ECE:5450 Machine Learning

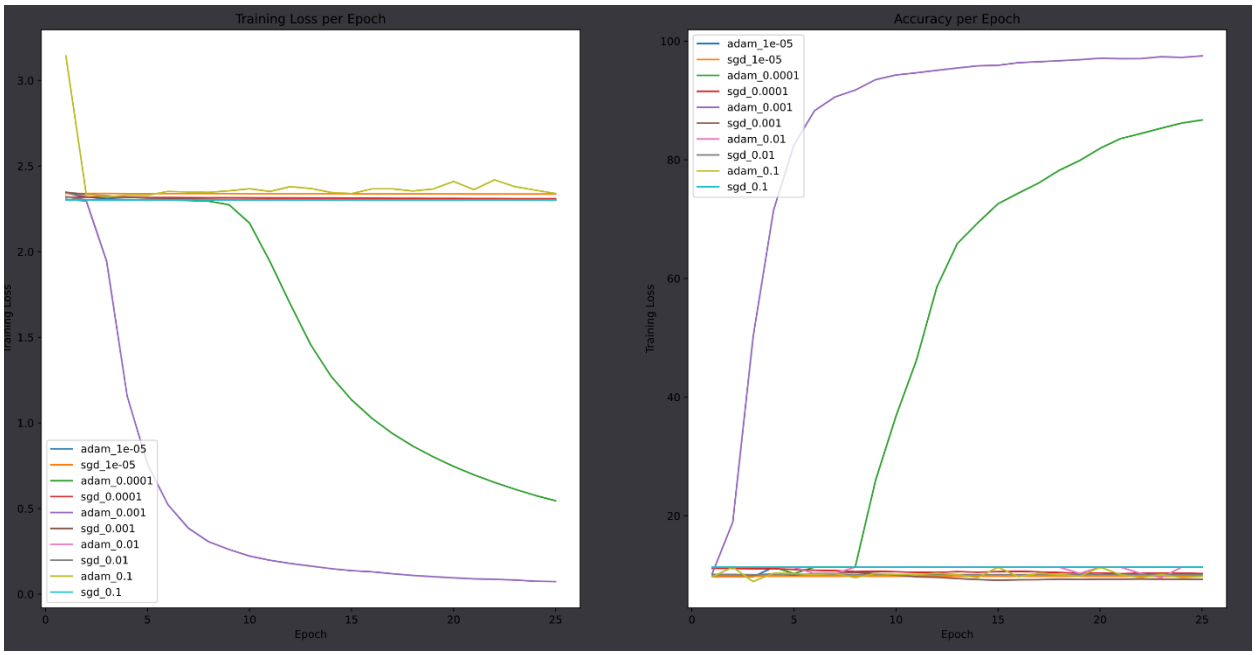
12/18/20

Ryan McDaniel

Project Part 2 Write Up

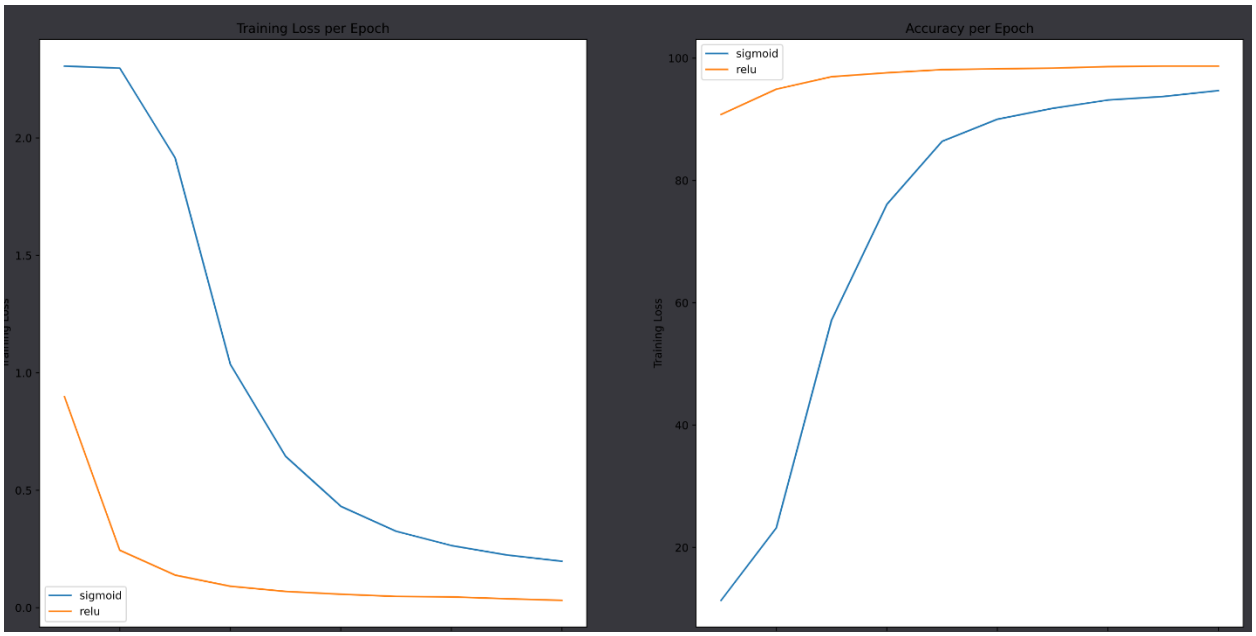
I added this document here because there are a lot of notebooks to go through. The notes in this file summarize my thoughts about the figures and outputs in each of the notebooks. Please refer to the notebooks for the plots and some of my other observations, but I will try to embed as many of the graphics from them in here. There will be a lot of code that does not seem entirely related to the project, but because I frequently reviewed the history of my CNNs, I adopted some programming strategies for ease of use. Fortunately, I was able to implement most of that correctly, and now I am sitting with about 6 GB worth of networks that I have tested.

1. I checked the learning rates for Adam and SGD. I noticed that SGD provided no convergence regardless of the rate. Adam converged with differing speeds between the range of 0.001 and 0.0001. 0.001 was faster than 0.0001. I found it odd that SGD did not converge, but upon further readings, I think this was because the baseline SGD is simply not good for our CNN. Other types of SGDs such as those with momentum would work better and would likely provide better accuracy than Adam. I decided for the rest of the project, I would stick with Adam because it typically has faster convergence speed and would make it easier to debug issues I had often created with my CNNs.



2. The activation functions were difficult to conceptualize for me at first, but I see how with the Sigmoid, the area of highest variance has such little length that the calculations throughout the network essentially vanish into meaningless values as they grow toward the more planar sections of it. I saw that with ReLU, I had faster convergence and epoch speed and higher test

