AMERICAN SIGN LANGUAGE IMAGE RECOGNITION USING NEURAL NETWORKS

Destin Saba, Nathan de Oliveira, Cole Cathcart, Rana El Sadig

https://github.com/destinsaba/ASL_Alphabet_Classifier

University of Calgary

ABSTRACT

Sign language is a vital communication tool for the Deaf and Hard of Hearing community. However, communication barriers persist between sign language users and non-signers. Convolutional neural networks, widely used for image recognition, offer a promising solution for automated sign language recognition to bridge this gap. This project presents a neural network-based approach for classifying static American Sign Language (ASL) alphabet gestures. A pre-trained ResNet-50 model with transfer learning was used to classify images from a publicly available ASL dataset, and was able to achieve 99.47% accuracy on an unseen test dataset. Additionally, a real-time webcam interface was developed to demonstrate the model's live inference capabilities. Results demonstrate deep learning's strong potential to enhance accessibility and communication for the DHH community.

1. INTRODUCTION

Sign language is a crucial mode of communication within the Deaf and Hard of Hearing (DHH) community, used by approximately 70 million people worldwide [14]. Despite its significance, a considerable communication barrier persists between sign language users and those unfamiliar with it. This barrier impacts educational, social, and professional opportunities for DHH individuals [14].

Automated Sign Language Recognition (SLR) systems have the potential to bridge this gap by facilitating real-time translation of sign language gestures into readable text or audible speech, improving inclusivity and accessibility [5]. By providing accurate translations instantly, these systems can significantly enhance the quality of life and social integration of sign language users [5].

Advancements in deep learning, especially Convolutional Neural Networks (CNNs), have demonstrated substantial potential in complex image recognition tasks [2,5]. CNNs learn spatial features and recognize patterns in visual data, making them a strong candidate for gesture classification [6].

The primary goal of this project is to develop and validate a CNN-based model for ASL gesture classification. ResNet-50 is a deep CNN architecture with 50 layers that uses residual learning through skip connections to enable efficient training of deep networks [8]. In this project, transfer learning will be applied using the ResNet-50 architecture pre-trained on ImageNet to classify images of static American Sign Language (ASL) alphabet gestures. The ASL alphabet comprises 26 static gestures corresponding directly to the English alphabet letters (Fig. 1). By automating ASL alphabet recognition, the proposed system seeks to significantly enhance accessibility for the DHH community, bridging critical communication barriers.

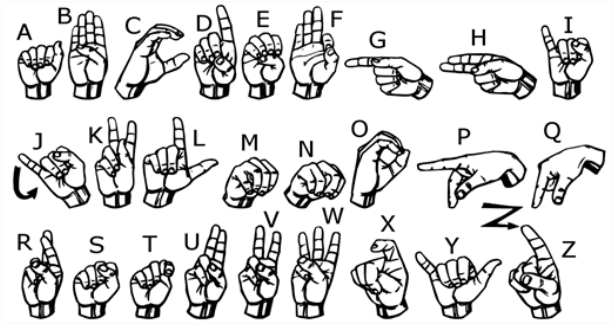

Fig. 1. 26 ASL gestures and their associated letters in the English alphabet [1].

A publicly available dataset (ASL Dataset) was used consisting of 87,000 images—covering 29 ASL gestures (the 26 letters plus *space*, *delete*, and *nothing*)—to fine-tune the model [3].

2. RELATED WORK

Research on automated Sign Language Recognition (SLR) has significantly expanded due to advancements in deep learning and increased computational power. Several distinct approaches have emerged, primarily categorized into sensor-based methods, which employ wearable devices with embedded sensors, and image-based methods, which rely solely on visual data captured by cameras [12].

Sensor-based systems typically utilize specialized gloves equipped with sensors that detect finger bending, hand position, and motion. Although these methods are often accurate and reliable due to precise motion tracking, they suffer from practical limitations such as high costs, discomfort, limited mobility, and reduced applicability in real-world scenarios [5, 9]. Therefore, image-based recognition techniques have become increasingly popular for their non-intrusive nature and accessibility, especially given the widespread availability of high-quality cameras on consumer devices [5].

