Drifting Field Policy: A One-Step Generative Policy via Wasserstein Gradient Flow

Juil Koo **Mingue Park** **Jiwon Choi** **Yunhong Min** **Minhyuk Sung**
KAIST
{63days, kicikicik}@kaist.ac.kr, jwchoi1529@gmail.com,
{dbsghd363, mhsung}@kaist.ac.kr

Abstract

We propose **Drifting Field Policy (DFP)**, a non-ODE one-step generative policy built on the *drifting model* paradigm [10]. We frame the policy update as a reverse-KL Wasserstein-2 gradient flow toward a soft target policy, so that each DFP update corresponds to a gradient step in probability space. By construction, this gradient is decomposed into an ascent toward higher action-value regions and a score matching with the anchor policy as a trust region. We further derive a simple, tractable surrogate of the otherwise intractable update loss, akin to behavior cloning on top-K critic-selected actions. We find empirically that this mechanism uniquely benefits the drifting backbone owing to its non-ODE parameterization. With one-step inference, DFP achieves state-of-the-art performance on several manipulation tasks across Robomimic and OGBench, outperforming ODE-based policies.

1 Introduction

Offline-to-online reinforcement learning (RL) has emerged as a practical paradigm for continuous control [42, 30, 43, 3], where an RL agent is first pretrained on static demonstrations and then refined through online interaction. To faithfully capture the multimodal action distributions of real-world demonstrations beyond unimodal Gaussians, the field has increasingly turned to generative policies, with ODE-based backbones such as diffusion and flow policies [62, 21, 11, 45, 12] proving particularly effective. To further meet the low-latency demands of deployment, recent work has converged on one-step inference, realized by one-step variants of these ODE backbones [14, 18, 56, 27] that amortize ODE trajectory integration into a single forward pass.

Despite the success of generative policies in offline behavior cloning [4, 24, 48, 9], RL finetuning exposes a structural burden for ODE-based parameterizations: a reward signal defined at the action must propagate back through the entire ODE trajectory, posing a non-trivial output-to-trajectory credit assignment problem [50, 35, 11, 40, 5, 28]. Crucially, this burden persists even in one-step variants, whose training objective is still defined along the ODE path. This motivates us to seek a one-step policy parameterization that bypasses trajectory integration entirely, so that output-level reward signals can act directly at the action level.

We propose **Drifting Field Policy (DFP)**, a non-ODE one-step generative policy built on *drifting models* [10]. The policy is a single-pass pushforward map $\pi_\theta(\cdot|s) = [f_\theta(\cdot, s)]_\# p_\epsilon$ from a prior p_ϵ to the action space, with no time variable. Drifting models are trained via a *drifting field* update that combines attraction toward a target distribution with repulsion from the current model distribution, driving π_θ toward the target. In RL fine-tuning, with the soft policy improvement target $\pi^+ \propto \pi_{\text{old}} \exp(Q/\alpha)$ [33, 20], we frame the corresponding drifting field update as a reverse-KL Wasserstein-2 gradient flow [6] that minimizes $\mathrm{KL}(\pi_\theta \| \pi^+)$: the ideal drift field follows the steepest-descent direction toward π^+ on probability space. We further show that this gradient is structurally

Preprint.

decomposed into a $\nabla_a Q$ ascent direction and a score matching with the anchor policy π_{old} as a trust region.

However, π^+ in this KL is intractable due to the normalizing constant. We therefore propose a simple yet effective tractable surrogate that replaces π^+ with the top-K critic-selected actions as positive targets. The training objective under this approximation is akin to behavior cloning on these K self-generated candidates, thus easy to implement with no architectural change. We prove this surrogate has bounded approximation error to the ideal update. Since DFP performs gradient descent directly on *probability space*, each step shifts the policy distribution directly at the action output, in contrast to diffusion and flow policies [62, 21, 11, 45, 66] that update the velocity prediction defining their ODE, spreading each signal across the ODE trajectory. Ablations confirm this distinction: the same top-K supervision yields only marginal gains on a MeanFlow backbone [18].

We summarize our contributions as follows: (i) We introduce the first application of drifting models to RL fine-tuning, framing the ideal drifting-field update as a Wasserstein-2 gradient flow direction on probability space — the steepest-descent direction toward the soft policy improvement target π^+ (Sec. 3). (ii) We derive a tractable top-K surrogate of the otherwise intractable target π^+ with a bounded approximation error to the ideal policy improvement loss (Proposition 1), and show that this top-K supervision is particularly effective on the drifting backbone, in contrast to ODE-based one-step backbones whose velocity-level updates can spread each positive's signal across the ODE trajectory (Sec. 3.2). (iii) On 12 tasks across Robomimic [39] and OGBench [44] benchmarks, DFP achieves state-of-the-art performance on 9 of 12 tasks and second-best on the remaining 3, outperforming prior ODE-based generative policies, such as QC-FQL [36] and MVP [66], by a large margin on average (Sec. 5).

2 Preliminaries

In this section, we briefly review the three building blocks of our method: the offline-to-online RL setting and the off-policy actor-critic framework (Sec. 2.1), drifting models as a one-step generative paradigm (Sec. 2.2), and the Wasserstein gradient flow interpretation of the drifting field (Sec. 2.3).

2.1 Offline-to-Online Reinforcement Learning

We consider a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A} \subseteq \mathbb{R}^d$, transition dynamics $P(s'|s,a)$, reward $r(s,a) \in \mathbb{R}$, and discount factor $\gamma \in [0,1)$. The objective of reinforcement learning is to find a policy $\pi(\cdot|s)$ maximizing the expected discounted return $J(\pi) = \mathbb{E}_\pi\big[\sum_{k=0}^{\infty} \gamma^k r(s_k, a_k)\big]$.

A common training strategy in offline-to-online RL is to first pretrain a policy on a static offline dataset $\mathcal{D}_{\text{offline}}$ (offline stage) and then fine-tune it with a limited budget of online interactions (online stage). We denote by $\mathcal{D}$ the replay buffer that accumulates both $\mathcal{D}_{\text{offline}}$ and the online transitions, and by π_β the (potentially unknown) behavior distribution that generates the data in $\mathcal{D}$.

We adopt the standard off-policy actor-critic framework [29], in which the critic Q_ϕ estimates the expected discounted return for each state-action pair via a temporal difference loss, and the actor π_θ is updated to maximize Q_ϕ under a behavioral constraint [64, 15, 59] that keeps it close to π_β. The two are jointly trained by minimizing:

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\big[(Q_\phi(s,a) - r - \gamma\, Q_{\bar{\phi}}(s', a'))^2\big], \quad a' \sim \pi_\theta(\cdot|s'), \tag{1}$$

$$\mathcal{L}_\pi(\theta) = -\mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi_\theta(\cdot|s)}[Q_\phi(s,a)], \quad \text{s.t. } D(\pi_\theta(\cdot|s),\, \pi_\beta(\cdot|s)) \le \varepsilon, \tag{2}$$

where $Q_{\bar{\phi}}$ is a target network [37, 41] and D is a divergence between π_θ and π_β. In Sec. 3, we instantiate this constrained actor objective with a novel one-step generative policy built on drifting models.

2.2 Drifting Models

Drifting models [10] are a recent paradigm for one-step generative modeling. Let $p := p_{\text{data}}$ denote the data distribution on $\mathbb{R}^d$ and $p_\epsilon := \mathcal{N}(0, I)$ a prior on $\mathbb{R}^m$. Instead of describing transport from p_ϵ to p through a stochastic process or its ODE counterpart at inference as in diffusion and flow models [23, 55, 57, 38], drifting models shift the dynamics to training: they directly parameterize

single-pass pushforward map $f_\theta : \mathbb{R}^m \to \mathbb{R}^d$ trained so that the model distribution $q := [f_\theta]_\# p_\epsilon$ matches p.

The core of drifting models is the *drifting field* $\mathbf{V}_{p,q} = \mathbf{V}_p^+ - \mathbf{V}_q^-$, a vector field that comprises an attraction term $\mathbf{V}_p^+$ and a repulsion term $\mathbf{V}_q^-$, both constructed via kernel mean shift:

$$\mathbf{V}_p^+(x) = \frac{\mathbb{E}_{y^+\sim p}\big[k(x,y^+)(y^+ - x)\big]}{\mathbb{E}_{y^+\sim p}[k(x,y^+)]}, \qquad \mathbf{V}_q^-(x) = \frac{\mathbb{E}_{y^-\sim q}\big[k(x,y^-)(y^- - x)\big]}{\mathbb{E}_{y^-\sim q}[k(x,y^-)]}, \tag{3}$$

where $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{>0}$ is a similarity kernel with bandwidth h (e.g., the Gaussian kernel $k(x,y) = \exp(-\|x-y\|^2/(2h^2))$), and $y^+ \sim p$ and $y^- \sim q$ denote positive and negative samples drawn from the two distributions. Intuitively, the attraction term pulls generated samples toward nearby data, while the repulsion term pushes them away from one another to prevent mode collapse.

By construction, the drifting field is anti-symmetric, $\mathbf{V}_{p,q}(x) = -\mathbf{V}_{q,p}(x)$, which implies $q = p \Rightarrow \mathbf{V}_{p,q} \equiv 0$. Under mild kernel regularity conditions, the converse also holds in the sense that $\mathbf{V}_{p,q} \approx 0 \Rightarrow q \approx p$ [10]. Drifting models are trained to satisfy $\mathbf{V}_{p,q} = 0$ via fixed-point regression with the stop-gradient operator "sg" on the drifted target:

$$\mathcal{L}_{\text{drift}}(\theta;\, p, q) \;=\; \mathbb{E}_{\epsilon\sim p_\epsilon}\big[\|x - \text{sg}(x + \mathbf{V}_{p,q}(x))\|^2\big], \text{ where } x = f_\theta(\epsilon). \tag{4}$$

The drift loss is parameterized by the positive (target) distribution p and the negative (source) distribution q, and the same form applies to any choice of (p, q) beyond unconditional generative modeling. In Sec. 3, we exploit this flexibility by plugging in different positive distributions for policy improvement and behavior cloning.

