Flow Matching Policy Optimization with Mirror Descent and Entropy Constraints

Ting Gao, Stavros Orfanoudakis, Nan Lin, Winnie Daamen, Serge Hoogendoorn, Elvin Isufi
Delft University of Technology
{T.Gao-1, S.Orfanoudakis, N.Lin, W.Daamen, S.P.Hoogendoorn, E.Isufi-1}@tudelft.nl

Abstract

Balancing policy expressiveness with the exploration-exploitation trade-off is a core challenge in online Reinforcement Learning (RL). While Stochastic Differential Equation (SDE)-based diffusion policies can represent complex, multimodal action distributions, they suffer from two critical limitations: their stochastic reverse processes render entropy intractable (necessitating heuristic exploration), and computing policy gradients through long denoising chains is expensive and unstable. In this work, we show that ODE-based flow matching inherently resolves these issues by enabling both simulation-free policy optimization and tractable entropy computation. Building on this, we introduce Flow Matching Policy Optimization with Mirror Descent and Entropy Constraints (FMER). Our framework exploits this insight in three ways. First, we derive an analytic entropy objective that corrects for the density distortion caused by the $\tanh$ transformation (mapping an unbounded latent space to bounded actions), thereby facilitating entropy-constrained policy optimization. Second, we theoretically establish that minimizing an advantage-weighted conditional flow matching loss acts as a simulation-free surrogate for policy mirror descent. This steers the velocity field toward high-value regions while entirely avoiding backpropagation through the ODE solver. Finally, we dynamically tune the mirror descent temperature based on the effective sample size to enforce a robust trust region during training. Empirical evaluations demonstrate that FMER achieves superior performance on the challenging sparse-reward FrankaKitchen environment, while maintaining competitive results across standard dense-reward MuJoCo benchmarks. Code is available at `https://github.com/gtanaelle/FMER.git`.

1 Introduction

In continuous control RL, standard unimodal Gaussian policies often fail to capture complex action landscapes. Many tasks exhibit multiple distinct, state-dependent high-return behaviors, necessitating policies capable of representing multi-peaked distributions [10]. This has motivated the development of generative policy classes that can represent rich, multimodal action distributions by transforming noise samples into actions conditioned on the state. Diffusion policies transform this via iterative denoising [36, 38], often relying on simulating a reverse-time SDE. Meanwhile, Flow Matching (FM) formulations [19, 22] provide an ODE-based alternative that learns the underlying transport by directly regressing a vector field, closely related to Continuous Normalizing Flows [4].

Diffusion policies have first gained prominence in offline RL [1, 37], typically formulated as behavior cloning of expert data. However, extending these models to online RL is non-trivial due to the absence of pre-collected expert data. Existing online approaches fall into two categories: (1) using the diffusion model purely as an approximation tool and backpropagating policy gradients through the sampling chain [36], a process that is computationally expensive and susceptible to high-variance gradients

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Table 1: Conceptual comparison between FMER and diffusion- and flow-based baselines.

Method	Policy Mirror Descent	Entropy Constrained	Unbiased Entropy Objective	Simulation-Free Training
DIPO [38]	✗	✗	✗	✓
DPMD [21]	✓	✗	✗	✓
QVPO [6]	✗	✓	✗	✓
SAC-Flow [42]	✗	✓	✗	✗
FLAME [17]	✗	✓	✗	✗
FMER (Ours)	✓	✓	✓	✓

[6, 29]; (2) leveraging the generative capability to iteratively improve the policy by regressing towards high-value samples selected by a critic [6, 21, 38]. Our work follows the second paradigm.

While SDE frameworks excel at approximating complex distributions, they face a fundamental limitation in online RL: their stochastic reverse process renders policy entropy intractable, yet without tractable entropy, maximum entropy exploration requires crude heuristics (e.g., uniform distribution assumptions [6]). Removing stochasticity by switching to ODE-based methods recovers tractable entropy but backpropagating through the ODE integration is numerically unstable [6, 42].

Recently, several ODE-based FM policies have emerged. For instance, [42] address gradient instability using sequence modeling techniques, while [17] achieve one-step FM generation within a maximum entropy framework. However, these approaches overlook the policy mirror descent perspective, which imposes crucial behavioral constraints to prevent policy collapse in extremely sparse environments or under perturbed critic estimations. Additionally, in their pursuit of reducing integration steps, these prior methods compromise on closed-form entropy by relying on entropy surrogates or augmented ODEs.

As shown in Table 1, existing methods trade off at least one of three desirable properties: tractable entropy, simulation-free training, and a principled mirror descent trust region. FMER is designed to satisfy all three simultaneously. We propose Flow Matching Policy Optimization with Mirror Descent and Entropy Constraints (FMER), which resolves these challenges by adopting the Optimal Transport (OT) conditional FM objective [19] to preserve simulation-free training, deriving a closed-form entropy expression that corrects for the $\tanh$ density distortion in bounded action spaces, and incorporating a mirror descent trust region via advantage-weighted updates. Each design choice directly addresses one of the three challenges above. While FMER shares the ideal policy mirror descent target with DPMD [21], it realizes this target through simulation-free weighted conditional FM rather than reweighted denoising score matching.

Our contributions are threefold:

- **FM Policy Optimization with Mirror Descent and Entropy Constraints.** We propose FMER, which leverages an ODE-based Flow Matching policy to improve expressiveness. FMER incorporates an explicit entropy constraint that directly encourages high-entropy policies, thereby improving exploration. In addition, FMER adopts the standard policy mirror descent target (shared with DPMD [21]) and contributes a simulation-free realization of it with self-normalized, ESS-adaptive weights.
- **Theoretical Foundations.** We derive a tractable entropy objective with an unbiased divergence estimator, which accounts for the density distortion induced by $\tanh$ mapping from the unbounded FM latent space to a bounded action space. This enables closed-form entropy-constrained optimization without surrogates or augmented ODEs. Under regularity and realizability assumptions, we further prove that any global minimizer of the weighted conditional FM loss induces the ideal policy mirror descent target, enabling efficient simulation-free training.
- **Empirical Validation.** We evaluate FMER against eight baselines spanning standard RL, diffusion-based RL, and FM-based RL methods. Experiments are conducted on 2D multi-goal environments, multi-task long-horizon and sparse-reward FrankaKitchen tasks, and dense-reward MuJoCo benchmarks. Together, these experiments and ablation studies validate that policy multimodality, principled entropy control, and mirror descent trust regions each play a necessary role in FMER's performance.

2 Related work

Diffusion policies. Diffusion policies parameterize the policy network as a diffusion model and first gained prominence in offline RL [3, 12, 14, 37]. Online extensions include DIPO [38], which imitates Q-gradient-updated replay actions; QSM [28], which aligns score estimates with critic gradients; DACER [36] and DIME [2], which backpropagate policy gradients through the diffusion sampling chain; and critic-weighted regression methods such as QVPO [6] and DPMD [21]. These approaches face expensive optimization through long sampling chains, and heuristic entropy approximations that bias exploration.

Flow Matching (FM) Policies. FM replaces stochastic reverse-time dynamics with a deterministic ODE over a learned vector field [4, 19], offering simulation-free training and easier-to-control sampling. Prior offline applications [30, 40] and online methods including Flow-GRPO [20], FPO [22], FPMD [5], GoRL [39], and FM-IRL [35] have explored this direction. RWFM and GRPO [27] improve demonstration-initialized action-chunk planners, whereas FMER learns from scratch through online interaction in sequential decision problems. RFM [18] bridges diffusion policies and flow policies under the MaxEnt principle. Most related to ours are SAC-Flow [42] and FLAME [17], which forgo conditional FM's simulation-free advantage by backpropagating through ODE integration, reducing steps for efficiency at the cost of closed-form entropy: SAC-Flow mitigates gradient instability via recurrent architectures, while FLAME achieves single-step generation but relies on entropy surrogates or augmented ODE dynamics. In contrast, FMER preserves simulation-free training and derives a closed-form entropy expression. See Appendix B.3–B.5 for detailed comparisons with prior works.

3 Preliminaries

3.1 Reinforcement learning

The environment is modeled as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma)$ where $\mathcal{S}$ and $\mathcal{A}$ are the continuous state and action spaces. The transition density $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$ describes the likelihood of transitioning to state s_{i+1} given action a_i in state s_i. The bounded reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ gives a scalar reward after an action is taken. ρ_0 denotes the initial state density, and $\gamma \in [0, 1)$ is the discount factor. The RL policy is represented as $\pi(a|s) : \mathcal{S} \times \mathcal{A} \to [0, \infty)$, denoting the probability density of taking action a_i at state s_i. Starting from an initial state s_0, the agent interacts with the environment to get a trajectory by following the policy π. We denote the trajectory distribution induced by the policy as $\mathcal{T} \sim \rho_\pi$.

The value function $V_\pi(s) = \mathbb{E}_{\mathcal{T}\sim\rho_\pi}[\sum_{i=0}^{\infty} \gamma^i r_i \mid s_0 = s]$ estimates the expected discounted future reward starting from any given state $s_0 = s$. Similarly, the action-value function $Q_\pi(s, a) = \mathbb{E}_{\mathcal{T}\sim\rho_\pi}[\sum_{i=0}^{\infty} \gamma^i r_i \mid s_0 = s, a_0 = a]$ estimates the expected return after taking action a at state s. The advantage function $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$ measures the relative benefit of taking action a over the average action at state s. The goal of RL is to maximize the expected cumulative reward $J(\pi) = \mathbb{E}_{\mathcal{T}\sim\rho_\pi}[\sum_{i=0}^{\infty} \gamma^i r_i]$.

3.2 Flow Matching

Flow Matching [19] is a generative framework that models a probability path p_t transporting a simple noise distribution p_0 (at $t = 0$) to a complex data distribution p_{data} (at $t = 1$). This transport is governed by a time-dependent ODE:

$$\frac{d}{dt} x_t = v_\theta(x_t, t), \tag{1}$$

where v_θ is a neural vector field parameterized by θ. To train this field, it is standard to define a target conditional probability path [19]. In this work, we utilize the Optimal Transport (OT) path, which linearly interpolates between noise $x_0 \sim \mathcal{N}(0, \mathbf{I})$ and data $x_1 \sim p_{\text{data}}$: $x_t = (1 - t)x_0 + tx_1$.

This path corresponds to a conditional vector field $u_t^{\text{target}}(x_t|x_0, x_1)$ that generates the trajectory from x_0 to x_1. For the Optimal Transport (OT) path, this vector field is the constant velocity:

$$u_t^{\text{target}}(x_t|x_0, x_1) = \frac{d}{dt} x_t = x_1 - x_0. \tag{2}$$

Since the true marginal vector field is intractable, [19] shows that minimizing the Conditional Flow Matching (CFM) loss is equivalent to minimizing the intractable marginal FM loss (up to a constant):

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,x_1\sim p_{\text{data}},x_0\sim p_0}\left[\left\|v_\theta(x_t,t) - u_t^{\text{target}}(x_t|x_0,x_1)\right\|^2\right]. \tag{3}$$

4 Method

In this section, we introduce the Flow Matching Policy Optimization with Mirror Descent and Entropy Constraints (FMER) framework.

To stabilize learning, Policy Mirror Descent [16, 33] formulates the policy update as a constrained optimization problem that maximizes the expected advantage while penalizing the Kullback-Leibler (KL) divergence from the current policy π_k:

$$\pi_{k+1} = \arg\max_\pi \mathbb{E}_{s\sim\mathcal{D}}\Big[\mathbb{E}_{a\sim\pi(\cdot|s)}[A^{\pi_k}(s,a)] - \tau D_{\text{KL}}(\pi(\cdot|s)\,\|\,\pi_k(\cdot|s))\Big] \tag{4}$$

where τ is the temperature parameter controlling the strength of the KL penalty, and $\mathcal{D}$ is the replay buffer. Following the same reasoning as AWR [26], denoting $Z(s) = \mathbb{E}_{a\sim\pi_k(\cdot|s)}\left[\exp\left(\frac{A^{\pi_k}(s,a)}{\tau}\right)\right]$, Eq. (4) yields a closed-form objective (proof in Appendix A.4):

$$\max_\theta \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_k(\cdot|s)}\left[\frac{1}{Z(s)}\exp\left(\frac{A^{\pi_k}(s,a)}{\tau}\right)\log\pi_\theta(a|s)\right]. \tag{5}$$

We write $A = A^{\pi_k}$ for brevity. To encourage exploration and prevent premature convergence, we are inspired by the entropy-constrained formulation of SAC [11]. We optimize the policy parameters θ to maximize the advantage-weighted log-likelihood subject to a minimum entropy constraint $\bar{H}$:

$$\min_\theta \mathcal{J}(\theta) = -\mathbb{E}_{s\sim\mathcal{D},a\sim\pi_k(\cdot|s)}\left[\frac{1}{Z(s)}\exp\left(\frac{A(s,a)}{\tau}\right)\log\pi_\theta(a|s)\right] \quad \text{s.t.} \quad \mathbb{E}_{s\sim\mathcal{D}}[H(\pi_\theta(\cdot|s))] \geq \bar{H}, \tag{6}$$

where $H(\pi_\theta(\cdot|s))$ is the policy entropy. We handle this constraint using a Lagrangian formulation:

$$\mathcal{L}(\theta,\alpha) = -\mathbb{E}_{s\sim\mathcal{D},a\sim\pi_k(\cdot|s)}\left[\frac{1}{Z(s)}\exp\left(\frac{A(s,a)}{\tau}\right)\log\pi_\theta(a|s)\right] + \alpha\big(\bar{H} - \mathbb{E}_{s\sim\mathcal{D}}[H(\pi_\theta(\cdot|s))]\big), \tag{7}$$

where $\alpha \geq 0$ is the Lagrange multiplier, dynamically adjusted via dual gradient descent [11] to maintain the target entropy $\bar{H}$. The critic estimates the standard discounted reward return without future entropy terms; entropy enters only through the actor constraint. FMER is therefore an entropy-constrained policy optimization method rather than a full maximum-entropy RL algorithm. Appendix B.2 details our rationale for not jointly optimizing policy mirror descent and the entropy constraint.

