BRIDGING SUCCESSOR MEASURE AND ONLINE POLICY LEARNING WITH FLOW MATCHING-BASED REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

The Successor Measure (SM), a powerful method in reinforcement learning (RL), describes discounted future state distributions under a policy, and it has recently been studied using generative modeling techniques. Although SM is a powerful predictive object, it lacks compact representations tailored for online RL. To address this, we introduce *Successor Flow Features* ($\texttt{SF}^2$), a representation learning framework that bridges SM estimation with policy optimization. $\texttt{SF}^2$ leverages flow-matching generative models to approximate successor measures, while enforcing a structured linear decomposition into a time-invariant embedding and a time-dependent projection. This yields compact, policy-aware state-action features that integrate readily into standard off-policy algorithms like TD3 and SAC. Experiments on DeepMind Control Suite tasks show that $\texttt{SF}^2$ improves sample efficiency and training stability compared to strong successor feature baselines. We attribute these gains to the compact representation induced by flow matching, which reduces compounding errors in long-horizon predictions.

1 INTRODUCTION

A key reason for the success of deep reinforcement learning (RL) in complex, sequential real-world tasks is its ability to learn meaningful representations automatically. This is often achieved through neural network architectures with specific inductive bias and efficient representation training algorithms. Effective representations generalize well across different observations and environments, give accurate value estimation, make efficient planning over long decision sequences, and achieve robustness when encountering new observed states (Kulkarni et al., 2016). However, how to find a general and efficient representation learning method that does not rely entirely on reward functions but focuses on environment dynamics is a crucial problem, especially for complex tasks with a continuous state space, sparse reward, and long decision sequences.

Successor Representation (SR) (Dayan, 1993) offers a promising approach by decoupling the reward function from environment dynamics. It captures the expected future state occupancy under a given policy, offering a dual interpretation: it can be viewed as a linear basis for state-action value functions, or equivalently, as a compact representation of infinite-horizon discounted visitation distributions. However, SR is inherently limited to discrete state spaces. To overcome this limitation, Successor Feature (SF) (Barreto et al., 2017) were introduced, incorporating a reward-relevant feature mapping along with a Temporal Difference (TD)-style learning algorithm. A key challenge, however, remains the design of an appropriate feature mapping, which is still an open problem (Ollivier, 2025).

More recently, **Successor Measure** (SM) emerged as a generalization of SR that directly models the discounted future state occupancy. Unlike SF, which relies on predefined features, SM describes distributions in principle in infinite-dimensional space and is typically estimated via generative models. Recent advances such as the *geometric horizon model* (GHM) (Thakoor et al., 2022), the γ-*model* (Janner et al., 2020), and TDFlow (Farebrother et al., 2025) utilize generative modeling and generalized TD learning to estimate SM, facilitating applications in policy evaluation and transfer learning. In particular, TDFlow builds on recent progress in generative modeling, specifically **flow matching** techniques (Lipman et al., 2022), enabling direct, simulation-free trajectory generation

between distributions. This makes it highly efficient for continuous and high-dimensional settings. Furthermore, the mixed nature of flow matching aligns naturally with TD-style training.

The stability and efficiency of flow matching make it exceptionally well-suited for modeling SM, as it mitigates compounding errors over long horizons and enhances scalability in complex environments. However, **online RL** imposes stricter demands, requiring representations that not only retain the predictive power of SM but also adapt quickly to new experiences. These requirements motivate a new framework that unifies the robust long-term forecasting of SM, the stable and efficient generative learning of flow matching, and the rapid adaptability essential for online RL.

In this paper, we introduce **Successor Flow Features** ($\texttt{SF}^2$), a new framework that leverages flow matching to approximate SM. $\texttt{SF}^2$ enforces a structured decomposition into a **time-invariant low-dimensional embedding** of state-action pairs and a **time-dependent projection operator**. This design offers several key advantages: (i) the time-invariant embedding is tailored for online RL, enabling joint training with value functions and seamless integration into existing algorithms like TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018); (ii) the time-dependent projection enables generative models to reconstruct SM while decoupling policy-dependent and environmental structure. We evaluate $\texttt{SF}^2$ by integrating it into TD3 and SAC on continuous-control benchmarks from Mujoco Playground (Zakka et al., 2025). Our results demonstrate improved average performance over standard baselines. While preliminary, these findings indicate that combining flow-based generative modeling with successor representations is a highly promising direction for scalable RL representation learning. Our contributions are threefold:

- We propose a generative model with a linear projection structure for approximating SM using flow matching.
- We introduce $\texttt{SF}^2$, an informative representation for online RL that is trained jointly with value functions.
- We provide empirical evaluation on challenging continuous-control tasks, together with diagnostic studies of bootstrapped flow matching.

To the best of our knowledge, $\texttt{SF}^2$ is the first approach to explicitly integrate successor measures with policy optimization for online RL representation learning. We emphasize that $\texttt{SF}^2$ is an initial step in this direction rather than a complete solution: rigorous theoretical guarantees and broader empirical validation remain open challenges.

2 PRELIMINARY

In this paper, we use uppercase serif fonts to denote a set S, $\mathscr{P}(\mathsf{S})$ to denote the space of probability measures over a measurable set S, uppercase capital letters to denote random variables (e.g., S) and $\mathbf{R}^n$ to denote the n-dimensional real space.

Markov Decision Process We consider a discounted Markov decision process $\mathcal{M} = (\mathsf{S}, \mathsf{A}, P, r, \gamma)$, which includes the state space S, discrete or continuous action space A, transition kernel $P : \mathsf{S} \times \mathsf{A} \to \mathscr{P}(\mathsf{S})$, reward function $r : \mathsf{S} \times \mathsf{A} \to \mathbf{R}$, and discount factor $\gamma \in [0, 1)$. Following the setting of Blier et al. (2021), the state space is measurable (either continuous or discrete). In an MDP, an agent interacts with the environment by observing the current state $s_t \in \mathsf{S}$, selecting an action $a_t \in \mathsf{A}$ according to policy π, and then receiving a reward $r(s_t, a_t)$ while transitioning to a new state $s_{t+1} \sim P(\cdot|s_t, a_t)$. The objective of reinforcement learning algorithms is to find a policy $\pi : \mathsf{S} \to \mathscr{P}(\mathsf{A})$ that maximizes the expected cumulative reward, or the value function $V^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s\right]$ for any state $s \in \mathsf{S}$, where $\mathbb{E}^\pi$ denotes expectation under the distribution induced by policy π interacting with the MDP. The value function satisfies the Bellman equation: $V^\pi(s) = \sum_{a \in \mathsf{A}} \pi(a|s) \left[r(s, a) + \gamma \sum_{s' \in \mathsf{S}} P(s'|s, a) V^\pi(s')\right]$.

Flow Matching Flow Matching (FM) (Lipman et al., 2022) is a technique used in generative modeling to learn mappings between distributions. Define a time-dependent diffeomorphic map $\phi_k = \phi(\cdot, k) : \mathbf{R}^n \times [0, 1] \to \mathbf{R}^n$ governed by an Ordinary Differential Equation (ODE) : $\frac{dx_k}{dk} = v(x_k, k)$ with time $k \in [0, 1]$, where $x_k := \phi_k(x_0)$. We use the notation k rather than t for the ODE's time parameter to distinguish it from the timestep t used in MDPs. A Continuous Normalizing Flow, one kind of Neural Ordinary Differential Equations (Chen et al., 2018), is employed to parameterize the vector field $v_k = v(\cdot, k) : \mathbf{R}^n \times [0, 1] \to \mathbf{R}^n$ as $u_\theta(\cdot, k)$ and determines the flow dynamics.

To find the training target of the parameterized time-dependent vector field, FM introduces a mixture representation approach to estimate the marginal vector field as a mixture of conditional vector fields that condition on each data point from p_{target}. For example, given a prior distribution $p_0(x) = \mathcal{N}(0, I_n)$ and a sampled data point x_1 from the target distribution $p_1(x) := p_{\text{target}}(x)$, FM constructs conditional probability paths $p_k(x|x_1) = \mathcal{N}(x; \mu_k(x_1), \sigma_k^2(x_1)I_n)$. The corresponding conditional time-dependent diffeomorphic map and the conditional vector field are given by $\phi_k(x, x_1) = \sigma_k(x_1)x + \mu_k(x_1)$ and $\frac{d\phi_k(x,x_1)}{dk}$, respectively. The training objective is

$$\mathcal{L}(\theta) = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I),x_1\sim p_{\text{target}}(x),k\sim\mathcal{U}(0,1)}\left[\left\|u_\theta(\phi_k(\epsilon,x_1),k) - \frac{d\phi_k(\epsilon,x_1)}{dk}\right\|^2\right],$$

where $\mathcal{U}(0,1)$ represents the uniform distribution over the interval $[0,1]$. This objective minimizes the squared difference between the parameterized vector field and the target vector field at randomly sampled time points, states, and noise values. The time parameter k is uniformly sampled to ensure the model learns the entire trajectory from the prior distribution to the target distribution.

Samples are generated by solving $x_1 = x_0 + \int_0^1 u_\theta(x_k, k)dk$ where $x_0 \sim p_0$ with standard ODE solvers (Gautschi, 2011). When additional conditions are imposed as c for the vector fields $u_\theta(x, k, c)$ and the sampled data conforms to the distribution $x \sim q(\cdot|\, c)$, the FM is also capable of constructing conditional generative models.

