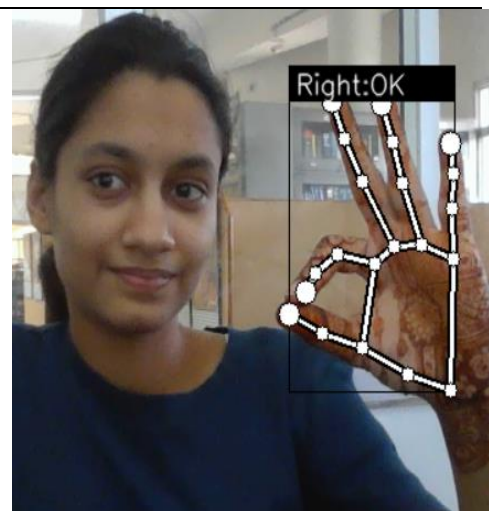




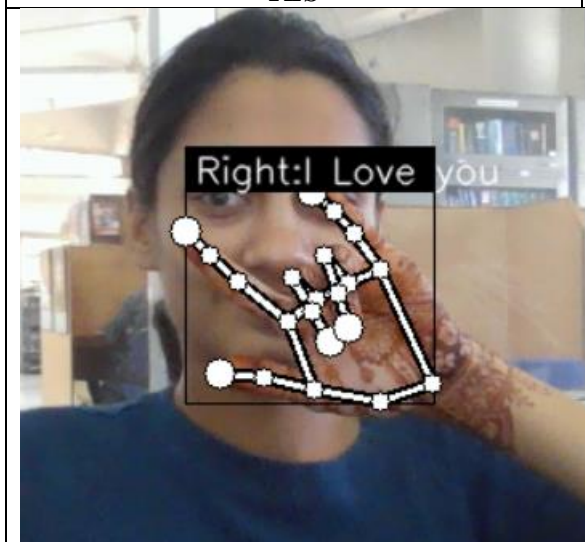




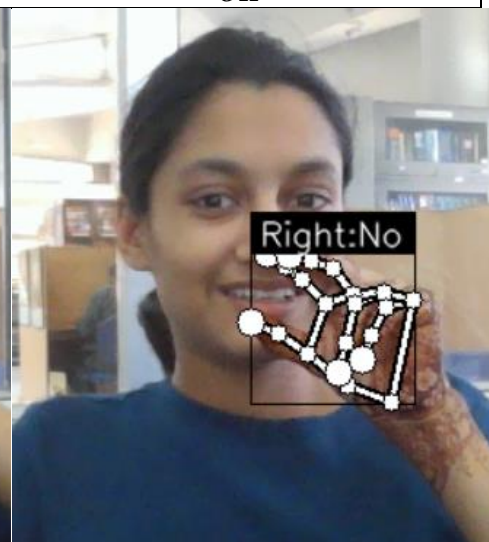
*YES*



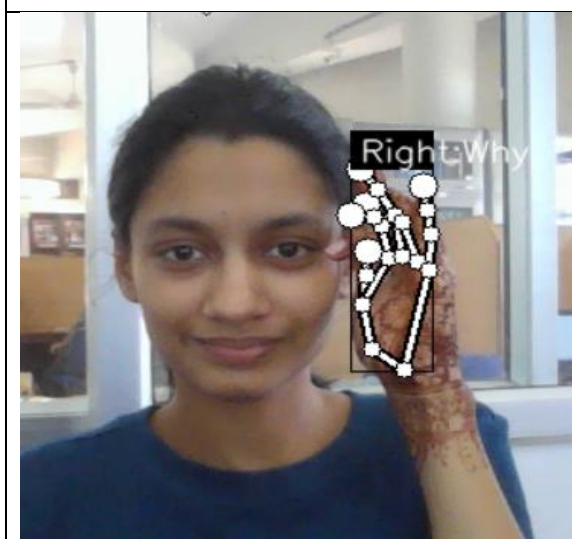
*OK*



*I LOVE YOU*



*NO*



*WHY*



*ALSO*





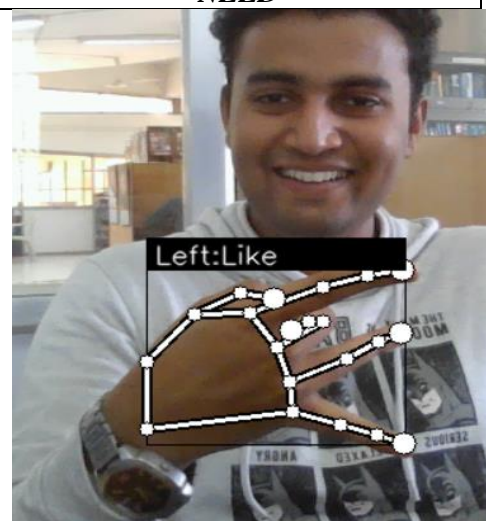
*EAT*



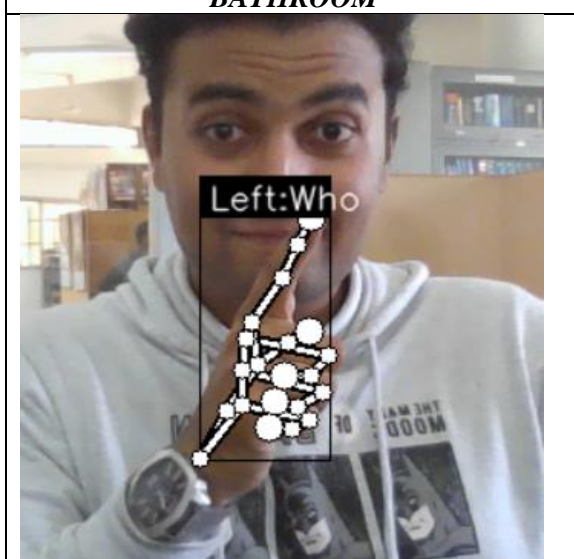
*NEED*



*BATHROOM*



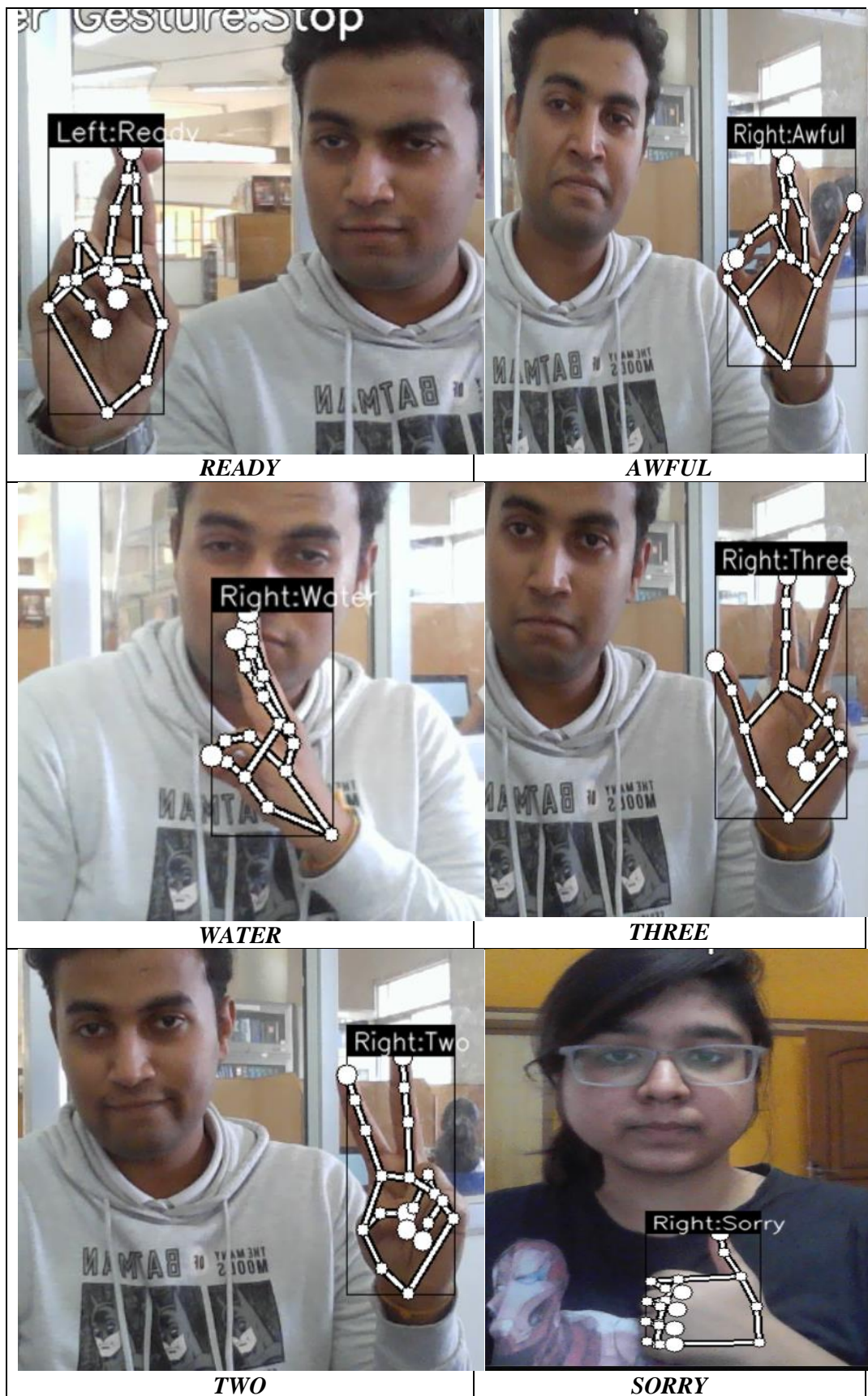
*LIKE*



*WHO*



*FUNNY*



## **7. CONCLUSION AND FUTURE SCOPE**

### **7.1 CONCLUSION**

Based on the above-mentioned information, we conclude that our project is able to facilitate communication between mute and non-mute people by recognizing words being signed that will lead to stronger bonds between people. It can effectively recognize signs on the word level and can provide realistic communication, in an easy and convenient way. This was done using technologies like python, TensorFlow and deep learning models like Keras Sequential model. Our proposed system can attain high accuracy of 99% which is evidently better than other existing systems. We were able to achieve this by performing hyperparameter tuning using the keras tuner. Our project also has a larger vocabulary when compared to other models that are able to recognize signs at the word level. By making a web application, the accessibility of this functionality can be increased, and can help more people. Through the app, the user will be able to enter the language in which they wish to hear the message that is trying to be communicated by the mute person. With the language selection option provided in the web app it can be used by a wide range of audience since the output can be heard in different languages. The text to speech conversion allows the end user to observe the output both in the form of text and speech, making it more convenient for them. Thus, our developed model accurately recognizes signs and gives output in different languages with a good accuracy. The model is also unaffected by drawings on the hand, such as mahendi, and is also able to recognize signs accurately in different background and lighting conditions.

### **7.2 LIMITATIONS**

- Our system is limited to identifying only 30 words.
- Although several different sign languages are used around the world, our model is trained on only American Sign Language out of all the other existing ones.



- Because the words are translated in other languages and we translate the word signed from English language to that chosen language, there is sometimes a lag or delay in the output being said out loud.

### **7.3 FUTURE SCOPE**

- The dataset of our model can be increased to recognise more than 30 words since that would increase the usability of the application.
- The lag or delay between the word signed and the word being said out loud can also be decreased in the future. This will enhance the user experience.
- The model can be extended to recognize words from other sign language like Indian Sign Language, Portuguese Sign Language, etc.
- The website can be fully deployed to make it more accessible by the related audience.

## 8. VISIBLE OUTPUT

The following paper, written by us, was published and presented in the conference named ‘International Conference on Cognitive & Intelligent Computing 2022’

### **A Review on Sign Language Recognition Techniques**

**S. Rakesh<sup>1</sup> , M. Venu Gopalachari<sup>1</sup> , Ishika Gupta<sup>1</sup> , Kritika Agarwal<sup>1</sup> , Ganji Nishanth<sup>1</sup>**

**1 IT Department, Chaitanya Bharathi Institute of Technology, Hyderabad**

**[srakesh\\_it@cbit.ac.in](mailto:srakesh_it@cbit.ac.in)**

**Abstract:** Not many people understand sign language, which poses a problem to people who can only communicate with sign language. They are unable to perform basic day-to-day activities due to this problem. Thus, many solutions were developed to help such people communicate effectively and live relatively easier life. Many papers from reputed journals were studied to gain an understanding of the working of these solutions. Based on our research, there are two kinds of solutions proposed: a glove-based approach where mute people can wear the gloves, and sign what they want to communicate and the device placed on the glove can convert the gestures signed to speech or text, which a non-mute person can easily understand and comprehend. The second solution is a computer vision based. By using a camera, the gestures signed by a person can be captured, processed, and identified. The output can be given in the form of text or speech or both. The rest of this review paper goes in-depth about the various solutions proposed. Their methodology, advantages and disadvantages are analyzed, compared, and discussed.

