ECE324 Assignment 4

Frederick Boyd

# Input Dataset

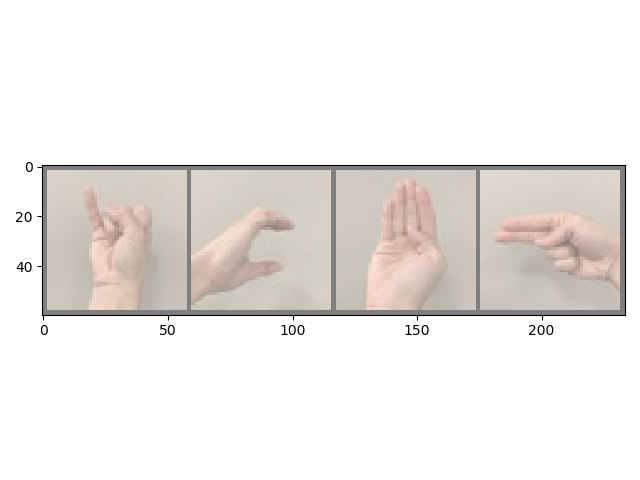


Figure 1: Four example images of ASL signs. The letters from left to right are: I, C, B, and H.

# Model Building and Testing

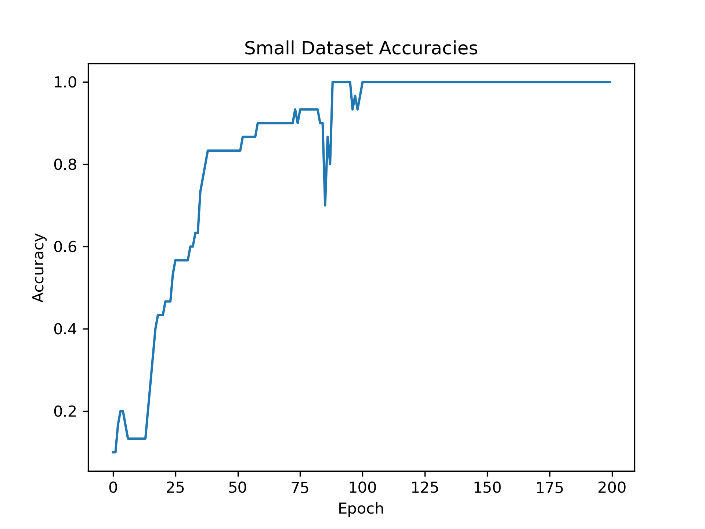
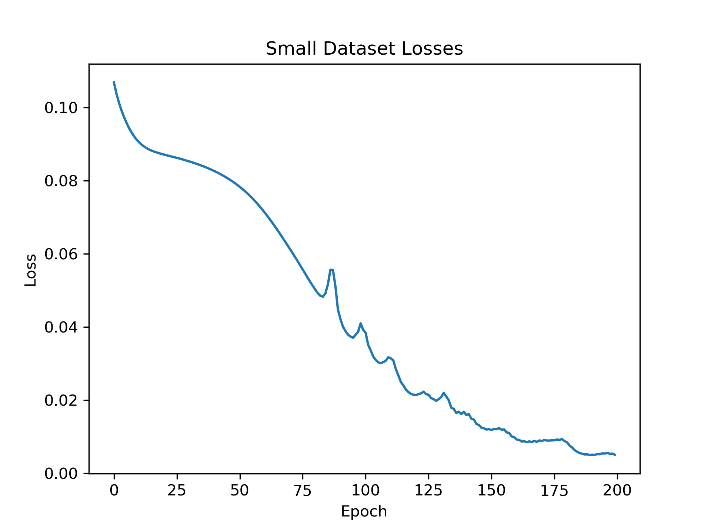
 

Figure 2, 3: Accuracies and loss plotted against epoch for the smaller dataset, which only contains 30 photos.

