Generalize and Guide: Decomposing Rewards for Few-Shot Inverse Reinforcement Learning

Ziyi Liu, Grace Zhang

Keywords: Few-Shot Learning, Inverse Reinforcement Learning, Reward Decomposition.

Summary

Inverse reinforcement learning (IRL) provides a powerful framework for learning from demonstrations. However, real-world tasks often exhibit substantial natural variations (e.g., picking up mugs with varying shapes), making it impractical to collect demonstrations that fully specify a new task under every possible scenario. In practice, while demonstrations for the target task are limited, it is often easier to obtain datasets of heterogeneous but related behaviors. This motivates the problem of *few-shot IRL with multi-task demonstrations (FM-IRL)*, where an agent must learn a new task with substantial variations from only a limited number of target-task demonstrations, together with sufficient demonstrations of related tasks and online agent experience. To do so, we must both recover the expert distribution of the new task and provide guidance when the agent deviates from it. We introduce Multitask discriminator Proximity-Guided IRL (MPG), which learns two complementary reward components: (1) a *generalizable* discriminator that transfers shared structure across related tasks to identify expert behavior in a new task, and (2) a proximity function that measures how far a state deviates from expert behavior and provides corrective *guidance* during exploration. We demonstrate the effectiveness of our method on multiple challenging navigation and manipulation tasks under significant variations (e.g., object configurations, table layouts, and initial robot poses), resulting in an average 35.4% increase in success rate over the next best method.

Contribution(s)

1. We introduce a novel problem setting: *few-shot IRL with multi-task demonstrations (FM-IRL)*, where an agent must learn a new task from limited demonstrations, while mainly leveraging information from demonstrations of related tasks and online agent experience.
   Context: Prior work on learning from limited demonstrations has primarily focused on using imitation learning or meta-learning, which either cannot leverage online policy optimization or require access to multi-task environments. In practice, although interaction with multi-task environments is often difficult or impractical in real-world settings, the agent typically has access to two readily available sources of information: (1) expert demonstrations from related tasks, and (2) online interaction with the target task environment.
2. We propose Multitask discriminator Proximity-Guided IRL (MPG), a reward decomposition framework that generates robust reward for expert distribution and provides guidance for recovery from unseen states.
   Context: To generalize beyond few-shot demonstrations, our key insight is twofold: demonstrations from other tasks can be leveraged to infer the expert distribution of a new task, and online interaction can be used to estimate temporal distance to this distribution, providing corrective guidance when the agent deviates.
3. We establish strong performance in the FM-IRL setting across diverse navigation and manipulation domains.
   Context: We evaluate on maze navigation, block stacking, and robot manipulation tasks in FactorWorld. Averaged across all tasks, our method improves the success rate by 35.4% over behavioral cloning (BC), the strongest overall baseline. Our results demonstrate robustness under substantial variations and limited expert supervision.

Generalize and Guide: Decomposing Rewards for Few-Shot Inverse Reinforcement Learning

Ziyi Liu[1,†], **Grace Zhang**[1,†]
{ziyiliu29,graciez168}@gmail.com

[1]**University of Southern California**

[†] indicating equal contribution

Abstract

Inverse reinforcement learning (IRL) provides a powerful framework for learning from demonstrations. However, real-world tasks often exhibit substantial natural variations (e.g., picking up mugs with varying shapes), making it impractical to collect demonstrations that fully specify a new task under every possible scenario. In practice, while demonstrations for the target task are limited, it is often easier to obtain datasets of heterogeneous but related behaviors. This motivates the problem of *few-shot IRL with multi-task demonstrations (FM-IRL)*, where an agent must learn a new task with substantial variations from only a limited number of target-task demonstrations, together with sufficient demonstrations of related tasks and online agent experience. To do so, we must both recover the expert distribution of the new task and provide guidance when the agent deviates from it. We introduce Multitask discriminator Proximity-Guided IRL (MPG), which learns two complementary reward components: (1) a *generalizable* discriminator that transfers shared structure across related tasks to identify expert behavior in a new task, and (2) a proximity function that measures how far a state deviates from expert behavior and provides corrective *guidance* during exploration. We demonstrate the effectiveness of our method on multiple challenging navigation and manipulation tasks under significant variations (e.g., object configurations, table layouts, and initial robot poses), resulting in an average 35.4% increase in success rate over the next best method.

1 Introduction

Reinforcement Learning (RL) provides a powerful framework for sequential decision-making, but its reliance on well-shaped and often hand-engineered reward functions remains a significant bottleneck (Sutton & Barto, 2018; Amodei et al., 2016). Inverse Reinforcement Learning (IRL) (Ng & Russell, 2000) offers an alternative by inferring a reward function directly from expert demonstrations. However, many realistic settings include natural *intra-task variations* (e.g., picking up mugs with different shapes and placements), where collecting sufficient demonstrations for every scenario is prohibitively expensive, limiting the applicability of traditional IRL.

Prior work on learning from limited demonstrations has primarily focused on using imitation learning (IL) (Finn et al., 2017; Yu et al., 2018) or meta-inverse reinforcement learning (meta-IRL) (Xu et al., 2019; Yu et al., 2019). However, IL methods struggle to generalize beyond the demonstrated state distribution, while meta-IRL typically requires access to multi-task training environments—an assumption that is often impractical in real-world settings. For example, demonstrations of robotic cleaning tasks across diverse furniture configurations may be available, but recreating the corresponding environments is difficult. Importantly, the agent often still has access to two readily available sources of information: (1) expert demonstrations from related tasks, and (2) online interaction

with the target task environment. Motivated by this, we introduce a novel setting: *few-shot IRL with multi-task demonstrations (FM-IRL)* (see Table 1).

Table 1: Comparison of assumptions across related settings for learning from limited demonstrations. In practical robotics scenarios, although interaction with *Multi-task Env.* is difficult to obtain, *Multi-task Demos* are often available. Our setting reflects this scenario by leveraging *Multi-task Demos* while requiring interaction *only* with the *Target-task Env.*, without access to *Multi-task Env.*.

Problem Setting	Target-task Demos	Multi-task Demos	Target-task Env.	Multi-task Env.
Traditional IRL	✓		✓	
Few-Shot IL	✓	✓		
Meta-IRL	✓	✓	✓	✓
FM-IRL (Ours)	✓	✓	✓	

This setting closely mirrors how humans learn a new task with substantial variations under limited supervision. For example, preparing a dish after being shown how just once. Beginners often (1) start by drawing on prior experience from preparing similar dishes—distilling useful behaviors and constraints that should be encouraged or avoided across related tasks. They then (2) refine their behavior through practice. When making mistakes or drifting away from behaviors consistent with the target dish (e.g., misplacing ingredients), they should return to the expert behavior. Through this interplay between **knowledge transfer** and **corrective feedback**, humans can master a new dish without needing demonstrations for every kitchen configuration.

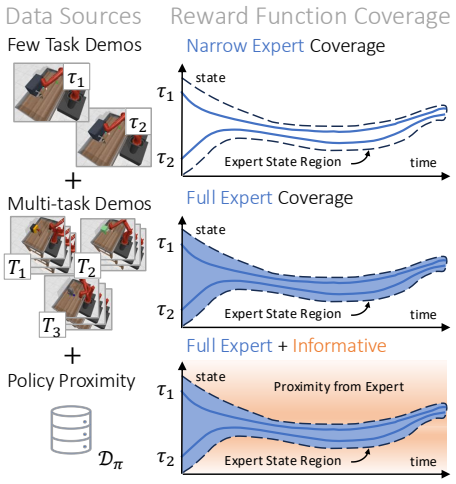


Figure 1: We learn a generalizeable and informative reward by making use of multi-task demonstrations and policy proximity.

Can we enable agents to similarly observe, practice, and master a new task with minimal supervision despite significant intra-task variation? We propose **Multitask discriminator Proximity-Guided IRL (MPG)**, a novel few-shot IRL method that decomposes the reward function into two synergistic components (Figure 1): (1) a demonstration-conditioned multi-task discriminator that *transfers knowledge* by leveraging shared structure across related tasks to generalize expert behavior to a new task under diverse variations, and (2) a proximity function that provides informative, *corrective feedback* in non-expert regions, by estimating the agent's distance to the expert state distribution.

Overall, we identify the challenge of under-specification in realistic tasks with natural variations and propose the problem setting of *FM-IRL*. Towards solving this problem setting, we propose *MPG*, a novel method that learns a generalizable and informative reward function for effective few-shot IRL. Our experimental results on maze navigation, block stacking, and robot manipulation tasks in FactorWorld (Xie et al., 2024), demonstrate that MPG achieves state-of-the-art performance against 7 other baselines in this problem setting, with an average 35.4% success rate improvement over the next best-performing method.

