NON-ADVERSARIAL INVERSE REINFORCEMENT LEARNING VIA SUCCESSOR FEATURE MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

In inverse reinforcement learning (IRL), an agent seeks to replicate expert demonstrations through interactions with the environment. Traditionally, IRL is treated as an adversarial game, where an adversary searches over reward models, and a learner optimizes the reward through repeated RL procedures. This game-solving approach is both computationally expensive and difficult to stabilize. In this work, we propose a novel approach to IRL by *direct policy optimization*: exploiting a linear factorization of the return as the inner product of successor features and a reward vector, we design an IRL algorithm by policy gradient descent on the gap between the learner and expert features. Our non-adversarial method does not require learning a reward function and can be solved seamlessly with existing actor-critic RL algorithms. Remarkably, our approach works in state-only settings without expert action labels, a setting which behavior cloning (BC) cannot solve. Empirical results demonstrate that our method learns from as few as a single expert demonstration and achieves improved performance on various control tasks.

1 INTRODUCTION

In imitation learning (Abbeel & Ng, 2004; Ziebart et al., 2008; Silver et al., 2016; Ho & Ermon, 2016; Swamy et al., 2021), the goal is to learn a decision-making policy that reproduces *behavior* from demonstrations. Rather than simply mimicking the state-conditioned action distribution as in behavior cloning (Pomerleau, 1988), interactive approaches like Inverse Reinforcement Learning (IRL; Abbeel & Ng, 2004; Ziebart et al., 2008) have the more ambitious goal of synthesizing a policy whose long-term occupancy measure approximates that of the expert demonstrator by some metric. As a result, IRL methods have proven to be more robust, particularly in a regime with few expert demonstrations, and has lead to successful deployments in real-world domains such as autonomous driving (Bronstein et al., 2022; Vinitsky et al., 2022; Igl et al.). However, this robustness comes at a cost: approaches to IRL tend to involve a costly bi-level optimization.

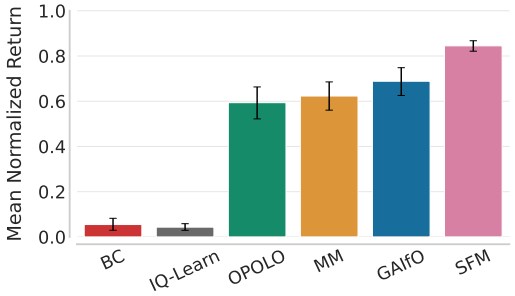

Figure 1: Comparing Mean Normalized Return on 10 tasks from DMC (Tassa et al., 2018) suite of our method SFM against the offline method BC (Pomerleau, 1988), the non-adversarial IRL method IQ-Learn (Garg et al., 2021), and the state-only adversarial methods OPOLO Zhu et al. (2020), MM (Swamy et al., 2021) and GAIfO (Torabi et al., 2018), where the agents are provided a single expert demonstration. Our state-only non-adversarial method SFM achieves higher mean normalized return. Error bars show the 95% bootstrap CIs.

Specifically, modern formulation of many IRL methods (e.g., Garg et al., 2021; Swamy et al., 2021) involve a min-max game between an adversary that learns a reward function to maximally differentiate between the agent and expert in the outer loop and a Reinforcement learning (RL) subroutine over this adversarial reward in the inner loop. However, all such methods encounter a set of well-documented challenges: (1) optimizing an adversarial game between the agent and the expert can be unstable, often requiring multiple tricks to stabilize training (Swamy et al., 2022), (2)

the inner loop of this bi-level optimization involves repeatedly solving a computationally expensive RL problem (Swamy et al., 2023), and (3) the reward function class must be specified in advance. Moreover, many approaches to imitation learning require knowledge of the actions taken by the demonstrator. This renders many forms of demonstrations unusable, such as videos, motion-capture data, and generally any demonstrations leveraging an alternative control interface than the learned policy (e.g., a human puppeteering a robot with external forces). As such, it is desirable to build IRL algorithms where the imitation policies learn from only expert states.

These challenges lead us to the following research question: *Can a non-adversarial approach to occupancy matching recover the expert's behavior without action labels?* To address this question, we present a new approach to IRL, called Successor Feature Matching (SFM), which provides a remarkably simple algorithm for imitation learning. Our contributions are the following:

Occupancy matching via reduction to reinforcement learning. We revisit the earlier approaches to *feature matching* (Abbeel & Ng, 2004; Ziebart et al., 2008), that is, matching the accumulation of discounted state or state-action base features along the expert's trajectory. For this task, we propose to estimate expected cumulative sum of features using Successor Features (SF; Barreto et al., 2017) – a low-variance, fully online algorithm that employs temporal-difference based methods for learning. Leveraging the benefits of SF, we demonstrate that *feature matching can be achieved by direct policy search* via policy gradients. In doing so, our method Successor Feature Matching (SFM) achieves strong imitation performance using off-the-shelf RL algorithms, circumventing bilevel optimization.

Imitation from a single state-only demonstration. When the learned base features are action-independent, we show that SFM can imitate an expert without knowledge of the actions it took in its demonstrations. This accommodates a variety of expert demonstration formats, such as video and motion-capture, where action labels are naturally absent. Through our experiments, we demonstrate that SFM successfully learns to imitate from as little as a single expert demonstration, without action labels—as highlighted in Figure 1. To our knowledge, SFM is the *only* online method capable of learning from a single unlabeled expert trajectory without requiring an expensive and difficult-to-stabilize bilevel optimization (Swamy et al., 2022). Additionally, rather than manually pre-specifying a class of expert reward functions (Swamy et al., 2021), SFM *adaptively* learns a class of reward functions from data using unsupervised RL techniques. As a result, SFM outperforms its competitors across a wide range of imitation learning benchmarks.

2 Related Work

Inverse Reinforcement Learning (IRL) methods typically learn via adversarial game dynamics, where prior methods assumed the base features are known upfront (Abbeel & Ng, 2004; Ziebart et al., 2008; Syed & Schapire, 2007; Syed et al., 2008) The advent of modern deep learning architectures led to methods (e.g. Ho & Ermon, 2016; Swamy et al., 2021; Fu et al., 2018) that do not estimate expected features, but instead learn a more expressive reward function that captures the differences between the expert and the the agent. The class of Moment Matching (MM; Swamy et al., 2021) methods offers a general framework that unifies existing algorithms through the concept of moment matching, or equivalently Integral Probability Metrics (IPM; Sun et al., 2019). In contrast to these methods, our approach is non-adversarial and focuses on directly addressing the problem of matching expected features. Furthermore, unlike prior methods in Apprenticeship Learning (AL; Abbeel & Ng, 2004) and Maximum Entropy IRL (Ziebart et al., 2008), our work *does not* assume the knowledge of base features. Instead, SFM leverages representation learning technique to extract relevant features from the raw observations. The method most similar to ours is IQ-Learn (Garg et al., 2021), a non-adversarial approach that utilizes an inverse Bellman operator to directly estimate the value function of the expert. Our method is also non-adversarial, but offers a significant advantage over IQ-Learn: it does not require knowledge of expert actions during training—it is a state-only imitation learning algorithm (Torabi et al., 2019). However, many existing state-only methods also rely on adversarial approaches (Torabi et al., 2018; Zhu et al., 2020; Gangwani & Peng, 2020). For instance, GAIfO (Torabi et al., 2018) modifies the discriminator to use state-only inputs, OPOLO Zhu et al. (2020) uses a similar discriminator and an inverse dynamics model to predict actions for expert transitions to regularize the agent, and R2I (Gangwani & Peng, 2020) proposed learning an indirect function to enable agents to imitate when the transition dynamics change. In contrast, SFM is a non-adversarial approach from learning from state-only demonstrations.

Successor Features (SF; Barreto et al., 2017) generalize the idea of the successor representation (SR; Dayan, 1993) by modeling the expected cumulative state features discounted according to the time of state visitation. Instead of employing successor features for tasks such as transfer learning (Barreto et al., 2017; Lehnert et al., 2017; Barreto et al., 2018; Abdolshah et al., 2021; Wiltzer et al., 2024), representation learning (Le Lan et al., 2022; Farebrother et al., 2023; Ghosh et al., 2023; Le Lan et al., 2023), exploration (Zhang et al., 2017; Machado et al., 2020; Jain et al., 2024), or zero-shot RL (Borsa et al., 2019; Touati & Ollivier, 2021; Touati et al., 2023; Park et al., 2024), our approach harnesses SFs for IRL, aiming to match expected features of the expert. Within the body of work on imitation learning, SFs have been leveraged to pre-train behavior foundation models capable of rapid imitation (Pirotta et al., 2024) and within adversarial IRL typically serves as the basis for estimating the value-function that best explains the expert (Lee et al., 2019; Filos et al., 2021; Abdulhai et al., 2022). In contrast, our work seeks to directly match SFs through a policy-gradient update without requiring large diverse datasets or costly bilevel optimization procedures.

3 PRELIMINARIES

Reinforcement Learning (RL; Sutton & Barto, 2018) typically considers a Markov Decision Process (MDP) defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, P_0)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition kernel, $r : \mathcal{S} \times \mathcal{A} \to [-1, 1]$ is the reward function, γ is the discount factor, and P_0 is the initial state distribution. Starting from the initial state $s_0 \sim P_0$ an agent takes actions according to its policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ producing trajectories $\tau = \{s_0, a_1, s_1, \dots\}$. The value function and action-value are respectively defined as $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) | S_0 = s]$ and $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) | S_0 = s, A_0 = a]$ where $\gamma \in [0, 1)$ represents the discount factor. The performance is the expected return obtained by following policy π from the initial state, given by $J(\pi) = \mathbb{E}_{s_0 \sim P_0}[\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,|\, S_0 = s]]$, and can be rewritten as $J(\pi) = \mathbb{E}_{s_0 \sim P_0}[V^\pi(s_0)]$.

The Successor Representation (SR; Dayan, 1993) provides the expected occupancy of future states for a given policy. For tabular state spaces, temporal-difference learning can be employed to estimate the SR. Successor Features (SF; Barreto et al., 2017) generalize the idea of the successor representation by instead counting the discounted sum of state features $\boldsymbol{\psi}^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \phi(S_t, A_t) | S_0 = s, A_0 = a]$ after applying the feature mapping $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$. The SR is recovered when ϕ is the identity function, with ϕ typically serving as a form of dimensionality reduction to learn SFs in continuous or large state spaces. In practice, SFs can be estimated via temporal difference learning (Sutton, 1988) through the minimization of the following objective,

$$\mathcal{L}_{SF}(\theta; \bar{\theta}) = \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \left[\| \phi(s, a) + \gamma \boldsymbol{\psi}^\pi_{\bar{\theta}}(s', \pi(s')) - \boldsymbol{\psi}^\pi_\theta(s, a) \|_2^2 \right], \tag{1}$$

where the tuple (s, a, s') is sampled from dataset $\mathcal{D}$ and $\boldsymbol{\psi}_\theta$ denotes a parametric SF model. The parameters $\bar{\theta}$ denote the "target parameters" that are periodically updated from θ by taking a direct copy or through a moving average. For tasks where the reward function can be expressed as a combination of base features ϕ and a weight vector $w \in \mathbb{R}^d$ such that $r(s, a) = \phi(s, a)^T w$, the performance of a policy π can be rewritten as $Q^\pi(s, a) = \boldsymbol{\psi}^\pi(s, a)^T w$ and $J(\pi) = \mathbb{E}_{s_0 \sim P_0, a \sim \pi(s_0)}[\boldsymbol{\psi}^\pi(s_0, a)]^T w$ (Barreto et al., 2017).

