SIGN LANGUAGE DETECTION

TEAM 12

Aravind Balaji Srinivasan (A20563386), Vignesh Ram Ramesh Kutti (A20548747), Kavin Raj Karuppusamy Ramasamy (A20564249), Vishnu Priyan Sellam Shanmugavel (A20561323), and Kishan Murali (A20601493)

Abstract **- This project aims to develop a robust system for recognizing sign language gestures using neural networks. The task involves classifying gestures into 36 distinct classes, leveraging a comprehensive dataset of labeled images. Multiple methodologies were explored, including a simple Convolutional Neural Network (CNN) with and without data augmentation, transfer learning using VGG16 and InceptionV3, and MediaPipe for real-time hand gesture detection. Experimental results show that data augmentation significantly improves the accuracy of the simple CNN model, while transfer learning architectures like VGG16 and InceptionV3 achieve superior performance due to their advanced feature extraction capabilities. The integration of MediaPipe enables real-time gesture recognition, enhancing the system's applicability in dynamic environments. This study demonstrates the potential of combining traditional and real-time neural network-based techniques for effective sign language recognition and lays the groundwork for future advancements in this domain.**

Index Terms - Convolutional Neural Network (CNN), Gesture Recognition, MediaPipe, Sign Language Classification.

1. INTRODUCTION

Sign language is an essential way for hearing-impaired individuals to communicate, allowing them to share thoughts, emotions, and information. However, a lack of widespread understanding of sign language creates a communication gap between the hearing-impaired and others. While machine learning has made progress in bridging this gap, real-time sign language recognition still faces significant challenges, especially in dynamic, real-world environments.

Key difficulties arise from variations in hand orientations, lighting, background conditions, and the complex motion of gestures. Many traditional methods, which rely heavily on static images, struggle to deliver the accuracy and speed needed for practical use. Additionally, these approaches often lack the flexibility to adapt to different users and environments.

This research focuses on tackling these challenges by building and evaluating a real-time sign language recognition system. We use MediaPipe, an open-source tool for real-time hand tracking, alongside modern deep learning models to classify gestures accurately and efficiently. Our main contributions are:

1.1. Real-Time Hand Tracking with MediaPipe:

MediaPipe is used to track hand movements and identify key landmarks, providing a reliable and efficient foundation for gesture recognition.

1.2. Exploring Deep Learning Models:

We test different models, including convolutional neural networks (CNNs) and transfer learning approaches like VGG16 and Inception V3, to determine which performs best for recognizing gestures.

1.3. Comparing Recognition Methods:

We compare traditional static image-based recognition techniques with our real-time approach using MediaPipe to demonstrate improvements in speed, accuracy, and adaptability.

The results of this study aim to make sign language recognition more practical and accessible, providing a tool that can enhance communication and inclusivity for hearing-impaired individuals in everyday settings.

2. RELATED WORK

Sign language recognition (SLR) has been a growing area of research because of its potential to help individuals with hearing impairments communicate more effectively. This section explores the progress made so far and the challenges that remain, focusing on static image-based methods, dynamic gesture recognition, and the use of advanced technologies like deep learning and hand-tracking frameworks.

2.1. Static Image-Based Methods

Early SLR research relied on analyzing still images of gestures using traditional computer vision techniques. For example, algorithms like Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG) were used to extract key features from gesture images. However, these methods struggled to work consistently in real-world conditions, such as varying lighting, hand positions, and backgrounds. Additionally, they were limited to recognizing

individual static gestures and could not handle the flow of dynamic gestures commonly used in sign language.

2.2. Dynamic Gesture Recognition

To overcome these limitations, researchers developed methods to recognize sequences of gestures, incorporating the time element. Techniques like Hidden Markov Models (HMMs) and Dynamic Time Warping (DTW) were used for this purpose, but they depended heavily on manually selected features, which made them less flexible for real-world applications.