2 Related Work

2.1 Few-Shot Imitation Learning

Imitation learning (IL) aims to replicate expert behavior by learning directly from expert demonstrations. In this paper, we distinguish IL methods as those that learn a policy directly without inferring

or optimizing a reward function. Early approaches address few-shot IL use Behavior Cloning (BC) (Finn et al., 2017; Duan et al., 2017; Yu et al., 2018). Hakhamaneshi et al. (2021) extract skill models from an offline dataset to facilitate few-shot IL, while Dance et al. (2021) learn a demonstration-conditioned policy with access to ground-truth rewards. Other works explore offline IL (Luo et al., 2023; Xu et al., 2022; Chang et al., 2021), but do not explicitly address the challenge of few-shot imitation. Overall, IL methods suffer from compounding errors and cannot improve through online interactions without learning a reward function. In response to this, Reddy et al. (2020) propose a simple, sparse reward label to allow for policy optimization through RL. Meanwhile, Chae et al. (2022) handle environment dynamic variations by imitating multiple experts in different dynamics.

2.2 Few-Shot Inverse Reinforcement Learning

The most common approach to solve few-shot IRL is meta-learning, including context-based and gradient-based methods (Fu et al., 2018; Ziebart et al., 2008). Context-based methods (Chen et al., 2023; Seyed Ghasemipour et al., 2019; Yu et al., 2019) learn a latent context variable to represent tasks and meta-train a context-conditioned reward function. Gradient-based methods (Xu et al., 2019) learn a good initialization for the reward function, which can be quickly adapted to a new task through a one-step gradient update. Meta-learning aims to enable rapid adaptation to a meta-test suite of tasks, whereas our approach focuses on sample-efficient learning for a single target task.

Chen et al. (2021) propose DVD, a multi-task video success discriminator that generalize across task variations from a few robot demos but does not employ RL to learn a policy. Similarly, Xie et al. (2018) develop a success classifier for goal-conditioned tasks but do not learn a reward function. Our work can be viewed as an extension of these ideas. Other works study demonstration-efficient IRL in multi-task (Gleave & Habryka, 2018) and multi-agent (Filos et al., 2021) settings.

2.3 Proximity-based Rewards

Popular IRL methods (Ho & Ermon, 2016; Fu et al., 2018) learn reward functions by discriminating between agent and expert behaviors, which may provide limited guidance in non-expert states. To address this, recent work introduces reward shaping mechanisms that estimate some form of proximity to the expert. Examples includes a progress estimator for goal-conditioned tasks (Lee et al., 2021), Euclidean distance between agent's and expert's state-action pairs (Hakhamaneshi et al., 2021), geometric distance between agent and expert distribution (Dadashi et al., 2021; Haldar et al., 2022), and a transition-based reachability discriminator Chiang et al. (2024). While these approaches provide useful guidance in non-expert states, they do not account for generalization across task variations with limited demonstrations.

3 Few-Shot IRL with Multi-Task Demonstrations Problem Formulation

IRL addresses the problem of learning sequential decision-making tasks from demonstrations. We consider these tasks to be Markov decision problems (MDPs) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho, \mathcal{R})$: state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities $\mathcal{T}$, initial state distribution ρ, and underlying reward function $\mathcal{R}$. We assume $\mathcal{R}$ is unkown and must instead be inferred from a set of demonstrations $\mathcal{D}$ from an expert policy $\pi^*(a|s)$. The goal is to learn a reward function $\tilde{\mathcal{R}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that explains the expert behavior and can then be used to learn a policy $\pi(a|s)$.

We focus on a *few-shot* setting where the task exhibits significant *intra-task variations*, arising from the initial state distribution ρ_i (e.g., varying agent positions or object configurations), yet only a small set of demonstrations $\mathcal{D}_{target}$ is available. This limited data is insufficient for traditional IRL to learn a reward that generalizes to unseen instances. To overcome this challenge, we additionally provide a large multi-task demonstration dataset $\mathcal{D}_{multi} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T\}$. Each task i has its own distinct underlying reward function $\mathcal{R}_i$, transition dynamics $\mathcal{T}$, and initial state distribution ρ_i but shares the same state and action spaces $(\mathcal{S}, \mathcal{A})$. The agent's ultimate goal is to leverage the broad knowledge in $\mathcal{D}_{multi}$, information in $\mathcal{D}_{target}$, and online samples $\mathcal{D}_\pi$, to learn a reward function $\tilde{\mathcal{R}}$

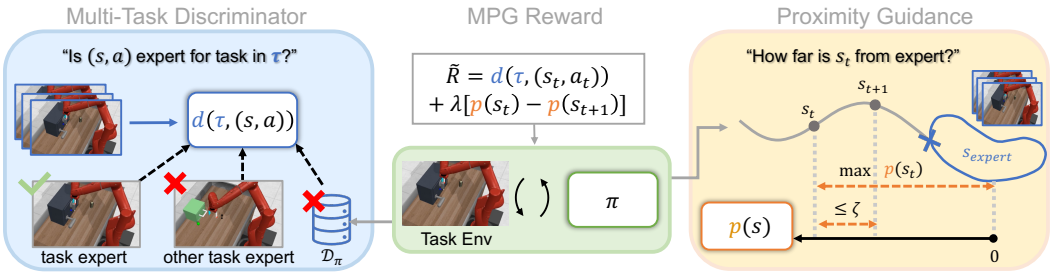


Figure 2: Our approach learns a two-part reward function, $\tilde{R}$. The multi-task discriminator extracts common structure across tasks, by predicting whether a state-action pair (s^i, a^i) is expert for the task in demonstration τ^j (i and j index tasks), to approximate expert distribution. The proximity reward estimates a state's proximity to the expert states by maximizing proximity $p_\theta(s_t)$, constrained by the triangle inequality $p_\theta(s_t) \leq p_\theta(s_{t+1}) + \zeta$. Finally, the combined reward $\tilde{R}$ integrates expert recognition and proximity guidance, enabling policy optimization via RL.

that enables an RL agent to learn a policy π that successfully solves unseen instances of the target task.

4 Method: Multitask discriminator Proximity-Guided IRL

To learn a new task from limited demonstrations, we propose MPG. Its key insight consists of two complementary components: (1) learning shared structure from a multi-task dataset to promote target task generalization (Section 4.1), and (2) acquiring recovery guidance for non-expert states through online interaction with the target environment (Section 4.2). These components enable effective policy optimization under limited demonstrations and broad task variations (Figure 2).

4.1 Multi-task Discriminator

To recognize expert behavior across intra-task variations, we propose a demonstration-conditioned discriminator $d_\phi(\tau, (s, a))$ to facilitate knowledge transfer across tasks. Given a task demonstration τ and a state and action pair (s, a), d_ϕ predicts whether (s, a) originates from the expert distribution for τ's task. Inspired by the DVD framework (Chen et al., 2021), we use two separate encoders to process τ and (s, a), respectively. The resulting embeddings are concatenated and passed through a network composed of two fully connected layers, which outputs the probability that (s, a) is consistent with expert behavior for the task described by τ.

We train d_ϕ using binary classification loss on expert demonstrations from all tasks, $\mathcal{D}_{target} \cup \mathcal{D}_{multi}$. We sample trajectories and state-action tuples from the same task as positive classification pairs and state-action tuples from different tasks as negative pairs. To prevent overfitting to expert data, we additionally incorporate online policy samples by labeling them as negative pairs, analogous to adversarial imitation learning methods (Ho & Ermon, 2016). The resulting loss combines binary classification between task demonstrations with the adversarial objective (Equation 1), where we use $\mathcal{D}$ to denote the combined dataset $\mathcal{D}_{target} \cup \mathcal{D}_{multi}$ and $\mathcal{D}_i$ to refer to the demonstrations for task i from $\mathcal{D}$.

$$
\begin{aligned}
L_{multi} = \mathbb{E}_{\tau^i, (s^i, a^i) \sim \mathcal{D}_i}[\log(1 - d(\tau^i, s^i, a^i)] \\
+ \mathbb{E}_{\tau^i \sim \mathcal{D}_i, (s^j, a^j) \sim \mathcal{D}_j, i \neq j}[\log(d(\tau^i, s^j, a^j))] + \mathbb{E}_{\tau \sim \mathcal{D}, (s,a) \sim \pi}[\log(d(\tau, s, a))]
\end{aligned} \tag{1}
$$

For notational simplicity, we denote the target task discriminator as $d(s, a)$, where the demonstration τ is implicitly sampled from $\mathcal{D}_{target}$.

4.2 Proximity Function

To provide informative rewards in non-expert states that effectively guide the policy back towards the expert state distribution, we introduce a proximity function $p_\theta(s)$. This function estimates the temporal distance between a state s and the expert state distribution (i.e., the minimum number of steps to reach an expert state). Directly computing this distance is impractical, as it requires rolling out the policy from each state and changes dynamically as the policy explores the environment.

Inspired by quasimetric learning (Wang et al., 2023), we adopt their chain analogy to build intuition for our proximity objective. Consider two objects connected by multiple chains, each consisting of links of fixed length. If the objects are pulled apart, their separation is limited by the shortest chain, revealing the length of the "optimal" chain. Similarly, in our setting there are multiple paths from a state s to the expert distribution through policy transitions. Assuming each transition incurs a fixed temporal cost, the distance from s to the expert distribution is determined by the shortest such path, which we estimate as the proximity measure.

