CV for American Sign Language - Interpretation of Sign Language

Members: Yu-Hsin Wu (USC ID: 3434391900), Yu-Hung Kung (USC ID: 3431428440)

1. Abstract:

American Sign Language (ASL) is a visual language primarily used by deaf individuals in North America, conveying meaning through gestures and facial expressions. However, most people have not fully learned this language, creating significant barriers in communication between sign language users and those unfamiliar with it. This gap hinders effective interaction and mutual understanding.

With advancements in machine learning and image recognition technologies, these communication barriers can be addressed. By recognizing ASL gestures, sign language can be translated into text, enabling easier comprehension for non-signers and bridging the language gap between the two groups.

2. Introduction

ASL is a visual language primarily used by the deaf community in North America. Since most people have not fully learned ASL, there is a huge communication barrier between sign language users and the general public. This language gap impedes effective interaction and understanding. To bridge this gap, we leverage machine learning to develop a model that can accurately recognize and interpret ASL gestures. By translating sign language into text, we aim to promote communication between sign language users and non-sign language users, creating a more inclusive society.

Our primary objective is to create a robust and efficient machine-learning model that can accurately recognize ASL signs. To achieve this, we will conduct a comprehensive analysis of various preprocessing techniques, model architectures, and hyperparameter tuning strategies. First, we recognize the importance of training data diversity. We hope that the data used for training should include real-world practical situations, including variations in lighting conditions, hand position, rotation, and handedness. If the dataset lacks sufficient diversity, we will use pre-processing techniques to supplement the training data. Second, we will explore a range of pre-trained models, including ResNet18, ResNet34, ResNet50, and MobileNetV2. By evaluating the accuracy and efficiency of each model, we aim to identify the most suitable architecture for ASL gesture recognition. Furthermore, we will find the most suitable hyperparameters for the model. Our goal is to find the best-performing model that balances accuracy and efficiency.

This initial research serves as a foundation for future work. Our final goal is to create a system that can accurately and simultaneously convert ASL gestures into written language, allowing ASL users to effortlessly communicate with people who are unfamiliar with ASL.

3. Dataset and Features

In this project, we are going to train our model based on the ASL Alphabet dataset ("ASL Alphabet"). There are 29 classes in total, of which 26 are alphabets A to Z, 'delete', 'space', and 'nothing'. This dataset contains 87000 images (3000 images for each class) for training, which are 200 x 200.

Figure 1. The image of each class. From top left to bottom right, they correspond to the alphabet A to Z, 'del', 'nothing', and 'space'.

Our inspection of the asl_alphabet dataset revealed that all training images are based on right-handed signers. However, this doesn't reflect real-world scenarios accurately since approximately 10% of the people are left handers. Likewise, there are also gestures for left handed sign language users. Considering this significant proportion, it's crucial to incorporate left-handed sign representations in our dataset and model training process. This approach will enhance our ASL alphabet recognition system's robustness and real-world applicability, ensuring accurate performance for both right-handed and left-handed users.

Additionally, we observed that the asl_alphabet dataset includes images with varying brightness, angles, and scales. This diversity reduces the need for extensive data preprocessing. However, the primary limitation of the dataset remains its lack of representation for left-handed signers. We will present our proposed solution to address this issue in the following section.

4. Methods

A. Problem Statement:

Left and Right handed signing expression:

According to [4] Handedness prevalence in the deaf: Meta-Analysis, meta-analyses on handedness prevalence in the deaf population show that approximately 10% to 20% of sign language users are left-handed. Also, the asl_alphabet dataset contains only right-handed sign gesture images. In real world scenarios, we have to apply data preprocessing to elevate the robustness of the dataset and model generalization.

To recognize ASL and translate sign language into text, enabling more people to understand ASL, we trained models on the ASL Alphabet dataset. However, we observed that all the images in the dataset were captured using right-hand gestures, overlooking the possibility of left-hand usage in real-world scenarios. To address this limitation, we applied `RandomHorizontalFlip()` to generate mirrored images representing left-hand gestures.

By incorporating these flipped images during training, we aimed to enhance the model's ability to learn scenarios closer to real-world usage. Subsequently, we trained the model with these mirrored images and evaluated whether the inclusion of left-hand gestures improved the model's accuracy in recognition of left-hand gestures.

