TW-CRL: Time-Weighted Contrastive Reward Learning for Efficient Inverse Reinforcement Learning

Yuxuan Li[1], **Yicheng Gao**[2], **Ning Yang**[3] *, **Stephen Xia** [1]
[1]Northwestern University
[2]University of Southern California
[3]Institute of Automation, Chinese Academy of Sciences
yuxuanli2023@u.northwestern.edu gaoyiche@usc.edu, ning.yang@ia.ac.cn, stephen.xia@northwestern.edu

Abstract

Episodic tasks in Reinforcement Learning (RL) often pose challenges due to sparse reward signals and high-dimensional state spaces, which hinder efficient learning. Additionally, these tasks often feature hidden "trap states"—irreversible failures that prevent task completion but do not provide explicit negative rewards to guide agents away from repeated errors. To address these issues, we propose Time-Weighted Contrastive Reward Learning (TW-CRL), an Inverse Reinforcement Learning (IRL) framework that leverages both successful and failed demonstrations. By incorporating temporal information, TW-CRL learns a dense reward function that identifies critical states associated with success or failure. This approach not only enables agents to avoid trap states but also encourages meaningful exploration beyond simple imitation of expert trajectories. Empirical evaluations on eight benchmarks demonstrate that TW-CRL surpasses state-of-the-art methods, achieving improved efficiency and robustness.

Code — https://github.com/yuxuanli224284/TW-CRL
Extended version — https://arxiv.org/abs/2504.05585

Introduction

An episodic task is a common Reinforcement Learning (RL) problem that has a well-defined termination condition (Sutton, Precup, and Singh 1999). Many real-world problems, such as robotic manipulation and navigation, are episodic by nature, often featuring complex state and action spaces alongside sparse reward signals (Andrychowicz et al. 2018; Zhu et al. 2020; Yu et al. 2019). The sparsity of rewards significantly complicates the training process by providing a positive reward only when the goal is achieved, leaving most steps uninformative. As a result, agents may spend excessive time in states that provide no meaningful feedback, slowing down the learning process. Moreover, certain "trap states" can make the goal permanently unreachable after a single mistake. These states do not explicitly offer negative rewards, but act as hidden traps that derail the agent's attempts to succeed.

Designing dense reward functions is an effective approach to address challenges in RL environments with sparse rewards (Gehring et al. 2022; Sutton et al. 2011; Gershman and Daw 2017; Chan et al. 2024). However, how to develop such reward functions remains a challenging task. A straightforward approach is to use distance-based rewards, where the agent receives rewards based on its distance from the goal (Trott et al. 2019). While this method is intuitive, distance-based reward may not be effective under more complex environments where the distance between current state and goal state is difficult to measure. Another promising approach to providing dense rewards is Inverse Reinforcement Learning (IRL) (Ng and Russell 2000; Abbeel and Ng 2010), which infer reward functions from expert demonstrations. These methods assign higher rewards to states or state-action pairs that more closely align with expert behavior; however, this imitation-based strategy can limit exploration and cause agents to converge to suboptimal behaviors (Mavor-Parker et al. 2022). As a result, agents may fail to discover alternative strategies such as more efficient paths. Additionally, most IRL-based approaches rely solely on successful demonstrations, which reduces data efficiency, especially in complex environments where failures are common during early training (Shiarlis, Messias, and Whiteson 2016). By ignoring failed experiences, these methods miss valuable information that could help agents learn to avoid undesirable trajectories, ultimately slowing down the learning process and reducing the robustness of the learned policies.

To address these challenges, we propose Time-Weighted Contrastive Reward Learning (TW-CRL), a novel IRL approach that trains the agent based on a learned accurate and environment-aligned dense reward function. TW-CRL contains two key components: Time-Weighted function and Contrastive Reward Learning. The Time-Weighted function incorporates temporal information from trajectories, providing denser and more informative feedback. This allows the agent to receive more accurate rewards that reflect the importance of different states over time. Contrastive Reward Learning utilizes both successful and failed demonstrations, which enables the agent to efficiently locate goals while avoiding repeatedly falling into the same failure patterns, ultimately improving learning efficiency and average episodic return.

The key contributions of this paper are as follows: (1) We define and formalize the trap state problem in episodic RL tasks, highlighting its impact on agent performance. (2) We introduce TW-CRL, a novel IRL method that effectively

*Corresponding author.

learns dense and accurate reward functions from both successful and failed demonstrations.

Related Work

IRL Approaches Most existing IRL methods focus primarily on imitating expert behavior. Maximum Entropy IRL (MaxEnt) (Ziebart et al. 2008) encourages diverse actions while maintaining consistency with expert demonstrations by maximizing entropy. Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon 2016) and Adversarial Inverse Reinforcement Learning (AIRL) (Fu, Luo, and Levine 2017) applies adversarial learning to align the agent's state-action distribution with that of the expert. Although recent research (Hugessen, Wiltzer, and Berseth 2024; Naranjo-Campos, Victores, and Balaguer 2024; Liu and Zhu 2025) continues to advance these methods, they often overlook the valuable insights that can be gained from failed demonstrations. As a result, their ability to recognize and avoid potential traps in complex environments is limited. TW-CRL overcomes this challenge by leveraging both successful and failed demonstrations, allowing agents to explore beyond simple imitation.

Learning from Failures in RL Recent studies have explored leveraging failures to improve learning. Inverse Reinforcement Learning from Failure (IRLF) (Shiarlis, Messias, and Whiteson 2016) helps agents imitate expert behavior while avoiding past failures but does not fully analyze failure patterns. Self-Adaptive Reward Shaping (SASR) (Ma et al. 2024) adjusts rewards based on past success rates to balance exploration and exploitation. Other approaches such as (Chen, Paleja, and Gombolay 2020; Reddy, Dragan, and Levine 2019; Vecerik et al. 2018) and (Hugessen, Wiltzer, and Berseth 2024) incorporate suboptimal demonstrations to aid training but typically rely only on successful yet imperfect trajectories, ignoring completely failed ones. (Gao et al. 2019; Xie et al. 2019; Guo et al. 2023; Wu et al. 2024) also focus on failure utilization but overlook the timing and sequence of failures. Unlike these methods, TW-CRL incorporates temporal information from both successful and failed demonstrations, leading to a more informative reward function.

