CS8803-Fall24 Course Project
Learning to Navigate: An Imitation Learning Framework for Path Planning in Adversarial Environments

Connor Bossard, Sophia Imhof, Sidney Wright, Elias Izmirlian
Georgia Institute of Technology
cbossard3@gatech.edu, simhof3@gatech.edu,
eizmirlian3@gatech.edu, swright92@gatech.edu

June 21, 2025

Abstract: This project seeks to explore training reinforcement learning agents to navigate complex and dynamic maze environments. Our approach pre-trains an RL agent through imitation learning methods using A* generated "expert" trajectories on a static maze environment. The pretrained agent is then fine-tuned in a dynamic environment with moving adversarial agents. Our method improves sample efficiency and enhances performance across various maze sizes.

Keywords: Deep Reinforcement Learning, CS8803, Course Project

1 Project Links

Link to Presentation: https://mediaspace.gatech.edu/media/DRL+Project+Presentation/1_98ww5ybv

Link to Github Implementation: https://github.gatech.edu/simhof3/DRL-Project

2 Introduction

Solving complex navigation tasks with moving obstacles is a challenge for reinforcement learning (RL) agents. This problem is important for use cases such as autonomous driving or flying, where dynamic obstacles create unpredictability that an RL agent must be able to adapt in real time. Expert path planning algorithms such as RRT (continuous) and A* (discrete) can solve static mazes efficiently [3], but would struggle in a dynamic environment with moving obstacles. Our goal is to develop an RL agent that successfully navigates complex maze environments with dynamic obstacles.

Our project will focus on a discrete action space so we will utilize A*. A* is a search algorithm that finds the shortest path from a starting position to end position by considering the estimated cost to move to a given cell in the maze and cost to the goal using a heuristic function. A* is effective in planning long-term routes in static environments. However, it does not respond well to dynamic planning problems in which obstacles need to be avoided (requires frequent re-planning)[2]. On the other hand, RL has been shown to be learn to avoid these dynamic obstacles without re-planning [1]. Training an RL agent on long-term path-planning scenarios is computationally expensive. As the observation space grows, training significantly increases in complexity.

Therefore, for a path-planning problem in an environment with dynamic obstacles or adversarial agents, we hypothesize that thoughtful pretraining could improve performance. Pretraining will be done using A* generated trajectories as an expert agent during imitation learning. Our intuition is this will decrease training time for our RL agent in the dynamic environment and increase overall performance. This approach balances the efficiency and performance of path-planning search algorithms with the adaptability of reinforcement learning.

3 Related Work

Dynamic obstacle avoidance is a well studied application of reinforcement learning, partially due to the more difficult nature of path planning in an environment that is not solely contingent on the agent actions and transition dynamics. For example, [4] demonstrates multiple agent decentralized collision avoidance, where agents avoid each other through dynamic path planning rather than communicating with each other. This differs from our setting, where we have a single "good" agent which is learning dynamic path planning, and one or more "bad" agents that act as moving obstacles.

Previous research demonstrates a number of viable solution methodologies to this kind of environment, such as careful design of a reward function and experience replay using sample weighting as exemplified in [1]. This approach is designed for an environment with a continuous state and action space with a high degree of freedom, which causes a dynamic obstacle to create additional complications, such as more difficult collision detection. Their comprehensive reward function and sample weighting requires a lot of domain knowledge be encoded in the design of their agent, whereas our approach attempts to encode this domain knowledge through expert trajectories for a simpler problem, and then learn a more difficult path planning policy using this initialization.

In this way, our approach is somewhat similar to curriculum learning for dynamic obstacle avoidance, as seen in [5]. In (Wang et. al) the authors use curriculum learning - a methodology which involves learning sub-tasks and combining policies for these sub-tasks to learn a more complicated task. The authors found success for environments with moving obstacles when a task hierarchy (or "curriculum") is designed to transfer specific functions such as distance and pace functions that are appropriate to the difficulty of the environment. The authors also found that training from scratch was ineffective for more difficult environments with moving obstacles. While our approach does not involve an ordered task hierarchy, we do adopt a similar approach of transferring certain portions of policy into our agent before continuing to learn.

4 Methods

4.1 Environment

We are using a custom 2D grid maze gym environment throughout our experiments. The maze is initialized as a matrix with each cell value representing the state of the maze in that position. A given cell can be occupied by a free space, a wall, our RL agent, a bad agent, or a goal location. There is only one RL agent and goal location in each maze. The maze is randomly initialized in a way that ensures there is a passage from the starting location to the goal location.

4.1.1 Static Maze

For the static environment, we place a single bad agent in the maze during maze initialization and do not alter its position.