2.3 Drifting Field as a Wasserstein Gradient Flow

A recent line of work [6, 22] reveals that the drifting field $\mathbf{V}_{p,q}$ of Sec. 2.2 is precisely the particle velocity of a Wasserstein-2 gradient flow (WGF) under KDE-smoothed densities. We recall this identification, which underpins the policy-space treatment in Sec. 3.

Wasserstein-2 gradient flow. Equip the space $\mathcal{P}_2(\mathbb{R}^d)$ of probability measures with finite second moment with the Wasserstein-2 distance W_2 [2, 51]. An absolutely continuous curve $\{q_t\}_{t\geq 0} \subset \mathcal{P}_2(\mathbb{R}^d)$ is characterized by a time-dependent velocity field $v_t : \mathbb{R}^d \to \mathbb{R}^d$ that transports each particle along $\dot{x}_t = v_t(x_t)$ and induces the marginal evolution $\partial_t q_t + \nabla \cdot (q_t v_t) = 0$ through the continuity equation. For a smooth functional $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^d) \to \mathbb{R}$ with first variation $\frac{\delta \mathcal{F}}{\delta q}$, the W_2 gradient flow [25] of $\mathcal{F}$ is the absolutely continuous curve whose velocity field is the steepest-descent direction $v_t(x) = -\nabla_x \frac{\delta \mathcal{F}}{\delta q_t}(x)$, equivalently characterized by the PDE

$$\partial_t q_t = \nabla \cdot \left(q_t \nabla_x \frac{\delta \mathcal{F}}{\delta q_t} \right), \tag{5}$$

the W_2-gradient flow interpretation of "$\dot{q}_t = -\nabla \mathcal{F}(q_t)$" on $\mathcal{P}_2$. Specializing to $\mathcal{F}(q) = \text{KL}(q\|p)$, the velocity reduces to the score difference (See Appendix C.1 for derivation)

$$v_t(x) = \nabla_x \log p(x) - \nabla_x \log q_t(x), \tag{6}$$

which attracts particles toward p, repels them from the current q_t, and monotonically dissipates $\text{KL}(q_t\|p)$ so that $q_t \to p$ in W_2 as $t \to \infty$ [2].

Drifting field as KDE-approximated WGF velocity. The scores in Eq. (6) are not available in closed form, since p and q_t are typically only accessible through samples. Replacing each density with its KDE smoothing $\mu_{\text{kde}}(x) := \int k_h(x,y)\, d\mu(y)$ under a Gaussian kernel $k_h(x,y) = \exp(-\|x-y\|^2/(2h^2))$ yields the score identity [8]

$$h^2\, \nabla_x \log \mu_{\text{kde}}(x) = \frac{\int k_h(x,y)\,(y-x)\, d\mu(y)}{\int k_h(x,y)\, d\mu(y)}. \tag{7}$$

Substituting Eq. (7) into Eq. (6), one can obtain the following identity:

$$h^2\, [\nabla_x \log p_{\text{kde}}(x) - \nabla_x \log q_{\text{kde}}(x)] = \mathbf{V}_p^+(x) - \mathbf{V}_q^-(x) = \mathbf{V}_{p,q}(x). \tag{8}$$

Consequently, the drifting loss of Eq. (4) is a parametric KDE-WGF descent of $\text{KL}(q\|p)$, a viewpoint we leverage in Sec. 3 by specializing (p, q) to the policy learning.

3 Drifting Field Policy

We propose **Drifting Field Policy (DFP)**, a novel one-step generative policy method for RL fine-tuning based on drifting models [10]. The policy is a single-pass map $f_\theta : \mathbb{R}^k \times \mathcal{S} \to \mathcal{A}$ that induces a state-conditional pushforward distribution on the action space,

$$\pi_\theta(\cdot|s) \;:=\; [f_\theta(\cdot, s)]_\# p_\epsilon. \tag{9}$$

DFP trains π_θ with the drifting training loss $\mathcal{L}_{\text{drift}}(\theta;\, p, q)$ of Sec. 2.2, identifying the model distribution q with π_θ and instantiating the positive distribution p at two complementary targets: the Q-maximizing target π^+ for policy improvement and the behavior distribution π_β for data anchoring. Sec. 3.1 derives the training objective with its Wasserstein gradient flow interpretation and tractable top-K surrogate; Sec. 3.2 contrasts with diffusion and flow policies.

3.1 Policy Improvement via Wasserstein Gradient Flow

Let π_{old} denote a trust-region anchor policy. At each iteration, the goal is to update the current policy π_θ to maximize Q within a trust region around π_{old}, given by the standard KL-regularized objective from optimal control [26, 60] and policy search [52, 47, 1, 34, 42, 46]:

$$\pi^+(\cdot|s) := \arg\max_\pi \; \mathbb{E}_{a\sim\pi(\cdot|s)}\big[Q_\phi(s,a)\big] \;-\; \alpha\, D_{\text{KL}}\big(\pi(\cdot|s)\,\big\|\,\pi_{\text{old}}(\cdot|s)\big), \tag{10}$$

with temperature $\alpha > 0$. This optimal policy π^+ admits the closed-form solution [33]:

$$\pi^+(a|s) = \frac{\pi_{\text{old}}(a|s)\exp(Q_\phi(s,a)/\alpha)}{Z(s)}, \quad Z(s) = \int \pi_{\text{old}}(a'|s)\exp(Q_\phi(s,a')/\alpha)\, da'. \tag{11}$$

To realize this update under the drifting model parameterization, we instantiate the drifting loss $\mathcal{L}_{\text{drift}}(\theta;\, p, q)$ of Sec. 2.2 with $p = \pi^+$ as the positive target and $q = \pi_\theta$ as the negative source. Letting $\hat{a} := f_\theta(\epsilon, s)$,

$$\mathcal{L}_{\text{PI}}(\theta) = \mathcal{L}_{\text{drift}}(\theta;\, \pi^+, \pi_\theta) \;=\; \mathbb{E}_{s,\epsilon}\big[\|\hat{a} - \text{sg}(\hat{a} + \mathbf{V}_{\pi^+,\pi_\theta}(\hat{a}|s))\|^2\big], \tag{12}$$

where $\mathbf{V}_{\pi^+,\pi_\theta} = \mathbf{V}^+_{\pi^+} - \mathbf{V}^-_{\pi_\theta}$ is the drifting field between π^+ and π_θ.

Remark 1 ($\mathcal{L}_{\text{PI}}$ as $\nabla_a Q$ ascent with score matching regularization)**.** *Specializing the result of Sec. 2.3 to $(p, q) = (\pi^+, \pi_\theta)$, $\mathbf{V}_{\pi^+,\pi_\theta}$ is the KDE-approximated Wasserstein-2 gradient flow velocity of $\text{KL}(\pi_\theta\|\pi^+)$ on policy space [6],*

$$\mathbf{V}_{\pi^+,\pi_\theta}(a|s) \;=\; h^2\big[\nabla_a \log \pi^+_{\text{kde}}(a|s) - \nabla_a \log \pi_{\theta,\text{kde}}(a|s)\big], \tag{13}$$

so in the ideal nonparametric continuous-time limit, this field gives a KL-dissipating update direction toward π^+. Substituting $\nabla_a \log \pi^+ = \frac{1}{\alpha}\nabla_a Q_\phi + \nabla_a \log \pi_{\text{old}}$ into Eq. (13) *and the small-bandwidth limit $\log p_{\text{kde}} \to \log p$ yields the structural decomposition*

$$\mathbf{V}_{\pi^+,\pi_\theta}(a|s) \;\simeq\; \underbrace{\frac{h^2}{\alpha}\nabla_a Q_\phi(s,a)}_{\nabla_a Q \text{ ascent}} \;+\; \underbrace{h^2\left(\nabla_a \log \pi_{\text{old}}(a|s) - \nabla_a \log \pi_\theta(a|s)\right)}_{\text{trust region around } \pi_{\text{old}} \text{ via score matching}}, \tag{14}$$

revealing that the ideal drift field contains an action-space $\nabla_a Q$ ascent component at temperature α regularized by score matching between π_θ and the anchor π_{old}, requiring neither the critic Jacobian $\nabla_a Q_\phi$ of DDPG-style actor updates [37] nor an explicit KL computation. Derivation in Appendix C.1.

Tractable surrogate. While Eq. (12) provides the desired soft policy update [33, 20] without an explicit critic Jacobian or KL computation, it is itself not directly tractable: $\pi^+ \propto \pi_{\text{old}}\exp(Q_\phi/\alpha)$ has an intractable normalization $Z(s)$ and cannot be sampled directly. Thus, we propose a simple yet effective surrogate loss as follows. A natural starting point is self-normalized importance sampling: drawing N candidate actions $a^{(1)}, \ldots, a^{(N)} \overset{\text{i.i.d.}}{\sim} \pi_{\text{old}}(\cdot|s)$ and weighting them by $w_j \propto \exp(Q_\phi(s, a^{(j)})/\alpha)$ to yield an estimator of expectations under π^+. We observe, however, that an even simpler scheme — replacing the weights with a uniform hard top-K cutoff, so the top-K candidates by Q_ϕ are equally weighted — is empirically robust to K over a wide range and admits a bounded approximation error to $\mathcal{L}_{\text{PI}}$ (Proposition 1).

