**Lab 7: DIGITS**

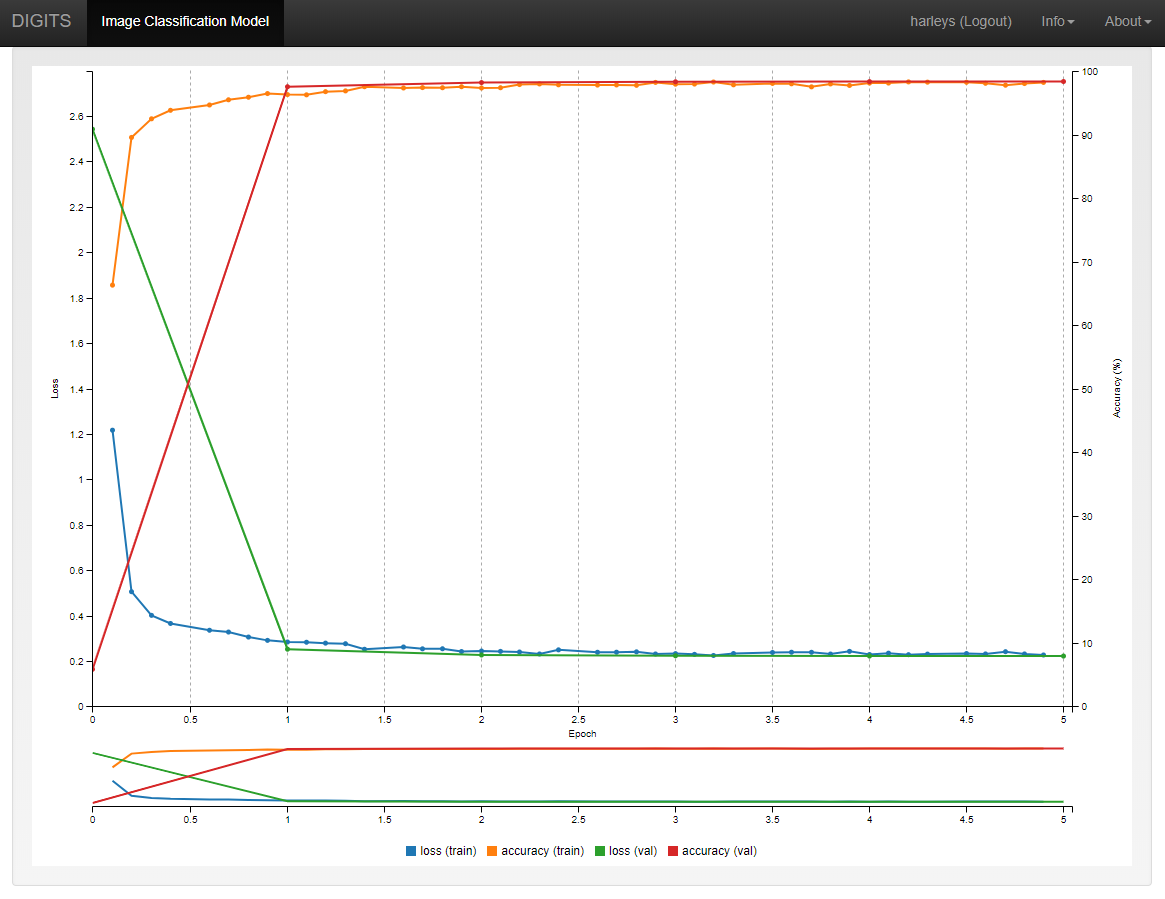
Stuart Harley

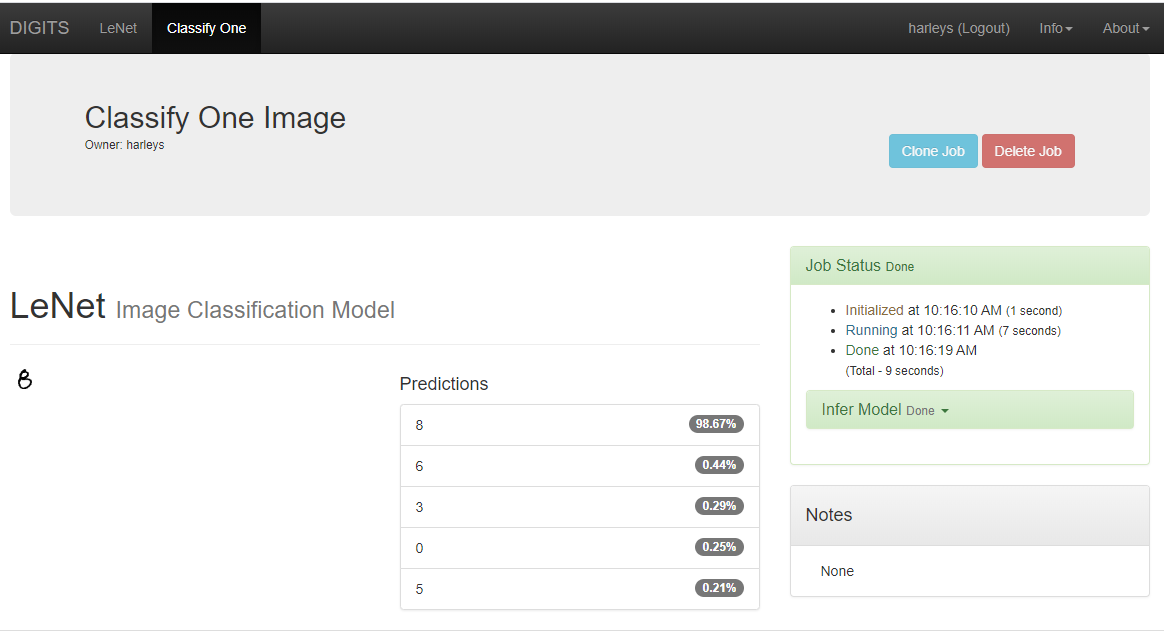
Introduction: In this lab I will be training a Deep Neural Network using DIGITS. I will train and test various models and data sets to evaluate multiple aspects of training a DNN model. I will get some experience adjusting hyperparameters, augmenting data, trying different DNN architectures.

