Memory and Curiosity in POMDP Imitation Learning

Ryan Hsiang
Department of Electrical Engineering
National Taiwan University
b11901040@ntu.edu.tw

Hubert Chon-Ho Chang
Department of Electrical Engineering
National Taiwan University
b11901028@ntu.edu.tw

Han Jun, Ko
Department of Electrical Engineering
National Taiwan University
b11901071@ntu.edu.tw

Jia Lin, Fang
Department of Electrical Engineering
National Taiwan University
b11901003@ntu.edu.tw

Abstract

Imitation learning in POMDP faces unique challenges due to the limited information provided by the environment. Especially when trained with a fully observable expert, whose behavior may appear confusing to the agent when given insufficient information. As a result, effective information gathering and utilization of memory and exploration techniques is crucial for learning. In this work, we look to address these issues by developing an imitation learning framework that integrates memory-based architectures for efficient information retention. In addition, we propose three distinct approaches to enhance exploration, including online fine-tuning, history imitation, and data collection. Our results show that our framework is able to improve training speed and performance of behavior cloning while encouraging the exploration of unseen states or teaching the expert to collect better data to fit the agent in memory-related tasks.

1 Introduction

In many real-world scenarios, such as robotics and autonomous driving, agents operate under uncertainty, receiving incomplete or noisy information about their environment. Such situations are aptly modeled as Partially Observable Markov Decision Processes (POMDPs), where the agent must make decisions based on limited observations rather than full state information. This partial observability poses significant challenges for imitation learning, a paradigm where agents learn to perform tasks by mimicking expert demonstrations.

Discrepancies arise when experts have access to complete state information and agents do not. Agents may find expert actions perplexing without the underlying context, leading to suboptimal imitation. Therefore, effective strategies for information gathering, memory utilization, and exploration are crucial. Memory mechanisms enable agents to retain and integrate past observations, constructing a more comprehensive understanding of the environment over time. Exploration techniques allow agents to seek out informative experiences, facilitating generalization beyond the expert demonstrations.

To enhance the agent's information retention capabilities, we incorporate memory-based models into behavior cloning for training in a partially observable environment. More specifically, we evaluate the performance of behavior cloning trained on the *Door Key* and *Memory* task on Minigrid[3] using three strategies to encode observation history: LSTM, Transformer, and frame-stacking, and compare them to a single-state representation. In the *Door Key* task, our encoding strategies were able to reach optimal play faster than a single-state encoding while showing better training stability. In the *Memory* task, Our memory architectures outperform the single-state representation, achieving 100% success

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

rate compared to 50% for the single-state representation. We found that frame-stacking was able to reach perfect play in the least number of epochs.

We propose three approaches to help the agent learn to explore.

1. **Online Fine-Tuning:** After training behavior cloning with the observation encoder offline, we fine-tune the pre-trained policy online using Proximal Policy Optimization. The fine-tuned policy was able to effectively explore, achieving an average reward only slightly lower than the fully observable expert.
2. **History Imitation:** Using a capable pre-trained history encoder, we leveraged GAIL to provide similarity scores between the history embeddings of the expert and the agent as rewards. However, this method achieved only a 50% success rate, with the agent learning segments of the expert's movements but failing to capture the critical exploration component. Interestingly, about 10% of the samples displayed some evidence of exploration, but the interaction steps required significantly exceeded those of training PPO from scratch.
3. **Collecting Better Data:** Instead of using the expert starting from the left side, this approach modifies the related work A2D [1], aiming to teach the expert starting from the aisle to collect better data for the agent to learn more efficiently. This approach creates a reward-learnable environment and a corresponding expert to fit the agent's behavior based on the reward from the real environment that the agent interacts with.

2 Related Work

Imitation learning in POMDPs has garnered significant attention, with various approaches proposed to address the challenges posed by partial observability. Gangwani et al. [4] introduced a method for learning belief representations in imitation learning within POMDPs. Their approach jointly trains the belief module with the policy using a task-aware imitation loss, ensuring alignment with the policy's objectives. They also incorporate belief regularization techniques, such as multi-step prediction of dynamics and action sequences, to enhance robustness.

