Noah Olsen
Group 9
Machine Learning II

Individual Report

Introduction:

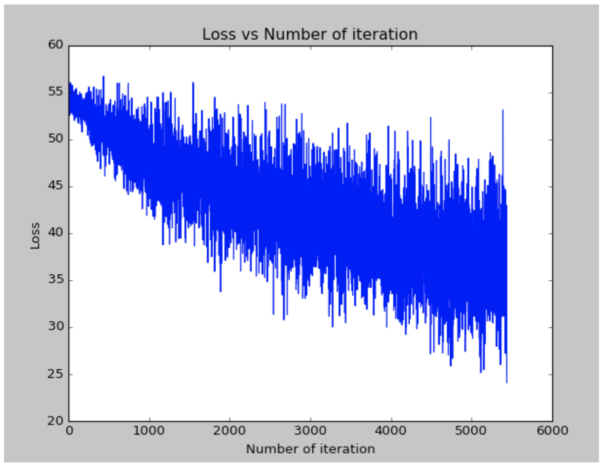
For our project we decided to look at a very large (over 87,000 images and over 1GB) dataset of different letters of the sign language alphabet. Our goal is to train an original Convolutional Neural Network to identify these signs. We then attempted to compare our model with popular models that we then tweaked to work with our specific dataset. This problem has significant applications especially when it comes to the ability of computers to interact with people using American Sign Language.

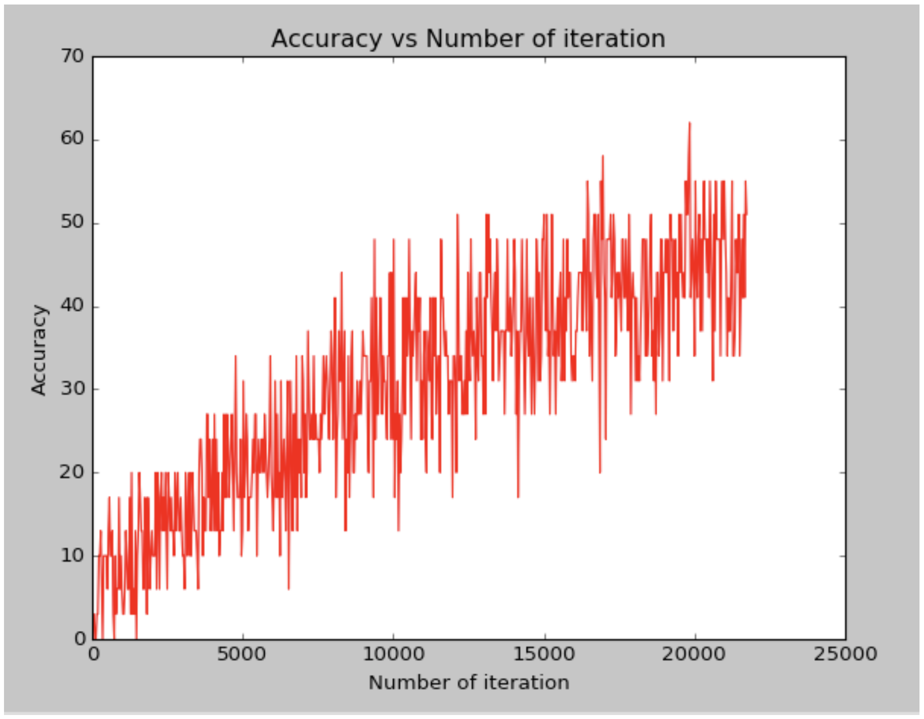
My individual work:

I was responsible for several tasks in our project. Firstly, I was responsible for all the preprocessing to make sure the data was in place where we could bring it into Pytorch. Secondly, I wrote the code that turned our raw JPG files into a labeled dataset that Pytorch could train on using the ImageFolder function. Next, I worked on getting the three models to run doing significant debugging on all three models that implemented. I was in charge of writing the code to visualize network performance

