SNARL: SELF-NORMALIZING ANCHORED MAXIMUM-ENTROPY REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Expressive generative policies can capture complex action distributions, but maximum-entropy (MaxEnt) reinforcement learning requires evaluating the policy log-density $\log \pi_\theta(a \mid s)$, which is generally intractable for implicit generators. We introduce Self-Normalizing Anchored MaxEnt Reinforcement Learning (SNARL), an off-policy framework that learns a separate state-conditioned energy model to estimate the policy's negative log-density from samples. SNARL augments spatial and temporal score matching with an anchor derived from the Gaussian limit of variance-preserving corruption, approximately normalizing the energy model as the policy evolves. The resulting model estimates the policy's negative log-density in a single forward pass. By separating policy log-density estimation from action generation, SNARL enables maximum-entropy learning with a one-step pushforward actor that generates each action in a single network evaluation. Experiments show that SNARL is competitive with strong diffusion and flow baselines across high-dimensional continuous-control benchmarks, excels on MuJoCo Playground humanoid locomotion tasks, and provides approximately $14\times$ faster policy inference than DIME.

1 INTRODUCTION

Expressive distribution families have become the prevalent policy representation for continuous control. Approaches based on diffusion models and flow matching (Janner et al., 2022; Wang et al., 2023; Lipman et al., 2023; Ding et al., 2024) can represent multimodal action distributions, which a unimodal Gaussian cannot. Recent advances in robot learning, for instance, leverage flow-based policies to carry out long-horizon manipulation in unstructured environments (Chi et al., 2025; Black et al., 2025b;a).

Maximum-entropy reinforcement learning (MaxEnt RL) underlies some of the most sample-efficient off-policy algorithms for continuous control, such as Soft Actor–Critic (SAC) (Haarnoja et al., 2018). The MaxEnt RL objective augments the task reward with the discounted cumulative entropy of the policy over future states, which improves exploration, smoothens the optimization landscape, and provably solves a class of robust reinforcement learning (RL) problems (Fox et al., 2016; Haarnoja et al., 2017; 2018; Eysenbach & Levine, 2022). To estimate the entropy, SAC evaluates the policy's log-density at sampled actions, which it achieves by representing the policy as a reparametrized squashed Gaussian. For implicit generative policies, evaluating the log-density of sampled actions can be intractable or computationally expensive, making the entropy term difficult to compute during MaxEnt training.

For these implicit generative policies, two challenges arise. First, MaxEnt updates require policy log-density estimates as the conditional action distribution changes during learning. The second challenge is deployment cost: diffusion- and flow-based policies can require many network function evaluations (NFEs) per action, and these repeated evaluations increase inference latency.

Existing work on the first challenge forgoes exact MaxEnt RL in favor of alternative optimization objectives or tractable surrogates of the entropy estimate. For example, DIME and AMDP derive lower bounds on the policy entropy (Celik et al., 2025; Thilges et al., 2026), while SAC Flow uses the entropy of the entire action generation process as a proxy for the policy entropy (Zhang et al., 2026). MaxEntDP estimates policy log-density by numerically integrating the diffusion model's noise prediction errors across noise levels (Dong et al., 2025). To address the second challenge,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

MFPO (Dong et al., 2026) reduces NFE by learning the average velocity fields with MeanFlow (Geng et al., 2026), enabling low-NFE action generation. Outside MaxEnt RL, consistency policies (Ding & Jin, 2024; Chen et al., 2023) and flow policies (Chen et al., 2025) have also achieved 1-NFE action generation.

In this work, we propose a method for jointly training a generic reparametrized policy and an energy model approximating its negative log-density. The method mitigates the first challenge by providing policy entropy estimates through energy-model evaluations at sampled actions. It also mitigates the second challenge by facilitating the training of an expressive 1-NFE policy model approximately consistent with the MaxEnt RL objective.

Key to training a reliable log-density model is keeping it approximately normalized, but this has been elusive (Hyvärinen, 2005; Gutmann & Hyvärinen, 2012; Andreas et al., 2015). Our approach is inspired by dual score matching (DSM) (Guth et al., 2025), which learns an energy function to estimate the negative log-density of a distribution from its samples via score matching in both sample space and diffusion time. For a fixed target distribution, these objectives identify the energy up to a global additive constant, which can subsequently be estimated and subtracted. This scheme, however, cannot be directly applied to the state-conditioned policy in RL, where the policy must already be usable during training without state-dependent bias.

We introduce Self-Normalizing Anchored MaxEnt Reinforcement Learning (SNARL), which applies space and time score matching to conditional policy densities under variance-preserving (VP) corruption (Guth et al., 2025; Yu et al., 2025). The VP corruption provides an anchor point: at high noise levels, the corrupted policy distribution approaches a standard Gaussian. Adding this anchor to the score-matching objective continually adjusts the model's state-dependent energy level toward normalization. The resulting energy network then provides an approximate negative log-density estimate for SAC-like critic, actor, and temperature updates. Crucially, the energy model is trained from current-policy samples and calibrated online through Gaussian anchoring, without requiring likelihood computation along the action generative path. This decoupling allows us to train a policy that generates actions in a single network evaluation, accelerating the control loop.

To summarize, SNARL offers three advantages:

- **Policy Log-density estimation from samples.** A separate energy network learns the policy's negative log-density from samples and provides estimates in a single forward pass for MaxEnt actor-critic updates.
- **Online energy normalization.** A Gaussian anchor loss, jointly optimized with space and time score matching under VP corruption, approximately normalizes state-conditioned energies during policy learning.
- **Efficient one-step control.** With a 1-NFE pushforward actor, SNARL achieves competitive final returns and outperforms all evaluated baselines on all four MuJoCo Playground humanoid locomotion tasks. Action inference is approximately $14\times$ faster than DIME in our batch-size-one GPU benchmark.

2 Related work

MaxEnt RL with Tractable Policy Densities. Soft actor-critic (SAC) commonly uses a tanh-squashed diagonal-covariance Gaussian policy (Haarnoja et al., 2018). Normalizing flows expand the policy class through invertible transformations while retaining tractable log-densities (Ward et al., 2019; Mazoure et al., 2020). Within this family, MEow jointly parametrizes the policy and soft Q-function, making both policy log-densities and soft values tractable (Chao et al., 2024).

MaxEnt RL with Intractable Policy Densities. For diffusion and flow policies, existing methods derive tractable entropy bounds or introduce alternative regularizers based on the action sampling process. DIME (Celik et al., 2025) derives a variational lower bound on policy entropy from the noising and denoising transition densities. This bound enters the Bellman backup, while policy improvement minimizes a reverse KL objective over diffusion paths. AMDP (Thilges et al., 2026) derives a continuous-time lower bound on policy entropy from the KL divergence between its diffusion process and a reference process. It uses reciprocal adjoint matching with regression targets

derived from critic gradients, avoiding backpropagation through the full diffusion trajectory. SAC Flow (Zhang et al., 2026) uses the total log-density of a noise-augmented generative flow with Gaussian transitions for entropy regularization in actor and critic updates. FLAC (Lv et al., 2026) replaces explicit policy entropy regularization with a penalty on integrated squared velocity to control deviation from a fixed high-entropy reference action distribution. SQL (Haarnoja et al., 2017) uses amortized Stein variational gradient descent to approximate Boltzmann policies induced by the soft Q-function, but its importance-sampling estimates of soft values can scale poorly with action dimension. S2AC (Messaoud et al., 2024) instead tracks changes in sample log-density using a small-step approximation to the Jacobian log-determinants of the sampling updates. MFPO (Dong et al., 2026) uses derivatives of the actor's velocity field to train an auxiliary network that approximates the integral governing log-density changes during few-step MeanFlow sampling (Geng et al., 2026).

Score-based Log-Density Estimation. Score-matching method learns log-density gradients from samples without evaluating a normalization constant, leaving the additive level of an associated energy function undetermined (Hyvärinen, 2005; Vincent, 2011). Choi et al. (2022) recover density ratios by integrating the time derivatives of log-density along a probability path connecting the two distributions. Yu et al. (2025) learn time scores by regression through conditional time-score matching, then estimate log-density by numerically integrating these scores over time and adding a known reference log-density. Dual Score Matching (DSM) (Guth et al., 2025) instead learns an energy function through both space- and time-score matching and approximately normalizes the energy model after training by adjusting the remaining global offset. DSM is not directly applicable to RL policies for two reasons: (1) a state-conditioned log-density model requires a more complicated state-dependent energy normalization; and (2) this normalization cannot be deferred and must instead concur with the reinforcement learning process, for the energy model to be usable in soft Bellman backups. SNARL applies DSM to a VP noising process to allow adding an anchor loss that keeps the energy model approximately normalized in all states during training.

3 Preliminaries

We consider a discounted Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma, d_0)$ with continuous state space $\mathcal{S}$ and bounded d-dimensional action space $\mathcal{A} \subseteq \mathbb{R}^d$. Here, p denotes the transition dynamics, r a bounded reward function, $\gamma \in [0, 1)$ the discount factor, and d_0 the initial state distribution. We consider a reparametrized stochastic policy $a = f_\theta(s, \varepsilon)$ with $\varepsilon \sim \mathcal{N}(0, I)$ and assume that it admits a density $\pi_\theta(a \mid s)$. We use d_{π_θ} for its unnormalized discounted state–action occupancy measure

$$d_{\pi_\theta}(s, a) = \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s, a_t = a). \tag{1}$$

Maximum Entropy Framework. MaxEnt RL augments discounted return with policy entropy (Fox et al., 2016; Haarnoja et al., 2017; 2018):

$$J_{\mathrm{MaxEnt}}(\pi_\theta) = \mathbb{E}_{(s,a)\sim d_{\pi_\theta}}\big[r(s, a) + \tau \mathcal{H}(\pi_\theta(\cdot \mid s))\big], \tag{2}$$

where $\mathcal{H}(\pi_\theta(\cdot \mid s)) = -\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[\log \pi_\theta(a \mid s)]$ is the policy entropy and the temperature $\tau > 0$ controls its tradeoff with the reward. Policy evaluation uses the soft Bellman recursion

$$Q^{\pi_\theta}(s, a) = r(s, a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)}\Big[\mathbb{E}_{a'\sim\pi_\theta(\cdot|s')}\big[Q^{\pi_\theta}(s', a') - \tau \log \pi_\theta(a' \mid s')\big]\Big]. \tag{3}$$

For fixed Q and τ, policy improvement seeks to maximize at each state the objective

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\big[Q(s, a) - \tau \log \pi_\theta(a \mid s)\big]. \tag{4}$$

Both updates require evaluating the policy log-density $\log \pi_\theta(a \mid s)$. We consider policies that can be sampled but whose log-density may be intractable, motivating its estimation from policy samples.

Dual Score Matching. DSM learns an energy function to estimate negative log-density from samples through space- and time-score matching (Guth et al., 2025). For a fixed noise level t, an energy function $U_\psi(\cdot, t)$ is normalized when $Z_\psi(t) = \int_{\mathbb{R}^d} \exp[-U_\psi(x, t)]\,\mathrm{d}x = 1$, and approximately normalized when $\log Z_\psi(t)$ is close to zero. Provided $0 < Z_\psi(t) < \infty$, adding $\log Z_\psi(t)$ to

$U_\psi(x,t)$ yields a normalized energy. Normalization alone does not ensure that the energy accurately estimates the target negative log-density. For a fixed target density p_0, let $x \sim p_0$ and $z \sim \mathcal{N}(0, I)$ be independent. Variance-exploding (VE) corruption produces $b = x + \sqrt{t}\, z$, whose density is denoted by p_t. DSM trains an energy network $U_\psi(b,t)$, with the goal of approximating $-\log p_t(b)$, using two regression objectives at each noise level $t > 0$:

$$\begin{aligned} \ell_{\text{space}}(\psi, t) &= \mathbb{E}_{x,z}\left[\left\|\nabla_b U_\psi(b,t) - \frac{z}{\sqrt{t}}\right\|^2\right], \\ \ell_{\text{time}}(\psi, t) &= \mathbb{E}_{x,z}\left[\left(\partial_t U_\psi(b,t) - \frac{d}{2t} + \frac{\|z\|^2}{2t}\right)^2\right]. \end{aligned} \tag{5}$$

The regression targets for the space gradient and time derivative of U_ψ have conditional means $-\nabla_b \log p_t(b)$ and $-\partial_t \log p_t(b)$, respectively, which are the negatives of the process's space and time *scores*. Intuitively, space-score matching learns the energy's landscape at each noise level, while time-score matching learns how energy values change with the noise level. With exact matching, $U_\psi(b,t)$ and the negative log-density $-\log p_t(b)$ have the same space gradients and time derivatives and thus differ only by a global constant c. DSM estimates this constant after training using a Gaussian approximation at a high noise level, and subtracts it to approximately normalize the energy function. The resulting network estimates the negative log-density $-\log p_0(x)$ with a single forward pass of the approximately normalized $U_\psi(x, t_{\min})$ at a low noise level $t_{\min} \approx 0$. For conditional densities $p_t(b \mid s)$, as in RL, exact matching instead leaves a state-dependent offset $c(s)$ that affects the entropy term in soft Bellman backups. This motivates a training objective that jointly learns the policy's distributional shape and approximately normalizes the state-conditioned energy model as the policy evolves.