* Learning rate: 0.2
* Batch size: 30
* Execution time: 21.35s
* Number of epochs required to achieve 100% accuracy: ~110
* Total number of parameters to be trained: 98 830
* Required memory: 0.38MB

# Using the Full Data Set

## Data Loading and Splitting

1. *Which method is better? Why?*

It is a better idea to put some of everyone’s samples into both the training and validation set. Doing this ensures that the neural network isn’t becoming biased towards any one person’s hand. As such, the model can generalize the look of each letter better because it sees a wider variety of data.

1. *What fraction of train/validation split did you use? Why?*

An 80% training, 20% validation split was used because there is well over 1000 photos, which means that only allocating 20% of the data to validation still gives an adequate number of test cases to test the generalization of the neural network.

1. *Report the mean and standard deviation of the image training data that you use.*

* Mean: 1.6679
* Standard deviation: 3.1018

## Hyperparameter Search

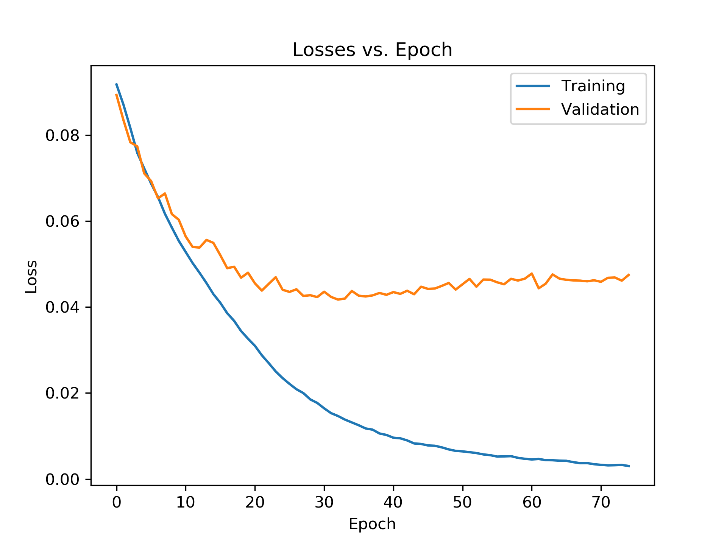
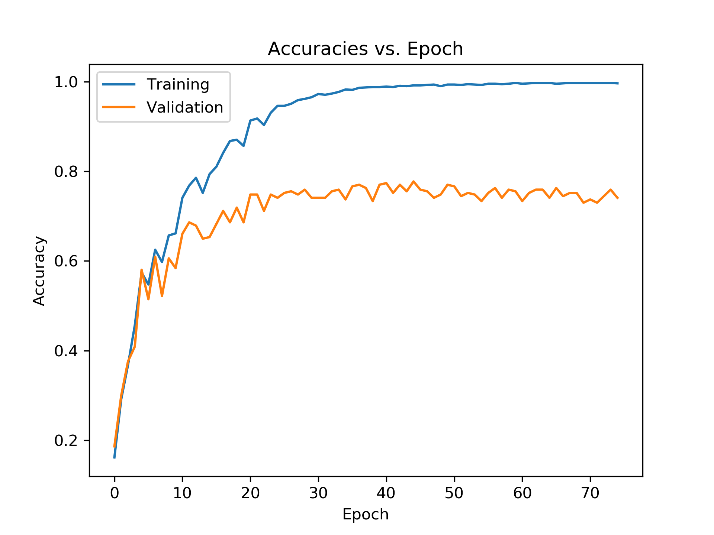
* Number of neurons on first layer: 8
* Learning rate: 0.1
* Batch size: 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *# Convolutional Layers* | *1* | *2* | *4* | *1* | *2* | *4* |
| *# Kernels* | *10* | *10* | *10* | *30* | *30* | *30* |
| *Validation Accuracy* | 0.412 | 0.693 | 0.471 | 0.099 | 0.657 | 0.675 |
| *Training Execution Time [s]* | 103.00 | 216.30 | 362.90 | 30.0 | 567.60 | 569.33 |
| *# Epochs to Train* | 40 | 60 | 60 | 10 | 40 | 50 |
| *# Parameters in Network* | 58698 | 12808 | 3188 | 175898 | 43628 | 25568 |
| *Parameter Memory [MB]* | 0.22 | 0.05 | 0.01 | 0.67 | 0.17 | 0.1 |

