

---

# RecoveryDAgger: Query-Efficient Online Imitation Learning Through Recovery Policy

---

**Wen-Ai Ho**

Department of Electrical Engineering  
National Taiwan University  
Taipei, Taiwan  
b13901006@ntu.edu.tw

**Sheng-Yu Cheng**

Department of Electrical Engineering  
National Taiwan University  
Taipei, Taiwan  
b13901088@ntu.edu.tw

**Sheng-Hsun Chang**

Department of Electrical Engineering  
National Taiwan University  
Taipei, Taiwan  
b13901165@ntu.edu.tw

**Aaron Wu**

Department of Electrical Engineering  
National Taiwan University  
Taipei, Taiwan  
b13901011@ntu.edu.tw

## Abstract

Imitation learning enables agents to acquire complex behaviors from expert demonstrations, yet standard behavior cloning (BC) often suffers from covariate shift and compounding errors in sequential decision-making tasks. Interactive methods such as DAgger alleviate this issue by querying the expert on states visited by the learner, but they typically require frequent expert supervision, resulting in high annotation cost.

In this work, we propose RecoveryDAgger, a query-efficient interactive imitation learning framework that augments DAgger-style training with a learned recovery mechanism. Instead of immediately querying the expert in risky states, RecoveryDAgger first invokes a recovery policy that locally corrects the agent’s behavior by ascending the gradient of a learned Success Q-function, which estimates the probability of task completion. Expert queries are reserved for truly novel states where recovery is unreliable, thereby reducing redundant supervision. Experiments on the PointMaze navigation task demonstrate that RecoveryDAgger significantly reduces the number of expert queries while achieving comparable success rates to strong query-efficient baselines. Our work establishes the effectiveness of integrating learned recovery policies into interactive imitation learning to enhance query efficiency.

## 1 Introduction

Imitation learning has become a widely adopted paradigm for training agents to perform complex tasks by leveraging expert demonstrations [1, 2]. Among existing approaches, Behavior Cloning (BC) is one of the most commonly used methods, where a policy is trained in a supervised manner on an offline dataset collected from expert trajectories. While effective in simple settings, BC often struggles in sequential decision-making problems due to its covariate shift issue. When the agent deviates from the expert’s trajectory and encounters states that are not covered by the offline dataset, prediction errors can accumulate over time, leading to compounding errors and ultimate task failure [3, 4].

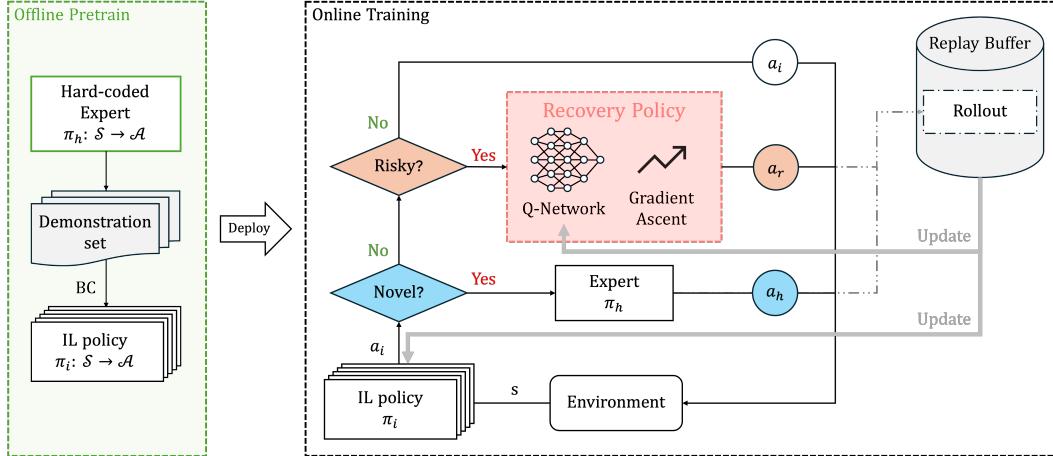
To mitigate this issue, interactive imitation learning methods were introduced to address this limitation. [5] A representative example is DAgger (Dataset Aggregation) [6], which allows the agent

to query an expert online and collect corrective demonstrations on states visited by the learned policy. Although this strategy significantly improves robustness, it introduces a new challenge: query inefficiency [7]. In DAgger, the expert is queried at every time step, which can be impractical when expert annotations are costly and time-consuming.

