ASL Sign Language Analysis

Savan Patel, Alaa Srour, Mai Sayed, Michael Payne

University of Nebraska-Lincoln

Abstract

This project leverages advancements in computer vision to bridge communication gaps for individuals who are deaf or hard of hearing. Current American Sign Language (ASL) recognition systems often face challenges with usability and accuracy, particularly in varying conditions such as inconsistent lighting, diverse backgrounds, and complex hand gestures.

Our proposed framework addresses these limitations by integrating advanced computer vision techniques to extract hands from images, utilizing methods like deep learning-based hand detection. This step enhances gesture recognition by isolating hand regions, reducing noise, and ensuring robust input for classification models. This project aims to improve ASL recognition, enabling seamless communication between signers and non-signers while addressing key limitations of existing solutions.

1.INTRODUCTION

This project addresses the critical challenge of improving the usability and accuracy of American Sign Language (ASL) recognition systems, which often struggle with real-world complexities such as varying lighting, diverse backgrounds, and intricate hand gestures.

The solution begins with hand detection using a single-shot detector model to accurately identify palm regions, followed by a landmark model for 3D hand landmark localization. This two-step approach isolates hands from noisy backgrounds and standardizes hand pose representation, even in cases of occlusions or poor lighting. For gesture classification, we explored both machine learning and deep learning methodologies. In the machine learning approach, we utilized hand landmark coordinates to extract features such as fingertip positions and finger lengths. These features were used to train a Support Vector Machine (SVM) and a Random Forest classifier. On the deep learning side, we employed custom CNN model training as well as transfer learning using pre-trained models such as VGG16, InceptionV3, and MobileNetV2. These models were fine-tuned to classify ASL gestures across 29 classes, including the full alphabet and additional gestures like SPACE, DELETE, and NOTHING. VGG16 achieved the highest accuracy but at the cost of computational efficiency, while InceptionV3 and MobileNetV2 offered a balance between accuracy and performance.

Additionally, we leveraged MediaPipe's 21 key hand landmarks to extract features like hand dimensions and orientation, structuring them into tabular data for further gesture analysis. This feature extraction enhanced the performance of machine learning classifiers, demonstrating the effectiveness of combining geometric and visual information for gesture recognition. By integrating hand detection, landmark-based feature extraction, and deep learning techniques, our methodology addresses the limitations of existing ASL recognition systems and provides a robust solution.

2.OBJECTIVES

1. **Sign recognition through grouped classification**
2. **Individual Sign Classification for All 24 Signs**
3. **Fine-tuning Pre-trained Vision Models for Image Recognition**
4. **Feature Extraction for Gesture Analysis**

3.RELATED WORK

Modern frameworks for sign language translation have added the ability to translate video, which aids in the number of words that can be translated within a given period of time. This expands the context of the model over the time dimension, which adds additional complexity.

One paper from Google [2] showcases a framework built on top of the pre-existing pose estimation model, PoseNet. This has the added benefit of including features for the entire human body. Taking this approach removes the need to feed the entire HD image through the model, and frame-to-frame optical flow can be calculated for each landmark position. Finally, they developed a long-short-term memory (LSTM) architecture to expand the context to utilize the aforementioned time dimension.

Even though pose estimation is used in many of these frameworks as a preliminary step to aid in real time performance, better performance has been achieved in cases where compute power is not as limited through the use of transfer learning. Leveraging these larger

deep-learning models provides more flexibility in the classifier's ability to generalize across new situations. [4] More datasets have been released over time that include external data outside of images and video. It has allowed researchers to create "Continuous Dynamic Sign Language" models that utilize more modalities than just video or images, such as depth data. [5]

State-of the art models like DFCNet+ [6] are able to translate video frames into "dynamic motion trajectories" that are then converted into glosses. Glosses are another textual representation of the visual components of sign language outside of full words that can assist translators.

4.DATASET

The project utilizes the American Sign Language (ASL) Alphabet Dataset [1], a publicly available Kaggle dataset comprising high-resolution images representing the ASL alphabet. Each image corresponds to a single letter of the alphabet and is uniformly sized at 200×200 pixels. The dataset consists of a total of 87,000 images, distributed across 29 classes, with approximately 3000 images per class. Among these, 26 classes correspond to the English alphabet letters (A-Z), while the remaining three classes represent SPACE, DELETE, and NOTHING. For the purpose of this project, only the 26 alphabetic letters were retained, with the SPACE, DELETE, and NOTHING classes removed during preprocessing.

