Stochastic MEANFLOW POLICIES: ONE-STEP GENERATIVE CONTROL WITH ENTROPIC MIRROR DESCENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Online off-policy reinforcement learning (RL) is governed by two coupled design dimensions: policy representation (e.g. Gaussian vs generative policies) and optimisation methodology (e.g. soft actor-critic (SAC) vs mirror descent (MD)). Gaussian policies offer fast inference and tractable entropy estimation but have limited ability to model multimodal action distributions, whereas generative policies provide richer action distributions at the cost of iterative sampling or intractable entropy estimation. On the optimisation side, SAC-style entropic exploration and MD updates can be interpreted as minimising distinct Kullback-Leibler divergences: SAC-style exploration performs soft policy improvement towards a value-induced Boltzmann distribution, whereas MD constrains updates to remain close to the previous policy. Combining SAC-style entropy and MD regularisation supports exploration and conservative policy updates, but can yield multimodal targets beyond the representational capacity of unimodal Gaussian policies.

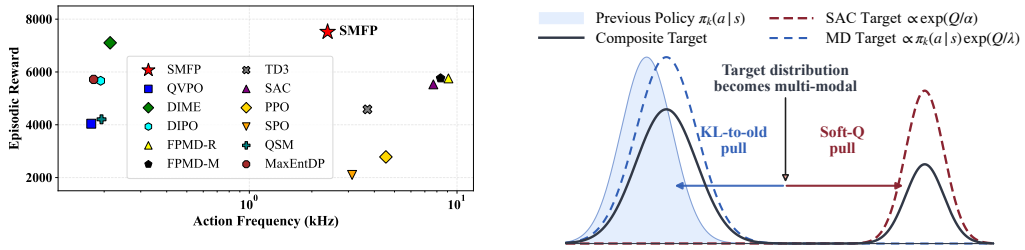
This mismatch calls for an expressive policy with efficient sampling and explicit entropy control. To this end, we introduce Stochastic MeanFlow Policies (SMFP), a one-step generative policy class that combines MeanFlow-based noise-to-action mappings with Gaussian reparameterisation. The latent mapping represents diverse action modes, while the learnable conditional Gaussian scale yields an analytic entropy lower bound. We use this bound to regulate stochasticity alongside direct value guidance and advantage-weighted MeanFlow regression, making entropy-regularised mirror-descent learning practical with one-step generative policies. Empirically, on seven MuJoCo benchmarks, SMFP achieves strong performance against both Gaussian and generative baselines while maintaining single-step inference efficiency. Complementary experiments demonstrate multimodal action modelling and effective exploration in sparse-reward navigation.

1 INTRODUCTION

Online off-policy reinforcement learning (RL) has become a dominant paradigm for solving high-dimensional continuous control tasks, ranging from robotic manipulation to autonomous navigation (Levine, 2018; Gu et al., 2017; Margolis & Agrawal, 2023; Tang et al., 2025). Its success is largely governed by two key design axes: policy representation (*how actions are modelled*) and optimisation methodology (*how policies are updated*).

Policy representation involves a trade-off between tractability and expressivity. Standard actor-critic methods (Fujimoto et al., 2018; Haarnoja et al., 2018) favour tractable policy parameterisations. Gaussian policies provide efficient likelihood and entropy evaluation and fast inference, but have limited flexibility in representing separated action modes. In contrast, diffusion and flow-based policies (Hansen-Estruch et al., 2023; Ghosh et al., 2024; Ding et al., 2024; Celik et al., 2025; Dong et al., 2025; Zhang et al., 2026) offer substantially greater expressivity and can model complex multimodal behaviours. However, these benefits come at the cost of intractable likelihoods and entropy estimation, as well as high inference latency due to iterative sampling (Ho et al., 2020; Liu et al., 2023). One-step MeanFlow policies (Wang et al., 2026) reduce sampling overhead, but their implicit action densities make explicit entropy control challenging.

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.



(a) (b)

Figure 1: (a) Ant-v4 episodic reward versus action frequency on an NVIDIA RTX 5090. (b) Schematic of the composite SAC-MD target, which balances value improvement with proximity to the previous policy. When separated high-value action regions have support under that policy, the target can retain multiple modes, motivating expressive action modelling.

Along another axis of optimisation methodology, a similar tension arises between exploration-oriented entropy maximisation, as in SAC (Haarnoja et al., 2018), and stability-focused trust-region methods such as mirror descent (Wang et al., 2019; Tomar et al., 2022). Entropy-regularised approaches maximise a soft value function augmented with an entropy term, promoting broad exploration and mitigating premature convergence. However, they typically rely on soft maximisation and lack explicit control over policy updates, which can lead to instability during rapid learning. In contrast, mirror-descent-based methods (Beck & Teboulle, 2003; Raskutti & Mukherjee, 2015) constrain or penalise KL divergence from the previous policy, encouraging conservative updates and mitigating abrupt policy changes. Fundamentally, these two optimisation approaches can be interpreted as minimising distinct Kullback-Leibler divergences (Tomar et al., 2022), capturing complementary aspects of policy optimisation: exploration vs stability.

Can SAC-style exploration be combined with the stability of mirror-descent updates? These objectives are complementary: value maximisation favours high-value actions, while mirror descent regularises deviations from the previous policy. Their composite target can retain multiple modes when separated high-value action regions have support under that policy, as illustrated schematically in Figure 1(b). However, standard Gaussian policies are inherently unimodal and therefore fail to represent such targets. This mismatch between optimisation objectives and policy representation raises a fundamental question: *can expressive generative policies bridge this gap?*

SAC-style entropy regularisation requires a tractable estimate of policy entropy. However, for expressive generative policies, exact likelihoods and entropies are generally intractable and often rely on costly estimators or trajectory-level approximations (Hutchinson, 1989; Celik et al., 2025; Zhang et al., 2025). In contrast, diagonal Gaussian parameterisations (Kingma & Welling, 2014; Rezende et al., 2014; Yu et al., 2021) enable efficient stochastic sampling and closed-form entropy control, suggesting a principled route to bridge generative expressivity with entropy-driven optimisation. To this end, we introduce Stochastic MeanFlow Policies (SMFP), which equip a one-step MeanFlow mapping (Wang et al., 2026) with a learnable conditional Gaussian perturbation. The mapping represents diverse action modes, while the conditional scale provides an analytic entropy lower bound. Optimising this surrogate together with direct value gradients and advantage-weighted MeanFlow regression gives the policy explicit stochasticity control without iterative action generation.

Empirically, SMFP achieves strong performance across seven MuJoCo continuous-control benchmarks, matching or outperforming iterative diffusion and flow baselines while retaining efficient one-step inference. Prior work reports limited practical gains from mirror descent in conventional off-policy actor-critic methods (Neumann et al., 2025). Our ablations show that entropy regularisation and mirror-descent regression each improve SMFP, supporting their joint use within the proposed policy. Complementary evaluations on Push-T and sparse-reward AntMaze further demonstrate multimodal action modelling and effective exploration, respectively (Figs. 2b and 3).

2 Related Work

Maximum Entropy Reinforcement Learning. Maximum entropy (MaxEnt) RL, standardised by algorithms like SAC (Haarnoja et al., 2018), balances exploration and exploitation by augmenting the reward objective with entropy regularisation. CrossQ (Bhatt et al., 2024) improves sample efficiency through batch renormalisation (Ioffe, 2017), while BRO (Nauman et al., 2024) combines larger critics with strong regularisation and optimistic exploration. Both retain Gaussian policies with limited flexibility in modelling separated action modes (Chi et al., 2024; Pan et al., 2026). LSAC (Ishfaq et al., 2025) promotes exploration through uncertainty-aware critic learning. Expressive MaxEnt policies include SVGD-based samplers (Haarnoja et al., 2017) and generative models, including diffusion-based algorithms like DIME (Celik et al., 2025), DACER (Wang et al., 2024), and MaxEntDP (Dong et al., 2025) that approximate score functions or entropy surrogates, alongside flow-based methods such as ReinFlow (Zhang et al., 2025) and SAC Flow (Zhang et al., 2026) that leverage velocity reparameterisation or noise injection for tractable entropy maximisation. Recent FLAC (Lv et al., 2026) avoids action-density estimation through kinetic-energy-regularised bridge matching, but regularises paths rather than action entropy.

Mirror Descent Policy Optimisation. Trust-region and proximal methods, notably TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017), seek to stabilise learning by limiting departures from the current policy. Mirror Descent Policy Optimisation (MDPO) (Tomar et al., 2022; Neu et al., 2017; Vieillard et al., 2020) formalises this principle through policy improvement regularised by a Bregman divergence. While recent research aims to instantiate this framework using highly expressive generative models, such as Diffusion (Ho et al., 2020; Song et al., 2021) and Flow Matching (Liu et al., 2023), the direct application of the theoretical MDPO update is impeded by the intractability of evaluating exact likelihoods for these priors. Consequently, prior work uses tractable surrogates to bypass exact density evaluation: DIPO (Yang et al., 2023) improves sampled actions using critic gradients and trains a diffusion policy to imitate them, QVPO (Ding et al., 2024) fits diffusion policies with a Q-weighted variational objective, and FlowRL (Lyu et al., 2025) uses value-weighted flow matching with Wasserstein-2 regularisation.

One-Step Generative Policy. Recent advancements have sought to bypass the computational burden of iterative sampling by adopting one-step generative policies, including distillation-based methods (Park et al., 2025), Consistency Policies (Ding & Jin, 2024), Shortcut Models (Espinosa-Dice et al., 2025), and MeanFlow formulations (Chen et al., 2025; Wang et al., 2026). Although latent noise allows these mappings to generate stochastic actions, the induced marginal density is generally implicit. Efficient one-step sampling therefore does not by itself provide a tractable mechanism for explicit entropy control in online RL.

More Discussion. Unlike prior one-step flow or MeanFlow policies that primarily focus on deterministic fast action generation or mirror-descent projection Chen et al. (2025); Wang et al. (2026); Espinosa-Dice et al. (2025), SMFP explicitly constructs a stochastic one-step generative policy whose conditional variance admits a tractable entropy lower bound. This makes the policy compatible with SAC-style entropy regularisation while retaining the stability of off-policy mirror descent. Thus, the key contribution is not merely replacing diffusion sampling with MeanFlow, but enabling a tractable objective in which entropy-driven exploration, value-weighted mirror descent, and one-step generative inference are jointly optimised.