Inverse Reinforcement Learning (IRL; Ng et al., 2000; Abbeel & Ng, 2004; Ziebart et al., 2008) is the task of deriving behaviors using demonstrations through interacting with the environment. In contrast to RL where the agent improves its performance using the earned reward, Inverse Reinforcement Learning (IRL) involves learning without access to the reward function; good performance is signalled by expert demonstrations. As highlighted in Swamy et al. (2021), this corresponds to minimizing an Integral Probability Metric (IPM) (Sun et al., 2019) between the agent's state-visitation occupancy and the expert's which can be framed as a task to minimize the imitation gap given by:

$$J(\pi_E) - J(\pi) \leq \sup_{f \in \mathcal{F}_\phi} \left[\mathop{\mathbb{E}}_{\tau \sim \pi} \sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) - \mathop{\mathbb{E}}_{\tau \sim \pi_E} \sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) \right] \tag{2}$$

where $\mathcal{F}_r : \mathcal{S} \times \mathcal{A} \to [-1, 1]$ denotes the class of reward basis functions. Under this taxonomy, the agent being the minimization player selects a policy $\pi \in \Pi$ to compete with a discriminator that picks a reward moment function $f \in \mathcal{F}_r$ to maximize the imitation gap, and this min-max game is framed as $\min_\pi \max_{f \in \mathcal{F}} J(\pi_E) - J(\pi)$.

Algorithm 1 Successor Feature Matching (SFM)

Require: Expert demonstrations $\tau^E = \{s_0^i, a_0^i, \ldots, s_{T-1}^i, a_{T-1}^i\}_{i=1}^M$
Require: Base feature loss $\mathcal{L}_{\mathsf{feat}}$ and initialized parameters $\theta_{\mathsf{feat}} = (\phi, \mathsf{f})$
Require: Initialized actor π_μ, SF network and target $\boldsymbol{\psi}_\theta, \boldsymbol{\psi}_{\bar{\theta}}$, replay buffer $\mathcal{B}$
while Training **do**
  Observe state s and execute action $a = \pi_\mu(s)$ to get next state s'
  Add transition to replay buffer $\mathcal{B} \leftarrow \mathcal{B} \cup (s, a, s')$
  Compute expected features of expert $\hat{\boldsymbol{\psi}}^E = \frac{1}{M}\sum_{i=1}^M \sum_{t=0}^{T-1} \gamma^t \phi(s_t^i)$
  Sample minibatch $\mathcal{D} = \{(s, a, s')\} \sim \mathcal{B}$
  Update SF network using $\nabla_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \|\phi(s) + \boldsymbol{\psi}_{\bar{\theta}}(s', \pi_\mu(s')) - \boldsymbol{\psi}_\theta(s, a)\|_2^2$
  Update actor with $\nabla_\mu \frac{1}{2}\|(1-\gamma)^{-1}\mathbb{E}_{s,s'\sim\mathcal{D}}[\boldsymbol{\psi}_\theta(s, \pi_\mu(s)) - \gamma\boldsymbol{\psi}_{\bar{\theta}}(s', \pi_\mu(s'))] - \hat{\boldsymbol{\psi}}^E\|_2^2$
  Update base feature parameters using $\nabla_{\theta_{\mathsf{feat}}} \mathcal{L}_{\mathsf{feat}}(\theta_{\mathsf{feat}})$
end while

By restricting the class of reward basis functions to be within span of some base-features ϕ of state-action pairs such that $\mathcal{F}_\phi \in \{f(s,a) = \phi(s,a)^T w_f\}$, the imitation gap becomes:

$$\begin{aligned} J(\pi_E) - J(\pi) &\leq \sup_{f\in\mathcal{F}_\phi} \mathop{\mathbb{E}}_{\tau\sim\pi} \sum_{t=0}^{\infty} \gamma^t \phi(s_t, a_t)^\top w_f - \mathop{\mathbb{E}}_{\tau\sim\pi_E} \sum_{t=0}^{\infty} \gamma^t \phi(s_t, a_t)^\top w_f \\ &= \sup_{f\in\mathcal{F}_\phi} \left(\mathop{\mathbb{E}}_{s\sim P_0, a\sim\pi} [\boldsymbol{\psi}^\pi(s,a)] - \mathop{\mathbb{E}}_{s\sim P_0, a\sim\pi_E} [\boldsymbol{\psi}^E(s,a)] \right)^\top w_f, \end{aligned} \tag{3}$$

where $\boldsymbol{\psi}^E(s,a)$ denotes the successor features (SF) of the expert policy π_E for a given state s and action a. Under this assumption, the agent that matches the SF with the expert will minimize the performance gap across the class of restricted basis reward functions. Solving this objective of matching expected features between the agent and the expert has been studied in prior methods where prior methods have often resorted to an adversarial game (Ziebart et al., 2008; Abbeel & Ng, 2004; Syed & Schapire, 2007; Syed et al., 2008). In the sequel, we introduce a non-adversarial approach that updates the policy greedily to align the SFs between the expert and the agent, rather than learning a reward function to capture their behavioral divergence.

4 Successor Feature Matching (SFM)

In this section, we will describe SFM – a state-only non-adversarial algorithm for matching expected features between the agent and the expert. SFM distinguishes itself in two crucial manners; namely, it derives a feature-matching imitation policy by *direct policy optimization* via policy gradient ascent, and it *learns* base features simultaneously during training. Notably, the base features depend on states only to accommodate state-only demonstrations. These components are described in subsection 4.1 and subsection 4.2, respectively. Consequently, the SFM training loop closely resembles that of familiar actor-critic methods, avoiding challenges such as bilevel optimization.

4.1 Policy Optimization

The key intuition behind SFM is that successor feature matching in the ℓ_2-norm can be accomplished directly via policy gradient ascent—this allows us to leverage powerful actor-critic algorithms for IRL, as intuited by equation 3. Towards this end, we define a potential function defined as the Mean Squared Error (MSE) between the expected features of the expert and the agent, given by:

$$U(\mu) = \frac{1}{2}\|\widehat{\boldsymbol{\psi}}^{\pi_\mu} - \widehat{\boldsymbol{\psi}}^E\|_2^2, \tag{4}$$

where π_μ is a policy parameterized by μ, $\widehat{\boldsymbol{\psi}}^{\pi_\mu} = \mathbb{E}_{s\sim P_0, a\sim\pi(s)}[\boldsymbol{\psi}^{\pi_\mu}(s,a)]$ and $\hat{\boldsymbol{\psi}}^E = \mathbb{E}_{s\sim P_0, a\sim\pi_E(s)}[\boldsymbol{\psi}^E(s,a)]$ represents the expected SF of agent and expert conditioned on the initial state distribution P_0. Note that $\nabla U(\mu) = (\widehat{\boldsymbol{\psi}}^{\pi_\mu} - \widehat{\boldsymbol{\psi}}^E)^\top \nabla_\mu \widehat{\boldsymbol{\psi}}^{\pi_\mu}$. Interpreting $\widehat{\boldsymbol{\psi}}^\pi$ as a value function for a *vector-valued* reward (the base features), it becomes clear that the latter term is simply a vector of standard policy gradients. This suggests a method for matching the expert successor

features with a simple actor-critic algorithm. With the state-only base feature function $\phi : \mathcal{S} \to \mathbb{R}^d$, the expected features of the expert can be estimated using the demonstrations. Here, the SF for the expert using M demonstrations $\{\tau^i = \{s_1^i, a_2^i, ..., s_T^i, a_T^i\}\}_{i=1}^M$ of length T is obtained by

$$\widehat{\boldsymbol{\psi}}^E = \frac{1}{M} \sum_{i=1}^{M} \sum_{t=1}^{T} \gamma^t \phi(s_t^i). \tag{5}$$

Taking inspiration from off-policy actor-critic methods for standard RL tasks (Fujimoto et al., 2018; 2023; Haarnoja et al., 2018), SFM maintains a deterministic actor and a network to estimate the SF for agent's policy. Here, instead of having a critic to estimate the expected returns, the agent has a network to predict the expected features. The network to predict SF of the agent is a parameterized and differentiable function with parameters θ and is denoted by $\boldsymbol{\psi}_\theta$. To obtain actions for a given state, SFM maintains a deterministic actor π_μ with parameters μ. To learn policies without an adversarial game for this task, we propose to optimize this non-linear objective where our method leverages the prowess of the Deterministic Policy Gradient (DPG) (Silver et al., 2014) algorithm. We now describe the loss functions and Algorithm 1 provides a training procedure of SFM.

To update the actor network π_μ, we first show how SFM estimates the SF $\widehat{\boldsymbol{\psi}}_\theta^\pi$ of the current policy under the initial state distribution.

Proposition 1. *Let $\mathcal{B}$ denote a buffer of trajectories sampled from arbitrary stationary Markovian policies in the given MDP with initial state distribution P_0. For any deterministic policy π,*

$$\widehat{\boldsymbol{\psi}}^\pi := \mathbb{E}_{s\sim P_0}[\boldsymbol{\psi}^\pi(s, \pi(s))] = (1-\gamma)^{-1} \mathbb{E}_{(s_t, s_{t+1})\sim\mathcal{B}} \left[\boldsymbol{\psi}^\pi(s_t, \pi(s_t)) - \gamma \boldsymbol{\psi}^\pi(s_{t+1}, \pi(s_{t+1}))\right]. \tag{6}$$

The proof of Proposition 1 is deferred to Appendix A. Proposition 1 presents a method for estimating the SF for the agent conditioned on the initial state distribution. The proposed derivation can utilize samples coming from a different state-visitation distribution and uses an off-policy replay buffer in this work. Similar to standard off-policy RL algorithms (Fujimoto et al., 2018; 2023; Haarnoja et al., 2018), SFM maintains a replay buffer $\mathcal{B}$ to store the transitions and use it for sampling. This buffer allows us to make good use of all state transitions for the purpose of estimating the initial-state successor features with temporal difference learning.

Note that the potential function defined in Equation 4 depends only on the initial state distribution and does not specify a way of updating the actor for any other state. By substituting Equation 6 into Equation 4, we express the potential function representing the gap between the expected features of the agent and those of the expert in terms of features at states visited by the agent (and not just initial states). Thus, we define the loss $\mathcal{L}_G$ for the actor, which we call the SF-Gap loss, according to

$$\mathcal{L}_G(\mu) := \frac{1}{2} \left\| \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{s,s'\sim\mathcal{B}} [\boldsymbol{\psi}_\theta(s, \pi_\mu(s)) - \gamma \boldsymbol{\psi}_{\bar{\theta}}(s', \pi_\mu(s'))] - \widehat{\boldsymbol{\psi}}^E \right\|_2^2, \tag{7}$$

where we use the target network $\boldsymbol{\psi}_{\bar{\theta}}$ to get SF at the next state. We can see that Equation 7 approximates the potential function on states sampled from the replay buffer $\mathcal{B}$. To obtain the gradients with respect to the actor parameters μ, we propose using the Deterministic Policy Gradient (DPG) algorithm (Silver et al., 2014) that estimates the gradients by applying the chain-rule over Equation 7.

Proposition 2. *The gradients of the actor for a batch of sampled transitions from the replay buffer obtained by applying the DPG (Silver et al., 2014) algorithm to Equation 7 is*

$$\nabla_\mu \mathcal{L}_G(\mu) = \sum_{i=1}^{d} z_i (1-\gamma)^{-1} \mathbb{E}_{s,s'\sim\mathcal{B}} \left[\nabla_\mu \pi_\mu(s) \, \nabla_a \boldsymbol{\psi}_{\theta,i}(s,a)\big|_{a=\pi_\mu(s)} \right], \tag{8}$$

where $z_i = (1-\gamma)^{-1} \mathbb{E}_{s,s'\sim\mathcal{B}} \left[\boldsymbol{\psi}_{\theta,i}(s, \pi_\mu(s)) - \gamma \boldsymbol{\psi}_{\bar{\theta},i}(s', \pi_\mu(s'))\right] - \widehat{\boldsymbol{\psi}}_i^E$, $\boldsymbol{\psi}_{\theta,i}$ denotes the SF at the ith dimension for the current policy, and $\widehat{\boldsymbol{\psi}}_i^E$ is the ith dimension of SF of expert policy.

We provide the details of this derivation in Appendix A. Proposition 2 provides a way to estimate the gradients for the actor and optimize the objective defined in Equation 4. So far, we have illustrated a procedure for iteratively reducing the mean squared error between the expected SFs of a policy and those of the expert; we now justify in which sense our method accomplishes the goals of IRL.

