PROJECT REPORT

ON

"INDIAN SIGN LANGUAGE RECOGNITION USING MACHINE LEARNING"

**Submitted in Partial Fulfillment of the Requirement
for the award of a Degree of**

Bachelor of Technology

In

Computer Science & Technology

Submitted By

Aryan Bhanot, Vibhay Bakshi, Yash Poonia

2K22CSUN01130, 2K22CSUN01157, 2K22CSUN01159

Under the Guidance of **Dr. Mamta Arora**

DEPARTMENT OF COMPUTER SCIENCE & TECHNOLOGY

MANAV RACHNA UNIVERSITY

FARIDABAD, HARYANA (INDIA)

(Formerly Manav Rachna College of Engineering)

Dec, 2024

ACKNOWLEDGEMENTS

I would like to thank my teacher who gave me a golden opportunity to work on this project. I'd also like to express my gratitude to my project mentor **Dr. Mamta Arora**. I must also thank my parents and friends for the immense support and help during this project. Without their help, completing this project would have been very difficult.

Name of Students: Aryan Bhanot, Vibhay Bakshi, Yash Poonia
Roll Numbers: 2K22CSUN01130, 2K22CSUN01157, 2K22CSUN01159

CANDIDATE’S DECLARATION

I hereby declare that the work presented in this project report entitled “**Indian Sign Language Detection using Machine Learning**”, is an authentic record of our own work carried out at Manav Rachna University.

Supervisor Name: Dr. Mamta Arora

Student Name: Aryan Bhanot

Roll No: 2K22CSUN01130

Student Name: Vibhay Bakshi

Roll No: 2K22CSUN01157

Student Name: Yash Poonia

Roll No: 2K22CSUN01159

Date: 25th November 2024

INDEX

Abstract

Sign language, which is commonly used all over the world to communicate faster, is not just a medium to express yourself in less time, but also a medium for individuals affected with disabilities to communicate regularly due to the lack of ability to communicate using normal language. To create a medium that would pass through this barrier, AI techniques were thought to be the most useful, wherein a sign language prediction system can be integrated to enforce a teaching interface helping these individuals to get their hands on sign language learning. The focus of this research was on processing video files containing sign language gestures and predicting the signs depicted within those videos. For this purpose, the study used the "INCLUDE" dataset, which consists of MOV format video files. The analysis focused on specific sign language gestures, extracting frames and landmarks from the videos associated with these signs. To have this data in a reliable form that could be used with the model, it was normalized. For this, various techniques including StandardScaler, LabelEncoder, and SMOTE, were applied. This study considered various approaches including Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and a hybrid Convolutional Neural Network (CNN) + LSTM model. The best-performing model was the one that included both CNN and LSTM layers and achieved a final validation accuracy of 93.89%, with precision, recall, and F1 scores all reaching 94%. This model was then integrated into the app, which features a server built using Python's Flask library. The server includes an endpoint to receive sign prediction requests from the app, developed in React Native. Future work can focus on expanding the dataset and improving accuracy by introducing additional techniques.

1. Introduction

Sign language is an important medium of communication for millions worldwide, especially for individuals with disabilities, but due to the lack of understanding and education, there is a big communication gap that exists among people despite the existence of sign language, especially in countries like India, where Indian Sign Language (ISL) is primarily used. For maintaining inclusivity and a sense of ease in communication despite the communication barriers that exist the development of sign language recognition (SLR) systems is important. In the context of human-computer interaction (HCI), the techniques of Vision-based SLR were found to be effective as they can be useful in real-time sign language recognition and translation. These systems typically operate through processes like image preprocessing, feature extraction, and gesture classification, with feature extraction being critical to overall accuracy and performance. Recent research has demonstrated various methods for improving SLR accuracy, from traditional approaches like Bag of Visual Words (BoVW) and Support Vector Machines (SVM) to advanced deep learning models, such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, which have achieved high accuracy in real-time ISL recognition. Furthermore, advancements in multimedia and accessibility technologies—such as MediaPipe and OpenCV—have provided robust tools for tracking and segmenting gestures, expanding the potential for SLR applications across diverse platforms, including video conferencing and multimedia-sharing sites. Despite these advancements, challenges remain in achieving contextually accurate and standardized ISL recognition models suitable for real-world deployment. This study sought to address these challenges by presenting a motion-based, real-time ISL recognition system, utilizing a trained LSTM model to enhance gesture prediction accuracy. The study's primary objectives were to explore diverse hand sign languages, apply neural network algorithms, and employ image processing techniques to achieve reliable live translation of ISL. Through this research, the aim was to contribute to the field of accessible technology, fostering an inclusive environment that supports effective communication for the hearing-impaired community in India.

2. Aim and Objectives

This research aimed to utilize the existing image processing and AI techniques including models like Long Short-Term Memory (LSTM) and libraries like OpenCV, and MediaPipe for predicting the sign languages based on video inputs by extracting the frames from videos and further extracting landmarks from the video. Further, this study aimed to create an app using React Native wherein users with disabilities can learn sign language by either searching for a sign or testing their signing skills in the app's play mode by seeing a sign prompt and inputting the corresponding sign.

