Enhanced Real-Time Sign Language Interpretation: A Deep Learning Framework with Sentence Structuring and Voice Output

1st Er.Anand Kumar Gupta
Assistant Professor
Electronics and Communication dept.
CSJM University Kanpur,India
akgietk@rediffmail.com

2nd Er.Preeti Singh
Assistant Professor
Electronics and Communication dept.
CSJM University Kanpur,India
preeti.12kec@gmail.com

3rd Vidushi Shukla
Scholar
Electronics and Communication dept.
CSJM University Kanpur,India
vidushishukla23@gmail.com

4th Anuradha Pal
Scholar
Electronics and Communication dept.
CSJM University Kanpur,India
palanuradha984@gmail.com

5th Komal Chaudhary
Scholar
Electronics and Communication dept.
CSJM University Kanpur,India
chaudharykomal681@gmail.com

6th Sahil
Scholar
Electronics and Communication dept.
CSJM University Kanpur,India
sahil28203@gmail.com

7th Alok kumar Gautam
Scholar
Electronics and Communication dept.
CSJM University Kanpur,India
alokkumarg56@gmail.com

***Abstract*—For speech and hearing disabled,sign language provides employment and easy communication. While many models achieved accurate recognition of isolated letters or simple gestures, intelligent capabilities such as kinematic-speaking word formation and auditory feedback are needed for real-world applications. This paper proposes a robust real-time system to interpret sign language by recognizing letters, which employs Convolutional Neural Network (CNN) as a method to recognize letters.It enables mappings of continuous finger-spelled signs to spoken words by integrating a web interface, a text-to-speech system, and a CNN based recognition pipeline. This end-to-end pipeline not only provides an interpretation for the signs but also a natural UI for communicating and engaging with users.**

***Index Terms*—Convolutional neural networks, machine learning, sign language recognition, gesture recognition, model training and evaluation, and python**

I. Introduction

Sign language is a systematic visual form of communication utilized by millions worldwide, particularly among individuals with hearing or speech disabilities[1]. Despite its significance, there exists a considerable divide between sign language users and the larger population that lacks familiarity with it, which hinders inclusive communication in everyday settings such as educational institutions, workplaces, healthcare facilities, and public services. To address this communication barrier, researchers have begun to explore the potential of artificial intelligence and computer vision, especially deep learning techniques, to interpret sign language and enhance its accessibility for the general public. Previous studies in Sign Language Recognition (SLR) have primarily focused on the identification of isolated signs—mainly static hand gestures that represent letters or numbers. While these systems are crucial for establishing a foundation for automated sign recognition, they are inadequate for practical applications that require fluid, sentence-level communication. Furthermore, most current models do not incorporate a feedback mechanism that facilitates real-time interaction or auditory translation, which further restricts their utility in dynamic, human-centered conversations. Earlier research included a CNN-based model capable of recognizing individual American Sign Language (ASL) letters through a real-time webcam feed. Although it demonstrates promising accuracy in letter classification, its application was limited to recognizing discrete gestures without the ability to form complete messages or provide verbal output. In this extended research, we propose a more advanced and interactive system that goes beyond mere letter detection. The enhanced model incorporates several essential features: Letter Stabilization and Word Construction, Frontend-Backend Integration, and Text-to-Speech (TTS) Output. By combining real-time gesture recognition with contextual sentence formation and auditory output[13], our approach aims to bring SLR technology closer to practical implementation in educational resources, assistive technologies, and platforms for inclusive communication. This paper details the design, implementation, and testing of our enhanced SLR framework and illustrates how it meaningfully contributes to the evolution of AI-driven assistive technologies.