To reduce the reliance on frequent expert queries, a growing body of work has explored query-efficient variants of DAgger that selectively request expert supervision based on state-dependent criteria [8, 9, 10, 11], such as uncertainty estimates [9], prediction discrepancies, or risk measures [10]. These methods aim to identify critical states where expert guidance is necessary, thereby lowering annotation costs while maintaining performance.

However, existing query-efficient online imitation learning methods typically treat expert supervision as the only form of intervention once their querying criterion is triggered, regardless of how that criterion is defined. In practice, this assumption is overly restrictive. Even when the agent encounters risky states, the agent can instead be corrected by an auxiliary recovery policy, without immediate querying the expert.

Motivated by this insight, we propose RecoveryDAgger, a novel interactive imitation learning framework that augments DAgger-style training with a recovery mechanism. Its architecture is illustrated in Figure 1. Instead of immediately querying the expert when encountering a risky state, RecoveryDAgger first attempts to correct the agent’s behavior using a learned recovery policy. Expert queries are therefore reserved for unseen states or irrecoverable situations where the recovery policy is insufficient, thereby reducing redundant expert supervision while maintaining performance.



**Figure 1: Overview of the proposed framework.** An expert policy first generates an offline demonstration dataset, which is used to pretrain the imitation learning (BC) policy. During online training, the agent interacts with the environment using the IL policy. When a state is detected as risky, a recovery policy optimizes an action via gradient ascent on the Q-network; when a state is identified as novel, the agent queries the expert policy. All executed state-action pairs are stored in the replay buffer and are subsequently used to update both the Q-network and the IL policy.

Our contributions can be summarized as follows:

- We propose RecoveryDAgger, a query-efficient interactive imitation learning method that integrates a gradient-based recovery policy to address risky states without immediate expert intervention.
- We provide visualizations and analyses of states where the agent invokes the recovery policy versus querying the expert, offering insights into how RecoveryDAgger improves query efficiency.
- We empirically demonstrate that RecoveryDAgger significantly reduces expert queries while achieving comparable performance to existing query-efficient baselines.

## 2 Related works

### 2.1 Query-Efficient DAgger Variants

**Discrepancy-based Methods.** Discrepancy-based approaches aim to reduce the expert burden in DAgger by estimating the deviation between the learner and the expert, and only querying the expert when this deviation exceeds a predefined threshold. A representative method is **SafeDAgger** [10], which trains a safety classifier to predict the discrepancy  $\epsilon = \|\pi(s) - \pi^*(s)\|^2$  in the action space and triggers expert intervention when  $\epsilon > \tau$ .

However, a large discrepancy does not necessarily imply that expert intervention is required. In unconstrained or low-risk regions (e.g., free space), imprecise actions may still lead to successful outcomes, whereas in constrained regions such as bottlenecks, even small deviations can result in failure. [12] Moreover, employing a fixed global threshold fails to capture the state-dependent nature of safety. This highlights a fundamental limitation of discrepancy-based criteria: they measure local action disagreement without directly accounting for state-dependent risk or task-level consequences, which can lead to suboptimal querying decisions.

**Uncertainty-based Methods.** Uncertainty-based approaches aim to reduce expert queries by estimating the uncertainty of the learner and intervening when the agent is insufficiently confident about its actions. A representative method is **EnsembleDAgger** [9], which maintains an ensemble of  $K$  policies trained with different data shuffles and monitors the variance among their predicted actions. High variance across the ensemble indicates elevated uncertainty, suggesting that the agent lacks sufficient knowledge about the current state and that expert intervention may be beneficial. Accordingly, the expert is queried in states that are either deemed unsafe or exhibit high uncertainty.

Recent works like [13] and [14] also focus on monitoring uncertainty to detect possible failures. The former trains a scalar signal to capture the uncertainty of the agent, while the latter uses MC-dropout to estimate uncertainty. Collectively, these methods demonstrate that uncertainty estimation provides informative signals about the agent’s epistemic knowledge of the current state and the following decision can be made accordingly.

Nonetheless, uncertainty-based criteria alone may lead to conservative querying behavior, as high uncertainty does not always imply imminent task failure or safety violations, particularly in regions that are rarely visited but low-risk.