Algorithm 1 Drifting Field Policy (DFP), online fine-tuning

Input: BC-pretrained policy $\pi_\theta(\cdot|s) = [f_\theta(\cdot, s)]_{\#}p_\epsilon$ and critic Q_ϕ, replay buffer $\mathcal{D}$ initialized with offline data, N candidates for $\mathcal{L}_{\text{top-}K}$, N' candidates for best-of-N' execution [19], N_{gen} samples of $\hat{a} \sim \pi_\theta(\cdot|s)$
Initialize old policy $\pi_{\text{old}} \leftarrow \pi_\theta$
for online training step $k = 1, 2, \ldots$ **do**
  Observe s_k; draw $\{a^{(i)}\}_{i=1}^{N'} \sim \pi_\theta(\cdot|s_k)$ and execute $a_k^\star \leftarrow \arg\max_j Q_\phi(s_k, a^{(j)})$
  Receive s_{k+1}, r_k; append $(s_k, a_k^\star, r_k, s_{k+1})$ to $\mathcal{D}$
  Sample mini-batch $\{(s^{(b)}, a^{(b)})\}_{b=1}^{B} \sim \mathcal{D}$
  for $b = 1, 2, \ldots, B$ **do** ▷ in parallel
    Generate $\hat{a}^{(b,1)}, \ldots, \hat{a}^{(b,N_{\text{gen}})} \sim \pi_\theta(\cdot|s^{(b)})$ ▷ $\hat{a}$ shared by both $\mathcal{L}_{\text{BC}}$ and $\mathcal{L}_{\text{top-}K}$
    Draw N candidates $\tilde{a}^{(b,1)}, \ldots, \tilde{a}^{(b,N)} \sim \pi_{\text{old}}(\cdot|s^{(b)})$
    Select $P_K(s^{(b)}) = \{\tilde{a}^{(b,j)} : j \in \text{argTopK}_{j\in[N]} Q_\phi(s^{(b)}, \tilde{a}^{(b,j)})\}$
  end for
  Update θ by minimizing $\mathcal{L}_{\text{BC}}(\theta) + \lambda \mathcal{L}_{\text{top-}K}(\theta)$ via Eqs. (18), (16)
  Update ϕ via the Bellman backup of Eq. (1)
  Update old policy: $\theta_{\text{old}} \leftarrow \tau_{\text{EMA}}\, \theta + (1 - \tau_{\text{EMA}})\, \theta_{\text{old}}$
end for

Denoting the resulting positive set by

$$P_K(s) := \{a^{(j)} : j \in \text{argTopK}_{j\in[N]} Q_\phi(s, a^{(j)})\}, \qquad a^{(j)} \overset{\text{i.i.d.}}{\sim} \pi_{\text{old}}(\cdot|s), \tag{15}$$

the tractable surrogate is

$$\mathcal{L}_{\text{top-}K}(\theta) = \mathcal{L}_{\text{drift}}(\theta;\, P_K, \pi_\theta) = \mathbb{E}_{s,\epsilon}\big[\|\hat{a} - \text{sg}(\hat{a} + \mathbf{V}_{P_K,\pi_\theta}(\hat{a}|s))\|^2\big], \quad \hat{a} \sim \pi_\theta(\cdot|s), \tag{16}$$

with the drifting field $\mathbf{V}_{P_K,\pi_\theta}$ taking the empirical set P_K as the positive set and π_θ as the negative distribution.

Proposition 1 (Bounded approximation error of $\mathcal{L}_{\text{top-}K}$ to $\mathcal{L}_{\text{PI}}$)**.** *Let $\rho := K/N$. With bounded Lipschitz kernel k of Sec. 2.2, assume $Q_\phi(s, A)$ for $A \sim \pi_{\text{old}}(\cdot|s)$ admits a strictly positive density at its $(1-\rho)$-quantile $q^\rho(s)$. Define the ρ-level-set tilting $\tilde{\pi}^\rho(a|s) := \rho^{-1}\, \mathbb{1}[Q_\phi(s, a) \geq q^\rho(s)]\, \pi_{\text{old}}(a|s)$. As $N \to \infty$ with ρ fixed,*

$$\mathcal{L}_{\textit{top-}K}(\theta) \to \mathcal{L}_{\text{drift}}(\theta;\, \tilde{\pi}^\rho, \pi_\theta), \qquad \big|\mathcal{L}_{\text{drift}}(\theta;\, \tilde{\pi}^\rho, \pi_\theta) - \mathcal{L}_{\text{PI}}(\theta)\big| \leq C\, \overline{\text{TV}}\big(\tilde{\pi}^\rho, \pi^+\big), \tag{17}$$

for any $\rho \in (0, 1]$ and a finite constant C depending only on the kernel and action-space diameter, where $\overline{\text{TV}}(p, q) := \mathbb{E}_s[\text{TV}(p(\cdot|s), q(\cdot|s))]$. The derivation combines Glivenko-Cantelli on the empirical $(1-\rho)$-quantile with TV-Lipschitz continuity of the kernel mean shift; full proof in Appendix C.2.

To anchor π_θ to the behavior distribution π_β, we additionally use a behavior cloning (BC) drift loss with empirical positives drawn from the replay buffer $\mathcal{D}$:

$$\mathcal{L}_{\text{BC}}(\theta) = \mathcal{L}_{\text{drift}}(\theta;\, \mathcal{D}, \pi_\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D},\,\epsilon}\big[\|\hat{a} - \text{sg}(\hat{a} + \mathbf{V}_{\mathcal{D},\pi_\theta}(\hat{a}|s))\|^2\big], \quad \hat{a} \sim \pi_\theta(\cdot|s). \tag{18}$$

Although $\mathcal{L}_{\text{BC}}$ and $\mathcal{L}_{\text{top-}K}$ originate from different objectives — data anchoring versus Q-maximizing policy improvement — they are both instances of $\mathcal{L}_{\text{drift}}(\theta;\, p, q)$ (Eq. (4)), sharing the same negative $q = \pi_\theta$ and differing only in the empirical positive set:

- $\mathcal{L}_{\text{BC}}(\theta) = \mathcal{L}_{\text{drift}}(\theta;\, \mathcal{D}, \pi_\theta)$: replay buffer $\mathcal{D}$ as the positive (data anchor);
- $\mathcal{L}_{\text{top-}K}(\theta) = \mathcal{L}_{\text{drift}}(\theta;\, P_K, \pi_\theta)$: top-K self-generated set $P_K(s)$ as the positive (Q-improvement).

The combined loss $\mathcal{L}(\theta) = \mathcal{L}_{\text{BC}}(\theta) + \lambda \mathcal{L}_{\text{top-}K}(\theta)$ with $\lambda > 0$ anchors π_θ to π_β while pushing it toward higher-Q regions. The shared drift form makes implementation simple: a single drift-loss routine evaluates both terms by swapping the positive set between $\mathcal{D}$ and $P_K(s)$. Algorithm 1 summarizes the full online fine-tuning procedure.

3.2 Why Drifting Models for RL Finetuning?

Drifting policies (DFP) and few-step diffusion or flow policies [66, 45] share the pushforward representation $\pi_\theta(\cdot|s) = [f_\theta(\cdot, s)]_{\#}p_\epsilon$ but parameterize f_θ in structurally different ways. Drifting policies parameterize f_θ *directly* as a single-pass network whose output is the action itself. Diffusion and flow policies parameterize f_θ *indirectly* through a time-indexed velocity field $v_\theta(a(t), t, s)$ along stochastic processes or their ODE counterparts [23, 38]; few-step variants [18, 14, 56] amortize the velocity integration along the trajectory into one or a few network calls but retain the time-indexed velocity field parameterization. We highlight two consequences of this choice for RL finetuning.

Probability space descent vs. velocity-field re-fitting. In the ideal nonparametric view, DFP corresponds to a WGF descent direction on *probability space*: the drifting field transports policy samples toward π^+ along the steepest-descent direction of $\mathrm{KL}(\pi_\theta \,\|\, \pi^+)$ through the pushforward map $f_\theta(\epsilon, s)$. Diffusion and flow policies [62, 11], including one-step variants [66, 54, 13], do not admit such a direct descent: they retain a time-indexed velocity prediction $v_\theta(a(t), t, s)$ from which an action is generated through ODE integration [57, 38, 23, 55], and one-step inference (1 NFE) [56, 14, 18] does not eliminate this ODE dependence at training time. Shifting π_θ toward π^+ therefore requires globally re-fitting v_θ, manifesting as two coupled burdens: a self-consistency constraint along the ODE (e.g., the MeanFlow identity [18]) that couples the velocity field across time in one-step variants, and trajectory-level credit assignment that spreads the Q-improvement signal across the integration path [50, 40, 35]. DFP sidesteps both: a single-pass pushforward map has no ODE structure, so output-level supervision realizes a direct WGF descent direction on probability space.

Built-in repulsion from current samples. The policy improvement loss of Eq. (12) pairs attraction toward Q-improvement targets ($\mathbf{V}^+_{\pi^+}$) with repulsion from the current policy's own samples ($\mathbf{V}^-_{\pi_\theta}$), which typically lie in lower-Q regions than the targets. We conjecture that this built-in repulsion further pushes the policy away from low-Q regions, a structural mechanism not present in diffusion or flow parameterizations.

4 Related Work

Offline-to-Online RL. A key challenge in offline-to-online RL is balancing the need to stay on the offline data support, avoiding out-of-distribution actions, against the exploration needed for online improvement. Existing methods address this intricate balance through (i) conservative critic regularization [31, 30, 43], (ii) behavior-cloning regularization on the policy [42, 59], and (iii) replay strategies mixing offline and online transitions [3, 58]. While crucial for stability, these methods ultimately depend on the underlying policy parameterization to support both behavior cloning during offline pretraining and effective improvement under continually shifting targets during online fine-tuning, motivating our novel generative policy introduced below.

Few-Step Generative Policies. In continuous-control RL, recent work replaces Gaussian policies [37, 16, 20] with generative policies that capture multimodal action distributions [62, 21, 11]. The field has since shifted toward few-step or one-step backbones for inference efficiency [66, 63, 54, 12, 13]. These backbones, however, still generate actions through a time-indexed SDE/ODE trajectory. When applied to RL fine-tuning, they require self-consistent updates across all intermediate timesteps to adapt to a shifting target distribution. We instead adopt a non-ODE policy based on *drifting models* [10] that generates actions in a single forward pass without any time variable, reducing adaptation to shifting only the network's output rather than re-aligning the entire generation trajectory.

Drifting Models. Drifting models [10] are one-step generative models that evolve the pushforward distribution at training time via stop-gradient regression onto a kernel-based mean-shift target, outperforming ODE-based few-step models [18, 56, 14] on image generation. Follow-up analyses relate the drift field to score matching [32, 61] and to the particle velocity of a Wasserstein-2 gradient flow under KDE-smoothed densities [6, 22]. Concurrent works extend drifting models to control: Ada3Drift [65] and KDP [49] only target offline imitation learning, while DBPO [17] targets offline-to-online fine-tuning but projects the drifting policy onto a unimodal Gaussian for

Table 1: Success rate (%) on Robomimic [39] and OGBench [44] tasks under the offline-to-online RL. Each cell reports `mean` ± `std` over 5 seeds. Best result per column in **bold**; second-best underlined.

