Imitation Learning with Temporal Logic Constraints

Zining Fan
Rutgers University
zf140@scarletmail.rutgers.edu

He Zhu
Rutgers University
hz375@cs.rutgers.edu

Abstract

Designing reinforcement learning agents to satisfy complex temporal objectives expressed in Linear Temporal Logic (LTL), presents significant challenges, particularly in ensuring sample efficiency and task alignment over infinite horizons. Recent works have shown that by leveraging the corresponding Limit Deterministic Büchi Automaton (LDBA) representation, LTL formulas can be translated into variable discounting schemes over LDBA-accepting states to maximize a lower bound on the probability of formula satisfaction. However, the resulting reward signals are inherently sparse, making exploration of LDBA-accepting states increasingly difficult as task horizons lengthen to infinity. In this work, we address these challenges by leveraging finite-length demonstrations to overcome the exploration bottleneck for LTL objectives over infinite horizons. We segment demonstrations and agent exploratory trajectories at LDBA-accepting states and iteratively guide the agent within each segment to learn to reach these accepting states. By incentivizing the agent to visit LDBA-accepting states from arbitrary states, our approach increases the probability of LTL formula satisfaction without the need for extensive or lengthy demonstrations. We demonstrate the applicability of our method across a variety of high-dimensional continuous control domains. It achieves faster convergence and consistently outperforms baseline approaches.

1 Introduction

Linear Temporal Logic (LTL) has been extensively studied as an alternative framework for specifying objectives for reinforcement learning (RL) agents [54, 27, 13, 62, 3, 16, 63]. LTL provides a powerful and flexible language to define tasks with temporal dependencies, such as“cycle between two subgoals while always avoiding unsafe regions" or “eventually reach a goal after completing a sequence of subtasks" [2]. Designing RL agents to satisfy these objectives is particularly challenging when considering infinite horizons, where the agent must maintain behavior that satisfies the objectives indefinitely. These challenges are compounded by the need to ensure sample efficiency in high-dimensional, continuous systems.

Several works [28, 63, 13] have proposed proxy reward schemes to derive policies from the Limit Deterministic Büchi Automaton (LDBA) representation of LTL specifications. A trajectory satisfies an LTL formula if and only if it visits an LDBA-accepting state infinitely often. However, these proxy rewards, defined over LDBA-accepting states, are inherently sparse, posing challenges for effective exploration toward such states.

To address these limitations, we propose a novel framework, TiLoIL (Temporal Logic Imitation Learning), which leverages Learning from Demonstrations to mitigate reward sparsity in policy optimization for LTL objectives. The high-level idea is to combine the structure of LDBAs with finite-length demonstrations to guide exploration. Specifically, TiLoIL segments expert demonstrations and agent exploratory trajectories at each visit to LDBA-accepting states. Each segment represents a sub-trajectory directed toward some accepting state. Using each segment of agent trajectories, TiLoIL incentivizes the agent to learn how to effectively reach an LDBA-accepting state by imitating

Preprint. Under review.

the expert behavior in the expert trajectory segments. In essence, TiLoIL encourages the agent to consistently seek LDBA-accepting states from any non-accepting states, thereby increasing the likelihood of satisfying infinite-horizon LTL formulas. Along the way, TiLoIL learns a reward function for reaching LDBA-accepting states by contrasting successful trajectories with unsuccessful ones. This reward function is then leveraged for policy training, hence reducing the need for extensive or lengthy demonstration data.

Moreover, TiLoIL uses the inherent multistage structure of LTL formulas to further densify the reward function to reach the LDBA-accepting states. At each stage, the reward function provides the agent with rich reward signals specifically tailored to that stage. This staged approach improves the efficiency of the learning process compared to treating LDBA-accepting state reaching as a single monolithic procedure.

We demonstrate the effectiveness of TiLoIL across a range of tasks in high-dimensional continuous systems, which have historically posed significant challenges for LTL-based RL methods. Our method achieves faster convergence and outperforms baseline approaches, demonstrating superior generalization to unseen scenarios. By leveraging the synergy between LTL specifications, LDBA representations, and demonstration data, TiLoIL provides a practical and scalable solution for designing RL agents that satisfy complex temporal objectives.

2 Background and problem setup

This section sets up our reinforcement learning problem to solve tasks specified by Linear Temporal Logic (LTL).

2.1 Linear temporal logic (LTL)

LTL [50] is a specification language that combines atomic propositions (APs) and logical operators to describe system behaviors and temporal properties. An atomic proposition AP represents a basic indivisible statement about the state of a system that can be true or false at a given time. The logical operators include: not ($\neg$), and ($\wedge$), and implies ($\rightarrow$); and the temporal operators include: next (X), repeatedly/always/globally (G), eventually (F), and until (U). Appendix C defines the syntax and semantics of LTL formulas. For a complete introduction, we refer the reader to Baier and Katoen [7].

Example. Consider the *FlatWorld* environment in Fig. 1 with $AP = \{y, g, r, b\}$ where r labels the red region, g labels the green, y labels the yellow, and b labels the blue. If the task is to eventually reach the yellow zone and remain there, we express this as $\varphi = \mathsf{FG}y$. Alternatively, if we require the agent to oscillate infinitely between the yellow, green, and red zones while avoiding the blue zone, we express this as $\varphi = \mathsf{GF}(y \wedge \mathsf{XF}(g \wedge \mathsf{XF}r)) \wedge \mathsf{G}\neg b$, which combines the properties of safety, reachability, and progress.

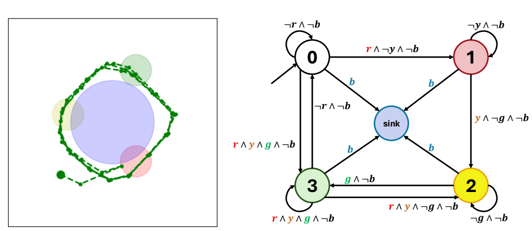

Figure 1: Left: FlatWorld Cycle environment with LTL spec $\varphi = \mathsf{GF}(y \wedge \mathsf{XF}(g \wedge \mathsf{XF}r)) \wedge \mathsf{G}\neg b$. Right: LDBA for φ accepts paths reaching state 3 infinitely. Blue region b leads to a sink.

From LTL to LDBA. The satisfaction of LTL formulas can be formally defined using Limit Deterministic Büchi Automata (LDBAs). An LDBA can be derived from any LTL formula φ and keeps track of the progression of φ satisfaction [57].

Definition 2.1 (Limit Deterministic Büchi Automaton (LDBA))**.** An LDBA is a tuple $\mathcal{L} = (\mathcal{B}, \Sigma \cup \mathcal{E}, P^{\mathcal{B}}, \mathcal{B}^{\star}, b_0)$, where $\mathcal{B}$ is a finite set of states, $\Sigma = 2^{AP}$ is an alphabet over atomic propositions, $P^{\mathcal{B}} : \mathcal{B} \times \Sigma \cup \mathcal{E} \to \mathcal{B}$ is a transition function, $\mathcal{B}^{\star} \subseteq \mathcal{B}$ is a set of accepting states, and $b_0 \in \mathcal{B}$ is the initial state. There exists a mutually exclusive partitioning of $\mathcal{B} = \mathcal{B}_D \cup \mathcal{B}_N$ such that $\mathcal{B}^{\star} \subseteq \mathcal{B}_D$ and for $(b, a) \in (\mathcal{B}_D \times \Sigma)$, then $P^{\mathcal{B}}(b, a) \subseteq \mathcal{B}_D$. $\mathcal{E}$ is a set of “jump" actions, also known as epsilon-transitions, for $b \in \mathcal{B}_N$ that transitions to $\mathcal{B}_D$ without evaluating any atomic propositions.

Example. In the *FlatWorld* environment (Fig. 1, left) and its corresponding LTL specification $\varphi = \mathsf{GF}(y \wedge \mathsf{XF}(g \wedge \mathsf{XF}r)) \wedge \mathsf{G}\neg b$, the LDBA is shown in Fig. 1 (right).

An infinite sequence of LDBA actions $(a_i)_{i=0}^{\infty} \in \Sigma^{\infty}$ induces a path $p = (b_i)_{i=0}^{\infty}$ according to $b_{i+1} = P^{\mathcal{B}}(b_i, a_i)$.

Definition 2.2 (Limit Deterministic Büchi Automaton (LDBA))**.** An LDBA $\mathcal{L}$ accepts a path $(b_i)_0^{\infty}$ if and only if the path visits an accepting state $b^\star \in \mathcal{B}^\star$ infinitely often, that is: $\forall t \in \mathbb{N}, \exists t' > t$ such that $b_{t'} \in \mathcal{B}^\star$.

Given an LTL formula φ, we translate it into an LDBA $\mathcal{L}$. Satisfaction of φ by an infinite sequence of AP evaluations (ω-word) corresponds directly to the acceptance of the path $(b_i)_{i=0}^{\infty}$ induced by the ω-word in $\mathcal{L}$ (Def. 2.2).

2.2 Product MDP with LDBA

A Markov Decision Process (MDP) is defined as a tuple $\mathcal{M} = (S, A, P, \mu_0)$, where S and A represent the state and action spaces, which may be continuous or discrete. The transition kernel $P : S \times A \to \Delta(S)$ describes the system's dynamics, representing probability distributions over S. $\mu_0 \in \Delta(S)$ specifies the initial state distribution.

To integrate atomic propositions (APs) into MDP states, as in previous work [63, 62, 6], we assume the existence of a labeling function $\mathcal{F} : S \to \Sigma$ which returns the atomic propositions that are true in that state. The labeling function can be thought of as a collection of event detectors that activate whenever the propositions in AP are satisfied within the environment.

We define control policies over the product space of an MDP $\mathcal{M}$ and an LDBA $\mathcal{L}$, aiming to generate accepting trajectories that satisfy an LTL formula φ from which the LDBA $\mathcal{L}$ is derived.

Definition 2.3. [Product MDP] Given MDP $\mathcal{M} = (S, A, P, \mu_0)$ and LDBA $\mathcal{L} = (\mathcal{B}, \Sigma \cup \mathcal{E}, P^{\mathcal{B}}, \mathcal{B}^\star, b_0)$, a product MDP $\mathcal{M}^\times = (S^\times, A^\times, P^\times, \mu_0^\times)$ synchronizes $\mathcal{M}$ with $\mathcal{L}$, where $S^\times = S \times \mathcal{B}$, $A^\times = A \cup \mathcal{E}$, and $\mu_0^\times(s, b) = \mu_0(s) \cdot 1_{b=b_0}$. The transition kernel $P^\times$ is defined using the labeling function $\mathcal{F}$ as follows:

$$P^\times((s', b') \mid (s, b), a) = \begin{cases} P(s' \mid s, a), & \text{if } a \in A,\ b' = P^{\mathcal{B}}(b, \mathcal{F}(s')) \\ 1, & \text{if } a \in \mathcal{E},\ b' = P^{\mathcal{B}}(b, a),\ s = s' \\ 0, & \text{otherwise} \end{cases}$$

We say that (s, b) is an LDBA-accepting product state if $b \in \mathcal{B}^*$. Def. 2.3 allows the connection of trajectories (and, consequently, policies) to the satisfaction of a given LTL formula. Consider an LTL formula φ and its corresponding LDBA $\mathcal{L}$, along with a trajectory $\tau = (s_i, b_i)_{i=0}^{\infty}$ in the product MDP $\mathcal{M}^\times$. The trajectory $\tau \models \varphi$ (i.e., τ satisfies φ) if and only if $\mathcal{L}$ accepts the path $(b_i)_{i=0}^{\infty}$, the projection of τ onto the LDBA states.