**Keywords-** Deep Learning, Sign Language, Sign to Speech

### **Introduction**

Most of us in today’s society do not realize how powerful words can be and how huge of an impact they can have on our lives. Effective communication helps us in expressing our thoughts, ideas and feelings. It helps us build lasting relationships in both personal

and professional life. Unfortunately, not many people have the privilege to easily communicate with each other using speech. There are people who are unable to speak. They cannot use their voice to articulate words. Thus, sign language was created to enable such people to communicate. Sign language is a medium of communication using visual hand gestures called signs. There are more than 70 million people in the world, who use sign language. But there is not a single universal sign language that people use to communicate. There are more than 300 sign languages that are used across the world. They can differ from county to country. Even within the same country, sign languages can have subtle differences.

### **Literature survey**

In reference [1], the authors of the paper have proposed an algorithm that uses YCbCr color space, COG and template matching for segmenting the hand, detecting the fingers and then identifying the hand gesture from real imagery to help the deaf and mute people. To discover isolated indicators, the authors of [2] use two appearance-based algorithms, I3D and TimeSformer, and one posed base approach, SPOTER, on two publically available datasets, AUTSL and WLASL300. They increased the accuracy of the WLASL300 dataset by applying the CMA-ES optimization technique. They've also proposed Neural Ensembler, an ensemble approach based on a Transformer model. The authors of [3] used techniques such as Fast Accelerated Segment Test (FAST), Scale-Invariant Feature Transformation (SIFT), and Convolution Neural Networks (CNN) for effective feature extraction, which is required for automatic gesture detection. FAST and SIFT are used to recognise and calculate features from pictures, respectively. CNN was utilised for classification, with FAST-SIFT characteristics hybridised. [5] Describes a method for capturing a still hand image frame with a webcam. These frames were enhanced by post-processing. The sign language is then translated into English text using feature extraction and classification techniques. The text to speech API is used to transform this translation to speech. [6] Used a user-defined dataset with around 2000 photos, approximately 400 for each of its classes, to develop and implement a sign language recognition model based on a CNN. To train the dataset, they used a Pre-Trained SSD Mobile net V2 architecture. The authors of [7] used transfer learning and deep CNN fine tuning to recognise 32 Arabic sign language hand movements. The

networks were fed regular 2D images of various Arabic Sign Language data. [9] uses Blob analysis to extract the hand gesture from the image considering that the largest blob is the hand. Then the blob was resized, sampled and length encoding was done. To identify the hand signs, the authors of [11] took a different approach. It employs algorithms (Gauss Laplace Algorithm) and methods for performing sign language translation. The vision of an efficient system for translating sign language to text is well within reach, but the challenges lie in optimization. A coloured tape is applied to the fingers to serve as an input. We must show these fingers to a computer that has MATLAB installed. In [12], this paper Sign Language Recognition is done using Convolutional Neural Network. This system is able to interpret static hand gestures. This system converts uploaded video to text and then converts that text to a gif. This model uses dense layer and softmax regression in the classification stage in CNN implementation.