4 Maximum entropy RL with a self-normalizing energy

In this section, we apply space- and time-score matching under variance-preserving (VP) corruption to log-density estimation of a generic reparametrized policy, and introduce an anchor loss to encourage an approximately normalized energy function at each state during training. We then use the resulting estimates in the critic, actor, and temperature updates of SNARL. By decoupling log-density estimation from action generation, SNARL supports 1-NFE pushforward actors without requiring them to provide tractable log-densities.

4.1 Learning a Self-Normalizing Energy by Endpoint Anchoring

The space gradient and time derivative identities below apply denoising score matching (Vincent, 2011) and conditional time-score matching (Yu et al., 2025) to the policy density at each fixed state. We state them together with the Gaussian endpoint used by the novel anchor loss (12).

Lemma 1 (informal)**.** *For a fixed state s and noise level $t \in (0,1)$, draw $a \sim \pi_\theta(\cdot \mid s)$ and $z \sim \mathcal{N}(0, I)$ independently. We denote the corrupted action at noise level t by $a^{(t)}$, defined under VP corruption as*

$$a^{(t)} = \sqrt{1-t}\, a + \sqrt{t}\, z, \qquad t \in (0,1). \tag{6}$$

Let $p_t(\cdot \mid s)$ denote the conditional density of $a^{(t)}$ and $U(s,x,t) = -\log p_t(x \mid s)$ its normalized energy at an action coordinate $x \in \mathbb{R}^d$. The derivatives of U with respect to the action input and noise level, evaluated at $x = a^{(t)}$, satisfy

$$(\nabla_x U)(s, a^{(t)}, t) = \mathbb{E}\left[\frac{z}{\sqrt{t}} \,\middle|\, s, a^{(t)}\right], \tag{7}$$

$$(\partial_t U)(s, a^{(t)}, t) = \mathbb{E}\left[\frac{d}{2t} - \frac{\|z\|^2}{2t} + \frac{\langle z,\, a\rangle}{2\sqrt{t}\sqrt{1-t}} \,\middle|\, s, a^{(t)}\right]. \tag{8}$$

These expectations condition the joint sampling process in (6) on the corrupted action. As $t \to 1$, the corrupted action distribution approaches a standard Gaussian, giving the normalized energy limit at each fixed $x \in \mathbb{R}^d$:

$$\lim_{t \to 1} U(s,x,t) = \frac{\|x\|^2}{2} + \frac{d}{2}\log(2\pi). \tag{9}$$

The formal statement and proof are provided in Appendix A.1.

The identities in (7) and (8) express the energy derivatives as conditional means of targets computable from sampled actions and Gaussian noise. We use these targets to train U_ψ through space- and time-score matching. Both score losses are invariant to adding a state-dependent offset $c(s)$ to the energy, which renders it unnormalized. We introduce an anchor loss that regresses energy values toward the known negative log-density of the Gaussian reference in (9). We call this joint training self-normalization: it encourages approximate normalization without explicitly computing a normalization constant or applying a separate post hoc correction.

All three losses below use states $s \sim \mathcal{D}$ sampled from the replay buffer of a reinforcement learning algorithm. For the score losses, at each state we independently draw an action $a \sim \pi_\theta(\cdot \mid s)$ from the current policy, Gaussian noise $z \sim \mathcal{N}(0, I)$, and a noise level $t \sim p_T$, then construct $a^{(t)}$ using (6). Here, $p_T(t) = 1/(t \log(t_{\max}/t_{\min}))$ is the log-uniform density on $[t_{\min}, t_{\max}]$, where $t_{\min} \approx 0$ and $t_{\max} \approx 1$, avoiding the $t = 0$ and $t = 1$ extremes where the scores diverge. The three losses are

$$\mathcal{L}_{\text{space}}(\psi) = \mathbb{E}_{s,a,z,t}\left[\frac{t}{d}\left\|(\nabla_x U_\psi)(s, a^{(t)}, t) - \frac{z}{\sqrt{t}}\right\|^2\right], \tag{10}$$

$$\mathcal{L}_{\text{time}}(\psi) = \mathbb{E}_{s,a,z,t}\left[\left(\frac{t}{d}\right)^2\left((\partial_t U_\psi)(s, a^{(t)}, t) - \frac{d}{2t} + \frac{\|z\|^2}{2t} - \frac{\langle z,\, a\rangle}{2\sqrt{t}\sqrt{1-t}}\right)^2\right], \tag{11}$$

$$\mathcal{L}_{\text{anchor}}(\psi) = \mathbb{E}_{s,z}\left[\frac{1}{d^2}\left(U_\psi(s, z, t_{\max}) - \frac{\|z\|^2}{2} - \frac{d}{2}\log(2\pi)\right)^2\right]. \tag{12}$$

Here, $(\partial_t U_\psi)(s, a^{(t)}, t)$ denotes the partial derivative with respect to the noise level, holding s and the action input fixed. The weights t/d and $(t/d)^2$ follow DSM (Guth et al., 2025). For the anchor, as a heuristic, we divide the energy error by the action dimension before squaring, yielding the factor $1/d^2$. Preliminary experiments found no benefit in tuning these coefficients for the three losses. We therefore minimize their sum:

$$\mathcal{L}(\psi) = \mathcal{L}_{\text{space}}(\psi) + \mathcal{L}_{\text{time}}(\psi) + \mathcal{L}_{\text{anchor}}(\psi). \tag{13}$$

Policy parameters are held fixed during energy updates. The Gaussian reference is therefore an approximation at $t_{\max}$. Appendix A.2 derives the score and anchor objectives and Appendix A.3 analyzes the resulting normalization error.

4.2 SNARL for Maximum-Entropy Actor–Critic Learning

We combine the energy network from Section 4.1 with a reparametrized actor $a = f_\theta(s, \varepsilon)$ and a critic $Q_\phi(s, a)$. We use $U_\psi(s, a, t_{\min})$ as an estimate of $-\log \pi_\theta(a \mid s)$ in the critic, actor, and temperature updates described below.

Energy Update. To track the current policy's negative log-density, we train the energy network using fresh policy actions at replayed states. We include both states s and next states s' from each replay minibatch because the actor and temperature updates evaluate the energy at s, whereas the critic evaluates it at s'. At these states, we construct the three losses described in Section 4.1 and minimize $\mathcal{L}(\psi)$ in (13) while holding the policy fixed. Appendix B.2 describes the energy network architecture.

Critic Update. We present the critic update in scalar form to show how the learned energy enters the soft Bellman target. For a replayed transition (s, a, r, s'), we sample the next action from the current policy and form the target and squared temporal-difference loss:

$$y(r, s', a') = r + \gamma\big[Q_\phi(s', a') + \tau U_\psi(s', a', t_{\min})\big], \qquad a' = f_\theta(s', \varepsilon'), \tag{14}$$

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D},\ \varepsilon'\sim\mathcal{N}(0,I)}\left[\big(Q_\phi(s, a) - y(r, s', a')\big)^2\right]. \tag{15}$$

The target y is held fixed when differentiating $\mathcal{L}_Q$. Eq. (15) illustrates the semantics of the critic loss, but our implementation instead uses a categorical distributional critic trained with a cross-entropy loss. The mean of its predicted return distribution gives $Q_\phi(s, a)$ (Appendix B.1).

Actor Update. The actor maximizes the objective in (4), using the learned energy to estimate the entropy term:

$$J_{\text{actor}}(\theta) = \mathbb{E}_{s\sim\mathcal{D},\ \varepsilon\sim\mathcal{N}(0,I)}\left[Q_\phi\big(s, f_\theta(s,\varepsilon)\big) + \tau U_\psi\big(s, f_\theta(s,\varepsilon), t_{\min}\big)\right]. \tag{16}$$

Gradients propagate through the reparametrized sampled action $f_\theta(s,\varepsilon)$ while ϕ, ψ, and τ are held fixed. Action generation requires only a forward pass through f_θ. We update the actor once every G_π gradient steps, while the critic and energy network update at every step. This allows the energy network G_π updates to improve its estimate of the current policy's negative log-density before the next actor update changes the policy. Section 5.3 evaluates the effect of the number of energy updates per actor update on returns.

Temperature Update. Following SAC (Haarnoja et al., 2018), we auto-adjust τ according to the difference between the estimated policy entropy and a target entropy $\mathcal{H}_{\text{tar}}$:

$$\mathcal{L}_\tau(\tau) = \tau\mathbb{E}_{s\sim\mathcal{D},\ \varepsilon\sim\mathcal{N}(0,I)}\left[U_\psi\big(s, f_\theta(s,\varepsilon), t_{\min}\big) - \mathcal{H}_{\text{tar}}\right]. \tag{17}$$

We minimize $\mathcal{L}_\tau$ with respect to $\log\tau$ while holding the actor and energy parameters fixed. We use $\mathcal{H}_{\text{tar}} = -d$ by default.

Algorithm 1 SNARL

1: **Initialize** critic Q_ϕ, energy U_ψ, reparametrized actor f_θ, and temperature τ; set G_π energy updates per actor update
2: **repeat**: Act with $a = f_\theta(s,\varepsilon)$, $\varepsilon\sim\mathcal{N}(0,I)$, and store (s,a,r,s') in $\mathcal{D}$
3: Sample minibatch $\mathcal{B} = \{(s,a,r,s')\}\sim\mathcal{D}$
4: Sample $t\sim p_T$ (log-uniform on $[t_{\min}, t_{\max}]$)
5: Construct $a^{(t)}$ from fresh policy actions at the states and next states in $\mathcal{B}$ via (6)
6: Update U_ψ by minimizing (13)
7: Update Q_ϕ by minimizing (15)
8: Update f_θ by maximizing (16), every G_π steps
9: Update τ by minimizing (17)

5 Experiments

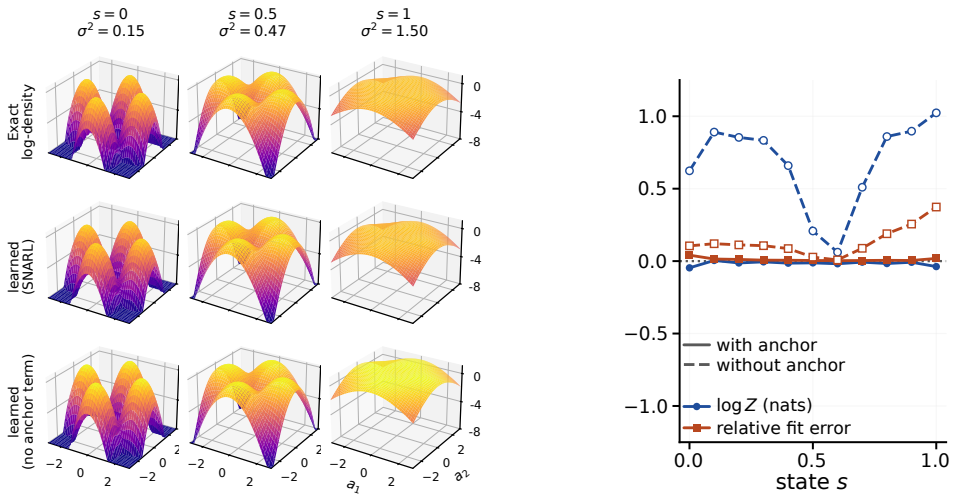
We first assess log-density estimation and energy normalization for fixed conditional distributions, then test tracking under prescribed distribution changes and policy updates driven by the learned energy. We then evaluate the control returns and action inference time of SNARL with a 1-NFE reparametrized policy. Finally, ablations examine how energy estimation, anchoring, the number of energy updates per actor update, and actor architecture affect control performance.