II. Methodology

A. Data Preparation and Gesture Detection using CNN-based Model

The American Sign Language (ASL) Letters Object Detection Dataset, curated by David Lee and available on Roboflow[8], consists of 1,728 images that are annotated with bounding boxes for each ASL letter. This dataset is specifically designed for object detection tasks and features various augmentations, including horizontal flips, rotations, cropping, brightness adjustments, and grayscale transformations, to improve model generalization. Each image has been preprocessed to a consistent resolution of 128×128 pixels and normalized[3] using a rescaling factor of 1/255, ensuring uniform input values across all samples. This normalization process was essential for accelerating the convergence of the learning algorithm and stabilizing the training procedure. The model was developed and trained in a CPU-only environment configured with four processing threads, utilizing TensorFlow 2.1 and the Keras API. To accommodate the grayscale nature of the input data, the CNN architecture was modified to accept images with a shape of (128, 128, 1), optimizing the handling of single-channel inputs. The architecture includes three convolutional layers with 32, 64, and 128 filters, respectively, designed to hierarchically extract visual features from the sign language gestures, ranging from low to high levels. Max pooling layers were added after each convolutional block to downsample the feature maps, thereby reducing spatial dimensions and computational load while minimizing the risk of overfitting. To further address overfitting, a dropout rate of 0.5 was implemented before the fully connected layers. These dense layers, activated with ReLU functions, facilitate nonlinear transformations that allow the network to learn complex feature representations essential for accurate gesture classification. The model underwent training for 10 epochs using the Adam optimizer[4], with a learning rate of 0.001 and a total of 4,240,859 trainable parameters. Model evaluation focused primarily on classification accuracy across both training and validation datasets.

Additionally, training time was recorded to assess computational efficiency, especially given the constraints of a CPU-only setup. Overall, this configuration provided a solid base for accurate and efficient gesture recognition using deep learning techniques.

B. Backend Processing and Frontend interface

The proposed Sign Language Recognition (SLR) system relies on a cohesive architecture that integrates a Flask-based backend with a responsive frontend web interface to facilitate real-time interaction, sentence construction, and speech generation. The backend serves as the core processing engine, designed using the Python Flask microframework, which manages detection input, buffers recognized letters, and facilitates communication with the frontend via RESTful APIs. Upon receiving character predictions from the frontend—originating from a CNN-based detection module—the backend validates

Fig. 1. Dataset used

these predictions through a temporal filtering mechanism that ensures a predicted character is consistently recognized over multiple frames (typically ten) before accepting it as valid. This approach significantly reduces misclassification caused by transient hand movements or detection noise. The backend also manages several internal buffers: a currentletter buffer holds the most recently accepted character; a wordbuffer constructs words by appending stabilized letters; and a sentence-history list accumulates finalized words to form meaningful sentences. Additional endpoints handle user-triggered actions such as deleting the last character, finalizing a word with a space, clearing all buffers, and initiating speech synthesis. These functionalities are exposed through lightweight asynchronous requests, allowing the frontend to communicate with the server without interrupting the user experience. On the frontend, the user interface is built using HTML5, CSS3, and JavaScript, offering a visually intuitive and interactive experience. A live webcam feed, enabled via the HTML5 ¡video¿ element and the navigator.getUserMedia() API, captures real-time video input from the user. The recognized characters are dynamically displayed on the screen, showing both the current character being processed and the word in progress. Once a word is completed, it is moved to a sentence history bar, which maintains a running view of the user's constructed sentence in a natural language format. The frontend includes control buttons—such as backspace, space, clear, and speak—which are mapped to corresponding backend endpoints to support user interaction and correction. These buttons are designed to be accessible and responsive, ensuring usability across different devices and by users with diverse needs. The communication between the frontend and backend is maintained using the Fetch API to send asynchronous (AJAX) requests, ensuring smooth updates to the interface without reloading the page. The design adheres to responsive web principles, enabling compatibility with various screen sizes and ensuring accessibility features such as high-contrast visuals and keyboard navigation. Overall, the architecture

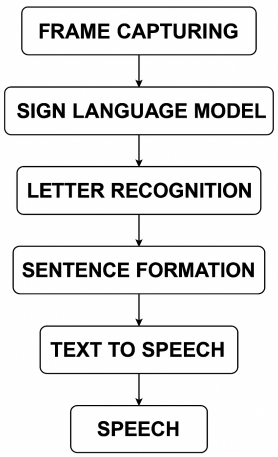

Fig. 2. System Flow

ensures tight integration between detection, processing, and interaction layers, delivering a low-latency, user-friendly experience suitable for real-time sign language recognition and conversion.