accuracy than the Sigmoid function. This observation and the understanding I have seem to agree.



3. My early stopping strategy was relatively easy. I investigated scheduling different learning rates and then eventually stopping, but that process seemed a bit too intensive for my little understanding of CNNs. Instead, I opted to keep track of the best epoch, and after 5 or 10 epochs of no validation loss gains, I returned the best epoch of the CNN. As far as my actual data goes for this task, I saw that the early stopping and no early stopping CNNs stopped at the same epoch essentially. I found this odd, but it might indicate that the MINST gradient has a minimum that is just slightly lower than what I thought I had seen before. I knew that my early stopping strategy was not to blame here, so I continued with 10 epochs before quitting, which the rest of my algorithms use.

```
regular network: 50 epochs 99.190 % accuracy
early stop network: 20 epochs 99.190 % accuracy
```

4. For data augmentation, I chose some basic transformations like rotations. I was careful not to make my rotations turn into the case where 6 became 9 and vice versa. These augmentations appeared to not do much for the accuracy. Upon further reading, I saw that this was more of a property of the MINST dataset. I ran some mini experiments with the CIFAR-10 set as I was constructing the networks and data pipelines for it, but I forgot to save those. They did show that data augmentation increased test accuracy. I think it is obvious though that without data augmentation, it is more likely that the CNN is just memorizing the data set than recognizing a pattern.

```
regular network: 18 epochs 99.080 % accuracy
aug stop network: 37 epochs 99.080 % accuracy
```

