POLICY OPTIMIZATION VIA DRIFTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) of diffusion and flow-based policies has become a popular paradigm in robot learning, where actions are continuous and multimodal. Sampling from these generative policies, however, costs many sequential network evaluations and incurs latency, making them a poor fit for dynamic tasks with tight feedback loops. Drifting models offer an appealing alternative. Empirically, they are as expressive as diffusion and flow models, but they can produce a sample with a single network evaluation. In this work, we propose reward drifting, an on-policy policy gradient optimization method. Starting from the theory of Wasserstein gradient flows, we derive a single fixed-point objective and justify that optimizing it is equivalent to increasing the expected discounted reward. Our method does not require access to action-likelihoods or surrogates. Empirically, we evaluate our algorithm across a range of manipulation tasks, including Push-T, dexterous in-hand reorientation, and long horizon bimanual manipulation. Our method offers substantially faster inference than diffusion and flow-matching baselines fine-tuned with existing RL methods.

1 INTRODUCTION

Diffusion and flow-based generative policies have seen great success in robot learning (Chi et al., 2025; Black et al., 2024) and generalize across a wide range of tasks (Barreiros et al., 2026; Intelligence et al., 2026). However, these policies require integrating the learned flow field at inference time, and can be time-consuming, limiting their application to quasi-static robotic tasks. Reliance on heuristics such as action chunking (Zhao et al., 2023), with frequent open loop rollouts, further limit the reactivity of the robot policy. On the other hand, optimizing these policies with online reinforcement learning is similarly challenging. Standard policy gradient methods (Schulman et al., 2017b) require the likelihood of the sampled action under the current policy, which is not easily available for these generative policies: the likelihood is defined implicitly through a multi-step sampling process, and estimating it is costly and imprecise (Ai et al., 2026).

Various workarounds address these limitations. To reduce latency, one can first train a full multi-step model and then distill it into a few-step or one-step student (Prasad et al., 2024; Frans et al., 2025). To enable policy optimization, the denoising chain can be reinterpreted as a Markov decision process whose individual steps have tractable Gaussian likelihoods, as in DPPO (Ren et al., 2025). These workarounds add a training stage and hyperparameters, and the resulting pipeline is considerably more complex. A policy class that produces actions in one step and admits a direct training signal for reinforcement learning would remove the need for distillation and for surrogate likelihoods altogether.

To address the challenges above, we look at a new class of generative models, called *drifting* models (Deng et al., 2026). A drifting model samples actions in a single forward pass; hence, it is fast at inference time. However, it does not provide action likelihoods, so the standard policy gradient algorithms are not applicable. Post-training a drifting policy with reinforcement learning thus still requires rethinking policy optimization.

We develop such a method by studying drifting through the lens of transport. We show that the drifting objective approximates a Wasserstein gradient flow on the distribution of generated actions. This view connects drifting to Wasserstein policy gradient flows, a connection that motivates an on-policy objective for drifting. Our loss is computed from rollouts sampled by the current policy and the rewards they receive in the environment. It does not require the likelihood, a surrogate Markov

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.

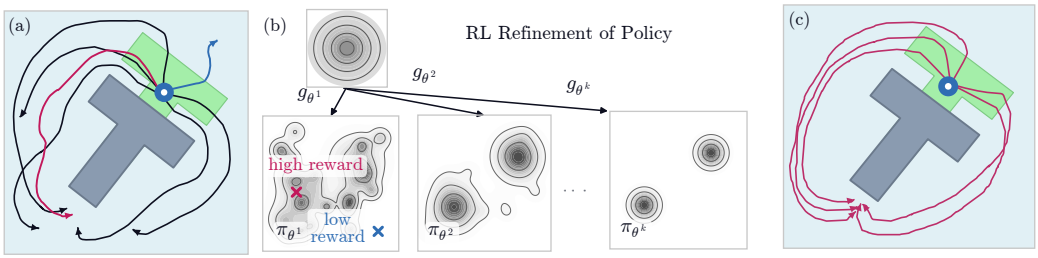

Figure 1: Illustration of Reward Drifting. $(\mathbf{a})$ Rollouts of the initial policy, obtained by imitation learning. $(\mathbf{b})$ a one-step generator g_θ maps Gaussian noise $z \sim \mathcal{N}(0, I)$ to actions; during RL post-training, sampled actions are drifted toward high-reward actions and away from low-reward ones, so the action distribution sharpens from π_{θ^1} to π_{θ^k} without ever evaluating a likelihood or a denoising chain. $(\mathbf{c})$ Rollouts of the post-trained policy sharpening on the high reward region.

decision process over denoising steps, or an additional training stage. This is illustrated in Figure 1 where an initialized policy obtained through imitation learning steadily improves and concentrates its actions around higher reward regions.

Our contributions are as follows. (i) We establish a rigorous connection between drifting models and Wasserstein policy gradient flows (WPGF) and use it to derive *reward drifting*, an on-policy objective that improves a one-step drifting policy directly from environment interaction, with no action likelihood, surrogate Markov decision process, or distillation stage. (ii) We evaluate reward drifting on three robotic manipulation tasks: Push-T, dexterous in-hand reorientation, and long-horizon bimanual transport. (iii) On all three, it matches or exceeds the success rate of recent generative post-training baselines while reducing inference latency by at least $3\times$.

2 Related Work

Generative models. Generative models, including normalizing flows (Rezende & Mohamed, 2015; Dinh et al., 2017; Papamakarios et al., 2021), energy-based models (LeCun et al., 2006; Du & Mordatch, 2019), variational autoencoders (Kingma & Welling, 2014), and generative adversarial networks (Goodfellow et al., 2014), have a long history in representing complex distributions. Modern generative models, however, increasing rely on diffusion (Ho et al., 2020; Song et al., 2020) and flow (Lipman et al., 2022; Liu et al., 2022) processes, which iteratively transport noise to a complex distribution. Although these models are highly expressive, at inference time they iteratively simulate an expensive flow map. To reduce this computational overhead, early efforts focused on distilling trained multi-step models into student networks with fewer steps (Salimans & Ho, 2022; Song et al., 2023; Yin et al., 2024). Subsequent work trained few-step models from scratch by learning variable-length jumps (Frans et al., 2025) or the average velocity over an interval (Geng et al., 2025). More recently, drifting models (Deng et al., 2026) have adapted the transport machinery of diffusion and flow models while overhauling their training process. They effectively amortize the cost of iterative refinement from inference to training time. Consequently, they achieve the expressivity of their predecessors but require only a single network evaluation at inference. Emerging theoretical research frames these drifting models as simulating a gradient flow that minimizes the KL divergence between the data and modeled distributions (Turan et al., 2026; Lai et al., 2026; Gretton et al., 2026). Drawing on the empirical success of (Deng et al., 2026), we leverage the mathematical framework of gradient flows not for generative modeling, but for policy optimization in RL.

Generative policies for RL. Modern generative frameworks do not provide access to actions' likelihoods. This property is problematic for policy gradient optimization and is a main reason why policy optimization needs to be rewritten for these policies. Some offline methods sidestep the likelihood by combining the generative loss with a critic: Wang et al. (2022) add a Q-maximization term to the diffusion objective, Park et al. (2025) keep the flow model as a behavior prior and train a separate one-step actor against Q, Li & Levine (2026) use adjoint matching to push the critic's action gradient through the sampler step by step, and Psenka et al. (2023) match the policy's score to the action gradient of Q directly. Among online methods, Ren et al. (2025) treat the denoising chain as its own

Markov decision process and run proximal policy optimization (PPO) (Schulman et al., 2017b) on it, and Zhang et al. (2025) inject noise into a flow policy to recover a tractable likelihood; both pay for the likelihood with a rollout of the entire denoising chain per action. Recently, certain variations of RL have been explored with drifting policies: Houssaini et al. (2026) regularize an offline actor with a drift toward dataset actions; Koo et al. (2026) frame the update as a Wasserstein gradient flow toward a soft target policy and realize it with critic-selected actions; Gao et al. (2026) pretrain a one-step drifting policy and fine-tune it with PPO through a stochastic variant that has an exact likelihood, keeping a deterministic single-pass path for deployment; and Zhang (2026) drift toward positive-advantage only. We share the single-pass policy with these papers; the difference is that our update is the reward's Wasserstein gradient, computed from samples, rather than a surrogate built from a critic's ranking or a likelihood recovered by adding Gaussian noise.

Policy optimization as transport. A separate line of work treats policy optimization variationally, gradually evolving the policy distribution to maximize reward. The underlying calculus relies on Wasserstein gradient flows over the space of probability measures (Jordan et al., 1998; Ambrosio et al., 2005). In the context of RL, this optimization is referred to as Wasserstein policy gradient flow (WPGF), which deforms a policy by transporting it along the Wasserstein gradient field of the value functional. Extensive research investigates the theoretical properties and convergence of WPGFs (Sutton et al., 1999; Zhang et al., 2018; Dus, 2026; Šiška & Zhang, 2026; Zhu et al., 2026b), particularly for continuous control (Pfau et al., 2025; Zhu et al., 2026a; Kerimkulov et al., 2025; 2026). Building upon this rich literature, our work establishes explicit connections to design a practical, one-step generative policy that scales to diverse robot manipulation tasks.