3 Preliminaries

Online reinforcement learning is modelled as an infinite-horizon Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^d$ is the continuous action space, p is the transition density, r is a bounded reward function, and $\gamma \in [0, 1)$ is the discount factor. We use h to index the discrete RL time step, so the dynamics and rewards are given by $p(s_{h+1}|s_h, a_h)$ and $r(s_h, a_h)$, respectively. In the online off-policy setting, the agent collects experience through environment interaction and stores transitions in a replay buffer $\mathcal{D}$ for off-policy updates. We consider two policy optimisation frameworks: Maximum Entropy RL and Policy Mirror Descent. For notation, the policy and Q-network are parameterised by θ and ϕ, respectively; α denotes the SAC entropy coefficient, and λ denotes the Policy Mirror Descent regularisation coefficient.

3.1 Maximum Entropy Reinforcement Learning

Maximum Entropy RL augments the return with an entropy bonus to encourage exploration. SAC (Haarnoja et al., 2018) instantiates this objective in an off-policy actor-critic framework by maximising $\mathbb{E}_\pi\left[\sum_{h=0}^{\infty}\gamma^h\big(r(s_h,a_h)+\alpha\mathcal{H}(\pi(\cdot|s_h))\big)\right]$, where α sets the entropy weight.

Soft Policy Evaluation. We evaluate $Q_\phi(s,a)$ using the soft Bellman operator

$$\mathcal{T}^\pi Q(s_h,a_h) \triangleq r(s_h,a_h)+\gamma\mathbb{E}_{s_{h+1}\sim p(\cdot|s_h,a_h)}\left[V^\pi(s_{h+1})\right], \tag{1}$$

where $V^\pi(s)=\mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q_\phi(s,a)-\alpha\log\pi(a|s)\right]$. The critic parameters are updated by minimising the squared Bellman residual.

Soft Policy Improvement. The policy improvement step can be written as projecting the parametric policy onto the Boltzmann distribution induced by the soft Q-function:

$$\mathcal{L}_\pi(\theta)=\mathbb{E}_{s_h\sim\mathcal{D}}\left[D_{\mathrm{KL}}\left(\pi_\theta(\cdot|s_h)\middle\|\frac{\exp\big(\frac{1}{\alpha}Q_\phi(s_h,\cdot)\big)}{Z(s_h)}\right)\right]. \tag{2}$$

Using the reparameterisation trick, this yields the standard tractable SAC actor loss

$$\mathcal{L}_\pi(\theta)=\mathbb{E}_{\substack{s_h\sim\mathcal{D}\\ a_h\sim\pi_\theta}}\left[\alpha\log\pi_\theta(a_h|s_h)-Q_\phi(s_h,a_h)\right], \tag{3}$$

where a_h is sampled from the reparameterised policy. In standard SAC, π_θ is typically parameterised as a squashed Gaussian, which enables efficient sampling and likelihood evaluation, with limited flexibility in modelling separated action modes.

3.2 Policy Mirror Descent

Mirror Descent (MD) provides a general framework for regularised policy optimisation in reinforcement learning (Schulman et al., 2015; Tomar et al., 2022). Rather than prescribing a specific objective, MD constrains each policy update to remain close to the previous policy while optimising a chosen policy objective. In this work, we consider a value-based instantiation, where the policy is improved with respect to the action-value estimate. Given π_k, we minimise

$$\mathcal{L}_{\mathrm{MD}}(\pi)=\mathop{\mathbb{E}}_{s\sim\mathcal{D},a\sim\pi}\left[-Q^{\pi_k}(s,a)+\lambda D_{\mathrm{KL}}(\pi(\cdot|s)\,\|\,\pi_k(\cdot|s))\right]. \tag{4}$$

For this value-based objective, the update admits an analytical solution, where the optimal policy is proportional to an energy-based reweighting of the prior policy π_k:

$$\pi_{k+1}(a|s)=\frac{\pi_k(a|s)\exp\left(\frac{1}{\lambda}Q^{\pi_k}(s,a)\right)}{Z(s)}, \tag{5}$$

where $Z(s)=\int\pi_k(a'|s)\exp(\frac{1}{\lambda}Q^{\pi_k}(s,a'))\,da'$ is the partition function. This multiplicative update shifts probability mass towards high-value actions while keeping the policy close to the previous iterate π_k, thereby improving update stability.

3.3 Flow Matching and MeanFlow Variants

Flow Matching learns to transform noise into data via time-dependent velocities, whereas MeanFlow enables one-step sampling by modelling the time-averaged velocity.

Flow Matching. Flow matching (Lipman et al., 2023; Liu et al., 2023) is a generative modelling framework that learns a continuous time-dependent velocity field bridging a simple base distribution and a complex data distribution. In our context, let $e\sim p_{\text{prior}}$ be sampled from a simple prior (e.g. a standard Gaussian) and $a\sim p_{\text{data}}$ from the empirical action data distribution. We consider an interpolation $a_t=(1-t)\,a+t\,e$ for $t\in[0,1]$. Flow matching trains a velocity field $v_\theta(a_t,t)$ to predict the ground-truth displacement between a and e. Specifically, one trains v_θ by minimising:

$$\min_\theta \mathop{\mathbb{E}}_{\substack{a\sim p_{\text{data}},\\ e\sim p_{\text{prior}},\,t\sim\mathcal{U}(0,1)}}\left[\|v_\theta(a_t,t)-(e-a)\|_2^2\right],^{1} \tag{6}$$

[1] State s is also a model input; for compact notation, we denote (s,a_t) simply by a_t.

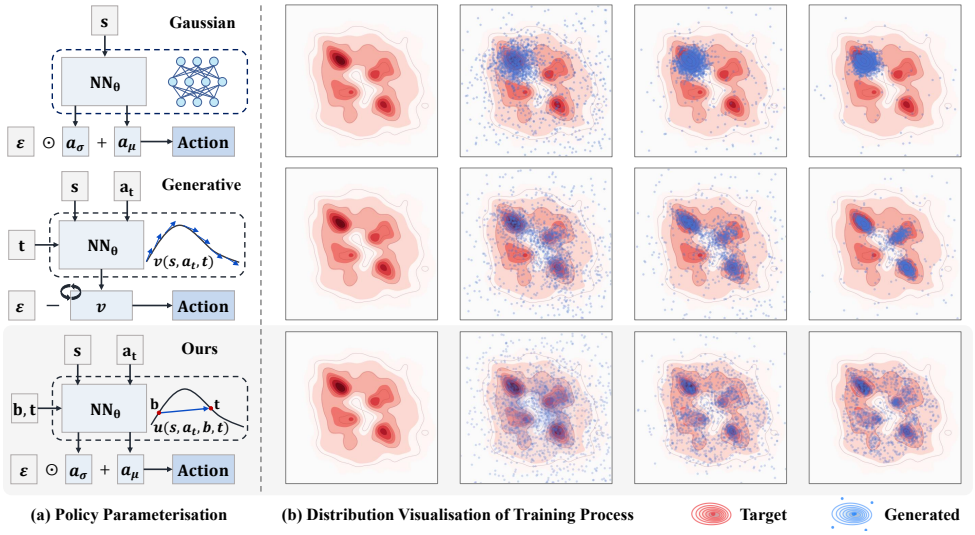


Figure 2: Comparison of policy parameterisations and action-distribution evolution on Push-T. (a) Gaussian, conventional flow-based, and SMFP policies. Conventional flow policies estimate instantaneous velocities $v(s, a_t, t)$; SMFP uses the MeanFlow relation over $[b, t]$ to learn a direct noise-to-action mapping, with Gaussian reparameterisation providing a tractable entropy lower bound. (b) Training evolution of generated action distributions on Push-T. Red contours indicate target modes and blue samples indicate generated actions. SMFP maintains samples around the separated target modes across the displayed stages, illustrating multimodal action modelling. The complementary AntMaze evaluation in Fig. 3 examines exploration and sustained goal-reaching performance.

where $a_t = (1-t)a+te$ as above. By learning to predict $e-a$, the model captures the time-dependent instantaneous velocity needed to transform a into e (or vice versa).

MeanFlow and Variants. MeanFlow (Geng et al., 2025) builds on flow matching by introducing an *average velocity* field $u(a_t, b, t)$. For any $0 \le b < t \le 1$, the average velocity is defined as the time-average of the instantaneous velocity v over the interval $[b, t]$:

$$u(a_t, b, t) \;:=\; \frac{1}{t-b}\int_b^t v(a_\tau, \tau)\, d\tau. \tag{7}$$

Differentiating both sides with respect to t (treating b as constant) yields the *MeanFlow Identity* (Geng et al., 2025):

$$u(a_t, b, t) \;=\; v(a_t, t) \;-\; (t-b)\,\frac{d}{dt}u(a_t, b, t). \tag{8}$$

MeanFlow establishes a principled relationship between the average velocity u and the instantaneous velocity v. Crucially, it enables *one-step* generation: by choosing $b = 0$ and $t = 1$ in (7), one obtains the one-step sampler $\hat{a} = e - u_\theta(e, b = 0, t = 1)$. This formula generates $\hat{a}$ through a single velocity estimate from the learned u_θ, without time integration.

For Q-learning, residual MeanFlow (Wang et al., 2026) directly parameterises the mapping from latent noise to actions:

$$f_\theta(a_t, b, t) = a_t - u_\theta(a_t, b, t). \tag{9}$$

This data-prediction parameterisation (Li & He, 2025) outputs an action in one forward pass. Its differentiable noise-to-action mapping also allows direct optimisation through the critic.

4 Stochastic MeanFlow Policy Framework

SMFP combines one-step generative action modelling with explicit stochasticity control and off-policy improvement (Fig. 2a). A residual MeanFlow mapping represents diverse action regions, while a learnable conditional Gaussian perturbation controls dispersion around its outputs. We first define this policy, then derive a tractable entropy surrogate and combine it with value-weighted MeanFlow regression for off-policy learning.

4.1 STOCHASTIC MEANFLOW POLICY ARCHITECTURE

We use the residual MeanFlow formulation (Wang et al., 2026) as the policy backbone. It implements data prediction (x-prediction) (Li & He, 2025), enabling one-step generation and direct Q-gradient propagation without differentiating through an ODE solver.

