Project Report - ECE 176

Yann Baglin-Bunod
ECE Department
PID: A17045308

Sebastian Castaneda
ECE Department
PID: A17052333

Abstract

This project aims to use Convolutional Neural Networks (CNNs) to interpret American Sign Language (ASL) from images and videos. In doing so, we hope to create a real-time ASL recognition system. The project will explore the intricacies of CNNs, specifically RESNET10, focusing on dataset selection, feature selection, architecture optimization, and the potential to create real-time applications. The innovation and contribution of this work lie in the addition of gray-scale skeleton images of the hands as a fourth channel and the optimization of the CNN architecture for ASL recognition.

1 Introduction

The motivation for this project stems from the need to improve communication accessibility for the deaf and hard-of-hearing community through technology. Understanding ASL is crucial for bridging the communication gap, and automated systems can play a significant role. Our project aims to address the challenge of interpreting the ASL alphabet through a real-time recognition system using CNNs, acknowledging the complexity of sign language's visual and dynamic nature.

2 Related Work

Previous works in ASL recognition have largely focused on static images or slow video processing. For example, research by Starner et al. (1998) demonstrated early success in glove-based ASL recognition, while more recent studies have leveraged deep learning for improved accuracy without wearable sensors.

Our approach is distinguished by its emphasis on real-time processing and the use of advanced CNN architectures, which have shown significant potential in related image and video recognition tasks. In a similar vein, the work by Mallikharjuna Rao K et al. on Indian Sign Language Recognition emphasizes the development of a real-time, accurate system utilizing advanced CNNs, marking a step forward in the field by addressing the unique challenges of ISL recognition.

Our method, integrating skeletal data as a fourth channel in the image data, expands upon these foundations, offering a novel approach to enhancing the feature set available for CNNs in the context of ASL recognition. This addition aims to refine the model's ability to discern nuanced gestures that are instrumental to obtaining effective sign language interpretation.

3 Method

Our method utilizes a Residual Network (Resnet) architecture designed to capture the intricacies of ASL gestures. The network extracts spatial features from individual frames and temporal features across sequences. Training involves a combination of real-world ASL datasets and augmented data to enhance the model's robustness to varied environments and signing styles.

Preprint. Under review.

The optimizer used is SGD with momentum and Nesterov acceleration, enhancing the ResNet model's training. It speeds up convergence by using a velocity vector in the direction of persistent loss reduction and anticipates future positions of parameters for more accurate updates. This results in faster and more efficient model optimization, particularly valuable for complex tasks like ASL gesture recognition, by efficiently navigating the weight space of deep networks.

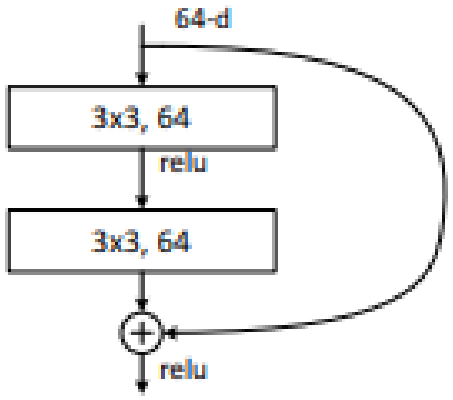
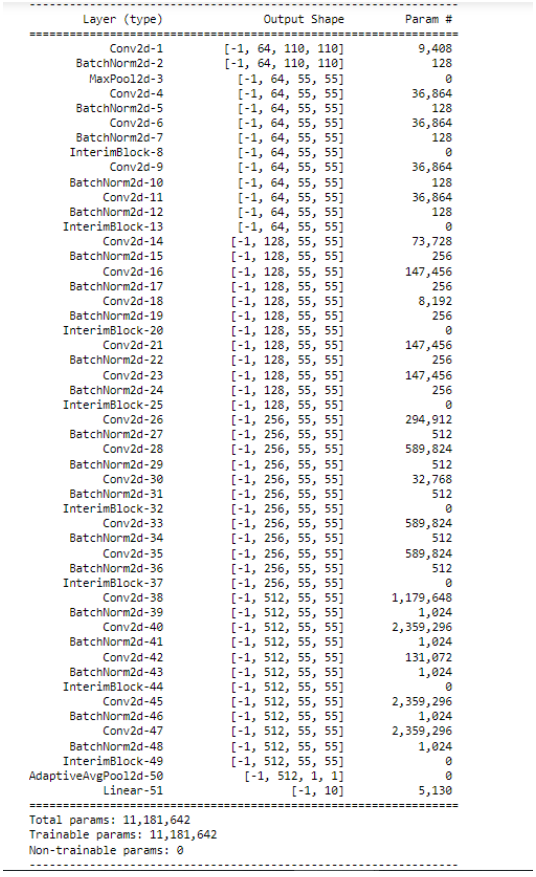


Figure 1: Single residual block from a ResNet architecture

Resnet implements a deep neural network architecture that introduces "residual blocks" to enable the training of much deeper networks than were previously feasible (representation of residual block Fig 1). Each residual block adds the input of the block to its output, helping to mitigate the vanishing gradient problem by allowing gradients to flow through the network more effectively during training. This innovation addresses the issue where adding more layers to a deep neural network paradoxically led to higher training error, due to difficulties in optimizing the network. ResNet's architecture has been highly influential, achieving state-of-the-art performance in various computer vision tasks and setting new benchmarks for deep learning models. Figure 2 shows the structure of the cnn used for this report.



Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 110, 110]	9,408
BatchNorm2d-2	[-1, 64, 110, 110]	128
MaxPool2d-3	[-1, 64, 55, 55]	0
Conv2d-4	[-1, 64, 55, 55]	36,864
BatchNorm2d-5	[-1, 64, 55, 55]	128
Conv2d-6	[-1, 64, 55, 55]	36,864
BatchNorm2d-7	[-1, 64, 55, 55]	128
InterimBlock-8	[-1, 64, 55, 55]	0
Conv2d-9	[-1, 64, 55, 55]	36,864
BatchNorm2d-10	[-1, 64, 55, 55]	128
Conv2d-11	[-1, 64, 55, 55]	36,864
BatchNorm2d-12	[-1, 64, 55, 55]	128
InterimBlock-13	[-1, 64, 55, 55]	0
Conv2d-14	[-1, 128, 55, 55]	73,728
BatchNorm2d-15	[-1, 128, 55, 55]	256
Conv2d-16	[-1, 128, 55, 55]	147,456
BatchNorm2d-17	[-1, 128, 55, 55]	256
Conv2d-18	[-1, 128, 55, 55]	8,192
BatchNorm2d-19	[-1, 128, 55, 55]	256
InterimBlock-20	[-1, 128, 55, 55]	0
Conv2d-21	[-1, 128, 55, 55]	147,456
BatchNorm2d-22	[-1, 128, 55, 55]	256
Conv2d-23	[-1, 128, 55, 55]	147,456
BatchNorm2d-24	[-1, 128, 55, 55]	256
InterimBlock-25	[-1, 128, 55, 55]	0
Conv2d-26	[-1, 256, 55, 55]	294,912
BatchNorm2d-27	[-1, 256, 55, 55]	512
Conv2d-28	[-1, 256, 55, 55]	589,824
BatchNorm2d-29	[-1, 256, 55, 55]	512
Conv2d-30	[-1, 256, 55, 55]	32,768
BatchNorm2d-31	[-1, 256, 55, 55]	512
InterimBlock-32	[-1, 256, 55, 55]	0
Conv2d-33	[-1, 256, 55, 55]	589,824
BatchNorm2d-34	[-1, 256, 55, 55]	512
Conv2d-35	[-1, 256, 55, 55]	589,824
BatchNorm2d-36	[-1, 256, 55, 55]	512
InterimBlock-37	[-1, 256, 55, 55]	0
Conv2d-38	[-1, 512, 55, 55]	1,179,648
BatchNorm2d-39	[-1, 512, 55, 55]	1,024
Conv2d-40	[-1, 512, 55, 55]	2,359,296
BatchNorm2d-41	[-1, 512, 55, 55]	1,024
Conv2d-42	[-1, 512, 55, 55]	131,072
BatchNorm2d-43	[-1, 512, 55, 55]	1,024
InterimBlock-44	[-1, 512, 55, 55]	0
Conv2d-45	[-1, 512, 55, 55]	2,359,296
BatchNorm2d-46	[-1, 512, 55, 55]	1,024
Conv2d-47	[-1, 512, 55, 55]	2,359,296
BatchNorm2d-48	[-1, 512, 55, 55]	1,024
InterimBlock-49	[-1, 512, 55, 55]	0
AdaptiveAvgPool2d-50	[-1, 512, 1, 1]	0
Linear-51	[-1, 10]	5,130

Total params: 11,181,642
Trainable params: 11,181,642
Non-trainable params: 0

Figure 2: Resnet structure used for project

Figure 2 illustrates a ResNet model's structure, detailing various layers including convolutional (Conv2d), batch normalization (BatchNorm2d), pooling (MaxPool2d and AdaptiveAvgPool2d), and intermediate residual blocks (InterimBlock). The model depth is shown by the sequence of layers, each followed by normalization to stabilize learning. Residual blocks allow training deeper networks by adding inputs to outputs, preventing gradient vanishing. The final pooling layer reduces spatial dimensions before classification. The model has over 11 million trainable parameters, indicating its capacity for complex feature learning.

4 Experiments

The primary dataset used is the American Sign Language Dataset by Akash, featuring images of ASL signs. The data set contains 87,000 images (.jpg) which are 200x200 pixels. There are 29 classes, of which 26 are for the letters A-Z and 3 classes for SPACE, DELETE and NOTHING. We created a 90/10 training testing split on our data.

We also use another dataset: This data set consists of a set of 870 TEST images 200x200 pixels (.jpg). Each image contains a hand making the shape of an ASL letter (with some variation).

Our experiments focus on model training, hyper-parameter optimization, and real-time performance evaluation. Results show our system's ability to accurately recognize signs with minimal latency. An ablation study further demonstrates the importance of each CNN layer and the impact of data quantity on model performance.