Fig. 1. Sample Images

4.1. PREPROCESSING

The dataset was divided into training, validation, and testing subsets to facilitate robust model evaluation. To ensure a balanced and representative split, 80% of the data was allocated for training, and 20% testing. Furthermore 20% was re-allocated from the training set for validation. During preprocessing, all images were resized to 64×64 pixels to standardize input dimensions while reducing computational overhead. The values of all pixels were changed to the range [0, 1] by division by 255. Normalization helps in maintaining consistency and accelerating model convergence.

4.2. AUGMENTATION

In order to make the training data more diverse, and improve the capability of the model, several augmentation techniques were performed on the training set. Each of these transformations introduces variability while preserving the semantic meaning of the images, thereby effectively simulating different real-world conditions. These included:

Rotation: The model was trained to allow for random rotations of up to 20 degrees, hence enabling the model to recognize signs regardless of minor angular variations in hand orientation. The images used for training were randomly horizontally and vertically shifted by up to 20% of the image dimensions to simulate slight positional variations in the placement of gestures.

Zooming: Random zooming within a range of ±20% was performed to make this model generalize well to any changes in hand size or distance from the camera.

Horizontal Flipping: The images were flipped horizontally with a probability of 50%, considering left and right-handed persons where their gestures could appear as mirror images of each other.

Vertical Flipping: Augmentation like vertical flipping were not performed as in the real-world scenario the signs will never be flipped and also it can change the interpretation of the signs.

These augmentations were made using the ImageDataGenerator class in TensorFlow/Keras, which applies the transformations on the wing at runtime during training. By contrast, neither the validation nor testing subsets have been augmented, to make the performance evaluation of the model comparable.

For objective 4, additional preprocessing was also performed. Images were loaded in batches to prevent memory overload. Each batch contains 500 images of uniform dimensions. Images were converted to grayscale to simplify computations and reduce memory usage while preserving critical information.

5.METHODOLOGY

5.1. OBJECTIVE 1: Sign recognition through grouped classification

We initially wanted to classify the different hand images into groups based on their similarities, meaning closed fist, and each number of fingers extended at once. This would give us the ability to quickly validate our methodology instead of focusing on the intricacies involved in the large number of class types.

In order to detect initial palm locations, a single-shot detector model is employed. This task is complex due to

the need to work across a variety of hand sizes and scales, as well as detecting occluded and self-occluded hands. Training on specifically palms helps to reduce this error but limits the flexibility and ability of the model to generalize. It uses an encoder-decoder feature extractor for greater scene context awareness, and minimized focal loss during training.

After detecting the palm, the landmark model optimizes landmark positions in 3d spaces within the detected hand regions through regression. This approach allows the model to learn a consistent internal hand pose representation and remain robust even in cases of partially visible hands and self-occlusions, which was particularly important for our dataset that included widely-varying lighting conditions.

Choosing this route for implementation had downsides, in part because it is better suited for real-time and embedded scenarios. This means the training workload cannot be easily parallelized or utilize any graphics acceleration to train a large model and detection has to be done individually before the classification model can be trained. This presents a major issue because the entire dataset must be loaded into memory before training can occur.

Once we were able to extract landmarks from the entire dataset, there were two methods of training a model that we wanted to attempt. The first approach was to train a Support Vector Machine based on a simplified representation of the hand model. This would be done using a list of normalized 2D vectors describing each fingertip's position in relationship to the wrist. The second was using all 21 (x,y) coordinate pairs to train a Random Forest classifier.

The main goal of an SVM is to find the best line (or hyperplane) that is able to separate different classes with the most margin. The support vectors are the critical data points closest to the hyperplane that determine its position and orientation.

Importantly for our dataset, SVMs can handle non-linearly separable data. For non-linear data, kernel functions are used to transform the input space into a higher-dimensional feature space where the classes become linearly separable. We chose the Radial Basis Function kernel for our testing and validation. A sigmoid kernel could similarly be used in its place. The choice of how to handle outliers depends on whether a hard margin or soft margin approach is being used. The main trade-off to be aware of when fine-tuning the margin is to maximize the margin from the hyperplane and minimize misclassifications.