4.1.2 Dynamic Maze

For the dynamic environment, we assume the bad agent moves to a random valid state, however it could easily be adapted to different bad agent algorithms like pure pursuit. We chose this bad agents algorithm because it shows the RL agent's ability to learn to avoid the bad agent without increasing the difficulty of the environment to much due to the resources and time we had this semester.

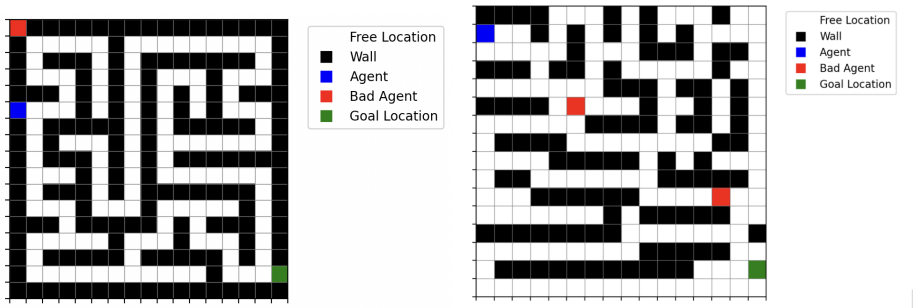

Figure 1: Static maze environment(left) and Dynamic maze environment with moving adversarial agents (right).

4.2 Algorithms

4.2.1 Generating Expert Trajectories

To generate a buffer of expert trajectories for our imitation algorithms, we used A* to generate a path for a series of randomly initialized mazes. Then, we followed the path through the maze and stored the corresponding state/action pairs in the buffer.

4.2.2 Behavior Cloning

We used a traditional Behavior Cloning algorithm, where the expert trajectories were generated as mentioned above. We pretrained the PPOs policy network by keeping the value network as is. The policy was updated using batch gradient descent on the loss between the policies log likelihood actions compared to the expert trajectories action.

4.2.3 DAGGER

As an alternative and more sophisticated imitation learning algorithm to vanilla Behavior Cloning we also implemented: DAGGER. The initial policy passed into the DAGGER algorithm was the pretrained Behavior Cloning policy. Using DAGGER helped us reach out-of-distribution states that simply creating a policy based off A* did not reach. DAGGER is able to reach these unfamiliar states by sampling from the current policy for the next action, and stepping with that action in the environment. However, it runs A* on the current state to add the expert action to the replay buffer. This sampling of the next state leads to more variance in the states reached leading to a more robust pretrained policy. The pseudo-code for DAGGER is shown below in Figure 2.

4.2.4 PPO

After pre-training the policy network with Behavior Cloning or DAGGER, we continued training on PPO, because of its stability and ease of implementation. PPO is a policy gradient method,

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

Figure 2: DAGGER psuedocode.

with a clipped surrogate objective to limit the change to the network in a single step. Unlike BC or DAGGER, which rely on supervised learning with expert demonstrations, PPO is an on-policy RL agent that learns directly from interactions with the environment. The pseudo-code for PPO is shown below in Figure 3.

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.
7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.
8: **end for**

Figure 3: PPO psuedocode.

5 Experimental Results

First, we were able to show that training the agent with BC and DAGGER outperformed a PPO agent in terms of sample efficiency and accuracy on a static environment. This shows the benefit of pretraining with an imitation learning algorithm rather than using a curriculum learning framework. For a 7x7 maze, the accuracy of a random agent was 15 percent, the accuracy of the PPO agent with 5 million steps was 72 percent, while the accuracy of the BC and DAGGER agents were both 99+ percent. These trends continued for larger mazes. For a 11x11 maze, the accuracy of a random agent was 2 percent, the accuracy of the PPO agent with 5 million steps was 4 percent, while the accuracy of the BC and DAGGER agents were 19 and 39 percent respectively.

Next, we used the BC and DAGGER policies from the static environment as the starting policy for the dynamic mazes. We were able to show that despite the environment changing significantly, pretraining on the simpler, static environment yielded improved average rewards and accuracy. For the small dynamic maze, the accuracy with PPO was 34%, which increased to 43% and 64% for Behavior Cloning and DAGGER respectively. In Figure 5, we can see that the PPO agents pretrained with BC converged faster to a higher reward on both the small and medium maze, despite that pretraining was done on a different environment.