Learning Outcomes:

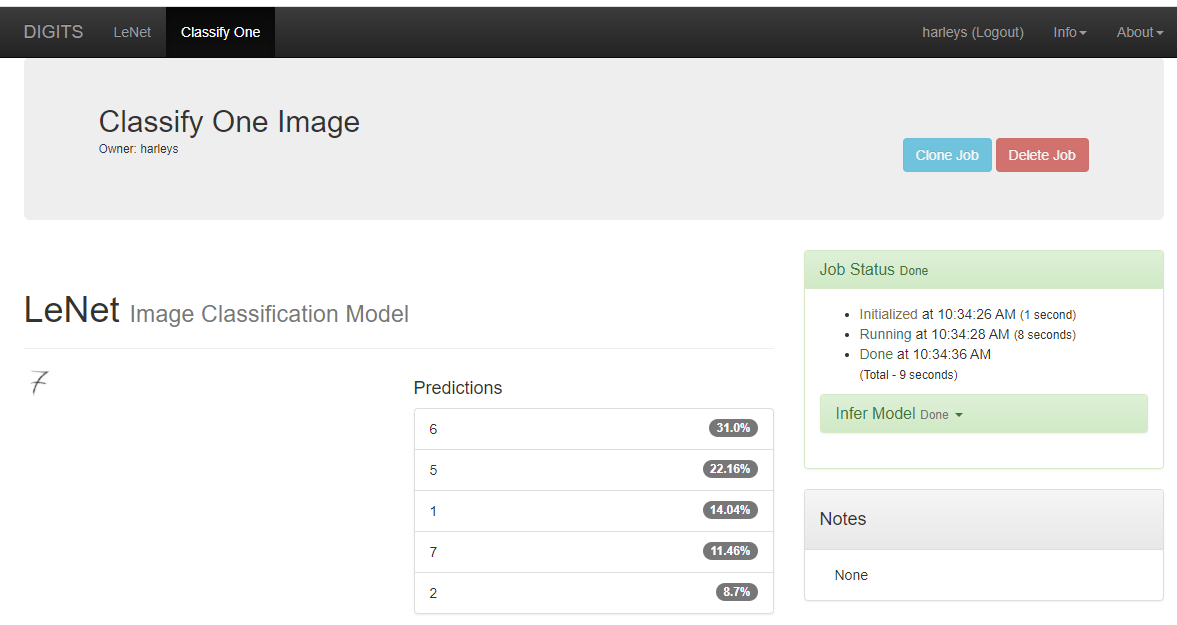
* DNN model training
* DNN model inference
* Hyperparameter tuning
* Data Augmentation Introduction
* Overfitting

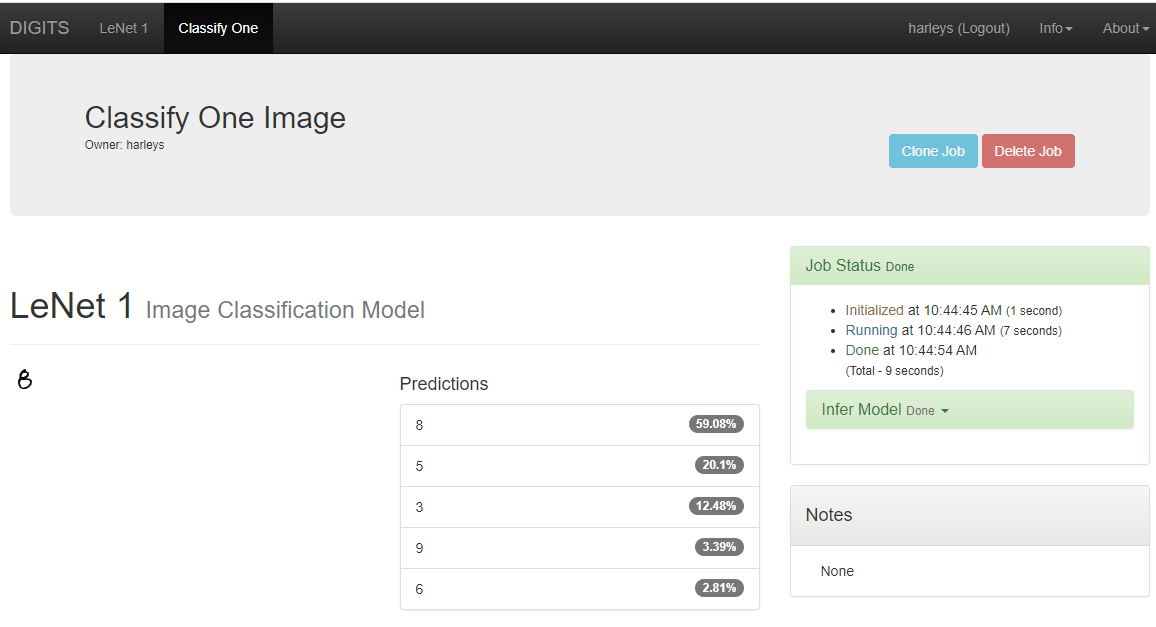
Data: The dataset that I will be using is the MNIST database of handwritten digits. It has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image of 28x28 pixels.