We evaluate SNARL on six MuJoCo tasks (Todorov et al., 2012; Towers et al., 2025), seven DeepMind Control tasks (Tassa et al., 2018), and four humanoid joystick tasks in MuJoCo Playground (Zakka et al., 2025). DIME (Celik et al., 2025) serves as a common diffusion-policy baseline across all three benchmarks. For each control task, we run SNARL with 10 random seeds and report the interquartile mean (IQM) return with 95% bootstrap confidence intervals (Agarwal et al., 2021).

5.1 Energy Estimation and Tracking

We first train one conditional energy network on a two-dimensional Gaussian mixture whose component variance changes with the state, using states sampled from a continuous range. Figure 1 shows that the network captures the different log-density shapes across states. We assess normalization using $\log Z(s)$, where $Z(s) = \int_{\mathbb{R}^2}\exp[-U_\psi(s,a,t_{\min})]\,\mathrm{d}a$ and a normalized energy function has $\log Z(s) = 0$.[1] The anchor improves normalization: including it reduces the maximum $|\log Z(s)|$ over 11 evaluation states from 1.024 to 0.046. After subtracting the mean energy error at each state, the two models have similar fit errors, indicating similar recovery of the distributional shape (Appendix D.3).

[1] For the exact normalized density, the integration procedure yields a numerical error of $|\log\widehat{Z}(s)| \approx 5\times10^{-4}$.



(a) (b)

Figure 1: Energy estimation and normalization on a conditional Gaussian mixture. Panel (a) compares the exact log-density (top) with $-U_\psi(s, a, t_{\min})$ learned with the anchor (middle) and without it (bottom) at three states. Panel (b) reports $\log Z(s)$ and relative energy fit errors at 11 states; solid and dashed lines denote the models with and without the anchor, respectively. The fit errors include additive offsets. Appendix D.3 defines the evaluation metrics.

We next examine whether the energy estimator can track the negative log-density of a distribution that changes throughout training. In Appendix D.4, a two-dimensional Gaussian mixture follows a prescribed trajectory, allowing comparison with its exact negative log-density throughout the experiment. Increasing the number of energy updates between target changes reduces the tracking error (Figure 8). The error remains higher than the static-target level while the distribution moves and decreases when it stops. These results show how additional energy updates improve log-density tracking under sustained distribution changes.

To test tracking when the estimate also guides policy updates, Appendix D.5 uses a 29-dimensional Gaussian mixture actor and compares learning with the estimated entropy against a control using exact entropy. We measure the energy estimate's response to uniformly widening the policy, which tests the entropy signal available to the actor in that direction. With more energy updates between actor updates, this response recovers toward the exact value; with only one energy update per actor update, it deteriorates and does not recover during the run (Figure 10). At a fixed number of actor updates, more energy updates per actor update narrow the objective gap relative to the exact-entropy control (Table 7). This experiment connects improved tracking of the actor's entropy response with better optimization of the reward-plus-entropy objective.

5.2 Control Performance and Inference Efficiency

MuJoCo. We compare SNARL with DIME, SAC Flow, FlowRL, and MFPO on six MuJoCo tasks over 10^6 environment steps (Figure 2). SNARL achieves the highest final IQM returns on HalfCheetah-v4, Walker2d-v4, and Humanoid-v4, with returns close to the best baseline on the remaining three tasks. These results show that SNARL's 1-NFE action generation model does not compromise competitive control performance against the evaluated diffusion and flow policies.

DeepMind Control. We compare SNARL with DIME, SAC Flow, FlowRL, and TD-MPC2 (Hansen et al., 2024) on seven DeepMind Control tasks over 10^6 environment steps (Figure 3). SNARL with a 1-NFE policy achieves returns comparable to the top performing baseline, DIME with a diffusion policy, on all domains but dog-run.

MuJoCo Playground. We evaluate G1 and T1 humanoid locomotion on flat and rough terrain in MuJoCo Playground (Figure 4). SNARL achieves significantly higher final IQM returns on all four tasks. The largest margin over the strongest baseline occurs on T1 rough terrain.

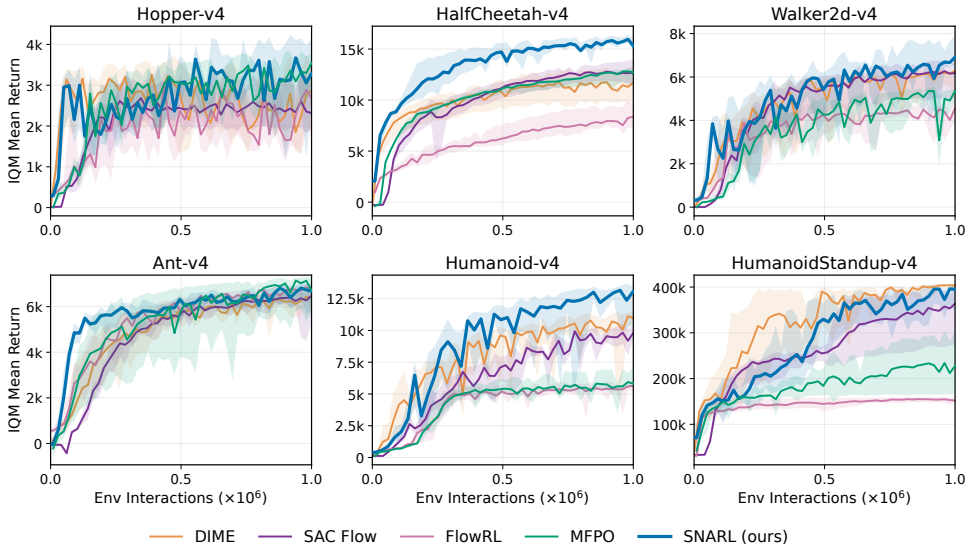


Figure 2: Learning curves on six MuJoCo tasks. With a 1-NFE actor, SNARL achieves the highest final IQM returns on 3 out of 6 tasks, and returns close to the best baseline on the remaining tasks.

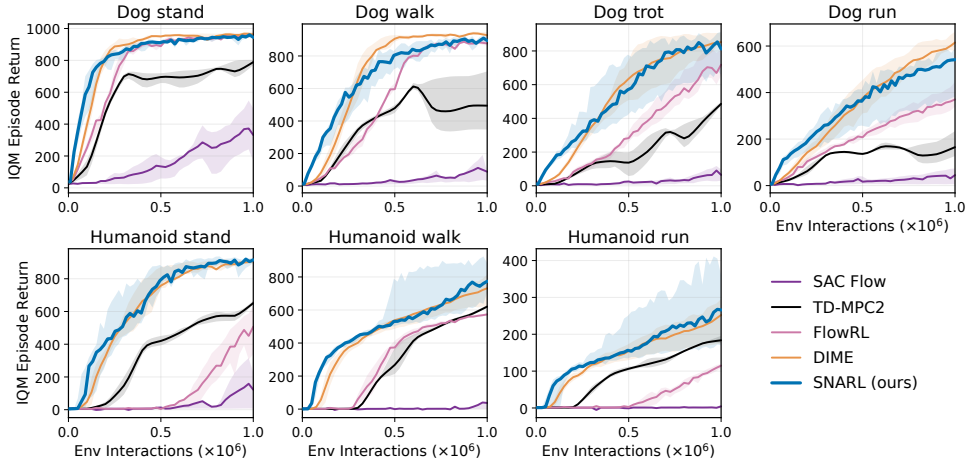


Figure 3: Learning curves on seven DeepMind Control tasks. With a 1-NFE actor, SNARL achieves final returns similar to DIME on six tasks, with lower returns on dog-run.

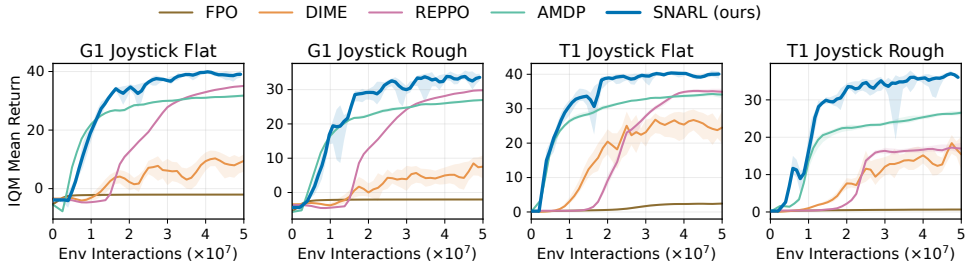


Figure 4: Learning curves on four humanoid joystick tasks in the GPU simulator of MuJoCo Playground, covering G1 and T1 on flat and rough terrain. SNARL achieves the highest final IQM returns on all four tasks, with the largest margin over the strongest baseline on T1 rough terrain.

Action inference efficiency. We measure action inference latency at batch size one on a single NVIDIA TITAN RTX and average the timings across the six MuJoCo tasks (Table 1). With one

network evaluation per action, SNARL reduces mean inference latency from DIME's 0.500 ms to 0.036 ms; MFPO's mean latency is 0.148 ms. SNARL reduces mean action inference latency by a factor of approximately 14 relative to DIME.

5.3 Ablation Studies

We examine how energy estimation, anchor loss, and the number of energy updates per actor update affect policy learning in SNARL. We also evaluate the estimator's compatibility with a multi-NFE flow actor. Appendix D.1 reports a sensitivity analysis of the replay buffer capacity.

Energy estimation and anchoring. We compare SNARL with two ablations on DMC dog-trot to examine the roles of energy estimation and anchoring. The first ablation removes only the anchor loss, retaining the energy network and both score losses. The second ablation removes the entropy terms from the critic and actor objectives altogether, effectively setting $\tau = 0$ and disabling the energy modeling. Both ablations reduce performance, with a larger decrease when the anchor loss is removed (Figure 5(a)). Without the anchor loss, the estimated entropy shifts upward and remains above the full method's estimates (Figure 5(b)). The return improvement from retaining the anchor corresponds to its normalization benefit in the controlled experiments of Section 5.1.

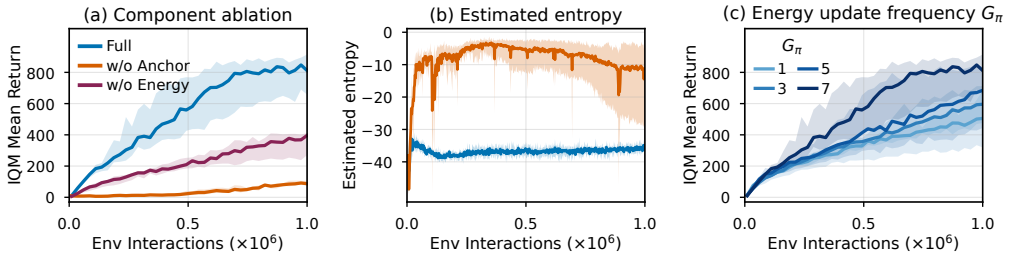


Figure 5: Ablations on DMC dog-trot. Panel (a) compares SNARL (Full) with removal of the anchor loss or the energy model. Panel (b) reports estimated entropy with and without the anchor, using the same curve colors as in (a). Panel (c) varies the number of energy updates per actor update, G_π. Shaded regions indicate 95% bootstrap confidence intervals.

Energy updates per actor update. More energy updates between actor updates give the estimator more opportunities to fit the current policy's negative log-density before the policy changes. We vary the number of energy updates per actor update, $G_\pi \in \{1, 3, 5, 7\}$, on DMC dog-trot (Figure 5(c)). With more energy updates per actor update, SNARL achieves higher IQM returns over the final five evaluations. This trend is consistent with the controlled diagnostics in Section 5.1, where additional energy updates improve log-density tracking.