Computing $\log\pi_\theta(a|s)$ requires solving an ODE, and directly optimizing Eq. (7) by backpropagating gradients through the ODE integration induces training instability [6, 42]. To address these challenges, we substitute the log-likelihood term with the Weighted Conditional Flow Matching ($\mathcal{L}_{\text{W-CFM}}$) loss, which serves as a *simulation-free and tractable* proxy. Together with the entropy regularization term $\mathcal{L}_{\text{ent}}$ (introduced in Sec. 4.3), we define the unified FMER loss:

$$\mathcal{L}_{\text{FMER}}(\theta) = \mathcal{L}_{\text{W-CFM}}(\theta) + \alpha\mathcal{L}_{\text{ent}}(\theta). \tag{8}$$

The remainder of this section is organized as follows: Sec. 4.1 presents the policy representation, Sec. 4.2 details the $\mathcal{L}_{\text{W-CFM}}(\theta)$ loss, Sec. 4.3 describes the entropy loss $\mathcal{L}_{\text{ent}}(\theta)$, and Sec. 4.4 presents the complete training procedure. The policy training pipeline is illustrated in Figure 1.

4.1 Flow Matching Policy Representation

We parameterize $\pi_\theta(a|s)$ using a latent FM framework. To ensure valid actions in bounded continuous control environments (e.g., $a \in (-1,1)^d$), we define the flow in an unbounded latent space $x_t \in \mathbb{R}^d$ and obtain actions via $a = \tanh(x_1)$, where $x_1 \sim p_1(\cdot|s)$ and $p_1(\cdot|s)$ represents the *terminal* distribution generated at $t = 1$. This guarantees valid actions by design. While squashing actions

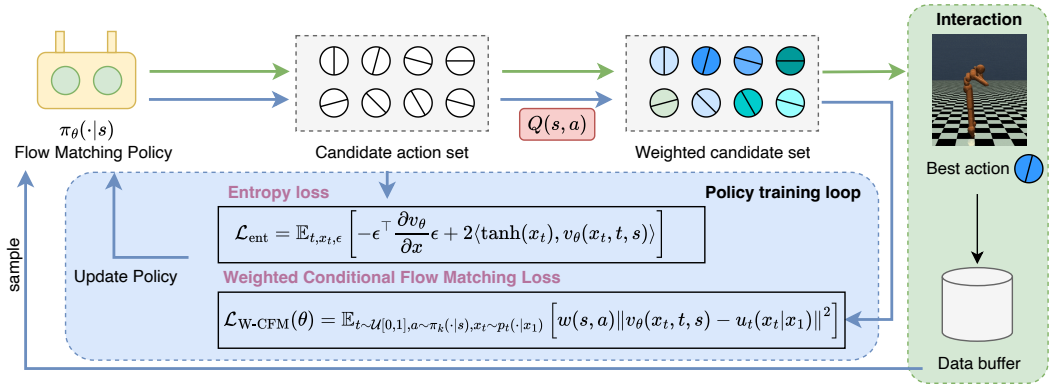


Figure 1: Overview of the FMER framework. Left (Policy Training): The policy generates a set of candidate actions, which are evaluated by the critic (Q-network) to compute advantage-based weights. These weights guide the Weighted Conditional FM loss, while an entropy regularization term ensures exploration. Right (Interaction): During data collection, the agent samples multiple candidates and executes the highest-value action to store in the replay buffer.

with a $\tanh$ operator is common in Gaussian policies [7, 11], computing the entropy of a flow policy requires special care to account for the density distortion induced by this transformation (Section 4.3).

The policy is trained by regressing v_θ toward the Optimal Transport target velocity $x_1 - x_0$ (Eq. 2-3), where $x_1 = \text{arctanh}(a)$ and $x_t = (1-t)x_0 + tx_1$:

$$\mathcal{L}_{\pi\text{-CFM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],x_0\sim p_0,(s,a)\sim\mathcal{D}}\left[\|v_\theta(x_t,t,s) - (\text{arctanh}(a) - x_0)\|^2\right]. \tag{9}$$

At inference, actions are generated by integrating the ODE via Euler steps [25]. The trajectory is discretized into N steps of size $\Delta t = 1/N$ and $x_0 \sim \mathcal{N}(0, \mathbf{I})$:

$$x_{t+\Delta t} = x_t + \Delta t \cdot v_\theta(x_t,t,s), \quad t \in \left\{0, \tfrac{1}{N}, \ldots, \tfrac{N-1}{N}\right\}. \tag{10}$$

The resulting terminal latent state x_1 is then transformed into the executable action via $a = \tanh(x_1)$.

4.2 Policy Optimization via Weighted-CFM

Eq. 9 can be used for behavior cloning. In online RL, we leverage a learned value function Q_ψ (parameterized by ψ) to guide policy updates. As shown in Figure 1, at each iteration k, we sample actions from the current policy π_k, evaluate them via Q_ψ, and apply advantage-based weights to the Flow Matching loss, steering the vector field toward higher-value regions of the action space.

Theorem 1 (Weighted Flow Matching Surrogate)**.** *Fix a temperature $\tau > 0$, a reference policy π_k, and an advantage function A. For each state s, let $w(s,a) = \frac{\exp(A(s,a)/\tau)}{Z(s)}$ where $Z(s) := \mathbb{E}_{a\sim\pi_k(\cdot|s)}[\exp(A(s,a)/\tau)]$, $x_1 = \text{arctanh}(a)$, and $x_t = (1-t)x_0 + tx_1$. Consider the Weighted Conditional Flow Matching loss*

$$\mathcal{L}_{\text{W-CFM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],s\sim\mathcal{D},a\sim\pi_k(\cdot|s),x_0\sim p_0}\left[w(s,a)\|v_\theta(x_t,t,s) - (x_1 - x_0)\|^2\right] \tag{11}$$

Assume $0 < Z(s) < \infty$ for $\mathcal{D}$-almost every state, and that the velocity network is expressive enough to represent the target marginal velocity field. Under the integrability and regularity conditions, every global minimizer θ^* induces the ideal policy mirror descent target

$$\pi_{\theta^*}(a \mid s) = \pi^*(a \mid s) := \frac{\pi_k(a \mid s)\exp(A(s,a)/\tau)}{Z(s)} \tag{12}$$

for $\mathcal{D}$-almost every s and almost every a.

Remark. The weight $w(s,a)$ in Eq. (11) depends only on the terminal action $a = \tanh(x_1)$, see Appendix B.1 for why the endpoint weight is the appropriate choice.

The full proof is provided in Appendix A.1. Connecting advantage-weighted policy improvement to vector field regression is non-trivial. Whereas methods such as SAC-Flow [42] attempt to mitigate the gradient instability inherent to backpropagating through the ODE, Theorem 1 enables a simulation-free training paradigm that entirely circumvents this issue, highlighting its practical significance.

In practice, we approximate the state-dependent normalizing constant using M candidate actions $\{a^{(j)}\}_{j=1}^{M} \sim \pi_k(\cdot|s)$: $\hat{Z}(s) = \frac{1}{M}\sum_{l=1}^{M} \exp\left(A(s, a^{(l)})/\tau\right)$. Substituting $\hat{Z}(s)$ into the empirical weighted-CFM objective and absorbing the Monte Carlo factor $1/M$ into the weights gives:

$$\hat{w}(s, a^{(j)}) = \frac{\exp\left(A(s, a^{(j)})/\tau\right)}{\sum_{l=1}^{M} \exp\left(A(s, a^{(l)})/\tau\right)}. \tag{13}$$

Although omitting $Z(s)$ preserves the ideal statewise minimizer under the assumptions of Theorem 1, retaining its estimate normalizes the total candidate weight for each state. Unlike top-1 candidate selection for policy fitting, this soft weighting retains information from multiple high-quality actions. As illustrated in Figure 2, top-1 selection discards this information and can be brittle to critic estimation errors such as overestimation.

Since the state-only baseline $V^{\pi_k}(s)$ is constant across candidates, it cancels during normalization. The weights can therefore be computed directly from $Q_\psi(s, a^{(j)})$, without learning a separate value network. We discuss connections to advantage-weighted and softmax policy methods in Appendix B.3 and connections to policy mirror descent methods in Appendix B.5.

Adaptive tuning of τ. The temperature τ controls the KL penalty strength in the policy mirror descent objective, governing the strictness of the trust region around π_k. Large τ flattens the weights, pulling updates toward the behavior distribution; small τ sharpens them, allowing more aggressive deviation toward high-advantage actions. We tune τ adaptively by monitoring the Effective Sample Size (ESS) of the normalized weights $\{\hat{w}_l(s)\}_{l=1}^{M}$:

$$\mathrm{ESS}(s) = \frac{1}{\sum_{l=1}^{M} \hat{w}_l(s)^2}, \tag{14}$$

which ranges from 1 (one weight dominates) to M (uniform weights). We select a target $\mathrm{ESS}^* \in (1, M)$. Similar to the update of α, we adjust τ at each step to maintain the target ESS^*: decreasing τ when $\mathrm{ESS} > \mathrm{ESS}^*$ (overly uniform weights, constraint too strict) and increasing it when $\mathrm{ESS} < \mathrm{ESS}^*$ (few actions dominate, constraint too loose).

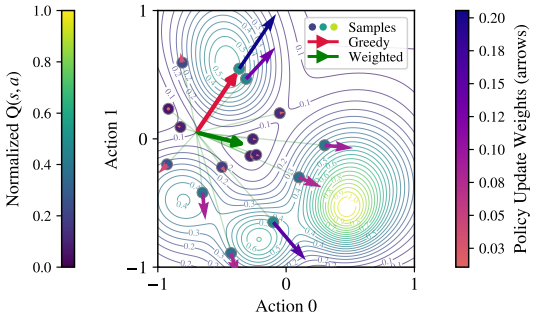


Figure 2: Weighted vs. greedy policy updates. Contours show the ground truth Q-value landscape for a fixed state. From 16 sampled candidate actions, the top-1 (greedy) update moves toward the highest-valued sample (red arrow), leading to a local maximum. In contrast, the weighted update (green) aggregates all candidates via Q-based weights (small arrows) towards the global optimal.

4.3 Entropy-Regularized Framework

The stochastic reverse process of diffusion-based policies renders their entropy intractable. In Section 4.1, we apply a nonlinear transformation $a = \tanh(x_1)$ to map from the unbounded FM latent space into the bounded action space. The quantity of interest is the entropy of *action* instead of the latent x_1. *While prior work either ignores or approximates the probability distortion introduced by this change of variables, our work closes this gap with a closed-form formulation.*

Theorem 2 (Entropy of Latent Flow, restated from [4, 19] for the OT path)**.** *Consider a latent vector field $v_\theta(x, t, s)$ that transports a base distribution p_0 to the target latent distribution p_1 at $t = 1$. For the unbounded latent space $\mathbb{R}^d$, the entropy $H^x(p_1(\cdot|s))$ can be computed as:*

$$H_1^x(p_1) = H_0^x + \int_0^1 \mathbb{E}_{x_t \sim p_t(\cdot|s)} \left[\nabla_x \cdot v_\theta(x_t, t, s)\right] dt, \tag{15}$$

where $H_0^x = \frac{d}{2}(1 + \ln(2\pi))$ is the entropy of the base Gaussian noise $p_0 = \mathcal{N}(0, \mathbf{I})$, d is the latent dimension, and $\nabla_x \cdot v_\theta = Tr(\nabla_x v_\theta)$ is the divergence of the vector field.

Algorithm 1 Flow Matching Policy Optimization with Mirror Descent and Entropy Constraints

Input: Parameters $\theta, \psi, \alpha, \tau$; Target entropy $\bar{H}$; Replay buffer $\mathcal{D}$.
for each epoch **do**
  // Interaction
  Observe s; generate M candidates $\{a^{(j)}\}_{j=1}^{M}$ via ODE (Eq. 10); select a with best Q value.
  Execute a, observe r, s', store (s, a, r, s') in $\mathcal{D}$.
  // Training
  for K gradient steps **do**
    Sample batch $B \sim \mathcal{D}$.
    Critic: Select best of M actions; compute targets y (Eq. 20); update ψ (Eq. 19).
    Policy: Sample M actions from $\pi_\theta(\cdot|s)$, compute weights w (Eq. (13)), sample flow time t; update θ minimizing $\mathcal{L}_{\text{W-CFM}} + \alpha\mathcal{L}_{\text{ent}}$ (Eqs. 11, 18).
    Update α; update τ; update target networks.
  end for
end for

The proof is provided in Appendix A.2. This theorem provides a tractable mechanism to control policy exploration directly through the vector field divergence. To ensure simulation-free optimization consistent with $\mathcal{L}_{\text{W-CFM}}$, we evaluate divergence along the Optimal Transport path: $x_t = (1-t)x_0 + tx_1$. By sampling $t \sim \mathcal{U}[0,1]$, the entropy is unbiasedly estimated by:

$$H_1^x \approx H_0^x + \mathbb{E}_{t\sim\mathcal{U}[0,1], x_t\sim p_t(\cdot|s)} \left[\nabla \cdot v_\theta(x_t, t, s)\right]. \tag{16}$$

Proposition 1 (Divergence under Tanh Transformation)**.** *Consider the coordinate transformation $a = \tanh(x)$ mapping any latent state $x \in \mathbb{R}^d$ to the bounded action space $(-1, 1)^d$. Let $v_\theta(x, t, s)$ be the vector field in the latent space. The corresponding induced vector field in the action space, $v^a(a, t, s)$, satisfies the following divergence relationship at any point x:*

$$\nabla_a \cdot v^a(a, t, s) = \nabla_x \cdot v_\theta(x, t, s) - \sum_{i=1}^{d} 2\tanh(x_i) v_{\theta,i}(x, t, s), \tag{17}$$

where $v_{\theta,i}$ is the i-th component of the latent vector field.