3 Background

RL Setup. We describe the RL problem as a Markov decision process (MDP) (Sutton & Barto, 2018), a tuple $(\mathcal{S}, \mathcal{A}, P, r, \mu, \gamma)$ where $\mathcal{S}$ is the set of states s, $\mathcal{A}$ the set of actions a, $P(s' \mid s, a)$ the transition probability of reaching state s' after taking action a in state s, $r(s, a)$ the reward for that action, μ the distribution of the initial state, and $\gamma \in (0, 1)$ a discount factor that weights near-term rewards over distant ones. A policy $\pi(a \mid s)$ specifies how the agent acts in the environment. In particular, in state s an action is drawn $a \sim \pi(\cdot \mid s)$, and as a result, the agent receives the reward $r(s, a)$ and moves to $s' \sim P(\cdot \mid s, a)$. In robotic manipulation, for example, the state may be an image or another observation of the robot and its surroundings, while the action may be a single control or a short sequence of control outputs.

To quantify the performance of a policy, we introduce its value function. Here, we formulate everything using maximum entropy RL, where, for a fixed policy π, the soft Bellman expectation equations define its soft state value V^π and action value Q^π:

$$\begin{aligned} V^\pi(s) &= \int \pi(a \mid s)\Big(Q^\pi(s, a) - \tau \log \pi(a \mid s)\Big)\, da, \\ Q^\pi(s, a) &= r(s, a) + \gamma \int P(s' \mid s, a)\, V^\pi(s')\, ds'. \end{aligned} \tag{1}$$

Intuitively, $V^\pi(s)$ is the total discounted reward the agent can expect if it starts from s and follows π, and $Q^\pi(s, a)$ is the expected reward if it takes action a and then follows π afterwards. The entropy term with weight τ keeps the policy from collapsing onto a single action and rewards it for staying stochastic (Haarnoja et al., 2018; Schulman et al., 2017a). The RL objective is then maximizing the expected value function of states:

$$J[\pi] \coloneqq \mathbb{E}_{s_0 \sim \mu}\big[V^\pi(s_0)\big]. \tag{2}$$

Therefore, policy optimization can be understood variationally as finding a π that maximizes J.

Drifting models and transport. Unlike diffusion and flow-based models, drifting models produce a sample in a single network evaluation. They have recently been adopted to parameterize policies in RL (Gao et al., 2026; Koo et al., 2026; Houssaini et al., 2026). A drifting policy is the pushforward of Gaussian noise through a state-conditioned map, $\pi_\theta(\cdot \mid s) \coloneqq \big(g_\theta(\cdot, s)\big)_{\#}\mathcal{N}(0, I)$. In other words, to sample an action in state s, one draws $z \sim \mathcal{N}(0, I)$ and evaluates the network $a = g_\theta(z, s)$.

We can understand drifting models through the lens of Wasserstein gradients (Gretton et al., 2026). Formally, for any functional $F[\pi]$, the Wasserstein gradient vector field is written

$$u = \nabla \frac{\delta F[\pi]}{\delta \pi}, \tag{3}$$

where $\delta F/\delta\pi$ is the first variation of F and ∇ is the gradient in sample space; intuitively, $u(a)$ tells us how to nudge a sample sitting at a so that F increases as quickly as possible. Turan et al. (2026); Lai et al. (2026); Gretton et al. (2026) show in various ways that drifting models approximate different discretizations of the Wasserstein gradient flows. In the context of generative models, the functional $F[\pi_\theta]$ is the negative KL divergence $-\mathrm{KL}(\pi_\theta \,\|\, \pi_{\text{data}})$ and the Wasserstein gradient field is the score difference of the two distributions:

$$-\nabla \frac{\delta \mathrm{KL}(\pi_\theta \| \pi_{\text{data}})}{\delta \pi_\theta} = \underbrace{\nabla \log \pi_{\text{data}}}_{\text{attraction force}} - \underbrace{\nabla \log \pi_\theta}_{\text{entropy repulsion force}} \approx \hat{u}_\theta. \tag{4}$$

Therefore, maximizing the functional amounts to attracting samples toward the data and repelling them from the model's samples.

In the original drifting paper Deng et al. (2026), at each step, θ^k gets updated to θ^{k+1} using the following rule:

$$\theta^{k+1} = \arg\min_\theta \mathcal{L}(\theta, \theta^k), \tag{5}$$

$$\mathcal{L}(\theta, \theta^k) = \mathbb{E}_z \| g_\theta(z) - \Big(g_{\theta^k}(z) + \hat{u}_{\theta^k}(g_{\theta^k}(z)) \Big) \|^2 + \lambda \mathbb{E}_z \| g_\theta(z) - g_{\theta^k}(z) \|^2, \tag{6}$$

with the goal of finding a fixed point $\theta^\star$, such that $\theta^\star = \arg\min_\theta \mathcal{L}(\theta, \theta^\star)$. Here, λ is a proximity penalty that ensures the g_θ-generated samples do not deviate too much from g_{θ^k}. Because the field $\hat{u}$ in equation 6 is evaluated at the current step θ^k, we refer to it as *explicit drifting*; this is because it is simulating an explicit Euler solver of the Wasserstein gradient flow.[1]

Wasserstein gradient flows are, however, more commonly discretized by the implicit JKO scheme (Jordan et al., 1998), $\pi^{k+1} = \arg\max_\pi F[\pi] - \frac{1}{2\Delta t} W_2^2(\pi, \pi^k)$ with W_2 the Wasserstein-2 distance. If we assume T is the optimal transport map from π^k to π^{k+1}, JKO can also be understood as an implicit Euler updated along this field: $T(a) - a = \Delta t\, u(T(a))$ with u the Wasserstein gradient of F at the *new* distribution π^{k+1}, evaluated at the *new* point $T(a)$ (Ambrosio et al., 2005, Lemma 10.1.2). Now focus on the proximal penalty $\mathbb{E}_z \| g_\theta(z) - g_{\theta^k}(z) \|^2$, it is already minimizing a transport cost for the map $\hat{T}_\theta = g_\theta \circ g_{\theta^k}^{-1}$, because both are generated from the same noise z. Hence, by minimizing the proximal penalty, we are letting $\hat{T} \approx T$. Thus, an implicit Euler step with $\hat{T}_\theta \approx T$ is roughly equivalent to JKO. We call this variant *implicit drifting*,

$$\theta^{k+1} = \arg\min_\theta \tilde{\mathcal{L}}(\theta, \theta^k), \tag{7}$$

$$\tilde{\mathcal{L}}(\theta, \theta^k) = \mathbb{E}_z \| g_\theta(z) - \Big(g_{\theta^k}(z) + \hat{u}_\theta(g_\theta(z)) \Big) \|^2 + \lambda \mathbb{E}_z \| g_\theta(z) - g_{\theta^k}(z) \|^2. \tag{8}$$

Appendix A makes this precise and relates them to parametric JKO (Zhang et al., 2018; Han et al., 2026). In the context of RL, we first need to derive the gradient field for maximizing the expected value $J[\pi_\theta]$ rather than for minimizing $\mathrm{KL}(\pi_\theta \| \pi_{\text{data}})$, and then show how an objective of the form above can be built on it for policy optimization.

4 Wasserstein Gradients for RL

Theorem 1 (Wasserstein policy gradient). *Let $\rho^\pi(s) = \mathbb{E}_{s_0 \sim \mu} \sum_{t \geq 0} \gamma^t \Pr(s_t = s \mid s_0, \pi)$ be the discounted occupancy of a state. Then the Wasserstein gradient field of J from equation 2 is*

$$\nabla_a \frac{\delta J}{\delta \pi(a \mid s)} = \rho^\pi(s) \Big(\underbrace{\nabla_a Q^\pi(s,a)}_{\textit{reward force}} - \underbrace{\tau \nabla_a \log \pi_\theta(a \mid s)}_{\textit{entropy repulsion force}} \Big). \tag{9}$$

[1] Deng et al. (2026) do not explicitly write their objective in this form, but one can show that this is equivalent to their loss involving stop-gradient (see Appendix A).

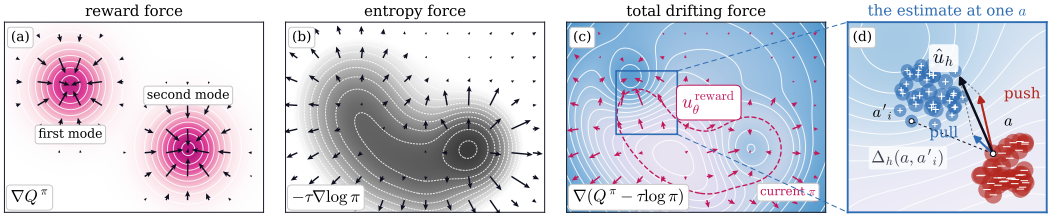

Figure 2: Estimating $u_\theta^{\text{reward}}(\cdot, s)$. (**a**) shows a bimodal Q^π function alongside the gradient field increasing returns; (**b**) shows a misaligned policy alongside its entropy repulsion force; (**c**) shows the sum of these two forces and indicates directions of improvement for the policy; (**d**) showing the mean-shift-based estimator of the total drifting force by taking a linear combination of displacements. Some displacements push ($w_i(a) < 0$), and others pull ($w_i(a) > 0$).

