International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V73I3P102

Volume 73 Issue 3, 13-21, March 2025

Original Article

SpeakEasy: A Tool for People with Communication Disabilities

Vijay Shelake[1], Sujata Deshmukh[2], Max Gonsalves[3], Vedant Chawardol[4], Saville Dsilva[5], Ivan Dsouza[6]

[1,2,3,4,5,6]*Department of Computer Engineering, Fr. Conceicao Rodrigues College of Engineering, University of Mumbai, Maharashtra, India.*

[1]*Corresponding Author : vijaysnew12@gmail.com*

Received: 23 July 2024 Revised: 06 February 2025 Accepted: 13 February 2025 Published: 28 March 2025

Abstract - *In this era of digitalization, where everyone is connected, people with communication disabilities may find it difficult to fully participate and engage in interactions. This paper aims to deliver an AI-powered communication system that fulfils the needs of individuals with communication disabilities. It provides the facility to convert Indian Sign Language (ISL) to text and speech in a regional language, providing inclusivity. It will empower people with verbal communication disabilities by breaking the barriers put up by sign language and help them express themselves to others without any restraint. The ultimate purpose of this tool is to create a more inclusive society wherein communication barriers are eliminated for individuals with disabilities. The MediaPipe Holistic Library is used to map key points and extract data for prediction to facilitate the conversion of sign language to text. A sequential framework along with an LSTM and Dense layer is incorporated to identify the signs by action recognition. Once a prediction has been made, a text prompt and audio of the predicted text are played. Through this chain of processes, it is possible to develop a multi-feature Sign Language Recognition system.*

Keywords - *Communication disabilities, Indian Sign Language, MediaPipe, Neural network, Long Short-Term Memory (LSTM).*

1. Introduction

Communication is an important part of our lives. It enables us to express our thoughts and opinions, helping us integrate into society since not every individual possesses the ability to hear and speak, which makes it difficult and challenging for these individuals to communicate effectively and fit into societal norms. Individuals with hearing and vision impairments can communicate with one another thanks to sign language recognition technology. Without this technology, they have significant difficulties in daily life since it inhibits their capacity to express themselves, understand information, and engage with others. According to the 2011 Census of India, approximately 26.8 million people (2.21% of the total population) had some form of disability. Among them, 5.07 million (18.9%) had hearing disabilities, while 1.98 million (7.4%) had speech disabilities. The development and enhancement of sign language recognition technology can enable individuals to live more independent and satisfying lives, promoting inclusion and bridging communication gaps in society. Earlier tools of similar scope were all mainly built around the American Sign Language convention and then translated into English. There were a few works that were making use of the Indian Sign Language convention for translation purposes to English. These implemented learning through image data for each alphabet [1-5]. Our contributions include, rather than using image data for training and testing purposes, utilizing keypoint data extracted using MediaPipe Holistic. An emphasis is given priority to more conversational actions for which a brand-new dataset was created. After translation, the result will be displayed in Marathi and played through the Text-to-Speech. Furthermore, an option for two-way communication is available wherein users can select an action which will be played via an embedded video.

2. Literature Review

The review of literature with the state-of-the-art mechanisms to extract the data and the prediction model to handle communication disabilities are discussed as follows:

In this study [1], it was found that Convolutional Neural Networks (CNNs) use batch normalization (batch norm) to increase training speed and stability. In this paper, some ideas from adaptive filter theory are employed to demonstrate how BatchNorm effectively controls the eigenvalues of input autocorrelation matrices, thereby enhancing the rate of convergence and stability. It also brings out the alignment between BatchNorm and Normalized Least Mean Square (NLMS) optimization principles, especially during initial training steps, which can be beneficial for understanding modern neural network operations at a deeper level. In the work [2], the hand movement data were collected using MediaPipe, where 21 points on the palm were extracted in

terms of coordinates. These numerical values were transformed into a NumPy array and introduced as input to an LSTM model for detecting sign language- represented by a string showing the detected alphabet ('A' – 'Z'). Additionally, a larger dataset with more variety of samples, cross-validation application and conducting statistical analysis for validating the accuracy and reliability of the model is aimed at enhancing scientific rigour by these researchers.

This study [3] explores the effectiveness of deep learning, a neural network-inspired machine learning branch in numerous applications, especially for working with large datasets, which has made it popular. This work focuses on classifying the MNIST dataset using TensorFlow and examining the impact of various activation functions such as eLU, sigmoid, soft plus, softsign, tanh, and ReLU within Convolutional Neural Networks (CNNs). It is revealed by the findings that ReLU gives better classification accuracy rates and provides insights into tuning deep learning models.

In this paper [4], the information about complications in gesture recognition and low accuracy are some of the problems that dynamic signal language recognition faces in hearing-impaired communities where this form of communication is crucial. This research combines MediaPipe and RNN models (GRU, LSTM, Bi-LSTM) to tackle the frame dependency problem using DSL10-Dataset. The results from experiments show a more than 99% accuracy rate, which shows that this methodology actually moves towards better dynamic sign language recognition technology.