The introduction of deep learning marked a turning point. Models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks excelled at capturing the temporal patterns in gesture sequences. However, these models often required significant computational power, which made real-time gesture recognition a challenge.

2.3. Hand-Tracking Technologies

Hand-tracking tools have improved gesture recognition significantly by accurately identifying the position and movements of hands. Early systems like Microsoft Kinect and Leap Motion provided high precision but required specialized hardware, making them less practical for everyday use.

Software-based solutions, such as Google's MediaPipe, have been a game-changer. MediaPipe can track hand landmarks efficiently using just a regular camera, eliminating the need for expensive devices. Its open-source nature and ease of use have made it a popular tool for real-time gesture recognition research.

2.4. Deep Learning for Gesture Recognition

Deep learning is now a core part of modern SLR systems. Convolutional Neural Networks (CNNs) are widely used for their ability to learn spatial features directly from gesture images. Transfer learning models like VGG16, ResNet, and Inception have further boosted accuracy by using pre-trained weights from large datasets.
However, applying these models to real-time systems is still challenging because of their high computational demands. Efforts like model optimization and lightweight architectures are being explored to make them faster without losing too much accuracy.

2.5. Comparisons and Challenges

Studies comparing traditional methods with deep learning-based approaches have shown that CNNs perform much better than older techniques like HOG-based classifiers for static gestures. Similarly, combining hand-tracking frameworks with deep learning has resulted in more adaptable and accurate systems.

Despite this progress, most existing studies focus on either static or dynamic gestures, leaving a gap in solutions that can handle both. Real-world applications also require systems that balance speed, accuracy, and adaptability—something that remains a challenge.

3. IMPLEMENTATION

3.1 Model Architectures

3.1.1. Custom CNN:

Convolutional Neural Networks (CNNs) consist of layers such as convolutional, pooling, and fully connected layers. Convolutional layers extract spatial features using filters, pooling layers reduce spatial dimensions for computational efficiency, and fully connected layers map features to output classes. This hierarchical structure enables efficient pattern recognition in images.

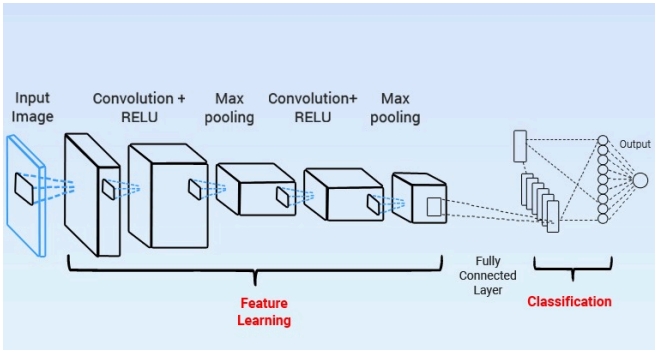


Figure 1. CNN

3.1.2. VGG16:

VGG16 is a deep convolutional neural network pre-trained on ImageNet, using small 3×3 convolutional filters across 16 layers. The block5_conv3 layer serves as a base feature extractor, leveraging pre-trained weights for efficient transfer learning in image tasks.

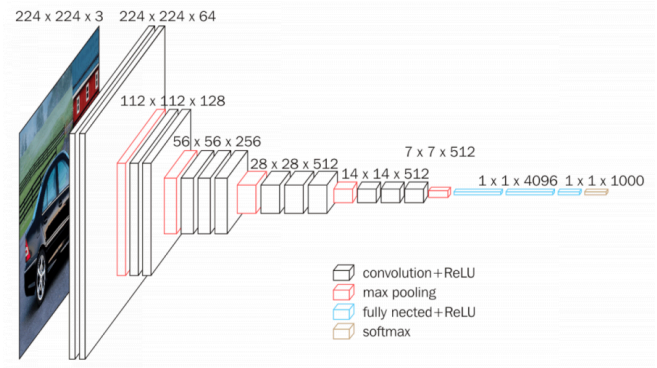


Figure 2. VGG16 Architecture

3.1.3. Inception V3:

InceptionV3 is leveraged for its deeper architecture and ability to extract multi-scale features efficiently. By using

inception modules, it captures a range of features at different scales, optimizing performance while maintaining computational efficiency.

