# SCORE-BASED ONE-STEP MEANFLOW POLICY OPTIMIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Diffusion and flow matching have emerged as expressive policy classes in reinforcement learning, but their reliance on multi-step denoising imposes substantial computational overhead at inference time, which is particularly problematic in online RL. MeanFlow offers a promising alternative by learning an average velocity field that maps noise to data in a single network evaluation. However, MeanFlow requires samples from the target distribution to construct its target velocity field, which are unavailable in online RL, and existing workarounds that use best-of-$N$ selections as training targets can only reinforce actions the current policy already produces. We propose Score-Based One-step MeanFlow Policy Optimization (SOM), an actor-critic algorithm that constructs the target velocity field directly from Q-function gradients via score estimation and the probability flow ODE, thereby transporting noise toward high-value modes. To stabilize score estimation under an evolving critic, we adapt the inverse temperature to control candidate-weight concentration via the effective sample size (ESS) across diffusion times, and anneal the target ESS over training. In fully online RL, SOM achieves the highest normalized return across five MuJoCo locomotion benchmarks with a single generation step, while substantially reducing inference latency compared to diffusion-based policies.

## 1 INTRODUCTION

Reinforcement learning in continuous action spaces has long been dominated by Gaussian policies, yet their unimodal nature fundamentally limits expressivity: they cannot capture the complex, multimodal action structure that arises in many control tasks. To overcome this, recent works have adopted generative models as policy classes. In particular, diffusion models (Ho et al., 2020; Song et al., 2021) and flow matching (Lipman et al., 2023) have emerged as promising alternatives to Gaussian policies in RL (e.g., Wang et al., 2023; Chi et al., 2025; Wang et al., 2024; Psenka et al., 2024; Ma et al., 2025). These approaches leverage the expressivity of generative models to represent complex, multimodal action distributions, which can enable richer exploration and capture diverse optimal behaviors.

Despite these advantages, such policies require multiple, iterative network evaluations to generate a single action. This introduces significant computational overhead at inference time, which is particularly problematic in online RL, where policy inference is continuously interleaved with data collection. While various approaches have been proposed to reduce the number of sampling steps (e.g., Song et al., 2023; Frans et al., 2025), MeanFlow (Geng et al., 2026) has recently emerged as a particularly promising remedy. Whereas flow matching learns an instantaneous velocity field that must be integrated over multiple discretization steps, MeanFlow learns the average velocity over a time interval, allowing noise to be mapped to data in a single network evaluation.

However, directly applying MeanFlow to online RL is not straightforward. Like flow matching, MeanFlow is trained by regressing onto a velocity constructed from samples of the target distribution. In online RL, the desired policy—one that concentrates on high-value actions—is defined only implicitly through the learned Q-function; replay-buffer actions reflect past behavior policies rather than this improved target, and no samples from it are directly available. A recent approach, MVP (Zhan et al., 2026), sidesteps this by treating the highest-valued of several candidate actions



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.

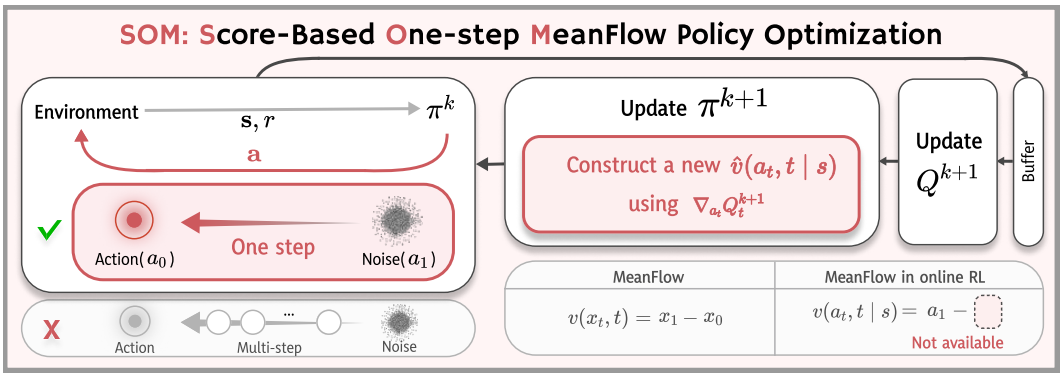



Figure 1: **Overview of SOM.** SOM generates actions in a single step, unlike multi-step diffusion and flow-matching policies. Because target-distribution samples are unavailable in online RL, SOM constructs the MeanFlow target velocity from the gradient of the current Q-function.

as imitation targets, but such best-of-$N$ targets can only reinforce actions the current policy already produces and may scale poorly as action dimensionality grows.

In this work, we propose **Score-Based One-step MeanFlow Policy Optimization (SOM)**, which constructs the target velocity field directly from the Q-function. Taking the Boltzmann distribution $\pi(a \mid s) \propto \exp(\alpha Q(s, a))$ as the target, we use the probability flow ODE to express the target velocity through the scores of its noise-perturbed marginals. These scores are intractable, so we estimate them using the Monte Carlo estimator of iDEM (Akhound-Sadegh et al., 2024), originally proposed for sampling from Boltzmann densities, which aggregates critic gradients at sampled candidate actions weighted by their Q-values. Unlike best-of-$N$ selection, the resulting target exploits gradient information and can move the policy beyond the support of its current samples. Our main contributions are as follows:

- We propose SOM, an actor-critic algorithm that trains one-step MeanFlow policies in online RL by constructing the target velocity field from Q-function gradients via score estimation and the probability flow ODE, rather than from best-of-$N$ target samples.
- We introduce an effective-sample-size (ESS) based scheme that adapts the inverse temperature to directly control candidate-weight concentration in the score estimator across diffusion times, and anneals the target ESS over training to account for the evolving reliability of the critic.
- SOM achieves the highest normalized return across five MuJoCo locomotion benchmarks, with the largest gain on the high-dimensional Humanoid task, while generating each action in a single step and substantially reducing inference latency compared to diffusion-based policies.

## 2 Related Work

**Generative policies in online RL.** In online RL, samples from the target policy are unavailable, and existing diffusion policies differ in how the critic's signal enters training: QVPO (Ding et al., 2024) optimizes a Q-weighted variational bound, DIPO (Yang et al., 2023) fits actions improved along $\nabla_a Q$, and DACER (Wang et al., 2024) backpropagates the Q-value objective through the reverse diffusion process. Closest to our approach, QSM (Psenka et al., 2024) regresses the policy's score onto $\alpha \nabla_a Q$ at the noisy action, while SDAC and DPMD (Ma et al., 2025) reweight denoising score matching with exponentiated Q-values, with SDAC recovering the score of the noise-perturbed Boltzmann marginal from Q-values alone. SOM targets the same marginal score, but estimates it from critic gradients at candidate actions, using self-normalized weights whose concentration is controlled through the effective sample size.

**One-step generative policies.** These prior methods all require iterative multi-step denoising at inference, which is costly in online RL. MVP (Zhan et al., 2026) achieves one-step action generation

with a MeanFlow policy (Geng et al., 2026), trained to imitate the highest-valued of $N$ candidate actions sampled from the current policy, so that policy improvement relies solely on best-of-$N$ selection. SOM also uses a one-step MeanFlow policy, but trains it toward a target velocity derived from Q-function gradients, allowing it to move beyond the support of its current samples.

# 3 Background

## 3.1 Reinforcement Learning

We consider a Markov decision process $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma \rangle$ with state space $\mathcal{S}$, continuous action space $\mathcal{A}$, transition kernel $\mathcal{P}(s' \mid s, a)$, reward function $r$, initial state distribution $\rho_0$, and discount factor $\gamma \in [0, 1)$ (Sutton et al., 1998). We seek a policy $\pi$ maximizing the expected discounted return $J(\pi) = \mathbb{E}_\pi \left[\sum_{\ell=0}^{\infty} \gamma^\ell r(s_\ell, a_\ell)\right]$, and the action-value function $Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{\ell=0}^{\infty} \gamma^\ell r(s_\ell, a_\ell) \mid s_0 = s, a_0 = a\right]$ is the return of taking $a$ in $s$ and then following $\pi$. Actor-critic methods alternate between estimating $Q^\pi$ and improving $\pi$ with respect to it. Analogous to soft policy iteration (Haarnoja et al., 2018), we take the improved policy to be the Boltzmann distribution induced by the current critic, $\pi_{\text{new}}(a \mid s) \propto \exp\big(\alpha Q^{\pi_{\text{old}}}(s, a)\big)$, where the inverse temperature $\alpha > 0$ controls how strongly the policy concentrates on high-value actions.

## 3.2 VE SDE and Probability Flow ODE

Score-based diffusion models perturb data with a forward SDE $dx_t = f(t)x_t\, dt + g(t)\, dw_t$, where $w_t$ is a standard Wiener process (Song et al., 2021). We use the variance-exploding (VE) SDE, which sets $f(t) = 0$ and $g^2(t) = \frac{d\sigma^2(t)}{dt}$ for an increasing noise scale $\sigma(t)$ on $t \in [0, 1]$. With $h(t) = \int_0^t g^2(\tau)\, d\tau = \sigma^2(t) - \sigma^2(0)$, its Gaussian perturbation kernel is $p_t(x_t \mid x_0) = \mathcal{N}(x_t; x_0, h(t)\mathbf{I})$. Every such SDE admits a deterministic probability flow ODE whose trajectories share the same marginals $p_t$ (Song et al., 2021); for the VE SDE, its velocity is $v(x_t, t) = -\frac{1}{2}\frac{d\sigma^2(t)}{dt}\nabla_{x_t} \log p_t(x_t)$. A velocity field transporting between noise and data is therefore fully determined by the scores of the perturbed marginals, which is the connection SOM exploits to construct target velocities from the critic.

## 3.3 MeanFlow

The probability flow ODE defines an instantaneous velocity field, so generating a sample requires integrating it over many steps. MeanFlow (Geng et al., 2026) instead models the *average velocity* over an interval $[r, t]$, $u(x_t, r, t) \triangleq \frac{1}{t-r}\int_r^t v(x_\tau, \tau)\, d\tau$, where $v$ is the velocity field of a flow that transports data at $t = 0$ to noise at $t = 1$. Multiplying both sides by $(t - r)$ and differentiating with respect to $t$ yields the MeanFlow identity $u(x_t, r, t) = v(x_t, t) - (t-r)\frac{d}{dt}u(x_t, r, t)$, where the total derivative $\frac{d}{dt}u = v(x_t, t)\,\partial_{x_t}u + \partial_t u$ is computed with a Jacobian-vector product. A network $u_\theta$ is trained to satisfy this identity by minimizing

$$\mathcal{L}(\theta) = \mathbb{E}_{t,r,x_t}\big\|u_\theta(x_t, r, t) - \text{sg}(u_{\text{tgt}})\big\|_2^2, \quad u_{\text{tgt}} = v(x_t, t) - (t - r)\big(v(x_t, t)\,\partial_{x_t}u_\theta + \partial_t u_\theta\big), \quad (1)$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator, and a sample is then generated in a single evaluation as $x_0 = x_1 - u_\theta(x_1, 0, 1)$ from a prior sample $x_1$. Constructing $u_{\text{tgt}}$ requires the velocity $v(x_t, t)$; standard MeanFlow replaces it with the conditional velocity $\epsilon - x_0$ of the linear path $x_t = (1 - t)x_0 + t\epsilon$ (Lipman et al., 2023), which requires samples $x_0$ from the target distribution.

## 3.4 Iterated Denoising Energy Matching (iDEM)

iDEM (Akhound-Sadegh et al., 2024) targets distributions defined by an energy function, $p_0(x) \propto \exp(-E(x))$, accessible only through $E$ and $\nabla E$ rather than samples. Under the VE kernel, the posterior is $p(x_0 \mid x_t) \propto \exp(-E(x_0))\,\mathcal{N}(x_0; x_t, h(t)\mathbf{I})$, and the score of the perturbed marginal satisfies $\nabla_{x_t} \log p_t(x_t) = \mathbb{E}_{p(x_0|x_t)}[-\nabla E(x_0)]$. iDEM estimates this expectation by self-normalized

importance sampling with the Gaussian proposal $\mathcal{N}(x_t, h(t)\mathbf{I})$:

$$\nabla_{x_t} \log p_t(x_t) \approx \sum_{i=1}^{K} w_i \big( -\nabla E(x_{0|t}^{(i)}) \big), \qquad w_i = \frac{\exp(-E(x_{0|t}^{(i)}))}{\sum_{j=1}^{K} \exp(-E(x_{0|t}^{(j)}))}, \tag{2}$$

where $x_{0|t}^{(i)} = x_t + \epsilon^{(i)}$ with $\epsilon^{(i)} \sim \mathcal{N}(0, h(t)\mathbf{I})$ denotes a candidate for the clean sample $x_0$ proposed around $x_t$. The query point $x_t$ can be arbitrary and need not be generated from a sample of $p_0$.

## 4 Score-Based One-step MeanFlow Policy Optimization

We now construct the MeanFlow target velocity directly from the critic, without samples from the target distribution. Throughout this section, we instantiate the generic variable $x$ of Section 3 with the action $a$ conditioned on the state $s$; the subscript $t \in [0, 1]$ denotes diffusion time, distinct from the environment step $\ell$. We first define the noise-perturbed marginals of the Boltzmann target (Section 4.1) and estimate their scores from the critic (Section 4.2), then convert these scores into a target velocity via the probability flow ODE (Section 4.3). We further adapt the inverse temperature through an ESS-based controller (Section 4.4) and describe implementation details (Section 4.5).

### 4.1 Time-Dependent Energy Function

The MeanFlow objective (Equation (1)) requires the velocity $v(a_t, t \mid s)$, which standard training obtains from target-distribution samples. In online RL, no such samples exist: the desired policy is defined only implicitly through the critic and shifts as the critic is updated. Rather than imitating best-of-$N$ selections from the current policy, we derive the target velocity directly from the critic's gradient, which can move the policy toward high-value actions beyond those it currently samples.

Specifically, we take as the target the Boltzmann distribution of Section 3.1 induced by the current critic, $p_0(a_0 \mid s) = \exp(\alpha Q(s, a_0))/Z(s)$ with $Z(s) = \int \exp(\alpha Q(s, a_0))\, da_0$, where the inverse temperature $\alpha > 0$ determines how sharply the target concentrates on high-value actions. Its score is available in closed form as $\nabla_{a_0} \log p_0(a_0 \mid s) = \alpha \nabla_{a_0} Q(s, a_0)$. The probability flow ODE, however, requires the scores of the noise-perturbed marginals $p_t$ at every $t \in [0, 1]$, which we characterize next through a time-dependent energy function that preserves the Boltzmann structure of the target across timesteps.

Under the VE kernel $p_t(a_t \mid a_0) = \mathcal{N}(a_t; a_0, h(t)\mathbf{I})$ of Section 3.2, which gradually corrupts a clean action $a_0$ into Gaussian noise as $t$ increases, the induced density at time $t$ is

$$p_t(a_t \mid s) = \int p_t(a_t \mid a_0)\, p_0(a_0 \mid s)\, da_0 = \frac{1}{Z(s)} \int p_t(a_t \mid a_0) \exp\big(\alpha Q(s, a_0)\big)\, da_0.$$

We accordingly define the time-dependent energy function

$$Q_t(s, a_t) = \log \int p_t(a_t \mid a_0) \exp\big(\alpha Q(s, a_0)\big)\, da_0, \tag{3}$$

which smooths the scaled critic $\alpha Q$ with the Gaussian kernel and reduces to $\alpha Q(s, a_t)$ as $h(t) \to 0$. Under this definition, the density at time $t$ retains the Boltzmann form $p_t(a_t \mid s) = \exp(Q_t(s, a_t))/Z(s)$ with the same normalization constant, so its score reduces to $\nabla_{a_t} \log p_t(a_t \mid s) = \nabla_{a_t} Q_t(s, a_t)$. Computing $Q_t$ itself is generally intractable, as it requires the log-expectation of exponentiated Q-values under the forward noising kernel. Our objective, however, only requires its gradient $\nabla_{a_t} Q_t(s, a_t)$, which is sufficient to construct the probability flow velocity field and can be approximated directly from the critic without ever evaluating $Q_t$.

