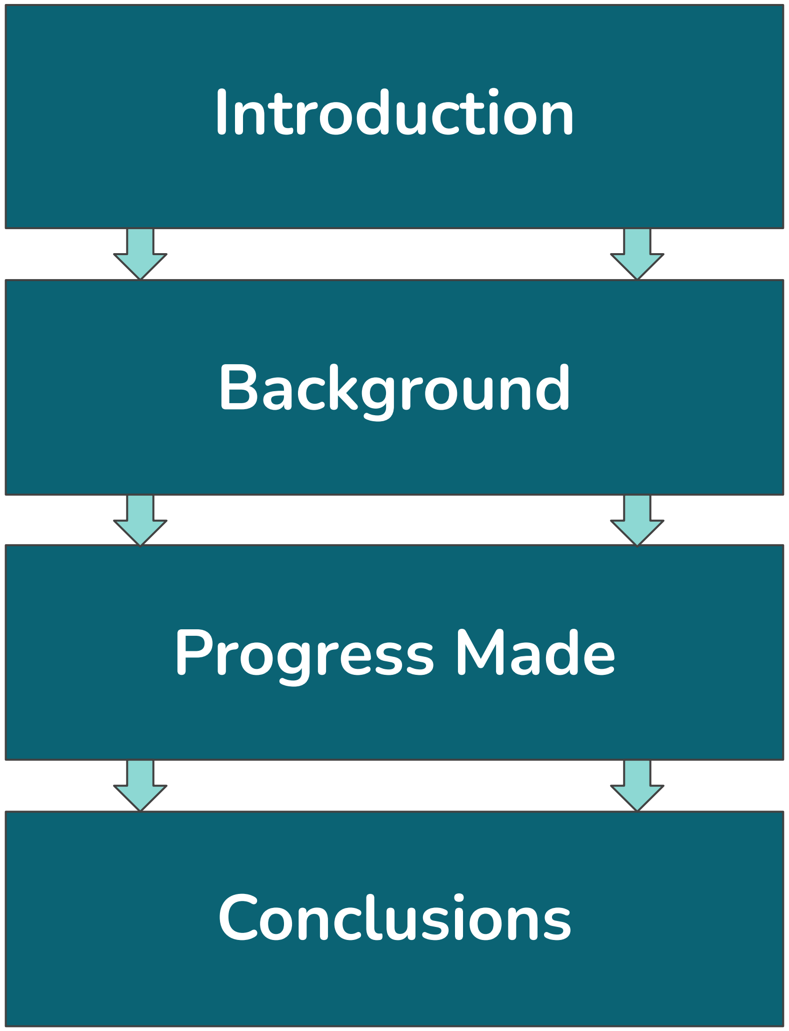
Controlling A Bipedal Robot Via Machine Learning

By Jade Millan

Roadmap



Problem Introduction

- The goal of this project was to better understand how to use machine learning to control a bipedal robot’s walking behavior in a novel direction.
- The novel direction chosen was to use behavior cloning on a a capture-point inverse kinematics system and inverse dynamics system.
- The end goal is to have successful tests in simulation before transferring the model to hardware

Why Legged Robots?

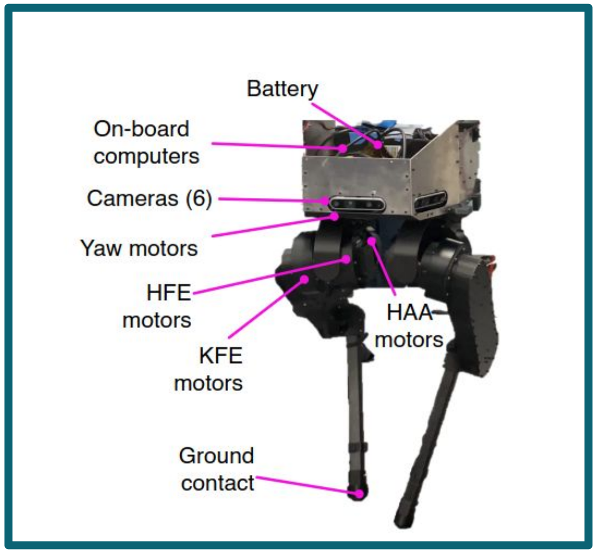
More Robust

Less Capable

Legged robots are currently being used in cave exploration, package delivery, and inspection tasks for offshore power plants!

Hardware and Kinematics

The current hardware is a passive ankle robot focused on leg motion planning and testing learning based controllers



Currently, the robot walks via a capture-point inverse kinematics system and inverse dynamics system. A "capture point" is a desired point for the robot's foot to be placed during the walking cycle.

$$\mathbf{u}_m = -\boldsymbol{\xi}_{m+1}^{\text{des}} + \boldsymbol{\xi}_m e^{\omega T_s}.$$

This system will be referred to as the "expert" for the rest of the presentation

Why Machine Learning (ML)?

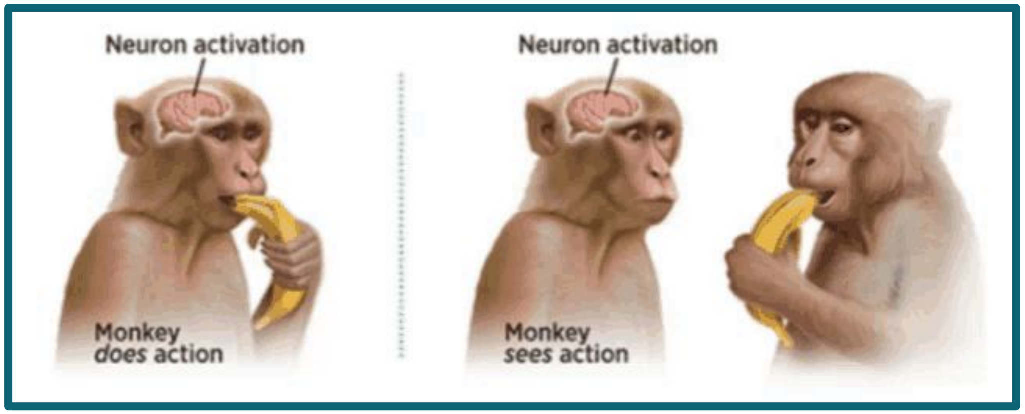
Compared to kinematics, ML offers more adaptability and robustness

- Kinematics works well on a hard, flat, and often known surface
- The adaptability of a learned controller makes traversing rough terrain easier
- Traversing rough terrain is a particular challenge for bipedal robots

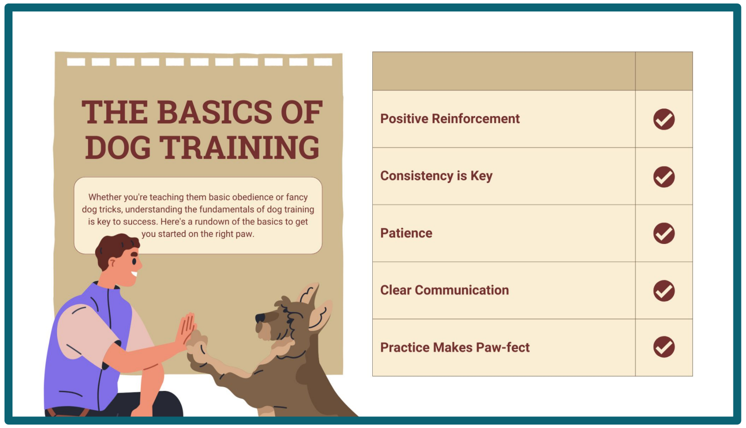
ML Background

The two fundamental types of ML used are Behavior Cloning (BC) and Reinforcement Learning (RL)

Analogy to Behavior Cloning



Analogy to Reinforcement Learning



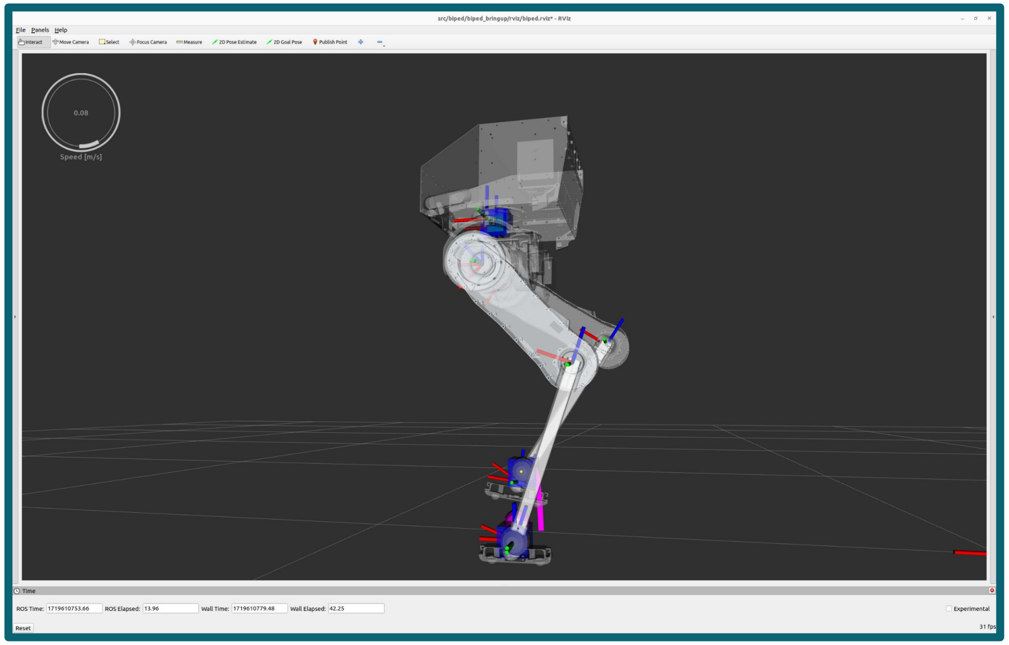
THE BASICS OF DOG TRAINING

Whether you're teaching them basic obedience or fancy dog tricks, understanding the fundamentals of dog training is key to success. Here's a rundown of the basics to get you started on the right paw.