**Risk-based Methods.** More recently, ThriftyDAgger [11] was developed, which combines the epistemic uncertainty mentioned above and success rate to form the query criteria. By utilizing a Q-network to predict the probability of future task completion, ThriftyDAgger ensures expert intervention occurs only during novel or high-risk scenarios.

Subsequent work has adopted ThriftyDAgger as a representative robot-gated imitation learning baseline for evaluating query-efficient methods. For instance, [15] proposed the Adaptive Intervention Mechanism (AIM) and compare it with ThriftyDAgger in both continuous and discrete environments. It shows that ThriftyDAgger has been accepted as a strong baseline in query-efficient imitation learning community.

Nevertheless, ThriftyDAgger and similar works treat expert query as the only form of intervention, ignoring the potential of leveraging information from previously visited states and executed actions to guide recovery.

### 2.2 Recovery Methods

Beyond the scope of imitation learning, safety has always been a main issue in reinforcement learning [16, 17]. Some works adopt the concept of recovery, which aims at guiding the agent back on track [18]. One of the promising method is RecoveryRL [19], which decomposes the agent into two policies:  $\pi_{task}$  and  $\pi_{rec}$ . By leveraging a safety critic  $Q_{risk}$  to identify whether the current state is unsafe, the agent switches between the two policies, avoiding unsafe regions while completing tasks simultaneously.

The notion of unsafe states in RecoveryRL closely aligns with the risky states identified in risk-aware imitation learning methods such as ThriftyDAgger. Inspired by this connection, we incorporate a

model-free recovery mechanism into query-efficient imitation learning by performing gradient-ascent on  $Q_{risk}$  intuitively. Through guiding the agent back to safe regions with local gradient, our method precludes redundant expert queries and enhances query efficiency.

### 3 Problem formulation

We formulate the imitation learning problem as a Markov Decision Process (MDP) defined by the tuple  $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0, T)$ , where  $\mathcal{S}$  is the state space,  $\mathcal{A}$  is the action space,  $P(s'|s, a)$  is the transition probability from state  $s$  to  $s'$  given action  $a$ ,  $r(s, a)$  is the reward function,  $\gamma \in [0, 1]$  is the discount factor,  $\rho_0$  is the initial state distribution, and  $T$  is the time horizon.

At every time step during training, an expert policy  $\pi_{exp} : \mathcal{S} \rightarrow \mathcal{A}$  and a recovery policy  $\pi_{rec} : \mathcal{S} \rightarrow \mathcal{A}$  are available during training. At each time step  $t$ , the agent has three choices:

$$a_t = \begin{cases} \pi_h(s_t) & \text{if Query Condition Met (e.g., } q_t = 1) \\ \pi_r(s_t) & \text{if Recovery Condition Met} \\ \pi_i(s_t) & \text{otherwise} \end{cases} \quad (1)$$

The mechanism for selecting among these action sources is specified by the query and recovery criteria described later. We define a query indicator variable  $q_t \in \{0, 1\}$  to denote whether the expert is queried at time  $t$ , where

$$q_t = \begin{cases} 1, & \text{if the expert is queried at time } t, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

and a total query cost over an episode as

$$C(\pi_\theta) = \mathbb{E} \left[ \sum_{t=0}^T q_t \right]. \quad (3)$$

The goal of RecoveryDagger is to learn a policy  $\pi$  that maintains high task performance while minimizing the total query cost  $C$ .

$$\max_{\pi_\theta} J(\pi_\theta) \quad \text{s.t.} \quad C(\pi) \leq B, \quad (4)$$

where  $J(\pi)$  denotes the expected task performance, such as the success probability or the expected return under policy  $\pi$ .

### 4 Preliminary

ThriftyDAgger [11] is an interactive imitation learning framework which narrows down expert queries to only novel and risky states.