Yue et al. [5] addressed the challenge of integrating an agent's history into memory for decision-making in POMDPs. They introduced the concept of memory dependency pairs to capture the expert's memory mechanisms used in decision-making, enabling the agent to better mimic expert behavior.

Warrington et al. [1] explored advancements in learning optimal policies for partially observed Markov decision processes (POMDPs). They address critical limitations in existing imitation learning approaches, particularly the challenges posed by asymmetric information—where an expert has access to data unavailable to the trainee. To overcome these issues, they introduce a novel algorithm, Adaptive Asymmetric DAgger (A2D). This approach iteratively refines the expert's policy, ensuring it provides guidance that aligns with the trainee's constraints.

These studies highlight the importance of memory mechanisms and exploration strategies in imitation learning within partially observable settings. Our work builds upon these foundations by integrating memory-based architectures and proposing new exploration approaches to enhance learning from non-exploring experts.

3 Problem Formulation

A partially observable Markovian decision process is defined as a 7-tuple containing the following:

- $S = \{s_1, s_2, ..., s_n\}$ is a set of states
- $A = \{a_1, a_2, ..., a_m\}$ is a set of actions
- T is a set of conditional transition probabilities $T(s'|s, a)$ for state transitions conditioned on the action.
- $R : S \times A \to \mathbb{R}$ is the reward function.
- $\Omega = \{o_1, o_2, ..., o_k\}$ is a set of observations.
- O is a set of observation probabilities $O(o|s', a)$ conditioned on the state and action.

- $\gamma \in [0, 1]$ is the discount factor.

In a partially observable setting, the environment has a well-defined state S that produces the environment's observations, but is hidden to the agent. As a result, the agent must construct its own state representation, i.e. its belief of the environment state. The agent's *belief state* is Markovian by assumption, and is defined as the probability distribution over possible states conditioned on the agent's information of the environment,

$$b(s) = P[S = s].$$

Under this setting, we investigate the behavior cloning approach in imitation learning. Behavior Cloning formulates imitation learning as a supervised learning problem. Given expert demonstrations, the agent learns a policy π that maps the observation o to an action. Concretely, the goal of Behavior cloning is to solve the following optimization problem:

$$\underset{\pi}{\operatorname{argmin}} \, \mathbb{E}_{s \in S}[\mathcal{L}(\pi(s), \pi^*(s))].$$

where π^* is the expert policy and $\mathcal{L}$ is the loss function.

A special case is when the expert has full observability of the environment but the agent does not, that is, the expert policy uses the environment state s as input while the agent only observes o:

$$\underset{\pi}{\operatorname{argmin}} \, \mathbb{E}_{s \in S, o \in \Omega}[\mathcal{L}(\pi(o), \pi^*(s))].$$

In addition to the original supervised learning problem, the agent must also learn a mapping from the partial observation o to a belief state $b(s) = P[S = s|\, o]$ that serves as an alternative to s:

$$\underset{\pi}{\operatorname{argmin}} \, \mathbb{E}_{s \in S, o \in \Omega}[\mathcal{L}(\pi(b), \pi^*(s))].$$

In general, this is impossible to achieve, as the observation o inherently lacks the complete information present in the full state s. As a result, reconstructing the full state s from the limited information of observations is crucial in any learning task in a partially observable MDP. One possible solution is to collect past observations $\{o_t, o_{t-1}, ..., o_1\}$, which may provide information not available in a single observation o_t. The hope is that the belief state

$$b(s) = P[S = s|\, o_t, o_{t-1}, ..., o_1],$$

can better approximate s. Two key challenges emerge: First, given the observations, the agent must learn an accurate belief state. This is addressed using deep learning methods. Sensible approaches include:

- **Frame Stacking**: The belief state is constructed by concatenating a fixed number of recent observations:
$$b_t = [o_t, o_{t-1}, \ldots, o_{t-k+1}].$$
- **Recurrent Neural Networks**: The belief state is updated sequentially using a recurrent model that processes the current observation along with the previous belief state:
$$b_t = \mathrm{RNN}_\phi(b_{t-1}, o_t).$$
- **Transformers**: The belief state is inferred by processing the entire sequence of observations up to the current time step using an attention mechanism:
$$b_t = \mathrm{Transformer}_\phi(o_t, o_{t-1}, \ldots, o_1).$$

Secondly, the observations collected by the agent must contain sufficient information to reconstruct the belief state and enable effective decision-making. This requires exploration strategies that help the agent collect diverse, informative data.