* Number of convolutional layers: 2
* Number of kernels: 10

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *# Neurons* | *8* | ***32*** | *8* | *32* | *32* | *8* |
| *Learning Rate* | *0.1* | ***0.1*** | *0.01* | *0.01* | *0.1* | *0.01* |
| *Batch Size* | *32* | ***4*** | *32* | *32* | *32* | *32* |
| *Validation Accuracy* | 0.478 | **0.753** | 0.526 | 0.580 | 0.746 | 0.233 |
| *Training Execution Time [s]* | 193.39 | **110.00** | 824.90 | 539.62 | 370.00 | 469.00 |
| *# Epochs to Train* | 100 | **40** | 150 | 250 | 120 | 150 |
| *# Parameters in Network* | 12808 | **47632** | 12808 | 47632 | 47632 | 12808 |
| *Parameter Memory [MB]* | 0.05 | **0.18** | 0.05 | 0.18 | 0.18 | 0.05 |

The bolded hyperparameters in the table above produced the best results with a validation accuracy of 0.753. It appears that a single or four convolutional layers negatively impacted the results. Increasing the number of kernels to 30 did not result in a significant improvement in validation accuracies. Increasing the number of neurons in the first layer from 8 to 32 improved the accuracy slightly. A batch size of 4 gave slightly better accuracies as well compared to a batch size of 32.



Figures 4, 5: Plots of accuracies and losses against epochs for the best combination of hyperparameters. The exact hyperparameters is the bolded entry in the tables above.

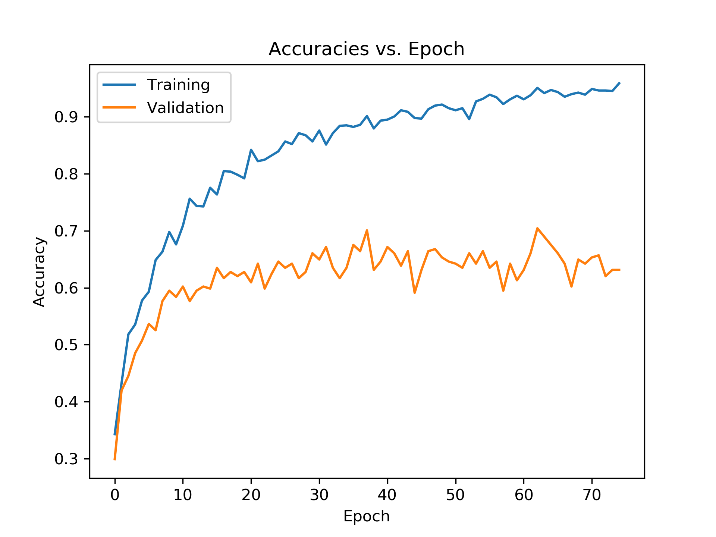
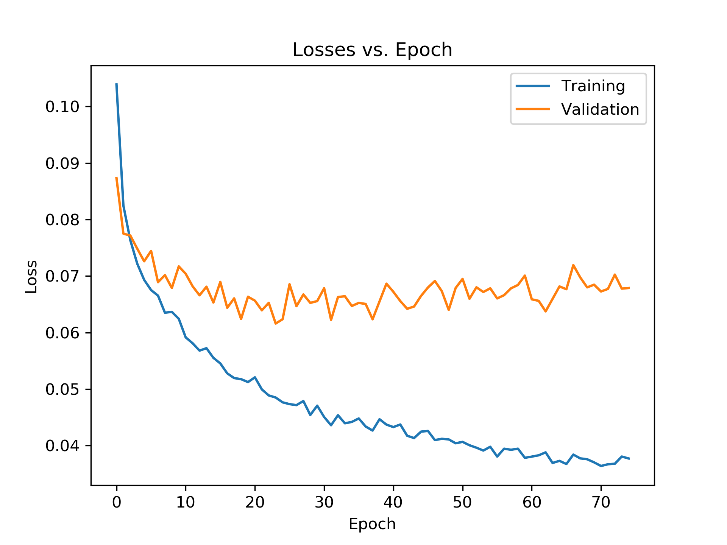
# Batch Normalization and Cross Entropy Loss Function