Real-Time Application Challenges in ASL Recognition:

In today's society, the majority of people lack understanding or familiarity with ASL, creating a significant communication barrier for sign language users when interacting with the general public. However, advancements in machine learning offer a potential solution to address this issue.

Through our exploration of ASL gesture recognition applications, we observed that existing models demonstrate a certain level of recognition accuracy. Nevertheless, real-world applications require careful consideration of computational efficiency, as the time required for processing must be minimized. Real-time responses are crucial to aligning with practical use cases.

Therefore, we aim to investigate the computational time of various models, striving to identify one that achieves both low latency and high accuracy, making it more suitable for real-world ASL recognition applications.

B. Preprocessing Steps:

To ensure compatibility with the ResNet architecture that we plan to use for model training, the images will need to be resized from their original 200 x 200 dimensions to 224 x 224. This step is crucial as ResNet is optimized for input images of size 224 x 224, and resizing ensures consistency across all training data. In addition to resizing, we applied the following preprocessing techniques:

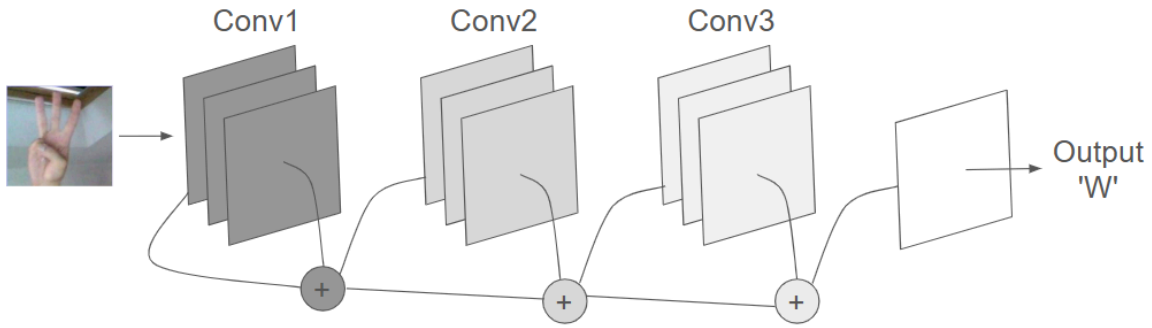

Figure 2. The ResNet architecture.

1. Normalization: Scale pixel values to a range between 0 and 1 or standardize to zero mean and unit variance, which helps accelerate convergence during training.

2. Horizontal Flip: In order to tackle with handedness, we applied the RandomHorizontalFlip() to flip the image horizontaly with probability 0.5. This approach will transform a proportion of the original right-handed sign image into the left-handed sign image. Figure 3 shows the comparison of left handed and right handed gestures.

3. Random Rotation: Although the asl_alphabet dataset contains sign gestures in different angles, we decided to apply additional rotation to enhance the effect. By applying RandomRotation(10), the image will be randomly rotated between [-10, 10] degrees.

4. Grayscale Conversion: Although the images are in RGB format, we may explore grayscale conversion to reduce computational complexity if color information is deemed non-critical.

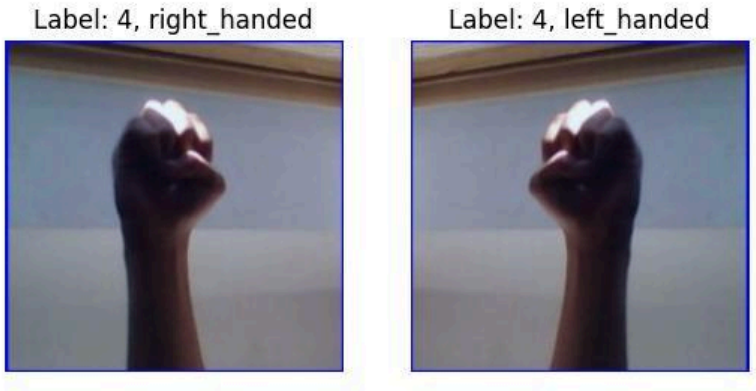

Figure 3. On the left is the original image from asl_alphabet dataset, by applying horizontal flip, we can get the same class of gestures expressed in the left hand.