Risk Awareness in RL Risk-aware methods in RL focus on training agents to avoid risky situations (García and Fernández 2015). Approaches such as Risk-Averse Imitation Learning (RAIL) (Santara et al. 2017; Lam et al. 2022; Jaimungal et al. 2021) and Risk-Sensitive Generative Adversarial Imitation Learning (RS-GAIL) (Lacotte et al. 2018) use the Conditional Value at Risk (CVaR) metric to assess and manage risk, while Inverse Risk-Sensitive Reinforcement Learning (IRSRL) (Ratliff and Mazumdar 2017) employs convex risk measures to minimize potential dangers. Recent studies (Fei, Yang, and Wang 2021; Zhao et al. 2025; Yang et al. 2024; Sun et al. 2023) also follow similar strategies by relying on predefined risk measures based on expert knowledge or historical data. In contrast, TW-CRL automatically identifies and avoids risky regions without requiring prior knowledge or explicit risk metrics.

Method

In this section, we introduce TW-CRL, a novel IRL method that utilizes temporal information from both failed and successful demonstrations in episodic tasks. We first introduce the background of TW-CRL, which includes the definition of episodic tasks and the proposed definition of trap states . Next, we describe how we use the Time-Weighted function to extract temporal information from trajectories, and detail the use of Contrastive Reward Learning loss function to train a reward function that helps the agent identify unseen trap states and goal states. Then, we outline the entire TW-CRL process. Finally, we discuss how TW-CRL could be extended to tasks types beyond episodic tasks with clear sets of successful terminal states.

Episodic Tasks

An episodic MDP (Sutton, Precup, and Singh 1999; Domingues et al. 2020; Neu and Pike-Burke 2020) is formally defined as a tuple $(\mathcal{S}, \mathcal{A}, T, \mu, P, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, T is the number of steps in one episode, μ is the distribution of the initial state, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the dynamics which gives the distribution of the next state s' given the current state s and the action a, and r is the reward function given by the environment. Episodic tasks, modeled by episodic MDPs, have a well-defined termination condition. In contrast, continuous tasks are tasks without clear termination conditions and proceeds indefinitely.

Trap State

Consider an episodic task with state space $\mathcal{S}$. For now, we consider tasks with horizon T and clear success/failure labels, meaning that there exists a non-empty set of goal states $\mathcal{S}_{\text{goal}} \subseteq \mathcal{S}$ that contains all terminal states indicating successful completion. Once the agent reaches a goal state $s_{\text{goal}} \in \mathcal{S}_{\text{goal}}$, it remains in the same state until the end of that episode. On the other hand, if the agent fails to reach a goal state within horizon T, then we consider this episode as failure.

In early stages of training, it is common to observe that the agent repeatedly fails in similar regions of the state space.

These regions are not explicitly labeled in the environment and typically do not provide any negative reward to signal the mistake; instead, they act as unseen traps that hinder progress toward the goal. We name these states as "trap states"—states that capture the agent and make it difficult to reach a goal state unless the policy is sufficiently strong.

Concretely, we define the trap states as follows:

Definition 0.1 (Trap state)**.** *A trap state is a state $s_{trap} \in \mathcal{S}$ such that, if s_{trap} is a trap state, then regardless of action a, the next state s'_{trap} is always a trap state. Additionally, all goal states are not trap states. Formally,*

$$\mathcal{S}_{trap} = \{s_{trap} \in \mathcal{S} - \mathcal{S}_{goal} \mid \sum_{s' \in \mathcal{S}_{trap}} P(s' \mid s_{trap}, a) = 1, \forall a \in \mathcal{A}\}. \tag{1}$$

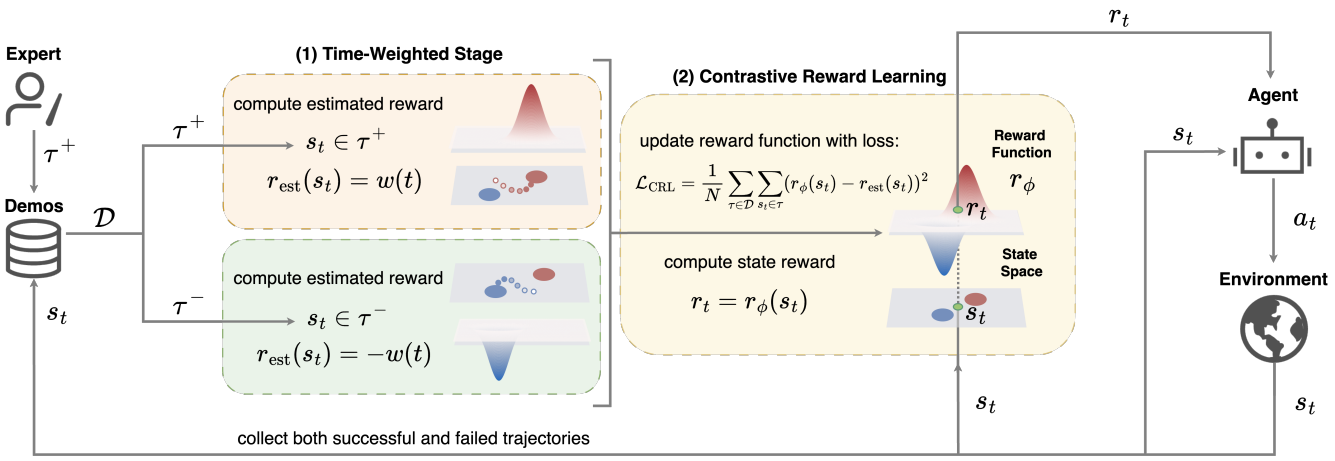


Figure 1: Overview of TW-CRL.

