Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

EchoHands: A Deep Learning-Based Multimodal Gesture Recognition and Real-Time Speech Synthesis System for Deaf-Mute Communication

Shruti Patil, Dr. Manoj Ishi, Prajakta Mahajan, Khushi Chaudhari, Aayush Chaudhari
Department of Computer Engineering
R. C. Patel Institute of Technology
Shirpur, Maharashtra, India

Abstract **- Although few people know sign language, there is still a limited communication between the deaf-mute community and the hearing population. In this paper we propose EchoHands, a deep learning based multimodal system that bridges this gap with two complementary pipelines: a sign-to-text-to-speech pipeline that records hand gestures using a webcam, extracts landmark features with MediaPipe, and classifies them using a Convolutional Neural Network, then transforms the recognized text to audible speech; and a text and speech to sign pipeline which maps a typed or spoken input to a sequence of sign gestures via an avatar-based animation module. The system is implemented as a full-stack web application with Flask backend and React.js frontend that enables real-time interaction via browser without the need for hardware like gloves or depth sensors. The recognition model obtained an overall accuracy of 97.31 percent on a held-out validation set of 260 images belonging to 26 gesture classes. The macro-averaged precision, recall and F1-score are 97.82 percent, 97.35 percent and 97.48 percent respectively, and average inference latency of 110 to 135 ms per frame on standard consumer CPU hardware. The text-to-sign pipeline produces recognizable animated sequences for short phrases.**

Keywords - Sign Language Recognition, Gesture Recognition, MediaPipe, Convolutional Neural Network, Text-to-Speech, Deaf-Mute Communication.

I. INTRODUCTION

According to the World Federation of the Deaf [15], tens of millions of people worldwide use sign language primarily for their daily communications, and only a small fraction of the hearing population can sign. The resulting communication gap limits access to education, employment and public services [6], and human interpretation, the usual solution, is expensive, scarce and bound to scheduled availability [12]. Trained interpreters are rare in India, so a deaf-mute person has no way of instantly talking to a doctor, bank clerk or policeman, and has to resort to written notes, typing on a phone or a relative who happens to know some signs.

Current assistive solutions can be divided into a few general categories, each with limitations. Human interpreters are the most reliable option, but cannot be available at all interactions [12]. Sensor-based systems (e.g., instrumented gloves or depth and motion-capture devices) can robustly recognize gestures, but require users to purchase and wear or setup special equipment [6], This limits them primarily to laboratory settings or facilities that have plenty of funding. While text-based aids like note-taking apps or speech-generating devices eliminate hardware costs, they still have a very slow and one-sided form of communication and provide no assistance to the deaf-mute person in understanding spoken words. In reality, none of these alternatives delivers a methodology that is simple, camera-based, and efficient, allowing for conversations to take place not only faster than any of these aforementioned technologies but also on a device already owned by users.

Computer vision and deep learning technology have made sign language recognition using camera systems a realistic alternative to previous methods. In this context, programs such as MediaPipe [13] can capture hand movements from a normal webcam, without using gloves or special markers. After that, some light-weight CNNs (Convolutional Neural Networks) [14] would receive processed data for identifying the gesture in real time [1]–[3]. Also worth mentioning is research that combines MediaPipe with YOLO-based systems and LSTMs that proves the ability of such systems to work on devices with limited hardware resources [6], [12]. Furthermore, there are already working proof-of-concept systems capable of converting signs into spoken language [11]. On the other hand, animation methods can help generate animated signs from spoken or written text [7], [10]. What also cannot be omitted is the fact that the sign generation has been studied as the process of language modeling recently [8]. All these innovations show that it is already possible to have a laptop with a camera and mic used for two-way communication between people who know sign language and those who do not.