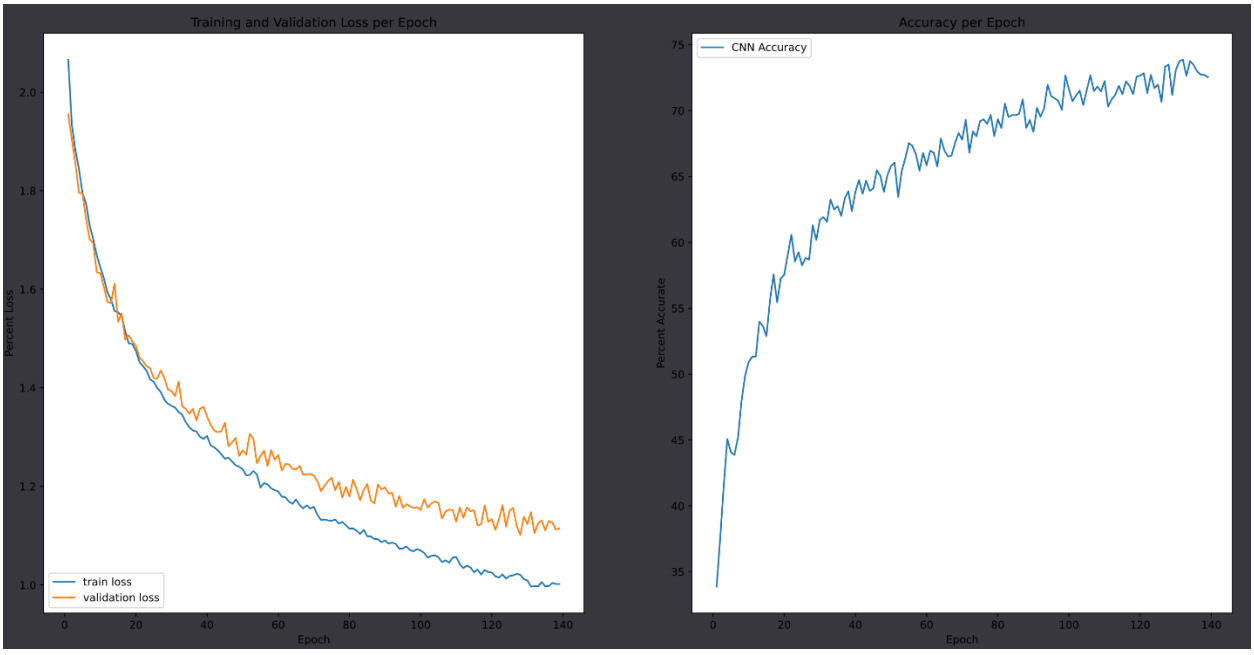
5. Depth versus width was an interesting topic. I read around several forums, GitHub issues, and some abstracts about this. Several times I read that wider networks were more likely to simply memorize the dataset, so people often had to employ dropouts to decrease that skew. Whereas deeper networks were better able to extract patterns rather than memorize them. Sometimes I read that the gradient problem can be more difficult as well as the time complexity if you do not check the dimensionality well enough. As far as my experiment with it on the MINST dataset, I saw no difference between the accuracies. I do believe what the research has to say about wider versus deeper networks, and I will adjust my CIFAR-10 network accordingly.

```
wideular network: 48 epochs 97.500 % accuracy
deep stop network: 49 epochs 97.500 % accuracy
```

CIFAR 10 Results

I am excited that I managed to get a 70-75% accuracy range with my own personal network, and this is the first time I have ever worked with machine learning. I saw online that several people had published networks that get to 90-something percent accuracy, and I decided to stay on my own because this is genuinely interesting. Unfortunately, I have a concern with grading our networks' performance to each other when people can literally copy code from someone's notebook, refactor it a little, and then compete for the performance credit.

My neural network uses 4 convolution layers, 2 max pooling layers, and 4 fully connected layers. I decided to use the SELU activation function because it insanely increased my epoch speed as opposed to using batch normalization. Because the SELU function itself can help normalize data, it took care of that extra processing. I also used some affines, crops, rotations, and random ordering for the data augmentation. Below is my loss and accuracy graphs for the 140 epochs of training.



I also trained a secondary neural network to try to outperform mine. I ended up using the fancier nn.Sequential syntax for rapid layer changing, but I did not save much of my results during the optimization process to show here. Instead, I finalized the second network and graphed it as well. Refer to the 'net2.ipynb' for additional details. A chief observation I found is that dropout was really not productive for the network. Dropout seemed to worsen convergence time and overall accuracy. My original reason to use it was because I suspected some neurons in the system were 'dead', and I could bring them back to life this way. I ended up not beating my original network but still marginally very close. The calculations per epoch were much faster, and the number of epochs before stopping was about 15% less.