The proof of Proposition 1 can be found in Appendix A.3. Eq. (17) allows us to optimize the action-space entropy $\pi(a|s)$ by adjusting the latent flow. To maximize entropy (minimize negative entropy), we define the entropy regularization loss $\mathcal{L}_{\text{ent}}$ as:

$$\mathcal{L}_{\text{ent}} = -\mathbb{E}_{t,x_t} \left[\nabla_x \cdot v_\theta(x_t, t, s) - 2\langle \tanh(x_t), v_\theta(x_t, t, s)\rangle\right]. \tag{18}$$

Appendix B.4 details the relationship between our FMER and existing MaxEnt methods. In practice, computing the divergence $\nabla_x \cdot v_\theta(x) = \sum_{i=1}^{d} \frac{\partial v_{\theta,i}}{\partial x_i}$ in Eq. (18) scales as $\mathcal{O}(d^2)$. To maintain the computational efficiency necessary for high-dimensional benchmarks, we use an unbiased Hutchinson trace estimator [8, 13]. This reduces the per-step complexity to $\mathcal{O}(d)$, ensuring training remains computationally feasible (see Appendix B.6 for details). Hutchinson estimation is only used for actor training and not for action generation, training and inference costs are reported in Appendix C.7.

Remark. Since H_0^a depends only on the fixed base distribution p_0 via a constant Jacobian correction to H_0^x, it is constant w.r.t. θ and can be omitted from the optimization objective (see Appendix B.7 for the full derivation).

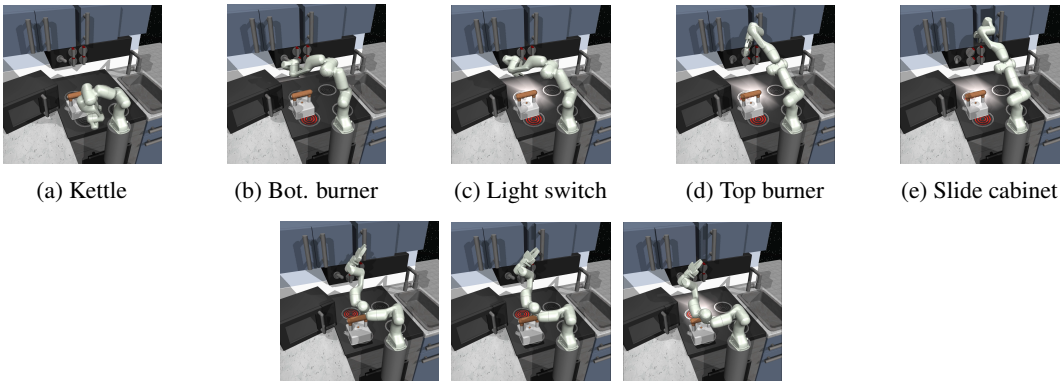
4.4 Training Procedure

The complete procedure is summarized in Algorithm 1. For the value network update, the critic parameters ψ are updated to minimize the Bellman Mean Squared Error (MSE). Given a transition tuple (s_i, a_i, r_i, s_{i+1}) sampled from the replay buffer $\mathcal{D}$, the loss is defined as:

$$\mathcal{L}_Q(\psi) = \mathbb{E}_{\mathcal{D}} \left[\left(y_i - Q_\psi(s_i, a_i)\right)^2\right], \tag{19}$$

where y_i is the regression target constructed using the target networks $\bar{Q}_{\bar{\psi}_1}, \bar{Q}_{\bar{\psi}_2}$. To mitigate value overestimation, we employ the Clipped Double Q-learning mechanism [7]:

$$y_i = r_i + \gamma \min_{j=1,2} \bar{Q}_{\bar{\psi}_j}(s_{i+1}, a_{i+1}), \tag{20}$$



(a) Kettle (b) Bot. burner (c) Light switch (d) Top burner (e) Slide cabinet

(f) Top burner. (g) Slide cabinet (h) Light switch

Figure 4: Agents working in the default FrankaKitchen environment with a budget of 280 steps. Top row (a-e): FMER is able to finish 5 tasks. Bottom row (f-h): The best baseline model (DIPO) is able to finish 3 tasks. In this visualization, we use the best seed for both FMER and DIPO.

where $a^*_{i+1} = \arg\max_{a^{(m)}} \min_{j=1,2} \bar{Q}_{\bar{\psi}_j}(s_{i+1}, a^{(m)})$ and $\{a^{(m)}\}_{m=1}^{M} \sim \pi_\theta(\cdot|s_{i+1})$. As detailed in Section 4.2, we optimize the weighted FM loss using M sampled actions. During data collection, following [6, 14], we execute the highest-valued of M candidates to interact with the environment. We store the resulting transition in the replay buffer to ensure the replay buffer accumulates high-reward transitions. Similarly, we compute the target for the critic update (Eq. 20) using the highest-valued candidate to reduce Bellman backup variance.

5 Experiments

In this section, we first visualize the multimodal expressiveness of FMER on a simple multi-goal environment, then evaluate performance on long-horizon sparse-reward multi-task FrankaKitchen tasks and dense-reward MuJoCo benchmarks. Finally, we present ablation studies to quantify the individual contribution of each component.

5.1 Visualizing Policies for *Multi-Goal*

To compare the multimodal capacity of different stochastic policy families, we use the standard 2D multi-goal environment commonly adopted for interpretability studies [10]. We compare SAC (Gaussian policy), DPMD (diffusion policy), and FMER (our flow matching policy). Figure 3 shows state-conditioned action selection across the environment after convergence. As indicated by the red arrows, SAC and DPMD consistently favor a subset of optimal regions (right and bottom), even when alternative peaks are equally optimal. In contrast, FMER adapts to the local geometry and points toward the nearest of the four equal-valued maxima. This indicates a diverse multimodal behavior facilitated by the tractable entropy control of our ODE-based formulation. Further visualization of the action generation process is provided in Appendix C.3.

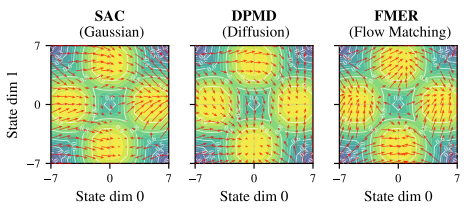


Figure 3: State-conditioned policy visualization on the 2D multi-goal task. Background colors display the ground-truth state-value landscape, with bright areas marking optimal goals. Red arrows show the selected actions for SAC [11], DPMD [21], and FMER, illustrating each method's navigation strategy.

5.2 Benchmarks on FrankaKitchen and Mujoco

We evaluate FMER against eight baselines across FrankaKitchen [9] and MuJoCo [34], spanning standard RL (PPO [31], SAC [11], TD3 [7]), diffusion-based (DIPO [38], QVPO [6], DPMD [21]),

Table 2: Normalized task completion rate on FrankaKitchen environment and performance on MuJoCo-v5 environment. Values are presented as Mean (Std) across 5 random seeds. For each seed, the policy is evaluated over 10 independent episodes.

Method	FrankaKitchen (Task Completion Rate)				MuJoCo-v5			
	N=1	N=2	N=4	N=7	Hopper	HalfCheetah	Walker	Humanoid
PPO	2.00 (4.47)	0.00 (0.00)	1.00 (2.24)	8.57 (7.82)	1198.02 (360.64)	5982.74 (810.60)	4598.21 (222.17)	811.94 (78.20)
SAC	40.00 (54.77)	32.00 (24.90)	2.50 (3.54)	12.29 (4.24)	3318.64 (132.80)	10295.33 (1170.43)	4441.51 (536.85)	**5326.73 (58.93)**
TD3	0.00 (0.00)	11.00 (24.60)	15.00 (17.32)	5.71 (7.82)	**3662.14 (61.58)**	9496.59 (901.14)	4887.65 (630.96)	318.33 (208.02)
DIPO	0.00 (0.00)	33.00 (30.33)	14.50 (21.39)	23.14 (18.36)	3408.50 (297.31)	9516.47 (1417.20)	5028.66 (217.36)	5087.46 (100.04)
DPMD	60.00 (54.77)	**50.00 (0.00)**	25.00 (17.68)	20.57 (8.67)	3397.57 (278.06)	10706.06 (817.59)	4901.24 (351.59)	5084.60 (27.90)
QVPO	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	18.86 (9.44)	3422.73 (413.41)	10309.60 (1398.37)	4846.39 (397.71)	5064.21 (48.82)
SAC-Flow	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.57 (1.28)	2930.95 (510.33)	9933.18 (254.50)	5233.78 (728.87)	4857.14 (2763.20)
FLAME	8.00 (17.89)	10.00 (22.36)	6.50 (14.53)	18.29 (7.79)	2593.95 (432.33)	10900.69 (1478.33)	4407.20 (143.72)	5207.44 (345.98)
FMER	**80.00 (44.72)**	48.00 (4.47)	**46.50 (3.79)**	**40.29 (21.03)**	2913.45 (642.67)	**12332.38 (903.67)**	**5282.89 (437.57)**	5286.12 (65.35)

and flow-based (SAC-Flow [42], FLAME [17]) methods. Table 1 provides a conceptual comparison; full hyperparameters are in Appendix C.1–C.2. All methods are trained for 1M environment steps over 5 random seeds. Off-policy methods include a 10K-step random warm-up and all policies are evaluated every 10K steps over 10 independent episodes. Results report mean maximum return over the final 10% of training (steps 900K–1M) following [36]; training time comparisons are in Appendix C.7.

FrankaKitchen. The FrankaKitchen benchmark [9] features a 9-DoF Franka robot with sparse rewards across interactable kitchen objects, evaluated on four task configurations $N \in \{1, 2, 4, 7\}$ (setup details in Appendix C.4). The environment is sparse: the agent has 280 steps per episode and receives a binary reward of 1 only upon completing a sub-task. It also exhibits a multimodal value distribution, as multiple distinct interaction sequences can satisfy the same goal. As shown in Table 2 and Figure 4, FMER outperforms all baselines across nearly all configurations with increasingly superior results at higher N.

Gaussian policies (PPO, SAC, TD3) struggle with the non-Gaussian optimal behavior distribution; SAC degrades from $N = 2$ to $N = 4$ as action multimodality grows beyond what a unimodal policy can capture. Among generative baselines, DPMD's KL trust region locks onto early successes but prevents escape from local optima at higher N; QVPO's uniform-distribution entropy objective and SAC-Flow's biased entropy surrogate both limit effective exploration in sparse settings. FMER's tractable entropy and flow-matching backbone enable principled exploration and multimodal target tracking throughout, yielding consistently superior performance. Full results and analysis, including an additional comparison with the on-policy method FPO [22], are provided in Appendix C.5.

MuJoCo-v5. Following prior diffusion- and FM-based work [6, 21, 42], we include MuJoCo as a regression test for whether FMER's multimodal machinery incurs unnecessary costs in simpler settings. As shown in Table 2, FMER attains the highest returns on HalfCheetah and Walker2d, and outperforms all other generative methods on Humanoid (trailing only SAC). However, FMER underperforms off-policy baselines on Hopper. We attribute this to the entropy constraint sustaining exploration longer than necessary in this relatively simple unimodal environment. Training curves and detailed analysis are provided in Appendix C.6.

5.3 Ablation study

Ablation on Best-of-M Action Selection. As described in Section 4.4, FMER selects the highest-valued action among M candidates for both environment interaction and critic-target computation. To examine whether this selection mechanism explains FMER's advantage, we compare it with DPMD, the strongest PMD-based baseline on FrankaKitchen. Both methods use best-of-M selection by default, with $M = 32$ for DPMD and $M = 8$ for FMER. We evaluate both methods under their default settings and a matched no-argmax ablation, in which a single action is sampled for environment interaction and critic-target computation, removing both argmax operations. The candidate sets used for actor training remain unchanged.

Table 3: Ablation on best-of-M action selection on FrankaKitchen. Task completion rates (%) are presented as Mean (Std).

Method	$N = 1$	$N = 2$	$N = 4$	$N = 7$
DPMD, default	60.00 (54.77)	**50.00 (0.00)**	25.00 (17.68)	20.57 (8.67)
FMER, default	80.00 (44.72)	48.00 (4.47)	**46.50 (3.79)**	**40.29 (21.03)**
DPMD, no argmax	0.00 (0.00)	0.00 (0.00)	5.00 (11.18)	0.86 (1.28)
FMER, no argmax	**100.00 (0.00)**	50.00 (35.36)	26.50 (2.24)	14.29 (0.00)

We remark that, although $N = 1$ offers fewer reward opportunities, it is also the simplest task. As shown in Table 3, candidate argmax does not fully explain FMER's advantage over DPMD. Under the matched no-argmax setting, FMER continues to learn across all task configurations, whereas DPMD barely succeeds. Moreover, removing candidate argmax does not reduce FMER's mean performance on $N = 1$ or $N = 2$, but substantially reduces it on $N = 4$ and $N = 7$. This comparison complements the entropy ablation: it rules out candidate argmax as the sole explanation for the performance gap, while acknowledging its substantial contribution on the harder long-horizon settings, likely because it helps collect rewarding transitions and produces stronger critic targets.