Pictured: First training loss plot of MNIST Data set run for 5 epochs (default batch size).

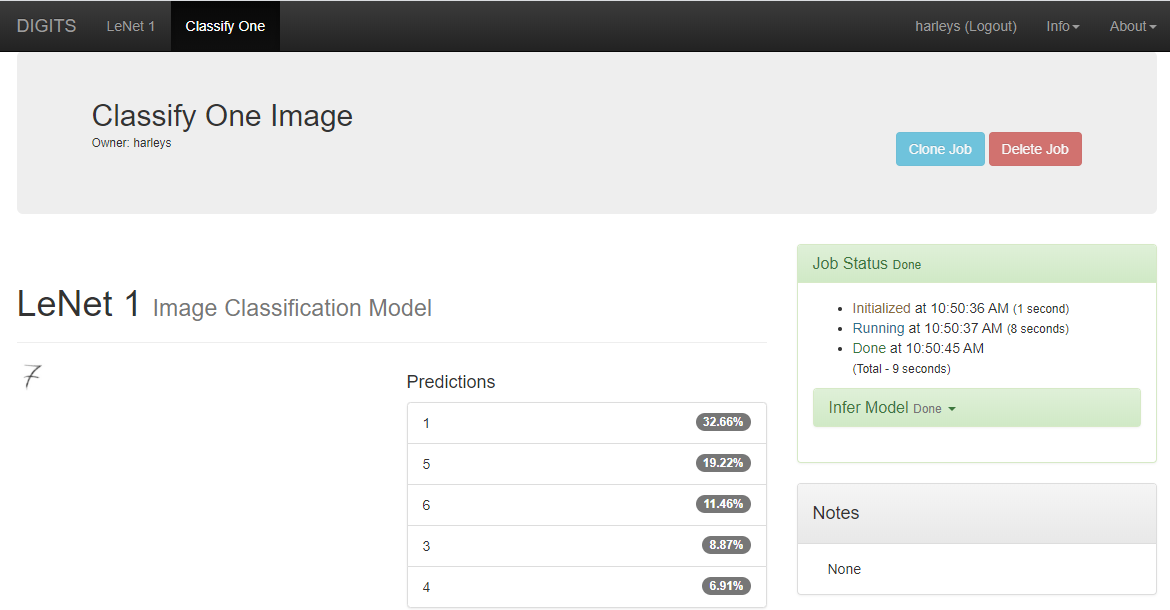
Pictured: Classification of one image of a hand-drawn “8”. As shown, it classifies this as an 8 with 98.67% accuracy. This is correct as the number is an 8. It is not surprising that this number was classified correctly, since this 8 is pretty normal looking so it shouldn’t confuse it with another number.

Pictured: Classification of one image of a hand-drawn “7”. As shown, it incorrectly classifies this as a 6 with 31% accuracy, a 5 with 22% accuracy, a 1 with 14% accuracy, a 7 with only 11.5% accuracy, and a 2 with 9% accuracy. Clearly, the network does not know how to classify this image. I hypothesize that this image was incorrectly classified because of the middle horizontal bar. My guess would be that the network was not trained with any 7’s that had middle bars since not many people draw their 7’s with them. Therefore, it is identifying that middle bar as the middle horizontal bar of a 6 and a 5.