### **Methodology:**

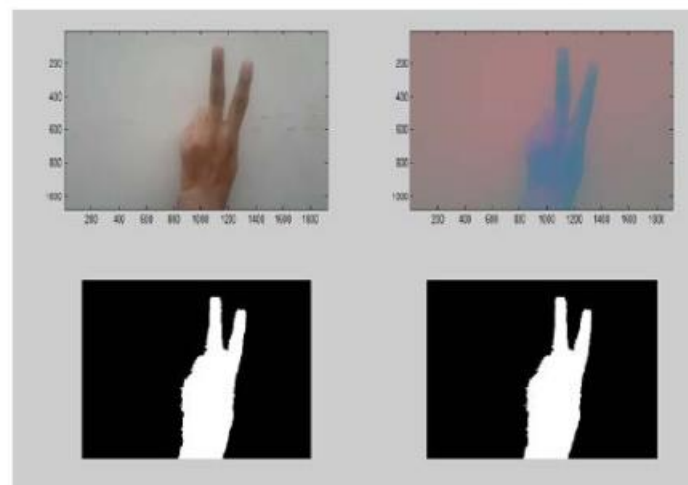


Fig. 1. Image conversion.

According to the methodology proposed by [1] , the user will have to upload their video of performing a sign. The video is then divided into frames and each frame is used as an image. To segment the hand from the background, the RGB image is converted to YCbCr color space. The centroid of gravity (COG) is then calculated. The farthest distance from the centroid to the tip of the longest active finger in the particular gesture

is taken as the radius to draw a circle. The circle consists of the gesture. Thus, COG was used to segment the hand. The unknown gesture is then compared with the preset template models of individual gestures to identify the gesture. [2] Implemented I3D (a family of convolutional neural network models for video classification) with three inputs: Crop and Resize, Masked, and OptFlow. As input, 16 frames from each video were taken. The Kinetics400 dataset was used to pre-train all I3D models. The authors followed the original Facebooksearch Github implementation. The authors implemented SPOTER (sign pose-based transformer) on top of body-pose sequences using Pytorch. They followed Boháek M., Hrz M. Sign Pose-Based Transformer for Word-Level Sign Language Recognition training. The SGD optimizer with a learning rate of 0.001 was used. Momentum and decay were both set to zero. The initial weights of the model were drawn from a uniform distribution of [0,1]. On WLASL300, they trained SPOTER for 100 epochs. They trained SPOTER with OpenPose and MMPose on the AUTSL dataset for 12 and 35 epochs, respectively, until each model variant converged on the training split. The authors used a neural ensemble to combine the outputs of individual models and make a final decision. PyTorch was used to implement it.

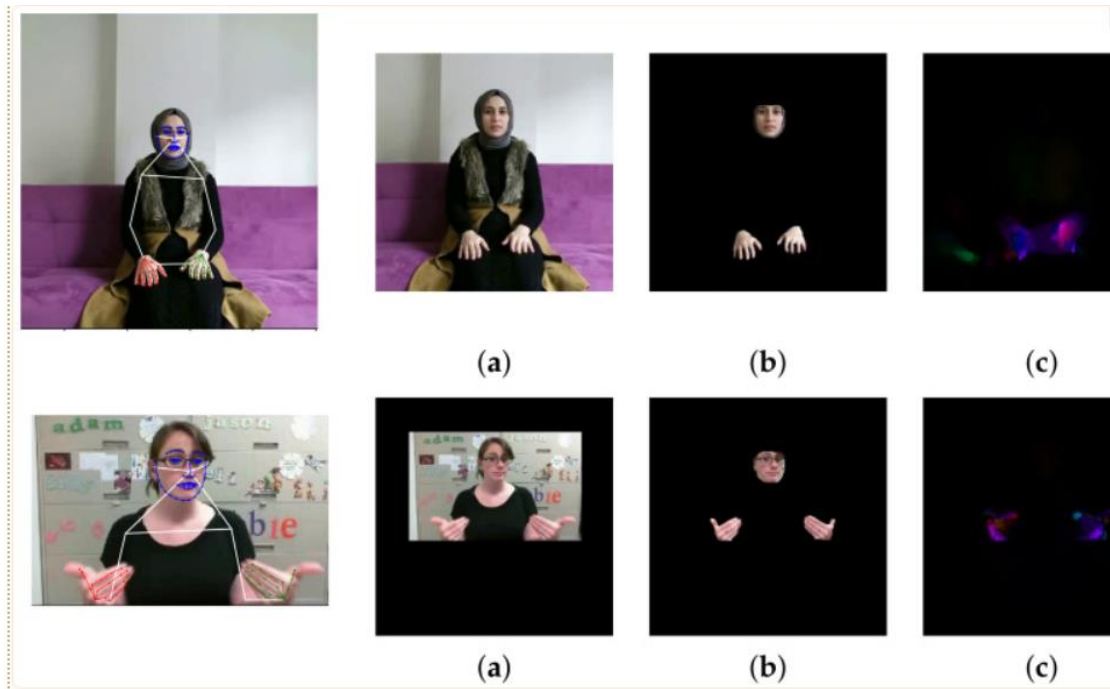


Fig. 2. Different data modalities are given as input.

FAST is a corner detection method that is used to extract feature points, which are then used in many computer vision tasks such as tracking and mapping. SIFT is a computer vision algorithm that detects, describes, and matches local features in images while remaining independent of image scale and rotation. [3] authors proposed three major phases for sign recognition: data preprocessing, feature extraction, and CNN training and testing. The stored static single-handed images are resized in the first phase, and then data augmentation is performed on those resized images. FAST techniques are then used to localize key points. In the third phase, the value of these localized key points was calculated using SIFT. These values were then fed into CNN for training and classification. Keras, a Python-based library, was used to implement and test the system. The different steps described in [5] to detect signs are: first, the hand gestures are captured by video, they are then divided into frames. Image preprocessing is then done and then feature extraction is performed using HOG (Histogram of Oriented Gradients). SVM (Support vector machines) is used for classification and recognition. The text is given as the output. The google API is used for converting text to speech to give audio as the output as well. The authors have created their own dataset of 26 English alphabets of ISL. 4800 images were used for training and 1200 images were used for testing. In paper [6] the three steps were recognized to design the model- Obtaining footage of the user signing and taking them as input, classifying each frame in the video as a sign, and reconstructing and displaying the most likely Sign from classification scores to provide that as output. They made use of A Convolutional Neural Network), a Deep Learning system that can take an input image and assign importance to various elements and objects in the image, as well as distinguish between them. When compared to other classification systems, a ConvNet requires far less preprocessing. ConvNets can pick up these filters and properties with the right training, but basic approaches require filter engineering by hand. A dataset of 2000 images with 400 images divided among 5 classes was used to train this model. The suggested method in [9] makes use of a skin color model to recognize the hand in the image, and additional preprocessing is carried out to eliminate unnecessary noise and regions. Since the hand is the biggest blob in the image, blob analysis is used to extract the hand gesture from the picture. Then, in order to remove the size variance limitation, the blob is shrunk to a standard size. The junction point between the grid line and the boundary

is then extracted. The hand gesture's border is represented using the Freeman chain coding. In order to reduce the duration of encoding for chain code run-length is completed. Its shape number is discovered by locating the first variation in the chain code. Each motion may be uniquely identified using a number.



Fig. 3. Gauss laplace edge detection.

In [11], a colourful tape is wrapped around the fingers and used as an input. We must present these fingers to a computer that has MATLAB installed. The output of the computer is translated to digital and processed by the microcontroller before responding as voice via the speaker. To generate the output in this project, we used a microcontroller, a voice IC, and a speaker. Power supply and voice IC are employed as hardware components (aPR33a3). MATLAB is the software utilized.