C. Model Architecture

In this study, we explored two different deep learning models: Residual Neural Network (ResNet) and MobileNet. The ResNet architecture is built on traditional neural networks with the addition of residual learning parameters when connecting different layers. This design allows the network to learn the relationship between the input and output within sub-networks. On the other hand, MobileNet is specifically designed for mobile devices, offering low latency compared to traditional machine learning models. It incorporates depthwise separable convolutions, which effectively reduce the number of parameters and computational load while maintaining excellent prediction performance.

We utilized pre-trained models in PyTorch, including ResNet18, ResNet34, ResNet50, and MobileNetV2. These models were initialized with pre-trained weights to use feature extraction capabilities derived from large-scale datasets. To fine-tune these architectures for our specific task, the parameters were frozen during the initial training phase to preserve general feature representations. After 10 epochs, the parameters were unfrozen to allow the models to learn task-specific features. During this phase, the learning rate was adjusted to 1e-5 in order to achieve stable and effective fine-tuning. This ensured that the model weights adapted appropriately without overfitting or deviating significantly from the pre-trained values.

5. Results

A. Training Loss and Training Accuracy

We separated the dataset into three parts: training, validation and testing with proportion 70%, 15%, 15% respectively. Figure 4 shows the training loss and

accuracy curves during training. We can observe that the model has a significant performance elevation at epoch 10 since we unfreeze the parameters and lower the learning rate to 1e-5. All the models have an excellent performance on the test set, nearly 100% correctness.

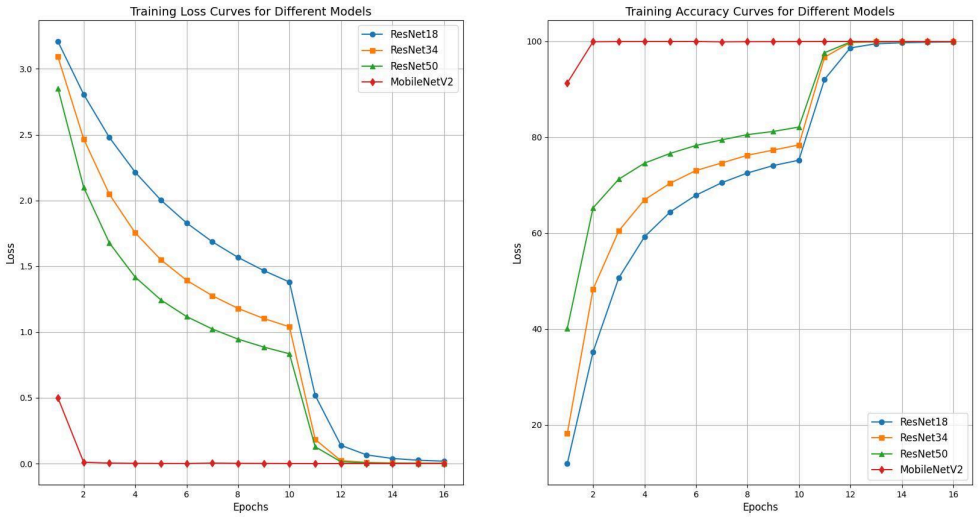

Figure 4. Training loss and accuracy of 4 different model architectures. We froze the pre-trained weights in the first 10 epochs. After epoch 10, we unfreezed the weights and lowered the learning rate to 1e-5.

B. Model Generalization

In order to analyze the model generalization, we trained models with three different train datasets: raw dataset, one containing both right-handed and left-handed images, and one with grayscale images with both handedness and rotation. Firstly, we trained models with raw datasets. Table 1 shows the accuracy of each model to a different test set.

Model	right hand only	left hand only	mixture(90/10)
ResNet 18	99.99%	83.48%	98.30%
ResNet 34	100%	86.32%	98.76%
ResNet 50	99.97%	95.70%	99.52%
MobileNetV2	99.93%	93.89%	99.36%

Table 1: Test accuracy of each model trained with a dataset without left-handed images evaluated on three different test datasets: one containing only right-handed images, one containing only left-handed images, and a mixed dataset consisting of 90% right-handed and 10% left-handed images.

From Table 1, we can see that there are differences in accuracy between right-handed and left-handed test sets since models are trained with right-handed only training sets. We also have another testing set, which is the mixture set with 90% right-handed and 10% left-handed images. This testing set simulates the distribution of the real-world sign language population. From the consequences, we can conclude that we need to add additional left-handed images into the training set in order to tackle this issue.