Entropy constraint. In Figure 5, we demonstrate the necessity of entropy regularization by deactivating $\mathcal{L}_{\text{ent}}$ at different m training steps. When deactivated from the start ($m = 0$), the policy collapses to near-deterministic behavior: ESS keeps being above ESS^*, driving $\tau \to 0$. Although clipping τ from below prevents numerical instability, it cannot avert the underlying policy collapse. In contrast, deactivating the constraint late in training ($m = 5 \times 10^5$) has limited impact, as sufficient exploration has already been established. In both cases, entropy drops immediately upon deactivation, confirming the efficiency of entropy control with α. Appendix D.2 validates the theoretical and practical necessity of the $\tanh$ correction (Proposition 1), and Appendix D.3 evaluates the computational trade-offs of exact trace calculation.

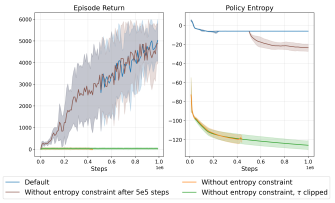


Figure 5: Training curves (left) and entropy evolution (right) when removing the entropy constraint after m steps on Walker2d-v5.

Advantage-weighting ablations. Figure 6 ablates the mirror descent weighting by replacing the soft advantage weights in Eq. (11) with uniform weights, reducing the loss to standard behavior cloning over all sampled actions regardless of quality. This significantly harms the performance: assigning equal weight to high-quality and low-quality actions pulls the policy toward mediocre behavior, confirming that exponential advantage weighting is a necessary mechanism for directed policy improvement rather than a soft heuristic. Appendix D.4 shows that ESS-based adaptive τ avoids environment-specific hyperparameter tuning. We provide further ablation studies on candidate size in Appendix D.5, and on ODE steps and solvers in Appendix D.6.

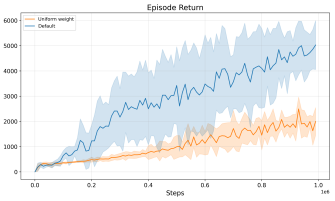


Figure 6: Training curves comparing advantage-based weight and uniform weight on Walker2d-v5.

We repeat both the entropy and advantage-weighting ablations on sparse-reward FrankaKitchen ($N = 7$), with results reported in Appendix D.1.

6 Conclusion

We presented FMER, an FM policy optimization framework for online RL that addresses two fundamental challenges: policy expressiveness and the exploration-exploitation tradeoff. We show that the policy mirror descent update can be realized via advantage-weighted conditional flow matching loss, enabling simulation-free training without backpropagating through the ODE. We further derive a closed-form tractable entropy expression which explicitly corrects for the density distortion induced by the $\tanh$ transformation from the unbounded latent space to the bounded action space. Empirical results validate each contribution: the 2D multi-goal environment confirms the multimodal expressiveness of the flow matching policy, FrankaKitchen and MuJoCo benchmarks demonstrate that FMER achieves superior or competitive performance across sparse- and dense-reward settings, and ablation studies isolate the necessity of both mirror descent weighting and closed-form entropy control. Limitations and future work are discussed in Appendix B.8.

Acknowledgment

This work was jointly supported by the EMERALDS project under the Horizon Europe Research and Innovation Programme (Grant Agreement No. 101093051) and the Transport & Mobility Institute at Delft University of Technology. The authors acknowledge the use of computational resources of the DelftBlue supercomputer, provided by Delft High Performance Computing Centre. This work also used the Dutch national e-infrastructure with the support of the SURF Cooperative using grant no. EINF-16178.

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/forum?id=sP1fo2K9DFG`.

[2] Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. DIME: diffusion-based maximum entropy reinforcement learning. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025. URL `https://openreview.net/forum?id=Aw6dBR7Vxj`.

[3] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/forum?id=42zs3qa2kpy`.

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

[5] Tianyi Chen, Haitong Ma, Na Li, Kai Wang, and Bo Dai. One-step flow policy mirror descent. *arXiv preprint arXiv:2507.23675*, 2025.

[6] Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

[7] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, 2018. URL `https://proceedings.mlr.press/v80/fujimoto18a.html`.

[8] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=rJxgknCcK7`.

[9] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1025–1037. PMLR, 30 Oct–01 Nov 2020. URL `https://proceedings.mlr.press/v100/gupta20a.html`.

[10] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, 2018. URL `https://proceedings.mlr.press/v80/haarnoja18b.html`.

[12] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[13] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[14] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego,*

CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980.

[16] Guanghui Lan. Policy mirror descent for reinforcement learning: Linear convergence, new sampling complexity, and generalized problem classes. *Mathematical programming*, 198(1):1059–1106, 2023.

[17] Zeqiao Li, Yijing Wang, Haoyu Wang, Zheng Li, and Zhiqiang Zuo. Boosting maximum entropy reinforcement learning via one-step flow matching, 2026. URL https://arxiv.org/abs/2602.01606.

[18] Zeyang Li, Sunbochen Tang, and Navid Azizan. Reverse flow matching: A unified framework for online reinforcement learning with diffusion and flow policies, 2026. URL https://arxiv.org/abs/2601.08136.

[19] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

[20] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di ZHANG, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=oCBKGw5HNf.

[21] Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=6Anv3KB91z.

[22] David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients, 2025. URL https://arxiv.org/abs/2507.21053.

[23] Safa Messaoud, Skander Charni, Elaa Bouazza, Ali Pourghasemi Fatideh, and Halima Bensmail. Particles don't care about z: Towards scaling entropy estimation of unnormalized densities. In *Forty-third International Conference on Machine Learning*.

[24] Safa Messaoud, Billel Mokeddem, Zhenghai Xue, Linsey Pang, Bo An, Haipeng Chen, and Sanjay Chawla. S 2 ac: Energy-based reinforcement learning with stein soft actor critic. In *International Conference on Learning Representations*, volume 2024, pages 22323–22336, 2024.

[25] Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning (ICML)*, 2025.

[26] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[27] Samuel Pfrommer, Yixiao Huang, and Somayeh Sojoudi. Reinforcement learning for flow-matching policies. *arXiv preprint arXiv:2507.15073*, 2025.

[28] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[29] Allen Z. Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=mEpqHvbD2h.

[30] Quentin Rouxel, Clemente Donoso, Fei Chen, Serena Ivaldi, and Jean-Baptiste Mouret. Extremum flow matching for offline goal conditioned reinforcement learning. In *24th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2025, Seoul, Republic of Korea, September 30 - October 2, 2025*, pages 981–988. IEEE, 2025. doi: 10.1109/HUMANOIDS65713.2025.11203207. URL https://doi.org/10.1109/Humanoids65713.2025.11203207.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

[32] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[33] Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. *arXiv preprint arXiv:2005.09814*, 2020.

[34] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[35] Zhenglin Wan, Jingxuan Wu, Xingrui Yu, Chubin Zhang, Mingcong Lei, Bo An, and Ivor Tsang. Fm-irl: Flow-matching for reward modeling and policy regularization in reinforcement learning, 2025. URL https://arxiv.org/abs/2510.09222.

[36] Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, and Shengbo Li. Diffusion actor-critic with entropy regulator. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[37] Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

[38] Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *CoRR*, abs/2305.13122, 2023. doi: 10.48550/ARXIV.2305.13122. URL https://doi.org/10.48550/arXiv.2305.13122.

[39] Chubin Zhang, Zhenglin Wan, Feng Chen, Fuchao Yang, Lang Feng, Yaxin Zhou, Xingrui Yu, Yang You, Ivor Tsang, and Bo An. Evolving diffusion and flow matching policies for online reinforcement learning, 2026. URL https://arxiv.org/abs/2512.02581.

[40] Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=HA0oLUvuGI.

[41] Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=ACagRwCCqu.

[42] Yixian Zhang, Shu'ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. SAC flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zZvWj4JrYj.

Contents

A Proofs **14**

A.1 Proof of Theorem 1 . . . 14

A.2 Proof of Theorem 2 . . . 15

A.3 Proof of Proposition 1 . . . 16

A.4 Derivation of the Policy Mirror Descent closed-form objective . . . 17

B Model design **18**

B.1 Clarification of $w(s, a)$ along the latent path . . . 18

B.2 Why not a joint optimization? . . . 18

B.3 Connections to Advantage-Weighted and Softmax Policy Methods . . . 19

B.4 Connections to MaxEnt methods . . . 19

B.5 Connections to Policy Mirror Descent Methods . . . 20

B.6 Hutchinson's trace estimator . . . 21

B.7 Remark on action-space entropy . . . 21

B.8 Limitations and future work . . . 21

C Experiments **21**

C.1 Baseline implementation . . . 21

C.2 FMER implementation . . . 22

C.3 Action generation trajectory in *multi-goal* environment . . . 22

C.4 FrankaKitchen environment setup . . . 22

C.5 FrankaKitchen performance . . . 23

C.6 MuJoCo training curves and dynamics . . . 24

C.7 Training and inference time . . . 25

D Ablation study **25**

D.1 Entropy and advantage-weighting ablations on FrankaKitchen . . . 25

D.2 Ablation on $\tanh$ correction in entropy derivation . . . 26

D.3 Ablation on divergence computation: Hutchinson vs. Exact Jacobian . . . 27

D.4 Ablation on adaptive τ . . . 27

D.5 Ablation on candidate size . . . 27

D.6 Ablation on ODE steps and solvers . . . 27

A Proofs

A.1 Proof of Theorem 1

Theorem 1 (Weighted Flow Matching Surrogate). Fix a temperature $\tau > 0$, a reference policy π_k, and an advantage function A. For each state s, let $w(s, a) = \frac{\exp(A(s,a)/\tau)}{Z(s)}$ where $Z(s) := \mathbb{E}_{a\sim\pi_k(\cdot|s)}\left[\exp(A(s, a)/\tau)\right]$, $x_1 = \text{arctanh}(a)$, and $x_t = (1 - t)x_0 + tx_1$. Consider the Weighted Conditional Flow Matching loss

$$\mathcal{L}_{\text{W-CFM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],s\sim\mathcal{D},a\sim\pi_k(\cdot|s),x_0\sim p_0}\left[w(s, a)\|v_\theta(x_t, t, s) - (x_1 - x_0)\|^2\right]. \tag{21}$$

Assume $0 < Z(s) < \infty$ for $\mathcal{D}$-almost every state, and that the velocity network is expressive enough to represent the target marginal velocity field. Under the integrability and regularity conditions, every global minimizer θ^*

induces the ideal policy mirror descent target

$$\pi_{\theta^*}(a \mid s) = \pi^*(a \mid s) := \frac{\pi_k(a \mid s)\exp(A(s,a)/\tau)}{Z(s)} \tag{22}$$

for $\mathcal{D}$-almost every s and almost every a.

Proof. Eq. (54) in Appendix A.4 identifies π^* as the ideal Policy Mirror Descent (PMD) optimizer:

$$\pi^*(a \mid s) = \frac{1}{Z(s)}\pi_k(a \mid s)\,\exp\left(\frac{A(s,a)}{\tau}\right) = w(s,a)\pi_k(a \mid s). \tag{23}$$

Let $T = \operatorname{arctanh}$ and define $q_1^*(\cdot \mid s) = T_{\#}\pi^*(\cdot \mid s)$ as the distribution of $x_1 = T(a)$ for $a \sim \pi^*(\cdot \mid s)$.

We assume that actions lie in $(-1,1)^d$, that $\mathbb{E}_{s\sim\mathcal{D}}\,\mathbb{E}_{x_1\sim q_1^*(\cdot|s)}\|x_1\|^2 < \infty$, and that each v_θ is regular enough for its ODE flow and continuity equation to have unique solutions on $[0,1]$.

Reweighting the endpoint distribution. The advantage weights can be absorbed into the endpoint distribution. Substituting Eq. (23) into Eq. (21), we obtain

$$\begin{aligned}
\mathcal{L}_{\text{W-CFM}}(\theta) &= \mathbb{E}_{s\sim\mathcal{D}}\left[\int \pi_k(a \mid s)w(s,a)\,\mathbb{E}_{t,x_0}\left[\|v_\theta(x_t,t,s) - (T(a) - x_0)\|^2\right] da\right] \\
&= \mathbb{E}_{s\sim\mathcal{D}}\left[\int \pi^*(a \mid s)\,\mathbb{E}_{t,x_0}\left[\|v_\theta(x_t,t,s) - (T(a) - x_0)\|^2\right] da\right] \\
&= \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{t,x_0,\,x_1\sim q_1^*(\cdot|s)}\left[\|v_\theta(x_t,t,s) - (x_1 - x_0)\|^2\right]\right].
\end{aligned} \tag{24}$$

The last equality follows because sampling $a \sim \pi^*(\cdot \mid s)$ and setting $x_1 = T(a)$ is equivalent to sampling $x_1 \sim q_1^*(\cdot \mid s)$. Throughout, $t \sim \mathcal{U}[0,1]$, $x_0 \sim p_0$, and the endpoint are sampled independently, conditional on s. Thus, weighting samples from π_k by w/Z yields the same population objective as unweighted Conditional Flow Matching (CFM) with the PMD target π^*, without requiring direct samples from π^*.

Define the inner conditional flow matching loss by

$$\mathcal{L}_{\text{CFM},s}(v) := \mathbb{E}_{t,x_0,x_1}\left[\|v(x_t,t,s) - (x_1 - x_0)\|^2\right]. \tag{25}$$

Then Eq. (24) becomes

$$\mathcal{L}_{\text{W-CFM}}(\theta) = \mathbb{E}_{s\sim\mathcal{D}}\left[\mathcal{L}_{\text{CFM},s}(v_\theta)\right]. \tag{26}$$

Recovering the ideal PMD policy. For each fixed state s, the inner loss $\mathcal{L}_{\text{CFM},s}$ in Eq. (26) is the standard CFM objective in [19] with endpoint distribution $q_1^*(\cdot \mid s)$. The assumed second-moment bound on x_1 and the finite second moment of the Gaussian prior p_0 ensure that the minimum state-averaged CFM loss is finite. By the conditional-mean decomposition underlying [19, Theorem 2], the conditional and marginal flow matching objectives differ only by a constant independent of v. Consequently, the minimizing field is

$$u_t^*(x,s) = \mathbb{E}[x_1 - x_0 \mid x_t = x, t, s], \tag{27}$$

whose flow transports p_0 to $q_1^*(\cdot \mid s)$ [19, Theorem 1]. Under the network expressiveness assumption, a single network parameter choice attains the minimum of each statewise loss for $\mathcal{D}$-almost every state. Every global minimizer θ^* of their average must therefore satisfy $v_{\theta^*} = u_t^*$ almost everywhere along the interpolation paths.