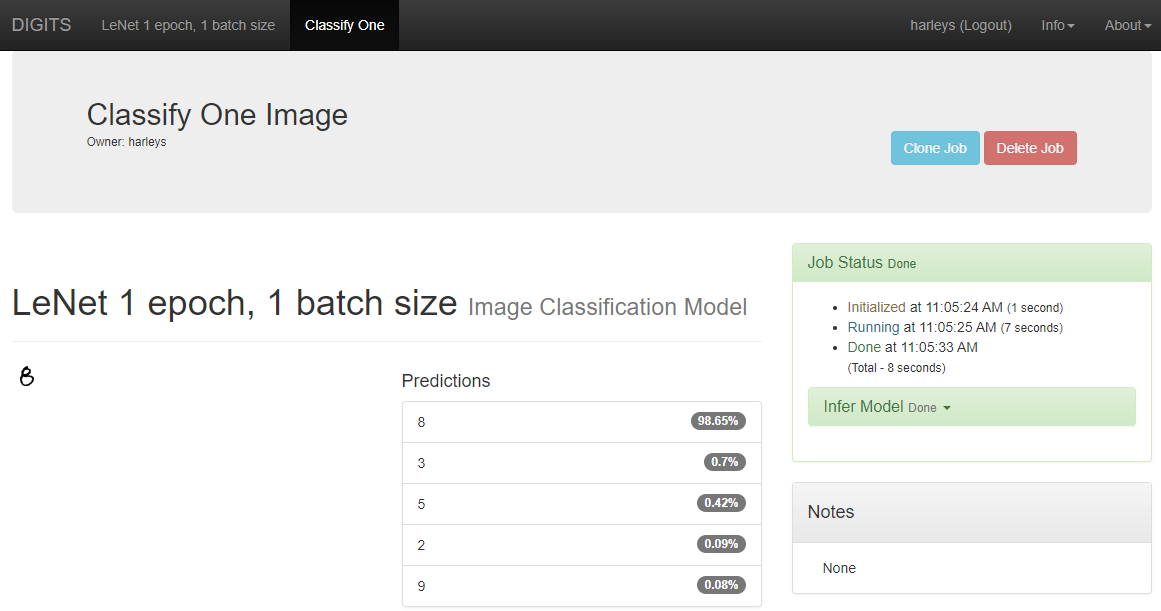


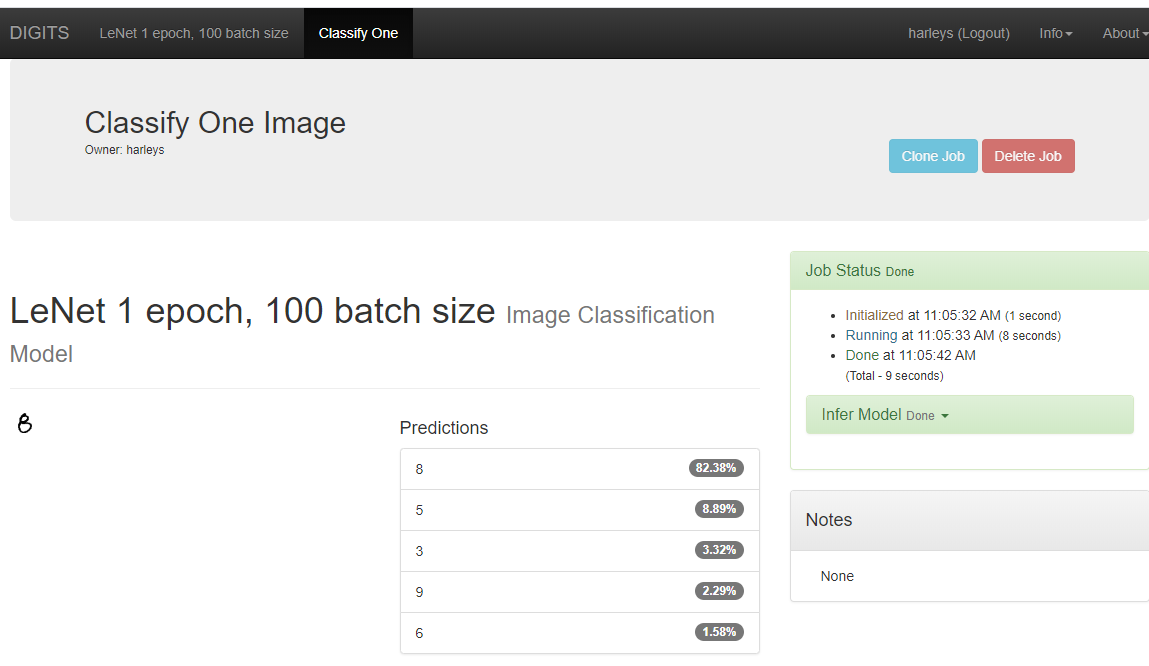
Pictured: Classification of the same image of a hand-drawn “8”. However, this image is classified on a model that only trained for **one epoch**. As shown, it classifies this as an 8 with 59.08% accuracy. This is correct as the number is an 8 however, 59% implies that the model is not actually sure that this is an 8. It is not surprising that this number was classified with only 59% accuracy because it was only trained for 1 epoch.

Pictured: Classification of the same image of a hand-drawn “7” with a horizontal bar. However, this image is also classified on a model that only trained for **one epoch**. This model did not identify this as a 7 for its top 5 possible classifications.



Pictured: Classification of the hand-drawn 8 on 2 models. Each of 1 epoch but one with a batch size of 1 and one with a batch size of 100.