3. Related Work

Vision-based sign language recognition (SLR) systems are a big step in human-computer interaction (HCI) [1]. These systems help deaf and mute people communicate better across different regions. The process usually has three parts: preparing images, finding features, and recognizing gestures. Feature extraction is very important because it provides the key information for the system to work well. This paper reviews the different feature extraction methods used in vision-based SLR systems. It also talks about how these methods are grouped based on their principles and uses. The paper focuses on Indian Sign Language (ISL), looking at recent improvements, current challenges, and what can be done in the future. Understanding feature extraction helps improve SLR systems, making communication easier for hearing-impaired people. A range of approaches have been developed to bridge communication between hearing individuals and those who use sign language as in [2]. Traditional systems often utilize avatars or animations for visualizing sign language, with varying degrees of complexity in translating spoken or text inputs to signs. Early models, such as those leveraging IBM Model 1-3, focus on word-by-word translation and reordering lexemes to suit target sentence structures. These approaches, however, face challenges with contextual accuracy and meaningful sentence formation. Furthermore, recent research has explored machine learning and natural language processing (NLP) techniques for improved parsing and translation, using tools like WordNet for synonym matching and similarity measurements, although such systems still encounter issues in capturing nuanced meanings or finding equivalent signs. Innovations have included the use of video concatenation methods to depict ISL Gloss for each translated word, enhancing clarity by reducing sentence complexity. Additionally, there is growing interest in leveraging multimedia platforms like YouTube to deliver more interactive, human-like visual outputs. Emerging tools in this domain also include image-based recognition systems that use CNN and TensorFlow to translate physical signs into text and audio, focusing on select languages but indicating broader potential with future language expansion. The proposed system thus addresses gaps by using simplified gloss structures, aiming to deliver clear ISL animations through YouTube, improving accessibility, and paving the way for further enhancements as ISL dictionaries expand. Sign language recognition systems are essential to bridge the communication gap for the approximately 63 million people in India with hearing impairments [3], many of whom rely on Indian Sign Language (ISL) for daily communication. Recognizing the lack of understanding of sign language among the general population, recent efforts have focused on developing standardized recognition models for ISL. A prominent approach involves the Bag of Visual Words (BoVW) technique, which facilitates real-time ISL prediction by employing various feature detectors and descriptors like SIFT, SURF, ORB, STAR + BRIEF, and FAST + FREAK. Through comparative analysis, it was observed that the SURF detector paired with a Support Vector Machine (SVM) classifier yielded a 99.94% accuracy and a rotation tolerance of up to 5 degrees. Additionally, the potential of deep learning methods, particularly Convolutional Neural Networks (CNNs), was explored, showing a remarkable 100% accuracy for ISL recognition. Comparative experiments with the American Sign Language (ASL) dataset also highlighted the robustness of CNN models, achieving 99.77% accuracy for ASL, while traditional methods with SURF and SVM yielded lower performance at 65.41% accuracy. These findings underscore the effectiveness of feature-based methods and CNNs in achieving high-accuracy sign language recognition, paving the way for practical, standardized ISL systems. Indian Sign Language (ISL) detection as discussed in [4] has become increasingly vital in addressing communication barriers between hearing and non-hearing communities in India. As ISL is the primary mode of communication for the deaf and hard of hearing in India, there is a pressing need for systems that facilitate understanding among those unfamiliar with the language. Recent advancements propose an ISL detection system utilizing deep convolutional neural networks (CNNs), MediaPipe, and OpenCV to enable real-time gesture recognition. This system detects and classifies ISL gestures through real-time hand tracking, gesture segmentation, feature extraction, and classification using a trained CNN model. Enhanced with a user-friendly interface, the system includes accessibility features like text-to-speech, image-to-speech, text-to-image conversion, and webcam input for text display. Such a comprehensive system holds significant promise for fostering communication between deaf and hearing individuals, offering a deployable solution to bridge communication gaps effectively in real-world scenarios. In light of the limited awareness and understanding of sign language among the general population [5], the need for effective sign language recognition systems is essential. Hearing-impaired individuals primarily rely on sign language, which incorporates gestures, facial expressions, and body movements to communicate. Since various countries have distinct sign language systems, this study focuses on an approach to recognizing Indian Sign Language (ISL) gestures. The proposed model employs a motion-based ISL recognition method powered by deep learning. Using OpenCV, gestures are captured in real-time, key points are identified through MediaPipe, and a trained Long Short-Term Memory (LSTM) model predicts the gestures. The dataset used for training was custom-developed by the research team, resulting in an average accuracy of 92%. This system has practical applications for real-time ISL recognition and can be integrated into video-conferencing platforms, enhancing accessibility for hearing-impaired individuals. The sign language recognition system proposed by Kumar et al. (2016) [6] integrates convolutional neural networks (CNNs) and Random Forest classifiers for gesture identification. It employs OpenCV and MediaPipe for real-time hand tracking, preprocessing gesture images, and extracting key landmarks. The CNNs are pivotal in feature extraction, significantly improving prediction accuracy. User interaction is facilitated through a Tkinter-based graphical user interface (GUI), while text-to-speech feedback is provided using Pyttsx3. Visualization tools like Matplotlib help analyze model strengths and misclassifications, aiding iterative optimization. This system aims to make sign language more accessible in diverse real-world settings. In the earlier work of Vogler and Metaxas (1999) [7], the challenge of scaling American Sign Language (ASL) recognition was addressed using Parallel Hidden Markov Models (PaHMMs). ASL recognition involves handling simultaneous hand movements, which increases modeling complexity. PaHMMs manage this by independently modeling each process, eliminating the need for extensive combinatorial training data. Demonstrating robust recognition of a 22-sign ASL vocabulary, this approach offers scalability and robustness, paving the way for more comprehensive gesture recognition systems. Building on deep learning, Camgoz et al. (n.d.) [8] introduced SubUNets, a framework for sequence-to-sequence tasks. SubUNets decompose tasks into domain-specific subnetworks, leveraging intermediate representations to incorporate expert knowledge and enable transfer learning. This structure achieves state-of-the-art performance in continuous sign language recognition by combining CNNs for spatial features, BLSTM layers for temporal dynamics, and CTC loss for alignment-free training. This modular approach excels in handling weakly labeled datasets and complex spatio-temporal challenges, highlighting its adaptability to real-world problems. Focusing on Indian Sign Language (ISL), Wadhawan and Kumar (2020) [9] developed a CNN-based recognition system incorporating convolutional, pooling, and dropout layers to prevent overfitting. This model processes RGB images by resizing and normalizing them before classification, achieving 99.9% accuracy using the Adam optimizer and demonstrating strong generalization on large datasets. This advancement surpasses prior ISL recognition methods, emphasizing its reliability for practical applications. Similarly, Jie Huang et al. (2015) [10] proposed a 3D CNN model utilizing Microsoft Kinect data, capturing spatial and temporal features from color, depth, and skeleton inputs. This architecture alternates convolution and subsampling layers to enhance feature extraction, followed by a multilayer perceptron for classification. The model outperformed traditional Gaussian Mixture Model-Hidden Markov Models (GMM-HMM) and earlier 3D Convolutional Network methods, especially in multi-channel scenarios. In contrast, Mehdi and Khan (2002) [11] explored sensor gloves for gesture recognition, emphasizing minimal training. Using an artificial neural network (ANN) with a 7-sensor input layer, this system achieved an 88% accuracy rate. However, variations in samples from non-sign language users posed challenges, revealing opportunities for improvement with diverse training data. Jiang et al. (n.d.) [12] advanced sign language recognition through skeleton-based and multimodal approaches. Their SL-GCN model builds spatio-temporal graphs for motion analysis, while SSTCN uses body features for recognition. Coupled with 3D CNNs for RGB, depth, and optical flow inputs, their multimodal ensemble method significantly enhanced recognition accuracy, achieving top rankings in benchmark challenges. Amrutha and Prabu (2021) [13] took a different route, applying edge-based methods like Prewitt and Canny for Arabic Sign Language Recognition (SLR). They used principal component analysis (PCA) and convex hull techniques for dimensionality reduction, with KNN classifiers achieving 65% accuracy in controlled environments. While performance was limited, the study highlighted the potential for real-time systems with larger datasets and advanced classifiers. Zuo et al. (2023) [14] introduced the NLA-SLR framework, which processes sign language videos through data pre-processing, a Video-Keypoint Network (VKNet), and a head network. Key points extracted as heatmaps feed into VKNet's dual-stream architecture, facilitating feature exchange between video and key points. The head network incorporates language-aware label smoothing and inter-modality mixup to leverage both visual and linguistic features, achieving high recognition accuracy by combining modalities effectively. Finally, Cui et al. (n.d.) [15] developed a system for continuous sign language recognition using CNNs with temporal convolution for spatiotemporal feature extraction and bidirectional LSTMs for sequence learning. The three-stage optimization process includes end-to-end training with CTC for alignment, feature extractor tuning, and sequence learning refinement. This approach significantly enhances performance, especially on tasks requiring accurate sequential predictions.