### 4.2 Estimation of the Time-Dependent Energy Gradient

The Boltzmann target of Section 4.1 is an energy-based distribution with energy $E(s, a) = -\alpha Q(s, a)$, so the iDEM estimator of Section 3.4 applies directly. Under the VE kernel, the posterior over clean actions is $p(a_0 \mid a_t, s) \propto \exp(\alpha Q(s, a_0))\, \mathcal{N}(a_0; a_t, h(t)\mathbf{I})$, and the score identity gives $\nabla_{a_t} Q_t(s, a_t) = \mathbb{E}_{p(a_0 \mid a_t, s)}[\alpha \nabla_{a_0} Q(s, a_0)]$, that is, the posterior average of the critic gradient

over clean actions consistent with $a_t$. Estimating this expectation by self-normalized importance sampling with $K$ candidates drawn from $\mathcal{N}(a_t, h(t)\mathbf{I})$ yields

$$\nabla_{a_t} Q_t(s, a_t) \approx \alpha \sum_{i=1}^{K} w_i \, \nabla_a Q(s, a_{0|t}^{(i)}), \qquad w_i = \frac{\exp\left(\alpha Q(s, a_{0|t}^{(i)})\right)}{\sum_{j=1}^{K} \exp\left(\alpha Q(s, a_{0|t}^{(j)})\right)}, \tag{4}$$

where $a_{0|t}^{(i)} = a_t + \epsilon^{(i)}$ with $\epsilon^{(i)} \sim \mathcal{N}(0, h(t)\mathbf{I})$. The estimator requires only critic values and action gradients at the $K$ candidates, and no samples from the target distribution. Since $Q_t$ is the logarithm of a Gaussian-smoothed $\exp(\alpha Q)$, the estimator approximates the gradient of a smoothed critic landscape: at large $t$ it aggregates critic gradients over a wide neighborhood of $a_t$, whereas as $h(t) \to 0$ it reduces to $\alpha \nabla_a Q(s, a_t)$. The inverse temperature $\alpha$ determines how strongly the weights $w_i$ concentrate on the highest-valued candidates, which we control in Section 4.4.

### 4.3 Velocity Field and Its Relation to the Score Function

The probability flow ODE of Section 3.2 connects the target velocity to the scores of the noise-perturbed marginals. Since the score of $p_t(a_t \mid s)$ reduces to $\nabla_{a_t} Q_t(s, a_t)$ (Section 4.1), the velocity field transporting the Boltzmann target is

$$v(a_t, t \mid s) = -\frac{1}{2} \frac{d\sigma^2(t)}{dt} \nabla_{a_t} Q_t(s, a_t), \tag{5}$$

so that the transport is determined entirely by the gradient of the time-dependent energy. In practice, we replace $\nabla_{a_t} Q_t$ with its estimate in Equation (4), denoting the resulting velocity by $\hat{v}(a_t, t \mid s)$, and substitute it for $v$ in the MeanFlow target of Equation (1), giving $\hat{u}_{\text{tgt}} = \hat{v}(a_t, t \mid s) - (t - r)\big(\hat{v}(a_t, t \mid s)\, \partial_{a_t} u_\theta + \partial_t u_\theta\big)$. The training objective is then

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a_0)\sim\mathcal{D},\,(r,t),\,a_t\sim p_t(a_t|a_0)} \Big[ \big\| u_\theta(a_t, r, t, s) - \mathrm{sg}(\hat{u}_{\text{tgt}}) \big\|_2^2 \Big], \tag{6}$$

where $0 \le r \le t \le 1$ (see Appendix A for time sampling and Appendix H for adaptive L2 weighting), and $a_t$ is obtained by perturbing a replay-buffer action with the VE kernel. Unlike the conditional velocity $\epsilon - x_0$ of standard MeanFlow, which depends on the particular pair $(x_0, \epsilon)$ that generated $x_t$ and therefore recovers the correct field only when $x_0$ is drawn from the target distribution, $\hat{v}(a_t, t \mid s)$ depends only on the query point $a_t$. Training thus reduces to regressing onto a known target field, and $a_t$ need not be drawn from the target marginal; replay-buffer actions perturbed by the VE kernel only determine where this field is fitted, and suffice as long as they cover the region traversed by the flow.

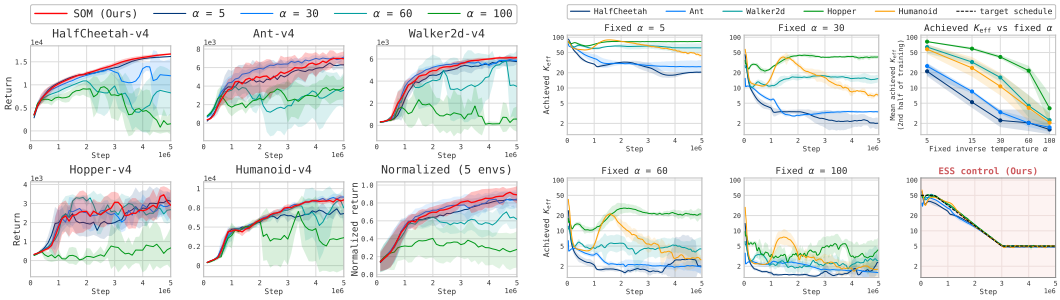
### 4.4 ESS-Based Adaptation of the Inverse Temperature

As noted in Section 4.2, the inverse temperature $\alpha$ determines how strongly relative Q-value differences among the $K$ candidates translate into concentration of the weights $w_i$ in Equation (4). These differences, however, vary across training steps, environments, and diffusion times: the critic changes as it is updated, the Q-landscape differs from task to task, and the proposal variance $h(t)$ grows with $t$, so that candidates at larger $t$ span a wider range of Q-values. Consequently, a single fixed $\alpha$ cannot maintain consistent candidate-weight concentration across training stages, environments, and diffusion times.

**Controlling candidate-weight concentration.** The degree of concentration trades off the strength of the policy-improvement signal against the variance of the estimator. Diffuse weights average over many candidates and yield stable but weak improvement signals, whereas concentrated weights emphasize the highest-valued candidates but rely on few effective samples, increasing finite-sample variance; both extremes degrade learning (Appendix B.3). Since a fixed $\alpha$ does not maintain a consistent level of concentration, we instead control the concentration itself.

We quantify this concentration by the effective sample size (ESS) of the weights (Kish, 1965), which we refer to as the effective number of candidates:

$$K_{\text{eff}}(s, a_t; \alpha) = \frac{1}{\sum_{i=1}^{K} w_i(\alpha)^2} \in [1, K]. \tag{7}$$




(a) Performance with fixed $\alpha$ (b) Concentration under fixed $\alpha$

Figure 2: **ESS control avoids environment-specific inverse-temperature tuning. (a)** Training curves of SOM and of fixed-$\alpha$ variants; the bottom-right panel shows the normalized return averaged over five environments. An $\alpha$ that performs well in one environment can perform poorly in another (details in Appendix B.2). **(b)** Candidate-weight concentration under fixed $\alpha$. The achieved $K_{\text{eff}}$ varies substantially across environments and over training as candidate Q-value differences change; the upper-right panel shows its mean over the second half of training for each $\alpha$. In contrast, ESS control (bottom right) adapts $\alpha$ so that $K_{\text{eff}}$ follows a common target schedule (dashed) across environments. All curves show the mean over five seeds with 95% confidence intervals.

Uniform weights yield $K_{\text{eff}} = K$, whereas domination by a single candidate gives $K_{\text{eff}} \approx 1$.

For fixed, non-degenerate candidate Q-values, $K_{\text{eff}}$ decreases monotonically in $\alpha$, so each target concentration in $(1, K)$ is attained by a unique $\alpha$. The value of $\alpha$ that attains a given concentration, however, depends on the candidate Q-value differences and therefore changes across environments and over training. As shown in Figure 2(a), no single fixed $\alpha$ performs consistently across environments, and Figure 2(b) offers an explanation: for the same fixed $\alpha$, the achieved $K_{\text{eff}}$ varies substantially across environments and throughout training. ESS control instead specifies the desired concentration directly and adapts $\alpha$ to realize it. Indeed, a single fixed target $K_{\text{eff}}$ shared across all environments performs comparably to selecting a fixed $\alpha$ separately for each environment in most cases (Appendix B.2). ESS thus provides a more transferable control variable: rather than tuning $\alpha$ for each environment, we specify the desired concentration in ESS space and let the inverse temperature absorb differences in candidate Q-values.

**Diffusion-time-dependent adaptation.** Because candidate Q-value differences also vary with diffusion time, we partition $[0, 1]$ into $B$ equal-width bins, shared across all environments, and maintain an independent inverse temperature $\alpha_{n,b}$ for each bin, where $n$ indexes gradient updates. Let $\overline{K}_{\text{eff},n,b}$ denote the mean ESS over the minibatch samples whose diffusion time falls in bin $b$, and let $K^{\star}_{\text{eff}}(n)$ denote the target ESS, scheduled as described below. We update

$$\log \alpha_{n+1,b} = \log \alpha_{n,b} + \eta \, \frac{\overline{K}_{\text{eff},n,b} - K^{\star}_{\text{eff}}(n)}{K^{\star}_{\text{eff}}(n)}, \tag{8}$$

so that $\alpha$ increases when the observed ESS exceeds the target and decreases otherwise, controlling candidate-weight concentration independently across diffusion-time bins. Using a separate inverse temperature per bin means that the resulting velocity field is no longer the probability flow of a single Boltzmann target; we treat this as a practical approximation that keeps the score estimator well conditioned across noise levels (see Appendix C for ESS diagnostics and bin-count ablations, and Appendix C.3 for an exact-velocity toy analysis of terminal mode allocation). In practice, we clip $\alpha$ to a fixed range for numerical stability (Appendix H).

**Scheduling the target during training.** A fixed target keeps the candidate-weight concentration constant throughout training. We instead use a high-to-low target schedule shared across environments, motivated by the expectation that critic estimates are less reliable early in training. A high target initially spreads weight over many candidates, limiting the influence of individual overestimated actions on the score estimate; as training progresses, a lower target places greater weight on high-value candidates and sharpens the policy-improvement signal. The schedule thus encodes a prior over training stages rather than measuring critic reliability or adapting the target to it. Specifically, we hold $K^{\star}_{\text{eff}}$ at $K/2$ during an initial warm-up, anneal it geometrically (linearly in log ESS)

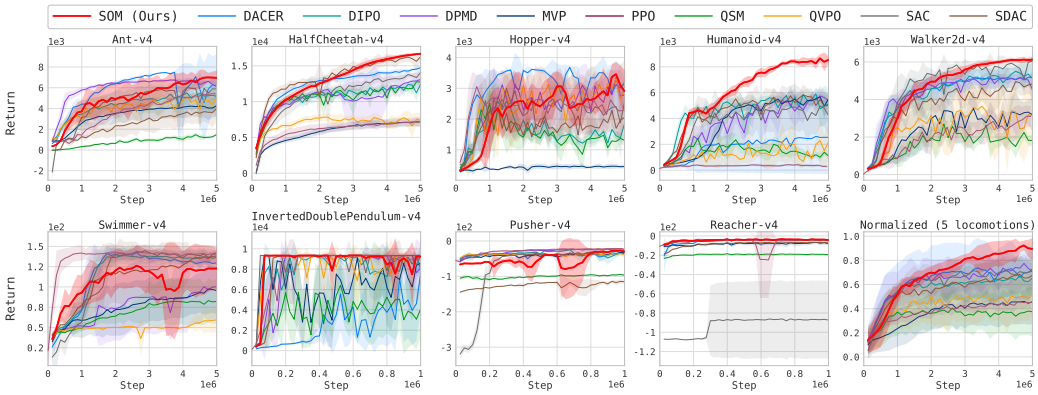



Figure 3: **Training curves on MuJoCo benchmarks.** Curves show the mean over five random seeds, with shading indicating 95% confidence intervals. The bottom-right panel shows the return min–max normalized over all methods and evaluation points per environment, averaged over the five locomotion tasks.

to $K/20$ over an intermediate portion of training, and keep it fixed thereafter. We compare fixed targets and the reversed schedule in Appendix B.3 and Appendix B.4.

### 4.5 Implementation Details

**Critic.** We train the critic via a distributional Bellman objective with twin Q-networks $Q_{\phi_1}, Q_{\phi_2}$ and corresponding target networks following DACER (Wang et al., 2024).

**Best-of-$N$.** Similar to MVP (Zhan et al., 2026), we wrap the one-step MeanFlow policy with best-of-$N$ selection, choosing the highest-valued of $N$ single-step candidate actions at each state, and use this policy both for acting and for computing critic targets so that the critic evaluates the executed policy. This selection complements the score-based actor update by re-ranking samples at little additional cost, which can help when gradient-based improvement slows in complex Q-landscapes. The overall procedure is described in Algorithm 1 and illustrated in Figure 1. We evaluate the effect of $N$, including $N = 1$, in Appendix H.4.

## 5 Experiments

### 5.1 MuJoCo Benchmarks

**Benchmarks.** We evaluate on nine tasks from the MuJoCo v4 suite (Todorov et al., 2012; Towers et al., 2024), with action dimensionalities ranging from 1 to 17. Our main comparison uses five locomotion tasks, `Ant-v4`, `HalfCheetah-v4`, `Hopper-v4`, `Humanoid-v4`, and `Walker2d-v4`, for which we report final returns and a normalized score in Table 1; training curves for the remaining four tasks, `Swimmer-v4`, `Pusher-v4`, `Reacher-v4`, and `InvertedDoublePendulum-v4`, are shown in Figure 3. Each algorithm is trained with five random seeds for 5M environment steps, except for 1M steps on `Pusher-v4`, `Reacher-v4`, and `InvertedDoublePendulum-v4`, and we report the mean and 95% confidence intervals.

**Baselines.** We compare against generative and non-generative policy baselines. The generative baselines comprise the diffusion-based actors DACER (Wang et al., 2024), DIPO (Yang et al., 2023), DPMD (Ma et al., 2025), QSM (Psenka et al., 2024), QVPO (Ding et al., 2024), and SDAC (Ma et al., 2025), and the one-step MeanFlow policy MVP (Zhan et al., 2026); since MVP was originally proposed for offline-to-online RL, we train it from scratch in the same fully online setting. The non-generative baselines are SAC (Haarnoja et al., 2018) and PPO (Schulman et al., 2017). All methods follow the training protocol above, with hyperparameters listed in Appendix H, except that PPO, being on-policy, is trained for 5M environment steps on `Pusher-v4`, `Reacher-v4`, and `InvertedDoublePendulum-v4`, with its curves rescaled accordingly in Figure 3.

Table 1: **Final performance on MuJoCo locomotion tasks.** Returns after 5M environment steps, averaged over the last five evaluations (mean ± 95% CI over five seeds). **Bold** and underline mark the highest and second-highest means in each column. Normalized: per-environment min–max normalization using the minimum and maximum over all methods, seeds, and evaluation points in that environment, averaged over the five environments.

| Algorithm | Ant-v4 | HalfCheetah-v4 | Hopper-v4 | Humanoid-v4 | Walker2d-v4 | Normalized |
|---|---|---|---|---|---|---|
| DACER | $5757.4 \pm 3044.9$ | $14458.4 \pm 120.9$ | $3097.4 \pm 556.1$ | $2535.1 \pm 2821.1$ | $5129.5 \pm 507.0$ | 0.693 |
| DIPO | $5263.7 \pm 1377.5$ | $11981.9 \pm 605.4$ | $1517.6 \pm 154.5$ | $5239.9 \pm 552.9$ | $5325.4 \pm 535.2$ | 0.643 |
| DPMD | $6364.0 \pm 295.7$ | $12803.4 \pm 1170.7$ | $2947.4 \pm 366.9$ | $5530.4 \pm 469.2$ | $4977.9 \pm 192.7$ | 0.739 |
| MVP | $4156.1 \pm 1060.5$ | $7134.4 \pm 650.1$ | $460.2 \pm 77.6$ | $5408.4 \pm 137.8$ | $3255.6 \pm 738.7$ | 0.454 |
| PPO | $5338.0 \pm 222.9$ | $7094.7 \pm 478.8$ | $2345.7 \pm 716.4$ | $337.4 \pm 89.7$ | $2926.5 \pm 703.8$ | 0.447 |
| QSM | $1290.6 \pm 64.3$ | $12214.0 \pm 313.9$ | $1264.4 \pm 77.2$ | $1140.5 \pm 135.3$ | $1857.6 \pm 134.0$ | 0.364 |
| QVPO | $4601.6 \pm 662.3$ | $7348.2 \pm 1090.5$ | $3020.9 \pm 224.9$ | $1765.8 \pm 804.4$ | $2915.2 \pm 1320.5$ | 0.501 |
| SAC | $5545.1 \pm 1089.0$ | $13625.4 \pm 301.2$ | $2297.4 \pm 171.4$ | $4532.3 \pm 1483.7$ | $5716.2 \pm 259.3$ | 0.702 |
| SDAC | $3694.7 \pm 1107.9$ | $16034.3 \pm 621.8$ | $1976.6 \pm 259.3$ | $5223.1 \pm 245.6$ | $4822.5 \pm 422.5$ | 0.669 |
| **SOM (Ours)** | $\mathbf{6947.1 \pm 552.5}$ | $\mathbf{16543.7 \pm 165.8}$ | $\mathbf{3122.2 \pm 445.4}$ | $\mathbf{8451.7 \pm 310.1}$ | $\mathbf{6109.3 \pm 126.9}$ | **0.901** |

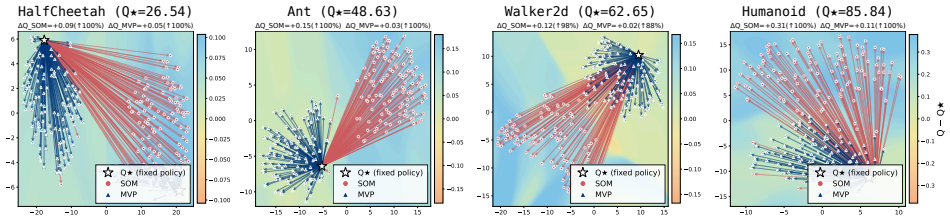



Figure 4: **Effect of the update rule from a shared frozen policy.** t-SNE of actions from SOM (red circles) and MVP (dark-blue triangles), both fine-tuned from the same frozen policy ($\star$, value $Q^\star$) and critic; the background shows $Q - Q^\star$ and panel titles report the mean gain $\Delta Q$. Under the same best-of-32 selection, SOM attains larger gains in all environments. Details in Appendix D.