Here, s_{trap} is a trap state, $\mathcal{S}_{\text{trap}}$ is the set of all trap states, a is any action in the action space $\mathcal{A}$, and s' is the next state determined by the environment based on s_{trap}. Once the agent enters $\mathcal{S}_{\text{trap}}$, the next state remains within the set of trap states regardless of the action taken. For example, in a push environment (e.g., PandaPush), if the robot accidentally pushes an object off the table, it becomes difficult to recover and complete the task. These kind of trap states are induced by the environment, where some states in the environment are absorbing states that could not be escaped from. In other cases, trap states emerge because of the policy being used. This happens because, when interacting with the environment using a weak or flawed policy, the agent tends to visit similar failure-prone states—i.e., each such policy has its own effective trap region.

More generally, we extend the concept of trap states to all the states that can "trap" a non-negligible fraction of policies: once the agent enters it, only a strong enough policy can escape. Thus, the difficulty of escaping a trap state depends on policy quality.

In practice, we treat failed trajectories as a progression toward a region where each state is more difficult to escape. Specifically, although the terminal state of a failed trajectory is not guaranteed to be an absorbing trap state as defined in Definition 0.1, we treat it as a proxy for a state that is able to trap a large proportion of policies. The rationale is that the trajectory did not reach the goal, and its endpoint reflects a region from which successful continuation was unlikely under the executed policy. Thus, the terminal state of a failed demonstration is, in a posterior sense, more likely to be a trap state than a random state along the trajectory.

In order to guide the policy to learn to avoid trap states, we design a reward function that supplies negative feedback for visiting such inferred trap states.

Time-Weighted Function

In this section, we describe how TW-CRL extracts temporal information from demonstrations. As discussed before, in successful demonstrations, the agent gradually moves closer to the goal as the episode progresses, making states that appear later in the trajectory more likely to be goal states. Conversely, in failed demonstrations, later states are more likely to be trap states. To capture this temporal feature, we model the dynamics of the agent as an absorbing Markov chain. For failed demonstrations, the trap states are absorbing states and non-trap states are transient states. In a failed demonstration, recall that we assume the final state is always a trap state ($s_T \in \mathcal{S}_{\text{trap}}$). Additionally, we also assume that the initial state is not a trap state ($s_0 \notin \mathcal{S}_{\text{trap}}$). Under these assumptions, given the prior knowledge that a demonstration is a failed demonstration, we can derive the conditional probability of a state s_t of timestep t being a trap state as follows:

$$P(s_t \in \mathcal{S}_{\text{trap}} | s_T \in \mathcal{S}_{\text{trap}}) = \frac{1-(1-f(t,T))^t}{1-(1-f(t,T))^T} \quad (2)$$

where $f : \mathbb{N}\times\mathbb{N} \to \mathbb{R}$ is a function that indicates the probability of a non-trap state transitioning into a trap state in timestep t given total horizon T. The derivation of Equation 2 is in Appendix A.1. In practice, $f(t,T)$ can be adjusted to meet different needs of specific environments. In our experiments, we let $f(t,T) \propto e^{\alpha t}$, where $\alpha > 0$ is a hyperparameter. This is because we observe an exponential increase with respect to the timestep matches the best with our environments. After normalization to ensure that $f(0,T) = 0$ and $f(T,T) = 1$, we have

$$f(t,T) = \frac{e^{\alpha t} - 1}{e^{\alpha T} - 1} \quad (3)$$

Given this assumption, Equation 2 can be further simplified:

$$P(s_t \in \mathcal{S}_{\text{trap}} | s_T \in \mathcal{S}_{\text{trap}}) \approx 1 - (1 - \frac{e^{\alpha t} - 1}{e^{\alpha T} - 1})^t \quad (4)$$

For successful demonstrations, the goal states are absorbing states, and non-goal states are transient states. Similar to our conclusion above, we can derive that:

$$P(s_t \in \mathcal{S}_{\text{goal}} | s_T \in \mathcal{S}_{\text{goal}}) \approx 1 - (1 - \frac{e^{\alpha t} - 1}{e^{\alpha T} - 1})^t \quad (5)$$

The derivations of Equation 4 and Equation 5 are in Appendix A.2. We can observe that the right hand side of Equation 4 and Equation 5 are the same. We thus use it as our Time-Weighted function $w(t)$ for both successful and failed demonstrations:

$$w(t) = 1 - (1 - \frac{e^{\alpha t} - 1}{e^{\alpha T} - 1})^t \quad (6)$$

Further discussions regarding the properties of $w(t)$ is in Appendix A.3. As we'll discuss later, the Time-Weighted function is used as a weighting function for states in different timesteps within each demonstration. By adopting the simplified conditional probabilities derived above as our Time-Weighted function, we make sure that higher weights are assigned to states that have heavier impacts on the success or failure of the trajectory they're in. Specifically, the weight assigned to each state equals the probability of the agent entering $\mathcal{S}_{\text{goal}}$ / $\mathcal{S}_{\text{trap}}$ in that timestep.

Contrastive Reward Learning

In TW-CRL, the reward function r_ϕ is modeled using a deep neural network and trained in a supervised manner. The training process involves creating a dataset from both successful and failed demonstrations. Concretely, successful and failed demonstrations are converted into supervised learning labels using the Time-Weighted function introduced previously. The key idea is to frame the estimation of $r_\phi(s_t)$ as a supervised regression problem with contrastive labels, where successful demonstrations serve as "positive" examples and failed demonstrations as "negative" examples.