Image-based recognition methods leverage powerful deep learning architectures, particularly Convolutional Neural Networks (CNNs), which excel in capturing spatial features from images. For example, AlexNet, one of the pioneering CNN architectures, has shown promising results in ASL recognition tasks. Cayamcela and Lim [10] employed AlexNet and GoogLeNet models trained on the ASL Dataset. The fine-tuned AlexNet model attained an accuracy of 99.39%, demonstrating the model's strong performance capabilities for ASL recognition.

Results from similar studies show variations based on preprocessing techniques and training methodologies. Alsharif et al. [5] conducted a comparative analysis using five deep learning models — AlexNet, ConvNeXt, EfficientNet, ResNet-50, and VisionTransformer. They reported exceptional accuracy with ResNet-50 (99.98%), followed closely by EfficientNet (99.95%), ConvNeXt (99.51%), and AlexNet (99.50%). In contrast, VisionTransformer achieved only 88.59%, primarily attributed to input data resizing requirements. These differences highlight the sensitivity of CNN-based models to variations in dataset preprocessing, model architecture, and hyperparameter tuning.

Overall, while CNN-based approaches consistently achieve high accuracy, there remains variability based on factors such as preprocessing and model-specific architectural nuances. ResNet architectures, especially ResNet-50, consistently demonstrate top performance due to their depth and ability to mitigate vanishing gradients through skip connections [8]. Future research may further explore hybrid approaches, data augmentation techniques, and optimization of transformer-based models to address existing limitations.

3. MATERIALS AND METHODS

This section describes the dataset, preprocessing techniques, the neural network architecture, and training approach used to develop an image classification model for recognizing static ASL alphabet hand gestures.

3.1. Dataset Preprocessing and Augmentation

The model was trained and validated using the public dataset known as the ASL Alphabet dataset accessible on Kaggle [3]. This dataset contains 87,000 images representing 29 distinct classes: the 26 letters of the American Sign Language alphabet, plus three additional gesture classes—space, delete, and nothing. Each class consists of 3000 images which are 200 x 200 pixels, captured in various lighting conditions and hand sizes, providing a robust enough dataset for transfer learning.

To develop the model, the dataset was divided into a development and test set with a ratio of 80%, and 20%, respectively. The development set was further split into training and validation sets, also with an 80/20 ratio. The dataset split is visualized in Figure 2. The dataset was deemed large enough to not warrant employing cross-validation.

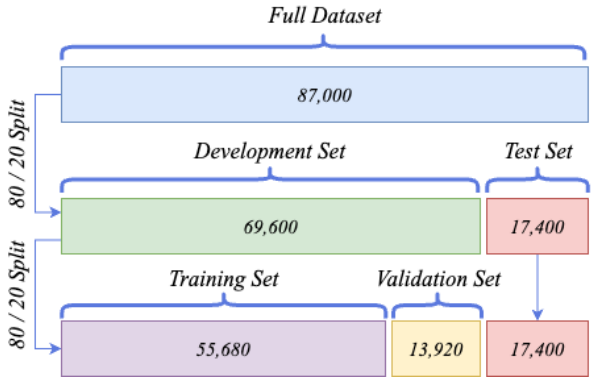

Fig. 2. Size of the training, validation and test datasets.

Since the ResNet-50 architecture was pre-trained on the ImageNet dataset, data preprocessing included image transformations consistent with ImageNet specifications. Images were resized to 224×224 pixels, converted to tensors, and normalized using ImageNet standard mean values [0.485, 0.456, 0.406] and standard deviations [0.229, 0.224, 0.225].

With the aim of improving model robustness for live inference, some data augmentation was applied to the training samples. First, random rotations up to 10 degrees were applied to help account for variations in live camera angles. Random cropping up to 80% of the original image size was also applied, which provides the model additional exposure to gestures that may be closer to the edge of the frame. Finally, the color jitter transform was used, which randomly adjusts image characteristics including brightness, contrast, saturation, and hue, with the aim of improving the model's ability to work with different camera settings. It is worth noting that no data augmentation that involved random flips, or large rotations (> 10 degrees) were applied,

since in some cases these transformations would alter the meaning of the gesture altogether, which would negatively impact model performance.