4. Research Methodology

This research methodology covers the critical steps from data collection, and model training to evaluating the best model and developing a prediction system based on that model as shown in *Fig. 1*.

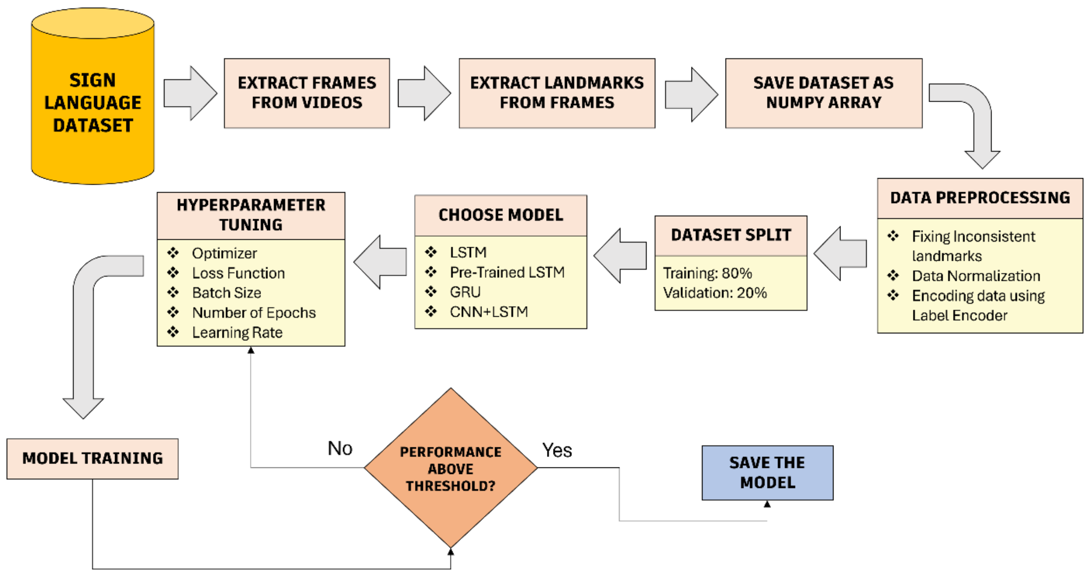


***Figure 1**. Methodology*

Dataset

The first step was to find relevant data, and the "INCLUDE" dataset was deemed suitable for the purpose. It provides a collection of video clips for several signs in Indian Sign Language (ISL) as shown in *Fig. 2*. The dataset consists of 4,292 video clips capturing 263 unique signs, organized into 15 categories. It was developed to support machine learning applications in sign language recognition and includes native signers. A curated subset called INCLUDE-50 facilitates faster model evaluation. This dataset offers separate training and testing sets, along with shell scripts for efficient download and integration. Licensed under Creative Commons Attribution 4.0, the "INCLUDE" dataset supports open research in ISL recognition. Initially, a set of signs featuring adjectives was considered which later shifted to greeting-based signs. Dataset Link: https://zenodo.org/records/4010759

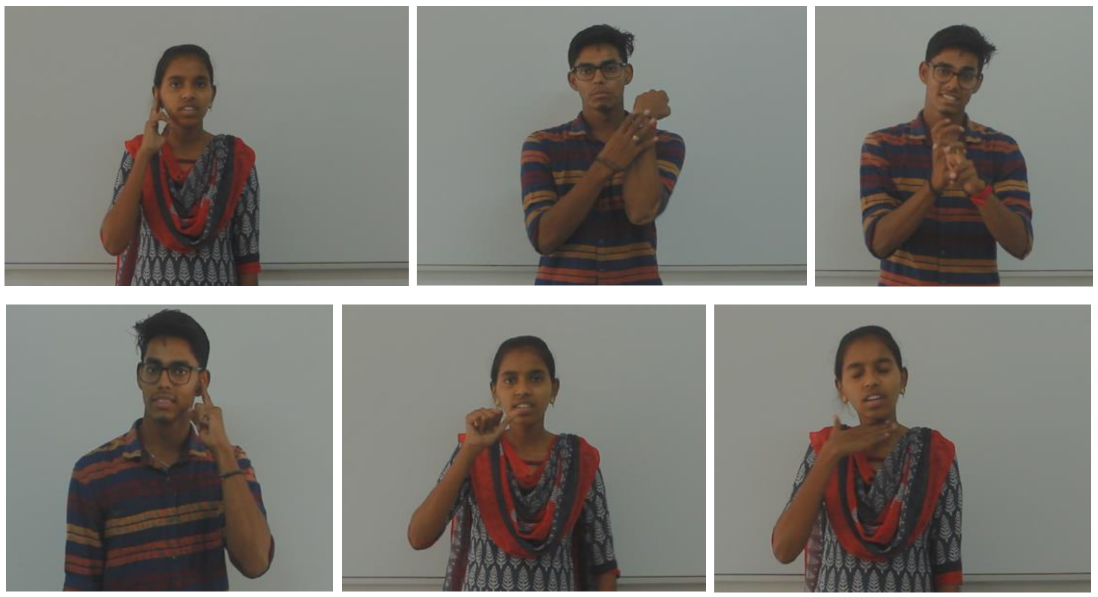


***Figure 2**. A glimpse of Indian sign languages*

Data Preprocessing and Visualization

The first step in data preprocessing involved converting the video clips into a form compatible with the model training process. This included the following steps:

1. Extracting Frames: Each video was processed using the OpenCV library. The video was captured with cv2.VideoCapture(), and frames were extracted using the method cap.read().
2. Extracting Landmarks: For each frame, the landmarks were extracted by converting the frames from BGR to RGB using OpenCV's cvtColor() method. The 3D coordinates (x, y, z) for each body part were then extracted.
3. Saving dataset as a Numpy array: Two lists—one for labels and one for landmarks—were populated and saved as a Numpy array.
4. Label Distribution: The label occurrences were visualized, revealing unequal distribution ("Good Morning": 1400, "Alright": 1417, "Good Afternoon": 1485, "How Are You": 1676, "Hello": 1300) as shown in *Fig. 3., and Fig. 4*. The minority classes were oversampled using SMOTE (Synthetic Minority Over-sampling Technique), resulting in a distribution of 1301 for each sign.

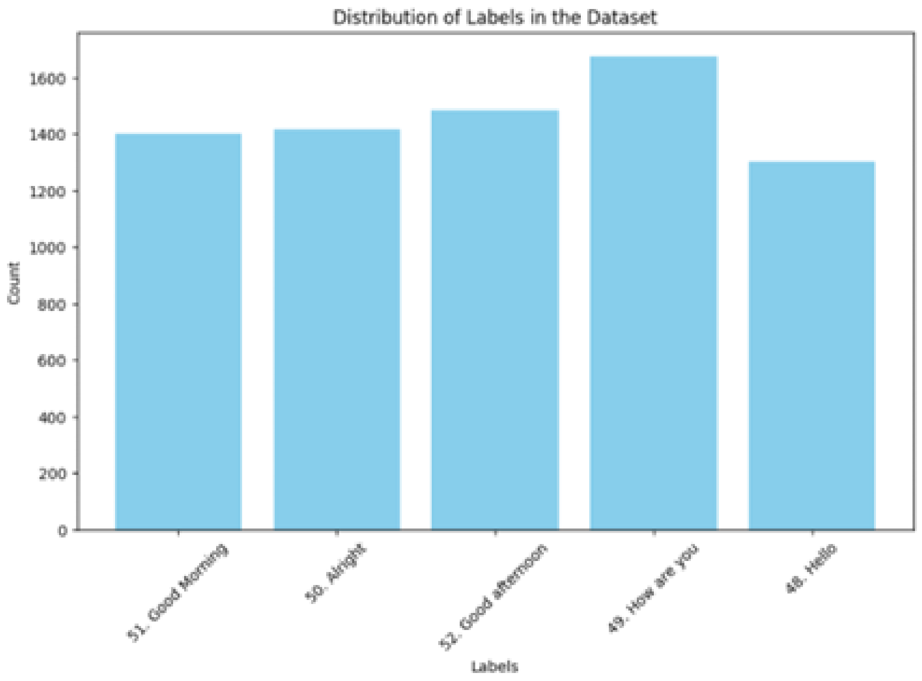


Figure 3. *Label Distribution initially*

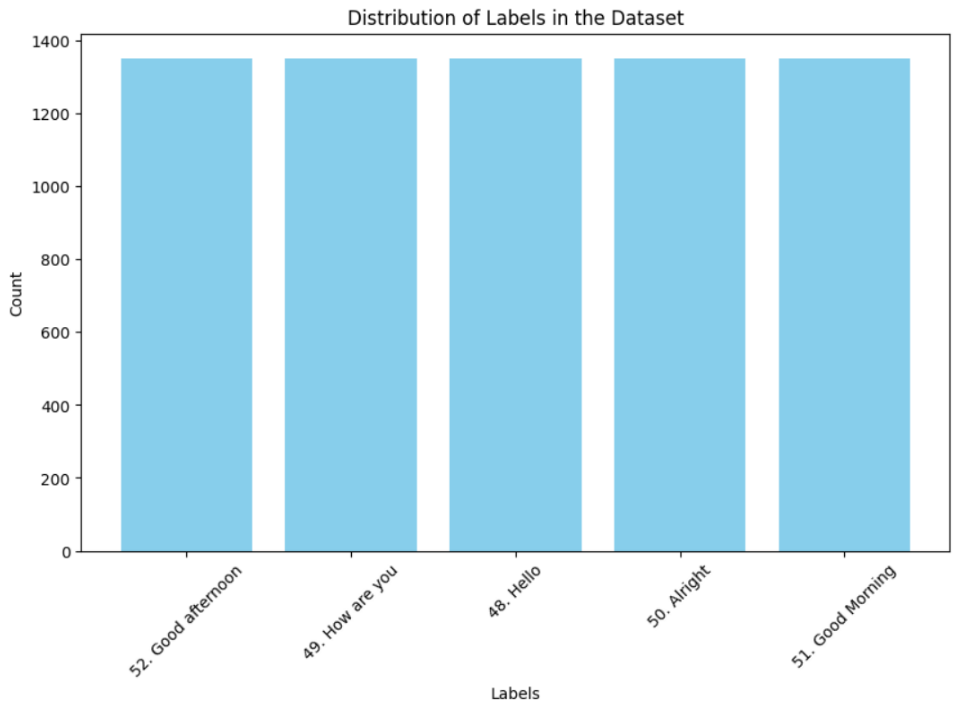


Figure 4. *Label Distribution finally*

5. Normalization: The data was normalized using StandardScaler() to prevent features from disproportionately influencing the results.
6. Encoding the signs: A label encoder was used to convert string data (e.g., "hello", "good morning") into numerical values.
7. Tensor Conversion: Data was converted into tensors for batch processing during training and evaluation.
8. Batch Preparation: A DataLoader was used to create mini-batches for efficient model training and to avoid memory overload.

