A REAL-TIME SYSTEM TO BRIDGE COMMUNICATION GAPS BETWEEN DEAF AND HEARING COMMUNITIES USING MACHINE LEARNING AND COMPUTER VISION

Alex Zhu [1], Carlos Gonzalez [2]

[1] Head Royce School, 4315 Lincoln Ave, Oakland, CA 94602
[2] Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Communication barriers between the deaf and hearing communities remain a significant challenge due to the lack of widespread knowledge of American Sign Language (ASL) [1]. Motivated by a personal experience at a Boy Scouts summer camp, I developed a real-time ASL translating app to bridge this gap [2]. The app leverages Google's MediaPipe for precise hand landmark detection and the PointNet model for gesture recognition, translating ASL letters into text in real time [3]. Built with Flutter and Dart for a seamless cross-platform experience, the app integrates a Flask-based backend for efficient processing.

Key challenges, including environmental variability and achieving computational efficiency, were addressed through data augmentation, model optimization, and extensive testing. The experimentation demonstrated high accuracy and usability, validating the app's effectiveness across diverse real-world scenarios. Future plans include expanding capabilities to recognize full ASL sentences, integrating text-to-speech functionality, and leveraging cloud storage for scalability.

This project exemplifies how technology can foster inclusivity, creating a practical tool to empower communication and bridge societal gaps.

KEYWORDS

ASL Recognition, Real-Time Translation, Machine Learning, Inclusive Communication

1. INTRODUCTION

Communication is an essential aspect of human connection, yet for millions of deaf and hard-of-hearing individuals, the lack of understanding of American Sign Language (ASL) among the hearing population creates a significant barrier [4]. Over 430 million people worldwide experience hearing loss requiring intervention, and in the United States alone, approximately 500,000 rely on ASL as their primary mode of communication. Despite these numbers, ASL literacy among the general population remains critically low, often leading to feelings of isolation, exclusion, and frustration for those in the deaf community.

David C. Wyld et al. (Eds): SIPO, BDHI, NET, SOEA, CSML, AISCA, MLIOB, DNLP – 2025
pp. 113-121, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.150411

This issue is particularly pressing in real-time, high-stakes situations where clear communication is vital. I experienced this firsthand during a Boy Scouts summer camp when a deaf individual who had lost their way tried to communicate with our group using ASL. None of us understood the gestures, leaving us unable to provide the support they needed. This encounter underscored the urgent need for tools that bridge the communication gap and foster understanding between hearing and deaf individuals [5].

Addressing this problem is essential for promoting inclusivity and breaking down systemic barriers faced by the deaf community. Beyond one-on-one interactions, this issue impacts education, employment, healthcare, and daily life, where seamless communication is critical. By developing a real-time ASL translating app, we can create a transformative solution that empowers the deaf community, fosters stronger societal connections, and demonstrates that inclusivity and accessibility are fundamental rights, not optional luxuries. This project is a step toward a world where language barriers no longer hinder understanding.

1. CNNs for ASL Recognition

CNNs classify ASL gestures by extracting spatial features from images. This method achieves high accuracy in static gesture recognition under controlled conditions. However, it struggles with dynamic gestures, environmental variations, and real-time processing due to high computational demands. My project improves this by using MediaPipe for precise hand tracking and PointNet for processing 3D spatial data, enhancing robustness and real-time capabilities.

2. CNN-SVM Hybrid Models

This approach combines CNNs for feature extraction with SVMs for classification, improving accuracy for distinguishing similar gestures [6]. However, it increases computational complexity, lacks scalability with large datasets, and is ineffective for dynamic gestures. My project addresses these issues by replacing SVMs with the PointNet model, optimizing real-time performance, and supporting dynamic gesture recognition.

3. Real-Time Neural Networks

This method uses neural networks to classify gestures from video frames, enabling real-time ASL recognition. Limitations include sensitivity to environmental changes, difficulty with nuanced gestures, and high computational requirements. My project integrates advanced tools like MediaPipe and PointNet to enhance environmental adaptability, dynamic gesture support, and computational efficiency.