Successor Measure The SM (Blier et al., 2021) is a probability distribution over states that captures the expected discounted future state visitations under a given policy, a transition kernel, and a state-action pair. Formally, for a policy π, the SM $\mu^\pi(\mathsf{X}|s,a)$ represents the probability of visiting state $s' \in \mathsf{X} \subseteq \mathsf{S}$ when starting from state-action pair (s,a) and following policy π, with geometric discounting:

$$\mu^\pi(\mathsf{X}|s,a) = (1-\gamma)\mathbb{E}_{(s_1,s_2,\ldots,s_t,\ldots)\sim P^\pi}\left[\sum_t \gamma^t \mathbb{1}_{s_t\in\mathsf{X}} \mid s_0 = s, a_0 = a\right],$$

where the expectation is taken over all possible trajectories generated by starting at state s, taking action a, and then following policy π for all subsequent steps. The indicator function $\mathbb{1}_{s_t\in\mathsf{X}}$ equals 1 when the state at time t belongs to the set X and 0 otherwise.

Similar to the Bellman equation for the value function, the SM satisfies the Bellman equation (Blier et al., 2021):

$$\mu^\pi(\mathsf{X}|s,a) = (1-\gamma)P(\mathsf{X}|s,a) + \gamma\sum_{s'\in\mathcal{S}} P(s'|s,a)\sum_{a'\in\mathcal{A}}\pi(a'|s')\mu^\pi(\mathsf{X}|s',a'). \tag{1}$$

This recursive formulation reveals that the SM can be interpreted as a mixture distribution between the immediate state distribution induced by the transition kernel (with weight $1-\gamma$) and the bootstrapped future state distribution (with weight γ). The conditional generative models can be employed to predict the SM utilizing the recursive form equation 1. In general, the learning objective is formulated as a maximum likelihood estimation problem, which aims to find the optimal generative model by solving:

$$\max_\mu \mathbb{E}_{\mathsf{X}\sim(1-\gamma)P(\cdot|s,a)+\gamma\mathbb{E}_{s'\sim P^\pi(\cdot|s,a),a'\sim\pi(\cdot|s')}[\mu^\pi(\cdot|s',a')]}[\log\mu^\pi(\mathsf{X}|s,a)]. \tag{2}$$

While this objective provides a general framework, the specific loss function needs to be adapted according to the choice of generative model. For instance, in the Geometric Horizon Model (GHM) (Thakoor et al., 2022) (also referred to as the γ-model (Janner et al., 2020)), different implementations employ distinct training losses, such as those based on VAE (Kingma et al., 2013) or GAN (Goodfellow et al., 2014). In this work, we focus on the best-performing variant, the flow matching used in TDFlow (Farebrother et al., 2025), which utilizes a modified flow matching loss. We will elaborate on its details in the next section. A further discussion about the benefits of using flow matching for SM through an explicit mixture viewpoint can be found in Appendix A.2.

3 SUCCESSOR FLOW FEATURE

3.1 FLOW MATCHING FOR SUCCESSOR MEASURE LEARNING

The SM's mixture structure is particularly well-suited for flow matching approaches, allowing us to directly model the interpolation between immediate transitions and future state distributions. For learning $\mu^\pi(s'|s,a)$ with its corresponding parameterized time-dependent vector field $u_\theta(x,k,s,a)$, we utilize its natural mixture representation:

$$\mu^\pi(s'|s,a) = (1-\gamma)P(s'|s,a) + \gamma \mathbb{E}_{s''\sim P(\cdot|s,a),a''\sim\pi(\cdot|s'')}\mu^\pi(s'|s'',a''),$$

which yields the corresponding FM training objective for the parameterized vector field $u_\theta(x,k,s,a)$ on given tuple (s,a,s'):

$$\begin{aligned}
\mathcal{L}_{\text{flow}}(\theta) =& (1-\gamma)\mathcal{L}_P(\theta) + \gamma \mathcal{L}_{\text{bootstrapping}}(\theta) \\
=& (1-\gamma)\mathbb{E}_{\epsilon,k,s'\sim P(\cdot|s,a)}\left[\left\| u_\theta(\phi_k(\epsilon,s'),k,s,a) - \frac{d\phi_k(\epsilon,s')}{dk}\right\|^2\right] \\
&+ \gamma \mathbb{E}_{\epsilon,k,a'\sim\pi(\cdot|s'),s_e\sim\mu_\theta(\cdot|s',a')}\left[\left\| u_\theta(\phi_k(\epsilon,s_e),k,s,a) - \frac{d\phi_k(\epsilon,s_e)}{dk}\right\|^2\right], \qquad (3)
\end{aligned}$$

where $\mathcal{L}_P(\theta)$ is used to learn the transition distribution and $\mathcal{L}_{\text{bootstrapping}}(\theta)$ uses a temporal difference form where bootstrapping distributions induced by $u_\theta(\cdot,\cdot,s',a')$ serve as $\mu^\pi(\cdot|s',a')$. However, this sampling procedure introduces substantial computational costs due to the need for multiple network evaluations during the generation process.

To enhance training efficiency, we leverage the fact that sampling from the SM can be achieved through an ODE solver(Euler method as an example): $s_e \sim \mu_\theta(\cdot|s,a)$ is equivalent to $s_e = \text{Euler}(\epsilon, u_\theta(\cdot,\cdot,s,a))$ where $\epsilon \sim \mathcal{N}(0, I_n)$. Motivated by the TD^2-CFM loss formulation from (Farebrother et al., 2025), we directly align the vector fields conditioned on different state-action pairs at the same noise level instead of generating full successor states and then comparing them. This approximation avoids expensive ODE integration while preserving the consistency between local flow directions. Intuitively, if two vector fields agree on their evolution, their generated distributions will also agree.

$$\mathcal{L}_{\text{bootstrapping}}(\theta) \approx \mathbb{E}_{\substack{\epsilon,k,a'\sim\pi(\cdot|s') \\ x_k=\text{ODE}(\epsilon,k,u_\theta(\cdot,\cdot,s',a'))}}\left[\|u_\theta(x_k,k,s,a) - u_\theta(x_k,k,s',a')\|^2\right]. \qquad (4)$$

This approach aligns the vector fields conditioned on different state-action pairs at the same noise level, eliminating the need to fully generate denoised states and then apply the ϕ_k transformation. This form substantially reduces the need for small integration steps in the ODE solver, decreasing computational overhead while maintaining performance. We provide a detailed analysis of the trade-off between computational efficiency and model performance in our ablation studies presented in Section 4.3. It is noted that the final loss used for learning is based on expectation over transition tuples from the current policy π. We model conditional vector fields on the latent space induced by the flow parameterization (Section 2). This avoids explicit density ratios on S and ensures training targets are defined even when the SM is provided implicitly via pushforwards.

3.2 SUCCESSOR FLOW FEATURE FROM ESTIMATED SUCCESSOR MEASURE

Following the estimation of the SM, we employ a linear projection formulation to derive a compact feature representation.

Definition 3.1 (Successor Flow Feature) *We define the Successor Flow Feature (SF^2) on state-action pair (s,a) as the output of the mapping $\psi : \mathbf{R}^{dim_S} \times \mathbf{R}^{dim_A} \to \mathbf{R}^d$, which generate the time-dependent conditional vector field $u(s',k,s,a)$ as a linear projection with a time-conditioned matrix field $\zeta : \mathbf{R}^{dim_S} \times [0,1] \to \mathbf{R}^{d\times dim_S}$:*

$$u(s',k,s,a) = \zeta(s',k)^\top \psi(s,a),$$

where $\psi(s,a)$ is time-invariant and captures the sufficient dimension reduction property.

In contrast to conventional conditional generative models that combine conditions, timestamps, and noised inputs through complex non-linear transformations, our approach employs a time-invariant feature $\psi(s, a)$ that interacts only at the final stage with the matrix field ζ. This architectural choice promotes the encoding of temporal structures within ζ that are essential for effective downstream representation learning. This linear projection approach has been explored in prior work (Shribak et al., 2024), which extracts spectral features from environmental transition dynamics to enhance reinforcement learning performance.

The representation function $\psi(s, a)$ achieves the Sufficient Dimension Reduction (SDR) (Fukumizu et al., 2009) by establishing conditional independence $s_e \perp\!\!\!\perp (s, a) \mid \psi(s, a)$, i.e. $\mu^\pi(s_e|s, a) = \mu^\pi(s_e|\psi(s, a))$, thereby ensuring that the extracted representations comprehensively capture all relevant information about how state-action pairs relate to successor states. Additionally, this formulation exhibits universal approximation properties (Sasaki & Hyvärinen, 2018), enabling it to theoretically approximate any target function to arbitrary accuracy, which makes it particularly effective for modeling SM across diverse policies and environments.