3.2. Transfer Learning with ResNet-50

Transfer learning was selected due to its effectiveness in image recognition tasks, especially with datasets of moderate size. Transfer learning leverages previously learned features from large-scale image datasets, enhancing training efficiency and accuracy when applied to specialized domains such as ASL recognition [10].

The ResNet-50 architecture was chosen for its proven capability on the dataset selected for this project. Its success is based on its use of skip connections, which enable the training of deeper networks without performance degradation due to a vanishing gradient [8].

The architecture of the selected neural network consists of two main parts, the pre-trained ResNet-50 model (trained on ImageNet dataset) and a fully connected layer customized for the specific task at hand with 29 output classes, as shown in Figure 3.

To train the model, the gradients of the convolutional layers in the ResNet-50 model were frozen. These layers extract features from the input images that will be leveraged by the fully connected layers to perform classification. To fine-tune the model to suit the ASL dataset, parameters in the last two fully connected layers were unfrozen. The two fully connected layers provided the model flexibility to learn and classify ASL gestures without discarding previously learned features.

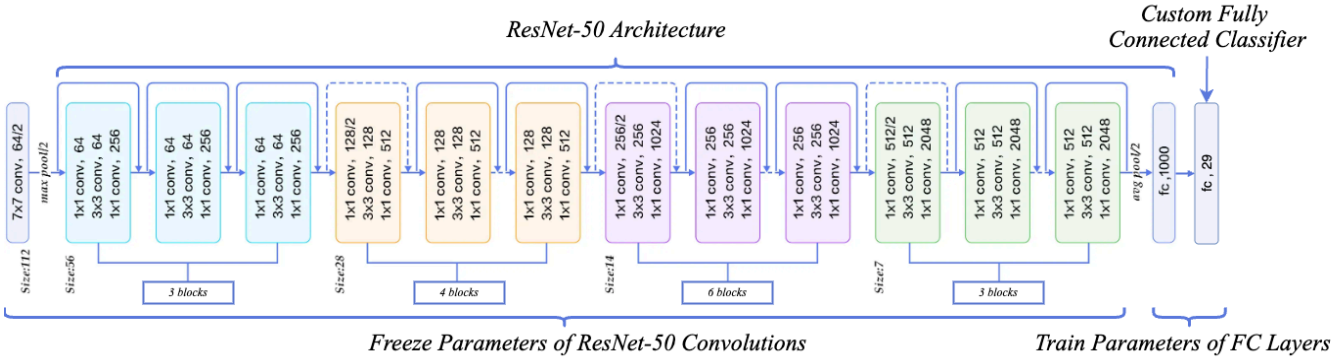

Fig. 3. Transfer learning model architecture, including the ResNet-50 architecture along with the fully connected classification layer (diagram modified from [4]).

3.3. Model Training and Hyperparameter Tuning

The model was trained using the University of Calgary's Advanced Research Computing (ARC) cluster. The ARC cluster imposes a limit of 1 GPU per job, which in the case of this project was a Tesla T4 GPU. The GPU, along with limits on CPU cores and memory, influenced the selection of certain hyperparameters in the interest of reducing model training time to enable more training iterations.

Model training was implemented in PyTorch, leveraging the torchvision library to access the pretrained ResNet-50 model and various dataset utilities. The model was trained using a structured approach where certain hyperparameters were kept constant, while others were adjusted between runs. The hyperparameters that were kept constant across training iterations included:

- Optimizer: The AdamW optimizer was selected due to its straightforward implementation, computational efficiency, and intuitive hyperparameters [7].
- Loss Function: Cross-entropy loss was selected as it is widely used and proven appropriate for multi-class classification problems.

The hyperparameters that were tuned across training iterations included:

- Scheduler: An exponential learning rate scheduler was added during training to improve model convergence and reduce overfitting [15]. With a final gamma value of 0.9, the scheduler multiplies the learning rate by 0.9 after each epoch.
- Initial Learning Rate: After multiple iterations, a learning rate of 0.001 was determined to provide an optimal balance between convergence speed and stability. Higher learning rates were found to cause the loss function to diverge, and lower learning rates required too many epochs to converge.