Stochastic Reparameterisation. The MeanFlow mapping generates diverse actions from latent noise, while Gaussian reparameterisation equips each latent-conditioned output with an explicit density and a learnable noise scale. Figure 2a illustrates how this mechanism extends Eq. 9:

$$g_\theta(a_t, b, t) = \underbrace{a_t - u_\theta(a_t, b, t)}_{Action_\mu} + \underbrace{\sigma_\theta(a_t, b, t)}_{Action_\sigma} \odot \epsilon, \tag{10}$$

where g_θ denotes the generative policy network and $a_t = (1-t)a + te$ is the interpolated action. Here, e and ϵ are independent standard Gaussian noise variables. The two noise variables play different roles: e determines the latent generative branch of the one-step MeanFlow mapping, while ϵ parameterises the conditional stochasticity around that branch and is the source of the tractable entropy surrogate. Here, u_θ defines the residual mean $a_t - u_\theta$, σ_θ predicts the conditional standard deviation, and $\odot$ denotes element-wise multiplication. The one-step generative policy is then obtained by setting $b = 0$, $t = 1$, and $a_1 = e$, yielding the predicted action $\hat{a} = g_\theta(a_t = e, b = 0, t = 1)$.

Optimisation Objective. We combine the SAC objective (Eq. 3) with a KL penalty relative to the previous policy $\pi_{\theta_{\text{old}}}$ (Eq. 4). Entropy encourages exploration, whereas the KL penalty moderates changes relative to the previous policy. At iteration k, the combined objective is:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{s\sim\mathcal{D}}\Big[- Q(s, g_\theta(s)) + \alpha \log \pi_\theta(\cdot|s) + \lambda D_{\text{KL}}(\pi_\theta(\cdot|s) \| \pi_{\theta_{\text{old}}}(\cdot|s))\Big], \tag{11}$$

where α weights entropy regularisation, and λ parameterises the inverse mirror descent step size. Here, g_θ serves as the reparameterised sampler for action generation, while π_θ denotes the induced density for computing the entropy and KL terms. This formulation presents two computational challenges specific to implicit generative policies: (1) calculating the log-likelihood $\log \pi_\theta(a|s)$ for a flow-based distribution without explicit density, and (2) computing the KL divergence between two stochastic MeanFlow policies. To address these, we derive a tractable entropy surrogate in Sec. 4.2 and formulate the mirror descent training objective in Sec. 4.3.

4.2 TRACTABLE ENTROPY REGULARISATION

For generative policies, SAC-style entropy regularisation is challenging because the marginal density is implicit. Density tracking and trajectory-level approximations can add substantial overhead to online RL (Hutchinson, 1989; Grathwohl et al., 2019; Celik et al., 2025; Zhang et al., 2025). SMFP instead uses its explicit conditional Gaussian policy to control stochasticity. Conditioned on $e \sim \mathcal{N}(0, I)$, the action has a tractable density, while the marginal policy $\pi_\theta(a|s) = \int \pi_\theta(a|s, e)p(e)\,\mathrm{d}e$ retains the expressivity of a continuous mixture. Proposition C.1 establishes a conditional-entropy lower bound on the marginal entropy. Dropping the additive Gaussian constant gives

$$\widetilde{\mathcal{H}}(\pi_\theta \mid e) = \mathbb{E}_e\left[\sum_{i=1}^{d} \log \sigma_\theta^{(i)}(a_t, b, t)\right]. \tag{12}$$

This lower bound makes the expected log-standard deviation a tractable measure of conditional dispersion. It complements the variation induced by the latent mapping, giving the expressive policy an explicit mechanism for controlling stochasticity. Proposition C.2 provides a complementary diffuse-prior interpretation. In practice, we encourage a minimum noise scale through a hinge penalty, which activates when the normalised log-standard deviation falls below a threshold:

$$\mathcal{L}_{\text{Ent}}(\theta) = \mathbb{E}\left[\text{ReLU}\left(\kappa_\sigma - \frac{1}{d}\sum_{i=1}^{d} \log \sigma_\theta^{(i)}(a_t, b, t)\right)\right]. \tag{13}$$

Here, d is the action dimension and κ_σ is the threshold for the average log-scale. The penalty encourages dispersion below this threshold and becomes inactive above it, where the remaining actor terms continue to shape the scale. Appendix C.1 relates this regulariser to the entropy bound; Figs. 4(a)–4(b) evaluate its effect on learning.

4.3 Optimisation via Off-Policy Mirror Descent

For the value-based mirror-descent component, Eq. (5) gives the energy-based target $\pi^*(\cdot|s) \propto \pi_{\theta_{\text{old}}}(\cdot|s)\exp(Q(s,\cdot)/\lambda)$. This target motivates weighted regression towards high-value actions under the previous policy, alongside the entropy control in Sec. 4.2.

Since π^* is a non-parametric and unnormalised target, we do not sample from it directly. Instead, following weighted policy projection (Abdolmaleki et al., 2018; Chen et al., 2025), we use samples from the previous policy as proposals and weight them according to their relative target density.

$$\omega(s,a) \propto \exp\left(\frac{Q(s,a)}{\lambda}\right) \propto \exp\left(\frac{A^{\pi_{\theta_{\text{old}}}}(s,a)}{\lambda}\right), \tag{14}$$

where the second proportionality holds because the state-value baseline is action-independent.

For SMFP, however, the marginal likelihood required by this projection is intractable. We therefore retain the weighting induced by the mirror-descent target, but replace the marginal log-likelihood fitting term with the tractable stochastic MeanFlow regression surrogate:

$$\mathcal{L}_{\text{SMF}}(\theta; a, s) = \mathcal{L}_\delta\left(g_\theta(a_t, b, t) - \text{sg}(g_{\text{tgt}})\right), \tag{15}$$

where $\mathcal{L}_\delta$ is the Huber loss, and $\text{sg}(\cdot)$ denotes the stop-gradient operator. Following the MeanFlow identity, the target is

$$g_{\text{tgt}} \triangleq a_t + (t-b-1)v(a_t,t) + \sigma_\theta \odot \epsilon - (t-b)\left[\texttt{jvp}(g_\theta,\cdot) - \texttt{jvp}(\sigma_\theta,\cdot)\odot\epsilon\right], \tag{16}$$

with the derivation provided in Appendix C.2. In practice, noisy online value estimates can make the exponential weights in Eq. (14) unstable. Following QVPO (Ding et al., 2024), we use the positive part of the estimated advantage:

$$\omega_{\text{adv}}(s,a) = \max\big(0, Q(s,a) - V(s)\big), \tag{17}$$

with the rationale given in Appendix C.3. This also avoids a separate value network: $V(s)$ is estimated by averaging the Q-values of N_{adv} actions drawn directly from the pre-update policy. The resulting mirror-descent regression objective is

$$\mathcal{L}_{\text{MD}}(\theta) = \mathbb{E}_{a\sim\pi_{\theta_{\text{old}}}}\left[\omega_{\text{adv}}(s,a)\mathcal{L}_{\text{SMF}}(\theta; a, s)\right]. \tag{18}$$

We combine these components into a tractable actor objective: the direct Q-gradient improves generated actions, the conditional entropy-floor loss maintains stochasticity, and the advantage-weighted MeanFlow loss fits high-value policy samples. Together, these terms implement the value-improvement, entropy-control, and policy-regularisation roles motivated by Eq. (11):

$$\mathcal{L}(\theta) = \mathbb{E}_{s\sim\mathcal{D}}\left[-Q(s, g_\theta(\cdot)) + \alpha\mathcal{L}_{\text{Ent}}(\theta) + \lambda\mathcal{L}_{\text{MD}}(\theta)\right]. \tag{19}$$

Algorithm 1 summarises training. For interaction and critic targets, value-guided (best-of-K) selection chooses the highest-Q action from K_b and K_t candidates, respectively, with one policy evaluation per candidate. Implementation details appear in Appendix B.1.

Algorithm 1 Stochastic MeanFlow Policy Optimisation

Initialise replay buffer $\mathcal{D}$, policy g_θ, critic Q_ϕ, and target critic $Q_{\bar{\phi}}$.
for $i = 1$ **to** N **do**
  Collect transitions with value-guided actions and store them in $\mathcal{D}$.
  Sample a mini-batch (s, a, r, s') from $\mathcal{D}$ and set $\theta_{\text{old}} \leftarrow \theta$.
  ▷ Train Stochastic One-Step MeanFlow Policy π_θ
  Draw N_{adv} actions directly from $\pi_{\theta_{\text{old}}}(\cdot \mid s)$; estimate $V(s)$ and weights via Eq. (17).
  Construct targets (Eq. (16)) and update θ by minimising Eq. (19). ▷ SMFP Loss
  ▷ Train Critic Q_ϕ
  Select value-guided next actions and update ϕ via Eq. (21). ▷ Critic Loss
  Update the target critic $\bar{\phi}$ by exponential moving average.
return One-step SMFP policy π_θ.

Table 1: Performance comparison of SMFP with 15 other online RL algorithms across 20 random seeds. The corresponding learning curves are shown in Fig. 7.