### **Result Analysis:**

The major advantage of [1] was that their algorithm was able to detect the hand gesture from any background (robust to background) and was able to identify the hands of different skin tones, thereby giving a better result. But at the same time, it was computationally expensive and could not detect sign language in real-time, since the user would have to provide a video of them. The individual models implemented in [2] gave suboptimal results. Appearance-based approaches were able to distinguish two similar signs, but it required a larger training dataset. Pose based approach required a smaller training dataset, but was unable to distinguish between similar signs. By using ensemble techniques, better accuracies were achieved: 96.37 % in AUTSL dataset and 73.87 % in the WLASL300 dataset. The neural ensemble method proposed by the

authors has high potential for future research. But isolated sign language recognition still poses a problem in an uncontrolled environment. The findings of the methodologies suggested in [3] were tested on 24 alphabet sets and 10-digit sets, 34 Indian Sign Language motions, and two publicly accessible datasets, the Jochen Trisech Dataset (JTD) and the NUS-II dataset. The proposed FiST CNN model outperformed previous models such as CNN and SIFT CNN. In terms of accuracy and calculation time, FiST CNN outperformed. The FiST CNN attained accuracy of 97.89%, 95.68%, 94.90%, and 95.87% for ISL-alphabets, MNIST, JTD, and NUS-II, respectively. The model cannot detect word-level signals, which is a drawback.

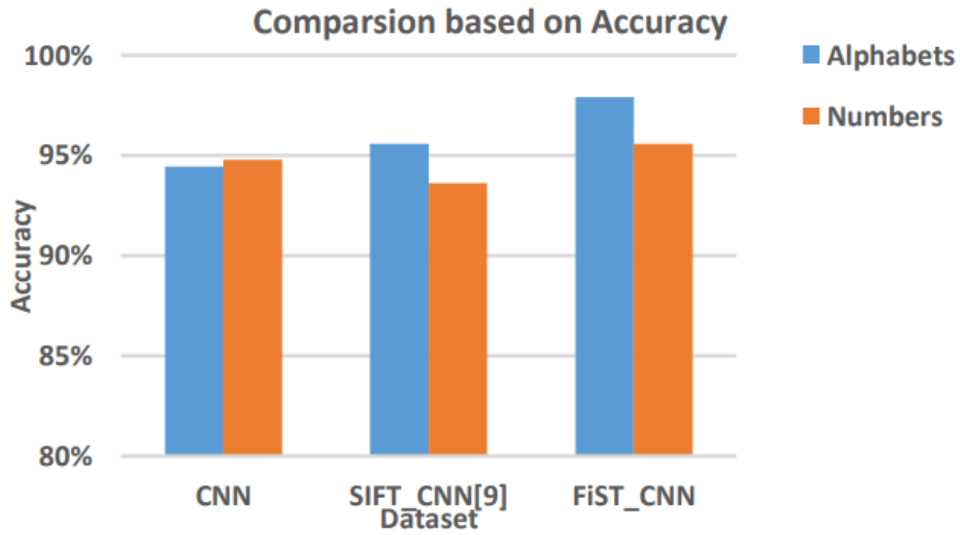


Fig. 4. Accuracy Comparison.

The overall accuracy of the model proposed by [5] was 88%. But this application is not portable. It could be made into a mobile application for convenience. Also, it cannot identify word-level signs. The same drawback is mentioned in [3]. The proposed system in [6] gave accurate results under controlled light and intensity. As a result, by extending the dataset, the model may be simply extended on a wide scale. However, this approach has significant drawbacks, such as environmental conditions such as low light intensity and unmanaged backdrop that reduced detection accuracy. By converting the gesture to a shape number, the object descriptor used in [9] to convey boundary information made categorization simple and straightforward. After analysing a few NSL movements, this method for gesture detection is determined to be straightforward



and satisfactory. It still has to be tested on additional gestures, though. In [11], despite using a low-quality camera, this system was able to process them and produce a video output. All input images were captured by webcam or an android device. To minimize the time and memory requirements associated with pattern recognition, the captured image is converted into binary form after reading into MATLAB. The equivalent image is then shown using PCA, which calculates the Eigen vectors with the minimum equivalent distance for the input gesture. After looking up the input image in the database, it provides the corresponding hand gesture and alphabet. The Computation may differ highly based on the type of background in images. Conclusion: From the above-mentioned information, it can be concluded that there are several solutions proposed to help the mute people communicate. The two main types of solutions offered are glove based and computer vision based. While it is a breakthrough that signs can now be converted to speech or text by using various machine learning and deep learning techniques, there are still some challenges to overcome. Some of the most common disadvantages found are that most of the solutions work well with just numbers and letters. The person who is signing would have to spell out all the words to be able to communicate a sentence. This is not a way of realistic communication. Moreover, a sophisticated design has not been developed yet for the glove based approach. It would be very inconvenient for the person to wear a glove with a microcontroller placed on top of it, as it would neither be comfortable nor portable. Thus, we propose a solution where we build a mobile app that can convert sign language to speech (in the language preference of the user) by capturing the signs via camera and then identifying the signs real time by using deep learning techniques. This will ensure a more realistic communication between the mute and non-mute people and could also be easily used by people, since it would be an app.

## **References:**

1. Himanshu Gupta, Aniruddh Ramjiwal, Jasmin T. Jose: Vision Based Approach to Sign Language Recognition. In: International Journal of Advances in Applied Sciences (IJAAS), Vol. 7, No. 2, June 2018

2. Marek Hruží, Ivan Gruber, Jakub Kanis , Matyáš Boháček , Miroslav Hlaváč, Zdenek Krňoul: One Model is Not Enough: Ensembles for Isolated Sign Language Recognition. In: PubMed Central, July 2022.
3. Akansha Tyagi, Sandhya Bansal: Hybrid FiST-CNN Approach for Feature Extraction for Vision-Based Indian Sign Language Recognition. In: The International Arab Journal of Information Technology, Vol. 19, No. 3, May 2022.
4. Walid Hasan, Nadia Naji Gabeal: Implementation Smart Gloves for Deaf and Dumb Disabled. In: International Science and Technology Journal. Volume 19. October 2019.
5. Omkar Vedak, Prasad Zavre, Abhijeet Todkar, Manoj Patil: Sign Language Interpreter using Image Processing and Machine Learning. In: International Research Journal of Engineering and Technology (IRJET), Volume: 06 Issue: 04 | Apr 2019.
6. Aman Pathak, Avinash Kumar, Priyam, Priyanshu Gupta, Gunjan Chugh: Real Time Sign Language Detection In:International Journal for Modern Trends in Science and Technology,2022
7. Yaser Saleh, Ghassan F. Issa:Arabic Sign Language Recognition through Deep Neural Networks Fine-Tuning In:International Journal of Online and Biomedical Engineering (iJOE) 16(05),2020
8. Sanjay S, Dineshkumar S, Dr. Suresh M, Vasanthakumar S, Saravana kumar K, Mohamed Rifayee Hussain Z: Development of Smart Glove for Deaf-mute People In: Nat. Volatiles & Essent. Oils Journal, 2021
9. V. Thap, J. Sunuwar and R. Pradhan:Finger Spelling Recognition for Nepali Sign Language: JCSE International Journal of Computer Sciences and EngineeringVolume-3, Issue-6 ,2019
10. Osama R. Shahin, Ahmed I. Taloba and Rady El Rwelli: Gesture based Arabic Sign Language Recognition for Impaired People based on Convolution Neural Network: (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12, No. 12, 2021