Policy architecture. We replace the 1-NFE pushforward actor with the 4-NFE flow actor from SAC Flow (Zhang et al., 2026), keeping the energy estimator and all other training components unchanged. The flow actor achieves a slightly higher final IQM return on Humanoid-v4, while the 1-NFE actor performs better on both G1 joystick tasks (Appendix D.2).

6 Conclusion

SNARL decouples policy log-density estimation from action generation by learning a separate energy model from current policy samples and approximately normalizing it through Gaussian anchoring. This separation enables MaxEnt learning with a 1-NFE pushforward actor without requiring the actor to provide a tractable log-density. Controlled experiments show that the Gaussian anchor improves energy normalization and that additional energy updates improve tracking of changing log-densities. Across continuous-control benchmarks, SNARL achieves competitive or superior returns, with approximately $14\times$ faster action inference than DIME in our benchmark. Future work could improve training efficiency by adapting the number of energy updates to changes in the policy.

Reproducibility Statement

The result the method rests on is Lemma 1; its proof is Appendix A.1, the derivation of the training objective (13) from it is Appendix A.2, and the level error incurred by anchoring at a finite noise level rather than in the limit (9) is quantified in Appendix A.3. The parameterization of the energy network, including its additive term that depends on the state and noise level and the derivatives used by the score losses, is given in Appendix B.2. Algorithm 1 states the update order, and each of the four updates appears as a display in Section 4.2 specifies the objective and update rule for each component. Appendix C gives the hyperparameter configuration of every arm we ran, per task family, together with the evaluation protocol.

For SNARL, we report IQM returns over 10 seeds per control task with 95% confidence intervals from stratified bootstrapping. The environment-step budgets are stated with the benchmark comparisons. Appendix C records the baseline sources, seed counts, and statistical reporting conventions.

Use of Large Language Models

Large language models were used as an assistant in the preparation of this work: for prose editing and LaTeX work on the manuscript, for routine coding support in the implementation and the analysis scripts, and for reviewing drafts of the derivations, where they surfaced candidate errors that the authors then checked and corrected themselves. The research question, the method, the experimental design and the interpretation of the results are the authors' own. Every derivation, claim and number in this paper has been verified by the authors, who take full responsibility for the entire content, including any portion produced with such assistance. No large language model meets the criteria for authorship or contributorship under the ICLR policy, and none is listed as an author.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Jacob Andreas, Maxim Rabinovich, Michael I. Jordan, and Dan Klein. On the accuracy of self-normalized log-linear models. In *Advances in Neural Information Processing Systems*, volume 28, pp. 1783–1791. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/4a08142c38dbe374195d41c04562d9f8-Paper.pdf.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 2017.

Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. CrossQ: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *International Conference on Learning Representations*, 2024. arXiv:1902.05605.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming

Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems*, 2025b. arXiv:2410.24164.

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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 6958–6977. PMLR, 2025. arXiv:2502.02316.

Chen-Hao Chao, Chien Feng, Wei-Fang Sun, Cheng-Kuang Lee, Simon See, and Chun-Yi Lee. Maximum entropy reinforcement learning via energy-based normalizing flow. In *Advances in Neural Information Processing Systems*, volume 37, pp. 56136–56165, 2024.

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

Yuhui Chen, Haoran Li, and Dongbin Zhao. Boosting continuous control with consistency policy. *arXiv preprint arXiv:2310.06343*, 2023.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Kristy Choi, Chenlin Meng, Yang Song, and Stefano Ermon. Density ratio estimation via infinitesimal classification. In *International Conference on Artificial Intelligence and Statistics*, pp. 2552–2573. PMLR, 2022.

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.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 53047–53066, 2024.

Xiaoyi Dong, Jian Cheng, and Xi Sheryl Zhang. Maximum entropy reinforcement learning with diffusion policy. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 13963–13983. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/dong25b.html.

Xiaoyi Dong, Xi Sheryl Zhang, and Jian Cheng. Mean flow policy optimization. In *Proceedings of the 43rd International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2026. arXiv:2604.14698.

Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. In *International Conference on Learning Representations*, 2022. arXiv:2103.06257.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *32nd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2016.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2026.

Florentin Guth, Zahra Kadkhodaie, and Eero P. Simoncelli. Learning normalized image densities via dual score matching. In *Advances in Neural Information Processing Systems*, volume 38, pp. 89796–89826. Curran Associates, Inc., 2025. arXiv:2506.05310.

Michael U. Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(11):307–361, 2012. URL https://www.jmlr.org/papers/v13/gutmann12a.html.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024. arXiv:2310.16828.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 2022.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. arXiv:2210.02747.

Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 93967–93990. Curran Associates, Inc., 2025. doi: 10.52202/085713-3143. arXiv:2506.12811.

Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, and Xiao Ma. FLAC: Maximum entropy RL via kinetic energy regularized bridge matching. In *Proceedings of the 43rd International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2026. arXiv:2602.12829.

Bogdan Mazoure, Thang Doan, Audrey Durand, Joelle Pineau, and R. Devon Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. In *Proceedings of the 3rd Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 430–444. PMLR, 2020. arXiv:1905.06893.

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, pp. 22323–22336, 2024.

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

Serge Thilges, Onur Celik, Denis Blessing, Emiliyan Gospodinov, and Gerhard Neumann. Scalable maximum entropy reinforcement learning for diffusion policies via adjoint matching. *arXiv preprint arXiv:2606.22630*, 2026.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-4916. Datasets and Benchmarks Track; arXiv:2407.17032.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2023. arXiv:2208.06193.

Patrick Nadeem Ward, Ariella Smofsky, and Avishek Joey Bose. Improving exploration in soft-actor-critic with normalizing flows policies. In *ICML Workshop on Invertible Neural Networks and Normalizing Flows*, 2019. arXiv:1906.02771.

Hanlin Yu, Arto Klami, Aapo Hyvärinen, Anna Korba, and Omar Chehab. Density ratio estimation with conditional probability paths. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 73146–73174. PMLR, 2025. URL https://proceedings.mlr.press/v267/yu25l.html.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo playground. *arXiv preprint arXiv:2502.08844*, 2025. Software: https://github.com/google-deepmind/mujoco_playground.

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 *International Conference on Learning Representations*, volume 2026, pp. 121983–122016, 2026.

A Derivation

We first state and prove the identities underlying Section 4.1, then derive the regression objectives and analyze the offset selected by anchoring at a finite noise level. We use x for a fixed action coordinate and $a^{(t)}$ for a corrupted action sampled through (6).

A.1 Formal statement and proof of Lemma 1

The time derivative below specializes the VP conditional time score of Yu et al. (2025, Section 4 and Appendix B.1) to our policy density and noise parameterization. The derivation also identifies the Gaussian endpoint and remaining state-dependent offset needed for the normalization analysis.

Lemma A.1 (VP energy derivatives and Gaussian endpoint)**.** *Fix a state s. Let $a \sim \pi_\theta(\cdot \mid s)$ have a density and finite second moment, and let $z \sim \mathcal{N}(0, I)$ be independent of a. For $t \in (0, 1)$, define $a^{(t)} = \sqrt{1-t}\, a + \sqrt{t}\, z$, let $p_t(\cdot \mid s)$ be its density, and set $U(s, x, t) = -\log p_t(x \mid s)$. Then $p_t(x \mid s) > 0$, and U is continuously differentiable in (x, t). For every fixed $t \in (0, 1)$ and $x \in \mathbb{R}^d$,*

$$\nabla_x U(s, x, t) = \mathbb{E}\left[\frac{z}{\sqrt{t}} \,\middle|\, s, a^{(t)} = x \right], \tag{18}$$

$$\partial_t U(s, x, t) = \mathbb{E}\left[\frac{d}{2t} - \frac{\|z\|^2}{2t} + \frac{\langle z,\, a \rangle}{2\sqrt{t}\sqrt{1-t}} \,\middle|\, s, a^{(t)} = x \right]. \tag{19}$$

The time derivative holds s and x fixed. At every fixed x,

$$\lim_{t \to 1} U(s, x, t) = \frac{\|x\|^2}{2} + \frac{d}{2} \log(2\pi). \tag{20}$$

If a differentiable function $\hat{U}(s, x, t)$ has the same space and time derivatives as U throughout $\mathbb{R}^d \times (0, 1)$, then $\hat{U}(s, x, t) = U(s, x, t) + c(s)$ for some constant $c(s)$ independent of x and t. Matching the limit in (20) fixes $c(s) = 0$.

Proof. We suppress the fixed state s in the proof. The Gaussian corruption kernel gives

$$U(x, t) = -\log p_t(x), \qquad p_t(x) = \int \pi_\theta(a) p(x \mid a, t)\, \mathrm{d}a, \tag{21}$$

where $p(x \mid a, t) = \mathcal{N}(x; \sqrt{1-t}\, a, tI)$. This kernel is strictly positive. On compact subsets of $\mathbb{R}^d \times (0,1)$, its first derivatives are bounded by $C(1 + \|a\|^2)$ for a finite constant C. The finite second moment therefore permits differentiation under the integral and gives continuous derivatives.

In the Gaussian conditional path notation of Yu et al. (2025, Appendix B), our kernel has mean $\mu_t(a) = \sqrt{1-t}\, a$ and variance $k_t = t$, with the clean action a as the conditioning variable. Our noise coordinate runs from data to Gaussian noise, reversing their VP path convention. Substituting this mean and variance into their conditional time score formula and changing the sign for $U = -\log p_t$ gives the target inside the expectation in (19).

Let $\partial_\bullet$ denote either a derivative in an action coordinate or the time derivative at fixed x. Differentiating (21) gives

$$\partial_\bullet U(x,t) = -\frac{\partial_\bullet p_t(x)}{p_t(x)} \tag{22}$$

$$= -\frac{\int \pi_\theta(a) p(x \mid a, t) \partial_\bullet \log p(x \mid a, t)\, \mathrm{d}a}{p_t(x)} \tag{23}$$

$$= \mathbb{E}_{a \sim p_t(\cdot \mid x)}\big[-\partial_\bullet \log p(x \mid a, t)\big]. \tag{24}$$

Here $p_t(a \mid x) = \pi_\theta(a) p(x \mid a, t)/p_t(x)$ is the conditional density of the clean action given $a^{(t)} = x$. It expresses conditioning of the joint draw $a \sim \pi_\theta$, $z \sim \mathcal{N}(0, I)$ in (6).

For the VP kernel,

$$-\log p(x \mid a, t) = \frac{\big\|x - \sqrt{1-t}\, a\big\|^2}{2t} + \frac{d}{2}\log(2\pi t). \tag{25}$$

Write $R = x - \sqrt{1-t}\, a$. Its spatial derivative gives $\nabla_x[-\log p(x \mid a, t)] = R/t$. At $x = a^{(t)}$, $R = \sqrt{t}\, z$; equivalently, in the conditional expectation at fixed x, $z = R/\sqrt{t}$. Substituting into (24) proves (18).

For the time derivative, hold x and a fixed. The derivative of the signal scale contributes

$$\partial_t \|R\|^2 = 2\,\langle R,\, \partial_t R\rangle = \frac{\langle R,\, a\rangle}{\sqrt{1-t}}. \tag{26}$$

Together with the derivatives of $1/(2t)$ and $(d/2)\log(2\pi t)$, this yields

$$\partial_t[-\log p(x \mid a, t)] = \frac{d}{2t} - \frac{\|R\|^2}{2t^2} + \frac{\langle R,\, a\rangle}{2t\sqrt{1-t}}. \tag{27}$$

Substituting $R = \sqrt{t}\, z$ into (24) gives

$$\partial_t U(x,t) = \mathbb{E}_{a \sim p_t(\cdot \mid x)}\left[\frac{d}{2t} - \frac{\|z\|^2}{2t} + \frac{\langle z,\, a\rangle}{2\sqrt{t}\sqrt{1-t}}\right], \tag{28}$$

which proves (19). The additional term relative to VE corruption comes from differentiating $\sqrt{1-t}$ and has a positive sign in the regression target.