Appendix B provides a proof based on standard variational calculus and the Bellman equations. The gradient field contains two interacting forces: an attraction toward higher Q values and a regularizing entropy force. Compared to equation 4, the entropy repulsion force is still present, but instead of attracting samples toward a given data distribution, the field pushes them toward higher returns Q^π. Theorem 1 has already been explored in other contexts by Zhang et al. (2018), Pfau et al. (2025), and Zhu et al. (2026b); in fact, Zhu et al. (2026b, Theorem 3.1) have recently proved global exponential convergence of this exact flow. Our goal here is simply to estimate this field and plug it into the fixed point updates of the previous section. Writing

$$u_\theta^{\text{reward}}(a, s) := \nabla_a Q^{\pi_\theta}(s, a) - \tau \nabla_a \log \pi_\theta(a \mid s), \tag{10}$$

and abbreviating $g_\theta := g_\theta(z, s_t)$, $g_k := g_{\theta^k}(z, s_t)$, $\hat{u}_k := \hat{u}_{\theta^k}^{\text{reward}}(g_k, s_t)$ and $\hat{u}_\theta := \hat{u}_\theta^{\text{reward}}(g_\theta, s_t)$ for the estimated field at the old and the new samples, the explicit and implicit versions of the fixed point loss can be written as

$$\mathcal{L}(\theta, \theta^k) = \mathbb{E}_{z,s_t}\left[\gamma^t\left(\left\|g_\theta - g_k - \hat{u}_k\right\|^2 + \lambda \left\|g_\theta - g_k\right\|^2\right)\right], \tag{11}$$

$$\tilde{\mathcal{L}}(\theta, \theta^k) = \mathbb{E}_{z,s_t}\left[\gamma^t\left(\left\|g_\theta - g_k - \hat{u}_\theta\right\|^2 + \lambda \left\|g_\theta - g_k\right\|^2\right)\right], \tag{12}$$

with $z \sim \mathcal{N}(0, I)$ and s_t the states of a rollout of π_{θ^k}. Note that the occupancy factor ρ^π of equation 9 does not appear explicitly because letting the current policy run freely in the environment and weighting the states met by γ^t already gives a weight ρ^{π_θ} to each state in expectation. Next, we finish the last missing piece of reward drifting, which is estimating the field u_θ^{reward}.

5 Mean-shift Estimation and Algorithmic Details

The previous section left us with a field u_θ^{reward} made of two forces: the reward force $\nabla_a Q(s, a)$, and the entropy force $\nabla \log \pi(a \mid s)$ — i.e., the score function. We have illustrated these force fields in Figure 2. Here, we show a bimodal Q function along with its reward force on panel (a), the current policy distribution alongside its entropy force on panel (b), and the total field u_θ^{reward} panel (c). We will use kernel-based mean-shift estimators (Fukunaga & Hostetler, 1975) of these forces, similar to Deng et al. (2026), that rely on samples $a'_i \sim \pi(\cdot \mid s)$. The kernel here is a generalized Gaussian $\kappa_h(x, y) = \exp\left(-\|x - y\|^p / p h^p\right)$ with h denoting the bandwidth.

We first focus on the entropy force, which does not require new machinery. Similar to Turan et al. (2026); Gretton et al. (2026); Lai et al. (2026), we approximate the score function by smoothing π with the kernel κ_h for some $h > 0$. Here and below, $\mathbb{E}[\cdot]$ denotes expectation over the action a' sampled from the current policy $\pi(\cdot \mid s)$:

$$\nabla_a \log \pi(a \mid s) = \lim_{h \to 0} \nabla_a \log \mathbb{E}[\kappa_h(a, a')] = \lim_{h \to 0} \frac{1}{h^p} \frac{\mathbb{E}[\Delta_h(a', a)\, \kappa_h(a, a')]}{\mathbb{E}[\kappa_h(a, a')]}, \tag{13}$$

where we denote the action displacement as $\Delta_h(a', a) := (a' - a) \cdot \|a' - a\|^{p-2} = \nabla_a \log \kappa_h(a, a')$. One of these displacements is visualized in Figure 2 (d) as dotted gray line for demonstration. The

Algorithm 1 Explicit Reward Drifting

input g_θ, h, τ, λ, M, K
for $k = 0, \dots, K-1$ **do**
  $\theta^k \leftarrow \theta$
  roll out π_{θ^k} from $s_0 \sim \mu$; keep states s_t
  for each s_t **do**
    $z_i \sim \mathcal{N}(0, I),\ a'_i \leftarrow g_{\theta^k}(z_i, s_t),\ i \le G$
    $q_i \leftarrow$ return of π_{θ^k} from (s_t, a'_i)
    $\hat{u}_i \leftarrow \sum_l w_l(a'_i)\,\Delta_h(a'_l, a'_i)$
  end for
  for $m = 1, \dots, M$ **do**
    $\tilde{a}_i \leftarrow g_\theta(z_i, s_t)$ for all s_t, i
    $\ell_i \leftarrow \|\tilde{a}_i - a'_i - \hat{u}_i\|^2 + \lambda\|\tilde{a}_i - a'_i\|^2$
    gradient step on $\sum_t \gamma^t \frac{1}{G} \sum_i \ell_i$
  end for
end for
return g_θ

Algorithm 2 Implicit Reward Drifting

input g_θ, h, τ, λ, M, K
for $k = 0, \dots, K-1$ **do**
  $\theta^k \leftarrow \theta$
  roll out π_{θ^k} from $s_0 \sim \mu$; keep states s_t
  $z_i \sim \mathcal{N}(0, I),\ a'_i \leftarrow g_{\theta^k}(z_i, s_t),\ i \le G$
  for $m = 1, \dots, M$ **do**
    for each s_t **do**
      $\tilde{a}_i \leftarrow g_\theta(z_i, s_t)$ with the shared noise z_i
      $\tilde{q}_i \leftarrow$ return of π_θ from $(s_t, \tilde{a}_i)$
      $\hat{u}_i \leftarrow \sum_l w_l(\tilde{a}_i)\,\Delta_h(\tilde{a}_l, \tilde{a}_i)$
    end for
    $\ell_i \leftarrow \|\tilde{a}_i - a'_i - \hat{u}_i\|^2 + \lambda\|\tilde{a}_i - a'_i\|^2$
    gradient step on $\sum_t \gamma^t \frac{1}{G} \sum_i \ell_i$
  end for
end for
return g_θ

bandwidth parameter h controls the smoothness of the vector field. In theory, one should set $h \to 0$ but setting h to a larger value also works in practice (Deng et al., 2026; Turan et al., 2026).

We now derive the reward force. The trick here is to weight the actions by $e^{Q(s,a)}$: the actions that result in higher return should pull with more force. Mathematically:

$$\nabla_a Q(s,a) = \nabla_a \log \frac{e^{Q(s,a)}\,\pi(a \mid s)}{\pi(a \mid s)} = \lim_{h\to 0} \nabla_a \log \frac{\mathbb{E}\Big[\kappa_h(a,a')\,e^{Q(s,a')}\Big]}{\mathbb{E}[\kappa_h(a,a')]} \tag{14}$$

$$= \lim_{h\to 0} \frac{1}{h^p} \frac{\mathbb{E}\Big[\Delta_h(a',a)\,\kappa_h(a,a')\,e^{Q(s,a')}\Big]}{\mathbb{E}\Big[\kappa_h(a,a')\,e^{Q(s,a')}\Big]} - \nabla_a \log \pi(a \mid s). \tag{15}$$

Combining the two kernel expectations in equation 13 and equation 15 allows us to write u_θ^{reward} as:

$$h^p \cdot u_\theta^{\text{reward}}(a,s) = \frac{\mathbb{E}\Big[\Delta_h(a',a)\ \kappa_h(a,a')\,e^{Q(s,a')}\Big]}{\mathbb{E}\Big[\kappa_h(a,a')\,e^{Q(s,a')}\Big]} - (1+\tau)\frac{\mathbb{E}[\Delta_h(a',a)\,\kappa_h(a,a')]}{\mathbb{E}[\kappa_h(a,a')]}. \tag{16}$$

Recall that τ is the entropy weight in the formula for the reward force equation 10 and appears here because $u_\theta^{\text{reward}} = \nabla Q - \tau \nabla \log \pi$.

Finally, we now have the full recipe to estimate the reward field by taking a Monte-carlo weighted sum of displacements. Sample $a'_1, \dots, a'_G \sim \pi_\theta(\cdot \mid s)$ alongside their returns $q_i = Q(s, a'_i)$ using Monte-carlo rollouts, then estimate

$$\hat{u}_h(a,s) \approx \sum_{i=1}^{G} w_i(a) \cdot \Delta_h(a'_i, a), \text{where,} \qquad \begin{cases} \alpha_i(a) & = \text{softmax}_i\big(\log \kappa_h(a,a'_i) + q_i\big), \\ \beta_i(a) & = \text{softmax}_i\big(\log \kappa_h(a,a'_i)\big), \\ w_i(a) & = \alpha_i(a) - (1+\tau)\beta_i(a) \end{cases} \tag{17}$$

We have illustrated this in Figure 2 (d). Some actions a'_i push away when $(1+\tau)\beta_i(a)$ is large, and some pull when their $\alpha_i(a)$ are large. Their summation provides the total estimated force $\hat{u}_h$. One can show that as G becomes larger and h becomes smaller $\hat{u}_h$ converges to u_θ^{reward}.[2]

The final step is substituting $\hat{u}_h$ into equation 11 for the explicit drifting and equation 12 for the implicit drifting. Overall, the method culminates to two algorithms 1 and 2 which are on-policy optimization algorithms. That said, even though the implicit method in Algorithm 2 is a more standard way to optimize the WPGF, it requires substantially more interactions to obtain $\tilde{q}_i$ and with the same number of learning steps is always comparatively worse than explicit drifting (see Figure 11 of Appendix C). Therefore, from now on, we will only use the explicit drifting method for our experiments.

[2] In practice, we can use the advantage $A(s,a)$ by subtracting the group average or a learned critic $V(s)$ from all the q_is to reduce variance. This does not change the expected value of the force-field $\hat{u}_h$.