Now, consider a policy $\pi : S^\times \to \Delta(A^\times)$. The probability that π satisfies φ can be expressed as the expected value of the indicator for trajectories satisfying φ: $P(\pi \models \varphi) = \mathbb{E}_{\tau \sim \pi}\left[1_{\tau \models \varphi}\right]$. Optimizing a policy π for the satisfaction of an LTL specification φ can therefore be formulated as finding $\pi^\star \in \arg\max_{\pi \in \Pi} P(\pi \models \varphi)$ where Π denotes the space of all admissible policies.

2.3 Imitation learning for LTL-Constrained tasks

To maximize the probability of satisfying the LTL formula $P(\pi_\phi \models \varphi)$, an RL algorithm incentivize the agent to visit LDBA-accepting states as frequently as possible [27, 63]. However, this approach presents a significant challenge for exploration due to the inherent sparsity of the feedback: the agent receives rewards only upon making substantial progress toward task completion, such as reaching an accepting state in the LDBA. Visits to multiple non-accepting states within the LDBA during exploration may not provide meaningful learning signals, as there is no guidance from unexplored regions of the LDBA. **Our main idea** is to leverage the global structure of the LDBA and integrate it with expert task demonstrations to generate a dense reward signal along the LDBA paths for efficient agent exploration.

Expert Demonstrations. TiLoIL assumes expert demonstrations $\mathcal{D}_{\text{expert}}$ in the form of sequences of MDP states in S observed during the execution of a task by an expert policy, providing information about the regions of the state space relevant for task completion. First, we do not assume that demonstrations are collected in the product MDP. This ensures that the discriminator cannot exploit LDBA state information as a shortcut to bias its decisions, which could result in uninformed learning signals. Second, our demonstrations consist solely of state trajectories, reducing the burden of generating detailed action sequences. This choice also aligns with our goal of learning policies in the action space $A^\times$ of the product MDP. Since expert demonstrations operate within the raw MDP action space A, the demonstrated actions do not include the "jump" actions in $A^\times$.

Imitation Learning. The introduction of the Generative Adversarial Imitation Learning (GAIL) algorithm [29] has driven significant advances in scalable deep imitation learning methods [23, 24, 39, 34, 21, 8, 48]. Beyond adversarial approaches, several imitation learning algorithms aim to match the state action distributions of the expert and the agent through non-adversarial techniques, such as non-parametric models [38], random network distillation [64], support estimation [10], Wasserstein distance minimization [15], and moment matching [59].

In this paper, we adopt the GAIL framework. Recent studies [5] have shown that GAIL and its extensions consistently perform well at varying sample sizes. However, TiLoIL is not restricted to adversarial approaches and can be extended to other distribution matching imitation learning techniques. TiLoIL shapes reward signals using a discriminator $f_\psi(s)$, which is trained to distinguish between states from high-quality trajectories B^+—comprising expert demonstrations $\mathcal{D}_{\text{expert}}$—and states from low-quality trajectories B^-, consisting of the agent's own exploratory rollouts $\tau \sim \pi_\phi$ sampled from $M^\times$. States that resemble those in B^+ receive higher rewards from the discriminator f_ψ, while those similar to B^- receive lower rewards. TiLoIL employs an iterative training procedure, alternating between updating the discriminator f_ψ and the policy π_ϕ. The policy is optimized to generate trajectories that are increasingly indistinguishable from expert behavior, by maximizing the shaped reward $r_\psi((s,b),a) = \tanh(f_\psi(s))$. Given a discount factor γ, the policy objective is: $J_\pi(\phi) = \mathbb{E}_{\tau\sim\pi_\phi}\left[\sum_{t=0}^{\infty} \gamma^t\, r_\psi((s_t,b_t),a_t)\right]$.

Imitation Learning with LTL Constraints. TiLoIL forms a multi-objective optimization problem involving both probabilistic satisfaction of an LTL formula φ and the imitation learning objective J_π:

$$\pi^* = \arg\max_{\pi_\phi \in \Pi} \left(P(\pi_\phi \models \varphi), J_\pi(\phi)\right) \tag{1}$$

3 Imitation learning with LTL constraints

We emphasize that the learning objectives in our problem formulation, as presented in Eq. 1, are *not* in conflict. Since we assume access to expert demonstrations $\mathcal{D}_{\text{expert}}$, among the set of policies that mimic $\mathcal{D}_{\text{expert}}$ (i.e., maximize $J_\pi(\phi)$), there exists at least one policy π_ϕ that adheres to the specified LTL formula φ. Conversely, there exists one policy satisfying the specification φ (i.e., $P(\pi_\phi \models \varphi)$) that also closely aligns with the behavior demonstrated by the expert. We formalize this intuition below:

Theorem 3.1. Let π_1 and π_2 be two policies with corresponding occupancy measures ρ_{π_1} and ρ_{π_2}. For any Linear Temporal Logic (LTL) formula φ, the difference in the probabilities of satisfying φ under these policies is bounded by twice the total variation distance between their occupancy measures:

$$|P[\pi_1 \models \varphi] - P[\pi_2 \models \varphi]| \leq 2D_{\text{TV}}(\rho_{\pi_1}, \rho_{\pi_2}), \tag{2}$$

where the total variation (TV) distance between the distributions ρ_{π_1} and ρ_{π_2} is given by $D_{\text{TV}}(\rho_{\pi_1}, \rho_{\pi_2}) = \frac{1}{2}\int_\tau |\rho_{\pi_1}(\tau) - \rho_{\pi_2}(\tau)|\, d\tau$.

Let π_1 be the expert policy π_E from which demonstrations are collected, and π_2 be the imitation learning policy π_ϕ. This theorem shows that minimizing the total variation distance between π_E and π_ϕ leads to policies that maximize LTL satisfaction. The use of GAIL for imitation learning to minimize the JS (Jensen–Shannon) divergence reduces this TV distance, driving policies toward optimal satisfaction of LTL properties. The proof is in Appendix D.

Main Challenge. Theorem 3.1 does not directly apply in practice, as the infinite-length demonstrations required by LTL tasks are impractical to generate. For tasks with cyclic structures (e.g., Fig. 1), it is unrealistic to assume that expert demonstrations contain a large number of visits to LDBA-accepting states, due to the cost and complexity of constructing such demonstrations. Instead, we assume that demonstrations may include only one or two visits to an accepting state. This highlights a fundamental tension between the theoretical requirement of infinite visits to LDBA-accepting states and the inherently finite-horizon nature of expert demonstrations. First, the limited learning signal from such sparse demonstration data makes it challenging for the agent to generalize its behavior to satisfy LTL constraints over infinite horizons. Second, since environment states near LDBA-accepting states are typically unique to expert demonstrations and differ substantially from the agent's exploratory behavior, particularly in the early stages of training, the discriminator may develop a reward function that assigns higher rewards to these states. Consequently, optimizing $J_\pi(\phi)$ under finite-length demonstrations could lead the agent to become trapped near an LDBA-accepting state, where the rewards are higher. In this way, the agent lacks incentive to actually reach the accepting state and then leaves it to initiate another trail aimed at reaching the accepting state again. This would result in suboptimal behavior, as the policy fails to reach LDBA-accepting states frequently enough.

3.1 Segmented imitation

Our core idea to address the main challenges is segmenting expert demonstrations[1] and agent exploratory trajectories in $M^\times$ based on visits to LDBA-accepting states, where each segment represents a subtrajectory toward an accepting state. From the agent's perspective, each segmented rollout can start in any state within the state space and the goal is to reach an LDBA-accepting state; when the agent reaches an accepting state, the next rollout starts directly from its current state, with the goal remaining the same: reaching an LDBA-accepting state. This structure incentivizes the agent to continuously seek LDBA-accepting states, regardless of the starting point of each trail, thus encouraging their infinite visits and improving $P(\pi_\phi \models \varphi)$ the probability of LTL formula satisfaction. Even if the demonstration includes only one single segment, we can use it repeatedly as a guide to help the agent learn to efficiently reach LDBA-accepting states within each segment of the agent's rollout, thus optimizing $J_\pi(\phi)$. We present ***an example in Fig. 5*** to visualize this procedure. We formalize our method based on off-policy Q learning.

Q-Function Update. The Q-function $Q_\theta((s,b),a)$ for state and action pairs from $M^\times$ is updated by minimizing the Bellman residual using data sampled from a sampled replay buffer B. The loss function is defined as follows:

$$J_Q(\theta) = \mathbb{E}_{((s,b),a,(s',b'))\sim B}\left[\frac{1}{2}\left(Q_\theta((s,b),a) - \hat{y}\right)^2\right], \tag{3}$$

The target value $\hat{y}$ is defined as:

$$\hat{y} = \begin{cases} 1/(1-\gamma), & b \in \mathcal{B}^\star \\ R((s,b)) + \gamma \mathbb{E}_{a'\sim\pi_\phi(\cdot|(s',b'))}\left[Q_{\texttt{targ}}\right], & b \notin \mathcal{B}^\star. \end{cases} \tag{4}$$

where $R((s,b)) = \tanh(f_\psi(s))$ is the reward from the learned discriminator function f_ψ that separates B^+ as segmented expert demonstrations and B^- as segmented policy rollouts. Here, $Q_{\texttt{targ}} = Q_{\bar{\theta}}((s',b'),a') - \alpha \log \pi_\phi(a'|(s',b'))$ is the target Q value computed using a target network with parameters $\bar{\theta}$, α is the temperature parameter controlling the trade-off between reward and entropy, and $\log \pi_\phi(a'|(s',b'))$ is the entropy term used to encourage exploration.

Intuitively, we use the discriminator reward in $R((s,b))$ to train Q_θ to optimize $J_\pi(\phi)$ in our learning objectives in Eq. 1, encouraging the agent to mimic expert behavior to reach LDBA-accepting states. Upon reaching an LDBA-accepting state (s,b) such that $b \in \mathcal{B}^\star$, we directly set the target value for $Q((s,b),a)$ (for any action a) to $\frac{1}{1-\gamma}$. First, this Q value is sufficiently large to incentivize the agent to reach the LDBA-accepting state (s,b), rather than lingering nearby, thereby ensuring continuous progress toward satisfying the LTL constraint. Second, in this way, each segmented rollout does not interfere with others, effectively addressing the main challenge.

Policy (π_ϕ) Update. The policy $\pi_\phi(a|(s,b))$ is updated by minimizing the entropy-regularized expected Q-value, balancing exploration and exploitation. The loss is:

$$J_\pi(\phi) = \mathbb{E}_{(s,b)\sim B}\left[\mathbb{E}_{a\sim\pi_\phi(\cdot|(s,b))}\left[\alpha \log \pi_\phi(a|(s,b)) - Q_\theta((s,b),a)\right]\right]. \tag{5}$$

Here, the entropy term $\alpha \log \pi_\phi(a|(s,b))$ promotes exploration, while the Q-value term $-Q_\theta((s,b),a)$ encourages reward maximization.

The following theorem bounds the Q-function update in Eq. 3 relative to the optimal Q-value under the assumption of infinite-horizon expert demonstrations:

[1] To segment expert demonstrations, we utilize the state labeling function $\mathcal{F}$ to project the demonstrations and simulate the resulting projected trajectories on the LDBA.