3.3 Connection to Successor Representation and Diffusion Spectral Representation

Connection to Successor Representation Let's consider the one step gradient updating on the parameterized ψ neural network with parameters θ using equation 3 with equation 4 under a transition tuple (s, a, s') and sampled $k \sim \mathcal{U}(0, 1), \epsilon \sim \mathcal{N}(0, I_n)$. When $k \to 0$, let $\phi_k(\epsilon, x) = kx + (1 - k)\epsilon$, $\frac{d\phi_k(\epsilon,x)}{dk} = x - \epsilon$, we have:

$$\mathcal{L}_{\text{flow}}(\theta) = (1-\gamma)\left\|\zeta(ks'+(1-k)\epsilon, k)^\top\psi(s, a) - (s'-\epsilon)\right\|^2 + \gamma\left\|\zeta(x_k, k)^\top\psi(s, a) - \zeta(x_k, k)^\top\psi(s', a')\right\|^2.$$

As k approaches 0, we can make the approximation:

$$\zeta(ks' + (1 - k)\epsilon, k) \approx \zeta(\epsilon, 0), \text{ and } \zeta(x_k, k) \approx \zeta(\epsilon, 0).$$

The intermediate point x_k is approximately obtained through a single ODE transformation step:

$$x_k \approx \epsilon + k\zeta(\epsilon, 0)^\top\psi(s', a').$$

Substituting these approximations into the loss function yields the semi-gradient, where we stop the gradient backpropagation on the bootstrapped target. We have

$$\nabla_\theta\mathcal{L} \approx 2\left[(1-\gamma)\left(\zeta(\epsilon, 0)^\top\psi(s, a) - (s'-\epsilon)\right) + \gamma\left(\zeta(\epsilon, 0)^\top\psi(s, a) - \zeta(\epsilon, 0)^\top\psi(s', a')\right)\right]\nabla_\theta\psi(s, a),$$

which can be rewritten more concisely as:

$$\nabla_\theta\mathcal{L} \approx 2\left[\psi(s, a)^\top\zeta(\epsilon, 0) - \left((1-\gamma)(s'-\epsilon) + \gamma\psi(s', a')^\top\zeta(\epsilon, 0)\right)\right]\nabla_\theta\psi(s, a).$$

This formulation reveals a temporal difference learning structure where the target combines: (1) A direct supervision component $(1 - \gamma)(s' - \epsilon)$ representing immediate information. (2) A discounted bootstrapped component $\gamma\psi(s', a')^\top\zeta(\epsilon, 0)$ that propagates future representations.

Rearranging into a Bellman-like equation, we have

$$\psi(s, a) \leftarrow (1-\gamma)(\zeta(\epsilon, 0)^T)^+(s'-\epsilon) + \gamma\psi(s', a'),$$

where $(\cdot)^+$ denotes the Moore-Penrose pseudoinverse. This formulation reveals that our approach learns a Successor Representation with Dayan's definition (Dayan, 1993). In our case, $(1 - \gamma)(\zeta(\epsilon, 0)^T)^+(s' - \epsilon)$ serves as the basic feature that captures immediate transitions, while the recursive structure $\psi(s, a) = (1 - \gamma)$(immediate feature) $+ \gamma\psi(s', a')$ serves as the bootstrapped part. The process incorporates a novel element where the next state $s' - \epsilon$ undergoes Gaussian noise perturbation before being projected onto the column space defined by $(\zeta(\epsilon, 0)^T)^+$. This can be interpreted as learning a basis for the state space that is robust to perturbations, enabling more effective representation of the expected future state occupancy distribution. This yields an SR-like recursion on ψ under fixed ζ. We emphasize this is an approximation to motivate design; we do not claim exact equivalence to SR. The exploration of how the representation behaves and what properties it captures when k takes values significantly away from 0 remains an open question for future investigation.

Connection to Diffusion Spectral Representation (Shribak et al., 2024) As γ approaches zero, our approach bears resemblance to Diffusion Spectral Representation (Shribak et al., 2024), which employs diffusion models (Song et al., 2020) rather than flow matching and targets transition probabilities instead of SM. The incorporation of future state transitions enables features to encode transition dynamics across extended time horizons. In Section 4, we conduct empirical comparisons between $\texttt{SF}^2$ and its variant with $\gamma = 0$ to demonstrate that $\texttt{SF}^2$ achieves better area-under-curve(AUC) performance compared to approaches that focus solely on short-term transition prediction.

3.4 Practical Policy Optimization with Flow Successor Representation

In this paper, we consider combining the proposed representation learning method with standard online reinforcement learning algorithms on continuous action spaces. We choose SAC (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018) as base algorithms. The learned representation is only used for building the state-action value function $Q(\psi_\theta(s, a))$. And the policy will be implicitly influenced through the $\nabla_a Q(\psi_\theta(s, a))$.

To enhance learning stability and performance, we implement two complementary techniques:

Value Alignment: We augment the flow-matching objective with a value prediction component:$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{flow}} + \lambda\mathcal{L}_{\text{value}}$, where λ controls the relative weight of value prediction. The value loss follows the standard temporal difference formulation:

$$\mathcal{L}_{\text{value}} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q(\psi_\theta(s, a)) - (r + \gamma \max_{a'} Q(\psi_\theta(s', a'))\right)^2\right].$$

This approach is compatible with various RL algorithms and can incorporate techniques such as double Q-learning (Van Hasselt et al., 2016) for improved target estimation.

Generative Model Smoothing: We employ exponential moving average (EMA) target networks that update parameters according to $\theta_{\psi'} = (1 - \tau)\theta_{\psi'} + \tau\theta_\psi$ and $\theta_{\zeta'} = (1 - \tau)\theta_{\zeta'} + \tau\theta_\zeta$ during bootstrapping phases, consistent with established flow matching training practices (Lipman et al., 2022). We perform an ablation analysis of the effectiveness of this moving average coefficient in Section 4.3. Furthermore, the EMA-updated parameters θ'_ψ also work in the target network for the estimation of the value function, providing an additional layer of stability to learning dynamics. The overall training objective $\mathcal{L}_{\text{total}}$ for learning $\texttt{SF}^2$, when embedded in off-policy RL, is shown in Algorithm 1. For the TD3-based methods, $y' = r+\gamma \min(Q'_1, Q'_2)$, where Q'_1 and Q'_2 are the target Q-networks evaluated at the next state s' and next action a' sampled from the target policy. For the SAC-based method, $y' = r + \gamma \min(Q'_1, Q'_2) - \alpha \log \pi(a'|s')$, where α is the temperature parameter that determines the trade-off between maximizing expected reward and entropy, and is updated according to the original SAC paper (Haarnoja et al., 2018).

Algorithm 1 Training $\texttt{SF}^2$ within Off-Policy RL

Input: (state, action, next state, next action) tuple (s, a, s', a'), networks $(\psi, \zeta, \psi', \zeta')$ and target for value learning y', which depends on the based algorithm
Sample $\epsilon \sim \mathcal{N}(0, I)$, $k \sim \mathcal{U}(0, 1)$
$s_k = k \cdot s' + \epsilon \cdot (1 - k)$, $s_{target} = s' - \epsilon$
Compute features and next state loss:
$\mathcal{L}_{\text{flow}} = \|\psi(s, a)^T\zeta(s_k, k) - s_{target}\|_2^2$
Generate state x using numerical integration, start with $x = \epsilon$:
$k_{start}, k_{end} = 0, k$
$k_{mid} = \frac{1}{2}(k_{start} + k_{end})$
$dx = \psi'(s', a')^T\zeta'(x+\frac{1}{2}\psi'(s', a')^T\zeta'(x, k_{start}), k_{mid})$
$x = x + (k_{end} - k_{start})dx$
Compute generation loss:
$\mathcal{L}_{\text{bootstrapping}} = \|\psi(s, a)^T\zeta(x, k)-\psi'(s', a')^T\zeta'(x, k)\|_2^2$
$\mathcal{L}_{\text{value}} = (Q(\psi(s, a)) - y')^2$
Return $\mathcal{L}_{\text{total}} = (1 - \gamma)\mathcal{L}_P + \gamma\mathcal{L}_{\text{bootstrapping}} + \lambda\mathcal{L}_{\text{value}}$

4 Empirical Evaluation

4.1 Experimental Setup

We implemented all experiments using JAX (Bradbury et al., 2018) and Deepmind Haiku (Hennigan et al., 2020) to leverage hardware acceleration. For the DeepMind Control Suite (Tassa et al., 2018),

we utilized a GPU-accelerated version, the MuJoCo Playground (Zakka et al., 2025). All algorithms were edited from their respective implementations in the Brax library (Freeman et al., 2021). Each experiment is conducted on a single NVIDIA GeForce RTX 4090 GPU. Detailed architectural specifications, hyperparameter configurations, and environment-specific parameters are provided in Appendix E.

Flow sampling uses Euler integration with 2 function evaluations (NFEs) unless otherwise noted; we sample $k \sim \mathcal{U}(0, 1)$ and base noise $\epsilon \sim \mathcal{N}(0, I)$, and condition vector fields on (s, a) and (s', a') as specified in Section 3. We report wall-clock time for representative settings (Section 4.3) and keep baseline parameter counts comparable by reusing encoder widths and depths across methods (Appendix E).

4.2 Experiments on Continuous Action Spaces with Off-Policy Learning

We evaluate $\texttt{SF}^2$ on seven diverse tasks from the DeepMind Control Suite (Tassa et al., 2018), selected to represent a range of challenges in dynamics complexity, reward structure, and control difficulty. We integrate our approach with two commonly used off-policy algorithms: SAC (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018). For comparison with the SF method, we also include Chua et al. (2024), a strong SF method designed for the online RL setting. This work reports substantial improvements over prior SF approaches. To provide a fair and rigorous comparison, we implemented four baseline methods: TD3Sim/SACSim, which closely follows the approach described in the Chua et al. (2024) with the TD3/SAC algorithm, and TD3SimLap/SACSimLap , which removes the Q-function alignment constraint and incorporates an orthogonality objective for feature learning via the graph Laplacian. All methods use identical neural network architectures, the same number of environment steps, and the same number of parallel environments as our proposed approach.