Positive Reinforcement	✓
Consistency is Key	✓
Patience	✓
Clear Communication	✓
Practice Makes Paw-fect	✓

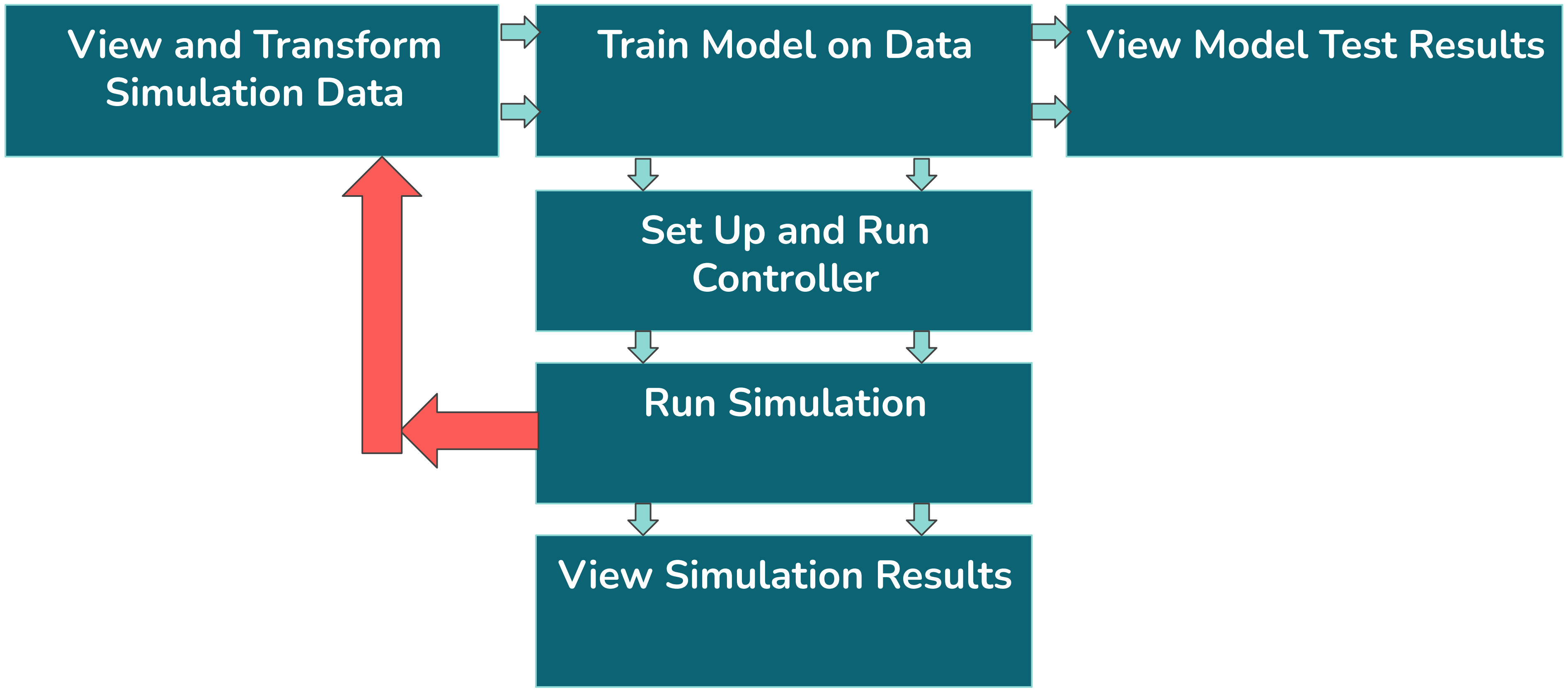
Previous Progress

- Previous project work included setting up the robot simulation and starting a BC model
- The simulation of the robot utilized Mujoco and ROS2, with simulation viewing done in RVIZ



ROS Time: 1719610753.66 ROS Elapsed: 13.96 Wall Time: 1719610779.48 Wall Elapsed: 42.25

Old Pipeline Structure



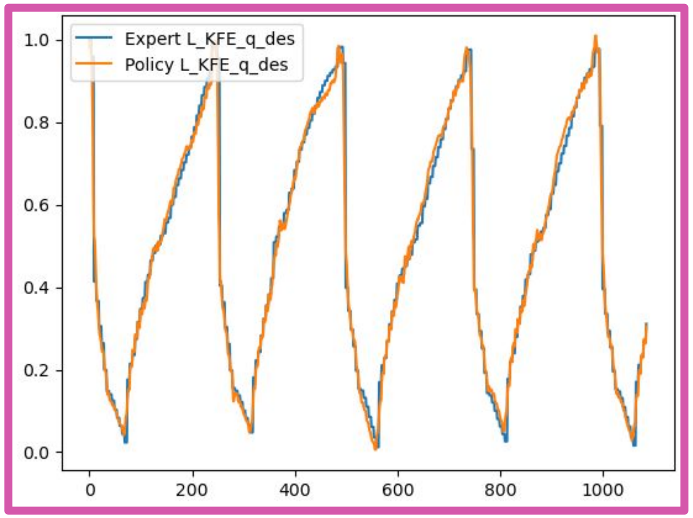
Progress With Behavior Cloning

After setting up (downloading Linux, necessary code, etc), the first step was to verify the integrity of the pipeline

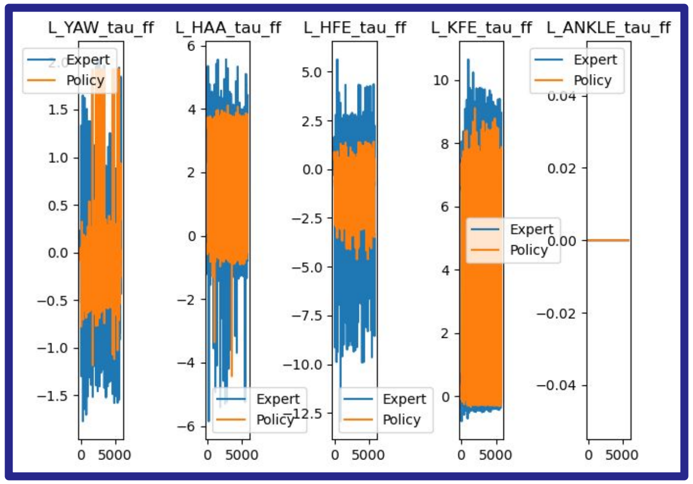
After making a few small changes to get things working, the first test and simulation results were able to be viewed

There are 41 joints or "states", each with their own sets of actions for a given time

