Robust Imitation Learning against Variations in Environment Dynamics

Jongseong Chae [1] **Seungyul Han** [2] **Whiyoung Jung** [1] **Myungsik Cho** [1] **Sungho Choi** [1] **Youngchul Sung** [1]

Abstract

In this paper, we propose a robust imitation learning (IL) framework that improves the robustness of IL when environment dynamics are perturbed. The existing IL framework trained in a single environment can catastrophically fail with perturbations in environment dynamics because it does not capture the situation that underlying environment dynamics can be changed. Our framework effectively deals with environments with varying dynamics by imitating multiple experts in sampled environment dynamics to enhance the robustness in general variations in environment dynamics. In order to robustly imitate the multiple sample experts, we minimize the risk with respect to the Jensen-Shannon divergence between the agent's policy and each of the sample experts. Numerical results show that our algorithm significantly improves robustness against dynamics perturbations compared to conventional IL baselines.

1. Introduction

Reinforcement Learning (RL) is a framework that produces optimal policies for tasks. Deep neural networks enable RL to handle complex tasks in various simulation environments (Mnih et al., 2015; Lillicrap et al., 2015; Fujimoto et al., 2018; Haarnoja et al., 2018; Schulman et al., 2015a; 2017). However, current RL still has limitations for deployment into the real world. Two of the main limitations are *robustness* and *design of reward function*. A typical RL algorithm interacts with a single environment and evaluates the policy with the interaction environment, so the policy becomes specialized to the trained environment and mostly fails when the underlying dynamics are perturbed from the trained environment. In the real world, underlying dynamics are highly likely to be perturbed. For example, consider autonomous driving with RL. The physical dynamics of an autonomous

---

[1]School of Electrical Engineering, KAIST, Daejeon, South Korea. [2]Artificial Intelligence Graduate School, UNIST, Ulsan, South Korea. Correspondence to: Seungyul Han <syhan@unist.ac.kr>.

driving car including handling, braking, the road friction coefficient of a rainy day change from those of a clear day. To cope with such uncertainty, one could consider learning an expert policy for each of all possible environment dynamics for a given task like car driving, estimating the realized dynamics, and using one of the learned expert policies for the estimated dynamics. However, estimating the realized environment dynamics is difficult because the dynamics of the environment depend on many correlated environment parameters such as gravity, mass, aging, etc. Furthermore, learning a policy for each of all possible dynamics perturbations is infeasible when the dynamics vary continuously.

Robust Reinforcement Learning (Robust RL) is a framework that produces a robust policy against such environment perturbations for a given task. The aim is to learn a policy that works well in all possible dynamics perturbations without estimating the perturbation. Typical robust RL allows the agent to interact in multiple environments (Derman et al., 2018; Mankowitz et al., 2018; 2019) and the policy optimizes the worst case of the expected returns in the multiple interaction environments. This agent can work well both in all the interaction environments and even in an unseen environment with similar dynamics. Even if such dynamics variation can be handled by robust RL, there still remains the issue of reward function design for many real-world control problems including our example of autonomous driving, since robust RL relies on a well-designed reward function. When we observe a human drive, it is difficult to know what reward the driver has for each of the driver's actions.

Imitation Learning (IL) has been developed to cope with such situations by learning a policy for a given task without a reward function (Torabi et al., 2018a; Finn et al., 2016; Syed et al., 2008). IL uses demonstrations generated from an expert for the task instead of a reward function, and the agent tries to mimic the expert. GAIL is one of the popular IL algorithms and tries to mimic an expert by matching occupancy measure, which is the unnormalized distribution of state-action pairs (Ho & Ermon, 2016). Up to now, however, most IL algorithms have been proposed for a single interaction environment with perfect or non-perfect expert demonstration to yield a policy that is specialized to the single interaction environment.

In this paper, we consider robust IL to learn a robust policy

by IL against continuous environment dynamics perturbation and propose a novel IL framework to learn a robust policy performing well over a range of continuous dynamics variation based on demonstrations only at a few sampled dynamics from the continuum, which does not require demonstrations from all the continuum and thus significantly reduces the amount of required demonstrations. The detail of the proposed framework will follow in the upcoming sections.

2. Related Works

Imitation Learning: IL aims to learn a policy by imitating an expert. Behavior Cloning (BC) (Torabi et al., 2018a) is an approach of IL based on supervised learning. Brantley et al. (2019) alleviated the covariate shift problem of BC. Another approach is adversarial imitation learning (Ho & Ermon, 2016; Torabi et al., 2018b) in which the agent imitates an expert by matching a positive measure. Fu et al. (2017) recovered the reward function using expert demonstration. Cross-domain IL (Gangwani & Peng, 2020; Liu et al., 2019; Viano et al., 2021; Raychaudhuri et al., 2021) considered the IL problem under dynamics mismatch between the expert and the learner.

The existing robust IL works addressed the IL problem with non-perfect demonstrations (Wu et al., 2019; Tangkaratt et al., 2020) or improved the stability of IL (Wang et al., 2017; Laskey et al., 2017), and their settings are different from our setting in this paper. Meta-IL (Duan et al., 2017; Finn et al., 2017; James et al., 2018; Zhou et al., 2019) and Meta-IRL (Xu et al., 2019; Yu et al., 2019) can learn a new task using a few demonstrations by leveraging experiences from similar tasks, whereas our framework doesn't require any demonstrations for test tasks. Multi-task IRL (Gleave & Habryka, 2018) proposed a Maximum Causal Entropy IRL framework for multi-task IRL and meta-learning to infer multiple reward functions for each task. Toyer et al. (2020) proposed a multi-task benchmark suite for evaluating the robustness of IL algorithms. ADAIL (Lu & Tompson, 2020) can learn an adaptive policy for environments of varying dynamics, but it assumed that collecting expert demonstrations in multiple environments is infeasible and used many simulation environments for domain randomization and environment encoding.

Robust Reinforcement Learning: Robust RL produces a robust policy over environment perturbations. Robust-MDP (Iyengar, 2005; Wiesemann et al., 2013) extends uncertainty transition set on MDP. Derman et al. (2018); Mankowitz et al. (2018; 2019) estimated the worst case of the expected return among multiple perturbed environments. Pinto et al. (2017) addressed the Robust RL problem by using the adversary. Tessler et al. (2019); Vinitsky et al. (2020) formalized criteria of robustness to action uncertainty.

3. Preliminaries

3.1. Markov Decision Process

An MDP is denoted by a tuple $\mathcal{M} = <\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma>$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ is the transition probability, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. A policy π is a (stochastic) mapping $\pi : \mathcal{S} \mapsto \mathcal{A}$. The return G_t is a discounted cumulative sum reward from time step t, i.e., $G_t = \sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i)$. The goal is to learn a policy π to maximize the expected return $J(\pi) = \mathbb{E}_{s_0 \sim \mu_0, \tau \sim \pi}[G_0]$ (Sutton & Barto, 2018), where $\tau = \{s_0, a_0, s_1, a_1, \ldots\}$ is an episode trajectory and $\mu_0(s)$ denotes the initial state distribution. The occupancy measure $\rho_\pi(s, a) = \pi(a|s) \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s|\pi, \mathcal{P})$ is the unnormalized state-action distribution induced by policy π, and $\mu_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s|\pi, \mathcal{P})$ is the unnormalized state distribution induced by policy π.
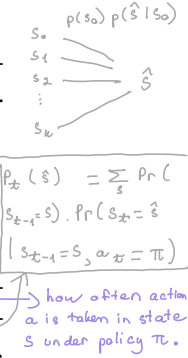
3.2. Generative Adversarial Imitation Learning

In IL, the agent does not receive a reward for its action. Instead, the agent learns a policy based on the demonstration of an expert without knowing the explicit expert policy. Typically, expert demonstration is given as a trajectory generated by the expert's policy, $\tau_E = \{s_0, a_0, s_1, a_1, \ldots\}$. Generative adversarial imitation learning (GAIL) (Ho & Ermon, 2016) is one of the popular IL methods using expert demonstration. Based on Proposition 3.1, GAIL seeks a policy of which occupancy measure is close to that of the expert so that the agent's policy π is close to the expert's policy π_E.

Proposition 3.1 (Theorem 2 of Syed et al. (2008) & Proposition 3.1 of Ho & Ermon (2016))**.** *In a single environment, the occupancy measure $\rho_\pi(s, a)$ satisfies the following Bellman flow constraint for each $(s, a) \in \mathcal{S} \times \mathcal{A}$:*

$$\rho_\pi(s, a) = \mu_0(s)\pi(a|s) + \gamma \int_{(s',a')} \mathcal{P}(s|s', a')\rho_\pi(s', a')\pi(a|s) \quad (1)$$

*and the policy π whose occupancy measure is ρ_π is unique. That is, the occupancy measure and the policy are in an **one-to-one** relationship.*

The policy π induces the occupancy measure ρ_π, and ρ_π maps to the unique policy π. Therefore, GAIL reproduces the expert's policy from the policy update (2), which matches the occupancy measures of the agent's policy and the expert's policy:

$$\min_\pi \mathcal{D}_{JS}(\bar{\rho}_\pi, \bar{\rho}_{\pi_E}) \quad (2)$$

$$\overset{(a)}{=} \min_\pi \mathbb{E}_{\rho_{\pi_E}}\left[\log \frac{\rho_{\pi_E}}{\rho_\pi + \rho_{\pi_E}}\right] + \mathbb{E}_{\rho_\pi}\left[\log \frac{\rho_\pi}{\rho_\pi + \rho_{\pi_E}}\right]$$

$$\overset{(b)}{=} \min_\pi \max_D \mathbb{E}_{\rho_{\pi_E}}[\log D(s, a)] + \mathbb{E}_{\rho_\pi}[\log(1 - D(s, a))]$$

where $\mathcal{D}_{JS}$ denotes the Jensen-Shannon (JS) divergence, and $\bar{\rho}_\pi$ and $\bar{\rho}_{\pi_E}$ are the normalized occupancy distributions from ρ_π and ρ_{π_E}, respectively. Here, (a) is valid since

the constant normalizer is irrelevant in minimization, and (b) is valid because the maximizing D value is given by $D(s, a) = \frac{\rho_{\pi_E}(s,a)}{\rho_\pi(s,a) + \rho_{\pi_E}(s,a)}$, where discriminator D distinguishes whether a given pair (s, a) is from expert or not.

Gradient Penalty: A variant of GAIL (Kostrikov et al., 2018) uses the gradient penalty (GP) proposed by Gulrajani et al. (2017) as a regularization term to enhance the stability of IL. The discriminator update of GAIL with GP is given by

$$\max_D \mathbb{E}_{\rho_{\pi_E}}\left[\log D(s, a)\right] + \mathbb{E}_{\rho_\pi}\left[\log(1 - D(s, a))\right] \quad (3)$$
$$+ \kappa \mathbb{E}_{\hat{x}} \left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2,$$

where $\hat{x} \sim (\epsilon \rho_\pi + (1 - \epsilon)\rho_{\pi_E})$ with $\epsilon \sim \text{Uniform}[0, 1]$, and κ is the regularization coefficient to control the GP term. We will call this GAIL+GP.

4. Motivation

The existing IL methods typically interact with a single nominal environment and try to imitate an expert that is specialized at the single nominal environment. For further discussion, we define three types of environment: the *interaction* environment, the *demonstration* environment and the *test* environment. The interaction environment is the one with which the agent interacts to obtain policy samples during the training, the demonstration environment is the one from which the expert demonstration is generated to train the agent, and the test environment is the actual test environment for the trained agent policy. The interaction environment and the demonstration environment are the same for conventional IL with a single nominal environment (SNE). We will refer to this IL training setting as SNE/SNE (interaction environment / demonstration environment). In most cases, IL trained in the SNE/SNE setting fails when the actual test environment dynamics are perturbed from the nominal dynamics, as seen in Figures 1a and 1b. In Figures 1a and 1b, the x-axis value denotes the ratio (in percentage) of the gravity (or mass) of the test environment to that of the nominal interaction/demonstration environment and the y-axis shows the mean return of the policy trained under the SNE/SNE setting at the corresponding x value. It is seen that the performance degrades severely as the test environment dynamics deviate from the nominal interaction/demonstration environment dynamics.

To handle such performance degradation, robust RL samples a few environments with perturbed dynamics. For example, in Figure 1c, three environments with gravity 50%, 150% and nominal 100% are sampled. Then, robust RL allows the agent to interact with these multiple sampled environments (MPE) so that the agent's policy can capture the various dynamics of the multiple environments. Then, robust RL typically solves $\max_\pi \min_{\mathcal{P}^i \in P} \mathbb{E}_\pi[G_t | \mathcal{P}^i]$, where $P = \{\mathcal{P}^i\}$ is the selected environment set. By maximizing

(a) Walker2d+Gravity: GAIL+GP

(b) Walker2d+Mass: GAIL+GP

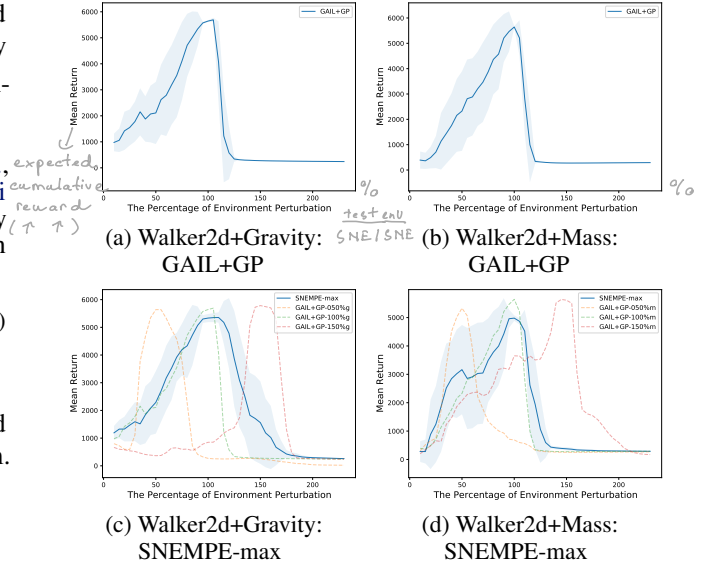(c) Walker2d+Gravity: SNEMPE-max

(d) Walker2d+Mass: SNEMPE-max

Figure 1: Performance of the policy trained in a single nominal environment against perturbation (left column - gravity perturbation and right column - mass perturbation): (a,b) - performance of GAIL+GP, and (c,d) - SNEMPE-max.

the worst-case expected return, the agent's policy can capture the varying dynamics in the selected environment set $P = \{\mathcal{P}^i\}$. However, robust RL requires a well-designed reward function, which we want to avoid.