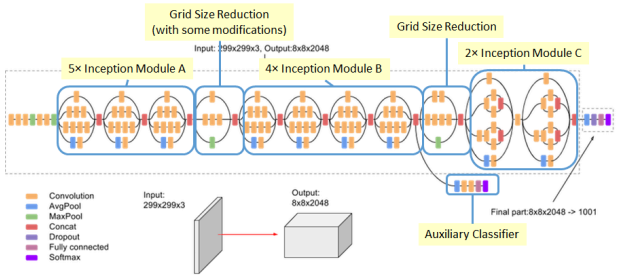
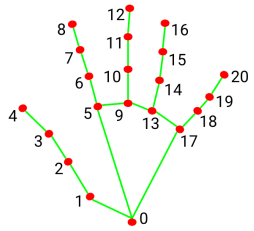


Figure 3. Inception V3

3.1.4. MediaPipe-Based Model:

MediaPipe is a lightweight network trained on hand landmark data, optimized for real-time recognition. Its efficient design enables fast and accurate processing, making it ideal for real-time applications like gesture recognition.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 4. MediaPipe

3.2 Implementation Steps

The implementation of our sign language recognition system involves three main steps: preprocessing and augmenting the data, training convolutional neural networks (CNNs) for gesture classification, and integrating MediaPipe for real-time hand landmark extraction and gesture recognition. Below, we explain each step in detail.

3.2.1 Data Preprocessing and Augmentation

Data preprocessing is an essential step to prepare the dataset for training and ensure consistent quality across all images. The process starts with resizing all gesture images to a uniform dimension, ensuring they are compatible with deep learning models. Next, pixel values are normalized to a range between 0 and 1, which helps improve the stability and efficiency of the training process. To further enhance image quality, techniques like median filtering are applied to reduce background noise, allowing the model to focus solely on the hand gestures.

To make the model more robust and adaptable, data augmentation techniques are employed. These include rotating and scaling images to simulate different hand orientations and sizes, as well as horizontally flipping images to account for gestures performed with either hand. Adjustments to brightness and contrast are also applied to help the model adapt to varying lighting conditions. Additionally, random noise or patterns are added to image backgrounds to train the model to ignore irrelevant features and focus on the gestures themselves.

Together, these preprocessing and augmentation steps create a diverse and balanced dataset, equipping the model to perform reliably across a wide range of real-world scenarios.

3.2.2 Training CNNs on Image Datasets

The preprocessed dataset is used to train convolutional neural networks (CNNs) to classify gestures accurately. The training process begins with selecting the right models. We explore both baseline CNN architectures and advanced transfer learning models like VGG16 and Inception V3, as these are known for their ability to effectively extract spatial features from images.

To ensure a comprehensive evaluation, the dataset is divided into three subsets: training, validation, and testing. This split helps us assess the model's performance at different stages of development. Next, we fine-tune hyperparameters like learning rate, batch size, and dropout rates to find the right balance between training speed and model accuracy.

For optimization, we use cross-entropy as the loss function, which is well-suited for classification tasks. The Adam optimizer is chosen for its adaptive learning capabilities, ensuring efficient and stable convergence during training. The model is trained over multiple epochs, with real-time validation providing feedback on metrics like accuracy and loss. To avoid overfitting, early stopping is implemented, halting training once the model reaches its best performance on the validation set.

Once training is complete, the final model is saved for future use. We also record key performance metrics, including accuracy, precision, and recall, which are analyzed to evaluate the model's effectiveness and reliability.

