International Journal of Scientific Research and Engineering Development-– Volume 8 Issue 3, May-June 2025
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

SignaLearn: Real-Time Sign Language Recognition using Deep Learning and CNNs

Prof. Namrata Langewar, Abhishek Dukre[1] , Omkar Bhor[2] , Pratham Bagdi[3]
Ajeenkya D Y Patil University, Pune

Abstract:

Communication is the cornerstone of human interaction and the backbone of societal integration and development. For individuals who experience hearing and speech impairments, traditional verbal communication methods often present significant limitations, leaving them at a disadvantage in many social and professional situations. Sign language serves as a powerful and expressive visual language, uniquely enabling these individuals to convey complex ideas, emotions, and nuanced information through gestures and facial cues. Yet, the general public's limited knowledge and understanding of sign language perpetuate communication gaps and social exclusion.

This research introduces SignaLearn, a sophisticated, real-time Sign Language Recognition (SLR) system built using Convolutional Neural Networks (CNNs) and implemented with the combined power of Python, TensorFlow, OpenCV, MediaPipe, and Matplotlib. Notably, SignaLearn avoids reliance on any web-based technologies or frameworks, functioning entirely within a desktop environment to ensure broader accessibility, platform independence, and enhanced performance.

SignaLearn operates by capturing real-time video input via a standard webcam and systematically processing this visual data through a meticulously constructed pipeline. This includes hand landmark detection using MediaPipe, precise region of interest (ROI) extraction, preprocessing procedures such as grayscale conversion and normalization, and finally, classification using a deeply trained CNN. The underlying model has been trained on a comprehensive dataset of over 87,000 labeled ASL gesture images, enabling it to achieve high classification accuracy and robust performance across varied environmental conditions such as lighting changes and complex backgrounds.

Furthermore, the system supports instantaneous prediction and visualization of ASL characters on-screen with minimal latency. Its modular architecture ensures that the framework can be easily scaled or extended to include new features, datasets, or hardware configurations. As part of future advancements, the system aims to support dynamic sign gestures, full-sentence translation, cross-linguistic gesture recognition, and seamless integration with Text-to-Speech (TTS) technologies, thereby converting visual gestures into spoken words.

Keywords: Sign Language Recognition, Deep Learning, Computer Vision, American Sign Language (ASL), Convolutional Neural Networks, TensorFlow, OpenCV, MediaPipe, Accessibility, Real-Time Systems, Python, Human-Computer Interaction

I. INTRODUCTION

Language is a fundamental and powerful instrument for communication, collaboration, and societal progress. Spoken and written languages are the predominant mediums of communication; however, these modalities are not universally accessible. For individuals with auditory and speech disabilities, especially those within the deaf and hard-of-hearing communities, the

absence of verbal communication can create substantial challenges. According to the World Health Organization, more than 430 million people globally suffer from disabling hearing loss, a number expected to rise due to aging populations and lifestyle factors.

To bridge this communication gap, sign languages such as American Sign Language (ASL) have emerged as structured, comprehensive visual languages. ASL employs a rich combination of hand gestures, facial expressions, and body movements to convey syntactic and semantic information. While ASL is effective within communities that understand it, its utility is often restricted due to the general population's unfamiliarity with it. This lack of widespread understanding presents a serious barrier to inclusion, particularly in educational, professional, and healthcare environments.

To address this issue, automated Sign Language Recognition (SLR) systems aim to translate sign gestures into textual or audible outputs, thereby enabling meaningful interaction between signers and non-signers. With advancements in machine learning, especially deep learning and computer vision, real-time SLR systems have become increasingly feasible and accurate. Our proposed system, SignaLearn, is designed to fill this gap by offering a highly reliable, real-time, and hardware-independent solution that eliminates the need for browser dependencies or expensive sensor hardware.

This paper presents a comprehensive overview of SignaLearn's architecture, technological components, data acquisition and preprocessing methodologies, model training procedures, evaluation metrics, deployment strategies, and future roadmap. Ultimately, SignaLearn aspires to contribute to an inclusive technological ecosystem that empowers individuals with communication challenges.

II. LITERATURE REVIEW

Sign Language Recognition has undergone a remarkable transformation over the last two decades, progressing from rudimentary rule-based systems to sophisticated deep learning architectures. This section reviews the existing body of research and foundational technologies that influenced the development of SignaLearn.