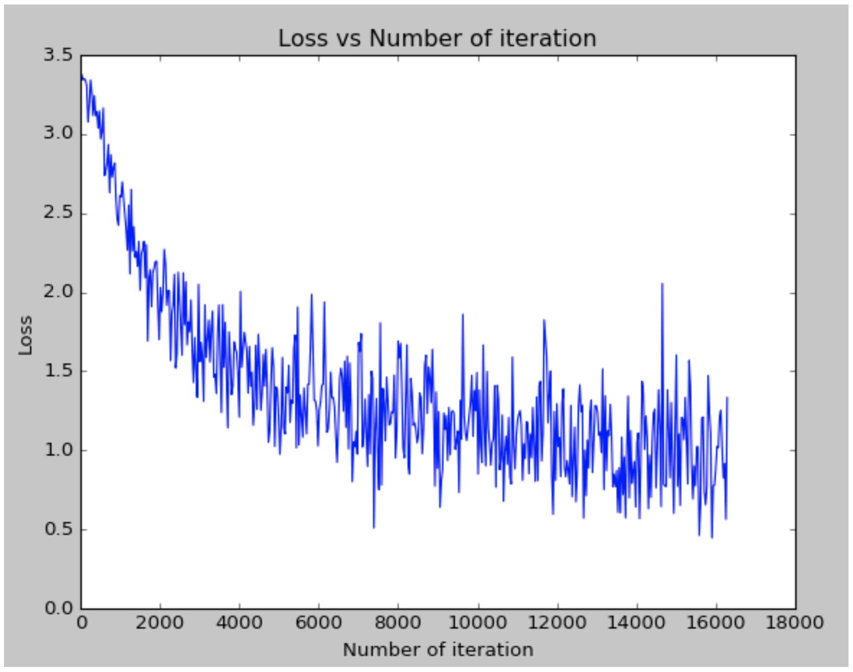
My Results:

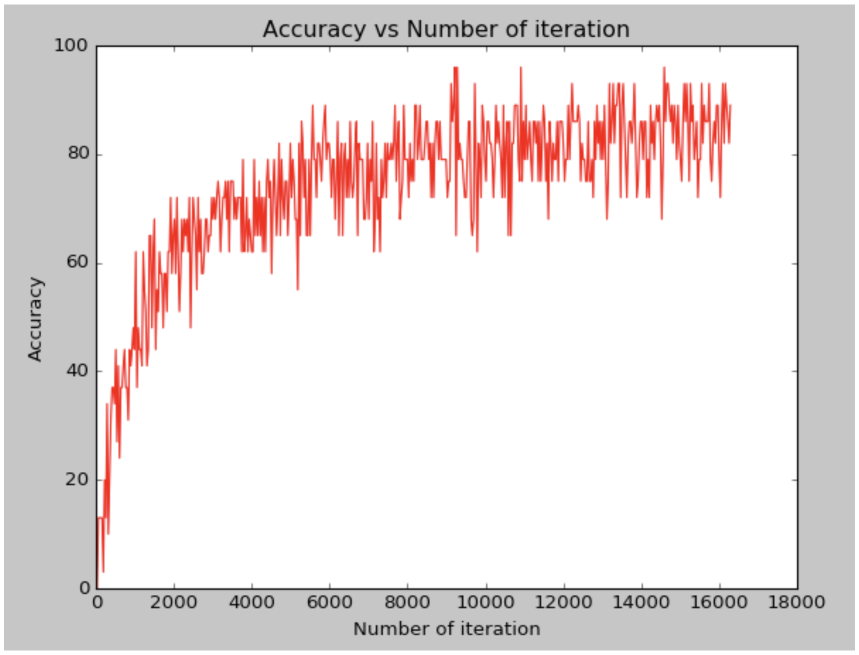
Our Original Model: This model struggled to get much better than 50% accuracy. However, it never converged but do to computational limits it didn't make sense to let it run for days and days. That being said we still saw significant improvement over the iterations and epochs as demonstrated in the charts below.





Adapted Resnet50: This was our strongest performing model. The combination of speed and extremely high accuracy (into the 90%s) was impressive to see. This model also converges very quickly.





VGG16: While this model performed fairly well, we couldn't get it to perform as well as our adapted Resnet model. Our final accuracy for this model hovered at around 80%. Additionally, this model was extremely slow to run.

Summary and Conclusion:

This biggest take away for us on this project was how efficient and accurate deep convolutional networks can be when properly optimized for the task at hand. We struggled mightily to have our convolutional predictor become semi accurate and even then, the model took an extremely long time to run.

Original Code:

The percentage of code that I found and modified from the source below was 20%.

References:

Data: https://www.kaggle.com/grassknoted/asl-alphabet

Main example in pytorch used for reference on CNN:
https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

Main example for use of the ImageFolder function to use folder names as class labels:
https://pytorch.org/tutorials/beginner/data_loading_tutorial.html