By the assumed uniqueness, the ODE

$$\frac{dX_t}{dt} = v_{\theta^*}(X_t,t,s), \qquad X_0 \sim p_0, \tag{28}$$

therefore produces $X_1 \sim q_1^*(\cdot \mid s)$. Transforming the terminal samples back to the action space gives

$$\pi_{\theta^*}(\cdot \mid s) = \tanh_{\#}\, q_1^*(\cdot \mid s) = \tanh_{\#}\big(T_{\#}\pi^*(\cdot \mid s)\big) = \pi^*(\cdot \mid s). \tag{29}$$

Since $\tanh \circ T$ is the identity, the induced action distribution equals π^*. Therefore, minimizing the population weighted CFM loss recovers the ideal PMD policy for $\mathcal{D}$-almost every state. □

A.2 Proof of Theorem 2

Theorem 2 (Entropy of Latent Flow [4, 19]). Consider a latent vector field $v_\theta(x,t,s)$ that transports a base distribution p_0 to the target latent distribution p_1 at $t = 1$. For the unbounded latent space $\mathbb{R}^d$, the entropy $H^x(p_1(\cdot|s))$ can be computed as:

$$H_1^x(p_1) = H_0^x + \int_0^1 \mathbb{E}_{x_t\sim p_t(\cdot|s)}\left[\nabla_x \cdot v_\theta(x_t,t,s)\right] dt, \tag{30}$$

where $H_0^x = \frac{d}{2}(1 + \ln(2\pi))$ is the entropy of the base Gaussian noise $p_0 = \mathcal{N}(0,\mathbf{I})$ and d is the latent dimension.

Proof. For simplicity, we use H_t to denote H_t^x in this proof. For all $x \in \mathbb{R}^d$ and $t \in [0,1]$, a flow model with vector field v_t and probability path $p_t(x)$ satisfies the continuity equation:

$$\frac{\partial p_t}{\partial t} = -\nabla \cdot (p_t v_t). \tag{31}$$

The entropy of the distribution p_t is defined as:

$$H_t = -\int p_t(x) \log p_t(x)\, dx. \tag{32}$$

Taking the time derivative:

$$\frac{d}{dt} H_t = -\int \left(\log p_t(x) \frac{\partial p_t(x)}{\partial t} + \frac{\partial p_t(x)}{\partial t} \right) dx. \tag{33}$$

Since $\int p_t(x)\, dx = 1$, the second term vanishes:

$$\frac{d}{dt} H_t = -\int \log p_t(x) \frac{\partial p_t(x)}{\partial t}\, dx. \tag{34}$$

Substituting Eq. (31) into the expression:

$$\frac{d}{dt} H_t = \int \log p_t(x) \nabla \cdot (p_t v_t)\, dx. \tag{35}$$

Assuming the probability mass vanishes at the boundaries, we apply integration by parts. Note that $\nabla \log p_t = \frac{\nabla p_t}{p_t}$:

$$\frac{d}{dt} H_t = -\int v_t \cdot \nabla p_t\, dx. \tag{36}$$

Using the identity $\nabla \cdot (p_t v_t) = p_t \nabla \cdot v_t + v_t \cdot \nabla p_t$ and the integration by parts:

$$\frac{d}{dt} H_t = \int p_t \nabla \cdot v_t\, dx. \tag{37}$$

Therefore,

$$\frac{d}{dt} H_t = \mathbb{E}_{x \sim p_t}[\nabla \cdot v_t]. \tag{38}$$

Integrating over time from $t = 0$ to $t = 1$:

$$H_1 = H_0 + \int_0^1 \mathbb{E}_{x \sim p_t}[\nabla \cdot v_t]\, dt. \tag{39}$$

Finally,

$$H_1 = H_0 + \mathbb{E}_{t \sim \mathcal{U}[0,1], x \sim p_t}[\nabla \cdot v_t]. \tag{40}$$

□

A.3 Proof of Proposition 1

Proposition 1 (Divergence under Tanh Transformation). Consider the coordinate transformation $a = \tanh(x)$ mapping any latent state $x \in \mathbb{R}^d$ to the bounded action space $(-1,1)^d$. Let $v_\theta(x,t,s)$ be the vector field in the latent space. The corresponding induced vector field in the action space, $v^a(a,t,s)$, satisfies the following divergence relationship at any point x:

$$\nabla_a \cdot v^a(a,t,s) = \nabla_x \cdot v_\theta(x,t,s) - \sum_{i=1}^{d} 2 \tanh(x_i) v_{\theta,i}(x,t,s), \tag{41}$$

where $v_{\theta,i}$ is the i-th component of the latent vector field.

Proof. Theorem 2 assumes an unbounded integration domain where boundary terms vanish. For bounded actions, we consider the domain $a \in (-1,1)^d$. We employ a change of variables where $a = \tanh(x)$ and the Flow Matching is performed in the unconstrained latent space $x \in \mathbb{R}^d$. The relationship between the velocity fields $v^a(a)$ in the action space and $v_\theta(x)$ in the latent space is given by the chain rule:

$$v_t^a(a) = \frac{da}{dt} = \frac{da}{dx}\frac{dx}{dt} = (1 - a^2) \odot v_\theta(x,t,s). \tag{42}$$

Note that here $(1 - a^2)$ acts element wise. The divergence in the action space is:

$$\nabla_a \cdot v_t^a(a) = \nabla_a \cdot \left((1 - a^2) \odot v_\theta(x) \right). \tag{43}$$

Using the product rule for divergence $\nabla \cdot (f\mathbf{v}) = \nabla f \cdot \mathbf{v} + f \nabla \cdot \mathbf{v}$:

$$\nabla_a \cdot v_t^a(a) = \sum_{i=1}^{d} \left(v_{\theta,i}(x) \frac{\partial(1-a_i^2)}{\partial a_i} + (1-a_i^2) \frac{\partial v_{\theta,i}(x)}{\partial a_i} \right). \tag{44}$$

The first term simplifies to $-2a_i v_{\theta,i}(x)$. For the second term, applying the chain rule yields:

$$\frac{\partial v_{\theta,i}(x)}{\partial a_i} = \frac{\partial v_{\theta,i}(x)}{\partial x_i} \frac{\partial x_i}{\partial a_i} = \frac{\partial v_{\theta,i}(x)}{\partial x_i} \frac{1}{1-a_i^2}. \tag{45}$$

Substituting Eq. (45) back into Eq. (44), the term $(1-a_i^2)$ cancels out:

$$\nabla_a \cdot v_t^a(a) = \sum_{i=1}^{d} \left(-2a_i v_{\theta,i}(x) + \frac{\partial v_{\theta,i}(x)}{\partial x_i} \right) = \nabla_x \cdot v_\theta(x) - \sum_{i=1}^{d} 2a_i v_{\theta,i}(x). \tag{46}$$

This confirms the relationship stated in Proposition 1. □

A.4 Derivation of the Policy Mirror Descent closed-form objective

We follow the same pipeline as in Appendix A, AWR [26]. AWR is an offline RL algorithm, here we focus on online RL. The data buffer $\mathcal{D}$ approximates on-policy samples from π_k. Our goal is to find a policy π that maximizes the expected advantage $A^{\pi_k}(s,a)$ with respect to the current policy π_k, subject to a trust-region constraint to stabilize learning. Using the expected advantage, we can formulate the following constrained policy search problem:

$$\arg\max_{\pi} \int_s d_{\pi_k}(s) \int_a \pi(a|s) A^{\pi_k}(s,a) da\, ds \quad \text{s.t.} \quad D_{\mathrm{KL}}(\pi(\cdot|s) \| \pi_k(\cdot|s)) \le \epsilon, \int_a \pi(a|s) da = 1, \forall s, \tag{47}$$

where $d_{\pi_k}(s)$ as the unnormalized discounted state distribution induced by the policy π_k [32].

We relax the hard KL constraint by converting it into a soft constraint with coefficient τ (acting as a temperature parameter). Following AWR [26], we adopt a global scalar multiplier τ rather than a state-dependent $\tau(s)$, which relaxes the pointwise KL constraint to an expected KL constraint over the state distribution d^{π_k}.

$$\begin{aligned} \arg\max_{\pi} \left(\int_s d_{\pi_k}(s) \int_a \pi(a|s) A^{\pi_k}(s,a) da\, ds \right) + \tau \left(\epsilon - \int_s d_{\pi_k}(s) D_{\mathrm{KL}}(\pi(\cdot|s) \| \pi_k(\cdot|s)) ds \right) \\ \text{s.t.} \int_a \pi(a|s) da = 1, \forall s. \end{aligned} \tag{48}$$

Next, we form the Lagrangian, with τ and $\beta = \{\beta_s | \forall s \in \mathcal{S}\}$ corresponding to the Lagrange multipliers:

$$\begin{aligned} \mathcal{L}(\pi, \tau, \beta) = \left(\int_s d_{\pi_k}(s) \int_a \pi(a|s) A^{\pi_k}(s,a) da\, ds \right) + \tau \left(\epsilon - \int_s d_{\pi_k}(s) D_{\mathrm{KL}}(\pi(\cdot|s) \| \pi_k(\cdot|s)) ds \right) \\ + \int_s \beta_s \left(1 - \int_a \pi(a|s) da \right) ds. \end{aligned} \tag{49}$$

Differentiating $\mathcal{L}(\pi, \tau, \beta)$ with respect to $\pi(a|s)$ results in:

$$\frac{\partial \mathcal{L}}{\partial \pi(a|s)} = d_{\pi_k}(s) A^{\pi_k}(s,a) - \tau d_{\pi_k}(s) \log \pi(a|s) + \tau d_{\pi_k}(s) \log \pi_k(a|s) - \tau d_{\pi_k}(s) - \beta_s. \tag{50}$$

Setting to zero and solving for $\pi(a|s)$ gives:

$$\log \pi(a|s) = \frac{1}{\tau} A^{\pi_k}(s,a) + \log \pi_k(a|s) - 1 - \frac{1}{d_{\pi_k}(s)} \frac{\beta_s}{\tau} \tag{51}$$

$$\pi(a|s) = \pi_k(a|s) \exp\left(\frac{A^{\pi_k}(s,a)}{\tau} \right) \exp\left(-\frac{1}{d_{\pi_k}(s)} \frac{\beta_s}{\tau} - 1 \right) \tag{52}$$

Since $\int_a \pi(a|s) da = 1$, the second exponential term is the partition function $\frac{1}{Z(s)}$ that normalizes the conditional action distribution:

$$Z(s) = \exp\left(\frac{1}{d_{\pi_k}(s)} \frac{\beta_s}{\tau} + 1 \right) = \int_{a'} \pi_k(a'|s) \exp\left(\frac{A^{\pi_k}(s,a')}{\tau} \right) da' \tag{53}$$

The optimal, non-parametric policy is therefore given by:

$$\pi^*(a|s) = \frac{1}{Z(s)} \pi_k(a|s) \exp\left(\frac{A^{\pi_k}(s,a)}{\tau} \right) \tag{54}$$

If π is represented by a function approximator (e.g., a neural network π_θ), the optimal policy π^* can be projected onto the manifold of parameterized policies by solving the following supervised regression problem via KL divergence minimization under the data distribution:

$$\arg\min_\theta \mathbb{E}_{s\sim\mathcal{D}}\left[D_{\mathrm{KL}}(\pi^*(\cdot|s)\|\pi_\theta(\cdot|s))\right] \tag{55}$$

$$= \arg\min_\theta \mathbb{E}_{s\sim\mathcal{D}}\left[D_{\mathrm{KL}}\left(\frac{1}{Z(s)}\pi_k(\cdot|s)\exp\left(\frac{A^{\pi_k}(s,\cdot)}{\tau}\right)\middle\|\pi_\theta(\cdot|s)\right)\right] \tag{56}$$

$$= \arg\max_\theta \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_k}\left[\frac{1}{Z(s)}\exp\left(\frac{A^{\pi_k}(s,a)}{\tau}\right)\log\pi_\theta(a|s)\right] \tag{57}$$

AWR [26] drops $Z(s)$ entirely, which is one source of its instability under heterogeneous Q-scales across states. FMER retains and empirically approximates it via the softmax denominator in Eq. (13). Specifically, $\sum_{l=1}^{M}\exp(A(s,a^{(l)})/\tau)$ serves as a self-normalized Monte Carlo estimate of $M\cdot Z(s)$ over the candidate action set $\{a^{(l)}\}_{l=1}^{M}\sim\pi_k(\cdot|s)$. This approximation becomes exact as $M\to\infty$. It also preserves the state-dependent normalization that ensures weights sum to one for each state. .

B Model design

B.1 Clarification of $w(s,a)$ along the latent path

We first clarify that x_t is an auxiliary flow-time state used only for velocity regression; it is not an action executed in the environment, and there is therefore *no independently defined action value $Q(s,x_t)$ (thus we cannot define $w(s,x_t)$)*. Only the terminal output $a=\tanh(x_1)$ is executed and evaluated by the critic. Accordingly, $w(s,a)$ assigns importance to a sampled terminal action and its associated conditional path.