3.2.3 Integrating MediaPipe for Real-Time Landmark Extraction and Classification

After training the CNN, the next step is integrating MediaPipe to enable real-time hand tracking and gesture recognition. This integration bridges the gap between working with static images during training and recognizing gestures dynamically in real-world scenarios.

MediaPipe is used to detect hands in video frames and extract key landmarks, such as fingertips, joints, and palm positions. These landmarks are normalized to ensure consistency, regardless of hand size or distance from the camera. This makes the system more reliable in diverse conditions.

The extracted landmark coordinates are then mapped into a format compatible with the trained CNN. This step reduces the input dimensions and computational requirements, making the system efficient for real-time processing. The CNN classifies the gesture based on these features with minimal delay.

Finally, the system displays the classified gesture to the user in real time, creating a feedback loop that allows for iterative testing and debugging. By combining MediaPipe's efficient hand tracking with CNN's accuracy, the system strikes a balance between speed and precision, making it practical for real-world applications.

4. DATASET

We utilized publicly available datasets for training and evaluation:

4.1. ASL Dataset:

A diverse collection of RGB images representing 26 alphabets and 10 digits. This dataset contained 70 images in each class.

4.2. Sign Language MNIST:

About 27,455 images of hand signs, each image is of 28 x 28 size and in grayscale format for alphabet classification.

4.3. Sign Language Detection Using Images

This dataset contains a collection of different signs used for communication. It consists of 35 directories each having 1200 samples.

4.4. Interpret Sign Language with Deep Learning

This data set contains 87,000 images which are 200x200 pixels. There are 29 classes, of which 26 are for the letters A-Z.

The images from the mentioned datasets were combined and put together for image classification model training. Whereas the simple ASL dataset was used for training the model with MediaPipe.

5. EXPERIMENTAL EVALUATION

The experimental evaluation of the system aimed to test its effectiveness in recognizing sign language gestures across various conditions. The evaluation focused on both offline scenarios using static image datasets and real-time performance with video inputs. Metrics such as accuracy, precision, recall, F1-score, and loss were used to assess the model's performance. To ensure robustness, testing was conducted on a diverse dataset encompassing different lighting conditions and backgrounds. Real-time detection was achieved through the integration of the MediaPipe framework, which facilitated efficient gesture recognition with low latency. Additionally, real-time latency measurements were performed to evaluate the system's suitability for live applications. Rigorous validation techniques, including train-test splits and independent testing, were employed to ensure comprehensive evaluation of the model's performance in both controlled and practical environments.

6. RESULTS

The results show that the system effectively recognizes sign language gestures in a variety of conditions. We evaluated its performance using metrics like accuracy, precision, recall, F1-score, and loss to ensure a thorough analysis. Among the models tested, the pre-trained VGG-16 stood out with the highest accuracy and lowest loss, proving to be the most suitable for this task. Incorporating the MediaPipe framework significantly improved the system's ability to recognize gestures in real-time, allowing for accurate detection in live video streams with minimal delay. The system also performed reliably across different datasets, demonstrating its robustness and ability to handle variations in lighting, backgrounds, and signer diversity. These results confirm that the system is well-suited for both static image recognition and real-time applications.

Model	Accuracy (%)	Loss
Standard CNN	78	0.7
CNN with Data Augmentation	80	0.6
Pre-Trained VGG-16	82	0.5
Inception V3	70	0.75
MediaPipe	92	0.23