This review by [5] discusses improvements in machine learning in automatic sign language recognition for the period between January 2019 and March 2022. The results showed a common use of LSTM and CNN models, while Transformer architectures are being proposed for continuous sign language recognition. Most feature extractors depend on CNNs or skeleton representations. The paper also emphasizes that research papers on the Deaf community should include encouraging words.

The paper [6] introduces a trainable neural network designed for recognizing isolated sign language. This system consists of three networks: Sign Recognition Network (SRN), Accumulative Motion Network (AMN) and Dynamic Motion Network (DMN). Here, DMN learns the spatiotemporal information of sign gestures by extracting key postures from sign videos. At the same time, AMN encodes motion into a single image using the accumulative video motion technique, and then these features are then fused and fed into the SRN for learning and classification. This paper [7] explains how sign language recognition uses LSTM. The LSTM model works similarly as recurrent neural networks. Sign language gestures are characterized by the LSTM neural network. A deep learning neural network is like a human brain in that it combines inputs, weights and biases to accomplish tasks such as object identification and classification. Deep learning algorithms work best when trained on huge amounts of data. The system's performance was assessed based on accuracy, precision and recall. The highest accuracy of 90-96% in training was observed for the proposed system that could classify seven gestures.

Research [8] shows how to use Google's Text-to-Speech model to convert the given text into speech in Python programming language. Using Google's text-to-speech API, the conversion of sign language to audio was implemented successfully. This is achieved by using a text-to-speech interpretation device, which uses a data processor to precisely comprehend the language in which a text was inputted and convert it to an audio file that can be played through a speaker.

To conclude, this research successfully demonstrates how text-to-speech conversion occurs using Google's API in Python, which helped add text to speech for recognized sign language words in our work. This paper [9] tells us about implementing Machine Learning solutions and how they are addressed using the MediaPipe framework developed by Google. MediaPipe uses pre-trained models in TensorFlow OpenCV to manipulate video and FFmpeg to handle audio data; it is also available for Android, iOS, C++, Python, and JavaScript.

In their research [10], the authors aimed to address communication challenges faced by individuals with hearing loss through a vision-based system for sign language recognition. They utilized the Media Pipe Holistic approach to capture hand movements, incorporating face, pose, and hand models. The suggested system utilized an adapted Long Short-Term Memory (LSTM) model for uninterrupted sign language sequences, resulting in a classification accuracy of 98%. The authors highlighted the system's potential applications in human-computer interaction, emphasizing the need to bridge the communication gap for individuals with hearing and speech impairments. The study also outlined the methodology, data collection using Media Pipe, and the integration of the YOLOv5 Algorithm for efficient communication translation.

The research [11] puts forward a real-time Sign Language (Indian Sign Language) recognition system for multiple dynamic signals using the framework offered by MediaPipe Holistic and an LSTM model. This study involves training an LSTM model to differentiate between different signs of Indian Sign Language using a dataset created of 24 dynamic gesture signs. The author used a pre-trained Holistic model of the MediaPipe framework to create the dataset as a feature extractor. Based on the insights provided by different research works in the literature, MediaPipe was chosen to extract key point data from the OpenCV feed. Sequential to build the neural network was chosen as the model along with LSTM and Dense for the prediction system. A frame rate of 30 was chosen from the same.

3. Issues and Findings

This section discusses the issues and findings of the state-of-the-art research related to people with communication disabilities and Indian sign language.

3.1. Lack of Knowledge of Indian Sign Language (ISL)

How does the lack of knowledge of Indian Sign Language (ISL) affect the system's inclusivity?

When people ignore or fail to support ISL, they unwittingly prevent deaf and hard-of-hearing people from enjoying the full benefits that these languages can offer. Many hearing-impaired individuals in India use ISL as their means of communication. When a system does not consider ISL, it results in this category being left out of numerous services requiring them to be active participants; therefore, they have no way through which they can get what they need, thus making them disadvantaged citizens who are against equal opportunities.

The solution is for an inclusive system that entails comprehensive support for ISL, such as training service providers on Indian Sign Language, developing resources in the language, and creating all accessible communication channels for users concerning sign language. For example, working closely with organizations specializing in deaf education and sign language interpretation will help reach this goal and buy communication devices like video relay services that enable one to communicate through an interpreter using a mobile device.

3.2. Lack of Translation Services in Local Languages

How does the absence of translation services in local languages affect the system's effectiveness?

The absence of translation services in various local languages limits the system's reach and influence. This diversity is found in India's linguistic population, and the absence of translations for their local languages means that many individuals cannot access vital information and services. In addition to this, it undermines efforts to promote multilingualism as well as cultural inclusivity. People with no fluency in major languages like Hindi or English are rendered irrelevant to important social, educational, or economic activities.

Therefore, we should invest in more robust language translation services within our systems to accommodate many local dialects to overcome this limitation. These could involve the use of native speakers of a language or linguistic experts; machine translation technology through which a database containing most commonly used phrases and terms can be established per language; community outreach programs that will help create awareness on localization concepts by training locals so they can become translators who are both accurate and culturally sensitive enough.

3.3. Absence of Real-Time Translation Mobile Application

How does the absence of a real-time translation mobile application hinder accessibility and communication?