In practice, we learn $p_\theta(s_t)$ by maximizing its value on policy states, pushing them away from the expert distribution, while enforcing local temporal consistency constraints. These constraints ensure that $p_\theta(s_t)$ is upper-bounded by the cumulative cost of any path from s_t to the expert distribution. Specifically, for any transition (s_t, s_{t+1}), the proximity of s_t can be at most one timestep greater than that of s_{t+1}, yielding the triangle inequality constraint $p_\theta(s_t) \leq p_\theta(s_{t+1}) + \zeta$, where ζ is a hyperparameter that defines the cost of one timestep. To ground the function, we anchor all expert states to have a proximity of 0. States farther from the expert distribution increase by ζ per step. The overall objective is given by:

$$\max_\theta \quad \underbrace{\mathbb{E}_{(s_t,s_{t+1})\sim\mathcal{D}_\pi} p_\theta(s_t)}_{\text{maximize proximity of policy states}} \qquad \text{s.t.} \quad \begin{cases} \underbrace{p_\theta(s_t) \leq p_\theta(s_{t+1}) + \zeta}_{\text{temporal consistency}}, & \forall (s_t, s_{t+1}) \in \mathcal{D}_\pi \\ \underbrace{p_\theta(s_e) = 0}_{\text{expert boundary condition}}, & \forall s_e \in \mathcal{D}_{target} \end{cases} \tag{2}$$

We optimize this objective using a Lagrangian relaxation, yielding the following maximization problem, where $(x)^+ = \max(x, 0)$:

$$\max_\theta \ \mathbb{E}_{(s_t,s_{t+1})\sim\mathcal{D}_\pi} \Big[\underbrace{p_\theta(s_t)}_{\text{maximize proximity}} - \underbrace{\alpha \left(p_\theta(s_t) - p_\theta(s_{t+1}) - \zeta\right)^+}_{\text{temporal consistency penalty}} \Big] - \underbrace{\beta\, \mathbb{E}_{s_e\sim\mathcal{D}_{target}} |p_\theta(s_e)|}_{\text{expert boundary penalty}} \tag{3}$$

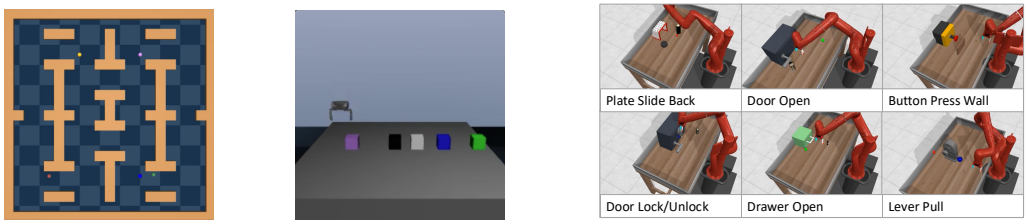
We set fixed Lagrangian multipliers ($\alpha = 100$, $\beta = 5$) and bound the output of p_θ to the range $[0, 1]$ using a sigmoid activation. While the discriminator operates in a multi-task setting, p_θ is trained *only* on target-task demonstrations and policy samples. To leverage information from the multi-task dataset and enrich the expert distribution for the target task, we additionally re-label policy states as expert if the multi-task discriminator's confidence exceeds a threshold, i.e., $d(s_t, a_t) > c_{thresh}$.

4.3 Multitask discriminator Proximity-Guided IRL

Our full reward function $\tilde{R}$, shown in Equation 4, combines the sparse, generalizable signal from the discriminator with the dense, informative guidance from the proximity function. Specifically, the discriminator score $d(s, a)$ is used directly as a reward to encourage expert-consistent behavior. The agent should also be rewarded for transitions that *reduce* its proximity to the expert distribution, so we incorporate the proximity improvement $p(s_t) - p(s_{t+1})$ with weight λ.

$$\tilde{R}(s_t, a_t, s_{t+1}) = \underbrace{d(s_t, a_t)}_{\text{expert behavior signal}} + \lambda \underbrace{\left[p(s_t) - p(s_{t+1})\right]}_{\text{proximity improvement}} \tag{4}$$

With the reward function defined, we summarize the training procedure of MPG in Algorithm 1. At each iteration, the policy collects transitions from the environment, after which we update d_ϕ



(a) Maze2D (b) Block Stacking (c) FactorWorld

Figure 3: We evaluate on navigation and manipulation tasks with diverse task configurations.

and p_θ. The policy π is then optimized using RL with rewards from $\tilde{R}$. Implementation details and hyperparameters are provided in Appendix D.7 and Appendix D.9.

Algorithm 1 MPG

Input: Target-task demos $\mathcal{D}_{target}$, multi-task demos $\mathcal{D}_{multi}$
Initialize policy π, discriminator d_ϕ, proximity function p_θ, replay buffer $\mathcal{D}_\pi$
for $i = 1, 2, \ldots, N$ **do**
    Collect transitions $(s_t, a_t, s_{t+1})_{t=0}^{T}$ by rolling out π in the target-task environment
    Add collected transitions to replay buffer $\mathcal{D}_\pi$
    Update discriminator d_ϕ using Eq. 1
    Update proximity function p_θ using Eq. 3
    Update policy π via RL with reward $\tilde{R}$ from Eq. 4
end for
Output: Trained policy π

5 Experiments

We answer the following questions in our experiments: (1) How effective is MPG compared to other methods that learn from limited demonstrations? (2) How does MPG's performance vary with the number of demonstrations and tasks in the multi-task dataset? (3) How do the components of MPG contribute to its performance?

Environment. We evaluate our methods on several IRL tasks in different environments (Figure 3): *Maze2D* from the D4RL benchmark (Fu et al., 2020): the agent navigates to a fixed goal location, distinguished by color, among four fixed objects. The intra-task variation comes from the agent's random starting position. *Block Stacking* (Pertsch et al., 2021): the agent's goal is to pick up a block of color X and place it on a block of color Y. Differently colored blocks have random initial positions creating intra-task variation. *FactorWorld* from Xie et al. (2024): a multi-task benchmark of manipulation tasks with intra-task variations in object position, table position, distractor objects & positions, and arm position. Each task additionally exhibits distinct dynamics. Further details for all environments are provided in Appendix C.

Baselines. To the best of our knowledge, no prior work directly tackles the FM-IRL setting. So we compare with state-of-the-art methods from closely related settings, augmenting them with additional assumptions where possible for a fair comparison (details in Appendix D). In short, we consider: *BC* clones the target task demonstrations only. *SQIL* (Reddy et al., 2020) is state-of-the-art online IL method. *GAIL* (Ho & Ermon, 2016) is a widely used IRL method. *MT-AIRL* is AIRL (Fu et al., 2018), an adversarial maximum entropy IRL method trained on multi-task demonstrations. To provide a comprehensive comparison, we include a meta-IRL baseline, *PEMIRL* (Yu et al., 2019), which meta-learns a reward function with access to multi-task training environments. *GoalPro* (Lee et al., 2021) learns a goal-proximity reward. *DVD* (Chen et al., 2021) learns a multi-task discriminator only using the demonstrations; we evaluate the discriminator with online RL.

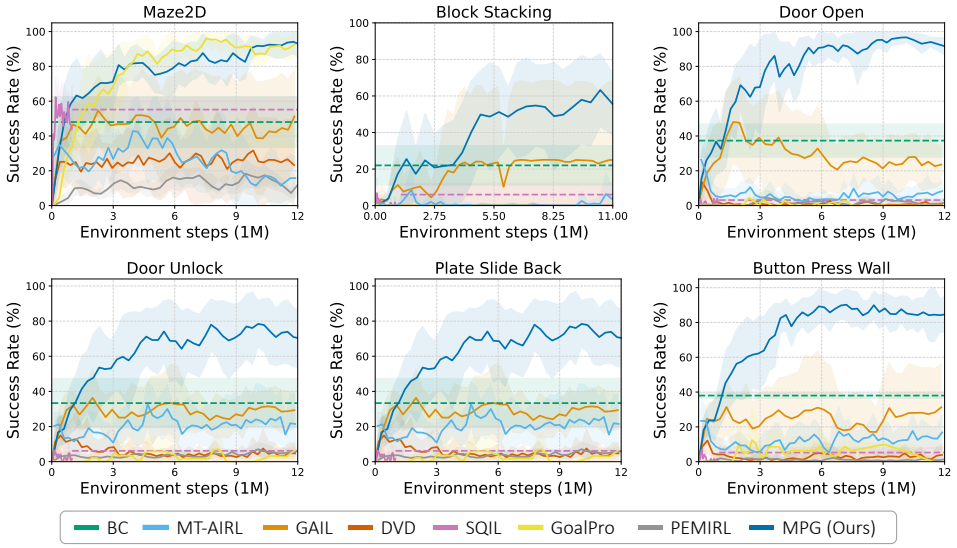


Figure 4: MPG comparison against all baselines. Dashed lines (BC, SQIL) denote performance at convergence.