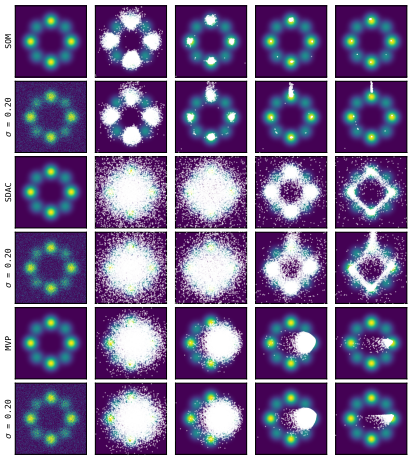
**Results.** Figure 3 shows the full training curves over all 9 tasks, and Table 1 reports final episode returns across the five locomotion benchmarks, together with the average of per-environment normalized performance. SOM achieves the highest mean return among the methods in Table 1 on all five locomotion benchmarks. On `Hopper-v4`, its margin over the next-best method, DACER, is small and their reported 95% confidence intervals overlap. The largest gain occurs on `Humanoid-v4`, where SOM exceeds the second-best method by about 53%.

Humanoid-v4 has the highest action dimensionality, consistent with our intuition that gradient information becomes more beneficial as action dimensionality grows. In addition, Table 2 shows that SOM achieves substantially lower inference latency than all diffusion-based baselines, highlighting the practical advantage of one-step action generation. The table also lists the denoising steps and best-of-$N$ candidates used for inference. Additional ablations on the ESS-based adaptation, including the controller, annealing schedule, target ESS, and number of candidates, are provided in Appendix B. Full implementation details, including hyperparameters, are provided in Appendix H.

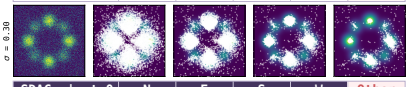
Table 2: **Inference latency on HalfCheetah-v4.** Single observation, mean ± 95% CI over 30 runs. $T_d$: denoising steps; $N_{\text{inf}}$: best-of-$N$ particles. Bold/underline: fastest/second fastest.

| Method | $T_d$ | $N_{\text{inf}}$ | Inference (ms) |
|---|---|---|---|
| SOM | 1 | 32 | $\mathbf{0.198 \pm 0.005}$ |
| QSM | 20 | 32 | $0.606 \pm 0.003$ |
| DPMD | 20 | 32 | $1.036 \pm 0.004$ |
| DIPO | 100 | 1 | $1.562 \pm 0.029$ |
| SDAC | 20 | 32 | $1.012 \pm 0.014$ |
| DACER | 20 | 1 | $0.703 \pm 0.010$ |
| QVPO | 20 | 32 | $1.044 \pm 0.027$ |

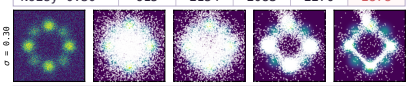
**Isolating the effect of the update rule.** We fine-tune SOM and MVP from the same frozen policy under the same frozen critic, and compare their actions with those of the frozen policy, all drawn with the same best-of-32 selection. As shown in Figure 4, both methods improve on the frozen policy, but SOM attains consistently larger Q gains in all four environments (e.g., $+0.31$ versus $+0.11$ on `Humanoid-v4`). This reflects the difference in update rules: MVP imitates best-of-$N$ selections, which can only reinforce actions the current policy already produces while the $N - 1$ rejected candidates receive no learning signal, whereas SOM's target velocity uses critic gradients and can move the policy beyond its current samples. Details of the setup are provided in Appendix D.




| SOM \| t=0 | N | E | S | W | Other |
|---|---|---|---|---|---|
| Clean | 1994 | 2410 | 2985 | 2594 | 17 |
| Noisy 0.20 | 1171 | 2451 | 3469 | 2681 | 228 |
| Noisy 0.30 | 2512 | 2210 | 2156 | 2522 | 600 |




| SDAC \| t=0 | N | E | S | W | Other |
|---|---|---|---|---|---|
| Clean | 1916 | 1961 | 1893 | 1989 | 2241 |
| Noisy 0.20 | 502 | 2398 | 2313 | 2012 | 2775 |
| Noisy 0.30 | 615 | 2154 | 2083 | 2270 | 2878 |




| MVP \| t=0 | N | E | S | W | Other |
|---|---|---|---|---|---|
| Clean | 0 | 9815 | 0 | 7 | 178 |
| Noisy 0.20 | 0 | 9746 | 0 | 1 | 253 |
| Noisy 0.30 | 0 | 9791 | 0 | 1 | 208 |

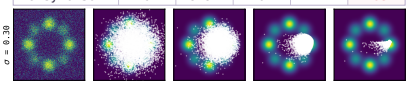



Figure 5: **Mode coverage under clean and noisy reward feedback.** For SOM, SDAC, and MVP (top to bottom), each block pairs clean training with training under Gaussian reward-observation noise ($\sigma = 0.20$, left; $\sigma = 0.30$, right). Each row shows the learned Q-landscape and the action distribution at sampler times $t = 0.75, 0.5, 0.25, 0$. Tables report, out of 10,000 samples at $t = 0$, the counts within a radius of 0.5 of the north, east, south, and west modes, and outside all four (Other). Details in Appendix G.

### 5.2 SOM Behavior on Bandit Environments

**Mode coverage.** We evaluate on a 2D bandit with four equally rewarded high-reward modes (Appendix E). Under clean reward feedback, SOM maps noise to all four modes in a single step, placing 99.8% of its samples near them and allocating them roughly evenly as the Boltzmann target implies. SDAC also covers all four modes but requires a 10-step sampler and leaves many samples between modes (78% near the modes), whereas MVP collapses onto a single mode (98% of samples), consistent with imitating best-of-$N$ selections that reinforce whichever mode the policy already favors.

**Robustness under imperfect critics.** In online RL, the critic is learned from imperfect feedback. To test how the resulting errors in the learned Q-landscape affect the policy, we add Gaussian noise to each reward observation during training. SOM still reaches all four modes (at least 94% of samples near them), whereas SDAC drops to about 72% and becomes more uneven across modes, and MVP remains collapsed (Figure 5; details in Appendix G). This robustness is consistent with the smoothing in SOM's score estimator (Section 4.2): at large $t$, it aggregates critic gradients over a wide neighborhood and averages out local critic errors, whereas at small $t$ it preserves structure near high-value modes.

## 6 Conclusion

We presented SOM, an actor-critic algorithm that trains one-step MeanFlow policies in fully online RL. Rather than imitating best-of-$N$ samples, SOM constructs the target velocity directly from Q-function gradients, estimating the scores of the noise-perturbed Boltzmann target with an iDEM-style estimator and converting them into a velocity field through the probability flow ODE. ESS feedback adapts the inverse temperature to keep candidate-weight concentration on a shared target across environments and diffusion times, while the target schedule gradually sharpens the improvement signal. Empirically, SOM achieves the highest normalized return on five MuJoCo locomotion tasks with a single generation step and substantially lower inference latency than diffusion-based policies.

**Limitations and Future Work.** SOM relies on the critic for both its target and its weighting, yet the target-ESS schedule follows a fixed prior rather than critic reliability; coupling it with uncertainty-aware critics is a natural extension. Extending SOM beyond MuJoCo to visual or partially observable settings, such as real-robot manipulation, is another important next step.

## AI USE STATEMENT

We used LLM-based tools to help identify relevant literature, review mathematical formulations, improve the clarity of the manuscript, and assist with code writing. We also provided research context to generate preliminary section drafts, which served as reference material for author-led writing and revision. The human authors reviewed all AI-assisted work and take full responsibility for the final manuscript, code, and reported results.

## REFERENCES

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, et al. Iterated denoising energy matching for sampling from boltzmann densities. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 760–786, 2024.

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.

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.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OlzB6LnXcS.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=uWj4s7rMnR.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://openreview.net/forum?id=HJjvxl-Cb.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 6840–6851, 2020.

Leslie Kish. *Survey Sampling*. John Wiley & Sons, New York, 1965.

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

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. In *International Conference on Machine Learning*, pp. 41837–41853. PMLR, 2025.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=35ahHydjXo.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 32211–32252, 2023.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

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, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang WU, Jingliang Duan, and Shengbo Eben Li. Diffusion actor-critic with entropy regulator. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=l0c1j4QvTq.

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

Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

Guojian Zhan, Letian Tao, Pengcheng Wang, Yixiao Wang, Yuxin Chen, Yiheng Li, Hongyang Li, Masayoshi Tomizuka, and Shengbo Eben Li. Mean flow policy with instantaneous velocity constraint for one-step action generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mIeKe74W43.

# A Algorithm Pseudocode

**Algorithm 1** Score-Based One-step MeanFlow Policy Optimization

**Function** $\pi(s; u_\theta, Q_\phi, N)$ ▷ Best-of-$N$ action selection
  Sample $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2_{\max}\mathbf{I})$ for $i = 1, \ldots, N$
  $a^{(i)} \leftarrow \epsilon^{(i)} - u_\theta(\epsilon^{(i)}, 0, 1, s)$ for $i = 1, \ldots, N$ ▷ One-step generation
  **return** $a^{(i^*)}$ where $i^* = \arg\max_i Q_\phi(s, a^{(i)})$

**Input:** Policy $u_\theta$, Critic $Q_\phi$, buffer $\mathcal{B}$, Monte Carlo sample size $K$, BoN size $N$
**for** each iteration **do**
  **for** each sampling step **do**
    $a \leftarrow \pi(s; u_\theta, Q_\phi, N)$
    Execute $a$, observe reward $R$ and next state $s'$
    Store transition $(s, a, R, s')$ in buffer $\mathcal{B}$
  **end for**
  **for** each update step **do**
    Sample mini-batch $\{(s, a, R, s')\} \sim \mathcal{B}$
    Update critic $Q_\phi$ following DACER (Wang et al., 2024)
    ▷ *Train MeanFlow policy* $u_\theta$
    Sample $z_1, z_2 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(-0.4, 1)$
    Set $t = \max_i \text{sigmoid}(z_i)$ and $r = \min_i \text{sigmoid}(z_i)$
    Get $a_t$ from Section 3.2
    Estimate $\nabla_{a_t} Q_t(s, a_t)$ using Equation (4)
    Compute $\hat{v}(a_t, t \mid s)$ by substituting the estimate into Equation (5)
    Update $\theta$ using adaptive weighting of Equation (6) (Appendix H)
  **end for**
**end for**
**return** One-step policy $\pi$

# B ESS Control and Hyperparameter Ablations

The ablations in this appendix use the same five Gymnasium MuJoCo v4 locomotion benchmarks as the main comparison (Todorov et al., 2012; Towers et al., 2024): `Ant-v4`, `HalfCheetah-v4`, `Hopper-v4`, `Humanoid-v4`, and `Walker2d-v4`.

## B.1 Removing the Controller and the Annealing Schedule

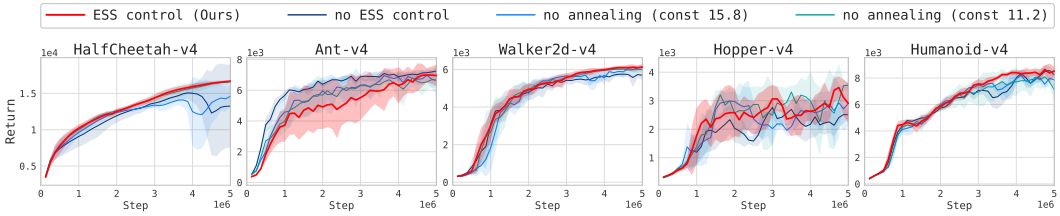



Figure 6: **Ablation of ESS feedback and target annealing.** Learning curves comparing our full ESS adaptation with *no ESS control* (fixed $\alpha = 15$) and *no annealing* (fixed target ESS). Mean over five seeds, shaded 95% CI.

Table 3: **Ablation of ESS feedback and target annealing.** Final performance, measured as the mean of the last five evaluations, ± 95% CI over five seeds. Parentheses give the relative change against Ours.

| | HalfCheetah | Ant | Walker2d | Hopper | Humanoid |
|---|---|---|---|---|---|
| Ours | **16544** ± **166** | 6947 ± 553 | **6109** ± **127** | **3122** ± **445** | **8452** ± **310** |
| no ESS control ($\alpha = 15$) | 12897 ± 5895 (−22.0%) | **7141** ± **167** (+2.8%) | 5699 ± 233 (−6.7%) | 2365 ± 244 (−24.3%) | 8276 ± 261 (−2.1%) |
| no annealing ($K_{\text{eff}} = 15.8$) | 14211 ± 1899 (−14.1%) | 6758 ± 306 (−2.7%) | 6028 ± 149 (−1.3%) | 2814 ± 732 (−9.9%) | 7862 ± 102 (−7.0%) |
| no annealing ($K_{\text{eff}} = 11.2$) | 16460 ± 352 (−0.5%) | 6585 ± 644 (−5.2%) | 5994 ± 131 (−1.9%) | 3071 ± 798 (−1.6%) | 7773 ± 546 (−8.0%) |

As shown in Figure 6 and Table 3, the ESS feedback and the target annealing play complementary roles, and their effects separate by environment. Removing the feedback controller degrades HalfCheetah and Hopper most, by 22% and 24%, and it does so by destabilizing training rather than by lowering a stable plateau: on HalfCheetah one of the five seeds collapses to 926 while the others finish between 14835 and 16573, which is what inflates that entry's confidence interval to ±5895. Freezing the target leaves HalfCheetah largely intact at the lower constant (−0.5% at $K_{\text{eff}} = 11.2$) but widens the seed spread at the higher one (10775–16096 at $K_{\text{eff}} = 15.8$), and it produces a consistent loss on Humanoid for both constants (−7.0% and −8.0%). Ant is the one environment that does not benefit from the controller, gaining 2.8% without it. Adapting $\alpha$ to hold the desired concentration and adapting that concentration over training therefore address distinct sources of sensitivity.

### B.2 A Shared ESS Replaces Environment-Specific $\alpha$ Tuning

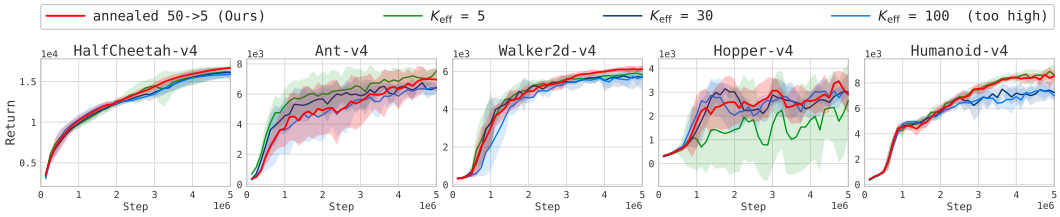



Figure 7: **Shared fixed ESS versus environment-specific inverse-temperature tuning.** Learning curves comparing a single fixed target $K_{\text{eff}}$ shared across all environments with the best fixed $\alpha$ from the tested grid selected separately for each environment. Mean over five seeds, shaded 95% CI.

As shown in Figure 7, a single fixed target $K_{\text{eff}}$ shared across all environments achieves performance comparable to selecting a fixed $\alpha$ separately for each environment. This is notable because the appropriate fixed $\alpha$ varies substantially across environments, as also illustrated in Figure 2. [1] In contrast, the same target ESS can be used across environments while the corresponding inverse temperature is adapted automatically to differences in the scale of candidate Q-values.

These results indicate that ESS serves as a more transferable control variable than the inverse temperature itself: rather than tuning $\alpha$ for each environment, we can specify a common desired candidate-weight concentration in ESS space. In Appendix B.3, we compare this fixed target with the training-dependent schedule across environments.

[1] The bottom-right panel of Figure 2(a) reports return min–max normalized per environment, using the minimum and maximum over all curves shown in that figure (all methods, seeds, and evaluation points in that environment), averaged over the five tasks; shading is the 95% CI across environments.