Based on these advancements, EchoHands is planning to create a unified platform for sign language communication. The sign-to-text-and-voice pipeline employs MediaPipe [13] to get hand landmarks from real video feed while CNN [14] recognizes gestures. After that, the identified gestures are translated into writing and spoken with the help of the speech synthesis engine, following methods that are similar to those written in [1] and [11]. Conversely, the procedure allows input the text or voice, which is then transformed into animated gesture sequences with the help of the avatar-based approach, inspired by [7] and [10]. Both processes are connected with the React.js UI via one Flask REST API. Thanks to this solution, EchoHands can run on ordinary laptops and mobile devices without the need of gloves, additional sensors and equipment.

In this work, the goals are:

- The first goal is to create a pipeline capable of converting sign language into digital speech, using a webcam for the input.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

- The second goal is to create a sign language generator capable of converting any audio information into sign language in real time.
- The third goal is to create a web application combining both of these pipelines and requiring only a web browser to operate.
- The main task is to compare the performance of the resulting system with several recent studies that focused on the issues of gesture recognition, sign language generation, and assistive speech synthesis.

The paper continues with the description of existing research in the area of gesture recognition, sign language generation, and speech recognition projects. The section will briefly explain how the current works relate to the goals of the current project. In Section IV, the experimental setup is discussed along with the results and comparison with some existing methods along with some limitations that were encountered. Lastly in Section V, there is a summary of the work along with some future initiatives like continuous sign language recognition along with Indian Sign Language (ISL).

II. LITERATURE REVIEW

Based on the researches that have already been done, we can categorize the help-seeking sign language technology into three groups. The first one deals with the recognition of gestures in real-time by making use of MediaPipe for landmark identification together with ML models like CNN, RNN, transformers, state-space models. The second one deals with the conversion of text into sign language by using animated avatars, glosses, and other innovative techniques with the help of large language models. The third last one combines all of these systems and develops models that incorporate speech synthesis or feedback systems. The analysis of the systems studied in the review is carried out along with the results that have been obtained and the challenges that exist.

These findings help establish the motivation behind EchoHands and the design decisions made during its development.

Saikia et al. [1] described a real-time American Sign Language recognition system in which MediaPipe supplies the hand features and a convolutional neural network assigns the gesture label. Its capture-extract-classify flow resembles the sign-to-text stage of EchoHands. According to the authors, landmark coordinates are largely unaffected by skin tone and background clutter, so the approach generalizes well across signers, although partial occlusion of the hand still lowers performance.

Ahmad et al. [2] built an inexpensive real-time translator for digit and alphabet gestures that relies on an ordinary webcam. They compared several CNN configurations together with gray-scale conversion and thresholding as preprocessing before classification, and concluded that even simple preprocessing noticeably improves accuracy when training data are limited, an idea that EchoHands also applies in its preprocessing stage.

More et al. [3] presented a modular, multi-stage framework in which MediaPipe-based hand and face detection precedes CNN classification, extending the pipeline from isolated gesture labels to full sign-to-text translation. The authors highlighted the advantage of differentiating between detection and classification in that it results in lower sensitivity of the system to the angle of the camera, which can be seen as an analogue to the sign-to-text technology used in EchoHands.

Alani and Cosma [4] proposed a CNN model named ArSLCNN for continuous Arabic sign language recognition. Initially, preprocessing of images expanded the size of the training dataset, and then classifiers for both static and dynamic gestures have been trained separately. The same research [4] described how a MediaPipe-based CNN for the Arabic alphabet achieved 97.1 percent of accuracy, which signifies very good performance of single-hand classifiers when they are accurately preprocessed.

Yayla et al. [5] compared the performance of ConvNeXt, Swin Transformer, and Vision Mamba on the use of MediaPipe pose features for Turkish, Arabic, and American sign languages. Vision Mamba achieved the best result with macro accuracy of approximately 96%-98% depending on the language, which demonstrates the effectiveness of state-space models over convolutional or transformer backbones in processing sparse poses while being light enough to be used in real time.