Model Architecture and Training

Models considered for this study included Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), due to their effectiveness in handling time-series data. These models were implemented using libraries like PyTorch and TensorFlow. A learning rate of 0.005 was used for the early LSTM model built with PyTorch (LSTM Model 1), and a learning rate of 0.001 was used for the other early model built with TensorFlow (LSTM Model 2), and the remaining models. A batch size of 32 was used for training most models, except for one LSTM Model 2, which used a batch size of 64. The Adam optimizer was used, and the CrossEntropyLoss function was used for the LSTM Model 1, while sparse_categorical_crossentropy was used for other models.

The architecture for each model was as follows:

1. LSTM Model 1: This model was built using PyTorch, and had three LSTM layers with 64, 32, and 16 neurons, respectively. A dropout rate of 0.4 was applied to each layer to reduce overfitting. The output from the final LSTM layer was passed through a fully connected layer, mapping it to the number of classes.
2. LSTM Model 2: This model was built using TensorFlow, and had three LSTM layers, with the first and second layers having 128 neurons, and the third having 64 neurons. return_sequences was set to true, and a dropout rate of 0.4 was applied after each layer. A Time Distributed Dense Layer with 64 neurons and a ReLU activation function was added. The final layer had 32 units to output only the last time step's output.
3. Pretrained Model (MobileNetV2): MobileNetV2 was used for feature extraction, with a shape of 224x224 RGB images. The first layer had 128 neurons, the second layer had 64 neurons, and dropout was applied throughout the layers. Fully connected layers performed the final classification.
4. GRU Model (TensorFlow): This model had 128 neurons in the first and second layers, and 64 neurons in the third layer. Dropout was applied throughout. The final layer applied a Softmax activation function to output probabilities for each class.
5. CNN+LSTM Model: This hybrid model started with a 1D Convolutional layer with 64 filters, followed by a MaxPooling layer. It then used a Bidirectional LSTM with 128 neurons, followed by an Attention layer and a Dense layer with 64 neurons. The final layer applied Softmax activation to output class probabilities.

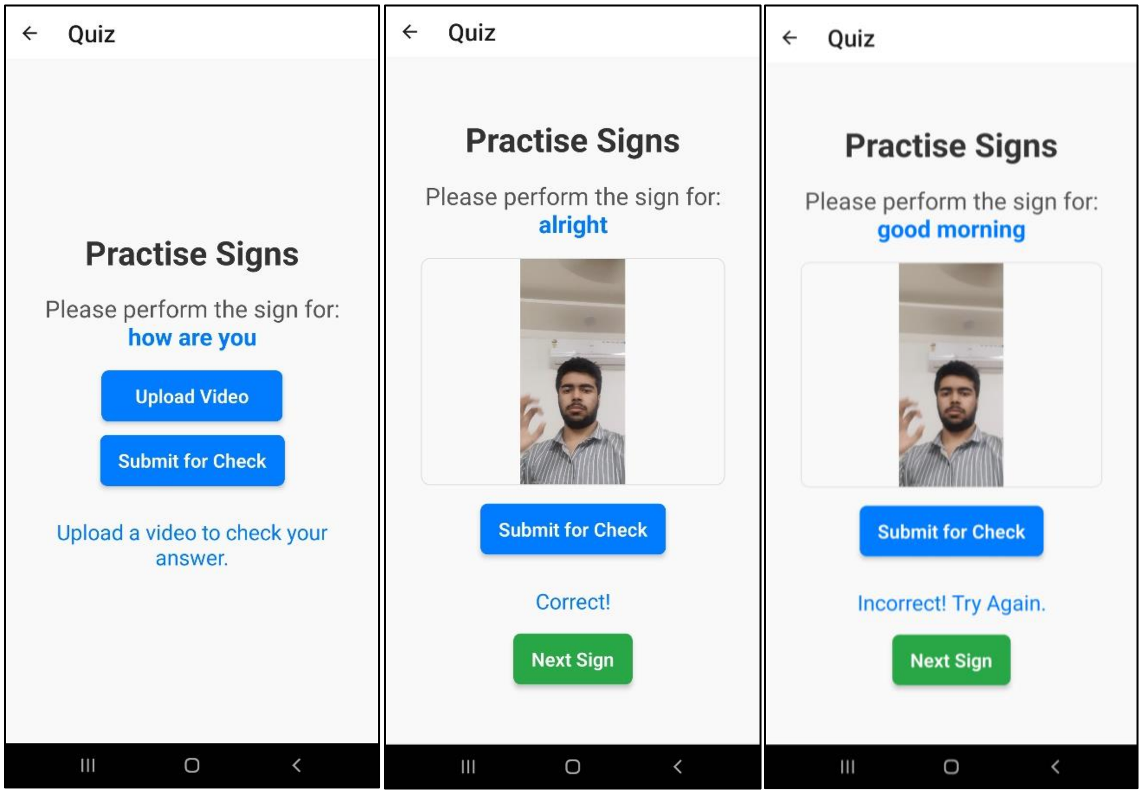
The models were evaluated based on their validation accuracy, recall, precision, and F1 score.

Initializing the Server Endpoint

After selecting the most optimal model, the next step was to build the mechanism for making real-time predictions using that model. The prediction process includes loading the preferred model, followed by defining the signs (“good morning,” “alright,” “good afternoon,” “how are you,” “hello”) that the model should return. Additionally, it involves extracting the frames of the input video and detecting the landmarks. If the landmarks are not detected, the code is designed to return the message: “No valid landmarks found in the video.” Otherwise, the landmarks are converted to array form, normalized to a compatible format, and inputted into the main model to predict the corresponding sign. A server endpoint was created using the Python library Flask to listen to client requests, including the video clips, and return the predicted sign associated with each video. The server was hosted locally on port 5000, and the function upload_video acts as the listener for this server, accepting a “POST” request with one argument: the input video.

Establishing the Client-Side Interface

The client-side interface was designed using the React Native framework. The necessary libraries were imported, including the “Expo” library and the “Expo Go” app, which simplified real-time app tracking during development. The interface included three options: the ability to predict a sign and to play a sign replication game, as shown in *Fig. 5*. In both the prediction section and the sign replication game, the user is prompted to pick a video from their device. Once the user selects a video and clicks the upload button, an HTTP request is sent to the Flask server, along with the selected video. After executing the server code, the result is displayed, as shown in *Fig. 6., and Fig. 7.*



Please perform the sign for: how are you

Upload a video to check your answer.

Please perform the sign for: alright

Correct!

Please perform the sign for: good morning

Incorrect! Try Again.