The absence of a real-time translation mobile app makes it hard for people to communicate and access information. In today's fast-paced world, the ability to translate spoken or written language instantly is crucial for seamless interaction across different languages. Without such an application, individuals must rely on slower, often less accurate, translation methods, which can be cumbersome and inefficient. This gap in service provision limits the system's ability to respond quickly to communication needs, particularly in dynamic environments such as business meetings, emergency services and travel.

To bridge this gap, developing and deploying a real-time translation mobile application that supports multiple languages, including regional dialects, is needed while providing accurate and instant translations. Incorporating advanced technologies like artificial intelligence and natural language processing into the app can make it more efficient than before. Moreover, the app should be user-friendly so that even those with limited experience in technology can operate it well. The developers may collaborate with technology firms as well as linguists' organizations, guaranteeing reliability and productivity.

4. Research Method

The proposed system for sign language prediction was developed using a neural network architecture with a sequential approach. This approach proved to be the most effective solution due to its flexibility in model construction.

4.1. Model Architecture Details

4.1.1. Neural Network Architecture

The architecture details of our neural network model are presented in Figure 1. The model consists of multiple layers, namely Long Short-Term Memory (LSTM) layers and Dense layers. Each LSTM layer is configured to output sequences, and the Dense layers have varying output sizes.

4.1.2. Layer Details

This neural network is designed to deal with data that unfolds over time by spotting patterns in the order of events. It includes multiple LSTM layers, which are good at remembering and bringing back information over long stretches of time. These LSTM layers are stacked one after the other to slowly reveal important features in the data. Referring to Figure 1, it can be inferred that the first LSTM layer, called lstm_3, outputs sequences of vectors with 64 elements. Lstm_4 then processes these sequences and produces higher-level descriptions with 128 elements. The last LSTM layer, lstm_5, reduces the complexity of the data by condensing it to 64 elements, providing a short yet meaningful description of the input sequences.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 30, 64)            442112

 batch_normalization_5 (Bat  (None, 30, 64)            256
 chNormalization)

 lstm_4 (LSTM)               (None, 30, 128)           98816

 batch_normalization_6 (Bat  (None, 30, 128)           512
 chNormalization)

 lstm_5 (LSTM)               (None, 64)                49408

 batch_normalization_7 (Bat  (None, 64)                256
 chNormalization)

 dense_3 (Dense)             (None, 64)                4160

 batch_normalization_8 (Bat  (None, 64)                256
 chNormalization)

 dense_4 (Dense)             (None, 32)                2080

 batch_normalization_9 (Bat  (None, 32)                128
 chNormalization)