Intuitively, we aim to train a reward function with labels $r_{\text{est}}(s_t)$ of the following form for each state s_t:

$$r_{\text{est}}(s_t) \propto P(s_t \in \mathcal{S}_{\text{goal}}) - P(s_t \in \mathcal{S}_{\text{trap}}), \quad (7)$$

However, in continuous environments, it is difficult to obtain the same state in both successful and failed demonstrations to estimate $P(s_t \in \mathcal{S}_{\text{goal}})$ and $P(s_t \in \mathcal{S}_{\text{trap}})$ for labeling. To work around this issue, we instead use the Time-Weighted function $w(t)$ introduced in Section to compute the estimated reward r_{est} for each state as follows:

$$r_{\text{est}}(s_t) = \begin{cases} w(t), & \text{if } s_t \in \tau^+ \\ -w(t), & \text{if } s_t \in \tau^- \end{cases} \quad (8)$$

For a state s_t in a successful demonstration τ^+, we assign a positive value $w(t)$ as its estimated reward, meaning the state is more likely to help achieve the goal. For a state s_t in a failed demonstration τ^-, we assign a negative reward $-w(t)$, meaning the state is more likely to cause failure. We show in Appendix A.4 that training the reward function using labels shown in Equation 8 is similar to Equation 7. We then introduce the **Contrastive Reward Learning loss function** to train the reward function in a simple way:

$$\mathcal{L}_{\text{CRL}} = \frac{1}{N}[\sum_{\tau \in \mathcal{D}^+} \sum_{s_t \in \tau} (r_\phi(s_t) - w(t))^2 + \sum_{\tau \in \mathcal{D}^-} \sum_{s_t \in \tau} (r_\phi(s_t) + w(t))^2], \quad (9)$$

where $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$ represents the collection of all trajectories from both successful and failed demonstrations, and N is the total number of states. Derivation of Equation 9 is in Appendix A.5. This loss function minimizes the Mean Squared Error (MSE) between the predicted rewards and the estimated reward of s_t. By minimizing the proposed loss function, the reward function is able to effectively learn to distinguish between states associated with successful task completion and those leading to failure.

TW-CRL: Complete Algorithm Pipeline

An overview of the TW-CRL framework is presented in Figure 1, and the corresponding pseudocode is provided in Appendix B. TW-CRL follows an iterative process that consists of training a reward function, optimizing a policy, and collecting new data through agent-environment interaction. Similar to other IRL methods, TW-CRL requires only successful demonstrations at the beginning, while failed demonstrations are collected later during the agent's interaction with the environment.

To begin with, TW-CRL estimates the reward for each state, $r_{\text{est}}(s_t)$, using the Time-Weighted function (Equation 6) and optimizes the reward function r_ϕ by minimizing the Contrastive Reward Learning loss (Equation 9). Once the reward function is trained, any suitable reinforcement learning algorithm POLICY-OPT(r, π) can be used to update the policy π_θ. We use TD3(Fujimoto, van Hoof, and Meger 2018) in our experiments. After the initial policy is trained, the agent interacts with the environment and generates new trajectories, which may include both successful and failed demonstrations. These new demonstrations are added to the dataset to further improve the reward function. After updating the reward function, the policy π_θ is retrained based on the updated rewards. This iterative process allows the policy to make better use of the improved reward signal, helping the agent focus on high-reward states while avoiding potential trap states.

Extending TW-CRL to Other Task Types

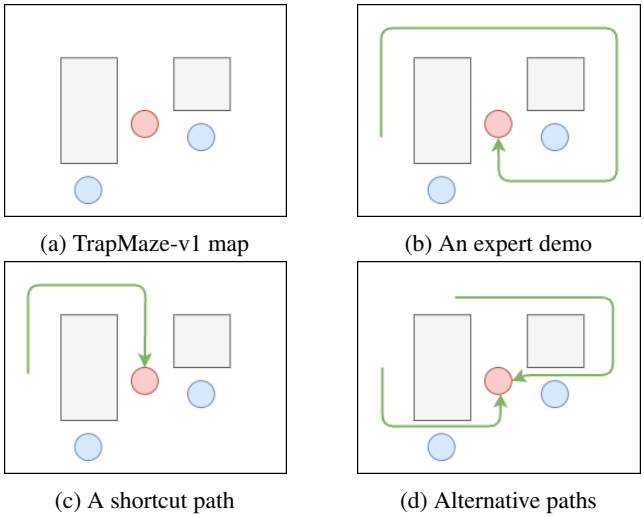
In the discussions above, we assume that the notion of "goal states" and "trap states" exist in the task. In other words, we assume that the task TW-CRL is solving must be an episodic task with terminal states that clearly signals success or failure. In this section, we discuss how TW-CRL could be applied to tasks beyond this category.

Extending to episodic tasks without clear success / failure terminal states. For this kind of tasks, we split all trajectories into success and failure according to the total episodic return. Specifically, we apply a threshold r_θ, which is a hyperparameter, to all the expert demonstrations as well as collected trajectories. All trajectories with episodic returns higher than or equal to r_θ are classified as successful trajectories, while others are classified as failed trajectories.

Environment	TW-CRL	SASR	MERIT	ICRL	GAIL	AIRL	MaxEnt
U-Maze	**273.05 ± 1.21**	192.72 ± 78.15	194.72 ± 19.01	173.73 ± 25.25	138.64 ± 20.25	208.21 ± 17.83	161.28 ± 41.38
TrapMaze-v1	**253.12 ± 9.46**	196.28 ± 32.89	51.20 ± 43.68	108.12 ± 36.52	159.06 ± 25.53	160.22 ± 11.25	129.70 ± 18.50
TrapMaze-v2	**201.27 ± 7.47**	63.68 ± 26.71	68.83 ± 20.59	72.60 ± 26.35	113.94 ± 50.26	48.95 ± 19.92	77.89 ± 30.10
MountainCar	**94.23 ± 0.07**	93.97 ± 0.09	91.70 ± 0.19	92.02 ± 0.10	91.80 ± 0.39	93.86 ± 0.53	91.71 ± 0.17
HumanStand	**144.69 ± 1.05**	128.22 ± 22.9	135.39 ± 3.37	57.89 ± 7.67	121.32 ± 9.35	134.83 ± 10.45	116.63 ± 20.82
Ant	**24.26 ± 0.12**	17.62 ± 0.87	17.34 ± 1.15	18.14 ± 0.40	15.22 ± 3.98	18.72 ± 1.43	-0.12 ± 0.01
PandaReach	**-5.98 ± 0.55**	-47.41 ± 0.51	-11.35 ± 19.85	-24.00 ± 12.83	-46.53 ± 2.28	-10.76 ± 5.93	-5.97 ± 2.28
PandaPush	**-34.89 ± 6.58**	-96.77 ± 2.34	-76.09 ± 14.64	-88.32 ± 3.42	-99.00 ± 1.40	-97.39 ± 3.00	-96.45 ± 2.54

Table 1: Average final return across benchmarks. The best performance is marked in bold, and ± represents the standard deviation over five runs.