Figure 5. *Sign Practice Section of the App*

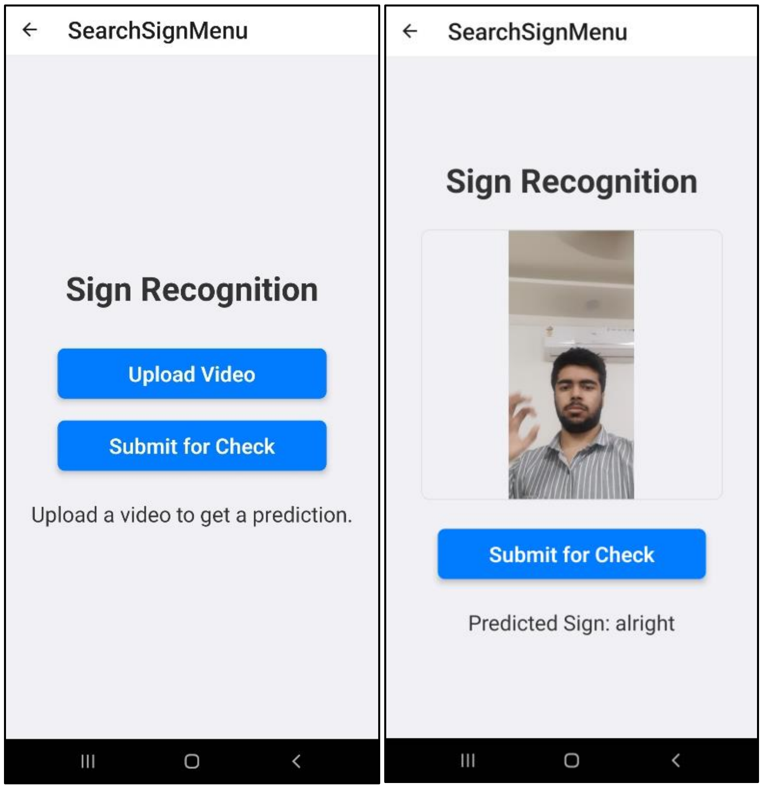


Figure 6*. Sign Prediction Section of App*

5. Results and Discussions

The accuracies of each model to the epoch cycle were visualized as shown in *Fig. 8., Fig. 9., Fig. 10., and Fig. 11.*