The test results were quite good, while the simulation showed convergence issues

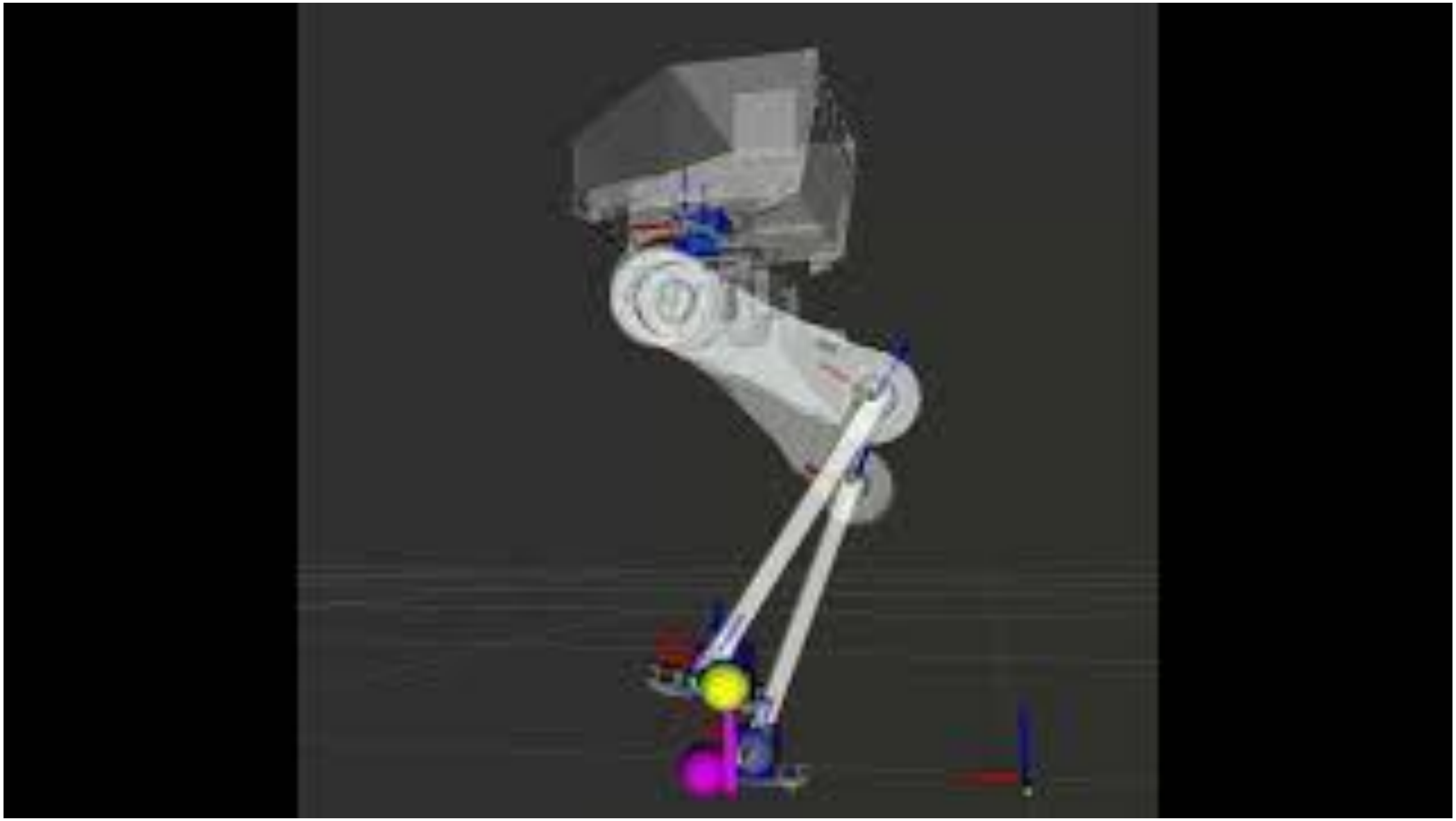


Test results for one joint

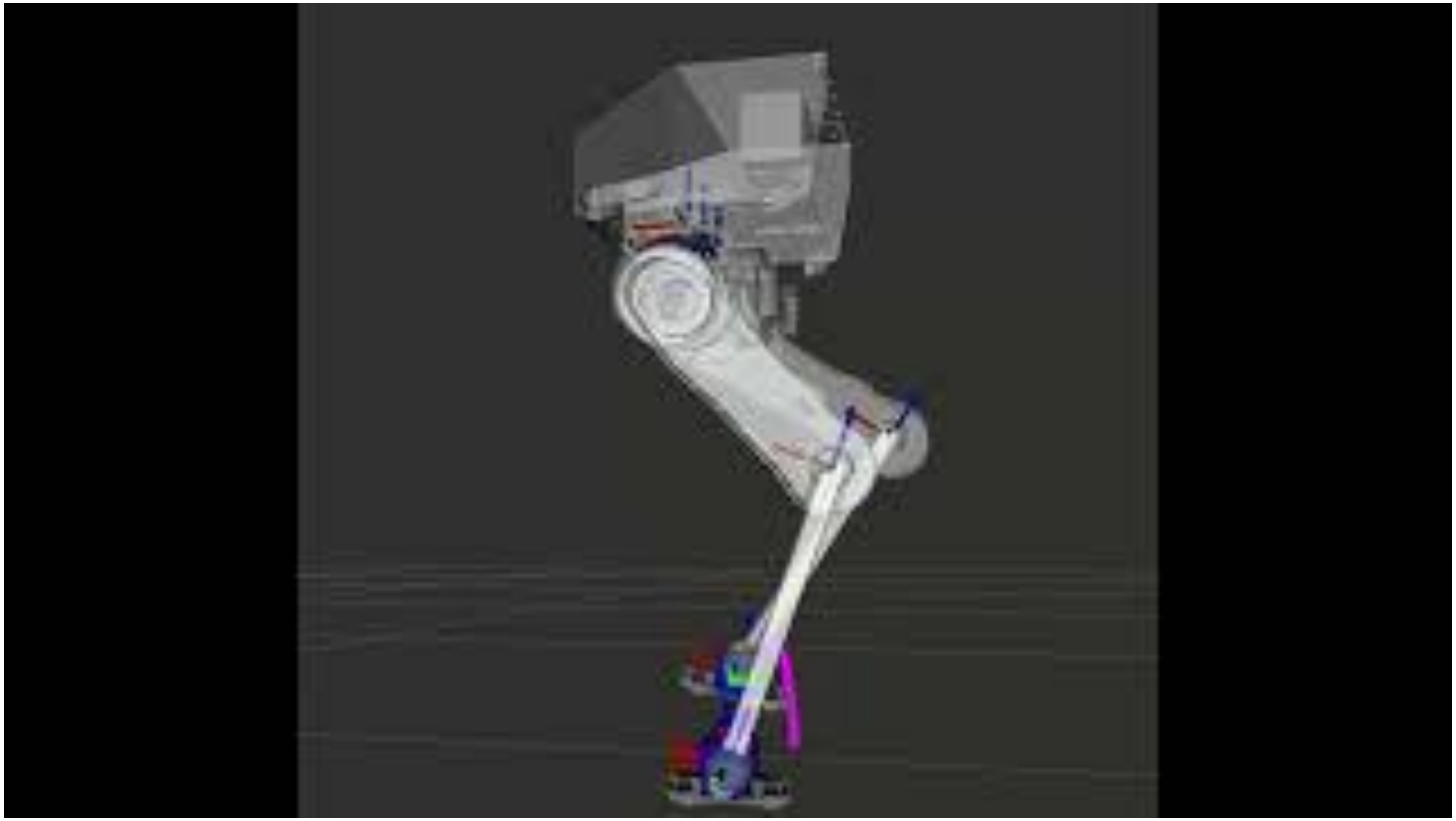


Simulation results for five joints

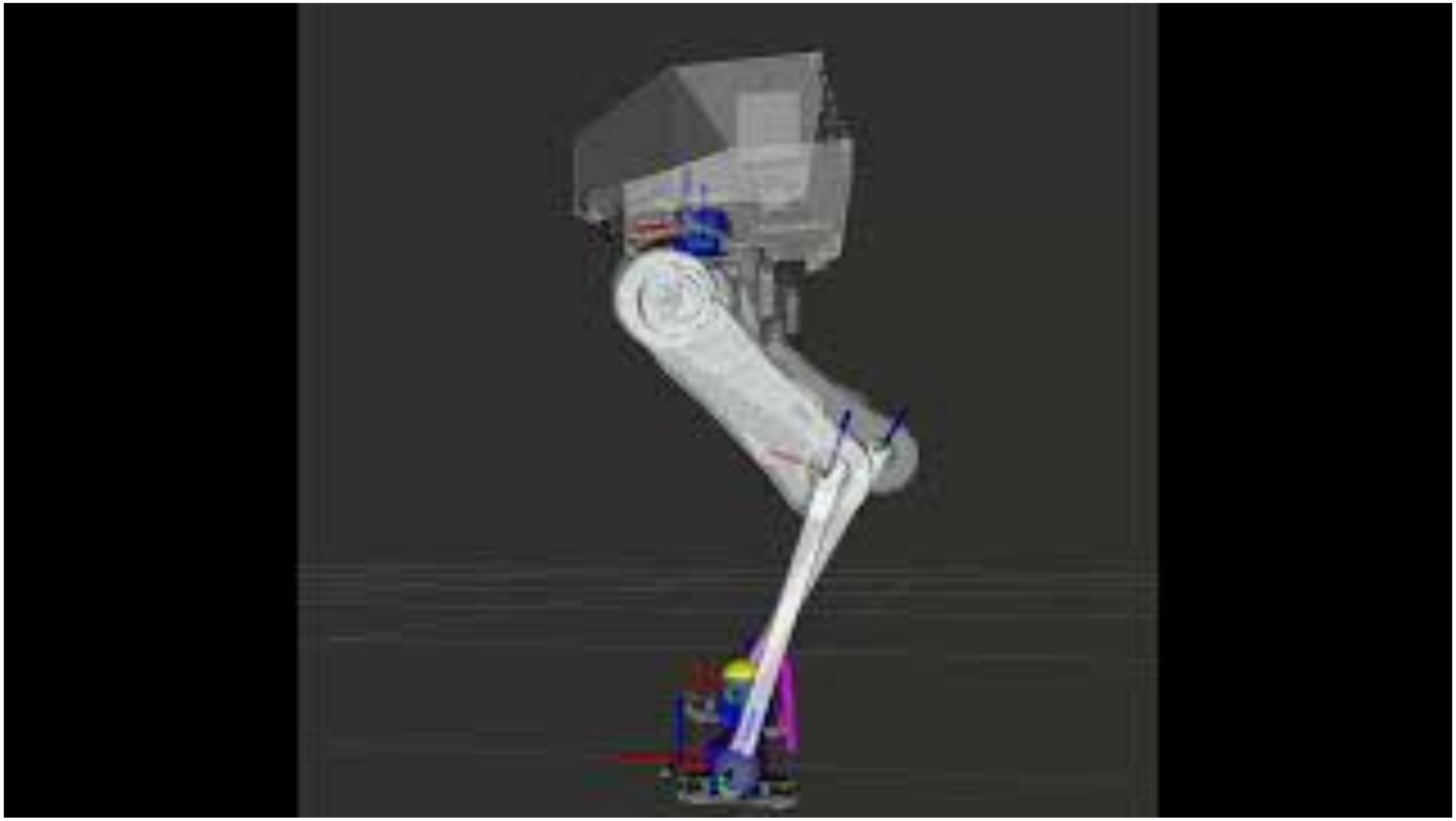
Biped Running With Expert Only



Biped Running With Expert and Policy



Biped “Running” With Only Policy



Major Fixes

Issue: Slow simulation

Fix: Introduce mlpfile format

Issue: Poor model/simulation performance

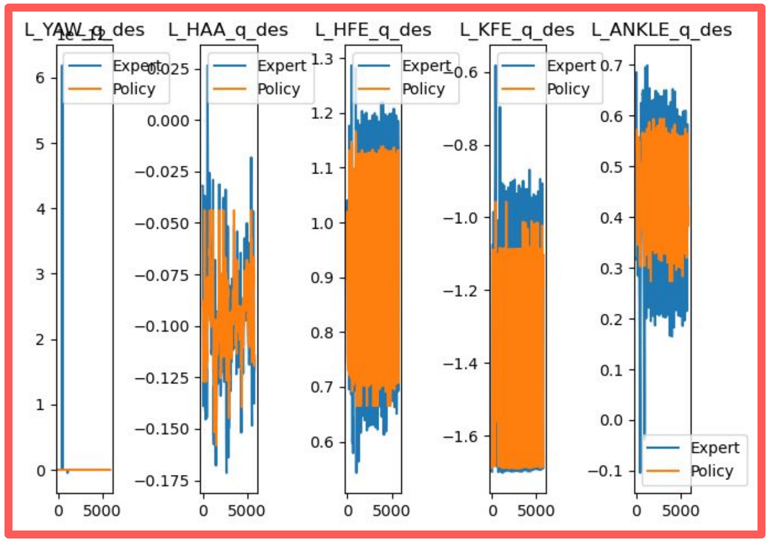
Fix: Use TensorBoard, improved policy architecture, and different model parameters