To identify novel states, ThriftyDAgger trains an ensemble of policies  $\{\pi_i\}_{i=1}^K$  with bootstrapped datasets from expert demonstrations. Each policy outputs an action  $a_i = \pi_i(s)$  given the current state  $s$ , and the variance among the actions indicates the epistemic uncertainty of the agent. The novelty score is defined as the variance of the ensemble actions:

$$\mathcal{N}(s) = \frac{1}{K} \sum_{i=1}^K \|a_i - \bar{a}\|^2, \quad \bar{a} = \frac{1}{K} \sum_{i=1}^K a_i. \quad (5)$$

To identify risky states, ThriftyDAgger employs a goal-conditioned action-value function  $Q_G^\pi(s, a)$  that estimates the probability of successfully completing the task when taking action  $a$  in state  $s$  under policy  $\pi$ , where

$$Q_G^\pi(s_t, a_t) = \mathbb{E}_\pi \left[ \sum_{t'=t}^{\infty} \gamma^{t'-t} \mathbb{1}_G(s_{t'}) \mid s_t, a_t \right]. \quad (6)$$

Here,  $\mathbb{1}_G(s)$  is an indicator function that equals 1 if the goal state is reached at state  $s$  and 0 otherwise. The risk score is defined as:

$$\mathcal{R}(s, a) = 1 - Q_G^\pi(s, a). \quad (7)$$

A state  $s$  is classified as *novel* if  $\mathcal{N}(s) > \delta_h$ , and *risky* if  $\mathcal{R}(s, a) > \beta_h$ .

## 5 Method

In this section, we first propose our RecoveryDAgger framework. Next, we introduce how to utilize Success Q, a rename for goal-conditioned action-value function, as recovery policy. Finally, we introduce Ensemble Success Q, which is an ensemble of multiple Success Q function. We also state how to utilize Ensemble Success Q as novel detector, risk detector as well as recovery policy.

### 5.1 RecoveryDAgger framework

As illustrated in Fig. 1, RecoveryDAgger comprises three main components: an imitation learning policy  $\pi_i$ , a recovery policy  $\pi_r$ , and an expert policy  $\pi_h$ . The IL policy is responsible for normal decision making and is trained via offline behavior cloning on the expert demonstration dataset.

During online training, at each time step, the agent first evaluates whether the current state is novel, which refers to out-of-distribution with respect to the training data. If the state is identified novel, the agent queries the expert policy  $\pi_h$  for supervision. If the state is not novel, the agent then assesses whether the state is risky using a risk detector based on the Success Q function. Risky states are states that are within the training distribution but may lead to task failure if executed naively by the IL policy. If the state is deemed risky, the agent invokes the recovery policy  $\pi_r$  to correct its behavior and guide it back to safer regions. Only when the state is neither novel nor risky does the agent execute the action from the IL policy  $\pi_i$ .

All transitions generated by these policies are collected in the replay buffer and used to iteratively update the IL policy and the Success Q function during online training. By integrating the recovery mechanism, RecoveryDAgger aims to reduce redundant expert queries in risky states while maintaining robust performance through expert supervision in novel states.

### 5.2 Success Q as recovery policy

Notice that the Success Q function represents the probability of successfully reaching the goal state after executing action under the current state. Under a risky but non-novel state, the objective of recovery is therefore to select an action that maximizes the predicted success probability, so as to steer the agent back to a safe region. For discrete action spaces, the recovery action can be obtained by directly enumerating all candidate actions and selecting the one with the highest Success Q value:

$$a_{\text{recovery}} = \arg \max_{a \in \mathcal{A}} Q_{\phi, \mathcal{G}}^{\pi}(s, a) \quad (8)$$

For continuous action spaces, explicitly enumerating all actions is infeasible. Instead, we exploit the fact that the gradient of the Success Q function with respect to the action indicates the direction toward higher success probability. Starting from the action proposed by the imitation policy, the recovery action can be obtained via iterative gradient ascent:

$$a_{i+1} = \text{projection}(a_i + \eta \nabla_a J(s, a_i), [a_{\min}, a_{\max}]), \quad i = 0, 1, 2, \dots, n \quad (9)$$

where

$$J(s, a) = Q_{\phi, \mathcal{G}}^{\pi}(s, a), \pi_r = a_n \quad (10)$$

and  $a_0$  is initialized as the action output by the robot policy and update it for  $n$  iterations in total. The projection guarantees the output action remains within the valid action set.

