International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(14s) 2026 pp. 913-921
https://doi.org/10.70917/ijcisim-2026-4288
Article

A Real-Time Web-Based Speech-to-American Sign Language Translation System Using a Pre-Trained CResD-Net

Bhagya Shree[1], Sonal Chawla[2]

[1]Department of Computer Science and Applications, Panjab University, Chandigarh, India.
shreebhagya91@gmail.com
[2]Department of Computer Science and Applications, Panjab University, Chandigarh, India.
sonal_chawla@pu.ac.in

Abstract: Communication between hearing disabled and society remain challenging as both rely on different modes of communication. Most of existing work in this field converts speech into text in offline mode. It makes them unsuitable for real-time conversations. This paper presents a real-time speech-to-American Sign Language (ASL) translation. Django framework is used to create front-end of the system. First, this system captures speech from a web application using browser microphone. Second, it processes the speech to convert in waveform and remove the silence. Third, it recognized the speech using a pretrained speech command recognition model. Then, it maps the recognized command to its corresponding American Sign Language video from the WLASL dataset. Finally, it displayed the predicted command and sign video on the browser. The system was evaluated using real-time speech input from five participants. Translation accuracy and average response time were evaluated using real-time speech. The experimental results demonstrate that the system can accurately recognize predefined speech commands and display the corresponding ASL videos interactively to users.
Keywords: American Sign Language, Speech command recognition, WLASL, Real-time translation, Assistive technology, Django.

1. Introduction

Communication is an important part of everyday life. It helps people share ideas, ask questions, learn new things and build relationships. However, communication between society and hearing disabled is often difficult because they use different ways to communicate. American Sign Language (ASL) as their primary language, while society communicate through speech. Both don't know each other's languages. This difference can make everyday conversations slow and challenging. Many technologies have been developed to reduce this communication gap. A system that directly converts speech into sign language can make communication more natural and easier.

Several speech-to-sign language systems have been proposed in recent years. These systems generate sign language using virtual avatars and complex graphics. These systems may be difficult to develop, deploy or use in real-world situations. A real-time speech-to-sign language translation system is presented in this work. The system listens to speech through a web browser, recognizes predefined speech commands and immediately displays the corresponding American Sign Language video. The complete application is developed using the Django framework. This application can run through a standard web browser. The performance of the system was evaluated using recognition accuracy and average response time.

The remainder of this paper is organized as follows. Section 2 reviews related studies on speech-to-sign language translation. Section 3 identifies the research gap identified from the work. Section 4 presents the research methodology of the system. Section 5 describes the implementation of the proposed system in open-source programming language python. Section 6 explains the evaluation results. Section 7 discusses the performance of this

system. Section 8 outlines the threats to validity and the limitations of the study. Finally, Section 9 concludes the paper and highlights directions for future work.

2. Related Work

Recent advances in speech-to-sign language translation have focused on improving recognition accuracy and real-time communication. Laaidi et al. [1] proposed an enhanced CNN-LSTM model with attention mechanisms and data augmentation for speech recognition. The model achieved 97.17% accuracy on an Arabic speech command dataset. Sardjono and Purwanto [2] compared 1D CNN, 2D CNN and 3D CNN architectures for voice command recognition. Their results showed that the 3D CNN achieved the highest recognition accuracy. Mnuarbek et al. [3] presented a speech-to-sign language translation system for Kazakh Sign Language. Their framework employed the NVIDIA FastConformer model for automatic speech recognition. Then text segmentation was performed. A Weighted Hybrid Similarity Score algorithm for sign retrieval was applied. A dataset containing 1,200 manually generated sign samples, including 95% static images and 5% dynamic gesture videos. The automatic speech recognition module achieved an average Word Error Rate (WER) of 10.55%, while the complete speech-to-sign translation system achieved 85.8% word translation accuracy, 70.5% sentence translation accuracy and an average response latency of 310 ms. Bittner et al. [4] introduced the S-Edge architecture, which reduced computational cost through temporal down-sampling while maintaining an accuracy between 90% and 95%. Liu et al. [5] proposed a dynamic convolution framework for keyword spotting and achieved 97% recognition accuracy on the Speech Commands dataset. Sun et al. [6] proposed a Swin-Transformer-based keyword spotting model that combined Temporal Convolutional Networks with MFCC features. Their method achieved 98.01% accuracy on the Google Speech Commands dataset while using fewer model parameters than conventional Transformer models. Gambhir et al. [7] developed a Conv2D-based architecture for low-resource keyword spotting and achieved 91.60% accuracy on the Speech Commands dataset.

