Interim Report 2

By Jade Millan

Work Done

The first thing done after the previous report was to attempt to unscale the data in the data formatting/viewing file. This resulted in a larger final loss reported by the model after training, 0.114. Testing the controller indicated that this was way off and not the right method. Debugging was done to figure out the issue, such as unscaling the model states. Scaling the actions but not the states resulted in poor results as well. Therefore, it appeared the initial belief of the simulation results being off due to data unscaling wasn't right. The default data scaling seemed to be the best option. Attempts were made to modify the layers in the model setup, changing their node count from 256 to 512 to 32. The loss function indicated that 64 nodes were the ideal amount. However, the simulation results were still poor.

The loss function was then examined to determine further issues. The default loss function was mean square error (MSE) and mean absolute error (MAE) and Huber loss were also tested. MAE resulted in the most realistic final loss based on the performance of the model on the test data. Optimizers were also tested, with the default being Adam. AdamW and RMSprop were also tested, and little to no change in the model performance was detected. AdamW was chosen as it proved to being a little more robust and computational expense wasn't an issue.

The next phase of the project included a new model format in mlpfile. The mlpfile was integrated with a final loss of about 0.03. The resulting performance on the test data appeared much improved.

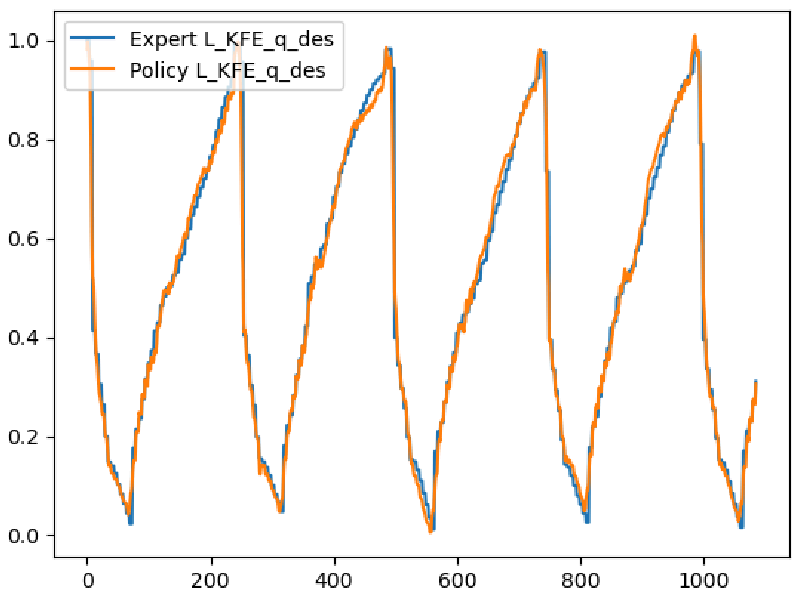

Figure 1: Initial mlpfile Test Data Performance

The simulation results were still quite off after this, so checks were made to ensure that the mlpfile was being saved correctly. Integration of the mlpfile proved a little difficult as integrating it in the simulation and data analysis files required a lot of code rewriting. This included the data being returned as a string in some cases (nearly impossible to work with). A solution to this was to reshape the actions array and ensure it was the right type.

After the mlpfile was fully integrated into the simulation, the next aspect of the project concerned improving the simulation results.

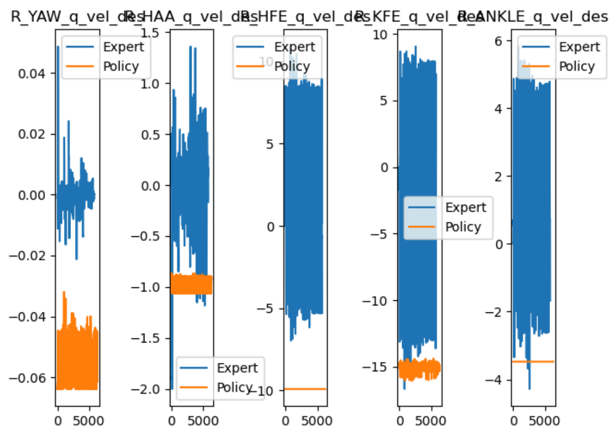

Figure 2: Post-Simulation Results With Poor Pipeline Integration

Some minor tweaks to the simulation setup were made, and the entire pipeline was rerun. This time, the simulation results were significantly improved.