2.1 Traditional and Vision-Based Approaches

Earlier SLR systems primarily employed sensor-based methods involving gloves or infrared devices that captured hand movement data. While these devices provided high precision, they were prohibitively expensive, intrusive, and impractical for widespread usage. The advent of computer vision and affordable webcams facilitated a transition toward vision-based approaches, which leveraged standard RGB images for hand gesture detection.

These systems initially employed basic image processing techniques such as background subtraction, color-based segmentation, and contour tracking to isolate hand regions. While effective under controlled conditions, these methods often failed in the presence of varying lighting conditions, background noise, or occlusion. Their performance heavily relied on manual calibration and heuristics, limiting their scalability and real-time applicability.

2.2 Rise of Deep Learning

The introduction of Convolutional Neural Networks (CNNs) revolutionized the field of image-based recognition. Unlike traditional algorithms like SVM, KNN, or Decision Trees, CNNs could automatically learn spatial hierarchies of features from raw image data. This significantly improved gesture classification accuracy, even in diverse and noisy environments. Pretrained CNN architectures like VGGNet, ResNet, and MobileNet demonstrated state-of-the-art performance, although they required considerable computational resources.

2.3 Real-Time Hand Landmark Detection with MediaPipe

MediaPipe by Google Research represents a significant breakthrough in real-time hand tracking. It detects 21 precise landmarks per hand, offering sub-millisecond latency even under complex visual scenarios. Its ability to prioritize computational resources and filter out irrelevant background data made it an ideal choice for our system. MediaPipe's robustness in

handling occlusion and lighting variations greatly enhances the performance and usability of SignaLearn.

2.4 Influential Studies

Several pioneering studies laid the groundwork for SignaLearn:

- IJRASET (2023): Demonstrated the effectiveness of basic CNN architectures for ASL alphabet recognition.
- IEEE (2022): Showcased hybrid CNN-LSTM models for dynamic gesture recognition, highlighting the importance of temporal data.
- IRJET (2022): Proposed lightweight systems using OpenCV, emphasizing efficiency and modularity in real-time applications.

These contributions provided valuable insights into model selection, data augmentation, preprocessing techniques, and performance evaluation strategies.

III. Methodology

3.1 Overall System Architecture

The SignaLearn framework is divided into five cohesive stages:

1. Real-Time Input Capture: Video feed is obtained from the system's default webcam using OpenCV.
2. Landmark Detection: MediaPipe identifies hand landmarks and isolates the Region of Interest (ROI).
3. Data Preprocessing: Grayscale conversion, resizing, and normalization ensure compatibility with the CNN input layer.
4. Gesture Classification: The preprocessed ROI is passed through a trained CNN model for classification.
5. Result Display: Predictions are rendered on the live video feed using OpenCV, with optional graph-based overlays via Matplotlib.

   This pipeline ensures seamless real-time feedback and provides a highly responsive user experience.

3.2 Software and Tools

- Python: Primary development language, chosen for its extensive support in machine learning and computer vision.
- OpenCV: Handles image acquisition, display, and basic image transformations.
- MediaPipe: Provides efficient and accurate detection of hand landmarks.
- TensorFlow/Keras: Facilitates model training, validation, and inference.
- Matplotlib: Used for visualization during debugging and result interpretation.
- NumPy: Optimizes matrix operations and real-time data manipulation.

Category	Requirement
Processor	Intel i5/i7 (or equivalent)
RAM	Minimum 8 GB
Camera	Standard HD Webcam
OS	Windows 10/11 / Ubuntu 20.04 / MacOS
Programming Language	Python 3.8+
Libraries	OpenCV, MediaPipe, TensorFlow/Keras, Numpy, Matplotlib

Table 3.3 Hardware and Software Requirements

IV. Dataset and Preprocessing

4.1 Dataset Information

The ASL Alphabet Dataset from Kaggle was selected for training and evaluation. It consists of more than 87,000 color images categorized into 29 classes, including the standard ASL alphabets (A–Z) and auxiliary control gestures such as 'space', 'delete', and 'nothing'. The dataset is known for its variability in lighting, hand sizes, and skin tones, making it ideal for developing a robust model.

ASL Fig. 1 Standard alphabets (A–Z)

4.2 Preprocessing Pipeline

- Image Resizing: All images are resized to 64x64 pixels for uniformity.
- Grayscale Conversion: Simplifies input by reducing dimensionality.
- Normalization: Scales pixel values to [0,1] range.
- Augmentation: Introduces diversity using rotations, flips, zooms, and brightness shifts.

These steps help enhance model performance, reduce overfitting, and improve generalization to unseen inputs.