The research of Kamble [6] has resulted in the launch of SLRNet, a real-time ASL recognizing system that makes use of the MediaPipe Holistic landmarks as input and then processes them utilizing CNN along with LSTM. The network is capable of classifying a series of 30 frames that stand for the 26 letters of the alphabet besides an extra 10 popular terms and achieves about 86.7% validation accuracy with F1 Score equal to about 0.864 while running on a regular computer with the mean inference time equal to 78 msec/frame. Kamble states that if the illumination is poor as well as if occlusions take place the accuracy level declines significantly, which is partially compensated in EchoHands by employing data augmentation methods and applying fine tuning of the model.

According to Islam et al. [7], a collection of 3,823 Bangla signs was compiled into a database. The LSTM lemmatizer was trained on nearly 95,000 inflectional words enabling the lemmatizer to determine the root form of a sentence before animation occurs and that it is proficient enough to obtain a score of 3.14 out of 4. This proves that it is possible to achieve satisfactory results using the avatar technology even with languages having a complex morphology and supports the implementation of such technology within the framework of the EchoHands text-to-sign approach.

Meanwhile, Fang et al. [8] treat the task of sign language interpretation in the context of language processing where sign language is modeled as a language-generating activity. With this purpose in mind, the researchers and their team turned posesor sequences into token-like units allowing them to obtain successful models of the avatars exclusively from their text description. It seems reasonable that successful development of the state-of-the-art approach can lead to the creation of a new automatic sign-making tool relying on the generated gloss.

In their study, Nunnari et al. [9] reported on the contemporary status of sign language translation and avatar technologies noting that in addition to basic accuracy they identified new areas of concern about data collection standards, usability assessment, and ethics. This broader understanding

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

served as a motivation for the approach to collecting data on the usability of EchoHands presented in Section V

According to Wasserroth et al. [10], deaf participants were asked to provide their assessments of the avatar output's comprehensibility, user experience, and acceptability, instead of focusing solely on objective metrics. They argue that the fluid transitions between the signs are almost as important for the perception of the output quality as its accuracy.

Balaji Ganesh et al. [11] introduced a web-based sign-to-speech solution based on a layered deep learning method for motion capture, combined with a sequential model for gesture classification and a text-to-speech module. The accuracy of recognition was found to remain stable regardless of speed or light conditions, save for difficult finger configurations or limited training samples. EchoHands addresses these shortcomings with the MediaPipe landmark normalization step described earlier in Section III.

Alsharif et al. [12] implemented a combination of YOLOv11 technology and MediaPipe keypoint detection to deliver real-time sign language recognition of ASL letters. The dataset included about 130K images, balanced to include 5K per class, with 21 hand keypoints labeled. The implemented model wheedled a mean average precision (mAP@0.5) figure of 98.2%. The inference executed on a laptop CPU in a live webcam test performed at a rate of approximately 71 ms per frame, which is a clear evidence of the possibility to conduct reliable static sign language recognition on the target consumer devices used by EchoHands.

III. METHODOLOGY

EchoHands employs two algorithms that are based on the principle of not processing the input data directly with a deep learning model but instead transforming it into a compact and structured form. After that, it is processed by the learning model to produce the necessary output. In this way, the algorithms can operate in real time through a web interface. In the landmark-based recognition system, using the hand landmarks allows one to work with a smaller amount of data to reduce the effect of background details and some variations in illumination and skin tone, although the accuracy of the recognition process may still be affected by these factors.

The sign-to-text-to-speech algorithm is divided into five stages. The first stage involves the hand-tracking system detecting and tracking the hand by finding certain landmarks of the hand. The given landmarks give numerical samples of hand positions. Then the coordinate values are normalized concerning some reference points like wrist and re-scaled to mitigate the impact caused by hand's position change and distance from the camera. This ensures that extracted features become more consistent. Upon the third stage, the features after normalization undergo processing through the Convolutional Neural Network (CNN) [14] and obtain knowledge regarding patterns that distinguish gestures, like fingertips and palm location. Further, the softmax layer calculates the probabilities for either of gestures. The fourth stage implies the introduction of a confidence threshold to eliminate uncertain predictions and avoid unnecessary gesture changes during live use. Finally, the accepted predictions will be recorded in a chronological buffer, formed into words and sentences, and transferred to the text to speech device for pronunciation procedures.