Firstly, much like existing actor-critic methods, our procedure can ensure convergence to a local minimum of the MSE SF-matching objective; this is a direct consequence of Agarwal et al. (2022). Thus, as in the case of general actor-critic, we expect that local optima achieve low MSE.

The next simple proposition demonstrates that policies achieving low MSE must achieve a low imitation gap, validating their competency in the IRL setting.

Proposition 3. *Let $\epsilon > 0$ and let μ be a policy parameter such that $\|\widehat{\boldsymbol{\psi}}^{\pi_\mu} - \widehat{\boldsymbol{\psi}}^{E}\|_2 \leq \epsilon$. Suppose the expert policy is optimal for the reward function $r(s) = w^\top \phi(s)$ for base features $\phi(s) \in \mathbb{R}^d$ and some $w \in [-B, B]^d$ for $B < \infty$. Then it holds that $J(\pi_E) - J(\pi_\mu) \leq \sqrt{d}B\epsilon$.*

Proposition 3 establishes that, for any tolerance $\epsilon > 0$, the imitation gap can be reduced to ϵ by approximately minimizing Equation 4 to within a margin of $O(d^{-1/2}\epsilon)$.

Notably, $\widehat{\boldsymbol{\psi}}^{E}$ can be computed without knowledge of the expert's actions, under the assumption that the base features are action-independent. We note that SFM is not fundamentally incapable of handling problems in which there is no state-only base feature map describing the expert's reward: in this case, we may simply learn base features defined on the state-action space. As a consequence, we would require knowledge of the expert's actions to compute equation 5. In many interesting applications, however, it is sufficient to model state-only base features, as we show in section 5, enabling SFM to learn strong imitation policies without access to expert actions.

Ultimately, Proposition 2 and Equation 1 provide drop-in replacements to actor and critic losses for standard actor-critic methods (Algorithm 1 describes the training procedure of SFM). Training an actor-critic with the corresponding gradients enables state-only non-adversarial method for IRL.

4.2 Base Feature Function

We described in section 3 that SF depends on a base feature function $\phi : \mathcal{S} \to \mathbb{R}^d$. In this work, SFM learns the base features jointly while learning the policy. Base feature methods are parameterized by pairs $\theta_{\mathsf{feat}} = (\phi, \mathsf{f})$ together with losses $\mathcal{L}_{\mathsf{feat}}$, where $\phi : \mathcal{S} \to \mathbb{R}^d$ is a state feature map, f is an auxiliary object that may be used to learn ϕ, and $\mathcal{L}_{\mathsf{feat}}$ is a loss function defined for ϕ and f. Below, we briefly outline the base feature methods considered in our experiments.

Random Features (Random): Here, ϕ is a randomly-initialized neural network, and f is discarded. The network ϕ remains fixed during training ($\mathcal{L}_{\mathsf{feat}} \equiv 0$).

Autoencoder Features (AE): Here, $\phi : \mathcal{S} \to \mathbf{R}^d$ compresses states to latents in $\mathbb{R}^d$, and $\mathsf{f} : \mathbb{R}^d \to \mathcal{S}$ tries to reconstruct the state from the latent. The loss $\mathcal{L}_{\mathsf{feat}}$ is given by the AE loss $\mathcal{L}_{\mathsf{AE}}$,

$$\mathcal{L}_{\mathsf{AE}}(\theta_{\mathsf{feat}}) = \mathbb{E}_{s\sim\mathcal{D}}\left[\left\|\mathsf{f}(\phi(s)) - s\right\|_2^2\right], \quad \theta_{\mathsf{feat}} = (\phi, \mathsf{f}). \tag{9}$$

Inverse Dynamics Model (Pathak et al., 2017, IDM): Here, $\mathsf{f} : \mathbb{R}^d \times \mathbb{R}^d \to \mathcal{A}$ is a function that tries to predict the action that lead to the transition between embeddings of consecutive states. The loss $\mathcal{L}_{\mathsf{feat}}$ is given by the IDM loss $\mathcal{L}_{\mathsf{IDM}}$,

$$\mathcal{L}_{\mathsf{IDM}}(\theta_{\mathsf{feat}}) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left\|\mathsf{f}(\phi(s), \phi(s')) - a\right\|_2^2\right], \quad \theta_{\mathsf{feat}} = (\phi, \mathsf{f}). \tag{10}$$

Forward Dynamics Model (FDM): Here, $\mathsf{f} : \mathbb{R}^d \times \mathcal{A} \to \mathcal{S}$ is a function that tries to predict the next state in the MDP given the embedding of the current state and the chosen action. The loss $\mathcal{L}_{\mathsf{feat}}$ is given by the FDM loss $\mathcal{L}_{\mathsf{FDM}}$,

$$\mathcal{L}_{\mathsf{FDM}}(\theta_{\mathsf{feat}}) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left\|\mathsf{f}(\phi(s), a) - s'\right\|_2^2\right], \quad \theta_{\mathsf{feat}} = (\phi, \mathsf{f}). \tag{11}$$

Hilbert Representations (Park et al., 2024, HR): The feature map ϕ of HR is meant to estimate a *temporal distance*: the idea is that the difference between state embeddings $f^*_\phi(s, g) = \|\phi(s) - \phi(g)\|$ approximates the amount of timesteps required to traverse between the states s, g. Here, f is discarded, and $\mathcal{L}_{\mathsf{feat}}$ is the HR loss $\mathcal{L}_{\mathsf{HR}}$,

$$\mathcal{L}_{\mathsf{HR}}(\theta_{\mathsf{feat}}) = \mathop{\mathbb{E}}_{(s,s')\sim\mathcal{D}} \mathop{\mathbb{E}}_{g\sim\mathcal{D}}\left[\ell_\tau^2\left(-\mathbb{1}(s \neq g) - \gamma\,\mathsf{sg}\{f^*_\phi(s', g)\} + f^*_\phi(s, g)\right)\right], \ \theta_{\mathsf{feat}} = (\phi, \emptyset), \tag{12}$$

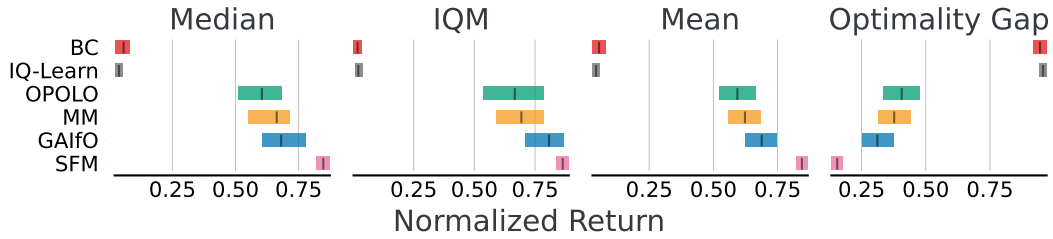

Figure 2: Rliable (Agarwal et al., 2021) plots of the proposed method SFM with an offline method BC (Pomerleau, 1988), a non-adversarial method IQ-Learn (Garg et al., 2021) that uses expert action labels and adversarial state-only methods: OPOLO (Zhu et al., 2020), MM (Swamy et al., 2021) and GAIfO (Torabi et al., 2018) across 10 tasks from DMControl suite (Tassa et al., 2018).

where sg denotes the stop-gradient operator, γ is the discount factor, and ℓ_τ^2 is the τ-expectile loss (Newey & Powell, 1987), as a proxy for the $\max$ operator in the Bellman backup (Kostrikov et al., 2021). In practice, $\text{sg}\{f_\phi^*(s', g)\}$ is replaced by $f_{\overline{\phi}}^*(s', g)$, where $\overline{\phi}$ is a delayed *target network* tracking ϕ, much like a target network in DQN (Mnih et al., 2015).

Finally, our framework does not preclude the use of adversarially-trained features, although we maintain that a key advantage of the framework is that it does not *require* adversarially-trained features. To demonstrate the influence of such features, we consider training base features via IRL.

Adversarial Representations (Adv): The features $\phi : \mathcal{S} \to \mathbf{R}^d$ are trained to maximally distinguish the features on states visited by the learned policy from the expert policy. That is, $\mathcal{L}_{\text{feat}}$ is given by $\mathcal{L}_{\text{Adv}}$ which adversarially maximizes an imitation gap similar to equation 3,

$$\mathcal{L}_{\text{Adv}}(\theta_{\text{feat}}) = -\left\|\mathbf{E}_{s\sim\pi}\phi(s) - \mathbf{E}_{s'\sim\pi_E}\phi(s')\right\|_2^2, \quad \theta_{\text{feat}} = (\phi, \emptyset). \tag{13}$$

In our experiments, we evaluated SFM with each of the base feature methods discussed above. A comparison of their performance is given in Figure 7. Our SFM method adapts familiar deterministic policy gradient algorithms, particularly TD3 (Fujimoto et al., 2018) and TD7 (Fujimoto et al., 2023), to policy optimization through the actor loss of Equation 6, and with the value function estimating the successor features corresponding to base features learned online. We provide implementation details in Appendix B, and demonstrate the performance of SFM in the following section.

5 Experiments

Through our experiments, we aim to analyze (1) how well SFM performs relative to competing non-adversarial and state-only adversarial methods at imitation from a single expert demonstration, (2) the robustness of SFM and its competitors to their underlying policy optimizer, (3) which features lead to strong performance in SFM, and (4) can SFM learn with stochastic policy optimizers? Our results are summarized in Figures 2, 4, 5, and 8 respectively, and are discussed in this section.

Ultimately, our results confirm that SFM indeed outperforms its competitors, achieving state-of-the-art performance on a variety of **single-demonstration** tasks, and even surpassing the performance of agents that have access to expert actions.

5.1 Experimental Setup

We evaluate our method on the 10 environments from the DeepMind Control (DMC) (Tassa et al., 2018) suite. Following the investigation in (Jena et al., 2020) which showed that the IRL algorithms are prone to biases in the learned reward function, we consider infinite horizon tasks where all episodes are truncated after 1000 steps in the environment. For each task, we collected expert demonstrations by training a TD3 (Fujimoto et al., 2018) agent for 1M environment steps. In our experiments, the agent is provided with a single expert demonstration which is sampled at the start and kept fixed during the training phase. The agents are trained for 1M environment steps and we report the mean performance across 10 seeds with 95% confidence interval shading and Rliable metrics (Agarwal et al., 2021). For the Rliable plots, we use the returns obtained by a random

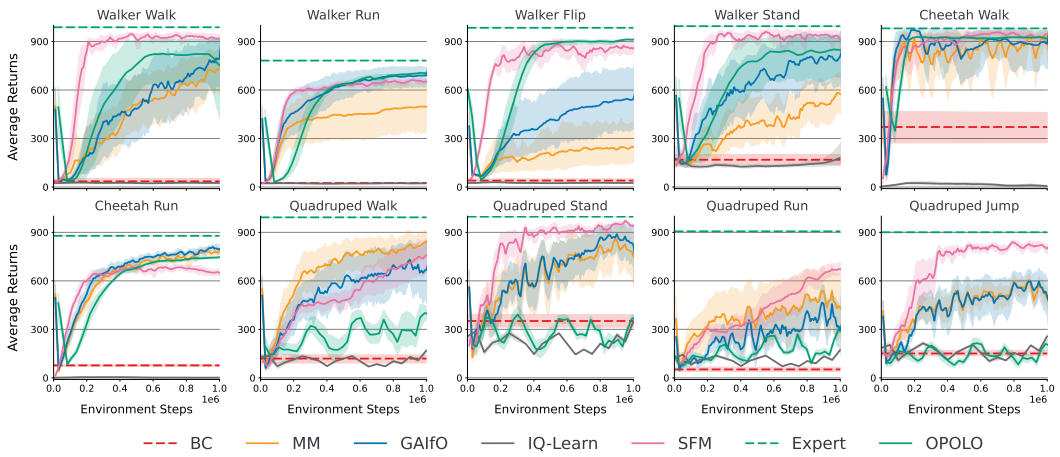

Figure 3: Per-task learning curves of IRL methods with a strong TD7 policy optimizer on single-demonstration imitation in DMC. Notably, IQ-Learn and BC require access to expert actions, while OPOLO, (state-only) MM, GAIfO, and SFM learn from expert states alone.