Method		NFE	HOPPER	WALKER2D	ANT	HALFCHEETAH	HUMANOID	HUMANOIDSTANDUP	SWIMMER
Gaussian Policy RL	PPO	1	3154.3 ± 426.2	3751.5 ± 609.1	2781.9 ± 74.1	4773.5 ± 53.4	713.7 ± 85.9	56368.4 ± 563.2	86.2 ± 12.3
	SPO	1	2212.8 ± 988.4	3321.8 ± 1328.9	2100.2 ± 302.4	4008.2 ± 246.8	797.4 ± 262.1	62875.2 ± 3521.8	58.4 ± 14.2
	TD3	1	3267.5 ± 8.5	3513.9 ± 40.7	4583.8 ± 69.5	10388.6 ± 80.4	5353.5 ± 53.7	105243.8 ± 637.1	80.4 ± 11.2
	SAC	1	2996.6 ± 111.9	4988.1 ± 80.0	5530.6 ± 1000.3	10616.9 ± 72.8	5159.7 ± 475.3	116073.6 ± 1254.4	69.2 ± 10.1
Diffusion Policy RL	DIPO	20	1295.4 ± 7.0	2181.7 ± 1625.7	5665.9 ± 14.7	9590.5 ± 67.5	4945.5 ± 898.6	105163 ± 2162.3	52.2 ± 7.3
	QSM	20	2154.7 ± 998.2	3613.4 ± 1443.5	4206.4 ± 143.5	3888.2 ± 632.6	4793.1 ± 229.5	97120.1 ± 927.1	61.6 ± 7.3
	QVPO	20	3364.4 ± 325.0	3826.3 ± 1909.4	4040.1 ± 1085.6	8759.4 ± 696.3	5176.6 ± 410.4	137754.7 ± 80529.5	112.4 ± 15.1
	DIME*	16	2567.6 ± 46.3	6447.8 ± 104.6	7103.6 ± 85.2	13464.1 ± 139.2	**11814.1 ± 164.1**	111336.9 ± 744.5	118.8 ± 4.5
	MaxEntDP	20	3520.9 ± 67.3	5142.2 ± 98.2	5717.9 ± 134.5	11281.5 ± 45.1	5749.6 ± 105.6	158656.4 ± 1432.8	90.3 ± 22.7
	DPMD	20	3317.9 ± 55.2	5023.2 ± 96.4	5931.8 ± 104.6	12931.4 ± 441.1	6835.4 ± 532.8	186483.3 ± 1613.1	80.2 ± 53.3
Flow Policy RL	SAC Flow-T	4	3457.1 ± 54.1	5676.5 ± 110.2	5787.6 ± 165.4	14897.8 ± 132.7	9821.1 ± 241.5	306937.8 ± 1876.3	101.5 ± 11.2
	SAC Flow-G	4	3440.3 ± 67.5	5094.1 ± 90.7	5847.1 ± 151.3	12432.8 ± 142.5	8163.3 ± 201.7	368505.1 ± 2248.2	98.2 ± 12.5
	FlowRL	1	2784.6 ± 215.4	4564.7 ± 312.1	5513.3 ± 287.6	8963.4 ± 198.5	5560.7 ± 145.2	151870.6 ± 1234.9	48.7 ± 9.6
	FPMD-R	1	3491.4 ± 61.2	4134.5 ± 596.5	5756.2 ± 82.5	11043.6 ± 612.2	6752.6 ± 326.7	205336.8 ± 1652.4	62.2 ± 10.3
	FPMD-M	1	3063.4 ± 693.2	4723.6 ± 312.4	5762.4 ± 152.5	9948.2 ± 652.2	6227.9 ± 621.1	168924.5 ± 2045.1	55.4 ± 9.6
Proposed	**SMFP**	1	**3825.8 ± 29.1**	**6776.5 ± 117.4**	**7518.3 ± 77.1**	**15769.3 ± 125.5**	10245.7 ± 93.5	**376132.6 ± 1107.4**	**137.2 ± 11.3**

* DIME parameterises its critic with a distributional Q-function, which can provide richer value information than the scalar Q-values commonly used in other methods.

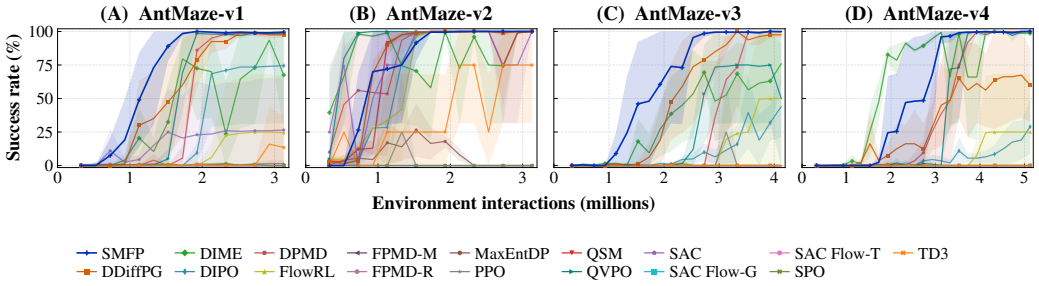


Figure 3: Exploration in sparse-reward AntMaze tasks. Curves show mean success rates over 10 training seeds; shaded regions indicate one standard deviation. NovelD-style intrinsic rewards are used by all compared methods.

5 EXPERIMENTS

We evaluate SMFP on MuJoCo benchmarks following the general experimental setup. We further visualise its multimodal action distributions on Push-T (Chi et al., 2024) and assess its exploration capabilities on the sparse-reward AntMaze tasks introduced in DDiffPG (Li et al., 2024).

MuJoCo Environments and Training Details. We evaluate on the seven widely adopted Gym MuJoCo benchmarks (Brockman et al., 2016; Todorov et al., 2012) to align with established baselines. For SMFP, we report mean episodic rewards and standard deviations over 20 random seeds. Baseline entries combine reproduced and comparable published results, selected as detailed in Appendix E.1.

MuJoCo Baselines. We compare three policy classes: (1) Gaussian or deterministic policies with single-step inference (1-NFE): PPO (Schulman et al., 2017), SPO (Xie et al., 2025), TD3 (Fujimoto et al., 2018), and SAC (Haarnoja et al., 2018). (2) Diffusion-based policies: DIPO (Yang et al., 2023), QSM (Psenka et al., 2024), QVPO (Ding et al., 2024), DIME (Celik et al., 2025), MaxEntDP (Dong et al., 2025), and DPMD (Ma et al., 2025). (3) Flow-based policies: SAC Flow (Zhang et al., 2026), FlowRL (Lyu et al., 2025), and FPMD (Chen et al., 2025).

5.1 COMPARATIVE PERFORMANCE EVALUATION

SMFP achieves the highest reported mean return on six of the seven MuJoCo tasks and ranks second on Humanoid, while using one-step action generation (Tab. 1). It also outperforms the Gaussian and deterministic baselines and the other one-step flow policies across the benchmark suite. These gains span locomotion and high-dimensional humanoid control, demonstrating effective policy optimisation across varied dynamics and action spaces with one-step generation. Appendix D.2 provides learning curves from reproduced runs (Fig. 7).

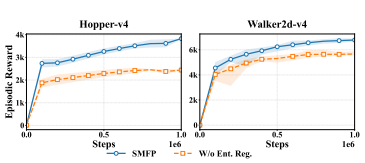


Figure 4(a): Comparison of SMFP and SMFP without the entropy regulariser. Curves show the mean over 5 seeds.

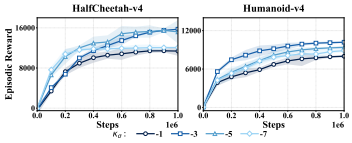


Figure 4(b): Performance comparison under different κ_σ settings. Curves show the mean over 5 seeds.

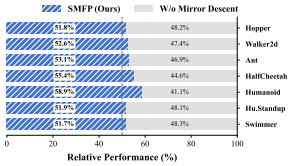


Figure 4(c): Relative performance of SMFP and its variant with mirror descent removed.

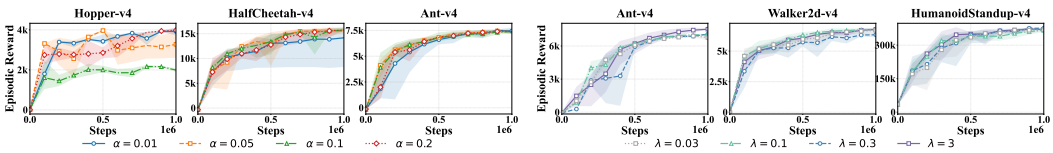


Figure 5: Sensitivity analysis of the entropy temperature parameter α and the mirror descent regularisation coefficient λ on MuJoCo benchmarks. Curves show the mean over 5 seeds.

Multimodal expressivity and exploration. Figure 3 compares SMFP with Gaussian, diffusion, and flow baselines on four sparse-reward AntMaze tasks (Li et al., 2024), using shared NovelD-style intrinsic rewards (Zhang et al., 2021) and matched interaction budgets. SMFP achieves and sustains near-ceiling success across all four tasks, including AntMaze-v3, the most challenging task in our evaluation (see Appendix D.1 for task details). These results support effective exploration and reliable goal-reaching under sparse rewards, without the hierarchical trajectory clustering or mode-specific critics used by DDiffPG (Li et al., 2024).

5.2 Further Analysis

Effect of the entropy regulariser. Removing the entropy regulariser slows learning and lowers final returns on both tasks in Fig. 4(a). Explicit control of conditional stochasticity therefore improves learning beyond the diversity supplied by the latent mapping alone.

Sensitivity to the noise-scale threshold. Fig. 4(b) evaluates the effect of the log-scale threshold $\kappa_\sigma \in \{-1, -3, -5, -7\}$ in Eq. (13). Performance varies non-monotonically with this threshold: less negative values enforce stronger stochasticity, while more negative values weaken the entropy constraint and may reduce exploration. Overall, $\kappa_\sigma = -3$ achieves the best trade-off, indicating that the noise-scale threshold affects the balance between exploration and refinement.

Effect of mirror-descent regularisation. We isolate the contribution of mirror descent by comparing SMFP with a variant that removes the mirror-descent term. As shown in Fig. 4(c), SMFP consistently outperforms this variant across all benchmarks, with relatively larger gains on high-dimensional tasks such as Humanoid. These gains support value-weighted policy fitting for control.

Sensitivity to optimisation hyperparameters. Fig. 5 studies the sensitivity of SMFP to the entropy temperature α and the mirror-descent coefficient λ. Overall, SMFP is stable across a moderate range of settings. The entropy temperature α has a more visible effect on some tasks, such as Hopper, where overly large values can slow policy refinement. Performance is comparatively robust to λ within the tested range. Appendix F lists the full hyperparameter settings.

6 Conclusion

In this work, we introduced Stochastic MeanFlow Policies (SMFP), a one-step generative policy framework for online off-policy reinforcement learning. SMFP augments MeanFlow policies with an explicit stochastic reparameterisation and a tractable conditional entropy surrogate, enabling SAC-style exploration to be combined with off-policy mirror-descent optimisation. This design addresses the tension between expressive action modelling, efficient inference, and tractable entropy control. Empirically, across seven MuJoCo benchmarks, SMFP achieves strong performance against both Gaussian and generative baselines with single-step inference. Push-T and AntMaze further demonstrate multimodal action modelling, sparse-reward exploration, and reliable goal attainment.