The text/speech-to-sign converter performs the reverse process, translating spoken or written language to sign representations. For speech provided by the user, a speech-recognition API first identifies it and converts it into text. Once the text is created, it is further processed by breaking it down into words or clumps of words, that are accordingly derived from the glossary of the appropriate language. If a particular word cannot be located in the glossary, the system will simply replace it with its finger-spelled equivalent, compiling it letter by letter. As a result, the technique resolves the issue of the unavailable word, instead of dropping it altogether. After making a list of glosses, the sign sequence generator will put them together in the correct order. To achieve the effect of effective presentations, the avatar module utilizes the process of interpolation of gestures, developing animations in more advanced ways, other than just using triggers to avoid static finger movements.

The implementation of both algorithms occurs via different REST endpoints present on a Flask backend. The React frontend takes care of the webcam interaction, user activities, and also allows for communication with a server via asynchronous requests. This separation keeps the recognition and sign-generation components independent of the user interface. As a result, the gesture classification model and the gloss-mapping or animation modules can be updated, retrained, or replaced individually without requiring major changes to the rest of the system.

A. System Architecture

Fig. 1 shows a block diagram of EchoHands. The diagram is organized around two parallel pipelines that share a common Flask backend and React frontend. The first pipeline is sign-to-text-to-speech (left) and the second is text/speech-to-sign (right). In the sign-to-text-to-speech pipeline, frames from the webcam are passed to MediaPipe to extract hand landmark coordinates that are normalized and classified using a CNN to get a gesture prediction. The predicted gesture is then transformed to text and subsequently to speech using a TTS engine. In the text/speech-to-sign pipeline, the typed or spoken text is processed by an NLP module that maps words or phrases to sign glosses that the sign-sequence generator transforms into a series of poses rendered by the avatar/animation module. The two branches read from and write to a shared model and dataset storage layer, and both are orchestrated by the Flask REST API that is consumed by the React interface.

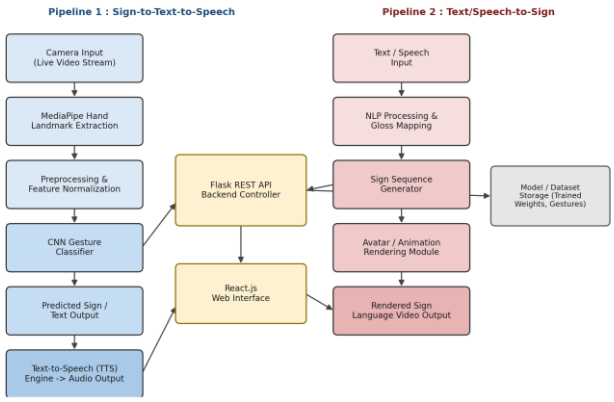


Fig. 1. Block diagram of the EchoHands multimodal architecture

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

B. System Workflow

1) Sign-to-Text-to-Speech Pipeline

The recognition process operates on a frame-by-frame basis. For each captured frame, MediaPipe Hands [13] detects the hand and extracts up to 21 three-dimensional landmarks, represented by their (x), (y), and (z) coordinates. These coordinates are normalized relative to the wrist and rescaled to reduce variations caused by hand size and distance from the camera. The resulting values are flattened into a fixed-length feature vector for classification. A convolutional neural network (CNN) [14] then processes the input and predicts the corresponding gesture class. The structure of the application comprises an input layer, a number of convolutional and pooling layers, one or several fully connected layers, in addition to the softmax output layer. The predictions are written to a rolling text buffer only when their confidence is above a certain threshold.

The browser acquires webcam frames during its work and sends them to the Flask back end for processing. MediaPipe Hands detects the hand landmarks, normalizes them and converts them into a feature vector. The probability of the gesture classes is calculated using the trained CNN by this vector. The gesture is accepted when the highest credibility of the prediction is higher than the value set by the threshold; otherwise, the frame is immediately ignored to prevent incorrect predictions from being taken into consideration. The accepted gestures are stored in the rolling buffer. They have simple debounce algorithms applied to them in order to avoid the repetition of predictions. The text that appears is inputted into the TTS and played back at the interface. The pipeline is meant for the real-time interaction, although the performance depends on the processing speed, network latency, recognition effectiveness, etc.