 dense_5 (Dense)             (None, 26)                858

=================================================================
Total params: 598842 (2.28 MB)
Trainable params: 598138 (2.28 MB)
Non-trainable params: 704 (2.75 KB)
_________________________________________________________________
```

Fig. 1 Layered structure of the model

Following the LSTM layers, the architecture incorporates densely connected layers (dense_3, dense_4, dense_5). These layers act as a bridge between the LSTM-derived features and the output layer, contributing to the model's ability to learn complex relationships in the data. In between the LSTM and Dense layers, there are also 5 Batch Normalization layers, which are done to enhance the stability of training, speed up convergence, provide regularisation, improve gradient flow, and boost generalization by stabilizing training, preventing overfitting and normalizing activations. The table abridges each layer's yield shapes and parameter tallies, giving bits of knowledge into the model's complexity. The entire number of trainable parameters shows the model's capacity to memorize the prepared information. These parameters are overhauled amid the preparing handle, permitting the show to adjust and capture complex designs. The non-trainable parameters, which stay settled amid preparation, may incorporate constants or parameters in non-trainable layers. This architecture is designed to effectively learn and represent sequential patterns in the input data, contributing to the model's overall performance. For the functioning of the overall detection system various steps in sequence are to be initiated and followed upon as displayed in Figure 2. this includes data collection via MediaPipe followed by the processing of this captured data from OpenCV. Then it is pushed through the recognition model consisting of the various layers (Figure 9) for a prediction.

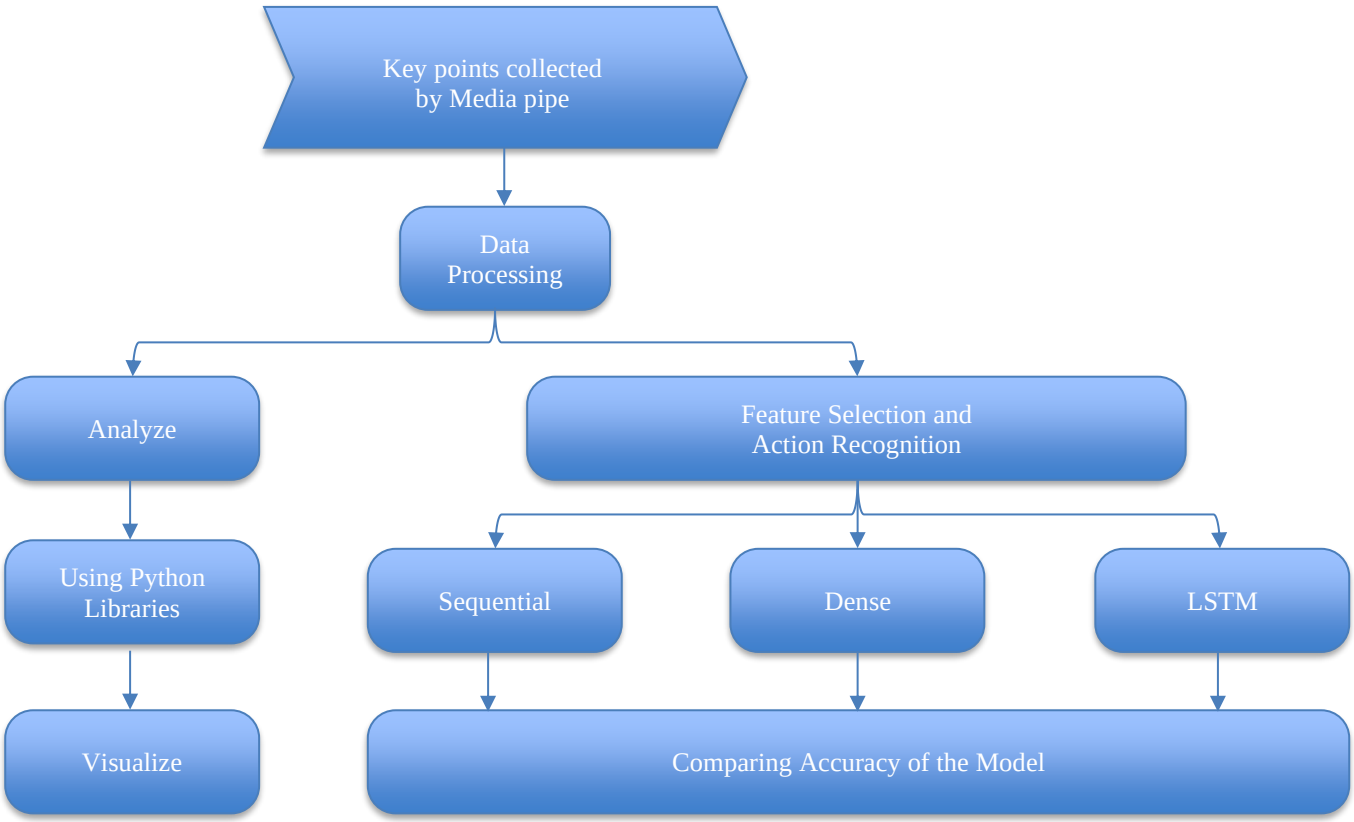

Fig. 2 Basic block diagram of the implementation of sign language recognition is done

4.1.3. Prediction Model

For the purposes of the prediction, a Sequential model type is chosen with six total layers, namely 3 LSTM (Long Short-Term Memory) layers, 3 Dense layers and 5 Batch Normalisation layers.

4.1.4. Sequential Model

A sequential model in the context of deep learning refers to a linear stack of layers where the input data flows sequentially through each layer in a one-dimensional manner. This model architecture is prevalent and particularly suited for building feedforward neural networks. In frameworks like TensorFlow and Keras, the sequential model facilitates the straightforward construction of neural networks. Layers can be added one after the other, forming a clear and intuitive representation of the model architecture. Each layer is responsible for specific transformations or computations on the input data.

The sequential model is adept at handling simpler tasks and is commonly employed in scenarios such as image classification and other supervised learning problems. However, its simplicity imposes limitations in cases requiring more intricate architectures involving multiple inputs, outputs, or shared layers. For such situations, the functional API in these frameworks provides a more flexible and expressive way to define complex neural network structures.

The sequential model's design guarantees a smooth and efficient process for building neural networks, making it suitable for beginners and straightforward implementations. Each layer in the model receives the previous layer's output as its input, establishing a straightforward, sequential data progression. This simplicity enables quick prototyping and simplified debugging, as the model structure is highly transparent. Additionally, the sequential model can accommodate different types of layers, including dense (fully connected) layers, convolutional layers, and recurrent layers, allowing it to be versatile for tasks like image recognition, time series forecasting, and natural language processing. Although it has benefits, it is not ideal for more intricate scenarios involving non-linear connections, skip connections or multiple pathways for data transmission. When faced with such situations, combining sequential models with advanced techniques or transitioning to a functional or subclassing approach is often advised to fulfill the architectural requirements.