Issue: Poor Data Visualization

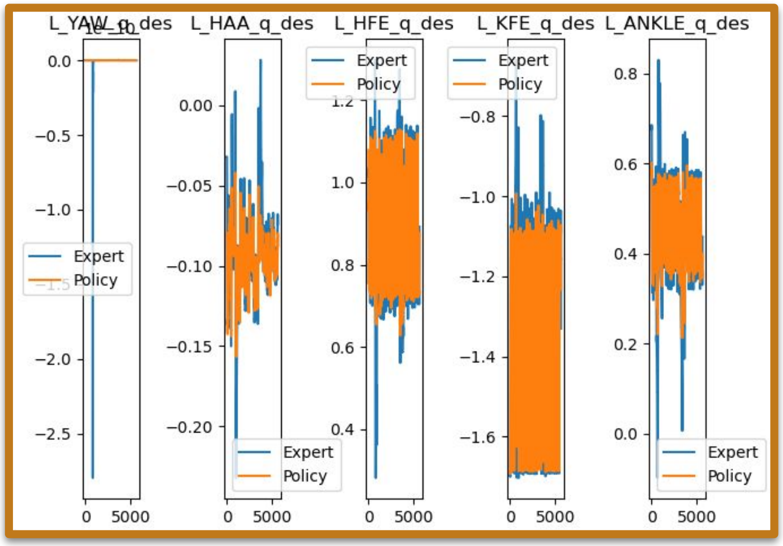
Fix: Change code setting up test data and simulation data graphs

Issue: Running pipeline multiple times without changing any code results in different simulation performances