- Number of Epochs: The number of epochs was determined dynamically using an early stopping criterion. After 5 epochs without a reduction in the validation loss, training stopped. Early stopping helped ensure the model had learned what it could from the data without wasting resources.
- Batch Size: The selected batch size varied between 16 and 64, as supported by empirical studies on CNNs [11]. A batch size of 32 was found to achieve the highest validation accuracy for this model.

Finally, to manage the overall training process, hyperparameters and losses were continuously monitored through the Weights and Biases (WandB) platform. WandB enabled easier tracking of hyperparameters and performance changes across multiple training iterations.

4. RESULTS AND DISCUSSION

4.1. Model Training

The model reached its early stopping criterion after 32 epochs. Based on the training and validation loss curves, the model appears to successfully converge to a minimum in the loss function (Fig. 4). The smooth and consistent decline in both training and validation losses indicates stable model learning without significant overfitting or underfitting.

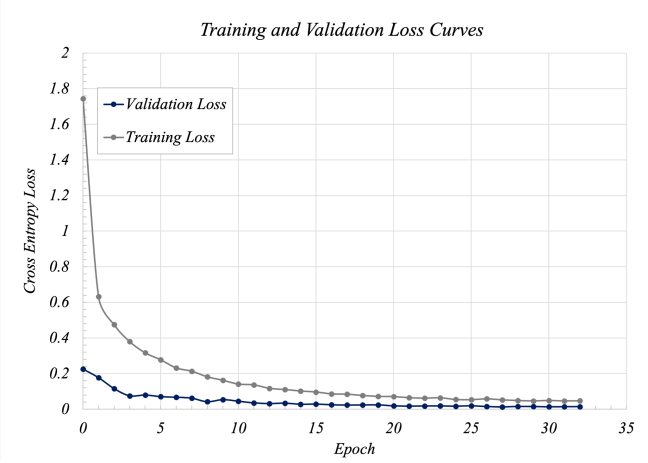

Fig. 4. Model training and validation loss curves plotted over the 32 epochs before the early stopping criterion was reached.

4.2. Model Performance

On the 17,428-sample test dataset, the model achieved 99.47% overall accuracy. As shown in the confusion matrix (Fig. 5), most gestures are classified with near-perfect precision and recall, reflecting the network's strong ability to distinguish between hand shapes.

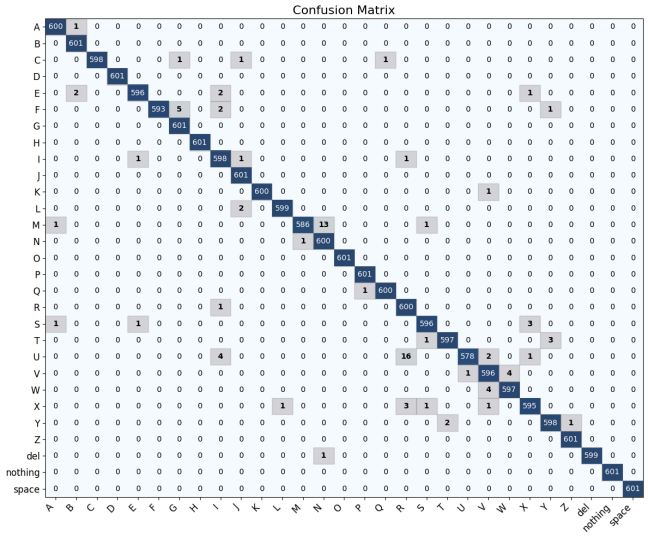

Fig. 5: Confusion matrix with true labels on the left border and predicted labels on the bottom border.

The few misclassifications occur primarily among letters whose gestures appear visually similar, such as M and N, or U and R, resulting in slightly lower precision or recall for these classes, as shown in Table 1. Visual examples of misclassifications are included in Figure 6. Nonetheless, each class maintains overall high performance. The macro average of precision, recall, and F1-score is 99.43%, proving the model's robustness across all classes.

Table 1. Precision and recall scores for the most challenging gestures to classify (M, N, R, U).