	Robomimic			Cube-double			Cube-triple			Cube-quadruple-100m			
Method	`lift`	`square`	`can`	`task2`	`task3`	`task4`	`task2`	`task3`	`task4`	`task2`	`task3`	`task4`	**Avg.**
BFN [19]	97.6 ± 2	32.8 ± 8	82.0 ± 2	86.0 ± 5	88.8 ± 5	27.2 ± 8	7.6 ± 9	6.8 ± 3	0.0 ± 0	32.4 ± 21	0.0 ± 0	0.0 ± 0	38.4
QC-BFN [36]	99.6 ± 1	88.4 ± 4	90.6 ± 3	99.8 ± 0	**99.8 ± 0**	92.6 ± 6	87.4 ± 10	80.8 ± 4	33.4 ± 9	95.8 ± 2	63.2 ± 10	74.2 ± 11	83.8
FQL [45]	96.8 ± 2	10.8 ± 7	58.4 ± 8	93.2 ± 8	91.2 ± 5	6.0 ± 6	0.4 ± 1	6.4 ± 8	0.0 ± 0	0.0 ± 0	0.0 ± 0	0.0 ± 0	30.3
QC-FQL [36]	**100.0 ± 0**	72.0 ± 9	**94.4 ± 2**	**100.0 ± 0**	**99.8 ± 0**	**99.8 ± 0**	88.2 ± 2	60.4 ± 12	51.4 ± 24	98.0 ± 2	85.0 ± 7	92.2 ± 7	86.8
MVP [66]	99.8 ± 0	79.4 ± 4	83.6 ± 5	98.4 ± 1	98.6 ± 1	94.8 ± 4	86.2 ± 4	57.2 ± 10	31.0 ± 20	96.6 ± 2	47.2 ± 30	91.2 ± 2	80.3
DFP (Ours)	**100.0 ± 0**	**93.2 ± 2**	90.6 ± 3	**100.0 ± 0**	99.6 ± 1	99.6 ± 1	**98.4 ± 1**	**91.6 ± 2**	**81.2 ± 6**	**99.6 ± 1**	**96.6 ± 2**	**99.0 ± 2**	**95.8**

PPO [53] updates, sacrificing its expressiveness. We are the first to interpret drifting model RL fine-tuning as a Wasserstein-2 gradient flow descent on policy space and derive an algorithm that updates the drifting policy toward high-reward actions within a trust region, with a tractable surrogate of the otherwise intractable policy improvement loss.

5 Experiments

Additional details and results are provided in the appendix: offline RL experiments (Appendix B.2), full offline-to-online results (Appendix B.1), implementation details and hyperparameters (Appendix A.3), and training and inference cost analysis (Appendix B.6).

5.1 Experimental Setup

Benchmarks. We follow the experimental setup of Mean Velocity Policy (MVP) [66], evaluating on 3 tasks from Robomimic [39] (`Lift`, `Can`, `Square`) under the multi-human (MH) datasets and 6 tasks from OGBench [44], with three tasks each from `cube-double` and `cube-triple` (`task-2/3/4` per environment). We further include three tasks from the `cube-quadruple` environment (`cube-quadruple-task-2/3/4`), resulting in a total of 12 tasks.

Baselines against DFP. We compare against the same baselines as MVP [66], spanning three categories: (i) *multi-step inference*: **BFN** [19] and **QC-BFN** [36] train a multi-step BC flow policy and perform Best-of-N extraction at inference (QC-BFN adds action-chunking on top of BFN); (ii) *distilled one-step*: **FQL** [45] and **QC-FQL** [36] distill a one-step policy from a multi-step BC flow policy under a Q-maximization objective via the critic Jacobian $\nabla_a Q_\phi$ (QC-FQL adds action-chunking on top of FQL); (iii) *teacher-free one-step*: **MVP** [66] trains a one-step MeanFlow [18] policy from scratch, jointly performing BC and policy improvement (the latter by imitating the best-of-N action) on the same network. MVP is the closest baseline to DFP: both are teacher-free one-step policies with action-chunking, differing only in the generative policy parameterization (MeanFlow [18] vs. drifting models [10]) and the corresponding training objectives, which we isolate in Sec. 5.4.

Hyperparameters. We use $N = 16$ candidate actions, with $K = 2$ on Robomimic [39] and $K = 4$ on OGBench [44], and the top-K loss coefficient $\lambda = 0.5$ by default unless otherwise stated. Full hyperparameter configurations are listed in Appendix A.3.

5.2 Comparisons to Baselines

As summarized in Tab. 1, DFP achieves the highest average success rate of 95.8%, ranking first on 9 of 12 tasks and second-best on the remaining 3, outperforming the strongest baseline QC-FQL [36] (86.8%) by $+9.0$ pp. Despite generating actions in a single forward pass, DFP surpasses even the best multi-step policy QC-BFN [36] (83.8%, $+12.0$ pp), with the gap widening on the harder `cube-triple` and `cube-quadruple` splits where multi-step BC alone is insufficient.

The most informative comparison is against MVP [66], the closest baseline: both methods train a single one-step policy with no multi-step teacher. DFP improves the average score by $+15.5$ pp

over MVP [66] and outperforms on every task, with particularly large gains on the multimodal long-horizon tasks (e.g., `cube-triple-task4`: $31.0 \to 81.2$; `cube-quadruple-task3`: $47.2 \to 96.6$). As shown by the training curves in Fig. 1, DFP outperforms MVP [66] consistently across time, demonstrating both faster convergence in the online stage and better behavior cloning capability in the offline stage. The full result table, including both offline and online phases, is in Appendix B.1.

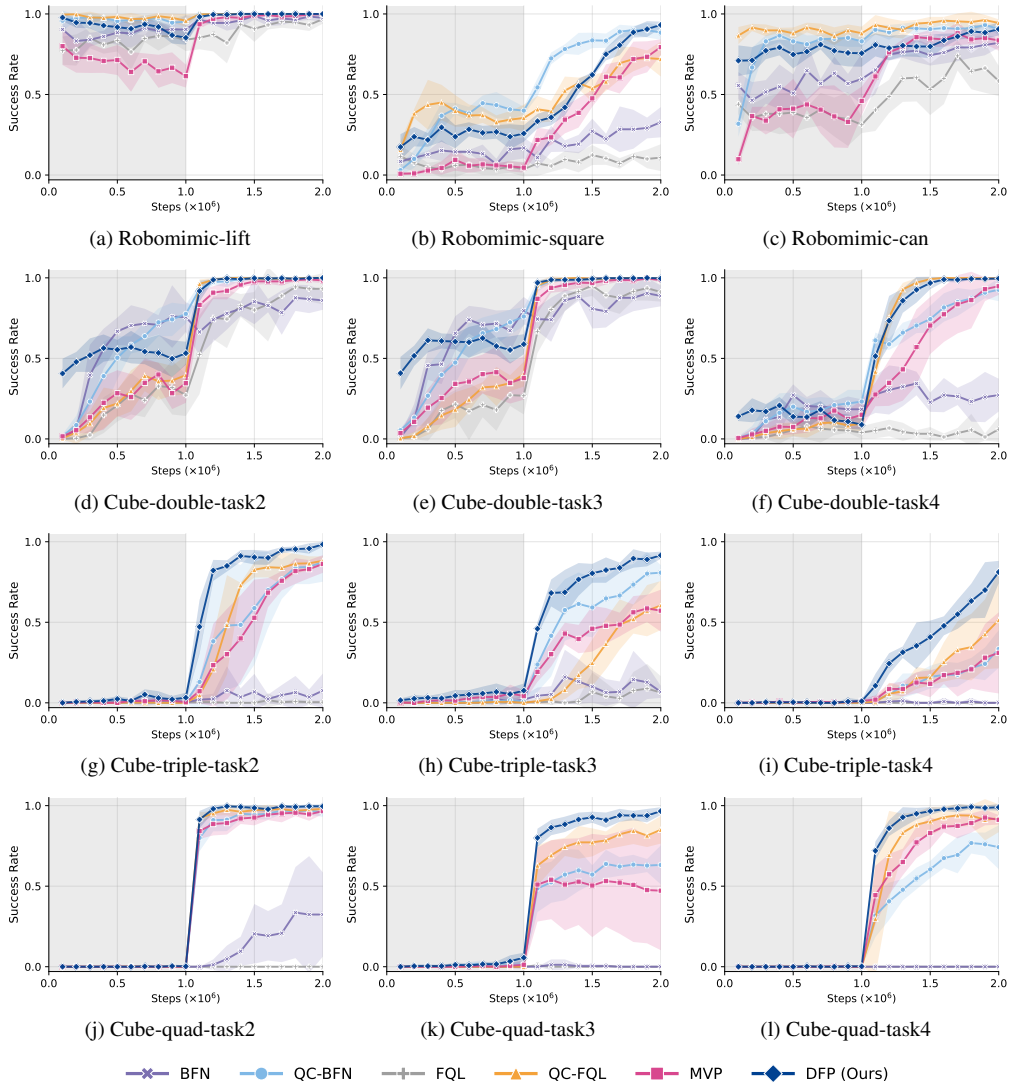


Figure 1: **Success rate over training steps across Robomimic [39] and OGBench [44].** Solid lines and shaded regions show the mean and 95% confidence interval over five runs. Gray and white backgrounds indicate the offline and online phases, respectively.

5.3 Analysis: Drifting vs. MeanFlow under Identical Loss

To isolate the contribution of each design choice that distinguishes DFP from MVP [66], we ablate two factors along two axes: the policy backbone (MeanFlow [18] vs. drifting model [10]) and the training objective (BC only vs. BC + $\mathcal{L}_{\text{top-}K}$). For the MeanFlow backbone, applying $\mathcal{L}_{\text{top-}K}$ reduces to applying its native MeanFlow identity loss to the top-K candidates. The results are summarized in Tab. 2. Even without $\mathcal{L}_{\text{top-}K}$ (BC only), DFP yields a $+8.1$ pp in average success rate over MVP, confirming the drifting parameterization's stronger behavior cloning capability. Adding $\mathcal{L}_{\text{top-}K}$ benefits the two backbones asymmetrically: MeanFlow gains only marginally ($80.3\% \to 82.7\%$, $+2.4$ pp), while the drifting backbone gains substantially ($88.4\% \to 95.8\%$, $+7.4$ pp).