Theorem 3.2. Let Q_θ be the learned soft Q-function trained using a modified target value $Q^{\text{target}} = \frac{1}{1-\gamma}$ in accepting states (s, b) where $b \in \mathcal{B}^\star$, and soft Bellman backups elsewhere. Assume $\pi_\phi(a \mid s, b) \propto \exp\left(\frac{1}{\alpha} Q_\theta((s, b), a)\right)$. Suppose that the reward function $R(s) = \tanh(f_\psi(s))$ is bounded: $R(s) \in [R_{\min}, R_{\max}]$. Let Q^* be the optimal Q-function under standard soft Bellman backups (without the modified target). Then, for any state-action pair $((s, b), a)$,

$$Q_\theta((s, b), a) - Q^*((s, b), a) \le \gamma^{k(s,b)} \cdot \delta_{\max}$$

where $k(s, b)$ is the number of steps it takes from (s, b) under π_ϕ to reach some accepting state (s', b') with $b' \in \mathcal{B}^\star$, and $\delta_{\max} := \frac{1}{1-\gamma} - \frac{R_{\min} + \alpha \mathcal{H}_{\min}}{1-\gamma}$ is the worst-case overestimation error at any accepting states. $\mathcal{H}_{\min}$ denotes the minimum entropy of π_ϕ.

The theorem shows that, while Q-values are inflated to make LDBA-accepting states attractive, this overestimation decays exponentially with distance. For states that appear early in a *segmented* trajectory—those farther from acceptance—the corresponding Q-values are not significantly overestimated. The learning process therefore remains grounded. The proof is in Appendix D.

3.2 Multi-Stage Discriminator Learning

The learning strategy described in Sec. 3.1 still faces challenges when long horizons are required to reach each LDBA-accepting state. We observe that many LTL tasks inherently consist of multiple stages. Take, for example, the FlatWorld Cycle task with the LTL specification $\varphi = \mathsf{GF}(y \wedge \mathsf{XF}(g \wedge \mathsf{XF}r)) \wedge \mathsf{G}\neg b$ and its LDBA, illustrated in Fig. 1. Any valid trajectory between the LDBA accepting states can be divided into three distinct stages. Initially, the LDBA state is 0 and the agent is in the white space for the first stage of reaching the red zone. Upon reaching the red zone (r), the LDBA transitions from 0 to 1 and the agent is the second stage of reaching the yellow zone. If the agent touches the blue region in the middle at any point, the LDBA transitions into a sink state and remains there for the rest of the episode. *How can we design a staged approach for LDBAs that makes the learning process more efficient compared to treating them as a single monolithic stage?*

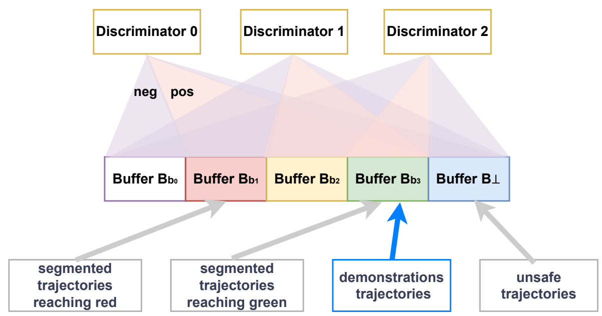

Figure 2: We illustrate the multistage discriminator for the FlatWorld Cycle environment in Fig.1. TiLoIL maintains separate stage buffers to store trajectories corresponding to different LDBA states. Here, the colors of the buffers match the corresponding automaton states in Fig. 1. Each segmented trajectory is assigned to only one stage buffer based on its maximal stage (Eq. 7). B_{b_0} holds segmented trajectories that do not enter any colored zones, while B_{b_1} stores segmented trajectories that visit the red region but do not reach yellow. For Discriminator 0, negative data are drawn from buffer B_{b_0} and buffer $B_\perp$ which stores unsafe trajectories, while positive data are sampled from buffers B_{b_1}, B_{b_2}, and B_{b_3}. As such, the discriminator provides reward signals that encourage the agent to reach the red region and beyond.

We begin by presenting key definitions, followed by an illustration of our proposed approach. Define $b_i \rightsquigarrow b_j$ as an acyclic path in the (graph representation of) LDBA $\mathcal{L}$, where b_i and b_j are states in $\mathcal{L}$, such that the path begins at b_i, ends at b_j, and *does not include any accepting states* $b_f \in \mathcal{B}^*$ other than possibly b_j if b_j is an accepting state:

$$b_i \rightsquigarrow b_j \iff \exists \, \text{path } (b_i, b_{i+1}, \ldots, b_j) \text{ in } \mathcal{L} \text{ such that}$$
$$b_k \notin \mathcal{B}^* \, \forall k \neq j, \text{ and } \forall k, \ell, \, k \neq \ell \implies b_k \neq b_\ell. \quad (6)$$

Define $b_i \rightsquigarrow^* b_j \iff (b_i \rightsquigarrow b_j \vee b_i = b_j)$. We define a sink state of an LDBA $\mathcal{L}$ as a state $b_s \in \mathcal{B}$ such that $P^{\mathcal{B}}(b_s, \cdot) = b_s$. Once the agent transitions to a sink state, it cannot escape, thereby failing to reach any accepting states. Sink states are useful for modeling safety properties, such as globally avoiding the blue obstacle shown in Fig. 1. We use $\textsc{Sink}(\mathcal{L})$ to denote all sink states of $\mathcal{L}$.

Multistage Discriminators. In a dense reward setting for multistage tasks, the reward of an environment state associated with an LDBA state b_l should exceed that of b_k if $b_k \rightsquigarrow b_l$, as this encourages the agent to progress toward the LDBA's accepting states. If each state in an LDBA $\mathcal{L}$ is viewed as an individual stage, a separate discriminator can be trained for each stage to serve as a dense reward for that particular stage. By training stage-specific discriminators, we can effectively guide the agent's progress through the different stages of the task. To train the discriminators for different stages, we establish positive and negative data for each discriminator. We assign a maximal stage to each trajectory τ, which is determined as the LDBA state that advanced the furthest towards accepting states among all LDBA states within τ:

$$\text{MaxStage}(\tau : ((s_0, b_0), \ldots, (s_N, b_N))) = b_i \text{ such that } \exists \text{ path } \rho = (b_i \rightsquigarrow b_f) \ \wedge \ b_f \in \mathcal{B}^*. \\ \forall b_j \in \{b_0, \ldots, b_N\} \setminus b_i, \ b_j \text{ not in } \rho \quad (7)$$

For the discriminator associated with the LDBA state b_k, positive data include trajectories τ^+ with maximal stage $\textsc{MaxStage}(\tau^+)$ progressing *beyond* b_k and *up to* an accepting state, formally expressed as $\exists b_f \in \mathcal{B}^*. \ b_k \rightsquigarrow \textsc{MaxStage}(\tau^+) \rightsquigarrow^* b_f$. Conversely, negative data consist of trajectories τ^- that only reach up to b_k, such that $\textsc{MaxStage}(\tau^-) \rightsquigarrow^* b_k$, or hit any sink state $\textsc{MaxStage}(\tau^-) \in \textsc{Sink}(\mathcal{L})$. ***An example is in Fig. 2***.

Once the positive and negative data for each discriminator for an LDBA state b_k have been established, we train a discriminator f_{b_k} that predicts if an MDP state $s \in S$ comes from the positive trajectories whose maximal stage is beyond the LDBA state b_k, as opposed to the negative trajectories that fail to progress beyond b_k or trap into sink states. The discriminator is trained using the BCE loss where positive data B^+ includes states from τ^+ trajectories and negative data B^- consists of states from τ^- trajectories. We note that B^+ always includes (segmented) expert demonstrations that surpass all LDBA states on the path toward accepting states.

Multistage Reward Formulation. The next step is to combine these discriminators to create a reward function that guides the agent to LDBA-accepting states. Our formulation is inspired by [45]. We define our learned reward function for a product MDP state in a multi-stage task as follows:

$$R((s, b)) = \frac{\textsc{SIdx}(b) + \beta \cdot \tanh(f_b(s))}{\mathcal{N}(b)} \quad (8)$$

where $\textsc{SIdx}(b)$ computes the length of the longest acyclic path in the LDBA from the initial state to b, serving as an approximation of the stage index of the product MDP state (s, b), and β is a hyperparameter. The $\tanh$ function is used to bound the output of the discriminators. As the range of the $\tanh$ function is $(-1, 1)$, any $\beta < \frac{1}{2}$ ensures that the reward of a state in stage $k + 1$ is always higher than that of stage k. In practice, we use $\beta = \frac{1}{3}$. Dividing it by $\mathcal{N}(b)$, where $\mathcal{N}(b)$ denotes the length of the longest acyclic path in the LDBA from the initial state to an accepting state through b, scales the rewards to values less than 1.

Main Algorithm. The main learning algorithm of TiLoIL is provided in Appendix E.

4 Experiments

This section empirically evaluates TiLoIL by addressing the following questions: (Q1) Does TiLoIL improve exploration in LTL-constrained tasks? (Q2) Are the new learning-from-demonstrations strategies in TiLoIL necessary to learn policies that align with the LTL constraints?

Baselines. To answer Q1, we compare TiLoIL with three state-of-the-art policy optimization algorithms for LTL task objectives: (1) LCER [63], a counterfactual experience replay scheme, (2) Cycler [55], a method focusing on cycle environments, and (3) DRL^2 [6], a direct exploration algorithm that encodes LDBAs as a Markov reward process for reward shaping. This evaluation focuses on TiLoIL's exploration capabilities relative to these RL-based approaches. For Q2, we compare TiLoIL with GAIL [29], PWIL [14] and SQIL [53]. As discussed in Sec. 2.3, TiLoIL can be integrated into any generative adversarial and distribution matching imitation learning methods. However, rather than evaluating a broad range of algorithms, our focus is on assessing how our proposed strategies improve the imitation learning algorithm that it builds on (GAIL). In the experiment, we combined SAC [25] with both TiLoIL and GAIL, as well as PWIL and SQIL, to perform policy updates on batches of data stored in a replay buffer of the agent experience.