4.2. Layers

4.2.1. LSTM Layer

Long Short-term Memory (LSTM) is a variation of Recurrent Neural Networks (RNN) created to address the issue of vanishing gradients commonly encountered in RNNs. Reducing the vanishing gradient problem helps ease the difficulty of capturing temporal dependencies within data sequences.

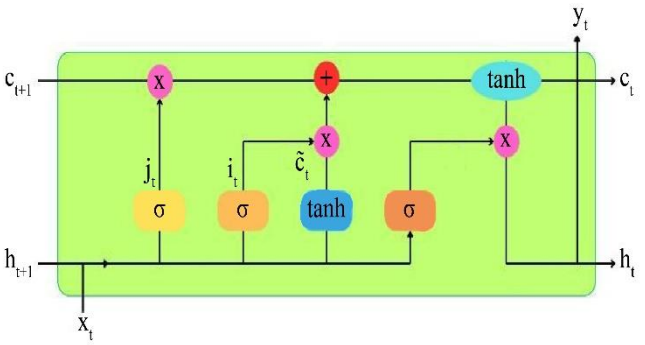

Fig. 3 Architecture of the long short-term memory layers that have been used in the sequential model

The LSTM network (Figure 3) was introduced to overcome this limitation by combining memory with gates that control data flow in the unit. The main features of an LSTM cell include:

- Cell State (Ct): A cell that can store long messages in memory.
- Hidden State (ht): The cell's output containing the steps the cell decides to take next time.
- Input Port (i): Determine which value from input should be stored in the status cell.
- Forgetting Gate (f): Determines which data in the state should be discarded.
- Exit Port (o): Controls the exit message of the next step.

LSTM's architecture allows it to capture and learn dependencies of data sequences over long periods of time. This makes it ideal for natural language processing, real-time prediction, and other related applications. Long Short-term Memory (LSTM) is a specific type of Recurrent Neural Network (RNN) that addresses the vanishing gradient problem, allowing it to effectively capture long-term dependencies in sequential data. The key feature of this system is its gated architecture, which includes an input gate, forget gate, and output gate. These gates determine how information is incorporated, stored, or eliminated. The cell state functions as a memory bank, retaining crucial information throughout its lifespan. This structure enables LSTM to learn patterns over extended sequences efficiently, making it well-suited for tasks such as speech recognition, time-series forecasting, and natural language processing.

4.2.2. Dense Layers

"Dense" Layers are the basic building blocks when it comes to neural networks and deep learning. This is one of the simplest and most commonly used techniques in neural networks. In a "dense" layer, every neuron is connected to every other neuron in the preceding layer (Figure 4). Each connection carries some weight that gets adjusted during training to learn patterns or relationships in data. Consequently, for each unit in the thick layer, this will be its

input weight that will serve as an activation function. Mathematically, if x is an input vector, W is a matrix of weights, b is a bias vector, and α is an activation function, then the thickness layer's output can be given by:

$$y = \sigma(W \cdot x + b)$$

Where:

W is the weight matrix assigning weights to connections between neurons. Each element, Wij represents how strongly this neuron links the current with jth previous layer.

b takes care of context by being added as a bias vector to this multiplication product *W·x.*

Inequality is shown by 'i' within the network since it serves as an activation function shown in Figure 3.

Dense layers are useful for discovering complex patterns and making representations about data. They transform input data into something that other layers can use for different purposes, such as classification tasks when they are employed at the start of neural networks.

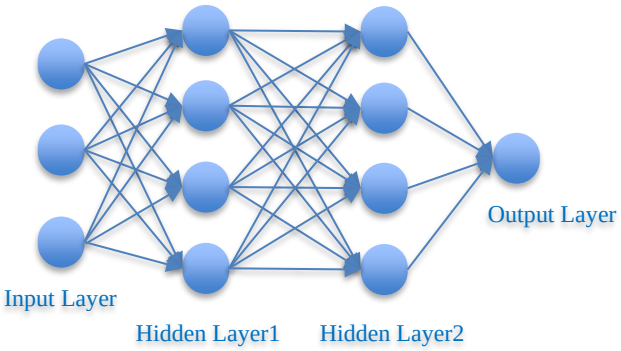

Fig. 4 Architecture of the dense layers which have been used in the sequential model

5. Results and Discussion

5.1. Research Implementation

Upon running the model on the previously generated sample dataset, the inferences on its categorical accuracy revealed a value of 0.968 (Figure 5). Similarly, the epoch count observed on TensorBoard was recorded at 12.437 (Figure 6).