Several researchers have also explored hybrid deep learning models for automatic speech recognition. Dhakal et al. [8] combined CNN, ResNet and Bidirectional LSTM for Nepali speech recognition and reported a Character Error Rate (CER) of 17.06%. Oruh et al. [9] compared ResNet, DenseNet, VGG-16 and LSTM-RNN models and found that LSTM-RNN achieved the highest recognition accuracy of 99.36%. Dhanjal and Singh [10] developed a multilingual speech-to-sign language translation system using MFCC, Hidden Markov Models (HMM) and 3D avatar generation, achieving 91% accuracy for English speech translation. Peguda et al. [11] proposed a multilingual speech-to-sign framework for six Indian languages using Wavelet-based MFCC, Gaussian Mixture Models (GMM) and LSTM for speech recognition and translation. Tang et al. [12] developed the Howl wake-word detection system using SpecAugment, noise augmentation and a Res-8 model. Their system achieved 97.8% accuracy on the Speech Commands dataset. Jayalakshmi et al. [13] presented an Indian Sign Language translation system that combined speech recognition, natural language processing and HamNoSys animation, reporting a Word Error Rate (WER) of 10.10%. Patel et al. [14] proposed the ES2ISL framework using Google Speech API, natural language processing and 3D avatar animation, achieving an average translation accuracy of 77%. Filippidou and Moussiades [15] compared IBM, Google and Wit speech recognition services. It was found that Google Speech Recognition produced the lowest recognition error. Georgila et al. [16] also compared several commercial speech recognition systems across multiple dialogue datasets. It also highlighted performance differences under different application domains.

Earlier studies established the foundation for speech recognition and sign language translation. Shahriar et al. [17] developed a smartphone-based communication system for Bangla speech and sign language translation. It achieved an average accuracy of 84.71%. Këpuska [18] compared Microsoft Speech API, Google API and CMU Sphinx for automatic speech recognition using Word Error Rate. El-Gayyar et al. [19] proposed a cloud-based framework for translating Arabic speech into Arabic Sign Language using Google Speech API and natural language processing. Gaida et al. [20] compared open-source speech recognition using Kaldi and CMU Sphinx toolkits. López-Ludeña et al. [21] presented a rule-based speech-to-sign language translation system that combined example-based and Interlingua translation methods for Arabic Sign Language. Overall, these studies highlight the growing potential of artificial intelligence to improve communication between society and hearing disabled.

3. Research Gap

Previous studies have shown that speech recognition models can achieve high accuracy. However, most existing systems either generate animated avatars, translate speech into text before sign generation or are evaluated only in offline environments. Only a few studies have developed a lightweight web-based application that directly converts live speech into real American Sign Language (ASL) videos.

There is also lack of real-time system evaluation. Most studies mainly report recognition accuracy or word error rate of the speech recognition model. Evaluation metrics such as response time and translation accuracy during live user interaction are often not reported.

To address these limitations, this work integrates a pretrained CResD-Net speech recognition model with the WLASL [22] dataset. A Django-based web application is deployed to achieve the task. The system converts spoken commands directly into corresponding ASL videos in real time using a browser interface. The proposed system was evaluated using live speech input using predefined words. The evaluation was done using translation accuracy and response time of conversion.

4. Research Methodology of Proposed System

The proposed system was designed to translate spoken commands into their corresponding American Sign Language (ASL) videos in real time. The Research methodology can be divided into five phases. Phase 1 captured the user's speech through a microphone and recorded the audio signal. Phase 2 preprocessed the recorded audio by removing silence and converting it into a spectrogram. Phase 3 recognized the spoken command using the pretrained CResD-Net speech recognition model. Phase 4 translated the recognized command by matching it with the corresponding American Sign Language video from the WLASL dataset. Finally, Phase 5 displayed the retrieved sign video in the web browser. The system also displayed the recognized command before presenting the corresponding ASL video. The complete methodology is illustrated in Figure 1.