For the endpoint, fix x. As $t \to 1$, $p(x \mid a, t) \to \mathcal{N}(x; 0, I)$ for every a. For $t \geq 1/2$, the kernel is bounded by $\pi^{-d/2}$ independently of a. Dominated convergence gives $p_t(x) \to \mathcal{N}(x; 0, I) > 0$, and taking the negative logarithm proves (20).

Finally, if $\hat{U}$ and U have the same two derivatives, their difference has zero gradient in (x, t). Since $\mathbb{R}^d \times (0,1)$ is connected, this difference is a constant, which may depend on the fixed state s. More specifically, matching the space derivative alone leaves a function $c(s,t)$, and matching the time derivative imposes $\partial_t c(s,t) = 0$. Matching the Gaussian endpoint then sets the remaining $c(s)$ to zero. □

Relation between the space and time derivatives. The VP densities also satisfy a Fokker–Planck equation. With the Ornstein–Uhlenbeck time coordinate $u = -\log(1-t)$, the channel has signal scale $e^{-u/2}$ and noise variance $1 - e^{-u}$. Transforming its density equation to t gives

$$\partial_t p_t(x) = \frac{1}{2(1-t)}\left[\Delta_x p_t(x) + \nabla_x \cdot \big(x p_t(x)\big)\right]. \tag{29}$$

Using $p_t = e^{-U}$, $\Delta_x p_t / p_t = \|\nabla_x U\|^2 - \Delta_x U$, and $\nabla_x \cdot (x p_t)/p_t = d - \langle x, \nabla_x U \rangle$, we obtain

$$\partial_t U = \frac{1}{2(1-t)} \left[\Delta_x U - \|\nabla_x U\|^2 + \langle x, \nabla_x U \rangle - d \right]. \tag{30}$$

Thus the exact time derivative is determined by the spatial derivatives of the energy. Matching it constrains the variation of the energy offset across noise levels, as shown in Lemma A.1.

A.2 From derivative identities to the training objective

Regression on sampled targets. We first show why the sampled targets in (10) and (11) can be used to learn the energy derivatives. Fix a state s and noise level t, and draw a and z as in (6). For the space loss, Lemma A.1 gives

$$\begin{aligned} &\mathbb{E}_{a,z}\left[\left\| (\nabla_x U_\psi)(s, a^{(t)}, t) - \frac{z}{\sqrt{t}} \right\|^2 \right] \\ &\quad = \mathbb{E}_{a,z}\left[\left\| (\nabla_x U_\psi)(s, a^{(t)}, t) - (\nabla_x U)(s, a^{(t)}, t) \right\|^2 \right] + C(s,t), \end{aligned} \tag{31}$$

where $C(s,t)$ is the expected squared difference between the sampled target $z/\sqrt{t}$ and its conditional mean, and is independent of ψ. To obtain this identity, add and subtract $(\nabla_x U)(s, a^{(t)}, t)$ inside the squared norm and expand. The cross term vanishes because the conditional mean of $z/\sqrt{t}$ given s and $a^{(t)}$ is exactly this derivative. Minimizing the sample regression loss therefore minimizes the expected error in the space derivative (Vincent, 2011).

The same argument applies to the time target in (8), giving the corresponding equivalence for (11). This is the conditional time score matching equivalence of Yu et al. (2025, Theorem 1), applied at a fixed state with the time score parameterized as $-\partial_t U_\psi$. Averaging these identities over replayed states and sampled noise levels, with the weights in (10) and (11), preserves this equivalence. Policy parameters remain fixed during these energy updates.

Anchor regression and joint optimization. Both derivative losses are unchanged by adding $c(s)$ to the energy. The Gaussian endpoint supplies an absolute reference: for $q = \mathcal{N}(0, I)$, its negative log-density at an independent reference action is $-\log q(a_{\text{ref}}) = \|a_{\text{ref}}\|^2/2 + (d/2)\log(2\pi)$. Dividing the energy error by the action dimension before squaring gives

$$\mathcal{L}_{\text{anchor}}(\psi) = \mathbb{E}_{s\sim\mathcal{D},\, a_{\text{ref}}\sim q}\left[\left(\frac{U_\psi(s, a_{\text{ref}}, t_{\max}) + \log q(a_{\text{ref}})}{d} \right)^2 \right], \tag{32}$$

which is (12). Evaluating this Gaussian reference at $t_{\max} < 1$ introduces the approximation analyzed in Appendix A.3.

All three losses use states sampled from the replay buffer. For the score losses, actions are fresh samples from the current policy, and Gaussian noise and noise levels are sampled independently as specified in Section 4.1. For the anchor, the reference action is independent of the state. We jointly minimize the three losses by their sum in (13).

A.3 The offset selected by anchoring at a finite noise level

The Gaussian endpoint is exact in the limit $t \to 1$, while the anchor is evaluated at $t_{\max} < 1$. We analyze the remaining energy offset when the two energy derivatives are matched exactly. In this setting, Lemma A.1 reduces the normalization analysis to choosing $c(s)$ in the family $U + c(s)$.

Proposition A.2 (Anchor bias under exact derivative matching)**.** *Fix a state s and assume that $\pi_\theta(\cdot \mid s)$ has a density and finite second moment. Let $q = \mathcal{N}(0, I)$ and $0 < t_{\max} < 1$. Suppose $\hat{U}$ matches the space and time derivatives in Lemma A.1 throughout $\mathbb{R}^d \times (0,1)$, so that $\hat{U}(s,x,t) = U(s,x,t) + c(s)$. Minimizing the population anchor loss at this state over $c(s)$ selects*

$$\begin{aligned} c^\star(s) &= \mathbb{E}_{a_{\text{ref}}\sim q}\left[-\log q(a_{\text{ref}}) + \log p_{t_{\max}}(a_{\text{ref}} \mid s) \right] \\ &= -\,\mathrm{KL}(q \,\|\, p_{t_{\max}}(\cdot \mid s)) \le 0. \end{aligned} \tag{33}$$

If in addition the policy is supported on $[-1,1]^d$*, then, as* $t_{\max} \to 1$*,*

$$\mathrm{KL}(q \,\|\, p_{t_{\max}}(\cdot \mid s)) = \frac{1 - t_{\max}}{2} \left\| \mathbb{E}_{a \sim \pi_\theta(\cdot|s)}[a] \right\|^2 + O\big((1 - t_{\max})^2\big)\,, \tag{34}$$

and $\mathrm{KL}(q \| p_{t_{\max}}(\cdot \mid s)) \le d(1 - t_{\max})/t_{\max}$*. For a centered policy, the divergence is* $O((1 - t_{\max})^2)$*.*

The proposition describes the offset selected within the family $U + c(s)$. Its leading magnitude depends on the squared mean action; a centered policy has no first-order term. The Gaussian regression target $\|a_{\mathrm{ref}}\|^2/2 + (d/2)\log(2\pi)$ has a distribution independent of $t_{\max}$. As $t_{\max} \to 1$, the magnitude of this offset tends to zero.

To prove the expansion, suppress the state and write $\rho = \sqrt{1 - t_{\max}}$. For $|\rho| < 1$, define $p_\rho(x) = \mathbb{E}_{a \sim \pi_\theta}[\mathcal{N}(x; \rho a, (1 - \rho^2) I)]$, so that $p_\rho = p_{t_{\max}}$ at the positive value of ρ used by the channel. The following symmetry determines the order of the remainder.

Lemma A.3 (Parity)**.** *The divergence* $\mathrm{KL}(q \| p_\rho)$ *is an even function of* ρ*. For bounded actions, it is four times continuously differentiable near* $\rho = 0$*. Its Taylor expansion through order three therefore has no odd powers and has remainder* $O(\rho^4)$*.*

Proof. The Gaussian covariance is unchanged when ρ changes sign, and

$$\begin{aligned} p_{-\rho}(x) &= \mathbb{E}_{a \sim \pi_\theta}[\mathcal{N}(x; -\rho a, (1 - \rho^2) I)] \\ &= \mathbb{E}_{a \sim \pi_\theta}[\mathcal{N}(-x; \rho a, (1 - \rho^2) I)] = p_\rho(-x). \end{aligned} \tag{35}$$

Since $q(-x) = q(x)$, changing variables $x \mapsto -x$ gives

$$\mathrm{KL}(q \| p_{-\rho}) = \int q(x) \log \frac{q(x)}{p_\rho(-x)} \, \mathrm{d}x = \mathrm{KL}(q \| p_\rho). \tag{36}$$

For bounded a and $|\rho| \le \rho_0 < 1$, the first four derivatives in ρ of the Gaussian log kernel are bounded by polynomials in $\|x\|$, uniformly in a and ρ. Differentiating its log integral expresses derivatives of $\log p_\rho(x)$ as conditional moments and cumulants of these quantities, with the same type of polynomial bounds. These bounds are integrable under q, permitting four differentiations under the integral defining the divergence. Taylor's theorem and evenness then give the stated remainder. □

Proof of Proposition A.2. Step 1: the offset selected by the anchor. Under the derivative matching assumption, Lemma A.1 gives $\hat{U}(s, x, t) = U(s, x, t) + c(s)$. Define $\Delta(x) = -\log q(x) - U(s, x, t_{\max})$. At the fixed state s, the anchor objective becomes

$$\begin{aligned} \mathcal{L}_{\mathrm{anchor}}(c; s) &= \frac{1}{d^2} \mathbb{E}_{a_{\mathrm{ref}} \sim q} \Big[\big(U(s, a_{\mathrm{ref}}, t_{\max}) + c + \log q(a_{\mathrm{ref}}) \big)^2 \Big] \\ &= \frac{1}{d^2} \mathbb{E}_{a_{\mathrm{ref}} \sim q} \Big[\big(c - \Delta(a_{\mathrm{ref}}) \big)^2 \Big]. \end{aligned} \tag{37}$$

This scalar regression has finite squared risk. Indeed, Jensen's inequality bounds $U(s, x, t_{\max})$ above by $\mathbb{E}_a[-\log \mathcal{N}(x; \sqrt{1 - t_{\max}}\, a, t_{\max} I)]$, which grows at most quadratically in x under the finite second moment assumption. The kernel upper bound gives $U(s, x, t_{\max}) \ge (d/2)\log(2\pi t_{\max})$, so $\Delta(x)^2$ is bounded by an integrable polynomial under q. Its minimizing constant is therefore

$$\begin{aligned} c^\star(s) &= \mathbb{E}_{a_{\mathrm{ref}} \sim q}[\Delta(a_{\mathrm{ref}})] \\ &= \mathbb{E}_{a_{\mathrm{ref}} \sim q}\big[-\log q(a_{\mathrm{ref}}) + \log p_{t_{\max}}(a_{\mathrm{ref}} \mid s) \big] \\ &= -\mathrm{KL}(q \,\|\, p_{t_{\max}}(\cdot \mid s)) \le 0. \end{aligned} \tag{38}$$

The factor $1/d^2$ rescales the objective without changing this minimizer.