AI USE STATEMENT

After completing the initial draft, we used LLM-based tools solely to correct grammar and improve the clarity and readability of the manuscript. Their use was limited to language editing and did not extend to research ideation, experimental design, scientific analysis, or the generation of substantive scientific content. All research contributions, analyses, and conclusions are the authors' own. No LLM is credited with authorship, and the authors take full responsibility for the accuracy and integrity of the entire manuscript, including all passages revised with LLM assistance.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility of our work, we provide detailed descriptions of the experimental environments, hyperparameters, and other experimental details used in our study in Section 5 and the appendices. Our code and model checkpoint will be released publicly upon acceptance to further support reproducibility and future research.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations (ICLR)*, 2018.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning (ICML)*, 2017.

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

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. DIME: Diffusion-based maximum entropy reinforcement learning. In *International conference on machine learning (ICML)*, 2025.

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

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*, 2024.

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 (NeurIPS)*, 2024.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Xiaoyi Dong, Jian Cheng, and Xi Sheryl Zhang. Maximum entropy reinforcement learning with diffusion policy. In *International Conference on Machine Learning (ICML)*, 2025.

Nicolas Espinosa-Dice, Yiyi Zhang, Yiding Chen, Bradley Guo, Owen Oertell, Gokul Swamy, Kianté Brantley, and Wen Sun. Scaling offline RL via efficient and expressive shortcut models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations (ICLR)*, 2019.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning (ICML)*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning (ICML)*, 2018.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

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

Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Haque Ishfaq, Guangyuan Wang, Sami Nur Islam, and Doina Precup. Langevin soft actor-critic: Efficient exploration through uncertainty-driven critic learning. In *International Conference on Learning Representations(ICLR)*, 2025.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Sergey Levine. Reinforcement learning and control as probabilistic inference: tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

Zechu Li, Rickmer Krohn, Tao Chen, Anurag Ajay, Pulkit Agrawal, and Georgia Chalvatzaki. Learning multimodal behaviors from scratch with diffusion policy gradient. *Advances in Neural Information Processing Systems(NeurIPS)*, 37:38456–38479, 2024.

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

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, and Xiao Ma. FLAC: Maximum entropy RL via kinetic energy regularized bridge matching. *arXiv preprint arXiv:2602.12829*, 2026.

Lei Lyu, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Annual Conference on Robot Learning*. PMLR, 2023.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized Markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

Samuel Neumann, Jiamin He, Adam White, and Martha White. Investigating the utility of mirror descent in off-policy actor-critic. In *Reinforcement Learning Conference (RLC)*, 2025.

Chaoyi Pan, Giri Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, et al. Much ado about noising: Dispelling the myths of generative robotic control. In *International Conference on Learning Representations (ICLR)*, 2026.

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-learning. In *International conference on machine learning (ICML)*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, pp. 4195–4205, 2023.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via Q-score matching. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Garvesh Raskutti and Sayan Mukherjee. The information geometry of mirror descent. *IEEE Transactions on Information Theory*, 61(3):1451–1457, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

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 (ICLR)*, 2021.

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes. *Annual Review of Control, Robotics, and Autonomous Systems*, 2025.

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 (IROS)*, 2012.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. In *International Conference on Learning Representations (ICLR)*, 2022.

Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist. Leverage the average: an analysis of KL regularization in reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Qing Wang, Yingru Li, Jiechao Xiong, and Tong Zhang. Divergence-augmented policy optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

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. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zeyuan Wang, Da Li, Yulin Chen, Ye Shi, Liang Bai, Tianyuan Yu, and Yanwei Fu. One-step generative policies with Q-learning: A reformulation of MeanFlow. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

Zhengpeng Xie, Qiang Zhang, Fan Yang, Marco Hutter, and Renjing Xu. Simple policy optimization. In *Proceedings of the 42nd International Conference on Machine Learning(ICML)*, volume 267, pp. 68813–68824. PMLR, 2025.

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.

Tianyuan Yu, Yongxin Yang, Da Li, Timothy Hospedales, and Tao Xiang. Simple and effective stochastic neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. NovelD: A simple yet effective exploration criterion. *Advances in Neural Information Processing Systems(NeurIPS)*, 34:25217–25230, 2021.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning flow matching policy with online reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yixian Zhang, Shu'ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. SAC flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. *International Conference on Learning Representations(ICLR)*, 2026.

A Limitations and Future Work

Although the hybrid optimisation framework is broadly applicable in principle, the tractable surrogate objective developed here relies on the reparameterised structure of the MeanFlow policy. Extending it to other generative policy classes may therefore require policy-specific entropy bounds and regularisation schemes. Within this framework, we keep α and λ fixed throughout training to avoid confounding scheduling effects and to isolate the contribution of the proposed objective and policy parameterisation. Adaptive weighting of α and λ is a natural direction for extending this framework. We will also investigate stronger critic architectures, such as CrossQ (Bhatt et al., 2024) and distributional Q-learning (Bellemare et al., 2017), for improving value estimation and overall training behaviour.

B SMFP Training Procedure

B.1 Implementation Details

Value-Guided Action Selection. Following the value-guided candidate selection used in QVPO and MeanFlowQL (Ding et al., 2024; Wang et al., 2026), we apply Best-of-K selection to favour high-value actions. For each state, SMFP generates K candidates in parallel using JAX `vmap`, with one network evaluation per candidate. The critic then selects the action:

$$a_\theta^K(s) \triangleq \underset{a\in\{a_1,\ldots,a_K\sim g_\theta(s,e,b=0,t=1)\}}{\operatorname{argmax}} Q(s,a). \tag{20}$$

This selection operates on completed action samples; each candidate retains single-step generation without iterative denoising or ODE integration. The candidate-size ablation in Fig. 8(a) characterises its effect, and Tab. 2 reports the computational cost of SMFP. Following QVPO (Ding et al., 2024), we use separate candidate counts for behaviour inference and target estimation, denoted by K_b and K_t, and choose a smaller target pool with $K_t = 0.5K_b$.

Critic Implementation. We train the critic using the standard target-Q architecture. Given critic parameters ϕ and target parameters $\bar{\phi}$, the critic is updated by minimising the Bellman error:

$$\mathcal{L}_{\text{Critic}}(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_\phi(s,a) - \bar{Q}\right)^2\right], \tag{21}$$

where the target is computed using the target critic with stop-gradient:

$$\bar{Q} = r + \gamma\left[Q_{\bar{\phi}}(s',a') + \alpha\sum_{i=1}^{d}\log\sigma_\theta^{(i)}(s',e)\right]_{\text{sg}}. \tag{22}$$

Here, a' is generated by the current one-step MeanFlow policy at s', and the entropy term follows the lower bound in Proposition C.1. The target parameters $\bar{\phi}$ are updated by an exponential moving average. Algorithm 1 in the main text summarises the training procedure.

C Proofs

C.1 Theoretical Derivations for Entropy Bounds

The stochastic reparameterisation in Eq. 10 separates the latent input e from the independent Gaussian perturbation ϵ. At a fixed state and input (a_t, b, t), randomness in ϵ induces the conditional density $\pi_\theta(\cdot \mid a_t, b, t) = \mathcal{N}(\mu_\theta, \Sigma_\theta)$, with mean $\mu_\theta = a_t - u_\theta(a_t, b, t)$ and covariance $\Sigma_\theta = \operatorname{diag}(\sigma_\theta^2)$. For one-step action generation, $a_t = e$, $b = 0$, and $t = 1$, so conditioning on e fixes the mean and scale of this Gaussian. We suppress the state argument below and use differential entropy throughout.

Integrating over e produces a continuous mixture of these conditional Gaussians. Its means and scales vary with the latent input, allowing the marginal policy to represent multiple action modes. The resulting marginal density is implicit, whereas each conditional entropy is available analytically. The following decomposition identifies the component of marginal entropy that SMFP can regulate directly:

Proposition C.1 (**Tractable Entropy Lower Bound**). *At a fixed state, consider the one-step policy with $a_t = e$, $b = 0$, and $t = 1$, positive conditional standard deviations, and finite entropies. The latent variable $e \sim \mathcal{N}(0, I)$ is distinct from Euler's number* e.

The marginal entropy decomposes into the expected entropy of the conditional action distribution and the mutual information between the generated action and e. In this notation, $\mathcal{H}(\pi_\theta \mid e)$ includes the expectation over latent inputs. Since mutual information is non-negative,

$$\mathcal{H}(\pi_\theta) = \mathcal{H}(\pi_\theta \mid e) + \underbrace{I_\theta(\pi_\theta; e)}_{\geq 0} \geq \mathcal{H}(\pi_\theta \mid e). \tag{23}$$

The conditional term affords an analytical solution based on the predicted Gaussian parameters:

$$\begin{aligned} \mathcal{H}(\pi_\theta \mid e) &= \mathbb{E}_e \left[\sum_{i=1}^{d} \log \sigma_\theta^{(i)}(a_t, b, t) \right] + \underbrace{\frac{d}{2} \log(2\pi \mathrm{e})}_{const.} \\ \Rightarrow \quad \tilde{\mathcal{H}}(\pi_\theta \mid e) &= \mathbb{E}_e \left[\sum_{i=1}^{d} \log \sigma_\theta^{(i)}(a_t, b, t) \right]. \end{aligned} \tag{24}$$

The additive Gaussian constant is independent of θ, so the expected sum of log-standard deviations supplies the parameter-dependent part of this lower bound. The mutual-information term captures the dependence of generated actions on the latent input. SMFP therefore combines latent-dependent action generation with direct control of the conditional dispersion around each generated mean.

Equation (13) implements this control using the average log-standard deviation across action dimensions. This quantity is the logarithm of the geometric mean of the conditional standard deviations, so the threshold acts on their aggregate scale while permitting different scales in different dimensions. Below κ_σ, minimising the hinge loss increases this average log-scale. Once the threshold is reached, that sample contributes no further expansion pressure from the entropy term.

The hinge is applied before the expectation in Eq. (13), allowing the penalty to respond to individual sampled inputs with low conditional dispersion. It is a soft regulariser whose strength is set by α in the actor objective. This construction makes the conditional scale directly trainable without evaluating the mutual-information term or the marginal density. The performance gains from the entropy regulariser and the sensitivity to κ_σ in Figs. 4(a)–4(b) support this form of stochasticity control.

The same surrogate has a diffuse-prior interpretation. For the conditional Gaussian policy and $p_0 = \mathcal{N}(\mu_0, \tau^2 I)$, the parameter-dependent part of the KL divergence separates conditional entropy from a second-moment penalty.