Now, consider robust IL. One simple approach is to apply the above robust RL principle to the IL setting. Here, we obtain expert demonstrations from multiple sampled demonstration environments and have a single policy interacting with the single nominal interaction environment. Then, we use discriminators to distinguish the policy samples from each of the multiple sampled expert demonstrations, and train the policy to follow the worst-case, i.e., the expert demonstration that is farthest from the policy sample based on the discriminator outputs. The performance of so learned policy in the perturbed test environment is shown in Figures 1c and 1d (the corresponding performance is denoted as SNEMPE-max). It is seen that the policy learned in such way improves robustness compared with conventional SNE/SNE IL in Figures 1a and 1b, but the performance is not satisfactory. This degradation implies that policy interaction with the single nominal environment is not enough to capture the dynamics variation even with expert demonstrations from multiple sampled demonstration environments. Thus, in order to fully capture the dynamics variation, we first sample a few environments with different dynamics from the continuous dynamics distribution and use these multiple sampled environments not only for expert demonstrations but also for policy interaction during the training. We refer to this setting as the MPE/MPE IL setting. In the remainder of this paper, we propose an efficient IL framework based on the MPE/MPE IL setting to yield a policy that performs

robustly against continuous environment dynamics variation based only on a few sampled dynamics for training.

5. Robust Imitation Learning against Variations in Environment Dynamics

5.1. Problem Formulation

We consider a collection of MDPs $\mathcal{C} = \{\mathcal{M} = <\mathcal{S}, \mathcal{A}, \mathcal{P}_\zeta, r, \gamma >, \zeta \in Z\}$, where the state and action spaces are the same for all members of the collection, the reward is unavailable to the agent, the transition probability modeling the dynamics is parameterized with parameter ζ, and the dynamics parameter ζ can be continuously varied or perturbed from the nominal value ζ_0 within the set Z. Among this continuous collection, we sample N MDPs with dynamics parameters $\zeta_1, \zeta_2, \cdots, \zeta_N$. We denote these N environments with dynamics $\mathcal{P}_{\zeta_1}, \cdots, \mathcal{P}_{\zeta_N}$ (simply denoted as $\mathcal{P}^1, \cdots, \mathcal{P}^N$) by $\mathcal{E}_1, \cdots, \mathcal{E}_N$. We assume that there exists an expert π_E^i for each environment $\mathcal{E}_i$, the expert π_E^i generates expert demonstration for the agent, but the expert policy π_E^i itself is not available to the agent. We also assume that the agent can interact with each of all sampled environments $\mathcal{E}_1, \cdots, \mathcal{E}_N$, and the initial state distributions of all interaction environments are the same as $\mu_0(s)$. Thus, according to our definition in the previous section, $\mathcal{E}_1, \cdots, \mathcal{E}_N$ are both demonstration and interaction environments, and the setting is MPE/MPE. Note that the expert demonstrations at $\mathcal{E}_1, \cdots, \mathcal{E}_N$ are partial information about the entire MDP collection $\mathcal{C}$. Our goal is for the agent to learn a policy π that performs well for all members in the MDP collection $\mathcal{C}$ based only on the expert demonstrations from and agent interaction with the sampled environments $\mathcal{E}_1, \cdots, \mathcal{E}_N$. We will refer to this problem as Robust Imitation learning with Multiple perturbed Environments (RIME).

Let us introduce a few more notations. $\rho_\pi^i(s, a) = \pi(a|s) \sum_{t=0}^\infty \gamma^t \Pr(s_t = s | \pi, \mathcal{P}^i)$ denotes the occupancy measure of π in the i-th interaction environment $\mathcal{E}_i$. $\mu_\pi^i(s) = \sum_{t=0}^\infty \gamma^t \Pr(s_t = s | \pi, \mathcal{P}^i)$ denotes the unnormalized state marginal of π in the i-th interaction environment $\mathcal{E}_i$. For simplicity, we denote $\rho_{\pi_E^j}^j(s, a)$ and $\mu_{\pi_E^j}^j(s)$ by $\rho_E^j(s, a)$ and $\mu_E^j(s)$, respectively. The expert demonstration τ_E^i is given by the state-action pair trajectory from expert policy π_E^i specialized in the i-th demonstration environment $\mathcal{E}_i$ with dynamics $\mathcal{P}^i$. $D_{ij}(s, a) : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a discriminator that distinguishes whether a state-action pair (s, a) is from policy π interacting with $\mathcal{E}_i$ or from expert π_E^j.

5.2. Direct Optimization in the Policy Space

In order to solve the RIME problem, one can consider the occupancy matching technique which is used in GAIL. As mentioned in Section 3.2, in the single environment setting,
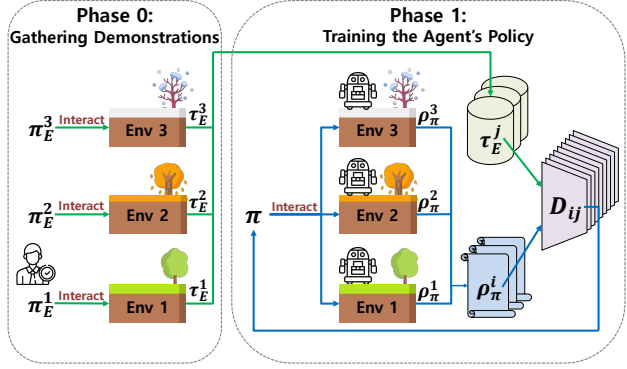

Figure 2: Overview of our algorithm. The blue line is the flow of policy samples ρ_π^i, and the green line is the flow of expert demonstrations $\tau_E^j \sim \rho_E^j$.

the occupancy measure ρ_π satisfies the Bellman flow constraint (1), and there exists a one-to-one mapping between the occupancy measure and the policy. By this relationship, the agent's policy can imitate the expert by matching its occupancy measure close to that of the expert. In the multiple environment setting, however, the situation is not so simple as in the single environment case. Suppose that the agent policy π interacts uniformly with N environments $\mathcal{E}_1, \cdots, \mathcal{E}_N$ with the same state-action space but different transition probabilities $\mathcal{P}^1, \cdots, \mathcal{P}^N$. Then, the occupancy measure of π becomes the mixture, i.e., $\rho_\pi = \frac{1}{N} \sum_{i=1}^N \rho_\pi^i$, and the corresponding Bellman flow equation is given by

$$\rho_\pi(s, a) = \frac{1}{N} \sum_{i=1}^N \rho_\pi^i(s, a) = \mu_0(s)\pi(a|s)$$

$$+ \gamma \frac{1}{N} \sum_{i=1}^N \int_{(s', a')} \mathcal{P}^i(s|s', a')\rho_\pi^i(s', a')\pi(a|s). \quad (4)$$

There exists a distinct characteristic in (4) from the single-environment equation (1). For simplicity of exposition, suppose that the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are discrete and finite with cardinalities $|\mathcal{S}|$ and $|\mathcal{A}|$, respectively. In the case of (1), we have a linear system of equations with $|\mathcal{S}||\mathcal{A}|$ unknowns $\rho_\pi(s, a), (s, a) \in \mathcal{S} \times \mathcal{A}$ and $|\mathcal{S}||\mathcal{A}|$ equations. Hence, we have a unique solution $\rho_\pi(s, a)$ if the kernel $\mathcal{P}(s|s', a')$ satisfies certain Markov chain conditions. In the case of (4), on the other hand, we have $N|\mathcal{S}||\mathcal{A}|$ unknowns $\rho_\pi^i(s, a), i = 1, \cdots, N$ but $|\mathcal{S}||\mathcal{A}|$ equations. So, the system is underdetermined, there exist infinitely many solutions for the set $\{\rho_\pi^i(s, a), i = 1, \cdots, N\}$, and hence the mixture $\rho_\pi = (1/N) \sum_i \rho_\pi^i$ can be infinitely many. Thus, the mapping from π to ρ_π can be one-to-many, so there is no guarantee to recover π from ρ_π unless we prove $\{\rho_\pi\} \cap \{\rho_{\pi'}\} = \emptyset, \forall \pi, \pi'$ such that $\pi \neq \pi'$. Hence, there is no guarantee for policy recovery from occupancy measure matching, and we need to consider a new approach to the RIME problem.

Our approach is not to use the occupancy measure as in

GAIL but to use the policy distribution itself. For the considered MPE/MPE setting, we propose the following objective function to solve the RIME problem:

unnormalized state marginal.

$$\min_{\pi} \mathbb{E}_{s \sim \frac{1}{N} \sum_{i=1}^{N} \mu_{\pi}^i} \left[\sum_{j=1}^{N} \lambda_j(s) \cdot \mathcal{D}(\pi(\cdot|s), \pi_E^j(\cdot|s)) \right], \quad (5)$$

where $\mathcal{D}$ is some divergence between two policy distributions, and $\sum_j \lambda_j(s) = 1$. The objective function (5) means that we want to design the agent policy π to appropriately imitate all expert policies $\pi_E^1, \cdots, \pi_E^N$ on the state samples generated by the agent policy interacting with all interaction environments. Here, $\lambda_j(s)$ is the weight to determine how much $\pi_E^j(\cdot|s)$ is imitated. Such an objective has been considered for *integration of expert machines* (Amari, 2016) and is well suited to our purpose. The key difference between (5) and (2) is that in (2), the distance between the occupancy measures of the agent and the expert is minimized based on Proposition 3.1, whereas in (5) the distance between the policy distribution of the agent and those of the multiple experts is minimized, not requiring the occupancy measures. However, the key challenge to the objective function (5) is that the expert policies $\pi_E^1, \cdots, \pi_E^N$ are not available but only their demonstrations are at hand. The following theorem is the first step to circumvent this difficulty.

Theorem 5.1. *If $\rho_{\pi}^i(s,a) > 0$, $\lambda_j(s) > 0$ for any $i, j \in \{1, \cdots, N\}$, $\gamma \in (0,1)$, and $\mathcal{D}$ in (5) is the Jensen-Shannon divergence, then the objective function (5) is expressed as*

$$\min_{\pi} \sum_{i=1}^{N} \sum_{j=1}^{N} \max_{D_{ij}} \left\{ \mathbb{E}_{(s,a) \sim \rho_{\pi}^i} \left[\frac{\lambda_j(s)}{2N} \log(1 - D_{ij}(s,a)) \right] \right.$$

$$\left. + \mathbb{E}_{s \sim \mu_{\pi}^i, a \sim \pi_E^j} \left[\frac{\lambda_j(s)}{2N} \log(D_{ij}(s,a)) \right] \right\} + \frac{\log 2}{1 - \gamma}, \quad (6)$$

where D_{ij} is a discriminator that distinguishes whether (s,a) is from policy π interacting with $\mathcal{E}_i$ or from expert π_E^j

Proof. See in Appendix A.1 □

5.3. Practical Methodology

Due to the second term $\mathbb{E}_{s \sim \mu_{\pi}^i(s), a \sim \pi_E^j(\cdot|s)}[\cdot]$ in (6), which is eventually replaced with sample expectation in implementation, we still require the expert policies π_E^j, $j = 1, \cdots, N$. However, π_E^j is not available. One way to circumvent this is to reproduce the expert policy π_E^j via Behavior Cloning or GAIL+GP by using its demonstration τ_E^j. However, we found that this method is not so effective. This is due to the classical generalization problem. That is, the reproduced expert policy $\hat{\pi}_E^j$ based on τ_E^j does not cover all states induced by π (i.e., $s \sim \mu_{\pi}^i$). For some states, $\hat{\pi}_E^j$ gives inappropriate actions to the agent policy, and these actions lead to learning failure. (The detailed description and experimental results of this approach are in Appendix B.1.) To circumvent this, using importance sampling, we modify (6) as follows:

$$\min_{\pi} \sum_{i=1}^{N} \sum_{j=1}^{N} \max_{D_{ij}} \left\{ \mathbb{E}_{(s,a) \sim \rho_{\pi}^i} \left[\lambda_j(s) \log(1 - D_{ij}(s,a)) \right] \right.$$

$$\left. + \mathbb{E}_{(s,a) \sim \rho_E^j} \left[\frac{\mu_{\pi}^i(s)}{\mu_E^j(s)} \lambda_j(s) \log(D_{ij}(s,a)) \right] \right\}, \quad (7)$$

where the last constant term $\log 2/(1 - \gamma)$ and the constant scaling factor $1/2N$ in (6) are removed. The difference of (7) from (6) is that for the expectation in the second term, the sample pair (s,a) is drawn from the expert trajectory, which facilitates implementation. Instead, we need the importance sampling ratio $\frac{\mu_{\pi}^i(s)}{\mu_E^j(s)}$. However, computing $\mu_{\pi}^i(s)$ and $\mu_E^j(s)$ for a continuous state space by the Bellman flow equation is difficult because we have an infinitely large space, and also the transition dynamics are unknown in the model-free case. In addition, computing $\mu_{\pi}^i(s)$ and $\mu_E^j(s)$ based on samples is also difficult unless we assume a predefined model distribution. One can consider applying histogram-based neural network approaches but then again faces the generalization issue. Hence, instead of computing $\mu_{\pi}^i(s)$ and $\mu_E^j(s)$, we directly estimate the ratio $\frac{\mu_{\pi}^i(s)}{\mu_E^j(s)}$ by using f-divergence (Sinha et al., 2020) (detailed implementation and experimental results are in Appendix D.2). However, we found that properly estimating $\frac{\mu_{\pi}^i(s)}{\mu_E^j(s)}$ and setting $\frac{\mu_{\pi}^i(s)}{\mu_E^j(s)}$ simply to 1 have almost the same results for most tasks. Thus, for algorithm simplicity, we set the importance ratio to 1 without estimating the ratio. Indeed, similar approaches were used in (Kostrikov et al., 2018; Liu et al., 2020).

With the importance sampling ratio set to 1, the optimization over π and D_{ij} in (7) is tractable. We can apply alternating optimization over π and D_{ij}. First, consider optimization over π for given D_{ij}. Note that π affects only the first term $\mathbb{E}_{\rho_{\pi}^i}[\cdot]$ in (7). In the first term, we have the weighting factor $\lambda_j(s)$ such that $\sum_{j=1}^{N} \lambda_j(s) = 1$, and determining proper $\lambda_j(s)$ is cumbersome. Thus, exploiting the fact $\sum_{j=1}^{N} \lambda_j(s) = 1$, we can rewrite the first term for given D_{ij} by pushing $\sum_{j=1}^{N}$ into the expectation based on the linearity of expectation, and obtain its upper bound as

$$\min_{\pi} \sum_{i=1}^{N} \mathbb{E}_{\rho_{\pi}^i} \left[\sum_{j=1}^{N} \lambda_j(s) \log(1 - D_{ij}(s,a)) \right]$$

$$\leq \min_{\pi} \sum_{i=1}^{N} \mathbb{E}_{\rho_{\pi}^i} \left[\max_j \log(1 - D_{ij}(s,a)) \right], \quad (8)$$

where $\mathbb{E}_{\rho_{\pi}^i}[\cdot]$ denotes $\mathbb{E}_{(s,a) \sim \rho_{\pi}^i}[\cdot]$, and the inequality is valid because $\sum_{j=1}^{N} \lambda_j(s)[\cdot]$ can be considered as an expectation ($\max_{D_{ij}}$ does not appear since D_{ij} is given for this step). Then, we optimize the upper bound of the objective function (8) for policy π.