11. R. Sruthi, B. Venkateswara Rao, P. Nagapraballika, G. Harikrishna and K. Narendra Babu: VISION BASED SIGN LANGUAGE BY USING MATLAB: International Research Journal of Engineering and Technology (IRJET), Volume: 05 Issue: 03 | Mar-2018
12. Salakapuri Rakesh, Avinash Bharadhwaj and E Sree Harsha: Sign Language Recognition using Convolutional Neural Network
13. Pooja S Bhore , Neha D Londhe , Sonal Narsale , Prachi S Patel , Diksha Thanambir: Smart Gloves for Deaf and Dumb Students: International Journal of Progressive Research in Science and Engineering Volume-1, Issue-8, November-2020
14. Zhenxing Zhou, Vincent W.L. Tam, Edmund Y. Lam: A Cross-Attention BERT-Based Framework for Continuous Sign Language Recognition In: IEEE, Volume 29,2022
15. Mohammedali, A. H., Abbas, H. H., & Shahadi, H. I. (2022). Real-time sign language recognition system. International Journal of Health Sciences, 6(S4), 10384–10407.
16. Dr. A. Ravi Kumar, Thakur Bhavana, Peekola Meghana Sri: A Deep Neural Framework for Continuous Sign Language Recognition by Iterative Training. In: Journal of Algebraic Statistics, Volume 13, No. 3, 2022, p. 4574-4584
17. Eman Aldhahri, Reem Aljuhani, Aseel Alfaidi, Bushra Alshehri, Hajer Alwadei, Nahla Aljojo, Areej Alshutayri & Abdulwahab Almazroi: Arabic Sign Language Recognition Using Convolutional Neural Network and MobileNet, In: Arabian Journal for Science and Engineering (2022)
18. Myron Darrel Montefalcon, Jay Rhald Padilla & Ramon Rodriguez: Filipino Sign Language Recognition Using Long Short-Term Memory and Residual Network Architecture, In: Springer, 2022
19. Ladan Khosravani pour, Ali Farrokhi: Language Recognition by Convolutional Neural Networks, In: Scientia Iranica, 2022

20. Archana Ghotkar, Atharva Gokhale & Durvesh Malpure: Real-Time Left- and Right-Hand Detection for Sign Language Recognition, In: Springer 2022
21. Barathi Subramanian, Bekhzod Olimov , Shraddha M. Naik , Sangchul Kim , Kil-Houm Park & Jeonghong Kim: An integrated mediapipe-optimized GRU model for Indian sign language recognition. In: Springer 2022
22. Boban Joksimoski, Eftim Zdravevski, Petre Lameski, Ivan Miguel Pires, Francisco José Melero, Tomás Puebla: Technological Solutions for Sign Language Recognition: A Scoping Review of Research Trends, Challenges, and Opportunities. In: IEEE 2022
23. Jerry John & Bismin V. Sherif: Hand Landmark-Based Sign Language Recognition Using Deep Learning, In: Springer 2022
24. Razieh Rastgoo, Kourosh Kiani & Sergio Escalera: Real-time isolated hand sign language recognition using deep networks and SVD, In: Springer 2022
25. Nigus Kefyalew Tamiru, Menore Tekeba & Ayodeji Olalekan Salau: Recognition of Amharic sign language with Amharic alphabet signs using ANN and SVM In: Springer 2022
26. Yunus Can Bilge, Ramazan Gokberk Cinbis, Nazli Ikizler-Cinbis: Towards Zero-shot Sign Language Recognition, In: IEEE 2022
27. Sakshi Sharma, Sukhwinder Singh: Recognition of Indian Sign Language (ISL) Using Deep Learning Model, In: Springer 2021
28. Nabah Inamdar, Zaid Inamdar, Sohel sheikh, Prof. Sunil A. Sushir: A Survey Paper on Sign Language Recognition, In: International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 10 Issue IV Apr 2022
29. V.Gowtham, S. Karthick, T. Karthikeyan, Dr. P. Elayaraja: Sign Language Recognition using Machine Intelligence for Hearing Impairment Person, In: IJERT 2022, Volume 10 Issue 9

30. Tanseem N. Abu-Jamie & Samy S. Abu-Naser: Classification of Sign-Language Using MobileNet - Deep Learning, In: International Journal of Academic Information Systems Research (IJAIR) 6 (7):29-40 (2022)

## BIBLIOGRAPHY

- [1] Himanshu Gupta, Aniruddh Ramjiwal, Jasmin T. Jose: Vision Based Approach to Sign Language Recognition. In: International Journal of Advances in Applied Sciences (IJAAS), Vol. 7, No. 2, June 2018
- [2] Marek Hruz, Ivan Gruber , Jakub Kanis , Matyáš Boháček , Miroslav Hlaváč, Zdenek Krňoul: One Model is Not Enough: Ensembles for Isolated Sign Language Recognition. In: PubMed Central, July 2022.
- [3] Akansha Tyagi, Sandhya Bansal: Hybrid FiST-CNN Approach for Feature Extraction for Vision-Based Indian Sign Language Recognition. In: The International Arab Journal of Information Technology, Vol. 19, No. 3, May 2022.
- [4] Walid Hasan, Nadia Naji Gabeal: Implementation Smart Gloves for Deaf and Dumb Disabled. In: International Science and Technology Journal. Volume 19. October 2019.
- [5] Omkar Vedak, Prasad Zavre, Abhijeet Todkar, Manoj Patil: Sign Language Interpreter using Image Processing and Machine Learning. In: International Research Journal of Engineering and Technology (IRJET), Volume: 06 Issue: 04 | Apr 2019.
- [6] Aman Pathak, Avinash Kumar, Priyam, Priyanshu Gupta, Gunjan Chugh: Real Time Sign Language Detection In:International Journal for Modern Trends in Science and Technology,2022
- [7] Yaser Saleh, Ghassan F. Issa:Arabic Sign Language Recognition through Deep Neural Networks Fine-Tuning In:International Journal of Online and Biomedical Engineering (iJOE) 16(05),2020
- [8] Sanjay S, Dineshkumar S, Dr. Suresh M, Vasanthakumar S, Saravana kumar K, Mohamed Rifayee Hussain Z:Development of Smart Glove for Deaf-mute People In: Nat. Volatiles & Essent. Oils Journal, 2021

- [9] V. Thap, J. Sunuwar and R. Pradhan: Finger Spelling Recognition for Nepali Sign Language: JCSE International Journal of Computer Sciences and Engineering Volume-3, Issue-6, 2019
- [10] Osama R. Shahin, Ahmed I. Taloba and Rady El Rwelli: Gesture based Arabic Sign Language Recognition for Impaired People based on Convolution Neural Network: (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12, No. 12, 2021
- [11] R. Sruthi, B. Venkateswara Rao, P. Nagapravallika, G. Harikrishna and K. Narendra Babu: VISION BASED SIGN LANGUAGE BY USING MATLAB: International Research Journal of Engineering and Technology (IRJET), Volume: 05 Issue: 03 | Mar-2018
- [12] Salakapuri Rakesh, Avinash Bharadhwaj and E Sree Harsha : Sign Language Recognition using Convolutional Neural Network
- [13] Pooja S Bhore, Neha D Londhe, Sonal Narsale, Prachi S Patel, Diksha Thanambir: Smart Gloves for Deaf and Dumb Students: International Journal of Progressive Research in Science and Engineering Volume-1, Issue-8, November-2020
- [14] Zhenxing Zhou, Vincent W.L. Tam, Edmund Y. Lam: A Cross-Attention BERT-Based Framework for Continuous Sign Language Recognition In: IEEE, Volume 29, 2022
- [15] Mohammedali, A. H., Abbas, H. H., & Shahadi, H. I. (2022). Real-time sign language recognition system. International Journal of Health Sciences, 6(S4), 10384–10407.
- [16] Dr. A. Ravi Kumar, Thakur Bhavana, Peekola Meghana Sri: A Deep Neural Framework for Continuous Sign Language Recognition by Iterative Training. In: Journal of Algebraic Statistics, Volume 13, No. 3, 2022, p. 4574-4584
- [17] Eman Aldhahri, Reem Aljuhani, Aseel Alfaidi, Bushra Alshehri, Hajer Alwadei, Nahla Aljojo, Areej Alshutayri & Abdulwahab Almazroi: Arabic Sign Language