Importantly, using the same weight along one conditional path does not give equal updates to paths leading to good and bad actions. A path ending at a high-advantage action is multiplied by $w(s,a_{\mathrm{high}})$, whereas a path ending at a low-advantage action is multiplied by the smaller $w(s,a_{\mathrm{low}})$, at every sampled flow time.

- **Consistency with the endpoint objective.** The use of the same endpoint weight across flow time follows from the endpoint-weighted AWR objective in Theorem 1. As given by Eq. (23), the target policy is derived by $w(s,a)$ *depending only on the terminal action*. Replacing $w(s,a)$ with a separate time- or latent-dependent weight $w_t(s,x_t,a)$ would define a different path-dependent objective and would no longer correspond to the same endpoint-weighted policy update.
- **Overlapping conditional paths.** Different flow matching paths can pass through the same latent region and prescribe different target velocities. Standard Conditional Flow Matching resolves this through the marginal vector field. For a fixed (x,t,s), consider

$$\ell(v)=\mathbb{E}\big[w(s,a)\,\|v(x,t,s)-(x_1-x_0)\|^2\,\big|\,x_t=x,t,s\big]\,. \tag{58}$$

  Differentiating with respect to v and setting the derivative to zero yields the optimal solution

$$v_w^*(x,t,s)=\frac{\mathbb{E}[w(s,a)(x_1-x_0)\,|\,x_t=x,t,s]}{\mathbb{E}[w(s,a)\,|\,x_t=x,t,s]}. \tag{59}$$

  Therefore, when paths toward high- and low-value actions overlap, the learned vector field does not branch. Instead, it aggregates their target velocities, with higher-advantage endpoints contributing more strongly.

B.2 Why not a joint optimization?

We acknowledge that our current Lagrangian in Eq. (7) decouples the constraints. However, we specifically avoid joint optimization to maintain the benefits of simulation-free training. Following the policy mirror descent in Eq. (47), if we were to include the entropy constraint directly into the primary maximization problem, the objective would become:

$$\begin{aligned}&\arg\max_\pi \int_s d_{\pi_k}(s)\int_a \pi(a|s)A^{\pi_k}(s,a)da\,ds\\ &\text{s.t.}\quad \mathbb{E}_{s\sim\mathcal{D}}[H(\pi(\cdot|s))]\geq\bar{H},\quad D_{\mathrm{KL}}(\pi(\cdot|s)\|\pi_k(\cdot|s))\leq\epsilon,\quad \int_a\pi(a|s)da=1,\forall s.\end{aligned} \tag{60}$$

For this joint maximization, we form the Lagrangian with τ, α, and $\beta = \{\beta_s | \forall s \in \mathcal{S}\}$ corresponding to the respective Lagrange multipliers:

$$\begin{aligned}\mathcal{L}(\pi, \tau, \alpha, \beta) = &\left(\int_s d_{\pi_k}(s) \int_a \pi(a|s) A^{\pi_k}(s,a) da\, ds\right) + \tau\left(\epsilon - \int_s d_{\pi_k}(s) D_{\mathrm{KL}}(\pi(\cdot|s) \| \pi_k(\cdot|s)) ds\right) \\ &+ \alpha\left(\mathbb{E}_{s\sim\mathcal{D}}[H(\pi(\cdot|s))] - \bar{H}\right) + \int_s \beta_s \left(1 - \int_a \pi(a|s) da\right) ds.\end{aligned} \tag{61}$$

Note that $H(\pi) = -\int \pi(a|s) \log \pi(a|s) da$. Similar to Appendix A.4, taking the derivative with respect to $\pi(a|s)$ yields the optimal policy π^*:

$$\pi^*(a|s) \propto \pi_k(a|s)^{\frac{\tau}{\alpha+\tau}} \exp\left(\frac{A^{\pi_k}(s,a)}{\alpha+\tau}\right) \tag{62}$$

Without the entropy constraint ($\alpha = 0$), the base policy $\pi_k(a|s)$ has an exponent of 1 and can be absorbed into importance sampling. However, with $\alpha > 0$, the fractional exponent $\frac{\tau}{\alpha+\tau}$ requires the explicit evaluation of $\pi_k(a|s)$. In our framework, this would require simulating the full backward ODE at every single training step, fundamentally destroying the simulation-free advantage of Conditional Flow Matching (CFM).

B.3 Connections to Advantage-Weighted and Softmax Policy Methods

Several prior works employ advantage-weighted or softmax-style objectives for diffusion- and flow-based policies. We clarify the key connections and distinctions below.

- **AWR [26].** As discussed in Section 4, our advantage-weighted policy objective shares the same trust-region motivation as AWR. The key distinction lies in optimization: directly applying AWR's update to a flow matching policy would require backpropagating through the full ODE integration chain, which is both computationally expensive and numerically unstable. We instead optimize a simulation-free surrogate by sampling $t \sim \mathcal{U}(0,1)$ and regressing the vector field without rolling out the ODE, with Theorem 1 providing the formal justification.
- **QVPO [6].** QVPO maximizes $\mathbb{E}_{s,a\sim\pi_k(\cdot|s)}\left[Q(s,a)\log\pi_\theta(a|s)\right]$, which requires $Q(s,a) \geq 0$ and it is an assumption that can be violated in general. QVPO addresses this via engineering fixes (`qcut`, `qadv`) whose guarantees do not transfer back to the original objective. By contrast, our weights arise directly from policy mirror descent (Section 4.2): using advantage values $A(s,a)$ rather than raw Q-values renders the surrogate relationship in Theorem 1 unconditional, requiring no assumption on reward sign or value scale.
- **IDQL [12].** IDQL applies value-function reweighting only at *inference time* via greedy argmax selection, while the diffusion model itself is trained as a pure behavior cloner without any value-function weighting. In our work, we integrate softmax-normalized advantage weights directly into the *training* objective at every gradient step, coupling policy improvement with representation learning throughout training.
- **SDAC [21].** Among the two methods proposed in [21], SDAC is the one that most resembles a softmax policy $\pi^*(a|s) \propto \exp(Q(s,a)/\lambda)$, whose normalizing constant $Z = \int \exp(Q(s,a)/\lambda)\, da$ is intractable. During training, SDAC samples a_t from an arbitrary proposal (e.g., uniform), draws noisy estimates $\tilde{a}_0 \sim \mathcal{N}(c_t^1 a_t,\, c_t^2)$ using diffusion coefficients c_t^1, c_t^2, and applies exponential weighting to these *noisy* samples; normalization serves only as a numerical stability fix. In our work, softmax weights are applied directly to clean candidate actions sampled from the current policy π_k, and are grounded in the advantage-weighted log-likelihood objective of Theorem 1. This provides a consistent, state-wise learning signal that is invariant to the absolute scale of Q-values across states.

B.4 Connections to MaxEnt methods

Recent flow-based methods approach entropy calculation in various ways. We note that FLAME [17] is a concurrent and independent work. Below, we contrast our method with these existing techniques:

- **ReinFlow [41]**. Designed for fine-tuning pre-trained policies, ReinFlow controls entropy by injecting noise into existing probability paths. However, in from-scratch online training, this noise injection provides no meaningful exploration signal and leads to policy collapse.
- **SAC-Flow [42]**. Whereas SAC-Flow addresses gradient instability during backpropagation through time (BPTT), our approach eliminates the need for BPTT by utilizing simulation-free training. SAC-Flow estimates entropy via a noise-augmented joint path density. This approach overestimates the true policy entropy, causing the optimizer to perceive the policy as more stochastic than it actually is, thereby diminishing the effective exploration pressure. While this bias may provide a stabilizing

effect in dense-reward settings, it fundamentally restricts exploration in sparse-reward environments, as confirmed by our experiments in Section 5.2: SAC-Flow collapses entirely on FrankaKitchen, underperforming even standard SAC, whereas FMER outperforms SAC-Flow by maintaining precise, closed-form entropy control throughout training.

- **FLAME [17]**. Although FLAME does not incorporate a policy mirror descent trust region in entropy computation. Both methods exploit the tractability of ODE-based flow matching for closed-form entropy computation and employ Hutchinson's estimator to efficiently approximate the vector field divergence. The key differences lie in architecture and action space handling. FLAME primarily targets discretization bias in one-step FM policies, and addresses this through a decoupled actor-critic design: the actor generates actions in a single step for low-latency inference, while the critic evaluates entropy over a multi-step integration grid to preserve regularization accuracy. For bounded action spaces, FLAME enforces boundary constraints directly on the reverse sampling process via a truncated normal distribution. In contrast, FMER does not constrain the sampling process itself. Instead, we formulate the entire flow in an unbounded latent space and derive an explicit analytical correction for the probability density distortion induced by the $\tanh$ transformation into the bounded action space (Proposition 1). This latent-space formulation preserves the theoretical guarantees of standard flow matching while remaining compatible with simulation-free training.
- **RFM [18]**. This work is theory-intensive. One major contribution of RFM is to place existing MaxEnt-style diffusion and flow policy methods under a unified framework: the noise-expectation family and the gradient-expectation family. The paper starts from the MaxEnt soft policy improvement target, whose closed-form solution is a Boltzmann distribution over actions. Because this Boltzmann distribution is unnormalized and difficult to sample from directly, the authors propose RFM, which converts the problem into posterior mean estimation. They use self-normalized importance sampling to estimate these posterior means and introduce Langevin Stein control variates to reduce sampling variance. Overall, RFM mainly follows the MaxEnt perspective and builds on the Boltzmann closed-form solution for soft policy improvement. In contrast, our paper incorporates the entropy constraint through a Lagrangian formulation, and our closed-form objective is derived from policy mirror descent.

Another line of work estimates policy entropy in closed form without a diffusion or flow backbone. S²AC [24] represents the policy as a Stein variational gradient descent (SVGD) sampler that transports particles from a learned Gaussian toward the energy-based distribution over Q-values through a fixed number of deterministic, invertible steps, so that the particle density can be tracked with the change-of-variables formula and its entropy approximated to first order using only first-order derivatives and inner products. The follow-up MET-SVGD [23] shows that this estimator breaks down beyond a few tens of dimensions; it restores the missing term with Hutchinson's estimator, derives a global invertibility condition, learns the kernel bandwidth and step size end to end, and adds a Metropolis–Hastings correction. FMER shares the goal of an expressive policy with analytic entropy but reaches it differently: Theorem 2 is the continuous-time change of variables along an ODE whose flow is invertible by construction, so no step-size, invertibility, or acceptance conditions arise, and Hutchinson's estimator is applied to the divergence of the learned velocity field rather than to the curvature of the target density. Both SVGD methods are full MaxEnt algorithms with entropy inside the soft Bellman target, whereas FMER imposes entropy only as an actor-side constraint, and neither accounts for bounded actions through an analytic density correction.

B.5 Connections to Policy Mirror Descent Methods

- **DPMD [21].** FMER and DPMD share the same ideal nonparametric policy mirror descent target. The distinction lies in how this shared target is realized and optimized. (1) Different regression objects. DPMD uses a diffusion policy, interpreted as a sequence of noise-perturbed energy-based models (EBMs), which can be optimized using reweighted denoising score matching (DSM). Its tractable target relies on a Gaussian corruption kernel $q_{t|0}(a_t \mid a_0)$, whose conditional score $\nabla_{a_t} \log q_{t|0}(a_t \mid a_0)$ is available analytically. FMER uses the deterministic OT path $x_t = (1-t)x_0 + tx_1$. Conditioned on a fixed endpoint pair (x_0, x_1), this path has a Dirac conditional distribution, so DPMD's Gaussian-kernel DSM objective does not directly apply to our endpoint-conditioned deterministic path. (2) Different finite-sample normalization. In FMER, we approximate $MZ(s)$ using $\sum_{l=1}^{M} \exp\big(A(s, a^{(l)})/\tau\big)$ over M candidates, producing self-normalized weights. DPMD, following the EBM/DSM route, instead uses unnormalized exponential Q-weights, while its running mean-and-variance normalization is an engineering fix for numerical stability. FMER's contribution is therefore a weighted-CFM realization of an existing PMD principle, combined with tractable, $\tanh$-corrected action-space entropy control.
- **FPMD [5]** FPMD is primarily designed to enable one-step flow-policy inference, whereas FMER retains multi-step ODE sampling and focuses on exploration in sparse-reward settings. For policy optimization, FPMD uses unnormalized exponential weights with a fixed temperature, while FMER uses candidate-wise self-normalized weights and adapts the temperature to maintain a target effective sample size. For exploration, FPMD uses scheduled Gaussian action noise, whereas FMER enforces a

minimum-entropy constraint using an analytical action-space entropy objective that accounts for the density distortion induced by the $\tanh$ transformation.

B.6 Hutchinson's trace estimator

In practice, computing the divergence $\nabla_x \cdot v_\theta(x) = \sum_{i=1}^{d} \frac{\partial v_{\theta,i}}{\partial x_i}$ in Eq. (18) requires the trace of the Jacobian matrix, which scales as $\mathcal{O}(d^2)$ because calculating each diagonal entry of the Jacobian requires a separate differentiation pass. This quadratic complexity becomes computationally prohibitive, especially for high dimensional action spaces. To maintain efficiency, we employ Hutchinson's trace estimator [8, 13], which provides an unbiased stochastic estimate of the divergence:

$$\nabla_x \cdot v_\theta(x) = \mathrm{Tr}\left(\frac{\partial v_\theta}{\partial x}\right) = \mathbb{E}_{p(\epsilon)}\left[\epsilon^\top \frac{\partial v_\theta}{\partial x}\epsilon\right], \tag{63}$$

where $\epsilon \in \mathbb{R}^d$ is a noise vector sampled from a distribution $p(\epsilon)$ with zero mean and unit variance. We use the Rademacher distribution as recommended by Hutchinson [13]. By leveraging modern automatic differentiation, this reduction to $\mathcal{O}(d)$ per-step complexity allows FMER to scale gracefully to complex high-dimensional environments, such as Humanoid.