The Random Forest is an ensemble learning method, meaning it combines multiple models (decision trees) to make predictions. This approach helps improve accuracy and reduce overfitting. The algorithm uses a technique called Bootstrap Aggregating (bagging) to create diverse subsets of the training data for each decision tree. This involves randomly sampling the data with replacement, so some samples may appear multiple times while others may not appear at all.

In addition to bagging the training data, Random Forest also uses feature bagging. At each split point in a decision tree, the algorithm randomly selects a subset of features to consider for splitting. This helps prevent any single feature from dominating the model and encourages diversity among the trees. This aspect is particularly useful for our application where recognizing the split between different important fingers that convey information and unimportant fingers if used in a closed fist.

The algorithm continues splitting nodes until a stopping point is met, such as reaching a maximum depth or having a minimum number of samples in each leaf node. This helps control the complexity of the model and prevent overfitting. When making predictions, each decision tree in the forest independently predicts the class of the input data. The final prediction is determined by aggregating the predictions of all the trees and choosing the majority vote for the determined class.

These two methods yielded mixed, but positive results, with the random forest approach outperforming SVM.

5.2. OBJECTIVE 2: Individual Sign Classification for All 24 Signs

To evaluate the performance of a model trained from scratch against pre-trained architectures (addressed in Objective 3), a custom Convolutional Neural Network (CNN) was designed and implemented. This model aimed to classify the 26 ASL alphabet letters, leveraging the processed dataset described earlier. The architecture was carefully selected to balance computational efficiency and accuracy while incorporating modern best practices to improve generalization and performance.

The custom CNN consists of three convolutional blocks for feature extraction, followed by fully connected layers for classification. Each convolutional block comprises:

Convolutional Layers: Filters of sizes 3×3 were used, with ReLU activation and "same" padding to preserve spatial dimensions.

Max Pooling Layers: Pooling operations with a 2×2 window were employed to reduce spatial dimensions while retaining essential features.

Batch Normalization: Batch normalization was applied after each convolutional layer to stabilize learning and improve convergence.

Dropout Regularization: Dropout layers were incorporated with a dropout rate of 20% to mitigate overfitting, particularly in deeper layers.

The model was trained on both augmented and non-augmented versions of the dataset to assess the impact of data augmentation on model performance.

Non-Augmented Dataset: The model was initially trained on the non-augmented dataset to establish a baseline performance. This experiment allowed an evaluation of the model's capacity to generalize without any additional variability introduced by augmentations.

Augmented Dataset: The same model was subsequently trained on the augmented dataset, which included transformations such as rotation, zoom, shifts, and flipping. This experiment aimed to evaluate how well the model could generalize when trained on more diverse data.

Early stopping was employed to monitor validation loss, with a patience of three epochs. This ensured that training was halted when the validation loss ceased to improve, preventing overfitting. Additionally, the best-performing weights from training were restored automatically. Both experiments were conducted for a maximum of 20 epochs, with a batch size of 32. The number of steps per epoch and validation steps were dynamically calculated based on the dataset size. Training accuracy, validation accuracy, and test accuracy were recorded for each experiment, and the results were compared.