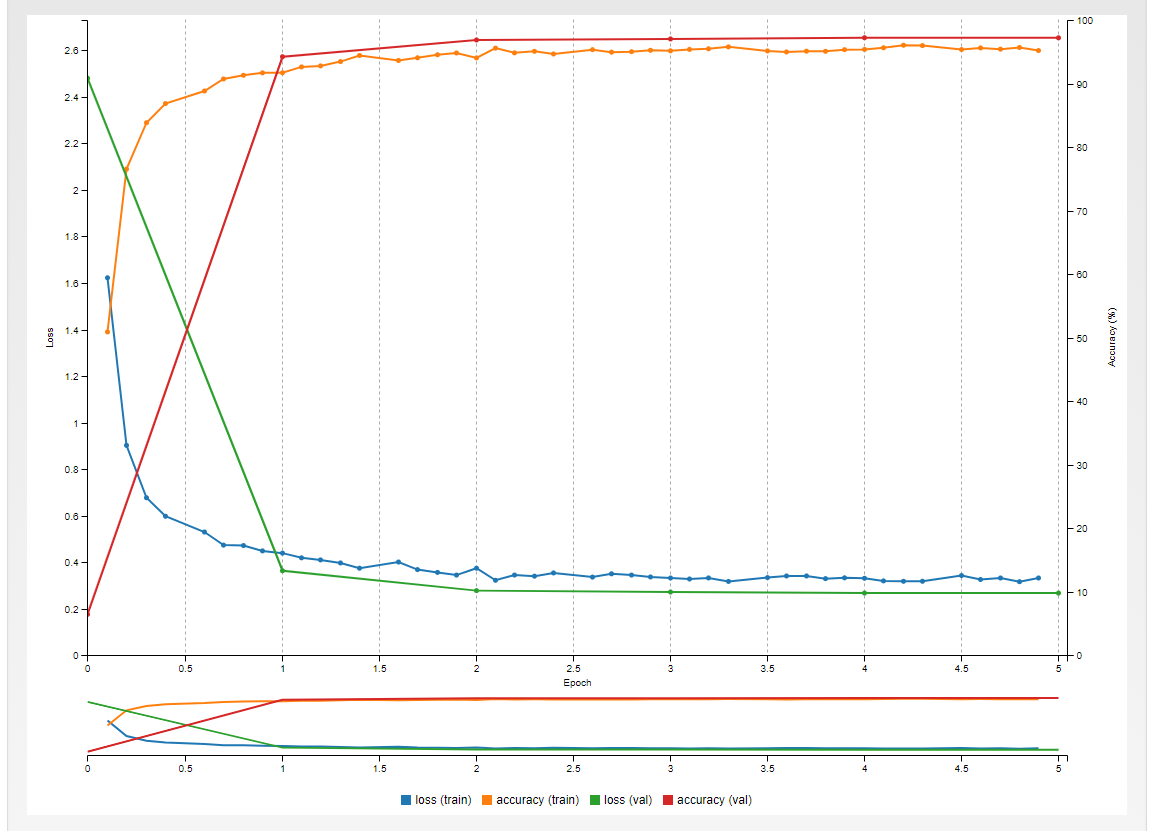


Analysis: When the batch size is lower, the number of epochs needed to train is lower because the model readjusts the weights more times so there is less averaging, and higher precision. When the batch size is higher, the number of epochs needed to train is higher because the model readjusts the weights less regularly, so there is more averaging, less precision, and a slower convergence.

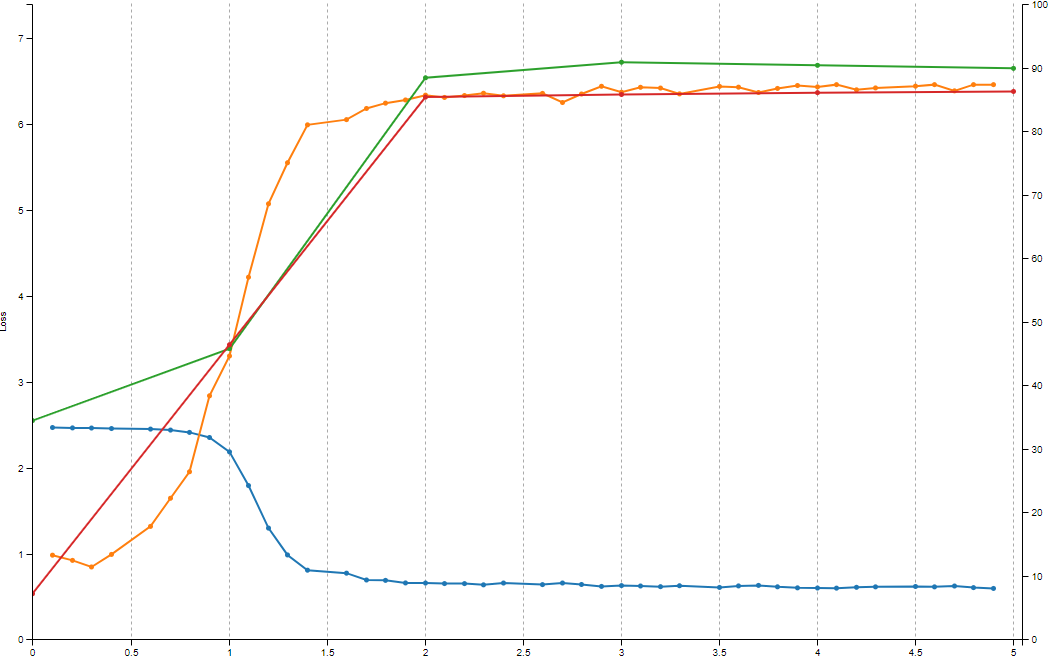
The resulting effect on accuracy is that with a higher batch size, the model will need more epochs in order to achieve the same accuracy as the model with the smaller batch size. If the models are run for the same number of epochs but different batch sizes, as shown above, then the model with the lower batch size will have a higher accuracy.