Extending to continuous tasks. For this kind of tasks, we follow common practice and set a horizon T. Each trajectory is then truncated at T timesteps. We can then treat it in the same way as episodic tasks without clear success / failure terminal states as described above. Since the value of T is often defined in the environment (as in all the environments we use introduced in the experiment section), there's no need to tune it as a hyperparameter.



(a) TrapMaze-v1 map (b) An expert demo

(c) A shortcut path (d) Alternative paths

Figure 2: Illustration of the map and potential trajectories in TrapMaze-v1.

Experiments

In this section, we conduct experiments to evaluate the performance of TW-CRL, visualize and analyze the reward function in TrapMaze-v1, examine TW-CRL's ability to generalize to various goal settings, and perform ablation studies. Details regarding hyperparameters, network architectures, additional training setups, and other information are provided in Appendix C.1, Appendix C.2, and Appendix C.4.

Benchmarks We use eight environments in total. This includes U-Maze and its two variants TrapMaze-v1 and TrapMaze-v2 which feature unknown trap states and randomized goal locations(Fu et al. 2020; de Lazcano et al. 2024), HumanoidStand, Ant(Todorov, Erez, and Tassa 2012),MountainCarCountinuous(Towers et al. 2024), PandaReach, and PandaPush(Gallouédec et al. 2021; Plappert et al. 2018). The environments are illustrated in Appendix C.1.

Baselines We evaluate TW-CRL against six baselines: four IRL methods (GAIL (Ho and Ermon 2016), AIRL (Fu, Luo, and Levine 2017), MaxEnt (Ziebart et al. 2008) and MERIT (Liu and Zhu 2025)), ICRL (Hugessen, Wiltzer, and Berseth 2024), which leverages suboptimal demonstrations, and SASR (Ma et al. 2024), which incorporates failed demonstrations. Detailed introductions for the baseline methods can be found in Appendix E.

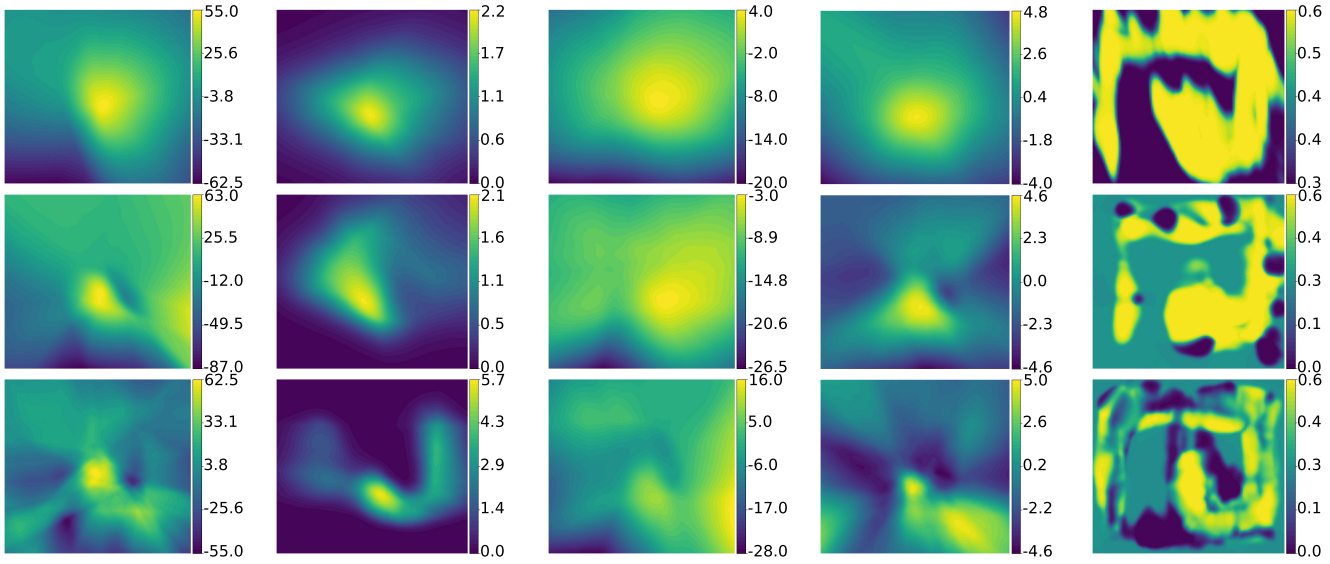
Comparison Evaluation

Table 1 report the average episodic returns and their variances across all tasks and baselines. We provide five successful demonstrations in the basic U-Maze environment, while TrapMaze-v1 and TrapMaze-v2 each receive 35 successful demonstrations, all ending in a randomly initialized goal located within a single designated grid area of the maze. During training, the same grid area is used for goal initialization, though the specific location within that grid varies each episode. In PandaReach and PandaPush, we offer 10 successful demonstrations. More details of experiment results are shown in the Appendix.

The results indicate that the performance of TW-CRL is better than or comparable to all baselines in terms of average episode return, with faster convergence and reduced variance. For example, in the U-Maze environment, the average final return in our algorithm demonstrates enhancements of 41.71%, 40.22%, 57.16%, 96.96%, 31.15%, and 69.33% over SASR, MERIT, ICRL, GAIL, AIRL, and MaxEnt, respectively.