5.1 Quantitative Results under FM-IRL

As shown in Figure 4, MPG consistently outperforms all baselines and achieves an average **35.4% success rate improvement** over the next best method, across all tasks. More results and details can be found in Appendix B.1.

MPG vs. IRL Baselines. We begin by comparing MPG against IRL methods. Although GAIL has access to the same data resources as MPG (few-shot target demonstrations, multi-task demonstrations, and policy samples), its performance remains poor in our setting (e.g., 20% to 40% success), as it cannot effectively leverage these heterogeneous sources of information. Similarly, MT-AIRL underperforms in our setting (e.g., up to 20% success), suggesting limited robustness to scarce demonstrations and task variation. PEMIRL does not work well (e.g., up to 10% success) due to the high sample cost of meta-training within a fixed training budget, highlighting the inefficiency of meta-IRL when targeting a single task.

MPG vs. IL Baselines. Across all tasks, BC appears to be a strong baseline (e.g., an average 35.9% success), suggesting that under limited demonstrations and substantial variations, inferring a reliable reward function is more challenging than directly learning a policy. In contrast, SQIL learns an RL policy but relies on an extremely sparse reward signal, which leads to poor performance on most tasks (e.g., up to 10% success), except for Maze2D, where the environment configuration is fixed.

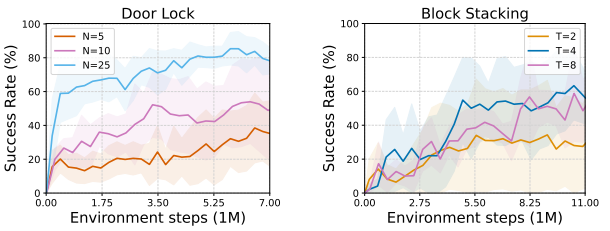
MPG vs. Discriminator/Proximity Baselines. Finally, we compare with prior work that shares a similar objective with parts of our method. GoalPro, which learns a proximity-based reward along *expert trajectories* rather than in *non-expert states*, fails to learn in most tasks (e.g., up to 5% success). It defines a dense reward function based on goal proximity for goal-conditioned tasks, and therefore, it performs well in Maze2D and achieves 90% success, where the goal is fixed. Despite using a multi-task success discriminator, DVD performs poorly (e.g., up to 5% success) due to its pre-trained reward function, which can be exploited by the policy.

5.2 Analysis: Sensitivity to Demonstration Quantity and Task Diversity

Demonstration Quantity. We analyze the effect of demonstration quantity on the Door-Lock task. As shown in Figure 5a, performance improves as the number of target demonstrations increases from 5 to 25. This suggests that additional demonstrations better characterize the expert distribution, enabling improved policy learning. Results on additional tasks are provided in Appendix B.2.

To further understand performance when demonstrations are abundant, we also evaluate a setting with 200 demonstrations. In this regime, MPG continues to perform strongly, achieving an average success rate of 85% across five tasks, while representative baselines—BC, SQIL, and GAIL—reach 70% on average. This indicates that MPG performs well both with limited and ample demonstrations, whereas the baselines require abundant demonstrations to achieve strong performance (see Appendix B.3 for details).

Task Diversity. We analyze the effect of task diversity by varying the number of tasks T in the multi-task dataset for the Block-Stacking task. As shown in Figure 5b, performance improves when T increases from 2 to 4, but plateaus from 4 to 8. This suggests that once a moderate level of task diversity is reached, additional tasks contribute limited new information. Analysis on more tasks can be found in Appendix B.2.

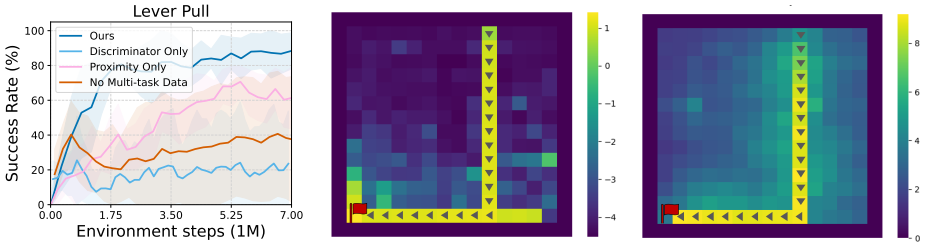


(a) Number of Target Demos (b) Number of Tasks in $\mathcal{D}_{multi}$

Figure 5: Data efficiency analysis. Effect of (a) demonstration quantity and (b) task diversity.

5.3 Ablations on MPG

We first ablate the two parts of our reward function $\tilde{R}(s, a)$ by training policies with either a DISCRIMINATOR ONLY reward or a PROXIMITY ONLY reward. In Figure 6a, while each part individually provides benefits, combining both yields the best performance for MPG. Additionally, removing the multi-task dataset (NO MULTI-TASK DATA) degrades performance, indicating that information from the dataset is important. See Appendix B.4 for additional results and details.



(a) Ablating Reward Terms (b) Discriminator Heatmap (c) Proximity Heatmap

Figure 6: Ablation of MPG's reward components.

We further illustrate the two components in a simple empty Minigrid environment (Chevalier-Boisvert et al., 2023) (Figure 6b, 6c). The red flag indicates the goal and arrows show the expert demonstration. Lighter colors correspond to higher rewards. Appendix C.4 gives more details about this environment. The two parts of our reward function provide complementary and informative learning signal: the discriminator generalizes to certain goal-reaching paths (e.g., directly above the goal), while the proximity provides a smooth gradient in non-expert regions.

Then, we evaluate the standalone contribution of the proximity reward by augmenting GAIL with our proximity function. In 3 out of 5 tasks, this improves GAIL's performance and stability, highlighting the usefulness of the proximity reward for providing guidance in non-expert regions. See details and results in Appendix B.5. Finally, we evaluate MPG under different hyperparameter setting and observe that it remains robust without careful fine-tuning (see Appendix B.4 for details).

6 Conclusion

We introduce a novel problem setting: few-shot IRL with multi-task demonstrations, aiming to learn a new task with diverse variations. To solve it, we propose MPG, a novel method that learns a two-part reward function: (1) a multi-task discriminator that recognizes expert behavior over task variations, and (2) a proximity reward that provides guidance in non-expert states. Finally, we demonstrate the effectiveness of our method across multiple navigation and manipulation environments, improving on the next best baseline by 35.4%; We provide more discussion in Appendix E.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016. URL https://arxiv.org/abs/1606.06565.

Jongseong Chae, Seungyul Han, Whiyoung Jung, Myungsik Cho, Sungho Choi, and Youngchul Sung. Robust imitation learning against variations in environment dynamics. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2828–2852. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/chae22a.html.

Jonathan Daniel Chang, Masatoshi Uehara, Dhruv Sreenivas, Rahul Kidambi, and Wen Sun. Mitigating covariate shift in imitation learning via offline data with partial coverage. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=7PkfLkyLMRM.

Annie S Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*, 2021.

Jiayu Chen, Dipesh Tamboli, Tian Lan, and Vaneet Aggarwal. Multi-task hierarchical adversarial inverse reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

Chia-Cheng Chiang, Li-Cheng Lan, Wei-Fang Sun, Chien Feng, Cho-Jui Hsieh, and Chun-Yi Lee. Expert proximity as surrogate rewards for single demonstration imitation learning. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=gzis9n5r7e.

Robert Dadashi, Leonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=TtYSU29zgR.

Christopher R. Dance, Julien Perez, and Théo Cachet. Demonstration-conditioned reinforcement learning for few-shot imitation. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2376–2387. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/dance21a.html.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

Angelos Filos, Clare Lyle, Yarin Gal, Sergey Levine, Natasha Jaques, and Gregory Farquhar. Psiphi-learning: Reinforcement learning with demonstrations using successor features and inverse temporal difference learning. *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on robot learning*, pp. 357–368. PMLR, 2017.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkHywl-A-.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Adam Gleave and Oliver Habryka. Multi-task maximum entropy inverse reinforcement learning, 2018. URL https://arxiv.org/abs/1805.08882.

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*, 2018.

Kourosh Hakhamaneshi, Ruihan Zhao, Albert Zhan, Pieter Abbeel, and Michael Laskin. Hierarchical few-shot imitation with skill transition models. In *Deep RL Workshop NeurIPS 2021*, 2021. URL https://openreview.net/forum?id=0nW7xnWnam.

Siddhant Haldar, Vaibhav Mathur, Denis Yarats, and Lerrel Pinto. Watch and match: Supercharging imitation with regularized optimal transport. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=ZUtgUA0Fuwd.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Neural Information Processing Systems*, 2016.