Table 4: **A single environment-independent ESS target replaces per-environment $\alpha$ tuning.** Final return from the last evaluation of each seed (mean $\pm$ 95% CI over five seeds). Percentages are relative to the per-environment oracle: the best fixed $\alpha$ selected *separately* for each environment.

| | HalfCheetah | Ant | Humanoid | Hopper | Walker2d |
|---|---|---|---|---|---|
| *Fixed $\alpha$, no ESS control — the best $\alpha$ differs per environment* | | | | | |
| $\alpha = 5$ | $\mathbf{16146}_{\pm 113}$ | $6291_{\pm 427}$ | $6752_{\pm 519}$ | $\mathbf{3117}_{\pm 348}$ | $5758_{\pm 213}$ |
| $\alpha = 15$ | $13241_{\pm 5731}$ | $\mathbf{7244}_{\pm 200}$ | $8250_{\pm 328}$ | $2508_{\pm 143}$ | $5695_{\pm 258}$ |
| $\alpha = 30$ | $11950_{\pm 3181}$ | $7132_{\pm 172}$ | $\mathbf{8903}_{\pm 232}$ | $2954_{\pm 565}$ | $\mathbf{5882}_{\pm 197}$ |
| $\alpha = 60$ | $8249_{\pm 4196}$ | $3544_{\pm 1662}$ | $7471_{\pm 2235}$ | $2674_{\pm 572}$ | $3519_{\pm 2825}$ |
| $\alpha = 100$ | $1546_{\pm 2025}$ | $3887_{\pm 1178}$ | $3620_{\pm 3589}$ | $665_{\pm 953}$ | $570_{\pm 1003}$ |
| Per-env tuned $\alpha$ (oracle) | 16146 | 7244 | 8903 | 3117 | 5882 |
| selected $\alpha$ | 5 | 15 | 30 | 5 | 30 |
| *One ESS target, identical across all environments* | | | | | |
| $K_{\text{eff}} = 5$ (constant) | $16049_{\pm 899}$ | $7515_{\pm 111}$ | $8629_{\pm 703}$ | $2646_{\pm 1323}$ | $5826_{\pm 290}$ |
| vs. oracle | $-0.6\%$ | $+3.7\%$ | $-3.1\%$ | $\mathbf{-15.1\%}$ | $-0.9\%$ |

### B.3 Freezing the Target Effective Sample Size

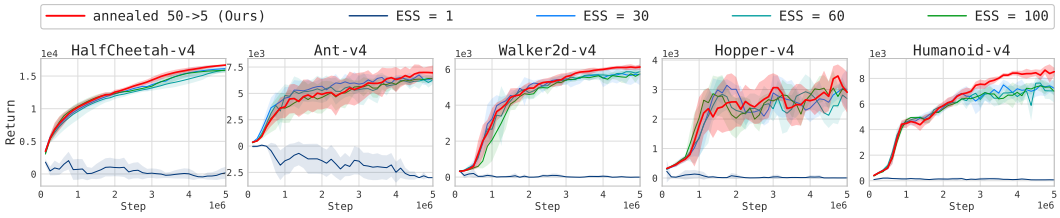



Figure 8: **Ablation of the target effective sample size.** Learning curves comparing our annealed target ESS schedule with fixed $K^{\star}_{\text{eff}} \in \{1, 30, 60, 100\}$. Mean over five seeds, shaded 95% CI.

Table 5: **Final performance with fixed target ESS.** Final performance of our annealed target ESS schedule and fixed $K^{\star}_{\text{eff}}$ values, measured as the mean of the last five evaluations. Values report mean $\pm$ 95% CI over five seeds. The fixed $K^{\star}_{\text{eff}} = 5$ results are the $K/20 \to K/20$ control reported in Table 7.

| Target ESS | HalfCheetah | Ant | Walker2d | Hopper | Humanoid |
|---|---|---|---|---|---|
| Ours ($50 \to 5$) | $\mathbf{16544 \pm 166}$ | $6947 \pm 553$ | $\mathbf{6109 \pm 127}$ | $\mathbf{3122 \pm 445}$ | $8452 \pm 310$ |
| $K^{\star}_{\text{eff}} = 1$ | $-127 \pm 661$ | $-2923 \pm 90$ | $-0 \pm 16$ | $3 \pm 1$ | $70 \pm 10$ |
| $K^{\star}_{\text{eff}} = 5$ | $16095 \pm 659$ | $\mathbf{7168 \pm 376}$ | $5856 \pm 180$ | $2248 \pm 1107$ | $\mathbf{8660 \pm 377}$ |
| $K^{\star}_{\text{eff}} = 30$ | $16042 \pm 625$ | $6453 \pm 206$ | $5707 \pm 495$ | $2869 \pm 328$ | $7316 \pm 281$ |
| $K^{\star}_{\text{eff}} = 60$ | $15671 \pm 349$ | $6187 \pm 309$ | $5827 \pm 105$ | $2506 \pm 493$ | $7273 \pm 178$ |
| $K^{\star}_{\text{eff}} = 100$ | $15819 \pm 410$ | $6359 \pm 138$ | $5646 \pm 220$ | $2919 \pm 305$ | $7276 \pm 191$ |

As shown in Figure 8 and Table 5, fixed targets $K^{\star}_{\text{eff}} \in \{1, 30, 60, 100\}$ yield lower mean final returns than our annealed schedule in all five environments. The table also reports the fixed target $K^{\star}_{\text{eff}} = 5$, which is the $K/20 \to K/20$ control in Table 7. Relative to this fixed target, annealing yields higher mean returns on HalfCheetah, Walker2d, and Hopper, whereas the fixed target performs better on Ant and Humanoid. The extreme target $K^{\star}_{\text{eff}} = 1$ severely degrades learning across all five environments. These mixed results suggest that the useful timing of candidate-weight concentration may vary across environments. No fixed target tested here gives the highest mean return on all five tasks, and our annealed schedule is a competitive shared choice. The schedule does not

measure task-specific critic reliability, so these return comparisons do not establish it as the cause of the task differences.

### B.4 Reversing the Annealing Direction

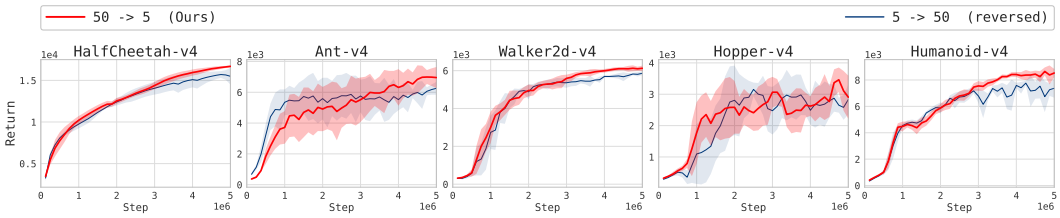



Figure 9: **Ablation of the annealing direction.** Learning curves comparing our high-to-low target ESS schedule ($K/2 \to K/20$) with the reversed low-to-high schedule ($K/20 \to K/2$). Both schedules traverse the same target ESS range over the same annealing window. Mean over five seeds, shaded 95% CI.

Table 6: **Final performance with reversed annealing.** Final performance of our high-to-low target ESS schedule and the reversed low-to-high schedule, measured as the mean of the last five evaluations. Values report mean ± 95% CI over five seeds. Parentheses give the relative change against Ours.

| Schedule | HalfCheetah | Ant | Walker2d | Hopper | Humanoid |
|---|---|---|---|---|---|
| Ours ($K/2 \to K/20$) | $\mathbf{16544 \pm 166}$ | $\mathbf{6947 \pm 553}$ | $\mathbf{6109 \pm 127}$ | $\mathbf{3122 \pm 445}$ | $\mathbf{8452 \pm 310}$ |
| Reverse ($K/20 \to K/2$) | $15593 \pm 577$ ($-5.7\%$) | $6061 \pm 462$ ($-12.8\%$) | $5783 \pm 107$ ($-5.3\%$) | $2745 \pm 358$ ($-12.1\%$) | $7224 \pm 568$ ($-14.5\%$) |

As shown in Figure 9 and Table 6, reversing the annealing direction degrades final performance across all five environments, with the largest reduction on Humanoid ($-14.5\%$). Importantly, the two schedules traverse the same target ESS range over the same annealing window and differ only in the order in which the targets are visited. Thus, the benefit of annealing cannot be explained simply by exposing the policy to a range of target ESS values; the ordering of those values also matters.

Our high-to-low schedule maintains more diffuse candidate weighting early in training and gradually increases concentration later. This ordering is consistent with the intended role of the schedule: using a more conservative score estimate early, when the critic is expected to be less reliable, and allowing stronger concentration as training progresses.

### B.5 The Annealing Ratio

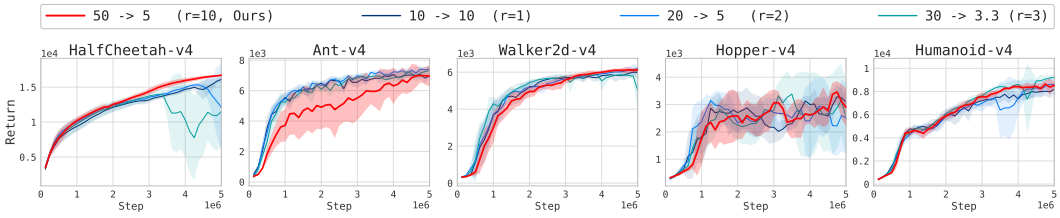



Figure 10: **Ablation of the annealing ratio.** Learning curves for schedules with a fixed geometric mean of 10 and ratios $r \in \{1, 2, 3\}$, corresponding to $10 \to 10$, $20 \to 5$, and $30 \to 3.3$, together with our $50 \to 5$ schedule and a warm-up-matched control. Mean over five seeds, shaded 95% CI.

A natural hypothesis is that the schedule is governed by the contrast between its early and late target ESS values, rather than by their absolute levels. We quantify this contrast by the annealing ratio $r =$

$\sqrt{K_{\text{start}}/K_{\text{end}}}$, where a larger $r$ corresponds to a larger change in target ESS over training. To isolate the effect of this ratio from the overall ESS level, we hold the geometric mean $\sqrt{K_{\text{start}}K_{\text{end}}} = 10$ fixed and vary $r \in \{1, 2, 3\}$, yielding the schedules $10 \to 10$, $20 \to 5$, and $30 \to 3.3$, respectively.

As shown in Figure 10, increasing $r$ at the fixed geometric mean is accompanied by increasing variability on HalfCheetah: the number of seeds finishing below 13,000 increases from $0/5$ to $1/5$ and $2/5$ for $r = 1, 2, 3$, respectively. However, our $50 \to 5$ schedule has an even larger ratio ($r \approx 3.16$) while remaining substantially more stable. This indicates that the annealing ratio alone does not explain the observed performance.

Instead, varying $r$ at a fixed geometric mean also changes the actual ESS endpoints. In particular, increasing $r$ lowers $K_{\text{end}}$ from 10 to 5 and 3.3. Thus, the relevant design choice is not simply the relative contrast between the initial and terminal ESS, but the actual ESS levels at which annealing starts and ends. We examine these endpoints directly in Appendix B.6 and Appendix B.7.

### B.6 Varying the Annealing Endpoints

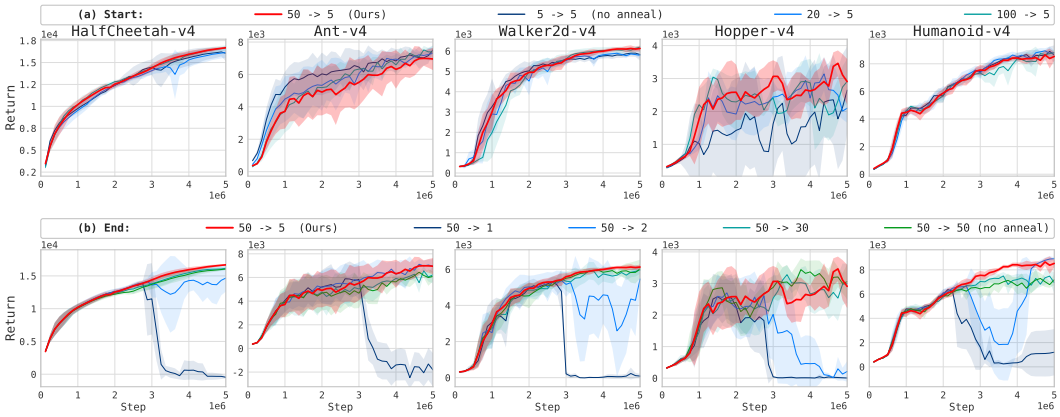



Figure 11: **Ablation of the annealing endpoints.** (a) The endpoint is fixed at $K/20$ and the starting target is varied over $\{K/20,\ K/5,\ K/2,\ K\}$. (b) The start is fixed at $K/2$ and the endpoint is varied over $\{K/100,\ K/50,\ K/20,\ 3K/10,\ K/2\}$. Here $K = 100$, and a schedule whose start and end coincide keeps the target constant. Mean over five seeds, shaded 95% CI.

Table 7: **Final performance across annealing endpoints.** Mean of the last five evaluations, $\pm$ 95% CI over five seeds. Parentheses give the relative change against our $K/2 \to K/20$ schedule. **Bold** marks the best value in each column.

| Schedule | HalfCheetah | Ant | Walker2d | Hopper | Humanoid |
|---|---|---|---|---|---|
| Ours ($K/2 \to K/20$) | **16544 ± 166** | 6947 ± 553 | **6109 ± 127** | **3122 ± 445** | 8452 ± 310 |
| *(a) Varying the start* | | | | | |
| $K/20 \to K/20$ | 16095 ± 659 (−2.7%) | 7168 ± 376 (+3.2%) | 5856 ± 180 (−4.1%) | 2248 ± 1107 (−28.0%) | 8660 ± 377 (+2.5%) |
| $K/5 \to K/20$ | 15992 ± 639 (−3.3%) | 7206 ± 221 (+3.7%) | 5815 ± 69 (−4.8%) | 2381 ± 413 (−23.7%) | **8829 ± 251** (+4.5%) |
| $K \to K/20$ | 16521 ± 320 (−0.1%) | **7311 ± 153** (+5.2%) | 6096 ± 117 (−0.2%) | 2365 ± 533 (−24.2%) | 8620 ± 334 (+2.0%) |
| *(b) Varying the end* | | | | | |
| $K/2 \to K/100$ | −349 ± 405 (−102.1%) | −1662 ± 1173 (−123.9%) | 113 ± 148 (−98.1%) | 19 ± 28 (−99.4%) | 1122 ± 1898 (−86.7%) |
| $K/2 \to K/50$ | 14077 ± 2455 (−14.9%) | 6943 ± 500 (−0.1%) | 4098 ± 1164 (−32.9%) | 146 ± 133 (−95.3%) | 8732 ± 171 (+3.3%) |
| $K/2 \to 3K/10$ | 16046 ± 300 (−3.0%) | 6134 ± 429 (−11.7%) | 5834 ± 169 (−4.5%) | 2755 ± 619 (−11.8%) | 7175 ± 280 (−15.1%) |
| $K/2 \to K/2$ | 15908 ± 228 (−3.8%) | 6173 ± 316 (−11.1%) | 5897 ± 458 (−3.5%) | 3060 ± 388 (−2.0%) | 7087 ± 224 (−16.1%) |

We vary one end of the schedule at a time around our $K/2 \to K/20$ setting (Figure 11, Table 7).

**The start is flexible, except on Hopper.** With the endpoint fixed at $K/20$, starting from $K/5$ or $K$ instead of $K/2$ keeps HalfCheetah, Ant, Walker2d and Humanoid within 5% of our schedule, and slightly improves Ant and Humanoid. Hopper is the exception: every alternative start costs it 24–28%, including the constant target $K/20$ that skips the high-ESS phase altogether.

**The end is sensitive in both directions.** With the start fixed at $K/2$, annealing too far concentrates the weights on too few candidates and destabilizes training. $K_{\text{end}} = K/100$ collapses in every environment, and $K/50$ collapses Hopper ($-95.3\%$) and Walker2d ($-32.9\%$). These collapses begin at about $3 \times 10^6$ steps, which is when the annealed target reaches its endpoint. Stopping the anneal early has the opposite effect: $K_{\text{end}} = 3K/10$ and $K/2$ train stably but settle lower, losing 15–16% on Humanoid and about 11% on Ant.

**No alternative is better across the board.** Our schedule is best on HalfCheetah, Walker2d and Hopper. The two environments where another setting wins are Ant, which peaks with a start of $K$ ($+5.2\%$), and Humanoid, which peaks with a start of $K/5$ ($+4.5\%$); both of those settings lose about 24% on Hopper. Measured against the best value in each environment, our schedule falls short by at most 5.0% (on Ant), whereas every alternative falls short by 18.7% or more in at least one environment. We therefore use $K/2 \to K/20$ in all environments rather than tuning the endpoints per task.