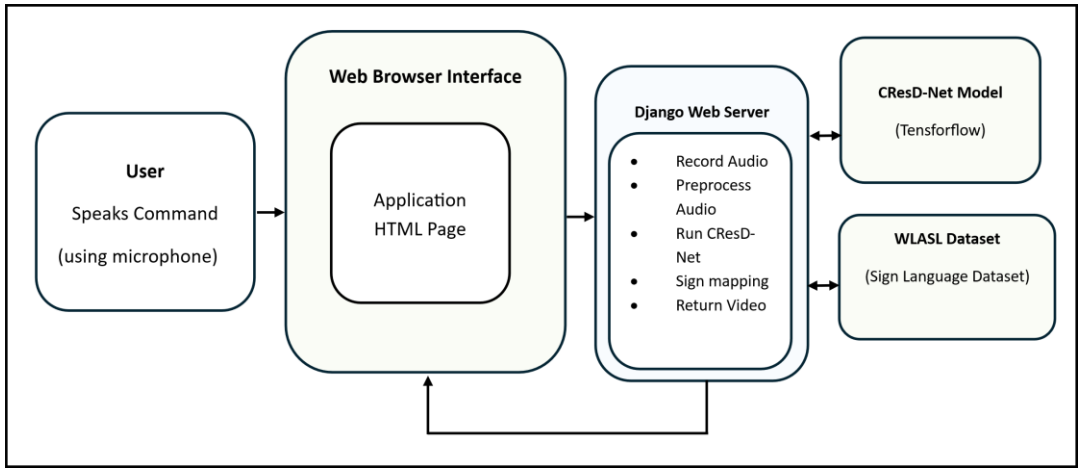

Figure 1. Methodology of the Proposed Real-Time Speech-to-ASL Translation System

A. Phase 1: Speech Acquisition

Speech was captured through the microphone available on the user's device using the SoundDevice library. The system automatically detected the available audio input device. If no microphone was found, the default system input device was used. The system recorded a 3-second audio clip at a sampling rate of 16 kHz. During recording, the Root Mean Square (RMS) energy of the signal was calculated to verify that speech had been captured. If the RMS value was below a predefined threshold, the recording was considered background noise. The user was asked to record again. The Direct Current offset was removed from the waveform to center the signal around zero. The system then identified the highest energy one-second speech segment by analyzing the audio in 50 ms windows. This approach removed the silence at the beginning and end of the recording while preserving the spoken command. Finally, the selected speech segment was normalized. It was saved as a WAV file for further processing.

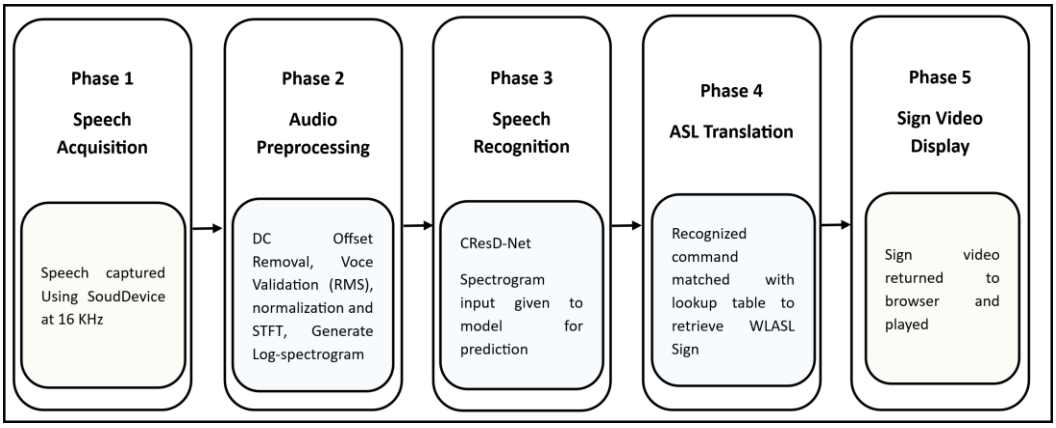

Figure 2. Five-tier architecture of the speech-to-ASL system.