Youngwoon Lee, Andrew Szot, Shao-Hua Sun, and Joseph J Lim. Generalizable imitation learning from observation via inferring goal proximity. *Advances in neural information processing systems*, 34:16118–16130, 2021.

Yicheng Luo, zhengyao jiang, Samuel Cohen, Edward Grefenstette, and Marc Peter Deisenroth. Optimal transport for offline imitation learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MhuFzFsrfvH.

Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pp. 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*. PMLR, 2021.

Siddharth Reddy, Anca D. Dragan, and Sergey Levine. {SQIL}: Imitation learning via reinforcement learning with sparse rewards. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1xKd24twB.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Seyed Kamyar Seyed Ghasemipour, Shixiang Shane Gu, and Richard Zemel. Smile: Scalable meta inverse reinforcement learning through context-conditional policies. *Advances in Neural Information Processing Systems*, 32, 2019.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (2nd edition)*. MIT Press, 2018.

Tongzhou Wang, Antonion Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. *ICML*, 2023.

Annie Xie, Avi Singh, Sergey Levine, and Chelsea Finn. Few-shot goal inference for visuomotor learning and planning. *2nd Conference on Robot Learning*, 2018.

Annie Xie, Lisa Lee, Ted Xiao, and Chelsea Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3153–3160. IEEE, 2024.

Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Kelvin Xu, Ellis Ratner, Anca Dragan, Sergey Levine, and Chelsea Finn. Learning a prior over intent via meta-inverse reinforcement learning. In *International conference on machine learning*, pp. 6952–6962. PMLR, 2019.

Lantao Yu, Tianhe Yu, Chelsea Finn, and Stefano Ermon. Meta-inverse reinforcement learning with probabilistic context variables. *Advances in neural information processing systems*, 32, 2019.

Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Code

We have made our code and data available to download here https://drive.google.com/drive/folders/1JbN2GX0005qrPSvRD3IEASOa9o5S3SMs?usp=sharing.

B Additional Results

B.1 Performance Comparison on More FactorWorld Tasks

Figure 7 contains comparison results for additional FactorWorld tasks that did not fit into the main paper. We plot the average and standard deviation (in shaded regions) over 5 seeds per method and roll out 10 episodes per evaluation. Our method out-performs the baseline methods in every task and displays similar trends as those discussed in Section 5.1. For GAIL, we evaluate two variants in which the multi-task dataset is treated as either negative or positive examples when training the discriminator, but neither variant achieves competitive performance. Here we only show the result of GAIL with multi-task dataset as negative examples. In Maze2D, SQIL achieves over 50% success, because variations of this task arise primarily from the agent's trajectory, where the environment configuration is fixed and the demonstrated trajectories can be revisited.

Figure 8 compares our method to an "oracle" BC policy trained with 2000 target-task demonstrations, serving as an approximate upper bound on imitation performance. This oracle is imperfect, as BC remains susceptible to compounding errors and covariate shift. Moreover, many of these tasks are very difficult RL problems (e.g., long-horizon maze navigation with narrow corridors, block stacking from random initial states), where training RL from scratch without demonstrations fails to reach oracle-level performance.

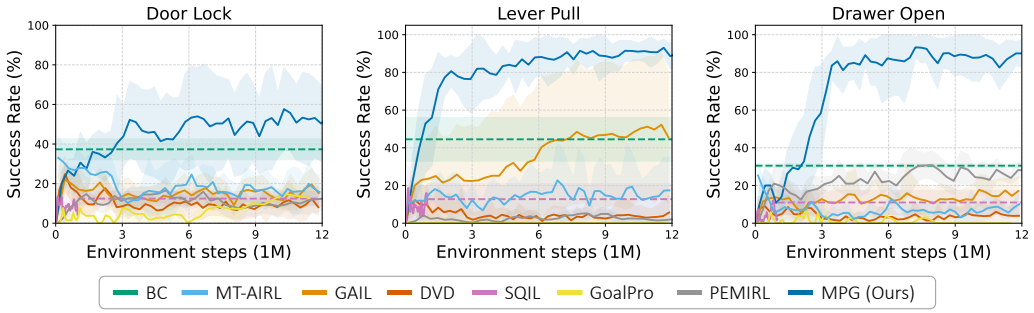


Figure 7: Remaining FactorWorld tasks that did not fit into the main paper. See Figure 4 for other tasks and experiment description.

B.2 Sensitivity to Demonstration Quantity and Task Diversity on More Tasks

For analysis and ablations, we evaluate on Maze, Block Stacking, and 3 out of 7 representative FactorWorld tasks. Figure 9 and Figure 10 report additional experiments varying (i) the number of target task demos and (ii) the number of tasks in the multi-task demo dataset.

Demonstration Quantity. Overall we see a similar trend as discussed in Section 5.2, where performance increases with the number of target demonstrations until some saturation level. For example, in Button Press Wall, MPG is able to achieve 80% success with only 5 demonstrations. For tasks that saturate earlier, such as Block Stacking and Plate Slide Back, it is probably that MPG saturates

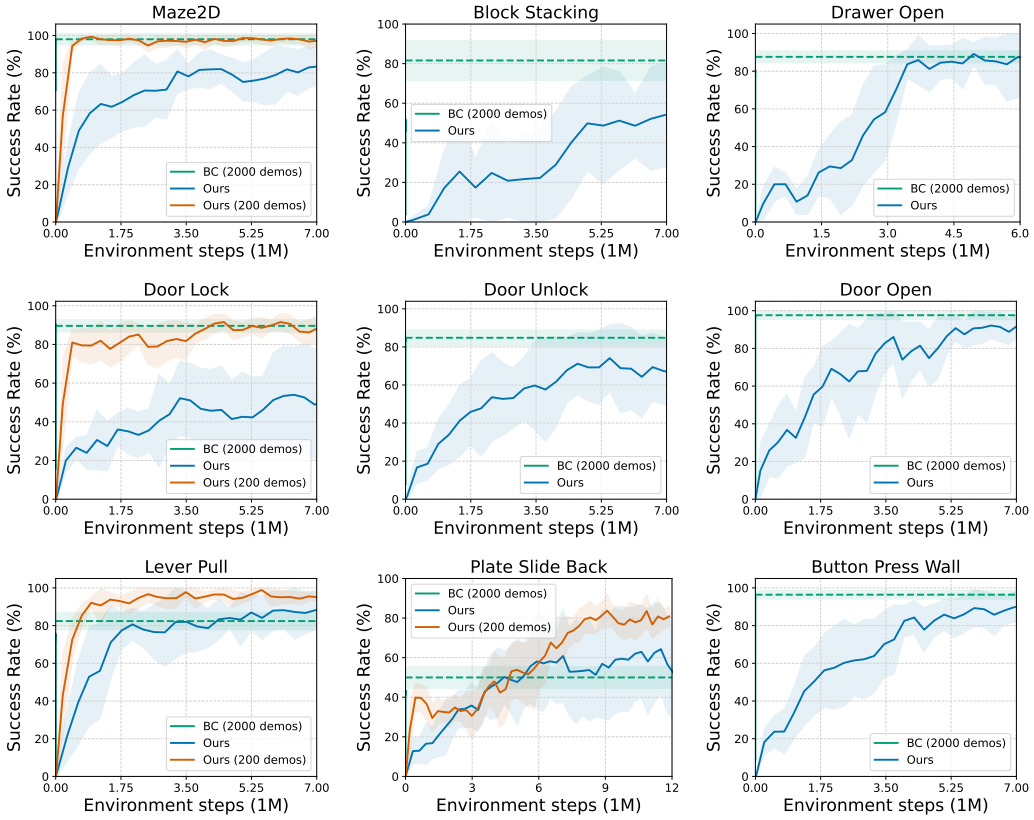


Figure 8: Comparisons with an "oracle" BC method given 2000 demonstrations.

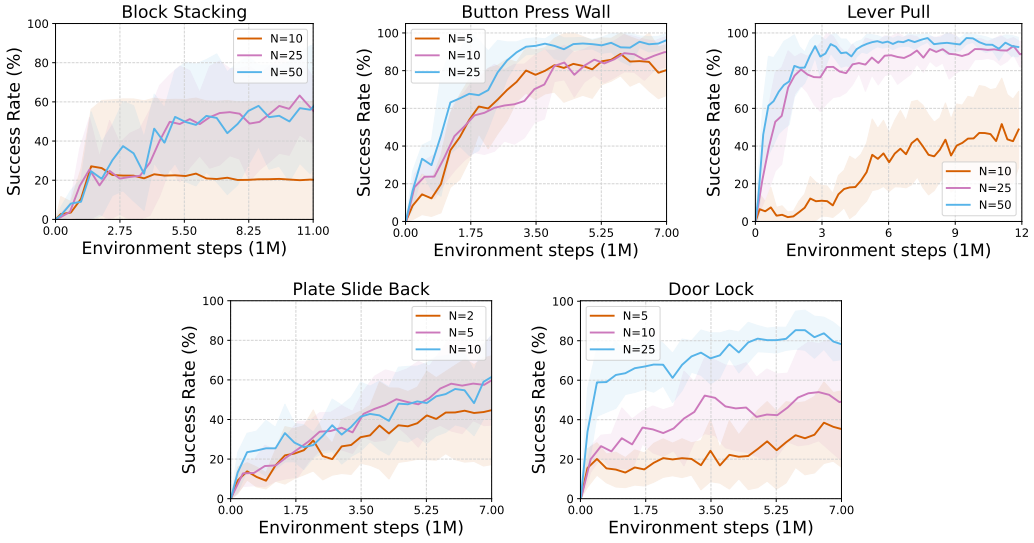


Figure 9: Analysis on the number of target task demonstrations in more tasks.

at this level and the remaining 40% performance requires a more sophisticated reward function or RL algorithm.