Next, consider the optimization of D_{ij} for given π. This optimization is simplified due to the following theorem:

Theorem 5.2. *The following maximization problem without the $\lambda_j(s)$ term has the same optimal solution for D_{ij} as* (7) *with $\mu^i_\pi(s)/\mu^j_E(s)$ set to 1 for given π:*

$$\max_{D_{ij}} \left\{ \mathbb{E}_{\rho^i_\pi} [\log(1 - D_{ij}(s,a))] + \mathbb{E}_{\rho^j_E} [\log(D_{ij}(s,a))] \right\}.$$

Proof. See in Appendix A.2 $\qquad \square$

Based on Theorem 5.2 and gradient penalty (GP), we finally derive the objective function of D_{ij} for given π as follows:

$$\max_{D_{ij}} \left\{ \mathbb{E}_{\rho^i_\pi} [\log(1 - D_{ij}(s,a))] + \mathbb{E}_{\tau^j_E} [\log(D_{ij}(s,a))] \right.$$
$$\left. + \kappa \mathbb{E}_{\hat{x}} \left(\|\nabla_{\hat{x}} D_{ij}(\hat{x})\|_2 - 1 \right)^2 \right\}, \quad (9)$$

where $\hat{x} = (s,a) \sim (\epsilon \rho^i_\pi + (1-\epsilon)\tau^j_E)$ with $\epsilon \sim \text{Unif}[0,1]$, and κ is the weight to control the GP term. Note that in (9) we added a gradient penalty term mentioned in Section 3.2 for stable learning, and $\mathbb{E}_{\rho^j_E}$ is replaced with $\mathbb{E}_{\tau^j_E}$.

Note that the number of discriminators D_{ij} is given by N^2, and increases quadratically as the number N of environments increases. We can reduce this number by using discriminator weight sharing which makes the discriminator models share a subset of their weights (Liu & Tuzel, 2016). The discriminators $D_{i1}, \cdots, D_{iN}$ share the weights of their input and hidden layers, and hence they can be implemented as one discriminator with N output nodes. We call this Weight-Shared Discriminator (WSD). For WSD $D_i^{\text{Weight-Shared}}$, the j-th output of its N output nodes corresponds to the output of D_{ij}, and its objective is given by $\sum_j V_{ij}$, where V_{ij} is the individual objective for D_{ij} in (9). Using WSDs $D_i^{\text{Weight-Shared}}$, $i = 1, \cdots, N$, the complexity of discriminators is reduced and is almost $\sim N$.

5.4. Comparison with Occupancy Measure Matching

Even without guarantee of the recovery of policy distribution from the occupancy measure in the case of MPE, we can still apply the occupancy measure matching technique to MPE/MPE. In this case, a reasonable objective is given by

$$\min_\pi \sum_{j=1}^N \lambda_j \mathcal{D}_{JS}(\bar{\rho}_\pi, \bar{\rho}^j_E), \quad (10)$$

where $\sum_j \lambda_j = 1$, and $\bar{\rho}_\pi$ and $\bar{\rho}^j_E$ are the normalized occupancy distributions obtained from ρ_π and ρ^j_E, respectively. (Other objectives are also considered in Section 6.) Then, we can derive an upper bound of (10) as follows:

$$\min_\pi \sum_{j=1}^N \lambda_j \mathcal{D}_{JS}(\bar{\rho}_\pi, \bar{\rho}^j_E)$$
$$\leq \min_\pi \sum_{i=1}^N \sum_{j=1}^N \frac{\lambda_j(1-\gamma)}{2N} \max_{D_{ij}} \left\{ \mathbb{E}_{\rho^i_\pi} [\log(1 - D_{ij}(s,a))] \right.$$
$$\left. + \mathbb{E}_{\rho^j_E} [\log D_{ij}(s,a)] \right\} + \log 2,$$

where the derivation of this upper bound is in Appendix B.2. Now consider the optimization of π for given D_{ij} in this case. Again, in order to handle λ_j, we can replace $\sum_j \lambda_j$ with $\max_j$ to yield another upper bound. Then, the objective function of π for given D_{ij} is given by

$$\min_\pi \sum_{i=1}^N \max_j \mathbb{E}_{\rho^i_\pi} [\log(1 - D_{ij}(s,a))]. \quad (11)$$

We refer to this method as Occupancy measure Matching in Multiple Environments (OMME). The key difference of the objective (11) from the proposed one in (8) is that the operation $\max_j$ is outside the expectation $\mathbb{E}_{\rho^i_\pi}[\cdot]$. Note that the order is not interchangeable since $\max_j$ is a nonlinear operation. We will see that this seemingly-slight difference makes a significant performance difference in Section 6.

6. Experiments

6.1. Experimental Settings

We considered our algorithm together with the following baselines:
- Behavior Cloning (BC): The policy is trained by supervised learning until validation errors of all expert demonstrations stop decreasing.
- GAIL-mixture: It is a variant of GAIL+GP for MPE. In this case, we have a single discriminator, and this discriminator distinguishes between all $\bar{\rho}^i_\pi$'s and all $\bar{\rho}^j_E$'s. Its objective function for π is $\min_\pi \mathcal{D}_{JS}(\sum_i \bar{\rho}^i_\pi/N, \sum_j \bar{\rho}^j_E/N)$.
- GAIL-single: It is another variant of GAIL+GP for MPE. In this case, we have multiple discriminators, and the objective function for π is $\min_\pi \sum_i \mathcal{D}_{JS}(\bar{\rho}^i_\pi, \bar{\rho}^i_E)$.
- OMME (closest to our algorithm): this is described already. The objective function is given by (10) with (11).

Detailed description of the baselines, implementation, expert demonstrations are in Appendix F. We considered two versions of the proposed algorithm: RIME and RIME+WSD. The only difference between RIME and RIME+WSD is the implementation of discriminators D_{ij}. RIME has the discriminators D_{ij} with the objective function (9) and hence the number of the discriminator networks is N^2. On the other hand, RIME+WSD uses weight-shared discriminator $D_i^{\text{Weight-Shared}}$ described at the end of Section 5.3.

We experimented the considered algorithms on MuJoCo tasks: Hopper, Walker2d, HalfCheetah and Ant (Todorov et al., 2012). Each expert demonstration contains 50 trajectories (i.e., episodes) of state-action pairs generated by the expert and one episode has 1000 timesteps. We considered gravity or mass for the considered tasks as our dynamics perturbation parameter ζ. The nominal value ζ_0 means 100% gravity or mass for each MuJoCo task. We trained all algorithms with 10M timesteps in the case of experiments with a 1-D dynamics parameter and with 5M timesteps in the

Table 1: Mean return / minimum return over the dynamics parameter range [50%,150%] in the 2 sampled environment case

Algorithm	Hopper +Gravity	Walker2d +Gravity	HalfCheetah +Gravity	Ant +Gravity
RIME (ours)	**2886.7 / 2332.4**	**4577.1 / 4260.9**	**4268.9 / 3712.0**	**4402.2 / 3909.9**
RIME+WSD (ours)	**2857.8 / 2333.2**	**4539.3 / 4235.8**	**4292.9 / 3802.5**	**4388.7 / 3871.8**
OMME	2020.3 / 1354.3	4467.2 / 3868.7	3854.4 / 3352.9	3787.8 / 2715.1
GAIL-mixture	1797.9 / 959.3	3286.7 / 1256.7	3688.6 / 2998.4	3614.1 / 2856.5
GAIL-single	1616.4 / 844.7	3210.0 / 1289.0	3571.8 / 2673.1	3314.3 / 2316.7
BC	1129.7 / 648.8	971.4 / 313.0	1299.5 / -18.3	2333.8 / 1988.6

Algorithm	Hopper +Mass	Walker2d +Mass	HalfCheetah +Mass	Ant +Mass
RIME (ours)	**3535.7 / 3255.6**	**4597.0** / 3965.8	3959.0 / **3156.7**	**4554.5 / 4417.5**
RIME+WSD (ours)	**3499.4 / 3238.4**	**4564.7 / 4174.9**	**4071.7** / 3254.7	**4539.6 / 4439.5**
OMME	3008.6 / 2741.4	4046.6 / 3460.0	3533.5 / 2732.0	4494.6 / 4343.2
GAIL-mixture	2334.5 / 1333.5	3493.8 / 1425.3	2794.7 / 1951.0	**4504.9** / 4301.7
GAIL-single	2194.1 / 1266.6	3031.5 / 1220.4	3164.9 / 1685.6	4031.1 / 3767.4
BC	726.5 / 453.9	962.6 / 607.4	474.2 / -132.9	3923.7 / 3519.0
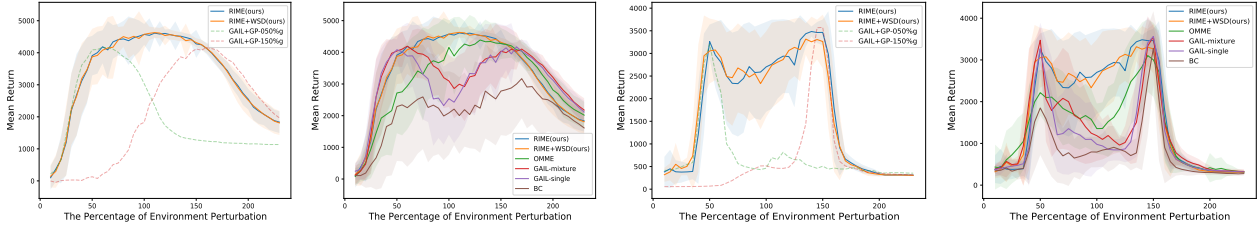
(a) Ant+Gravity: performance (b) Ant+Gravity: comparisons (c) Hopper+Gravity: performance (d) Hopper+Gravity: comparisons

Figure 3: Performance on the actual test environment with gravity-perturbed dynamics (the graphs with mass perturbation are in Appendix E.1)

case of experiments with 2-D dynamics parameters, and the algorithm for updating the policy is PPO (Schulman et al., 2017; 2015b).

6.2. Results

For the same task, we conducted 3 experiments. The first two correspond to the case in which a single dynamics parameter (gravity or mass) is perturbed from the nominal value, and the third is the case in which both gravity and mass parameters are perturbed. The setting for the first is $N = 2$ sampled environments with sampled gravity (or mass) parameters $50\%\zeta_0$ and $150\%\zeta_0$, and the setting for the second is $N = 3$ sampled environments with sampled gravity (or mass) parameters $50\%\zeta_0$, $100\%\zeta_0$ and $150\%\zeta_0$. In the third case, we sampled the joint dynamics of gravity and mass as $50\%\zeta_{0,g}50\%\zeta_{0,m}$, $50\%\zeta_{0,g}150\%\zeta_{0,m}$, $150\%\zeta_{0,g}50\%\zeta_{0,m}$ and $150\%\zeta_{0,g}150\%\zeta_{0,m}$ with $N = 4$. Note that in the third case, we want to cover the variation from 50% to 150% for both parameters and only sampled the four corner points in the joint gravity-mass parameter space.

With the sampled N environments, we trained the agent by applying the IL algorithms considered in Section 6.1. Then, in the 1-D perturbation case, we tested the trained agent policy in each of test environments of which dynamics parameter ζ varies from $10\%\zeta_0$ to $230\%\zeta_0$ with

$5\%\zeta_0$ step, i.e., $10\%\zeta_0, 15\%\zeta_0, \cdots, 230\%\zeta_0$. In the 2-D perturbation case, we tested the trained algorithms for each of test environments with dynamics parameters $[50\%, 70\%, \cdots, 150\%]\zeta_{0,g} \times [50\%, 70\%, \cdots, 150\%]\zeta_{0,m}$.

IL with 2 Sampled Environments (50%, 150%): Figure 3 shows the result in the case of 2 sampled environments with $\zeta = 50\%\zeta_0$ and $150\%\zeta_0$. Figures 3a and 3b show the test environment performance of the trained policies of the considered algorithms on the Ant+Gravity task, where the gravity parameter varies. As seen in Figure 3a, GAIL+GP trained at $50\%\zeta_0$ and GAIL+GP trained at $150\%\zeta_0$ perform well only around the trained dynamics. On the other hand, the proposed algorithm (RIME) performs well across all dynamics variation range between the two trained points. It is seen that in the middle the performance of RIME is even better than the peak of the single-environment-specialized GAIL+GP policy. Figure 3b shows the performance of other MPE IL algorithms. It is seen that other MPE IL algorithms' performance degrades for the unseen dynamics. Note that the performance sensitivity with respect to the dynamics parameter is mild in the case of Ant+Gravity. Figures 3c and 3d show the test environment performance for Hopper+Gravity in which the performance sensitivity with respect to the dynamics parameter is high. As seen in Figure 3c, in this case, GAIL+GP can perform only well in a very narrow region around the trained point. On the other hand, the proposed
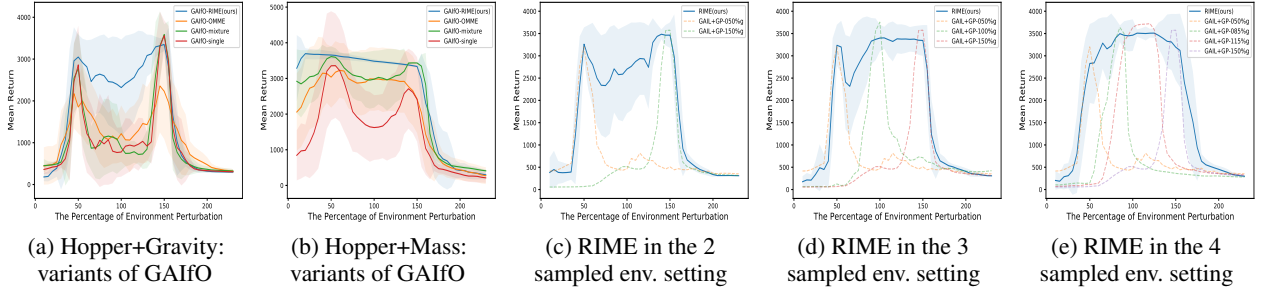
Table 2: Mean return / minimum return over the dynamics parameter range [50%,150%] in the 3 sampled environment case

Algorithm	Hopper +Gravity	Walker2d +Gravity	HalfCheetah +Gravity	Ant +Gravity
RIME (ours)	3164.4 / 2315.5	**5197.1 / 4820.6**	5012.8 / 4599.2	**4290.8 / 3485.3**
RIME+WSD (ours)	**3281.1 / 2764.5**	**5231.7 / 4894.4**	**5025.7 / 4727.9**	4168.6 / 3338.2
OMME	2878.8 / 2260.6	5106.9 / 4484.9	4693.4 / 4516.8	3689.2 / 2091.5
GAIL-mixture	2905.0 / 2289.9	4549.7 / 2543.5	4746.0 / 4297.6	3882.4 / 3431.8
GAIL-single	2533.9 / 1276.7	4104.0 / 2342.8	4512.7 / 4033.1	3619.5 / 3088.7
BC	798.6 / 448.2	791.6 / 594.9	1621.9 / 509.8	2188.6 / 1129.2

Algorithm	Hopper +Mass	Walker2d +Mass	HalfCheetah +Mass	Ant +Mass
RIME (ours)	**3597.1 / 3244.6**	**4752.9 / 4198.7**	5248.2 / 4637.3	**4506.1 / 4384.6**
RIME+WSD (ours)	**3585.7 / 3198.2**	4704.2 / **4260.7**	**5308.5 / 4868.8**	4417.0 / 4202.1
OMME	3109.3 / 2815.8	4495.2 / 3782.7	4802.9 / 4077.7	4268.0 / 4036.8
GAIL-mixture	3100.1 / 2525.6	**4824.1** / 3605.3	4237.1 / 3223.3	4368.6 / 4075.5
GAIL-single	1526.0 / 1011.7	4663.1 / 3667.6	4088.0 / 2901.7	4055.3 / 3603.3
BC	410.7 / 162.7	704.5 / 354.0	1046.7 / -385.2	4057.1 / 3740.1

(a) Hopper+Gravity: variants of GAIfO

(b) Hopper+Mass: variants of GAIfO

(c) RIME in the 2 sampled env. setting

(d) RIME in the 3 sampled env. setting

(e) RIME in the 4 sampled env. setting

Figure 4: (a-b): Performances when the algorithms use state-only expert demonstration (the graphs for other tasks are in Appendix D.3), (c-e): The performance of RIME with respect to N for Hopper+Gravity task

method performs well in the full unseen region between the two trained points. Note that the test performance of the proposed algorithm is superb in the unseen region as compared to other MPE IL baselines, as seen in Figure 3d.

