REAL-TIME AMERICAN SIGN LANGUAGE RECOGNITION USING DEEP LEARNING WITH RESIDUAL CONVOLUTIONAL NEURAL NETWORKS

Alexey Kresin
Department of Computer Science Hood College

Abstract

This paper presents a real-time American Sign Language (ASL) recognition system using deep learning models trained on the ASL Alphabet dataset. A CNN was developed enhanced with residual blocks for improved feature extraction and classification accuracy.
Temporal smoothing was introduced in the real-time pipeline to enhance stability in gesture recognition. Our system achieves high accuracy in identifying most ASL alphabet gestures in live camera feeds. Challenges with specific letters (e.g., A, B, and X) are discussed along with possible future improvements.

I. Introduction

American Sign Language (ASL) is a primary mode of communication for millions of individuals within the Deaf and hard-of-hearing community across North America. It is a complete visual language that uses hand gestures, facial expressions, and body movements to convey meaning, making it fundamentally different from spoken languages. Despite its widespread use, communication barriers still exist between ASL users and those unfamiliar with sign language, often limiting access to essential services, education, and social interaction.

Recent advancements in computer vision and artificial intelligence have significantly improved the feasibility of translating ASL gestures into readable text or speech. In particular, the development of deep learning techniques, such as convolutional neural networks (CNNs), has enabled more accurate and scalable recognition of complex hand shapes and motions. These technologies open the door to real-time ASL recognition systems that can bridge communication gaps by serving as automated interpreters in various settings.

Real-time ASL recognition has valuable applications in **assistive technology**, where it can support live translation for accessibility; in **educational tools**, where it aids ASL learners through interactive feedback; and in **human-computer interaction**, enabling gesture-based input systems. However, achieving reliable real-time performance in uncontrolled environments remains a technical challenge due to variations in lighting, hand orientation, skin tone, and background clutter.
This paper presents the design and implementation of a real-time ASL recognition system that leverages a custom-trained CNN with residual connections and a temporal smoothing mechanism. The system processes live webcam input, predicts static hand gestures corresponding to ASL alphabet letters, and displays the recognized sign along with its confidence level. The workflow spans from dataset selection and preprocessing to model training, evaluation, and deployment in a real-time inference pipeline.

II. Related work

Previous research in ASL recognition primarily focuses on static image classification using CNNs. Some approaches leverage hand segmentation or depth cameras. This work uses only RGB input, a modified CNN with residual blocks, and confidence-based temporal smoothing to balance accuracy and real-time speed. We build upon datasets and methods introduced by works such as [1] and [2], refining their real-time application.

III. Main Methodology

A. Data Set Strategy and Model Adaptation

The project initially began with the construction of a custom American Sign Language (ASL) dataset by capturing gesture samples using a webcam. This approach allowed for control over environmental variables such as lighting, background, and gesture framing. However, it quickly became apparent that the limited size and variability of the custom dataset restricted model generalization.

The small number of samples per class and inconsistencies in hand positioning led to overfitting and poor performance during validation. To address these limitations, the strategy was revised to utilize a publicly available ASL dataset containing labeled images for all 29 static signs. This dataset offered significantly more diversity in hand shapes, orientations, and conditions, thereby improving the robustness of model training. Transitioning to the pretrained dataset resulted in improved accuracy and accelerated development in both offline evaluation and real-time testing.

After transitioning to a pretrained model approach, the ASL Alphabet dataset was selected to support training and evaluation. This dataset includes 29 distinct classes, corresponding to the English alphabet letters A through Z, as well as additional labels for "space," "nothing," and "del." Each class contains approximately 3,000 labeled RGB images, offering a sufficient volume of data for effective training of a deep learning model. The dataset provided consistent image resolution and was captured under controlled conditions, which helped minimize variations in lighting and background. For model development, the dataset was partitioned into two subsets: 80% of the data was allocated for training, while the remaining 20% was used for validation. This division allowed for performance monitoring and overfitting prevention during the training process. The rich class distribution and image diversity within the dataset contributed significantly to improving classification accuracy and model generalization.

B. Preprocessing

Prior to training and deployment, all input images underwent a standardized preprocessing pipeline to ensure consistency and compatibility with the model architecture. Each image was resized to a fixed resolution of 200×200 pixels to reduce computational complexity while preserving essential spatial features required for gesture recognition. Following resizing, pixel values were normalized to the range [0, 1] by dividing by 255, which aids in faster convergence during training and improves numerical stability in the neural network.

Additionally, two specific classes—**'space'** and **'del'**—were excluded from the real-time prediction pipeline. Although these labels were present in the training dataset, they were consistently misclassified during testing due to their ambiguous hand gestures and high visual similarity to other signs. These inconsistencies were further amplified under variable real-world conditions such as lighting changes, background clutter, and hand orientation differences.