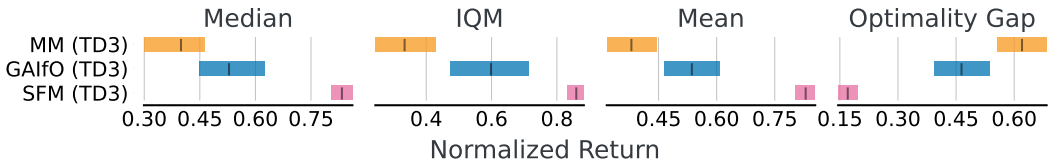

Figure 4: Performance of state-only IRL algorithms under the weaker TD3 policy optimizer.

policy and the expert policy to compute the normalized returns. Our implementation of SFM is in Jax (Bradbury et al., 2018) and it takes about ∼2.5 hours for one run on a single NVIDIA A100 GPU. We provide details about the implementation in Appendix B and hyperparameters in Appendix C.

Baselines. Our baselines include a state-only version of MM (moment matching) (Swamy et al., 2021), which is an adversarial IRL approach with the version where the integral probability metric (IPM) is replaced with the Jenson-Shannon divergence (which was shown to achieve better or comparable performance with GAIL (Swamy et al., 2022)). We implemented the state-only MM by changing the discriminator network to depend only on the state and not on the actions. Furthermore, we replace the RL optimizer in MM to TD7 (Fujimoto et al., 2023) to keep parity with the proposed method SFM. We compare SFM to another state-only baseline GAIfO (Torabi et al., 2018) where the discriminator learns to distinguish between the state transitions of the expert and the agent. Since, to our knowledge no official implementation of GAIfO is available, we implemented our version of GAIfO with a similar architecture to the MM framework. Here, we change the RL optimizer from TRPO (Schulman, 2015) in the paper to a recent RL optimizer like TD3 or TD7. Additionally, the adversarial approaches required Gradient Penalty (Gulrajani et al., 2017) on the discriminator, learning rate decay and the OAdam (Daskalakis et al., 2017) optimizer to stabilize learning. We also compare with OPOLO (Zhu et al., 2020) and use the official implementation for our experiments. Apart from state-only adversarial approaches, our baselines include behavior cloning (Pomerleau, 1988, BC) which is a supervised learning based imitation learning method trained to match actions taken by the expert. Lastly, we compare SFM with IQ-Learn (Garg et al., 2021) – a non-adversarial IRL algorithm which learns the Q-function using inverse Bellman operator (Piot et al., 2016). Notably, BC and IQ-Learn require the expert action labels in the demonstrations for learning.

5.2 Results

Quantitative Results Figure 2 presents the Rliable plots (Agarwal et al., 2021) obtained across DMC environments. We observe that the proposed method SFM learns to solve the task with a single demonstration and significantly outperforms the offline method BC (Pomerleau, 1988) and non-adversarial baseline IQ-Learn (Garg et al., 2021). Notably, SFM achieves this without using

Figure 5: Effect of different base feature functions on the performance of the agent. Here, we compare with Random, Inverse Dynamics Model (IDM), Hilbert Representations (Hilp), Autoencoder (AE), Adversarial (Adv) and Forward Dynamics Models (FDM).

the action labels in the demonstrations. We believe behavior cloning (BC) fails in this regime of few expert demonstrations as the agent is unpredictable upon encountering states not in the expert dataset (Ross & Bagnell, 2010). We further observe that SFM outperforms our implementation of state-only adversarial baselines– MM (Swamy et al., 2021) and GAIfO (Torabi et al., 2018) across all metrics. Furthermore, SFM has a significantly lower optimality gap, indicating that the baselines are more likely to perform much worse than the expert. Among the state-only adversarial approaches, GAIfO leverages a more powerful discriminator based on the state transition as compared to only states used in MM and thereby performs better. To further analyze the gains, we report the average returns across each task in Figure 3. Moreover, OPOLO does well on walker and cheetah domains. However, OPOLO regularizes the agent using a learned inverse dynamics module (IDM) and it is challenging to learn accurate models for quadruped domain. We observe that SFM converges faster when compared to leading methods, suggesting improved sample efficiency relative to its competitors. To highlight, SFM does not use techniques like gradient penalties (e.g., Gulrajani et al., 2017; Kodali et al., 2018) which are often required when training adversarial methods. Lastly, SFM outperforms MM and GAIfO on most tasks across the quadruped and walker domains.

Robustness with weaker policy optimizers In this work, the network architecture for SFM and state-only baselines is inspired from the TD7 (Fujimoto et al., 2023) framework. TD7 is a very recent algorithm presenting several tricks to attain improved performance relative to its celebrated predecessor TD3. As such, we also studied the performance of SFM as well as the state-only baselines built on TD3, in order to assess the robustness of these methods to the strength of their RL optimizer. Since the expert demonstrations were obtained using the TD3 algorithm, we believe the agents should recover expert behavior as the policy architecture is similar. The Rliable plots in Figure 4 present the efficacy of SFM to learn with simpler RL frameworks. Remarkably, the performance of SFM (TD3) is similar to the SFM (Figure 2) demonstrating the efficacy of our non-adversarial method to learn with other off-the-shelf RL algorithms. However, the adversarial baselines did not perform as well on top of TD3. In Figure 6 and table 3, we see that SFM attains significant performance gains across the tasks in the quadruped domain. In contrast, the baselines perform similarly on the cheetah and walker domains for both RL optimizers in the inner loop.

Ablation of base feature function In Figure 7, we study the performance of SFM under various base feature functions ϕ. We experiment with Random Features, Inverse Dynamics Models (IDM; Pathak et al., 2017), Hilbert Representations (Hilp; Park et al., 2024), Forward Dynamics Model (FDM), Autoencoder (AE) and Adversarial (Adv) features, as discussed in subsection 4.2. Through our experiments, we observe that FDM achieves superior results when compared with other base feature functions (Figure 5). In Figure 7 and Table 4, we see that, IDM features performed similarly to FDM on walker and cheetah domains, but did not perform well on quadruped tasks. We believe it is challenging to learn IDM features on quadruped domain and has been observed in prior works (Park et al., 2024; Touati & Ollivier, 2021). Similar trends were observed for Hilp features where Hilp based features did not do well on quadruped domain, where we suspect that learning the notion of temporal distance during online learning is challenging as the data distribution changes while training. Random features performed well on quadruped domain but did not perform well on cheetah and walker tasks. Autoencoder (AE) and advervsarial (Adv) features did well across Rliable metrics, however FDM features achieved better performance– we suspect that leveraging structure from the dynamics leads to superior performance. Moreover, learning adversarial features required tricks like gradient penalty and learning rate decay to stabilize training. We believe SFM can lever-

age any representation learning technique to obtain base features and a potential avenue for future work is to leverage pretrained features for more complex tasks to speed-up learning.

Extension to Stochastic Policy Optimizers So far, we present SFM with a deterministic policy and derive the update rule with DPG theorem in Proposition 2. In Appendix E, we describe how SFM can be learned with a stochastic actor. Here, the actor is parameterized to estimate the Gaussian distribution and reparameterization trick is used for computing the gradients. Moreover, we add an entropy regularizer while updating the policy for stability. Here, the SF network is updated with 1-step TD error as described in equation 1 with an action sampled from the actor. Algorithm 2 provides the training procedure of SF with stochastic policy. In Figure 8, we observe that SFM can learn with a stochastic policy and performs comparably to variants with deterministic polices.

6 Limitations

While SFM is simpler than IRL methods, it still doesn't theoretically alleviate the exploration problem that IRL methods encounter. A promising direction of future work would be to combine SFM with mechanisms like reset distribution (Swamy et al., 2023) or hybrid IRL (Ren et al., 2024) to improve sample efficiency. Currently, SFM incorporates exploration much like TD3 by adding noise to actions during training. Of course, this type of exploration strategy does not generally incur low regret, but it is not any less principled than actor-critic methods applied to deep RL. We highlight that SFM can leverage any existing exploration strategies common in RL—a potentially interesting direction is to use the successor features to derive exploration bonuses (Machado et al., 2020).

7 Discussion

We introduced SFM—a novel non-adversarial method for IRL that requires no expert action labels—via a reduction to a deterministic policy gradient algorithm. Our method learns to match the expert's successor features, derived from adaptively learned base features, using direct policy optimization as opposed to solving a minimax game. Through experiments on several standard imitation learning benchmarks, we have shown that state-of-the-art imitation is achievable with a non-adversarial approach, thereby providing an affirmative answer to our central research question.

Consequently, SFM is no less stable to train than its online RL subroutine. This is not the case with adversarial methods, which involve complex game dynamics during training. Much like the rich literature on GANs (Goodfellow et al., 2014; Gulrajani et al., 2017; Kodali et al., 2018), adversarial IRL methods often require several tricks to stabilize the optimization, such as gradient penalties, specific optimizers, and careful hyperparameter tuning.

Beyond achieving state-of-the-art performance, SFM demonstrated an unexpected feat: it is exceptionally robust to the policy optimization subroutine. Notably, when using the weaker TD3 policy optimizer, SFM performs almost as well as it does with a strong state-of-the-art TD7 optimizer. This is in stark contrast to the baseline methods, which performed considerably worse under the weaker policy optimizer. As such, we expect that SFM can be broadly useful and easier to deploy on resource-limited systems, which is often a constraint in robotics applications.

Interestingly, SFM follows a recent trend in model alignment that foregoes explicit reward modeling for direct policy optimization. This was famously exemplified by DPO (Rafailov et al., 2024) and its generalizations (Azar et al., 2024), which eliminate reward modeling from RLHF. It is worth noting that SFM, unlike DPO, *does* require modeling state features. However, the state features modeled by SFM are *task-agnostic*, and we found in particular that state embeddings for latent dynamics models suffice. We emphasize that this is a reflection of the more complicated dynamics inherent to general RL problems, unlike natural language problems which have trivial dynamics.

SFM is not the first non-adversarial IRL method; we note that IQ-Learn (Garg et al., 2021) similarly reduces IRL to RL. However, we showed that SFM substantially outperforms IQ-Learn in practice, and more importantly, it does so *without access to expert action labels*. Indeed, to our knowledge, SFM is *the first* non-adversarial state-only interactive IRL method. This opens the door to exciting possibilities, such as imitation learning from video and motion-capture data, which would not be possible for methods that require knowledge of the expert's actions. We believe that the simpler, non-adversarial nature of SFM training will be highly useful for scaling to such problems.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Majid Abdolshah, Hung Le, Thommen Karimpanal George, Sunil Gupta, Santu Rana, and Svetha Venkatesh. A new representation of successor features for transfer across dissimilar environments. In *International Conference on Machine Learning*, pp. 1–9. PMLR, 2021.

Marwa Abdulhai, Natasha Jaques, and Sergey Levine. Basis for intentions: Efficient inverse reinforcement learning using past experience. *CoRR*, abs/2208.04919, 2022.

Mridul Agarwal, Vaneet Aggarwal, and Tian Lan. Multi-objective reinforcement learning with nonlinear scalarization. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, pp. 9–17, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pp. 501–510. PMLR, 2018.

Diana Borsa, Andre Barreto, John Quan, Daniel J. Mankowitz, Hado van Hasselt, Remi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1VWjiRcKX.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Eli Bronstein, Mark Palatucci, Dominik Notz, Brandyn White, Alex Kuefler, Yiren Lu, Supratik Paul, Payam Nikdel, Paul Mougin, Hongge Chen, et al. Hierarchical model-based imitation learning for planning in autonomous driving. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8652–8659. IEEE, 2022.