Table 1 summarizes the robustness performance. We tested each algorithm at the test dynamics $50\%\zeta_0$, $55\%\zeta_0$, $\cdots$, $150\%\zeta_0$ with 5% quantization between the two sampled dynamics values 50% and 150%. We then averaged the performance over the test values and took the minimum performance over the test values. So, when the average and minimum values are equal, the test performance is flat across the tested region, showing the robustness over the variation. It is seen that the proposed algorithm is superior to other algorithms.

IL with 3 Sampled Environments (50%, 100%, 150%): Next, we tested the algorithms trained based on $N = 3$ with dynamics parameters $50\%\zeta_0$ and $100\%\zeta_0$ and $150\%\zeta_0$. This setting has more densely-sampled environments compared to $N = 2$. Table 2 shows the corresponding result. (Table 2 was constructed in a similar way to Table 1.) It is seen that the proposed algorithm is superior to others for a variety of tasks with wide ranges of perturbation.

2-D Perturbation Parameter Case: Table 3 summarizes the robustness performance of the algorithms on the test environments with 2-D perturbation (gravity and mass). Figure 5 shows the mean-return color plot for the performance of the algorithms for the Hopper task. It is seen that our proposed algorithm performs well within the entire 2-D parameter space $[50\%,150\%]\zeta_{0,g} \times [50\%,150\%]\zeta_{0,m}$ by only sampling the four corner points. With this result, we conjecture that even for higher dimensional perturbation, the proposed method with sampled environments only at the corner points performs well. Additional experimental results are available in Appendix E.

6.3. Ablation Studies

State-only Expert Demonstration: Torabi et al. (2018b) stated that demonstrations from various resources lack the information on expert's action and addressed the problem of Imitation from Observation (IfO). We tested the proposed RIME algorithm and GAIL variants in a situation in which state-only expert demonstrations are available. We trained the algorithms by using state-only demonstrations, which are variants of GAIfO, in the case of $N = 2$ sampled environments. The result is shown in Figures 4a and 4b. It is seen that RIME performs well across the test environment perturbation. This result indicates that our method can appropriately recover experts' preference over the state space.

Table 3: Mean return / minimum return over the dynamics parameter range $[50\%g, 150\%g] \times [50\%m, 150\%m]$ in the 4 sampled environments case with 2-dimensional perturbation parameters

| Algorithm	Hopper + (G&M)	Walker2d + (G&M)	HalfCheetah + (G&M)	Ant + (G&M)
RIME (ours)	**3043.3** / 2430.8	4463.4 / 3824.1	**3721.3** / 2753.1	**4671.7** / 4233.5
RIME+WSD (ours)	2936.9 / **2331.6**	**4646.4** / **4000.2**	3717.9 / **2891.7**	4651.4 / **4304.5**
OMME	2573.4 / 1986.4	4488.8 / 3029.3	3498.5 / 2502.2	4625.3 / 3594.5
GAIL-mixture	1636.4 / 712.0	3907.8 / 1245.1	3018.6 / 1982.3	3994.8 / 2746.1
GAIL-single	1684.9 / 840.0	3844.8 / 2484.2	3199.1 / 2072.6	3799.7 / 2194.1
BC	500.2 / 317.2	330.0 / 211.0	1289.3 / 30.2	1728.2 / 1032.7
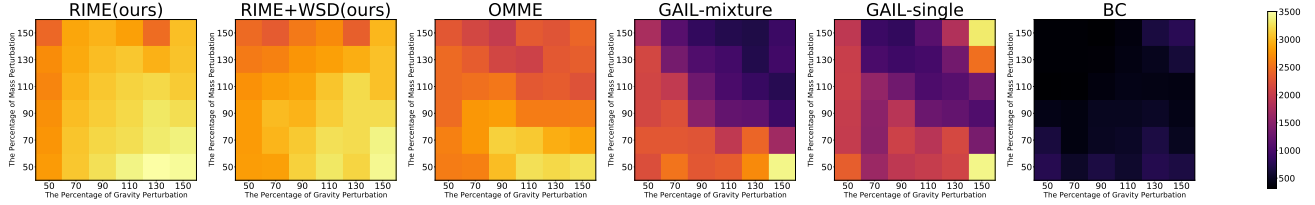
Figure 5: Performance on the test environment with both gravity and mass perturbation for Hopper (the graphs for other tasks are in Appendix E.3)

Impact of the Sample Size of Expert Demonstration: In the previous section, we used expert demonstrations containing 50 trajectories. However, there may not be sufficient expert demonstrations in the real world. Thus, we performed experiments by reducing the expert demonstration samples gradually from 50 trajectories. Due to space limitation, the result is in Appendix D.4. There, we can see that the proposed robust IL algorithm works quite well even if the expert demonstration length decreases.

Tendency over N: From Tables 1 and 2, we observe that the mean or minimum return performance of the proposed algorithm did not improve monotonically as N changes from 2 to 3. In certain cases, mean return or minimum return slightly decreased as N increases from 2 to 3, although the decrease is not severe. For example, in the case of Hopper+Gravity, the mean and minimum return values of 2886.7 and 2332.4 change to 3164.4 and 2315.5, as N increases from 2 to 3. In order to check the performance tendency with respect to N, we further tested the proposed algorithm trained with $N = 4$ sampled environments $\mathcal{E}_1, \cdots, \mathcal{E}_4$ with dynamics parameters $\zeta_1 = 050\%\zeta_0$, $\zeta_2 = 085\%\zeta_0$, $\zeta_3 = 115\%\zeta_0$ and $\zeta_4 = 150\%\zeta_0$. Figures 4c to 4e show the performance tendency as N increases. It is hard to say that at every step of N the performance increases as N increases, but there exists a tendency of improvement as N increases. Note that the test performance for $N = 4$ is smooth across the variation.

The source code of the proposed algorithm is available at https://github.com/JongseongChae/RIME.

7. Conclusion

In this paper, we have considered two issues for the deployment of RL for real-world control problems such as autonomous driving: robustness and proper reward design. To address these issues, we have introduced a new framework for robust IL based on multiple environments with dynamics parameters sampled from the continuous range of dynamics parameter variation. Since it is not obvious that one can recover the policy from the occupancy measure in the case of multiple environments, we have approached the problem by directly optimizing the agent policy in the policy space. We have formulated the problem as minimization of the weighted average of divergences from the agent policy to the multiple expert policies. Through a series of manipulations, we have shown that the proposed objective function can be expressed eventually as a formula with implementable familiar operations such as expectation, max and discrimination. We have evaluated the robustness of the resulting algorithm on MuJoCo tasks by varying gravity or/and mass parameter(s). Numerical results show that the proposed IL algorithm shows superior performance in robustness across a wide range of dynamics parameter variation based only on training with a few sampled environment dynamics.

Acknowledgement

References

Amari, S.-i. *Information geometry and its applications*, volume 194. Springer, 2016.

Brantley, K., Sun, W., and Henaff, M. Disagreement-regularized imitation learning. In *International Conference on Learning Representations*, 2019.

Derman, E., Mankowitz, D. J., Mann, T. A., and Mannor, S. Soft-robust actor-critic policy-gradient. *arXiv preprint arXiv:1803.04848*, 2018.

Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. *arXiv preprint arXiv:1703.07326*, 2017.

Durrett, R. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pp. 49–58, 2016.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pp. 357–368. PMLR, 2017.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Gangwani, T. and Peng, J. State-only imitation with transition dynamics mismatch. *arXiv preprint arXiv:2002.11879*, 2020.

Gleave, A. and Habryka, O. Multi-task maximum entropy inverse reinforcement learning. *arXiv preprint arXiv:1805.08882*, 2018.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Ho, J. and Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.

Iyengar, G. N. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

James, S., Bloesch, M., and Davison, A. J. Task-embedded control networks for few-shot imitation learning. In *Conference on Robot Learning*, pp. 783–795. PMLR, 2018.

Kostrikov, I. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., and Tompson, J. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

Laskey, M., Lee, J., Fox, R., Dragan, A., and Goldberg, K. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pp. 143–156. PMLR, 2017.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu, F., Ling, Z., Mu, T., and Su, H. State alignment-based imitation learning. *arXiv preprint arXiv:1911.10947*, 2019.

Liu, M., Zhou, M., Zhang, W., Zhuang, Y., Wang, J., Liu, W., and Yu, Y. Multi-agent interactions modeling with correlated policies. *arXiv preprint arXiv:2001.03415*, 2020.

Liu, M.-Y. and Tuzel, O. Coupled generative adversarial networks. *Advances in neural information processing systems*, 29, 2016.

Lu, Y. and Tompson, J. Adail: Adaptive adversarial imitation learning. *arXiv preprint arXiv:2008.12647*, 2020.

Mankowitz, D. J., Mann, T. A., Bacon, P.-L., Precup, D., and Mannor, S. Learning robust options. *arXiv preprint arXiv:1802.03236*, 2018.

Mankowitz, D. J., Levine, N., Jeong, R., Shi, Y., Kay, J., Abdolmaleki, A., Springenberg, J. T., Mann, T., Hester, T., and Riedmiller, M. Robust reinforcement learning for continuous control with model misspecification. *arXiv preprint arXiv:1906.07516*, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. *arXiv preprint arXiv:1703.02702*, 2017.

Raychaudhuri, D. S., Paul, S., Vanbaar, J., and Roy-Chowdhury, A. K. Cross-domain imitation from observations. In *International Conference on Machine Learning*, pp. 8902–8912. PMLR, 2021.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015a.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sinha, S., Song, J., Garg, A., and Ermon, S. Experience replay with likelihood-free importance weights. *arXiv preprint arXiv:2006.13169*, 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Syed, U., Bowling, M., and Schapire, R. E. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, pp. 1032–1039, 2008.

Tangkaratt, V., Charoenphakdee, N., and Sugiyama, M. Robust imitation learning from noisy demonstrations. *arXiv preprint arXiv:2010.10181*, 2020.

Tessler, C., Efroni, Y., and Mannor, S. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pp. 6215–6224. PMLR, 2019.

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

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018a.

Torabi, F., Warnell, G., and Stone, P. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018b.

Toyer, S., Shah, R., Critch, A., and Russell, S. The magical benchmark for robust imitation. *arXiv preprint arXiv:2011.00401*, 2020.

Viano, L., Huang, Y.-T., Kamalaruban, P., Weller, A., and Cevher, V. Robust inverse reinforcement learning under transition dynamics mismatch. *Advances in Neural Information Processing Systems*, 34, 2021.

Vinitsky, E., Du, Y., Parvate, K., Jang, K., Abbeel, P., and Bayen, A. Robust reinforcement learning using adversarial populations. *arXiv preprint arXiv:2008.01825*, 2020.

Wang, Z., Merel, J., Reed, S., Wayne, G., de Freitas, N., and Heess, N. Robust imitation of diverse behaviors. *arXiv preprint arXiv:1707.02747*, 2017.

Wiesemann, W., Kuhn, D., and Rustem, B. Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

Wu, Y.-H., Charoenphakdee, N., Bao, H., Tangkaratt, V., and Sugiyama, M. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, pp. 6818–6827. PMLR, 2019.

Xu, K., Ratner, E., Dragan, A., Levine, S., and Finn, C. Learning a prior over intent via meta-inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 6952–6962. PMLR, 2019.

Yu, L., Yu, T., Finn, C., and Ermon, S. Meta-inverse reinforcement learning with probabilistic context variables. *arXiv preprint arXiv:1909.09314*, 2019.

Zhou, A., Jang, E., Kappler, D., Herzog, A., Khansari, M., Wohlhart, P., Bai, Y., Kalakrishnan, M., Levine, S., and Finn, C. Watch, try, learn: Meta-learning from demonstrations and reward. *arXiv preprint arXiv:1906.03352*, 2019.