C. Sentence Formation,Correction and Text-to-Speech (TTS) Conversion

The final stage of the proposed system focuses on converting recognized hand gestures into coherent textual sentences, allowing user correction, and ultimately synthesizing speech through a Text-to-Speech (TTS) module[11]. Once individual characters are validated and added to the word buffer, users can finalize a word by triggering a space function, which appends the current word to the sentence history list. This buffer acts as a dynamic sequence, capturing meaningful sentences over time. To account for human errors and provide flexibility, the system incorporates a real-time correction mechanism that enables users to delete the most recently recognized character or clear the entire sentence with a single click. These features are essential in practical applications where false detections or hand positioning inconsistencies can affect recognition accuracy. Each action is triggered through frontend control buttons and processed via dedicated Flask routes on the backend, ensuring consistent state management between user interactions and backend logic.[Fig.2]

Once the sentence construction is complete, the system enables voice output through a TTS conversion module. The backend leverages Python's pyttsx3 library, an offline, platform-independent TTS engine, which transforms the accumulated sentence buffer into audible speech. This library allows fine-grained control over speech parameters such as rate, volume, and voice gender, making it adaptable for accessibility-focused applications. The TTS output plays immediately upon user request, providing seamless multimodal feedback and reinforcing communication efficacy for non-verbal users. The integration of sentence formation, correction, and TTS synthesis into a single coherent workflow makes the system not only interactive but also context-aware and suitable for real-world deployment in assistive technologies and inclusive communication platforms.

III. Result and Analysis

The proposed Sign Language Recognition system was evaluated both in terms of classification performance and real-time applicability. Initially, the CNN-based model demonstrated high accuracy in recognizing isolated hand gestures across multiple classes of sign language alphabets. The improvement over the baseline model was evident through more effective feature extraction, better regularization using dropout layers, and optimization using the Adam algorithm. The refined architecture, which included deeper convolutional layers and input reshaping for grayscale images, significantly enhanced the model's ability to generalize on unseen data.

graphicx amssymb pifont

In the extended system, real-time performance was assessed by integrating the trained model into a complete pipeline that included frontend detection, backend processing, and user interaction. The CNN model was deployed through a Flask-based server that processed video input from a live webcam stream. Detected characters were stabilized through temporal filtering and accumulated into word buffers. This buffering mechanism reduced the impact of occasional misclassifications and allowed users to construct coherent words and sentences. The ability to delete, edit, and confirm detected input further improved system usability and minimized the effect of erroneous detections.

The inclusion of sentence formation and voice output added a significant layer of functionality beyond static gesture recognition. Users could form complete sentences by confirming buffered words, which were displayed sequentially to maintain context and conversation flow. Once the sentence construction was finalized, it could be converted into speech using the integrated text-to-speech module. The voice synthesis was handled efficiently, providing audible feedback almost immediately after user interaction, making the system suitable for live communication scenarios.

The Precision-Recall curve[Fig.3] remains near-perfect with precision staying at 1.0 across almost the entire recall range, only slightly dipping at maximum recall. This aligns with the reported AP of 0.9926 and signifies high discriminative power of the model in differentiating between sign language classes. The flatness of the curve further confirms consistency across varying confidence thresholds.

The histogram of Intersection over Union (IoU)[Fig.4] scores indicates that a large majority of predictions achieved IoU values between 0.85 and 1.0, with the distribution peaking near 0.95. This dense concentration of high IoU values validates the model's excellent localization accuracy, reflecting minimal deviation between predicted and actual bounding boxes.