Class	Precision (%)	Recall (%)
M	99.66	97.67
N	98.03	99.50
R	96.47	100.00
U	99.65	96.01

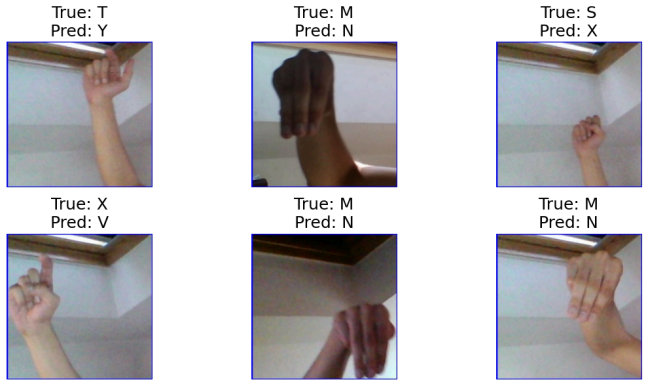

Fig. 6. Examples of misclassified samples, highlighting the model's relative difficulty differentiating 'M' from 'N'.

To verify that the network is making predictions based on gesture-related features, Grad-CAM heatmaps (Fig. 7) were generated. Grad-CAM uses the gradients flowing into the final convolutional layer to produce a localization map highlighting important regions in the image for making predictions [13]. The model's attention consistently centers on the hand region, suggesting it uses the gesture shape itself rather than background elements or other potential shortcuts.

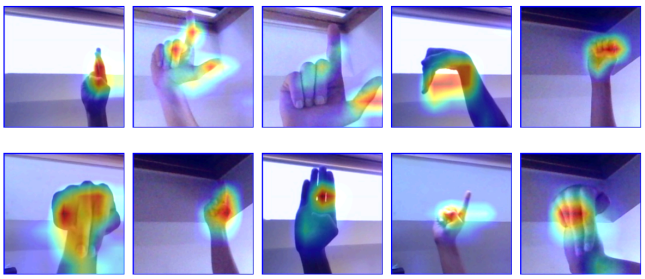

Fig. 7. Visualizing model attention with Grad-CAM heatmap, showing the model correctly attends to the hand gesture.

Finally, a confidence histogram (Fig. 8) indicates that correct predictions occur predominantly at higher confidence scores, reflecting the model's strong certainty in most cases. By contrast, misclassified samples cluster more heavily at lower confidence levels, highlighting the model's inherent uncertainty when faced with visually similar or ambiguous gestures.

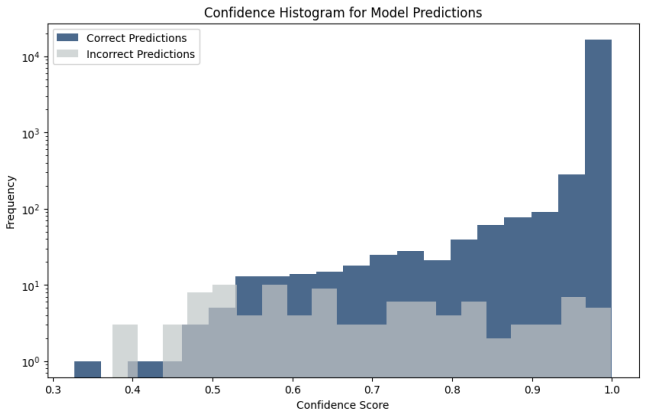

Fig. 8. Confidence histogram

4.3. Real-Time Inference Implementation

An objective of this project was to demonstrate the practical applicability of the trained model for real-time gesture recognition. To achieve this, a live webcam-based inference system was implemented using OpenCV for image capture and the best performing model for inference.

The inference pipeline included the following steps:

1. Capture real-time video frames from a webcam.
2. Preprocess each captured frame using the identical preprocessing steps applied during training (resize, normalization).
3. Perform real-time classification using the trained model.
4. Display the classification results directly on the video feed, providing instantaneous visual feedback.

The application was able to display predictions in real time with no latency. The speed of the real-time inference successfully illustrates the capability of the model to deliver real-time ASL alphabet classification, highlighting the potential for practical use in accessibility technologies for the DHH community. However, it is difficult to comment on the accuracy of the real-time predictions for two reasons:

1. The user must be fluent in ASL.
2. The model expects images with a relatively plain background.

The second point highlights a limitation with the trained model in real-world applications; gestures will rarely be viewed on a plain background. A future improvement to this solution could involve including a segmentation model that can isolate hand gestures from the background.