B.7 Remark on action-space entropy

Note that the Flow Matching process evolves in the unbounded latent space x, while the final policy entropy refers to the bounded action space a. The initial entropy in the action space, H_0^a, differs from the latent Gaussian entropy H_0^x by a constant Jacobian term. The final policy entropy H_1^a is obtained by evolving this base value according to the divergence of the induced vector field:

$$H_0^a = H_0^x + \mathbb{E}_{x_0 \sim p_0}\left[\sum_{i=1}^{d} \log\left(1 - \tanh^2(x_{0,i})\right)\right], \tag{64}$$

$$H_1^a \approx H_0^a + \mathbb{E}_{t\sim\mathcal{U}[0,1], x_t \sim p_t(\cdot|s)}\left[\nabla_a \cdot v^a(a,t,s)\right]. \tag{65}$$

Since H_0^x and the Jacobian expectation in Eq. (64) depend only on the fixed base distribution p_0, H_0^a is constant w.r.t. the policy parameters θ. Therefore, it can be omitted from the optimization objective without affecting the gradient.

B.8 Limitations and future work

Despite its strong empirical performance, FMER has several limitations that warrant future investigation. First, as shown in Table 9, while FMER is more efficient than diffusion-based baselines, it remains slower than FM-based counterparts such as SAC-Flow. This gap can be attributed to two factors: FMER requires more ODE integration steps (10 vs. 4), and its closed-form entropy computation, while theoretically principled, demands more computational resources than the entropy surrogates employed by SAC-Flow. This presents a tradeoff: FMER's closed-form entropy control yields superior exploration quality, particularly in sparse-reward settings, but at a higher per-step computational cost. Second, while the Hutchinson estimator reduces the divergence computation from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$, it introduces variance into the entropy gradient estimates, which may affect convergence. We encountered no computational or memory issues in the evaluated action spaces ($d = 3$–17), but the cost of entropy estimation also depends on the vector-field architecture and batch size. Substantially higher-dimensional actions or long action chunks may increase this cost considerably. Evaluating and improving scalability in these settings remains an important direction for future work. Third, while our closed-form entropy derivation accounts for the $\tanh$ transformation, it assumes the flow operates in a factorized latent space with an isotropic Gaussian base distribution; extending FMER to structured or non-isotropic base distributions would require revisiting the entropy correction in Proposition 1. Finally, all experiments are conducted in simulation, and the sim-to-real gap for flow-based policies in contact-rich manipulation tasks remains an open question. We view these as natural directions for future work rather than fundamental barriers to the approach.

C Experiments

C.1 Baseline implementation

We use the following codebases for the baseline implementations. Hyperparameters are set according to the specifications in their respective papers; otherwise, the default values are used.

- PPO: `https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html`
- SAC: `https://stable-baselines3.readthedocs.io/en/master/modules/sac.html`

- TD3: `https://stable-baselines3.readthedocs.io/en/master/modules/td3.html`
- DIPO: `https://github.com/mahaitongdae/diffusion_policy_online_rl/tree/main`
- QVPO: `https://github.com/wadx2019/qvpo/tree/main/agent`
- DPMD: `https://github.com/mahaitongdae/diffusion_policy_online_rl/tree/main`
- SAC-Flow: `https://github.com/Elessar123/SAC-FLOW/tree/master/from_scratch_code`
- FLAME: `https://github.com/lzqw/FLAME`

Table 4: Hyperparameters used in the baselines. † 20 for training/sampling, 1 for test. *4 for critic update, 32 for evaluation, 64 for actor update. **Note:** SAC-Flow utilizes multiple sub-networks for the actor with different depths and activations.

Hyperparameters	**FLAME**	**DIPO**	**DPMD**	**QVPO**	**SAC-Flow**	**SAC**	**TD3**	**PPO**
No. of hidden layers	3	3	3	2	Multiple	2	2	2
No. of hidden nodes	256	256	256	256	Multiple	256	256	256
Activation	mish	mish	mish	mish	Multiple	ReLU	ReLU	ReLU
Batch size	256	256	256	256	512	256	256	256
Discount for reward γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Reward scale	0.2	0.2	0.2	1	1	1	1	1
Target smoothing coefficient	0.005	0.005	0.005	0.005	N/A	0.005	0.005	N/A
Learning rate for actor	3×10^{-4}, annealing to 5×10^{-5}			3×10^{-4}	3×10^{-4}	3×10^{-4}	1×10^{-3}	3×10^{-4}
Learning rate for critic	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	1×10^{-3}	3×10^{-4}	1×10^{-3}	3×10^{-4}
Diffusion steps	$20/1^{\dagger}$	100	20	20	4	N/A	N/A	N/A
Candidate set size	32	1	32	$4/32/64^{*}$	-	N/A	N/A	N/A

C.2 FMER implementation

Table 5: Hyperparameters used in FMER (shared across all environments).

Hyperparameters	**Value**
No. of hidden layers	3
No. of hidden nodes	256
Activation	mish
Batch size	256
Discount for reward γ	0.99
Target smoothing coefficient	0.005
Shared learning rate for actor	3×10^{-4}, annealing to 5×10^{-5}
Learning rate for critic	3×10^{-4}
Advantage weight initialized τ	0.5
ODE steps	10
Target ESS*	3
Target entropy $\bar{H}$	Annealed from 0 to $-dim(\mathcal{A})$
Candidate set size	8
Initial τ value	0.5
Initial α value	e
Optimizer	Adam [15]

C.3 Action generation trajectory in *multi-goal* environment

Figure 7 visualizes, for the same state $s = (0, 0)$, the action samples produced by each policy at an early (10K), intermediate (20K), and converged (45K) stage of training. SAC initially concentrates samples in a broad, unimodal region and only gradually shifts mass toward a high-value basin, reflecting the limited ability of a single Gaussian to represent disconnected modes. In contrast, both DPMD and FMER allocate probability mass to the high-value region much earlier, indicating faster discovery of promising modes. Diffusion denoising trajectories are often indirect and vary across samples, whereas FMER induces smoother and more coherent transport toward high-value actions.

C.4 FrankaKitchen environment setup

The environment details are available at `https://robotics.farama.org/envs/franka_kitchen/`.

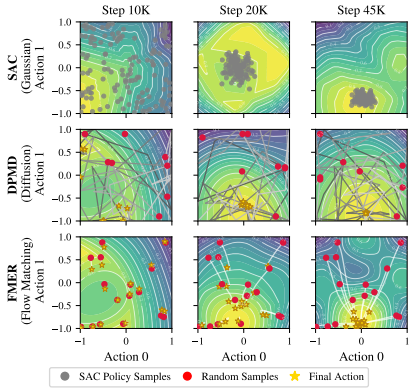


Figure 7: Policy evolution on the 2D multi-goal task for the same state $s = (0, 0)$. Background contours depict the learned critic $Q(s, a)$ over action space. For SAC, points show policy samples, and for the generative policies, red dots show the random noise samples and their sampling trajectory towards the final action.

Table 6: FrankaKitchen task description.

Tasks	**Description**
"top burner"	Turn the oven knob that activates the top burner
"bottom burner"	Turn the oven knob that activates the bottom burner
"light switch"	Turn on the light switch
"slide cabinet"	Open the slide cabinet
"hinge cabinet"	Open the left hinge cabinet
"microwave"	Open the microwave door
"kettle"	Move the kettle to the top left burner

Table 7: FrankaKitchen task compositions.

Tasks (N)	**Composition**
$N = 1$	"light switch"
$N = 2$	"light switch", "slide cabinet"
$N = 4$	"light switch", "slide cabinet", "bottom burner", "microwave"
$N = 7$	"light switch", "slide cabinet", "bottom burner", "microwave", "kettle", "top burner", "hinge cabinet"

Table 8: We report the mean number of accomplished tasks and the normalized task completion rate (%). Results reflect the maximum evaluation return achieved during the final 10% of training (steps 900K to 1M), averaged across 5 random seeds. Each evaluation point is computed over 10 independent environment episodes. Standard deviations are provided in parentheses.

	Accomplished Tasks				**Normalized Task Completion Rate (%)**			
Method	**N=1**	**N=2**	**N=4**	**N=7**	**N=1**	**N=2**	**N=4**	**N=7**
PPO	0.02 (0.04)	0.00 (0.00)	0.04 (0.09)	0.60 (0.55)	2.00 (4.47)	0.00 (0.00)	1.00 (2.24)	8.57 (7.82)
SAC	0.40 (0.55)	0.64 (0.50)	0.10 (0.14)	0.86 (0.30)	40.00 (54.77)	32.00 (24.90)	2.50 (3.54)	12.29 (4.24)
TD3	0.00 (0.00)	0.22 (0.49)	0.60 (0.69)	0.40 (0.55)	0.00 (0.00)	11.00 (24.60)	15.00 (17.32)	5.71 (7.82)
DIPO	0.00 (0.00)	0.66 (0.61)	0.58 (0.86)	1.62 (1.29)	0.00 (0.00)	33.00 (30.33)	14.50 (21.39)	23.14 (18.36)
DPMD	0.60 (0.55)	**1.00 (0.00)**	1.00 (0.71)	1.44 (0.61)	60.00 (54.77)	**50.00 (0.00)**	25.00 (17.68)	20.57 (8.67)
QVPO	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	1.32 (0.66)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	18.86 (9.44)
SAC-Flow	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.04 (0.09)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.57 (1.28)
FLAME	0.08 (0.18)	0.20 (0.45)	0.26 (0.58)	1.28 (0.54)	8.00 (17.89)	10.00 (22.36)	6.50 (14.53)	18.29 (7.79)
FPO	0.00 (0.00)	0.02 (0.04)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	1.00 (2.24)	0.00 (0.00)	0.00 (0.00)
FMER	**0.80 (0.45)**	0.96 (0.09)	**1.86 (0.15)**	**2.82 (1.47)**	**80.00 (44.72)**	48.00 (4.47)	**46.50 (3.79)**	**40.29 (21.03)**

C.5 FrankaKitchen performance

We observe several notable performance trends among the baselines:

- Task completion versus normalized task completion rate: Across most baselines, as the number of tasks N increases, the absolute number of completed tasks rises, whereas the overall normalized task

completion rate declines. The number of completed tasks rises because the number of total tasks rises. This suggests that agents readily exploit isolated, simpler sub-tasks for initial rewards. However, the fixed episode budget of 280 steps imposes a bottleneck on sequential execution, leaving insufficient time to complete subsequent sub-tasks.

- SAC exhibits a non-monotonic pattern across task configurations, with a sharp drop at $N = 4$ despite $N = 4$ being a superset of the $N = 2$ tasks. We hypothesize this reflects the limitation of Gaussian modelling. The Gaussian policy's limited expressiveness becomes particularly damaging at $N = 4$, where optimal behavior requires committing to one of several distinct sequential task orderings. A unimodal policy attempts to average over these modes and commits to none. The recovery at $N = 7$ is consistent with the partial-reward accumulation effect noted above: with eight available sub-tasks, even a poorly committed unimodal policy can accumulate rewards by accidentally completing simpler sub-tasks encountered during exploration.
- DPMD susceptibility to local optima: DPMD achieves the highest performance among baselines for low N. Its KL-divergence constraint acts as a trust region that quickly locks onto a single successful task early in training; however, as N increases, this rigidity hinders exploration and traps the policy in local optima.
- FPO performed substantially worse than FMER. A key distinction is the training regime: FPO is on-policy and updates only from recently collected rollout batches, whereas FMER repeatedly reuses replay-buffer transitions with a bootstrapped critic. Under our fixed 1M-step interaction budget, this gives FMER more policy and critic updates per environment transition. This gap is further amplified on FrankaKitchen by the sparse reward structure: successful transitions are rare, and FMER can repeatedly reuse them via its replay buffer.

Overall, these findings underscore the necessity of multimodal policy expressiveness and demonstrate the importance of principled exploration control and policy mirror descent.

C.6 MuJoCo training curves and dynamics

Training curves. As shown in Figure 8, FMER has a slower convergence rate than other baselines. This is because the Lagrangian multiplier α enforces an entropy constraint to promote early exploration and prevent premature mode collapse. While this sustained exploration enables the discovery of higher-reward solutions in complex spaces, it comes at the cost of more gradual convergence. On Hopper, it also causes performance to peak and then degrade in simpler, strictly unimodal environments.

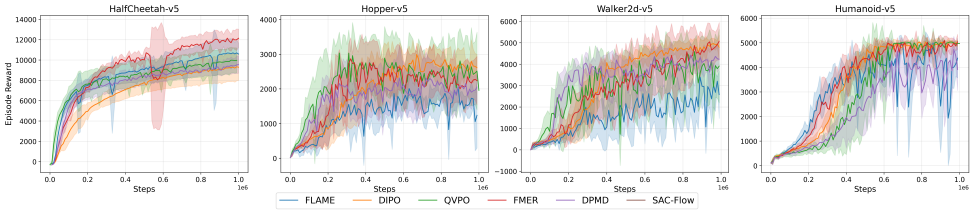
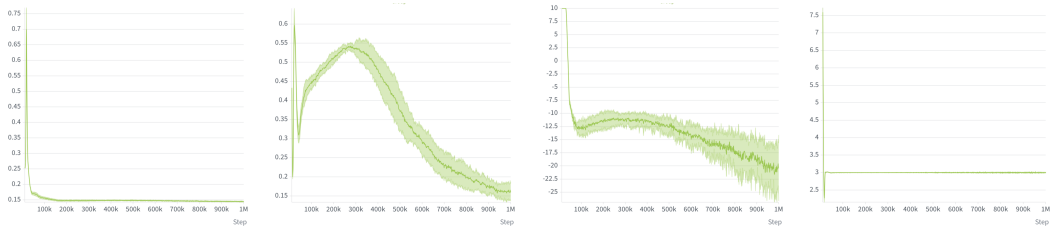


Figure 8: Training curves on four MuJoCo locomotion benchmarks. We report the average episodic return over 5 random seeds, with solid lines representing the mean and shaded regions indicating the standard deviation. For visual clarity, we compare FMER against the five top-performing baselines (QVPO, DPMD, DIPO, FLAME, SAC-Flow). Curves are smoothed with a 10,000-step window.