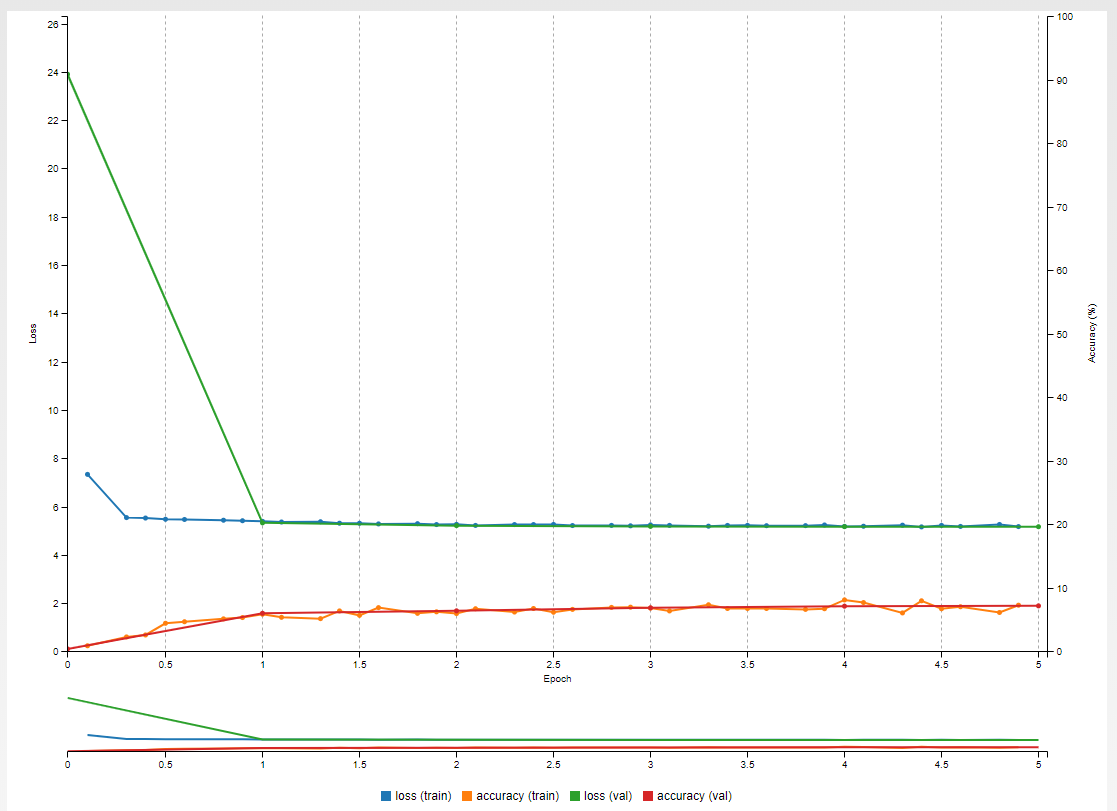
In terms of GPU utilization, the model with the 100-batch size sat at around 22% GPU utilization when running. The model with the 1 batch size also sat at around 22% GPU utilization when running. However, the model with the 1 batch size took much longer to run because it has to back propagate many more times.

Pictured: Training Loss plot for data run with Horizontal and Vertical Flipping data augmentation (5 epochs, default batch size). The classification is not much different from the original model we created, however there is slightly more loss. I imagine that this might come from the 6’s and 9’s, since when they are flipped they would appear the same. Because numbers have a specific vertical/horizontal orientation, being able to identify numbers upside down or backwards would not help identify more “realistic” images, because no sane person writes their numbers backwards or upside down. If there was an option to rotate the images by up to about 45 degrees, this would be more effective because people write their numbers slanted sometimes.



Pictured: Training Loss plot for data run with Whitening augmentation (5 epochs, default batch size). Whitening is described in DIGITS as “per-image whitening by subtracting its own mean, and dividing by its own standard deviation.” The goal of whitening is to make the edges of objects more prominent. However since our images are in black and white and they really isn’t any gray, the edges of our images are already very clear. So whitening actually makes the images less recognizable than the original images so the network has a lower accuracy rate.