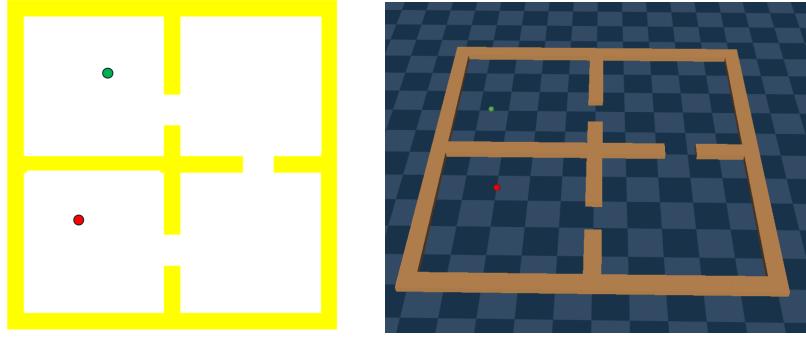
### 5.3 Ensemble Success Q

Since  $Q_{\phi, \mathcal{G}}^{\pi}$  is an approximation to  $Q_{\mathcal{G}}^{\pi}$ , using an ensemble of Success Q function simultaneously can provide a more reliable estimate. We construct an ensemble consisting of  $N$  independently trained Success Q functions  $\{Q_{\phi, \mathcal{G}, n}^{\pi}\}_{n=1}^N$ . The ensemble mean is defined as:

$$\bar{Q}_{\phi, \mathcal{G}}^{\pi}(s_t, a_t) = \frac{1}{N} \sum_{n=1}^N Q_{\phi, \mathcal{G}, n}^{\pi}(s_t, a_t) \quad (11)$$

The ensemble mean  $\bar{Q}$  provides a better approximation of the true success probability and is therefore used to define risk in the same manner as the single Success Q case:

$$\text{Risk}(s, a) = 1 - \bar{Q}_{\phi, \mathcal{G}}^{\pi}(s, a) \quad (12)$$



(a) Four-room map in 2D

(b) Four-room map in 3D

Figure 2: This figure illustrates the maze layout; the green spot denotes the start point and the red spot denotes the end point. The right panel shows the real 3D environment of PointMaze.

In addition to improved risk estimation, the ensemble structure allows us to quantify uncertainty in the Success Q prediction. Intuitively, estimating the success probability of the current  $(s, a)$  requires the model to have sufficient understanding of the resulting next state. If the next state is out-of-distribution, different models in the ensemble may yield divergent predictions due to epistemic uncertainty. In such cases, the transition should be considered novel. We therefore define novelty of the transition as the variance across the ensemble:

$$\text{Novel}(s, a) = \text{Var}(\bar{Q}_{\phi, \mathcal{G}}^{\pi}(s, a)) = \frac{1}{N} \sum_{n=1}^N (Q_{\phi, \mathcal{G}, n}^{\pi}(s, a) - \bar{Q}_{\phi, \mathcal{G}}^{\pi}(s, a))^2 \quad (13)$$

Heuristically, under a risky but non-novel state, the recovery policy aims to select an action that not only maximizes the predicted success probability, but also avoids transitions leading to poorly understood next states. This can be achieved by maximizing the ensemble mean of the Success Q function while penalizing high ensemble variance. Consequently, the recovery policy gets similar structure to the iterative recovery action finding algorithm stated in the section Success Q as recovery policy with

$$J(s, a) = \bar{Q}_{\phi, \mathcal{G}}^{\pi}(s, a) - \lambda \text{Var}(\bar{Q}_{\phi, \mathcal{G}}^{\pi}(s, a)), \quad (14)$$

where  $\lambda$  is a predetermined Hyperparameter controlling the trade-off between success maximization and novelty avoidance.

## 6 Experiment

### 6.1 Experimental settings

**Environment** We evaluate our methods on the PointMaze environment from Gymnasium [20]. This environment allows flexible design of navigation tasks, and we construct a 4-room navigation task, as illustrated in Figure 2. In this setting, the agent is required to navigate from a given start position to a target goal while passing through narrow corridors and avoiding collisions with the walls.

Under offline behavioral cloning (BC) training, small action errors can accumulate over time and eventually lead to task failure, making this environment particularly sensitive to compounding errors. As a result, PointMaze is well-suited for studying online imitation learning, where expert intervention is mainly required when the agent deviates into unseen states caused by accumulated errors or encounters locally risky states that are critical for successful navigation.