Surprisingly, we found that DAGGER did not yield significantly improved results when transferred

Table 1: Performance Metrics for Static and Dynamic Mazes (Small Maze)

Agent/Method	**Accuracy (%)**
Random Static Maze	15
PPO Static Maze (no pretraining)	72
PPO Static Maze with Pretrained BC	99+
PPO Static Maze with Dagger	99+
PPO Dynamic Maze (no pretraining)	34
PPO Dynamic Maze with Pretrained BC	43
PPO Dynamic Maze with Dagger	64

Table 2: Performance Metrics for Static and Dynamic Mazes (Medium Maze)

Agent/Method	**Accuracy (%)**
Random Static Maze	2
PPO Static Maze (no pretraining)	4
PPO Static Maze with Pretrained BC	19
PPO Static Maze with Dagger	39
PPO Dynamic Maze (no pretraining)	14
PPO Dynamic Maze with Pretrained BC	27
PPO Dynamic Maze with Dagger	26

to the dynamic mazes over Behavior Cloning despite having higher average reward and accuracy on the static maze during pretraining. We hypothesize that there is only so much "knowledge" that can be transferred over from the static environment to the medium-sized dynamic environment. For example, both these agents learned to avoid walls and move in the direction of the goal, however the degree to which they learned these may not be significant when they continued to learn in an RL framework on a different environment.

6 Discussion and Analysis

Our approach proved to be successful over our baseline performance for the dynamic mazes, supporting the intuition that pretraining with imitation learning using A* gives a performance boost to reinforcement learning algorithms such as PPO. We saw this performance boost in both the static and dynamic mazes. While we were more interested in observing the effect of a pretrained policy on PPO performance in the dynamic maze, the boost in accuracy seen in the pretrained PPO for the static maze (both medium and small) is also interesting. This could indicate that the clipped updates made by PPO work better when the policy has been initialized to have some foundation suited to the environment.

The imitation learning algorithm employed for policy initialization seems to have minimal impact on the agent performance in the dynamic maze which is somewhat surprising, as DAGGER is considered to be a more advanced imitation learning algorithm than Behavior Cloning. Potential reasons for this could include that the policy initialization on the static maze can only be so helpful. The nature of imitation learning algorithms learning from A* causes our pretrained policy to be better posed for general maze solving, but the moving obstacle adds a layer of complexity that has substantial effects on the optimal policy. Thus, having a better policy for a static maze does not necessarily translate to a better policy on the dynamic maze. Our results do support, however, that some policy initialization is beneficial.

Another interesting observation is the average reward over time on the medium maze seen in Figure 5 - we see that the PPO algorithm without pretraining outperforms the pretrained agent for the first million rollouts. Afterwards the average reward of the pretrained agent continues to increase while

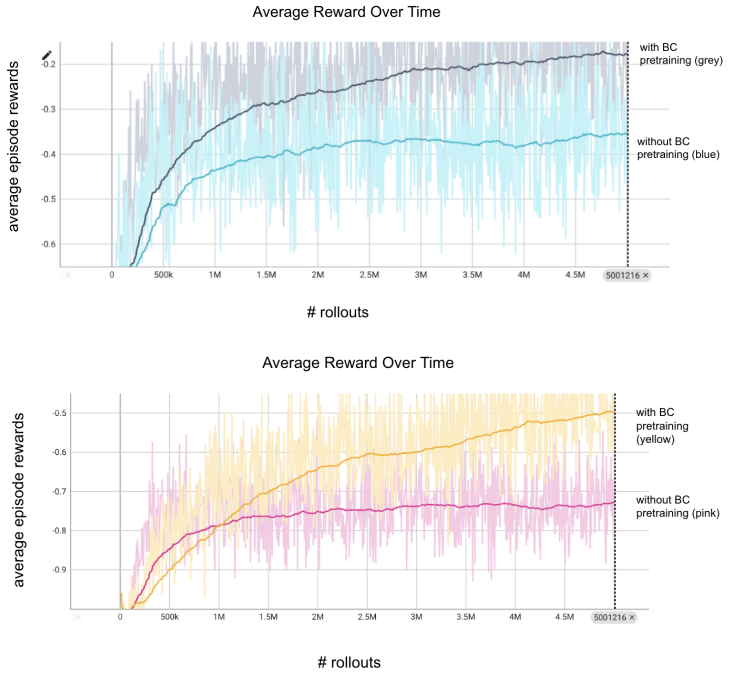

Figure 4: Comparison of average reward over time of PPO agent when pretraining with BC versus no pretraining on a small (top) and medium (bottom) maze.