B. Phase 2: Audio Preprocessing

The recorded WAV file was preprocessed before being passed to the speech recognition model. The audio signal was decoded using TensorFlow. It was adjusted to a fixed length of 16,000 samples by trimming longer recordings or padding shorter ones with zeros. A Short-Time Fourier Transform (STFT) was then applied to convert the waveform into a spectrogram. The logarithm of the spectrogram magnitude was computed to improve feature representation and reduce variations in signal amplitude. Finally, batch and channel dimensions were added to match the input format.

C. 3: Speech Recognition

Speech recognition was performed using the previously trained CResD-Net (Convolutional-Residual-Dense Network) model. The model was trained on the Google Speech Commands V2 dataset containing 35 command classes. The CResD-Net architecture consists of an initial 2D Convolutional Neural Network (CNN) layer, followed by four ResNet residual blocks and a Deep Neural Network (DNN) classifier. The 2D CNN with a 7×7 kernel and stride of 2 extracts low-level temporal and spectral features from the input spectrograms, such as formants and phoneme patterns. These features are then passed through the four ResNet blocks with progressively increasing filters 64, 128, 256 and 512. Residual skip connections enable efficient gradient propagation and facilitate the learning of complex hierarchical acoustic representations. The extracted high-level features are flattened and fed into a DNN comprising four fully connected Dense layers with 64, 128, 256 and 512 neurons, followed by a Softmax output layer that computes class probabilities and predicts the target speech or keyword class. This combination of CNN-based feature extraction, ResNet-based deep representation learning and DNN-based classification provides robust and accurate speech recognition performance. The trained TensorFlow model was loaded once when the application started and remained in memory throughout execution. It resulted in reducing prediction time. The generated spectrogram was passed through the model, which predicted the probability for each command class. The command with the highest probability was selected as the recognized speech command. The recognized command was displayed on the web interface before the corresponding ASL video was presented to the user. The offline performance of CResD-Net on the Google Speech Commands dataset served as the baseline performance. The real-time evaluation reported in this work measured the performance of the complete deployed system.

D. Phase 4: ASL Translation

After speech recognition, the predicted command was translated into American Sign Language using a predefined lookup table. Each of the 35 recognized commands was mapped directly to its corresponding video in the Word-Level American Sign Language (WLASL) dataset. Since the proposed system operated on isolated command words. No natural language processing or grammatical restructuring was required. The translation process therefore consisted of identifying the corresponding WLASL video associated with the recognized command and retrieving its file path.

E. Phase 5: Sign Video Presentation

The retrieved WLASL video was transmitted through the Django server and displayed using the browser's HTML5 video player. The browser automatically played the returned sign video after successful recognition, allowing users to immediately observe the translated ASL sign. Since pre-recorded human sign videos were used instead of

avatar animation, the translation process required minimal computational resources while preserving the natural appearance of sign language.

5. Implementation

The proposed speech-to-American Sign Language (ASL) translation system was implemented using the Django web framework, with TensorFlow serving as the deep learning inference engine. The user interface was developed using HTML, CSS and JavaScript. It enabled users to access the system through a standard web browser. The interface allowed users to record speech, view the recognized command and play the corresponding Sign.

Speech acquisition was performed using the SoundDevice library. The available audio input devices microphone was detected automatically. A three second audio clip was recorded at a sampling rate of 16 kHz to ensure that the complete spoken command was captured. The recorded signal was then validated using the Root Mean Square (RMS) energy to determine whether sufficient speech had been recorded. Recording was saved as a standard WAV file. The recorded audio was preprocessed using TensorFlow before being passed to the recognition model. The waveform was padded or truncated to 16,000 samples. A Short-Time Fourier Transform (STFT) was applied to generate a log spectrogram. The logarithmic transformation improved the representation of speech features and reduced the effect of amplitude variations. The resulting log spectrogram was reshaped to match the input dimensions required by the trained CResD-Net model.