To improve prediction reliability in live settings, the decision was made to remove these two classes from the active inference process while retaining them in the training dataset to preserve class structure and data balance.
This preprocessing strategy contributed to a more stable and accurate real-time classification pipeline, especially when paired with confidence filtering and temporal smoothing techniques.

C. Model Architecture

The model architecture was designed using a Convolutional Neural Network (CNN) backbone tailored for static hand gesture classification. The input layer accepts RGB images of size 200×200×3, corresponding to the preprocessed ASL hand sign frames.

The architecture begins with three sequential convolutional blocks, each consisting of a convolutional layer (Conv2D), followed by max pooling and batch normalization. These layers extract low- and mid-level spatial features, reduce dimensionality, and stabilize training by normalizing feature distributions across mini-batches.
To further enhance the model's ability to learn complex patterns, **two residual blocks** were incorporated after the initial convolutional layers. Each residual block consists of two convolutional layers with a **skip connection** that bypasses one or more layers and directly adds the input to the output. This architectural concept, originally introduced in ResNet, addresses the **vanishing gradient problem** and allows the network to learn identity mappings more effectively.

By enabling the flow of gradients through shortcut paths, residual blocks improve convergence, especially in deeper networks, and help preserve feature representations from earlier layers.

Following the convolutional and residual layers, the feature maps are **flattened** and passed through a fully connected **dense layer with 256 units**, followed by a **dropout layer with a rate of 0.3** to mitigate overfitting. The final layer is a **dense softmax layer with 29 output neurons**, each corresponding to one of the ASL classes.

The model was compiled using the **Adam optimizer**, which offers adaptive learning rates and efficient gradient updates. The **sparse categorical cross-entropy** loss function was selected due to its compatibility with integer-labeled multi-class classification tasks. The performance metric used during training was **classification accuracy**, reflecting the model's ability to correctly identify individual ASL gestures.

Table I
Summary of CNN Architecture and Parameter Distribution

Layer Type	Output Shape	Parameters
Input	(200, 200, 3)	0
Conv2D + BN ×3	Varies	~20K
Residual Blocks ×3	(100×100 to 25×25)	~280K
Dense (256)	(256)	4.7M
Output (Softmax)	(29)	7.5K
Total		5.03M

Table I summarizes the structure and parameter distribution of the implemented CNN architecture. The model begins with an input layer designed for RGB images of size **200×200×3**, followed by three convolutional layers paired with batch normalization, which collectively contain approximately **20,000 parameters**.

These layers extract low- to mid-level spatial features while maintaining training stability. The core of the network includes **three residual blocks**, spanning output resolutions from **100×100 to 25×25**, and contributing around **280,000 parameters**. These blocks enhance feature learning by allowing identity mappings through skip connections, improving gradient flow and convergence speed. A dense layer with **256 units** serves as the penultimate layer, accounting for the majority of parameters at **4.7 million**, and is followed by a

softmax output layer with **29 neurons** corresponding to each ASL class.

The total parameter count of the model is approximately **5.03 million**, indicating a moderate-size architecture that balances representational power and computational efficiency, suitable for deployment even on CPU-based systems.

D. Training

The model was trained for a total of 10 epochs using the TensorFlow deep learning framework. To prevent overfitting and reduce unnecessary training time, early stopping was implemented with a patience parameter of 3. This mechanism continuously monitored the validation loss during training, and automatically halted the process if no improvement was observed for three consecutive epochs. Such an approach ensures that the model does not continue training once it reaches its optimal generalization capability.

Training was conducted in a **local CPU-based environment**, which limited the training speed but remained sufficient due to the moderate size of the dataset and relatively shallow network architecture. While GPU acceleration could have significantly reduced the training time, the use of a CPU setup demonstrated that the model is lightweight enough to be trained and deployed in resource-constrained systems. Despite these limitations, the training process was completed successfully and achieved stable convergence within the specified number of epochs.

E. Real-time Prediction Pipeline

The model is deployed using OpenCV to read live video input. A Region of Interest (ROI) is defined where the user shows gestures. Predictions are made on this ROI after resizing and normalization. The system uses temporal smoothing with a queue of the last 10 predictions to enhance reliability.

F. Performance Improvement

To enhance the performance and stability of the model, two key strategies were incorporated into the training pipeline: the use of **residual blocks** and the implementation of **early stopping**. The addition of residual blocks significantly improved the network's ability to learn complex spatial features by enabling direct gradient flow through skip connections.