5. CONCLUSIONS

This project demonstrated the effectiveness of transfer learning with ResNet-50 for static ASL alphabet recognition, achieving 99.47% test accuracy. Real-time inference using a webcam interface confirmed the model's practical potential for accessibility applications.

Several metrics, such as high accuracy, fast inference, and efficient training, support the viability of this approach. However, the model occasionally misclassifies visually similar letters (e.g., *M* vs. *N*) and struggles with cluttered backgrounds. Moreover, it cannot recognize dynamic gestures, limiting its effectiveness in signing scenarios beyond fingerspelling.

Future enhancements could involve integrating hand-segmentation methods for more robust background handling, as well as exploring recurrent neural networks (RNNs) to learn from video footage of motion-based ASL signs. Taken together, these results reinforce the promise of deep learning in advancing accessible communication tools for the Deaf and Hard of Hearing community.

6. REFERENCES

[1] A. Akoum and N. Mawla, “Hand Gesture Recognition Approach for ASL Language Using Hand Extraction Algorithm,” J*ournal of Software Engineering and Applications*, vol. 8, no. 8, pp. 419–430, 2015, doi: 10.4236/jsea.2015.88041.

[2] A. Krizhevsky, I. Sutskever, and G.E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017, doi: 10.1145/3065386.

[3] A. Nagaraj, “ASL Alphabet,” *Kaggle*, 2018. [Online]. Available: https://doi.org/10.34740/KAGGLE/DSV/29550.

[4] A. Rastogi, “ResNet50,” *Dev Genius Blog*, Sep. 8, 2022. [Online]. Available: https://blog.devgenius.io/resnet50-6b42934db431.

[5] B. Alsharif *et al.*, “Deep Learning Technology to Recognize American Sign Language Alphabet,” *Sensors*, vol. 23, no. 18, p. 7970, Sep. 2023, doi: 10.3390/s23187970.

[6] D. Parashar *et al*., “A Deep Learning-Based Approach for Hand Sign Recognition Using CNN Architecture,” *Revue d’Intelligence Artificielle*, vol. 37, no. 4, pp. 937–943, 2023, doi:10.18280/ria.370414

[7] K. Diederik and J. Ba. “Adam: A Method for Stochastic Optimization,” 2014, https://doi.org/10.48550/arxiv.1412.6980.

[8] K. He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[9] M.A. Ahmed *et al.*, “A Review on Systems-Based Sensory Gloves for Sign Language Recognition,” *Sensors*, vol. 18, no. 7, p. 2208, 2018, doi: 10.3390/s18072208.

[10] M.E.M. Cayamcela and W. Lim, “Fine-tuning a Pre-trained Convolutional Neural Network Model to Translate American Sign Language in Real-time,” in *Proceedings of the 2019 International Conference on Computing, Networking and Communications* (ICNC), pp. 100–104, 2019, doi: 10.1109/ICNC.2019.8685613.

[11] R. Lin, "Analysis on the Selection of the Appropriate Batch Size in CNN Neural Network," in *International Conference on Machine Learning and Knowledge Engineering (MLKE)*, pp. 106-109, 2022, doi: 10.1109/MLKE55170.2022.00026.

[12] R. Rastgoo, K. Kiani, and S. Escalera, “Sign language recognition: A deep survey,” *Expert Systems with Applications*, vol. 164, p. 113794, 2021, doi: 10.1016/j.eswa.2020.113794.

[13] R. R. Selvaraju *et al*., “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–59, 2020, https://doi.org/10.1007/s11263-019-01228-7.

[14] World Federation of the Deaf (WFD), “Our Work,” Available online: https://wfdeaf.org/our-work (accessed on 8 March 2025).

[15] Z. Li and S. Arora. “An Exponential Learning Rate Schedule for Deep Learning,” 2019, https://doi.org/10.48550/arXiv.1910.07454

Repository containing the model training script, evaluation notebook, and live inference app: https://github.com/destinsaba/ASL_Alphabet_Classifier