Proposition C.2 (**Max-Entropy Derivation**). *For each fixed conditional Gaussian policy with finite second moments, the parameter-dependent KL objective converges to its negative entropy, up to an additive constant, as $\tau \to \infty$. Averaging over e yields the conditional-entropy surrogate in Proposition C.1.*

Proof. For the conditional policy, the Gaussian KL consists of a negative log-scale term, a squared distance between means, and a covariance term. Substituting the mean $a_t - u_\theta$ and removing the additive term $d \log \tau - d/2$, which is independent of θ, gives:

$$\mathcal{J}(\theta) = -\sum_i \log \sigma_\theta^{(i)} + \frac{1}{2\tau^2} \left(\|a_t - u_\theta - \mu_0\|^2 + \|\sigma_\theta\|_F^2 \right). \tag{25}$$

The two squared norms measure the conditional second moment about the prior mean. Increasing τ reduces their contribution relative to the negative log-scale term. For fixed θ, this finite second-moment contribution vanishes as $\tau \to \infty$, giving:

$$\lim_{\tau \to \infty} \mathcal{J}(\theta) = -\sum_i \log \sigma_\theta^{(i)}. \tag{26}$$

If the second moment is integrable over e and the expected log-scales are finite, the same decomposition holds after averaging over latent inputs. The limiting parameter-dependent objective is then the

negative conditional-entropy surrogate in Proposition C.1. This limit describes the objective for fixed policy parameters after the prior-dependent additive term has been removed.

The diffuse-prior calculation explains why the log-scale term favours conditional dispersion. The practical hinge loss uses this direction of regularisation below a prescribed threshold, while allowing the value objective to guide action quality. Thus, the lower-bound argument identifies the computable entropy component, and the diffuse-prior limit explains the associated regularisation. □

C.2 Derivation of the Stochastic MeanFlow Objective

In this section, we derive the training objective for Stochastic MeanFlow. Our goal is to extend the deterministic MeanFlow identity to support a stochastic generative policy g_θ, parameterised by a residual flow with learnable variance.

Following the reformulation principles introduced in MeanFlowQL (Wang et al., 2026), we define the stochastic policy $g_\theta(a_t, b, t)$ by decomposing the trajectory into a deterministic residual and a stochastic perturbation, and obtain the parameterisation:

$$g_\theta(a_t, b, t) = a_t - u_\theta(a_t, b, t) + \sigma_\theta(a_t, b, t) \odot \epsilon, \quad \text{where } \epsilon \sim \mathcal{N}(0, I). \tag{27}$$

Rearranging Eq. 27, we express the underlying velocity generator u_θ as:

$$u_\theta(a_t, b, t) = a_t - g_\theta(a_t, b, t) + \sigma_\theta(a_t, b, t) \odot \epsilon. \tag{28}$$

A fundamental property of the MeanFlow framework (Geng et al., 2025) is the consistency between the generator u_θ and the vector field $v(a_t, t)$. This is governed by the *MeanFlow Identity*:

$$u_\theta(a_t, b, t) = v(a_t, t) - (t - b)\frac{d}{dt}u_\theta(a_t, b, t). \tag{29}$$

Substituting the expression for u_θ (Eq. 28) into Eq. 29, we obtain:

$$a_t - g_\theta + \sigma_\theta \odot \epsilon = v(a_t, t) - (t - b)\frac{d}{dt}\left(a_t - g_\theta + \sigma_\theta \odot \epsilon\right). \tag{30}$$

Solving for g_θ and using the flow dynamics $\frac{da_t}{dt} = v(a_t, t)$, we obtain:

$$\begin{aligned} g_\theta &= a_t + \sigma_\theta \odot \epsilon - v(a_t, t) + (t - b)\left[\frac{da_t}{dt} - \frac{dg_\theta}{dt} + \frac{d\sigma_\theta}{dt} \odot \epsilon\right] \\ &= a_t + \sigma_\theta \odot \epsilon - v(a_t, t) + (t - b)v(a_t, t) - (t - b)\left(\frac{dg_\theta}{dt} - \frac{d\sigma_\theta}{dt} \odot \epsilon\right) \\ &= a_t + (t - b - 1)v(a_t, t) + \sigma_\theta \odot \epsilon - (t - b)\left(\frac{dg_\theta}{dt} - \frac{d\sigma_\theta}{dt} \odot \epsilon\right). \end{aligned} \tag{31}$$

For these time derivatives, we hold the sampled perturbation ϵ fixed and differentiate along the trajectory a_t, with b fixed. The derivative of $\sigma_\theta \odot \epsilon$ is therefore $(d\sigma_\theta/dt) \odot \epsilon$. Applying the chain rule to a differentiable function $f(a_t, b, t)$ gives:

$$\frac{d}{dt}f(a_t, b, t) = \nabla_{a_t} f \cdot \frac{da_t}{dt} + \nabla_b f \cdot \frac{db}{dt} + \nabla_t f \cdot \frac{dt}{dt}. \tag{32}$$

Given the flow dynamics $\frac{da_t}{dt} = v(a_t, t) = e - a$, $\frac{db}{dt} = 0$, and $\frac{dt}{dt} = 1$, this corresponds to the Jacobian-vector product (JVP) operator:

$$\frac{d}{dt}f(a_t, b, t) = \texttt{jvp}\big(f, (a_t, b, t), (v(a_t, t), 0, 1)\big). \tag{33}$$

Applying this operator to both g_θ and σ_θ in Eq. 31, we derive the explicit regression target g_{tgt}:

$$g_{\text{tgt}} \triangleq a_t + (t - b - 1)v(a_t, t) + \sigma_\theta \odot \epsilon - (t - b)\left[\texttt{jvp}(g_\theta) - \texttt{jvp}(\sigma_\theta) \odot \epsilon\right]. \tag{34}$$

Training Objective. Finally, to enforce the derived identity, we minimise the discrepancy between the predicted generator g_θ and the target g_{tgt}. We employ the Huber loss $\mathcal{L}_\delta$ for robustness against outliers and apply the stop-gradient operator $\text{sg}(\cdot)$ to the target to prevent unstable feedback loops during training. The Stochastic MeanFlow objective is formulated as:

$$\mathcal{L}_{\text{SMF}}(\theta) = \mathbb{E}_{t,b,\epsilon}\left[\mathcal{L}_\delta\left(g_\theta(a_t, b, t) - \text{sg}(g_{\text{tgt}})\right)\right], \tag{35}$$

where $\mathcal{L}_\delta$ denotes the Huber loss with threshold $\delta = 1.0$:

$$\mathcal{L}_\delta(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| \le \delta, \\ \delta(|x| - \frac{1}{2}\delta) & \text{otherwise.} \end{cases} \tag{36}$$

The scale-derivative term in the target accounts for the variation of the Gaussian perturbation along the flow trajectory. This preserves the residual MeanFlow relation when the policy is written in the stochastic form of Eq. (27). The stop-gradient operation makes the evaluated target fixed during each regression update.

In the full actor objective, the weighted SMF loss fits the residual mapping to actions favoured by the critic. The direct Q-gradient also passes through the reparameterised action, so it can shape both the mean and the conditional scale. The entropy-floor penalty supplies an additional gradient on the scale when conditional dispersion is low. These roles explain how value-directed generation and explicit stochasticity control are combined in Eq. (19).

C.3 Justification for Truncated Advantage Weighting.

The value-based mirror-descent target in Eq. (5) increases the probability of high-value actions relative to the old policy. To explain the resulting weights, we first consider a fixed state and hold the critic and old policy fixed in the projection argument. Assuming a finite partition function, the target is:

$$\pi^*(a|s) = \frac{1}{Z(s)}\pi_{\text{old}}(a|s)\exp\left(\frac{Q^{\pi_{\text{old}}}(s,a)}{\lambda}\right), \tag{37}$$

where λ is the regularisation coefficient and $Z(s)$ is the partition function normalising the distribution.

We use a forward KL projection, $\mathcal{D}_{\text{KL}}(\pi^*\|\pi_\theta)$, to motivate weighted policy fitting. The target entropy is independent of the parameters being fitted, so minimising this divergence reduces to minimising the expected negative log-likelihood under π^*. Using the old policy as a proposal expresses this expectation through a density ratio:

$$\begin{aligned} \mathcal{L}_{\text{MD}}(\theta) &= \mathbb{E}_{a\sim\pi^*}\left[-\log\pi_\theta(a|s)\right] \\ &= \mathbb{E}_{a\sim\pi_{\text{old}}}\left[\frac{\pi^*(a|s)}{\pi_{\text{old}}(a|s)}\left(-\log\pi_\theta(a|s)\right)\right] \\ &= \mathbb{E}_{a\sim\pi_{\text{old}}}\left[\frac{\frac{1}{Z(s)}\pi_{\text{old}}(a|s)\exp\left(\lambda^{-1}Q(s,a)\right)}{\pi_{\text{old}}(a|s)}\left(-\log\pi_\theta(a|s)\right)\right] \\ &= \mathbb{E}_{a\sim\pi_{\text{old}}}\left[\frac{1}{Z(s)}\exp\left(\frac{Q(s,a)}{\lambda}\right)\left(-\log\pi_\theta(a|s)\right)\right]. \end{aligned} \tag{38}$$

The ratio cancels the old-policy density, leaving an exponential weight determined by the critic. Centring by $V(s)$ expresses the same within-state preference in terms of advantage: an action-independent baseline multiplies all exponential weights at that state by a common factor. With $A^{\pi_{\text{old}}}(s,a) = Q(s,a) - V(s)$,

$$\begin{aligned} \frac{1}{Z(s)}\exp\left(\frac{Q(s,a)}{\lambda}\right) &= \frac{1}{Z(s)}\exp\left(\frac{A^{\pi_{\text{old}}}(s,a) + V(s)}{\lambda}\right) \\ &\propto \exp\left(\frac{A^{\pi_{\text{old}}}(s,a)}{\lambda}\right). \end{aligned} \tag{39}$$

The proportionality describes relative action weights at a fixed state; the normalisation factor can vary across states. For the practical generative-policy update, we use these advantage-based weights

with the tractable $\mathcal{L}_{\mathrm{SMF}}(\theta; s, a)$ in place of marginal likelihood fitting. The resulting MeanFlow regression surrogate retains the target preference for high-value actions:

$$\mathcal{L}(\theta) = \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_{\text{old}}}\left[\underbrace{\exp\left(\frac{A^{\pi_{\text{old}}}(s,a)}{\lambda}\right)}_{w(s,a)} \mathcal{L}_{\mathrm{SMF}}(\theta; s, a)\right]. \tag{40}$$

Exponential weights can concentrate the regression update on a small number of samples when advantages are large or λ is small. Since the advantages are estimated by an online critic, this concentration can also magnify value-estimation errors. A weighting rule with linear growth offers a more moderate response to large positive estimates while retaining a preference for actions above the value baseline.