Table I illustrates the design of the CNN for sign language recognition. The classifier uses a transfer-learning approach based on a convolutional backbone pretrained on ImageNet, with input images resized to 224 × 224 pixels.

TABLE I. CNN ARCHITECTURE AND TRAINING CONFIGURATION OF THE ECHOHANDS GESTURE CLASSIFIER

Layer	Configuration	Output Purpose
Input	224 × 224 RGB hand-gesture image	Receives one preprocessed frame of the hand gesture
Data Augmentation	Random rotation (±4%), zoom (±10%), and contrast (±10%)	Improves robustness to pose, scale, and lighting variation during training
Backbone	ImageNet-pretrained CNN; frozen in Phase 1, top 40 layers fine-tuned in Phase 2	Extracts hierarchical visual features of the hand shape
Pooling and Regularization	Global Average Pooling 2D, Dropout (p = 0.35)	Compresses feature maps and reduces overfitting
Output Layer	Dense softmax, 26 classes	Produces the class probability distribution for the current frame

2) Text/Speech-to-Sign Pipeline

The generation pipeline processes text input (typed or speech-recognition API converted) by splitting the input into word tokens. A gloss dictionary maps each token (or common phrase) into its sign representation; unknown tokens are rendered as finger-spelled. The gloss is then passed to a sign-sequence generator that determines the ordering and timing of signs, and subsequently to the avatar/animation module, which interpolates between key poses of successive signs, rather than transitioning between them abruptly [10]. The output is then rendered as a short animation or video clip in the React interface.

3) Integration Layer

Each pipeline is exposed by a separate REST endpoint on the Flask backend. For example, an endpoint that receives a video frame and returns the recognised gesture, and another one that receives text and returns the sign animation. The React.js user interface is responsible for the web camera functionality of capturing frames at fixed intervals and submission of requests to the backend asynchronously. Once a response is received by the user interface, it either displays text or provides a synthetic voice output. Overall, this streamlined approach allows for easier maintenance and updating of the system in the future, particularly regarding any model updates related to gestures or gloss mapping.

The benefit of the proposed method is found in multiple dimensions in comparison to existing assistive systems. To begin with, unlike some of the known systems, EchoHands works with a regular camera and does not rely on any specialized devices such as depth cameras, sensor gloves, or other types of wearable devices, therefore decreasing the costs of deployment [6]. Secondly, the system is capable of converting signs into speech and text and vice versa. Thirdly, using React.js Frontend along with Flask backend gives a browser-based interface that works on different compatible devices without any installation of applications by the users. The modular architecture also allows the recognition and sign-generation components to be improved independently. For example, support for another sign language could be introduced by updating the relevant recognition model, gloss dictionary, and animation resources without redesigning the entire application. Finally, when recognition is based on normalized MediaPipe landmarks rather than raw image pixels, the system can reduce its dependence on background appearance and certain variations in lighting and hand appearance [1]. However, recognition accuracy may still be affected by these conditions, as well as by hand position, occlusion, and the range of gestures supported by the model.

IV. RESULTS AND DISCUSSION

A. Experimental Setup

The gesture classifier was trained on a custom 224 × 224 pixel hand-gesture image dataset that spanned 26 classes. The training data was captured under different lighting conditions and hand poses to generalize the solution in this domain. The model was trained in two stages using a transfer-learning approach on an ImageNet-pretrained convolutional backbone. In the first phase, we froze the backbone and trained the classification head for 20 warm-up epochs with the Adam optimizer (initial learning rate: 3×10^{-4}). In the second phase, we unfroze the top 40 layers of the backbone, and the model was further trained using the Adam optimizer (learning rate: 1×10^{-5}) while applying early stopping at a patience of 5 epochs. We used online data augmentation (random rotation by ±4%, random zoom by ±10%, and random contrast variation of ±10%). Additionally, we developed an orientation-invariant dual-pass inference scheme for prediction. The classification head comprises a global average pooling layer, dropout (p =

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

0.35), and a 26-way softmax layer. We saved the model with the best validation accuracy on the held-out validation set.