### B.7 The Annealing Endpoint Grid

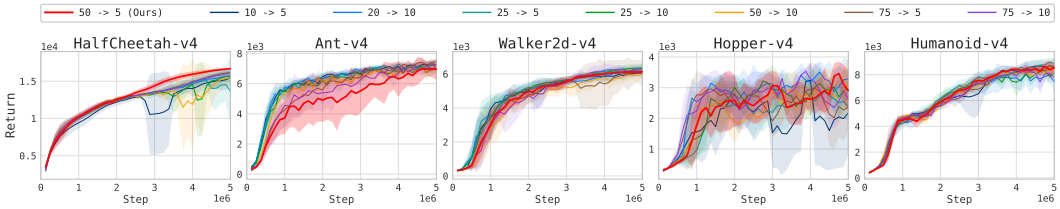



Figure 12: **Grid ablation of the annealing endpoints.** Learning curves for different $(K_{\text{start}}, K_{\text{end}})$ configurations spanning starting targets from 10 to 75 and terminal targets of 5 or 10, together with our $50 \to 5$ schedule. Mean over five seeds, shaded 95% CI.

Table 8: **Final performance across annealing endpoint configurations.** Final performance for different $(K_{\text{start}}, K_{\text{end}})$ configurations, measured as the mean of the last five evaluations. Values report mean $\pm$ 95% CI over five seeds. Parentheses give the relative change against our $50 \to 5$ schedule.

| Schedule | HalfCheetah | Ant | Walker2d | Hopper | Humanoid |
|---|---|---|---|---|---|
| | | | $K_{\text{end}} = 5$ | | |
| **Ours** ($50 \to 5$) | $16544 \pm 166$ | $6947 \pm 553$ | $6109 \pm 127$ | $3122 \pm 445$ | $8452 \pm 310$ |
| $25 \to 5$ | $14151 \pm 2964$ ($-14.5\%$) | $7252 \pm 154$ ($+4.4\%$) | $6253 \pm 258$ ($+2.4\%$) | $2563 \pm 424$ ($-17.9\%$) | $8780 \pm 425$ ($+3.9\%$) |
| $75 \to 5$ | $15960 \pm 310$ ($-3.5\%$) | $7324 \pm 188$ ($+5.4\%$) | $6055 \pm 769$ ($-0.9\%$) | $2235 \pm 950$ ($-28.4\%$) | $8493 \pm 214$ ($+0.5\%$) |
| | | | $K_{\text{end}} = 10$ | | |
| $10 \to 10$ | $15407 \pm 775$ ($-6.9\%$) | $6976 \pm 169$ ($+0.4\%$) | $5918 \pm 249$ ($-3.1\%$) | $3034 \pm 285$ ($-2.8\%$) | $8020 \pm 374$ ($-5.1\%$) |
| $20 \to 10$ | $15485 \pm 938$ ($-6.4\%$) | $7067 \pm 244$ ($+1.7\%$) | $6142 \pm 236$ ($+0.5\%$) | $3132 \pm 544$ ($+0.3\%$) | $7750 \pm 431$ ($-8.3\%$) |
| $25 \to 10$ | $15391 \pm 789$ ($-7.0\%$) | $6959 \pm 112$ ($+0.2\%$) | $6318 \pm 240$ ($+3.4\%$) | $2900 \pm 396$ ($-7.1\%$) | $8122 \pm 329$ ($-3.9\%$) |
| $50 \to 10$ | $15035 \pm 1584$ ($-9.1\%$) | $6904 \pm 288$ ($-0.6\%$) | $5912 \pm 302$ ($-3.2\%$) | $2837 \pm 736$ ($-9.1\%$) | $8242 \pm 333$ ($-2.5\%$) |

As shown in Figure 12 and Table 8, the effect of the annealing endpoints is environment dependent. Changing the starting target while keeping $K_{\text{end}} = 5$ can substantially affect individual environments: $25 \to 5$ reduces performance on HalfCheetah and Hopper by 14.5% and 17.9%, respectively, while $75 \to 5$ reduces Hopper by 28.4%. At the same time, these configurations slightly improve performance on Ant and Humanoid.

Changing the terminal target to $K_{\text{end}} = 10$ produces a different pattern. All tested configurations reduce performance on Humanoid ($-2.5\%$ to $-8.3\%$), while their effects on Ant, Walker2d, and Hopper are generally smaller and vary with the starting target. Overall, the grid does not reveal a single endpoint pair that consistently improves performance across environments. Our $50 \to 5$ schedule therefore provides a robust choice across the tested tasks without environment-specific endpoint tuning.

### B.8 Annealing Duration

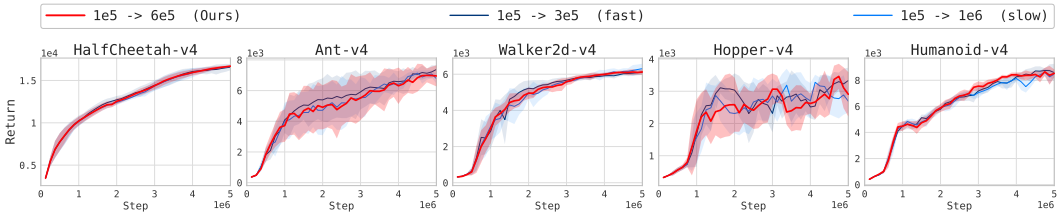



Figure 13: **Ablation of the annealing duration.** Learning curves for three annealing durations with identical target ESS endpoints ($50 \to 5$) and the same starting point ($10^5$ updates). The annealing ends at $3 \times 10^5$ (fast), $6 \times 10^5$ (ours), or $10^6$ (slow) updates. Mean over five seeds, shaded 95% CI.

Having fixed the endpoints and direction of the schedule, we next examine how sensitive performance is to the annealing duration. We keep the annealing start fixed at $n_{\text{start}} = 10^5$ updates and vary only its end, $n_{\text{end}} \in \{3, 6, 10\} \times 10^5$, corresponding to fast, default, and slow annealing, respectively.

As shown in Figure 13, changing the annealing duration has a relatively limited effect on performance compared with changing the endpoints or reversing the annealing direction. The largest changes relative to our setting are $+2.9\%$ on Humanoid with fast annealing and $-10.2\%$ on Hopper with slow annealing, while the remaining differences are smaller. This suggests that performance is more sensitive to which target ESS levels are visited and in what order than to the precise rate at which the schedule moves between them. We therefore use $10^5 \to 6 \times 10^5$ as our default annealing window without further tuning.

### B.9 The Number of Candidates

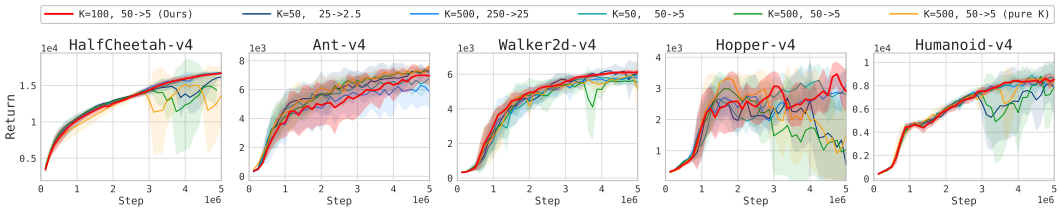



Figure 14: **Ablation of the number of candidates.** Learning curves for $K \in \{50, 100, 500\}$ under relative scaling of the target ESS schedule, absolute scaling with the terminal target fixed at 5, and a $K = 500$ variant that retains our $50 \to 5$ schedule. Mean over five seeds, shaded 95% CI.

Table 9: **Final performance across candidate budgets $K$.** Final performance when the candidate budget is changed together with the target schedule, either preserving the relative concentration $K_{\text{eff}}/K$ or the absolute endpoint $K_{\text{eff}}$, measured as the mean of the last five evaluations. Values report mean $\pm$ 95% CI over five seeds. Parentheses give the relative change against our $K = 100$ configuration.

| Budget & schedule | HalfCheetah | Ant | Walker2d | Hopper | Humanoid |
|---|---|---|---|---|---|
| Ours ($K = 100$, $50 \to 5$) | **16544 ± 166** | **6947 ± 553** | **6109 ± 127** | **3122 ± 445** | **8452 ± 310** |
| $K = 50$, $25 \to 2.5$ | 15631 ± 1064 (−5.5%) | 7223 ± 374 (+4.0%) | 5998 ± 530 (−1.8%) | 1202 ± 1222 (−61.5%) | 7605 ± 2186 (−10.0%) |
| $K = 50$, $25 \to 5$ | 16598 ± 314 (+0.3%) | 6537 ± 744 (−5.9%) | 5836 ± 229 (−4.5%) | 2800 ± 654 (−10.3%) | 8477 ± 354 (+0.3%) |
| $K = 500$, $250 \to 25$ | 16657 ± 184 (+0.7%) | 6040 ± 1190 (−13.1%) | 5776 ± 276 (−5.5%) | 2897 ± 500 (−7.2%) | 8265 ± 488 (−2.2%) |
| $K = 500$, $250 \to 5$ | 14582 ± 2330 (−11.9%) | 7266 ± 53 (+4.6%) | 5580 ± 240 (−8.7%) | 1170 ± 1338 (−62.5%) | 7831 ± 1395 (−7.3%) |
| $K = 500$, $50 \to 5$ | 12718 ± 4783 (−23.1%) | 7311 ± 185 (+5.2%) | 5694 ± 486 (−6.8%) | 1224 ± 1374 (−60.8%) | 8436 ± 325 (−0.2%) |

Changing the candidate budget $K$ changes both the absolute and relative meaning of a target ESS. For example, a target of $K_{\text{eff}} = 5$ corresponds to $10\%$ of the candidate pool when $K = 50$, but only $1\%$ when $K = 500$. We therefore vary $K \in \{50, 100, 500\}$ under two scaling conventions. Relative scaling preserves $K_{\text{eff}}/K$ by scaling both endpoints with $K$, yielding $25 \to 2.5$ at $K = 50$ and $250 \to 25$ at $K = 500$. Absolute scaling instead keeps the terminal target fixed at $K_{\text{eff}} = 5$. We additionally evaluate $K = 500$ while retaining our original $50 \to 5$ schedule.

As shown in Figure 14 and Table 9, neither absolute nor relative scaling alone transfers consistently across candidate budgets. This is particularly clear on Hopper. At $K = 50$, preserving the relative terminal ESS of $5\%$ lowers the absolute target to 2.5, resulting in a $61.5\%$ performance drop and $3/5$ seeds finishing below 1500 return. Increasing the terminal target to 5 ($10\%$ of $K$) substantially reduces this degradation. Conversely, at $K = 500$, keeping the absolute terminal ESS at 5 corresponds to only $1\%$ of the candidate pool and produces similarly large performance drops ($60.8$–$62.5\%$). Relative scaling to 25 ($5\%$ of $K$) substantially mitigates this degradation.

These observations suggest that both the absolute and relative levels of the terminal ESS matter. Among the tested configurations, the largest degradations on Hopper occur when either the absolute terminal ESS is below 5 or its relative value is below $5\%$ of the candidate pool. In contrast, configurations with an absolute terminal ESS of at least 5 and a relative terminal ESS of at least $5\%$ avoid these severe failures. A small absolute ESS leaves only a few candidates with substantial effective weight, whereas a small relative ESS corresponds to strong concentration relative to the candidate pool.

Our choice of $K = 100$ places the terminal target $K_{\text{eff}} = 5$ at $5\%$ of the candidate pool. Increasing the budget to $K = 500$ does not provide a consistent improvement across environments while requiring five times as many candidate evaluations. We therefore use $K = 100$ as our default candidate budget.

# C Diffusion-Time Binning and ESS Diagnostics

We compare the inverse temperature and resulting candidate-weight ESS across diffusion time in two separately trained settings. The fixed-temperature runs disable ESS feedback and use $\alpha = 30$ at every $t$ and in every environment. SOM instead updates a separate $\alpha_{n,b}$ in each of $B = 16$ diffusion-time bins to track a common ESS target at training update $n$. The two settings otherwise share the same hyperparameters; in the fixed-temperature runs, the 16-bin partition is retained only for logging and does not affect training. Table 10 reports the temperatures, and Table 12 reports the resulting ESS. We then assess how the number of bins affects return. The fixed-$\alpha = 30$ diagnostic is distinct from the $B = 1$ ablation: $B = 1$ still adapts one temperature by ESS feedback but shares it across all diffusion times.

## C.1 Inverse temperature and achieved ESS across diffusion time

For the per-bin diagnostics, an update with no samples in a bin is logged as zero. We divide the logged mean by that bin's occupancy rate, so the reported values average over updates in which the bin was populated. The second-half aggregates cover updates $[5 \times 10^5, 10^6)$ (2.5–5M environment steps). Occupancy is at least $98\%$ for bins 2–14 in this window; the correction matters more for the sparsely populated edge bins.

Table 10: **Inverse temperature by diffusion-time bin.** The fixed column shows the prescribed $\alpha = 30$. The $B = 16$ entries are occupancy-corrected controller outputs averaged over the second half of training and five seeds; all controlled runs use the same target-ESS schedule. Table 12 reports the candidate-weight ESS under these settings. Bins 0, 1, and 15 (†) are populated in only 3%, 60%, and 38% of updates, respectively.

| | | Fixed $\alpha = 30$ | $B = 16$ ESS control (Ours) | | | | |
|---|---|---|---|---|---|---|---|
| bin | $t$ | all envs | HalfCheetah | Ant | Walker2d | Hopper | Humanoid |
| 0† | 0.00–0.06 | 30 | 61.4 | 87.2 | 192.9 | 347.9 | 173.0 |
| 1† | 0.06–0.12 | 30 | 56.2 | 79.0 | 184.5 | 366.4 | 137.1 |
| 2 | 0.12–0.19 | 30 | 44.9 | 60.8 | 153.8 | 325.8 | 110.3 |
| 3 | 0.19–0.25 | 30 | 36.4 | 47.9 | 130.0 | 292.9 | 91.2 |
| 4 | 0.25–0.31 | 30 | 29.7 | 38.2 | 110.5 | 265.0 | 76.9 |
| 5 | 0.31–0.38 | 30 | 24.1 | 30.5 | 93.8 | 239.8 | 65.6 |
| 6 | 0.38–0.44 | 30 | 19.5 | 24.3 | 79.3 | 216.4 | 56.3 |
| 7 | 0.44–0.50 | 30 | 15.7 | 19.3 | 66.5 | 193.8 | 48.7 |
| 8 | 0.50–0.56 | 30 | 12.5 | 15.2 | 55.3 | 171.9 | 42.2 |
| 9 | 0.56–0.62 | 30 | 9.9 | 11.9 | 45.5 | 150.8 | 36.6 |
| 10 | 0.62–0.69 | 30 | 7.8 | 9.2 | 37.0 | 130.4 | 31.7 |
| 11 | 0.69–0.75 | 30 | 6.2 | 7.2 | 29.8 | 111.2 | 27.5 |
| 12 | 0.75–0.81 | 30 | 5.3 | 5.7 | 23.8 | 93.5 | 23.8 |
| 13 | 0.81–0.88 | 30 | 5.1 | 5.1 | 18.8 | 77.6 | 20.6 |
| 14 | 0.88–0.94 | 30 | 5.0 | 5.0 | 14.8 | 63.9 | 17.9 |
| 15† | 0.94–1.00 | 30 | 5.0 | 5.0 | 11.9 | 53.0 | 15.8 |

**Training-stage comparison.** To check whether the diffusion-time contrast in Table 10 arises solely from averaging over training, we compare early and late windows in Table 11. In every environment, the mean inverse temperature in bin 2 exceeds that in bin 14 within both windows: the ratio ranges from 3.0 to 6.2 early and from 5.1 to 11.9 late. Thus the bin-2 versus bin-14 contrast is present within each inspected stage, although its magnitude changes during learning. The mean $\alpha_2$ also rises between the windows. The early window contains the initial target ESS of 50 and the beginning of its anneal; the target reaches 5 only at update $6 \times 10^5$ (3M environment steps). A higher ESS target favors more diffuse candidate weights and hence a smaller $\alpha$, so the cross-window rise in $\bar{\alpha}_2$ does not by itself establish that the critic's Q-value contrasts grew. Moreover, the late ratios on HalfCheetah and Ant are affected by the $\alpha \geq 5$ lower bound on bin 14 (Table 10). These are diagnostics of controller-selected temperatures, not of the terminal action distribution.