Recognition Using Convolutional Neural Network and MobileNet, In: Arabian Journal for Science and Engineering (2022)

[18] Myron Darrel Montefalcon, Jay Rhald Padilla & Ramon Rodriguez: Filipino Sign Language Recognition Using Long Short-Term Memory and Residual Network Architecture, In: Springer, 2022

[19] Ladan Khosravani pour, Ali Farrokhi: Language Recognition By Convolutional Neural Networks, In: Scientia Iranica, 2022

[20] Archana Ghotkar, Atharva Gokhale & Durvesh Malpure: Real-Time Left- and Right-Hand Detection for Sign Language Recognition, In: Springer 2022

[21] Barathi Subramanian , Bekhzod Olimov , Shraddha M. Naik , Sangchul Kim , Kil-Houm Park & Jeonghong Kim: An integrated mediapipe-optimized GRU model for Indian sign language recognition. In: Springer 2022

[22] Boban Joksimoski, Eftim Zdravevski, Petre Lameski, Ivan Miguel Pires, Francisco José Melero, Tomás Puebla: Technological Solutions for Sign Language Recognition: A Scoping Review of Research Trends, Challenges, and Opportunities. In: IEEE 2022

[23] Jerry John & Bismin V. Sherif: Hand Landmark-Based Sign Language Recognition Using Deep Learning, In: Springer 2022

[24] Razieh Rastgoo, Kourosh Kiani & Sergio Escalera: Real-time isolated hand sign language recognition using deep networks and SVD, In: Springer 2022

[25] Nigus Kefyalew Tamiru, Menore Tekeba & Ayodeji Olalekan Salau: Recognition of Amharic sign language with Amharic alphabet signs using ANN and SVM In: Springer 2022

[26] Yunus Can Bilge, Ramazan Gokberk Cinbis, Nazli Ikizler-Cinbis: Towards Zero-shot Sign Language Recognition, In: IEEE 2022

[27] Sakshi Sharma, Sukhwinder Singh: Recognition of Indian Sign Language (ISL) Using Deep Learning Model, In: Springer 2021



- [28] Nabah Inamdar, Zaid Inamdar, Sohel sheikh, Prof. Sunil A. Sushir: A Survey Paper on Sign Language Recognition, In: International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 10 Issue IV Apr 2022
- [29] V.Gowtham, S. Karthick, T. Karthikeyan, Dr. P. Elayaraja: Sign Language Recognition using Machine Intelligence for Hearing Impairment Person, In: IJERT 2022, Volume 10 Issue 9
- [30] Tanseem N. Abu-Jamie & Samy S. Abu-Naser: Classification of Sign-Language Using MobileNet - Deep Learning, In: International Journal of Academic Information Systems Research (IJASIR) 6 (7):29-40 (2022)
- [31] Ketan Gomase, Akshata Dhanawade, Prasad Gurav, Sandesh Lokare: Sign Language Recognition using Mediapipe: International Research Journal of Engineering and Technology (IRJET) Volume: 09 Issue: 01 | Jan 2022
- [32] Sundar B , Bagyammal T: American Sign Language Recognition for Alphabets Using MediaPipe and LSTM: Elsevier, Science Direct, Procedia Computer Science215 (2022) 642–651
- [33] Subramanian, B., Olimov, B., Naik, S.M. et al. An integrated mediapipe-optimized GRU model for Indian sign language recognition. Sci Rep 12, 11964 (2022).

## APPENDIX

### Base Paper:

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 09 Issue: 01 | Jan 2022 www.irjet.net p-ISSN: 2395-0072 © 2021, IRJET | Impact Factor value: 7.529 | ISO 9001:2008 Certified Journal | Page 744

### Sign Language Recognition using Mediapipe

Ketan Gomase<sup>1</sup>, Akshata Dhanawade<sup>2</sup>, Prasad Gurav<sup>3</sup>, Sandesh Lokare<sup>4</sup>  
<sup>1,2,3,4</sup>Student, Department of Electronics & Telecommunication Engineering, Atharva College of Engineering, Mumbai, Maharashtra, India

----- \*\*\* -----

**Abstract** - Lack of speech is considered to be a real disability. People with these disabilities use a variety of methods to communicate with others, there are many forms available for their communication, one of the most common forms of communication as well as sign language. Sign language is used by deaf and hard of hearing people to share information with their community and others. Electronic recognition of sign language deals from signalling to touch and continues until text / speech production. Touch gestures can be classified as permanent and flexible. Steps in recognizing sign language are described in this study. Data acquisition, pre-processing and modification of data, feature extraction, segmentation and obtained results are assessed. Future research guides in this area are also recommended.

**Key Words:** Mediapipe, Sign language recognition [SLR], KNN, Hand Solution, Computer Interaction with Humans.

### 1. INTRODUCTION

Sign language using sign language is designed for the deaf community, which can be used as a means of communication between friends and family of the deaf and the deaf. Sign Language Recognition is one of the fastest growing and challenging areas of

research today. Many new strategies have been developed recently in this field. In this project, we will develop a program to translate sign language into OpenCV. It outlines a method that recognizes American Sign Language (ASL) and translates it into standard text.

### 1.1 Motivation and Background

Sign Language Recognition strive to develop algorithms and methods for accurately identifying the sequences of symbols produced and understanding their meaning. Many SLR methods mistreat the problem as Gesture Recognition (GR). Therefore, research has so far focused on identifying the positive characteristics and methods of differentiation in order to properly label a given signal from a set of potential indicators. However, sign language is more than just a collection of well-articulated gestures.

### 1.2 What is Gesture recognition?

Gesture recognition is a subject in computer science as well language technology for the purpose of translating a person touch with mathematical algorithms. The subdiscipline of computer vision. Gestures can come from any body movement or position but usually appears on the face or hand. The current focus in the field includes emotional recognition from facial and hand touch recognition. Users can use simple touch to control or interact with devices without the touch touching them. Many methods have been developed using cameras and computer vision algorithms to translate the signal language.

### 1.3 Sign Language

Sign languages [also known as sign languages] are languages that use visual cues to convey meaning. Sign languages are expressed in sign language as well as non-sign language objects. Sign languages are complete natural languages with their own grammar and dictionary. Sign languages are not universal and are not widely understood, although there are some striking similarities between sign languages. Linguists consider both spoken and signed communications to be natural forms of language, meaning that they both evolved into a vague aging process, one that lasted

longer and evolved over time without careful planning. Sign language should not be confused with body language, a form of communication without voice.

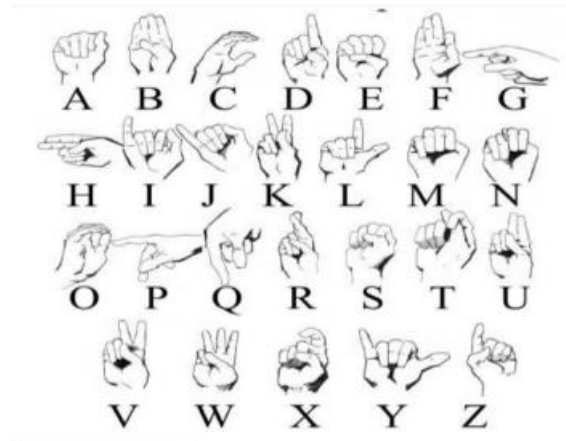


Figure 1: Sign Languages

#### 1.4 Mediapipe Framework

Mediapipe Hands is a reliable hand and finger tracking device solution. It uses machine learning (ML) to understand 21 3D local hand marks from just one frame. Although modern methods depend largely on the powerful desktop locations for discovery, our approach benefits real-time performance on mobile phones, even scales to many hands. We hope to give you this handy idea working on extensive research and development society will result in cases of to promote new applications and new research methods. MediapipeHands uses an integrated ML pipe of the many models working together: The palm detection model which works on the full image and returns the directdirected hand binding box. Hand gesture model applicable to image-cut region defined by a palm detector once returns 3D hand key points with high reliability. This strategy is similar to the one used in our Mediapipe Face Mesh solution, using a face detector and a face detector a landmark model.