6 EXPERIMENTS

We study the performance of reward drifting against four baselines methods for post-training generative models on three robotic manipulation tasks. Training runs for each baseline are initialized from behavior cloned (BC) policies of identical success rates. All BC policies are trained on the same dataset of demonstrations. All policies have roughly 1 million parameters. All methods use the environment's shaped reward on Push-T, and a sparse reward for reorientation and the transport task. Implementation details are provided in Appendix C.

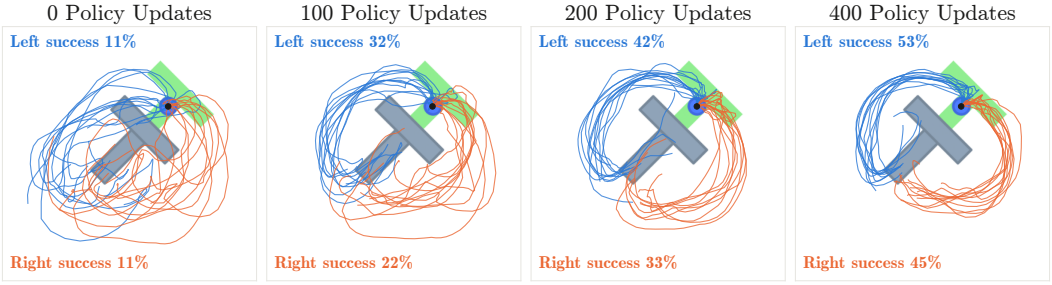

Figure 3: Evolution of PushT trajectories and their success rate during reward drifting. Note that both modes are retained over the course of training and the success rate improves across both modes.

6.1 BASELINES

Diffusion Policy Policy Optimization (DPPO). Ren et al. (2025) treat the denoising chain of a diffusion policy as a Markov decision process and apply PPO to it.

ReinFlow. Zhang et al. (2025) fine-tune a few-step flow-matching policy with PPO. They inject Gaussian noise at each integration step of the flow with a noise injection network that is learned jointly with the policy.

Drifting-based Policy Optimization (DBPO). Gao et al. (2026) fine-tune a one-step drifting policy with PPO by placing a Gaussian over the policy's output. Their method does not use the attraction-repulsion mechanism during the RL process and adds Gaussian noise directly to the actions for exploration.

MeanFlow Policy Optimization (MFPO). Dong et al. (2026) train a one-step MeanFlow policy with an off-policy actor-critic objective. The critic is a Q-function. We report the actor success rate without the best-of-N critic filtering.

6.2 TASKS

Push-T. We study the canonical Push-T task (Figure 3 and Figure 6 from Appendix C), in which a circular agent (the pusher) pushes a T-shaped block into a fixed goal pose. The observation is the agent position and the block pose. The policy predicts a chunk of four planar actions from four observation frames. An episode succeeds when the block covers at least 90 percent of the goal within 200 steps. The demonstrations are the 206 human teleoperation episodes from Chi et al. (2025).

Dexterous in-hand reorientation. We test our method on a contact-rich task (Figure 7) with the 12-DOF D'Claw hand adapted from Chen et al. (2023). The hand must rotate a held object to a random goal orientation. The observation is the joint state of the hand and the pose of the object relative to the goal. The policy outputs one joint-position target per step from a single observation frame. An episode succeeds when the orientation error falls below 0.4 radians with the hand and object at rest and fails when the object is dropped. Episodes last 120 steps. Demonstrations are obtained by running an expert policy in IsaacLab (Mittal et al., 2025) and adding gaussian noise.

Bimanual transport. We demonstrate our method on the long-horizon bimanual transport task (Figure 8) from Robomimic (Mandlekar et al., 2021). The action for a single timestep is 14-dimensional (7 degrees of freedom per robot arm), and the policy predicts a chunk of five actions from a single observation. An episode succeeds when the payload is inside the target bin and the trash is placed

in the trash bin. Episodes last up to 1200 control steps at 20 Hz. The 1029 demonstrations are generated with DexMimicGen (Jiang et al., 2025) in Mujoco Warp (DeepMind & NVIDIA, 2025).

6.3 RL Training Dynamics

Performance. We present the results in Figure 4 below. Reward drifting matches or exceeds the absolute success rate of the baselines and has comparable convergence speed during training. Table 1 highlights the effectiveness of reward drifting in reducing the inference latency; it is at least 3 times faster than the baselines at deployment. We also include an ablation of the entropy force coefficient τ in Figure 13. Preliminary results for vision-based reward drifting are shown in Figure 14.

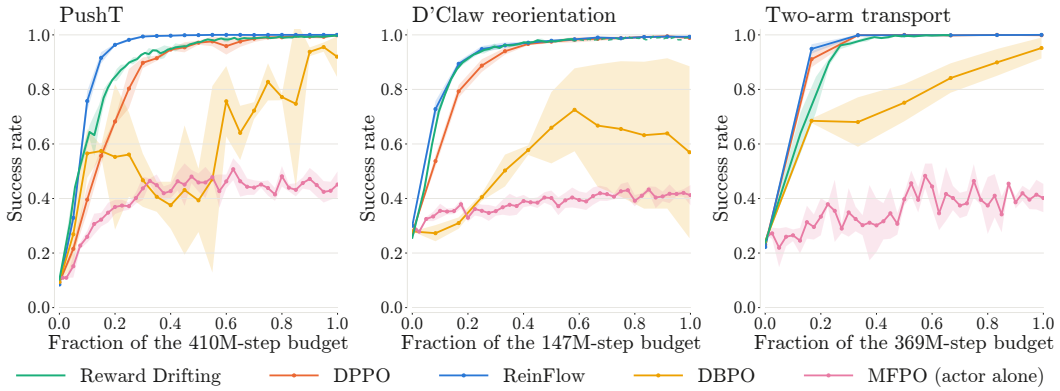

Figure 4: Success rate of the policy against fraction of the training budget for reward drifting and other generative post-training baselines across all 3 tasks. For a given task, each method is initialized from behavior cloning checkpoints of identical success rates. Our method trains more stably compared to the other one-step method, and is comparable to the multi-step flow-matching or diffusion-based methods while being at least 3 times faster at inference time. Note that MFPO has its own off-policy budget outlined in Table 6.

Sampling and Exploration Noise. The exploration noise used during training changes both the stability and the speed of convergence for reward drifting. We compare three schemes that perturb the deterministic action chunk $g(s, 0)$ of the one-step generator $g(s, z)$. *Scaled latent* draws one latent $z \sim \mathcal{N}(0, \zeta^2 I)$ per chunk, with ζ several times the training scale, and executes $g(s, z)$. *Antithetic latent* draws the same z and executes $g(s, 0) + \frac{1}{2}[g(s, z) - g(s, -z)]$ (Hammersley & Morton, 1956; Ren et al., 2019; Greensmith et al., 2004; Wu et al., 2019). *Gaussian action noise* adds independent $\mathcal{N}(0, \sigma^2)$ noise to every action element at every step. Note that for both the scaled latent and antithetic latent, the noise is *not* added to the actions directly, but introduced by perturbing the latent itself. All three are compared at the same action chunk spread (the standard deviation of the sampled chunks per action element).

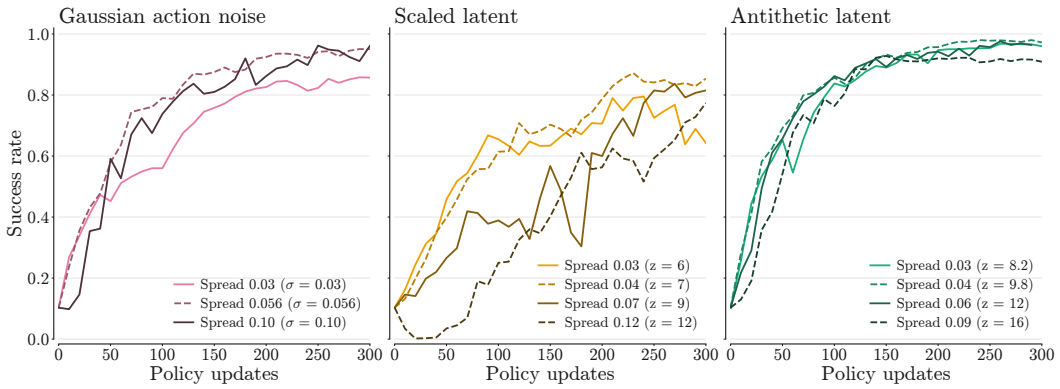

Figure 5: Comparison of exploration noise for reward drifting on PushT. Each panel varies the magnitude of the perturbation. The antithetic latent converges fastest and is the easiest to tune.

Figure 5 illustrates the training dynamics of the various sampling methods on PushT. We observe that antithetic latent noise and gaussian action noise both train stably. The scaled latent approach is harder to tune because it introduces bias into the generator's mean output (in addition to increasing the variance). This occurs because the scale ζ moves the latent outside of the generator's training range. Gaussian action noise is unbiased by construction. Antithetic sampling is less biased as a function of ζ, and is the easiest of the three to tune. Figure 9 compares antithetic and gaussian noise at matched action spread on all three tasks and we observe that antithetic noise performs better across seeds and training checkpoints.

We verify the difference in exploration quality visually on the PushT task to provide additional intuition (Figure 10). Antithetic noise perturbs the trajectory along coherent high value directions, whereas Gaussian noise spreads the variance over every step and dimension leading to less efficient exploration including regions of low or zero value.

Mode Retention Finally, we show (in Figure 3) that reward drifting visually improves the quality of the learned trajectories over the course of training. It also preserves the modes of the PushT policy while it is being post-trained and does not lead to mode collapse as the success rate improves.