The following hyperparameters were kept constant for this section:

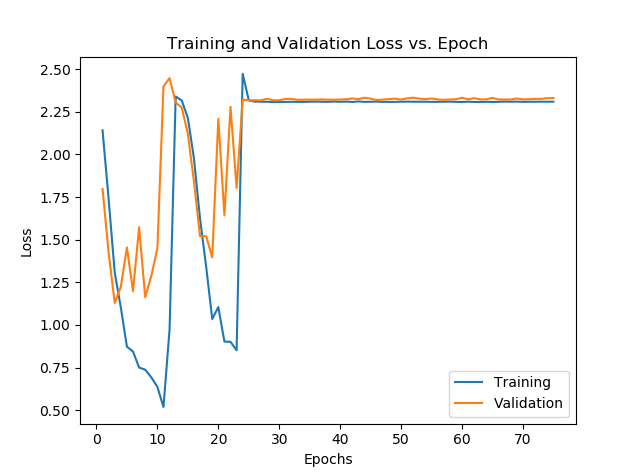
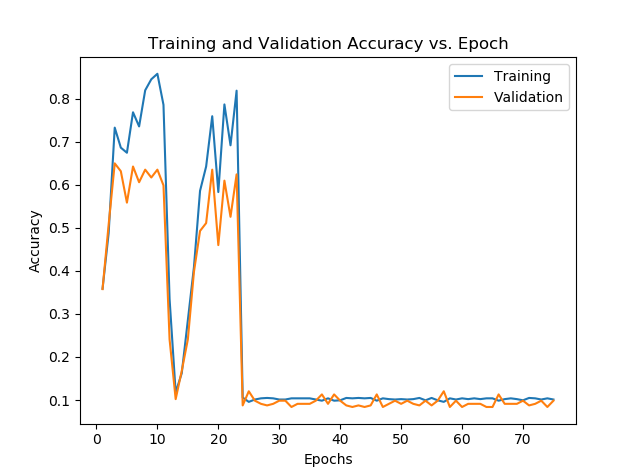
* Convolutional layers: 2
* Number of kernels: 10
* Neurons: 32
* Learning rate: 0.1
* Batch size: 4

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Batch Normalization* | *Cross Entropy Loss* | *Batch Normalization + Cross Entropy Loss* |
| *Validation Accuracy* | 0.631 | 0.610 | 0.631 |
| *Training Execution Time [s]* | 317.74 | 382.00 | 322.37 |
| *# Epochs to Train* | 60 | 25 | 40 |
| *# Parameters in Network* | 47736 | 47632 | 47736 |
| *Parameter Memory [MB]* | 0.18 | 0.18 | 0.18 |