Robert Dadashi, Adrien Ali Taiga, Nicolas Le Roux, Dale Schuurmans, and Marc G Bellemare. The value function polytope in reinforcement learning. In *International Conference on Machine Learning*, pp. 1486–1495. PMLR, 2019.

Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Jesse Farebrother, Joshua Greaves, Rishabh Agarwal, Charline Le Lan, Ross Goroshin, Pablo Samuel Castro, and Marc G Bellemare. Proto-value networks: Scaling representation learning with auxiliary tasks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=oGDKSt9JrZi.

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. In *International Conference on Machine Learning (ICML)*, 2021.

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-.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 10–15 Jul 2018.

Scott Fujimoto, David Meger, and Doina Precup. An equivalence between loss functions and non-uniform sampling in experience replay. *Advances in neural information processing systems*, 33: 14219–14230, 2020.

Scott Fujimoto, Wei-Di Chang, Edward J. Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For SALE: State-action representation learning for deep reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xZvGrzRq17.

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

Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. IQ-learn: Inverse soft-q learning for imitation. 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=Aeo-xqtb5p.

Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions. In *International Conference on Machine Learning (ICML)*, 2023.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

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*, pp. 1861–1870. PMLR, 2018.

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

M Igl, D Kim, A Kuefler, P Mougin, P Shah, K Shiarlis, D Anguelov, M Palatucci, B White, and S Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation, 2022. *URL https://arxiv. org/abs/2205.03195*.

Arnav Kumar Jain, Lucas Lehnert, Irina Rish, and Glen Berseth. Maximum state entropy exploration using predecessor and successor representations. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2024.

Rohit Jena, Siddharth Agrawal, and Katia Sycara. Addressing reward bias in adversarial imitation learning with neutral reward functions. *arXiv preprint arXiv:2009.09467*, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Naveen Kodali, James Hays, Jacob Abernethy, and Zsolt Kira. On convergence and stability of GANs. In *International Conference on Learning Representations (ICLR)*, 2018.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Charline Le Lan, Stephen Tu, Adam Oberman, Rishabh Agarwal, and Marc G. Bellemare. On the generalization of representations in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

Charline Le Lan, Stephen Tu, Mark Rowland, Anna Harutyunyan, Rishabh Agarwal, Marc G. Bellemare, and Will Dabney. Bootstrapped representations in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

Donghun Lee, Srivatsan Srinivasan, and Finale Doshi-Velez. Truly batch apprenticeship learning with deep successor features. *arXiv preprint arXiv:1903.10077*, 2019.

Lucas Lehnert, Stefanie Tellex, and Michael L Littman. Advantages and limitations of using successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1708.00102*, 2017.

Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5125–5133, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Whitney K Newey and James L Powell. Asymmetric least squares estimation and testing. *Econometrica: Journal of the Econometric Society*, pp. 819–847, 1987.

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

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. *arXiv preprint arXiv:2402.15567*, 2024.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Bilal Piot, Matthieu Geist, and Olivier Pietquin. Bridging the gap between imitation learning and inverse reinforcement learning. *IEEE transactions on neural networks and learning systems*, 28 (8):1814–1826, 2016.

Matteo Pirotta, Andrea Tirinzoni, Ahmed Touati, Alessandro Lazaric, and Yann Ollivier. Fast imitation via behavior foundation models. In *International Conference on Learning Representations (ICLR)*, 2024.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Juntao Ren, Gokul Swamy, Zhiwei Steven Wu, J Andrew Bagnell, and Sanjiban Choudhury. Hybrid inverse reinforcement learning. *arXiv preprint arXiv:2402.08848*, 2024.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668. JMLR Workshop and Conference Proceedings, 2010.

John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. Pmlr, 2014.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Wen Sun, Anirudh Vemula, Byron Boots, and Drew Bagnell. Provably efficient imitation learning from observation alone. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6036–6045. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/sun19b.html.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

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

Gokul Swamy, Sanjiban Choudhury, J Andrew Bagnell, and Steven Wu. Of moments and matching: A game-theoretic framework for closing the imitation gap. In *International Conference on Machine Learning*, pp. 10022–10032. PMLR, 2021.

Gokul Swamy, Nived Rajaraman, Matt Peng, Sanjiban Choudhury, Drew Bagnell, Steven Wu, Jiantao Jiao, and Kannan Ramchandran. Minimax optimal online imitation learning via replay estimation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=1mFfKXYMg5a.

Gokul Swamy, David Wu, Sanjiban Choudhury, Drew Bagnell, and Steven Wu. Inverse reinforcement learning without reinforcement learning. In *International Conference on Machine Learning*, pp. 33299–33318. PMLR, 2023.

Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. *Advances in neural information processing systems*, 20, 2007.

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

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MYEap_OcQI.

Eugene Vinitsky, Nathan Lichtlé, Xiaomeng Yang, Brandon Amos, and Jakob Foerster. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. *Advances in Neural Information Processing Systems*, 35:3962–3974, 2022.

Harley Wiltzer, Jesse Farebrother, Arthur Gretton, and Mark Rowland. Foundations of multivariate distributional reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2371–2378. IEEE, 2017.

Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. *Advances in neural information processing systems*, 33:12402–12413, 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.

A PROOFS

Before proving Proposition 1, we begin by proving some helpful lemmas. First, we present a simple generalization of a result from Garg et al. (2021).

Lemma 1. *Let μ denote any discounted state-action occupancy measure for an MDP with state space $\mathcal{S}$ and initial state distribution P_0, and let $\mathcal{V}$ denote a vector space. Then for any $f : \mathcal{S} \to \mathcal{V}$, the following holds,*

$$\mathbb{E}_{(s,a)\sim\mu}\left[f(s) - \gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}[f(s')]\right] = (1-\gamma)\mathbb{E}_{s\sim P_0}\left[f(s)\right].$$

Proof. Firstly, any discounted state-action occupancy measure μ is identified with a unique policy π^μ as shown by Ho & Ermon (2016). So, μ is characterized by

$$\mu(\mathrm{d}s\mathrm{d}a) = (1-\gamma)\pi^\mu(\mathrm{d}a \mid s)\sum_{t=0}^{\infty}\gamma^t p_t^\mu(\mathrm{d}s),$$

where $p_t^\mu(S) = \Pr_{\pi^\mu}(S_t \in S)$ is the state-marginal distribution under policy π^μ at timestep t. Expanding the LHS of the proposed identity yields

$$\begin{aligned}
&\mathbb{E}_{(s,a)\sim\mu}\left[f(s) - (1-\gamma)\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}[f(s')]\right] \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s\sim p_t^\mu}[f(s)] - \gamma\mathbb{E}_{(s,a)\sim\mu}\mathbb{E}_{s'\sim P(\cdot|s,a)}[f(s')] \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s\sim p_t^\mu}[f(s)] - (1-\gamma)\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{E}_{s\sim p_t^\mu}\mathbb{E}_{a\sim\pi^\mu(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}[f(s')] \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s\sim p_t^\mu}[f(s)] - (1-\gamma)\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{E}_{s\sim p_{t+1}^\mu}[f(s)] \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s\sim p_t^\mu}[f(s)] - (1-\gamma)\sum_{t=1}^{\infty}\gamma^t\mathbb{E}_{s\sim p_t^\mu}[f(s)] \\
&= (1-\gamma)\mathbb{E}_{s\sim P_0}[f(s)],
\end{aligned}$$

since $p_0^\mu = P_0$ (the initial state distribution) for any μ. □

Intuitively, we will invoke Lemma 1 with f denoting the successor features to derive an expression for the initial state successor features via state transitions sampled from a replay buffer.

Proposition 1. *Let $\mathcal{B}$ denote a buffer of trajectories sampled from arbitrary stationary Markovian policies in the given MDP with initial state distribution P_0. For any deterministic policy π,*

$$\widehat{\boldsymbol{\psi}}^\pi := \mathbb{E}_{s\sim P_0}[\boldsymbol{\psi}^\pi(s,\pi(s))] = (1-\gamma)^{-1}\mathbb{E}_{(s_t,s_{t+1})\sim\mathcal{B}}\left[\boldsymbol{\psi}^\pi(s_t,\pi(s_t)) - \gamma\boldsymbol{\psi}^\pi(s_{t+1},\pi(s_{t+1}))\right]. \quad (6)$$

Proof. Suppose $\mathcal{B}$ contains rollouts from policies $\{\pi_k\}_{k=1}^N$ for some $N \in \mathbb{N}$. Each of these policies π_k induces a discounted state-action occupancy measure μ_k. Since the space of all discounted state-action occupancy measures is convex (Dadashi et al., 2019), it follows that $\mu = \frac{1}{N}\sum_{k=1}^N \mu_k$ is itself a discounted state-action occupancy measure.

Consider the function $f : \mathcal{S} \to \mathbb{R}^d$ given by $f(s) = \boldsymbol{\psi}^{\pi}(s, \pi(s))$. We have

$$
\begin{aligned}
&\mathbb{E}_{(s_t, s_{t+1}) \sim \mathcal{B}}[f(s_t) - \gamma f(s_{t+1})] \\
&= \mathbb{E}_{k \sim \mathsf{Uniform}(\{1,\ldots,N\})} \mathbb{E}_{(s_t, a_t) \sim \mu_k, s_{t+1} \sim P(\cdot|s_t, a_t)}[f(s_t) - \gamma f(s_{t+1})] \\
&= \mathbb{E}_{(s_t, a_t) \sim \mu, s_{t+1} \sim P(\cdot|s_t, a_t)}[f(s_t) - \gamma f(s_{t+1})] \\
&= \mathbb{E}_{(s,a) \sim \mu} \left[f(s_t) - \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[f(s')] \right] \\
&= (1-\gamma) \mathbb{E}_{s \sim P_0}[f(s)],
\end{aligned}
$$

where the final step invokes Lemma 1, which is applicable since μ is a discounted state-action occupancy measure. The claim then follows by substituting $f(s)$ for $\boldsymbol{\psi}^{\pi}(s, \pi(s))$ and multipying through by $(1-\gamma)^{-1}$.

□

Proposition 2. *The gradients of the actor for a batch of sampled transitions from the replay buffer obtained by applying the DPG (Silver et al., 2014) algorithm to Equation 7 is*

$$
\nabla_{\mu} \mathcal{L}_G(\mu) = \sum_{i=1}^{d} z_i (1-\gamma)^{-1} \mathbb{E}_{s,s' \sim \mathcal{B}} \left[\nabla_{\mu} \pi_{\mu}(s) \, \nabla_a \boldsymbol{\psi}_{\theta,i}(s,a) \big|_{a=\pi_{\mu}(s)} \right], \tag{8}
$$

where $z_i = (1-\gamma)^{-1} \mathbb{E}_{s,s' \sim \mathcal{B}} \left[\boldsymbol{\psi}_{\theta,i}(s, \pi_{\mu}(s)) - \gamma \boldsymbol{\psi}_{\bar{\theta},i}(s', \pi_{\mu}(s')) \right] - \widehat{\boldsymbol{\psi}}_i^E$*,* $\boldsymbol{\psi}_{\theta,i}$ *denotes the SF at the* i*th dimension for the current policy, and* $\widehat{\boldsymbol{\psi}}_i^E$ *is the* i*th dimension of SF of expert policy.*