Table 1: Inference latency of the state-based policies: median wall-clock time in milliseconds of one policy call (one action chunk) on an NVIDIA RTX PRO 6000. All policies have 1.0M parameters. NFE is number of function evaluations for each action output. Eager: PyTorch eager mode, or op-by-op JAX for MFPO. Compiled: the entire call replayed as one CUDA graph, or JAX JIT for MFPO.

		Eager			Compiled		
Method	NFE	Push-T	Reorient	Transport	Push-T	Reorient	Transport
DPPO	20	6.04	5.95	6.06	0.94	0.93	0.94
ReinFlow	4	0.73	0.74	0.74	0.12	0.12	0.12
DBPO	1	0.82	0.76	0.82	0.17	0.16	0.19
MFPO (actor)	2	89.7	87.6	88.8	0.10	0.11	0.12
Reward Drifting (ours)	1	**0.16**	**0.16**	**0.17**	**0.04**	**0.04**	**0.04**

7 Conclusions, Limitations, and Future Work

We introduced reward drifting, an on-policy method for fine-tuning drifting policies inspired by WPGFs. The resulting policy is fast and the algorithm does not need require likelihood or surrogates. Overall, we believe this makes reward drifting a theoretically grounded and practically useful algorithm for robot learning.

Our work has a few limitations. Every training run in this work starts from a BC policy; when trained from a random initialization the method is not able to learn a policy that captures all of the modes for the task (Figure 12). A natural next step is end-to-end multimodal reinforcement learning without demonstrations, where the same drifting machinery would have to discover the modes rather than simply sharpen them. Second, the update is on-policy and estimates the field from Monte Carlo returns of a batch of actions at every visited state, which is sample-inefficient; the implicit variant additionally re-estimates these returns inside the inner loop, so it is applicable only when rollouts are inexpensive.

On the theoretical side, the algorithms are agnostic to the way the Wasserstein gradient is estimated, so any better estimator of the reward field or any better way to discretize and optimize Wasserstein policy gradient flows can improve the method overall. Work in this direction has already begun in generative modeling, where Gretton et al. (2026); Turan et al. (2026) study Sinkhorn-type proxies for the transport step. Bringing these estimators into the reinforcement learning setting may also result in methods with faster convergence.

7.1 AI Use Statement

AI tools were used for polishing text and orchestrating experiments on GPU clusters. The authors retain full responsibility for the scientific claims and results.

REFERENCES

Xinyue Ai, Yutong He, Albert Gu, Ruslan Salakhutdinov, J Zico Kolter, Nicholas Matthew Boffi, and Max Simchowitz. Joint distillation for fast likelihood evaluation and sampling in flow-based models, 2026. URL https://arxiv.org/abs/2512.02636.

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer, 2005.

Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11 (113):eaea6201, 2026.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84): eadc9244, 2023.

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

Google DeepMind and NVIDIA. Mujoco warp: A gpu-accelerated backend for mujoco built on nvidia warp, 2025. URL https://github.com/google-deepmind/mujoco_warp.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *International Conference on Learning Representations*, 2017.

Xiaoyi Dong, Xi Sheryl Zhang, and Jian Cheng. Mean flow policy optimization, 2026. URL https://arxiv.org/abs/2604.14698.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy-based models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Mathias Dus. Wasserstein formulation of reinforcement learning. an optimal transport perspective on policy optimization. *arXiv preprint arXiv:2604.14765*, 2026.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, volume 2025, pp. 34668–34684, 2025.

Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975.

Yuxuan Gao, Yedong Shen, Shiqi Zhang, Wenhao Yu, Yifan Duan, Jia Pan, Jiajia Wu, Jiajun Deng, and Yanyong Zhang. Drift-based policy optimization: Native one-step policy learning for online robot control. In *Proceedings of the 34th ACM International Conference on Multimedia*. ACM, 2026. URL https://arxiv.org/abs/2604.03540.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2025.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

Arthur Gretton, Li Kevin Wenliang, Alexandre Galashov, James Thornton, Valentin De Bortoli, and Arnaud Doucet. On the wasserstein gradient flow interpretation of drifting models. *arXiv preprint arXiv:2605.05118*, 2026.

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*, pp. 1861–1870. Pmlr, 2018.

John Michael Hammersley and Keith William Morton. A new monte carlo technique: antithetic variates. In *Mathematical proceedings of the Cambridge philosophical society*, volume 52, pp. 449–475. Cambridge University Press, 1956.

Jiaqi Han, Puheng Li, Qiushan Guo, Renyuan Xu, Stefano Ermon, and Emmanuel J Candès. One-step generative modeling via wasserstein gradient flows. *arXiv preprint arXiv:2605.11755*, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Anas Houssaini, Mohamad H Danesh, Amin Abyaneh, Scott Fujimoto, Hsiu-Chin Lin, and David Meger. Drift q-learning. *arXiv preprint arXiv:2606.00350*, 2026.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026. URL https://arxiv.org/abs/2604.15483.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16923–16930. IEEE, 2025.

Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

Bekzhan Kerimkulov, David Šiška, Łukasz Szpruch, and Yufei Zhang. Mirror descent for stochastic control problems with measure-valued controls. *Stochastic Processes and their Applications*, pp. 104765, 2025.

Bekzhan Kerimkulov, James-Michael Leahy, David Siska, Lukasz Szpruch, and Yufei Zhang. A fisher–rao gradient flow for entropy-regularised markov decision processes in polish spaces. *Foundations of Computational Mathematics*, 26(4):2395–2469, 2026.

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

Juil Koo, Mingue Park, Jiwon Choi, Yunhong Min, and Minhyuk Sung. Drifting field policy: A one-step generative policy via wasserstein gradient flow. *arXiv preprint arXiv:2605.07727*, 2026.

Chieh-Hsin Lai, Bac Nguyen, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yuki Mitsufuji, Stefano Ermon, and Molei Tao. A unified view of score-based and drifting models. *arXiv preprint arXiv:2603.07514*, 2026.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. *arXiv preprint arXiv:2601.14234*, 2026.

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

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation, 2021. URL https://arxiv.org/abs/2108.03298.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48104–48127. PMLR, 2025. URL https://proceedings.mlr.press/v267/park25f.html.

David Pfau, Ian Davies, Diana L Borsa, João Guilherme Madeira Araújo, Brendan Daniel Tracey, and Hado Van Hasselt. Wasserstein policy optimization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 49128–49149. PMLR, 2025. URL https://proceedings.mlr.press/v267/pfau25a.html.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

Hongyu Ren, Shengjia Zhao, and Stefano Ermon. Adaptive antithetic sampling for variance reduction. In *International Conference on Machine Learning*, pp. 5420–5428. PMLR, 2019.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

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

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

David Šiška and Yufei Zhang. A note on convergence of wasserstein policy optimization. *arXiv preprint arXiv:2605.22622*, 2026.

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

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, second edition, 2018. ISBN 9780262039246. URL https://mitpress.mit.edu/9780262039246/reinforcement-learning/.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Erkan Turan, Nicolas Dufour, and Maks Ovsjanikov. Generative drifting is secretly score matching: a spectral and variational perspective. *arXiv preprint arXiv:2603.09936*, 2026.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Mike Wu, Noah Goodman, and Stefano Ermon. Differentiable antithetic sampling for variance reduction in stochastic variational inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2877–2886. PMLR, 2019.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024.

Qi Zhang. Positive-only drifting policy optimization. *arXiv preprint arXiv:2604.16519*, 2026.

Ruiyi Zhang, Changyou Chen, Chunyuan Li, and Lawrence Carin. Policy optimization as wasserstein gradient flows. In *International Conference on machine learning*, pp. 5737–5746. PMLR, 2018.

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*, 38: 106282–106319, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Zhaoyu Zhu, Rui Gao, and Shuang Li. Wasserstein policy gradient for entropy-regularized linear-quadratic control. *arXiv preprint arXiv:2608.07433*, 2026a.

Zhaoyu Zhu, Rui Gao, and Shuang Li. Global convergence of wasserstein policy gradient for entropy-regularized reinforcement learning. *arXiv preprint arXiv:2605.26078*, 2026b.

A DRIFTING THROUGH THE LENS OF TRANSPORT

This appendix is for a reader who is more familiar with the language of optimal transport and wants to see what the two updates of the main text, explicit drifting equation 6 and implicit drifting equation 7, are as time discretizations of a Wasserstein gradient flow.

Setting. Let F be a generic functional to be maximized. To keep notation light we drop the state s (every statement below holds state by state, with s carried along as an argument of g_θ), write $z \sim p_z = \mathcal{N}(0, I)$, $q_\theta = (g_\theta)_\# p_z$ for the law of the generator, and $u[q] = \nabla \frac{\delta F}{\delta q}[q]$ for the Wasserstein gradient field. Along the iterates $\theta^1, \theta^2, \dots$ we abbreviate $q^k = q_{\theta^k}$, $u^k = u[q^k]$, and inside expectations $g = g_\theta(z)$, $g^k = g_{\theta^k}(z)$. The useful identity behind everything below is the displacement derivative of F. For a vector field ξ,

$$\frac{d}{d\varepsilon} F\big[(\mathrm{id} + \varepsilon\xi)_\# q\big]\Big|_{\varepsilon=0} \;=\; \mathbb{E}_{x\sim q}\big\langle u[q](x),\, \xi(x)\big\rangle. \tag{18}$$

The Wasserstein gradient flow of F is the solution to the continuity equation

$$\partial_t q_t = -\nabla \cdot (q_t\, u[q_t]), \tag{19}$$

which describes the law of samples X_t that follow the ODE

$$\dot{X}_t = u[q_t](X_t). \tag{20}$$

The two elementary time discretizations of equation 20 with step Δt are

$$X_{k+1} = X_k + \Delta t\, u[q^k](X_k), \qquad \text{(forward Euler)} \tag{21}$$

$$X_{k+1} = X_k + \Delta t\, u[q^{k+1}](X_{k+1}), \qquad \text{(backward Euler)} \tag{22}$$

where q^{k+1} denotes the law of X_{k+1}. Forward Euler is a formula: the field is evaluated at the old law and the old point, and the step is stable only for Δt below the inverse Lipschitz constant of u. Backward Euler is an equation: the field is evaluated at the new law and the new point, and the step is unconditionally stable for the linearized dynamics. The question is which of the two the drifting updates implement.