Step 2: expansion near the Gaussian endpoint. For bounded actions, write

$$\begin{aligned} u(a, x) &= \log \mathcal{N}(x; \rho a, (1 - \rho^2) I) - \log q(x) \\ &= -\frac{d}{2} \log(1 - \rho^2) - \frac{\|x - \rho a\|^2}{2(1 - \rho^2)} + \frac{\|x\|^2}{2}. \end{aligned} \tag{39}$$

Then $-\mathrm{KL}(q\|p_\rho) = \mathbb{E}_{x\sim q}[\log \mathbb{E}_{a\sim\pi_\theta}[e^{u(a,x)}]]$. The polynomial bounds in the proof of Lemma A.3 justify taking its Taylor coefficients under the Gaussian expectation. Expanding at each fixed x gives

$$u(a,x) = \rho\,\langle x,\, a\rangle + \frac{\rho^2}{2}\left(d - \|a\|^2 - \|x\|^2\right) + O(\rho^3), \tag{40}$$

and $\log \mathbb{E}_a[e^u] = \mathbb{E}_a[u] + \frac{1}{2}\operatorname{Var}_a(u) + O(\rho^3)$. Using $\mathbb{E}_q[x] = 0$, $\mathbb{E}_q[\|x\|^2] = d$, and $\mathbb{E}_q[x^\top M x] = \operatorname{tr}(M)$ yields

$$\begin{aligned} \mathbb{E}_q\mathbb{E}_a[u] &= \frac{\rho^2}{2}\left(d - \mathbb{E}_a[\|a\|^2] - d\right) + O(\rho^3), \\ \frac{1}{2}\mathbb{E}_q \operatorname{Var}_a(u) &= \frac{\rho^2}{2}\left(\mathbb{E}_a[\|a\|^2] - \|\mathbb{E}_a[a]\|^2\right) + O(\rho^3). \end{aligned} \tag{41}$$

For the second line, $\operatorname{Var}_a(u) = \rho^2 x^\top \operatorname{Cov}_a(a)x + O(\rho^3)$. The second moments cancel between the two lines. Lemma A.3 removes the cubic term in the divergence, giving

$$\mathrm{KL}(q\|p_{t_{\max}}) = \frac{1 - t_{\max}}{2}\,\|\mathbb{E}_{a\sim\pi_\theta}[a]\|^2 + O\big((1-t_{\max})^2\big)\,. \tag{42}$$

When the policy is centered, the leading term vanishes and the divergence is $O((1-t_{\max})^2)$.

Step 3: a bound at any finite noise level. Applying Jensen's inequality to the negative logarithm of the expected Gaussian kernel, followed by the formula for Gaussian KL divergence, gives

$$\begin{aligned} \mathrm{KL}(q\|p_{t_{\max}}) &\le \mathbb{E}_{a\sim\pi_\theta}[\mathrm{KL}(\mathcal{N}(0,I)\,\|\,\mathcal{N}(\rho a, t_{\max} I))] \\ &= \frac{(1-t_{\max})\left(\mathbb{E}_a[\|a\|^2] + d\right)}{2t_{\max}} + \frac{d}{2}\log t_{\max}. \end{aligned} \tag{43}$$

On $[-1,1]^d$, $\|a\|^2 \le d$ and $\log t_{\max} \le 0$, so $\mathrm{KL}(q\|p_{t_{\max}}) \le d(1-t_{\max})/t_{\max}$. This bound holds for every policy supported on the cube. For centered policies, the expansion in (42) gives the sharper quadratic order near the endpoint. □

B Network Architecture and Implementation

B.1 Distributional Critic

Distributional Q-learning models the distribution of returns, with its categorical formulation trained using cross-entropy against a projected Bellman target (Bellemare et al., 2017). This formulation is also adopted in DIME and MFPO (Celik et al., 2025; Dong et al., 2026). Following these methods, we represent returns as a categorical distribution over a fixed support and use its mean as the scalar value $Q_\phi(s,a)$. As in DIME, we adopt CrossQ's critic update, using batch renormalization and no target network for bootstrapping (Bhatt et al., 2024).

B.2 Energy Network

We use the inner product construction of Guth et al. (2025), implemented by a network g_ψ with a d-dimensional output, and add a learned scalar $C_\psi(s,t)$ that depends on the state and noise level but not on the action. For a generic action coordinate x, the energy is

$$U_\psi(s,x,t) = \tfrac{1}{2}\,\langle x,\, g_\psi(s,x,t)\rangle + C_\psi(s,t), \tag{44}$$

In particular, $U_\psi(s,0,t) = C_\psi(s,t)$, so the energy at the origin can vary with the conditioning state and the noise level. This avoids fixing that value to zero, as the inner product alone would do.

The action pathway uses bias-free linear maps, a mean-free normalization, and a smooth activation. The two derivatives used in the score losses, with x held fixed in the time derivative, are

$$\nabla_x U_\psi(s,x,t) = \tfrac{1}{2}\Big[g_\psi(s,x,t) + \nabla_x g_\psi(s,x,t)^\top x\Big], \tag{45}$$

$$\partial_t U_\psi(s,x,t) = \tfrac{1}{2}\,\langle x,\, \partial_t g_\psi(s,x,t)\rangle + \partial_t C_\psi(s,t). \tag{46}$$

We compute both derivatives by automatic differentiation of (44) and train the energy network through the score losses using double backpropagation. Thus the spatial score loss does not constrain the

additive term, while the time score loss constrains its time derivative together with that of the inner product. The learned function $C_\psi(s,t)$ is distinct from the residual ambiguity $c(s)$ in Lemma A.1: exact matching of both scores determines the total energy up to an additive function of the state alone. Adding any $c(s)$ to $C_\psi(s,t)$ leaves both derivatives unchanged, and the endpoint anchor supplies the information needed to fix this remaining freedom. The bias from the finite endpoint is characterized under exact score matching in Proposition A.2.

C Experiment Details

Baseline result sources. For MuJoCo, we reproduce DIME, SAC Flow, FlowRL, and MFPO using their respective open-source implementations (Celik et al., 2025; Zhang et al., 2026; Lv et al., 2025; Dong et al., 2026). Each method uses 10 random seeds per task. The DeepMind Control baseline curves in Figure 3 show IQM returns reported in prior work (Celik et al., 2025; Hansen et al., 2024; Zhang et al., 2026; Lv et al., 2025; 2026). For these DMC curves, the reported evaluation protocol uses five seeds for DIME, SAC Flow, and FlowRL, and three for TD-MPC2. Baseline learning curves in MuJoCo Playground are taken from publicly available results in the AMDP study (Thilges et al., 2026). The DIME results use the AMDP authors' reimplementation, and the AMDP curve uses their standard evaluation without best-of-N action selection. These baseline results report IQM returns over 10 seeds with 95% confidence intervals from stratified bootstrapping.

C.1 Efficiency Comparison

We measure action inference latency for seven algorithms at batch size one on a single NVIDIA TITAN RTX GPU and average the timings across the six MuJoCo tasks. Table 1 reports the mean latency and the number of network function evaluations (NFE) required to generate an action. SNARL uses one function evaluation and takes 0.036 ms on average, compared with 0.148–2.084 ms for the evaluated flow and diffusion policies.

Table 1: Mean action inference latency across six MuJoCo tasks at batch size one on a single NVIDIA TITAN RTX GPU. NFE counts neural function evaluations per action.

Algorithm	Type	NFE	Mean inference (ms)
SAC	Gaussian	1	0.018
SNARL	**Pushforward**	**1**	**0.036**
FlowRL	Flow	2	0.332
MFPO	Flow	2	0.148
SAC-Flow	Flow	4	0.190
FPO	Flow	10	2.084
DIME	Diffusion	16	0.500

C.2 Hyperparameters

Tables 2–4 list the settings for each benchmark. A dash denotes an inapplicable parameter. Hidden depth excludes the output layer, and hidden size denotes the layer width. Entropy targets retain each source's definition and sign convention. For SNARL, G_π denotes the number of energy updates per actor update.

For MuJoCo, we train the baselines using their respective official code releases. FlowRL's midpoint step uses two network evaluations. MFPO selects the highest-Q action among ten candidates at evaluation; its two sampling steps apply to each candidate.

For SNARL, the critic and energy network update at every gradient step, and all learning rates remain constant.

Table 2: Hyperparameters for the MuJoCo experiments in Figure 2. A dash indicates an inapplicable parameter.

Parameter	**DIME**	**SAC Flow**	**FlowRL**	**MFPO**	**SNARL**
Batch size	256	512	256	256	256
Replay capacity	10^6	10^6	10^6	10^6	10^6
Discount	0.99	0.99	0.99	0.99	0.99
$\mathcal{H}_{target}$	$4\text{dim}(\mathcal{A})$	0	–	$-0.5\text{dim}(\mathcal{A})$	$-\text{dim}(\mathcal{A})$
Optimizer	Adam	Adam	Adam	Adam	Adam
Actor learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Critic learning rate	3×10^{-4}	10^{-3}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Energy learning rate	–	–	–	–	3×10^{-4}
Temperature learning rate	1×10^{-4}	3×10^{-4}	–	3×10^{-4}	3×10^{-4}
Update-to-data ratio	2	1	1	1	1
Energy updates per actor update G_π	–	–	–	–	7
Warmup steps	5,000	50,000	–	–	5,000
Sampling steps	16	4	1	2	1
Energy $t_{\min}$	–	–	–	–	10^{-4}
Energy $t_{\max}$	–	–	–	–	0.99
Actor hidden depth	3	2†	2	3	4
Actor hidden size	256	96†	512	256	256
Critic hidden depth	2	2	3	3	2
Critic hidden size	2048	1024	256	256	2048
Energy hidden depth	–	–	–	–	3
Energy hidden size	–	–	–	–	256
Categorical bins	101	–	–	101	101

† For SAC Flow, depth denotes the number of Transformer decoder layers and size denotes the model dimension; the actor uses four attention heads.

Table 3: Hyperparameters for the DMC experiments in Figure 3. The SAC Flow column lists the MuJoCo reference settings from Table 2.

Parameter	**DIME**	**SAC Flow**	**FlowRL**	**TD-MPC2**	**SNARL**
Batch size	256	512	256	256	256
Replay capacity	10^6	10^6	10^6	10^6	10^6
Discount	0.99	0.99	0.99	0.99	0.99
$\mathcal{H}_{target}$	$4\dim(\mathcal{A})$	0	–	–	$-\dim(\mathcal{A})$
Optimizer	Adam	Adam	Adam	Adam	Adam
Actor learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Critic learning rate	3×10^{-4}	10^{-3}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Energy learning rate	–	–	–	–	3×10^{-4}
Temperature learning rate	1×10^{-4}	3×10^{-4}	–	–	3×10^{-4}
Update-to-data ratio	2	1	1	1	1
Energy updates per actor update G_π	–	–	–	–	7
Warmup steps	5,000	50,000	–	2,500	5,000
Sampling steps	16	4	1	–	1
Energy $t_{\min}$	–	–	–	–	10^{-4}
Energy $t_{\max}$	–	–	–	–	0.99
Actor hidden depth	3	2†	2	2	4
Actor hidden size	256	96†	512	512	256
Critic hidden depth	2	2	3	2	2
Critic hidden size	2048	1024	512	512	2048
Energy hidden depth	–	–	–	–	3
Energy hidden size	–	–	–	–	256
Categorical bins	101	–	–	101	101
Action repeat	–	–	–	2	–
Planning horizon	–	–	–	3	–
Planning iterations	–	–	–	8	–
Trajectory population	–	–	–	512	–
Policy trajectories	–	–	–	24	–
Model size	–	–	–	5M	–
Encoder learning rate	–	–	–	10^{-4}	–

† For SAC Flow, depth denotes the number of Transformer decoder layers and size denotes the model dimension; the actor uses four attention heads.

Table 4: Hyperparameters for the MuJoCo Playground experiments in Figure 4. Baseline settings are reported by AMDP (Thilges et al., 2026).

Parameter	**FPO**	**DIME**	**REPPO**	**AMDP**	**SNARL**
Batch size	–	256	1,024	1,024	2048
Buffer size	–	131,072	131,072	131,072	4×10^6
Discount	0.97	0.97	0.97	0.97	0.97
$\mathcal{H}_{target}$	–	$-4\dim(\mathcal{A})$	$-0.5\dim(\mathcal{A})$	$-\dim(\mathcal{A})$	$-\dim(\mathcal{A})$
Optimizer	Adam	Adam	Adam	Adam	Adam
Actor learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Critic learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Energy learning rate	–	–	–	–	3×10^{-4}
Temperature learning rate	–	1×10^{-4}	–	1×10^{-4}	3×10^{-4}
Update-to-data ratio	–	$1/32$	$1/32$	$1/32$	1/256
Energy updates per actor update G_π	–	–	–	–	7
Sampling steps	10	16	–	16	1
Energy $t_{\min}$	–	–	–	–	10^{-4}
Energy $t_{\max}$	–	–	–	–	0.99
Actor hidden depth	5	3	3	3	4
Actor hidden size	32	512	512	512	256
Critic hidden depth	5	3	3	3	4
Critic hidden size	256	512	512	512	256
Energy hidden depth	–	–	–	–	3
Energy hidden size	–	–	–	–	256
Categorical bins	–	101	151	151	101

C.3 ENVIRONMENTS

Table 5 lists the 17 tasks evaluated in Section 5.2, with their observation and action dimensions and control objectives.