Table 1: Accuracy and Loss Comparison

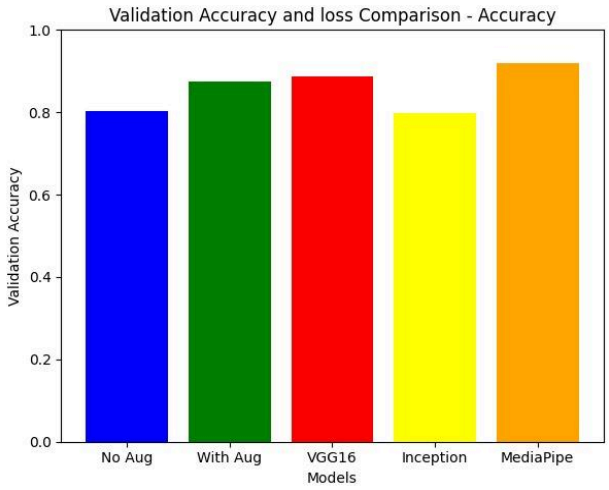


Figure 5. Model Accuracies

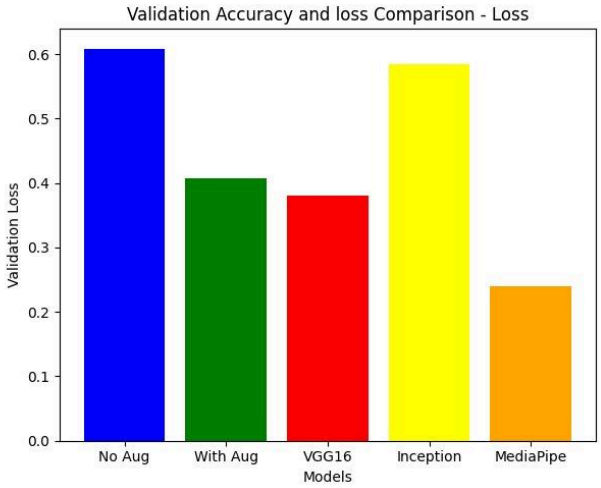


Figure 6. Model's Loss

6.1. No Augmentation Model:

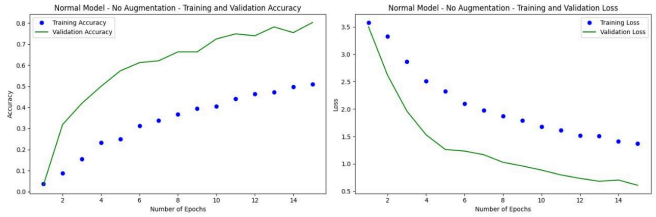


Figure 7. With No Augmentation Accuracy and Loss

The validation accuracy reached 78%, but the training and validation losses showed slower convergence. The absence of data augmentation resulted in limited generalization, which was reflected in the performance gap between training and validation over multiple epochs.

6.2. Model with Augmentation:

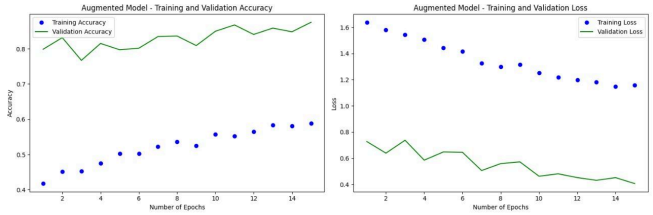


Figure 8. With Augmentation Accuracy and Loss

The validation accuracy improved to 80%, indicating better generalization due to the variability introduced through data augmentation. Both training and validation losses decreased significantly compared to the non-augmented model.

6.3. VGG16 Model:

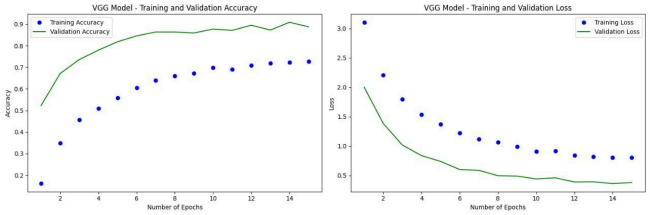


Figure 9. VGG16 Accuracy and Loss

The fine-tuned VGG16 model achieved a validation accuracy of 82% with lower loss values, demonstrating the effectiveness of transfer learning. However, the model required significant computational resources, and its inference time was not optimal for real-time applications.