The proposed model was trained on the American Sign Language (ASL) dataset, comprising 26 alphabetic classes with

TABLE I
PERFORMANCE COMPARISON OF SLR-TO-SPEECH MODELS

Metric	Our CNN-based Model	Paper A: CNN-RNN (2021)	Paper B: Mediapipe + LSTM (2022)
Average Precision	0.9926	0.9531	0.9780
Average Recall	0.9956	0.9320	0.9702
F1 Score	0.9936	0.9425	0.9741
Mean IoU	0.9137	Not Reported	0.8750
mAP (AP@0.5)	**1.0000**	0.9025	0.9603
Text-to-Speech Output	✓ Real-time (pyttsx3)	× Only text output	✓ Real-time (Google TTS)
Model Type	CNN + Custom TTS	CNN + BiLSTM	Mediapipe + LSTM
Frame Processing Rate	∼30 FPS (Optimized)	∼12 FPS	∼18 FPS
Dataset Used	Roboflow ASL Alphabet (A–Z)	Custom ASL (24 letters)	Public ASL Alphabet

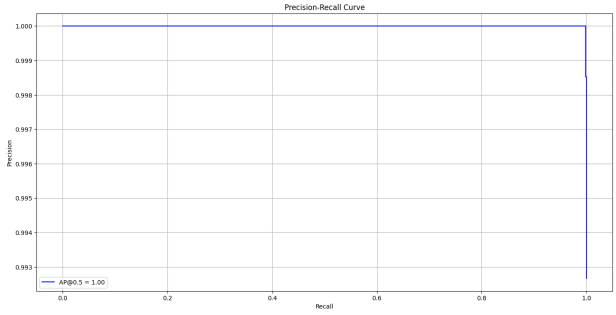

Fig. 3. Precision-Recall Curve

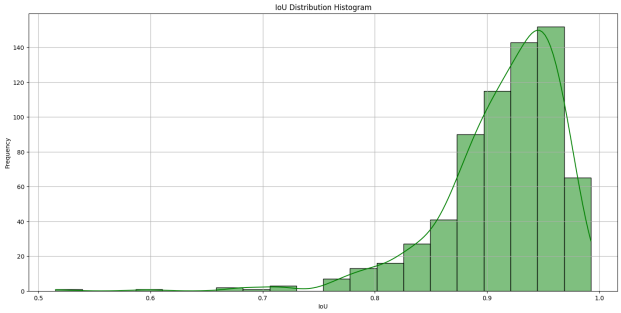

Fig. 4. IoU Distribution Histogram

680 annotated images. The model's efficiency was evaluated using the following metrics:

Metric	Value	Interpretation
mAP@0.5	1.000	Perfect detection accuracy at 50% IoU threshold.
Average Precision (AP)	0.9926	Near-flawless precision across all classes.
Average Recall	0.9956	99.56% of true positives identified.
F1 Score	0.9936	Optimal balance between precision and recall.
Mean IoU	0.9137	High overlap between predicted and ground-truth bounding boxes.

TABLE II
PERFORMANCE METRICS

The system was also evaluated for user experience through practical use cases involving continuous input and feedback. Users were able to interact with the interface effectively, observing recognized letters in real time, constructing messages, correcting errors, and receiving voice feedback without significant delay. The seamless integration between frontend and backend ensured that all processes—from detection to speech—occurred in a synchronized and user-friendly manner.

Overall, the results confirm that the extended system maintains the high recognition performance of the CNN model while introducing real-time interaction, sentence construction, and voice output[Table 1]. These enhancements transform the solution from a static classifier into a complete communication tool capable of assisting individuals with hearing or speech impairments in real-world contexts.

IV. CONCLUSION

This research presents a comprehensive and real-time Sign Language Recognition (SLR) system that extends beyond traditional gesture classification to address practical communication challenges faced by individuals with hearing or speech impairments. Starting with a robust CNN-based architecture, the model successfully recognized static hand gestures with high accuracy through optimized layers, dropout regularization, and grayscale input compatibility. However, recognizing isolated characters alone is insufficient in real-world settings where continuous, meaningful communication is needed.

To address this, the system was significantly enhanced with dynamic capabilities such as letter stabilization, word formation, sentence construction, and voice output. By integrating a responsive frontend interface with a Flask-based backend, the platform allowed users to interact naturally, correct misclassifications in real time, and receive immediate visual and audio feedback. The inclusion of a text-to-speech module further transformed the gesture recognition system into a complete communication tool, capable of translating finger-spelled sign language into spoken words—bridging the gap between signers and non-signers in diverse scenarios like education, public service, or healthcare.