Proof. For the loss function

$$
\mathcal{L}_G(\mu) = \frac{1}{2} \| (1-\gamma)^{-1} \mathbb{E}_{s,s' \sim \mathcal{B}}[\boldsymbol{\psi}_{\theta}(s, \pi_{\mu}(s)) - \gamma \boldsymbol{\psi}_{\bar{\theta}}(s', \pi_{\mu}(s'))] - \hat{\boldsymbol{\psi}}^E \|_2^2 \tag{14}
$$

the gradient for the actor is given by:

$$
\begin{aligned}
\nabla_{\mu} \mathcal{L}_G(\mu) &= \frac{1}{2} \nabla_{\mu} \sum_{i=1}^{d} \{ (1-\gamma)^{-1} \mathbb{E}_{s,s' \sim \mathcal{B}}[\boldsymbol{\psi}_{\theta}(s, \pi_{\mu}(s)) - \gamma \boldsymbol{\psi}_{\bar{\theta}}(s', \pi_{\mu}(s'))] - \hat{\boldsymbol{\psi}}^E \}^2 \\
&= \sum_{i=1}^{d} z_i \nabla_{\mu} \{ (1-\gamma)^{-1} \mathbb{E}_{s,s' \sim \mathcal{B}}[\boldsymbol{\psi}_{\theta}(s, \pi_{\mu}(s)) - \gamma \boldsymbol{\psi}_{\bar{\theta}}(s', \pi_{\mu}(s'))] - \hat{\boldsymbol{\psi}}^E \} \\
&= \sum_{i=1}^{d} z_i \{ (1-\gamma)^{-1} \nabla_{\mu} \mathbb{E}_{s,s' \sim \mathcal{B}}[\boldsymbol{\psi}_{\theta}(s, \pi_{\mu}(s)) \} \\
&= \sum_{i=1}^{d} z_i \{ (1-\gamma)^{-1} \mathbb{E}_{s,s' \sim \mathcal{B}}[\nabla_{\mu} \boldsymbol{\psi}_{\theta}(s, \pi_{\mu}(s)) \} \\
&= \sum_{i=1}^{d} z_i \{ (1-\gamma)^{-1} \mathbb{E}_{s,s' \sim \mathcal{B}}[\nabla_{\mu} \pi_{\mu}(a) \nabla_a \boldsymbol{\psi}_{\theta}(s,a)|_{a=\pi_{\mu}(s)} \}
\end{aligned}
$$

Here, we defined $z_i = (1-\gamma)^{-1} \mathbb{E}_{s,s' \sim \mathcal{B}}[\boldsymbol{\psi}_{\theta}(s, \pi_{\mu}(s)) - \gamma \boldsymbol{\psi}_{\bar{\theta}}(s', \pi_{\mu}(s'))] - \hat{\boldsymbol{\psi}}^E$. This completes the proof.

□

Proposition 3. *Let* $\epsilon > 0$ *and let* μ *be a policy parameter such that* $\| \widehat{\boldsymbol{\psi}}^{\pi_{\mu}} - \widehat{\boldsymbol{\psi}}^E \|_2 \leq \epsilon$*. Suppose the expert policy is optimal for the reward function* $r(s) = w^{\top} \phi(s)$ *for base features* $\phi(s) \in \mathbb{R}^d$ *and some* $w \in [-B, B]^d$ *for* $B < \infty$*. Then it holds that* $J(\pi_E) - J(\pi_{\mu}) \leq \sqrt{d} B \epsilon$*.*

Proof. Notably, we have that $J(\pi) = w^\top \widehat{\boldsymbol{\psi}}^\pi$. Thus,

$$\begin{aligned} |J(\pi_E) - J(\pi_\mu)| &= |w^\top (\widehat{\boldsymbol{\psi}}^E - \widehat{\boldsymbol{\psi}}^{\pi_\mu})| \\ &\leq \|w\|_\infty \|\widehat{\boldsymbol{\psi}}^E - \widehat{\boldsymbol{\psi}}^{\pi_\mu}\|_1, \end{aligned}$$

by the Hölder inequality. By assumption, we immediately have that $\|w\|_\infty \leq B$. Moreover, since $\|v\|_1 \leq \sqrt{d}\|v\|_2$ for any $v \in \mathbb{R}^d$, we have that

$$\begin{aligned} |J(\pi_E) - J(\pi_\mu)| &\leq B\left(\sqrt{d}\|\widehat{\boldsymbol{\psi}}^E - \widehat{\boldsymbol{\psi}}^{\pi_\mu}\|_2\right) \\ &\leq \sqrt{d}B\epsilon. \end{aligned}$$

□

B Implementation Details

Since SFM does not involve estimating a reward function and cannot leverage an off-the-shelf RL algorithm to learn a Q-funtion, we propose a novel architecture for our method. SFM is composed of 3 different components- actor π_μ, SF network $\boldsymbol{\psi}_\theta$, base feature function ϕ and f. Taking inspiration from state-of-the-art RL algorithms, we mantain target networks for both actor and the SF network. Since, SF network acts similarly to a critic in actor-critic like algorithms, SFM comprises of two networks to estimate the SF (Fujimoto et al., 2018). Here, instead taking a minimum over estimates of SF from these two networks, our method performed better with average over the two estimates of SF. To implement the networks of SFM, we incorporated several components from the TD7 (Fujimoto et al., 2023) algorithm. Moreover, unlike MM (Swamy et al., 2021), SFM did not require techniques like gradient penalty (Gulrajani et al., 2017), the OAdam optimizer (Daskalakis et al., 2017) and a learning rate scheduler.

B.1 Network Architecture

The architecture used in this work is inspired from the TD7 (Fujimoto et al., 2023) algorithms for continuous control tasks (Pseudo 2). We will describe the networks and sub-components used below:

- Two functions to estimate the SF ($\boldsymbol{\psi}_{\theta_1}, \boldsymbol{\psi}_{\theta_2}$)
- Two target functions to estimate the SF ($\boldsymbol{\psi}_{\bar{\theta}_1}, \boldsymbol{\psi}_{\bar{\theta}_2}$)
- A policy network π_μ
- A target policy network $\pi_{\bar{\mu}}$
- An encoder with sub-components f_ν, g_ν
- A target encoder with sub-components $f_{\bar{\nu}}, g_{\bar{\nu}}$
- A fixed target encoder with sub-components $f_{\bar{\bar{\nu}}}, g_{\bar{\bar{\nu}}}$
- A checkpoint policy π_c and the checkpoint encoder f_c
- A base feature function ϕ

Encoder: The encoder comprises of two sub-networks to output state and state-action embedding, such that $z^s = f_\nu(s)$ and $z^{sa} = g_\nu(z^s, a)$. The encoder is updated using the following loss:

$$\mathcal{L}_{\mathsf{Encoder}}(f_\nu, g_\nu) = \left(g_\nu(f_\nu(s), a) - |f_\nu(s')|_\times\right)^2 \tag{15}$$

$$= \left(z^{sa} - |z^{s'}|_\times\right)^2, \tag{16}$$

where s, a, s' denotes the sampled transitions and $|\ .\ |_\times$ is the stop-gradient operation. Also, we represent $\bar{z}^s = f_{\bar{\nu}}(s)$, $\bar{z}^{sa} = g_{\bar{\nu}}(\bar{z}^s, a)$, $\bar{\bar{z}}^s = f_{\bar{\bar{\nu}}}(s)$, and $\bar{\bar{z}}^{sa} = g_{\bar{\bar{\nu}}}(\bar{\bar{z}}^s, a)$.

SF network: Motivated by standard RL algorithms (Fujimoto et al., 2018; 2023), SFM uses a pair of networks to estimate the SF. The network to estimate SF are updated with the following loss:

$$\mathcal{L}_{\mathsf{SF}}(\boldsymbol{\psi}_{\theta_i}) = \|\mathsf{target} - \boldsymbol{\psi}_{\theta_i}(\bar{z}^{sa}, \bar{z}^{s}, s, a)\|_2^2, \tag{17}$$

$$\mathsf{target} = \phi(s) + \frac{1}{2}\gamma * \mathrm{clip}([\boldsymbol{\psi}_{\bar{\theta}_1}(x) + \boldsymbol{\psi}_{\bar{\theta}_2}(x)], \boldsymbol{\psi}_{\min}, \boldsymbol{\psi}_{\max}), \tag{18}$$

$$x = [\bar{\bar{z}}^{s'a'}, \bar{\bar{z}}^{s'}, s', a'] \tag{19}$$

$$a' = \pi_{\bar{\mu}}(\bar{\bar{z}}^{s'}, s') + \epsilon, \text{where } \epsilon \sim \mathrm{clip}(\mathcal{N}(0, \sigma^2), -c, c). \tag{20}$$

Here, instead of taking the minimum over the SF networks for bootstrapping at the next state (Fujimoto et al., 2018), the mean over the estimates of SF is used. The action at next state a' is samples similarly to TD3 (Fujimoto et al., 2018) and the same values of (z^s, z^{sa}) are used for each SF network. Moreover, the algorithm does clipping similar to TD7 (Fujimoto et al., 2023) on the predicted SF at the next state which is updated using $target$ (equation 18) at each time step, given by:

$$\boldsymbol{\psi}_{\min} \leftarrow \min(\boldsymbol{\psi}_{\min}, \mathsf{target}) \tag{21}$$

$$\boldsymbol{\psi}_{\max} \leftarrow \min(\boldsymbol{\psi}_{\max}, \mathsf{target}) \tag{22}$$

Policy: SFM uses a single policy network which takes $[z^s,\ s]$ as input and is updated using the following loss function described in section 4.

Upon every $target\_update\_frequency$ training steps, the target networks are updated by cloning the current network parameters and remains fixed:

$$(\theta_1, \theta_2) \leftarrow (\bar{\theta}_1, \bar{\theta}_2) \tag{23}$$

$$\mu \leftarrow \bar{\mu} \tag{24}$$

$$(\nu_1, \nu_2) \leftarrow (\bar{\nu}_1, \bar{\nu}_2) \tag{25}$$

$$(\bar{\nu}_1, \bar{\nu}_2) \leftarrow (\bar{\bar{\nu}}_1, \bar{\bar{\nu}}_2) \tag{26}$$

$$\tag{27}$$

Moreover, the agent maintains a checkpointed network similar to TD7 (Refer to Appendix F of TD7 (Fujimoto et al., 2023) paper). However, TD7 utilizes the returns obtained in the environment for checkpointing. Since average returns is absent in the IRL tasks, it is not clear how to checkpoint policies. Towards this end, we propose using the negative of MSE between the SF of trajectories generated by agent and the SF of demonstrations as a proxy of checkpointing. To highlight some differences with the TD7 Fujimoto et al. (2023) algorithm, SFM does not utilize a LAP Fujimoto et al. (2020) and Huber loss to update SF network, and we leave investigating them for future research.

Base Features: Since we use a base feature function ϕ, we have two networks- 1) To provide the embedding for the state, and 2) To predict the next state from the current state and action. Pseudo 1 provides the description of the network architectures and the corresponding forward passes.

State-only adversarial baselines: For the state-only MM method, we used the same architecture as TD7 (Fujimoto et al., 2023) or TD3 (Fujimoto et al., 2018) for the RL subroutine. We kept the same architecture of the discriminator as provided in the official implementation. However, we modified the discriminator to take only states as inputs. Additionally, we used gradient penalty and learning rate decay to update the discriminator, and OAdam optimizer (Daskalakis et al., 2017) for all networks. For GAIfO (Torabi et al., 2018), we used the same architecture as state-only MM. However, the discriminator takes the state-transition denoted as the state and next-state pair as input.