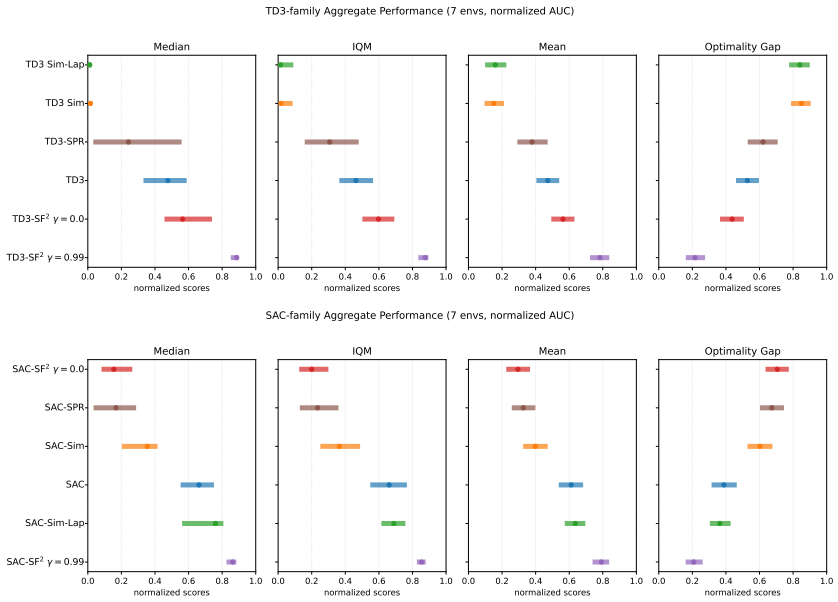


Figure 1: **IQM performance across DeepMind Control tasks.** Panels summarize the aggregate AUC for TD3 (upper) and SAC (lower) variants, comparing vanilla baselines, SF-based baselines, and our $\texttt{SF}^2$ with transition ($\gamma = 0.0$) and successor ($\gamma = 0.99$) horizons.

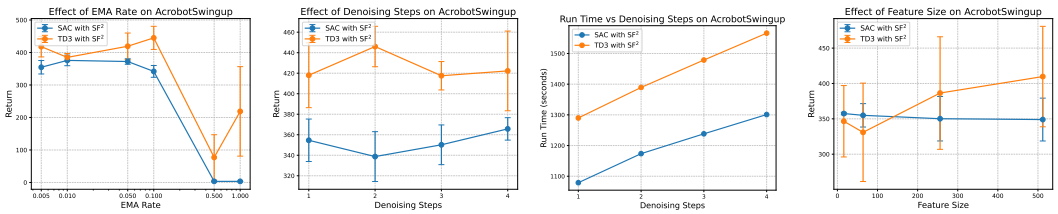
Following the suggestions described in Patterson et al. (2024), we summarize learning dynamics and aggregate performance across seven environments using 15 random seeds for each algorithm variant. For each algorithm variant, we extract evaluation reward curves (mean of episodic returns over evaluation steps using 1000 trajectories) and compute the area-under-curve (AUC). Then the AUC is normalized per environment by linearly scaling to $[0, 1]$ using that environment's minimum and maximum in order to compare across different environment. For each algorithm family (SAC or TD3), the aggregate panels report the median, interquartile mean (IQM; the mean between the 25th and 75th percentiles), mean, and optimality gap (one minus the mean), with 95th percentiles confidence

intervals computed using 5000 samples over normalized scores. The results in Figure 1 reveal that incorporating $\texttt{SF}^2$ enhances the performance of TD3 and SAC across most environments over their standard versions, with higher improvements when using the successor version ($\gamma = 0.99$). Our method consistently outperforms both baseline algorithms and uses the transition version ($\gamma = 0.0$), demonstrating the importance of incorporating a longer horizon in representation learning. Full per-environment learning curves for both TD3 and SAC are deferred to Appendix C.1 (Figure 6) .

Interestingly, the performance gains(compared with baseline) are more pronounced for TD3 than for SAC, suggesting that our method may be particularly beneficial for algorithms that struggle with exploration or representation learning, and a deterministic policy may improve the training efficiency for the online successor measure learning. Additionally, the reduced standard deviations in many cases indicate that $\texttt{SF}^2$ not only improves performance but also enhances stability. Approaches based on SF tend to struggle on sparse reward tasks based on TD3 experiments, as the majority of transitions yield a reward of 0.0, making it difficult to effectively learn the task weight w. In contrast, our method does not depend on this mechanism and thus avoids the associated performance degradation in such settings.

4.3 Hyperparameter Analysis

We examine the influence of three key hyperparameters in our method on the AcrobotSwingup task: exponential moving average (EMA) coefficient, number of denoising steps, and feature size. We use the mean episode return over the final 50k steps to show this.



(a) Analysis of exponential moving average coefficient τ on performance.

(b) Performance evaluation with varying numbers of denoising steps.

(c) Training time evaluation with varying numbers of denoising steps.

(d) Influence of feature size on ψ networks.

Figure 2: Systematic analysis of hyperparameter effects on $\texttt{SF}^2$ performance. (a) Exponential moving average coefficient τ demonstrating stability-performance trade-offs, (b) performance sensitivity to denoising step count in the sampling procedure, (c) computational cost scaling with respect to denoising steps, and (d) feature size effects on algorithm performance.

EMA Parameter. Figure 2a demonstrates that the EMA coefficient τ inversely correlates with performance. Peak results occur at $\tau = 0.1$ for TD3 and $\tau = 0.01$ for SAC, with performance declining as τ approaches 1.0. Despite considerable variance across configurations, these findings suggest that more stable target network updates (smaller τ values) enhance learning dynamics in our framework. The results highlight the critical importance of proper τ calibration, as large values can substantially impair policy effectiveness.

Denoising Steps. Our analysis reveals that varying the number of denoising steps produces comparable performance outcomes (Figure 2b), though computational costs increase proportionally with more steps (Figure 2c). We observe that even with minimal denoising steps (1-2), both algorithms maintain robust performance, suggesting that aligning the bootstrapping part can rely on a rough sampling process without requiring extensive iterative refinement. The computational efficiency analysis in Figure 2c further confirms that a small number of denoising steps provides an optimal balance between performance and computational overhead, as the default choice in experiments.

Feature Size. As shown in Figure 2d, the two algorithms respond differently to feature size changes. SAC with $\texttt{SF}^2$ demonstrates similar final 50k steps returns across various feature sizes with low variance, suggesting effective representation learning even in reduced dimensions. TD3 tends to benefit from larger feature sizes, though variance increases, while SAC remains stable across sizes.

This indicates that the deterministic policy gradient method particularly benefits from richer feature representations.

4.4 Computation Complexity

The additional time overhead of our method is due to the additional representation learning and the additional neural network. Here we use the comparison between TD3 and TD3 with $\texttt{SF}^2$ as an illustration. Each standard TD3 update step consists of one step of critic updating and one step of actor updating, which is consistent with TD3 with $\texttt{SF}^2$. The main distinction is that an additional feature representation update step: for each update step, TD3 with $\texttt{SF}^2$ performs seven forward passes and one backward pass through the ζ network, and two forward passes and one backward pass through the ψ network. Under identical experimental conditions, the original TD3 method's running time on AcrobotSwingup is 659 seconds, while, as reported in Figure 2c, $\texttt{SF}^2$ with 1 denoising step takes approximately 1300 seconds, about twice as long. Our experiments demonstrate significant improvements in downstream performance, justifying this trade-off. We will further optimize training costs in future work.

5 Related Work

Representation learning in RL. Reconstruction-based methods have been employed for feature extraction from observations (Hafner et al., 2019; Yarats et al., 2020). Contrastive learning techniques have emerged as a powerful paradigm for learning discriminative state representations (Laskin et al., 2020; Stooke et al., 2021; Zheng et al., 2023). bisimulation metrics offer a more formal approach to learning state abstractions by grouping behaviorally equivalent states (Zhang et al., 2021; Castro et al., 2021; 2023). World models learn to capture the environment's dynamics, allowing agents to plan or learn in a learned latent space (Gelada et al., 2019; Seo et al., 2022; Hafner et al., 2025). Spectral decomposition methods decompose state and actions into low-rank spectral features (Wang et al., 2021b; Yang & Wang, 2020; Shribak et al., 2024). $\texttt{SF}^2$ uniquely bridges successor measures and online RL: (1) Unlike world models (Gelada et al., 2019; Seo et al., 2022; Hafner et al., 2025) and spectral methods (e.g., (Shribak et al., 2024)) that ignore policy-dependent horizons, $\texttt{SF}^2$ explicitly encodes discounted future distributions via flow-matched successor measures; (2) While reconstruction methods focus on regenerating the observation, our method considers policy and environment dynamics; (3) Bisimulation methods (Zhang et al., 2021; Castro et al., 2021; 2023) emphasize state similarity with reward, but $\texttt{SF}^2$ also optimizes features $\psi(s, a)$ from environmental dynamics not only from the value alignment.

Successor Measure. SM predicts future state distributions under a given policy, effectively capturing the expected discounted future state occupancy. This concept is closely related to Successor Representations (Dayan, 1993) and Successor Features (Barreto et al., 2017), decoupling environment dynamics from reward structures, facilitating efficient policy evaluation and transfer. Blier et al. (2021) offers a formal mathematical definition of SM and introduces how to estimate it for value function evaluation. Wiltzer et al. (2024) further enhances these approaches by modeling the full distribution of future state occupancies, providing richer representations for downstream decision-making. GHMs and γ-models extend the notion of modeling discounted state visitation distributions, creating a continuum between model-free and model-based RL (Thakoor et al., 2022; Janner et al., 2020). Agarwal et al. (2025); Touati & Ollivier (2021) also employ SM to build a representation in the zero-shot RL setting under a precollected offline dataset, always need an extra exploration policy to collect, which allows for optimal policy inference under other given reward functions. $\texttt{SF}^2$ fundamentally advances this paradigm by introducing the subsequent flow characteristics $\psi(s, a)$ (Definition 3.1) as a linear decomposition of the flow field, enabling representation learning, which greatly expands the scope of application of previous methods.