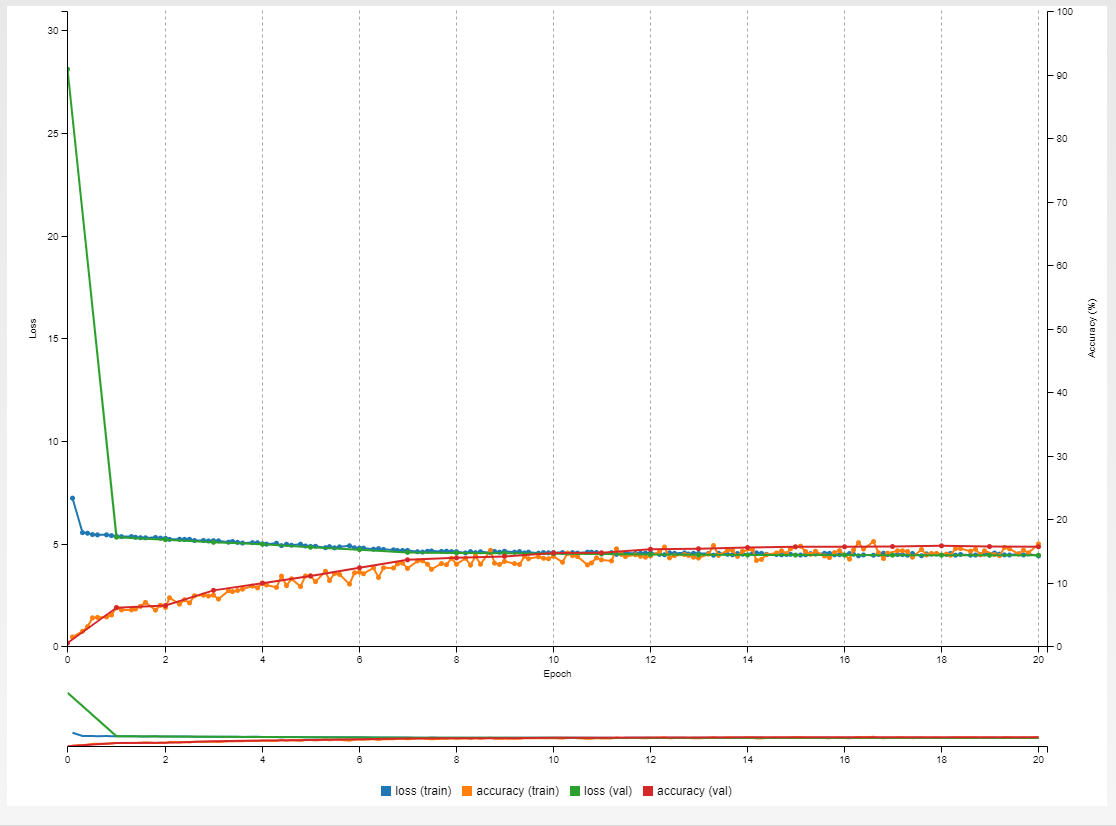
Part 6) Pictured: Training loss plot for a model using the Caltech images with 5 epochs, a batch size of 32 and a learning rate of .001. Time elapsed training: 4 min. Accuracy: <10%.

Pictured: Training loss plot for a model using the Caltech images with 20 epochs, a batch size of 32, and a .001 learning rate. This model took 16 minutes to complete. This was anticipated since I quadrupled the number of epochs, so it also quadrupled the time.

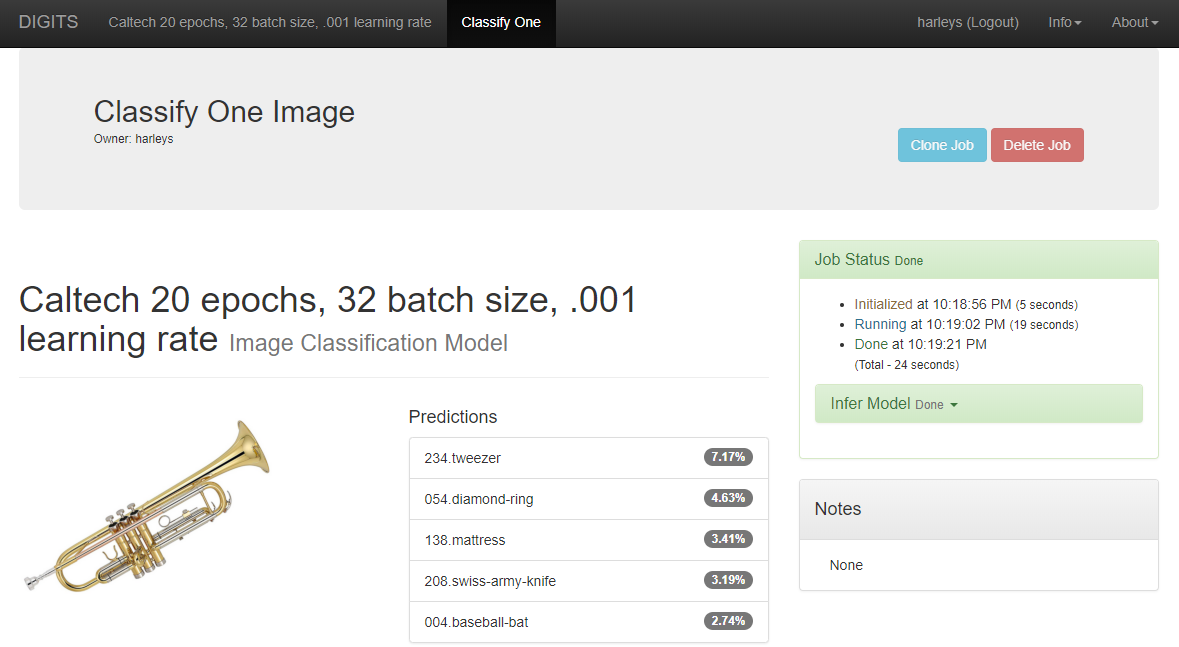
After trying different settings when creating models, this was the most accurate model I was able to create using the Caltech dataset. Horizontal/Vertical flipping made the accuracy slightly worse and loss slightly more. Contrast, noise, and an HSV shift did the same. Whitening made the model really bad, with loss above 40. Increasing the batch size made the model worse, but I believe if I increased the epochs by a proportional amount that the end result would probably be pretty similar. The only thing that I found worked to increase accuracy and decrease loss was to increase the number of epochs.