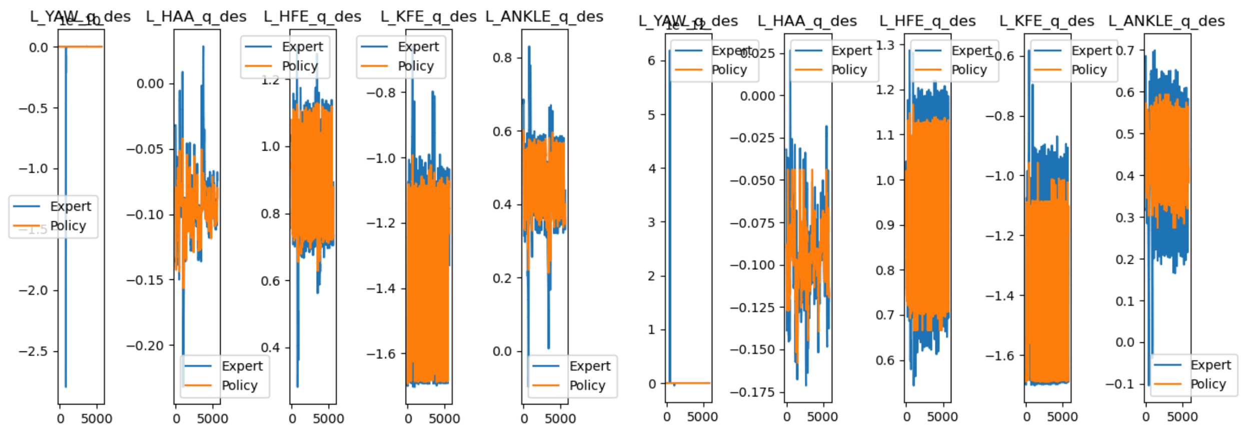

Figure 3: Post-Simulation mlpfile Results (left) Versus Original Results (right)

While not easily quantifiable, these graphs clearly show improvement with a noticeably higher policy convergence on the expert. However, it was observed that restarting the kernel or turning off the computer resulted in the data appearing like Figure 2 again. This shouldn't happen, and indicates a persisting issue. such as implementing early stopping and TensorBoard techniques for the model. With TensorBoard implementation, different amounts of epochs were tested.

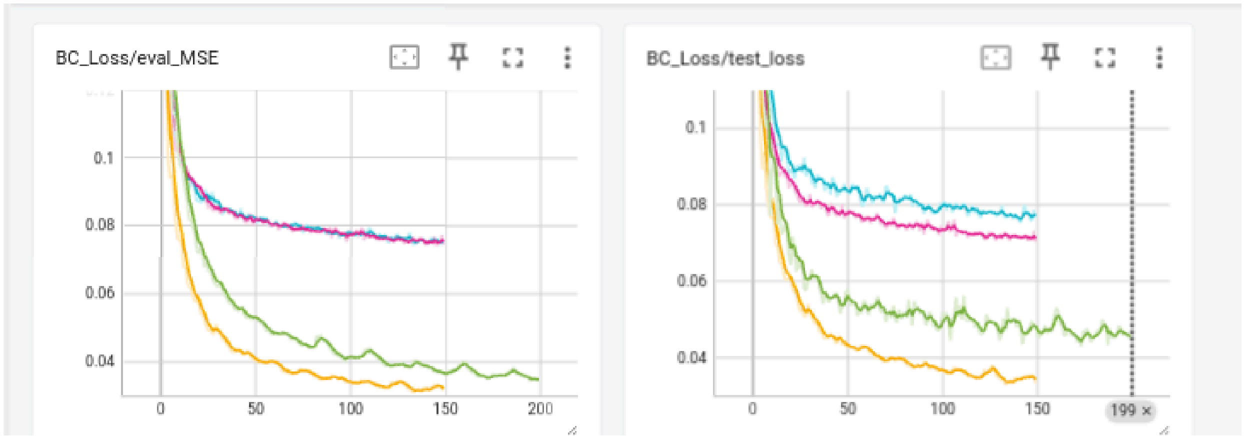

Figure 4: TensorBoard Loss Function Graphs

It was observed that more epochs up to 200 were the most ideal, with the loss function plateauing around ~180 epochs. Retraining the model and running the simulation saw improved training results. However, the post-simulation data was still quite poor. While doing this, it was noted that running the simulation would cause different results in the final data analysis, but this should only happen after the *controller* is run, not the visual part of the simulation in rviz. The final controller results were still very poor after all of this and it's clear the model isn't underfitting or overfitting the training data, a clear indication that the model is generally performing well. This gives a clue to the fundamental issue, the pipeline structure.