Table 11: **Selected inverse temperatures within two training-stage windows.** Early and late denote 0–1M and 4–5M environment steps, respectively, corresponding to updates $[0, 2 \times 10^5)$ and $[8 \times 10^5, 10^6)$ (one update per five environment steps). Bin 2 covers $t \in [0.12, 0.19]$ and bin 14 covers $t \in [0.88, 0.94]$. We compute each bin's occupancy-corrected mean $\bar{\alpha}_b$ over the window and five seeds, then report $\bar{\alpha}_2/\bar{\alpha}_{14}$, not the mean of per-update ratios.

| | Early training | | Late training | |
|---|---|---|---|---|
| Environment | $\bar{\alpha}_2$ | $\bar{\alpha}_2/\bar{\alpha}_{14}$ | $\bar{\alpha}_2$ | $\bar{\alpha}_2/\bar{\alpha}_{14}$ |
| HalfCheetah | 17.9 | 3.0× | 45.6 | 9.1× |
| Ant | 22.3 | 4.1× | 59.3 | 11.9× |
| Walker2d | 28.8 | 5.1× | 153.3 | 10.6× |
| Hopper | 40.7 | 6.2× | 332.6 | 5.1× |
| Humanoid | 25.5 | 3.8× | 94.8 | 6.2× |

**Resulting weight concentration.** Table 12 shows the ESS produced by the temperatures in Table 10. Across bins 2–14, fixed $\alpha = 30$ yields a 5–21-fold range in ESS across $t$. The $B = 16$ controller limits this range to at most 1.7-fold and keeps most values near the window-averaged tar-

get of 5.27. Thus a shared numerical $\alpha$ does not imply comparable candidate-weight concentration across noise levels.

Table 12: **Achieved $K_{\text{eff}}$ by diffusion-time bin.** Entries are occupancy-corrected means over the second half of training and five seeds from the separate runs described above. Fixed $\alpha = 30$ produces substantially different weight concentrations across diffusion time. The controlled target averages 5.27 over this window, and the $B = 16$ controller keeps most populated bins near that value. Bins 0, 1, and 15 ($^\dagger$) are populated in only 3%, 60%, and 38% of updates, respectively, making their estimates less reliable.

| | | Fixed $\alpha = 30$ (no control) | | | | | $B = 16$ control (Ours) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| bin | $t$ | HalfCheetah | Ant | Walker2d | Hopper | Humanoid | HalfCheetah | Ant | Walker2d | Hopper | Humanoid |
| $0^\dagger$ | 0.00–0.06 | 12.0 | 34.4 | 70.7 | 80.3 | 56.4 | 7.2 | 7.7 | 7.6 | 7.9 | 5.4 |
| $1^\dagger$ | 0.06–0.12 | 8.4 | 23.8 | 63.1 | 78.8 | 47.7 | 5.3 | 5.3 | 5.3 | 5.4 | 5.2 |
| 2 | 0.12–0.19 | 6.1 | 15.9 | 52.6 | 73.0 | 38.0 | 5.3 | 5.3 | 5.3 | 5.3 | 5.2 |
| 3 | 0.19–0.25 | 4.8 | 10.7 | 43.3 | 67.4 | 30.1 | 5.3 | 5.3 | 5.3 | 5.3 | 5.2 |
| 4 | 0.25–0.31 | 3.9 | 7.4 | 35.2 | 62.0 | 23.9 | 5.3 | 5.3 | 5.3 | 5.3 | 5.2 |
| 5 | 0.31–0.38 | 3.2 | 5.1 | 28.2 | 56.7 | 18.9 | 5.3 | 5.3 | 5.3 | 5.3 | 5.2 |
| 6 | 0.38–0.44 | 2.8 | 3.7 | 22.2 | 51.4 | 14.9 | 5.3 | 5.3 | 5.3 | 5.3 | 5.2 |
| 7 | 0.44–0.50 | 2.4 | 2.8 | 17.2 | 46.1 | 11.7 | 5.3 | 5.3 | 5.3 | 5.3 | 5.2 |
| 8 | 0.50–0.56 | 2.1 | 2.2 | 13.1 | 40.7 | 9.2 | 5.3 | 5.3 | 5.3 | 5.3 | 5.2 |
| 9 | 0.56–0.62 | 1.9 | 1.8 | 9.8 | 35.5 | 7.3 | 5.3 | 5.3 | 5.3 | 5.3 | 5.2 |
| 10 | 0.62–0.69 | 1.7 | 1.6 | 7.3 | 30.4 | 5.8 | 5.3 | 5.3 | 5.3 | 5.3 | 5.2 |
| 11 | 0.69–0.75 | 1.6 | 1.4 | 5.4 | 25.6 | 4.6 | 5.2 | 5.3 | 5.3 | 5.3 | 5.2 |
| 12 | 0.75–0.81 | 1.5 | 1.3 | 4.1 | 21.2 | 3.8 | 4.6 | 5.1 | 5.3 | 5.3 | 5.2 |
| 13 | 0.81–0.88 | 1.4 | 1.2 | 3.1 | 17.2 | 3.1 | 3.8 | 4.2 | 5.3 | 5.3 | 5.2 |
| 14 | 0.88–0.94 | 1.3 | 1.2 | 2.5 | 13.9 | 2.7 | 3.1 | 3.3 | 5.3 | 5.3 | 5.2 |
| $15^\dagger$ | 0.94–1.00 | 1.2 | 1.1 | 2.1 | 11.4 | 2.3 | 2.6 | 2.6 | 5.3 | 5.4 | 5.2 |

Tracking is not exact at the edges: at large $t$, HalfCheetah and Ant fall below the target, while bins 0, 1, and 15 are populated in fewer updates. In the former two environments, the selected $\alpha$ reaches 5.0 in the last bins (Table 10). Thus the diagnostics show that binwise control greatly reduces, but does not eliminate, variation in candidate-weight concentration across diffusion time.

### C.2 Effect of the number of diffusion-time bins

We compare a single ESS-controlled inverse temperature shared across all $t$ ($B = 1$) with controllers using $B \in \{4, 8, 16\}$ bins. Unlike the fixed-$\alpha = 30$ diagnostic above, every row in this comparison uses ESS feedback. For $B = 1$, the underlying exact-score field has the Boltzmann probability-flow interpretation of Sections 4.1 and 4.3 when the critic and shared $\alpha$ are frozen at a training update. The binwise variants trade this exact marginal interpretation for more uniform candidate-weight concentration across noise levels.

Table 13: **Final return versus the number of diffusion-time bins.** Mean of the last five evaluations, reported as mean $\pm 1.96$ SEM over five seeds. All rows use the ESS controller; $B = 1$ shares one inverse temperature across diffusion time. Bold denotes the highest mean in each environment.

| Bins | HalfCheetah | Ant | Walker2d | Hopper | Humanoid |
|---|---|---|---|---|---|
| $B = 1$ | $15809 \pm 673$ | $6910 \pm 479$ | $6068 \pm 48$ | $1449 \pm 995$ | $\mathbf{8872 \pm 281}$ |
| $B = 4$ | $16538 \pm 311$ | $6927 \pm 686$ | $5843 \pm 372$ | $2840 \pm 733$ | $8692 \pm 343$ |
| $B = 8$ | $16233 \pm 328$ | $\mathbf{7283 \pm 109}$ | $\mathbf{6240 \pm 209}$ | $2760 \pm 542$ | $8606 \pm 280$ |
| $B = 16$ (Ours) | $\mathbf{16544 \pm 166}$ | $6947 \pm 553$ | $6109 \pm 127$ | $\mathbf{3122 \pm 445}$ | $8452 \pm 310$ |

The largest difference between $B = 1$ and the binwise variants occurs on Hopper: the mean return rises from 1449 with one shared temperature to 2840, 2760, and 3122 with 4, 8, and 16 bins, respectively. The $B = 1$ result also has a wider interval across seeds. Results on the other four tasks are mixed: $B = 1$ has the highest mean on Humanoid, $B = 8$ on Ant and Walker2d, and $B = 16$ on HalfCheetah. These data support the use of multiple bins on Hopper, but do not establish that 16 bins are intrinsically better than 4 or 8. We use $B = 16$ as one default across tasks because

it gives the highest observed Hopper mean among the tested settings while remaining competitive elsewhere.

### C.3 TERMINAL MODE ALLOCATION UNDER BINWISE TEMPERATURE CONTROL

The preceding diagnostics establish how binwise control changes candidate-weight ESS and return, but do not identify the terminal distribution of the resulting flow. We therefore numerically integrate the exact-score velocity on a separate two-dimensional, four-mode toy energy landscape. This isolates the effect of changing $\alpha$ across diffusion time from score-estimation and one-step-network errors. The landscape uses a quadratic background term with coefficient $\lambda = 1.0$ and a mode-height contrast of $\Delta h = 0.030$. We use $B = 16$ bins, $K = 100$ candidates, target ESS 5, and the same inverse-temperature bounds $[5, 1000]$ as in the main experiments. The resulting controller schedule ranges from $\alpha_0 = 339.7$ at the data end to $\alpha_{15} = 8.5$ at the noise end; for example, $\alpha_8 = 21.9$.

We report the relative mass of each mode conditional on a sample lying in one of the four mode regions, and separately report the fraction of samples outside these regions. To test which part of the temperature schedule matters for mode allocation, we flatten either its low-noise half (`lo_flat`) or its high-noise half (`hi_flat`) while retaining the other half of the controller schedule. Table 14 shows that flattening the low-noise half changes the four-mode allocation by an $L_1$ distance of only $0.012$ from the controller, although the off-mode fraction rises from $0.0023$ to $0.0112$. Flattening the high-noise half changes the allocation much more ($L_1 = 0.368$). The tabulated runs start from the schedule-matched $t = 1$ marginal $p_1^{\alpha(1)}$. Because this initial marginal can change when the high-noise schedule is modified, we also repeat the comparison from a common Gaussian prior $\mathcal{N}(0, \sigma_{\max}^2 I)$. The same ordering holds: the corresponding $L_1$ distances are $0.007$ and $0.219$. Here $L_1$ is the sum of absolute differences between the four conditional mode masses.

Table 14: **Terminal mode allocation in the exact-velocity toy.** Mode columns give fractions conditional on assignment to one of the four modes; the off-mode column gives the fraction outside all four mode regions. The last column compares conditional mode masses with the controller using $L_1$ distance. The two interventions flatten only the indicated half of the diffusion-time temperature schedule.

| Schedule | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Off-mode | $L_1$ to controller |
|---|---|---|---|---|---|---|
| Controller | 0.201 | 0.282 | 0.379 | 0.138 | 0.0023 | — |
| `lo_flat` | 0.199 | 0.284 | 0.384 | 0.134 | 0.0112 | 0.012 |
| `hi_flat` | 0.350 | 0.316 | 0.277 | 0.056 | 0.0028 | 0.368 |

The path diagnostic places the emergence of the final relative mode allocation near $t \simeq 0.62$: there, its $L_1$ distance to the endpoint allocation is already $0.020$. As $t$ decreases further, the selected $\alpha$ rises by roughly 24-fold, while the measured off-mode fraction falls from $0.071$ to $0.0023$. Constant-temperature controls provide a complementary check: fixing $\alpha = 339.7$, the value at the data end, puts $0.957$ of the on-mode mass in mode 1, unlike the controller's $0.201$. Fixing $\alpha = 8.5$, the value at the noise end, gives a similar relative mode allocation to the controller but a larger off-mode fraction ($0.0275$). Thus the data-end temperature alone does not determine the terminal mode weights in this toy; the schedule encountered earlier along the reverse flow matters, while the later low-noise segment affects how tightly samples concentrate near modes.

For this toy, the controller's four relative mode masses can be summarized by a best-fitting single Boltzmann inverse temperature of approximately $11.3$. This is only a summary of *relative mode masses*: it does not establish that the full terminal density is a Boltzmann distribution, or restore a single-target probability-flow interpretation to the binwise velocity. Because this experiment integrates the exact-score velocity, it also does not determine the endpoint error of a learned one-step map. The conclusion is confined to the tested landscape and temperature interventions.

# D Experimental Setup and Results for the t-SNE Visualization

## D.1 Experimental Setup

This experiment isolates *policy improvement* from critic learning. We first train a source policy together with its critic until it is competent but clearly unconverged, and freeze both at that point. We then regenerate the replay buffer by rolling out the frozen policy, so that the frozen policy is a fixed point of a pure imitation update; otherwise a branch that only clones the buffer would drift toward actions the frozen policy never takes, for reasons unrelated to the update rule being studied.

Starting from this common initialization, we fine-tune SOM and MVP for the same number of steps with the critic held fixed, each branch sampling only from its own copy of the buffer. We then draw states from that buffer and, for each state, sample actions from the frozen policy, from SOM and from MVP, scoring all of them with the same frozen critic. Because the critic, the starting policy and the data distribution are shared, and because every action—including the frozen policy's own—is produced by the same best-of-$N$ selection, any difference between the resulting action distributions is attributable to the update rule alone. We report $\Delta Q = Q - Q^\star$, where $Q^\star$ is the value of the frozen policy's action at that state.

## D.2 Results for the t-SNE Visualization

For each state, we sample 120 actions from each of SOM and MVP and embed the sampled actions into two dimensions with t-SNE. We shade the background by a kernel-weighted interpolation of $\Delta Q$ over the embedded actions; the frozen policy's action anchors the background at zero, so green regions indicate actions with $\Delta Q > 0$. The color scale is normalized per panel, and is therefore comparable within a panel but not across panels. SOM samples are drawn as red circles, MVP samples as dark-blue triangles, and the frozen policy's action as a white star, with arrows from the star to each sample.

As shown in Figure 15, Figure 16, Figure 17 and Figure 18, both branches move away from the frozen policy toward higher $Q$, but by very different amounts: SOM attains several times the value gain of MVP, and in nearly every panel its entire sample cloud lies above $Q^\star$.