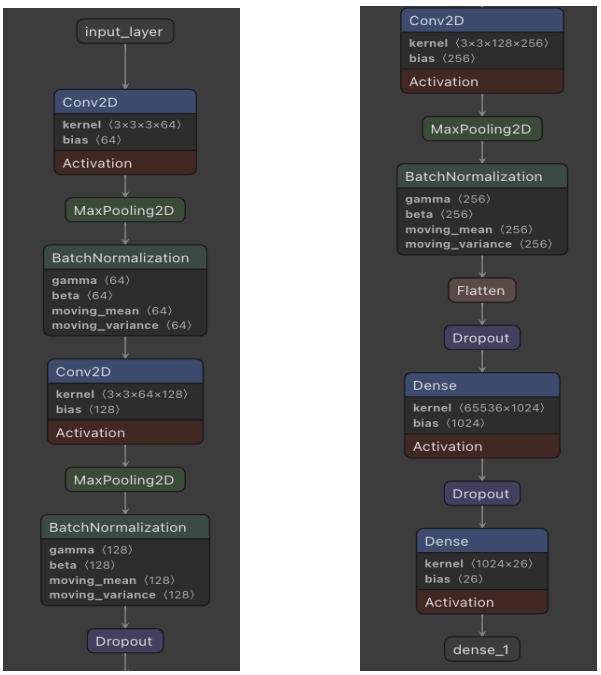

Fig. 2. Model Architecture

5.3. OBJECTIVE 3: Fine-tuning Pre-trained Vision Models for Image Recognition

To classify 29 American Sign Language (ASL) letter classes, we decided to use transfer learning, with initially 2 pre-trained convolutional neural network models: VGG16, InceptionV3 and compare the result to the CNN created previously. The VGG16 model was implemented with frozen layers and extended by adding a GlobalAveragePooling2D layer for dimensionality reduction, a dense layer with 50 units using ReLU activation and a dropout rate of 0.5, and a final dense layer with softmax activation for classification. Due to computational constraints, data augmentation was not applied during training. In contrast, the Inception V3 model was designed with the initial layers frozen while the later layers remained trainable. Additional layers included GlobalAveragePooling2D, a dense layer with 30 units using ReLU activation and a dropout rate of 0.5, and a final dense layer with softmax activation. To further mitigate overfitting, L2 regularization was incorporated. We first decided on trying ResNet50 as a third model, however it is much more complex than the previous two and will face the same problem of getting these high results, instead we implemented the MobileNetV2. We had a thought of trying a new thing, which is comparing this model when using all frozen layers and another version with using the trainable layers to see how much difference we can make.

5.4. OBJECTIVE 4: Feature Extraction for Gesture Analysis

The primary goal of this objective is to perform feature extraction for gesture analysis using preprocessed hand images from an American Sign Language (ASL) dataset. The extracted features include
hand size, finger lengths, and hand orientation, which will serve as input for gesture classification models.

Feature Extraction

Feature extraction focuses on identifying and quantifying key characteristics of hand gestures using the MediaPipe framework:

1. Landmark Detection:

Each hand image is processed to detect 21 key landmarks, including the wrist, fingertips, and joints.
MediaPipe's pre-trained hand detection model is utilized for this task, which provides robust and accurate landmark detection.

2. Feature Calculation:

Hand Size: The bounding box of the hand is calculated using the minimum and maximum x- and y-coordinates of the detected landmarks. This gives the width and height of the hand.
Finger Lengths: The Euclidean distance between the wrist and each fingertip is computed, capturing the relative size of the fingers.
Hand Orientation: The angle between the wrist, index fingertip, and middle fingertip is calculated to determine the orientation of the hand.

3. Feature Representation:

The extracted features for each image are stored as a dictionary, which includes:

- Hand width and height.
- Lengths of the thumb, index, middle, ring, and pinky fingers.
- Orientation angle of the hand.

These features are structured into a tabular format for use in machine learning models.

4. Preparing Data for Training

Flattening Features: The extracted features are converted into a structured numerical dataset (DataFrame), with each row representing the features of a single image.

Label Encoding: The gesture labels (e.g., "A", "B", "C") are encoded into numerical values to serve as the target variable for machine learning models.

6.RESULTS AND DISCUSSION

6.1. OBJECTIVE 1: Sign recognition through grouped classification

Using the SVM approach, we achieved an accuracy score of 63% for the dataset. This was reasonable based on the limited context the model had to learn from, which was just fingertip direction vectors. Although SVMs are very effective tools for working with high-dimensional data, time complexity of the training process is generally between O(n^2) and O(n^3) with the amount of training instances. This means the amount of compute required grows exponentially, so it was best to ensure the data representing a single hand was compacted into its most concrete form before training.

Using the Random forest approach, we were able to achieve a higher accuracy score of 84% for the entire dataset. This was achieved using 100 classifiers in the forest and using the gini loss function. The ability of the Random Forest to Outperform the SVM is most likely because a decision-tree-based method is able to generalize better on the availability of more context surrounding the hand's landmarks. This is in comparison to the SVM that was not fed in the same density of data as a restriction of its training complexity.