The proposed solution is a real-time American Sign Language (ASL) translating app that uses deep learning and computer vision to recognize ASL gestures and convert them into text or speech in real time. This app aims to bridge the communication gap between deaf individuals and those who do not understand ASL by providing an intuitive and accessible tool for seamless interaction.

This solution addresses the problem by leveraging the widespread availability of smartphones and advancements in artificial intelligence. Using the device's camera, the app captures ASL gestures, processes them using trained machine learning models, and translates them into text or audio outputs [7]. This ensures communication is not hindered by language barriers in diverse settings, including schools, workplaces, healthcare facilities, and emergency situations.

The app's reliance on cutting-edge deep learning models makes it highly effective, as these algorithms excel at recognizing complex gestures and adapting to individual variations in signing styles. Unlike traditional methods, such as hiring interpreters or using cumbersome sensor-based devices, this app offers a scalable, cost-effective, and user-friendly alternative that is always accessible.

Moreover, the app's continuous learning capability allows it to improve over time by analyzing and incorporating new data, ensuring its relevance across various users and contexts. By promoting independence and fostering inclusivity, this solution stands out as a superior approach to bridging the communication gap between the deaf and hearing communities, empowering both groups to interact more effectively and build stronger connections.

In Experiment A, the goal was to evaluate the model's ability to minimize loss and maximize accuracy during training using the ASL Alphabet dataset. The experiment involved training a PointNet model over multiple epochs with a loss function and optimizer to fine-tune parameters. Results showed a rapid decline in loss (approaching 0.01) and a steady increase in accuracy (nearing 0.999), indicating the model's robustness and effectiveness in recognizing ASL gestures. The findings reflect the strength of the dataset, preprocessing techniques, and model architecture in achieving high performance.

In Experiment B, user satisfaction was tested by having 10 participants evaluate the app's usability, speed, accuracy, visual appeal, and overall satisfaction. Participants rated their experience on a 1-10 scale after performing 10 gestures. High ratings for accuracy (8.8) and ease of use (8.9) demonstrated the app's effectiveness, while slightly lower scores for speed (8.1) and visual appeal (7.9) highlighted areas for improvement. These results reflect the app's strengths and its optimization needs.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. The machine learning model

One major challenge is ensuring the machine learning model can accurately recognize ASL gestures despite variations in lighting, hand positions, and backgrounds. These factors can significantly impact the accuracy of gesture recognition, leading to inconsistent results. To address this, I could use a diverse and extensive dataset that includes examples under varying conditions to train the model. Additionally, data augmentation techniques, such as altering brightness, contrast, and orientation, could help the model generalize better. Fine-tuning hyperparameters and leveraging pre-trained models optimized for gesture recognition could also enhance performance across different scenarios.

2.2. Real-time processing

Real-time processing is crucial for the app’s usability, but achieving this can be challenging due to the computational complexity of recognizing gestures and converting them into text or speech with minimal delay. To tackle this, I could optimize the integration of tools like MediaPipe for gesture detection and Flask for backend processing [10]. Using lightweight models and efficient algorithms could reduce computational overhead. Additionally, techniques like frame skipping and parallel processing could help maintain responsiveness without compromising accuracy, ensuring a seamless user experience even on less powerful devices.

2.3. Consistency

Ensuring the app works consistently across different devices is another significant challenge. Variations in camera quality, screen sizes, and processing capabilities could affect the app's functionality. To overcome this, I could design the app to be hardware-agnostic, testing it on a range of devices to identify and resolve compatibility issues. Employing adaptive resolution scaling and platform-specific optimizations could enhance performance across devices [9]. Additionally, leveraging cloud-based processing as a fallback for lower-spec devices could ensure reliability while maintaining accessibility for a wide audience.

3. SOLUTION

The program is structured around three major components: the front-end user interface, gesture recognition and processing (MediaPipe and PointNet model), and the backend server (Flask). These components work together to provide seamless real-time translation of ASL gestures into text.

The program begins at the Start Screen, where users access the app. The live camera feed captures ASL gestures and sends video frames to the gesture recognition system, which uses Google's MediaPipe to extract precise hand landmark coordinates. These spatial coordinates are then transmitted to the backend server via Flask.