The pretrained CResD-Net model was loaded once during Django application startup. It remained in memory throughout execution. It eliminated repeated model loading for each prediction request. Speech recognition was performed through a single forward pass of the network and the command with the highest prediction probability was selected as the recognized output together with its confidence score. The predicted command was matched against a predefined lookup table that maps each of the 35 Google Speech Commands to its corresponding WLASL video. For 2 words fingerspelling were also generated as same were not there in WLASL dataset. The system translated isolated speech commands. The translation process consisted of a direct lookup without requiring additional natural language processing or grammatical transformation. The Django server returned the recognized command and the corresponding video path. The browser then displayed the recognized command and automatically played the associated ASL video using the HTML5 video player.

To achieve better performance, the implementation combined in-memory model loading, efficient STFT-based preprocessing. The lookup-based sign retrieval and browser-based video playback was done for lightweight system. The total response time was affected by speech recording, processing, command prediction, network communication and sign video loading. The web application for real-time speech to Sign language conversion is illustrated in Figure 3.

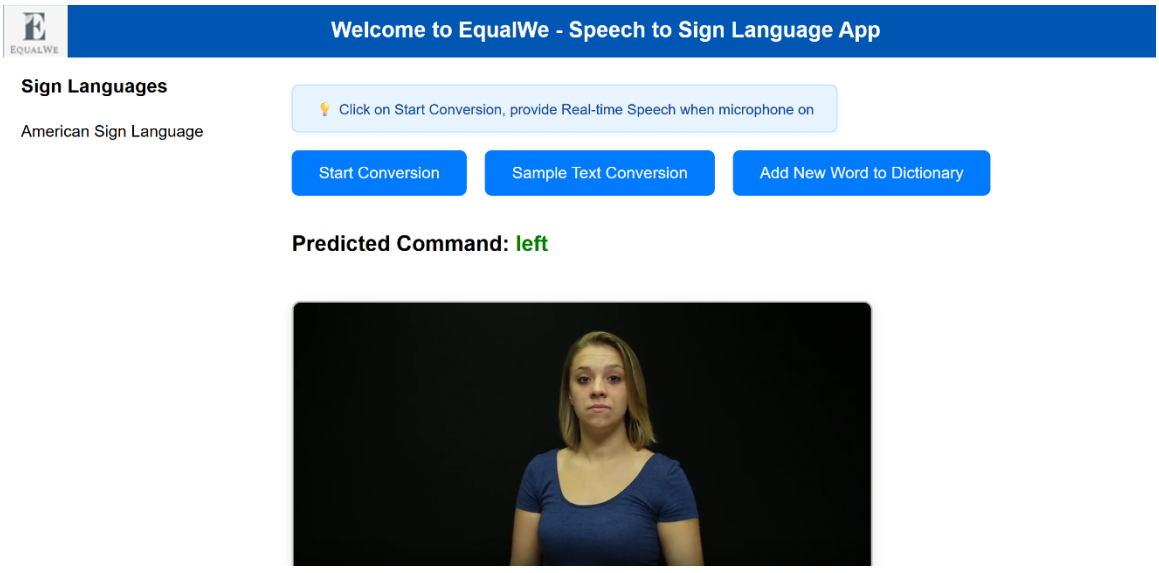

Figure 3. Real-time Web Application for Speech to Sign Language.

6. Evaluation Results

The proposed system was evaluated using live speech collected through the developed web application. The evaluation was designed to verify whether the deployed system could accurately recognize spoken commands and translate them into the corresponding American Sign Language (ASL) videos in real time. Five users including two male and three female speakers, participated in the evaluation. Each participant spoke the 35 commands supported by the Google Speech Commands dataset using their natural speaking style. Participants had different speaking accents and pronunciation styles to examine whether the system could reliably recognize commands under normal real-world usage. The recognized command was then translated into the corresponding ASL video using the WLASL dataset.