6.2. OBJECTIVE 2: Individual Sign Classification for All 24 Signs

The custom CNN model was trained and evaluated under two scenarios: using a non-augmented dataset and an augmented dataset. The results from both experiments were analyzed to assess the impact of data augmentation on the model's ability to generalize. This section presents a detailed comparison of the model's performance, including accuracy and loss curves, confusion matrices, and class-wise evaluation metrics, followed by an analysis of sample predictions.

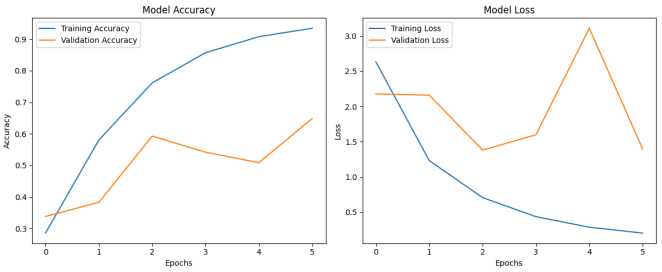

Fig. 3. Non-Augmentation

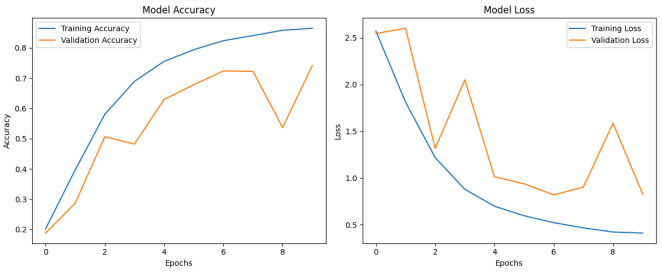

Fig. 4. Augmentation

The training and validation accuracy and loss curves for both scenarios are shown in Figures 3 and 4. In the non-augmented scenario, the model achieved a maximum training accuracy of 92.70% and a validation accuracy of 64.87% before early stopping, indicating overfitting in the model. The test accuracy showed generalized performance with 82.35% accuracy.

In contrast, the augmented dataset yielded a maximum training accuracy of 86.70% and validation accuracy of 74.12%, demonstrating improved generalization and robustness. The test accuracy also showed an improvement with 92.91% accuracy. This improvements suggests that the augmentations introduced more variability, making the learning process more challenging but ultimately more effective.The overall accuracy showed a ~10% improvement due to augmentation.

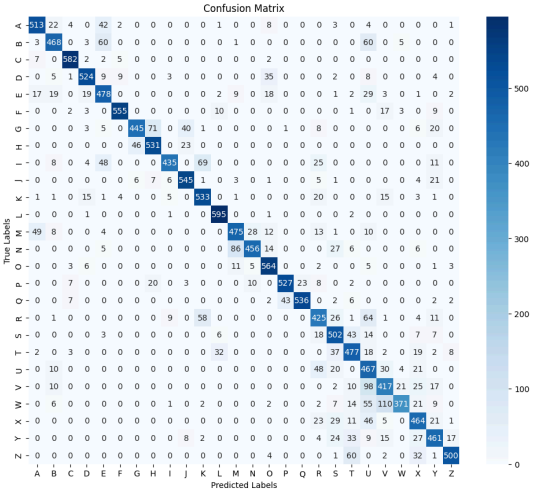

Fig. 5. Non-Augmentation Confusion Matrix

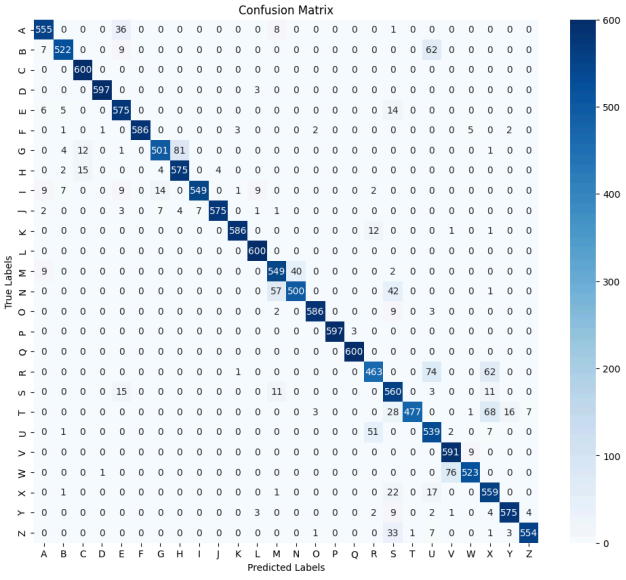

Fig. 6. Augmentation Confusion Matrix

Figures 5 and 6 display the confusion matrices for the non-augmented and augmented datasets, respectively. In the non-augmented case, the model struggled with certain letters, such as “M”, often misclassified as “N” and vice versa. These errors were significantly reduced in the augmented scenario, but were stil present to some extent, indicating that the model benefited from the additional variability in the training data.