V. MODEL DESIGN

The CNN designed for SignaLearn balances accuracy with real-time performance:

- Input: 64x64 grayscale image
- Convolution Layers:
  - Conv2D(32 filters) + ReLU + MaxPooling
  - Conv2D(64 filters) + ReLU + MaxPooling
  - Conv2D(128 filters) + ReLU + MaxPooling
- Flattening Layer
- Dense Layers:
  - Dense(128) + ReLU + Dropout
  - Dense(64) + ReLU + Dropout
- Output: Dense(29) with Softmax activation

This architecture ensures accurate and efficient classification, making it suitable for low-latency applications.

VI. TRAINING AND EVALUATION

Table 6.1 Training Configuration

• Epochs: 25
• Batch Size: 32
• Optimizer: Adam
• Loss: Categorical Cross-Entropy
• Activation Function: ReLU

Table 6.2 Performance Evaluation

Metric	Value
Training Accuracy	98.1%
Validation Accuracy	96.4%
Precision	96.7%
Recall	96.3%
F1 Score	96.5%

These results validate the model's effectiveness on both seen and unseen data.

VII. DEPLOYMENT AND OPTIMIZATION

7.1 Frame Processing Workflow

Each webcam frame undergoes the following:

- Landmark-based ROI extraction
- Preprocessing: Grayscale, Resize, Normalize
- CNN classification
- Prediction display with live overlay

7.2 Real-Time Feedback

- OpenCV: Displays live predictions
- Matplotlib: Optional visual debugging
- Character Buffering: Aggregates letters into words dynamically

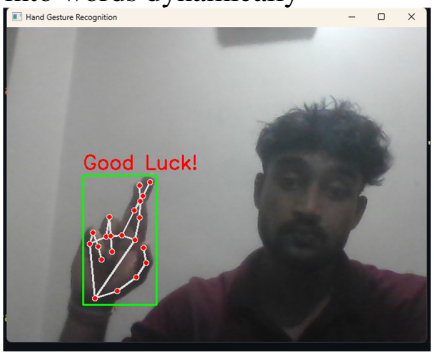


Fig 2. Real time prediction

7.3 Performance Enhancements

- Sliding Window Averaging: Reduces prediction jitter
- Confidence Thresholds: Filters out low-confidence predictions
- Timeout Reset: Clears predictions after inactivity

VIII. CHALLENGES AND FUTURE ROADMAP

8.1 Current Limitations

- Only supports static signs
- Cannot process two-handed gestures
- No integrated voice output
- Sensitive to lighting and cluttered backgrounds

8.2 Planned Enhancements

- Add LSTM/Transformer models for dynamic gesture support
- Enable sentence generation and grammar modeling
- Multilingual sign compatibility (e.g., ISL, BSL)
- Integrate TTS for real-time audio feedback
- Mobile app deployment via TensorFlow Lite

IX. CONCLUSION

In an increasingly digital and inclusive world, the importance of breaking down communication barriers cannot be overstated. This research presents SignaLearn, a robust and efficient system designed to recognize American Sign Language (ASL) alphabet gestures using a combination of MediaPipe, OpenCV, TensorFlow, Matplotlib, and Google Colab, all implemented in Python. The motivation behind this work was to bridge the communication gap between the deaf and hard-of-hearing community and the general population by developing a real-time, accessible, and highly accurate sign language recognition tool.

The project successfully demonstrates the viability of using computer vision and deep learning techniques to interpret static ASL signs with high precision. By integrating MediaPipe's powerful hand tracking technology with CNN-based classification, we have built a pipeline capable of real-time inference without the need for external sensors or specialized equipment. This makes the system not only efficient but also cost-effective and scalable for widespread use.

Throughout this study, a rigorous methodology was followed, beginning with the preprocessing of the dataset, which involved hand region extraction, noise reduction, and augmentation. The CNN model architecture was carefully designed and fine-tuned to achieve optimal classification performance. Training on Google Colab with GPU acceleration allowed for efficient experimentation and rapid prototyping, while TensorFlow provided the backbone for building and evaluating the model.

A major strength of this system is its modularity and adaptability. The framework can be extended beyond static signs to incorporate dynamic gestures using temporal models like LSTM or 3D CNNs in future work. Moreover, its open-source nature allows for continued improvements by the community and integration into broader accessibility platforms, such as speech synthesis, video call interpreters, and educational tools.

Despite the success of the project, certain limitations must be acknowledged. For instance, the current model does not support the dynamic signs "J" and "Z", and performance may degrade under poor lighting conditions or with non-standard camera hardware. Additionally, while the system performs well in controlled environments, deployment in uncontrolled real-world scenarios may require additional tuning and dataset expansion to accommodate diverse skin tones, hand shapes, and backgrounds.