Environments and Tasks. Our benchmarks, visualized in Fig. 3, are borrowed from LCER [63] and DRL^2 [6]. On the top left, *GridCircular Hard* is a discrete 2D grid world where the agent moves

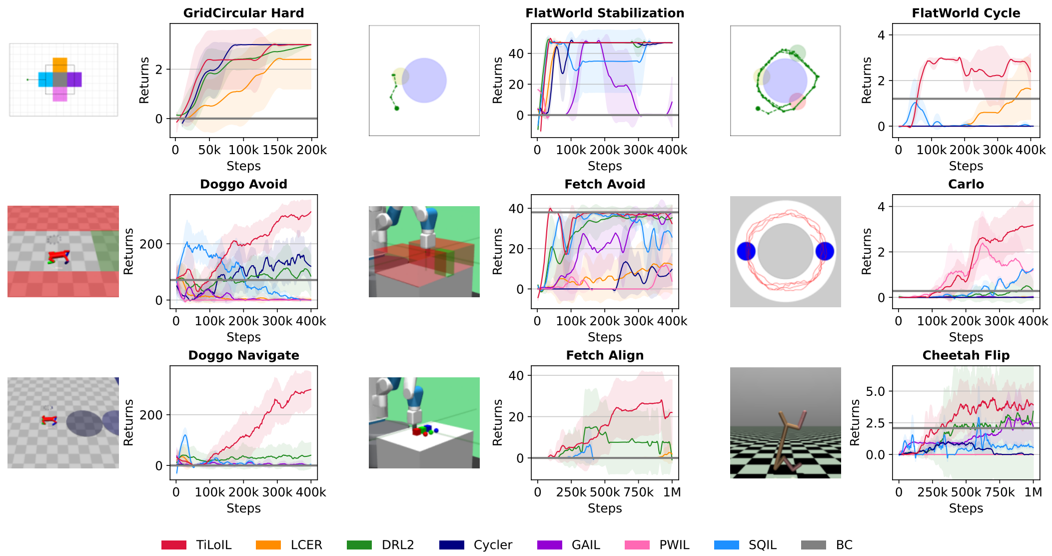

Figure 3: Returns under eventual discounting [63] comparing TiLoIL and the baselines over 10 random seeds.

in four cardinal directions or stays still[2]. The environment is a cross-shaped grid of five squares, where the center is an obstacle. The agent must repeatedly loop through the outer squares while avoiding the center. Next in the upper row, a point agent in *FlatWorld Stabilization* must stabilize in the yellow zone, while in *FlatWorld Cycle*, it oscillates between red, yellow, and green infinitely, always avoiding blue regions. On the bottom left, the Doggo agent, a 12-DoF quadruped robot [52], must (i) traverse a narrow corridor collision-free (*Doggo Avoid*) and (ii) sequentially navigate two designated zones (*Doggo Navigate*). In the middle bottom, a Fetch robotic arm [17] must (i) guide its gripper to a target position while minimizing lateral movements (*Fetch Avoid*) and (ii) achieve horizontal alignment of three cubes (*Fetch Align*). On the bottom right, in the *Carlo* environment, the agent drives a self-driving simulator based on a bicycle model counterclockwise on a circular track, repeatedly visiting two blue regions without crashing. In *Cheetah Flip*, the HalfCheetah agent performs frontflips to alternate between standing on its front and back legs infinitely. All task specifications in LTL and additional environment details are provided in Appendix H.

Demonstrations are generated by designing dense rewards and training individual policies for each stage of a task (e.g., training a policy to reach the yellow zone and another to reach the green zone in *FlatWorld Cycle*). Trajectories that successfully reach accepting states are collected by sequentially executing these policies. Both TiLoIL and the baselines—GAIL, PWIL, and SQIL—are provided with only **5 demonstrations**. In environments with cyclic structures—*GridCircular*, *FlatWorld Cycle*, *Carlo*, and *Cheetah Flip*—each trajectory contains at most **2 visits to LDBA accepting states**.

Results. In Fig. 3, the x-axis shows environment steps, and the y-axis shows mean cumulative returns under eventual discounting (App. H.3). The shaded region indicates the standard deviation. Computing the probability of LTL satisfactions over infinite horizons requires estimating policy occupancies, which is intractable. This return function, as a proxy for the likelihood of task satisfaction [63], counts visits to LDBA-accepting states, assigning a reward of 1 per visit with future discounts applied only at such states. This return function is solely used for *evaluation* in TiLoIL.

(Q1) Does TiLoIL help the learning process for LTL-constrained tasks? According to Fig. 3, TiLoIL significantly accelerates learning compared to the RL-based policy optimization approaches LCER, DRL^2 and Cycler. In our experience, Cycler faces challenges in scaling to high-dimensional continuous environments. Integrating the global structure of an LDBA and with expert task demonstrations, TiLoIL provides informative rewards in each LDBA transition, effectively guiding exploration toward accepting states, an advantage absent in LCER. Although DRL^2 also learns an intrinsic reward signal along LDBA paths to aid exploration, TiLoIL provides a stronger guidance through demonstrations, allowing faster transitions to LDBA-accepting states, particularly in complex environments, e.g.

[2]We compared TiLoIL with the baselines in a suit of discrete environments from [6] in Appendix F.

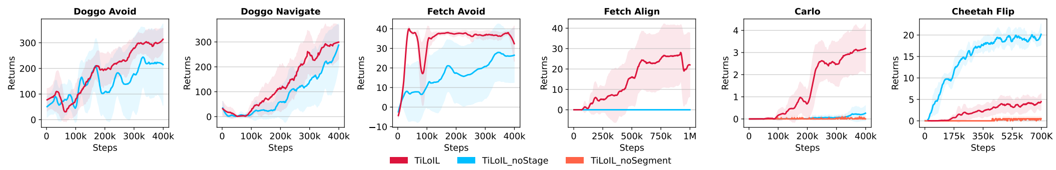

Figure 4: Ablation studies for TiLoIL over 10 random seeds using returns under eventual discounting. Results for TiLoIL-noSegment are omitted in *Doggo* and *Fetch* tasks. For these non-circular tasks, their LDBAs absorb at accepting states when safety is ensured. Thus, trajectory segmentation has negligible impact, leading to similar reward curves for TiLoIL-noSegment and TiLoIL in these tasks.

Fetch Align and *Carlo*. In *Carlo*, TiLoIL completes 3–4 rounds within the track in 500 steps, while the baselines struggle to complete even one.

(Q2) Are segmented imitation and multistage discriminator learning in TiLoIL necessary? Our results show that TiLoIL generates a significant lift over its baseline algorithms GAIL, PWIL and SQIL in the learning curves in Fig. 3 for all tasks with cyclic structures, such as *FlatWorld Cycle*, *Carlo*, and *Cheetah Flip*, highlighting the importance of segmented imitation for generalizing policies to repeated accepting-state visits over infinite horizons. In other multistage tasks, such as *Doggo Navigate* and *Fetch Align*, the improvements over these imitation learning baselines are attributed to the generation of stage-specific rewards that guide intermediate goal achievement, without which the learning signal from a monolithic discriminator may lead to a flat value landscape.

Ablation Studies. We assess the individual contributions of (a) segmented imitation and (b) multistage discriminator learning to the overall performance. The first ablation, TiLoIL-noSegment, only performs part (b) for multistage discriminator learning. The second ablation TiLoIL-noStage only performs part (a) for segmented imitation.Fig. 4 confirms the importance of segmented imitation, as circular tasks (*Carlo* and *Cheetah Flip*) cannot be solved without it. Multistage discriminator learning is also crucial to ensure sample-efficient learning in all tasks except *Cheetah Flip*. TiLoIL-noStage performs extremely well in this challenging task. We found that the multistage discriminator in TiLoIL incorporates successful trajectories that barely achieve standing on the back legs, sometimes leading the agent to just satisfy this subtask without generating sufficient momentum to round over.

Demonstration Size. We experimented with TiLoIL using *varying numbers of demonstrations* and found that it does not rely on a large number of demonstrations to bootstrap learning. See Appendix G.

5 Related work

Reinforcement learning from linear temporal logic (LTL) has advanced significantly, with many approaches leveraging structural insights to guide learning [54, 18, 30, 31, 12, 13]. Early methods aligned value and policy optimization using product MDPs and reward signals to encourage task satisfaction [9, 12, 13, 27, 28, 37, 43], leading to principled approaches that optimize lower bounds on formula satisfaction [56, 63]. To address reward sparsity, several methods emerged: rule-based approaches [43], accepting frontier function [27, 28], rewards for initial visits [11] or adapted annotated maps [67]. Automata structure has also been used to learn hierarchical [28, 31, 35], goal-conditioned [51], or modular [11] policies. Recent works further improve LTL-guided exploration via meta-learning [41, 62], temporal reward shaping [6], eventual discounting [63] and cyclic temporal constraints for recurring goals [55]. We discuss related work in broader contexts in Appendix B.

Conclusion. We present TiLoIL, a learning-from-demonstrations framework for policy optimization under LTL constraints. TiLoIL employs segmented imitation learning to guide the agent toward LDBA-accepting states following expert demonstrations from any state within the state space, thus optimizing the satisfaction of the LTL formula over infinite horizons. Furthermore, TiLoIL leverages the global structure of the LDBA, integrating it with expert task demonstrations to construct a dense reward signal along the LDBA paths, facilitating efficient exploration toward accepting states. Our results demonstrate that TiLoIL effectively solves diverse high-dimensional tasks with limited demonstrations, outperforming various baselines.

Limitations. TiLoIL is not designed to handle suboptimal demonstrations. Future work will focus on extending it to effectively leverage prior datasets containing suboptimal trajectories to guide exploration toward LTL satisfaction while still allowing flexibility to discover optimal behaviors.

References

[1] 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.

[2] David Abel, Will Dabney, Anna Harutyunyan, Mark K Ho, Michael Littman, Doina Precup, and Satinder Singh. On the expressivity of markov reward. *Advances in Neural Information Processing Systems*, 34:7799–7812, 2021.

[3] Rajeev Alur, Suguman Bansal, Osbert Bastani, and Kishor Jothimurugan. A framework for transforming specifications in reinforcement learning. In *Principles of Systems Design: Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, pages 604–624. Springer, 2022.

[4] Brandon Araki, Xiao Li, Kiran Vodrahalli, Jonathan DeCastro, Micah Fry, and Daniela Rus. The logical options framework. In *International Conference on Machine Learning*, pages 307–317. PMLR, 2021.

[5] Kai Arulkumaran and Dan Ogawa Lillrank. A pragmatic look at deep imitation learning. In Berrin Yanikoglu and Wray L. Buntine, editors, *Asian Conference on Machine Learning, ACML 2023, 11-14 November 2023, Istanbul, Turkey*, volume 222 of *Proceedings of Machine Learning Research*, pages 58–73. PMLR, 2023. URL `https://proceedings.mlr.press/v222/arulkumaran24a.html`.

[6] Marco Bagatella, Andreas Krause, and Georg Martius. Directed exploration in reinforcement learning from linear temporal logic. *arXiv preprint arXiv:2408.09495*, 2024.

[7] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[8] Lionel Blondé and Alexandros Kalousis. Sample-efficient imitation learning via generative adversarial nets. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 3138–3148. PMLR, 2019. URL `http://proceedings.mlr.press/v89/blonde19a.html`.

[9] Alper Kamil Bozkurt, Yu Wang, Michael M Zavlanos, and Miroslav Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10349–10355. IEEE, 2020.

[10] Kianté Brantley, Wen Sun, and Mikael Henaff. Disagreement-regularized imitation learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=rkgbYyHtwB`.

[11] Mingyu Cai, Mohammadhosein Hasanbeig, Shaoping Xiao, Alessandro Abate, and Zhen Kan. Modular deep reinforcement learning for continuous motion planning with temporal logic. *IEEE Robotics and Automation Letters*, 6(4):7973–7980, 2021.

[12] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila McIlraith. Non-markovian rewards expressed in ltl: guiding search via reward shaping. In *Proceedings of the International Symposium on Combinatorial Search*, volume 8, pages 159–160, 2017.

[13] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. Ltl and beyond: Formal languages for reward function specification in reinforcement learning. In *IJCAI*, volume 19, pages 6065–6073, 2019.

[14] Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. *arXiv preprint arXiv:2006.04678*, 2020.

[15] Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=TtYSU29zgR`.

[16] Giuseppe De Giacomo, Antonio Di Stasio, Moshe Vardi, Shufang Zhu, et al. Two-stage technique for ltlf synthesis under ltl assumptions. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, pages 304–314, 2020.

[17] R. de Lazcano, K. Andreas, J. J. Tai, S. R. Lee, and J. Terry. Gymnasium robotics. GitHub, 2023. URL `http://github.com/Farama-Foundation/Gymnasium-Robotics`.