JKO is backward Euler along the optimal transport map. The Jordan–Kinderlehrer–Otto scheme (Jordan et al., 1998) is a standard method for solving Wasserstein flows variationally. There, equation 19 is discretized by a variational problem in the space of laws,

$$q^{k+1} \;=\; \arg\max_q \Big(F[q] \;-\; \frac{1}{2\Delta t}\, W_2^2(q, q^k) \Big), \tag{23}$$

and q^k converges to the solution of equation 19 as $\Delta t \to 0$. Its first-order optimality condition, with T the optimal transport map from q^k to q^{k+1}, can be written as

$$\frac{T(x) - x}{\Delta t} \;=\; u[q^{k+1}]\big(T(x)\big) \qquad q^k\text{-a.e.}, \tag{24}$$

which is equation 22 for the pair $X_k = x$, $X_{k+1} = T(x)$: JKO *is* implicit Euler, with the pairing of old and new samples given by the optimal transport map. This is the Euler equation for the minimizer of the quadratic perturbation in Ambrosio et al. (2005, Lemma 10.1.2). Comparing the maximizer with the competitor $q = q^k$ gives $F[q^{k+1}] \geq F[q^k] + \frac{1}{2\Delta t} W_2^2(q^{k+1}, q^k)$ for every Δt, with no step-size condition, this means that JKO always improves the functional.

The parametric scheme. Drifting optimizes over parameters, not laws, so the natural cost is the shared-noise displacement

$$C(\theta, \theta^k) \;=\; \mathbb{E}_{z\sim p_z} \big\| g_\theta(z) - g_{\theta^k}(z) \big\|^2, \tag{25}$$

and the parametric JKO step is

$$\theta^{k+1} \;=\; \arg\max_\theta \Big(F[q_\theta] \;-\; \frac{1}{2\Delta t}\, C(\theta, \theta^k) \Big), \tag{26}$$

the scheme of Zhang et al. (2018) and Han et al. (2026). The shared noise is what makes this a transport cost. Old and new samples are generated from the same z, so the map $\hat{T}_\theta := g_\theta \circ g_{\theta^k}^{-1}$, which sends each old sample $g_{\theta^k}(z)$ to the new sample $g_\theta(z)$, pushes q^k to q_θ, and $C(\theta, \theta^k) = \int \|\hat{T}_\theta(x) - x\|^2 \, dq^k(x)$ is the transport cost along $\hat{T}_\theta$. Thus equation 26 is equation 23 with the optimal map T replaced by $\hat{T}_\theta$. Since $\hat{T}_\theta$ is an admissible but not necessarily optimal coupling, $C(\theta, \theta^k) \geq W_2^2(q_\theta, q^k)$, with equality exactly when $\hat{T}_\theta$ is the optimal map. In general parametric JKO is not exactly equation 23, but for an expressive network and a moderate step it is a good approximation of it.

Explicit drifting is forward Euler. The explicit update linearizes F at θ^k. By equation 18 with $\xi \circ g^k = g - g^k$,

$$F[q_\theta] \;\approx\; F[q^k] \;+\; \mathbb{E}_z\big\langle u^k(g^k),\; g - g^k \big\rangle, \tag{27}$$

and completing the square in equation 26,

$$\begin{aligned} \theta^{k+1} &= \arg\max_\theta \Big(\mathbb{E}_z\big\langle u^k(g^k),\, g - g^k \big\rangle \;-\; \tfrac{1}{2\Delta t}\, \mathbb{E}_z \big\| g - g^k \big\|^2 \Big) \\ &= \arg\min_\theta \; \mathbb{E}_z \Big\| g_\theta(z) - \Big[g_{\theta^k}(z) + \Delta t\, u^k\big(g_{\theta^k}(z)\big) \Big] \Big\|^2 . \end{aligned} \tag{28}$$

The target in brackets is a constant computed from the frozen generator, which is where the stop-gradient of Deng et al. (2026) comes from, and for every noise z it is exactly the forward Euler step equation 21 of the sample $g_{\theta^k}(z)$ with the field frozen at q^k.

This can be connected to the explicit loss of the main text. Completing the square once more in equation 6,

$$\|g - g^k - \hat{u}_k\|^2 + \lambda \, \|g - g^k\|^2 \;=\; (1+\lambda) \left\| g - g^k - \tfrac{1}{1+\lambda}\, \hat{u}_k \right\|^2 + \tfrac{\lambda}{1+\lambda} \, \|\hat{u}_k\|^2, \tag{29}$$

and the last term does not depend on θ. Hence the explicit loss with proximity weight λ is exactly equation 28 with

$$\Delta t \;=\; \frac{1}{1+\lambda}, \tag{30}$$

the field's own scale being absorbed into $\hat{u}_k$. The proximity weight is the time step: $\lambda = 0$, the choice of Deng et al. (2026), is the full step $\Delta t = 1$, and $\lambda \to \infty$ is $\Delta t \to 0$.

Implicit drifting is JKO along the shared-noise map. For the implicit update we keep the functional and evaluate its gradient at the new law and the new sample:

$$\theta^{k+1} \;=\; \arg\min_\theta \; \mathbb{E}_z \Big\| g_\theta(z) - \Big[g_{\theta^k}(z) + \Delta t\, u[q_\theta]\big(g_\theta(z)\big) \Big] \Big\|^2, \tag{31}$$

with $\Delta t = 1/(1+\lambda)$ as before, except that the leftover term $\frac{\lambda}{1+\lambda}\|\hat{u}_\theta\|^2$ now depends on θ; it is a penalty on the field at the new samples that vanishes at a fixed point of the flow, and for $\lambda = 0$ it is absent. The loss is non-negative and vanishes exactly when

$$g_\theta(z) \;=\; g_{\theta^k}(z) + \Delta t\, u[q_\theta]\big(g_\theta(z)\big) \qquad p_z\text{-a.e.} \tag{32}$$

Now write this along the shared-noise map. With $x = g_{\theta^k}(z)$ and $\hat{T}_\theta(x) = g_\theta(z)$, equation 32 reads

$$\frac{\hat{T}_\theta(x) - x}{\Delta t} \;=\; u[q_\theta]\big(\hat{T}_\theta(x)\big) \qquad q^k\text{-a.e.}, \tag{33}$$

which is the JKO optimality condition equation 24 with the parameteric transport $\hat{T}_\theta$ in place of the optimal map T and q_θ in place of q^{k+1}. Again, assuming g_θ is expressive enough we may assume $\hat{T}_\theta \approx T$ and implicit euler along $u = \frac{\hat{T}_\theta(x) - x}{\Delta t}$ is roughly a parameteric JKO step.

B Proof of Theorem 1

Proof. Let χ be an admissible perturbation, meaning $\int \chi(a \mid s)\, da = 0$ at every state so that $\pi + \epsilon\chi$ stays normalized, and write $\mathrm{D}X$ for the first variation of a policy-dependent quantity X in the direction χ.

Step 1: differentiate the soft value. By equation 1, $V^\pi(s) = \int \pi\,(Q^\pi - \tau \log \pi)\, da$, and both factors depend on the policy, so

$$\begin{aligned} \mathrm{D}V(s) &= \int \chi \left(Q^\pi - \tau \log \pi\right) da \;+\; \int \pi \left(\mathrm{D}Q - \tau\, \mathrm{D} \log \pi\right) da \\ &= g(s) \;+\; \int \pi(a \mid s)\, \mathrm{D}Q(s,a)\, da, \end{aligned} \tag{34}$$

where $g(s) := \int \chi(a \mid s) \big(Q^\pi(s,a) - \tau \log \pi(a \mid s)\big)\, da$, and the entropy term dropped because $\pi\, \mathrm{D} \log \pi = \mathrm{D}\pi = \chi$ integrates to zero.

Step 2: the implicit term is a resolvent. Neither r nor P depends on the policy, so $\mathrm{D}Q(s,a) = \gamma \int P(s' \mid s,a)\, \mathrm{D}V(s')\, ds'$. Substituting into equation 34 and writing $P_\pi(s' \mid s) = \int \pi(a \mid s)\, P(s' \mid s,a)\, da$ for the state kernel of π,

$$(I - \gamma P_\pi)\, \mathrm{D}V \;=\; g, \qquad \text{so} \qquad \mathrm{D}V \;=\; \sum_{t \geq 0} \gamma^t P_\pi^t\, g. \tag{35}$$

Averaging over $s_0 \sim \mu$ and recognizing $\sum_{t \geq 0} \gamma^t \mu P_\pi^t$ as the occupancy ρ^π,

$$\begin{aligned} \mathrm{D}J \;=\; \int \mu(s_0)\, \mathrm{D}V(s_0)\, ds_0 \;&=\; \int \rho^\pi(s)\, g(s)\, ds \\ &=\; \iint \chi(a \mid s)\, \rho^\pi(s) \Big(Q^\pi(s,a) - \tau \log \pi(a \mid s)\Big)\, da\, ds. \end{aligned}$$

Step 3: take the action gradient. The first variation is whatever pairs with χ, so

$$\frac{\delta J}{\delta \pi(a \mid s)} \;=\; \rho^\pi(s) \Big(Q^\pi(s,a) - \tau \log \pi(a \mid s)\Big), \tag{36}$$

and since $\rho^\pi(s)$ carries no dependence on a, taking ∇_a gives equation 9. □

C IMPLEMENTATION

C.1 MODEL TRAINING DETAILS

Table 2: Task Specification. Every on-policy method is run across the same number of parallel environments and the same simulator budget (number of steps) on a task. MFPO is off-policy and has a separate training schedule (Table 6).