Analysis of Learned Rewards

In this section, we analyze the learned reward functions across different methods using the TrapMaze-v1 environment to illustrate why TW-CRL outperforms other approaches. The simplified illustration of the TrapMaze-v1 environment map is shown in Figure 2a, where the red area represents the goal states, the blue area represents the trap states, and the gray blocks represent walls that the agent cannot get through. We provide a total of 35 successful demonstrations in which agents follow the main path to the goal, as illustrated in Figure 2b. Figure 3 visualizes the reward functions learned by



(a) TW-CRL (b) GAIL (c) AIRL (d) MaxEnt (e) SASR

Figure 3: Visualization of the reward function in the TrapMaze-v1 environment for TW-CRL and baseline methods. Each column represents a different method, and each row shows a training stage, with the final row illustrating the fully trained reward functions.

different methods throughout the training process. An exhaustive figure containing all baseline methods is shown in Appendix D.2.

The reward function visualizations reveal that GAIL, AIRL, and MaxEnt fail to recognize unseen trap states, and give high rewards only to states seen in successful demonstrations. As a result, these methods limit the agent to only imitating expert behavior without discovering better ways to reach the goal. On the other hand, SASR can detect unseen trap states by using failed demonstrations. It primarily assigns higher rewards to states that frequently appear in successful trajectories and lower rewards to those that frequently appear in failed trajectories. However, this approach still limits the agent's ability to explore and find better solutions.

TW-CRL overcomes these limitations by enabling the agent to avoid trap states and efficiently discover paths to the goal. This is achieved by 1) utilizing both successful and failed trajectories in the Contrastive Reward Learning process and 2) utilizing temporal information in our Time-Weighted function to only place high emphasis on later states that may have heavier impacts on the outcome of the trajectory. As a result, TW-CRL not only identifies both goal and trap states more accurately than imitation-only methods, but also encourages broader exploration. For example, the shortcut in Figure 2c and the alternative paths in Figure 2d were never shown in the expert demonstrations, but TW-CRL allows the agent to explore these paths. This is made possible by the design of the Time-Weighted function, where only states close to the terminal state are assigned high weights. By doing so, we avoid punishing or rewarding early states that are hardly related to the result of the trajectory.

	1	3	Any
TW-CRL	**253.12 ± 9.46**	**240.48 ± 4.44**	**221.33 ± 11.58**
SASR	196.28 ± 32.89	150.84 ± 50.95	122.78 ± 20.39
MERIT	51.20 ± 50.95	96.37 ± 11.94	67.93 ± 23.7
ICRL	108.12 ± 36.52	39.59 ± 17.93	90.12 ± 22.56
GAIL	159.06 ± 25.53	59.79 ± 22.87	55.25 ± 21.67

Table 2: Average return under three goal reset settings. 1, 3, and any. #goal represents the number of grids where the goal state might be in.

Generalization Analysis

We then evaluate TW-CRL's ability to generalize beyond the goal configurations provided in the original demonstrations in the TrapMaze-v1 environment, where the goal is randomly placed within one or several grid cells. Specifically, we provide demonstrations where the goal is initialized within a single grid cell, but during training and testing, the goal is allowed to be initialized within 1, 3, or any grid cells in the maze. Table 2 reports the average returns for each configuration. While all methods experience some drop in performance as the distribution of possible goal states broadens, TW-CRL remains consistently superior to the baselines, suggesting that its learned reward function can adapt successfully to unseen goal locations. Further results and details can be found in Appendix C.1 and Appendix D.1.

Ablation Study

Effect of the Time-Weighted function Figure 4 and Figure 5 (top) compare TW-CRL with the Time-Weighted function

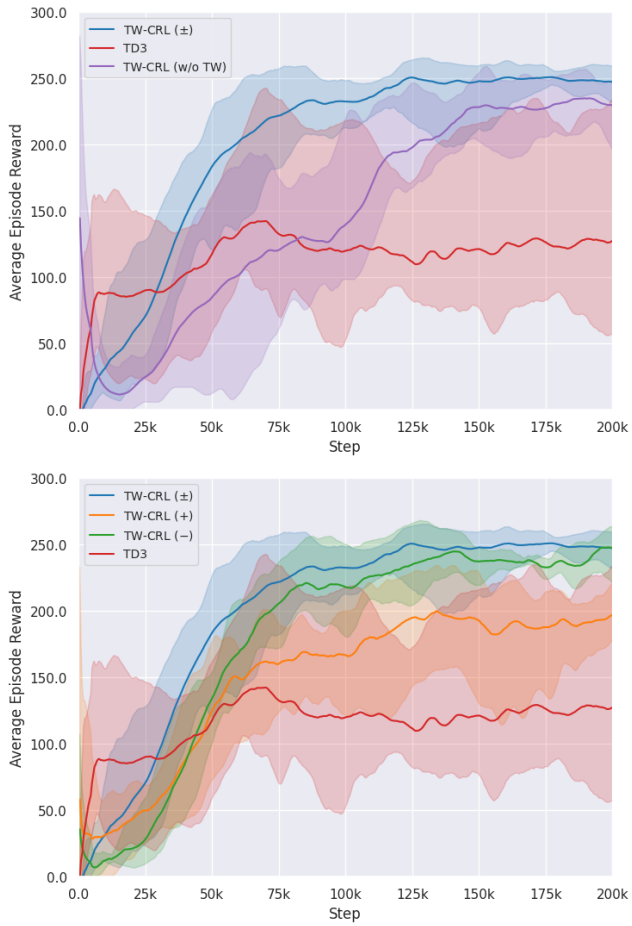


Figure 4: Ablation studies in TrapMaze-v1. The top figure shows the ablation of the Time-Weighted function, the bottom figure shows the ablation of the Contrastive Reward Learning loss function.