6 Conclusion

In this work, we proposed the Successor Flow Feature ($\texttt{SF}^2$) framework, which leverages flow matching and linear-spectral decomposition to address the challenges of estimating and integrating successor measures in online RL. By explicitly modeling the mixture structure of successor measures,

our method provides useful state-action representations that facilitate efficient online policy learning and planning. Through extensive empirical evaluations across discrete and continuous control tasks, we demonstrated that using $\texttt{SF}^2$ consistently improves performance over standard baselines. Our results underscore the promise of flow-based generative modeling for successor features, paving the way for future research on scalable, expressive, and efficient RL representations.

Reproducibility Statement

We facilitate reproducibility by providing an anonymized source-code repository in the supplementary materials. For every experiment, we specify the random seed used, and we document all implementation and training details in Appendix E and D. Together, these references are sufficient for independent researchers to replicate our reported results.

References

Siddhant Agarwal, Harshit Sikchi, Peter Stone, and Amy Zhang. Proto successor measure: Representing the behavior space of an agent in a dynamical system. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=s9SVlWOcLt.

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pp. 4055–4065. Curran Associates, Inc., 2017.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*, 2021.

Michał Bortkiewicz, Władek Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating Goal-Conditioned RL Algorithms and Research. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/pdf/2408.11052.

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/jax-ml/jax.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. MICo: Improved representations via sampling-based state similarity for markov decision processes. 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=wFp6kmQELgu.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. A kernel perspective on behavioural metrics for markov decision processes. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=nHfPXl1ly7. Expert Certification.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Raymond Chua, Arna Ghosh, Christos Kaplanis, Blake A Richards, and Doina Precup. Learning successor features the simple way. *Advances in Neural Information Processing Systems*, 37: 49957–50030, 2024.

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

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Rémi Munos, Alessandro Lazaric, and Ahmed Touati. Temporal difference flows. *arXiv preprint arXiv:2503.09817*, 2025.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Kenji Fukumizu, Francis R Bach, and Michael I Jordan. Kernel dimension reduction in regression. *The Annals of Statistics*, pp. 1871–1905, 2009.

Walter Gautschi. *Numerical analysis*. Springer Science & Business Media, 2011.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. 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. 2170–2179. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/gelada19a.html.

Ian J 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.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. 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. 2555–2565. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

Tom Hennigan, Trevor Cai, Tamara Norman, Lena Martens, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL http://github.com/deepmind/dm-haiku.

Michael Janner, Igor Mordatch, and Sergey Levine. Generative temporal difference learning for infinite-horizon prediction. *arXiv preprint arXiv:2010.14496*, 2020.

Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

Sotetsu Koyamada, Shinri Okano, Soichiro Nishimori, Yu Murata, Keigo Habara, Haruka Kita, and Shin Ishii. Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45716–45743, 2023.

Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/laskin20a.html.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 2025.

Yann Ollivier. Which features are best for successor features? *arXiv preprint arXiv:2502.10790*, 2025.

Andrew Patterson, Samuel Neumann, Martha White, and Adam White. Empirical design in reinforcement learning. *Journal of Machine Learning Research*, 25(318):1–63, 2024.

Stefano Peluchetti. Non-denoising forward-time diffusions. *arXiv preprint arXiv:2312.14589*, 2023.

Jing Peng and Ronald J Williams. Incremental multi-step q-learning. In *Machine Learning Proceedings 1994*, pp. 226–232. Elsevier, 1994.

Hiroaki Sasaki and Aapo Hyvärinen. Neural-kernelized conditional density estimation. *arXiv preprint arXiv:1806.01754*, 2018.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=Bf6on28H0Jv.

Dmitry Shribak, Chen-Xiao Gao, Yitong Li, Chenjun Xiao, and Bo Dai. Diffusion spectral representation for reinforcement learning. *arXiv preprint arXiv:2406.16121*, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning, 2021. URL https://openreview.net/forum?id=_SKUm2AJpvN.

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.

Shantanu Thakoor, Mark Rowland, Diana Borsa, Will Dabney, Rémi Munos, and André Barreto. Generalised policy improvement with geometric policy composition. In *International Conference on Machine Learning*, pp. 21272–21307. PMLR, 2022.

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

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72, 1969.

Gefei Wang, Yuling Jiao, Qian Xu, Yang Wang, and Can Yang. Deep generative learning via schrödinger bridge. In *International conference on machine learning*, pp. 10794–10804. PMLR, 2021a.

Tianhao Wang, Dongruo Zhou, and Quanquan Gu. Provably efficient reinforcement learning with linear function approximation under adaptivity constraints. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 13524–13536. Curran Associates, Inc., 2021b. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/70a32110fff0f26d301e58ebbca9cb9f-Paper.pdf.

Harley Wiltzer, Jesse Farebrother, Arthur Gretton, Yunhao Tang, André Barreto, Will Dabney, Marc G Bellemare, and Mark Rowland. A distributional analogue to the successor representation. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 52994–53016. PMLR, 2024. URL https://proceedings.mlr.press/v235/wiltzer24a.html.

Lin Yang and Mengdi Wang. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10746–10756. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/yang20h.html.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images, 2020. URL https://openreview.net/forum?id=HklE01BYDB.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carlo Sferrazza, Yuval Tassa, and Pieter Abbeel. Mujoco playground: An open-source framework for gpu-accelerated robot learning and sim-to-real transfer., 2025. URL https://github.com/google-deepmind/mujoco_playground.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

Ruijie Zheng, Xiyao Wang, Yanchao Sun, Shuang Ma, Jieyu Zhao, Huazhe Xu, Hal Daumé III, and Furong Huang. $\texttt{TACO}$: Temporal latent action-driven contrastive loss for visual reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ezCsMOy1w9.

A THEORETICAL FOUNDATION

A.1 CHALLENGES OF MIXED DISTRIBUTION IN GHMS

The learning procedure for GHMs using generative models, particularly those without explicit density estimation, follows a principled sampling-based strategy maximizing the objective in equation 2. Transition tuples $\{s, a, s'\}$ are sampled from given environment interactions or a pre-collected dataset. For each tuple, an indicator variable is sampled from a Bernoulli distribution with parameter γ to determine which distribution to be learned by GHMs: 1) with probability $1 - \gamma$, learn from the true observed successor state s'; and 2) with probability γ, learn from bootstrapped samples, where a state is sampled from the generative model, $\mu(\cdot|s', a')$, conditioned on the observed next state and a next action sampled according to the current policy. This procedure implements the recursive structure of the normalized successor measure (NSM) by sampling.

While theoretically sound, mixture-based learning struggles when γ is close to 1 [1]: the data distribution is dominated by the bootstrapped samples from the generative model, introducing significant bias and inaccuracy—especially early in training—with minimal anchoring to real data. In the following section, we demonstrate this via a Gaussian mixture example, where learning the mixed distribution grows increasingly difficult as γ approaches 1.

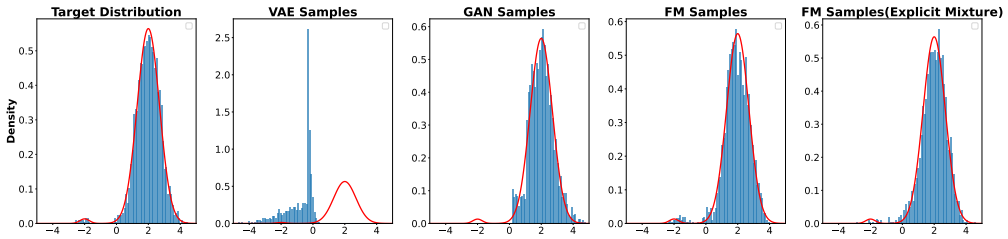


Figure 3: Visualization shows flow-matching methods better capture the multimodal structure of Gaussian mixtures compared to other generative models, such as GANs and VAEs, which is crucial for SM learning with explicit mixture targets.

A.2 EMPIRICAL VALIDATION ON MIXTURE DISTRIBUTIONS (GAUSSIAN AS AN EXAMPLE)

We evaluate several generative models on a one-dimensional Gaussian mixture that is a simplified yet insightful setting. The target distribution is shown in the left-most part of Figure 3. We train VAEs, GANs, and FM models, and observe that learning becomes difficult as the mixing parameter $\gamma \approx 1$.

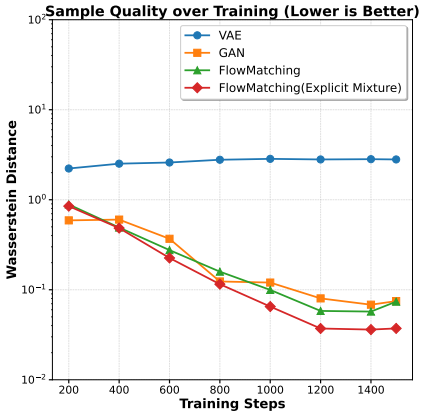


Figure 4: Wasserstein distance between true distribution and generated samples.