	PushT	D'Claw reorientation	Two-arm transport
Observation	5-d state, 4 frames	109-d state, 1 frame	59-d state, 1 frame
Action	2-d	12-d	14-d
Action chunk (predicted / executed)	4 / 4	1 / 1	5 / 5
Episode length (env steps)	200	120	1200
Parallel environments	4096	4096	2048
Success criterion	coverage ≥ 0.90	rotation error < 0.4 rad	both objects in bins
Discount per env step	0.999	0.998	0.999
On-policy budget (env steps)	4.1×10^8	1.5×10^8	3.7×10^8
On-policy budget (iterations)	500	300	150
BC initialization success rate	0.10	0.26	0.24
Held-out episodes per seed	8192	4096	2048

Table 3: Model Architectures. Every policy has 1.0M parameters on every task; per-task values are given as PushT / D'Claw / transport. Reward Drifting uses a one-step generator; the baselines use their reference implementations at matched size.

	Reward Drifting	DPPO	ReinFlow	DBPO	MFPO
Backbone	MLP, 3 hidden layers, SiLU	residual MLP, 3 hidden layers, ReLU	residual MLP, 3 hidden layers, ReLU	1-D UNet, kernel 5, GroupNorm	MLP, 3 hidden layers, GELU, LayerNorm
Conditioning	encoder MLP 2×256 (LayerNorm, SiLU)	concatenated	concatenated	FiLM	concatenated
Time embedding	–	sinusoidal, 16-d	sinusoidal, 16-d	–	two Fourier embedders, 128-d
Hidden width	608	696 / 670 / 654	696 / 670 / 654	[32, 64, 128] / 144 / 152	672 / 649 / 634
Params (M)	0.98 / 1.01 / 1.03	1.01 / 1.00 / 1.00	1.01 / 1.00 / 1.00	1.00 / 0.97 / 1.02	1.00 / 1.00 / 1.00
Sampling steps	1	20 (last 10 fine-tuned)	4 (20 at BC)	1	2
Exploration	antithetic latent noise, $z \in \mathbb{R}^{16}$	denoising noise, std ≥ 0.1	learned noise net 2×64, std 0.05–0.12	Gaussian, std 0.05–0.15	sampling noise of the flow
Critic	V, MLP 2×256, LayerNorm, SiLU	V, MLP 3×256, Mish	V, MLP 3×256, Mish	V, MLP 3×256, Mish	twin distributional Q, MLP 3×256

Table 4: Reward Drifting hyperparameters. The latent scale is computed at the beginning of training such that the policy's sampled action chunks reach the spread target and held fixed for the rest of training. Shared across tasks: 64 generator samples per state, kernel radii $\{0.2, 0.05, 0.02\}$, antithetic latent noise, advantages standardised over the batch, actor AdamW 2×10^{-5} with an EMA of 0.98 per update, critic AdamW 10^{-3} for 4 epochs at minibatch 4096 with 5 warmup updates, evaluation every 10 updates.

	PushT	D'Claw reorientation	Two-arm transport
Policy updates	1000	300	200
Chunk spread target (upper bound)	0.03 (0.10)	0.096 (0.25)	0.072 (0.20)
GAE λ	0.95	0.85	0.97
Advantage weight $\exp(A/\kappa)$	$\kappa = 1.0$	$\kappa = 0.7$	$\kappa = 1.0$
Evaluation environments	2048	1024	512

Table 5: On-policy baseline hyperparameters (PPO-style fine-tuning). All three run the iteration count, environments and rollout length of Table 2; one iteration collects one episode per environment. Values that differ by task are given as PushT / D'Claw / transport.

	DPPO	ReinFlow	DBPO
Discount	0.996 / 0.998 / 0.995	0.996 / 0.998 / 0.995	0.996 / 0.998 / 0.995
GAE λ	0.95	0.95	0.95
Actor learning rate	10^{-4} (constant)	4.5×10^{-5}, cosine to 2×10^{-5}	10^{-4} (constant)
Critic learning rate	10^{-3}	6.5×10^{-4}	10^{-4}
PPO clip	0.01	0.01	0.05
Target KL	1.0	1.0	0.1
Value coefficient	0.5	0.5	0.5
Entropy coefficient	–	0	0.01
Gradient clip	–	–	0.5
Critic warmup	10 iterations	10 iterations	10 iterations
Exploration	10 fine-tuned denoising steps	noise std in $[0.05, 0.12]$, learned; 4 inference steps	Gaussian std in $[0.05, 0.15]$

Table 6: MFPO hyperparameters (off-policy). MFPO collects with 1024 environments for 4000 collection steps, which is 1.6×10^7 / 4.1×10^6 / 2.0×10^7 env steps on PushT / D'Claw / transport. The actor is evaluated alone (no critic-guided candidate selection).

Collection	1024 environments, 4000 steps, 50 warm-up collection steps
Updates per collection step	64 update calls, one critic and one actor step each
Batch Size	256
Discount per chunk	0.996 / 0.998 / 0.995
Critic	twin distributional Q, clipped double Q, Adam 3×10^{-4}
Actor	Adam 10^{-5}

Table 7: Vision variant. Observations are features of a frozen DINOv3 ViT-S/16 (mean of the class and register tokens, 384-d per camera); the policy and training procedure are otherwise those of the state variant. Values are PushT / D'Claw / transport.

Image	112^2, 1 camera / 112^2, 1 camera / 80^2, 5 cameras
Observation window	4 frames / 8 frames / 1 frame per camera
Baseline RL	1024 / 512 / 512 environments, 500 / 300 / 150 iterations; otherwise as Table 5

C.2 Task Illustrations

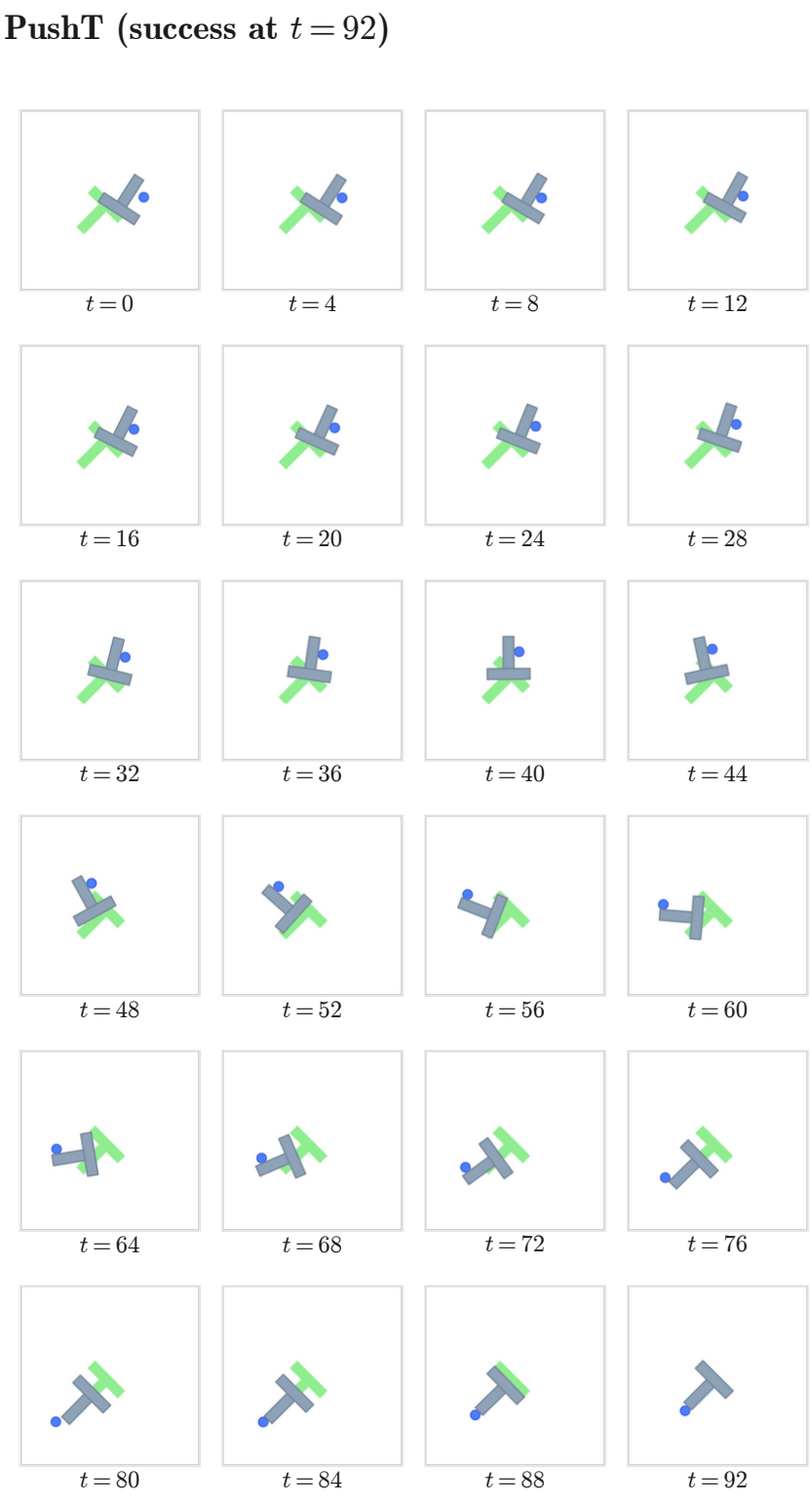

Figure 6: Filmstrip for a successful rollout of the Push-T task.