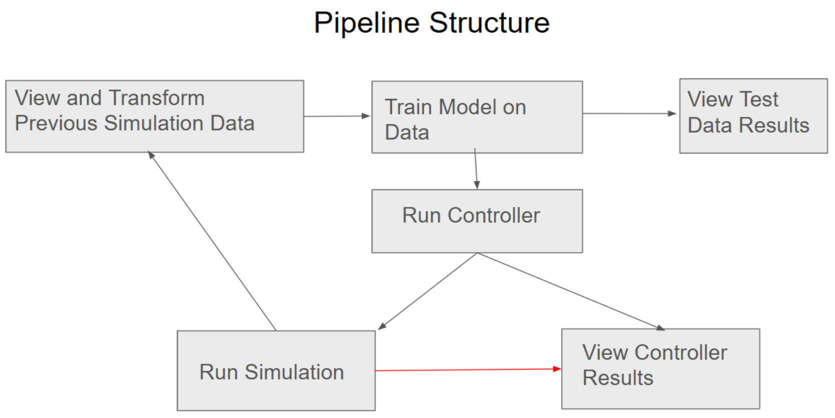

Figure 5: Pipeline Structure

Figure 5 shows the flow and structure of how the data is formatted, trained, and then used by the controller before being demonstrated by the simulation. The red arrow indicates the current happenings, that for some reason viewing the simulation affects or changes the controller results. The controller results should be very consistent as long as the model isn't altered or retrained, but it is very inconsistent and affected by things such as the kernel restarting or other simulation factors. To further examine this, the post-simulation data viewing was improved.

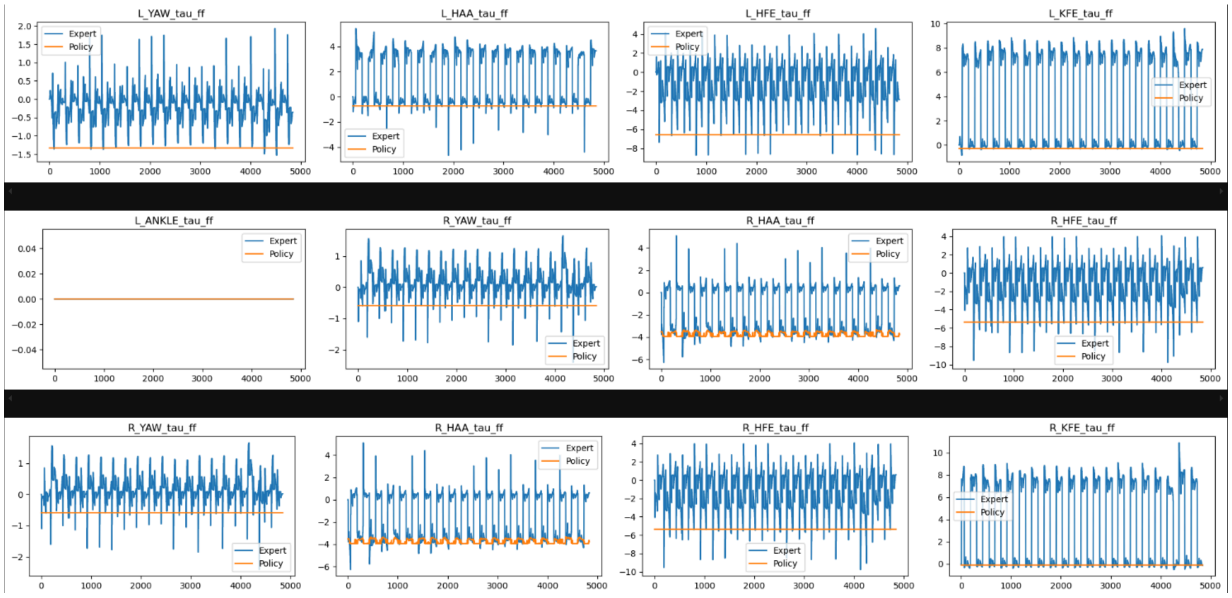

Figure 6: Improved Controller Results Viewing

The clearer look at the controller performance shows little to no action by the policy in most states, a sign of not poor model performance, but a fundamental issue with either the controller, simulation, or pipeline.

Remaining Goals

The most pressing goal for the remainder of the project is to finish correcting the controller performance and get the policy to be run effectively on the robot in simulation so that it is understood it works on hardware. After this is accomplished, the next goal would be to devise a unique behavior such as hurdling to train the robot to do. A tertiary goal would be to design hardware that could actuate the ankles of the robot and make them active.