Under $\gamma = 0.99$, Figure 3 provides visualizations of the final generated samples, and Figure 4 reports the Wasserstein distance (Vaserstein, 1969) between generated samples and the ground truth

[1]To obtain a longer horizon length and align with the discount factor used in the value function definition, the discount factor γ is expected to be very close to 1, e.g. $\gamma = 0.99$.

distribution over the training process. These results show how accurately each generative model captures the underlying mixture distribution. Specifically, visualization results show that VAEs and GANs struggle to capture the multimodal structure, while FM exhibits a stronger ability to model the target distribution.

In the next section, we will introduce how to explicitly leverage the mixture structure and corresponding weights to enhance the FM approach for more efficient learning of complex distributions.

A.3 Explicit Mixture Objective for Flow Matching

To enhance learning of mixture distributions $p^{(\text{mix})}(x) = \gamma p^{(1)}(x) + (1-\gamma)p^{(2)}(x)$, we introduce an **explicit mixture objective** within the flow matching framework. Unlike black-box density estimation, our method directly encodes the known compositional structure of the target distribution into the training objective. This improves computational efficiency and accelerates optimization convergence.

Let $u_\theta(x, k)$ denote the parameterized time-dependent vector field, the marginal distribution on time $k = 0$ is a standard Gaussian distribution, and $\phi_k(\epsilon, x)$ is the conditional time-dependent diffeomorphic map. Let $x \sim p^{(\text{mix})}, k \sim \mathcal{U}(0,1), \epsilon \sim \mathcal{N}(0, I_n)$ and $v_k(\epsilon, x) = \frac{d\phi_k(\epsilon,x)}{dk}$, the original flow matching objective for mixture distributions is,

$$\mathcal{L} = \mathbb{E}_{k,x,\epsilon}\|u_\theta(\phi_k(\epsilon,x),k) - v_k(\epsilon,x)\|^2 = \mathbb{E}_\epsilon\left[\int_0^1 p^{(\text{mix})}(x)\|u_\theta(\phi_k(\epsilon,x),k) - v_k(\epsilon,x)\|^2 dxdk\right].$$

Since p^{mix} is a mixture, we can decompose this into:

$$\begin{aligned}
\mathcal{L} &= \mathbb{E}_\epsilon\left[\int_0^1 \left(\gamma p^{(1)}(x) + (1-\gamma)p^{(2)}(x)\right)\|u_\theta(\phi_k(\epsilon,x),k) - v_k(\epsilon,x)\|^2 dxdk\right] \\
&= \gamma \underbrace{\mathbb{E}_{\substack{\epsilon,k \\ x^{(1)}\sim p^{(1)}}}\left\|u_\theta(\phi_k(\epsilon,x^{(1)}),k) - v_k(\epsilon,x^{(1)})\right\|^2}_{\mathcal{L}_1} + (1-\gamma)\underbrace{\mathbb{E}_{\substack{\epsilon,k \\ x^{(2)}\sim p^{(2)}}}\left\|u_\theta(\phi_k(\epsilon,x^{(2)}),k) - v_k(\epsilon,x^{(2)})\right\|^2}_{\mathcal{L}_2} \\
&= \gamma\mathcal{L}_1 + (1-\gamma)\mathcal{L}_2.
\end{aligned}$$

This explicit decomposition enables training by sampling from each mixture component and reweighting the loss according to its corresponding mixture weight. By leveraging the known structure of the mixture, this approach aligns closely with the recursive formulation of the NSM, which also exhibits a mixture form rooted in the Bellman equation, which is also noted in (Farebrother et al., 2025). As shown in Figure 3 and Figure 4, our proposed method, *FlowMatching (Explicit Mixture)*, outperforms standard baselines by effectively capturing the multimodal structure of the target distribution. It is worth noting that not only can flow matching methods exploit this form of distribution mixing, but also diffusion models (Song et al., 2020) and bridge-based models (Wang et al., 2021a), which all rely on the diffusion mixture representation (Peluchetti, 2023).

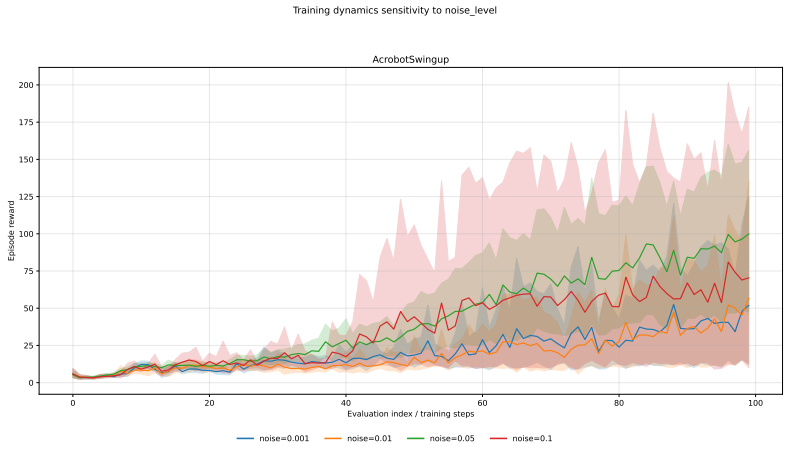


Figure 5: **SPR augmentation sweep.** Performance of SPR on AcrobotSwingup for different Gaussian noise magnitudes used in the augmentation pipeline. The noise level of 0.05 is selected for the cross-environment experiments.

B Experimental Setup and Comparisons

B.1 Comparison with Other Representation Learning Methods

To align with our method, we configure SPR (Schwarzer et al., 2020) with $K = 1$, so it relies only on the immediate successor transition, just like our approach. Since the SPR method was originally designed for image input, while our environment uses state input, we use Gaussian noise to simulate data augmentation. In order to select a suitable augmentation setting, we perform a hyperparameter sweep over the Gaussian noise magnitude on AcrobotSwingup environment. Following the sensitivity analysis in Figure 5, we adopt the best-performing noise standard deviation of 0.05 and reuse it across all 7 DeepMind Control Suite environments without further tuning. This configuration is used to generate the aggregate IQM statistics in Figure 1 and the full learning curves in Figure 6. Ensuring that improvements stem from representational benefits rather than environment-specific hyperparameter adjustments. In this configuration, our proposed method exhibits better performance.

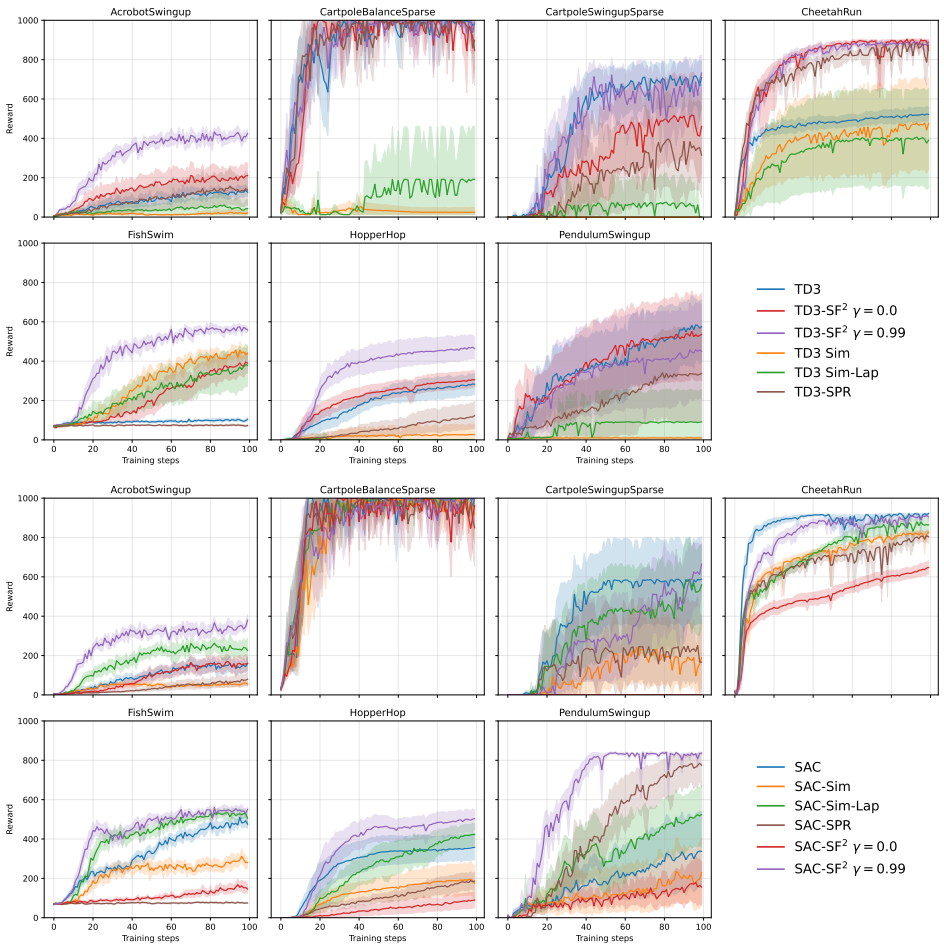


Figure 6: **Learning curves across tasks.** Each plot reports training curve with 95th percentile confidence intervals(bootstrap using 5000 samples) for TD3 (upper) and SAC (lower) with vanilla baselines, SF-based baselines, SPR(Schwarzer et al., 2020), and our $\texttt{SF}^2$ with transition ($\gamma = 0.0$) and successor ($\gamma = 0.99$) horizons.

C Experimental Results

C.1 Main Results on Default Settings

Figure 6 presents the full learning curves for 7 DeepMind Control Suite tasks using TD3 and SAC variants, complementing the aggregate IQM statistics shown in the main text.