(a) Weighted-CFM loss (b) Temperature τ (c) Policy entropy (d) Effective Sample Size

Figure 9: Training dynamics of FMER on Humanoid-v5, averaged over five seeds; shaded regions denote one standard deviation.

Training dynamics. Figure 9 reports the weighted-CFM loss, temperature τ, policy entropy, and ESS over training on Humanoid-v5, averaged over five seeds. The brief transient at the start reflects the untrained critic: all candidates initially receive similar values, so the weights are uniform and ESS is close to M; once the critic begins to discriminate, ESS undershoots and the controller raises τ to restore the target within a few thousand steps. Thereafter, the weighted-CFM loss decreases monotonically and plateaus early with negligible variance across seeds, indicating stable optimization of the surrogate objective, and the ESS remains at its target ($\mathrm{ESS}^* = 3$) for the rest of training, confirming the effectiveness of adaptive τ. Policy entropy drops quickly from its initial near-random level, plateaus while the entropy constraint is binding, and then decreases gradually as the target entropy $\bar{H}$ is annealed. The final entropy remains finite (a deterministic policy would have differential entropy $-\infty$), confirming that the policy stays stochastic while exploration is progressively reduced.

C.7 Training and inference time

Table 9 reports total training time and inference time on Humanoid-v5 for FMER and all diffusion- and FM-based baselines, with all models trained on a single NVIDIA V100 GPU with 5 CPU cores (25GB system RAM). For QVPO, we report the training time using the official PyTorch implementation. While a JAX port is available in the DPMD repository, our profiling revealed it to be less optimized, resulting in longer training times than the official PyTorch version.

Table 9: Total training time and inference time of 1 million environment interactions on Humanoid-v5.

Algorithm	Training time	Inference time
DIPO (JAX)	294 min	2.268 ms
DPMD (JAX)	276 min	1.125 ms
QVPO (PyTorch)	19.7 h	1.284 ms
SAC-Flow (JAX)	**126 min**	0.483 ms
FMER (JAX)	215 min	0.515 ms

Table 10: Training time (minutes) and inference time (ms) across environments with varying action dimensions (d), averaged over 5 random seeds.

Environment	**Action dimension d**	**Training time**	**Inference time**
Hopper-v5	3	167	0.471
HalfCheetah-v5	6	168	0.479
Walker2d-v5	6	171	0.480
Humanoid-v5	17	215	0.515

As shown in Table 9, FMER achieves substantially lower training time than diffusion-based methods, attributable to its smaller candidate action set and fewer ODE integration steps. SAC-Flow attains the shortest training time overall by further reducing ODE steps and replacing closed-form entropy with a surrogate estimate; however, as shown in Section 5.2, this approximation comes at the cost of exploration quality in sparse-reward settings.

We evaluate the impact of action dimensionality on training time in Table 10. As expected, training time scales with d due to the $\mathcal{O}(d)$ complexity of the Hutchinson estimator (Section 4.3).

D Ablation study

D.1 Entropy and advantage-weighting ablations on FrankaKitchen

To evaluate the roles of entropy regularization and advantage weighting in sparse-reward settings, we repeat the corresponding Walker2d ablations on FrankaKitchen with $N = 7$. All variants retain the default best-of-M action selection for environment interaction and critic-target computation. Results are reported as mean (standard deviation) across five random seeds.

Entropy regularization. We deactivate $\mathcal{L}_{\mathrm{ent}}$ either from the beginning of training or after 5×10^5 environment steps. For removal from the beginning, we also evaluate a variant that clips τ from below to prevent numerical instability.

As shown in Table 11, removing entropy regularization from the start leads to failure on this sparse-reward task. During training, ESS remains above its target, driving τ toward zero. Clipping τ prevents numerical instability but does not restore performance, and the final entropy remains substantially below that of default FMER. In

Table 11: Entropy ablation on FrankaKitchen ($N = 7$). Task completion rates are reported in percentages. NaN denotes a non-finite entropy estimate.

Setting	Task completion rate (%)	Final entropy
Default FMER	40.29 (21.03)	-8.71 (1.06)
Entropy removed after 5×10^5 steps	40.29 (19.73)	-10.43 (2.36)
Entropy removed from the start	0.00 (0.00)	NaN
Entropy removed from the start, τ clipped	0.00 (0.00)	-201.36 (11.91)

contrast, removing the constraint after 5×10^5 steps leaves the mean task completion rate unchanged despite a decrease in final entropy. These results are consistent with the Walker2d ablation and support the importance of entropy regularization during early exploration.

Advantage weighting. We replace the normalized exponential advantage weights with uniform weights, $\hat{w}_j(s) = 1/M$, while retaining entropy regularization and all other components. This reduces the actor's weighted-CFM term to unweighted conditional flow matching over the sampled candidate actions.

Table 12: Advantage-weighting ablation on FrankaKitchen ($N = 7$).

Setting	Task completion rate (%)
Default FMER	40.29 (21.03)
Uniform weights	33.71 (18.23)

Uniform weighting yields a lower mean task completion rate than default FMER (Table 12), consistent with the benefit of emphasizing higher-value candidate actions during policy fitting.

D.2 Ablation on $\tanh$ correction in entropy derivation

A natural alternative to our $\tanh$ correction formulation is to replace the $\tanh$ transformation with simple clipping and compute the entropy directly using Theorem 2. We first explain why clipping is theoretically incompatible with our framework, then provide empirical validation.

Our framework requires two properties that clipping violates:

- Non-invertibility: Theorem 1 requires mapping actions back to latent space via $x_1 = \text{arctanh}(a)$. Clipping is a many-to-one operation that maps all out-of-bound values to the boundary ± 1, making the inversion undefined and collapsing the latent representation.
- Non-differentiability: The entropy correction in Proposition 1 accounts for the probability density distortion induced by the latent-to-action transformation (Appendix A.3). This requires computing the derivative of the transformation, which is undefined at the clipping boundaries (± 1) with clipping operation.

Despite this incompatibility, we ran ablation experiments on MuJoCo to empirically compare $\tanh$ against clipping. In the clipping setup, we bypass the invertible latent mapping and use the environment actions directly as the flow targets during training ($x_1 = a$). Because no $\tanh$ transformation is applied, we omit the density correction from Proposition 1 and estimate the uncorrected entropy using the standard vector field divergence (Eq. (16)). During inference, the ODE is integrated to produce an unbounded terminal state x_1, which is then hard-clipped to ensure valid environment actions: $a = \text{clip}(x_1, -1, 1)$.

Table 13: Comparison of clipping and tanh transformations across different environments. Values represent mean returns with standard deviations in parentheses.

Method	**Hopper**	**HalfCheetah**	**Walker**	**Humanoid**
clipping	2704.55 (627.12)	7701.50 (598.25)	4733.82 (696.37)	4378.76 (2021.24)
$\tanh$	**2913.45 (642.67)**	**12332.38 (903.67)**	**5282.89 (437.57)**	**5286.12 (65.35)**

As shown in the Table 13, $\tanh$ consistently outperforms clipping across environments, validating both our architectural choice and the necessity of the closed-form entropy correction in Proposition 1.

D.3 Ablation on divergence computation: Hutchinson vs. Exact Jacobian

Table 14: Comparison of Hutchinson's estimator and exact trace computation on Walker2d-v5.

Method	Performance	Time (min)
Hutchinson	5282.89 (437.57)	171
Exact	5316.47 (254.97)	258

To evaluate the impact of the Hutchinson trace estimator, we compare it against an exact Jacobian trace computation on Walker2d-v5 ($d = 6$). As shown in Table 14, while the exact calculation reduces the standard deviation across seeds, the mean performance gain is limited, and the exact computation increases training time by nearly 50% even in this low-dimensional action space. Following [8], we adopt the unbiased Hutchinson estimator as it provides the necessary scalability with limited impact on final policy performance.

D.4 Ablation on adaptive τ

Table 15 ablates fixed versus dynamically tuned τ. No single fixed τ performs consistently and the best fixed value varies across benchmarks. Dynamic tuning achieves competitive performance across all environments without environment-specific tuning, making it a more practical default despite not uniformly dominating any single fixed τ.

Table 15: Sensitivity of FMER to temperature τ: fixed values versus ESS-based dynamic tuning.

	FrankaKitchen (finished task)				MuJoCo-v5			
Method	**N=1**	**N=2**	**N=4**	**N=7**	**Hopper**	**HalfCheetah**	**Walker**	**Humanoid**
$\tau = 0.2$	0.20 (0.45)	0.98 (0.67)	1.00 (0.67)	**3.10 (1.21)**	2927.44 (531.97)	11632.94 (859.00)	5025.65 (350.93)	**5399.06 (152.65)**
$\tau = 0.5$	0.60 (0.55)	**1.42 (0.39)**	1.36 (0.41)	2.84 (1.44)	2972.15 (344.36)	10888.09 (1873.16)	**5504.34 (144.18)**	5310.64 (87.56)
$\tau = 1.0$	0.00 (0.00)	0.42 (0.58)	0.56 (0.88)	1.82 (1.10)	**3222.88 (289.04)**	10920.01 (2624.63)	5430.12 (293.93)	5228.06 (148.65)
Dynamic	**0.80 (0.45)**	0.96 (0.09)	**1.86 (0.15)**	2.82 (1.47)	2913.45 (642.67)	**12332.38 (903.67)**	5282.89 (437.57)	5286.12 (65.35)

D.5 Ablation on candidate size

Table 16 gives performance difference with different candidate size. Training time increases with candidate size (more forward/backward passes). For dense tasks (Walker2d, Humanoid), smaller sets ($M = 4, 8$) suffice as the value function provides accurate guidance. Sparse tasks (FrankaKitchen) benefit from larger sets ($M = 16$) for broader exploration. Excessively large sets ($M = 32$) introduce value estimation noise, degrading performance universally. $M = 8$ is a robust default setting.

Table 16: Performance and training time (minutes) across candidate sizes M and environments (5 random seeds).

	Walker2d-v5		Humanoid-v5		Franka (N=2)		Franka (N=7)	
Candidate Size	Time	Perf	Time	Perf	Time	Perf	Time	Perf
4	151	5321.36	167	5243.07	192	1.00	193	2.50
8	171	5282.89	215	5286.12	220	0.96	217	2.82
16	225	4889.52	285	5129.35	275	1.40	274	2.90
32	320	4657.87	422	5048.57	369	1.24	370	1.96

D.6 Ablation on ODE steps and solvers

We investigate the impact of ODE integration steps and solver choices on model performance and training efficiency.

As shown in Table 17, performance remains stable across $N \in [5, 20]$. This range requires fewer steps than most diffusion baselines, maintaining a computational advantage (see Table 9). These results confirm that the straight paths conditioned on Optimal Transport require less precise integration compared to the curved trajectories typical of standard diffusion models.

Table 18 demonstrates that higher-order solvers increase the Number of Function Evaluations (NFE) per step while offering limited performance improvements. Applying a 4th-order Runge–Kutta (RK4) solver with $N = 2$ underperforms significantly, despite utilizing a similar NFE to the Euler method with $N = 10$. This suggests that too few integration steps cause insufficient trajectory coverage regardless of solver order. Euler remains the most efficient default.

Table 17: Training time (minutes) and performance on Walker2d across ODE integration steps (N), averaged over 5 random seeds. Performance is presented as Mean (Std).

N	Time	Perf
5	147	5493.64 (282.53)
10	171	5282.89 (437.57)
20	226	5591.64 (153.13)
50	409	5186.86 (519.63)

Table 18: Training time (minutes) and performance on Walker2d across ODE solvers, averaged over 5 random seeds. Performance is presented as Mean (Std).

ODE solver	ODE steps	NFE	Time	Perf
Euler	10	10	171	5282.89 (437.57)
Runge–Kutta 2	10	20	219	5519.10 (395.64)
Runge–Kutta 4	10	40	340	5486.24 (184.02)
Runge–Kutta 2	5	10	168	5234.53 (172.47)
Runge–Kutta 4	2	8	164	3988.16 (2367.62)

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in abstract and introduction are supported by the proofs in Appendix A and experiments in Section 5.

   Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please see Appendix B.8.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The proof of each theorem and proposition is provided in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The algorithm is described in Algorithm 1 and codes are provided in an anonymous GitHub for reviewers. The GitHub will be made public available upon acceptance.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: This paper has described where to find each environment used in the experiment online. Codes are provided in an anonymous GitHub for reviewers. The GitHub will be made public available upon acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Please see Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Experiments are executed with 5 random seeds. For each random seed, 10 random runs are used for evaluation. Results are reported with standard deviation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please see Appendix C.7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: Our research adheres to the code of ethics by using only synthetic, non-personal data from a simulated environment, posing no risk to human subjects, privacy, or societal harm, and we commit to open-sourcing our code and models with proper documentation to ensure reproducibility and responsible use.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: This paper presents work whose goal is to advance the field of reinforcement learning. Experiments are conducted on robot controlling in well-controlled simple tasks in simulated environments.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not pose any such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the environment used in the experiments in Section 5.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The codebase and implementation details are shared alongside the paper.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve experiments with human participants.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve experiments with human participants.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.