The PointMaze environment features a continuous state space consisting of the agent’s 2D position and velocity, and a continuous action space corresponding to the forces applied along the x- and y-axes. The task adopts a sparse reward structure: a positive reward is provided only when the agent successfully reaches the goal. Episodes terminate upon goal achievement, collision with a wall, or exceeding the time limit.

**Expert Policy & Offline Dataset.** The expert policy is implemented with a rule-based controller that gives actions toward the goal based on the current state. The expert is deterministic and achieves a 100% success rate in the PointMaze task. To train the initial BC policy, we collect an offline dataset consisting of 100 expert trajectories with random start points in the initial room. We empirically find that 100 expert trajectories are sufficient to learn a reasonable initial BC policy for this environment with success rate around 20% to 30%.

**Evaluation Metrics.** We evaluate the performance of different methods based on two metrics:

1. Success Rate: the percentage of episodes where the agent successfully reaches the goal within the time limit without colliding with walls.
2. Total Expert Queries: the average number of expert queries made per episode during training.

To highlight the trade-off between task performance and expert burden, we plot success rate against total expert queries for each method.

**Implementation Details.** All policies and critics are parameterized by multilayer perceptrons. The safety critic uses an MLP with architecture [10, 256, 256, 1] with ReLU activations and Sigmoid output. Novelty estimation employs an ensemble of  $K = 5$  policy networks with identical architectures.

All methods are initialized using behavior cloning on an offline expert dataset (split 9:1 for training and validation). The BC policy is trained using Adam ( $\eta = 10^{-3}$ ), batch size 100, and 5 epochs. The safety critic replay buffer is initialized with BC data and trained with batch size 50, positive sample fraction 0.1, discount factor  $\gamma = 0.9999$ , and learning rate  $10^{-3}$ .

During online training, agents run for 30 iterations, collecting 2000 environment steps per iteration and evaluating performance using 100 test episodes. Novelty and risk thresholds are initialized from offline data and updated online.

## 6.2 Main Result

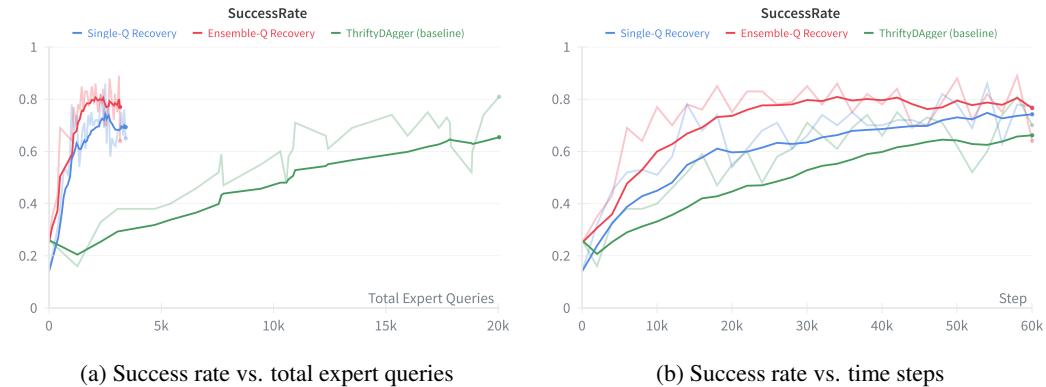


Figure 3: In our experiments on the PointMaze environment, we compare Single-Q recovery and Ensemble-Q recovery against the baseline ThriftyDagger.

We illustrate the overall performance of all methods in Figure 3. All methods start from the same BC-pretrained policy, starting at a success rate of approximately 0.2 to 0.3. We compare ThriftyDagger (baseline) with our Single-Q Recovery and Ensemble-Q Recovery. Table 1 summarizes the success rate and the number of expert queries for each method. Results are collected at the training step corresponding to the highest success rate achieved within 30 epochs of online training.

Figure 3a plots success rate against total number of expert queries. Both recovery methods are significantly more query efficient: they reach a success rate of around 0.7 with 2k expert queries, whereas ThriftyDagger requires roughly 20k queries to approach the same performance. Among the two recovery variants, Ensemble-Q recovery improves faster in the early stage and remains

Table 1: Performance comparison on the task.