Task Diversity. Increasing the number of tasks T in the multi-task dataset sometimes yields negligible improvement (e.g., Lever-Pull), whereas in other tasks (e.g., Block-Stacking), performance

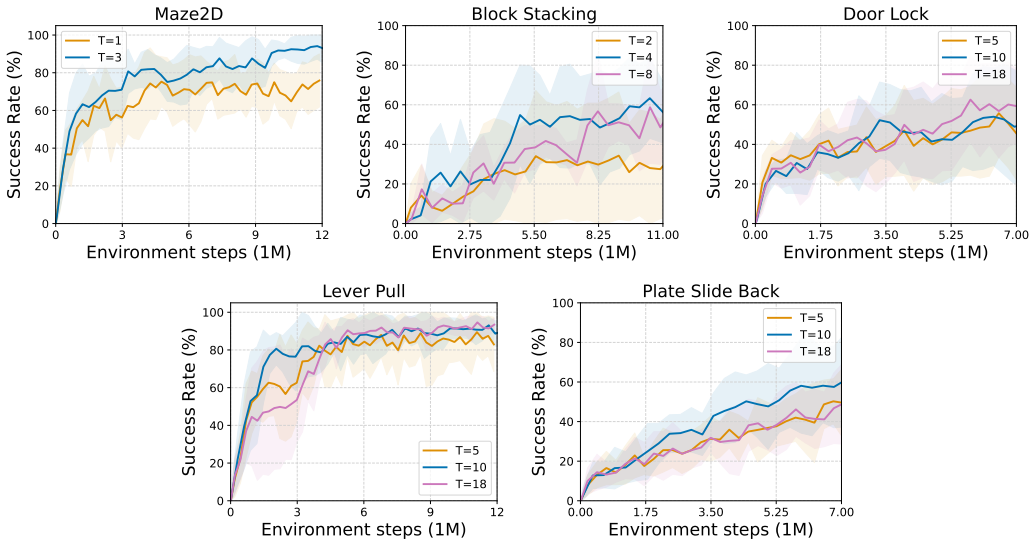


Figure 10: Analysis on the number of tasks in the multi-task dataset in more tasks.

improves when increasing T from 2 to 4. This suggests that once a minimum level of task diversity is reached, performance saturates.

B.3 Performance with Ample Target-task Demonstrations

In Figure 11, MPG performs competitively with traditional IL and IRL methods, confirming that MPG's reward function remains sound when sufficient task-specific data is available. As expected, however, MPG no longer exhibits a clear advantage, since dense demonstrations sufficiently cover the expert trajectory distribution. These results further demonstrate that GAIL and SQIL are strong IRL baselines, and their degradation in our main experiments arises from limited demonstrations and intra-task variations rather than inherent methodological weakness.

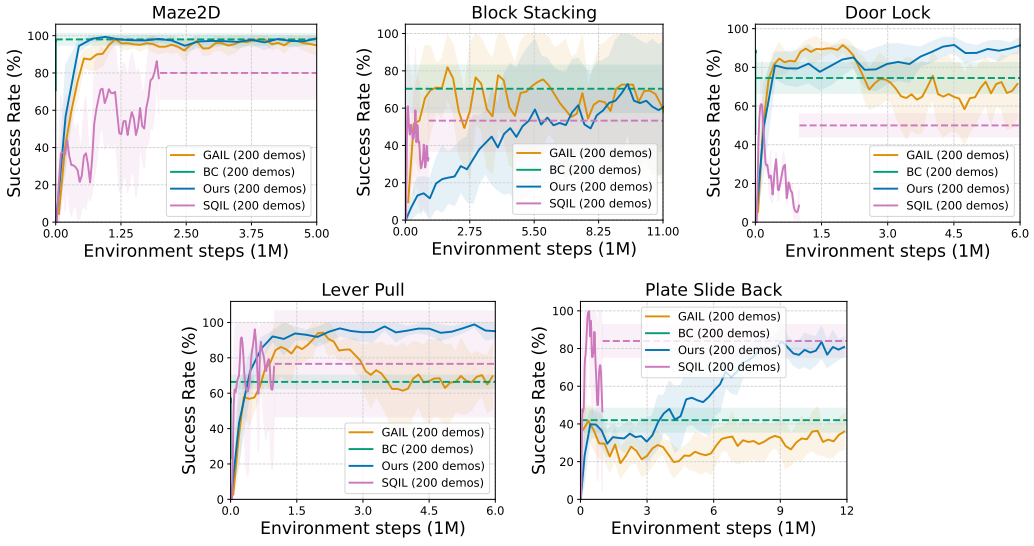


Figure 11: Comparison of MPG with traditional IL and IRL approaches in a standard IL setting with ample demonstrations.

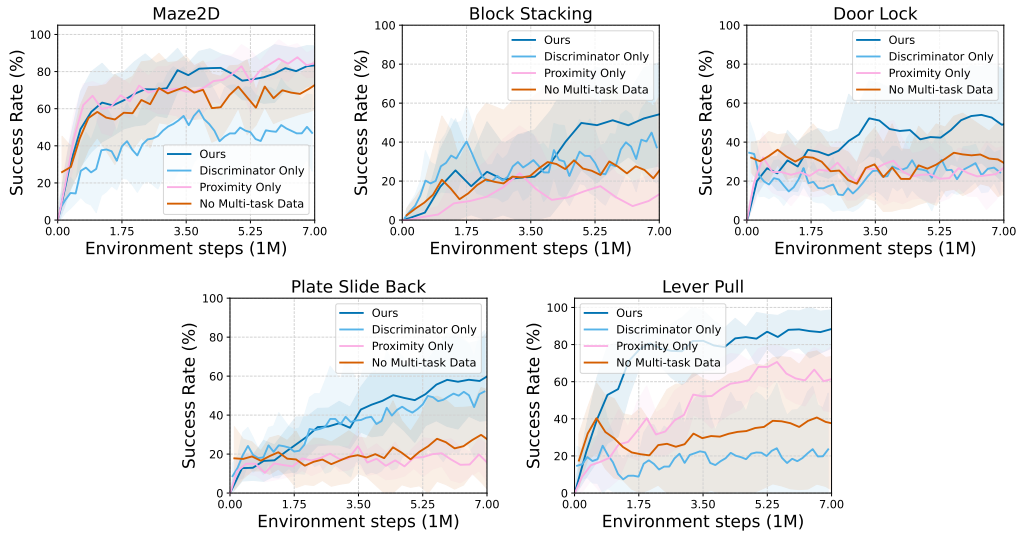


Figure 12: Ablations over more tasks in supplement to Figure 6

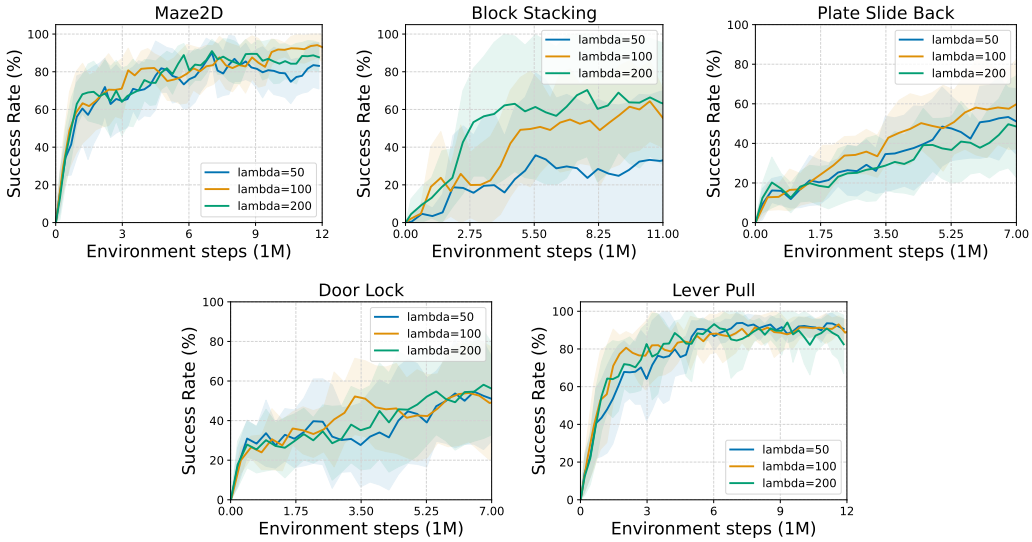


Figure 13: Analysis on lambda λ, the coefficient of the proximity reward, in more tasks.