A. Proofs

A.1. Proof of Theorem 5.1

Theorem 5.1 If $\rho_\pi^i(s,a) > 0$, $\lambda_j(s) > 0$ for any $i, j \in \{1, \cdots, N\}$, $\gamma \in (0,1)$, and $\mathcal{D}$ in eq. (5) in the main paper is the Jensen-Shannon divergence, then eq. (5) in the main paper is expressed as

$$\min_\pi \sum_{i=1}^N \sum_{j=1}^N \max_{D_{ij}} \left\{ \mathbb{E}_{(s,a)\sim\rho_\pi^i} \left[\frac{\lambda_j(s)}{2N} \log(1 - D_{ij}(s,a)) \right] + \mathbb{E}_{s\sim\mu_\pi^i, a\sim\pi_E^j} \left[\frac{\lambda_j(s)}{2N} \log(D_{ij}(s,a)) \right] \right\} + \frac{\log 2}{1-\gamma}.$$

Proof.

$$\min_\pi \mathbb{E}_{s\sim\frac{1}{N}\sum_{i=1}^N \mu_\pi^i} \left[\sum_{j=1}^N \lambda_j(s) \mathcal{D}_{JS}(\pi(\cdot|s), \pi_E^j(\cdot|s)) \right]$$

$$= \min_\pi \int_{s\in\mathcal{S}} \frac{1}{N} \sum_{i=1}^N \mu_\pi^i(s) \sum_{j=1}^N \lambda_j(s) \mathcal{D}_{JS}(\pi(\cdot|s), \pi_E^j(\cdot|s))$$

$$= \min_\pi \int_{s\in\mathcal{S}} \frac{1}{N} \sum_{i=1}^N \mu_\pi^i(s) \sum_{j=1}^N \frac{\lambda_j(s)}{2} \left\{ \int_{a\in\mathcal{A}} \pi(a|s) \log \frac{2\pi(a|s)}{\pi(a|s)+\pi_E^j(a|s)} + \pi_E^j(a|s) \log \frac{2\pi_E^j(a|s)}{\pi(a|s)+\pi_E^j(a|s)} \right\}$$

$$= \min_\pi \int_{s\in\mathcal{S}} \frac{1}{N} \sum_{i=1}^N \mu_\pi^i(s) \sum_{j=1}^N \frac{\lambda_j(s)}{2} \left\{ \int_{a\in\mathcal{A}} \pi(a|s) \log \frac{\pi(a|s)}{\pi(a|s)+\pi_E^j(a|s)} + \pi_E^j(a|s) \log \frac{\pi_E^j(a|s)}{\pi(a|s)+\pi_E^j(a|s)} \right\}$$

$$+ \int_{s\in\mathcal{S}} \frac{1}{N} \sum_{i=1}^N \mu_\pi^i(s) \sum_{j=1}^N \lambda_j(s) \log 2$$

$$\overset{(a)}{=} \min_\pi \int_{s\in\mathcal{S}} \frac{1}{N} \sum_{i=1}^N \mu_\pi^i(s) \sum_{j=1}^N \frac{\lambda_j(s)}{2} \left\{ \int_{a\in\mathcal{A}} \pi(a|s) \log \frac{\pi(a|s)}{\pi(a|s)+\pi_E^j(a|s)} + \pi_E^j(a|s) \log \frac{\pi_E^j(a|s)}{\pi(a|s)+\pi_E^j(a|s)} \right\}$$

$$+ \int_{s\in\mathcal{S}} \frac{1}{N} \sum_{i=1}^N \mu_\pi^i(s) \log 2$$

$$\overset{(b)}{=} \min_\pi \sum_{i=1}^N \sum_{j=1}^N \int_{s\in\mathcal{S}} \int_{a\in\mathcal{A}} \mu_\pi^i(s) \frac{\lambda_j(s)}{2N} \left\{ \pi(a|s) \log \frac{\pi(a|s)}{\pi(a|s)+\pi_E^j(a|s)} + \pi_E^j(a|s) \log \frac{\pi_E^j(a|s)}{\pi(a|s)+\pi_E^j(a|s)} \right\} + \frac{\log 2}{1-\gamma}$$

$$= \min_\pi \sum_{i=1}^N \sum_{j=1}^N \int_{s\in\mathcal{S}} \int_{a\in\mathcal{A}} \left\{ \pi(a|s)\mu_\pi^i(s) \frac{\lambda_j(s)}{2N} \log \frac{\pi(a|s)\mu_\pi^i(s)}{\pi(a|s)\mu_\pi^i(s)+\pi_E^j(a|s)\mu_\pi^i(s)} \right.$$

$$\left. + \pi_E^j(a|s)\mu_\pi^i(s) \frac{\lambda_j(s)}{2N} \log \frac{\pi_E^j(a|s)\mu_\pi^i(s)}{\pi(a|s)\mu_\pi^i(s)+\pi_E^j(a|s)\mu_\pi^i(s)} \right\} + \frac{\log 2}{1-\gamma}$$

$$= \min_\pi \sum_{i=1}^N \sum_{j=1}^N \int_{s\in\mathcal{S}} \int_{a\in\mathcal{A}} \left\{ \rho_\pi^i(s,a) \frac{\lambda_j(s)}{2N} \log \frac{\rho_\pi^i(s,a)}{\rho_\pi^i(s,a)+\pi_E^j(a|s)\mu_\pi^i(s)} \right.$$

$$\left. + \pi_E^j(a|s)\mu_\pi^i(s) \frac{\lambda_j(s)}{2N} \log \frac{\pi_E^j(a|s)\mu_\pi^i(s)}{\rho_\pi^i(s,a)+\pi_E^j(a|s)\mu_\pi^i(s)} \right\} + \frac{\log 2}{1-\gamma}$$

$$\overset{(c)}{=} \min_\pi \sum_{i=1}^N \sum_{j=1}^N \max_{D_{ij}} \left\{ \mathbb{E}_{(s,a)\sim\rho_\pi^i} \left[\frac{\lambda_j(s)}{2N} \log(1 - D_{ij}(s,a)) \right] + \mathbb{E}_{s\sim\mu_\pi^i, a\sim\pi_E^j(\cdot|s)} \left[\frac{\lambda_j(s)}{2N} \log(D_{ij}(s,a)) \right] \right\} + \frac{\log 2}{1-\gamma},$$

where (a) holds by the definition of $\lambda_j(s)$, (b) holds due to Lemma A.1 below with the condition of $\gamma < 1$, (c) holds due to $D_{ij} \in [0,1]$; for any non-negative $(a,b) \in \mathbb{R}^2\backslash\{0,0\}$, the function $f \to a\log(f) + b\log(1-f)$ has maximum at $\frac{a}{a+b}$ in $[0,1]$. Thus if we represent $\rho_\pi^i(s,a) \cdot \lambda_j(s)/2N$ and $\mu_\pi^i(s) \cdot \pi_E^j(a|s) \cdot \lambda_j(s)/2N$ as $g(s,a)$ and $h(s,a)$ respectively, then we have the optimal solution $D_{ij}^*(s,a) = \frac{h(s,a)}{g(s,a)+h(s,a)} = \frac{\pi_E^j(a|s)\mu_\pi^i(s)}{\rho_\pi^i(s,a)+\pi_E^j(a|s)\mu_\pi^i(s)} = \frac{\pi_E^j(a|s)\mu_\pi^i(s)}{\pi(a|s)\mu_\pi^i(s)+\pi_E^j(a|s)\mu_\pi^i(s)} = \frac{\pi_E^j(a|s)}{\pi(a|s)+\pi_E^j(a|s)}.$ $\square$

Lemma A.1 (Lemma for proof of Theorem 5.1). *Let $f_T^i(s) = \sum_{t=0}^{T} \gamma^t Pr(s_t = s|\pi, \mathcal{P}^i)$ and $\gamma \in (0,1)$. Then, we have*

$$\int_{s \in \mathcal{S}} \mu_\pi^i(s) = \frac{1}{1-\gamma}$$

Therefore, $\int_{s \in \mathcal{S}} \frac{1}{N} \sum_{i=1}^{N} \mu_\pi^i(s) = \frac{1}{N} \sum_{i=1}^{N} \int_{s \in \mathcal{S}} \mu_\pi^i(s) = \frac{1}{1-\gamma}$.

Proof. For fixed s and i, $0 \leq \Pr(s_t = s|\pi, \mathcal{P}^i) \leq 1$ because it is a probability. Since $\gamma < 1$, we have

$$f_T^i(s) = \sum_{t=0}^{T} \gamma^t \Pr(s_t = s|\pi, \mathcal{P}^i) \leq \sum_{t=0}^{T} \gamma^t < \sum_{t=0}^{\infty} \gamma^t = \frac{1}{1-\gamma} < \infty.$$

Also, by the definition of the discount factor γ mentioned in Section 3.1, its condition $0 < \gamma < 1$, which implies that $\{f_T^i(s)\}$ is a non-negative and monotone increasing sequence of positive measures with respect to T. Hence, by the monotone convergence theorem (Theorem 1.5.7 in (Durrett, 2019)), $\lim_{T \to \infty} \int_{s \in \mathcal{S}} f_T^i(s) = \int_{s \in \mathcal{S}} \lim_{T \to \infty} f_T^i(s)$. Therefore, we have

$$\int_{s \in \mathcal{S}} \mu_\pi^i(s) = \int_{s \in \mathcal{S}} \lim_{T \to \infty} f_T^i(s) = \lim_{T \to \infty} \int_{s \in \mathcal{S}} f_T^i(s)$$

$$= \lim_{T \to \infty} \int_{s \in \mathcal{S}} \sum_{t=0}^{T} \gamma^t \Pr(s_t = s|\pi, \mathcal{P}^i)$$

$$= \lim_{T \to \infty} \sum_{t=0}^{T} \gamma^t \int_{s \in \mathcal{S}} \Pr(s_t = s|\pi, \mathcal{P}^i)$$

$$= \lim_{T \to \infty} \sum_{t=0}^{T} \gamma^t = \frac{1}{1-\gamma},$$

where $N, T \in \mathbb{N}$. □

A.2. Proof of Theorem 5.2

Theorem 5.2 can be rewritten as follows:

Theorem 5.2 The two following maximizing problems have the same optimal solution.

$$\max_{D_{ij}} \left\{ \mathbb{E}_{(s,a) \sim \rho_\pi^i} \left[\lambda_j(s) \log(1 - D_{ij}(s,a)) \right] + \mathbb{E}_{(s,a) \sim \rho_E^j} \left[\lambda_j(s) \log(D_{ij}(s,a)) \right] \right\} \tag{12}$$

$$\max_{D_{ij}} \left\{ \mathbb{E}_{(s,a) \sim \rho_\pi^i} \left[\log(1 - D_{ij}(s,a)) \right] + \mathbb{E}_{(s,a) \sim \rho_E^j} \left[\log(D_{ij}(s,a)) \right] \right\}. \tag{13}$$

Proof.

$$(12) = \int_{(s,a) \in \mathcal{S} \times \mathcal{A}} \rho_\pi^i(s,a) \lambda_j(s) \log(1 - D_{ij}(s,a)) + \rho_E^j(s,a) \lambda_j(s) \log(D_{ij}(s,a))$$

$$(13) = \int_{(s,a) \in \mathcal{S} \times \mathcal{A}} \rho_\pi^i(s,a) \log(1 - D_{ij}(s,a)) + \rho_E^j(s,a) \log(D_{ij}(s,a)).$$

For any non-negative $(a,b) \in \mathbb{R}^2 \setminus \{0,0\}$, the function $f \to a\log(f) + b\log(1-f)$ has maximum at $\frac{a}{a+b}$ in $[0,1]$. $\rho_\pi^i(s,a) \lambda_j(s)$, $\rho_E^j(s,a) \lambda_j(s)$ can be represented as $g(s,a)$ and $h(s,a)$, respectively. Therefore, the optimal solution of (12) $D_{ij}^*(s,a)$ becomes $\frac{h(s,a)}{g(s,a)+h(s,a)} = \frac{\rho_E^j(s,a)}{\rho_\pi^i(s,a)+\rho_E^j(s,a)}$, which is the same as the optimal solution of (13). □

B. Detailed Descriptions

B.1. Description for Reproduced Expert Policy

In order to optimize (6), expert policies π_E^j, $j = 1, \cdots, N$ are required. However, π_E^j's are not available explicitly to us, but we can use expert demonstration τ_E^j, which is in form of state-action pairs generated by the expert policy π_E^j in the j-th demonstration environment $\mathcal{E}_j$. In this section, we evaluate an algorithm with the objective function (6) in the main paper. In order to compute the second term $\mathbb{E}_{s \sim \mu_\pi^i, a \sim \pi_E^j}[\cdot]$ in the objective function, we reproduce the expert policy $\pi_E^j(\cdot|s)$ by behavior cloning (BC) and GAIL+GP mentioned in Section 3.2 by using the given expert demonstration. Before we optimize the objective function, each expert policy π_E^j is first trained in the j-th demonstration environment $\mathcal{E}_j$ by using the j-th expert demonstration τ_E^j.

With the above experimental setup, we tested the case $N = 1$ of the objective function (6) as follows:

$$\min_\pi \max_{D_{11}} \left\{ \mathbb{E}_{(s,a) \sim \rho_\pi^1} \left[\frac{\lambda_1(s)}{2} \log(1 - D_{11}(s,a)) \right] + \mathbb{E}_{s \sim \mu_\pi^1, a \sim \pi_E^1} \left[\frac{\lambda_1(s)}{2} \log(D_{11}(s,a)) \right] \right\} + \frac{\log 2}{1 - \gamma}, \qquad (14)$$

where $\lambda_1(s)$ is equal to 1 by the definition of $\lambda_j(s)$. This setting is SNE/SNE. The agent policy is trained in the nominal interaction environment, and the expert π_E^1 is also trained in the same environment. We evaluated the corresponding performance with 10 random seeds. Figure 6 shows the results of the mean returns of both the expert's and the agent's policies in the nominal test environment. In most cases, the agent policy either has almost the same performance as the expert policy or totally fails to learn. Thus, learning is unstable. It implies that if the reproduced expert policy $\hat{\pi}_E^j$ covers the states induced by the agent policy π, then the agent policy can work well as the expert. On the other hand, if the reproduced expert $\hat{\pi}_E^j$ does not cover the states of the agent policy, then the agent policy fails to learn for the given task.

In practice, it is highly likely that we will have an expert demonstration that covers only a limited region of the entire state-action space. Furthermore, the reproduced expert policy by an IL method would visit a limited region of the entire state space during the training phase. These two reasons can cause extrapolation error. Due to this error, the reproduced expert policy may sample an action that seems to be a non-expert action for a given state. This inappropriate action will give incorrect information to the agent policy.

| (a) Hopper: | (b) Walker2d: | (c) HalfCheetah: | (d) Ant: |
| reproduced expert via BC | reproduced expert via BC | reproduced expert via BC | reproduced expert via BC |

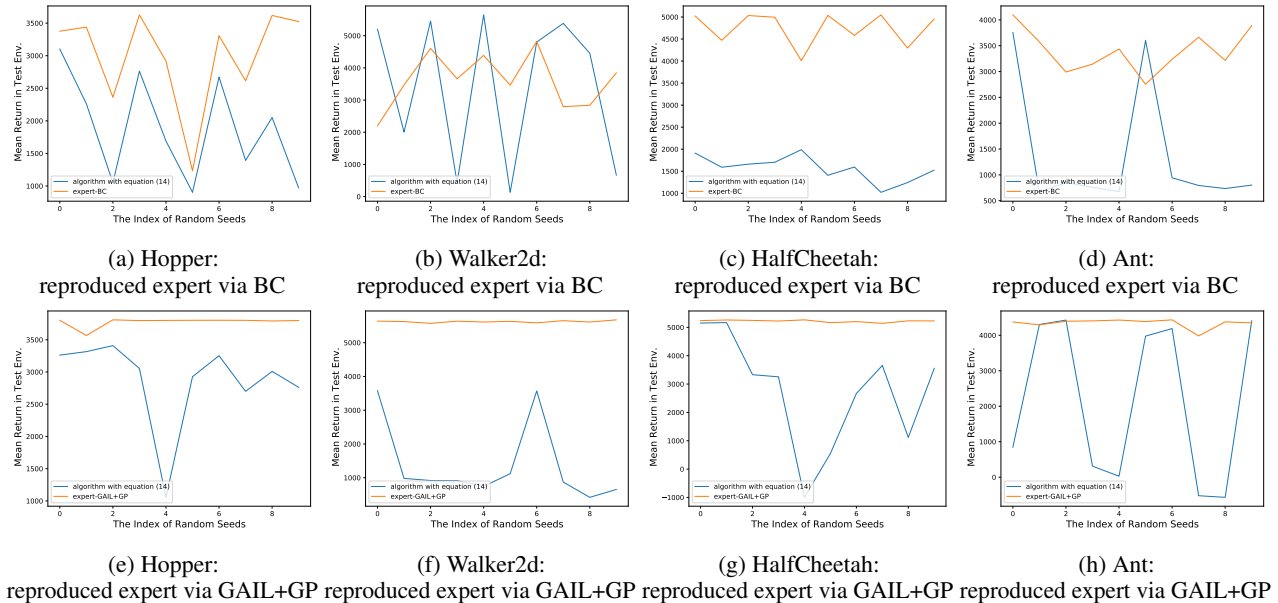| (e) Hopper: | (f) Walker2d: | (g) HalfCheetah: | (h) Ant: |
| reproduced expert via GAIL+GP | reproduced expert via GAIL+GP | reproduced expert via GAIL+GP | reproduced expert via GAIL+GP |

Figure 6: Mean return performance of IL algorithm solving (14) with the reproduced expert policy $\hat{\pi}_E^j$. The x-axis is the index of 10 random seeds and the y-axis is the mean return. The orange lines - the performance of reproduced experts, and the blue line - IL algorithm solving (14) with the reproduced expert policy $\hat{\pi}_E^j$.