Pseudo 1. *Base Feature Network Details*

Variables:

```
phi_dim = 128
```

Base Feature Network ϕ to encode state:

```
l0 = Linear(state_dim, 512)
l2 = Linear(512, 512)
l3 = Linear(512, phi_dim)
```

Base Feature ϕ Forward Pass:

```
input = state
x = Layernorm(l1(x))
x = tanh(x)
x = ReLU(x)
phi_s = L2Norm(l3(x))
```

FDM Network:

```
l0 = Linear(phi_dim + action_dim, 512)
l1 = Linear(512, 512)
l2 = Linear(512, action_dim)
```

FDM Network Forward Pass:

```
input = concatenate([phi_s, action])
x = ReLU(l0(x))
x = ReLU(l1(x))
action = tanh(l2(x))
```

Pseudo 2. *SFM Network Details*

Variables:

```
phi_dim = 128
zs_dim = 256
```

Value SF Network:

▷ SFM uses two SF networks each with similar architechture and forward pass.

```
l0 = Linear(state_dim + action_dim, 256)
l1 = Linear(zs_dim * 2 + 256, 256)
l2 = Linear(256, 256)
l3 = Linear(256, phi_dim)
```

SF Network ψ_θ Forward Pass:

```
input = concatenate([state, action])
x = AvgL1Norm(l0(inuput))
x = concatenate([zsa, zs, x])
x = ELU(l1(x))
x = ELU(l2(x))
sf = l3(x)
```

Policy π Network:

```
l0 = Linear(state_dim, 256)
l1 = Linear(zs_dim + 256, 256)
l2 = Linear(256, 256)
l3 = Linear(256, action_dim)
```

Policy π Forward Pass:

```
input = state
x = AvgL1Norm(l0(input))
x = concatenate([zs, x])
x = ReLU(l1(x))
x = ReLU(l2(x))
action = tanh(l3(x))
```

State Encoder f Network:

```
l1 = Linear(state_dim, 256)
l2 = Linear(256, 256)
l3 = Linear(256, zs_dim)
```

State Encoder f Forward Pass:

```
input = state
x = ELU(l1(input))
x = ELU(l2(x))
zs = AvgL1Norm(l3(x))
```

State-Action Encoder g Network:

```
l1 = Linear(action_dim + zs_dim, 256)
l2 = Linear(256, 256)
l3 = Linear(256, zs_dim)
```

State-Action Encoder g Forward Pass:

```
input = concatenate([action, zs])
x = ELU(l1(input))
x = ELU(l2(x))
zsa = l3(x)
```

C HYPERPARAMETERS

In Table 1, we provide the details of the hyperparameters used for learning. Many of our hyperparamters are similar to the TD7 (Fujimoto et al., 2023) algorithm. Important hyperparameters include the discount factor γ for the SF network and tuned it with values $\gamma = [0.98, 0.99, 0.995]$ and report the ones that worked best in the table. Rest, our method was robust to hyperparameters like learning rate and batch-size used during training.

Name	**Value**
Batch Size	1024
Discount factor γ for SF	.99
Actor Learning Rate	5e-4
SF network Learning Rate	5e-4
Base feature function learning Rate	5e-4
Network update interval	250
Target noise	.2
Target Noise Clip	.5
Action noise	.1
Environments steps	1e6

Table 1: Hyper parameters used to train SFM.

D EXTENDED RESULTS

In this section, we provide the tables with average returns across tasks from DMControl Suite (Table 2, 3 & 4) and per-environment training runs for our study with weak policy optimizers and base feature functions ((Fig. 6 & 7).

Task	BC	IQ-Learn	OPOLO	MM	GAIfO	SFM
Cheetah Run	77.0 ± 11.1	1.4 ± 1.4	747.5 ± 6.7	$\mathbf{781.6 \pm 30.7}$	$\mathbf{777.2 \pm 45.0}$	648.8 ± 35.9
Cheetah Walk	371.1 ± 163.3	5.7 ± 6.9	919 ± 26.3	895.6 ± 128.2	885.2 ± 236.2	$\mathbf{945.1 \pm 33.9}$
Quadruped Jump	150.8 ± 29.7	260.7 ± 12.0	198.9 ± 50.6	489.4 ± 104.6	505.8 ± 192.3	$\mathbf{799.1 \pm 47.8}$
Quadruped Run	52.1 ± 22.4	174.7 ± 7.7	291.5 ± 43.9	433.9 ± 347.4	289.4 ± 227.3	$\mathbf{671.7 \pm 65.9}$
Quadruped Stand	351.6 ± 68.5	351.1 ± 25.7	378.7 ± 37.3	752.2 ± 271.9	804.7 ± 211.5	$\mathbf{941.6 \pm 25.7}$
Quadruped Walk	119.0 ± 40.9	171.6 ± 11.2	391.8 ± 57.9	$\mathbf{844.7 \pm 138.7}$	656.1 ± 321.2	759.9 ± 177.5
Walker Flip	39.6 ± 17.7	25.0 ± 2.2	$\mathbf{913.6 \pm 2.5}$	249.1 ± 230.8	544.0 ± 313.1	856.9 ± 64.5
Walker Run	24.5 ± 6.1	22.4 ± 1.6	$\mathbf{706.2 \pm 7.9}$	496.8 ± 264.3	690.7 ± 101.9	653.6 ± 26.7
Walker Stand	168.5 ± 48.9	181.1 ± 135.8	846.2 ± 256.9	574.2 ± 209.3	810.4 ± 250.3	$\mathbf{909.4 \pm 96.9}$
Walker Walk	35.1 ± 29.6	25.3 ± 2.6	738.9 ± 399.7	725.3 ± 234.8	792.8 ± 242.2	$\mathbf{916.5 \pm 43.4}$

Table 2: Returns achieved by BC, IQ-Learn, OPOLO, state-only MM, GAIfO and SFM across tasks on the DMControl Suite. The average returns and standard deviation across 10 seeds are reported.

Environment	MM (TD3)	GAIfO (TD3)	SFM (TD3)
Cheetah Run	439.6 ± 138.6	$\mathbf{674.0 \pm 27.7}$	514.7 ± 77.9
Cheetah Walk	859.6 ± 165.3	$\mathbf{873.9 \pm 58.2}$	829.7 ± 226.3
Quadruped Jump	308.8 ± 115.9	334.3 ± 159.8	$\mathbf{821.6 \pm 27.5}$
Quadruped Run	107.0 ± 22.8	94.4 ± 23.1	$\mathbf{705.6 \pm 57.3}$
Quadruped Stand	449.7 ± 206.1	381.8 ± 216.0	$\mathbf{946.3 \pm 20.5}$
Quadruped Walk	201.0 ± 175.8	347.3 ± 246.4	$\mathbf{829.3 \pm 86.9}$
Walker Flip	328.7 ± 287.8	774.4 ± 276.1	$\mathbf{865.5 \pm 37.7}$
Walker Run	530.2 ± 163.1	600.9 ± 105.0	$\mathbf{606.5 \pm 30.2}$
Walker Stand	575.8 ± 245.8	764.8 ± 220.7	$\mathbf{934.0 \pm 49.6}$
Walker Walk	395.1 ± 351.1	769.7 ± 258.1	$\mathbf{880.1 \pm 75.8}$

Table 3: Comparison of state-only IRL methods using the weaker TD3 policy optimizer. This table presents returns achieved by state-only MM (TD3), GAIfO (TD3) and SFM (TD3) across tasks on the DMControl Suite. The average returns and standard deviation across 10 seeds are reported.

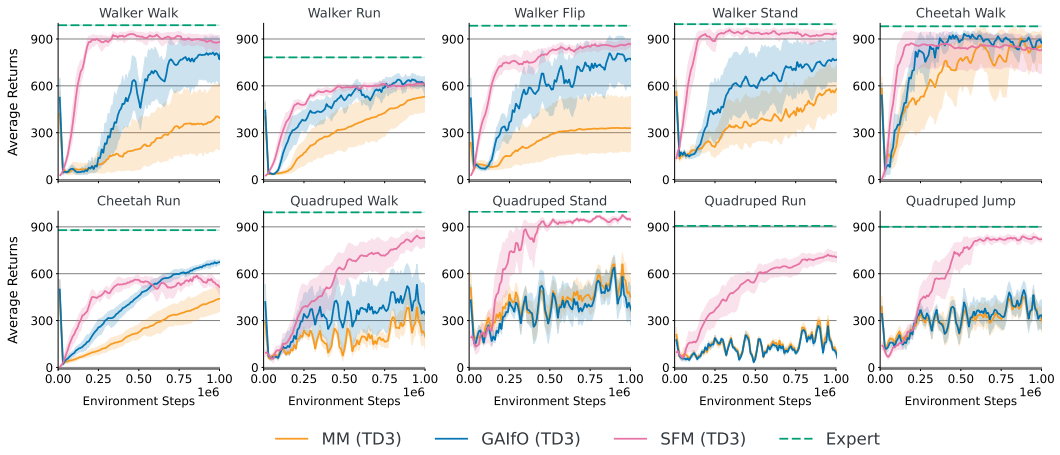

Figure 6: Comparison of state-only IRL methods using the weaker TD3 policy optimizer. Notably, only SFM consistently maintains strong performance with the weaker policy optimizer.

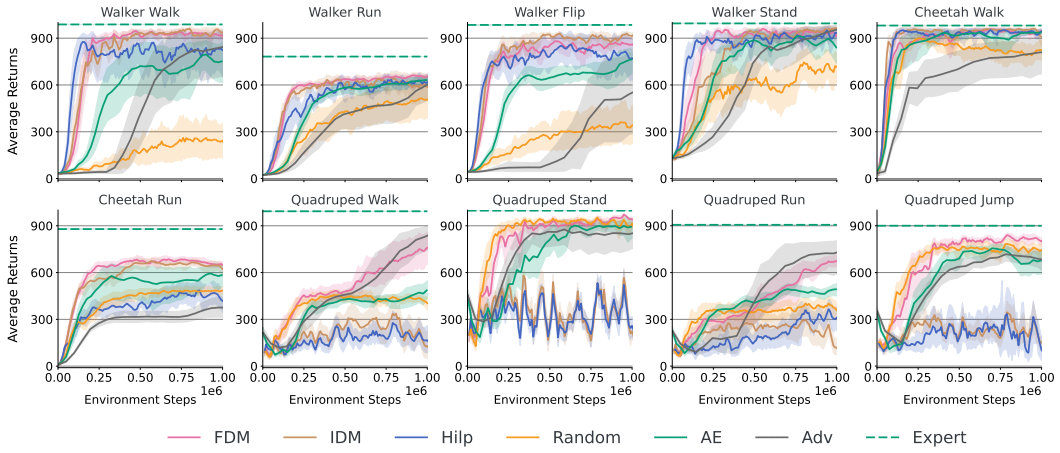

Figure 7: Effect of different base feature functions on the performance of the agent. Here, we compare with Random, Inverse Dynamics Model (IDM) (Pathak et al., 2017), Hilbert Representations (Hilp) (Park et al., 2024), Autoencoder (AE), Adversarial representations (Adv) and Forward Dynamics Models (FDM). FDM was found to work best across DMC tasks. Note that all base feature functions were jointly learned during training.

E SFM WITH STOCHASTIC POLICY

To extend SFM to stochastic policies, we propose having an agent comprising of a stochastic actor parameterized to predict the mean and standard deviation of a multi-variate gaussian distribution. Here, for a given state s, the action is sampled using $a \sim \pi_\mu(.|s)$. The SF network architecture ψ_θ is same as the SFM (TD3) variant, where the network estimates the SF for a state-action pair. The SF-network can be updated using 1-step TD error using the base features of the current state similar to equation 1. To update the actor, we first propose a modification of Proposition 1 to estimate the expected features of the agent for the initial state distribution for a stochastic policy.

