Class project 1

Matouš Bohoněk, 2025/03

# Behavioral cloning

## Results

For experimenting with pure behavioral cloning I chose two tasks: the Walker2d which did not achieve 30% (just barely at 29.8%) of the expert return and the HalfCheetah which achieved an impressive 96% return.

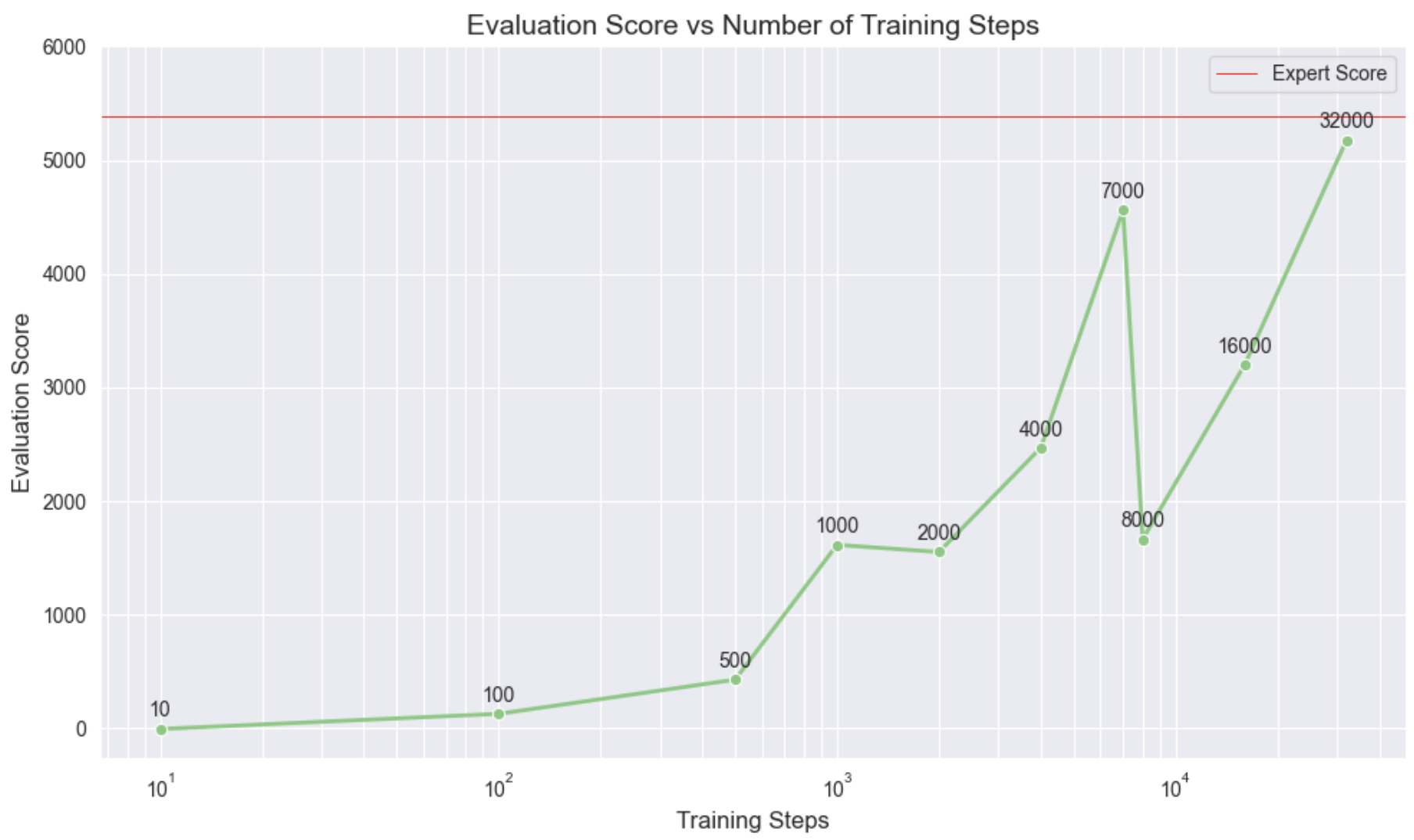
|  | Walker2d |  | HalfCheetah |  |
| --- | --- | --- | --- | --- |
|  | Eval | Train | Eval | Train |
| Mean return | 1604 | 5383 | 3872 | 4035 |
| STD of return | 1505 | 54 | 86 | 33 |
| # of eval runs | 270 | 2 | 100 | 2 |

For both experiments the maximum episode length was set to 1000 and the eval\_batch\_size to 100,000.

When comparing the observation and action spaces, we see that both Walker2d and HalfCheetah have the same number of actions, with HalfCheetah having only one fewer observation (16 versus 17). Despite this dimensional similarity, their performance differs dramatically. This difference stems most likely from the movement mechanics: Walker2d requires precise bipedal balance with each step, having a higher center of mass , while HalfCheetah has a lower center of mass and much wider base. The Walker2d agent can easily tip over, resulting in terminal failure, while HalfCheetah can recover from suboptimal actions. This perfectly illustrates how errors can compound quadratically in environments where small mistakes lead to catastrophic, non-reversible situations—a classic example of the distribution shift problem in imitation learning. Walker2d's high standard deviation further supports this analysis; sometimes the agent 'gets lucky' and progresses far, while other times an early mistake causes immediate failure. Conversely, HalfCheetah demonstrates that pure behavioral cloning can succeed in more forgiving environments that allow recovery from errors.