MuJoCo. We use six Gymnasium v4 tasks with CPU-based MuJoCo simulation (Todorov et al., 2012; Towers et al., 2025). These tasks cover locomotion with different body structures and recovery to a standing posture.

DeepMind Control. We use the CPU implementation of the DeepMind Control Suite (Tassa et al., 2018), with four dog tasks and three humanoid tasks. These tasks test upright standing and locomotion at different target speeds.

MuJoCo Playground. We use the GPU-accelerated MuJoCo XLA (MJX) simulator in MuJoCo Playground (Zakka et al., 2025). The Unitree G1 and Booster T1 humanoids track joystick commands on flat and rough terrain. The actor and energy network use policy observations, while the critic additionally receives privileged information.

Table 5: Actor and critic observation dimensions, action dimensions, and objectives for our control tasks.

Task	Actor obs. dim.	Critic obs. dim.	Act. dim.	Description
MuJoCo (CPU)				
Hopper-v4	11	11	3	Hop forward while maintaining balance.
HalfCheetah-v4	17	17	6	Run forward with a planar articulated body.
Walker2d-v4	17	17	6	Walk forward with a planar biped.
Ant-v4	27	27	8	Move a quadruped forward.
Humanoid-v4	376	376	17	Move a humanoid forward without falling.
HumanoidStandup-v4	376	376	17	Make a lying humanoid stand up.
DeepMind Control (CPU)				
dog-stand	223	223	38	Keep the dog standing upright.
dog-walk	223	223	38	Move at the target walking speed.
dog-trot	223	223	38	Move at the target trotting speed.
dog-run	223	223	38	Move at the target running speed.
humanoid-stand	67	67	21	Keep the humanoid standing upright.
humanoid-walk	67	67	21	Walk at the target speed.
humanoid-run	67	67	21	Run at the target speed.
MuJoCo Playground (GPU)				
G1JoystickFlatTerrain	103	216	29	Track joystick commands on flat terrain.
G1JoystickRoughTerrain	103	216	29	Track joystick commands on rough terrain.
T1JoystickFlatTerrain	85	180	23	Track joystick commands on flat terrain.
T1JoystickRoughTerrain	85	180	23	Track joystick commands on rough terrain.

The energy network uses the same observations as the actor.

D Additional Ablation Study

D.1 Replay Capacity

We investigate the effect of replay buffer capacity in longer (5×10^7 environment steps) MuJoCo Playground training processes, where the buffer fills during training. Holding other settings fixed, we compare capacities of one, two, four, and eight million transitions on the G1 and T1 rough-terrain tasks (Figure 6). Final IQM returns improve as capacity increases from one to four million transitions on both tasks, but appear to saturate there.

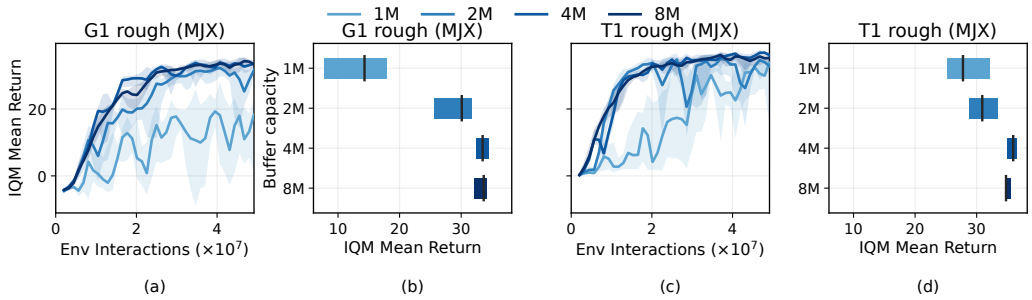


Figure 6: Replay buffer capacity, in transition count, on the two rough-terrain joystick tasks of MuJoCo Playground. Panels (a) and (c) plot the IQM over 10 seeds against environment steps with 95% stratified bootstrap bands; (b) and (d) summarize the IQM and its confidence interval, averaged over the last five evaluations of each training process.

D.2 Policy Architecture

Figure 7 provides the learning curves for the actor comparison in Section 5.3. On the G1 joystick tasks, we include AMDP (Thilges et al., 2026) as a multi-NFE diffusion baseline, since MuJoCo Playground was not included in the published SAC Flow study.

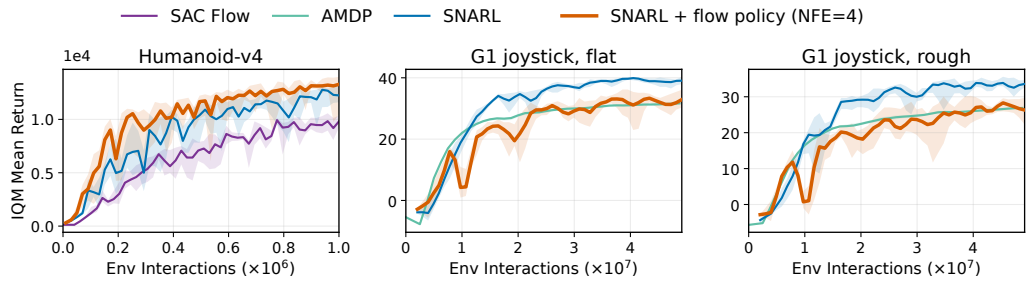


Figure 7: SNARL with a 4-NFE flow policy (orange) instead of a 1-NFE pushforward actor (blue), with all other components held fixed. IQM over 10 seeds with 95% stratified bootstrap bands.

D.3 Ablation of the Anchor Loss

The two models in Figure 1 differ only in whether $\mathcal{L}_{\text{anchor}}$ is included in (13). They use the same random seed, network architecture, VP corruption, and 2.56×10^5 gradient steps, and are evaluated at the same 11 states.

Both models use the energy parameterization in Appendix B.2, including the learned scalar term $C_\psi(s, t)$. This term is independent of the action but can depend on the noise level, so its time derivative enters time score matching. It differs from the remaining ambiguity $c(s)$, which is independent of both action and noise level and therefore leaves both score derivatives unchanged.

We evaluate normalization using the numerical estimates of $\log Z(s)$ defined in Section 5.1. To measure energy fit, we compute the root mean square error against $-\log p_{\text{mix}}(a \mid s)$, giving equal

weight to grid points in the region within eight nats of the peak log-density, and divide by the range of the exact energy over that region. This relative fit error includes additive energy offsets. To compare distributional shape, we subtract the mean energy error at each state before computing the same relative fit error. This centering is used only for the shape diagnostic; Figure 1(b) reports the original errors.

Table 6 summarizes normalization across the 11 evaluation states.

Table 6: Energy normalization across the evaluation states. The standard deviation measures variation across states within a single training run.

	Mean $\log Z(s)$	Std. $\log Z(s)$	Max. $\lvert \log Z(s) \rvert$
With anchor	-0.016	0.014	0.046
Without anchor	$+0.675$	0.291	1.024

D.4 Tracking error against a moving target

The learned energy never converges to the negative log-density of a target distribution that keeps changing. Its error settles above the error floor for a static target and stays there, by a factor of 1.22 to 3.09 over twelve conditions, at 6.9 to 26.1 standard errors. The excess falls with the number of energy steps per policy change and rises with the speed of the change, monotonically in both. So the actor reads a standing tracking error rather than a decaying one, which is not what the transient alone suggests.

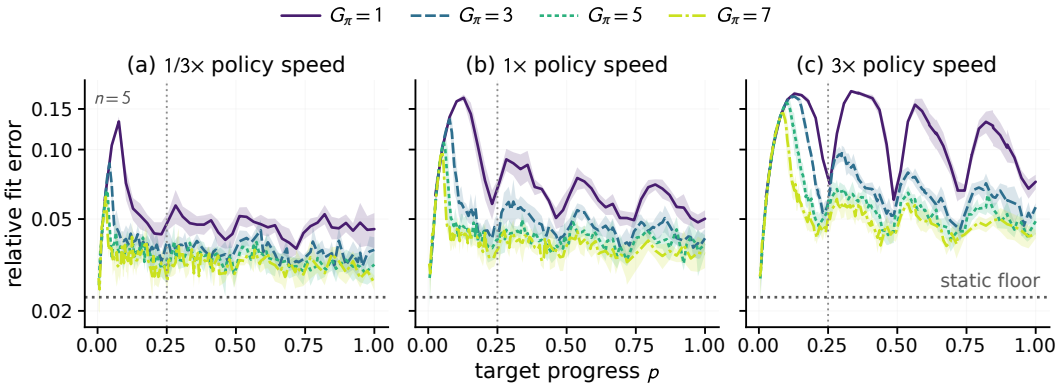


Figure 8: Tracking a moving target at three speeds, $d = 2$, five seeds, bands ± 1 standard deviation. One speed per panel, relative fit error against the target's position. The dotted vertical marks where the variance and weight channels freeze into rigid rotation, the horizontal one the error floor for a static target at the same training budget.

Measuring that needs the exact $-\log \pi_\theta$ along the whole trajectory. So the policy is replaced by a target that moves on a schedule, and G_π becomes the number of energy steps per move. Four periods and three speeds run at $d = 2$, five seeds each, with the deployed objective, corruption instance and variance-preserving schedule. The control is a cell whose target never moves, run to the same budget at each frozen position. Its converged error is 0.0229 ± 0.0004, flat to ± 0.0012 across those positions, so it measures the fit and not the place.

The target rotates its four modes through π, two mode spacings, and shrinks every component variance threefold. Progress along that path is $p \in [0, 1]$. The variance and mixture weight channels finish at $p = 1/4$ and then hold, so the last three quarters are rigid rotation at constant angular velocity: a co-moving steady state rather than a transient. That is what makes a converged tracking error meaningful at all. After $p = 1$ the target stops and each arm trains on it for the rest of its budget.

The speed is calibrated rather than chosen. On a G1 joystick task the shipped policy's action spread moves by 1.55×10^{-4} per energy step, at the median of 975 logged intervals. Spreading the target's

variance channel over 3,549 moves per quarter turn reproduces that rate to 0.2% at the deployed period $G_\pi = 2$. The same path is walked at three times and at a third of that speed, chosen so that two pairs of cells share the drift per energy step exactly, one over the move count times the period, while the step size differs threefold. Matching travel rather than nats is the conservative direction: at $d = 2$ the channel moves 14.5 times fewer nats per update than the $d = 29$ policy it is calibrated against.

Figure 8 shows every arm settling above that floor and staying there. The excess runs from $+0.0051 \pm 0.0006$ at the slow speed with $G_\pi = 7$ to $+0.0480 \pm 0.0029$ at the fast speed with $G_\pi = 1$. Nine of the twelve cells are still falling slowly over the co-moving phase, so these are upper bounds on the asymptote, but the amount left to fall is small next to the excess. The motion causes it rather than an exhausted budget. When the target stops, the $G_\pi = 1$ cells give up 101%, 87% and 91% of their excess at the slow, deployed and fast speeds. And at 40, 60, 80 and 100% of one shared absolute budget, an arm still turning sits above an arm that has stopped by $+0.004$ to $+0.020$, positive in all twelve comparisons at 2.9 to 18.1 standard errors.