The first-order expansion around $A(s, a) = 0$ gives a local motivation for linear dependence on advantage:

$$\exp\left(\frac{A(s,a)}{\lambda}\right) \approx 1 + \frac{A(s,a)}{\lambda}. \tag{41}$$

The expansion contains a uniform fitting component and an advantage-dependent component. Following QVPO (Ding et al., 2024), the practical update retains the positive-advantage component. This choice changes the regression weights and concentrates fitting on actions above the value baseline.

Using signed advantages directly would give negative coefficients to a non-negative regression loss. Rectification instead assigns zero weight to actions at or below the baseline and a positive weight proportional to the estimated improvement for the remaining actions:

$$w(s,a) = \max\left(0, Q(s,a) - V(s)\right) \propto \left(A^{\pi_{\text{old}}}(s,a)\right)^{+}. \tag{42}$$

The baseline is estimated by averaging the critic values of N_{adv} actions drawn directly from the pre-update policy, as described in Sec. 4.3. This provides a state-dependent reference using the existing critic, without training a separate value network. Substituting the positive-advantage weights gives the practical regression objective:

$$\mathcal{L}_{\text{Final}}(\theta) = \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_{\text{old}}}\left[\left(A^{\pi_{\text{old}}}(s,a)\right)^{+} \mathcal{L}_{\mathrm{SMF}}(\theta; s, a)\right]. \tag{43}$$

The positive-advantage weighting has the following useful properties:

1. Monotonicity. Weights are zero at or below the value baseline and strictly increasing for positive advantages. Higher-valued actions therefore receive larger regression weights.

2. Linear Weight Growth. For positive advantages, increasing $Q - V$ changes the weight linearly. This avoids the exponential amplification of large positive critic estimates. The Huber loss addresses a different source of sensitivity: its derivative with respect to the regression residual stops growing once the residual enters the linear regime. Positive-advantage weighting and Huber regression therefore moderate sensitivity to value estimates and fitting errors, respectively.

3. State-Dependent Value Centring. Subtracting $V(s)$ makes action selection depend on improvement relative to the current state. Adding the same state-dependent offset to Q and V leaves the weights unchanged. A positive rescaling of both quantities preserves which actions receive non-zero weight and their relative weights, while rescaling the regression term. In the full actor objective, λ controls the strength of this term relative to the direct Q loss and entropy penalty.

The weighting supplies value guidance through regression targets, complementing the direct Q-gradient through generated actions in Eq. (19).

C.4 Discussion: Entropic Mirror Descent with Generative Policies

SAC-style entropy regularisation and mirror descent provide complementary guidance for policy improvement. The value-based MD target reweights the old policy towards actions with higher critic values, while entropy regularisation encourages stochastic action generation. When several separated action regions have high value and support under the old policy, the MD target can retain multiple

modes. A latent generative mapping can represent these regions, and the conditional Gaussian scale gives SMFP a tractable way to regulate dispersion around its generated means.

The practical actor objective combines these roles through three gradient paths. Advantage-weighted MeanFlow regression fits actions favoured by the critic, the direct Q term provides differentiable value guidance, and the entropy-floor penalty acts when the conditional scale falls below its threshold. The scale remains learnable under the value objective after this penalty becomes inactive. This allows SMFP to adapt action quality and stochasticity within the same one-step policy.

The action-distribution visualisation in Fig. 2b illustrates this distinction. In the displayed setting, the Gaussian policy concentrates on a dominant mode, and the conventional flow policy captures multiple modes with increasingly concentrated samples. SMFP retains broader coverage around the target modes. The entropy ablation in Fig. 4(a) provides complementary task-level evidence: removing the regulariser slows learning and lowers final returns. Together, the visualisation and ablation support combining generative expressivity with explicit control of conditional stochasticity.

D Additional Experimental Results

D.1 Exploration in Sparse-Reward AntMaze Tasks

Tasks and environment. We evaluate AntMaze-v1–v4 from Li et al. (2024) in a Gymnasium/MuJoCo port of the released task definitions. The port retains the maze layouts, designated start and goal locations, and episode horizons: 500 steps for v1–v2 and 700 for v3–v4. Each task has sparse goal rewards; v2–v4 have two possible goals. Figure 6 shows the four layouts.

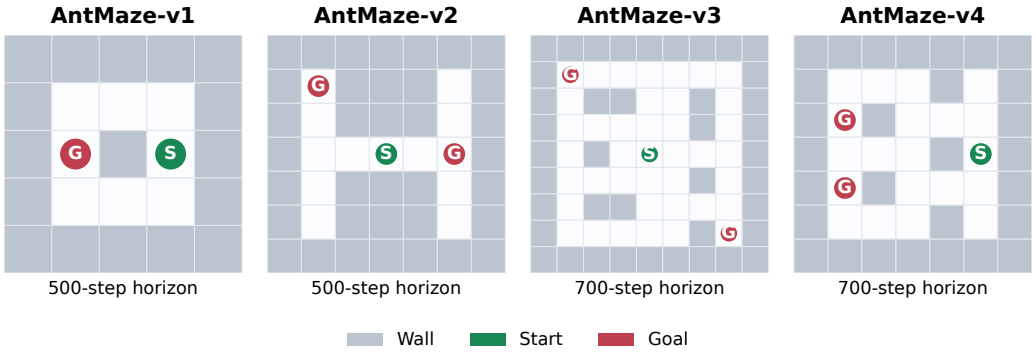


Figure 6: AntMaze task layouts, redrawn from the released definitions (Li et al., 2024). Grey cells are walls; S and G mark designated start and goal cells. The panels show maze geometry rather than sampled trajectories. Episode horizons are 500 steps for v1–v2 and 700 for v3–v4.

Reproduction protocol. We compare SMFP and the baselines in this port with matched environment-interaction budgets. Figure 3 in the main text reports mean success rates over 10 training seeds for every method, with shaded regions showing one standard deviation. All compared AntMaze methods use a NovelD-style intrinsic reward during training (Zhang et al., 2021). SMFP retains its one-step stochastic policy, conditional-entropy regularisation, and off-policy mirror-descent updates, without trajectory clustering, mode-specific critics, multimodal training batches, or mode embeddings.

Goal-reaching results. We assess exploration through the acquisition and retention of goal-reaching behaviour. Figure 3 shows that SMFP learns successful navigation across all four mazes and sustains high success during later training. The learning trajectories support effective exploration and reliable goal attainment under the shared intrinsic-reward protocol. The Push-T visualisation in Fig. 2b separately examines the representation of distinct action modes.

D.2 MuJoCo Learning Curves

Figure 7 complements the aggregate results in Table 1 by showing learning dynamics across the seven MuJoCo benchmarks. The curves use reproduced runs under a common training budget and evaluation protocol.

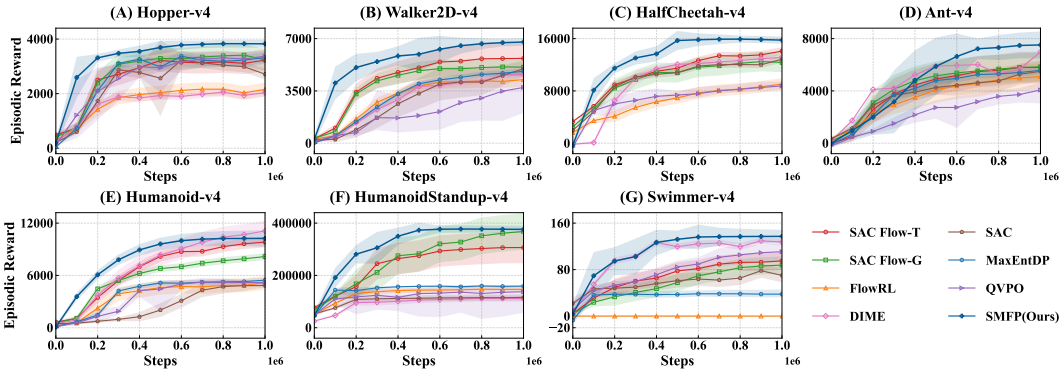


Figure 7: Comparison of evaluation performance across 7 benchmarks. All methods are trained for 1M environment steps. The x-axis denotes environment steps, and the y-axis denotes episodic reward.

D.3 Hyperparameter Sensitivity and Analysis

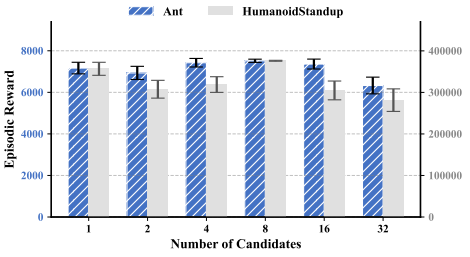


Figure 8(a): Impact of the number of candidates K on performance across two tasks using best-of-K action selection.

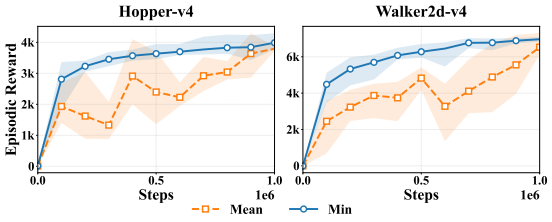


Figure 8(b): Sensitivity analysis of Q-network aggregation strategies.

Effect of candidate size K in value-guided selection. Fig. 8(a) illustrates the impact of candidate pool size K. Among the tested settings, $K = 8$ gives the strongest performance on these two tasks. Larger pools (e.g., $K = 32$) can reduce returns, plausibly because maximising over more candidates amplifies critic estimation errors. These results motivate our default choice of $K = 8$.

Sensitivity to Q-value aggregation. We further analyse the target Q-value aggregation strategy `q_agg`. As shown in Fig. 8(b), using the *min* operator consistently outperforms the *mean* operator. This suggests that mean aggregation is less robust in our maximum-entropy framework, whereas the conservative *min* aggregation provides more stable target estimates and improves final performance. Accordingly, we use *min* as the default Q-aggregation strategy in SMFP.