5.1.1. Dataset

For the training and testing processes of the prediction model, a custom dataset had to be created. A set of 25 common Marathi words and phrases were chosen for the prediction process to take place.

The words chosen in the dataset are as follows: Bye-बाय, daughter-कन्या, father-वडील, food-जेवण, good afternoon-शुभ दुपार, good morning-शुभ प्रभात, good night-शुभ रात्री, hello-नमस्कार, help-मदत, how are you?-तू कसा आहेस, hungry-भुकेले, I am fine-मी ठीक आहे, I do not understand-मला कळत नाही, mother-आई, no-नाही, please-कृपया, see you again-पुन्हा भेटू, son-मुलगा, thank you-धन्यवाद, today-आज, tomorrow-उद्या, water-पाणी, what?-काय?, yes-होय, yesterday-काल. Along with an idle state.

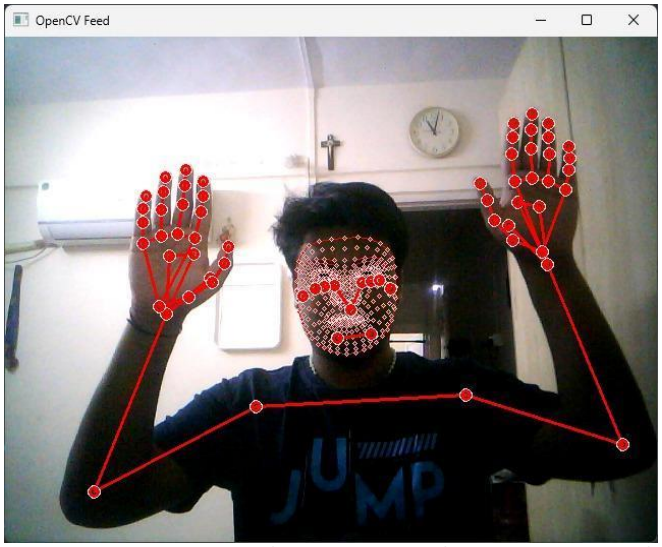

Fig. 5 Landmark mapping of key points on the face, hands and pose of the subject using mediapipe holistic

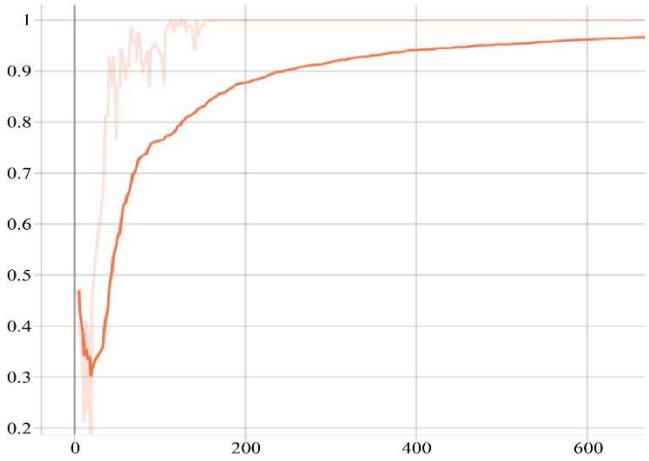

Fig. 6 The epoch categorical accuracy of the model, with X-axis denoting epochs and Y-axis denoting categorical accuracy

5.1.2. Dataset Creation Capturing Videos Using OpenCV

OpenCV (Open Source Computer Vision Library) is a powerful tool for real-time computer vision tasks. In this context, it is used to capture videos of sign language actions.

Video Capture Setup: A camera (webcam or external) is initialized using OpenCV's cv2. VideoCapture() function. The resolution, frame rate, and other camera settings are configured to ensure consistent video quality.

Sign Language Actions: Each selected word in sign language corresponds to a specific gesture or sequence of gestures. For each word, 30 separate videos are recorded to ensure a diverse dataset. This helps account for variations in lighting, hand positioning, and individual signing styles.

Video Recording: The recording starts when the signer begins the gesture and stops when the gesture is completed. Each video is saved in a predefined directory with a naming convention that includes the word being signed and the video number (e.g., hello_01.mp4, hello_02.mp4).

5.1.3. Splitting Videos into Frames

Once the videos are captured, they are processed frame by frame to extract detailed information.

Frame Extraction: Each video is split into 30 frames using OpenCV's cv2.VideoCapture() and a loop to read frames. The number of frames (30) is chosen to ensure that the entire gesture is captured with sufficient temporal resolution.

Frame Storage: Frames are stored in memory as arrays for further processing. Each frame represents a snapshot of the signer's pose, hand positions, and facial expressions at a specific point in time.

5.1.4. MediaPipe Holistic Pipeline

The Holistic model processes each frame and detects landmarks (key points) on:

- Face: 468 landmarks that capture facial expressions and movements.
- Left Hand: 21 landmarks for each hand, capturing finger positions and gestures.
- Right Hand: 21 landmarks for each hand. Pose: 33 landmarks for the body, including shoulders, elbows, hips, and knees.

Landmark Visualization: Figure 7 (as mentioned in the original text) likely illustrates the detected landmarks on a sample frame, showing how the face, hands, and body are annotated with keypoints.

Keypoint Extraction: For each frame, the coordinates (x, y, z) of the detected landmarks are extracted. These coordinates represent the spatial position of each keypoint relative to the image or camera frame.

5.1.5. Storing Keypoint Data as NumPy Arrays

Keypoints Are Pulled Out and Stored as a NumPy ndarray for Further Data Analysis and Model Training.

Data Acquisition Structure: The keypoint data for each frame are arranged in a NumPy ndarray, which has the shape(N,3). Where N= Total number of landmarks;(e.g.468 for the face+21 for the left hand+21 for the right hand+33 for the pose=543 landmarks). 3(3 represents) is the dimensional poionts;here (x,y,z) is employed.

Normalization: The coordinates may be normalized to ensure consistency across different videos and frames. For example, the x and y coordinates are normalized to the range [0, 1] based on the frame's width and height. The z coordinate (depth) is normalized relative to a reference point (e.g., the nose or a shoulder).