This architectural enhancement mitigated the vanishing gradient problem often encountered in deeper convolutional networks and allowed the model to retain critical low-level information across layers. As a result, the residual blocks contributed to faster convergence and improved classification accuracy, particularly for visually similar ASL gestures. In parallel, early stopping was employed as a form of regularization by monitoring validation loss and halting training when no improvement was observed for a specified number of epochs.

This technique prevented overfitting, reduced training time, and ensured that the final model retained only the most generalizable parameters. Together, these modifications led to a more robust and efficient learning process, with increased accuracy and stability during both validation and real-time testing phases.

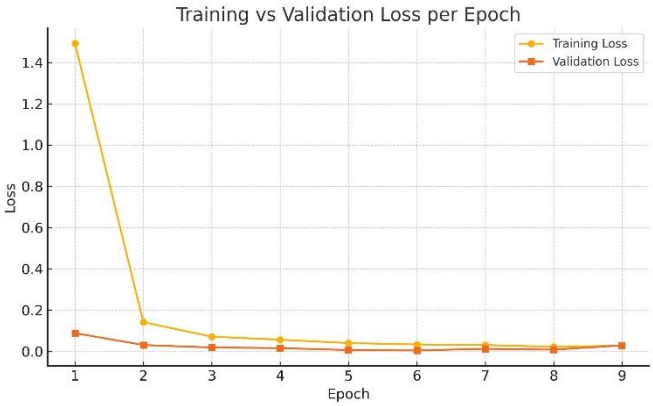

Fig 1. Training and validation loss over epochs after adding res blocks and early stopping

Fig 1 illustrates the model's decreasing loss, showing effective convergence by the 6th epoch with minimal overfitting.

IV. EXPERIMENTS

A. Accuracy

- Validation Accuracy: ~97%
- Real-Time Accuracy: ~90% on clearly captured gestures.

B. Real-Time Performance

- Model inference speed: ~30ms/frame
- Webcam processing resolution: 640×480
- Average FPS (on CPU): 10–15 fps

C. Confidencce Threshold

A confidence threshold of 50% was used to filter low-confidence predictions, reducing noise.

The ASL recognition system demonstrated strong performance across both offline validation and real-time deployment scenarios. The model achieved a **validation accuracy of approximately 97%**, confirming its ability to generalize well on clean, labeled data.

During real-time testing, the system maintained an average accuracy of **around 90%** when gestures were clearly framed and consistently presented. The model was also capable of running at **10–15 frames per second on a CPU**, with an **inference speed of ~30 ms per frame**, enabling practical responsiveness in live settings. To further improve prediction stability, a **confidence threshold of 50%** was applied, effectively filtering out low-confidence outputs and enhancing

the reliability of the system in unpredictable real-world environments. These results highlight the effectiveness of the chosen architecture and preprocessing pipeline in delivering accurate and responsive ASL recognition.

V. Identified Issues

While most letters were correctly recognized, a few letters consistently failed in real-time conditions as presented below.

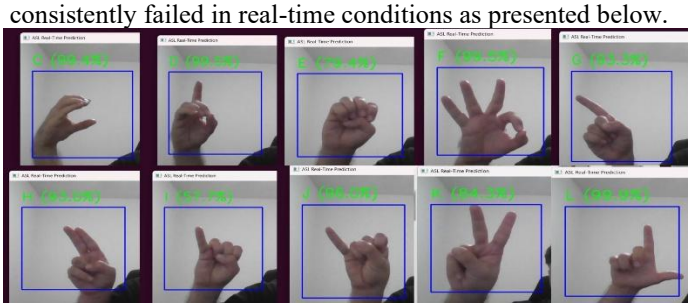

Fig 2. Real-time application execution showing correctly identified letters: C, D, E, F, G, H, I, J, K, L

As shown in **Fig 2** all letters were identified with high confidence except letter I, just 57.7% confidence level, the rest showing confidence over 80%. The gesture for "I" involves extending only the pinky finger, which may resemble other ASL letters like:

"J" – Starts similarly but includes movement (not captured in a static image).

"Y" – Pinky + thumb extended; confusion possible if the thumb is not clearly hidden.

If the thumb isn't fully curled or partially visible, the model may be uncertain.