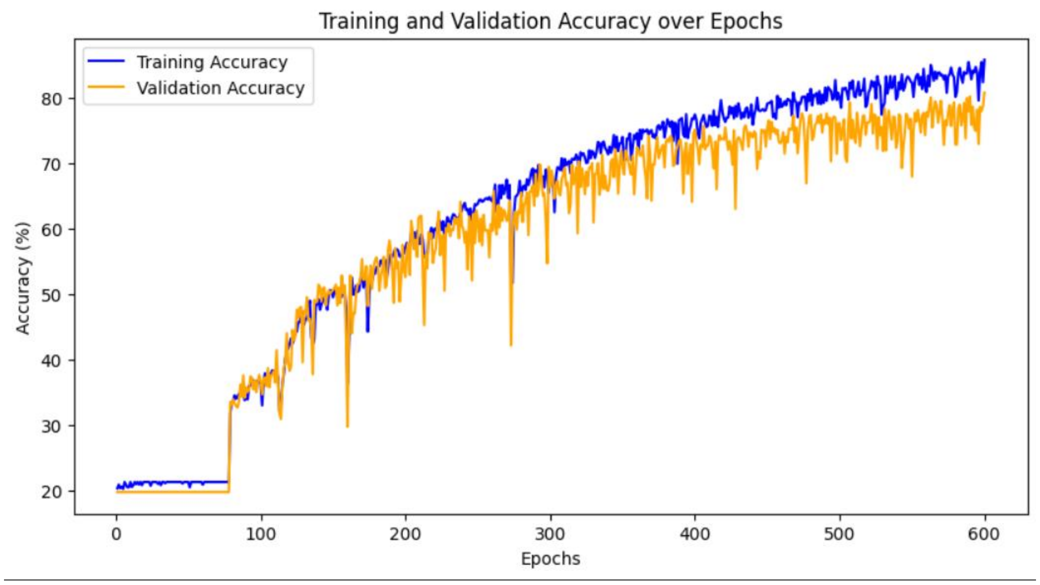


Figure 8*. Training and validation accuracy with each epoch of LSTM Model 1.*

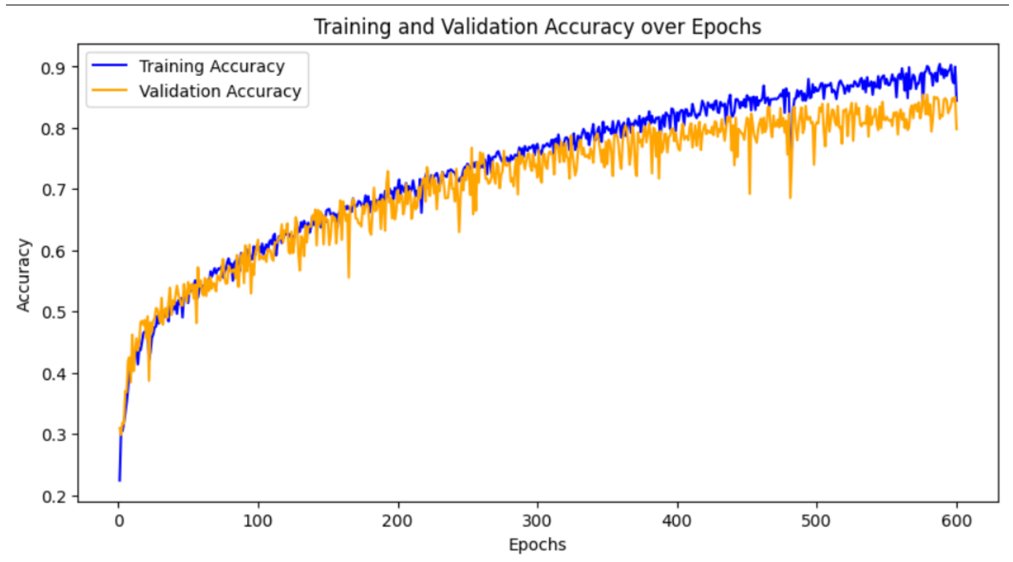


Figure 9*. Training and validation accuracy with each epoch of LSTM Model 2.*

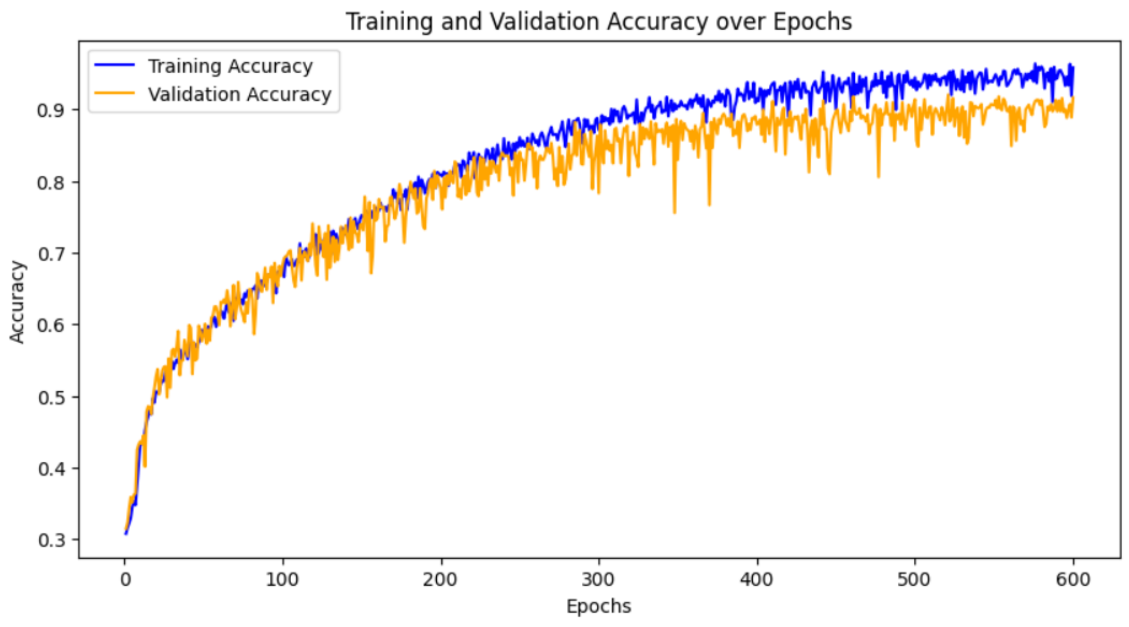


Figure 10. *Training and validation accuracy with each epoch of GRU.*

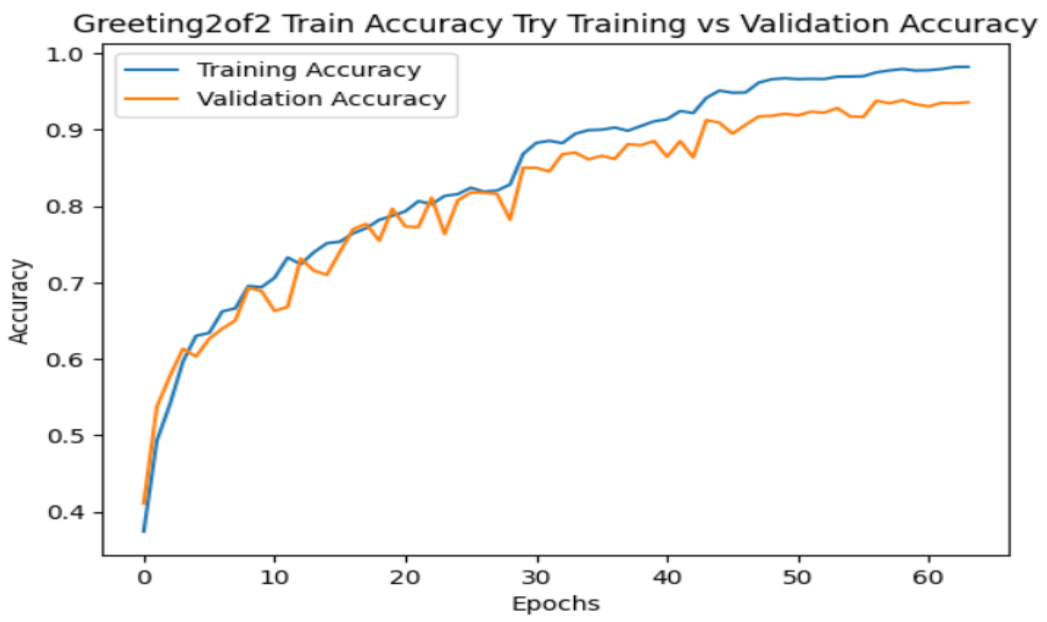


Figure 11. *Training and validation accuracy with each epoch of CNN+LSTM.*

After evaluating the performance of the models, the following results were observed:

- LSTM Model 1: Achieved a validation accuracy of 79.73% at 1200 epochs, with a training accuracy of 84.68% (precision, recall, and F1 score approximately 0.79) as shown in *Table 1*.
- LSTM Model 2: Achieved a validation accuracy of 85.82% at 1200 epochs, with a training accuracy of 94.83% (precision, recall, and F1 score approximately 0.85) as shown in *Table 2*.
- Pretrained Model (MobileNetV2): Achieved a validation accuracy of 90.16% at 1200 epochs, with a training accuracy of 96.35% (precision, recall, and F1 score approximately 0.9).
- GRU Model (TensorFlow): Achieved a validation accuracy of 91.66% at 600 epochs, with a training accuracy of 96.19% as shown in *Table 3*. This model performed well in terms of accuracy, but its precision, recall, and F1 score were significantly lower due to the inefficiency of the model architecture in handling long-sequence data.
- CNN+LSTM Model (TensorFlow): Achieved a validation accuracy of 93.61% at only 64 epochs, with a training accuracy of 98.25% (precision, recall, and F1 score approximately 0.94) as shown in *Table 4*.