By applying horizontal flip and rotation to the raw training set, we trained four different models based on the processed training set. Table 2 shows the outcomes of each model to different testing sets. From the consequences, we can imply that horizontal flip benefits the model for classifying left-handed gestures. For ResNet18, the accuracy of left-handed images increased from 83.48% to 99.92%, which increased 19.69% of the model performance.

Model	right hand only	left hand only	mixture(90/10)
ResNet 18	99.88%	99.92%	99.92%
ResNet 34	100%	99.98%	100%
ResNet 50	100%	99.98%	100%
MobileNetV2	99.86%	99.85%	99.85%

Table 2: Test accuracy of each model trained with a dataset with left-handed images evaluated on three different test datasets: one containing only right-handed images, one containing only left-handed images, and a mixed dataset consisting of 90% right-handed and 10% left-handed images.

We mentioned RGB to Grayscale preprocessing in the pre-processing step. Since we already concluded that horizontal flip and rotation bring advantages for training, we apply the grayscale transform to evaluate the performance. Table 3 shows the results of ResNet18 trained with grayscale images, evaluated with three different testing sets.

Model	right hand only	left hand only	mixture(90/10)
ResNet 18	99.95%	99.96%	99.95%

Table 3: Test accuracy of ResNet 18 model trained with grayscale images evaluated on three different test datasets: one containing only right-handed images, one containing only left-handed images, and a mixed dataset consisting of 90% right-handed and 10% left-handed images.

From the results of table 3, compared with table 2, we can conclude that grayscale transform can benefit models only on a tiny scale.

C. Confusion Matrix

Although all four models achieve an impressive accuracy exceeding 99%, a closer examination of their confusion matrices reveals subtle differences in their classification performance. Figure 5 presents the confusion matrices for the ResNet18, ResNet34, ResNet50, and MobileNetV2 models. These visualizations offer valuable insights into the specific classes where each model occasionally falters.

Upon careful analysis of the confusion matrices, we observe that the models demonstrate near-flawless performance. Remarkably, each model makes fewer than five misclassifications out of the entire test set of 13,050 images. This exceptional accuracy underscores the robustness of these deep learning architectures in handling the given classification task.