Since the speech recognition model had already achieved 99.02% offline recognition accuracy on the Google Speech Commands V2 dataset in the previous study. The objective of this evaluation was not to retrain or revalidate the recognition model. It was to assess the performance of the complete real-time system. Therefore, the evaluation focused on real-time recognition accuracy and system response time while processing live microphone input. All experiments were conducted under normal indoor conditions using the browser-based application and a standard microphone. Table 1 shows the parameters used in the evaluations.

Table 1. Parameters of Evaluation.

Parameter	**Value**
Participants	5 (2 Male, 3 Female)
Supported speech commands	35
Speech dataset	Google Speech Commands V2
Sign language dataset	WLASL
Recording device	Browser microphone
Recording environment	Indoor environment
Evaluation metrics	Real-time recognition accuracy, Response time

The performance of the proposed system was evaluated using two evaluation metrics.

Real-time recognition accuracy measured the percentage of spoken commands that were correctly recognized by the system. As each recognized command is directly mapped to its corresponding WLASL video, successful recognition also results in correct ASL translation.

$$\text{Recognition Accuracy (\%)} = \frac{\text{Number of Correctly Recognized Commands}}{\text{Total Spoken Commands}} \times 100$$

Response time measures the total time required for speech o Sign language conversion. The measured time includes speech processing, feature extraction, model prediction, command lookup and video retrieval.

$$\text{Average Response Time} = \frac{\sum_{i=1}^{N} T_i}{N}$$

where T_i represents the response time for ith spoken command and N is the total number of evaluated commands. Table 2 presents the evaluation results achieved by the system in real-time environment.

Table 2. Real-Time System Performance.

Parameter	**Value**
Real-time recognition accuracy (%)	90.67
Average response time (ms)	0.93 (3 seconds recording + 0.93 seconds for prediction and sign generation)

The results presented in Table 2 demonstrate the effectiveness of the proposed browser-based speech-to-ASL translation system when used with live speech from speakers having different accents and pronunciation styles. Real-time accuracy was 90.67%, which was lower than the 99.02% offline accuracy previously reported for CResD-Net on clean speech. The average audio recording duration was approximately 3 seconds. The mean model prediction and sign-video retrieval time was 0.93 seconds per command. The evaluation confirmed that the system could accurately recognize supported speech commands with a low response time. This ensures that system is fit for real time translation.

7. Discussion

The experimental results demonstrated that the proposed system can successfully translate spoken commands in real time. The evaluation was conducted using live speech from two male and three female participants with different speaking accents and pronunciation styles. The system was able to recognize the supported speech commands accurately. It indicated that the pretrained CResD-Net model generalises well to real-time microphone input.

Unlike the previous study, which evaluated the CResD-Net model using the Google Speech Commands V2 dataset under controlled offline conditions. This work evaluated the performance of the complete deployed application. The reported recognition accuracy therefore reflects the behaviour of the entire speech-to-sign translation pipeline, including live speech recording, audio preprocessing, speech recognition, command mapping and ASL video retrieval.

The measured response time shows that the proposed system is suitable for real-time communication. The trained model is loaded into memory only once during application startup and the ASL translation is performed using a lightweight lookup table. The computational overhead remains low/ It allows users to receive the translated sign shortly after speaking. The proposed system is the use of real WLASL sign videos that make it realistic than animated systems. Displaying human-performed signs provides a more natural visual representation of American Sign Language. It avoids the additional computational cost required for avatar rendering.

The system performs well for the supported commands. But It is currently limited to the 35 speech commands included in the Google Speech Commands dataset. Commands outside this vocabulary cannot be recognized or translated. In addition, recognition performance may be affected by excessive background noise or recording conditions that differ significantly from those used during model training. Expanding the vocabulary and supporting continuous speech translation remain important directions for future work.

8. Threats to Validity

Several factors may influence the generalizability of the reported results. The evaluation was performed under normal indoor conditions using standard microphones. Variations in microphone quality, speaker distance, pronunciation, speaking speed and background noise may influence recognition accuracy. Although the system automatically validates the recorded speech using RMS energy, extracts the highest energy speech segment and normalizes the audio before prediction, different recording environments may still affect performance.