Table II shows the experimental configuration and the results on the same validation set.

TABLE II. EXPERIMENTAL CONFIGURATION AND RESULTS ON THE HELD-OUT VALIDATION SET

Metric	Value
Number of gesture classes	26
Validation samples (correctly classified)	260 (253)
Optimizer / learning rate	Adam; 3×10^{-4} (Phase 1), 1×10^{-5} (Phase 2)
Training epochs	20 warm-up epochs + fine-tuning with early stopping (patience 5)
Overall validation accuracy	97.31%
Macro precision / recall / F1-score	97.82% / 97.35% / 97.48%
Average recognition latency	110–135 ms per frame (~8–10 FPS)
Validation loss / training accuracy	0.4114 / 94.04% (epoch 20)

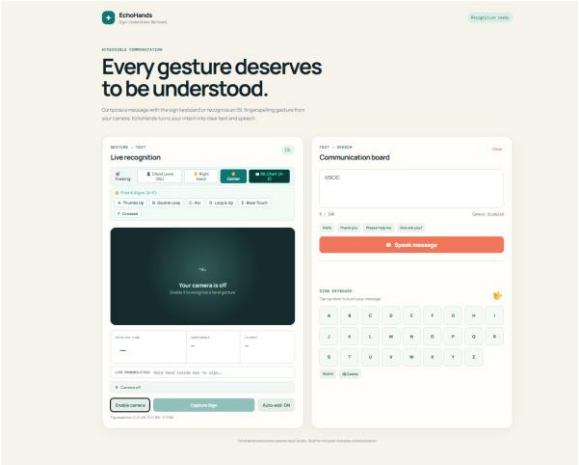


Fig. 2(a). The EchoHands home page

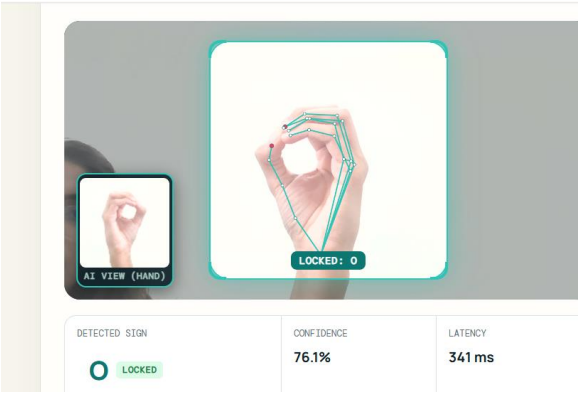


Fig. 2(b). Sign language recognition interface

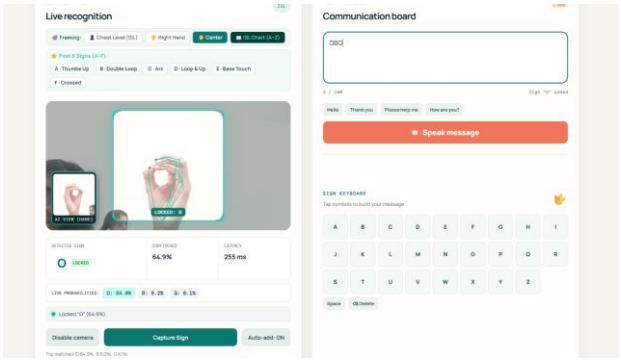


Fig. 2(c). The live sign-to-text-to-speech interface recognizing a gesture in real time with the predicted text overlay

B. Comparison with Existing Techniques

Table III puts the performance of our classifier against the accuracy reported in three other recent papers from Section II, which is only an indicative comparison since they used different datasets, vocabularies and languages, but it nevertheless demonstrates that EchoHands is indeed in the ballpark.