3.1 Environment

In practice, there are several ways to create partial observability in an environment, including adding observation noise, masking full observations, or other environment-specific constraints. In this work, we achieve partial observability by limiting the agent's view of the environment. Specifically, we use

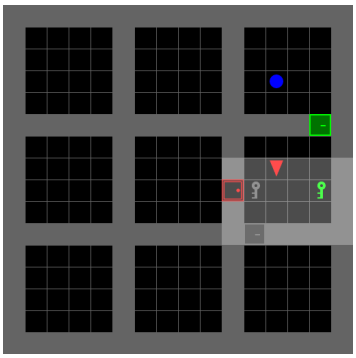


Figure 1: The Minigrid environment, the agent's view is a 7x7 grid obstructed by walls

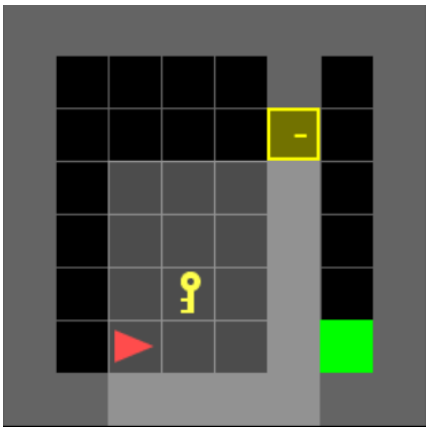
Num	Name	Action
0	left	Turn left
1	right	Turn right
2	forward	Move forward
3	pickup	Pick up an object
4	drop	Unused
5	toggle	Toggle/activate an object
6	done	Unused

Table 1: Action Space of Minigrid

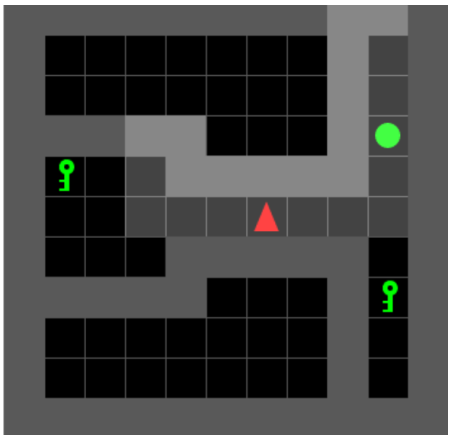
Minigrid(1) developed by the Farama foundation, a collection of grid-world environments designed for reinforcement learning research. In this environment, the partially observable agent receives a 7x7 view of the grid directly in front of it. Each tile is represented by a 2-dimensional tuple containing the `OBJECT_ID`, `COLOR_IDX`, describing the object type (floor, wall, key, etc.) and color of each tile. The agent's view may be obstructed by walls. The agent's action space is given by (1). The reward for each episode is

$$\text{Reward} = 1 - 0.9 \times \left(\frac{\texttt{step\_count}}{\texttt{max\_steps}} \right),$$

for success, and 0 for failure. We select the *Door Key* and *Memory* tasks in Minigrid to investigate our methods.



(a) Door Key task



(b) Memory task

3.1.1 Door key

In Door Key, the agent starts from a random position. It must first pick up the key, positioned randomly in the left room, then navigate to the goal in the right room by opening the door. The walls and door of the environment is fixed. Since the observation only contains the state of each tile in front of the agent, it must remember some key information:

- The location of the door
- The location of the key
- Whether it has picked up the key.

We hypothesize that an effective encoding of observation history will assist with remembering this information.

3.1.2 Memory

In memory, the agent starts in the hallway, it has to go through the hallway to a split. At each end of the split is an object, one of which is the same as the object in the left room. The agent has to remember the object in the left room and go to the matching object at the split. The location and type of the object are chosen randomly. To solve this task consistently, the agent must

- Learn to navigate to the left room.
- Learn to navigate from the left room to the goal.
- Remember the object in the left room.

In addition to an effective encoding strategy, the agent must learn to explore states that may not directly lead to the goal, even at the cost of taking more steps.