The system was evaluated using five limited participants with different accents and pronunciation styles. A larger and more diverse group of participants would provide stronger evidence of its performance across wider populations. The evaluation was limited to the 35 predefined speech commands available in the Google Speech Commands dataset. Therefore, the reported results cannot be directly generalized to larger vocabularies or continuous speech.

The evaluation focused on real-time recognition accuracy and response time. Other evaluations using noisy environments, different recording devices and speakers from different regions would provide a more comprehensive understanding of system robustness. The system supports only word level translation and cannot translate complete sentences. Future work will focus on expanding vocabulary, supporting continuous speech.

9. Conclusions

This paper presented a real time speech-to-American Sign Language (ASL) translation system. It was developed using Django, TensorFlow and the WLASL dataset. The system recognized spoken commands from a browser microphone using a pre-trained CResD-Net speech recognition model. It further retrieves the corresponding ASL video from the WLASL dataset. The system provides fast response within 3.93 seconds including 3 seconds of recording time. The browser-based implementation also makes the system easy to access. The evaluation focused on real-time recognition accuracy, translation accuracy and response time.

In future work, the system will be extended to support a larger vocabulary. Evaluation on continuous speech translation will be performed. Evaluation will be performed with a greater number of users.

References

1. N. Laaidi, M. Telmem, M. Lamrini and H. Satori, "Arabic speech command recognition using an enhanced CNN-LSTM model with attention and data augmentation," vol. 29, no. 1, 2026.
2. T. A. Sardjono and D. Purwanto, "A performance comparison of 1D,tang 2D and 3D CNN architectures for robot voice command classification," Engineering, Technology & Applied Science Research, vol. 16, no. 1, pp. 32110-32119, 2026.
3. A. Mnuarbek, A. Bekarystankyzy, M. Turdalyuly, D. Oralbekova and A. Dyussemkhanov, "Speech-to-sign gesture translation for Kazakh: Dataset and sign gesture translation system," Computers, vol. 15, no. 3, Art. no. 188, 2026, doi: 10.3390/computers15030188.
4. M. Bittner, D. Schnöll, M. Wess and A. Jantsch, "Efficient and interpretable raw audio classification with diagonal state space models," Machine Learning, vol. 114, no. 8, 2025.
5. R. Liu, W. Chen and X. Zhang, "Dynamic convolution models for cross-frontend keyword spotting," Scientific Reports, vol. 15, no. 1, Art. no. 16745, 2025.
6. C. Sun, B. Chen, F. Chen, Y. Leng and Q. Guo, "Speech keyword spotting method based on SWIN-Transformer model," International Journal of Computational Intelligence Systems, vol. 17, no. 1, Art. no. 61, 2024.
7. P. Gambhir, A. Dev, P. Bansal and D. K. Sharma, "End-to-end multi-modal low-resourced speech keywords recognition using sequential Conv2D nets," ACM Transactions on Asian and Low-Resource Language Information Processing, vol. 23, no. 1, pp. 1-21, 2024.
8. M. Dhakal, A. Chhetri, A. K. Gupta, P. Lamichhane, S. Pandey and S. Shakya, "Automatic speech recognition for the Nepali language using CNN, bidirectional LSTM and ResNet," in Proc. Int. Conf. Inventive Computation Technologies (ICICT), 2022, pp. 515-521.
9. J. Oruh, S. Viriri and A. Adegun, "Long short-term memory recurrent neural network for automatic speech recognition," IEEE Access, vol. 10, pp. 30069-30079, 2022.
10. A. S. Dhanjal and W. Singh, "An automatic machine translation system for multi-lingual speech to Indian sign language," Multimedia Tools and Applications, vol. 81, no. 3, pp. 3259-3294, 2022.
11. J. Peguda, V. S. S. Santosh, Y. Vijayalata, A. Deepa and V. Mounish, "Speech to sign language translation for Indian languages," in Proc. 8th Int. Conf. Advanced Computing and Communication Systems (ICACCS), 2022, pp. 1131-1135.