#### 1.5 Objective

The objective of this project was to create a neural network that could distinguish between the American Sign Language (ASL) alphabet characters, if a handwritten signature is provided. This project is the first step in creating a potential sign language

translator, which can take communication in sign language and translate it into written and oral language. Such a translator can greatly reduce the barrier between many deaf and hard of hearing people so that they can better communicate with others in their daily activities.

## 1.6 Summary

Improving sign language application for the deaf can be it is very important, as they will be able to easily communicate with them and those who do not understand sign language. Our program aims to take a basic step to close the connection the gap between the common people and the deaf and dumb using sign language. The main focus of this work is creativity a vision-based system for identifying spelled characters ASL. Reason for choosing a vision-based system has to do with the fact that it provides a simple and accurate way how to communicate between a person and a computer. In this report, the various stages of 36:26 were considered sections of the English Alphabets (a to z) We have used Google's Mediapipe Framework Mediapipe Solutions has improved hand recognition model and can find 21 3D Landmarks of Palm

## 3.LITERATURE SURVEY

In recent years, much research has been done on sign language recognition. This recognition technology is divided into two categories: -

### 2.1 Vision Based Approach

This method takes pictures on camera as touch data. The vision-based approach focuses heavily on touch-captured images and brings out the main and recognizable feature. Colour belts were used at the beginning of the vision-based approach. The main disadvantage of this method was the standard colour to be applied to the fingers. Then use bare hands instead of coloured ribbons. This creates a challenging problem as these systems require background, uninterrupted lighting, personal frames and a camera to achieve real-time performance. In addition, such systems must be developed to meet the requirements, including accuracy and robustness.



Figure 2: Sample of Vision Based Technique

Theoretical analysis is based on how people perceive information about their environment, yet it is probably the most difficult to use effectively. Several different methods have been tested so far. The first is to build a three-dimensional human hand model. The model is compared to hand images with one or more cameras, and the parameters corresponding to the shape of the palm and the combined angles are estimated. These parameters are then used to create the touch phase. The second is to take a picture using the camera and extract certain features and those features are used as input in the partition algorithm to separate.

### 3. IMPLEMENTATION METHODOLOGY

Mediapipe Hands uses a Machine Learning Pipeline that integrates multiple co-working models: A palm-type acquisition model that works in a complete image and returns a fixed hand-held bounding box. A handwriting model that works with a cropped image location defined by a palm detector and restores 3D reliable key points. So, to make a Web site we have to photograph at least 25-30 images per mark and with this model we can get 21 hand points. i.e., links  $[x, y, z]$ .  $x$  and  $y$  are common to say  $[0.0, 1.0]$  the width and height of the image respectively. The  $z$  represents the depth of the landmark and the depth of the arm at the root, and the smaller the value the closer the camera becomes. After making the Website can predict the sign with the help of the Appropriate Model. We will use the KNN algorithm.

#### 3.1 Hardware & Software Requirement

- 1) Windows computer or Linux, Python installed and Libraries.
- 2) CMOS sensor (Webcam) 3) Hand Touch for Visibility

Computer Software We Used to Recognize Project Signature Recognition:

- 1) Python Installed Windows Os or Linux Os Machine.
- 2) CPU - Intel core i5 9th Gen.
- 3) GPU - Nvidia GTX 1050 Ti.
- 4) 720p60 Web Camera
- 5) Python 3.8.6 and IDE like VS, Spyder etc.
- 6) Libraries: OpenCV, TensorFlow, Keras, Mediapipe and many more basic
- 7) KNN (The closest neighbours) from the Sklearn Library of Python.

### 3.2 Result

This sign language receiver can detect hand and produce coordinators and will be able to recognize letters (A-Z). All signs will appear in real time.





- [2] Umang Patel & Aarti G. Ambekar "Moment based sign language recognition for Indian Languages" "2017 Third International Conference on Computing, Communication, Control and Automation (ICCUBEA).
- [3] Lih-Jen kau, Wan-Lin Su, Pei-Ju Yu, Sin-Jhan Wei "A Realtime Portable Sign Language Translation System" Department of Electronic Engineering, National Taipei University of Technology No.1, Sec. 3, Chung-Hsiao E. Rd., Taipei 10608, Taiwan, R.O.C.
- [4] Obtaining hand gesture parameters using Image Processing by Alisha Pradhan and B.B.V.L. Deepak ,2015 International Conference on Smart Technology and Management (ICSTM).
- [5] Vision based sign language translation device (conference paper: February 2013 by Yellapu Madhuri and Anburajan Mariamichael).
- [6] K-nearest correlated neighbour classification for Indian sign language gesture recognition using feature extraction (by Bhumika Gupta, Pushkar Shukla and Ankush Mittal).
- [7] Vikram Sharma M, Virtual Talk for deaf, mute, blind and normal humans, Texas Instruments India Educator's conference, 2013.
- [8] Hand in Hand: Automatic Sign Language to English Translation, by Daniel Stein, Philippe Dreuw, Hermann Ney and Sara Morrissey, Andy Way.
- [9] Er. Aditi Kalsh, Dr. N.S. Garewal "Sign Language Recognition System" International Journal of Computational Engineering Research 2013
- [10] Prakash B Gaikwad, Dr. V.K. Bairagi, "Hand Gesture Recognition for Dumb People using Indian Sign Language", International Journal of Advanced Research in Computer Science and Software Engineering, pp:193-194, 2014.
- [11] [https://en.wikipedia.org/wiki/Sign\\_language](https://en.wikipedia.org/wiki/Sign_language)

## Activation Functions:

Sigmoid activation function:

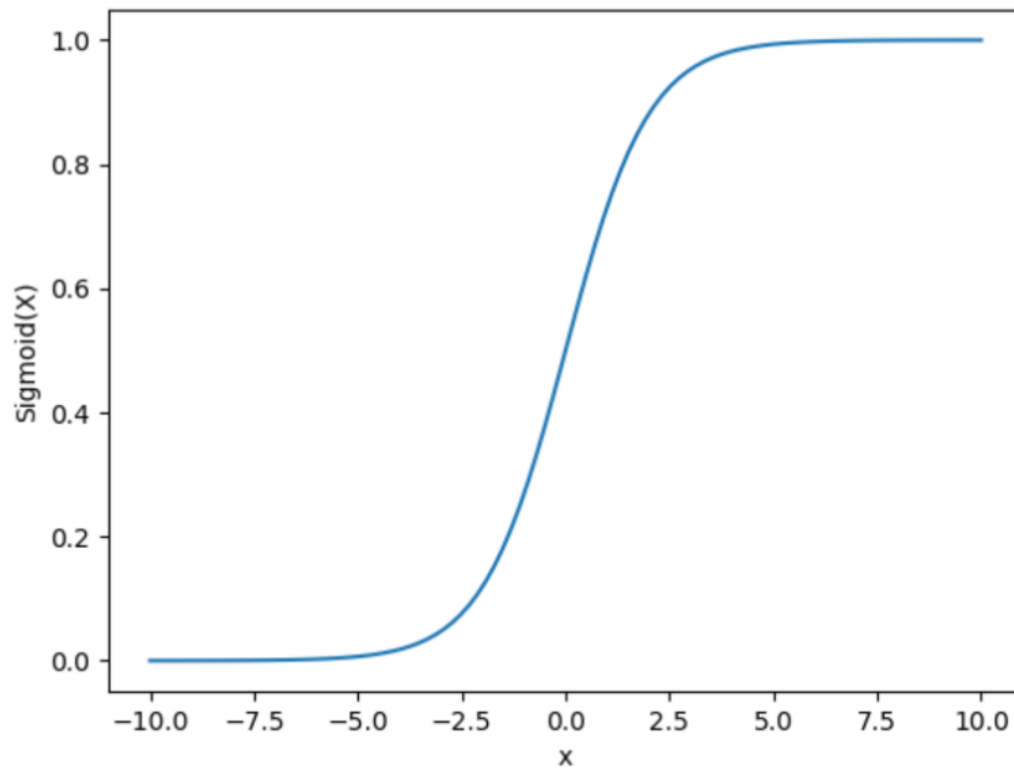


Fig 10.1: Sigmoid Activation Function

$$S(x) = \frac{1}{1 + e^{-x}}$$

$S(x)$  = sigmoid function

$e$  = Euler's number

Fig 10.2: Sigmoid activation function formula

The sigmoid activation function is an 'S' shaped graph and it is non-linear in nature. It is usually used in the output layer of the neural network since it provides the output

between 0 and 1. It is thus used in binary classification problems. If the output of the function is greater than 0.5, then it is classified as 1, else 0.