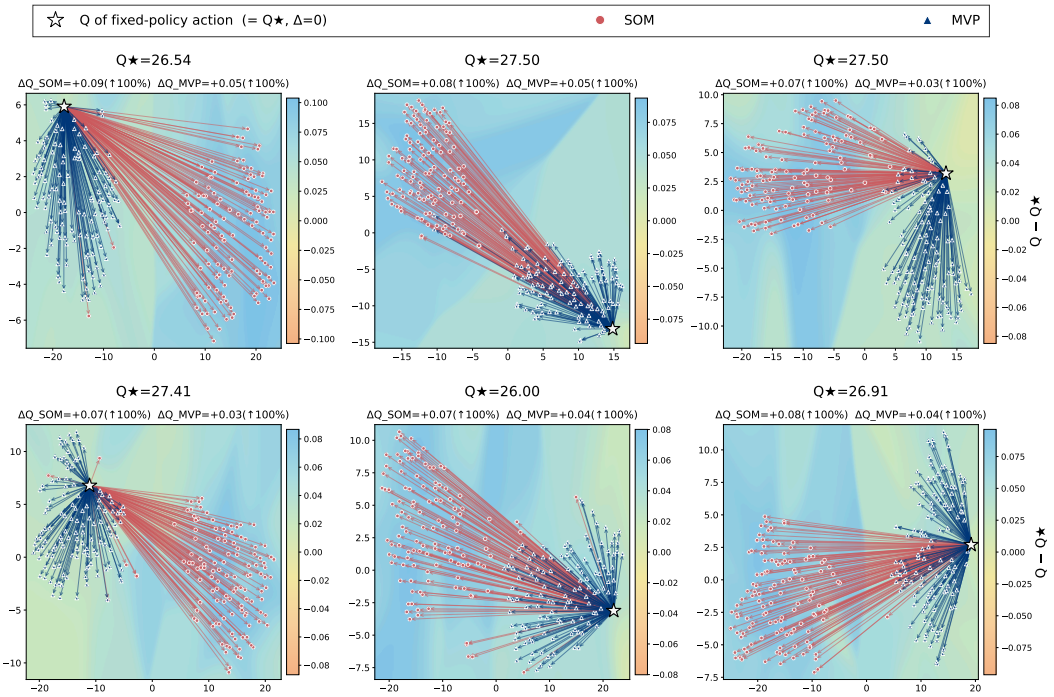



Figure 15: **HalfCheetah-v4**

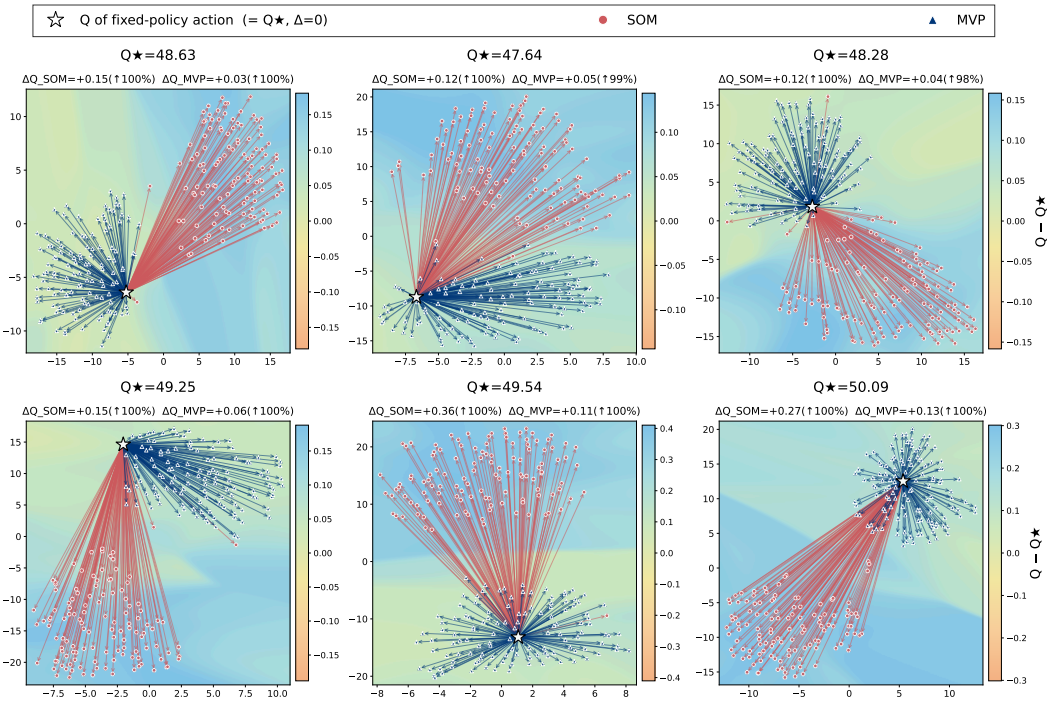



Figure 16: **Ant-v4**

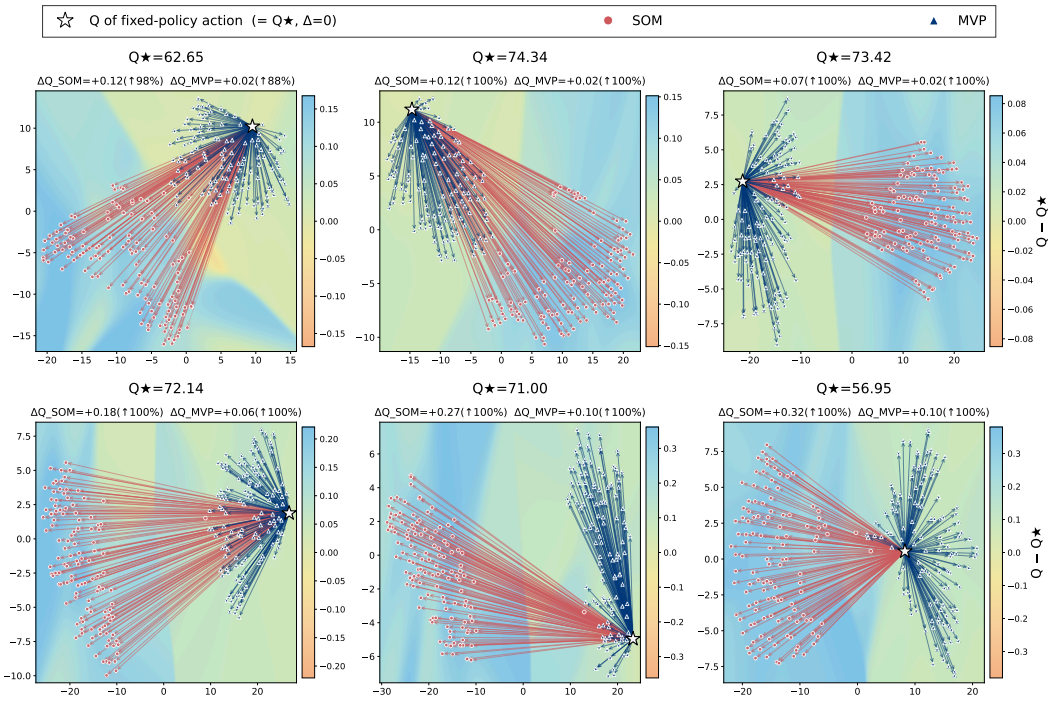



Figure 17: **Walker2d-v4**

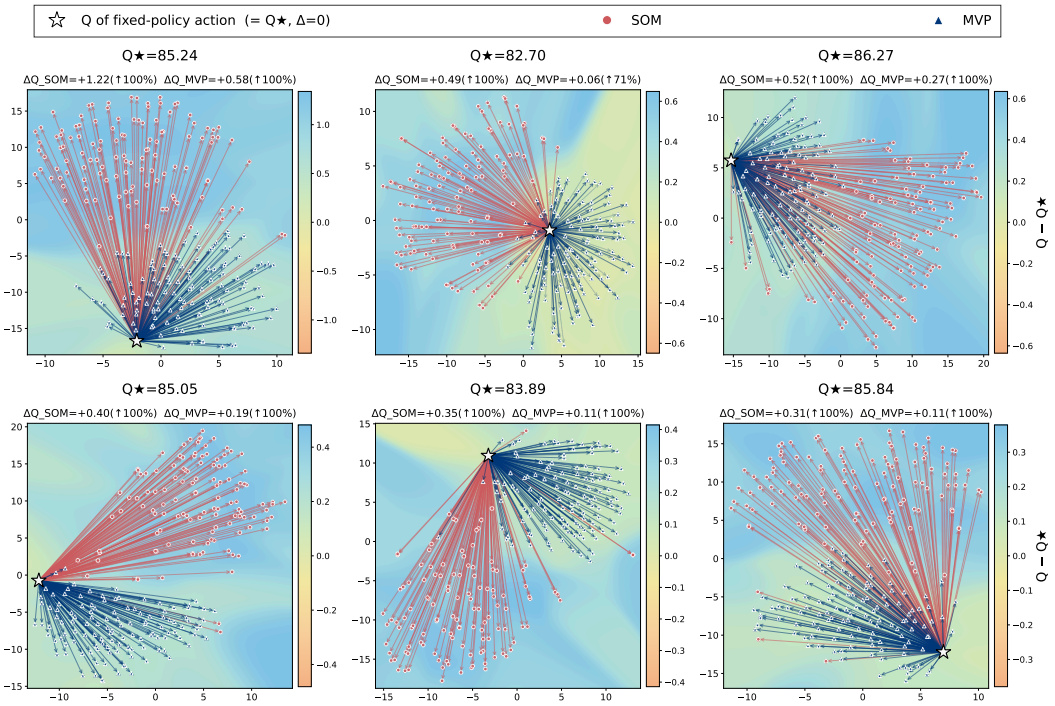



Figure 18: **Humanoid-v4**

# E 2D BANDIT TASKS: BANDIT, TWO-MOONS, AND CHECKERBOARD

## E.1 EIGHT MODE BANDIT

**Reward function** The reward is defined as a Gaussian-mixture density with $G = 8$ isotropic components $\mathcal{N}(\mu_i, \sigma^2 I_2)$ with $\sigma = 0.3$. The component centers

$$\mu_i = \sqrt{2}\left(\cos\frac{2\pi i}{8}, \sin\frac{2\pi i}{8}\right), \quad i = 0, \ldots, 7,$$

are uniformly distributed on a circle of radius $\sqrt{2}$. To create interleaved high- and low-reward modes, we assign alternating mixture weights $w_i = 2$ for even $i$ and $w_i = 1$ for odd $i$. The final reward is given by the normalized mixture density, producing a smooth multimodal reward landscape with values in $[0, 1]$.

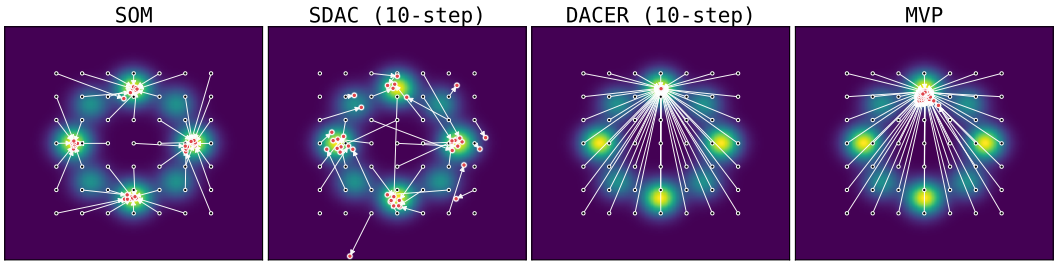



Figure 19: **Generative trajectories.** Arrow plots from a $7 \times 7$ grid of $x_T$ (black) to $x_0$ (red) on the eight-Gaussian reward. SDAC and DACER with their full 10-step rollouts. SOM (Ours) and MVP, both 1-step by design.

**Mode coverage in the clean bandit.** Figure 19 rolls out each learned policy from a fixed $7 \times 7$ grid of initial noise $x_T$ and draws the endpoints $x_0$ and their trajectories. With their full 10-step reverse process, the diffusion baselines concentrate samples on high-reward modes: SDAC distributes them across all four high-reward modes, while DACER concentrates on one. Among the one-step samplers, MVP also collapses to a single mode, whereas SOM maps the grid to all four. This trajectory view complements the sample-count comparison in Figure 5.

## E.2 TWO-MOONS

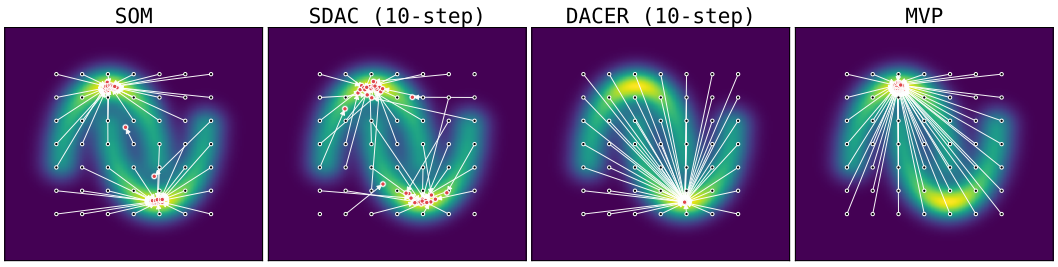



Figure 20: **Two-Moons Results.** Arrow plots from a $7 \times 7$ grid of $a_1$ (black) to $a_0$ (red) on the two-moon reward. SDAC and DACER with their full 10-step rollouts. SOM (Ours) and MVP, both 1-step by design.

As shown in Figure 20, SOM successfully transports actions toward the high-reward modes of the two-moon landscape while preserving coherent and stable transport trajectories. Compared to diffusion-based baselines with iterative denoising steps, SOM achieves comparable mode-seeking behavior using a single-step policy parameterization. In contrast, DACER exhibits more diffuse transport dynamics, whereas MVP occasionally produces less structured trajectories around the multimodal regions.

### E.3 CHECKERBOARD

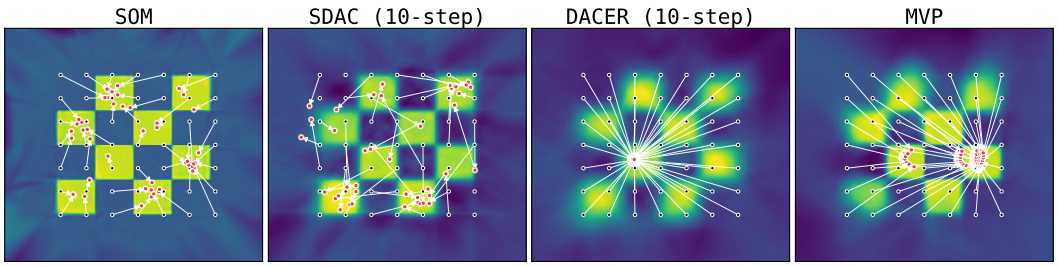



Figure 21: **Checkerboard results.** Arrow plots from a $7 \times 7$ grid of $a_1$ (black) to $a_0$ (red) on the checkerboard reward. SDAC and DACER with their full 10-step rollouts. SOM (Ours) and MVP, both 1-step by design.

As shown in Figure 21, SOM accurately transports actions toward high-reward regions in the checkerboard landscape. Compared to iterative diffusion-based baselines, SOM achieves clearer mode separation and more stable convergence behavior under a single-step policy parameterization. DACER exhibits highly diffuse transport patterns, whereas MVP tends to concentrate trajectories toward a limited subset of modes.

## F COMPARISON OF ESTIMATED AND ANALYTIC SCORE FIELDS

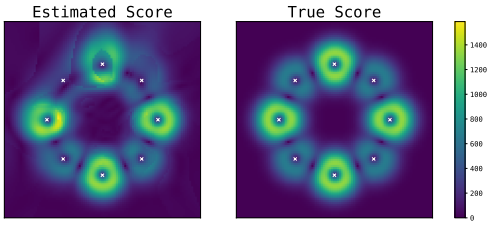



Figure 22: **Estimated field and analytic reward-gradient reference on the eight-Gaussian bandit at $t = 0$.** Left: the critic-based estimate $\hat{s}$ (maximum magnitude $\approx 1588$). Right: the analytic reference $\alpha \nabla r$ (maximum magnitude $\approx 1328$). Both panels share a single color scale; brighter regions indicate larger gradient magnitude, and the eight mode centers are marked with $\times$. The fields have correlation $0.958$ and a ratio of maximum magnitudes of $1.20$.

**Analytic reward-gradient reference.** The eight-mode reward is a peak-normalized weighted Gaussian sum,

$$r(x) = \frac{1}{c} \sum_{g=1}^{G} w_g \mathcal{N}(x; \mu_g, \Sigma_g), \qquad c = \max_x \sum_{g=1}^{G} w_g \mathcal{N}(x; \mu_g, \Sigma_g).$$

For a fixed inverse temperature $\alpha$, the log-gradient of the unnormalized Boltzmann factor $\tilde{p}_0(x) = \exp(\alpha r(x))$ is

$$\nabla_x \log \tilde{p}_0(x) = \alpha \nabla r(x) = \frac{\alpha}{c} \sum_{g=1}^{G} w_g \, \mathcal{N}(x; \mu_g, \Sigma_g) \, \Sigma_g^{-1} (\mu_g - x). \tag{9}$$

This reference differs from the score of the reward mixture, $\nabla \log r(x)$: taking the logarithm of the reward would compare the estimate with a different field.

**Result and scope.** Figure 22 shows that the displayed estimate and the analytic, unsmoothed reward-gradient reference have similar spatial structure. The comparison checks their alignment at the displayed time; it does not isolate critic approximation error or establish the accuracy of the estimator at nonzero noise levels.

# G Robustness under Two Reward Perturbations

## G.1 Random Gaussian Noise

At every reward query during training, an independent Gaussian sample with mean $0$ and standard deviation $\sigma$ is added directly to the observed scalar reward, so the same action returns a different value each time it is queried. In this setting $\sigma$ characterizes the magnitude of measurement noise on the reward value itself; it does *not* describe any spatial scale in the action space. The results for $\sigma = 0.2$ and $\sigma = 0.3$ are shown in Figure 23 and Figure 24, respectively.

## G.2 Radius for Counting Points That Reach a Mode

The 2D bandit has four high-reward modes lying on the cardinal axes (north, east, south, west) at distance $\sqrt{2} \approx 1.414$ from the origin. For each trajectory endpoint, we count it as having reached a given mode if its Euclidean distance to that mode's center is at most $0.5$. The per-mode counts $\mathrm{N}/\mathrm{E}/\mathrm{S}/\mathrm{W}$ and the total shown below each panel are obtained with this fixed tolerance of $0.5$. Figure 5 selects the terminal counts for the displayed clean and noisy conditions; the figures below show the complete timestep progression.