This asymmetry follows from the parameterization-level distinction discussed in Sec. 3.1. On the drifting backbone, the top-K supervision is applied directly to generated actions through the push-forward map f_θ in a single forward pass, approximating a WGF descent direction toward the high-Q candidates. On MeanFlow, the same K targets must instead enter through its identity loss [18] and re-fit v_θ globally, where the self-consistency constraint couples v_θ across time and trajectory-level credit assignment spreads the supervision along the integration path; the same targets therefore yield a smaller shift in π_θ. See Appendix B.3 for training curve comparisons.

5.4 Ablation Study

Refer to Sec. B.4 and Sec. B.5 in the appendix for more comprehensive ablation results.

Table 2: Ablation on the $\mathcal{L}_{\text{top-}K}$ for MVP [66] and DFP. Success rate (%) on Robomimic and OGBench tasks under the offline-to-online protocol. Each cell reports `mean` $\pm$ `std` over 5 seeds. Best result per column in **bold**; second-best underlined.

			Robomimic			Cube-double			Cube-triple			Cube-quadruple-100m			
Method	**Backbone**	$\mathcal{L}_{\text{top-}K}$	`lift`	`square`	`can`	`task2`	`task3`	`task4`	`task2`	`task3`	`task4`	`task2`	`task3`	`task4`	**Avg.**
MVP [66]	**MeanFlow**	✗	99.8 ± 0	79.4 ± 4	83.6 ± 5	98.4 ± 1	98.6 ± 1	94.8 ± 4	86.2 ± 4	57.2 ± 10	31.0 ± 20	96.6 ± 2	47.2 ± 30	91.2 ± 2	80.3
MVP w/ $\mathcal{L}_{\text{top-}K}$	**MeanFlow**	✓	**100.0 ± 0**	81.6 ± 5	86.2 ± 6	99.6 ± 1	98.8 ± 1	96.6 ± 1	78.0 ± 15	60.6 ± 13	30.4 ± 11	97.2 ± 1	69.2 ± 15	94.2 ± 4	82.7
DFP w/o $\mathcal{L}_{\text{top-}K}$	**Drifting**	✗	**100.0 ± 0**	88.6 ± 2	90.4 ± 5	99.2 ± 1	**99.6 ± 1**	96.0 ± 3	91.4 ± 3	83.2 ± 4	31.2 ± 7	97.6 ± 1	88.8 ± 3	95.2 ± 3	88.4
DFP (Ours)	**Drifting**	✓	**100.0 ± 0**	**93.2 ± 2**	**90.6 ± 3**	**100.0 ± 0**	**99.6 ± 1**	**99.6 ± 1**	**98.4 ± 1**	**91.6 ± 2**	**81.2 ± 6**	**99.6 ± 1**	**96.6 ± 2**	**99.0 ± 2**	**95.8**

Table 3: Top-K loss weight λ ablation on cube-quadruple tasks, `mean` and `std` over 5 seeds. **Bold**/underline: best/second-best.

Cube-4.	**task 2**	**task 3**	**task 4**
$\lambda = 0.1$	99.0 ± 1	86.2 ± 8	96.4 ± 2
$\lambda = 0.5$	**99.6 ± 1**	**96.6 ± 2**	99.0 ± 2
$\lambda = 1.0$	98.4 ± 1	**96.6 ± 2**	**99.2 ± 0**
$\lambda = 5.0$	99.2 ± 0	94.2 ± 4	98.0 ± 2

Table 4: K ablation. Success rate (%) averaged over tasks per environment and 5 seeds. **Bold**/underline: best/second-best.

	Robo.	**Cube-2.**	**Cube-3.**	**Cube-4.**	**Avg.**
$K = 1$	95.0	99.7	54.5	94.2	85.8
$K = 2$	**94.6**	**100.0**	76.2	96.8	91.9
$K = 4$	93.9	99.7	**90.4**	**98.4**	**95.6**
$K = 8$	88.6	99.8	85.5	96.2	92.5

Effect of the top-K loss weight λ. We perform an ablation study on the $\mathcal{L}_{\text{top-}K}$ weight λ. As reported in Tab. 3, we vary $\lambda \in \{0.1, 0.5, 1.0, 5.0\}$ and observe that DFP is robust to the weight.

Effect of the number of positives K. With the candidate pool size fixed at $N = 16$, we vary $K \in \{1, 2, 4, 8\}$ in $\mathcal{L}_{\text{top-}K}$, reported in Tab. 4. While a sweet spot exists, $\mathcal{L}_{\text{top-}K}$ is overall robust to K: even the worst setting ($K = 1$, 85.8%) is already comparable to the best baseline (QC-FQL, 86.8%).

6 Conclusion

We present *Drifting Field Policy* (DFP), a novel one-step generative policy grounded in a Wasserstein-2 gradient flow interpretation of drifting model training. The resulting top-K drift loss updates the policy toward high-reward actions within a trust region around the previous policy, with a tractable surrogate of the policy improvement objective. On Robomimic [39] and OGBench [44] benchmarks, DFP achieves 95.8% average success rate, outperforming all baselines including multi-step variants. Ablations attribute the gain to both the drifting parameterization and the top-K supervision, whose synergy is unique to the drifting backbone.

Limitations. DFP is currently studied on simulated continuous-action manipulation tasks; extensions to high-dimensional observations and sim-to-real deployment remain future work. Performance of the proposed top-K loss depends on the quality of the learned critic, a limitation shared with actor-critic methods broadly. Finally, while we provide structural analyses and supporting ablations for the advantages of our non-ODE parameterization over ODE-based policies, a deeper theoretical understanding of these distinctions remains an open question.

References

[1] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. In *ICLR*, 2018.

[2] L. Ambrosio, N. Gigli, and G. Savaré. *Gradient Flows: In Metric Spaces and in the Space of Probability Measures*. Springer, 2005.

[3] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data. In *ICML*, 2023.

[4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control. 2025.

[5] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine. Training diffusion models with reinforcement learning. In *ICLR*, volume 2024, 2024.

[6] J. Cao, Z. Wei, and Y. Liu. Gradient flow drifting: Generative modeling via wasserstein gradient flows of kde-approximated divergences. *arXiv preprint arXiv:2603.10592*, 2026.

[7] H. Chen, C. Lu, Z. Wang, H. Su, and J. Zhu. Score regularized policy optimization through diffusion behavior. In *ICLR*, 2024.

[8] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[9] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. 2023.

[10] M. Deng, H. Li, T. Li, Y. Du, and K. He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

[11] S. Ding, K. Hu, Z. Zhang, K. Ren, W. Zhang, J. Yu, J. Wang, and Y. Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *NeurIPS*, 2024.

[12] Z. Ding and C. Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *ICLR*, 2024.

[13] N. Espinosa-Dice, Y. Zhang, Y. Chen, B. Guo, O. Oertell, G. Swamy, K. Brantley, and W. Sun. Scaling offline rl via efficient and expressive shortcut models. *arXiv preprint arXiv:2505.22866*, 2025.

[14] K. Frans, D. Hafner, S. Levine, and P. Abbeel. One step diffusion via shortcut models. In *ICLR*, 2025.

[15] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. In *NeurIPS*, 2021.

[16] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *ICML*, 2018.

[17] Y. Gao, Y. Shen, S. Zhang, W. Yu, Y. Duan, J. Wu, J. Deng, Y. Zhang, et al. Drift-based policy optimization: Native one-step policy learning for online robot control. *arXiv preprint arXiv:2604.03540*, 2026.

[18] Z. Geng, M. Deng, X. Bai, J. Z. Kolter, and K. He. Mean flows for one-step generative modeling. In *NeurIPS*, 2025.

[19] S. K. S. Ghasemipour, D. Schuurmans, and S. S. Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *ICML*, 2021.

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

[21] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[22] P. He, O. Khangaonkar, H. Pirsiavash, Y. Bai, and S. Kolouri. Sinkhorn-drifting generative models. *arXiv preprint arXiv:2603.12366*, 2026.

[23] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[24] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, 2022.

[25] R. Jordan, D. Kinderlehrer, and F. Otto. The variational formulation of the Fokker–Planck equation. *SIAM Journal on Mathematical Analysis*, 1998.

[26] H. J. Kappen. Linear theory for control of nonlinear stochastic systems. *Physical review letters*, 2005.

[27] D. Kim, C.-H. Lai, W.-H. Liao, N. Murata, Y. Takida, T. Uesaka, Y. He, Y. Mitsufuji, and S. Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *ICLR*, 2024.

[28] J. Kim, T. Yoon, J. Hwang, and M. Sung. Inference-time scaling for flow models via stochastic generation and rollover budget forcing. In *NeurIPS*, 2026.

[29] V. Konda and J. Tsitsiklis. Actor-critic algorithms. In *NeurIPS*, 1999.

[30] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

[31] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. In *NeurIPS*, 2020.

[32] C.-H. Lai, B. Nguyen, N. Murata, Y. Takida, T. Uesaka, Y. Mitsufuji, S. Ermon, and M. Tao. A unified view of drifting and score-based models. *arXiv preprint arXiv:2603.07514*, 2026.

[33] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[34] S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *NeurIPS*, 2014.

[35] Q. Li and S. Levine. Q-learning with adjoint matching. In *ICLR*, 2026.

[36] Q. Li, Z. Zhou, and S. Levine. Reinforcement learning with action chunking. In *NeurIPS*, 2025.

[37] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

[38] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *ICLR*, 2023.

[39] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[40] D. McAllister, S. Ge, B. Yi, C. M. Kim, E. Weber, H. Choi, H. Feng, and A. Kanazawa. Flow matching policy gradients. In *ICLR*, 2026.

[41] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 2015.

[42] A. Nair, A. Gupta, M. Dalal, and S. Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[43] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. In *NeurIPS*, 2023.

[44] S. Park, K. Frans, B. Eysenbach, and S. Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *ICLR*, 2025.

[45] S. Park, Q. Li, and S. Levine. Flow q-learning. In *ICML*, 2025.

[46] X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[47] J. Peters, K. Mulling, and Y. Altun. Relative entropy policy search. In *AAAI*, 2010.

[48] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. 2024.