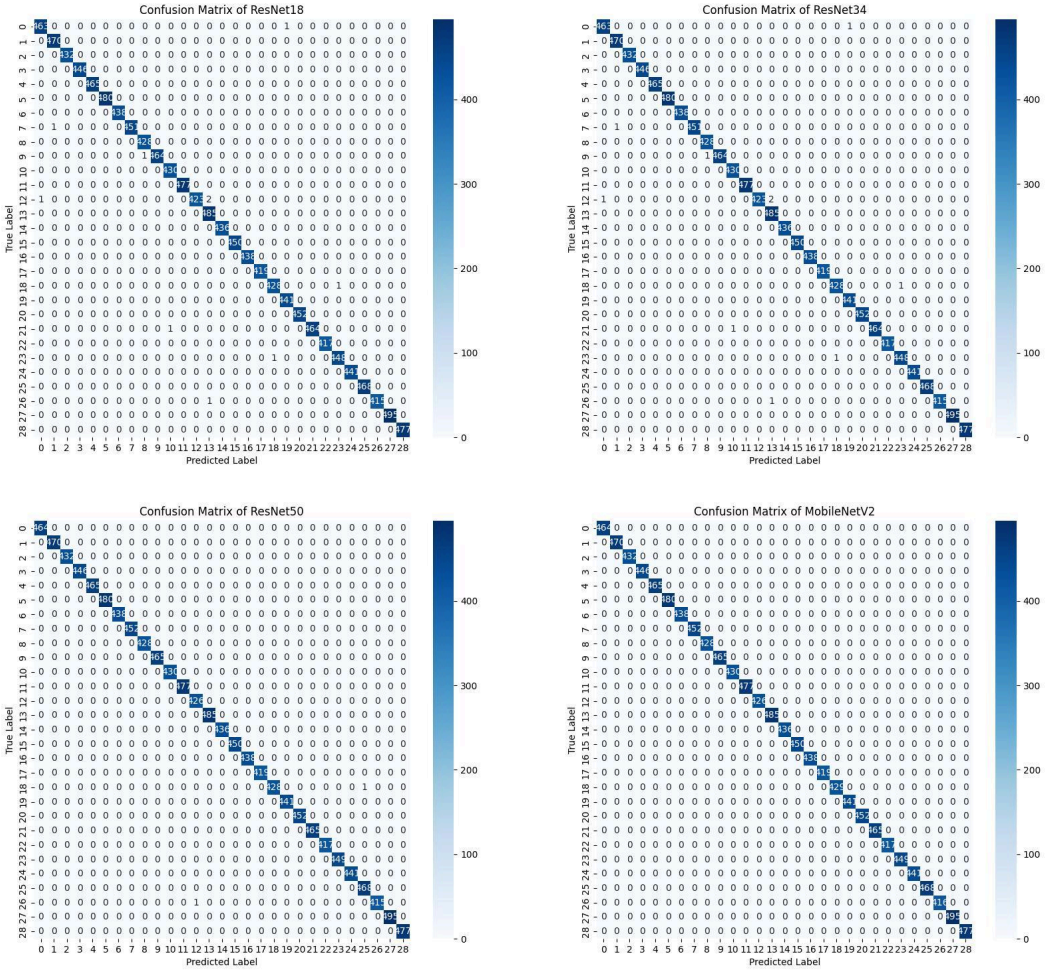

Figure 5. Confusion matrix of ResNet18, ResNet34, ResNet50, and MobileNetV2.

D. Model execution time

Because our goal is to enable the ASL recognition model to produce results immediately after receiving input images, model execution time becomes critically important. The tables below summarize the execution times of different models. On datasets with generalized preprocessing, ResNet18 demonstrates significantly shorter execution times compared to ResNet34, ResNet50, and MobileNet. These results highlight the efficiency and suitability of ResNet18 for real-time applications.

Dataset with horizontal flip and random rotation:

Model	ResNet 18	ResNet 34	ResNet 50	MobileNet
Time	423.69 ms	691.03 ms	1556.85 ms	671.41 ms

Table 4: Execution time of different models trained on a dataset containing images of both handedness types, evaluated on a mixed test dataset consisting of 90% right-handed and 10% left-handed images.

Dataset with grayscale, horizontal flip, and random rotation:

Model	right hand only	left hand only	mixture(90/10)
ResNet 18	693.07 ms	695.15 ms	692.54 ms

Table 5: Execution time of ResNet18 model trained on a dataset containing grayscale images of both handedness types, evaluated on three different datasets: one containing only right-handed images, one containing only left-handed images, and a mixed dataset consisting of 90% right-handed and 10% left-handed images.

E. Summary

We investigated the impact of different model architectures and dataset preprocessing methods on the performance of gesture recognition and analyzed the generalization ability and execution efficiency of the models. During the training process, pre-trained weights were frozen initially, and then unfrozen at the 10th epoch, which significantly improved model performance, achieving nearly 100% accuracy on the test set.

To align the model with real-world applications, we further enhanced its generalization ability. During training, in addition to using the original dataset, we incorporated datasets containing left-handed images and applied grayscale processing and rotation. The results demonstrated that including more comprehensive datasets

effectively improved the accuracy of left-handed image recognition. We also attempted to enhance the model by incorporating grayscale processing. However, we found that this method had a limited impact on improving the model's performance, possibly because our model already achieved a high level of accuracy. In terms of execution efficiency, we evaluated the performance of models with excellent generalization. The results showed that ResNet18 achieved not only high performance but also shorter execution times. Therefore, ResNet18 is well-suited as a real-time ASL recognition model.

In conclusion, the experimental results indicate that incorporating data augmentation methods such as horizontal flipping and rotation effectively enhances the model's generalization ability. Among these models tested, ResNet18 stands out for its high performance and shorter execution time, making it an excellent choice for real-time ASL recognition.

6. Discussion

Based on the above results, our current research proves that adding data such as horizontal flipping and rotation can effectively enhance the generalization ability of the ASL gesture recognition model. At the same time, we determined that ResNet18 can recognize test images in the shortest time and is very suitable as a real-time recognition model. While the current test setup handles static input images, our goal is to extend this project to real-time applications. We expect that future systems will be able to capture gestures via a real-time camera, recognize them immediately, and display the results on the screen. This application allows people who do not understand ASL to quickly understand the meaning of ASL gestures and reduce communication barriers between the two parties.

7. References

[1] Real-time Object Detection and Classification for ASL alphabet
[2] Sign Language Recognition Based on Computer Vision
[3] ASL Alphabet dataset
[4] Handedness prevalence in the deaf: Meta-Analysis