D.4 Training and Inference Time.

Tab. 2 benchmarks computational efficiency on Ant-v4. SMFP attains the highest return among the listed methods with an inference latency of 0.42 ms. One-step generation and parallel candidate evaluation through JAX's `vmap` keep best-of-K selection efficient, yielding more than an order-of-magnitude lower inference latency than the listed iterative diffusion baselines. Training-time gains are smaller: SMFP also samples from the old policy and optimises the MeanFlow objective. The results therefore support a strong return–latency trade-off, while training remains more expensive than for the simple Gaussian and deterministic baselines.

Table 2: Training and inference efficiency on Ant-v4. SMFP achieves the highest return among the listed methods with low inference latency.

Method	SMFP	QVPO	DIME	DIPO	TD3	SAC	PPO	SPO	QSM	MaxEntDP
Training Time (h)	2.7	5.2	3.5	9.6	0.4	2.2	0.3	0.3	0.8	2.3
Inference Time (ms)	0.42	5.77	4.68	5.20	0.27	0.13	0.22	0.32	5.15	5.62
Inference Steps	1	20	16	20	1	1	1	1	20	20
Performance	7518.3	4040.1	7103.6	5665.9	4583.8	5530.6	2781.9	2100.2	4206.4	5717.9

E Implementation Details.

E.1 Experimental Details

Environments. We evaluate seven OpenAI Gym MuJoCo tasks (Brockman et al., 2016; Todorov et al., 2012). In our runs, each method receives 1M (10^6) environment steps under a common evaluation protocol. SMFP results are means and standard deviations over 20 seeds; Tab. 1 supplements reproduced baseline scores with comparable published results when stronger (Appendix E.1).

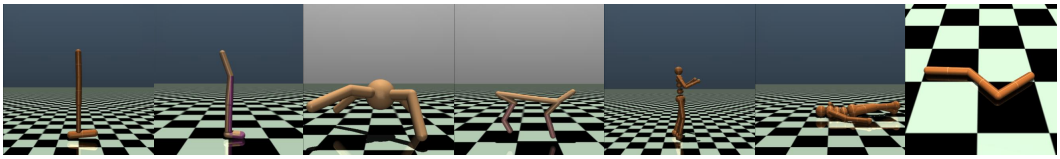


Figure 9: MuJoCo environments used in our experiments: Hopper-v4, Walker2d-v4, Ant-v4, HalfCheetah-v4, Humanoid-v4, HumanoidStandup-v4 and Swimmer-v4.

Table 3: Comparison of policy representations and optimisation components. **Gaussian**: Uses Gaussian or deterministic components; SMFP uses latent-conditioned Gaussians. **Generative**: Uses diffusion or flow-based policies. **SAC-Style**: Uses entropy-regularised updates. **MD / guided fitting**: Includes mirror descent and related value-guided policy fitting. FlowRL uses value-weighted flow matching under Wasserstein-2 regularisation.

Family	**Algorithm**	**Representation Level**		**Optimisation Level**	
		Gaussian Policy	**Generative Policy**	**SAC (Ent. Reg.)**	**MD / guided fitting**
Gaussian	PPO (Schulman et al., 2017)	✓	✗	✗	✗
	SPO (Xie et al., 2025)	✓	✗	✗	✗
	TD3 (Fujimoto et al., 2018)	✓*	✗	✗	✗
	SAC (Haarnoja et al., 2018)	✓	✗	✓	✗
Diffusion	DIPO (Yang et al., 2023)	✗	✓	✗	✓
	QSM (Psenka et al., 2024)	✗	✓	✗	✓
	QVPO (Ding et al., 2024)	✗	✓	✗	✓
	DIME (Celik et al., 2025)	✗	✓	✓	✗
	MaxEntDP (Dong et al., 2025)	✗	✓	✓	✗
	DPMD (Ma et al., 2025)	✗	✓	✗	✓
Flow	SAC Flow (Zhang et al., 2026)	✗	✓	✓	✗
	FlowRL (Lyu et al., 2025)	✗	✓	✗	✓
	FPMD (Chen et al., 2025)	✗	✓	✗	✓
Ours	**SMFP**	✓	✓	✓	✓

* TD3 uses a deterministic actor with added Gaussian noise for exploration and target policy smoothing.

Baselines. We compare our approach against three categories of online RL algorithms: (1) Classical Model-Free RL: Methods employing Gaussian or deterministic policies with efficient single-step inference (1-NFE), including PPO (Schulman et al., 2017) (`https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`), SPO (Xie et al., 2025) (`https://github.com/MyRepositories-hub/Simple-Policy-Optimization`), TD3 (Fujimoto et al., 2018) (`https://github.com/sfujim/TD3`), and SAC (Haarnoja et al., 2018) (`https://github.com/toshikwa/soft-actor-critic.pytorch`). (2) Diffusion-Based RL: Generative policy algorithms such as DIPO (Yang et al., 2023)

(https://github.com/BellmanTimeHut/DIPO), QSM (Psenka et al., 2024) (https://github.com/escontra/score_matching_rl), QVPO (Ding et al., 2024) (https://github.com/wadx2019/qvpo), DIME (Celik et al., 2025) (https://github.com/ALRhub/DIME), MaxEntDP (Dong et al., 2025) (https://github.com/diffusionyes/MaxEntDP), and DPMD (Ma et al., 2025) (https://github.com/mahaitongdae/diffusion_policy_online_rl). These methods typically involve iterative sampling, though some are optimised for efficiency. (3) Flow-Based RL: Emerging flow-matching approaches, including SAC Flow (Zhang et al., 2026) (https://github.com/Elessar123/SAC-FLOW), FlowRL (Lyu et al., 2025) (https://github.com/bytedance/FlowRL), and FPMD (Chen et al., 2025). Tab. 3 compares these baselines. We use their published hyperparameters as the starting point for reproduction and tuning.

We tuned reproduced baselines, including on environments absent from their original evaluations. For each baseline and environment in Tab. 1, we report the stronger of our reproduction and a comparable published or benchmark result, where available. SMFP entries use our 20-seed evaluation. The learning curves in Fig. 7 use reproduced runs only, with a common training budget and evaluation protocol; table values and curves can therefore differ.

Critic network architecture. Following MeanFlowQL (Wang et al., 2026), we use multilayer perceptrons (MLPs) with four hidden layers of 512 units each for the critics. To improve training stability, we also apply layer normalisation to the value networks.

Actor network architecture. The actor is a shallow Transformer adapted from DiT (Peebles & Xie, 2023), with 3 layers, 2 attention heads, and hidden size 256. Feature embeddings encode the observation and action inputs, while time and positional embeddings provide the conditioning used in the MeanFlow objective and its Jacobian-vector products. We use residual mixing with scale 0.1, gradient clipping at norm 1.0, and a zero-initialised final projection.

F HYPERPARAMETERS

Table 6 lists the shared architecture and optimisation settings for the MuJoCo experiments. Table 5 gives the task-specific mirror-descent coefficient λ.

Table 4 provides default values and candidate ranges for new environments. We recommend starting from these defaults, then searching the listed candidates where needed. For sequential tuning, fix the entropy coefficient α before varying the mirror-descent coefficient λ.

Table 4: Recommended Hyperparameter Search Space for New Environments

Parameter	**Default**	**Candidate Values**	**Description**
q_agg	min	$\{\text{min}, \text{mean}\}$	Target Q aggregation method
κ_σ	-3	$\{-1, -3, -5, -7\}$	Mean log-scale threshold
α	0.2	$\{0.01, 0.05, 0.1, 0.2, 0.4, 1\}$	Entropy temperature in SAC
λ	0.3	$\{0.03, 0.1, 0.3, 1, 3, 10\}$	Mirror descent regularisation coefficient
num_candidates (K_b)	8	$\{1, 4, 8, 16, 32\}$	Number of candidates

Table 5: **Task-Specific Hyperparameter Settings.** We report the mirror descent regularisation coefficient (λ) used for each environment.

Environment	HOPPER-V4	WALKER2D-V4	HALFCHEETAH-V4	ANT-V4	HUMANOID-V4	HUMANOIDSTANDUP-V4	SWIMMER-V4
Lambda (λ)	3	3	0.3	0.3	0.3	0.3	3

Table 6: Hyperparameters in Experiments

Parameter	Value	Description
General Settings		
batch_size	256	Number of samples per batch
q_agg	min [+]	Target Q aggregation method
noise_type	gaussian	Noise distribution type
optimiser	Adam (Kingma & Ba, 2014)	Network optimiser
gradient steps	1000000	Total gradient updates
discount γ	0.99	Same as QVPO (Ding et al., 2024)
lr	$3e{-4}$	Learning rate
lr_schedule	cosine_with_warmup	Learning rate schedule
lr_min_ratio	0.1	Minimum ratio in cosine scheduler
Critic Network		
value_hidden_dims	(512, 512, 512, 512)	Hidden layer sizes
layer_norm	True	Apply layer normalisation
gradient clipping	1.0	Maximum gradient norm
network initialisation	kaiming_init	Network initialisation techniques
Actor Network		
actor_hidden_dims	256	Policy hidden size
actor_depth	3	Number of transformer layers
actor_num_heads	2	Number of attention heads
actor_layer_norm	False	Use layer norm in actor
discount	0.99	Discount factor γ
tau	0.005	Target network update rate
tanh_squash	False	Use tanh at actor output
use_output_layernorm	False	Norm at final actor output
gradient clipping	1.0	Maximum gradient norm
network initialisation	zero_init	Network initialisation techniques
κ_σ	-3 [+]	Mean log-scale threshold
time_steps	100	Time steps of MeanFlow (same as (Wang et al., 2026))
Best-of-K Selection		
num_candidates (K)	8[+]	Number of candidates
action_mode	best	Action selection strategy
N_d	64	Number of samples from old policy(Ding et al., 2024)
Transform	$qadv$	Q-weight transformation function (Ding et al., 2024)
SMFP		
α	0.2 [+]	Entropy temperature hyperparameter (SAC)
λ	Task-dependent	See Tab. 5

[+] Task-dependent. For new environments, we recommend performing a grid search over the candidate values (refer to Tab. 4) to achieve optimal performance.