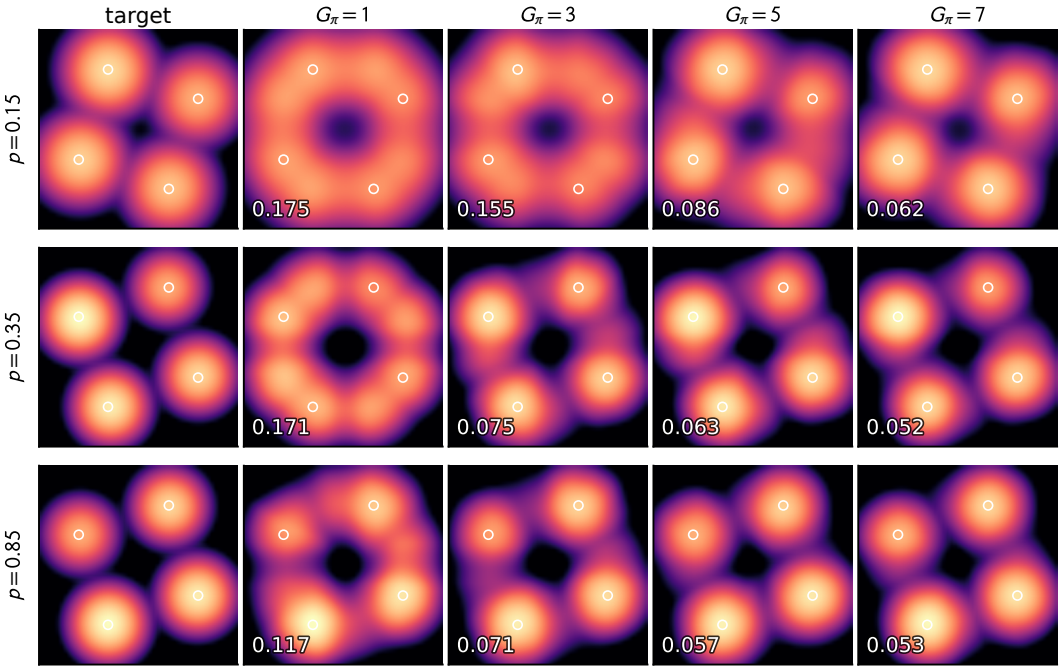


Figure 9: Learned $-U_\psi$ against the rotating target at three times the deployed speed, $d = 2$, one seed, one state, colour shared across panels. Rings mark the target's current modes; the number in each learned panel is its relative fit error. The first row is on the rising limb, the other two in the co-moving steady state, and all three sit near the worst phase of Figure 8(c)'s oscillation.

Figure 9 shows what the period buys as shape, drawn at three times the deployed speed. At the deployed speed the two longest periods do not separate in shape at all: their weakest barriers between modes and saddles differ by at most 0.19 nats at $p \in \{0.1, 0.5, 0.8\}$, and at 0.8 the order inverts, 3.94 for $G_\pi = 5$ against 3.75 for $G_\pi = 7$. Tripling the speed orders all four arms on every row. On the rising limb the target separates each mode from the saddle beside it by 2.73 nats at its weakest pair; $G_\pi = 1$ and 3 keep 0.39 and 0.44 of that and read as rings, while $G_\pi = 5$ keeps 1.68 and $G_\pi = 7$ keeps 2.46. A ring assigns nearly the same energy to the modes and to the saddles between them, which is the failure a Bellman entropy bonus cannot tolerate, and the two short periods give it up rather than holding the right shape at a lag. $G_\pi = 1$ is still a ring once the target is co-moving, 0.38 against its 4.30 at $p = 0.35$ where the other three hold 2.59, 2.98 and 3.39, and it recovers only 1.40 of 4.38 by $p = 0.85$. The fit errors order the same way on all three rows, from 0.175 down to 0.052.

These are the rows where the speed separates the arms most, not typical ones. The error here carries a periodic component with the target's own quarter-turn period, the oscillation in Figure 8(c), and all three rows sit near its worst phase: over five seeds the $G_\pi = 1$ to 7 spread is 0.112, 0.121 and 0.075 at these positions against 0.017 at $p = 0.25$. At one seed the ordering carries over and the level does

not, six of these twelve cells sitting more than one seed standard deviation from their mean across five seeds.

At this resolution the excess depends on that drift rate alone, not on the granularity as well. Two pairs of cells share it to the last bit. They agree to 0.4 and 0.1 standard errors over the moving phase, to 0.1 and 0.6 over the co-moving segment, and to 1.9 each on the converged level. Those statistics are averages in p rather than over logged events: the two cells of a pair sample the same interval of the path at different densities, which puts up to 6.0 standard errors of quadrature difference into an unweighted event mean.

The round fits no exponent for how the excess grows with speed. Progress is the elapsed energy steps times that rate, so the steps spent co-moving and the angular speed are anticorrelated exactly -1 in logarithms across this ladder. No slice of the design varies the speed at a fixed position and a fixed budget, so a slope in the rate would describe the ladder rather than measure a response. Separating the two needs a ladder in the path length at a fixed move count. It also draws no lag curve. Lag is read as the target position whose negative log-density best matches the current energy estimate, on a grid of 0.039 radians. Eleven of the twelve cells fall below one grid step over the co-moving phase, so a lag panel would show the instrument rather than the run; those eleven are recorded as bounds. The lag that *is* resolvable, on the rising limb, replicates the round with a shorter path in all twelve cells to within 1.8 standard errors.

The setting bounds this subsection more than the statistics do. It is at two action dimensions, where the log-density is drawable. The target is imposed rather than produced by the policy, so nothing feeds back from the estimate to the thing being estimated. And the estimate is scored against $-\log \pi_\theta$ rather than through a return. Section D.5 closes the first two by putting the energy in the loop at $d = 29$, at the cost of a policy class that is no longer the one-step pushforward actor.

D.5 What the actor reads while the policy moves

The period is worth its cost at every rung of the closed loop. Against an oracle arm given the exact entropy, where 0 is the oracle, the objective gap in Table 7 runs from -125.5 ± 0.7 at $G_\pi = 1$ to -3.2 ± 0.4 at $G_\pi = 7$, and the width response the actor consumes rises over the same ladder from 0.0002 to 0.921, where 1 is exact. More energy steps per policy change leave more of that response intact, monotonically.

Here the policy is a Gaussian mixture with one independent mixture per state, so $-\log \pi_\theta$ can be evaluated exactly and differentiated through reparameterized samples, allowing an exact-entropy control. Two arms then differ in one thing, where the entropy in (16) comes from: the oracle uses $-\mathbb{E}[\log \pi_\theta]$ and the energy arm uses $\mathbb{E}[U_\psi]$. The reward, the initialization, the seeds, the number of policy updates, the reparameterized samples and a shared energy warm start are common to both. The action dimension is 29, matching the joystick tasks, and the policy's drift per update is calibrated against a measured run rather than chosen. Five seeds.

What the actor consumes when it resists collapse is not the level of U_ψ but its response to width. Widening every coordinate of π_θ by a factor e^δ raises the exact differential entropy by $d\,\delta$ to first order, so we report

$$\Gamma = \frac{\mathbb{E}_{a' \sim \pi_\theta^{(\delta)}}\big[U_\psi(s, a', t_{\min})\big] - \mathbb{E}_{a \sim \pi_\theta}\big[U_\psi(s, a, t_{\min})\big]}{d\,\delta}, \tag{47}$$

where $\pi_\theta^{(\delta)}$ is that widened policy. For $U_\psi = -\log \pi_\theta$ this is 1 in the same limit: the right-hand side is then the derivative of a cross entropy whose Kullback-Leibler part is stationary at the policy itself.

At the first policy update every arm reads Γ between 0.903 and 0.906, so what follows is attributable to the period alone. By update $4{,}000$ of $40{,}000$ they are already ordered by it, from 0.365 at $G_\pi = 1$ to 0.672 at $G_\pi = 7$. Figure 10 shows what happens after. Every arm dips, and every arm but $G_\pi = 1$ then turns and climbs back, later the shorter the period: $G_\pi = 7$ bottoms at 0.627 by update $14{,}000$, while $G_\pi = 1$ never recovers and leaves the policy about 129 nats of entropy below the oracle. So the energy-based entropy response does recover while the policy moves when enough energy steps are taken per policy change, and which of the two happens is visible within the first tenth of the run.

Neither mechanism that fits a version of this experiment with two arms survives the ladder. The energy estimate cannot resolve the negative log-density of a policy narrower than $\sqrt{t_{\min}}$, and the

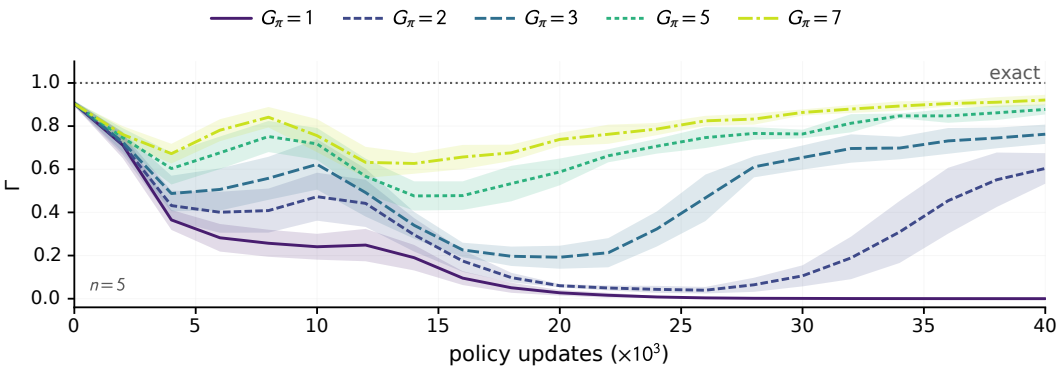


Figure 10: Gain in the width derivative Γ against policy updates, closed loop at $d = 29$, five seeds, bands ± 1 standard deviation. Every arm starts near 0.9 and dips; all but $G_\pi = 1$ recover, later the shorter the period. The dotted line is the exact value for $U_\psi = -\log \pi_\theta$.

Table 7: Closed loop at $d = 29$, five seeds, against an oracle arm given the exact entropy. Gap in expected reward plus τ times entropy, paired by seed. Final Γ.

G_π	1	2	3	5	7
objective gap	-125.5 ± 0.7	-21.3 ± 1.9	-13.1 ± 0.8	-5.3 ± 0.5	-3.2 ± 0.4
Γ	0.0002	0.604	0.762	0.877	0.921

collapsing arm does end well inside that region. But $G_\pi = 2$ also enters that region, reaching 0.69 of $\sqrt{t_{\min}}$ with its response down at 0.039, and returns in five seeds of five. A sweep over the drift rate at $G_\pi = 1$ closes the other direction: three seeds whose width never fell below 0.72, 0.81 and 1.08 of $\sqrt{t_{\min}}$ collapsed anyway, the last without entering that region at all. No threshold in how far the policy narrows separates the outcome.

Table 7 holds the number of actor updates fixed, so the arm at period G_π also takes G_π times as many energy steps. Raising the period spends one resource to buy another, and no sweep holds both fixed, so both edges are run. Pinning the energy steps instead at $80{,}000$, so that $G_\pi = 1$ takes twice the actor updates, the gaps to the oracle are -145.0 ± 4.5 and -33.4 ± 2.4 at $G_\pi = 1$ and 2, paired at each arm's own update count over five seeds. The period is worth more than the actor updates it costs at both edges.

The gap at the $G_\pi = 2$ edge is a lower bound rather than an estimate. That arm's objective is still rising when its $40{,}000$ updates run out, at 3.5 per thousand updates against 0.03 for the $G_\pi = 1$ arm. And a replicate of the same training path on different hardware moves the statistic by about 8 objective units there, twice the standard error quoted within a single job, so a gap of that order is not resolvable.

Neither edge attributes the effect to granularity rather than to the drift rate. The policy's total travel is the drift per energy step times the number of energy steps, with no G_π in it, so matching both would match the motion at every period and leave only the interleaving. But that drift rate is an outcome of the run rather than a setting: in the pair designed to match it the two measured rates still differ by 5%, enough on its own to account for most of the separation. Under a scripted target, where the rate is exact, the two matched pairs of Section D.4 agree across a threefold range of interleaving, which makes the rate difference the simplest reading here. Deciding it in the closed loop takes a rate ladder rather than a single pair.

Three distances from the deployed algorithm remain. The policy is a mixture rather than the one-step pushforward actor $a = f_\theta(s, \varepsilon)$ of Section 4, which is what makes the exact control possible. One reward landscape and one temperature are used throughout. And the estimate is scored against $-\log \pi_\theta$ rather than through a return, so this measures what the period does to the estimate the actor reads, not what it is worth in control, which is Section 5's job.