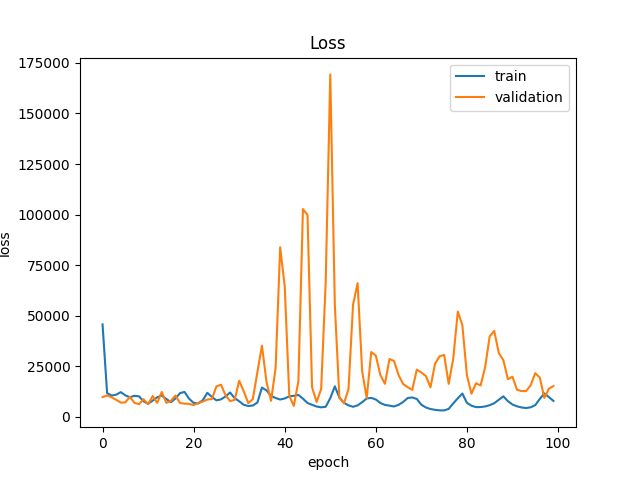
Journal of Experimentation

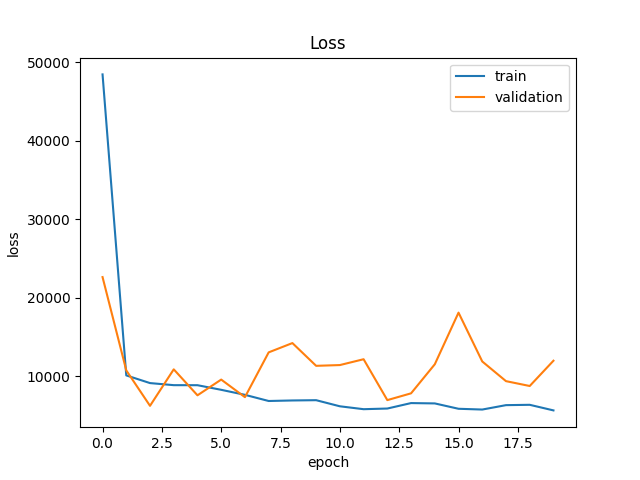
By: C1C Jason Tilley and C2C Kayleb Klapp

1. We adjusted our code to allow for the preprocessing from binary files from Kayleb’s previous project. This made for easy processing and randomization of the process bagged files. This required us to rewrite almost all the data generation code, and preprocessing code to allow for the new formatted files, and lessons learned from the previous milestone on preprocessing. These changes were randomizing the images to allow for better generalization.
2. We then use image score and the image and fed both of these into the model to see performance. We also created a kernel to pass over our image with a height = 0.35 and width =0.04, we also shifted it down by 75 pixels to allow it to focus more on what’s immediately in front of it. We also reduced the dimensions of our dense layer to help the converging.
3. To help the model converge faster, we decreased the image size from 360 \* 280 to 320 \* 160, this did help convergence
4. After further consideration, we decided that heading was not adding valuable information because of its lack in variance and state change. We also reduced down to 3 dense layers as we thought 20 million parameters was overkill. We tried training with only the kernel image to see if that image allows for data that is more valuable. The model didn’t converge as planned.
5. We moved back to not shifting the kernel to see if we can adjust the kernel to allow for lower loss, and flipped all the images on the horizontal axis as well as throttle. This doubled our data size. We also combined two kernels to allow for a more precise area of focused on our altered image. We ran the model for 250 epochs, but it performed best on epoch 29. We ran this on our rover, and although it did not follow the lines, it ran faster and smoother.
6. To fix this we wanted to train on the previous throttle’s frame, because due to a logic error you cannot predict and create a score on the same frame’s throttle. We also threw out about 50% of our data that we determined would not help in the convergence of our model due to sporadic training. Now we make a subdirectory for the flipped data. Model did very good; I think we just need a wider kernel. This one is the best one by the standpoint of smoothness and speed. But it's very obvious it can't see very far or wide.
7. We generate data with noise for both flipped and unflipped images, which also have their own subdirectories. The extra overhead makes our data generator comically slow. In the trainer instead of going through exactly 10 files in each subdirectory, we dynamically allow it to go through all the files, making it more flexible of a preprocessor.
8. We then added a dense layer, and ran it for 250 epochs. This model was over-trained as demonstrated by our rover not performing correct, although smooth in its speed and steering, it did not follow the line correctly. We decided that the flipped data may not be correct/beneficial for the training. We trained a new model on just one orientation of data.
9. We also realized that the remote was altering the throttle, so our previous testing on step 8 was not valid. The new model ran similarly to model 11, smooth and adjusted speed for turns, but did not turn when required. Our final attempt was to add more training data on turns based on if a turn occurred in our training data. This allowed us to weight the turns more heavily.
10. Our last attempt was to over emphasize turns. We did this by replicating the data that showed it was turning more in the csv’s. We have white threshold of 205 per channel, and kernel that emphasizes far away rather than close with two matrix’s added together. We ran it for 40 epochs. We also added flipped images back and and reduced the noise. This is shown on Model 7.

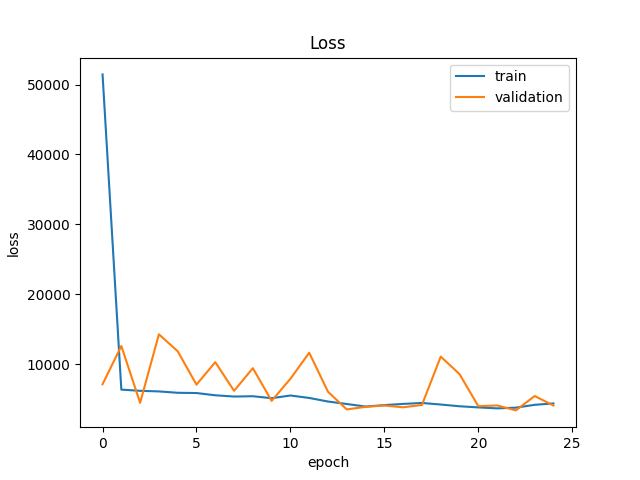
Model 1:



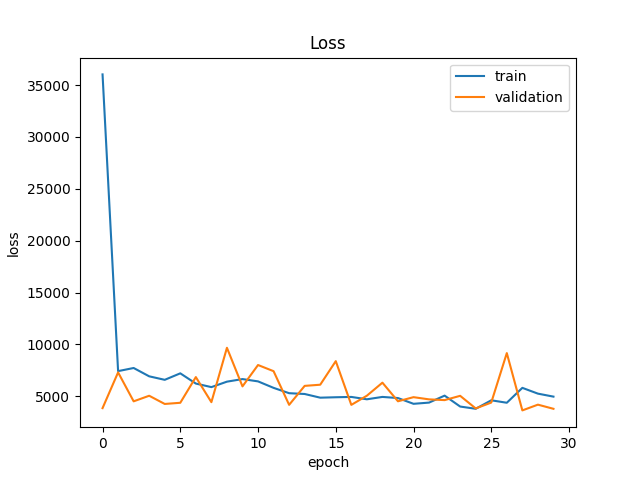
Model 2:



Model 3



Model 4



Model 5