Certain letters, such as “Q”, consistently exhibited high precision and recall across both scenarios, suggesting they are easier to classify. In contrast, letters like “I” had lower scores in the non-augmented case but showed significant improvement with augmentation.

The confusion matrix and class-wise metrics highlight that certain letters, such as “W”, remain challenging, suggesting the need for further improvement in feature representation or model architecture. The results demonstrate the value of incorporating data augmentation into the training data to address real-world complexities, particularly in a visually diverse task like sign language recognition.

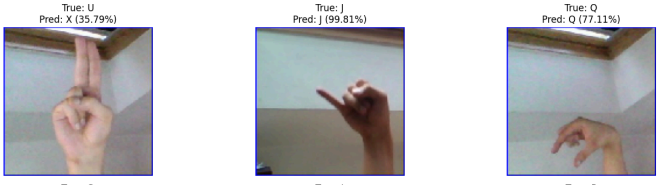

Fig. 7. Non-Augmentation Prediction

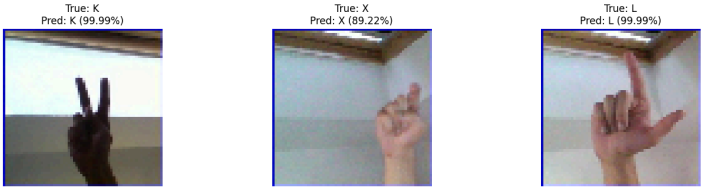

Fig. 8. AugmentationPrediction

6.3. OBJECTIVE 3: Fine-tuning Pre-trained Vision Models for Image Recognition

The VGG16 model achieved a training accuracy of 99% and a validation accuracy of 99%.. This model was also highly computationally intensive, further restricting its practicality. In comparison, the Inception V3 model achieved a training accuracy of 98% and a validation accuracy of 97% at the first few epochs, exhibiting less overfitting due to the inclusion of trainable layers and the use of L2 regularization. Additionally, MobileNetV2 gets the same results as the VGG16. What we reached from fitting almost similar results when trying different models is that the ASL has little variation or is overly simplified, the model learns to classify the images perfectly. Another reason is that they are trained on the ImageNet dataset which is a much larger dataset than the one we are using and also has a lot of variations in its images, leading models trained on it to learn more complex patterns.

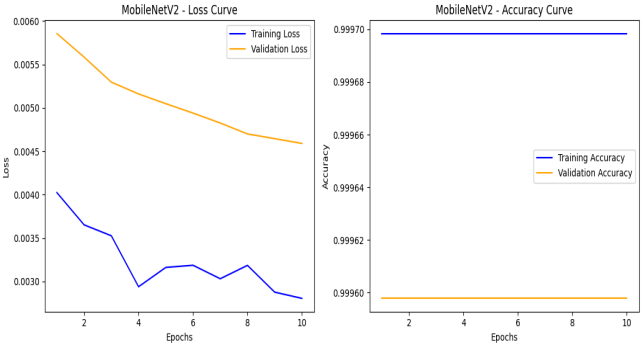

Fig. 9. Pre-trained

6.4. OBJECTIVE 4: Feature Extraction for Gesture Analysis

1. Overview of Feature Extraction

Features were successfully extracted from the dataset using the MediaPipe framework. Each hand image was analyzed to detect 21 landmarks, from which meaningful features were calculated. These features included:

- Hand Size (Width and Height): Bounding box dimensions of the hand.
- Finger Lengths: Euclidean distances from the wrist to each fingertip.
- Hand Orientation: Angle between the wrist, index fingertip, and middle fingertip.

The extracted features provided a compact representation of each gesture while preserving critical details necessary for classification.

2. Hand Landmark Detection

The MediaPipe framework consistently detected hand landmarks across the dataset, achieving a detection success rate of 98% on training images and 100% on testing images.