4 Method

In the Door Key task, there is little discrepancy between the partially-observable agent and the fully-observable expert's information. Both are able to identify the location of the door and the key within the first few frames. As a result, conventional behavior cloning should suffice without the need for additional exploration. The key insight here is how different observation encoding might affect learning efficiency and episodic success rate.

On the other hand, the Memory task poses a more significant challenge. The expert will be able to identify the target object in the first frame, however, the agent can only observe the target object once it has entered the left room. As a result, the expert will enter the split without turning and checking the left room. If the agent were to naively follow this trajectory, it would have no way of knowing which goal to enter. Therefore, in addition to remembering the target object, the agent must learn to explore states not traversed by the expert to obtain critical information.

To simplify this task, we adopt a two-step approach. First, the agent is trained to remember the object and navigate to the goal. Subsequently, the agent is trained to develop exploration strategies. Specifically, we move the agent's starting point to the left room, so that it is able to observe the target object without the need for exploration. This setup ensures that the agent can initially concentrate on learning to navigate to the goal associated with the correct object, thereby testing the effectiveness of its observation encoding. After which, we can train the agent on exploration strategies.

4.1 Observation Encoding

To develop an effective belief state of the environment, we test various memory-based architectures against a single-state representation. We run behavior cloning on the two MiniGrid tasks, Memory and Door Key, using the following encoding strategies:

- **Frame Stacking**: A history of recent observations with a length of 5 is concatenated directly and then passed through a convolutional neural network (CNN) to effectively exploit the spatial and temporal features of the observations.
- **Long Short-Term Memory (LSTM)**: The agent uses a recurrent architecture to process sequential observations and maintain a hidden state that captures temporal dependencies.
- **Transformers**: The agent encodes all past observations using a transformer architecture with attention mechanisms.
- **Single-State**: The agent relies solely on the current observation, with no memory or historical context, and uses a CNN architecture similar to that of frame stacking.

The embeddings output by the encoders are then passed through the policy network, which consists of two fully connected layers.

4.2 Exploration

Once we have developed a strong BC agent starting from the left room. We explore three distinct methods to improve the ability of agents operating under partial observability to effectively explore and perform tasks.

4.2.1 Method 1: Online Fine-Tuning

The first method involves utilizing online interactions with the environment to fine-tune the agent. In this approach, we take the agent pre-trained on behavior cloning starting from the left room, place it at the midpoint of the hallway and fine-tune using Proximal Policy Optimization (PPO). Through online fine-tuning, we hope that the agent takes advantage of the navigation and memory skills learned from the pretraining phase, and is more sample-efficient than online RL without pre-training.

4.2.2 Method 2: History Imitation

This method operates on the hypothesis that trajectories that interact with or observe the object of interest inherently encode critical information within their observation embedding vectors.

The approach begins by training an encoder capable of representing observation histories effectively, i.e. our pre-trained BC agent. Next, an agent, referred to as the Exploration Network, is trained to imitate these history embeddings without relying on environmental rewards. The objective is to enable the agent to explore by replicating encoded historical patterns. Starting from a midpoint in the environment, the agent attempts to recreate the encoded history. However, if the object of interest (e.g., on the left) has not been observed, it becomes impossible to generate a comparable history embedding.

The Generative Adversarial Imitation Learning (GAIL) framework is employed to train both the discriminator and the Exploration Network (the generator). The reward for the generator is defined as the similarity score between the expert's history embeddings and the agent's history embeddings, eliminating the dependence on any environment-derived rewards.

4.2.3 Method 3: Collect Better Data

The third method is based on [1], this method is designed to solve partially observable behavior cloning from a fully observable expert. In this method, expert data are exactly what the agent will see during the testing time because the expert starts from the aisle, just like the agent. The problem is that the agent is partially observable, so the agent can only guess an answer if it directly clones the fully observable expert. Our idea is to let the expert learn to collect better data to fit the agent's behavior as follows:

$$\text{Minimize } KL(\pi_{\hat{\theta}^*}(a|s), \pi_{\psi}(a|o))$$

based on the reward of the agent interacting with the environment.

However, following the previous two methods, the DP expert is also used in this method, meaning that the DP can not be updated using gradient descent as in the previous work. As a result, we create a new expert and a corresponding reward-learnable environment to better fit the agent. There are two experts now, one for the original environment, and the other for the reward-learnable environment.