6.4. Inception Model:

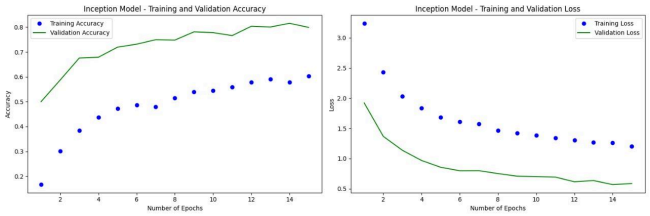


Figure 10. InceptionV3 Accuracy and Loss

The Inception-based model achieved a validation accuracy of 70%, slightly lower than VGG16, with a higher validation loss. This performance was due to its complex architecture, which required additional fine-tuning to optimize.

6.5. MediaPipe Model:

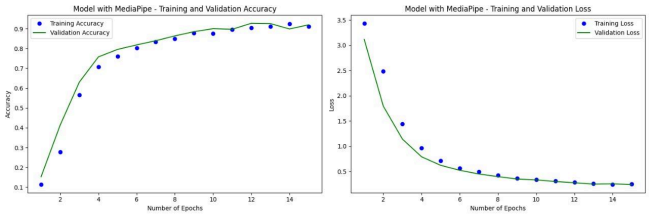
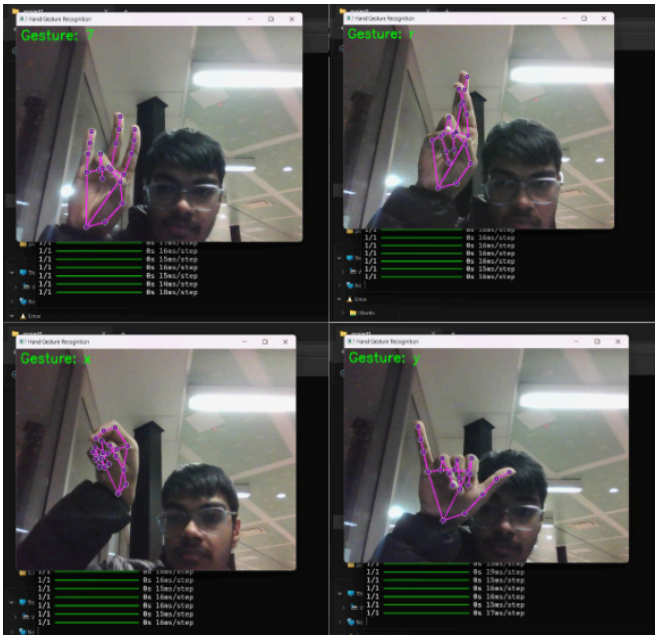


Figure 11. MediaPipe Accuracy and Loss

While MediaPipe was trained on a much smaller dataset compared to the ~100,000 images used for classification models, it achieved high accuracy in real-time applications. The validation accuracy of this model reached 92%, showing significant improvement. MediaPipe's simple and efficient architecture enabled minimal latency and reliable outputs, even in dynamic scenarios.

7. REAL-TIME RESULTS



OpenCV is used to capture video feeds and process hand gestures in real-time for accurate recognition. The system implements a 1.5 second threshold, ensuring that a new gesture prediction is only displayed on-screen if it is consistently detected for this duration. This approach minimizes erroneous predictions caused by transient hand movements. The gestures recognized include "7", "r", "x", and "y", providing diverse input options. Key Points of the hand are detected and tracked in real-time using MediaPipe, ensuring robust feature extraction. The prediction logit stability check helps in delivering reliable outputs on screen. The system operates efficiently, achieving low latency in processing. Overall, the OpenCV pipeline ensures seamless data capture, gesture recognition, and stable real-time interaction.