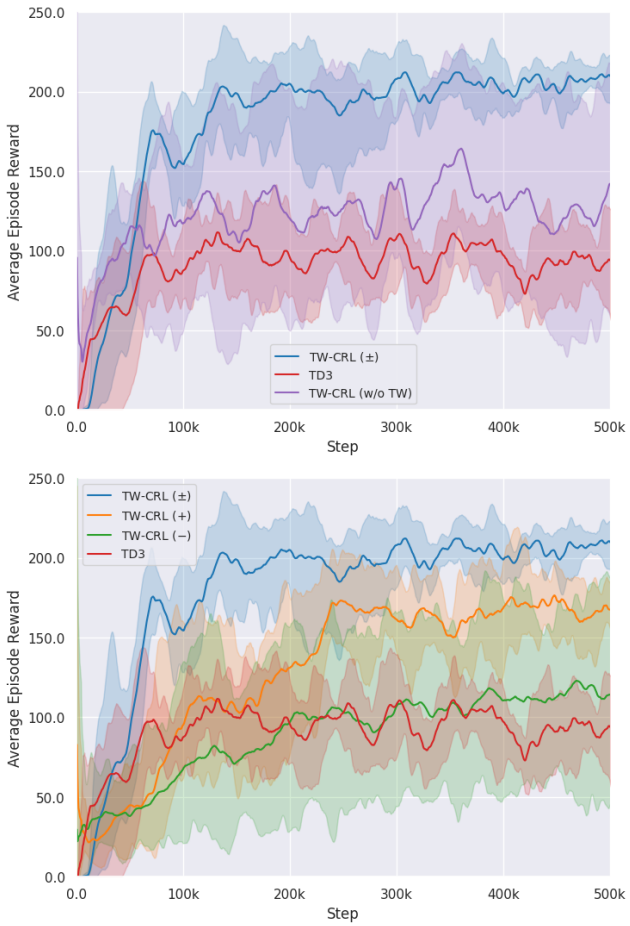


Figure 5: Ablation studies in TrapMaze-v2. The top figure shows the ablation of the Time-Weighted function, the bottom figure shows the ablation of the Contrastive Reward Learning loss function.

(blue) to a simpler approach using constant rewards of $+1$ for success states and -1 for failure states (purple). Results show that using time-weighted rewards accelerates convergence and achieves higher final returns in both TrapMaze-v1 and TrapMaze-v2. The advantage is even clearer in TrapMaze-v2, where the time-weighted approach significantly outperforms the non-time-weighted version and vanilla TD3. These findings suggest that time weighting is particularly beneficial in complex tasks.

Effect of using both successful and failed demonstrations Figure 4 and Figure 5 (bottom) compare four different settings: (1) TW-CRL with both successful and failed demonstrations (blue), (2) TW-CRL with only successful demonstrations (orange), (3) TW-CRL with only failed demonstrations (green), and (4) vanilla TD3 without demonstrations (red). The results show that using both successful and failed demonstrations (blue) achieves the fastest convergence and highest final reward. In contrast, using only successful (orange) or failed (green) demonstrations results in slower learning and slightly lower performance, though both still surpass TD3 (red). This trend is more pronounced in the more complex TrapMaze-v2 environment, confirming that combining both types of demonstrations provides a stronger learning signal.

Conclusion

In this paper, we propose TW-CRL, an inverse reinforcement learning approach that infers reward functions by leveraging temporal information from both successful and failed demonstrations. By incorporating the Time-Weighted function and Contrastive Reward Learning, TW-CRL learns an accurate and informative reward function, enabling the agent to efficiently locate goals while avoiding repeated failures, thereby enhancing learning efficiency and increasing task success rates. Additionally, TW-CRL enhances exploration beyond expert demonstrations by discovering alternative high-value paths or shortcuts that were not explicitly demonstrated. Empirical evaluations demonstrate that TW-CRL outperforms baseline methods, showcasing its effectiveness in enhancing agent performance and learning efficiency in complex tasks.

Acknowledgments

Ning Yang was supported by the National Natural Science Foundation of China under Grants 62301559.

References

Abbeel, P.; and Ng, A. Y. 2010. Inverse Reinforcement Learning. In *Encyclopedia of Machine Learning*.

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, P.; and Zaremba, W. 2018. Hindsight Experience Replay. *arXiv*, 1707.01495.

Chan, A. J.; Sun, H.; Holt, S.; and van der Schaar, M. 2024. Dense Reward for Free in Reinforcement Learning from Human Feedback. *arXiv*, 2402.00782.

Chen, L.; Paleja, R.; and Gombolay, M. 2020. Learning from Suboptimal Demonstration via Self-Supervised Reward Regression. *arXiv*, 2010.11723.

de Lazcano, R.; Kallinteris, A.; Tai, J. J.; Lee, S. R.; and Terry, J. 2024. Gymnasium Robotics. https://github.com/Farama-Foundation/Gymnasium-Robotics. Accessed: 2024-10-01.

Domingues, O. D.; M'enard, P.; Kaufmann, E.; and Valko, M. 2020. Episodic Reinforcement Learning in Finite MDPs: Minimax Lower Bounds Revisited. *ArXiv*, abs/2010.03531.

Fei, Y.; Yang, Z.; and Wang, Z. 2021. Risk-Sensitive Reinforcement Learning with Function Approximation: A Debiasing Approach. In *International Conference on Machine Learning*.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *arXiv*, 2004.07219.

Fu, J.; Luo, K.; and Levine, S. 2017. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning. *ArXiv*, abs/1710.11248.

Fujimoto, S.; van Hoof, H.; and Meger, D. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*.

Gallouédec, Q.; Cazin, N.; Dellandréa, E.; and Chen, L. 2021. panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS*.

Gao, Y.; Xu, H.; Lin, J.; Yu, F.; Levine, S.; and Darrell, T. 2019. Reinforcement Learning from Imperfect Demonstrations. *arXiv*, 1802.05313.

García, J.; and Fernández, F. 2015. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16: 1437–1480.

Gehring, C.; Asai, M.; Chitnis, R.; Silver, T.; Kaelbling, L. P.; Sohrabi, S.; and Katz, M. 2022. Reinforcement Learning for Classical Planning: Viewing Heuristics as Dense Reward Generators. *arXiv*, 2109.14830.

Gershman, S. J.; and Daw, N. D. 2017. Reinforcement Learning and Episodic Memory in Humans and Animals: An Integrative Framework. *Annual Review of Psychology*, 68: 101–128.