Method	Training			Testing
	Expert Queries	Recovery Queries	Total Steps	Success Rate
BC	-	-	-	0.26
ThriftyDAgger	20048	-	58029	0.81
Single-Q Recovery (Ours)	2504	9232	54027	0.86
Ensemble-Q Recovery (Ours)	2693	9296	50025	0.88

consistently higher than Single-Q recovery over most of the query budget, showcasing a more reliable recovery decision.

Figure 3b shows the success rate as a function of environment interaction steps. Both Single-Q Recovery and Ensemble-Q Recovery consistently outperform ThriftyDAgger throughout training, achieving higher success rates with fewer environment interactions. In particular, Ensemble-Q Recovery exhibits faster performance improvement and stabilizes at a higher success rate earlier in training. These results indicate that our recovery-based methods improve learning efficiency while significantly reducing reliance on expert queries, without sacrificing final task performance.

In conclusion, the result supports our key claim : we reduce unnecessary expert queries without sacrificing task performance.

### 6.3 Mechanism Analysis

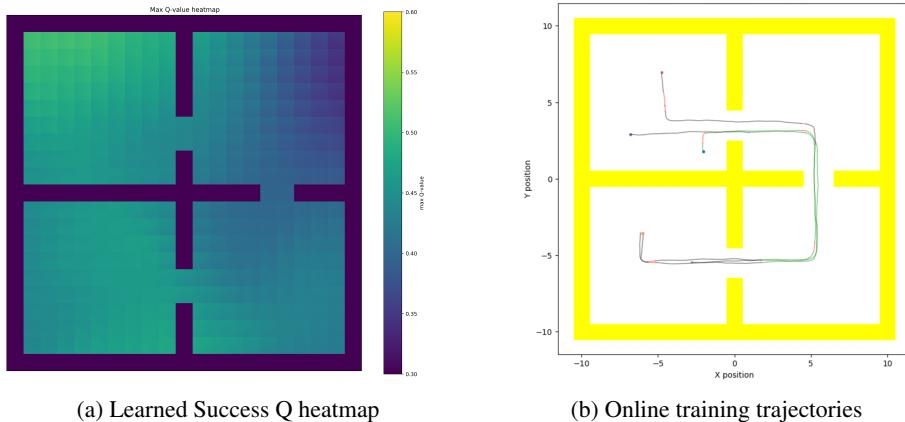


Figure 4: **Visualization of the learned Success Q function and its effect on online training deployment.** Figure 4a shows the learned Success Q function over the 2D state space. Figure 4b depicts the online training trajectories, where black trajectories correspond to the IL agent’s executed paths during training, where red segments indicate novel states queried to the expert and green segments denote recovery actions.

Figure 4a visualizes the learned Success Q-values over the state space, which estimate the likelihood of eventual task success. The higher Q-values are concentrated in task-relevant regions and gradually decrease toward riskier or less favorable states, indicating that the Success Q function effectively captures meaningful progress signals toward task completion. The upper-left region exhibits higher max Q-values, as states in this area are well covered by the random-starting-point expert demonstrations.

The effect of this learned value function is illustrated in Figure 4b, which shows representative online training trajectories. When the agent enters low-value regions, recovery actions guide the trajectory back toward higher-value areas by following value gradients. Notice that when the agent encounters novel states where the Success Q function provides insufficient or unreliable estimates, expert intervention is still required to supply corrective supervision.

Together, these visualizations show that the learned Success Q-function provides reliable guidance for recovery, enabling the agent to correct suboptimal behavior on its own during training.

## 7 Discussion

### 7.1 Why RecoveryDAgger improves query efficiency

Our results suggest that a significant portion of expert queries in existing query-efficient imitation learning methods is spent on recoverable risky states rather than genuinely novel states, which does not fully utilize the power of online imitation learning. In environments such as PointMaze, many failures arise from local deviations—e.g., near gate between rooms or wall boundaries—where the agent remains within the training distribution but requires fine-grained corrective actions.

By introducing a recovery policy optimized directly on the Success Q function, RecoveryDAgger enables the agent to locally ascend toward safer regions, namely the region with highest success rate, without immediately querying the expert. This mechanism effectively decouples novelty handling from risk mitigation, allowing expert supervision to be reserved for states that are truly falling out of distribution of behavior cloning dataset. This explains why RecoveryDAgger achieves comparable success rates with substantially fewer expert queries, especially in later stages of online training.