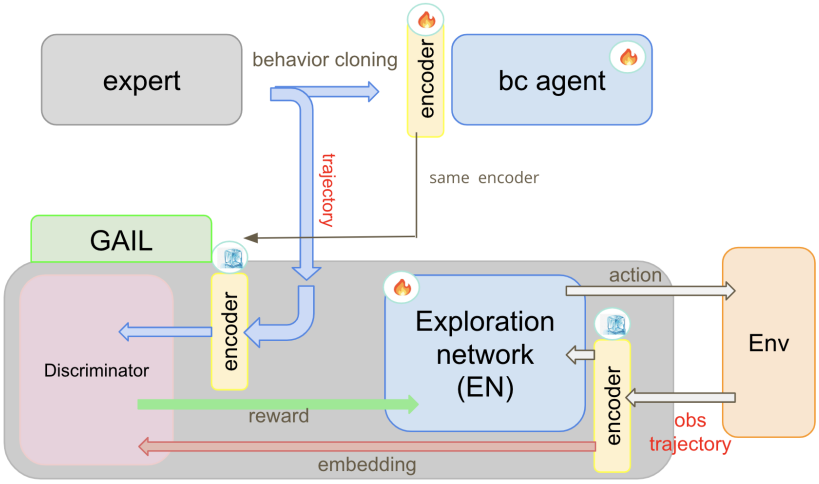


Figure 3: Visualization of the History Imitation process. The image illustrates how historical trajectories are encoded and used to train the agent in partially observable environments.

We will use the agent's expert chooser of the agent to decide which of the experts is used for data collection based on the agent's encoder.

The two experts first do DP on their corresponding environment, then the expert chooser will decide which of the experts to collect the data. The expert data will be used to update the agent's encoder and policy network, and then the agent will interact with the real environment and the reward will be used to update the expert chooser and the reward learnable environment.

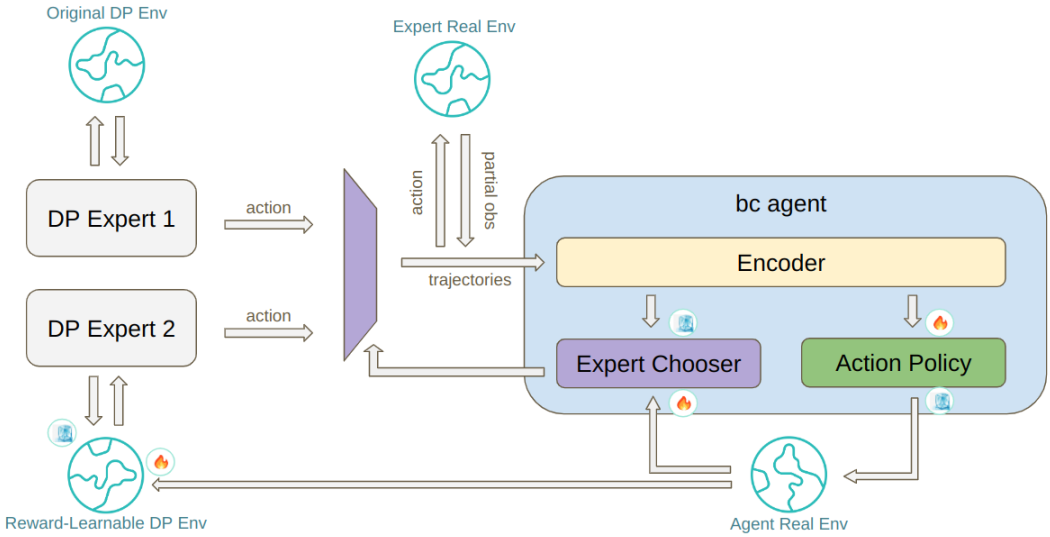


Figure 4: Visualization of the Collect Better Data process. Two DP experts first DP with their environments, and then the agent uses its expert chooser to choose one of the experts to interact with the environment and collect the corresponding partial observation. After

5 Experiment

5.1 Observation Encoding

We evaluate three different approaches for history encoding: LSTM, Transformer, and frame stacking, and compare them to the baseline of a single-state representation. We train the agent on the

