**Friday, September 15, 2023**

I submitted my research proposal. I am currently researching transfer learning, as it seems extremely similar to the double-stage training I want to implement. With transfer learning, a pretrained neural network is first obtained before certain portions of the network are frozen. With a convolutional network used for image classification, the fully-connected layers which form the classifier and return a one-hot encoded output are usually the ones not frozen. Then, a new fully-connected network is added onto the end of the convolution layers with the appropriate number of layers for the number of classes in the new dataset. The difference here is that the raw outputs from the convolutional layers hold no useful information in themselves. While they are feature extraction layers, similar to the encoder of an autoencoder, the output from a convolution doesn’t tell us anything about what features it is extracting.

To better understand the workflow for transfer learning, I found two libraries which implement transfer learning with a relatively simple API: TensorFlow Model Maker and MediaPipe. TensorFlow Model Maker, upon closer scrutiny, was recently deprecated so I went with MediaPipe. I already have a dataset of images taken from a car driving on a road, so I first sorted my dataset into the format MediaPipe is looking for by finding the nearest label and placing each picture in the dataset into a folder with the corresponding label. MediaPipe uses a nicer data loading scheme that what I normally use myself, so I had to fix some bugs with tensorflow-datasets to make sure my custom data was being parsed properly. Then I started training my model, which took about 30 minutes (I have over 18,000 samples in my dataset). The model had 28% accuracy and a loss of about 2.4 on the test dataset. I will investigate later whether this is actually an issue – if a steering value one bin adjacent to the correct one is predicted, for example, that is usually fine.

**Monday, September 18, 2023**

I started reproducing the results of previous researchers with a variational autoencoder. The simulation used is quite graphically intensive as it simulates the vehicle’s motion in Unity in order to capture photorealistic camera frames for machine learning. The installation is quite complex due to the Conda package manager they use. Although I have Conda installed on my system, it was an older version, so I had to spend some time updating it and making sure the default Python environment was greater than 3.7 because the code needed the modern Python type hinting. I started installing the package about halfway through class, however, my computer blue-screened ten minutes later, corrupting the installation. I spent the rest of class deleting the corrupted files and resetting Conda’s environment variables so that I can try again next week.

**Wednesday, September 20, 2023**

I made some progress with the transfer learning and evaluated the model on my dataset. There were quite a few bugs with MediaPipe accepting OpenCV and Matplotlib image objects. I also had to translate the image labels into the actual numerical steering value. This highlights the limitations of the categorical MediaPipe network – there is no spatial relationship between each category label. Here is a subset of the evaluations on my dataset:

A blue and orange lines

Description automatically generated

I will have to go home to evaluate on my personal computer with a GPU to determine if this is actually a viable driving policy. In the graph, the blue line is the actual steering data while the orange is predicted steering based on the corresponding image of the road. Clearly, it doesn’t match the inputs, but the jitter could possibly be solved through a smoothing filter. I believe using tranfer learning on a model with a single output neuron may help – although a standard CNN with a single output was a complete failure previously, transfer learning may allow the suitable features to be extracted. I found the [Keras Applications API](https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html) which has several pre-trained models such as MobileNet, EfficientNet, and GoogLeNet.