### 7.2 Relationship to Prior Query-Efficient DAgger variants

Unlike prior query-efficient DAgger variants such as SafeDAgger and ThriftyDAgger, which treat expert querying as the sole response to unsafe or uncertain states, RecoveryDAgger introduces an intermediate recovery mechanism. While discrepancy-based, uncertainty-based, and risk-based criteria remain useful for detecting critical states, our results indicate that querying the expert is not always necessary once such a state is detected. In many cases, a locally optimal corrective action can be derived from a learned Success Q function.

From this perspective, RecoveryDAgger can be viewed as an orthogonal extension to existing query-efficient frameworks: it does not replace their query criteria, but instead augments them with a learned recovery option that further reduces annotation cost.

### 7.3 Limitations

Despite its advantages, RecoveryDAgger has several limitations. First, the effectiveness of the recovery policy depends on the quality of the learned Success Q function. In highly non-smooth or sparse-reward environments, inaccurate gradients may lead to suboptimal recovery actions. Second, the gradient-based recovery procedure introduces additional computational overhead at each time step, which may limit scalability in environments with high-dimensional action spaces or strict real-time constraints. Finally, RecoveryDAgger assumes that risky states are locally recoverable. In tasks where failures are irreversible or require long-horizon planning, recovery alone may be insufficient, and expert intervention remains necessary.

## 8 Conclusion

In this paper, we introduced RecoveryDAgger, a query-efficient online imitation learning framework that integrates a learned recovery policy into the DAgger paradigm. By leveraging a Success Q function to locally correct risky behaviors, RecoveryDAgger reduces redundant expert queries while maintaining strong task performance. Empirical results on the PointMaze environment demonstrate that many expert interventions in existing methods can be effectively replaced by recovery actions derived from learned value gradients.

This work highlights the importance of distinguishing between novelty and recoverability in interactive imitation learning, which potentially reduces expert burden through learned corrective policy. Future work includes deploying RecoveryDAgger on real-world robotic systems, where intervention cost, latency, and safety constraints play a critical role. Extending the framework to long-horizon and safety-critical settings will also be an interesting avenue for future research.

## References

- [1] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

- [2] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.
- [3] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.
- [4] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3(1):297–330, 2020.
- [5] Yigit Korkmaz and Erdem Bıyık. Mile: Model-based intervention learning. *arXiv preprint arXiv:2502.13519*, 2025.
- [6] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.
- [7] Zhenghao Mark Peng, Wenjie Mo, Chenda Duan, Quanyi Li, and Bolei Zhou. Learning from active human involvement through proxy value propagation. *Advances in neural information processing systems*, 36:77969–77992, 2023.
- [8] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.
- [9] Kunal Menda, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Ensembledagger: A bayesian approach to safe imitation learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5041–5048. IEEE, 2019.
- [10] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.
- [11] Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. *arXiv preprint arXiv:2109.08273*, 2021.
- [12] Hanbit Oh, Masaki Murooka, Tomohiro Motoda, Ryoichi Nakajo, and Yukiyasu Domae. Self-augmented robot trajectory: Efficient imitation learning via safe self-augmentation with demonstrator-annotated precision. *arXiv preprint arXiv:2509.09893*, 2025.
- [13] Chen Xu, Tony Khuong Nguyen, Patrick Miller, Robert Lee, Paarth Shah, Rares Andrei Ambrus, Haruki Nishimura, and Masha Itkina. Uncertainty-aware failure detection for imitation learning robot policies. In *CoRL Workshop on Safe and Robust Robot Learning for Operation in the Real World*.
- [14] Yuchen Cui, David Isele, Scott Niekum, and Kikuo Fujimura. Uncertainty-aware data aggregation for deep imitation learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 761–767. IEEE, 2019.
- [15] Haoyuan Cai, Zhenghao Peng, and Bolei Zhou. Robot-gated interactive imitation learning with adaptive intervention mechanism. *arXiv preprint arXiv:2506.09176*, 2025.
- [16] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. A review of safe reinforcement learning: Methods, theories and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [17] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.
- [18] Amin Abyaneh, Mahrokh G Boroujeni, Hsiu-Chin Lin, and Giancarlo Ferrari-Trecate. Contractive dynamical imitation policies for efficient out-of-sample recovery. *arXiv preprint arXiv:2412.07544*, 2024.
- [19] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021.
- [20] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.