Assignment 3 Report: CNN

Chandler Ault

COS 429: Computer Vision

10/30/20

Architecture:

I decided to create my baseline CNN similar to the one shown in the LeCun paper. My model had two convolutional layers. Both layers had a filter size of two and the first outputted 6 filters while the second outputted 16 filters. I chose such a small filter size because in lecture Pr. Russakovsky said that sometimes smaller filter sizes and deeper models perform better. In retrospect, I think I would have gotten better performance if I had chosen a larger filter size even a small increase to a filter size of 3. After each convolution layer was a pool layer with filter size and stride of two. In my limited experience, I think this filter size was sufficient for a pooling layer, but It would have been useful to have the convolutional layers render down the original image before the pooling layer. After these four layers had a ReLu activation layer and flatten layer to prepare the data for the linear layer. The linear layer had 576 nodes in and 10 nodes out (one node for each class). Finally, I had a Softmax layer since this is a categorization problem and need probabilities.

Model:

Convolutional Layer: size = 2, filters = 6

Pool Layer: size = 2, stride = 2

Convolutional Layer: size = 2, filters = 16

Pool Layer: size = 2, stride = 2

ReLu Layer

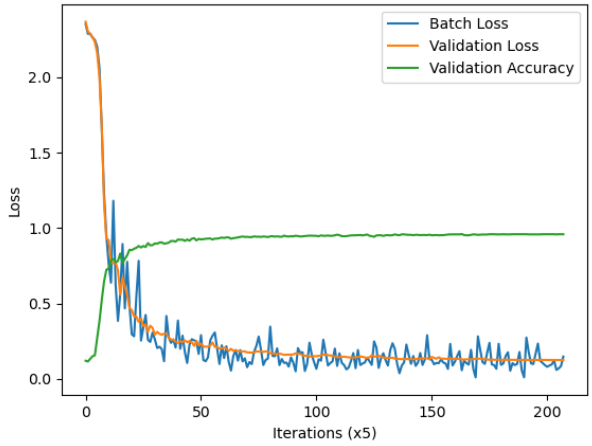
Flatten Layer

Linear Layer: in = 576, out = 10

Softmax Layer

Results:

I used a learning rate of 0.02 and a weight decay of 0.0005. These were around the provided values in the code. When the accuracy plateaued around 93%, I switched to a lower weight decay and learning rate of 0.00005 and 0.0075 respectively to find the minimum better. I had a batch size of 128. Finally, due to limited time training, I implemented SGD with momentum from the beginning to help train faster and because it was highly recommended. I used the first 1000 test images as a validation set.



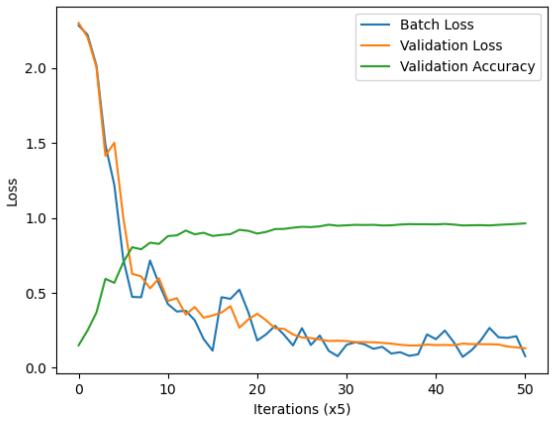
Final Accuracy: 96.64%

I sampled the loss after five batches or iterations, so that is what is represented on the x-axis. I am very pleased with the performance and loss in the model. The validation loss almost perfectly reflects the batch loss indicating I was not overfitting.

Note: I achieved higher accuracy with longer training, but I did not save the validation loss correctly and lost the data.

Further Exploration:

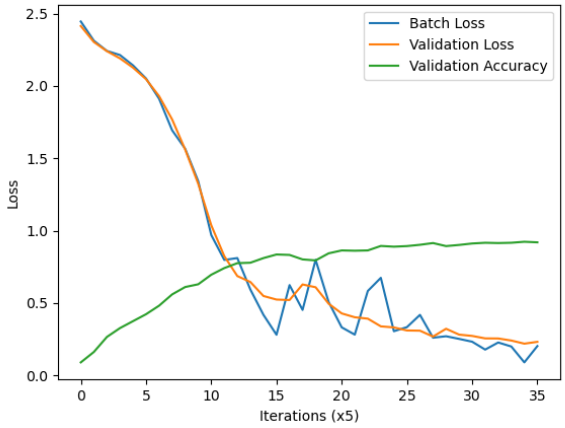
Although I wanted to make my network deeper, but I felt like my filter size of two was so small I needed to increase layer width. Thus, I kept the same number of layers but varied my layer width. The first alteration I made was increasing the filter size of my first two convolutional layers to 4 and 3 respectively. This achieved very good results and trained exceedingly faster than my original model to a satisfactory accuracy. After this, I increased the filter depth to 12 and 32 for my first and second convolutional layers respectively. This had the predictable effect of ballooning the runtime of my model training and did not achieve as good of accuracy after a comparable amount of time. Thus, I think that my original number of filters performed relatively well, and the new filter depth was greater than necessary to achieve 96% accuracy. I do think that increasing the filter size had an entirely beneficial effect from my findings. As mentioned above, I realized after beginning training on my original model that having a filter size of 2 was entirely too small.



Test Accuracy: 96.48%

Filter Size: 4 and 3

Filter Depth: 6 and 16



Test Accuracy: 93.30%

Filter Size: 4 and 3

Filter Depth: 12 and 32

Conclusion:

In conclusion, I believe having slightly larger filter sizes is incredibly beneficial because it achieved comparable performance in a fraction of the time. There is certainly a point of diminishing returns in filter depth, however. The added filter depth does not provide due justification since the accuracy isn't increased by that much compared to the runtime. Furthermore, I think the disproportionate number of filters to number of categories in the classification does little for accuracy. Thus, if given additional time, I would take my second model and experiment with additional layers to try and improve performance as discussed in lecture.