B.2. Description for Occupancy measure Matching in Multiple Environments (OMME)

Equation (10) in the main paper is rewritten here as

$$\min_{\pi} \sum_{j=1}^{N} \lambda_j \mathcal{D}_{JS}(\bar{\rho}_{\pi}, \bar{\rho}_E^j), \tag{15}$$

where $\sum_j \lambda_j = 1$. We assume $\gamma \in (0, 1)$, and as in (Ho & Ermon, 2016; Wu et al., 2019), $\bar{\rho}_{\pi}^i = (1 - \gamma)\rho_{\pi}^i$ and $\bar{\rho}_E^j = (1 - \gamma)\rho_E^j$ are the normalized occupancy distributions from π in $\mathcal{E}_i$ and π_E^j.

Then, we have

$$\min_{\pi} \sum_{j=1}^{N} \lambda_j \mathcal{D}_{JS}(\bar{\rho}_{\pi}, \bar{\rho}_E^j) = \min_{\pi} \sum_{j=1}^{N} \lambda_j \mathcal{D}_{JS}(\frac{1}{N} \sum_{i=1}^{N} \bar{\rho}_{\pi}^i, \bar{\rho}_E^j)$$

$$\overset{(a)}{\leq} \min_{\pi} \sum_{j=1}^{N} \lambda_j \frac{1}{N} \sum_{i=1}^{N} \mathcal{D}_{JS}(\bar{\rho}_{\pi}^i, \bar{\rho}_E^j) = \min_{\pi} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\lambda_j}{N} \mathcal{D}_{JS}(\bar{\rho}_{\pi}^i, \bar{\rho}_E^j)$$

$$= \min_{\pi} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\lambda_j}{2N} \left\{ \int_{(s,a) \in \mathcal{S} \times \mathcal{A}} \bar{\rho}_{\pi}^i(s,a) \log \frac{2\bar{\rho}_{\pi}^i(s,a)}{\bar{\rho}_{\pi}^i(s,a) + \bar{\rho}_E^j(s,a)} + \bar{\rho}_E^j(s,a) \log \frac{2\bar{\rho}_E^j(s,a)}{\bar{\rho}_{\pi}^i(s,a) + \bar{\rho}_E^j(s,a)} \right\}$$

$$= \min_{\pi} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\lambda_j}{2N} \left\{ \int_{(s,a) \in \mathcal{S} \times \mathcal{A}} \bar{\rho}_{\pi}^i(s,a) \log \frac{\bar{\rho}_{\pi}^i(s,a)}{\bar{\rho}_{\pi}^i(s,a) + \bar{\rho}_E^j(s,a)} + \bar{\rho}_E^j(s,a) \log \frac{\bar{\rho}_E^j(s,a)}{\bar{\rho}_{\pi}^i(s,a) + \bar{\rho}_E^j(s,a)} \right\}$$

$$+ \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\lambda_j}{2N} \left\{ \int_{(s,a) \in \mathcal{S} \times \mathcal{A}} \bar{\rho}_{\pi}^i(s,a) + \bar{\rho}_E^j(s,a) \right\} \log 2$$

$$\overset{(b)}{=} \min_{\pi} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\lambda_j}{2N} \left\{ \int_{(s,a) \in \mathcal{S} \times \mathcal{A}} \bar{\rho}_{\pi}^i(s,a) \log \frac{\bar{\rho}_{\pi}^i(s,a)}{\bar{\rho}_{\pi}^i(s,a) + \bar{\rho}_E^j(s,a)} + \bar{\rho}_E^j(s,a) \log \frac{\bar{\rho}_E^j(s,a)}{\bar{\rho}_{\pi}^i(s,a) + \bar{\rho}_E^j(s,a)} \right\} + \log 2$$

$$= \min_{\pi} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\lambda_j(1-\gamma)}{2N} \left\{ \int_{(s,a) \in \mathcal{S} \times \mathcal{A}} \rho_{\pi}^i(s,a) \log \frac{\rho_{\pi}^i(s,a)}{\rho_{\pi}^i(s,a) + \rho_E^j(s,a)} + \rho_E^j(s,a) \log \frac{\rho_E^j(s,a)}{\rho_{\pi}^i(s,a) + \rho_E^j(s,a)} \right\} + \log 2$$

$$= \min_{\pi} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\lambda_j(1-\gamma)}{2N} \max_{D_{ij}} \left\{ \mathbb{E}_{(s,a) \sim \rho_{\pi}^i} [\log(1 - D_{ij}(s,a))] + \mathbb{E}_{(s,a) \sim \rho_E^j} [\log D_{ij}(s,a)] \right\} + \log 2,$$

where (a) holds by the convexity of the Jensen-Shannon divergence, (b) holds by the definition of λ_j.

C. Algorithm: Robust Imitation Learning against Variations in Environment Dynamics

---

Algorithm 1 Robust Imitation learning with Multiple perturbed Environments (RIME)

Input: The number of sampled environments N, sampled environments $\mathcal{E}_1, \ldots, \mathcal{E}_N$, expert demonstrations $\tau_E^1, \ldots, \tau_E^N$, policy parameter θ, parameter of discriminators $\{\phi_{ij}\}$, the number of learning iterations n_{epoch}, the weight of GP κ.
Initialize all parameters θ, $\{\phi_{ij}\}$.
for $k = 1$ **to** n_{epoch} **do**
   for $i = 1$ **to** N **do**
      Sample trajectories $\tau_\pi^i \sim \pi_\theta$ in $\mathcal{E}_i$
      for $j = 1$ **to** N **do**
         Update the discriminator $D_{\phi_{ij}}$ by maximizing (9)
      end for
   end for
   for $i = 1$ **to** N **do**
      Update the policy π_θ by minimizing (8) using PPO
   end for
end for

D. Ablation Studies

D.1. Ablation Study for an Algorithm Trained in the SNE/MPE Setting

To see the effect of interacting with MPE, we evaluated SNEMPE-max described in Section 4 in the perturbed test environments. This algorithm is obtained by simply applying the robust RL principle to the IL setting. Furthermore, it is a variant of our algorithm (7) applied to the SNE/MPE setting.

We used three expert demonstrations which are generated by their experts in demonstration environments with perturbations $050\%\zeta_0$, $100\%\zeta_0$, $150\%\zeta_0$, where ζ_0 is the nominal dynamics value. With three expert demonstrations, we trained this algorithm in the nominal interaction environment with ζ_0. It has discriminators D_{1j}, and the objective function for the discriminator D_{1j} is the same as our discriminator's objective function (9). The objective function for the policy is given by

$$
\min_\pi \mathbb{E}_{(s,a)\sim\rho^1_\pi} \left[\max_j \log(1 - D_{1j}(s,a)) \right]. \tag{16}
$$

Figure 7 shows that SNEMPE-max fails when the underlying environment dynamics are perturbed from those of the interaction environment. It is seen that SNEMPE-max trained in a single interaction environment cannot properly capture the diverse dynamics of multiple demonstration environments.

(a) Hopper+Gravity   (b) Hopper+Mass   (c) Walker2d+Gravity   (d) Walker2d+Mass

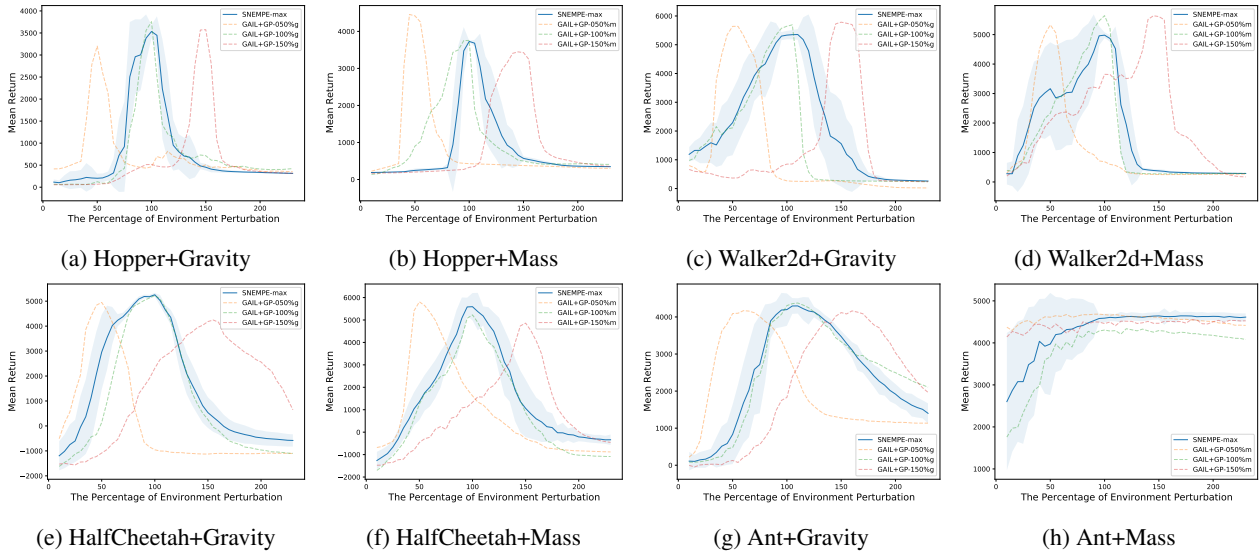(e) HalfCheetah+Gravity   (f) HalfCheetah+Mass   (g) Ant+Gravity   (h) Ant+Mass

Figure 7: The performance of SNEMPE-max over environment perturbations.

D.2. Ablation Study with Importance Sampling Ratio $\frac{\mu_\pi^i(s)}{\mu_E^j(s)}$ Estimator

To exactly compute $\mu_\pi^i(s)$ in the importance sampling ratio $\frac{\mu_\pi^i(s)}{\mu_E^j(s)}$ in (7), we need many interactions with the interaction environment $\mathcal{E}_i$, which could increase the sample complexity in practice. To avoid this sample complexity issue, we can estimate the ratio $\frac{\mu_\pi^i(s)}{\mu_E^j(s)}$ directly. With an estimated ratio $\frac{\mu_\pi^i(s)}{\mu_E^j(s)}$, eq. (7) is replaced with

$$\min_\pi \sum_{i=1}^N \sum_{j=1}^N \max_{D_{ij}} \left\{ \mathbb{E}_{\rho_\pi^i} \left[\lambda_j(s) \log(1 - D_{ij}(s,a)) \right] + \mathbb{E}_{\rho_E^j} \left[\tilde{w}_{ij}(s) \lambda_j(s) \log(D_{ij}(s,a)) \right] \right\}, \tag{17}$$

where $\tilde{w}_{ij}(s)$ is a given estimator of the ratio $\frac{\mu_\pi^i(s)}{\mu_E^j(s)}$. So, the policy in (17) affects only the first term $\mathbb{E}_{\rho_\pi^i}[\cdot]$ and hence the objective function for the policy update is the same as (8). The objective function for the discriminator D_{ij} is given by

$$\max_{D_{ij}} \mathbb{E}_{\rho_\pi^i} \left[\log(1 - D_{ij}(s,a)) \right] + \mathbb{E}_{\rho_E^j} \left[\tilde{w}_{ij}(s) \log(D_{ij}(s,a)) \right] \tag{18}$$

In the same way as in Theorem 5.2, the optimal discriminator is given by $D_{ij}^* = \frac{\tilde{w}_{ij}(s) \lambda_j(s) \rho_E^j(s,a)}{\lambda_j(s) \rho_\pi^i(s,a) + \tilde{w}_{ij}(s) \lambda_j(s) \rho_E^j(s,a)} = \frac{\tilde{w}_{ij}(s) \rho_E^j(s,a)}{\rho_\pi^i(s,a) + \tilde{w}_{ij}(s) \rho_E^j(s,a)}$.
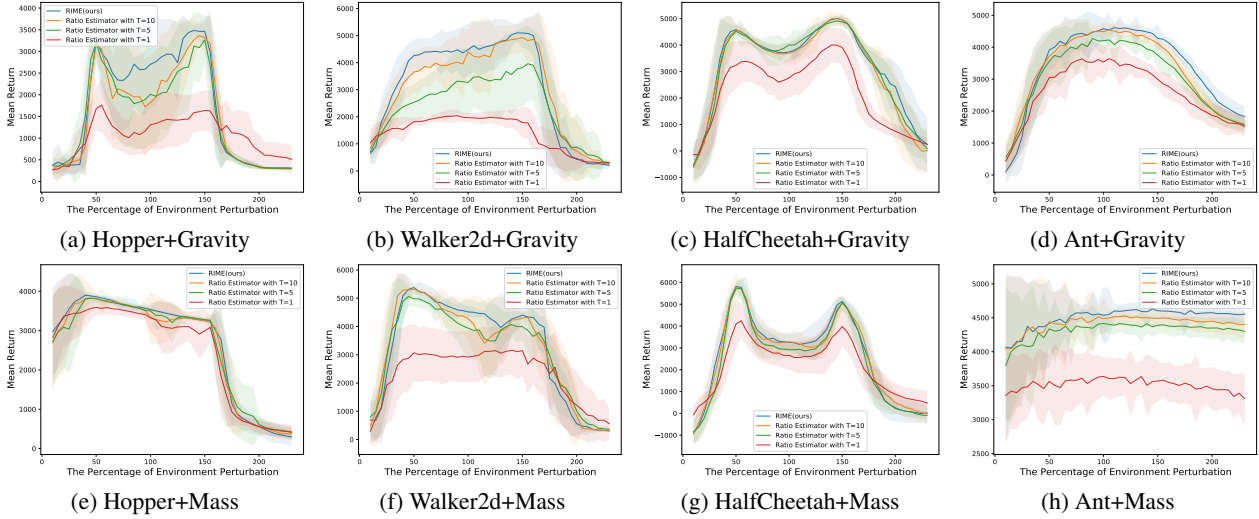

Figure 8: Comparison with LFIW method in the $N = 2$ sampled environments case

Now we explain how to estimate the ratio $\tilde{w}_{ij}(s) := \frac{\mu_\pi^i(s)}{\mu_E^j(s)}$ by a method of estimating probability measure ratio proposed in (Sinha et al., 2020). They proposed the Likelihood-Free Importance Weights (LFIW) method, which estimates the ratio of two probability measures by using the lower bound of f-divergence between the two measures. They showed that in (19), the equality holds at $w = \frac{dP}{dQ}$, so they estimated the probability measure ratio $w(x)$ by maximizing (19):

$$D_f(P||Q) \geq \mathbb{E}_P \left[f'(w(x)) \right] - \mathbb{E}_Q \left[f^*(f'(w(x))) \right], \tag{19}$$

where P and Q are probability measures, and D_f is an f-divergence. However, directly using the probability measure ratio $w(x)$ may cause learning failure due to the finite sample size issue in practice. To address this issue, the LFIW applies the self-normalization to the probability measures ratio $w(x)$ over Q with a temperature hyperparameter T.

$$\tilde{w}(x) = \frac{w(x)^{1/T}}{\mathbb{E}_Q \left[w(x)^{1/T} \right]} \tag{20}$$

By replacing P and Q with μ_π^i and μ_E^j, we can estimate the importance sampling ratio $\frac{\mu_\pi^i(s)}{\mu_E^j(s)}$. Figure 8 shows that our proposed method (RIME) which simply sets $\frac{\mu_\pi^i(s)}{\mu_E^j(s)}$ to 1 has almost same performance as the proposed method using the estimated importance sampling ratio by LFIW for all tasks.