Fix: Investigate pipeline integrity

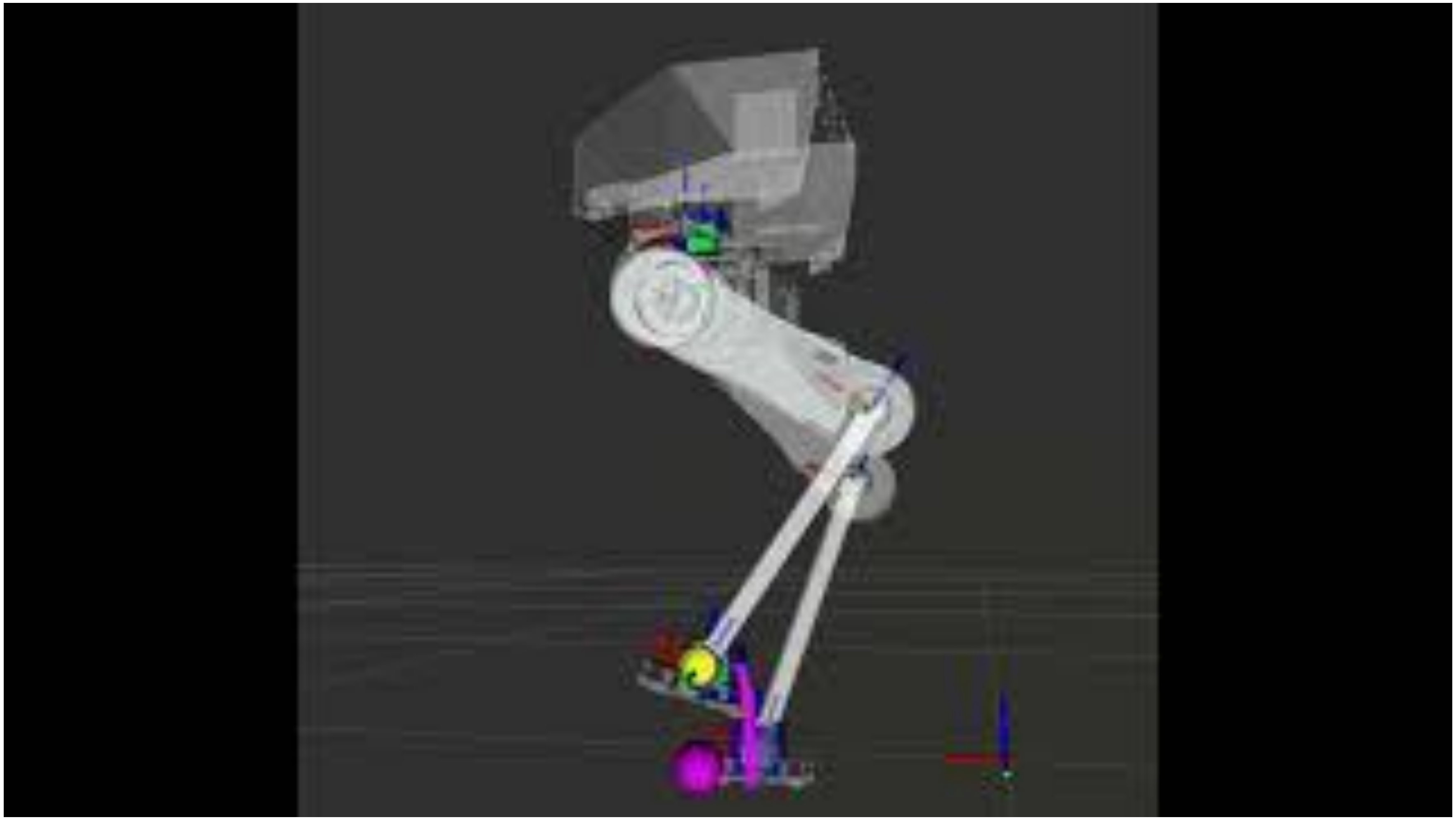


Before mlpfile on old policy format

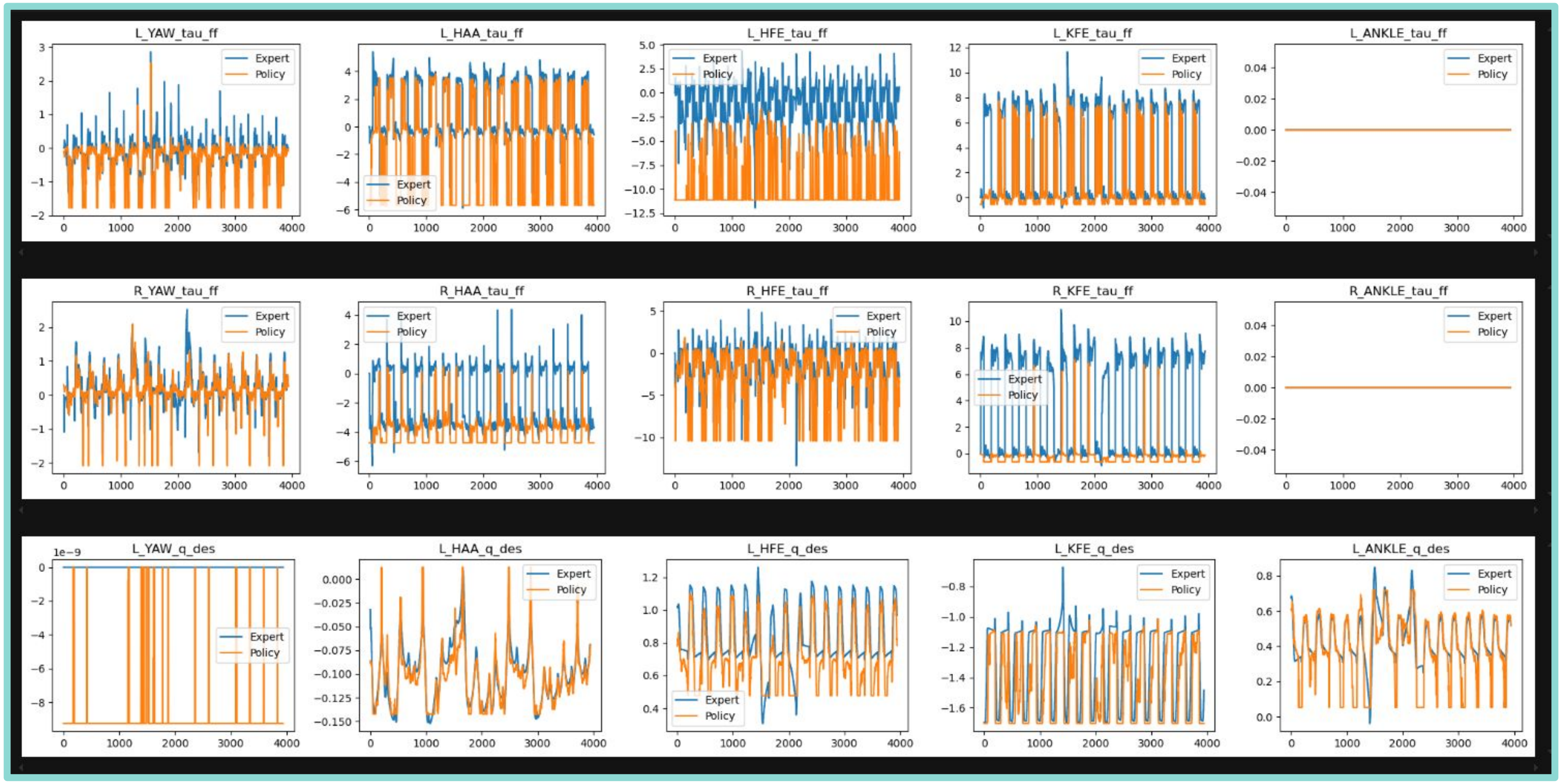


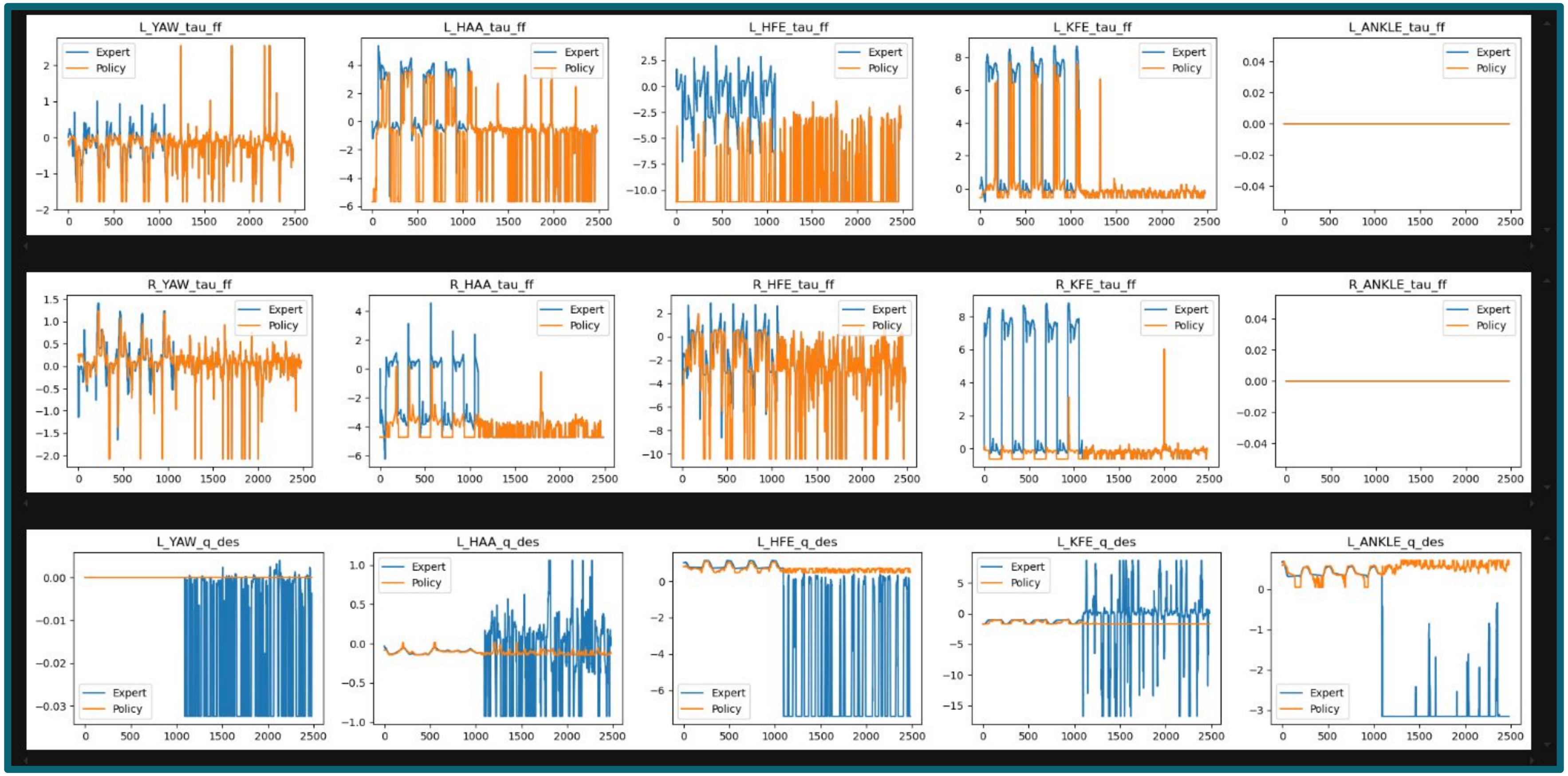
With mlpfile

Simulation With Mlpfile



Improving Data Visualization

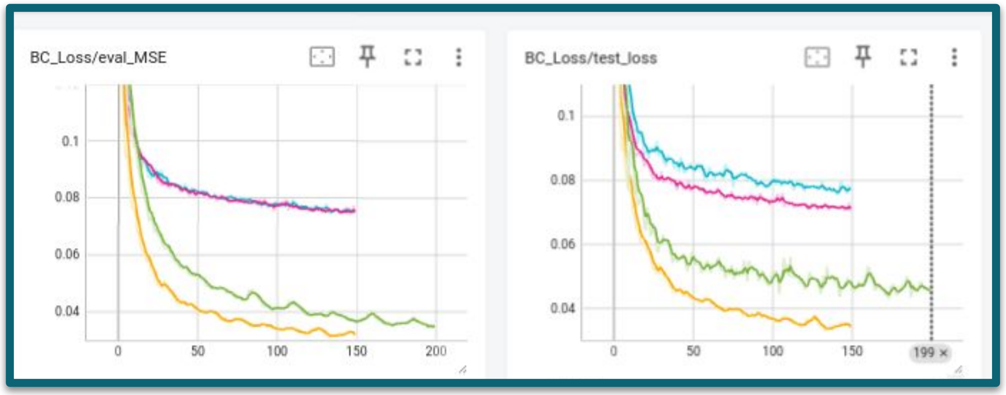




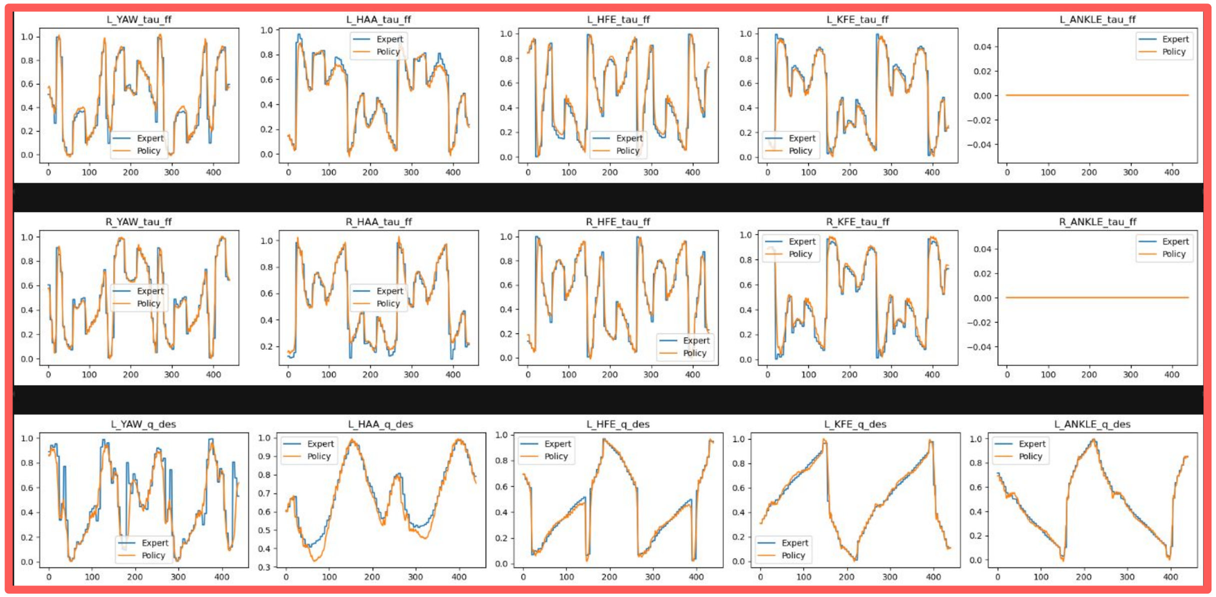
After multiple runs, the model performance gets worse and the robot still crashes, despite not changing any code

Verifying Model Quality

Using TensorBoard, we can visualize the outputs of the loss functions on both the training and test data



Improved test data performance shows mostly great model performance, so what is the issue?