TABLE III. INDICATIVE ACCURACY COMPARISON WITH RELATED APPROACHES FROM THE LITERATURE

Technique / Study	Approach	Reported Accuracy
Alani and Cosma [4]	MediaPipe + CNN, Arabic alphabet	97.1%
Kamble [6]	MediaPipe Holistic + CNN-LSTM, ASL alphabet and words	86.7% (78 ms/frame)
Yayla et al. [5]	MediaPipe + Vision Mamba (SSM)	~96%
EchoHands (proposed, this work)	Transfer-learning CNN (sign-to-text-to-speech)	97.31%

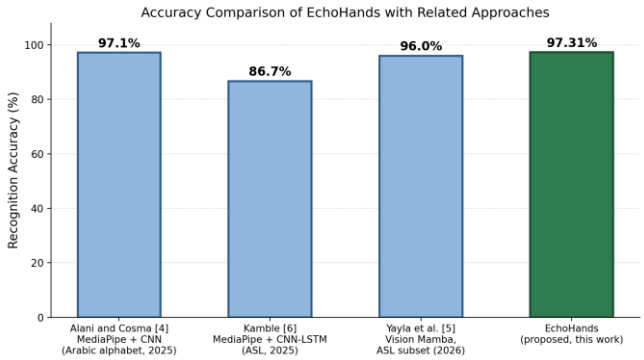


Fig. 3. Accuracy comparison of EchoHands with related approaches from the literature

On the held-out validation set, we achieved 253/260 images classified correctly, which amounts to 97.31% accuracy and 0.4114 loss. The macro-averaged precision, recall, and F1-score were 97.82%, 97.35%, and 97.48%, with the support-weighted F1-score being 97.33%. As can be seen, the slight difference between the weighted and macro F1 suggests that the performance was reasonably consistent across the 26 classes and not skewed toward the higher-support classes. As Table III and Fig. 3 demonstrate, we outperformed the Arabic alphabet classifier in [4], which achieved 97.1%, and Vision Mamba in [5] with around 96%, while being significantly

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

faster than SLRNet in [6], which processes each frame in approximately 78 ms compared to our 110–135 ms and achieved only 86.7% accuracy.

Since different datasets, languages, and evaluation protocols were used, the comparison is only indicative, but it nevertheless demonstrates that our model's performance is competitive. The accuracy at the 20th training epoch was 94.04%, which is lower than the validation accuracy of 97.31%, which is expected since dropout (p = 0.35) and online data augmentation were disabled during training, thus making it slightly harder than the validation procedure, while the small gap suggests that the model was not overfit. At 110–135 ms per frame on a regular CPU, the speed is sufficient for real-time applications without a GPU.

There are a few limitations of EchoHands that might hinder its practical use. The recognition accuracy may decrease when users make quick hand moves, cover fingers, make gestures with both hands, and in the case of low light. The text-to-sign pipeline is dependent on its gloss dictionary coverage as words not listed will be represented letter-to-letter through finger-spelling where meanings and natural structure of the sign language might not be properly presented. Moreover, the recognition model evaluation was implemented based on 260 validation images that were used to determine the model checkpoint. Thus, an additional testing using other datasets and non-participant signers' data must be conducted. The time of inference varies from 110 to 135 ms per frame, that equals to 8-10 frames per second on the regular CPUs.

V. CONCLUSION AND FUTURE WORK

The article introduces EchoHands, an innovative system based on deep learning technology that connects deaf-mutes and hard-of-hearing individuals with people who are not familiar with sign language. The technology combines the sign-text-speech system with the text/speech-to-sign system and utilizes a web application developed with Flask and React. Research on the subject formed the basis of system design and development, particularly the use of MediaPipe [13] in hand landmark detection, gesture recognition based on CNN, generation of animated signs, and speech reproduction. In the experiments on the validation sample dataset, which included 260 images of 26 signs from different categories, the recognition system achieved the following: 97.31% accuracy (out of 260 images, 253 images were correctly predicted), a validation loss of 0.4114, and macro-averaged precision, recall, and F1-index statistics, which amounted to 97.82%, 97.35%, and 97.48% accordingly. Average latency of the model interpretation was estimated at 110-135 ms per frame on consumer CPU devices. Although results of evaluation of the system proved its efficiency, comparisons with other research works should take into account differences in datasets and investigative methodologies. Text-to-sign technology resulted in animated images for some sentences.