B.4 Ablation Results on More Tasks

Reward Components. Figure 12 contains results ablating the two components of MPG's reward function: the multi-task discriminator and proximity function. Across additional tasks, we see the same trend that combining both components consistently outperforms either component alone.

Multi-Task Dataset. We also evaluate our method with and without the multi-task demonstrations(Figure 12). To implement, we train the multi-task discriminator without the multi-task dataset, thus the positive example is only the pairs from the target task. We see that the "No Multi-task Data" yields performance comparable to the "Proximity Only" ablation in most environments, indicating that the discriminator's primary benefit stems from leveraging multi-task demonstrations.

Hyperparameters. We further ablate the proximity reward weight λ (Figure 13) and timestep factor ζ (Figure 14). Performance remains stable across a broad range of values for both hyperparameters, suggesting that MPG does not require careful fine-tuning. In practice, robust performance

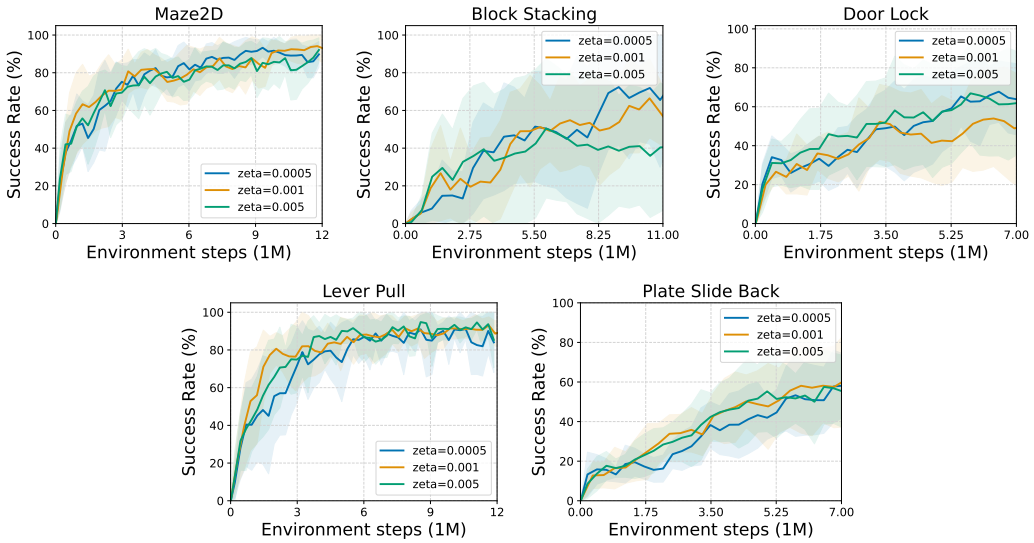


Figure 14: Analysis on zeta ζ, the proximity timestep factor, in more tasks.

is achieved when λ is on the same order as the discriminator reward and ζ is scaled relative to the inverse of the maximum episode length. Figure 13 contains results ablating the proximity reward coefficient λ for additional tasks. We see the same trend here that performance is relatively robust to a range of λ's within one order of magnitude, with the exception being Block Stacking at $\lambda = 50$.

Figure 14 demonstrates similar robustness to ζ across tasks. These findings further support the hyperparameter stability discussed in Section D.9.

B.5 Effect of the Proximity Reward

We analyze the contribution of the proximity component in MPG's reward and its interaction with the multi-task discriminator. The proximity function estimates the number of steps away a state is from an expert state, providing informative feedback in non-expert regions. Theoretically, this same reward can be combined with traditional IRL methods that do not use multi-task data.

To evaluate its standalone benefit, we augment GAIL with the proximity reward (Figure 15). In 3 out of 5 tasks, this improves GAIL's performance and stability, with gains of up to 20% in Plate Slide Back, supporting the usefulness and soundness of the proximity reward on its own.

For the FactorWorld tasks, 200 target demonstrations were required for "GAIL + Proximity" to yield consistent improvements. For Lever Pull and Door Lock, we additionally tuned the re-labeling hyperparameter c_{thresh} (0.4 and 0.6, respectively). These changes improve discriminator generalizable and increase re-labeled expert states. Across all tasks, MPG outperforms "GAIL + Proximity" , validating the benefit of the multi-task discriminator, even in data-rich settings.

C Environment Details

C.1 Maze2D

We base our implementation on the Maze environment from the D4RL benchmark Fu et al. (2020). As show in Figure 3a, there are four balls placed in fixed locations, resulting in four tasks. The starting positions of the agent are randomly sampled. The state space is the agent's position, velocity, and positions of four balls, and then outputs an x- and y-velocity to navigate in the maze. Episodes have a horizon of 1500 timesteps. For the target task we use two demonstrations, and for the multi-

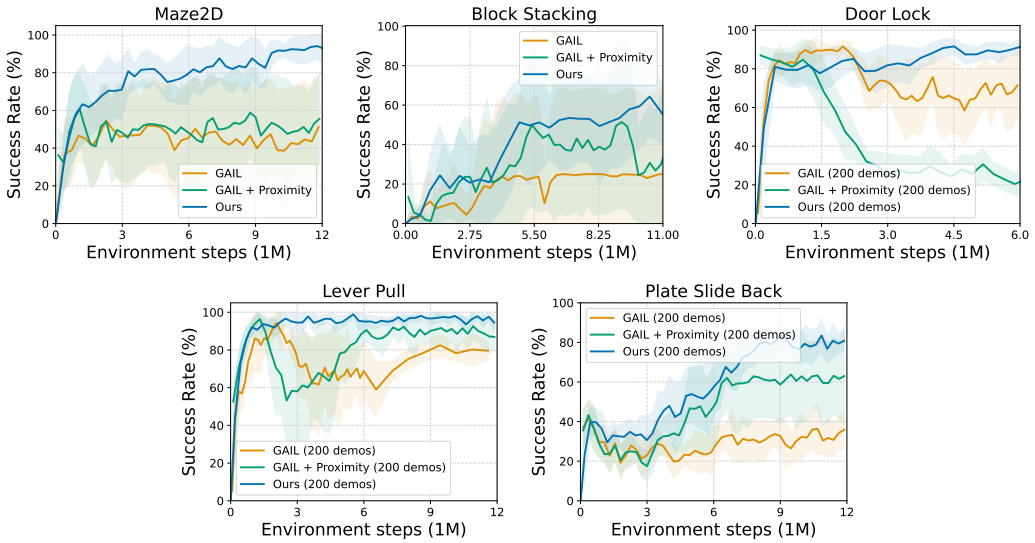


Figure 15: Analysis of our proximity reward by combining it with GAIL without multi-task data (GAIL + Proximity).

task dataset we use 200 demonstrations for each of the remaining three tasks, all gathered by a planner-based policy provided in Pertsch et al. (2021).

C.2 Block Stacking

We use the implementation from Pertsch et al. (2021), there are five blocks on the ground with five different colors. The five block starting positions are randomly generated. In each task, the agent aims to pick up a block with color X and place it on a block with color Y (X and Y are two different colors selected from five colors). Different tasks have different pick-place colors. The state space contains the gripper's position, opening angle, velocity, and the position of the gripper fingers. It also includes the position and orientation of the block in quaternions. The action space consists of an (x, z)-displacement and a continuous action representing the degree of the robot gripper's opening. We collect 200 demonstrations for each task using a planner from Pertsch et al. (2021) and use 25 demonstrations for the target task. The target task is to stack the purple block on top of the blue block. The three tasks in the multi-task demonstration dataset are: purple on top of green, black on top of blue, and green on top of white. Episodes have a horizon of 500 timesteps.

This task is very unforgiving: dropping a block prematurely drives the policy out of distribution, from which recovery is difficult. We hypothesize that our proximity reward mitigates such errors by penalizing those transitions more than other less harmful non-expert behaviors.

C.3 FactorWorld

We utilize the implementation provided by Xie et al. (2024), which extends the Meta-World benchmark (Yu et al., 2020) by introducing various factors of variations. In our experiments, we incorporate variations in object position, table position, and arm position, and include distractor objects with diverse initial positions and shapes. The agent observes in state space, the 3D position of its end effector, how open its gripper is, the 3D positions of the one or two objects on the tabletop, table position, the goal position, and its previous state. The action space is the end effector position delta along with the normalized torque input to the gripper. We evaluate performance on seven tasks from the benchmark, using between 2 and 25 demonstrations for each task (Table 2). Since these tasks vary by difficulty, what is considered too few demonstrations varies. Additionally, we leverage an offline dataset consisting of 10 tasks randomly selected from the following set of 18 tasks, none of which are target tasks: reach, push, pick-place, dial-turn, drawer-close, button-press, peg-insert-side, window-

open, sweep-into, basketball, door-close, faucet-open, hammer, handle-press-side, pick-out-of-hole, plate-slide, plate-slide-side, handle-pull. Each of these tasks has 200 demonstrations, collected by Meta-World's open-source hard-coded policies. The maximum number of timesteps per episode is capped at 500.