Number of Epochs	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
100th	1.4222	36.23%	1.3737	36.53%
200th	0.9363	57.10%	1.0096	52.54%
300th	0.7136	68.47%	0.7786	66.31%
400th	0.5267	77.40%	0.6064	75.06%
500th	0.4842	79.99%	0.6235	76.06%

***Table 1**. Metrics of LSTM Model 1*

Number of Epochs	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
100th	0.8905	60.42%	0.8524	61.72%
200th	0.6884	70.71%	0.7192	69.81%
300th	0.5585	77.80%	0.7067	71.89%
400th	0.4390	82.86%	0.6028	77.56%
500th	0.3493	86.94%	0.5826	80.90%

Table 2. *Metrics of LSTM Model 2*

Number of Epochs	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
100th	0.7495	67.49%	0.7320	68.22%
200th	0.4925	80.96%	0.5398	78.98%
300th	0.3107	88.27%	0.5923	78.32%
400th	0.2471	91.19%	0.3583	87.91%
500th	0.1595	93.93%	0.3266	91.41%

Table 3. *Metrics of GRU*

Number of Epochs	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
10th	0.6813	68.72%	0.6745	68.89%
20th	0.4931	78.75%	0.4811	79.67%
30th	0.3578	85.32%	0.3633	85.03%
40th	0.2183	91.03%	0.2864	88.53%
50th	0.1016	97.10%	0.2089	92.10%

Table 4. *Metrics of CNN+LSTM*

Each model was then compared based on the highest training and validation accuracy achieved, as shown in *Table 5*. It was concluded that the CNN+LSTM model performed the best, achieving the highest training and validation accuracy, that too in the least number of epochs. This model also performed well on real-time video inputs and had the highest precision, recall, and F1 score. Compared to the other models, the CNN+LSTM model was much more stable, while the other models showed significant fluctuations in their accuracies.

Models	No. of epochs	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
LSTM Model 1	594	0.8322	0.4166	0.8165	0.4653
LSTM Model 2	963	0.9077	0.298	0.8966	0.4078
Pre-Trained LSTM (MobileNetV2)	1196	0.9617	0.1243	0.9191	0.3579
GRU	521	0.9558	0.1321	0.9208	0.2841
CNN+LSTM	59	0.9782	0.0772	0.9389	0.1829

Table 5*. The metrics of models at their highest achieved validation accuracy*

6. Conclusion and Future Work

This research successfully trained a model to predict sign language based on video inputs, achieving high validation accuracy (93.89%) along with promising performance metrics such as precision, recall, and F1 score. The study also successfully built a React Native app, that acted as an interface between users and the Python Flask server for sign prediction. The app allows users to learn and improve their knowledge of Indian Sign Language. Future work could explore additional signs and further improve the model's accuracy by expanding the dataset. Further, the ability to predict signs from a live camera feed could be added.

References

[1] A. Tyagi and S. Bansal, “Feature Extraction Technique for Vision-Based Indian Sign Language Recognition System: A Review,” 2021, pp. 39–53. doi: 10.1007/978-981-15-6876-3_4.

[2] H. Monga, J. Bhutani, M. Ahuja, N. Maid, and H. Pande, “Speech to Indian Sign Language Translator,” 2021. doi: 10.3233/apc210172.

[3] T. Kurre, T. Katta, S. A. Burla, and N. Neelima, “Real-Time Indian Sign Language Recognition Using Image Fusion,” in *Advances in Cognitive Science and Communications*, A. Kumar, S. Mozar, and J. Haase, Eds., Singapore: Springer Nature Singapore, 2023, doi: 10.1007/978-981-19-8086-2_58, pp. 599–605.

[4] B. Surya, N. V. S. Krishna, A. S. SankarReddy, B. V Prudhvi, P. Neeraj, and V. H. Deepthi, “An Efficient Real-Time Indian Sign Language (ISL) Detection using Deep Learning,” in *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2023, pp. 430–435. doi: 10.1109/ICICCS56967.2023.10142596.

[5] A. Ganpatye and S. Mane, “Motion Based Indian Sign Language Recognition using Deep Learning,” in *2022 2nd International Conference on Intelligent Technologies (CONIT)*, 2022, pp. 1–6. doi: 10.1109/CONIT55038.2022.9848275.

[6] A. Kumar, K. Thankachan, and M. M. Dominic, “Sign language recognition,” in 2016 3rd International Conference on Recent Advances in Information Technology (RAIT), IEEE, Mar. 2016, pp. 422–428. doi: 10.1109/RAIT.2016.7507939.

[7] C. Vogler and D. Metaxas, “Parallel hidden Markov models for American sign language recognition,” in Proceedings of the Seventh IEEE International Conference on Computer Vision, IEEE, 1999, pp. 116–122 vol.1. doi: 10.1109/ICCV.1999.791206.

[8] N. Cihan Camgoz, S. Hadfield, O. Koller, and R. Bowden, “SubUNets: End-to-end Hand Shape and Continuous Sign Language Recognition.”, [Online] Available: https://openaccess.thecvf.com/content_iccv_2017/html/Camgoz_SubUNets_End-To-End_Hand_ICCV_2017_paper.html

[9] A. Wadhawan and P. Kumar, “Deep learning-based sign language recognition system for static signs,” Neural Comput Appl, vol. 32, no. 12, pp. 7957–7968, Jun. 2020, doi: 10.1007/s00521-019-04691-y.

[10] Jie Huang, Wengang Zhou, Houqiang Li, and Weiping Li, “Sign Language Recognition using 3D convolutional neural networks,” in 2015 IEEE International Conference on Multimedia and Expo (ICME), IEEE, Jun. 2015, pp. 1–6. doi: 10.1109/ICME.2015.7177428.

[11] S. A. Mehdi and Y. N. Khan, “Sign language recognition using sensor gloves,” in Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP ’02., IEEE, 2002, pp. 2204–2206 vol.5. doi: 10.1109/ICONIP.2002.1201884.

[12] S. Jiang, L. Wang, Y. Bai, K. Li, and Y. Fu, “Skeleton Aware Multi-modal Sign Language Recognition.” [Online]. Available: https://github.com/jackyjsy/

[13] K. Amrutha and P. Prabu, “ML Based Sign Language Recognition System,” in 2021 International Conference on Innovative Trends in Information Technology (ICITIIT), IEEE, Feb. 2021, pp. 1–6. doi: 10.1109/ICITIIT51526.2021.9399594.

[14] R. Zuo, F. Wei, and B. Mak, “Natural Language-Assisted Sign Language Recognition,” in 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2023, pp. 14890–14900. doi: 10.1109/CVPR52729.2023.01430.

[15] R. Cui, H. Liu, and C. Zhang, “Recurrent Convolutional Neural Networks for Continuous Sign Language Recognition by Staged Optimization.” [Online] Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Cui_Recurrent_Convolutional_Neural_CVPR_2017_paper.html