`MiniGrid-DoorKey-8x8-v0` and `MiniGrid-MemoryS11-v0` environment in Minigrid and use the agent's episode success rate as the evaluation metric. In the Door Key task, all encoders are able

(a) Door Key (b) Memory

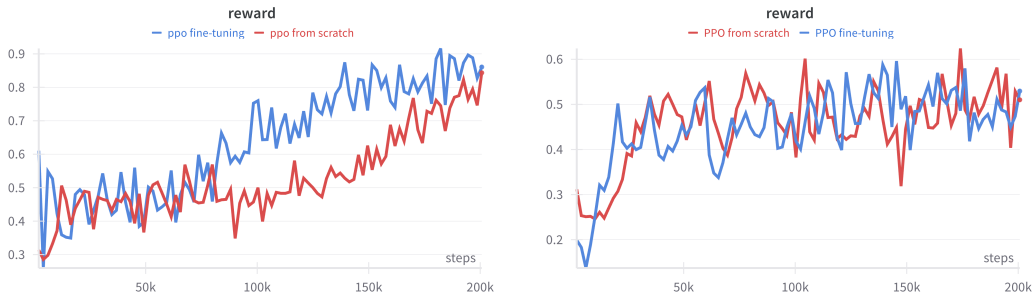
Figure 5: Comparison of encoder performance across two MiniGrid tasks.

to reach optimal play in approximately the same number of epochs, with transformer being the fastest. We also observed that our encoding strategies showed better training stability compared to a single-state. In the Memory task, all encoders reach around 50% success rate within the first few epochs, indicating that the agent has learned to navigate to the goal. However, frame-stacking was able to reach 100% success rate in 100 epochs, followed by transformer in 200 epochs, then LSTM in about 1000 epochs. Despite LSTM taking the most epochs to reach perfect play, it is by far the fastest in terms of training time due to its ability to efficiently handle sequential data.

5.2 Exploration

5.2.1 Method 1: Online Fine-Tuning

Since frame-stacking showed the best results from our previous experiment and is easiest to implement. We take its trained BC policy and fine-tune with PPO. We compare its sample efficiency and performance with PPO training from scratch. We run this experiment on two Memory environments, `MiniGrid-MemoryS7-v0` and `MiniGrid-MemoryS11-v0`, which have sizes 7 and 11 respectively. Our results show that PPO fine-tuning does not significantly outperform training PPO from scratch,



(a) `MiniGrid-MemoryS7-v0` (b) `MiniGrid-MemoryS11-v0`

Figure 6: Performance of PPO fine-tuning vs PPO from scratch.

nor is it more sample-efficient. We observed that PPO fine-tuning in the smaller environment performed well in the initial steps, indicating that it was able to translate some of the skills learned from the pre-training phase. However, it quickly suffered from *policy collapse*[2], where the critic becomes overoptimistic in unseen states in the initial phase of training. As a result, the performance of both agents after 200k steps is similar, with the pre-trained agent slightly outperforming the other. As for the larger 11x11 environment, the agent is unable to make use of the pre-trained skills, presumably due to the larger map size, and both approaches show a similar learning curve.

5.2.2 Method 2: History Imitation

While the concept seemed intuitively sound, our experiments revealed that the Exploration Network primarily learns only the latter part of the expert's history. This occurred because the discriminator became too strong too quickly. As a result, the agent consistently moved to the right and randomly chose a side, achieving only a 50% success rate.

Analyzing the reward provided by the discriminator, we observed that it frequently overpowered the generator. We suspect this issue arises from the agent starting in the middle of the environment, which makes it relatively easy for the discriminator to differentiate between the agent's and the expert's trajectories. Introducing a threshold for the discriminator's effectiveness could potentially mitigate this issue and lead to better performance. Although success rate did not improve, about 20% of the samples traverse to the regions on the left, which shows potential for better exploration.