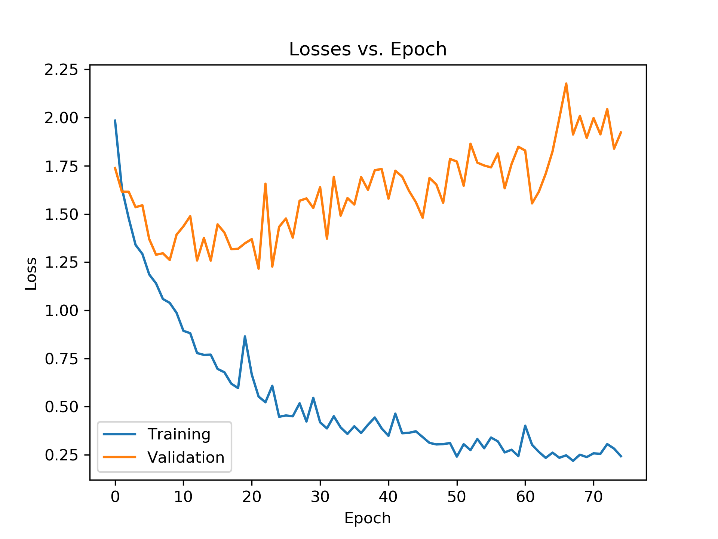
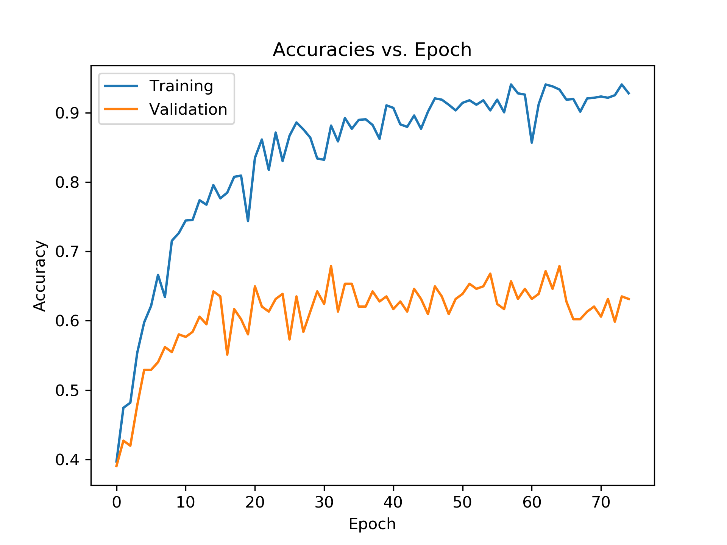
* Plots of just adding batch normalization can be found in Figures 6 and 7
* Plots of just adding cross entropy loss can be found in Figures 8 and 9
* Plots of adding both batch normalization and cross entropy loss can be found in Figures 10 and 11



Figures 6, 7: Plots of losses and accuracies against epochs when just using batch normalization on the best neural network from the Hyperparameter Search section.



Figures 8, 9: Plots of losses and accuracies against epochs when just using cross entropy loss on the best neural network from the Hyperparameter Search section.



Figures 10, 11: Plots of losses and accuracies against epochs when using both batch normalization and cross entropy loss on the best neural network from the Hyperparameter Search section.

Just adding batch normalization appears to make the neural network perform slightly worse than the best one from the Hyperparameter Search section. However, its overall shape is the same with steadily increasing accuracies and decreasing losses.

Just adding cross entropy loss makes the neural network’s performance significantly worse. There is a sudden decrease in accuracy around 10 epochs before it goes back up. However, the accuracy ends up plummeting to 0.1 again at around 25 epochs. After this point, the accuracies and losses remain relatively constant. From these results, it appears that just adding cross entropy loss is not a good idea for this specific neural network.

Adding both batch normalization and cross entropy loss to the neural network results in a hybrid between the two individual results. Although the general shape of the accuracy plot tends upward, the validation accuracy seems to start decreasingly slightly starting around the 35th epoch. This behaviour can be seen more clearly in the loss plot where the validation loss initially decreases, but slowly starts to increase starting from the 20th epoch.