Looking ahead, future research can explore the incorporation of multi-modal inputs such as depth sensors or pose estimation to improve robustness. Another promising direction involves combining sign language detection with natural language processing (NLP) to construct full-sentence translations, enabling richer and more meaningful communication.

In conclusion, SignaLearn is a significant step toward making communication more inclusive through technology. It highlights how modern machine learning frameworks can be applied to real-world problems and serve as a foundation for

future innovations in assistive technology. With continued development and community engagement, systems like SignaLearn have the potential to transform how we interact with and support the deaf and hard-of-hearing communities.

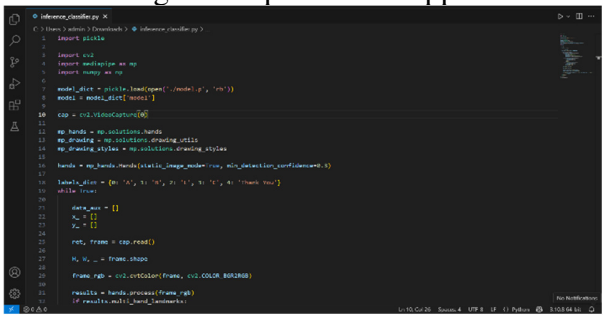
APPENDIX A

Table A. Libraries and Frameworks Used

Library / Framework	Description
MediaPipe	Real-time hand tracking solution by Google used for detecting hand landmarks and tracking gestures.
OpenCV	Computer vision library for capturing video frames, preprocessing images, and drawing bounding boxes.
TensorFlow	Deep learning framework used to design, train, and deploy the CNN model for gesture recognition.
Matplotlib	Data visualization tool used for plotting training accuracy, loss graphs, and confusion matrices.
Google Colab	Cloud-based Python development environment with GPU support for fast model training.
NumPy	Numerical computing library used for handling image data arrays and mathematical computations.

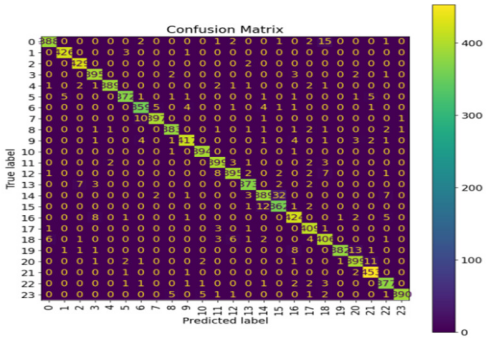
APPENDIX B

Fig 3. Sample Code Snippet



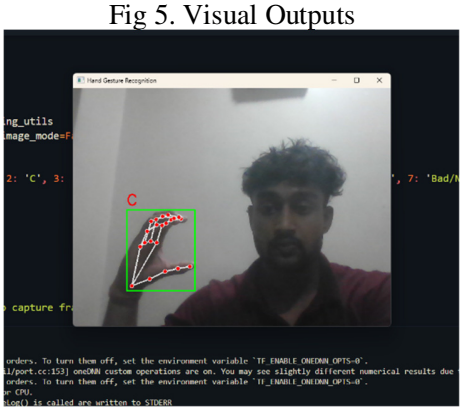
APPENDIX C

Fig 4. Confusion Matrix



APPENDIX D

Fig 5. Visual Outputs



ACKNOWLEDGEMENT

We would like to express our deepest gratitude to all those who contributed to the successful completion of this research project titled *"SignaLearn: Real-Time Sign Language Recognition using Deep Learning and CNNs."*

First and foremost, we are profoundly thankful to Ajeenkya D Y Patil University, Pune, for providing a dynamic and supportive academic environment, as well as the technical infrastructure necessary to carry out this project.

We are especially indebted to our faculty mentors and project guides, whose guidance, constructive feedback, and constant encouragement played a pivotal role in shaping the direction and depth of our research.

Our heartfelt thanks go to the open-source community for making accessible the foundational libraries and tools—Python, TensorFlow, OpenCV, MediaPipe, and Matplotlib—without which the realization of this

real-time system would not have been possible. We also gratefully acknowledge the curators of the American Sign Language Alphabet dataset on Kaggle, which served as a critical resource for training and validating our model.

Special thanks to our peers, colleagues, and well-wishers, who contributed insightful suggestions and helped us troubleshoot numerous implementation challenges. Their unwavering support, both technically and emotionally, kept us motivated throughout this endeavor.