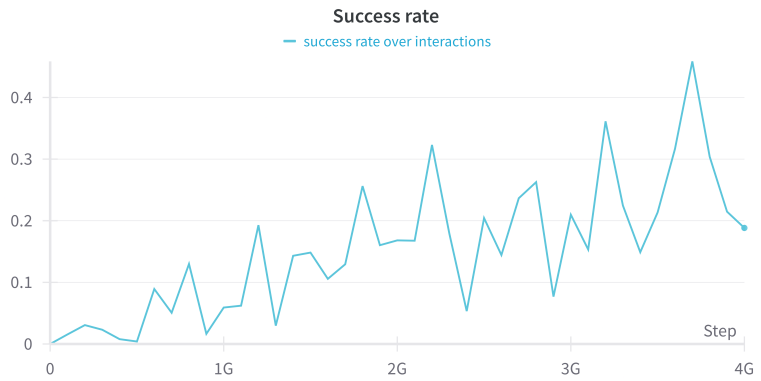


Figure 7: This figure shows how method 2 only achieves 50% success rate

5.2.3 Method 3: Collecting Better Data

For simplicity, we replace the expert chooser by first collecting data using the original expert with constant steps, then using the other expert to go to the correct terminal. To discourage the expert from collecting data that cannot help the agent to do better than randomly guessing, we modify the reward of the original environment so that arriving at the wrong terminal gets a negative reward, instead of 0. For the reward-learnable environment, we try different reward-shaping methods as follows.

1. **Assign reward based on state-action pairs** We first try to provide the reward from the real environment to all of the state-action pairs from interactions with the reward-learnable environment as follows:

   $$\text{Reward}(s_t, a_t) = \gamma \times \text{Reward}(s_{t-1}, a_{t-1}) + r\,,$$

   where r denotes the reward of the trajectory from the real environment.

   However, we find that the reward seems too sparse for the expert to learn to go left to see the target.

2. **Assign reward based on states** We then try to add the reward to all the states from interactions the reward-learnable environment as follows:

   $$\text{Reward}(s_t) = \gamma \times \text{Reward}(s_{t-1}) + r\,,$$

   where s_t denotes the state the expert enters.

   We find that the reward-learnable environment tends to forget what it learns from the interaction if the discount factor is too small. We find that the expert can learn to go left if the discount factor is more than 0.9.

After letting the expert go left and then go right, we want to start to study the expert chooser. However, we find that the agent still can not learn to go to the correct terminal even if the agent has learned to go left first. We speculate the issue is that it is too hard for the encoder to learn to remember if the target does not appear at a regular position. As a result, it will definitely fail if we directly start to study the expert chooser (the expert chooser is based on the encoder).

6 Conclusion

In this work, we addressed the challenges of imitation learning in Partially Observable Markov Decision Processes (POMDPs). We evaluated various observation encoding architectures in two MiniGrid environments, DoorKey and Memory. Our results demonstrate that observation encoders outperform single-state representations in terms of both training efficiency and overall performance. Additionally, we introduced three methods to facilitate exploration in partially observable settings. While these methods did not achieve better success rate than online RL training, these strategies are able to guide the agent to traverse states not present in the expert data.

Future work includes building upon our framework to develop more robust behavior cloning agents with improved sample efficiency, or applying our framework to other partially observable environments, such as robotics, autonomous driving, and environments with continuous action spaces.

7 References

[1] Andrew Warrington and J. Wilder Lavington and Adam Ścibior and Mark Schmidt and Frank Wood. "Robust Asymmetric Learning in POMDPs". In: *CoRR* abs/2012.15566 (2020). DOI: https://doi.org/10.48550/arXiv.2012.15566.

[2] Anonymous. "Finetuning from Offline Reinforcement Learning: Challenges, Trade-offs and Practical Solutions". In: *Submitted to Transactions on Machine Learning Research* (2023). Rejected. URL: https://openreview.net/forum?id=PffrUvAOGy.

[3] Farama Foundation. *Minigrid Environments*. URL: https://minigrid.farama.org/environments/minigrid/. (accessed: 12.29.2024).

[4] Tanmay Gangwani and Joel Lehman and Qiang Liu and Jian Peng. "Learning Belief Representations for Imitation Learning in POMDPs". In: *CoRR* abs/1906.09510 (2019). DOI: https://doi.org/10.48550/arXiv.1906.09510.

[5] Yue Wang and Swarat Chaudhuri and Lydia E. Kavraki. "Bounded Policy Synthesis for POMDPs with Safe-Reachability Objectives". In: *CoRR* abs/1801.09780 (2018). DOI: https://doi.org/10.48550/arXiv.1801.09780.