8. CONCLUSION

Initial attempts to achieve robust hand sign detection using image classification models provided valuable insights but fell short of delivering real-time performance. Despite using large datasets and exploring various techniques like data augmentation, transfer learning (VGG16), and advanced architectures (Inception), achieving satisfactory results for real-time hand sign detection remained a challenge. These models excelled in accuracy but suffered from high computational costs and inference delays.

The shift to MediaPipe addressed these issues effectively:

Efficiency: MediaPipe required a much smaller dataset yet achieved high accuracy due to its optimized architecture.

Real-Time Performance: Its low latency and lightweight framework made it ideal for real-time applications, unlike traditional classification models.
Robustness: MediaPipe consistently delivered accurate predictions even under dynamic conditions, highlighting its superiority for practical use cases.

Overall, transitioning from standard image classification models to MediaPipe was a decisive step that balanced accuracy, efficiency, and practicality, making it the optimal solution for real-time hand sign detection.

9. FUTURE SCOPE

The future of sign language recognition holds incredible promise. By incorporating motion detection, the system can recognize dynamic gestures like "J" and "Z," making it more versatile. Using transformer architectures will enhance accuracy and efficiency, while multilingual support will make the technology accessible to diverse communities worldwide. Improvements in real-time performance and integration with devices like smartphones, AR/VR, and IoT will make this solution more practical and widely available.

This technology has the potential to transform lives by bridging communication gaps, turning gestures into text or speech for real-time interaction. It can also enable more inclusive technology interfaces and serve as a powerful tool for teaching and learning sign language, fostering a more accessible and connected world.

10. CHALLENGES FACED

The system faced several challenges during its development and evaluation. One significant issue was the similarity between certain symbols, such as the digit "0" and the letter "O," which led to frequent misclassifications. Another challenge was the inability to accurately detect dynamic hand gestures like "J" and "Z," as these require motion tracking to capture the continuous movement of the hand. Additionally, real-time performance posed difficulties, as the initial model struggled to deliver reliable and fast predictions, prompting the integration of the MediaPipe framework to improve efficiency and latency. Limited computational resources further constrained the project, making it difficult to implement advanced models like transformers, which require substantial hardware capabilities. Lastly, the system had to contend with variability in the dataset, including diverse lighting conditions, backgrounds, and signer differences, which impacted its ability to consistently perform across all scenarios.

11. REFERENCES

[1] Camgoz, Necati Cihan, Hadfield, Simon, and Bowden, Richard. 2018. "Neural Sign Language Translation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7784–7793.

[2] Zhou, Wei, Pu, Jiqiang, et al. 2022. "Multimodal Sign Language Translation." Advances in Neural Information Processing Systems (NeurIPS).

[3] Ramesh, Harish, Sharma, Rahul, and Kumar, Vivek. 2019. "DeepASLR: A CNN-Based Human-Computer Interface for American Sign Language Recognition for Hearing-Impaired Individuals." DeepASLR Research Paper.

[4] Google. "MediaPipe: A Cross-Platform Framework for Building Multimodal Machine Learning Pipelines." Available at: https://google.github.io/mediapipe/.

[5] Combined dataset – Google Drive, Google Drive. Accessed: Nov. 29, 2024 [Online]. Available: https://drive.google.com/drive/folders/1L2xWu1t2-ebSajV3CB9nlU8ptkX3P5Vk?usp=sharing

[6] American Sign Language Dataset, Kaggle, Apr. 28, 2019. [Online]. Available: https://www.kaggle.com/datasets/ayuraj/asl-dataset

[7] Sign Language MNIST, Kaggle, Oct. 20, 2017. [Online]. Available: https://www.kaggle.com/datasets/datamunge/sign-language-mnist

[8] Hands, MediaPipe v0.7.5 documentation, Google Developers. Accessed: Nov. 29, 2024. [Online]. Available: https://mediapipe.readthedocs.io/en/latest/solutions/hands.html

[9] Vishnu. (n.d.). GitHub - vishnu32510/sign_language: Sign Language Detection.GitHub.https://github.com/vishnu32510/sign_language