12. R. Tang, J. Lee, A. Razi, I. Bicking, J. Cambre, I. Bicking, J. Kaye and J. Lin, "Howl: A deployed, open-source wake word detection system," in Proc. 2nd Workshop for NLP Open Source Software (NLP-OSS), 2020, pp. 61-65.
13. D. S. Jayalakshmi, H. Salpekar, H. K. Kiran Raj, R. Rahul and Shobha, "Augmenting Kannada educational video with Indian Sign Language captions using synthetic animation," in Proc. World Conf. Smart Trends in Systems, Security and Sustainability (WS4), 2020.
14. B. D. Patel, H. B. Patel, M. A. Khanvilkar, N. R. Patel and T. Akilan, "ES2ISL: An advancement in speech to sign language translation using 3D avatar animator," in Proc. IEEE Canadian Conf. Electrical and Computer Engineering (CCECE), 2020.
15. F. Filippidou and L. Moussiades, "A benchmarking of IBM, Google and Wit automatic speech recognition systems," in Artificial Intelligence Applications and Innovations, 2020, pp. 73-82.
16. K. Georgila, A. Leuski, V. Yanov and D. Traum, "Evaluation of off-the-shelf speech recognizers across diverse dialogue domains," in Proc. 12th Int. Conf. Language Resources and Evaluation (LREC), 2020.
17. R. Shahriar, A. G. M. Zaman, T. Ahmed, S. M. Khan and H. M. Maruf, "A communication platform between Bangla and sign language," in Proc. IEEE Region 10 Humanitarian Technology Conf. (R10-HTC), 2018.
18. V. Këpuska, "Comparing speech recognition systems (Microsoft API, Google API and CMU Sphinx)," International Journal of Engineering Research and Applications, vol. 7, no. 3, pp. 20-24, 2017.
19. M. M. El-Gayyar, A. S. Ibrahim and M. E. Wahed, "Translation from Arabic speech to Arabic Sign Language based on cloud computing," Egyptian Informatics Journal, vol. 17, no. 3, pp. 353-365, 2016.
20. C. Gaida, P. L. Lange, R. Petrick, P. Proba, A. Malatawy, D. Suendermann-Oeft and D. Suendermann-Oeft, "Comparing open-source speech recognition toolkits," Technical Report, DHBW Stuttgart, Stuttgart, Germany, 2014.
21. V. López-Ludeña, R. San-Segundo, C. González-Morcillo, J. C. López and J. M. Pardo, "Increasing adaptability of a speech into sign language translation system," Expert Systems with Applications, vol. 40, no. 4, pp. 1312-1322, 2013.
22. D. Li, C. Rodriguez, X. Yu and H. Li, "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison," in Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV), 2020, pp. 1459–1469.

Author Biographies

Ms Bhagya Shree received her Master of Computer Applications (MCA) degree from Panjab University, Chandigarh, India. She is currently pursuing her PhD in the Department of Computer Science and Applications at Panjab University under the supervision of Prof. Sonal Chawla. She is also working as a Senior Software Engineer-1 at Precisely and has previously served as an Assistant Manager at AlertEnterprise. Her research interests include machine learning, real-time problem-solving, and signal processing.

Dr. Sonal Chawla is a Professor in the Department of Computer Science and Applications at Panjab University, Chandigarh, India, and also serves as the Honorary Director of the Centre for IAS and Other Competitive Examinations at the same university. She has held several key academic and administrative roles, including Chairperson of the department (2014–2017), Placement Coordinator (2009–2011, 2017–2019), Convener of the Board of Studies for multiple undergraduate and postgraduate computer science programs, and Coordinator of the PGDCA correspondence course under USOL. She is also an elected member of the Panjab University Senate (since 2021) and Principal Investigator for the RUSA-supported Skill Enhancement project. Her research interests include Artificial Intelligence, Machine Learning, Semantic Web Applications, Operating Systems, and Relational Database Management Systems. She has led research projects funded by RUSA, AICTE (3 years), and UGC (2 years), and has over 35 publications in reputed journals and conferences.