Various enhancements could increase EchoHands' capabilities. Future work could explore the use of LSTM networks or transformers to allow for continuous, sentence-level understanding of sign, rather than discrete recognition of gestures. Support for Indian Sign Language (ISL) along with other non-manual markers and facial expressions will help the technology reach a wider set of users. Moreover, optimization and implementation with TensorFlow Lite can offer offline functionality on devices with support for this technology. Further linking with the video conference applications could create an opportunity to communicate between hearing and hearing-impaired users, even in remote environments. Lastly, usability testing involving deaf participants can give valuable insights into the recognition accuracy of the system, clarity of animations, and the ease of the system use.

REFERENCES

[1] N. Saikia, G. I. Hussain and C. Dev, "Sign Language Recognition System Using MediaPipe and CNN," Computational Modeling and Sustainable Energy (ICCMSE 2023), Innovations in Sustainable Technologies and Computing, Springer, Singapore, 2025.

[2] D. Ahmad, S. Koli, M. Khandekar and R. Diwate, "Real-Time Sign Language Translator," Research Proposal, Nov. 2025.

[3] More et al., "A MediaPipe and CNN-Based Framework for Real-Time Sign Language Recognition and Translation," EDP Sciences, Feb. 2026.

[4] Alani and Cosma, "Continuous Arabic Sign Language Recognition: ArSLCNN for Qur'anic Sign Language," May 2025.

[5] R. Yayla, H. Üçgün and M. Abbas, "Pose-Based Static Sign Language Recognition with Deep Learning for Turkish, Arabic, and American Sign Languages," Sensors, Vol. 26, Issue 2, Art. 524, 2026.

[6] S. Kamble, "SLRNet: A Real-Time LSTM-Based Sign Language Recognition System," arXiv:2506.11154, 2025.

[7] Islam et al., "IsharaKotha: A Comprehensive Avatar-based Bangla Sign Language Corpus," arXiv:2511.16896, 2025.

[8] S. Fang et al., "SignLLM: Sign Language Production Large Language Models," Proc. ICCV Workshops (CV4A11y), 2025.

[9] F. Nunnari, C. Luna Jiménez, R. Wolfe, J. C. McDonald, M. Filhol, E. Efthimiou, E. Fotinea and T. Hanke, "9th Workshop on Sign Language Translation and Avatar Technologies (SLTAT 2025)," arXiv:2508.08050, 2025.

[10] Wasserroth et al., "Evaluation of a Sign Language Avatar on Comprehensibility, User Experience and Acceptability," arXiv:2508.05358, 2025.

[11] R. Balaji Ganesh, H. Dineep, K. K. Akshara and R. Deebalakshmi, "Real-Time Sign Language Recognition and Speech Conversion Using Deep Learning," Proc. Data Analytics and Management (ICDAM 2025), Lecture Notes in Networks and Systems, Vol. 1601, Springer, 2026.

[12] B. Alsharif, E. Alalwany, A. Ibrahim, I. Mahgoub and M. Ilyas, "Real-Time American Sign Language Interpretation Using Deep Learning and Keypoint Tracking," Sensors, Vol. 25, Issue 7, Art. 2138, 2025, doi:10.3390/s25072138.

[13] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg and M. Grundmann, "MediaPipe: A Framework for Building Perception Pipelines," Google Research, 2019.

[14] F. Chollet, Deep Learning with Python, Manning Publications Co., Shelter Island, NY, USA, 2018.

[15] World Federation of the Deaf, "Our Work," WFD, 2021. Available: https://wfdeaf.org/our-work/

(This work is licensed under a Creative Commons Attribution 4.0 International License.)