At the backend, the PointNet machine learning model processes the gesture coordinates to identify the ASL gesture being signed. This model assigns a label, such as "A," "B," or "C," representing the recognized letter. The backend ensures accurate predictions by leveraging the trained model, which has been fine-tuned for diverse conditions.

Once the gesture is identified, the translated text is sent back to the front-end interface, where it is displayed on the screen in real time. The UI also includes control buttons for functionalities like stopping the camera or capturing a snapshot, ensuring the app is user-friendly.

The app is developed using Flutter and Dart in Android Studio for a cross-platform solution, while Flask facilitates backend processing. MediaPipe and PointNet power the gesture recognition system. Together, these components ensure the program runs smoothly from capturing gestures to delivering real-time translations, bridging the communication gap effectively.

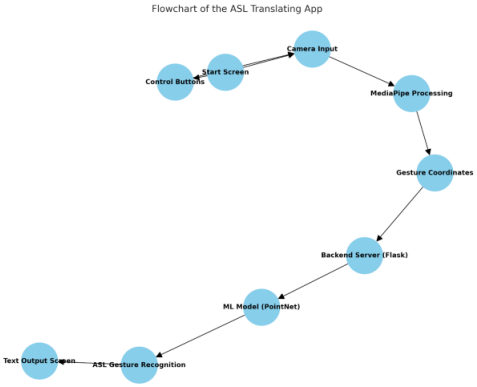

Figure 1. Overview of the solution

The gesture recognition and processing component is essential for interpreting ASL gestures. It utilizes Google's MediaPipe to detect hand landmarks, extracting spatial coordinates through advanced computer vision techniques [8]. By leveraging neural networks, this component transforms raw video input into actionable data, enabling real-time gesture recognition and seamless integration with the program's backend.

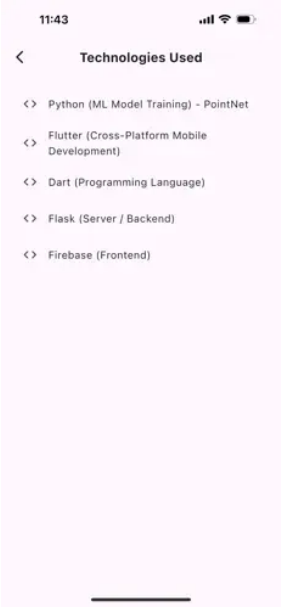

Figure 2. Screenshot of the APP

4. EXPERIMENT

4.1. Experiment 1

Experiment A is to evaluate the effectiveness of a machine learning model in accurately recognizing and translating ASL letters into text using a structured training and testing pipeline.

The experiment focuses on training and testing a machine learning model to recognize ASL gestures using the ASL Alphabet dataset and PointNet architecture. Data preprocessing includes normalization, augmentation, and splitting into training, validation, and testing sets. The model is trained using Cross-Entropy Loss, the Adam Optimizer, and early stopping over multiple epochs. Its performance is evaluated using metrics like accuracy and F1-score on unseen data. Integration with Google's MediaPipe allows real-time gesture recognition by extracting hand landmarks. Testing in varied real-world scenarios ensures robustness, with the ultimate goal of creating an accurate, efficient app for seamless ASL-to-text translation.

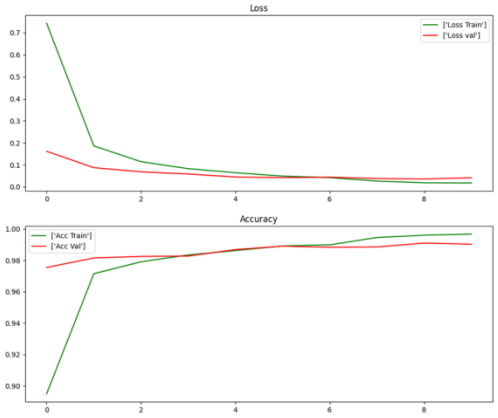

Figure 3. Figure of experiment 1

The data from Experiment A demonstrates rapid and consistent improvement in both loss and accuracy metrics. After just two training epochs, the loss sharply declines, approaching a minimal value near 0.01, indicating that the model is quickly minimizing errors during prediction. Simultaneously, the accuracy increases significantly, nearing 0.999 within the same timeframe, showcasing the model's ability to correctly classify ASL gestures with exceptional precision.