Tanh activation function:

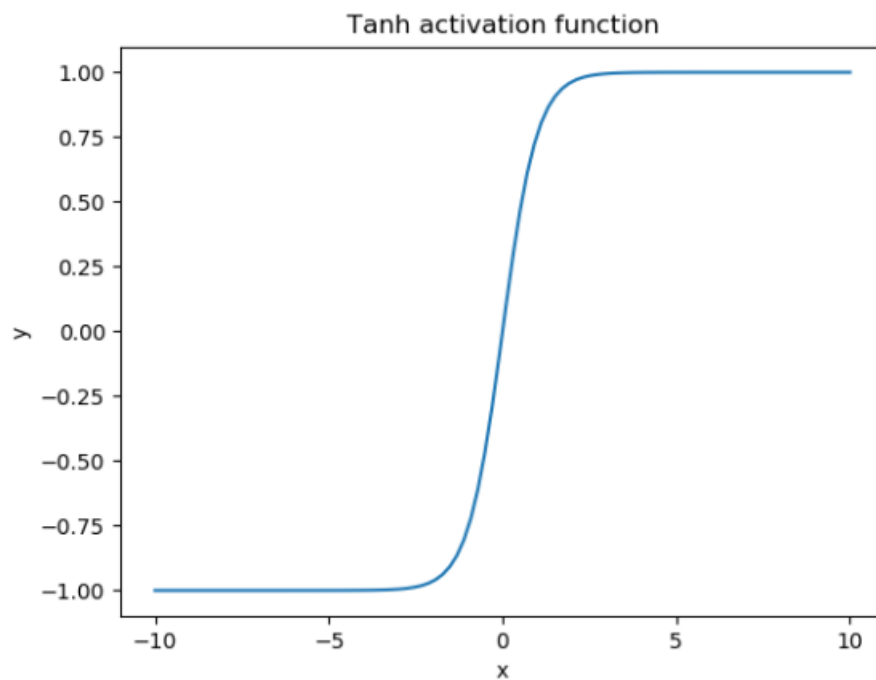


Fig 10.3: Tanh activation function

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

Fig 10.4: Tanh activation function formula

The tanh (Tangent Hyperbolic) activation function is also a non-linear function. It provides an output between -1 and 1 and is thus used in the hidden layers of the neural network. It helps in centering the data since it brings the mean of the hidden layers closer to 0. This makes learning easier for the next layers.

ReLU activation function:

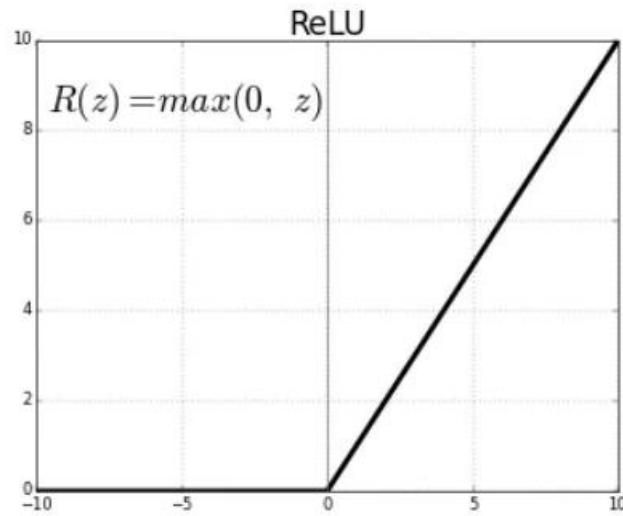


Fig 10.5: ReLU activation function

ReLU (Rectified Linear Unit) activation function is a non-linear activation function whose equation is  $\max(0, x)$ . If  $x$  is positive, then the output is  $x$ , else it is 0. It is the most widely used activation function in the hidden layers of neural network. It is also computationally less expensive since it uses simpler mathematical operations when compared to the other activation functions.

### Optimizers:

SGD with momentum:

$$W_{t+1} = W_t - V_t$$

$$\text{here, } V_t = \beta * V_{t-1} + \eta \Delta W_t$$

Fig 10.6: SGD with momentum formula

$W_{t+1}$  is the updated weight,  $W_t$  is the current weight,  $V_t$  is the current velocity,  $V_{t-1}$  is the velocity at the previous step,  $\beta$  is the momentum constant or decay factor (whose value is usually between 0 and 1).  $\beta$  value determines how much do the past

velocities contribute towards the momentum. The older the velocities, the lesser its impact on momentum. Newer the velocities, more the impact. Common values used for beta are 0.5, 0.9 and 0.99. Neta is the learning rate. The core principle behind this optimizer is that when the previous gradients point the algorithm to move towards the same direction, then the algorithm becomes more confident and moves towards the minima at a much higher speed. This allows the algorithm to reach the global minima faster. The history of previous velocities and the gradient at that point together is used to determine to jump to the next point. It decreases the unstability of moving towards the minima. It overcomes the local minima problem, but due to the oscillations, it takes time to converge at the local minima.

NAG (Nesterov Accelerated Gradient):

$$w_{t+1} = w_t - \text{update}_t$$

$$\text{update}_t = \gamma \cdot \text{update}_{t-1} + \eta \nabla w_{\text{look\_ahead}}$$

$$w_{\text{look\_ahead}} = w_t - \gamma \cdot \text{update}_{t-1}$$

Fig 10.7: Formula of NAG optimizer

It was introduced to overcome the oscillation problem of momentum. It enhances the momentum algorithm to dampen or reduce the number of oscillations to reach the global minima. This is done by first making the jump by taking into account the history of previous velocities only. Then from this new location (the look ahead point), the gradient of the point is calculated and based on this gradient another jump is taken. A single jump is not taken by considering the velocity and gradient together. Separate jumps are taken (first by considering the velocity history and the second after the gradient at the look ahead point). The potential problem with this algorithm is the local minima problem. Since we are decreasing the no. of oscillations and thus the momentum, the algorithm may get stuck in a local minima problem.

## Adagrad

$$W_t = W_{t-1} - \eta'_t \frac{\partial L}{\partial W(t-1)}$$
$$\eta'_t = \frac{\eta}{\text{sqrt}(\alpha_t + \epsilon)}$$

Fig 10.8: formula of Adagrad optimizer

$W_t$  is the updated weight.  $W_{t-1}$  is the previous weight. Alpha represents the different learning rates at each iteration. Epsilon is a small number used to avoid division by zero. Adagrad makes use of different learning for different parameters or features. Sparse features require a higher learning rate when compared to dense features. This allows it to reach the global minima faster. The biggest disadvantage is that this algorithm will not be able to reach the global minima, no matter how many epochs. It only reaches near it. According to the mathematical formula, the learning rate gets divided by a value. This decreases the learning rate and decreases the updates as well. Thus, it never converges to the global minima and thus adagrad is mostly not used in neural networks.