[18] Xuchu Ding, Stephen L Smith, Calin Belta, and Daniela Rus. Optimal control of markov decision processes with linear temporal logic constraints. *IEEE Transactions on Automatic Control*, 59(5):1244–1257, 2014.

[19] Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023.

[20] Zeyu Feng, Hao Luan, Kevin Yuchen Ma, and Harold Soh. Diffusion meets options: Hierarchical generative skill composition for temporally-extended tasks. *arXiv preprint arXiv:2410.02389*, 2024.

[21] Chelsea Finn, Paul F. Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *CoRR*, abs/1611.03852, 2016. URL `http://arxiv.org/abs/1611.03852`.

[22] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

[23] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[24] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Conference on robot learning*, pages 1259–1277. PMLR, 2020.

[25] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[26] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[27] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099*, 2018.

[28] Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Deep reinforcement learning with temporal logics. In *Formal Modeling and Analysis of Timed Systems: 18th International Conference, FORMATS 2020, Vienna, Austria, September 1–3, 2020, Proceedings 18*, pages 1–22. Springer, 2020.

[29] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[30] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116. PMLR, 2018.

[31] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

[32] Craig Innes and Subramanian Ramamoorthy. Elaborating on learned demonstrations with temporal logic specifications. *arXiv preprint arXiv:2002.00784*, 2020.

[33] Mathias Jackermeier and Alessandro Abate. Deepltl: Learning to efficiently satisfy complex ltl specifications for multi-task rl. In *The Thirteenth International Conference on Learning Representations*.

[34] Rohit Jena, Siddharth Agrawal, and Katia P. Sycara. Addressing reward bias in adversarial imitation learning with neutral reward functions. *CoRR*, abs/2009.09467, 2020. URL `https://arxiv.org/abs/2009.09467`.

[35] Kishor Jothimurugan, Suguman Bansal, Osbert Bastani, and Rajeev Alur. Compositional reinforcement learning from logical specifications. *Advances in Neural Information Processing Systems*, 34:10026–10039, 2021.

[36] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[37] Yiannis Kantaros. Accelerated reinforcement learning for temporal logic control objectives. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5077–5082. IEEE, 2022.

[38] Kee-Eung Kim and Hyun Soo Park. Imitation learning via kernel mean embedding. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3415–3422. AAAI Press, 2018. doi: 10.1609/AAAI.V32I1.11720. URL `https://doi.org/10.1609/aaai.v32i1.11720`.

[39] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

[40] Jan Křetínskỳ, Tobias Meggendorfer, Salomon Sickert, and Christopher Ziegler. Rabinizer 4: From ltl to your favourite deterministic automaton. In *International Conference on Computer Aided Verification*, pages 567–577. Springer, 2018.

[41] Borja G León, Murray Shanahan, and Francesco Belardinelli. Agent, do you see it now? systematic generalisation in deep reinforcement learning. In *ICLR Workshop on Agent Learning in Open-Endedness*.

[42] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

[43] Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3834–3839. IEEE, 2017.

[44] Jason Xinyu Liu, Ankit Shah, Eric Rosen, Mingxi Jia, George Konidaris, and Stefanie Tellex. Ltl-transfer: Skill transfer for temporal task specification. *arXiv preprint arXiv:2206.05096*, 2022.

[45] Tongzhou Mu, Minghua Liu, and Hao Su. Drs: Learning reusable dense rewards for multi-stage tasks. *arXiv preprint arXiv:2404.16779*, 2024.

[46] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[47] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[48] Manu Orsini, Anton Raichuk, Léonard Hussenot, Damien Vincent, Robert Dadashi, Sertan Girgin, Matthieu Geist, Olivier Bachem, Olivier Pietquin, and Marcin Andrychowicz. What matters for adversarial imitation learning? In Marc'Aurelio Ranzato, Alina

Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 14656–14668, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/7b647a7d88f4d6319bf0d600d168dbeb-Abstract.html.

[49] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[50] Amir Pnueli. The temporal logic of programs. In *18th annual symposium on foundations of computer science (sfcs 1977)*, pages 46–57. ieee, 1977.

[51] Wenjie Qiu, Wensen Mao, and He Zhu. Instructing goal-conditioned reinforcement learning agents with temporal logic objectives. *Advances in Neural Information Processing Systems*, 36, 2024.

[52] A. Ray, J. Achiam, and D. Amodei. Benchmarking safe exploration in deep reinforcement learning. GitHub, 2019. URL https://github.com/openai/safety-gym.

[53] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019.

[54] Dorsa Sadigh, Eric S Kim, Samuel Coogan, S Shankar Sastry, and Sanjit A Seshia. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 1091–1096. IEEE, 2014.

[55] Ameesh Shah, Cameron Voloshin, Chenxi Yang, Abhinav Verma, Swarat Chaudhuri, and Sanjit A Seshia. Deep policy optimization with temporal logic constraints. *arXiv preprint arXiv:2404.11578*, 2024.

[56] Daqian Shao and Marta Kwiatkowska. Sample efficient model-free reinforcement learning from ltl specifications with optimality guarantees. *arXiv preprint arXiv:2305.01381*, 2023.

[57] Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Křetínskỳ. Limit-deterministic büchi automata for linear temporal logic. In *International Conference on Computer Aided Verification*, pages 312–332. Springer, 2016.

[58] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. *arXiv preprint arXiv:1912.04443*, 2019.

[59] Gokul Swamy, Sanjiban Choudhury, J. Andrew Bagnell, and Steven Wu. Of moments and matching: A game-theoretic framework for closing the imitation gap. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 10022–10032. PMLR, 2021. URL http://proceedings.mlr.press/v139/swamy21a.html.

[60] Geraud Nangue Tasse, Devon Jarvis, Steven James, and Benjamin Rosman. Skill machines: Temporal logic composition in reinforcement learning. 2022.

[61] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis. Gymnasium, 2023. URL https://zenodo.org/record/8127025. Accessed: 2025-01-28.

[62] Pashootan Vaezipoor, Andrew C Li, Rodrigo A Toro Icarte, and Sheila A Mcilraith. Ltl2action: Generalizing ltl instructions for multi-task rl. In *International Conference on Machine Learning*, pages 10497–10508. PMLR, 2021.

[63] Cameron Voloshin, Abhinav Verma, and Yisong Yue. Eventual discounting temporal logic counterfactual experience replay. In *International Conference on Machine Learning*, pages 35137–35150. PMLR, 2023.

[64] Ruohan Wang, Carlo Ciliberto, Pierluigi Vito Amadori, and Yiannis Demiris. Random expert distillation: Imitation learning via expert policy support estimation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6536–6544. PMLR, 2019. URL `http://proceedings.mlr.press/v97/wang19d.html`.

[65] Yanwei Wang, Nadia Figueroa, Shen Li, Ankit Shah, and Julie Shah. Temporal logic imitation: Learning plan-satisficing motion policies from demonstrations. *arXiv preprint arXiv:2206.04632*, 2022.

[66] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[67] Zikang Xiong, Joe Eappen, Daniel Lawson, Ahmed H Qureshi, and Suresh Jagannathan. Co-learning planning and control policies using differentiable formal task constraints. *CoRR*, 2023.

[68] Daniel Yang, Davin Tjia, Jacob Berg, Dima Damen, Pulkit Agrawal, and Abhishek Gupta. Rank2reward: Learning shaped reward functions from passive video. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2806–2813. IEEE, 2024.

[69] Konrad Zolna, Alexander Novikov, Ksenia Konyushkova, Caglar Gulcehre, Ziyu Wang, Yusuf Aytar, Misha Denil, Nando de Freitas, and Scott Reed. Offline learning from demonstrations and unlabeled experience. *arXiv preprint arXiv:2011.13885*, 2020.

A Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

B Related Work Discussion

Learning from demonstrations. Learning from demonstration is particularly valuable when designing a reward function is challenging. It allows agents to learn desired behaviors by observing and mimicking expert demonstrations. Some methods utilize classification-based rewards, where a reward function is trained by classifying goals [58, 36, 19] or by categorizing demonstration trajectories [69]. However, these rewards are trained exclusively on offline datasets, making them vulnerable to exploitation by a reinforcement learning agent.

Unlike multi-task RL approaches for LTL [44, 60, 4, 33] that define subtasks for zero-shot generalization to long-horizon or unseen tasks, TiLoIL focuses on mitigating the inherent exploration challenges in LTL-guided tasks, making it orthogonal to these approaches. Compared to approaches that require large-scale offline pretraining for LTL tasks [20], TiLoIL learns effectively from a small number of expert trajectories. Temporal Logic Imitation [65] integrates high-level LTL planning with pre-existing low-level controllers for continuous motion execution, whereas TiLoIL jointly learns both in an end-to-end manner. [32] transforms LTL specifications into a differentiable loss function, while TiLoIL addresses tasks where the reward signal from LTL remains sparse where such a differentiable loss approach is infeasible.

Inverse reinforcement learning. The above issue can be solved by Inverse Reinforcement Learning. Inverse Reinforcement Learning (IRL) is a crucial tool in learning from demonstrations [1, 47]. It aims at uncovering the underlying reward function from observed behaviors, which is particularly useful in scenarios where reward structures are not explicitly defined. Recently, Adversarial Imitation Learning (AIL) [29, 39, 24, 23]methods have been introduced, functioning in a manner akin to Generative Adversarial Networks (GANs). In these approaches, a generator (the policy) is trained to maximize the confusion of a discriminator, while the discriminator, acting as a surrogate for the reward function, is trained to differentiate between the agent's trajectories and the expert demonstrations. The introduction of GAIL [29] has driven significant advances in scalable deep imitation learning methods [23, 24, 39, 34, 21, 8, 48]. Beyond adversarial approaches, several imitation learning algorithms aim to match the state action distributions of the expert and the agent through non-adversarial techniques, such as non-parametric models [38], random network distillation [64], support estimation [10], Wasserstein distance minimization [15], and moment matching [59].

Rank reward learning The above issue could be solved by decomposing the tasks into easier sub-tasks. Hierarchical Reinforcement Learning (HRL) [22, 46, 42] methods decompose policies into sub-policies, each designed to address specific sub-tasks. Some approaches learn rewards with underlying substructures. DrS [45]decompose tasks by stage indicators, rank2reward [68] by learning videos; some methods [6, 63, 55]combined Buchi Automaton with the problem of labeling subtasks.

C Linear Temproal Logic

Syntax of LTL The syntax of Linear Temporal Logic (LTL) is defined over a set of atomic propositions AP. A state labeling function $\mathcal{F} : S \to \Sigma$ maps each state $s \in S$ to a subset of atomic propositions $\Sigma = 2^{\mathsf{AP}}$, where Σ is the alphabet formed by the powerset of AP. An LTL formula φ is constructed using the following grammar:

$$\varphi ::= \mathsf{true} \mid \mathsf{false} \mid p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid \mathsf{X}\varphi \mid \mathsf{F}\varphi \mid \mathsf{G}\varphi \mid (\varphi \mathsf{U} \psi)$$

where $p \in \mathsf{AP}$, and φ and ψ are LTL formulas. The logical connectives true, false, $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction) are standard. The temporal operators are: X for "next", F for "eventually", G for "globally", and U for "until".