[49] G. Puthumanaillam and M. Ornik. Amortizing trajectory diffusion with keyed drift fields. *arXiv preprint arXiv:2603.14056*, 2026.

[50] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz. Diffusion policy policy optimization. In *ICLR*, 2025.

[51] F. Santambrogio. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Progress in Nonlinear Differential Equations and Their Applications. Birkhäuser, 2015.

[52] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, 2015.

[53] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[54] J. Sheng, Z. Wang, P. Li, and M. Liu. Mp1: Meanflow tames policy learning in 1-step for robotic manipulation. In *AAAI*, 2026.

[55] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

[56] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. In *ICML*, 2023.

[57] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

[58] Y. Song, Y. Zhou, A. Sekhari, J. A. Bagnell, A. Krishnamurthy, and W. Sun. Hybrid rl: Using both offline and online data can make rl efficient. In *ICLR*, 2023.

[59] D. Tarasov, V. Kurenkov, A. Nikulin, and S. Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *NeurIPS*, 2023.

[60] E. Todorov. Linearly-solvable markov decision problems. In *NeurIPS*, 2006.

[61] E. Turan and M. Ovsjanikov. Generative drifting is secretly score matching: a spectral and variational perspective. *arXiv preprint arXiv:2603.09936*, 2026.

[62] Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *ICLR*, 2023.

[63] Z. Wang, D. Li, Y. Chen, Y. Shi, L. Bai, T. Yu, and Y. Fu. One-step generative policies with q-learning: A reformulation of meanflow. In *AAAI*, 2026.

[64] Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[65] C. Xu, Y. Zou, Z. Feng, F. Meng, and S. Liu. Ada3drift: Adaptive training-time drifting for one-step 3d visuomotor robotic manipulation. *arXiv preprint arXiv:2603.11984*, 2026.

[66] G. Zhan, L. Tao, P. Wang, Y. Wang, Y. Li, Y. Chen, H. Li, M. Tomizuka, and S. E. Li. Mean flow policy with instantaneous velocity constraint for one-step action generation. In *ICLR*, 2026.

A Experimental Details

A.1 Environment Descriptions

We evaluate on 12 manipulation tasks drawn from the Robomimic benchmark [39] and the OGBench manipulation suite [44]. Robomimic uses a 7-DoF Franka Emika Panda arm, while the OGBench cube environments use a 6-DoF UR5e arm with a Robotiq 2F-85 parallel-jaw gripper. All tasks employ sparse, completion-style rewards. Fig. 2 visualizes all 12 tasks.

A.1.1 Robomimic (Multi-Human)

Datasets. We use the Multi-Human (MH) variant of each Robomimic task, collected via teleoperation through the RoboTurk platform by six operators stratified by skill (two "worse", two "okay", and two "better"). Each operator contributes 50 successful trajectories, yielding 300 heterogeneous demonstrations per task. The intentionally non-uniform action distribution makes MH a harder offline-learning setting than the cleaner Proficient-Human variants.

Tasks. Every task is solved by a single 7-DoF Franka Emika Panda arm in a tabletop workspace and terminates upon success, with a sparse reward.

- **Lift** (Fig. 2a) – Grasp a randomly placed cube and raise it above a height threshold. Tests basic grasping.
- **Can** (Fig. 2b) – Transport a coke can from one bin to a smaller target bin, requiring coordinated reach, grasp, and release.
- **Square** (Fig. 2c) – Pick up a square nut and thread it onto a peg with sub-centimeter tolerance. The most precision-sensitive task in the suite.

A.1.2 OGBench Cube Environments

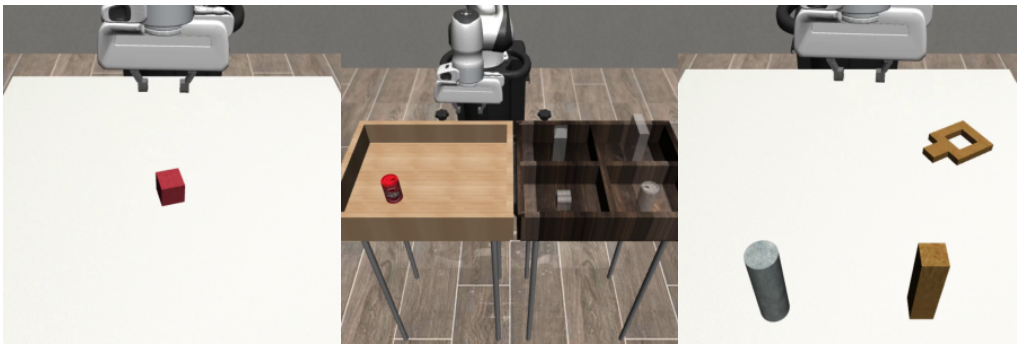
Common setup. The `cube-double`, `cube-triple`, and `cube-quadruple` environments share an identical 6-DoF UR5e arm with a Robotiq 2F-85 gripper but vary the number of cubes $N \in \{2, 3, 4\}$. We use the `play-singletask-task[N]-v0` variants, which fix an evaluation goal and relabel the unstructured `play` dataset with the corresponding reward function for offline pre-training. Among the five canonical evaluation goals (`task1`–`task5`) provided by each environment, we focus on `task2`–`task4`, which together cover representative skills including multi-object pick-and-place, structural manipulation, and combinatorial rearrangement. The reward is semi-sparse, defined as $r = -n_{\text{wrong}}$, where n_{wrong} counts cubes whose position has not yet reached its target within the environment's success tolerance; an episode terminates only when all N cubes simultaneously satisfy the goal criterion. Following OGBench's convention, success is determined by cube positions only, ignoring orientation.

cube-double (Fig. 2d–f) covers three multi-object skills corresponding to the OGBench-defined goals *double-pnp1* (`task2`), *double-pnp2* (`task3`), and *swap* (`task4`), where the last requires exchanging the positions of two occupied cubes.

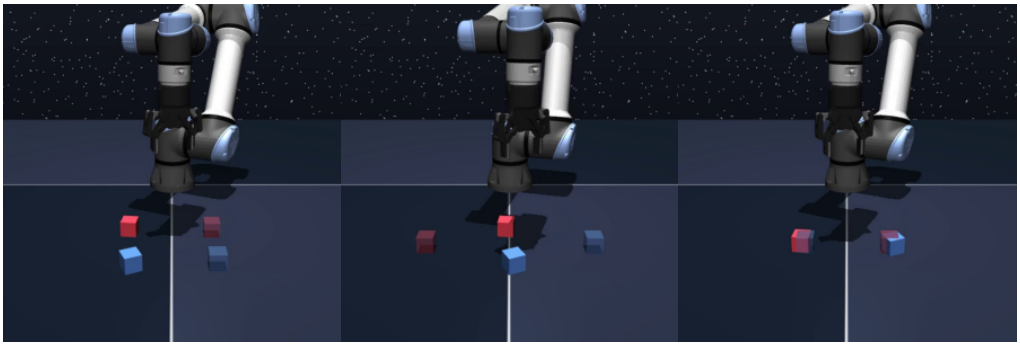
cube-triple (Fig. 2g–i) requires composing pick-and-place primitives in non-trivial sequences: *triple-pnp* rearrangement of three cubes (`task2`), *pnp-from-stack* which involves manipulating cubes from a stacked configuration (`task3`), and *cyclic permutation* of three cubes (`task4`).

cube-quadruple (Fig. 2j–l) amplifies long-horizon coordination, requiring up to four sequential pick-and-place subtasks per goal (Table 1 of [44]): *quadruple-pnp* rearrangement of four cubes (`task2`), *pnp-from-square* which manipulates cubes from a square configuration (`task3`), and a *4-cycle* permutation (`task4`) that cannot be decomposed into fewer than three pairwise swaps.

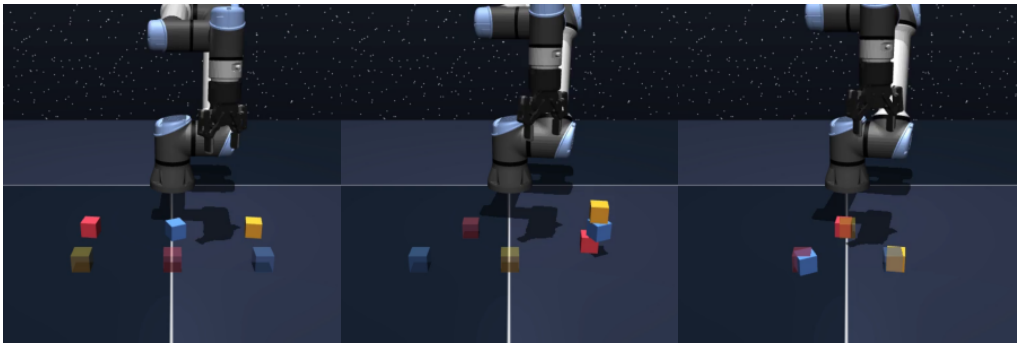
The progression from `cube-double` to `cube-quadruple` is intentionally combinatorial: the same primitives (pick-and-place, swap, cycle) recur, but the number of required subtasks scales with N (up to 1, 2, 4 atomic behaviors for $N = 2, 3, 4$ respectively, per OGBench Table 1), with the longest evaluation task requiring approximately 400 environment steps. This provides a controlled benchmark for long-horizon sequential reasoning and credit assignment.



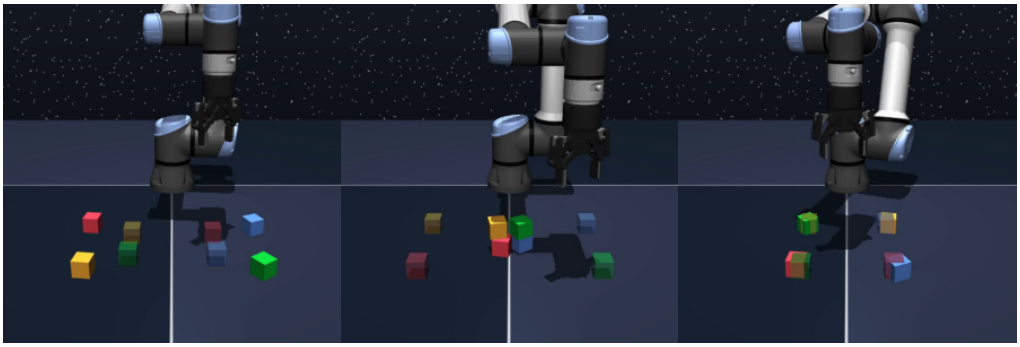
(a) Lift (b) Can (c) Square



(d) Cube-double-task2 (e) Cube-double-task3 (f) Cube-double-task4



(g) Cube-triple-task2 (h) Cube-triple-task3 (i) Cube-triple-task4



(j) Cube-quadruple-task2 (k) Cube-quadruple-task3 (l) Cube-quadruple-task4

Figure 2: **Visualization of the 12 manipulation tasks used in our experiments.** The top row shows the Robomimic Multi-Human tasks (Lift, Can, Square), and the remaining rows show the OGBench Cube environments with $N \in \{2, 3, 4\}$ cubes. Each panel depicts the initial configuration of a representative episode.