In our training we decided to use the Media Pipe Python package, which is an open-source framework for building pipelines to perform computer vision inference over arbitrary sensory data such as video or audio. The package allowed us to create images like this:

Figure 3: 'A' in ASL, skeletal representation

Figure 4: 'A' in ASL, RGB representation

Figure 5: 'A' in ASL, RGBA representation (skeleton + RGB)

The skeleton helped the model focus on the most relevant parts of the image for ASL recognition—the position and configuration of the fingers and palm. This reduced the impact of background noise and irrelevant details in the original image. It was passed into a fourth channel – serving as a hyperparameter.

Consider a convolution operation in the initial layer of a ResNet:

$$O = W * I + b \tag{1}$$

where:

- O is the output feature map,
- W is the weight matrix of the convolution filter,
- I is the input image with four channels (RGB and grayscale skeleton),
- b is the bias term,
- $*$ denotes the convolution operation.

In this equation, W will have dimensions that consider the fourth channel, allowing the network to learn filters that specifically respond to the skeletal structure, separate from color-based features.

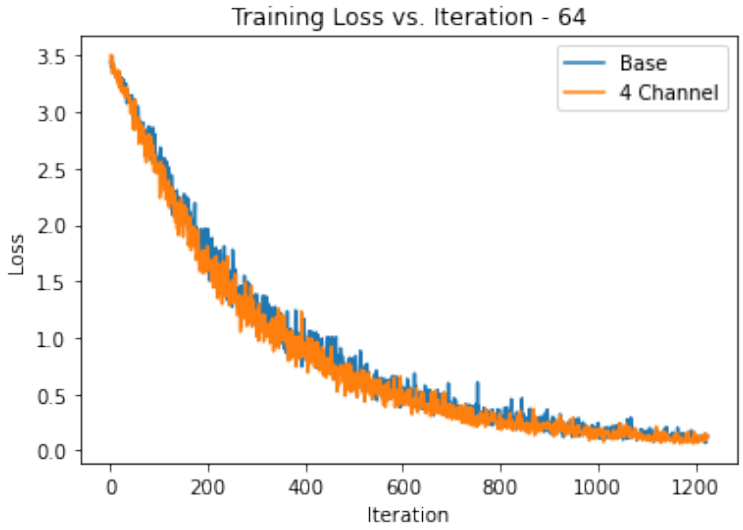


Figure 6: Loss using base training set (3 channels) and 4 Channel set, Batch Size = 64

We found that adding this fourth channel, lead to more accurate recognition of ASL signs as the model can better learn the nuances of hand shapes and positions that define different signs. As seen in Figure 7, the 4 channel input model outperforms the base model in a more stable learning, likely due to its ability to process more comprehensive information through additional input channels. These extra channels could be providing nuanced details that enhance feature detection, enabling the model to better handle the variability within ASL gesture data.

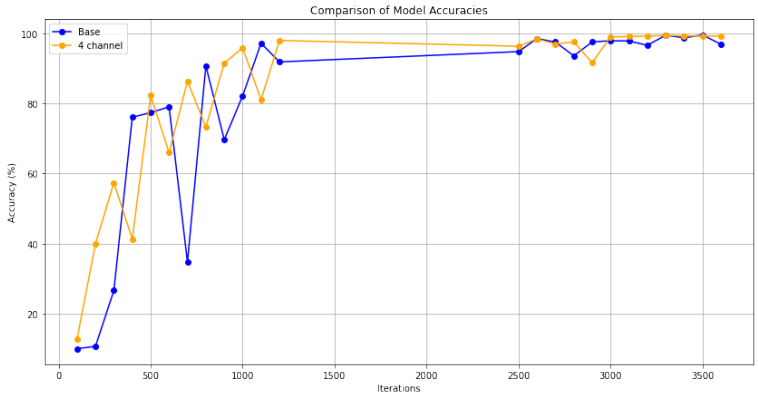


Figure 7: Accuracy 3 vs 4 channels

Results are discussed more here. Where we experimented with learning rate!

5 Supplementary Material

This video showcases the motivation, methodology, and results of the project, demonstrating the ASL recognition system in action.

References

1. Starner, T., Weaver, J., and Pentland, A. (1998). Real-time American sign language recognition using desk and wearable computer based video. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(12), 1371-1375.
2. Nagaraj, A. (2018, April 22). ASL alphabet. Kaggle. https://www.kaggle.com/datasets/grassknoted/asl-alphabet
3. He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385. Retrieved from https://arxiv.org/pdf/1512.03385.pdf
4. Rasband, D.(2018). ASL Alphabet Test. ASL Alphabet Images with a variety of backgrounds for validating a model. Kaggle. Retrieved from https://www.kaggle.com/datasets/danrasband/asl-alphabet-test/data
5. Mallikharjuna Rao K, Harleen Kaur, Sanjam Kaur Bedi, M A Lekhana. (2023). Image-based Indian Sign Language Recognition: A Practical Review using Deep Neural Networks. arXiv:2304.14710 [cs.CV]. https://arxiv.org/ftp/arxiv/papers/2304/2304.14710.pdf