This performance suggests that the model's architecture and training setup—such as the use of PointNet, effective loss function, and optimization techniques—are highly effective. The steep early improvements and steady convergence to near-perfect accuracy highlight the model's capacity to generalize well from the training data to unseen inputs. Such results also underline the robustness of the dataset and preprocessing techniques used.

Overall, the experiment validates the chosen methodology and reinforces the model's potential for real-time ASL-to-text translation in practical applications.

4.2. Experiment 2

Experiment B is to assess user satisfaction with the ASL translating app's usability, accuracy, and effectiveness in real-time ASL-to-text translation.

This experiment evaluates user satisfaction with the ASL translating app using a diverse group of 10 participants, including ASL users and non-users. Participants perform 10 ASL gestures using the app and rate their experience on accuracy, speed, ease of use, visual appeal, and overall satisfaction (1-10). Researchers observe interactions to note usability challenges. Feedback forms provide quantitative and qualitative insights, with metrics like mean and median satisfaction scores analyzed. The experiment aims to identify strengths and improvement areas, focusing on the app's usability, accuracy, and real-time performance. Results will guide enhancements to ensure the app effectively bridges communication gaps.

Participant	Accuracy	Speed	Ease of Use	Visual Appeal	Overall Satisfaction
Participant 1	9	8	9	8	9
Participant 2	8	7	8	7	8
Participant 3	10	9	10	9	10
Participant 4	9	8	9	8	9
Participant 5	7	6	7	6	7
Participant 6	9	9	10	9	9
Participant 7	10	10	10	9	10
Participant 8	8	7	8	7	8
Participant 9	9	8	9	8	9
Participant 10	9	9	9	8	9

Figure 4. Figure of experiment 2

The experiment results show strong user satisfaction overall. The mean scores for Accuracy, Ease of Use, and Overall Satisfaction are all high (8.8-8.9), with medians of 9.0, indicating consistency in user feedback. Speed has a slightly lower mean of 8.1 and median of 8.0, while Visual Appeal is the lowest-rated category with a mean of 7.9. The lowest recorded score across all categories is 6 (for Speed and Visual Appeal), and the highest is 10 (appearing in all categories).

The lower scores for Speed and Visual Appeal suggest areas for improvement, possibly due to delays in real-time processing or UI design preferences. These factors likely impacted overall satisfaction, emphasizing the need for optimization and design refinement. High accuracy and ease of use reflect the app's effectiveness and intuitive interface, demonstrating its success in bridging communication gaps. User feedback and real-world testing had the most significant impact on the results.

5. RELATED WORK

Convolutional Neural Networks (CNNs) have been widely used for American Sign Language (ASL) recognition, leveraging their ability to extract spatial features from visual data. These models classify static ASL gestures by analyzing image-based inputs, often achieving high accuracy in controlled environments. Pre-trained architectures, such as GoogLeNet and ResNet, are commonly fine-tuned for ASL classification tasks. While effective for static gestures, CNNs face limitations in recognizing dynamic movements, adapting to environmental variations, and requiring extensive labeled datasets. These challenges highlight the need for additional advancements to achieve robust, real-time ASL recognition [11].

The hybrid methodology combining Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs) is utilized for American Sign Language (ASL) gesture recognition. CNNs extract visual features from gesture images, while SVMs classify these features into categories such as letters or words. This approach enhances classification accuracy, particularly for similar gestures, and leverages the strengths of both methods. However, it faces limitations in scalability, computational complexity, and real-time dynamic gesture recognition, reducing its effectiveness in diverse environments. Further innovations, such as real-time processing and support for dynamic gestures, are required for broader applicability [12].

Real-time sign language recognition using neural networks processes video frames to identify ASL gestures by classifying patterns in hand shapes and movements. This approach enables practical, real-time communication applications and performs well with static and simple dynamic gestures in controlled environments. However, it faces challenges in handling environmental variability, complex gestures, and contextual elements like facial expressions and motion sequences. The high computational demands of real-time processing further limit its scalability. Innovations such as integrating advanced hand-tracking tools and 3D spatial data processing are required to address these limitations and improve versatility and robustness [13].

