HW2: Advanced Behavior Cloning
CS 6955: Adv Artificial Intelligence
University of Utah

Name: Seongil Heo
UID: u1527760
Date: January 26, 2026

ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning

- https://sites.google.com/view/thrifty-dagger/home
- https://github.com/ryanhoque/thriftydagger

Part 1: What did you learn about the algorithm/implementation that you didn't know when you started this?

Although ThriftyDAgger is designed to reduce expert intervention, I observed that control switching between the robot and the human occurred much more frequently than expected. In particular, even small instabilities in novelty or risk estimation often led to frequent expert queries, which prevented human intervention from being reduced as much as intended. This showed that the expert-request mechanism is not just a simple rule, but is highly sensitive to the accuracy of uncertainty estimation and the choice of thresholds.

I also found that successful learning required more repeated expert demonstrations of similar behaviors than I initially expected. While the algorithm aims to minimize expert data, in practice the policy needed multiple demonstrations in similar states to achieve stable performance. In other words, not only data diversity but also repeated demonstrations in similar situations played an important role in learning, which I realized through running the experiments.

Part 2: What evidence do you have that your code/algorithm worked or failed?

My goal was to reproduce Table 1 from the paper, but I was not able to fully complete it because injecting demonstrations into the system took much longer than expected. Nevertheless, the overall system ran correctly, and I observed some successful cases after starting from an initial success rate of 0

Initially, I tried to run the authors' original code directly in order to reproduce the exact results and then gradually upgrade it to a more stable and modern version. However, I spent more than six hours just on environment setup. I tried multiple combinations of Python and package versions, and also attempted various fixes related to MuJoCo and ARM-based compilation. A significant amount of time was spent troubleshooting installation and compatibility issues rather than working on the algorithm itself.

In the end, instead of reproducing the original environment, I heavily modified the code to make it run under a modern software stack. This allowed me to build a working system, even though I could not fully reproduce the final table in the paper.

Part 3: What were some of the biggest challenges you faced?

I used the code released by the authors. The biggest challenge was the issues related to the software environment and version compatibility. The authors' code was built on very old versions of MuJoCo and robosuite, and those versions were not officially supported on macOS ARM systems. In addition, pip installation was not available, which made it very difficult to reproduce the original execution environment used in the paper.

Another major issue was that the observation structure in robosuite has changed in the latest version. The provided .pkl demonstration data was based on a 51-dimensional state, while the current environment produces 67-dimensional observations. As a result, it was necessary to analyze which observation keys should be used and to modify the preprocessing pipeline to match the state representation assumed in the paper. This process was technically demanding and time-consuming.

Overall, reproducing the environment and aligning the state representation turned out to be more challenging than implementing the algorithm itself.

Part 4: How successful were you at getting things to work?

The original implementation was based on Python 3.6, mujoco-py 2.0.2.9, robosuite 1.2, and Gym. I updated the entire software stack to Python 3.12, MuJoCo 3.4.0, robosuite 1.5.2, and Gymnasium.

Robosuite in particular had undergone major structural changes, and the observation space differed from the version used in the paper. To make the state representation identical to the original setup, I modified the observation preprocessing by removing components such as *joint_pos*, *joint_acc*, and *eef_quat_site* from the observation vector. This ensured consistency with the dimensionality and structure of the demonstration data. I also updated device handling and rendering-related code to match the new APIs.

After these modifications, the experiments ran without interruption, and user input (for interactive data collection and control) was handled correctly.

Part 5: In hindsight, what might you have done differently if you were to go back in time and redo this assignment?

If I were to redo this assignment, I would not have spent time trying to install the authors' original environment on my Mac from the beginning. Instead, I would have immediately started by analyzing the code structure under the latest software stack and designing a clear strategy for mapping the new observation format to the one assumed in the paper.

I also learned that I should not rely entirely on AI-generated instructions at the early stage. In several cases, the AI suggested that older versions could be installed, while the official documentation later revealed that those versions were not supported on ARM-based systems. Checking the official documentation first would have saved a significant amount of time.

Part 6: How was AI helpful and/or frustrating in completing this assignment?

I used ChatGPT and Codex. The AI was very helpful for solving local, code-level problems. It performed particularly well when modifying functions or writing code to handle API changes. For relatively simple or well-scoped tasks, AI was able to provide fast and useful solutions.

However, for more complex problems, AI required very specific and detailed instructions to be effective. In situations involving multiple packages and system-level dependencies, its suggestions were not always reliable. In particular, it struggled with system-related tasks such as environment configuration, and with newer software versions where available information is still limited.

Overall, AI worked well for clearly defined coding tasks, but for more difficult system-level issues, it required precise guidance and additional verification through official documentation.