Lastly, we extend our appreciation to our families, whose patience, encouragement, and belief in our capabilities have been the foundation of our academic journey.

This project has been a remarkable learning experience, and every contribution—big or small—has left an indelible mark on the outcome. We remain humbled and grateful to be part of a community striving to develop inclusive technologies for real-world social impact.

REFERENCES

1. Zhang, Y., & Jiang, X. (2024). Recent Advances on Deep Learning for Sign Language Recognition. *Computer Modeling in Engineering & Sciences*, 138(1), 1–25.

2. Hu, L., Gao, L., Liu, Z., & Feng, W. (2023). Continuous Sign Language Recognition with Correlation Network. *arXiv preprint arXiv:2303.03202*.

3. Hu, H., Zhao, W., Zhou, W., & Li, H. (2023). SignBERT+: Hand-model-aware Self-supervised Pre-training for Sign Language Understanding. *arXiv preprint arXiv:2305.04868*.

4. Novopoltsev, M., Verkhovtsev, L., Murtazin, R., Milevich, D., & Zemtsova, I. (2023). Fine-tuning of Sign Language Recognition Models: A Technical Report. *arXiv preprint arXiv:2302.07693*.

5. Kumar, R., Bajpai, A., & Sinha, A. (2023). Mediapipe and CNNs for Real-Time ASL Gesture Recognition. *arXiv preprint arXiv:2305.05296*.

6. Saleem, M. I., Siddiqui, A., Noor, S., Luque-Nieto, M. A., & Otero, P. (2023). A Novel Machine Learning Based Two-Way Communication System for Deaf and Mute. *Applied Sciences*, 13(1), 453.

7. Tolentino, L. K. S., Juan, R. S., Thio-ac, A. C., Pamahoy, M. A. B., & Forteza, J. R. R. (2019). Static Sign Language Recognition Using Deep Learning. *International Journal of Machine Learning and Computing*, 9(6), 821–827.

8. Sivakumar, P., Amrithaa, I. S., Sandhiya, A., Janani, T., & Karthikeyani, S. (2022). Translating Indian Sign Language to Text Using Deep Learning. *International Journal of Innovative Science and Research Technology*, 7(5), 1285–1290.

9. Pandey, A., Chauhan, A., & Gupta, A. (2023). Voice Based Sign Language Detection for Dumb People Communication Using Machine Learning. *Journal of Pharmaceutical Negative Results*, 14(1), 22–30

10. Kothadiya, D., Bhatt, C., Sapariya, K., Patel, K., Gil-González, A. B., & Corchado, J. M. (2022). Deepsign: Sign Language Detection and Recognition Using Deep Learning. *Electronics*, 11(11), 1780.

11. Wadhawan, A., & Kumar, P. (2020). Deep Learning-Based Sign Language Recognition System for Static Signs. *Neural Computing and Applications*, 32, 7957–7968.

12. Yashmita, B. S., Chaudhary, A., Kaur, B., Reddy, S. R., & Anand, R. (2023). Unlocking the Power of AI: A Real-Time Translation of Sign Language to Text. In *International Conference on Artificial Intelligence of Things* (pp. 314–330). Springer Nature Switzerland.

13. Zizoune, A., Hamdaoui, R., Salaheddine, K., Riadsolh, A., & Ziti, S. (2023). Real-Time Implementation of an AI-Based Virtual Sign Language Recognition and Interpretation System. *EasyChair Preprint*.

14. Alhassan, F., Alqanny, A., Elharbi, M., & Elharbi, M. (2020). Technology-Based Services for Deaf and Dumb People. *International Journal of Data Science*, 5(2), 160–167.

15. Sanmitra, P. R., & Sowmya, V. S. (2021). Machine Learning Based Real-Time Sign Language Detection. *International Journal of Research in Engineering, Science and Management*, 4(6), 137–141.

16. Khaskheli, A. H., Mirani, S. H., & Arain, A. (2021). An Android App for Deaf and Dumb People (Sign Language Recognition–Smart Talk App). *International Journal of Advanced Research in Computer Science*, 12(1), 1–5. Link
17. Hossain, M. S., & Muhammad, G. (2019). Cloud-Assisted Sign Language Recognition Framework Using Smart Gloves. *IEEE Internet of Things Journal*, 6(1), 161–170.
18. Wu, Y., & Huang, T. S. (1999). Vision-Based Gesture Recognition: A Review. In *Proceedings of the International Gesture Workshop* (pp. 103–115). Springer.