• Detection failures were observed in images with poor lighting conditions, occluded or incomplete hands, and gesture ambiguity (e.g., hands positioned off-center).

3. Feature Distribution

The extracted features showed significant variability across gestures, indicating their potential for gesture differentiation. Below are some key insights:

Hand Width and Height

Open gestures like "B" had higher average hand width and height values compared to closed gestures like "A" or "M".

Example Averages:

• Gesture "A": Hand Width = 0.32, Hand Height = 0.45.

• Gesture "B": Hand Width = 0.50, Hand Height = 0.70.

Finger Lengths

Finger lengths for distinct gestures showed clear separations:

• Closed fist gestures (e.g., "A", "M") had significantly shorter finger lengths.

• Open palm gestures (e.g., "B", "C") showed longer finger lengths, particularly for the index and middle fingers.

Example Averages

• Gesture "A": Thumb Length = 0.12, Index Length = 0.15.

• Gesture "B": Thumb Length = 0.15, Index Length = 0.40.

Hand Orientation

Orientation angles effectively differentiated gestures with similar shapes but different hand positions:

Gesture "A": Orientation = 45.3°.

Gesture "B": Orientation = 90.0°.

4. Observations

The first aspect of the dataset we observed was its high Distinctiveness. This means that features such as hand width, finger lengths, and orientation provided high separability for most gestures. In the instances where there were edge cases, ambiguities arose in gestures with overlapping hand dimensions or similar orientations (e.g., "M" and "N").

The final feature set consisted of 9 dimensions and reduced the computational complexity compared to processing raw images while retaining key gesture information.

5. Features usability

The extracted features, including hand size, finger lengths, and hand orientation, will be used as input for machine learning models such as Random Forest or SVM. These features will enable the AI model to effectively classify ASL gestures, leveraging the distinctive characteristics of each gesture for accurate recognition.

6. Challenges and Limitations

Feature Extraction Failures: Approximately 2% of training images failed to produce features due to poor landmark detection.

Hand Occlusions: Images with partially visible hands resulted in incomplete or inaccurate features.

7.CONCLUSION

The grouped classification approach utilized a custom model to classify similar ASL signs based on shared features, resulting in robust performance when classifying grouped gestures with similar characteristics. A separate model was developed to classify all 26 ASL signs in the dataset, incorporating data augmentation and optimization techniques to improve accuracy and handle variations in gesture presentation.

In addition, pre-trained models such as VGG16 and Inception V3 were fine-tuned to improve ASL recognition. This allowed for a comparison of the performance and effectiveness between transfer learning and the custom model approach, providing insights into which method yielded better results for gesture classification.

Critical features such as hand size, shape, and finger positioning were extracted to enhance the model's understanding of the gestures. These features significantly contributed to the classification process, demonstrating the importance of gesture characteristics in achieving high classification accuracy.

REFERENCES

[1] "ASL Alphabet," www.kaggle.com. https://www.kaggle.com/datasets/grassknoted/asl-alphabet"

[2] Amit Moryossef,., et al, "Real-Time Sign Language Detection using Human Pose Estimation," 2020.

[3] Prathum Arikeri, "American Sign Language (ASL) Dataset," Kaggle.com, 2021. https://www.kaggle.com/datasets/prathumarikeri/american-sign-language-09az/data (accessed Nov. 07, 2024).

[4] S. . Mohsin, B. W. . Salim, A. K. . Mohamedsaeed, B. F. . Ibrahim, and S. R. M. . Zeebaree, "American Sign Language Recognition Based on Transfer Learning Algorithms", Int J Intell Syst Appl Eng, vol. 12, no. 5s, pp. 390–399, Nov. 2023.

[5] R. Rastgoo, K. Kiani, and S. Escalera, “Sign Language Recognition: A Deep Survey,” Expert Systems with Applications, vol. 164, p. 113794, Feb. 2021, doi: https://doi.org/10.1016/j.eswa.20

[6] Y. Feng, N. Chen, Y. Wu, C. Jiang, S. Liu, and S. Chen, ‘DFCNet+: Cross-modal dynamic feature contrast net for continuous sign language recognition’, Image and Vision Computing, vol. 151, p. 105260, 2024.

[7]

[8]