D.3. Ablation Study with State-only Expert Demonstration

GAIfO (Torabi et al., 2018b) uses state-only expert demonstration and reproduces the expert policy π_E by matching the state-transition occupancy measures induced by the π and π_E. Our algorithm (RIME) and other GAIL variant algorithms can directly be applied to this setting by using state-only expert demonstration instead of state-action expert demonstration. We refer to these methods as GAIfO-RIME, GAIfO-OMME, GAIfO-mixture, GAIfO-single.

We tested these GAIfO variants in the $N = 2$ sampled environment case (50% and 150%). Table 4 and Figure 9 show similar results to Table 1 and Figure 3 (the case with the state-action expert demonstration) for all the tasks except for Walker2d+Gravity and Ant. For Walker2d+Gravity, GAIfO-mixture and GAIfO-single have good performance around the interaction environments, but they are over-fitted to these environments and do not perform near the test environment with ζ_0. On the other hand, our method (GAIfO-RIME) performs well near the test environment with ζ_0. Therefore, the experimental results show that our method can properly recover the experts' preference over the state space. In the case of Ant+Gravity and Ant+Mass, all algorithms failed to learn, and we think this is due to the difficulty of optimization due to the large state space of the Ant task.

Table 4: Mean return / minimum return of GAIfO variants over the dynamics parameter range [50%,150%] in the $N = 2$ sampled environment case

Algorithm	Hopper +Gravity	Walker2d +Gravity	HalfCheetah +Gravity
GAIfO-RIME (ours)	**2758.9 / 2318.4**	**3767.7 / 3331.5**	**4247.9 / 3786.1**
GAIfO-OMME	1491.4 / 1063.3	2918.1 / 1935.6	3600.4 / 3152.0
GAIfO-mixture	1424.8 / 719.6	3551.4 / 1808.0	3446.6 / 2851.8
GAIfO-single	1376.6 / 765.7	3346.0 / 1384.0	3208.0 / 2252.2

Algorithm	Hopper +Mass	Walker2d +Mass	HalfCheetah +Mass
GAIfO-RIME (ours)	**3496.5 / 3335.5**	**4757.3 / 4336.3**	**4247.6 / 3611.5**
GAIfO-OMME	2937.7 / 2421.3	4154.2 / 3657.6	3906.5 / 3172.2
GAIfO-mixture	3188.3 / 2953.5	3872.0 / 2769.2	3239.8 / 2670.5
GAIfO-single	2269.2 / 1622.9	3478.5 / 1582.6	2947.1 / 1653.3

(a) Hopper+Gravity   (b) Walker2d+Gravity   (c) HalfCheetah+Gravity

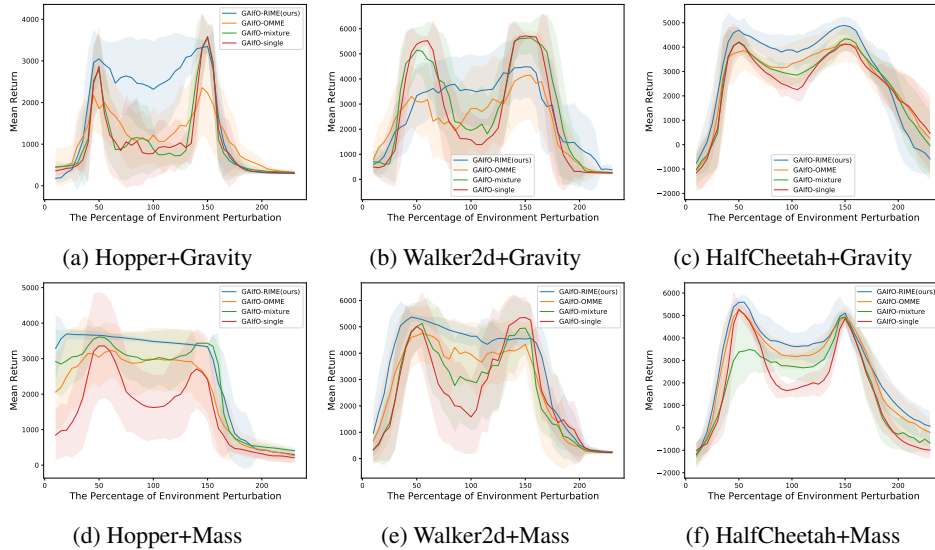(d) Hopper+Mass   (e) Walker2d+Mass   (f) HalfCheetah+Mass

Figure 9: Comparisons with variants of GAIfO in the $N = 2$ sampled environments case

D.4. Ablation Study according to the Size of Expert Demonstration

Considering the fact that expert demonstrations are costly to obtain, we tested our algorithm by reducing the amount of expert demonstration from the 50 trajectories (each trajectory with 1000 samples).

As seen in Table 5, for Hopper+Gravity, the robustness of our algorithm decreases as the size of expert demonstration decreases. However, for Ant+Gravity and Walker2d+Mass and HalfCheetah+Gravity and Ant+Mass, our algorithm using the reduced amount of expert demonstration still performs well. It seems that the amount of demonstration above a threshold is sufficient.

Table 5: Mean return / minimum return over the dynamics parameter range [50%,150%] in the $N = 2$ sampled environment case with expert demonstrations with various size

# of expert trajectories	Hopper +Gravity	Walker2d +Gravity	HalfCheetah +Gravity	Ant +Gravity
50	2886.7 / 2332.4	4577.1 / 4260.9	4268.9 / 3712.0	4402.2 / 3909.9
25	2774.7 / 2021.0	4455.5 / 4044.1	4290.3 / 3720.9	4445.5 / 3666.5
10	2570.8 / 1811.5	4243.1 / 3529.8	4244.5 / 3622.7	4554.7 / 4038.6
5	2323.2 / 1754.2	4514.0 / 3906.0	4219.5 / 3578.2	4562.7 / 3985.9

# of expert trajectories	Hopper +Mass	Walker2d +Mass	HalfCheetah +Mass	Ant +Mass
50	3535.7 / 3255.6	4597.0 / 3965.8	3959.0 / 3156.7	4554.5 / 4417.5
25	3510.4 / 3246.3	4608.7 / 4134.2	4185.2 / 3320.4	4602.1 / 4395.6
10	3352.2 / 2895.2	4552.2 / 3937.8	4025.4 / 3222.4	4662.6 / 4524.2
5	3365.5 / 2965.8	4614.4 / 4073.2	3698.9 / 2718.1	4679.9 / 4496.8

(a) Hopper+Gravity (b) Walker2d+Gravity (c) HalfCheetah+Gravity (d) Ant+Gravity

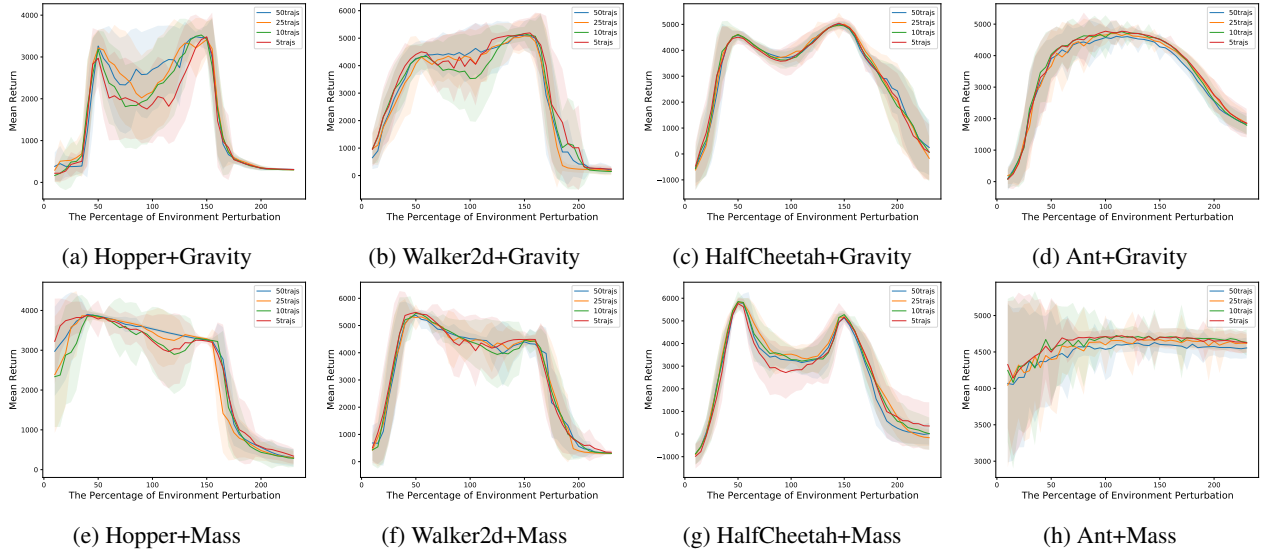(e) Hopper+Mass (f) Walker2d+Mass (g) HalfCheetah+Mass (h) Ant+Mass

Figure 10: The performance of RIME trained in the $N = 2$ sampled environment setting with various sizes of expert demonstrations.

E. Additional Experimental Results

E.1. Results in the $N = 2$ Sampled Environment Setting ($50\%\zeta_0$, $150\%\zeta_0$)

Here we provide all result plots in the 2 sampled environment setting for our algorithm and the baseline algorithms.

(a) Hopper+Gravity: performance

(b) Hopper+Gravity: comparisons

(c) Hopper+Mass: performance

(d) Hopper+Mass: comparisons

(e) Walker2d+Gravity: performance

(f) Walker2d+Gravity: comparisons

(g) Walker2d+Mass: performance

(h) Walker2d+Mass: comparisons

(i) HalfCheetah+Gravity: performance

(j) HalfCheetah+Gravity: comparisons

(k) HalfCheetah+Mass: performance

(l) HalfCheetah+Mass: comparisons

(m) Ant+Gravity: performance

(n) Ant+Gravity: comparisons
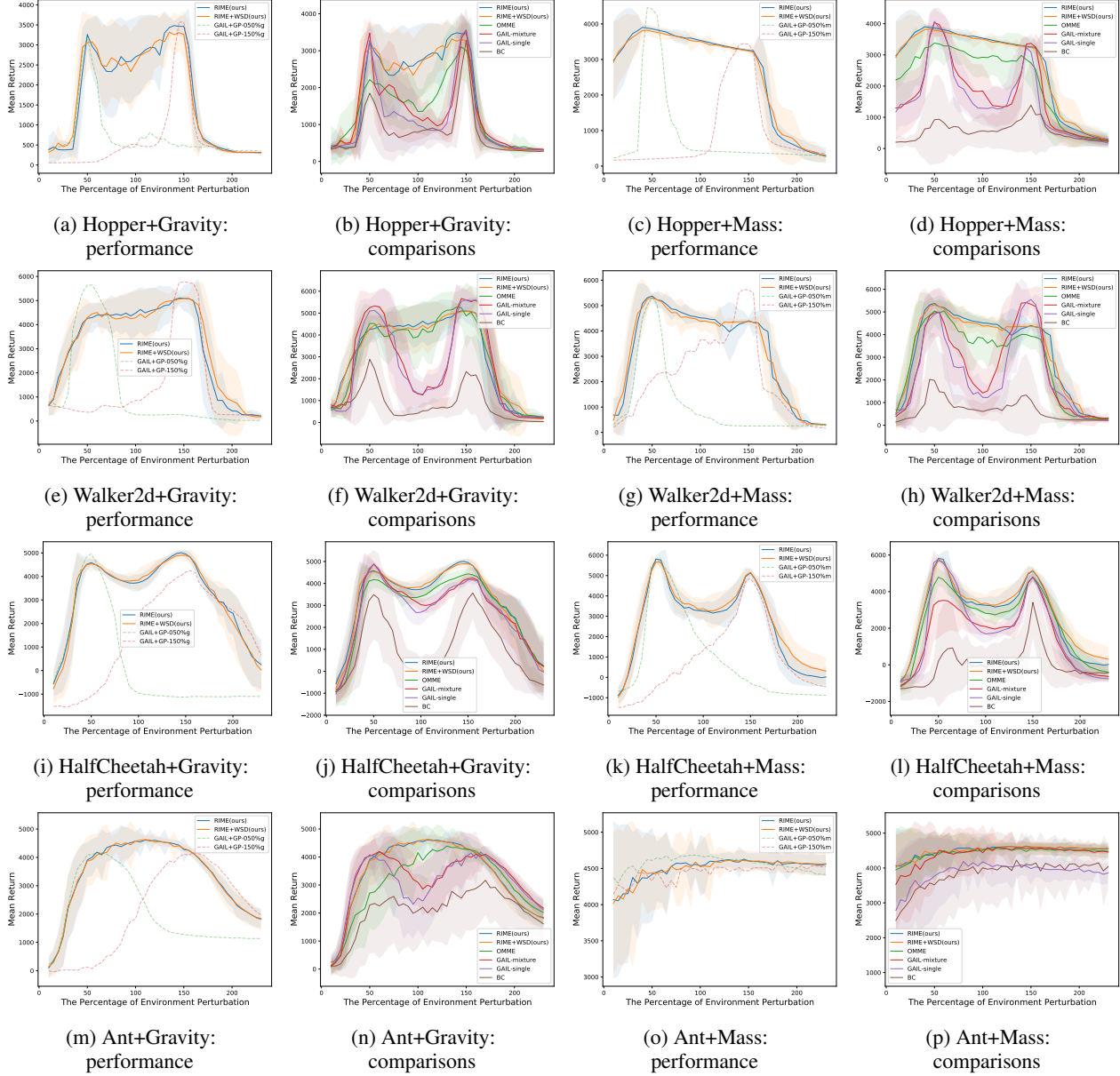
(o) Ant+Mass: performance

(p) Ant+Mass: comparisons

Figure 11: All experimental results in the $N = 2$ sampled environment setting ($50\%\zeta_0$, $150\%\zeta_0$).

