Individual Project Report
Andy Christian
DATS 6303
April 25, 2023

ASL Fingerspelling Image Classification

I. Introduction

The goal of the project was to provide a proof of concept for using machine learning to interpret American Sign Language (ASL). A data set was found containing images of ASL fingerspelling (alphabet letters). A simple Convolution Neural Network (CNN) was used to classify each image by its corresponding letter. As the project was successful, future work could improve on this project by using object detection with a live video feed to translate ASL in real time, including words as opposed to just letters.

II. Description of Individual Work

NOTE: a python file showing examples of the code I contributed to the project can be found inside the Code folder of my individual folder in the project repo. I commented out the code written by others and put blocks of ##### symbols around the sections of code which I added.

Daqian and Rajkumar worked on the preprocessing code and Rajkumar set up the baseline model. I reviewed the initial code base, identifying some errors and areas for improvement, specifically on how the data loader was being setup. However, it was Rajkumar who made and pushed the changes to the code based on my comments.

After the preprocessing and baseline model were cleaned up, I included code to add in several important components to ensure we could experiment with the model and do testing properly. This included adding in seeding to ensure reproducibility and creating a model save point to keep the best models produced by our optimizations. Additionally, I added in code for early stopping and learning rate scheduling to ensure we addressed issues of overfitting. I made a few other quality improvements, such as separating the final testing from the training loop so that it was only run after the model had finished training.

III. Results

All three of us conducted our own optimization experiments. For my optimization experiments, I played around with different learning rates, batch sizes, number of filters, kernel sizes of the filters, adding extra convolution layers, using dropout, including L2 regularization, and adjusting the patience of early stopping. In general, I found that larger batch sizes and smaller learning rates led to smoother convergence and very low loss values (0.02-0.01). However, despite these type of setups having low loss, the final model performance on the test data resulted in macro F1 scores around 0.9. When smaller batch sizes and larger learning rates were used, the loss values bottomed out around 0.1, but the macro F1 score on the test set reached 0.98.

With this particular data set, solving this particular problem, batch size and learning rate were the hyperparameters that really seemed to matter. Dropout and L2 regularization appeared to have little effect. If anything, including them caused the model to perform slightly worse on the test set. Increasing the number of convolutions layers, the number of filter maps and the kernal size of the filter maps also had a negative impact, and when made too large, caused training to fail completely. Ultimately, my best model had a batch size of 64, a learning rate of 0.001, and only three convolution layers with a relatively small number of filter maps (16,32,64). Dropout and L2 regularization were included. The macro F1 of the test set for my best model was 0.9874.

Hyperparameter for Andy's model	
Learning rate	0.001
Minibatch size	64
Kernel size	5 x 5
N of convolution layers	3
Dropout implemented in each layer	0.3
Filter maps per layer	16,32,64
Epoch Number	108
Optimizer	ADAM
L2 regularization (weight decay)	0.001
Early Stopping patience	30

Results for Andy's model	
Best N of Epoch	108
Train loss	0.109
Train accuracy	1.0
Validation loss	0.13175
Validation accuracy	1.0
F1 score for test set	0.9874

IV. Summary and Conclusion

The project was successful in achieving the goal. With a very simple CNN model, we were able to predict ASL fingerspelling with high accuracy, precision and recall. The results highlighted the value of using a CNN to solve this task, as the authors of the research paper from 2011, from which the data was taken, only obtained around 0.7 precision, on average, for each letter. They used a random forest model. It was strange to me that the model with the best predictions on the test data had relatively high loss (0.1) compared to other versions (0.01-0.02). Further investigation needs to be conducted for understanding why the model had worse predictions when if was more "confident" about the predictions it was making.

V. Code Breakdown

Total lines contributed: 46
Number of copied Lines: 20
Number of lines modified: 6
Number of unique Lines: 26
Score = (20-6)/(20+26)*100 = 30.43

VI. References

Bowden, R., Pugeault, N. (2011). Spelling It Out: Real–Time ASL Fingerspelling Recognition. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. https://doi.org/10.1109/ICCVW.2011.6130290

Geislinger, V. (2020). ASL Fingerspelling Images (RGB & Depth) (Version 2)[Data Set]. Kaggle. https://www.kaggle.com/datasets/mrgeislinger/asl-rgb-depth-fingerspelling-spelling-it-out