Semantics of LTL The semantics of LTL is defined over infinite sequences of states $\tau = s_0, s_1, s_2, \ldots$, where each state $s_i \in S$ is associated with a set of atomic propositions $\mathcal{F}(s_i) \subseteq \mathsf{AP}$.

We write $\tau, i \models \varphi$ to indicate that φ holds at position i of the sequence τ. The semantics are given inductively as follows:

$$
\begin{aligned}
&\tau, i \models \mathsf{true} \quad \text{(always holds)}\\
&\tau, i \not\models \mathsf{false} \quad \text{(never holds)}\\
&\tau, i \models p \quad \Longleftrightarrow \quad p \in \mathcal{F}(s_i) \quad \text{for } p \in \mathsf{AP}\\
&\tau, i \models \neg\varphi \quad \Longleftrightarrow \quad \tau, i \not\models \varphi\\
&\tau, i \models (\varphi \wedge \psi) \quad \Longleftrightarrow \quad \tau, i \models \varphi \text{ and } \tau, i \models \psi\\
&\tau, i \models (\varphi \vee \psi) \quad \Longleftrightarrow \quad \tau, i \models \varphi \text{ or } \tau, i \models \psi\\
&\tau, i \models \mathsf{X}\varphi \quad \Longleftrightarrow \quad \tau, i+1 \models \varphi\\
&\tau, i \models \mathsf{F}\varphi \quad \Longleftrightarrow \quad \exists j \geq i, \tau, j \models \varphi\\
&\tau, i \models \mathsf{G}\varphi \quad \Longleftrightarrow \quad \forall j \geq i, \tau, j \models \varphi\\
&\tau, i \models (\varphi \mathsf{U} \psi) \quad \Longleftrightarrow \quad \exists j \geq i, (\tau, j \models \psi \text{ and } \forall k \in [i, j), \tau, k \models \varphi).
\end{aligned}
$$

In English, the semantics of Linear Temporal Logic (LTL) are defined over infinite sequences of states $\tau = s_0, s_1, s_2, \ldots$, where each state s_i represents a snapshot of the system. A state labeling function $\mathcal{F} : S \to \Sigma$ assigns a set of atomic propositions (AP) to each state, indicating which propositions are true. The satisfaction of an LTL formula is evaluated along these sequences. For example, p holds at a state s_i if $p \in \mathcal{F}(s_i)$, while $\mathsf{X}p$ requires p to hold in the next state s_{i+1}. Temporal operators like $\mathsf{F}p$ ("eventually p") and $\mathsf{G}p$ ("globally p") extend this reasoning over future states. Similarly, $\varphi \mathsf{U} \psi$ ("φ until ψ") requires φ to hold continuously until ψ becomes true at some future state.

For instance, the formula $\mathsf{F}y$ expresses that the system must eventually reach a state where y holds, which could represent a robot reaching a target area. The formula $\mathsf{G}\neg b$ specifies that b, such as a hazardous condition, must always be avoided. More complex behaviors can also be modeled, such as $\mathsf{G}(\mathsf{F}r)$, which ensures the system repeatedly visits states where r is true, or $\mathsf{G}(p\mathsf{U}q)$, where p must hold until q becomes true. These examples demonstrate how LTL can express diverse temporal properties for dynamic systems.

D Proofs of Theorem 3.1 and 3.2

Theorem 3.1 Let π_1 and π_2 be two policies with corresponding occupancy measures ρ_{π_1} and ρ_{π_2}. For any Linear Temporal Logic (LTL) formula φ, the difference in the probabilities of satisfying φ under these policies is bounded by twice the total variation distance between their occupancy measures:

$$|P[\pi_1 \models \varphi] - P[\pi_2 \models \varphi]| \leq 2D_{\mathrm{TV}}(\rho_{\pi_1}, \rho_{\pi_2}), \tag{9}$$

where the total variation (TV) distance between the distributions ρ_{π_1} and ρ_{π_2} is given by

$$D_{\mathrm{TV}}(\rho_{\pi_1}, \rho_{\pi_2}) = \frac{1}{2} \int_\tau |\rho_{\pi_1}(\tau) - \rho_{\pi_2}(\tau)| \, d\tau. \tag{10}$$

Proof. The probability of satisfying φ under policy π_1 can be expressed as an expectation:

$$P[\pi_1 \models \varphi] = \mathbb{E}_{\tau \sim \rho_{\pi_1}}[\mathbf{1}(\tau \models \varphi)] = \int_\tau \mathbf{1}(\tau \models \varphi) \rho_{\pi_1}(\tau) d\tau. \tag{11}$$

Similarly, for policy π_2:

$$P[\pi_2 \models \varphi] = \mathbb{E}_{\tau \sim \rho_{\pi_2}}[\mathbf{1}(\tau \models \varphi)] = \int_\tau \mathbf{1}(\tau \models \varphi) \rho_{\pi_2}(\tau) d\tau. \tag{12}$$

Thus, the absolute difference between these probabilities is:

$$|P[\pi_1 \models \varphi] - P[\pi_2 \models \varphi]| = \left| \int_\tau \mathbf{1}(\tau \models \varphi)(\rho_{\pi_1}(\tau) - \rho_{\pi_2}(\tau)) d\tau \right|. \tag{13}$$

Applying the triangle inequality:

$$|P[\pi_1 \models \varphi] - P[\pi_2 \models \varphi]| \leq \int_\tau |\rho_{\pi_1}(\tau) - \rho_{\pi_2}(\tau)| \, d\tau. \tag{14}$$

By the definition of total variation distance,

$$\int_{\tau} |\rho_{\pi_1}(\tau) - \rho_{\pi_2}(\tau)|\, d\tau = 2D_{\mathrm{TV}}(\rho_{\pi_1}, \rho_{\pi_2}). \tag{15}$$

Thus, we obtain the desired bound:

$$|P[\pi_1 \models \varphi] - P[\pi_2 \models \varphi]| \leq 2D_{\mathrm{TV}}(\rho_{\pi_1}, \rho_{\pi_2}). \tag{16}$$

This completes the proof. □

Theorem 3.2 Let Q_θ be the learned soft Q-function trained using a modified target value $Q^{\text{target}} = \frac{1}{1-\gamma}$ in accepting states (s, b) where $b \in \mathcal{B}^\star$, and soft Bellman backups elsewhere. Assume $\pi_\phi(a \mid s, b) \propto \exp\left(\frac{1}{\alpha} Q_\theta((s, b), a)\right)$. Suppose that the reward function $R(s) = \tanh(f_\psi(s))$ is bounded: $R(s) \in [R_{\min}, R_{\max}]$. Let Q^* be the optimal Q-function under standard soft Bellman backups (without the modified target). Then, for any state-action pair $((s, b), a)$,

$$Q_\theta((s, b), a) - Q^*((s, b), a) \leq \gamma^{k(s,b)} \cdot \delta_{\max}$$

where $k(s, b)$ is the number of steps it takes from (s, b) under π_ϕ to reach some accepting state (s', b') with $b' \in \mathcal{B}^\star$, and $\delta_{\max} := \frac{1}{1-\gamma} - \frac{R_{\min} + \alpha \mathcal{H}_{\min}}{1-\gamma}$ is the worst-case overestimation error at any accepting states. $\mathcal{H}_{\min}$ denotes the minimum entropy of π_ϕ.

Proof. We prove the result by backward induction along the trajectory, using the soft Bellman equation and the definition of $\delta_{\max}$.

Base case: At the final step $k = k(s, b)$, the overestimation is at most:

$$\delta_{\max} = \frac{1}{1-\gamma} - \left(\frac{R_{\min} + \alpha \mathcal{H}_{\min}}{1-\gamma}\right),$$

Inductive step: Suppose at step $i + 1$ we have:

$$Q_\theta((s_{i+1}, b_{i+1}), a_{i+1}) \leq Q^*((s_{i+1}, b_{i+1}), a_{i+1}) + \gamma^{k-(i+1)} \delta_{\max}.$$

Then for step i:

$$\begin{aligned}
Q_\theta((s_i, b_i), a_i) &= R(s_i) + \gamma\, \mathbb{E}_{s_{i+1}, b_{i+1}} \left[\mathbb{E}_{a' \sim \pi_\phi(\cdot|(s_{i+1}, b_{i+1}))} \left[Q_\theta((s_{i+1}, b_{i+1}), a')\right]\right] \\
&\leq R(s_i) + \gamma\, \mathbb{E}_{s_{i+1}, b_{i+1}} \left[\mathbb{E}_{a' \sim \pi_\phi(\cdot|(s_{i+1}, b_{i+1}))} [Q^*((s_{i+1}, b_{i+1}), a') + \gamma^{k-(i+1)} \delta_{\max}]\right] \\
&\leq R(s_i) + \gamma\, \mathbb{E}_{s_{i+1}, b_{i+1}} \left[\mathbb{E}_{a' \sim \pi_*(\cdot|(s_{i+1}, b_{i+1}))} [Q^*((s_{i+1}, b_{i+1}), a') + \gamma^{k-(i+1)} \delta_{\max}]\right] \\
&= Q^*((s_i, b_i), a_i) + \gamma^{k-i} \delta_{\max}.
\end{aligned}$$

where π^* is the optimal policy under Q^*.

Thus, the overesitmation error at (s, b) can propagate at most $\gamma^{k(s,b)} \cdot \delta_{\max}$:

$$Q_\theta((s, b), a) \leq Q^*((s, b), a) + \gamma^{k(s,b)} \cdot \delta_{\max}.$$

□

Even though our method clips the Q-value at accepting states to a large constant (e.g., $\frac{1}{1-\gamma}$), the theoretical bound shows:

$$Q_\theta((s, b), a) - Q^*((s, b), a) \leq \gamma^{k(s,b)} \cdot \delta_{\max}$$

Algorithm 1 TiLoIL Main Algorithm

Require: MDP $\mathcal{M}$, LDBA $\mathcal{L} = (\mathcal{B}, \Sigma \cup \mathcal{E}, P^{\mathcal{B}}, \mathcal{B}^{\star}, b_0)$, Demonstration dataset $\mathcal{D} := \{\tau^0, \tau^1, \dots\}$

$\mathcal{M}^{\times} \leftarrow M \times \mathcal{L}$
Initialize policy π_ϕ, critic Q_θ, replay buffer B
Initialize discriminators $f_{b_0}, f_{b_1}, \dots$ for $b_k \in \mathcal{B} \setminus \mathcal{B}^*$
Initialize stage buffers $B_{b_0}, B_{b_1}, \dots$ for $b_k \in \mathcal{B}$
Fill demo $\mathcal{D}$ in the stage buffers for accepting states $\mathcal{B}^*$
for each iteration **do**
    Sample trajectories $\mathcal{T}$ by executing π_ϕ in $\mathcal{M}^{\times}$
    Segment $\mathcal{T}$ at LDBA-accepting states (s, b) such that $b \in \mathcal{B}^*$ to obtain $\{\tau_\pi^{0\times}, \tau_\pi^{1\times}, \dots\}$
    for each trajectory $\tau_\pi^{i\times}$ in $\{\tau_\pi^{0\times}, \tau_\pi^{1\times}, \dots\}$ **do**
        $b_j \leftarrow \textsc{MaxStage}(\tau_\pi^{i\times})$
        if $b_j \in \textsc{Sink}(\mathcal{L})$ **then**
            $B_\perp \leftarrow B_\perp \cup \{\tau_\pi^i\}$
        else
            $B_{b_j} \leftarrow B_{b_j} \cup \{\tau_\pi^i\}$
        end if
    end for
    $B \leftarrow B \cup \{\tau_\pi^{0\times}, \tau_\pi^{1\times}, \dots\}$
    for each gradient step for discriminators **do**
        for each state b_k in $\mathcal{B} \setminus \mathcal{B}^* \setminus \textsc{Sink}(\mathcal{L})$ **do**
            Sample positive data from $\bigcup B_{b_i}$ for all b_i s.t. $\exists b_f \in \mathcal{B}^*.\ b_k \rightsquigarrow b_i \rightsquigarrow^* b_f$
            Sample negative data from $\bigcup B_{b_i} \cup B_{b_k} \cup B_\perp$ for all b_i s.t. $b_i \rightsquigarrow^* b_k$
            Update f_{b_k} using BCE loss
        end for
    end for
    for each gradient step for the policy π **do**
        Sample from B
        Update Q_θ via Eq. 3 and π_ϕ via Eq. 5 by SAC
    end for
end for