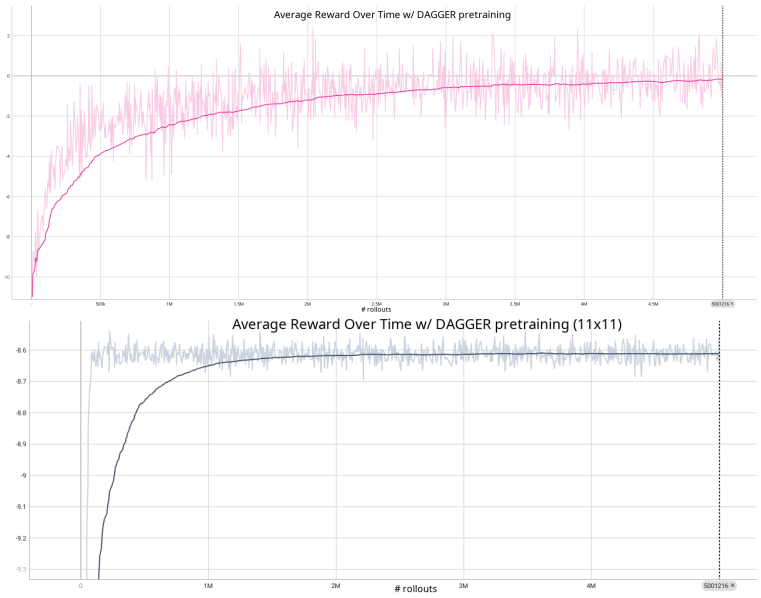

Figure 5: Comparison of average reward over time of PPO agent when pretraining with DAGGER versus no pretraining on a small (top) and medium (bottom) maze.

the non-pretrained agent seems to plateau. With a pretrained policy transferred from a different environment, you might expect initial performance boosts that decrease a little over time, but here we see the opposite. This could be because the initialized policy does not provide immediate benefit, as the difference in environment might lead to the policy biasing unhelpful actions at times. Thus,

PPO must correct certain parts of the pretrained policy which are detrimental before seeing long term benefits of the pretrained policy.

The performance of all agents in the medium mazes is significantly lower than in the small maze. Ideally, we would see not just a performance boost, but good performance from our pretrained PPO policy. We faced some challenges with the medium maze, as the larger state space made it more difficult for our baseline algorithms to learn. Since we had to perform tuning on the pretraining algorithm as well as PPO, the performance was also highly variable. A possible future direction would be to experiment with different policy network structures, such as a Convolutional Neural Net, as opposed to the MLP policy we employed. A CNN might be better suited to this task, and could even see larger benefits from being pretrained. It also might be helpful for the issue we faced with the medium maze of the state space being too large.

7 Conclusion

Our findings support the idea that a pretrained policy for a related, simpler task is beneficial to learning a more complicated task even if a subtask hierarchy is not created as it is in curriculum learning. We also demonstrate that transfer learning using imitation learning from an algorithm better suited to a simple task can have benefits for learning on a more complex task. While our algorithms were not able to achieve very high accuracy on the medium maze, we were able to demonstrate performance boosts gained from pretraining.

These findings are useful for applications involving dynamic obstacle avoidance, especially ones with limited access to the ”true” or more complex environment, as a simpler simulated environment can be used for pretraining. Our results indicate that this is especially true for smaller state spaces, but further research could demonstrate equal or greater benefits in larger state spaces.

Acknowledgments

Thank you to our TA's and Professor Garg for a great semester!

References

[1] P. Chen, J. Pei, W. Lu, and M. Li. A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. *Neurocomputing*, 497:64–75, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.05.006. URL https://www.sciencedirect.com/science/article/pii/S0925231222005367.

[2] D. Connell and H. M. La. Dynamic path planning and replanning for mobile robots using rrt. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1429–1434, 2017. doi: 10.1109/SMC.2017.8122814.

[3] Y. He, P. Wang, and J. Zhang. A comparison between a* rrt in maze solving problem. In *2021 3rd International Symposium on Robotics Intelligent Manufacturing Technology (ISRIMT)*, pages 333–338, 2021. doi: 10.1109/ISRIMT53730.2021.9596830.

[4] C. Jaewan, L. Geonhee, and L. Chibum. Reinforcement learning-based dynamic obstacle avoidance and integration of path planning. *Intel Serv Robotics*, 8(5):663–677, 2021. doi: https://doi.org/10.1007/s11370-021-00387-2.

[5] H.-C. Wang, S.-C. Huang, P.-J. Huang, K.-L. Wang, Y.-C. Teng, Y.-T. Ko, D. Jeon, and I.-C. Wu. Curriculum reinforcement learning from avoiding collisions to navigating among movable obstacles in diverse environments. *IEEE Robotics and Automation Letters*, 8(5):2740–2747, 2023. doi: 10.1109/LRA.2023.3251193.