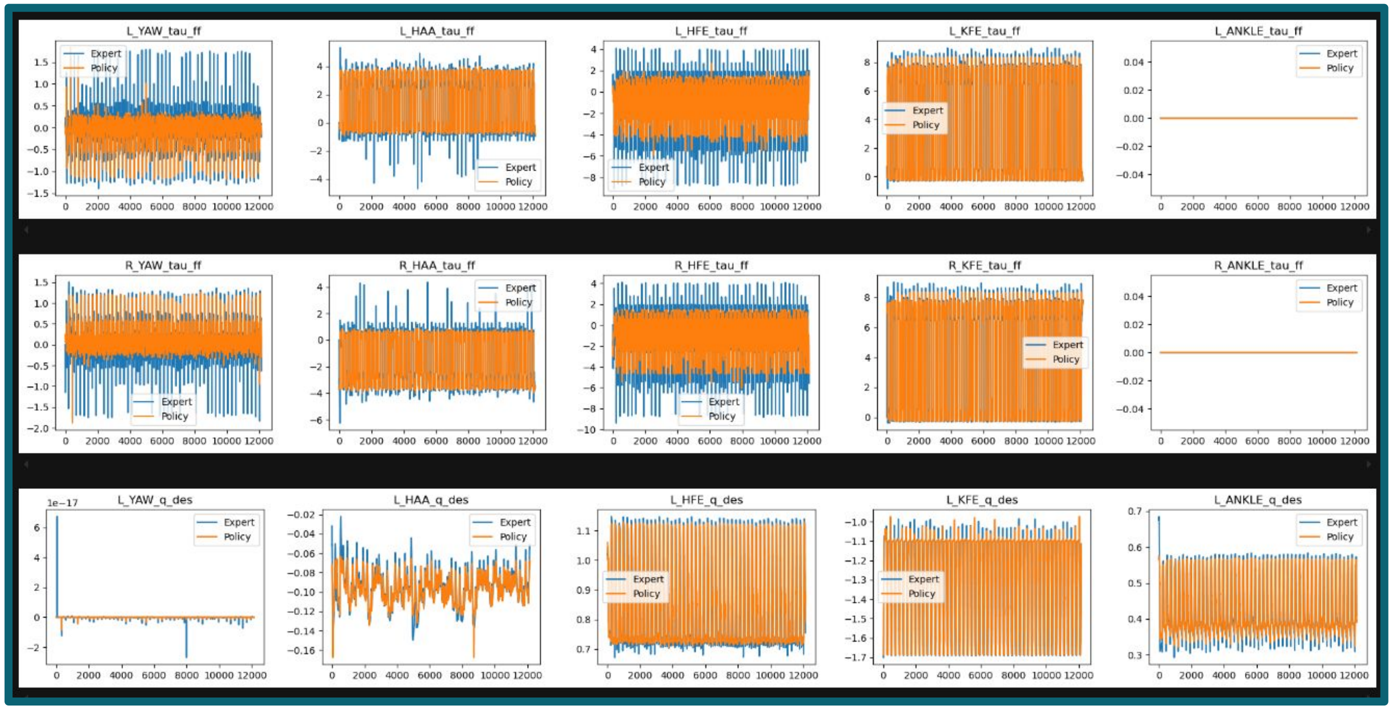
Verify Pipeline Integrity

Idea: Try to see if running the simulation for longer affects the training performance or length.

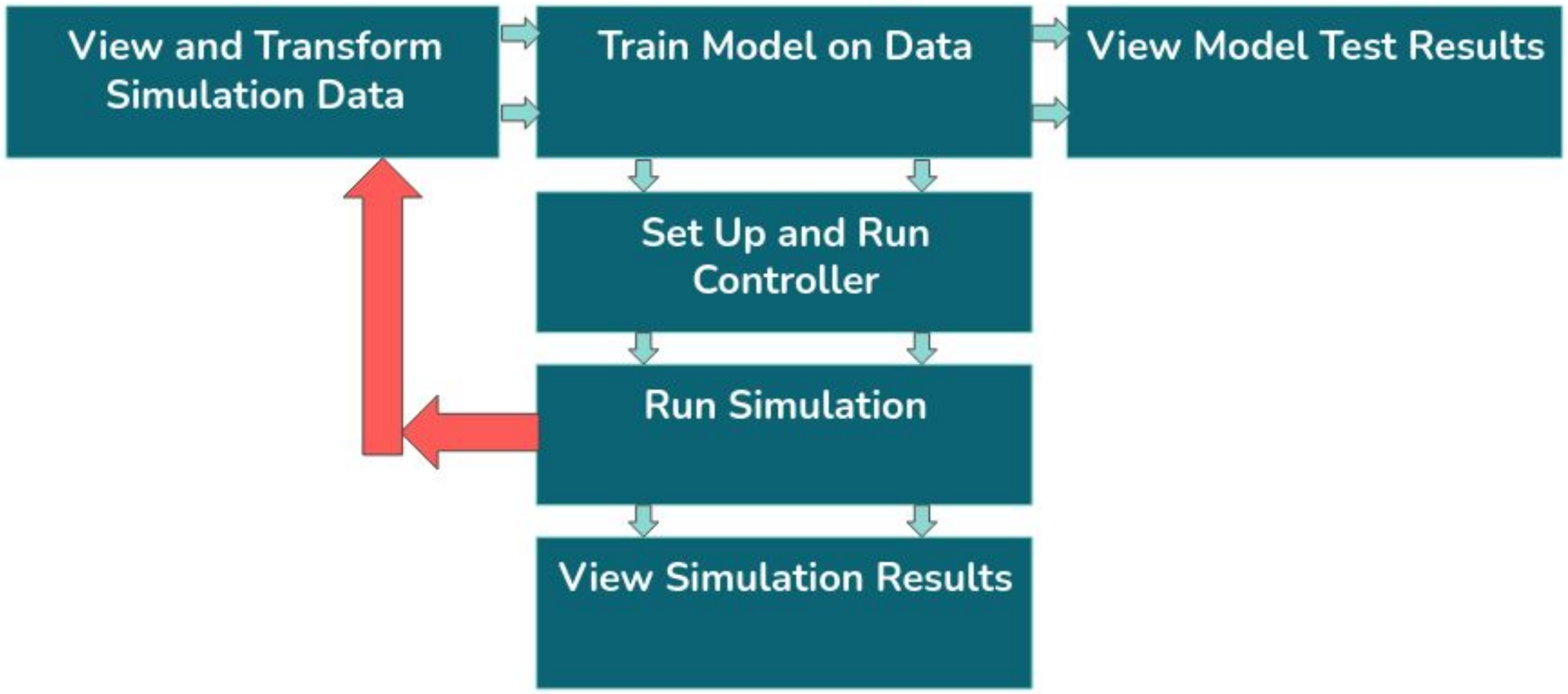
Simulation was run for much longer than usual.

Running the pipeline again resulted in much slower training time.

This shouldn't happen, indicative of a pipeline error!