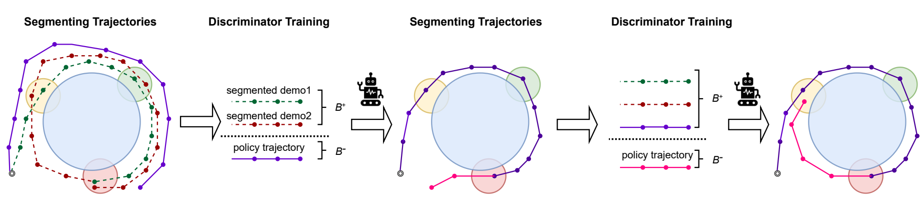

Figure 5: Segmented Imitation: the red and green dashed lines represent segmented demonstrations, while the purple and pink lines correspond to segmented trajectories generated by the policy. Through training, the policy learns to align with the segmented demonstrations.

E Main Algorithm

The main algorithm of TiLoIL is summarized in Algorithm 1. We use SAC [25] for policy training. In addition to the regular replay buffer B used in SAC, TiLoIL maintains different stage buffers B_{b_i} to store trajectories corresponding to different LDBA states b_i. Each trajectory is assigned to only one stage buffer based on its maximal stage (Eq. 7). During the training of the discriminators, we sample data from the union of multiple buffers. Policy updates are then guided by reward functions derived from these trained discriminators.

We visualize segmented imitation learning in Fig. 5.

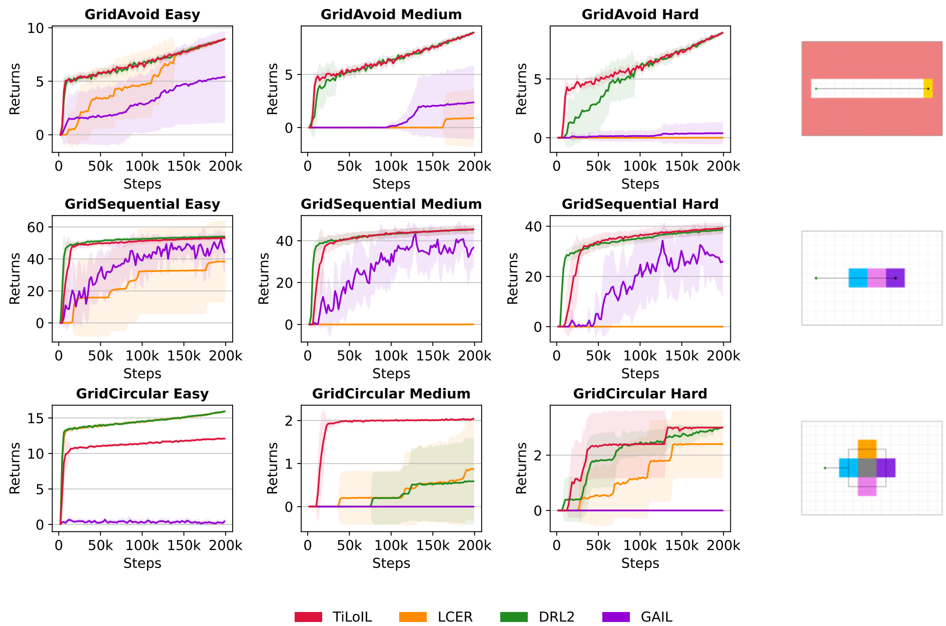

Figure 6: Discrete environment evaluation. The three rows represent reach-avoidance, sequential, and circular tasks, as shown on the right.

F Tabular Results

The evaluation environment is a deterministic 2D gridworld where the agent can move one unit in any of the four cardinal directions at each timestep or stay still.

Fig. 6 presents the performance of various algorithms across different GridWorld tasks under Tabular Q-Learning. The three rows correspond to distinct task categories: reach-avoidance (top row), sequential (middle row), and circular (bottom row), with increasing levels of difficulty (Easy, Medium, Hard) from left to right. The x-axis denotes the number of environment steps (up to 200k), while the y-axis represents the cumulative returns achieved.

The first row of Figure 6 shows the results in three *Grid Avoid* environments with LTL specification $\varphi = \mathsf{F}(a) \wedge \mathsf{G}\neg b$, with different grid sizes. The agent must navigate around a large area encompassing everything except a narrow corridor. The goal area is located at the end of this corridor, and the task becomes more challenging as the corridor's length increases. Here, the discriminators of our method will assign a bad reward whenever the agent exits the corridor, causing the LDBA to transition to a sink state. The discriminators direct the trajectory to align with the behavior observed in the demonstrations. Therefore, it has enhanced learning efficiency. Our approach shows a consistent and significant improvement in returns, especially in the hard setting.

The second and third row of Figure 6 shows the results in Sequential environments with LTL specification $\varphi = \mathsf{F}(a \wedge \mathsf{XF}(b \wedge \mathsf{XF}(c))$ and circular environments with **LTL** specification $\varphi = \mathsf{GF}(a \wedge \mathsf{XF}(b \wedge \mathsf{XF}(c))\mathsf{G}\neg b$. Here, a, b, and c are different zones. The agent needs to reach each zone in order. The hardness increases with the number of zones. For the sequential task, the number of zones increases progressively, with 3, 4, and 5 zones for the easy, medium, and hard levels, respectively. For the circular task, each mode has 2, 3, and 4 zones. In the first scenario, the agent must visit a specific sequence of zones in a predetermined order. Our method quickly achieves optimal returns, outperforming other methods. In the second scenario, the agent must repeat this sequence indefinitely while avoiding the center of the room. In this case, our method not only converges faster but also achieves significantly higher returns compared to other baselines, particularly in the medium and hard settings. For the *GridCircular Medium*, each trajectory has 72 steps; the *GridCircular Hard* 144 steps in each trajectory.

G Demonstration Sizes

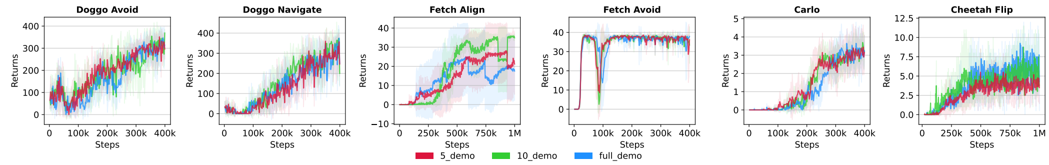

Figure 7: This figure illustrates our method with different numbers of demonstrations. The lines represent 5, 10, and full-size demonstrations. The full demonstration (number of episodes) varies across environments due to differences in episode length. Specifically, the full demonstration numbers are as follows: *Doggo Avoid* is 25, *Doggo Navigate* is 15, *Fetch Align* and *Fetch Avoid* are 100, *Carlo* is 111, and *Cheetah Flip* is 50.

Demonstrations can be difficult to obtain, especially in complex environments, making it important to work with a limited number of demonstrations. In Fig. 3, the results are based on 5 demonstrations. We aim to investigate whether more demonstration data would improve learning. As shown in Fig. 7, the demonstration size does not significantly limit our learning efficiency, though it has some impact on challenging tasks like Fetch Align and Cheetah Flip. This is mainly because TiLoIL trains multi-stage discriminators from stage-specific buffers. TiLoIL is able to leverage the agent's own behaviors—those that have progressed beyond certain LDBA states—as positive trajectories to train discriminators that guide learning for those trajectories that fail to meet such LDBA states, thereby reducing reliance on large demonstration datasets. We conclude that our method does not require large numbers of demonstrations.

We also compared our method with GAIL across varying numbers of demonstrations. As shown in Fig. 8, while GAIL improves with more demonstrations, TiLoIL consistently outperforms it by a significant margin.

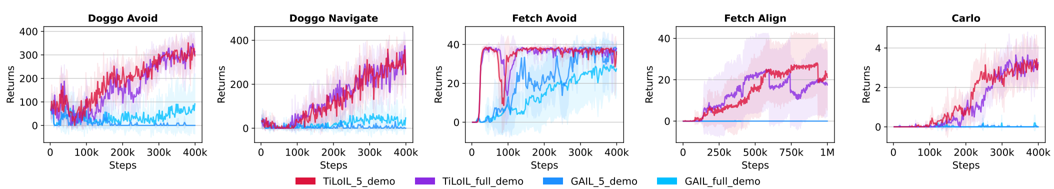

Figure 8: Returns under eventual discounting comparing TiLoIL with GAIL under different sizes of demonstrations. The full demonstration (number of episodes) varies across environments due to differences in episode length.

H Implementation Details

H.1 Environments and Tasks

The experiments presented in this paper are conducted within simulated environments.

H.1.1 Tabular Environments

The environments explored in the Fig. 6 are variations of the standard gridworld, represented as a 2D grid with the agent occupying a single cell. The action space is discrete, comprising five actions: four corresponding to movements in the cardinal directions and one no-op action. The observation space is 2-dimensional, consisting of the agent's x and y coordinates.

Although the dynamics are consistent across tasks, the labeling functions mapping MDP states to atomic propositions, as well as task-specific details, differ from each other.

Reach-avoidance Reach-avoidance is assessed in a gridworld without obstacles as shown in the first row of Fig.6. The agent starts at one end of a narrow corridor with a unit-width layout. While the opposite end of the corridor is unbounded, the atomic proposition a evaluates to true in all cells of the corridor located beyond a certain fixed distance from the starting position. Conversely, the AP z evaluates to true in all areas outside the corridor. The LTL formula is $\mathsf{F}(a) \wedge \mathsf{G}\neg z$. The task's difficulty increases with the distance to the target zone, set to 7, 9, and 11 for the easy, medium, and hard variants, respectively. Each episode lasts for the minimum number of steps required to reach the target zone, plus an additional 10 steps.

Sequential Sequential task is shown in the second row of Fig. 6, it also takes place in a gridworld without obstacles, consisting of contiguous 7×7 squares aligned horizontally. The agent starts at the center of the leftmost square, and in each subsequent square to the right, a different atomic proposition (AP) evaluates to true in alphabetical order. Specifically, a evaluates to true in the first square to the right of the starting position, b in the second, and so on. The task's difficulty is scaled by using progressively longer temporal logic formulas: $\mathsf{F}(a \wedge \mathsf{XF}(b \wedge \mathsf{XF}c))$, $\mathsf{F}(a \wedge \mathsf{XF}(b \wedge \mathsf{XF}(c \wedge \mathsf{XF}d)))$, and $\mathsf{F}(a \wedge \mathsf{XF}(b \wedge \mathsf{XF}(c \wedge \mathsf{XF}(d \wedge \mathsf{XF}e))))$ for the easy, medium, and hard variants, respectively. Each episode has a fixed length of 72 steps.