## Experiments

One set of hyperparameters that I decided to tune is the number of training steps. I thought that training only for the default 1000 steps might not be enough and, as it turns out, there is some benefit to training for longer periods of time. See the graph below for the Walker2d problem:

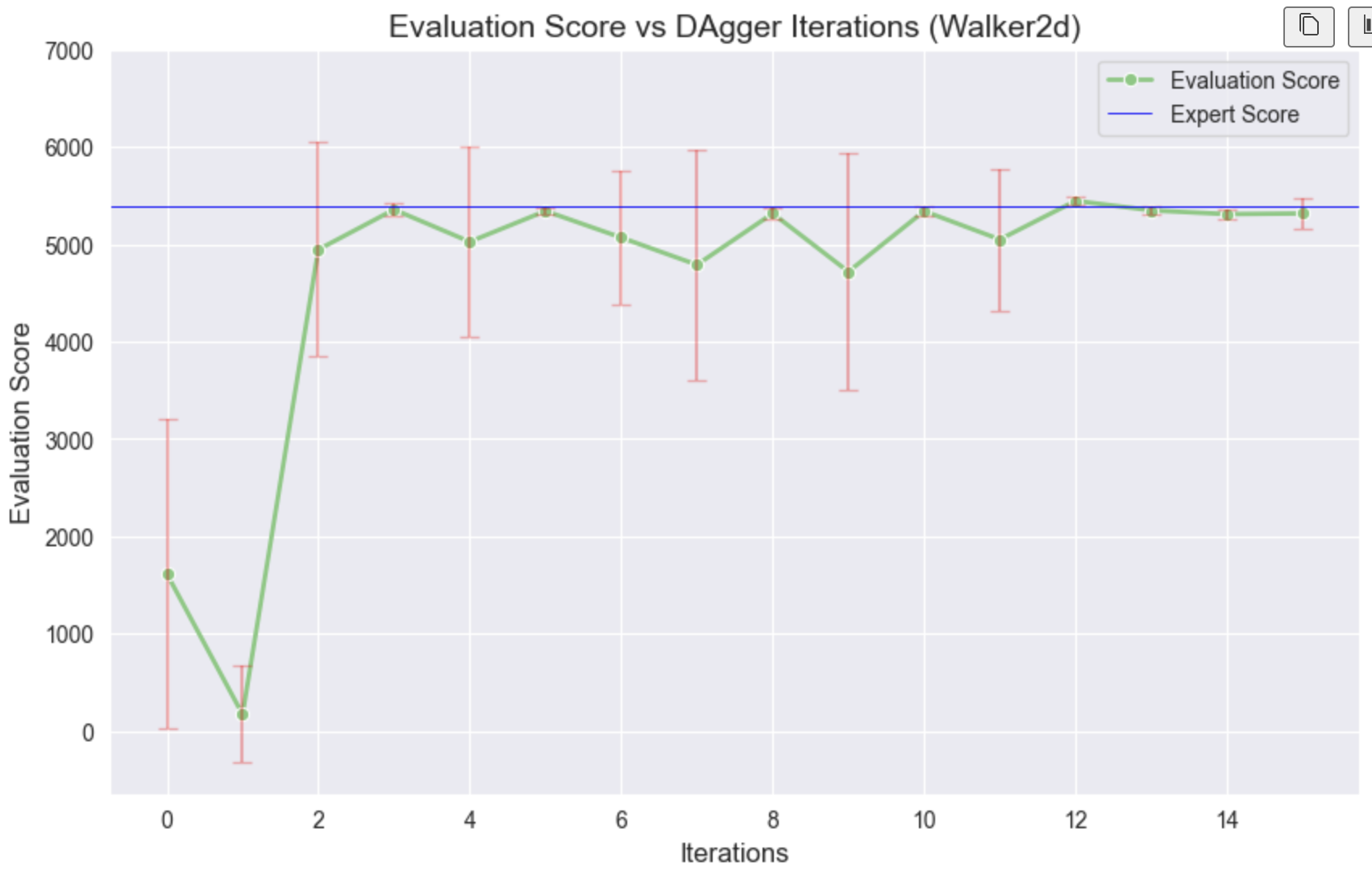


As we can see, adding more training on the same expert data proved to be beneficial, although the relationship isn’t perfectly linear. We can see that 7,000 seems optimal as adding more achieves a better score only way later, at 32,000 steps. All evaluations were calculated with maximum episode length set to 1000 and the eval\_batch\_size to 10,000. Network size and other hyperparameters were left untouched at their default values.

# 

# DAgger

Using dagger showed significant improvement in both tasks, achieving performance on par with the expert or even better in the case of HalfCheetah. See the graphs below:





All evaluations were calculated with maximum episode length set to 1000 and the eval\_batch\_size to 10,000. Network size and other hyperparameters were left untouched at their default values.

This improvement is thanks to the way DAgger helps the agent learn from its mistakes. In the expert data of Walker2d there probably weren’t many mistakes which meant the agent didn’t know how to recover when he made one. After using DAgger, the agent got trained on a more diverse dataset which included mistakes and subsequent recoveries which helped tremendously. The HalfCheetah has improved even beyond expert performance, which might be due to just having a bigger and more diverse dataset which helped the agent learn a better policy.