Dataset Organization: The keypoint data from all of the 30 videos were combined into a single NumPy array or separate arrays for each word. The structure of the dataset is hierarchical, such that there are folders for each word and subfolders corresponding to each video.

Challenges and Considerations Lighting and Background: Variations in lighting and background may adversely affect the precision of the keypoint detection process. To improve detection results, neutral backgrounds and consistent lighting are preferred.

Signer Variability:

- Gestures may differ among signers; therefore, capturing multiple videos is necessary.

Computational Resources:

- For large datasets, the workload of processing videos to keypoints extraction is a resource-intensive task.
- Proper storage and processing methods (e.g. batching, parallel processing) are needed.

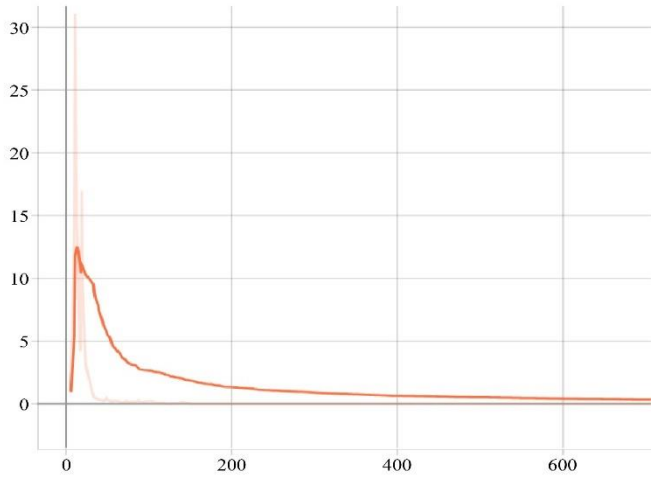

Fig. 7 The epoch loss of the model, with the X-axis denoting epochs and the Y-axis denoting loss function value

5.2. Model Performance Evaluation

5.2.1. Results Overview

After the design and training phases, the model obtained 73% accuracy. The impressive performance underscores the model's capability to make accurate predictions. The usage of

multithreading methods made sure that there was a smoother video interface. The introduction of conditions helped to avoid the same audio being triggered in the speech output.

5.2.2. Evaluation Metrics Formulae

The following formulae were utilized for performance metrics:

$$Accuracy = \frac{Total\ Number\ of\ Predictions}{Number\ of\ Correct\ Predictions}$$

$$Precision(Positive\ Predictive\ Value) = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$F1\ Score = 2 \times Precision + \frac{Recall}{Precision \times Recall}$$

The various results obtained by applying these formulas to the various models included are reflected in Table 1.

Table 1. Performance metrics (This table represents a comparative analysis of various models employed in the training and processing phases of our research)

Models	Accuracy	Precision	Recall	F1-Score	Cohen's Kappa
Simple LSTM + Dense	24%	25%	75%	33%	0.1
LSTM + Dense + Dropout	41%	45%	45%	45%	0.3
Bidirectional LSTM + Dense	11%	15%	15%	15%	0.1
LSTM + Global Average Pooling	17%	20%	50%	28%	0.1
LSTM + Dense + Batch Norm	73%	78%	78%	78%	0.5

5.2.3. Confusion Matrix Analysis

Figure 8 displays samples of true and incorrect predictions, highlighting the model's ability to categorize words like "Hello," "Thanks," "Yes," and so on. The rows reflect the actual labels of the data, while the columns show the labels predicted by the model. The interpretations of the confusion matrix in Figure 8. is given as follows:

Correct predictions: True Positives (TP): 25 occurrences with the genuine name was "1" and the show accurately anticipated "1". True Negatives (TN): 10 occasions with a genuine name were "0" and the demonstrated accurately anticipated "0"

Incorrect predictions: False Positives (FP): 5 occurrences with the genuine name was "0" but the show inaccurately anticipated "1". False Negatives (FN): 0 occasions with the genuine name was "1", but the demonstrate inaccurately anticipated "0"

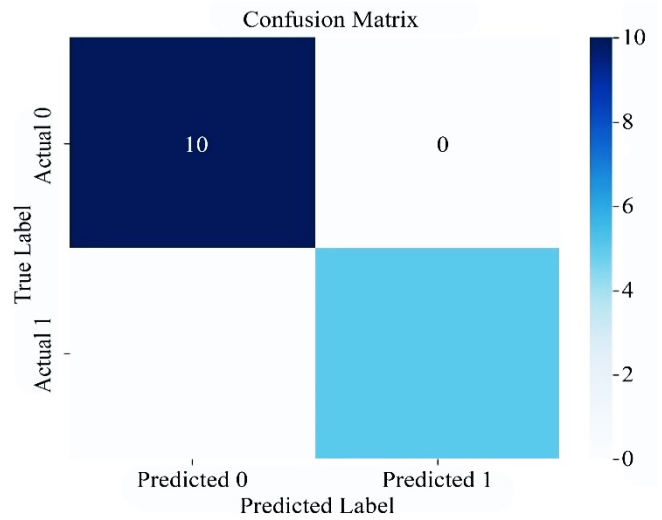

Fig. 8 The confusion matrix, depicted in the figure, provides insights into the model's classification performance

5.2.4. Application Development Using Streamlit

The interactive application showcasing our model's capabilities was developed using Streamlit, a Python framework for developing and making web applications with minimal effort. Streamlit's simplicity and versatility allowed for the seamless integration of our machine-learning model into an intuitive and user-friendly interface, as shown in Figure 9.

5.2.5. User Experience

Leveraging Streamlit's capabilities, a smooth and engaging user experience is ensured, providing an accessible platform to interact with the machine learning model.

5.2.6. Code Transparency