E.2. Results in the $N = 3$ Sampled Environment Setting ($050\%\zeta_0$, $100\%\zeta_0$, $150\%\zeta_0$)

Here we provide all result plots in the 3 sampled environment setting for our algorithm and the baseline algorithms.

(a) Hopper+Gravity: performance

(b) Hopper+Gravity: comparisons

(c) Hopper+Mass: performance

(d) Hopper+Mass: comparisons

(e) Walker2d+Gravity: performance

(f) Walker2d+Gravity: comparisons

(g) Walker2d+Mass: performance

(h) Walker2d+Mass: comparisons

(i) HalfCheetah+Gravity: performance

(j) HalfCheetah+Gravity: comparisons

(k) HalfCheetah+Mass: performance

(l) HalfCheetah+Mass: comparisons

(m) Ant+Gravity: performance

(n) Ant+Gravity: comparisons

(o) Ant+Mass: performance

(p) Ant+Mass: comparisons
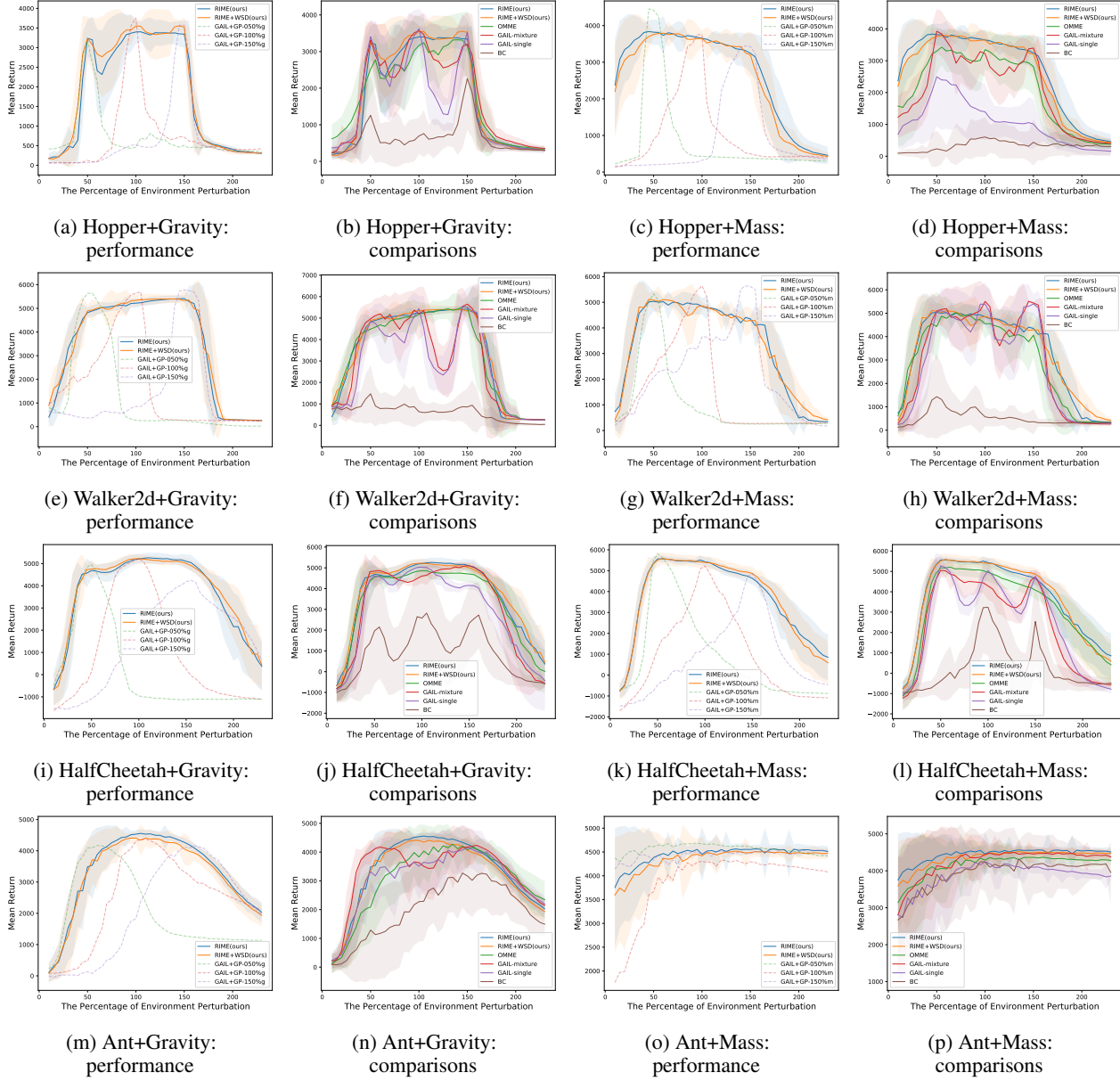
Figure 12: All experimental results in the $N = 3$ sampled environment setting ($50\%\zeta_0$, $100\%\zeta_0$, $150\%\zeta_0$)

E.3. Results in the 2-D Perturbation Case

Here we provide all result plots in the 2-D perturbation case for our algorithm and the baselines.

(a) Hopper+2-dim(G&M)

(b) Walker2d+2-dim(G&M)
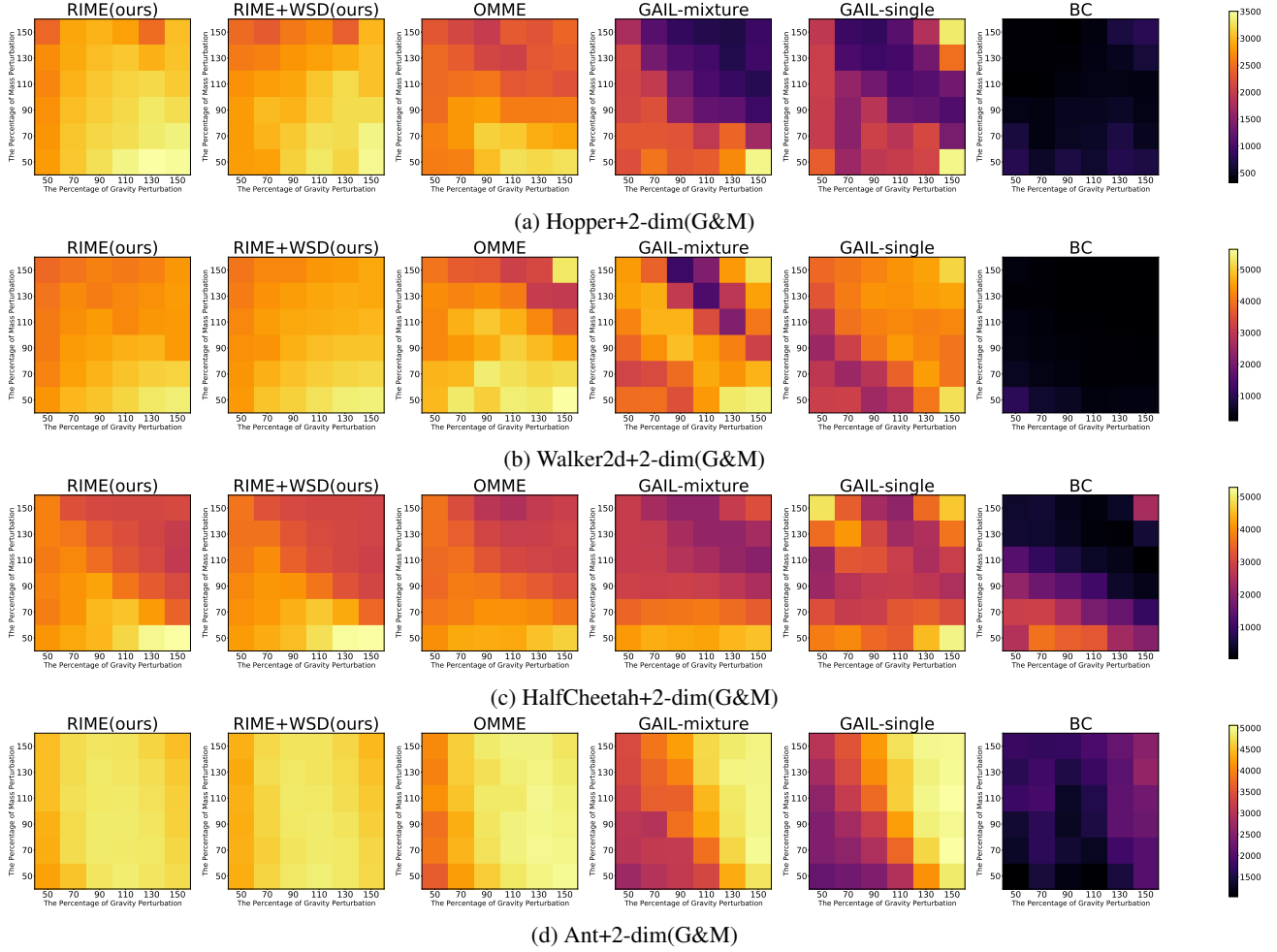

(c) HalfCheetah+2-dim(G&M)

(d) Ant+2-dim(G&M)

Figure 13: Experimental results in the $N = 4$ sampled environments case with 2-dim perturbation parameters (Gravity+Mass)

F. Experimental Information

F.1. Detailed Description of Other IL Baselines

We considered the following 4 IL baselines in Section 6. Here is the detailed description of these IL baselines.

1. Behavior Cloning (BC): All expert demonstrations are split out 70% training dataset and 30% validation dataset. The policy is trained by supervised learning until validation errors of each expert demonstration stop decreasing.

2. GAIL-mixture: It is a variant of GAIL+GP applied directly to the multiple interaction environment setting. There is a single discriminator, and this discriminator distinguishes between all $\bar{\rho}_\pi^i$'s and all $\bar{\rho}_E^j$'s. The objective function of this algorithm is (21), and the objective function without GP term can be reduced to $\min_\pi \mathcal{D}_{JS}(\sum_i \bar{\rho}_\pi^i/N, \sum_j \bar{\rho}_E^j/N)$. It minimizes the divergence between the mixture of the normalized occupancy distributions of the policy and the experts so that the mixtures are close. Thus, we call this algorithm GAIL-mixture.

$$\min_\pi \max_D \sum_{i=1}^N \left\{ \mathbb{E}_{\rho_\pi^i} \left[\log(1 - D(s,a)) \right] + \mathbb{E}_{\rho_E^i} \left[\log(D(s,a)) \right] + \kappa \mathbb{E}_{\hat{x}} \left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1 \right)^2 \right\} \tag{21}$$

3. GAIL-single: It is another variant of GAIL+GP applied directly to the multiple interaction environment setting. Unlike GAIL-mixture, there are multiple discriminators. The objective function of this algorithm is (22), and the objective function without GP term can be reduced to $\min_\pi \sum_i \mathcal{D}_{JS}(\bar{\rho}_\pi^i, \bar{\rho}_E^i)$. It minimizes the divergence between $\bar{\rho}_\pi^i$ and $\bar{\rho}_E^i$, which makes ρ_π^i close to ρ_E^i, for each i. Thus, we call it GAIL-single.

$$\min_\pi \sum_{i=1}^N \max_{D_i} \left\{ \mathbb{E}_{\rho_\pi^i} \left[\log(1 - D_i(s,a)) \right] + \mathbb{E}_{\rho_E^i} \left[\log(D_i(s,a)) \right] + \kappa \mathbb{E}_{\hat{x}} \left(\|\nabla_{\hat{x}} D_i(\hat{x})\|_2 - 1 \right)^2 \right\} \tag{22}$$

4. Occupancy measures Matching in Multiple Environments (OMME): This algorithm is a method obtained by matching occupancy measures in a different way from GAIL-mixture and GAIL-single. As mentioned in Section 5.4 & Appendix B.2, if we match occupancy measures, the objective function for the policy would be $\min_\pi \sum_i \max_j \mathbb{E}_{\rho_\pi^i} [\log(1 - D_{ij})]$, not $\min_\pi \sum_i \mathbb{E}_{\rho_\pi^i} [\max_j \log(1 - D_{ij})]$. Except the objective function for the policy, this algorithm is the same as our algorithm.

F.2. Model Architecture

We developed our code based on (Kostrikov, 2018). In our experiments, we used MLP that consists of two layers with 64 cells in each layer, and this network is used for the policy. For the discriminators, we used MLP that consists of two layers with 100 cells in each layer. We used PPO as the algorithm for updating the policy. The batch size is set to 2048, the number of update epochs for the policy at one iteration is set to 4, and the number of update epochs for the discriminator at one iteration is set to 5. Finally, the coefficient of the GP term is set to 10, and the coefficient of entropy for PPO is 0. The rest of the hyper-parameters are the same as those in (Schulman et al., 2017; 2015b).

F.3. Environments & Experts

Table 6: Interaction Environments & Expert Demonstrations

Task	Observation Space	Action Space	Environment Perturbation	Expert Performance
Hopper-v2	11 (Continuous)	3 (Continuous)	100%g & 100%m 50%g (Gravity) 150%g 50%m (Mass) 150%m	3817.7 ± 21.9 3717.0 ± 72.1 3620.1 ± 4.3 4415.7 ± 16.6 3442.0 ± 2.5
Walker2d-v2	17 (Continuous)	6 (Continuous)	100%g & 100%m 50%g 150%g 50%m 150%m	5617.6 ± 16.3 5612.4 ± 22.1 5791.8 ± 46.1 5359.0 ± 118.9 5616.7 ± 14.6
HalfCheetah-v2	17 (Continuous)	6 (Continuous)	100%g & 100%m 50%g 150%g 50%m 150%m	6106.3 ± 44.9 5396.7 ± 55.0 6171.4 ± 20.7 6159.7 ± 43.3 5889.9 ± 54.5
Ant-v2	111 (Continuous)	8 (Continuous)	100%g & 100%m 50%g 150%g 50%m 150%m	4136.5 ± 66.1 3618.8 ± 102.0 4182.6 ± 114.7 4255.3 ± 97.0 4399.8 ± 38.3
Task	Observation Space	Action Space	Environment Perturbation	Expert Performance
Hopper-v2	11 (Continuous)	3 (Continuous)	50%g+50%m 50%g+150%m 150%g+50%m 150%g+150%m	4042.7 ± 49.6 3473.4 ± 41.4 3789.8 ± 6.9 3386.8 ± 1.6
Walker2d-v2	17 (Continuous)	6 (Continuous)	50%g+50%m 50%g+150%m 150%g+50%m 150%g+150%m	6478.3 ± 84.2 4526.3 ± 33.0 5841.6 ± 49.9 5580.6 ± 12.5
HalfCheetah-v2	17 (Continuous)	6 (Continuous)	50%g+50%m 50%g+150%m 150%g+50%m 150%g+150%m	5015.4 ± 63.0 5466.5 ± 34.5 6051.8 ± 53.7 5843.9 ± 63.3
Ant-v2	111 (Continuous)	8 (Continuous)	50%g+50%m 50%g+150%m 150%g+50%m 150%g+150%m	3818.2 ± 97.4 3846.5 ± 106.6 4316.7 ± 114.9 4516.0 ± 77.3