D'Claw reorientation (success at $t = 39$)



Figure 7: Filmstrip for a successful rollout of the D'Claw In-hand Reorientation task.

Two-arm transport (success at $t = 327$)



Figure 8: Filmstrip for a successful rollout of the Two Arm Transport task.

C.3 Additional Figures and Tables

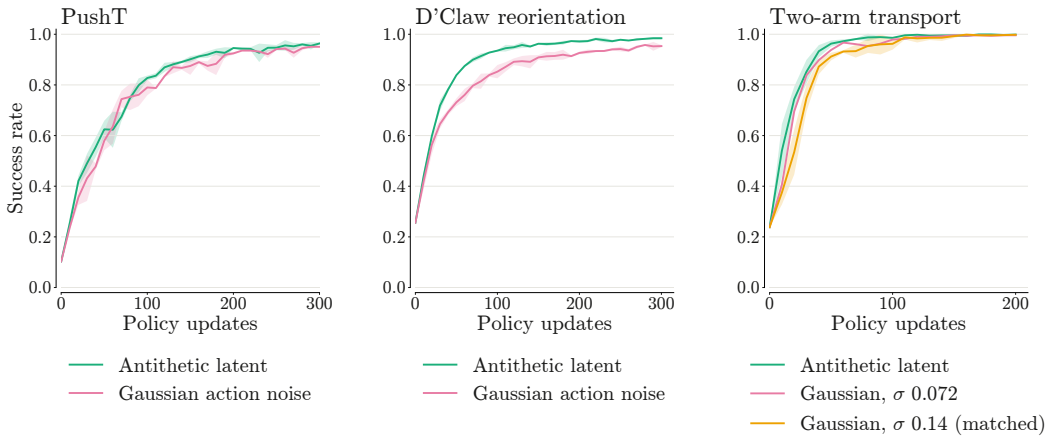

Figure 9: A comparison of training trajectories using antithetic latent sampling and gaussian action noise across all tasks. Antithetic sampling consistently outperforms and converges faster than directly adding gaussian noise to the actions.

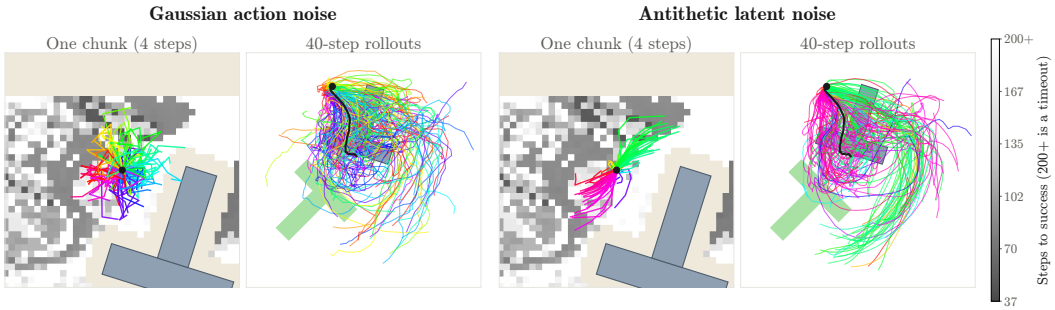

Figure 10: Exploration from a randomly picked mid-episode PushT state. For each noise sampling scheme, **Left**: 96 single exploration chunks (4 steps each). The grey heatmap shows the number of steps the same policy needs to succeed from each pusher position which can be interpreted as a proxy for the value function. Antithetic noise tends to sample more trajectories that have higher value. **Right**: 128 closed-loop rollouts of forty steps with fresh noise every chunk. The deterministic rollout is shown in black. Gaussian noise explores all directions uniformly including the white regions of the heatmap where the value function is zero. Antithetic noise does not allocate any exploration trajectories to the white region making its exploration more efficient.

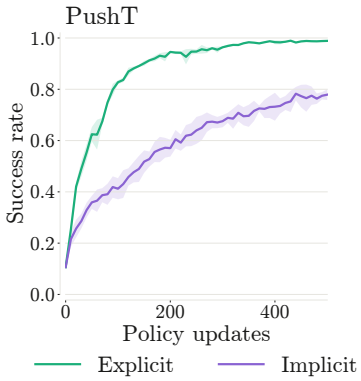

Figure 11: Implicit drifting underperforms compared to the simpler explicit drifting approach on the PushT task. The experimental setup remains consistent with other PushT evaluations of this work.

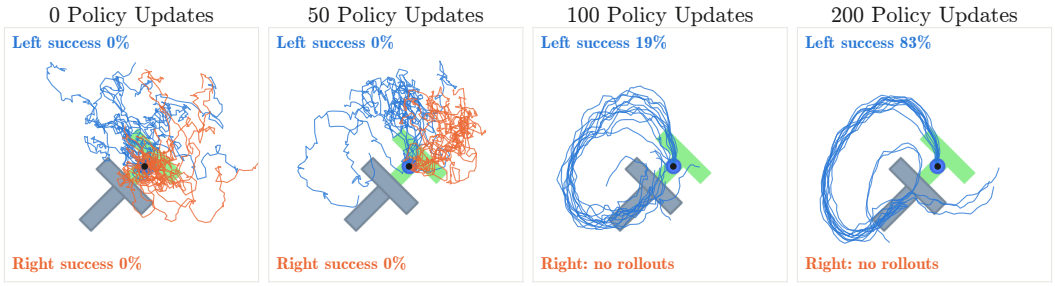

Figure 12: Reward drifting with a randomly initialized network (instead of a pre-trained BC model) on PushT. Initially, the trajectories are completely random. They become more structured as training progresses until one of the modes is discovered. This is then refined further by reward drifting. The policy only learns one of the modes successfully.

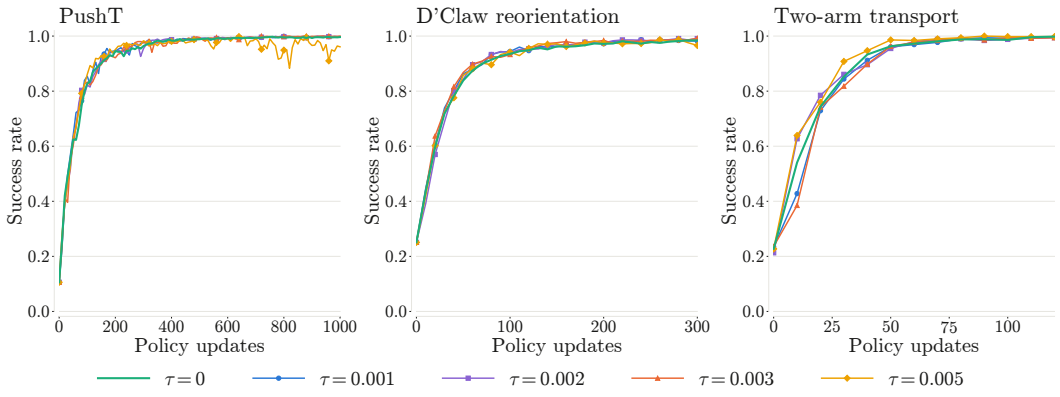

Figure 13: Ablation of the entropy force coefficient τ across all tasks. On the transport task, we observe that a higher value of τ can lead to marginally faster convergence.

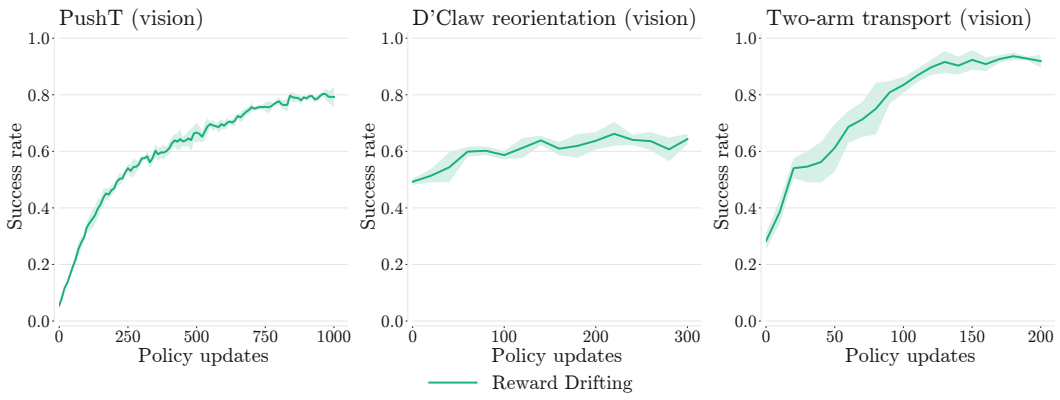

Figure 14: Preliminary results for vision-based reward drifting for the 3 tasks. Vision training uses pooled DINOv3 (Siméoni et al., 2025) features. Additional details can be found in Table 7. We observe an improvement in success rate across all tasks.