Circular Circular tasks are shown in the third row of Fig. 6 They consist of 5 contiguous 7×7 squares arranged in a cross formation. The central zone with 6×6 cells acts as an obstacle, labeled with the atomic proposition z, and the agent is unable to access it. The remaining 4 zones are labeled a, b, c, and d in a counterclockwise order, with the agent initialized near the first zone. Task difficulty is scaled by increasing the number of zones involved in the loop, with the following formulas used: $\mathsf{GF}(a \wedge \mathsf{XF}(b)) \wedge \mathsf{G}\neg z$ for the easy variant, $\mathsf{GF}(a \wedge \mathsf{XF}(b \wedge \mathsf{XF}c)) \wedge \mathsf{G}\neg z$for medium, and $\mathsf{GF}(a \wedge \mathsf{XF}(b \wedge \mathsf{XF}(c \wedge \mathsf{XF}d))) \wedge \mathsf{G}\neg z$ for hard. Each episode has 72 steps for easy and medium mode; hard mode has 144 steps.

H.1.2 Continuous Environments

Carlo The Carlo environment (illustrated in the second row, last column of Fig. 3) is a simplified self-driving simulator based on a bicycle model for its dynamics. The agent observes its position, velocity, and heading (in radians), resulting in a 5-dimensional observation space. The agent controls its heading and throttle, with an action space of $[-1, 1]^2$. For this domain, we use a circular track where the agent starts at the center of the road at an angle of $\frac{(1+2i)\pi}{4}$, $\quad i = 0$ and drives counterclockwise around the circle without crashing. The task is defined by $\mathsf{GF}(wp_0 \wedge \mathsf{XF}(wp_1)) \wedge \mathsf{G}\neg crash$, to visit the blue regions wp_0, wp_1 repeatedly while avoiding the gray region $crash$. The episode length is 500.

Doggo Doggo is a 12-DoF quadruped adapted from the most challenging tasks in Safety-Gym [52], designed to navigate a flat plane. The observation space is 66-dimensional, the action space is 12-dimensional, and each episode lasts 500 steps. Similar to other reach-avoidance tasks, *Doggo Avoid* requires the agent to navigate directly to a distant goal along a straight path, avoiding any detours as shown in the first column, the second row of Fig. 3. In contrast, *Doggo Navigate* involves navigating through a sequence of two circular zones as shown in the first column, the third row of Fig. 3. These tasks are defined by the following specifications: $\mathsf{F}a \wedge \mathsf{G}\neg z$ for *Doggo Avoid* and $\mathsf{F}(a \wedge \mathsf{XF}(b))$ for *Doggo Navigate*. Both tasks have a fixed episode length of 500 steps.

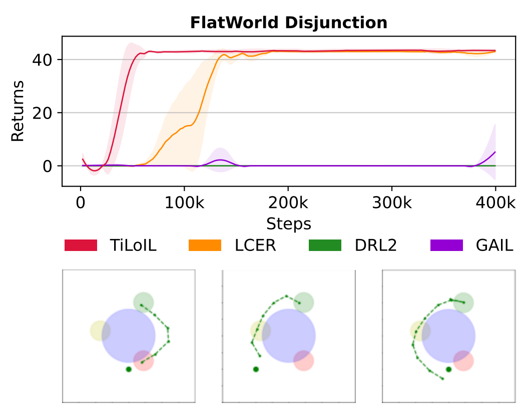

Figure 9: Disjunctive LTL objective in FlatWorld. The task requires reaching (yellow $\rightarrow$ green) or (red $\rightarrow$ green) while avoiding blue. The first row shows that returns under eventual discounting comparing TiLoIL with the baselines over 10 random seeds. The second row illustrates the trajectories of TiLoIL and LCER. The first image shows TiLoIL starting near the red zone, while the second shows TiLoIL starting near the green zone. The third image depicts the LCER trajectory from a red-zone start. Although LCER achieves similar returns to TiLoIL, it fails to adapt its path based on the initial state, always following yellow $\rightarrow$ green. In contrast, TiLoIL dynamically selects the optimal path (e.g., red $\rightarrow$ green when starting closer to red).

Table 1: Hyperparameters for Q-learning and SoftActorCritic.

HYPERPARAMETER	VALUE
γ	0.99
α	0.2
BUFFER SIZE	$1 \cdot 10^6$
BATCH SIZE	64
LEANING STARTS	2000
τ	$1 \cdot 10^{-4}$
Q LEARNING RATE	$3 \cdot 10^{-4}$
ACTOR LEARNING RATE	$3 \cdot 10^{-4}$
CRITIC LEARNING RATE	$3 \cdot 10^{-4}$
DISCRIMINATOR LEARNING RATE	$3 \cdot 10^{-4}$
DISCRIMINATOR UPDATE FREQUENCY	1 STEP
TARGET NETWORK UPDATE FREQUENCY	1 STEP

Fetch Fetch environments are based on the widely used Fetch robotic benchmark [17]. We evaluate two tasks where the agent controls the end effector position of a 7-DoF robotic arm using 4-dimensional actions over 50-step episodes. *Fetch Avoid*, inspired by the reach-avoidance task in tabular settings, requires the arm to fully extend while avoiding lateral movements (i.e., reaching the green zone and avoiding red zones as shown in the second row of Fig. 3). In this task, the observation space is 10-dimensional. *Fetch Align*, on the other hand, involves interacting with three cubes on one side of the table and aligning them horizontally at the center. For this task, the observation space includes information about the cubes and is 45-dimensional. The two tasks are defined by the following specifications: $\mathsf{F}a \wedge \mathsf{G}\neg z$ (reach the end of the table while avoiding lateral movements) for *Fetch Avoid*, and $\mathsf{F}(a \wedge \mathsf{XF}(b \wedge \mathsf{XF}c))$ (position the first, second, and third block sequentially) for *Fetch Align*.

FlatWorld The Flatworld environment (illustrated in the first row of Fig. 3) is a two-dimensional continuous world. The agent, represented by a green dot, starts at position $(-1, -1)$. The dynamics of the environment are defined by: $x = x + \frac{a}{10}$ where $x \in \mathbb{R}^2$ and $a \in [0, 1]^2$. Here we define three tasks *FlatWorld Stabilization*, *FlatWorld Cycle* and *FlatWorld Disjunction*. *FlatWorld Stabilization* also inspired by the reach-avoidance task in tabular settings, requires the agent to reach the yellow zone at left and avoid the blue zones in the middle. *FlatWorld Cycle* requires the agent to visit red, yellow, and green zones in order repeatedly while avoid the blue zone. *FlatWorld Disjunction* requires the agent to visit yellow then green, or red then green while avoid the blue zone. The three tasks are defined by the following specifications: $\mathsf{FG}(y) \wedge \mathsf{G}\neg b$ for *FlatWorld Stabilization*, $\mathsf{GF}(r \wedge \mathsf{XF}(g \wedge \mathsf{XF}y)) \wedge \mathsf{G}\neg b$ for *FlatWorld Cycle*, and $(\mathsf{F}(y \wedge \mathsf{XF}g) \vee \mathsf{F}(r \wedge \mathsf{XF}g)) \wedge \mathsf{G}\neg b$ for *FlatWorld Disjunction*. Fig. 9 shows the result for *FlatWorld Disjunction*. The episode length is 50 for all tasks.

HalfCheetah HalfCheetah [61], a standard environment in deep reinforcement learning, involves controlling a 6-DoF robot in a vertical 2D plane. In *Cheetah Flip*, the agent follows the formula $\mathsf{GF}(b \wedge \mathsf{XF}d)$, where each variable corresponds to a specific range of angles for the robot's main body. These angles represent the Cheetah standing on its front legs and standing on its back legs, respectively. The task requires the agent to perform a sequence of frontflips to satisfy the formula. The episode length is 1000.

H.2 Methods and Algorithms

We utilize SAC [25] and tabular Q learning [66] as the backbone RL algorithm. The architectures of the SAC networks are shown below:

- Actor Network:4-layer MLP, hidden units(256,256,256)
- Critic Networks:3-layer MLP, hidden units(256,256)
- Discriminator Networks(Reward):2-layer MLP,hidden units(32)

The corresponding hyperparameters are provided in Table 1. For challenging tasks, we allow more steps for initial random exploration. For *Cheetah Flip*, learning starts is $2 \cdot 10^4$.

H.3 Metrics

Performance is evaluated using cumulative return under eventual discounting [63], an RL-friendly proxy objective from prior work [27, 63] that optimizes a lower bound on the probability of LTL formula satisfaction.

$$\pi^\star_\gamma \in \arg\max_{\pi \in \Pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{i=0}^{\infty} \Gamma_i R^\times(b_i) \right] (=: V_\pi^\gamma),$$

where

$$R^\times(b_i) = 1_{\{b_i \in \mathcal{B}^*\}}, \quad \Gamma_0 = 1, \quad \Gamma_i = \prod_{t=0}^{i-1} \gamma^\times(b_t), \tag{17}$$

and

$$\gamma^\times(b_t) = \begin{cases} \gamma, & b_t \in \mathcal{B}^*, \\ 1, & \text{otherwise.} \end{cases} \tag{18}$$

All performance curves represent the mean estimated across 10 seeds, with shaded areas indicating variance.

H.4 Tools

Our codebase primarily utilizes NumPy [26] for numerical computations and Torch [49] for its autograd capabilities. Additionally, we partially automate the synthesis of LDBAs from LTL formulas using Rabinizer [40].

H.5 Computational Costs

All methods exhibit similar runtimes within their respective settings (tabular or continuous). Each experimental run was conducted using NVIDIA Quadro RTX 6000 GPU. On average, a single run took approximately 30 minutes in the tabular setting and 3 hours in the continuous setting.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, the abstract and introduction clearly state the main claims of the paper, including the background on LTL with RL, motivations and contributions. The conclusions made are based on a wide range of benchmarks and results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We included a clear discussion on the limitations. TiLoIL requires suboptimal demonstrations for imitation learning. So the future work will focus on using suboptimal demonstrations to guide agent exploration toward LTL satisfaction.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We included clear theorems and proof at Theorem. 3.1, Theorem. 3.2 and Appendix D. They are cross linked in the main paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provided the necessary details to reproduce the main experimental results in Appendix H.2. This includes that we use SAC and Q learning as a backbone RL algorithm. It shows the structure of the sac network, the structure of the discriminators, and the corresponding hyperparameters.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We include our code and demonstrations as a zip file in the supplementary material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We include the experimental details in Sec. 4 and Appendix H. Appendix H.2 shows the hyperparameters.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our experimental results shown in Fig. 3 are over 10 random seeds. The solid line represents the mean, and the shaded area indicates the standard deviation.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We include the compute resources in Appendix H.5

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work fully adheres to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

   Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

   Answer: [Yes]

   Justification: We discuss broader impacts in Appendix .A

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our work does not release any models, datasets, or tools that pose a high risk of misuse or dual-use concerns.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We mentioned the assets we used in Sec. 4, and cited to their original publications.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our work included new assets and are submitted in supplementary material.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve any crowdsourcing activities or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve human subjects. So there are no participant risks or Institutional Review Board (IRB) approval.

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We only use LLM for writing, editing, or formatting purposes. Our work does not involve the use of LLM in any important or original way.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.