Proposition 4. *Let $\mathcal{B}$ denote a buffer of trajectories sampled from arbitrary stationary Markovian policies in the given MDP with initial state distribution P_0. For any stochastic policy π,*

$$\widehat{\boldsymbol{\psi}}^{\pi} := (1-\gamma)^{-1}\mathbb{E}_{(s_t,s_{t+1})\sim\mathcal{B}}\left[\mathbb{E}_{a_t\sim\pi(s_t)}[\boldsymbol{\psi}^{\pi}(s_t,a_t)] - \gamma\mathbb{E}_{a_{t+1}\sim\pi(s_{t+1})}[\boldsymbol{\psi}^{\pi}(s_{t+1},a_{t+1})]\right]. \quad (28)$$

Environment	Random	AE	Hilp	IDM	FDM	Adv
Cheetah Run	484.4 ± 45.4	585.7 ± 93.6	417.8 ± 118.8	622.0 ± 69.5	$\mathbf{648.8 \pm 35.9}$	374.1 ± 113.7
Cheetah Walk	823.8 ± 107.6	$\mathbf{938.4 \pm 18.8}$	$\mathbf{944.5 \pm 18.6}$	908.2 ± 88.4	$\mathbf{945.1 \pm 33.9}$	812.5 ± 244.5
Quadruped Jump	744.4 ± 79.0	678.0 ± 126.6	101.5 ± 78.5	151.3 ± 59.0	$\mathbf{799.1 \pm 47.8}$	679.7 ± 144.1
Quadruped Run	356.8 ± 92.7	493.4 ± 57.6	311.8 ± 94.5	118.0 ± 86.1	671.7 ± 65.9	$\mathbf{737.0 \pm 196.8}$
Quadruped Stand	914.0 ± 33.1	895.3 ± 94.2	259.1 ± 102.0	222.0 ± 63.3	$\mathbf{941.6 \pm 25.7}$	858.4 ± 140.3
Quadruped Walk	402.8 ± 68.7	489.7 ± 68.4	166.0 ± 138.8	129.0 ± 145.6	759.9 ± 177.5	$\mathbf{849.1 \pm 140.8}$
Walker Flip	341.8 ± 204.4	765.5 ± 101.8	771.9 ± 197.0	$\mathbf{912.8 \pm 36.9}$	856.9 ± 64.5	565.1 ± 439.9
Walker Run	506.6 ± 181.8	632.2 ± 41.7	615.7 ± 53.8	589.5 ± 139.0	$\mathbf{653.6 \pm 26.7}$	620.5 ± 158.2
Walker Stand	715.9 ± 160.9	836.2 ± 140.0	934.8 ± 42.8	$\mathbf{960.6 \pm 22.9}$	909.4 ± 96.9	$\mathbf{964.2 \pm 31.8}$
Walker Walk	243.8 ± 189.7	752.5 ± 164.9	821.8 ± 242.5	$\mathbf{936.3 \pm 48.8}$	$\mathbf{916.5 \pm 43.4}$	849.6 ± 289.8

Table 4: Effect of different base feature functions on the performance of the agent. Here, we compare with Random, Inverse Dynamics Model (IDM) (Pathak et al., 2017), Hilbert Representations (Hilp) (Park et al., 2024), Autoencoder (AE), Adversarial representations (Adv), and Forward Dynamics Models (FDM). The table reports the returns achieved by each base feature function when trained with SFM across tasks on the DMControl Suite. The average returns and standard deviation across 10 seeds are reported. FDM was found to work best across DMC tasks. Note that all base feature functions were jointly learned during training.

Proof. In the proof of Proposition 1, we saw that

$$\mathbb{E}_{(s_t,s_{t+1})\sim\mathcal{B}}[f(s_t) - \gamma f(s_{t+1})] = (1-\gamma)\mathbb{E}_{s\sim P_0}[f(s)] \tag{29}$$

for any $f : \mathcal{S} \to \mathbb{R}^d$. Substituting $f : s \mapsto \mathbb{E}_{a\sim\pi(s)}[\boldsymbol{\psi}^\pi(s,a)]$, we have

$$\begin{aligned}
&(1-\gamma)^{-1}\mathbb{E}_{(s_t,s_{t+1})\sim\mathcal{B}}\left[\mathbb{E}_{a_t\sim\pi(s_t)}[\boldsymbol{\psi}^\pi(s_t,a_t)] - \gamma\mathbb{E}_{a_{t+1}\sim\pi(s_{t+1})}[\boldsymbol{\psi}^\pi(s_{t+1},a_{t+1})]\right] \\
&\quad= \mathbb{E}_{s\sim P_0}\mathbb{E}_{a\sim\pi(s)}[\boldsymbol{\psi}^\pi(s,a)] && \text{Equation 29} \\
&\quad\equiv \widehat{\boldsymbol{\psi}}^\pi,
\end{aligned}$$

completing the proof. □

Notably, in light of Proposition 4, Proposition 2 can again be used to update the policy parameters, defining

$$\begin{aligned}
\nabla_\mu\mathcal{L}_G(\mu) &= \sum_{i=1}^{d} z_i(1-\gamma)^{-1}\mathbb{E}_{s,s'\sim\mathcal{B}}\mathbb{E}_{a\sim\pi_\mu(s)}\left[\nabla_\mu\pi_\mu(s)\,\nabla_a\boldsymbol{\psi}_{\theta,i}(s,a)\right] \\
z &:= (1-\gamma)^{-1}\mathbb{E}_{s,s'\sim\mathcal{B}}\left[\boldsymbol{\psi}_\theta(s,\pi_\mu(s)) - \gamma\boldsymbol{\psi}_{\bar{\theta}}(s',\pi_\mu(s'))\right] - \widehat{\boldsymbol{\psi}}^E.
\end{aligned} \tag{30}$$

For certain policy parameterizations (e.g., Gaussian policies), the reparameterization trick Kingma & Welling (2013); Haarnoja et al. (2018) can be used to directly estimate the gradient $\nabla_\mu\pi_\mu(s)$ in Equation 30. We note that, to compute an unbiased gradient from samples, minibatches of (s, s') pairs must be sampled independently for the computation of z. That is, we estimate the gradient as follows,

$$\begin{aligned}
\nabla_\mu\mathcal{L}_G(\mu) &\approx \sum_{i=1}^{d}\hat{z}_i(1-\gamma)^{-1}\frac{1}{N_1}\sum_{j=1}^{N_1}\nabla_\mu\pi_\mu(s_{1,j})\nabla_a\boldsymbol{\psi}^\pi_{\theta,i}(s_{1,j},a)\big|_{a=a_{1,j}} \\
\hat{z} &= (1-\gamma)^{-1}\frac{1}{N_2}\sum_{j=1}^{N_2}\left[\boldsymbol{\psi}^\pi_\theta(s_{2,j},a_{2,j}) - \gamma\boldsymbol{\psi}^\pi_{\bar{\theta}}(s'_{2,j},a'_{2,j})\right] - \widehat{\boldsymbol{\psi}}^E \\
\{s_{1,j}\}_{j=1}^{N_1} &\overset{\text{iid}}{\sim} \mathcal{B}, \quad a_{1,j}\sim\pi_\mu(s_{1,j}) \\
\{(s_{2,j},s'_{2,j})\}_{j=1}^{N_2} &\overset{\text{iid}}{\sim} \mathcal{B}, \quad a_{2,j}\sim\pi_\mu(s_{2,j}), a'_{2,j}\sim\pi_\mu(s'_{2,j}).
\end{aligned} \tag{31}$$

Finally, to prevent the policy from quickly collapsing to a nearly-deterministic one, we also include a policy entropy bonus in our actor updates. The pseudocode is given in Algorithm 2.

We conduct experiments over the tasks from DMControl suite and present environment plots in Figure 8 and returns achieved in Table 5. Here, we observe that SFM can learn with a stochastic policy optimizer and achieve comparable performance with the deterministic subroutines.

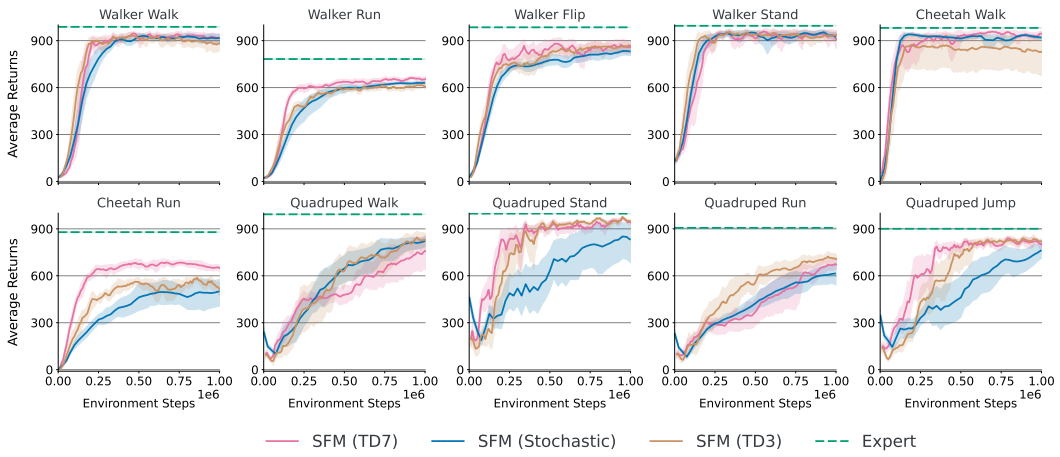

Figure 8: Comparison of variants of SFM with TD7, TD3 and an entropy regularized stochastic policy. We observe that SFM can be trained with stochastic polices. However, the variants with deterministic policy optimizers work better on some tasks than the stochastic policy.

Algorithm 2 Successor Feature Matching (SFM) (Stochastic)

Require: Expert demonstrations $\tau^E = \{s^i_0, a^i_0, \ldots, s^i_{T-1}, a^i_{T-1}\}^M_{i=1}$
Require: Base feature loss $\mathcal{L}_{\mathsf{feat}}$ and initialized parameters $\theta_{\mathsf{feat}} = (\phi, \mathsf{f})$
Require: Initialized actor π_μ, SF network with targets $\boldsymbol{\psi}_\theta, \boldsymbol{\psi}_{\bar{\theta}}$, replay buffer $\mathcal{B}$
while Training **do**
  Observe state s and execute action $a = \pi_\mu(s)$ to get next state s'
  Add transition to replay buffer $\mathcal{B} \leftarrow \mathcal{B} \cup (s, a, s')$
  Compute expected features of expert $\hat{\boldsymbol{\psi}}^E = \frac{1}{M}\sum_{i=1}^{M}\sum_{t=0}^{T-1}\gamma^t\phi(s^i_t)$
  Sample minibatch $\mathcal{D} = \{(s, a, s')\} \sim \mathcal{B}$
  Sample action at state s' using $a' \sim \pi_\mu(.|s')$
  Update SF network using $\nabla_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \|\phi(s) + \boldsymbol{\psi}_{\bar{\theta}}(s', a') - \boldsymbol{\psi}_\theta(s, a)\|^2_2$, where $a' \sim \pi_\mu(s')$
  Compute $\hat{\boldsymbol{\psi}}^\pi = (1-\gamma)^{-1}\mathbb{E}_{s,s'\sim\mathcal{D}}[\mathbb{E}_{a\sim\pi(s)}[\boldsymbol{\psi}_\theta(s, a)] - \gamma\mathbb{E}_{a'\sim\pi(s')}[\boldsymbol{\psi}_{\bar{\theta}}(s', a')]]$
  Update actor with $\nabla_\mu \frac{1}{2}\|\hat{\boldsymbol{\psi}}^\pi - \hat{\boldsymbol{\psi}}^E\|^2_2 + \alpha\mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_\mu(s)}[\log \pi_\mu(a|s)]$—see Equation 31
  Update base feature parameters using $\nabla_{\theta_{\mathsf{feat}}}\mathcal{L}_{\mathsf{feat}}(\theta_{\mathsf{feat}})$
end while

Environment	SFM (TD7)	SFM (TD3)	SFM (Stochastic)
Cheetah Run	648.8 ± 35.9	514.7 ± 77.9	500.9 ± 136.3
Cheetah Walk	945.1 ± 33.9	829.7 ± 226.3	918.8 ± 21.7
Quadruped Jump	799.1 ± 47.8	821.6 ± 27.5	764.0 ± 84.8
Quadruped Run	671.7 ± 65.9	705.6 ± 57.3	614.9 ± 113.4
Quadruped Stand	941.6 ± 25.7	946.3 ± 20.5	829.5 ± 224.1
Quadruped Walk	759.9 ± 177.5	829.3 ± 86.9	821.9 ± 56.3
Walker Flip	856.9 ± 64.5	865.5 ± 37.7	830.4 ± 45.4
Walker Run	653.6 ± 26.7	606.5 ± 30.2	630.6 ± 17.5
Walker Stand	909.4 ± 96.9	934.0 ± 49.6	925.7 ± 38.9
Walker Walk	916.5 ± 43.4	880.1 ± 75.8	916.3 ± 43.9

Table 5: Comparison of SFM with a stochastic policy and variants based on deterministic policy optimizers (TD3 & TD7). The table reports the returns achieved by each base feature function when trained with SFM across tasks on the DMControl Suite. The average returns and standard deviation across 10 seeds are reported.