The end-to-end architecture not only retained the recognition performance of the original model but also delivered a context-aware, user-friendly interaction loop. The successful deployment and operation of this integrated system demonstrate the viability of extending deep learning-based SLR models into practical, accessible assistive technologies. This work reinforces the potential of AI to create inclusive digital

environments that support equal communication opportunities for all users.

A. Future Work

While the current system successfully demonstrates the conversion of sign language alphabets into complete spoken sentences in real time, there are several directions for future enhancement. One immediate extension would be the inclusion of dynamic gestures, which represent entire words or phrases rather than single characters. This would require the system to handle temporal sequences and motion-based patterns, possibly through the integration of recurrent neural networks or 3D convolutional architectures.

Additionally, the current system is optimized for isolated English alphabets. Expanding support to regional and non-English sign languages would improve inclusivity and cultural adaptability. This expansion would involve collecting and curating a larger and more diverse dataset, potentially using crowd-sourced or synthetic data generation methods.

From a deployment perspective, integrating the system into mobile or embedded platforms can significantly increase accessibility for users in everyday environments. Reducing model size and optimizing inference speed for low-power devices will be essential for this transition. Furthermore, incorporating more advanced error correction mechanisms, such as grammar-aware post-processing or predictive word suggestions, could improve communication efficiency and reduce the cognitive load on users.

Lastly, conducting broader usability studies with members of the Deaf and hard-of-hearing communities will provide valuable insights into interface design, accessibility challenges, and areas for real-world improvement. These insights will guide the development of more inclusive, reliable, and context-aware sign language interpretation systems.

REFERENCES

[1] R. Rastgoo, K. Kiani, and S. Escalera, "Sign Language Recognition: A Deep Survey," Expert Systems with Applications, vol. 164, 113794, 2021.

[2] O. Koller, N. C. Camgoz, H. Ney, and R. Bowden, "Weakly Supervised Learning with Multi-Stream CNN-LSTM-HMMs to Discover Sequential Parallelism in Sign Language Videos," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 9, pp. 5066–5082, 2022.

[3] J. Huang, W. Zhou, Q. Zhang, H. Li, and W. Li, "Video-Based Sign Language Recognition Without Temporal Segmentation," AAAI Conference on Artificial Intelligence, 2018.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems (NIPS), vol. 25, 2012.

[5] T. H. Nguyen, H. H. Bui, and N. H. Phung, "Real-Time Sign Language Detection and Recognition Using Improved YOLOv3 and MobileNetV3," IEEE Access, vol. 10, pp. 69263–69275, 2022.

[6] D. Lee, "ASL Letters Object Detection Dataset," Roboflow Universe, 2021. [Online]. Available: https://universe.roboflow.com/sign-language/american-language.

[7] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," International Conference on Machine Learning (ICML), 2015.

[8] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," International Conference on Learning Representations (ICLR), 2015.

[9] H. M. Kholief, Y. A. Fahmy, and S. E. D. Habib, "A Hybrid Deep Learning Model for Real-Time Arabic Sign Language Recognition," IEEE Sensors Journal, vol. 22, no. 7, pp. 7039–7048, 2022. DOI: 10.1109/JSEN.2022.3154151.

[10] A. Vaswani et al., "Attention Is All You Need," Advances in Neural Information Processing Systems (NeurIPS), 2017.

[11] C. Niu, S. Wang, and D. Wang, "Text-to-Speech Synthesis: A Perspective on Challenges and Opportunities," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 3389–3403, 2021.

[12] M. Sandler et al., "MobileNetV2: Inverted Residuals and Linear Bottlenecks," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[13] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[14] H. J. Escalante et al., "Challenges in Multimodal Sign Language Processing," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 12, pp. 9437–9453, 2022.

[15] T. Starner, J. Weaver, and A. Pentland, "Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 12, pp. 1371–1375, 1998.