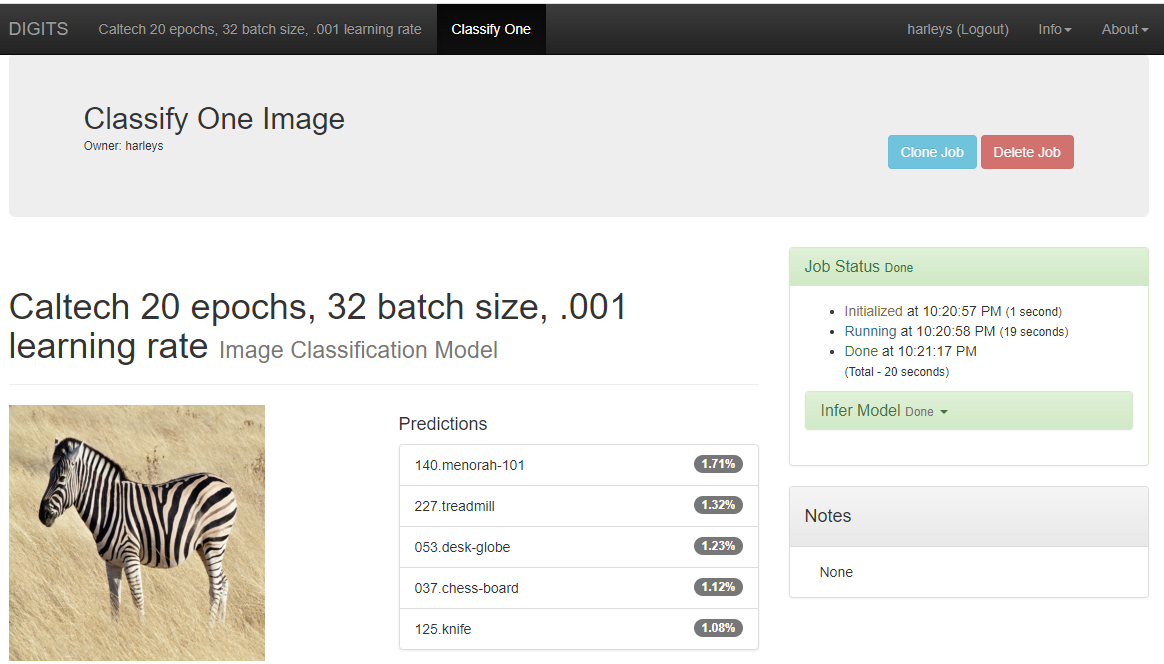
The accuracy after 20 epochs ended up being about 15.7% and the loss was at 4.4. This obviously better than when I ran the same settings just with 5 epochs. I think that this model could still improve more if I increased the number of epochs, but I am not sure by how much. I think this model using the Caltech database is able to improve for longer than the MNIST database because there is much more variance between the images. It is also a larger dataset with many more classifications. This means that the inner workings of the network can be fine tuned for longer and still see effective improvements.

I did not see any overfitting occurring. I assume that if there was overfitting occurring then the training data would have a much higher accuracy rate than the validation data, however the difference between the two on this graph was less than half a percent. I imagine overfitting would have a more significant difference.



Pictured: Classifications of a picture of a trumpet and a picture of a zebra using the Caltech model with 20 epochs.





Neither of these images’ classifications were in the actual dataset, I believe, so is was expected that the model would not be able to classify them correctly. For the trumpet I can understand maybe how it got tweezers, a ring, and a knife, because they are all metallic. They are obviously still very low percentages, but now as low as the zebra. For the zebra I thought it would do better since I know that a horse is a classification in the dataset. Obviously, a zebra has the same general shape as a horse, but I guess the model was confused about the coloring/stripes on the zebra, so it effectively had no idea what it was.

Conclusion: A model is only as good as the dataset it was trained on. If a dataset is given an image comparable to one in its dataset, it will not know how to classify it. There are ways to augment your dataset obviously. And if done correctly it can improve the accuracy of your model. However, the best way to create a good model is to start with a good dataset.