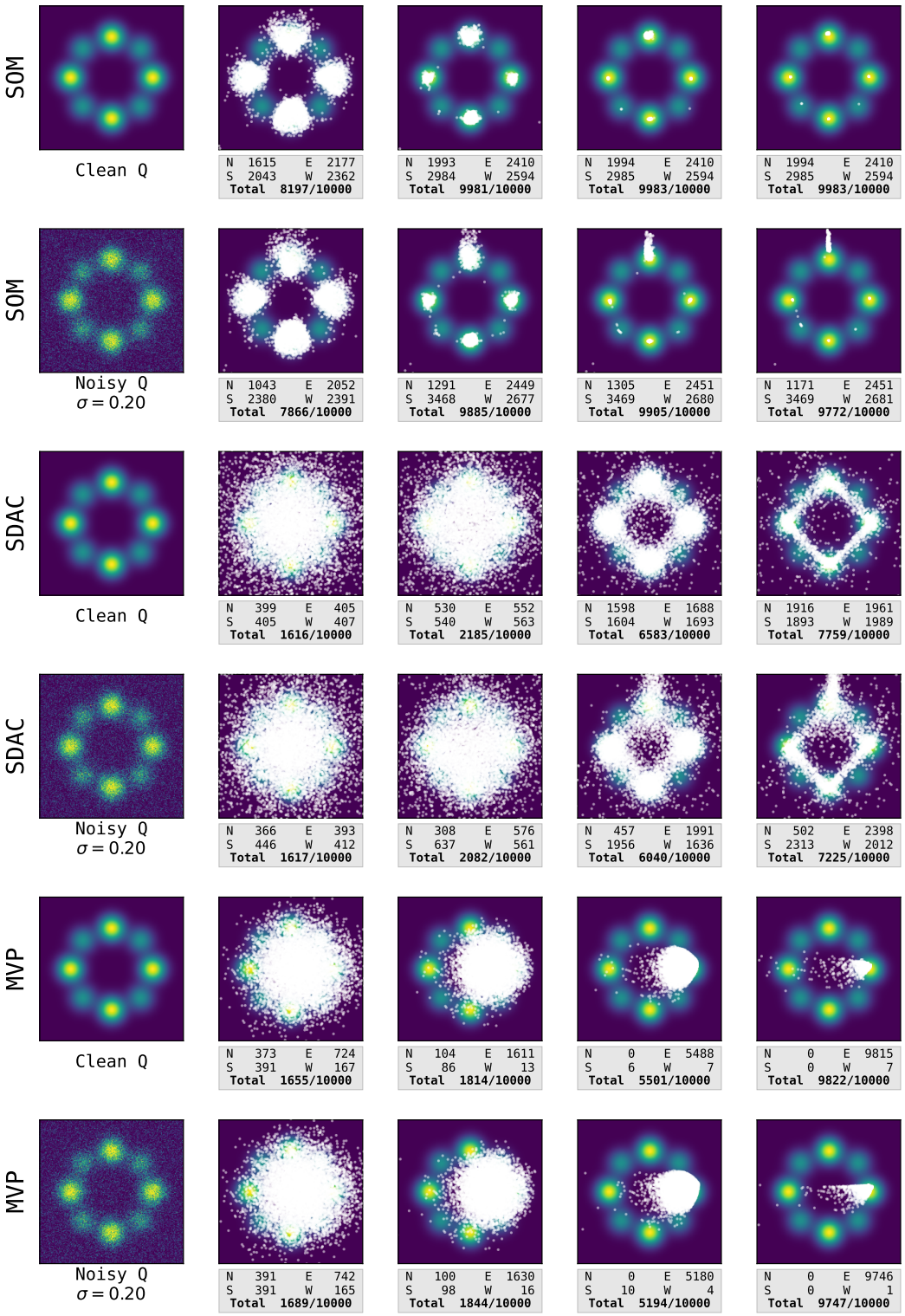



Figure 23: **Random Gaussian Noise** ($\sigma = 0.2$)**.**

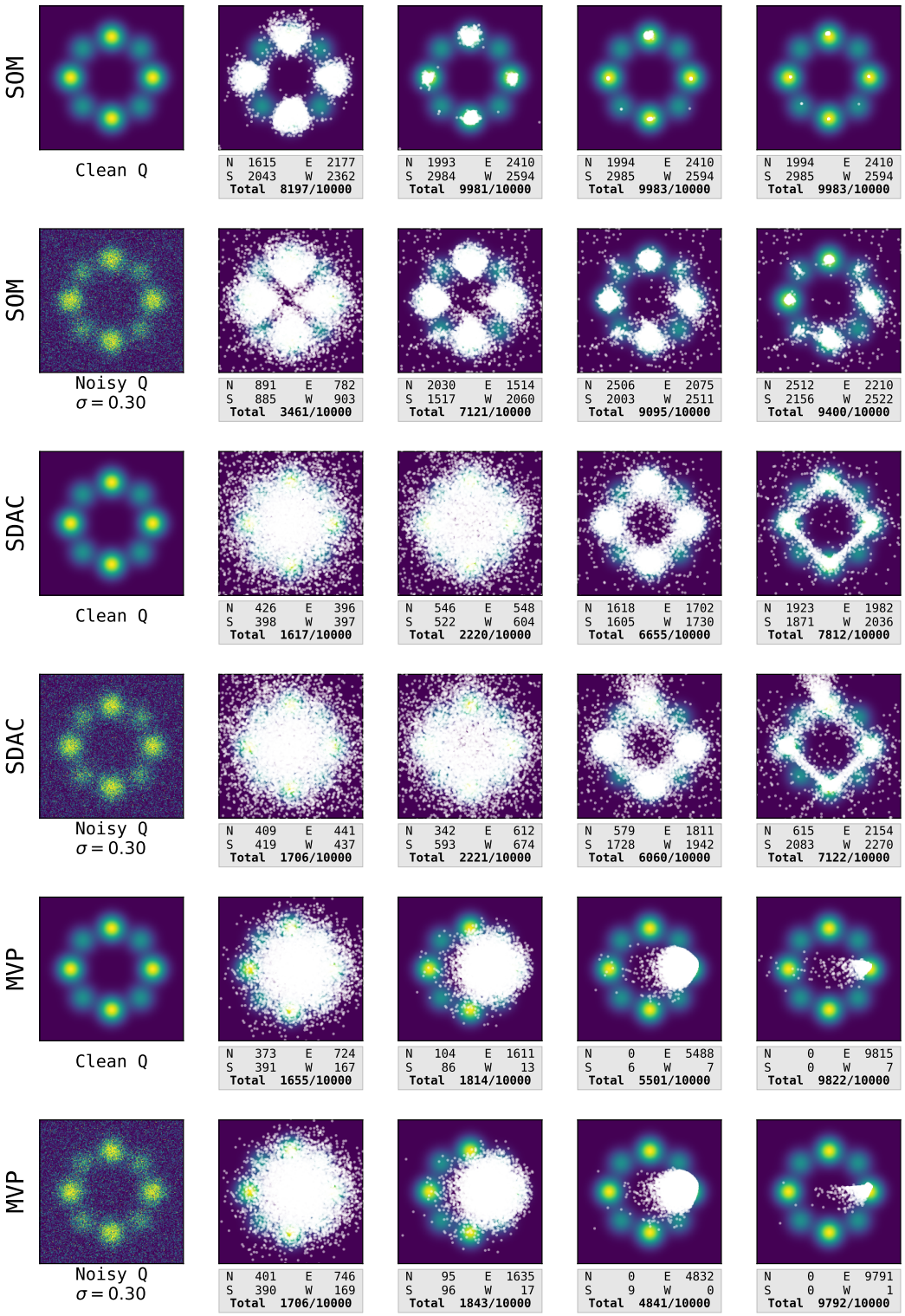



Figure 24: **Random Gaussian Noise** ($\sigma = 0.3$)**.**

# H Implementation Details

**SOM-specific settings.** Table 15 collects the default settings used by SOM in the MuJoCo experiments. The gain $\eta$ refers to the ESS feedback update in Equation (8).

Table 15: **Default SOM-specific settings.** The target ESS is held at 50 through update $10^5$, annealed geometrically (linearly in log ESS) to 5 by update $6 \times 10^5$, and held at 5 thereafter.

| Setting | Value |
|---|---|
| VE noise schedule | $\sigma(t) = \sigma_{\min}(\sigma_{\max}/\sigma_{\min})^t$, $t \in [0, 1]$ |
| Minimum noise $\sigma_{\min}$ | 0.1 |
| Maximum noise $\sigma_{\max}$ | 1.0 |
| Score-estimation candidates $K$ | 100 |
| Diffusion-time bins $B$ | 16 |
| ESS feedback gain $\eta$ | $10^{-4}$ |
| Inverse-temperature range | $\alpha \in [5, 1000]$ |
| Target ESS $K^\star_{\text{eff}}$ | $50 \to 5$ |

**Policy loss.** We adopt the adaptive loss-weighting formulation explored in MeanFlow (Geng et al., 2026) and apply it per action component. Let $e_j = [u_\theta(a_t, r, t, s) - \text{sg}(\hat{u}_{\text{tgt}})]_j$, and let $d_a$ denote the action dimension. The implemented actor loss is

$$\mathcal{L}_{\text{actor}} = \mathbb{E}\left[\frac{1}{d_a}\sum_{j=1}^{d_a} \text{sg}\Big((e_j^2 + 10^{-3})^{-1/2}\Big)\, e_j^2\right].$$

This uses the MeanFlow weighting family with $p = 0.5$ and $c = 10^{-3}$, with weights computed elementwise rather than from the full residual norm.

## H.1 Computational Resources

Note that we conducted all experiments using four NVIDIA RTX 4090 GPUs and AMD Ryzen Threadripper PRO 5995WX CPUs.

## H.2 Critic Normalization

We apply root mean square layer normalization (RMSNormalization) only to the first hidden layer of the critic. It normalizes the features by their root mean square without subtracting the mean, using a learnable element-wise scale and no additive bias.

### H.3 INVERSE-TEMPERATURE CLIPPING

We constrain the ESS controller's inverse temperature to $5 \leq \alpha \leq 1000$ in the default setting. Table 16 reports how often the lower bound is reached during training; the upper bound is never reached in these runs.

Table 16: **Frequency of lower-bound activation.** Fraction of sampled (diffusion-time bin, training update) pairs at which the controller reaches its lower bound, pooled over five seeds and all 16 bins within each environment-step interval. The upper bound $\alpha = 1000$ was never reached.

| Environment steps | Ant | HalfCheetah | Hopper | Humanoid | Walker2d |
|---|---|---|---|---|---|
| $\alpha_{\min} = 5$ (default) | | | | | |
| 0–1M | 33.8% | 40.1% | 3.4% | 19.9% | 17.3% |
| 1–2M | 38.9% | 40.8% | 0.0% | 0.0% | 5.5% |
| 2–3M | 23.1% | 22.1% | 0.0% | 0.0% | 0.0% |
| 3–4M | 16.5% | 22.4% | 0.0% | 0.0% | 0.0% |
| 4–5M | 21.5% | 25.1% | 0.0% | 0.0% | 0.0% |
| $\alpha_{\min} = 1$ | | | | | |
| 0–1M | 0.0% | 0.5% | 0.0% | 0.0% | 0.0% |
| 1–2M | 0.3% | 0.0% | 0.0% | 0.0% | 0.0% |
| 2–5M | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |

The lower bound is reached early in all five environments under the default setting. This is consistent with the target ESS starting at 50: a higher ESS requires more diffuse candidate weights, which the controller obtains by decreasing $\alpha$. As the target is annealed toward 5, the required $\alpha$ generally rises and the lower bound becomes inactive on Hopper and Humanoid after 1M steps and on Walker2d after 2M steps. Ant and HalfCheetah continue to reach the lower bound later in training. Their smaller achieved ESS at a shared fixed $\alpha$ in Table 12 is consistent with stronger relative Q-value contrasts among sampled candidates; the clipping frequencies alone do not establish a difference in the absolute scale of the critic. Lowering the bound to 1 makes bound activation rare and removes it after 2M steps in all five environments.

### H.4 BEST-OF-$N$ SELECTION

The best-of-$N$ candidates used to select an action for acting and critic-target computation are distinct from the $K = 100$ candidates used by the score estimator and ESS controller. To assess the contribution of best-of-$N$ selection, we retrain SOM with each value of $N$ while keeping all other settings fixed. The same $N$ is used during training and evaluation.

Table 17: **Number of best-of-$N$ candidates.** Return averaged over the last five evaluations (mean $\pm$ 95% CI over five seeds). Each setting uses the same $N$ for acting and critic-target computation during training and for evaluation. All other settings are fixed. **Bold** marks the highest mean in each column; the lower rows show changes relative to $N = 32$.

| $N$ | Ant | HalfCheetah | Hopper | Humanoid | Walker2d |
|---|---|---|---|---|---|
| 1 | $2307.7 \pm 1256.6$ | $3788.7 \pm 3066.7$ | $107.8 \pm 78.6$ | $7516.5 \pm 449.8$ | $3464.0 \pm 2082.5$ |
| 8 | $6788.2 \pm 606.3$ | $15641.9 \pm 423.4$ | $1638.2 \pm 1268.5$ | **$8561.0 \pm 469.7$** | $5444.9 \pm 576.1$ |
| 16 | $7189.1 \pm 139.6$ | $16232.9 \pm 401.4$ | $2879.2 \pm 852.5$ | $8138.9 \pm 1178.4$ | $5857.5 \pm 174.2$ |
| 32 **(Ours)** | $6947.1 \pm 552.5$ | **$16543.7 \pm 165.8$** | **$3122.2 \pm 445.4$** | $8451.7 \pm 310.1$ | $6109.3 \pm 126.9$ |
| 64 | **$7244.7 \pm 243.2$** | $16373.7 \pm 164.2$ | $2820.3 \pm 372.7$ | $7265.5 \pm 2640.2$ | **$6117.6 \pm 174.9$** |
| *Relative to* $N = 32$ | | | | | |
| $N = 1$ | $-66.8\%$ | $-77.1\%$ | $-96.5\%$ | $-11.1\%$ | $-43.3\%$ |
| $N = 8$ | $-2.3\%$ | $-5.5\%$ | $-47.5\%$ | $+1.3\%$ | $-10.9\%$ |
| $N = 16$ | $+3.5\%$ | $-1.9\%$ | $-7.8\%$ | $-3.7\%$ | $-4.1\%$ |
| $N = 64$ | $+4.3\%$ | $-1.0\%$ | $-9.7\%$ | $-14.0\%$ | $+0.1\%$ |

Table 17 shows that best-of-$N$ selection substantially affects benchmark return. On Hopper-v4, the mean return at $N = 1$ is $96.5\%$ below that at $N = 32$, and $N = 8$ remains $47.5\%$ below. The observed means at $N = 16$ and $N = 32$ are broadly similar across tasks, while $N = 64$ provides no consistent gain. We use $N = 32$ as the default. These returns include the effects of selection during both acting and critic-target computation; the frozen-critic comparison in Figure 4 holds best-of-$N$ fixed at $N = 32$ when comparing update rules.

### H.5 HYPERPARAMETER SETTINGS

For SOM, the entropy parameter adjusts the scale of Gaussian action noise during training. This parameter is distinct from the ESS-controlled score inverse temperature $\alpha_{n,b}$, whose feedback gain $\eta$ is given in Table 15.

Table 18: Hyperparameter settings for all algorithms.

| **Hyperparameter** | **SOM** | **SDAC** | **DPMD** | **QSM** | **DIPO** | **DACER** | **QVPO** | **SAC** | **PPO** | **MVP** |
|---|---|---|---|---|---|---|---|---|---|---|
| Replay buffer capacity | 1e6 | 1e6 | 1e6 | 1e6 | 1e6 | 1e6 | 1e6 | 1e6 | - | 1e6 |
| Buffer warm-up size | 1e5 | 3e4 | 3e4 | 3e4 | 3e4 | 3e4 | 3e4 | 3e4 | - | 3e4 |
| Batch size | 256 | 256 | 256 | 256 | 256 | 256 | 256 | 256 | 4000 | 256 |
| Discount factor $\gamma$ | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.995 | 0.99 |
| Target update rate $\tau$ | 0.005 | 0.005 | 0.005 | 0.005 | 0.005 | 0.005 | 0.005 | 0.005 | - | 0.005 |
| Reward scale | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 1.0 | 0.2 |
| Number of hidden layers | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Number of hidden nodes | 256 | 256 | 256 | 256 | 256 | 256 | 256 | 256 | 512 | 256 |
| Activations | GELU | Mish | Mish | ReLU | Mish | Mish | Mish | GELU | GELU | GELU |
| Diffusion steps | 1 | 20 | 20 | 20 | 100 | 20 | 20 | - | - | 1 |
| Entropy parameter update interval | 250 | 250 | 250 | - | - | 10,000 | - | - | - | - |
| Action gradient steps | - | - | - | - | 30 | - | - | - | - | - |
| Number of Gaussian distributions | - | - | - | - | - | 3 | - | - | - | - |
| Number of action samples | - | - | - | - | - | 200 | - | - | - | - |
| Entropy noise scale | 0.1 | 0.1 | 0.1 | - | - | - | - | - | - | - |
| Optimizer | Adam | Adam | Adam | Adam | Adam | Adam | Adam | Adam | Adam | Adam |
| Actor learning rate | 3e-4 | 3e-4 | 3e-4 | 3e-4 | 3e-4 | 3e-4 | 3e-4 | 3e-4 | 1e-4 | 1e-4 |
| Critic learning rate | 3e-4 | 3e-4 | 3e-4 | 3e-4 | 3e-4 | 3e-4 | 3e-4 | 3e-4 | 1e-3 | 1e-4 |
| Entropy parameter learning rate | 7e-3 | 7e-3 | 7e-3 | - | - | 3e-2 | - | 3e-4 | - | - |
| Target entropy | $-0.9 \cdot \dim(\mathcal{A})$ | $-0.9 \cdot \dim(\mathcal{A})$ | $-0.9 \cdot \dim(\mathcal{A})$ | - | - | $-0.9 \cdot \dim(\mathcal{A})$ | - | $-\dim(\mathcal{A})$ | - | |
| Number of candidates for best-of-$N$ | 32 | 32 | 32 | 32 | 1 | 1 | 32 | 1 | - | 32 |

Table 19: Additional PPO hyperparameters.

| **Hyperparameter** | **Setting** |
|---|---|
| GAE $\lambda$ | 0.95 |
| PPO clip $\epsilon$ | 0.2 |
| Target KL | 0.01 |
| Policy update steps | 80 |
| Value update steps | 80 |