Guo, Y.; Gao, J.; Wu, Z.; Shi, C.; and Chen, J. 2023. Reinforcement learning with Demonstrations from Mismatched Task under Sparse Reward. *arXiv*, 2212.01509.

Ho, J.; and Ermon, S. 2016. Generative Adversarial Imitation Learning. *arXiv*, 1606.03476.

Hugessen, A.; Wiltzer, H.; and Berseth, G. 2024. Simplifying Constraint Inference with Inverse Reinforcement Learning. In *First Reinforcement Learning Safety Workshop*.

Jaimungal, S.; Pesenti, S.; Wang, Y. S.; and Tatsat, H. 2021. Robust Risk-Aware Reinforcement Learning. *arXiv*, 2108.10403.

Lacotte, J.; Ghavamzadeh, M.; Chow, Y.; and Pavone, M. 2018. Risk-Sensitive Generative Adversarial Imitation Learning. *arXiv*, 1808.04468.

Lam, T.; Verma, A.; Low, B. K. H.; and Jaillet, P. 2022. Risk-aware reinforcement learning with coherent risk measures and non-linear function approximation. In *The Eleventh International Conference on Learning Representations*.

Liu, S.; and Zhu, M. 2025. In-Trajectory Inverse Reinforcement Learning: Learn Incrementally Before An Ongoing Trajectory Terminates. *arXiv*, 2410.15612.

Ma, H.; Luo, Z.; Vo, T. V.; Sima, K.; and Leong, T.-Y. 2024. Highly Efficient Self-Adaptive Reward Shaping for Reinforcement Learning. *arXiv*, 2408.03029.

Mavor-Parker, A.; Young, K.; Barry, C.; and Griffin, L. 2022. How to Stay Curious while avoiding Noisy TVs using Aleatoric Uncertainty Estimation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 15220–15240. PMLR.

Naranjo-Campos, F. J.; Victores, J. G.; and Balaguer, C. 2024. Expert-Trajectory-Based Features for Apprenticeship Learning via Inverse Reinforcement Learning for Robotic Manipulation. *Applied Sciences*, 14(23).

Neu, G.; and Pike-Burke, C. 2020. A Unifying View of Optimism in Episodic Reinforcement Learning. *ArXiv*, abs/2007.01891.

Ng, A. Y.; and Russell, S. J. 2000. Algorithms for Inverse Reinforcement Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, 663–670. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 1558607072.

Plappert, M.; Andrychowicz, M.; Ray, A.; McGrew, B.; Baker, B.; Powell, G.; Schneider, J.; Tobin, J.; Chociej, M.; Welinder, P.; Kumar, V.; and Zaremba, W. 2018. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. *arXiv*, 1802.09464.

Ratliff, L. J.; and Mazumdar, E. 2017. Inverse Risk-Sensitive Reinforcement Learning. *arXiv*, 1703.09842.

Reddy, S.; Dragan, A. D.; and Levine, S. 2019. SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards. *arXiv*, 1905.11108.

Santara, A.; Naik, A.; Ravindran, B.; Das, D.; Mudigere, D.; Avancha, S.; and Kaul, B. 2017. RAIL: Risk-Averse Imitation Learning. *arXiv*, 1707.06658.

Shiarlis, K.; Messias, J. V.; and Whiteson, S. 2016. Inverse Reinforcement Learning from Failure. In *Adaptive Agents and Multi-Agent Systems*.

Sun, X.; Zhang, Q.; Wei, Y.; and Liu, M. 2023. Risk-Aware Deep Reinforcement Learning for Robot Crowd Navigation. *Electronics*, 12(23).

Sutton, R. S.; Modayil, J.; Delp, M.; Degris, T.; Pilarski, P. M.; White, A.; and Precup, D. 2011. Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Adaptive Agents and Multi-Agent Systems*.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.*, 112: 181–211.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033.

Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J. U.; De Cola, G.; Deleu, T.; Goulão, M.; Kallinteris, A.; Krimmel, M.; KG, A.; et al. 2024. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *arXiv prvolume arXiv:2407.17032*.

Trott, A.; Zheng, S.; Xiong, C.; and Socher, R. 2019. Keeping Your Distance: Solving Sparse Reward Tasks Using Self-Balancing Shaped Rewards. *arXiv*, 1911.01417.

Vecerik, M.; Hester, T.; Scholz, J.; Wang, F.; Pietquin, O.; Piot, B.; Heess, N.; Rothörl, T.; Lampe, T.; and Riedmiller, M. 2018. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. *arXiv*, 1707.08817.

Wu, M.; Siddique, U.; Sinha, A.; and Cao, Y. 2024. Offline Reinforcement Learning with Failure Under Sparse Reward Environments. In *2024 IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI)*, 1–5.

Xie, X.; Li, C.; Zhang, C.; Zhu, Y.; and Zhu, S.-C. 2019. Learning Virtual Grasp with Failed Demonstrations via Bayesian Inverse Reinforcement Learning. In *IROS*.

Yang, Z.; Jin, H.; Tang, Y.; and Fan, G. 2024. Risk-Aware Constrained Reinforcement Learning with Non-Stationary Policies. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '24, 2029–2037. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Yu, T.; Quillen, D.; He, Z.; Julian, R.; Hausman, K.; Finn, C.; and Levine, S. 2019. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. In *Conference on Robot Learning (CoRL)*, volume 1910.10897.

Zhao, Y.; Escamill, J. E. A.; Lu, W.; and Wang, H. 2025. RA-PbRL: Provably Efficient Risk-Aware Preference-Based Reinforcement Learning. *arXiv*, 2410.23569.

Zhu, Y.; Wong, J.; Mandlekar, A.; Martín-Martín, R.; Joshi, A.; Nasiriany, S.; Zhu, Y.; and Lin, K. 2020. robosuite: A Modular Simulation Framework and Benchmark for Robot Learning. In *arXiv prvolume arXiv:2009.12293*.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum Entropy Inverse Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*.