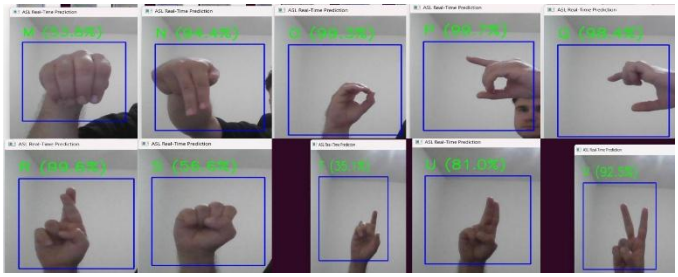

Fig 2. Real-time application execution showing correctly identified letters: M,N,O,P,Q,R,S,T,U,V

As shown in **Fig. 3**, most letters were recognized with high confidence, with the exception of the letters **M** (53.8%) and **T** (35.1%). For the letter **Q**, it was necessary to use the **left hand** to perform the sign because the model had been trained on left-handed images, while the right hand was used during testing. To address this mismatch, the input was horizontally flipped using the command roi = cv2.flip(roi, 1) to ensure the region of interest (ROI) matched the training data orientation.

Letter M had a low confidence rate just 53.8% possible reasons for this. ASL letters like M, N, and sometimes E or S involve very subtle differences in finger positioning.

For example: **M** = 3 fingers over the thumb, **N** = 2 fingers over the thumb, **S** = fist with thumb outside These look extremely similar in RGB images, especially without precise finger segmentation.

As shown in **Fig 3.** Letter T was identified with just 35.1% confidence. In ASL, the letter T is formed by placing the thumb between the index and middle finger, creating a subtle crossing gesture.

From the image, it looks like the **thumb is not clearly visible**, possibly tucked too far in or shadowed, making the sign resemble **L** or **D**, which use extended fingers.

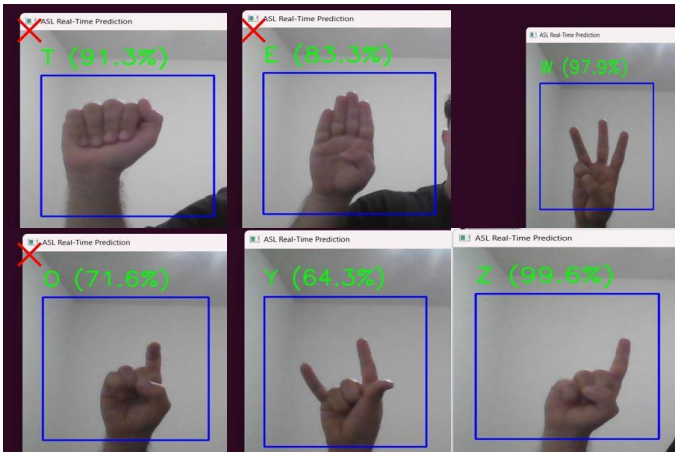

Fig 4. Real-time application execution showing correctly identified letters: W, Y, Z Letters A,B, X were not identified correctly.

As shown in **Fig. 4**, the letter **A** was not correctly identified in most cases and was frequently confused with the letter **N**. The letters **B** and **X** also exhibited poor recognition, likely due to insufficient separation in the training data. The poor recognition of the letter B may stem from its visual similarity to O or C when hand angles vary.

VI. Conclusion

A real-time American Sign Language (ASL) recognition system was developed and evaluated using a custom Convolutional Neural Network (CNN) architecture enhanced with residual connections and a temporal smoothing filter. The integration of residual blocks improved feature learning and model convergence, while temporal smoothing helped reduce noise and stabilize predictions in live video input.

The system demonstrated strong performance in controlled settings, accurately recognizing the majority of static ASL gestures with high confidence.

However, certain limitations were observed, particularly in the classification of gestures with visually similar hand shapes, such as **M**, **N**, and **T**. These confusions were more pronounced in uncontrolled environments where lighting conditions, hand orientation, and background variability introduced inconsistencies not well represented in the training data.

Additionally, the system's reliance on RGB images alone limited its ability to capture subtle finger positioning, especially when gestures involved occluded or overlapping fingers.

To address these challenges, several potential improvements are proposed. Integrating **hand landmark detection** (e.g., using Mediapipe) could allow the model to interpret fine-grained finger positions independent of lighting or skin tone. The incorporation of **depth sensing** or 3D hand pose estimation could further disambiguate similar gestures by capturing spatial relationships that are not easily distinguishable in 2D imagery. Finally, applying **attention mechanisms** within the model architecture could help prioritize relevant regions of the image, enhancing the model's ability to focus on critical gesture features. These enhancements would likely increase the system's robustness and accuracy, particularly in real-world deployment scenarios.

REFERENCES

[1] [1] Starner, T., Weaver, J., & Pentland, A. (1998). Real-time American Sign Language recognition using desk and wearable computer-based video.

[2] Ong, S. C. W., & Ranganath, S. (2005). Automatic sign language analysis: A survey and the future beyond lexical meaning.

[3] Grassknoted, "ASL Alphabet," Kaggle, 2017. [Online]. Available: https://www.kaggle.com/datasets/grassknoted/asl-alphabet