A.2 Baseline Method Details

We compare DFP against five recent generative model based policies for offline-to-online RL. All baselines share the off-policy actor-critic skeleton of Eqs. (1)-(2) but differ in the policy parameterization and how the actor is supervised. We summarize each below; full hyperparameters are listed in Appendix A.3.

FQL [45]. Flow Q-Learning trains a multi-step flow-matching behavior policy $\mu_\theta(s, z)$ with the standard flow-matching objective and jointly distills it into a one-step student $\mu_\omega(s, z)$ that is optimized to maximize Q_ϕ under a distillation regularizer to μ_θ. The one-step student is used at inference, eliminating iterative integration.

BFN [19]. Best-of-N is a critic-guided action-selection wrapper around a behavior-cloned multi-step flow policy: at every action call, N candidates are drawn from the BC policy and the one with the highest $Q_\phi(s, a^{(j)})$ is executed. We follow the formulation of EMaQ [19], which derives this scheme from the expected-max Bellman backup and uses the BC policy as the proposal. BFN exploits the critic only at execution time; the policy itself is never updated by Q_ϕ.

QC-BFN [36]. Q-chunking [36] applied to BFN. The actor predicts a temporally extended sequence of H future actions rather than a single action, and RL is run directly in the chunked action space with an unbiased H-step Bellman backup that mitigates exploration in long-horizon sparse-reward tasks. The BFN best-of-N wrapper is then applied on chunks at execution.

QC-FQL [36]. Q-chunking [36] applied to FQL. The chunked h-step Bellman backup of QC-BFN is retained, but the actor is the one-step FQL student rather than the multi-step BC flow, paying only one-step inference cost while inheriting the chunked exploration benefit.

MVP [66]. Mean Velocity Policy parameterizes the actor as a MeanFlow [18] velocity field $v_\theta(a^{(t)}, t, r, s)$ that models the *mean* velocity over an interval $[r, t]$, enabling a single Euler step from noise to action at inference. MVP introduces an instantaneous velocity constraint (IVC) at the interval boundary as an auxiliary regression loss to disambiguate the otherwise under-determined ODE. Online supervision relies on best-of-N environment rollouts: N actions are sampled from the current actor at every state, the highest-Q candidate is executed and stored, and the velocity field is regressed onto these critic-selected actions via standard MeanFlow self-consistency. MVP is the most direct empirical comparison for DFP since it shares the one-step regime and the best-of-N online recipe but uses an ODE-based backbone and single-positive supervision.

MVP w/ $\mathcal{L}_{\text{top-}K}$. A controlled ablation we introduce to isolate the effect of top-K supervision from the choice of backbone. The actor parameterization, IVC loss, and best-of-N rollout are inherited from MVP; the only modification is to extend the online actor supervision from a single critic-selected target to the same top-K multi-positive set $\mathcal{P}_K(s)$ used in DFP, regressing the MeanFlow velocity field onto each of the K positives. This isolates the question: does top-K supervision transfer to an ODE-based one-step backbone, or does the gain require the drifting parameterization? Sec. 5 reports that the gain is largely backbone-specific, supporting the latter.

A.3 Hyperparameters

Tab. 5 lists the hyperparameters used by DFP. The shared block (top) covers the actor-critic backbone, and the DFP-specific block (bottom) lists the additional hyperparameters introduced for the drifting field and the top-K surrogate loss. For all baselines, we follow the configurations reported in their original papers [45, 19, 36, 66].

B Additional Results

B.1 Full Offline-to-Online Results

Tab. 6 reports the full offline→online progression of all methods, complementing the online-only summary in Tab. 1. The expanded format makes the online-phase contribution of each method

Table 5: **Detailed hyperparameters.**

Parameter	Value
Shared	
Batch size	256
Discount factor (γ)	0.99
Optimizer	Adam
Learning rate	3×10^{-4}
Target network update rate (τ)	5×10^{-3}
Number of offline training steps	1×10^{6} (1M)
Number of online training steps	1×10^{6} (1M)
Total gradient steps	2×10^{6} (2M)
Policy network width	512
Policy network depth	4 hidden layers
Policy activation function	GELU
Policy layer normalization	False
Value network width	512
Value network depth	4 hidden layers
Value activation function	GELU
Value layer normalization	True
Value ensemble size	2
Value ensemble operator	MEAN
Chunking horizon	5 (1 for *FQL, BFN*)
DFP (Ours)	
Number of candidates for top-K (N)	16
Number of generated actions (N_{gen})	8
Drift weight (λ)	0.5
Actor EMA smoothing (τ_{EMA})	1×10^{-4}
Number of best-of-N' (N')	16^{*} / $4^{\dagger}$
Top-K positive set size (K)	4^{*} / $2^{\dagger}$
Kernel bandwidth (h)	$\{0.05\}^{*}$ / $\{0.005, 0.05\}^{\dagger}$

* OGBench (`cube`) / † Robomimic (`lift`, `can`, `square`).

explicit, isolating how effectively each parameterization absorbs buffer updates during fine-tuning. Consistent with the analysis in Sec. 5, **DFP** and **DFP w/o** $\mathcal{L}_{\textbf{top-}K}$ show the largest online gains on the long-horizon cube-triple and cube-quadruple splits, where the drifting backbone's output-level supervision and the top-K drift loss provide the most leverage.

B.2 Adaptation to Offline RL

While our main experiments in Sec. 5 follow the offline-to-online RL setup of MVP [66], DFP's algorithm is not tied to this setting. To probe the generality of our method, we evaluate DFP in a pure offline RL setting, where the algorithm is identical to Algorithm 1 but without the online environment interaction. The combined loss $\mathcal{L}_{\text{BC}}(\theta) + \lambda\, \mathcal{L}_{\text{top-}K}(\theta)$ is applied throughout training.

Offline RL Benchmarks. Following prior offline RL work [45], we evaluate on the 5 robot-manipulation environments from OGBench [44], and compare against the baselines reported therein. For each environment, we run the default tasks and report success rates averaged over 8 seeds. Baseline numbers are taken from prior work [45].

Offline RL Baselines. We compare against three categories of baselines: (i) *Gaussian policy*: Behavior Cloning (BC), Implicit Q-Learning (IQL) [30], and ReBRAC [59]; (ii) *diffusion-based policy*: Implicit Diffusion Q-Learning (IDQL) [21], SRPO [7], and CAC [12]; (iii) *flow-based policy*: FAWAC and FBRAC, the flow-based variants of AWAC [42] and BRAC [64] respectively, and Flow Q-Learning with its iterative form (FQL, IFQL) [45].

Table 6: **Offline-to-online RL full results.** Each cell shows offline → online (mean with std as subscript). Best result per column in **bold**; second-best underlined.