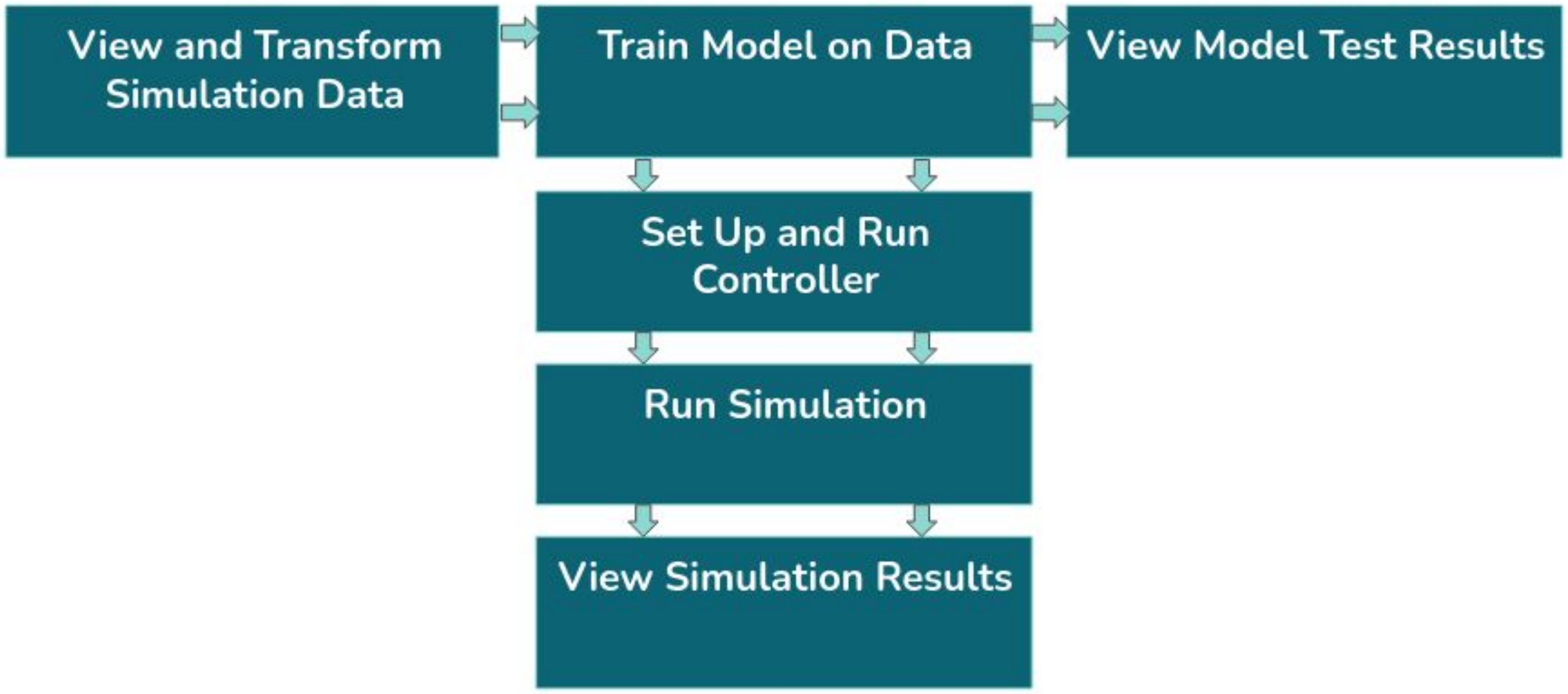


Old Pipeline



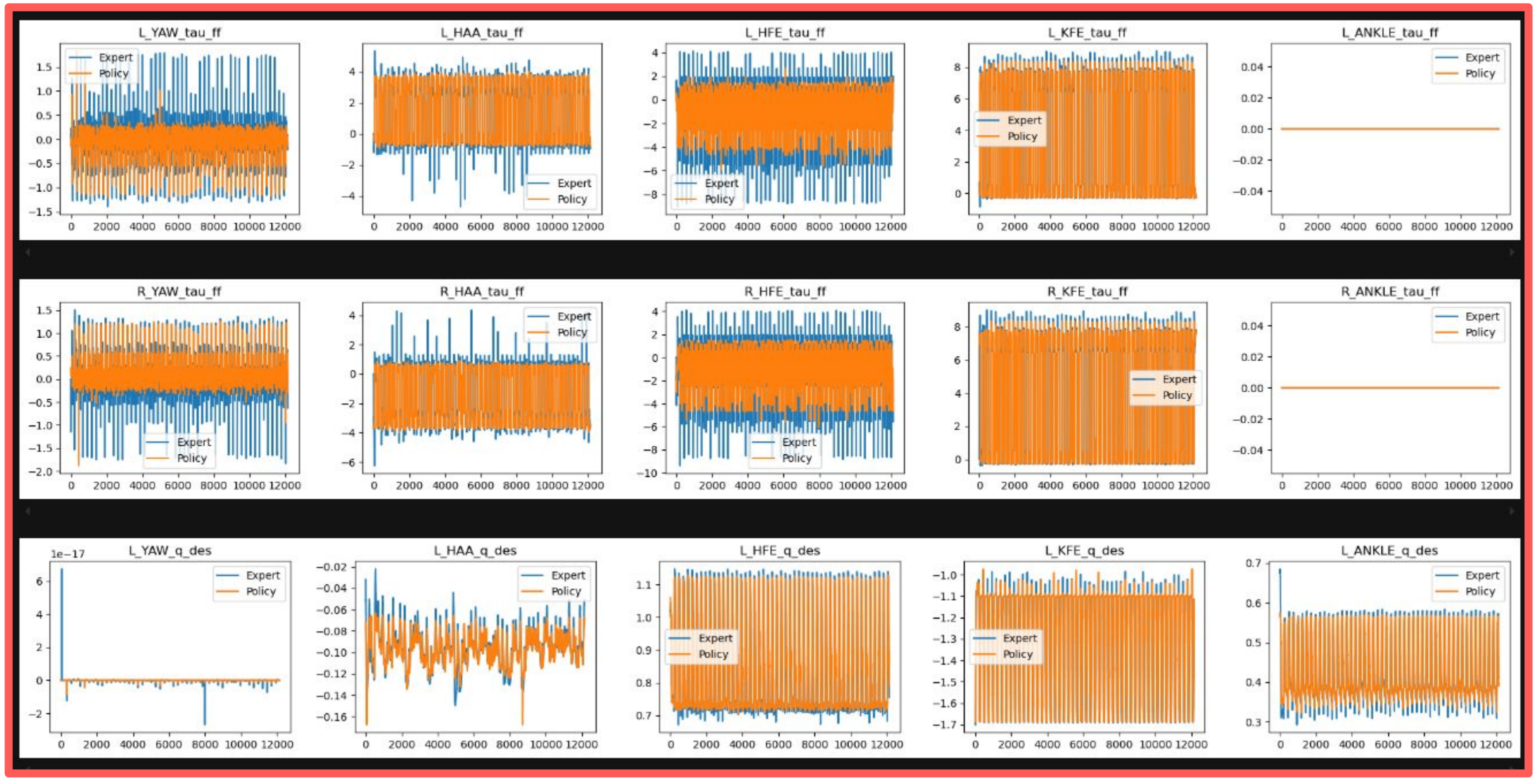
Solution: Remove the red arrows and create a training data file that doesn't get overwritten by the simulation

New Pipeline



Created a new training data file with ~370,000 lines of data, compared to the typical ~7k lines of data typically output by the simulation

Issues Persist

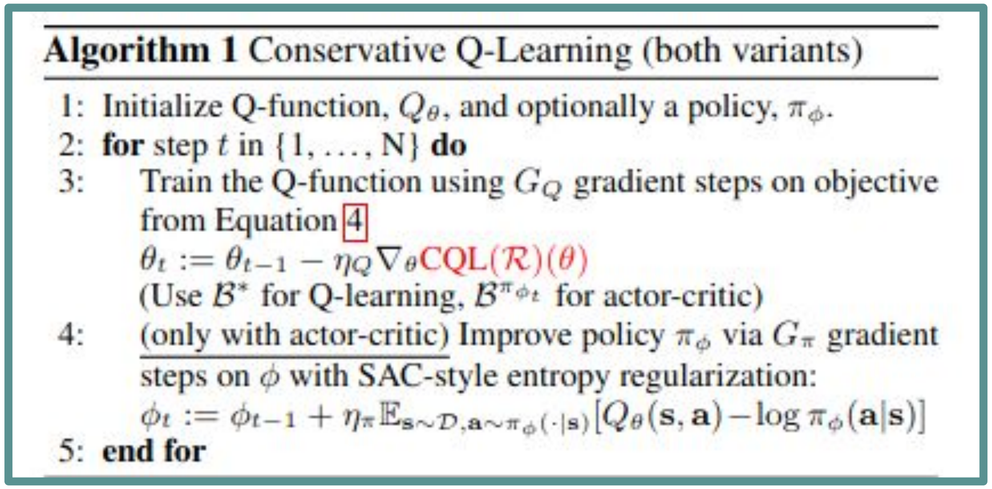


Despite training now taking 4-5 hours instead of 5-10 minutes, we still see the same issues, perhaps the issue is the model?

Introduce CQL

Offline RL is a form of learning that uses a static dataset and attempts to use concepts like rewards and random action sampling to learn a behavior.

Conservative Q-Learning (CQL) is a form of offline RL that introduces penalties for actions outside of the training data distribution.



Algorithm 1 Conservative Q-Learning (both variants)

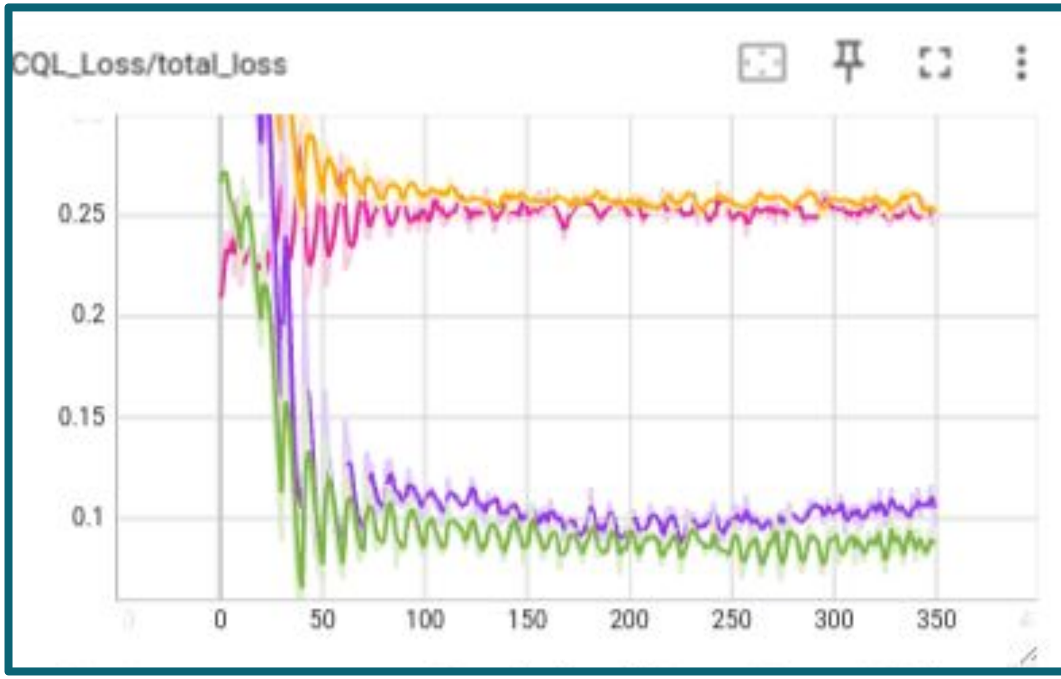
```
1: Initialize Q-function, $Q_\theta$, and optionally a policy, $\pi_\phi$.
2: for step $t$ in $\{1, \dots, N\}$ do
3:     Train the Q-function using $G_Q$ gradient steps on objective
       from Equation 4
       $\theta_t := \theta_{t-1} - \eta_Q \nabla_\theta \text{CQL}(\mathcal{R})(\theta)$
       (Use $\mathcal{B}^*$ for Q-learning, $\mathcal{B}^{\pi_{\phi_t}}$ for actor-critic)
4:     (only with actor-critic) Improve policy $\pi_\phi$ via $G_\pi$ gradient
       steps on $\phi$ with SAC-style entropy regularization:
       $\phi_t := \phi_{t-1} + \eta_\pi \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \pi_\phi(\cdot|\mathbf{s})}[Q_\theta(\mathbf{s}, \mathbf{a}) - \log \pi_\phi(\mathbf{a}|\mathbf{s})]$
5: end for
```

Progress With CQL

- CQL is overall much harder to use as it as many more parameters to tune and a more sensitive loss function.