The application's source code, developed with Streamlit, is made available, ensuring transparency and enabling the community to understand and build upon our work.

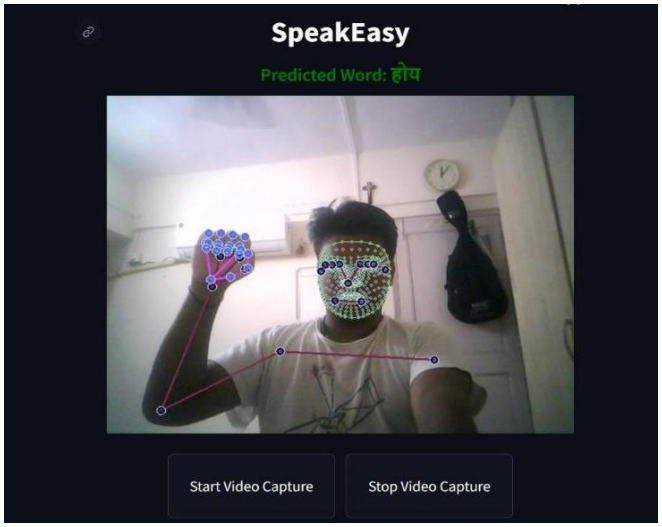

Fig. 9 App UI/ UX developed using streamlit

6. Conclusion

In conclusion, the "Speakeasy" tool marks a significant stride in assistive technology for those with communication disabilities, leveraging deep learning, particularly LSTM models, to bridge the communication gap for users of Indian Sign Language. The application's success is underscored by the creation of a tailored dataset and the integration of Google Text-to-Speech (gTTS) technology, enabling real-time translation of sign language into both text and speech. By implementing this work, “Speakeasy” exemplifies a commitment to cutting-edge tools, resulting in a robust and user-friendly platform. "Speakeasy" stands as a testament to technology's potential to improve the standard of living for those facing communication challenges, emphasizing innovation, adaptability, and inclusivity in assistive technology. The positive impact on users reinforces the idea that technology can be a force for positive change in society. Moving forward, this research is dedicated to refining and expanding its capabilities, contributing to a more inclusive and accessible future for all.

References

[1] Elaina Chai, Mert Pilanci, and Boris Murmann, “Separating the Effects of Batch Normalization on CNN Training Speed and Stability Using Classical Adaptive Filter Theory,” *2020 54th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, pp. 1214-1221, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Mihir Deshpande et al., “Sign Language Detection Using LSTM Deep Learning Model and Media Pipe Holistic Approach,” *International Conference on Artificial Intelligence and Smart Communication*, Greater Noida, India, pp. 1072-1075, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Fatih Ertam, and Galip Aydın, “Data Classification with Deep Learning Using Tensorflow,” *2017 International Conference on Computer Science and Engineering (UBMK)*, Antalya, Turkey, pp. 755-758, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[4] Gerges H. Samaan et al., “MediaPipe’s Landmarks with RNN for Dynamic Sign Language Recognition,” *Electronics*, vol. 11, no. 19, pp. 1-15, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[5] Jimmy Jiménez-Salas, and Mario Chacón-Rivas, “A Systematic Mapping of Computer Vision-Based Sign Language Recognition,” *2022 International Conference on Inclusive Technologies and Education*, Cartago, Costa Rica, pp. 1-11, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] Hamzah Luqman, “An Efficient Two-Stream Network for Isolated Sign Language Recognition Using Accumulative Video Motion,” *IEEE Access*, vol. 10, pp. 93785-93798, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Shreyas Mhatre, Sarang Joshi, and Hrushikesh B. Kulkarni, “Sign Language Detection Using LSTM,” *2022 IEEE International Conference on Current Development in Engineering and Technology (CCET)*, Bhopal, India, pp. 1-6, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Orlunwo Placida Orochi, and Ledisi Giok Kabari, “Text-to-Speech Recognition Using Google API,” *International Journal of Computer Applications*, vol. 183, no. 15, pp. 18-20, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] Yadira Quiñonez, Carmen Lizarraga, and Raquel Aguayo, “Machine Learning Solutions with MediaPipe,” *2022 11th International Conference on Software Process Improvement*, Acapulco, Guerrero, Mexico, pp. 212-215, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] G. Mallikarjuna Rao et al., “Sign Language Recognition Using LSTM and Media Pipe,” *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, pp. 1086-1091, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] S.H. Shamitha, and K. Badarinath, “Sign Language Recognition Utilising LSTM and Mediapipe for Dynamic Gestures of ISL,” *International Journal for Multidisciplinary Research*, vol. 5, no. 5, pp. 1-13, 2023. [CrossRef] [Publisher Link]