6. CONCLUSIONS

My project effectively bridges communication gaps, but there are limitations to address for enhanced usability. First, recognizing only individual letters limits the app's real-world application. Expanding to full ASL sentence recognition requires incorporating sequence models like Recurrent Neural Networks (RNNs) or Transformers to process temporal data [14]. Second, environmental variability, such as poor lighting or cluttered backgrounds, could affect accuracy. This can be mitigated by integrating advanced preprocessing techniques or augmenting training data to reflect diverse scenarios.

Another limitation is the absence of text-to-speech capabilities, which reduces accessibility for visually impaired users. Implementing a text-to-speech API could address this [15]. Cross-platform compatibility needs improvement to ensure seamless functionality across devices, achievable through frameworks like Flutter. Additionally, the lack of cloud-based storage limits scalability; incorporating cloud infrastructure allows for efficient updates and model sharing.

With more time, these enhancements would make my app a comprehensive tool for inclusive communication across diverse real-world settings.

My project demonstrates significant progress in bridging the communication gap between deaf individuals and non-ASL users. By integrating advanced technologies like MediaPipe and PointNet, it achieves real-time, accurate gesture recognition. With further refinements, such as dynamic gesture support and improved environmental adaptability, this app can make a lasting societal impact.

REFERENCES

[1] Rink, Isabel. "Communication barriers." Handbook of accessible communication. Frank & Timme, Berlin, 2024. 33-68.
[2] Ardiansyah, Andra, et al. "Systematic literature review: American sign language translator." Procedia Computer Science 179 (2021): 541-549.
[3] Charles, R. Qi, et al. "PointNet: Deep learning on point sets for 3D classification and segmentation." 2017 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, 2017.
[4] Newport, Elissa L., and Richard P. Meier. "The acquisition of American sign language." The crosslinguistic study of language acquisition. Psychology Press, 2017. 881-938.
[5] Angeli, Simon, Xi Lin, and Xue Zhong Liu. "Genetics of hearing and deafness." The Anatomical Record: Advances in Integrative Anatomy and Evolutionary Biology 295.11 (2012): 1812-1829.
[6] Niu, Xiao-Xiao, and Ching Y. Suen. "A novel hybrid CNN–SVM classifier for recognizing handwritten digits." Pattern Recognition 45.4 (2012): 1318-1325.
[7] Sullivan, Emily. "Understanding from machine learning models." The British Journal for the Philosophy of Science (2022).
[8] Siju, Ijas, and Prabu Selvam. "A Novel Approach for Lightweight Sign Language Recognition Leveraging Google Mediapipe and Deep Neural Net." 2024 First International Conference on Software, Systems and Information Technology (SSITCON). IEEE, 2024.
[9] Köster, Marcel, et al. "Platform-specific optimization and mapping of stencil codes through refinement." Proceedings of the 1st International Workshop on High-Performance Stencil Computations. Vol. 21. 2014.
[10] Chan, Jack, Ray Chung, and Jack Huang. Python API Development Fundamentals: Develop a full-stack web application with Python and Flask. Packt Publishing Ltd, 2019.
[11] Koller, Oscar, Hermann Ney, and Richard Bowden. "Deep learning of mouth shapes for sign language." Proceedings of the IEEE International Conference on Computer Vision Workshops. 2015.
[12] Gupta, Anuj Kumar, and Shaminder Singh. "Hand Gesture Recognition System Based on Indian Sign Language Using SVM and CNN." International Journal of Image and Graphics (2024): 2650008.

[13] Garcia, Brandon, and Sigberto Alarcon Viesca. "Real-time American sign language recognition with convolutional neural networks." Convolutional Neural Networks for Visual Recognition 2.225-232 (2016): 8.
[14] Medsker, Larry R., and Lakhmi Jain. "Recurrent neural networks." Design and Applications 5.64-67 (2001): 2.
[15] Hallahan, William I. "DECtalk software: Text-to-speech technology and implementation." Digital Technical Journal 7.4 (1995): 5-19.