Table 2: FactorWorld Number of Target Demos

Task	Drawer Open	Door Lock	Door Unlock	Plate Slide Back	Door Open	Lever Pull	Button Press Wall
# Demos	5	10	5	5	2	25	10

C.4 Minigrid

There are four tasks, each requiring the agent to reach a different corner of the room, with the target task being to reach the bottom-left corner. The agent's start position is randomly initialized. The environment's discrete action space includes movement directions: up, down, left, and right. The state space includes the maze layout, agent position, and direction. A single expert demonstration is provided for the target task. We provide 200 demonstrations for each of the remaining three tasks For interpretability, we consider a state-only discriminator and define the proximity function such that all expert states have proximity value 1, yielding high proximity within the expert distribution and progressively lower values as the agent deviates.

D Implementation Details

We use the robot learning code base from https://github.com/youngwoon/robot-learning for basic RL and imitation learning baselines and use default hyperparameters unless otherwise specified. All online methods use PPO (Schulman et al., 2017) as the RL algorithm except SQIL which uses the off-policy algorithm SAC (Haarnoja et al., 2018). For all methods, we initialize the policy with a BC trained policy and add an auxiliary BC loss to the policy loss function using Equation 5. We do not use BC to initialize PEMIRL since it infers and conditions on a context variable that cannot be pretrained with BC. We detail our own implementations of each method below.

$$L_{MSE} = \mathbb{E}_{(s,a)\sim\mathcal{D}_{target}} \|a - \pi(s)\|^2 \tag{5}$$

D.1 SQIL

We implement SQIL using the resources from Reddy et al. (2020) and use SAC as the off-policy RL algorithm. It gives sparse rewards (i.e., $+1$ only to transitions inside expert demonstrations and 0 elsewhere). To incorporate the other task data, we add it to the training data with labeled rewards of 0. For each batch of training data, we sample 50% from target task demonstrations, 40% from the policy replay buffer, and 10% from the multi-task demonstrations. This addition can provide better coverage of the environmnet especially early on in training.

We run SQIL until convergence, which often happened more quickly than the other methods because SAC tends to be more sample efficient than PPO. SQIL requires an off-policy RL algorithm. While our method could also use SAC, in practice, we found the generative adversarial training for the multi-task discriminator to be more stable with PPO.

D.2 GAIL

We train GAIL by treating the multi-task demonstrations as additional negative examples for the discriminator, in addition to the standard online policy samples.

D.3 MT-AIRL

We implement MT-AIRL by training a multi-task demonstration-conditioned discriminator using the same network architecture and training procedure as our multi-task discriminator. We train MT-AIRL on $\mathcal{D}_{target} \cup \mathcal{D}_{multi}$ using demonstration trajectories and expert state-action tuples from the same task as positive classification examples and state-action tuples from different tasks as negative examples.

D.4 DVD

We implement DVD and adapt the video-discriminator from a non-Markovian reward function to a state-action based reward function. Specifically, we input a demonstration trajectory including actions, and state-action tuple, and predict whether or not that state-action tuple exhibits expert behavior for the demonstrated task. Similar to our multi-task discriminator, we train DVD on $\mathcal{D}_{target} \cup \mathcal{D}_{multi}$ using trajectory and state-action tuples from the same task as positive samples and trajectory and state-action tuples from different tasks as negative examples. We train DVD for 200 gradient steps using batch size of 128 and learning rate of 1e-3 then use it as a reward function to train a policy with online RL.

D.5 PEMIRL

We use the implementation from Chen et al. (2023), and made the following changes to accommodate our problem setting. We ensure that sampling between demonstrations of different tasks are balanced so the target task gets sufficient training. Since PEMIRL learns a single policy over all tasks in the meta-training set, we use the same network architectures as the other methods but with double the width (512 hidden dimensions), in order to accommodate the higher capacity. In the Factorworld tasks, we removed pick-out-of-hole from the list of tasks that can be sampled for the multi-task demonstration set because training in the pick-out-of-hole environment frequently caused unstable simulation and training. To clarify, agents still received the same number of tasks and demonstrations, but the multi-task demonstrations were sampled out of 17 instead of 18 like the other comparison methods.

D.6 GoalPro

We use the implementation from Lee et al. (2021), with PPO as the RL algorithm. The limited target demonstrations are used by GoalPro to learn its reward function. We do not provide the additional multi-task demonstrations, as there is no straightforward way to assign goal-proximity labels for the target task.

D.7 MPG

Before starting online training. We pre-train p_θ only on the demonstration data by treating $\mathcal{D}_{target}$ as expert states and $\mathcal{D}_{multi}$ as non-expert states and optimizing the same objective Eq. 3.

During online training, we alternative between updating the policy π, multi-task discriminator d_ϕ, and proximity function p_θ, training the policy and multi-task discriminator adversarially while updating the proximity function with current policy samples. Initially, we collect 2000 steps of policy data from the environment, storing it in two separate buffers: the policy replay buffer $\mathcal{D}_\pi$ (for policy data with predicted rewards $\tilde{R}$) and the proximity dataset $\mathcal{D}_{prox}$. In our implementation, to facilitate balanced sampling of expert states, we maintain two separate buffers for expert and non-expert states that together make up $\mathcal{D}_{prox}$. Policy samples re-labeled as expert by the multi-task discriminator ($d(s, a) > c_{thresh}$) are added to the expert buffer, along with the target task demonstrations from $\mathcal{D}_{target}$. When training $p(s)$, we sample two minibatches of non-expert states (one for the maximization objective and one for the triangle inequality constraint) and one minibatch of expert states (for the expert proximity anchoring). We found that this improves sample efficiency. We use

a sigmoid output activation to cap our proximity values between 0 and 1, which 0 being closest to expert states and 1 being the furthest. The policy can be trained with any RL algorithm to optimize this combined reward.

D.8 General Hyperparameters

For all environments, we use a learning rate of 3e-4 for policy and 1e-3 for the reward function. We use PPO with a clip ratio of 0.2 and a batch size of 128. The proximity function is a feedforward network with 2 hidden layers of dimension 256 and tanh activation. The multi-task discriminator has the same architecture with an added lstm (2 layers, hidden dimension 128) to encode the demonstration trajectory, which is concatenated with the state-action tuple. The RL policy and critic are feedforward networks with 2 hidden layers of dimension 256 and relu activation.

D.9 MPG Hyperparameters

MPG has two main hyperparameters: the proximity reward weight λ, the cost of each timestep ζ. While both can be tuned for optimal performance, we provide the best practices for selecting good values based on the environment, without the need for extensive tuning. The hyperparameters used in our experiments are summarized in Table 3.

Selecting λ. We selected $\lambda = 100$ for all our experiments to match the proximity reward, $\lambda[p(s_t) - p(s_{t+1})]$ to the observed magnitude of the discriminator rewards. As we see in Figure 13, MPG is robust to a range of λ's within the same order of magnitude, so this hyperparameter is not very sensitive and can be selected without tuning.

Selecting ζ. ζ affects how quickly the proximity reward decreases as states get further away from the expert state distribution. A general rule of thumb is to choose ζ on the order of 1/Maximum episode length, and we found that the results with $\zeta = 0.001$ works well for all of our tasks with an episode length between 500 and 1500. This provides informative differences between states while allowing the model to cover a broad range of proximities. Ablations for ζ are shown in Figure 14. We see a performance drop-off in some tasks as ζ becomes too small, but performance is generally stable between $\zeta = 0.0005$ and $\zeta = 0.005$. When ζ is too small, there is very little change in $p_\theta(s_t)$, which can lead to higher variance rewards due to noise or errors in p_θ.

Tuning c_{thresh}. In practice, a fixed value of c_{thresh} in the range of 0.8-0.9 is generally sufficient.

Table 3: MPG hyperparameters.

Hyperparameter	Maze2D	Block Stacking	FactorWorld
Proximity Reward Weight λ	100	100	100
Proximity Timestep Factor ζ	0.001	0.001	0.001
$d(s, a)$ threshold c_{thresh}	0.9	0.9	0.8
Number of pretraining epochs	5	100	5
Maximum Episode Length	1500	500	500

E Limitations & Future Work

MPG requires a structured multi-task demonstration dataset from the same domain and agent. We hope that incorporating large pre-trained language and vision models can ease these assumptions in the future. In addition, while MPG is efficient in the number of demonstrations required, it is still fairly sample hungry when it comes to online training. Combining our reward function with more sophisticated pre-trained behavior models is a promising direction. Finally, since our reward function is trained online with the policy, it cannot be reused to train a new policy from scratch.