C.2 Robustness Analysis

C.2.1 Performance Under Reduced Training Budget

To rule out the possibility that our gains come only from a specific chosen training budget, we reran every experiment with a 1M-timestep budget and 15 random seeds. Figure 7 reports the IQM aggregates for TD3 and SAC under this stricter setting, showing that $\texttt{SF}^2$ still surpasses vanilla and SF-based baselines by a clear margin. Figure 8 then provides the per-environment learning dynamics, confirming that our method maintains its advantage even when learning is constrained to only 1M timesteps. Finally, we study how our approach benefits from more training budgets and more training times because we require additional training on both the generative and representation models. To further illustrate this, we plot the IQM of the AUC as we vary the number of gradient steps. Figure 9 shows that allocating more gradient steps to the feature updater steadily improves the IQM score,

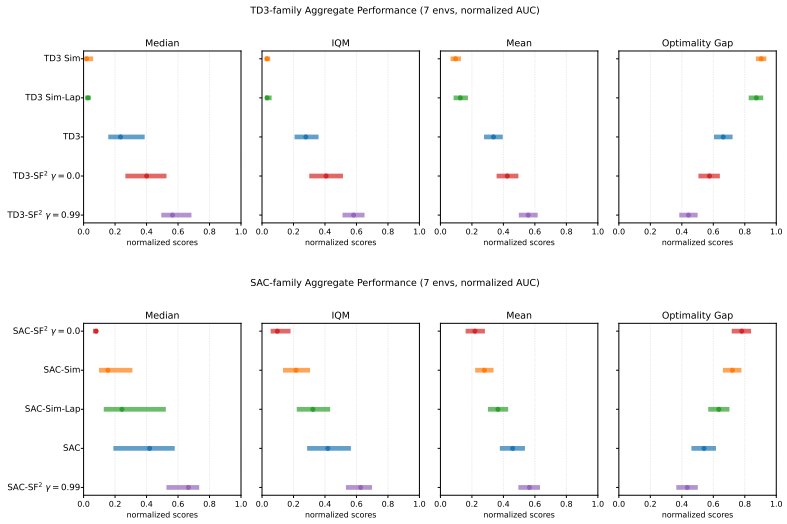


Figure 7: **IQM performance at 1M timesteps.** Aggregate AUC statistics for TD3 (upper) and SAC (lower) at a 1M-step training budget using 15 seeds.

highlighting that SF^2 benefits from thorough feature optimization rather than relying on a particular environment budget.

C.2.2 Generalization Under Observation Noise

To assess whether SF^2 has better generalization performance, we adopt the following setting. For each algorithm, we train on the 1M-timestep on the AcrobotSwingup environment using six random seeds (0–5), store the resulting checkpoints, and then evaluate the learned policies on perturbed versions of the environments in which Gaussian noise is injected into the observation space. The right panel of Figure 9 reports the AUC under different noise levels with 5 checkpoints with 1000 trajectories, evaluation with 95th percentile confidence interval using bootstrap with 5000 samples. Because SF^2 achieves higher performance even before the perturbation, it also sustains better AUC after the noise injection.

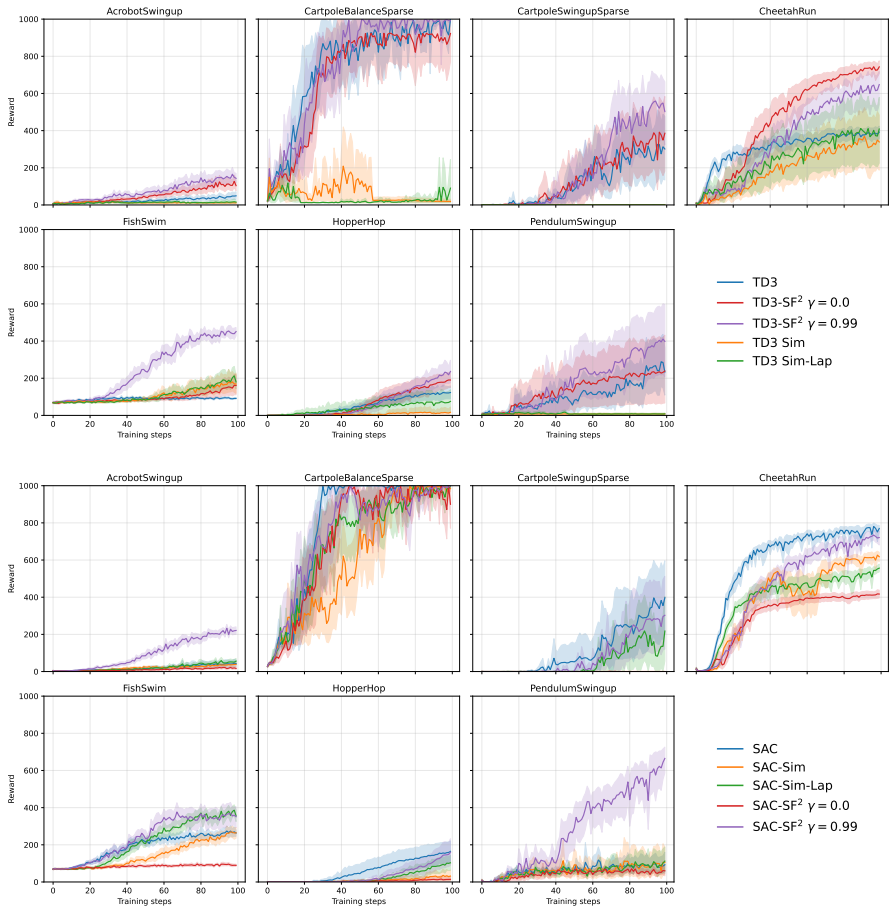


Figure 8: **Learning curves at 1M timesteps.** Each plot shows the median and 95% CIs (bootstrap, 5000 samples) over 15 seeds for TD3 (upper) and SAC (lower) with vanilla baselines, SF-based baselines, SPR (Schwarzer et al., 2020), and our $\texttt{SF}^2$ with transition ($\gamma = 0.0$) and successor ($\gamma = 0.99$) horizons.

C.3 Extension to Discrete Action Spaces with Image Input

We conducted additional experiments on the MinAtar benchmark from PGX (Koyamada et al., 2023) using PPO. Because our method constructs representations over state-action pairs (s, a), we replaced the standard PPO state-value estimator V with an action-value estimator $Q(s, a)$ and trained it with TD-λ target ($\lambda = 0.95$). We also adopted separate actor and critic networks for baseline and our methods. MinAtar provides image-like observations and a discrete action space; to make our approach applicable in this setting, we evaluated two implementations: (i) linear interpolation along image channels (shown as SF^2 w/o AE), and (ii) an autoencoder-based pipeline in which the generative model is trained in the latent space (shown as SF^2 w AE). All experiments were run with discount factor $\gamma = 0.9$. Results are presented in Figure 10, where we report both IQM (interquartile mean) statistics and training curves. These results provide preliminary evidence that our method can operate effectively in image-based environments with discrete actions.

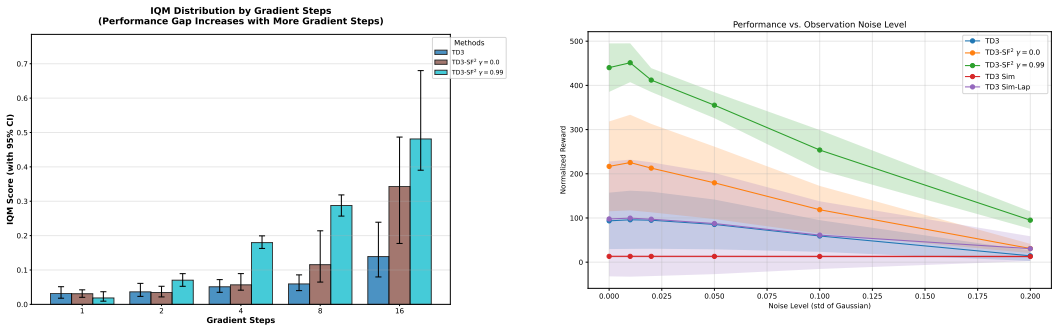


Figure 9: **Left:** Effect of training budget. IQM of the AUC as we vary the number of gradient steps dedicated to representation learning, highlighting the gains from additional feature optimization. **Right:** Generalization under Gaussian observation noise. Interquartile mean performance of policies trained for 1M steps (six seeds) when evaluated on Gaussian-noise-perturbed observations. $\texttt{SF}^2$ maintains the highest returns across both TD3 and SAC backbones.

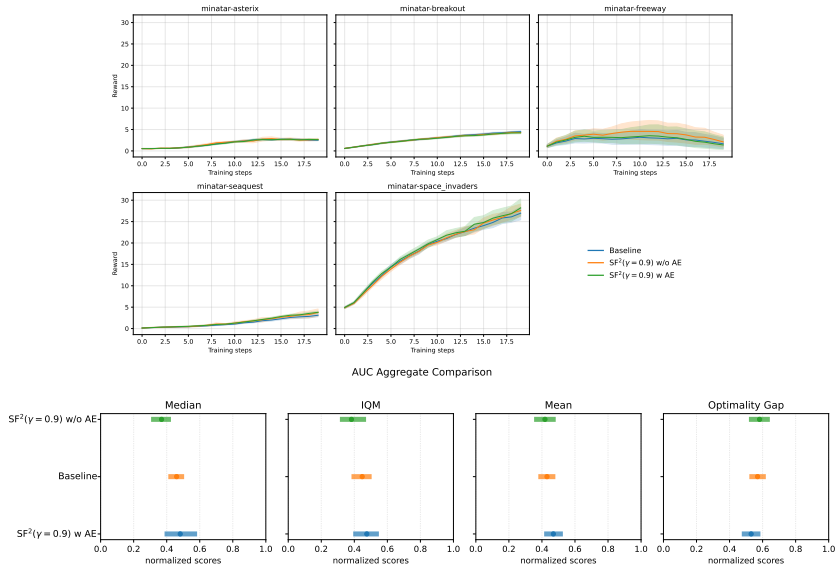


Figure 10: **MinAtar results. Upper** Learning curves for PPO variants on MinAtar environments with 95th percentile confidence intervals (bootstrap using 5000 samples). **Lower** Aggregate IQM performance. Interquartile mean AUC statistics across MinAtar environments comparing vanilla PPO, SF^2 without autoencoder (w/o AE), and SF^2 with autoencoder (w AE).