- CQL also relies on more architecture. Instead of a policy and policy manager, it requires a sample buffer, a policy, a Q-Network, and a manager.

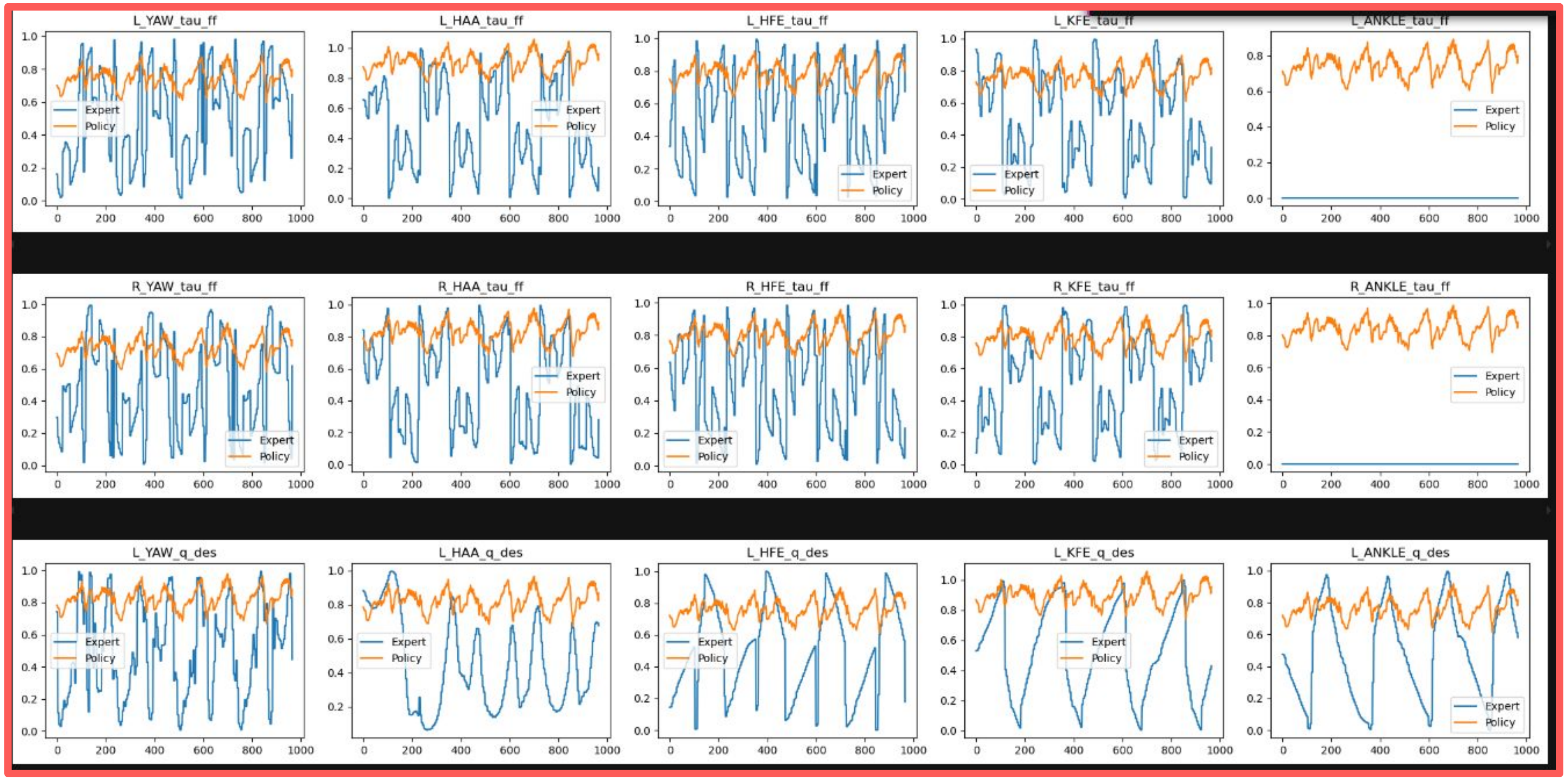
- One of the more parameters to turn is the “cql_alpha” parameter, which scales the penalty for out of distribution actions.



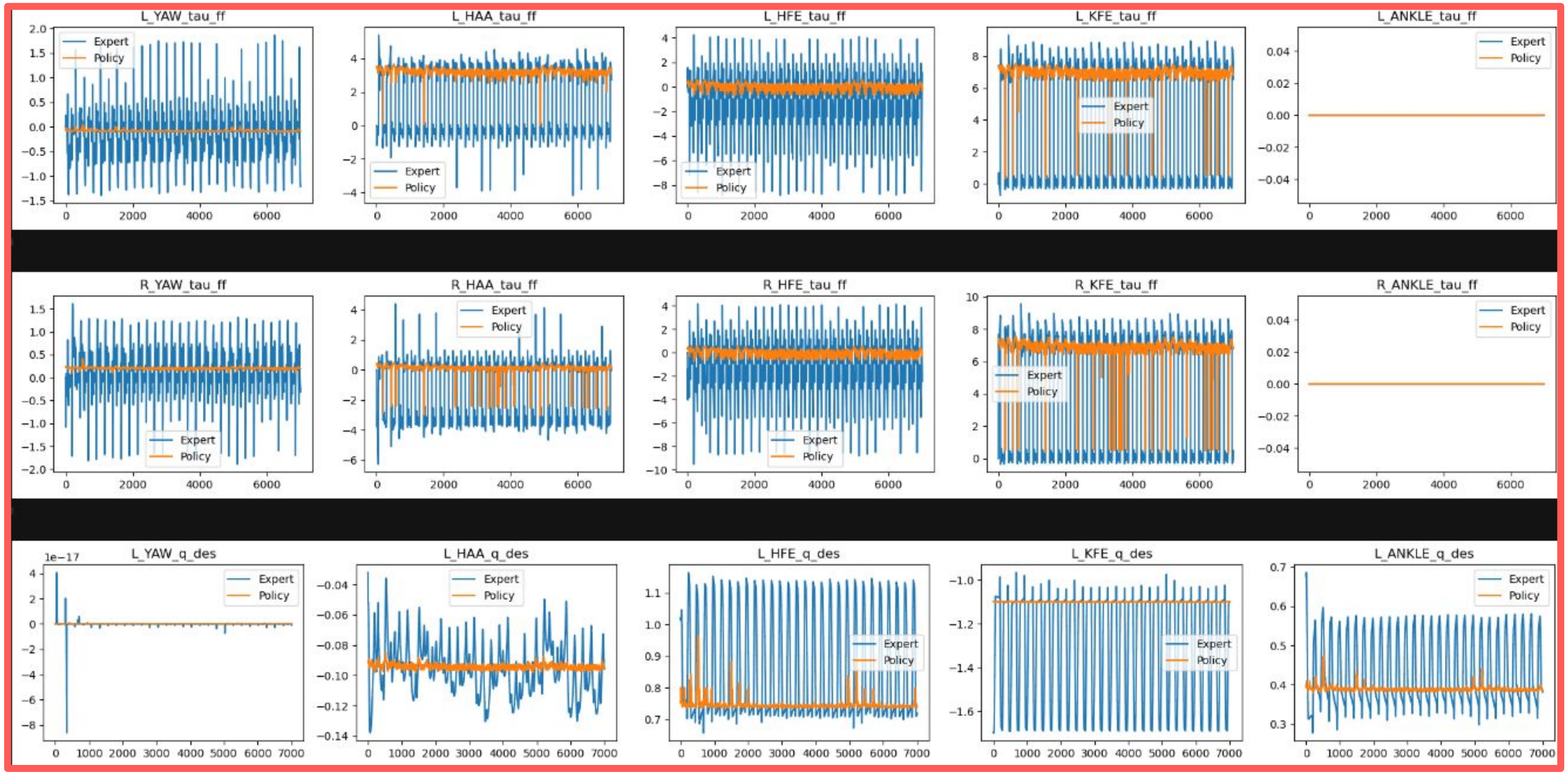
Green: 0.5

Purple: 0.1

CQL Test Results



CQL Simulation Results



Conclusions

- While improvements and progress were made on the pipeline and model training, getting the data to converge enough
- While more robust and “better” at learning behaviors, offline RL is generally much more difficult to use and isn’t necessarily best suited for this task
- When a BC model encounters slightly out of distribution states, it has a strong potential to diverge from the typical behavior completely (generally avoided by CQL). This causes the robot to crash
- When it comes to these models, testing on simulation is much more convenient than testing on hardware
- These models are more difficult to implement than online RL, the most common ML model in walking robotics

Personal Takeaways

- I came into this project with no prior ML experience (I’m a Mechanical Engineer)
- While it was difficult to learn on the fly, I gained a lot of knowledge about an area of robotics I have never seen before
- A lot of ML work that I did was very “black box” in nature, it was difficult to fully understand what the model was doing and why every time it was trained

Future Work

- Both the CQL and the BC models have room for improvement, whether it's continuing to adjust the model parameters or modifying the model architecture
- There may be some issue with the simulation or pipeline that need to be corrected that limit the output of the BC model
- The training data could be improved or made more robust
- A different model may perform better than the two tested
- The biped could eventually be taught novel behaviors such as hurdling

Acknowledgements

Mentor: Soon-Jo Chung

Co-Mentor: Sorina Lupu

Program: SURF/SFP

Special Thanks to the Class of ‘52 Fellowship

Questions?

Jade Millan