# Final Best Networks and Confusion Matrix

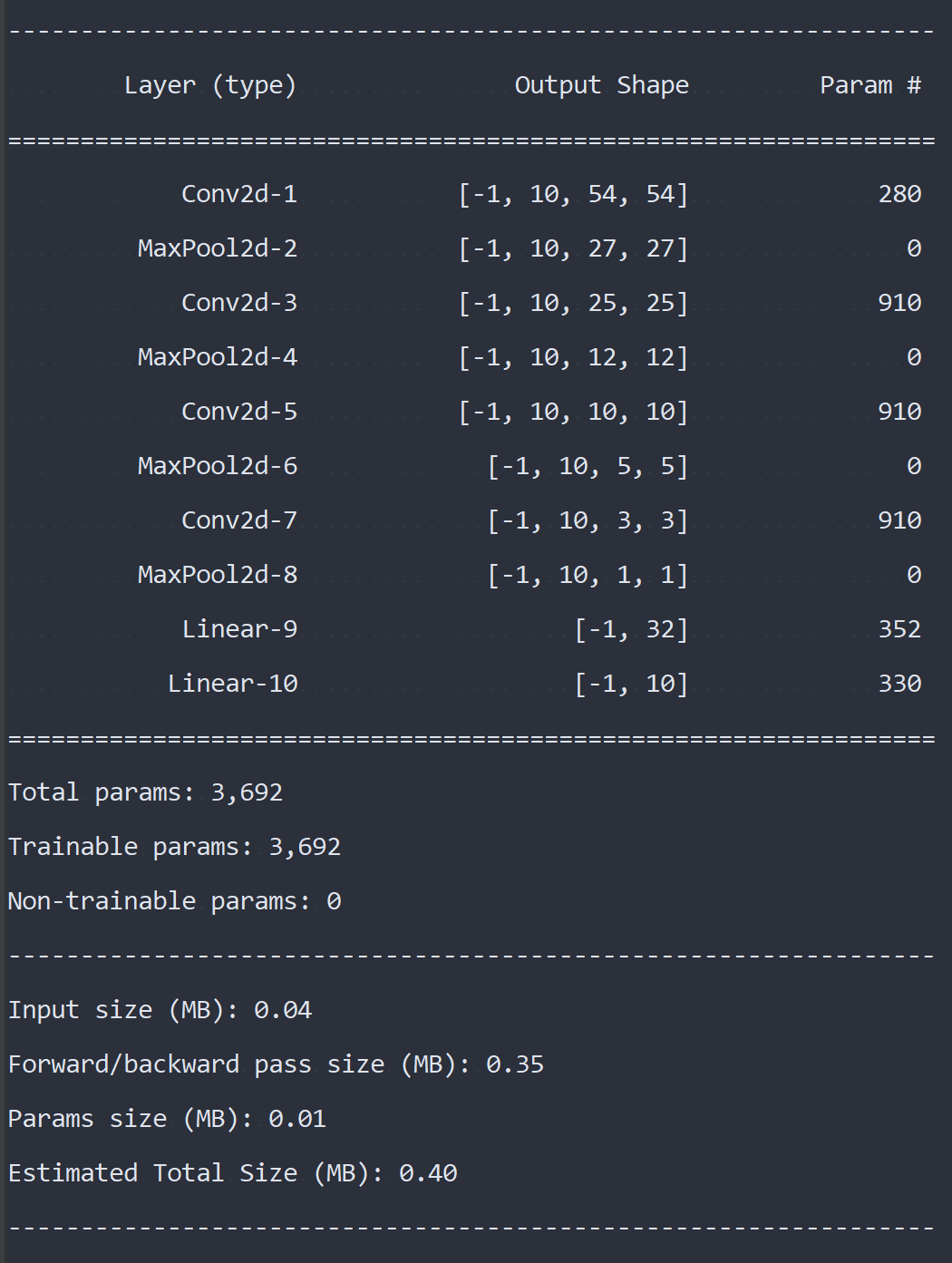
From testing various hyperparameters, the following values were found to produce the best results:

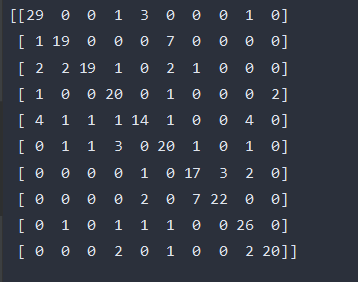
* Loss function: MSE
* Activation function: ReLU
* Number of convolutional layers: 2
* Number of kernels: 10
* Number of neurons: 32
* Learning rate: 0.1
* Batch size: 4

This neural network resulted in a validation accuracy of 75.3%, which is the highest I was able to find.

The highest accuracy neural network with a number of parameters below 5000 used the following hyperparameters:

* Loss function: MSE
* Activation function: ReLU
* Number of convolutional layers: 4
* Number of kernels: 10
* Number of neurons: 32
* Learning rate: 0.2
* Batch size: 32





The classes that got confused the most were the two 7s in the confusion matrix above. These correspond to the classes F, B and G, H. This confusion makes sense because when looking at the ASL symbols for F and B, they are very similar. The only difference between the two is the position of the thumb and second finger. Similarly, G and H have a very similar look.