D Detailed Algorithms

We demonstrate the detailed algorithm training process, combining SAC and TD3.

Algorithm 2 TD3 with $\texttt{SF}^2$

Initialize $\theta, \phi, \psi, \zeta$; target networks $\theta' \leftarrow \theta, \phi' \leftarrow \phi, \psi' \leftarrow \psi, \zeta' \leftarrow \zeta$; replay buffer $\mathcal{B}$
for episode = 1 to M **do**
  Initialize environment, get initial state s_0
  for step = 1 to T **do**
    $a_t = \pi_\theta(s_t) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$; execute a_t, observe r_t, s_{t+1}; store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$
    if $\mathcal{B}$ is large enough **then**
      for G gradient steps **do**
        Sample batch (s, a, r, s') from $\mathcal{B}$
        $y = r + \gamma \min_{i=1,2} Q_{\phi_i'}(\psi'(s', \pi_{\theta'}(s')))$
        $\phi_i \leftarrow \phi_i - \alpha_Q \nabla_{\phi_i}(Q_{\phi_i}(\psi(s,a)) - y)^2$
        $\psi \leftarrow \psi - \alpha_\psi \nabla_\psi (Q_{\phi_i}(\psi(s,a)) - y)^2$
        $(\psi, \zeta) \leftarrow (\psi, \zeta) - \alpha_{SR} \nabla_{\psi,\zeta} \mathcal{L}_{SR}$
        if step mod d = 0 **then**
          $\theta \leftarrow \theta - \alpha_\pi \nabla_\theta Q_{\phi_1}(\psi(s, \pi_\theta(s)))$
          $\theta' \leftarrow \tau\theta + (1-\tau)\theta', \phi' \leftarrow \tau\phi + (1-\tau)\phi', \psi' \leftarrow \tau\psi + (1-\tau)\psi', \zeta' \leftarrow \tau_\zeta \zeta + (1-\tau_\zeta)\zeta'$
        end if
      end for
    end if
  end for
end for

Algorithm 3 SAC with $\texttt{SF}^2$

Initialize θ, ϕ, ψ; target networks $\theta' \leftarrow \theta, \phi' \leftarrow \phi, \psi' \leftarrow \psi$; replay buffer $\mathcal{B}$
for episode = 1 to M **do**
  Initialize environment, get initial state s_0
  for step = 1 to T **do**
    $a_t = \pi_\theta(s_t) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$; execute a_t, observe r_t, s_{t+1}; store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$
    if $\mathcal{B}$ is large enough **then**
      for G gradient steps **do**
        Sample batch (s, a, r, s') from $\mathcal{B}$
        $y = r + \gamma \min_{i=1,2} Q_{\phi_i'}(\psi'(s', \pi_{\theta'}(s')))$
        $\phi_i \leftarrow \phi_i - \alpha_Q \nabla_{\phi_i}(Q_{\phi_i}(\psi(s,a)) - y)^2$
        $\psi \leftarrow \psi - \alpha_\psi \nabla_\psi (Q_{\phi_i}(\psi(s,a)) - y)^2$
        $(\psi, \zeta) \leftarrow (\psi, \zeta) - \alpha_{SR} \nabla_{\psi,\zeta} \mathcal{L}_{SR}$
        if step mod d = 0 **then**
          $\theta \leftarrow \theta - \alpha_\pi \nabla_\theta Q_{\phi_1}(\psi(s, \pi_\theta(s)))$
          $\theta' \leftarrow \tau\theta + (1-\tau)\theta', \phi' \leftarrow \tau\phi + (1-\tau)\phi', \psi' \leftarrow \tau\psi + (1-\tau)\psi', \zeta' \leftarrow \tau_\zeta \zeta + (1-\tau_\zeta)\zeta'$
        end if
      end for
    end if
  end for
end for

E NETWORK ARCHITECTURE AND HYPERPARAMETERS

E.1 DEEPMIND CONTROL SUITE IN SECTION 4

Parameter	Value
Network Architecture	
Hidden Layer Sizes for Q and policy	(512, 512, 512)
Q-Network Layer Normalization	True
Policy Network Layer Normalization	True
Feature Size	512
Zeta Network Hidden Layer Sizes	(512, 512)
Embedding Size	64
Activation Function	ReLU
Kernel Initializer	LeCun Uniform
Training Parameters	
Number of Timesteps	5M: CartpoleBalanceSparse, CartpoleSwingupSparse, FishSwim; 10M: Acrobot, Hopper, CheetahRun, PendulumSwingup
Number of Evaluations	100
Reward Scaling	1.0
Max Episode Length	1000
Normalize Observations	True
Action Repeat	1 (4 for PendulumSwingUp)
Learning Rate	1e-3
Number of Environments	128
Batch Size	512
Gradient Updates per Step	8
Max Replay Size	4,194,304 (1048576 * 4)
Min Replay Size	8192
Discounting Factor	0.99
Policy Delay	1
Noise Clip	0.3
Smoothing Noise	0.2
Exploration Noise	0.2
Optimizer Parameters	
Alpha Optimizer Learning Rate	3e-4
Policy Optimizer Learning Rate	1e-4
Q-Network Optimizer Learning Rate	1e-4
Psi-Zeta Optimizer Learning Rate	1e-4
Method-Specific Parameters	
TD3/SAC Gamma for Successor	0.99
TD3/SAC Tau Zeta	0.005
TD3/SAC Denoising Steps	2 Function Evaluations

Table 1: Network and Training Parameters

E.2 Two Gaussian Example in Section A.2

Parameter	Value
Distribution Parameters	
First Gaussian Mean (μ_1)	-2.0
Second Gaussian Mean (μ_2)	2.0
First Gaussian Std (σ_1)	0.3
Second Gaussian Std (σ_2)	0.7
Mixture Probability	0.01
Network Architecture	
Hidden Dimension	256
Activation Function	ReLU
Training Parameters	
Number of Steps	1500
Batch Size	32
Learning Rate	1e-3
Evaluation Interval	200
Logging Interval	100
EMA Decay Rate	0.995
Evaluation Parameters	
Sample Size for Evaluation	4000
Number of Integration Steps	100

Table 2: Two Gaussian Example Hyperparameters

F The use of LLMs

The authors used LLMs to polish the language and improve readability. All AI-generated content was thoroughly reviewed and revised by the authors, who take full responsibility for the final content.

G Schematic Visualization of SF^2

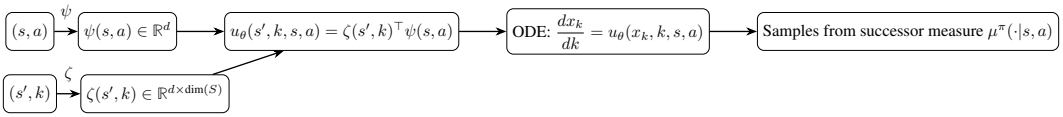


Figure 11: Schematic visualization of how ψ, ζ, and u interact. $\psi(s,a)$ encodes the time-invariant flow representation of the current state–action pair. $\zeta_k(s')$ provides a time-varying projection over future states s'. Their inner product defines the vector field $u(s',k,s,a)$, which describes how future-state probability mass evolves in the flow-matching objective.

H Experiments with CRL and Flow-Based Policy

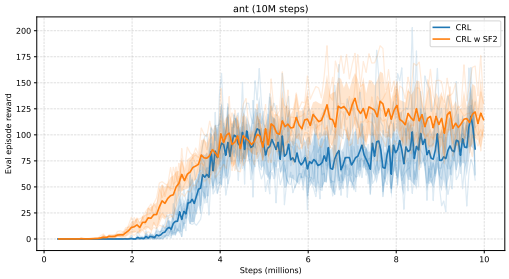


Figure 12: Return curve on Ant environment with CRL (Eysenbach et al., 2022) and CRL with SF^2 based on JaxGCRL. (Bortkiewicz et al., 2025)(5 seeds)

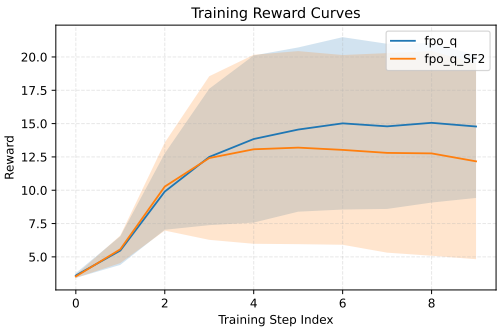


Figure 13: Return curve on CheetahRun environment with FPO (McAllister et al., 2025) with $Q(\lambda)$ (Peng & Williams, 1994) and FPO with $Q(\lambda)$ and SF^2. (5 seeds)