		Baselines						Ours	
Benchmark	**Task**	BFN [19]	QC-BFN [36]	FQL [45]	QC-FQL [36]	MVP [66]	MVP w/ $\mathcal{L}_{\text{top-}K}$	**DFP w/o $\mathcal{L}_{\text{top-}K}$**	**DFP (Ours)**
Robomimic	lift	$90.4_{\pm 7.1} \rightarrow 97.6_{\pm 2.2}$	$95.2_{\pm 3.4} \rightarrow 99.6_{\pm 0.5}$	$84.0_{\pm 5.7} \rightarrow 96.8_{\pm 2.3}$	$\mathbf{95.8}_{\pm 3.3} \rightarrow \mathbf{100.0}_{\pm 0.0}$	$61.4_{\pm 5.7} \rightarrow 99.8_{\pm 0.4}$	$61.4_{\pm 5.7} \rightarrow 100.0_{\pm 0.0}$	$85.2_{\pm 3.4} \rightarrow \mathbf{100.0}_{\pm 0.0}$	$85.2_{\pm 3.4} \rightarrow \mathbf{100.0}_{\pm 0.0}$
	square	$16.8_{\pm 5.2} \rightarrow 32.8_{\pm 7.6}$	$40.0_{\pm 2.0} \rightarrow 88.4_{\pm 3.6}$	$3.6_{\pm 3.6} \rightarrow 10.8_{\pm 6.7}$	$35.4_{\pm 6.7} \rightarrow 72.0_{\pm 8.7}$	$4.6_{\pm 2.5} \rightarrow 79.4_{\pm 3.9}$	$4.6_{\pm 2.5} \rightarrow 81.6_{\pm 5.0}$	$\mathbf{77.6}_{\pm 6.9} \rightarrow 88.6_{\pm 1.5}$	$\mathbf{77.6}_{\pm 6.9} \rightarrow \mathbf{93.2}_{\pm 1.6}$
	can	$59.6_{\pm 12.8} \rightarrow 82.0_{\pm 2.4}$	$83.0_{\pm 2.7} \rightarrow 90.6_{\pm 3.2}$	$31.2_{\pm 4.1} \rightarrow 58.4_{\pm 7.5}$	$\mathbf{88.0}_{\pm 3.3} \rightarrow \mathbf{94.4}_{\pm 1.8}$	$46.0_{\pm 7.1} \rightarrow 83.6_{\pm 5.2}$	$46.0_{\pm 7.1} \rightarrow 86.2_{\pm 5.6}$	$25.8_{\pm 4.1} \rightarrow 90.4_{\pm 4.5}$	$25.8_{\pm 4.1} \rightarrow 90.6_{\pm 3.2}$
Cube-double	task2	$75.6_{\pm 7.4} \rightarrow 86.0_{\pm 4.7}$	$\mathbf{77.6}_{\pm 4.7} \rightarrow 99.8_{\pm 0.4}$	$27.2_{\pm 10.4} \rightarrow 93.2_{\pm 7.8}$	$39.6_{\pm 9.7} \rightarrow \mathbf{100.0}_{\pm 0.0}$	$34.6_{\pm 9.8} \rightarrow 98.4_{\pm 1.3}$	$34.6_{\pm 9.8} \rightarrow 99.6_{\pm 0.5}$	$53.2_{\pm 5.8} \rightarrow 99.2_{\pm 0.8}$	$53.2_{\pm 5.8} \rightarrow \mathbf{100.0}_{\pm 0.0}$
	task3	$\mathbf{79.6}_{\pm 6.8} \rightarrow 88.8_{\pm 5.2}$	$76.0_{\pm 4.7} \rightarrow \mathbf{99.8}_{\pm 0.4}$	$26.8_{\pm 9.3} \rightarrow 91.2_{\pm 4.8}$	$40.2_{\pm 6.2} \rightarrow \mathbf{99.8}_{\pm 0.4}$	$37.8_{\pm 10.8} \rightarrow 98.6_{\pm 1.1}$	$37.8_{\pm 10.8} \rightarrow 98.8_{\pm 0.8}$	$58.8_{\pm 5.0} \rightarrow 99.6_{\pm 0.9}$	$58.8_{\pm 5.0} \rightarrow 99.6_{\pm 0.5}$
	task4	$18.4_{\pm 5.0} \rightarrow 27.2_{\pm 8.2}$	$\mathbf{23.2}_{\pm 5.8} \rightarrow 92.6_{\pm 5.7}$	$4.0_{\pm 3.2} \rightarrow 6.0_{\pm 6.3}$	$9.4_{\pm 3.6} \rightarrow \mathbf{99.8}_{\pm 0.4}$	$15.0_{\pm 7.0} \rightarrow 94.8_{\pm 4.3}$	$15.0_{\pm 7.0} \rightarrow 96.6_{\pm 0.5}$	$8.8_{\pm 2.6} \rightarrow 96.0_{\pm 2.8}$	$8.8_{\pm 2.6} \rightarrow 99.6_{\pm 0.5}$
Cube-triple	task2	$0.4_{\pm 0.9} \rightarrow 7.6_{\pm 8.5}$	$0.6_{\pm 0.9} \rightarrow 87.4_{\pm 9.8}$	$0.4_{\pm 0.9} \rightarrow 0.4_{\pm 0.9}$	$0.0_{\pm 0.0} \rightarrow 88.2_{\pm 2.2}$	$0.4_{\pm 0.5} \rightarrow 86.2_{\pm 4.4}$	$0.4_{\pm 0.5} \rightarrow 78.0_{\pm 15.4}$	$\mathbf{3.2}_{\pm 2.5} \rightarrow 91.4_{\pm 3.4}$	$\mathbf{3.2}_{\pm 2.5} \rightarrow \mathbf{98.4}_{\pm 1.1}$
	task3	$2.0_{\pm 2.0} \rightarrow 6.8_{\pm 3.0}$	$0.8_{\pm 0.8} \rightarrow 80.8_{\pm 3.8}$	$0.4_{\pm 0.9} \rightarrow 6.4_{\pm 8.2}$	$0.0_{\pm 0.0} \rightarrow 60.4_{\pm 12.3}$	$4.2_{\pm 2.9} \rightarrow 57.2_{\pm 10.5}$	$4.2_{\pm 2.9} \rightarrow 60.6_{\pm 12.6}$	$\mathbf{7.6}_{\pm 3.5} \rightarrow 83.2_{\pm 4.1}$	$\mathbf{7.6}_{\pm 3.5} \rightarrow \mathbf{91.6}_{\pm 1.8}$
	task4	$0.0_{\pm 0.0} \rightarrow 0.0_{\pm 0.0}$	$0.0_{\pm 0.0} \rightarrow 33.4_{\pm 9.4}$	$0.0_{\pm 0.0} \rightarrow 0.0_{\pm 0.0}$	$0.0_{\pm 0.0} \rightarrow 51.4_{\pm 24.2}$	$\mathbf{1.2}_{\pm 0.8} \rightarrow 31.0_{\pm 20.3}$	$\mathbf{1.2}_{\pm 0.8} \rightarrow 30.4_{\pm 10.9}$	$1.0_{\pm 0.7} \rightarrow 31.2_{\pm 6.5}$	$1.0_{\pm 0.7} \rightarrow \mathbf{81.2}_{\pm 5.6}$
Cube-quad	task2	$0.0_{\pm 0.0} \rightarrow 32.4_{\pm 21.0}$	$0.0_{\pm 0.0} \rightarrow 95.8_{\pm 2.3}$	$0.0_{\pm 0.0} \rightarrow 0.0_{\pm 0.0}$	$0.0_{\pm 0.0} \rightarrow 98.0_{\pm 2.0}$	$0.0_{\pm 0.0} \rightarrow 96.6_{\pm 1.5}$	$0.0_{\pm 0.0} \rightarrow 97.2_{\pm 1.5}$	$\mathbf{0.2}_{\pm 0.4} \rightarrow 97.6_{\pm 1.3}$	$\mathbf{0.2}_{\pm 0.4} \rightarrow \mathbf{99.6}_{\pm 0.9}$
	task3	$0.0_{\pm 0.0} \rightarrow 0.0_{\pm 0.0}$	$1.6_{\pm 1.8} \rightarrow 63.2_{\pm 9.7}$	$0.0_{\pm 0.0} \rightarrow 0.0_{\pm 0.0}$	$0.0_{\pm 0.0} \rightarrow 85.0_{\pm 6.9}$	$1.2_{\pm 2.2} \rightarrow 47.2_{\pm 29.8}$	$1.2_{\pm 2.2} \rightarrow 69.2_{\pm 14.8}$	$\mathbf{5.6}_{\pm 2.5} \rightarrow 88.8_{\pm 3.2}$	$\mathbf{5.6}_{\pm 2.5} \rightarrow \mathbf{96.6}_{\pm 1.9}$
	task4	$0.0_{\pm 0.0} \rightarrow 0.0_{\pm 0.0}$	$\mathbf{0.4}_{\pm 0.5} \rightarrow 74.2_{\pm 10.9}$	$0.0_{\pm 0.0} \rightarrow 0.0_{\pm 0.0}$	$0.0_{\pm 0.0} \rightarrow 92.2_{\pm 7.0}$	$0.0_{\pm 0.0} \rightarrow 91.2_{\pm 2.4}$	$0.0_{\pm 0.0} \rightarrow 94.2_{\pm 4.1}$	$0.2_{\pm 0.4} \rightarrow 95.2_{\pm 3.2}$	$0.2_{\pm 0.4} \rightarrow \mathbf{99.0}_{\pm 1.7}$
Average		28.6→38.4	**33.2**→83.8	14.8→30.3	25.7→86.8	17.2→80.3	17.2→82.7	27.3→88.4	27.3→**95.8**

Offline RL Results. Although our primary target is offline-to-online fine-tuning, DFP can also be trained from scratch in the pure offline RL setting by jointly optimizing behavior cloning and the top-K Q-maximization objective. As Tab. 7 shows, DFP attains the best success rate on `cube-double-task2` and `scene-task2` and remains close to the strongest baseline on `cube-single-task2` and `puzzle-4x4-task4`, indicating effectiveness in this setting as well.

Table 7: **Offline RL results on default OGBench tasks.** Mean with std as subscript, over 8 seeds. Best result per task in **bold**; second-best underlined.

	Gaussian Policies			Diffusion Policies			Flow Policies				Ours
Task	BC	IQL	ReBRAC	IDQL	SRPO	CAC	FAWAC	FBRAC	IFQL	FQL	**DFP**
cube-single-task2	3 ± 1	85 ± 8	92 ± 4	96 ± 2	82 ± 16	80 ± 30	81 ± 9	83 ± 13	73 ± 3	$\mathbf{97 \pm 2}$	95 ± 3
cube-double-task2	0 ± 0	1 ± 1	7 ± 3	16 ± 10	0 ± 0	2 ± 2	2 ± 1	22 ± 12	9 ± 5	36 ± 6	$\mathbf{41 \pm 4}$
scene-task2	1 ± 1	12 ± 3	50 ± 13	33 ± 14	2 ± 2	50 ± 40	18 ± 8	46 ± 10	0 ± 0	76 ± 9	$\mathbf{93 \pm 4}$
puzzle-3x3-task4	1 ± 1	2 ± 1	2 ± 1	0 ± 0	0 ± 0	0 ± 0	1 ± 1	2 ± 2	0 ± 0	$\mathbf{16 \pm 5}$	3 ± 2
puzzle-4x4-task4	0 ± 0	4 ± 1	10 ± 3	$\mathbf{26 \pm 6}$	7 ± 4	1 ± 1	0 ± 0	5 ± 1	21 ± 11	11 ± 3	20 ± 2

Hyperparameters. We list the offline-specific hyperparameters in Tab. 8. Unlike the offline-to-online setting, we disable both action chunking and best-of-N' execution for offline RL. All remaining hyperparameters follow the shared block of Tab. 5

Table 8: **Task-specific hyperparameters for DFP offline training.**

Hyperparameter	cube-single task2	cube-double task2	scene task2	puzzle-3x3 task4	puzzle-4x4 task4
Drift weight λ	0.45	0.55	0.5	0.5	0.5
Kernel bandwidth (h)	$\{0.05\}$	$\{0.05\}$	$\{0.01, 0.05\}$	$\{0.01, 0.05\}$	$\{0.01, 0.05\}$
Num. candidates for top-K (N)	16	16	16	16	32
Top-K positive set size (K)	8	8	8	8	16

B.3 Asymmetric Effect of $\mathcal{L}_{\text{top-}K}$ Across Backbones

Figure 3 shows the training curves of the backbone × loss ablation. Adding $\mathcal{L}_{\text{top-}K}$ to the MeanFlow backbone (**MVP** → **MVP w/ $\mathcal{L}_{\text{top-}K}$**) yields only a marginal gain that often falls within the seed variance, while the same supervision on the drifting backbone (**DFP w/o $\mathcal{L}_{\text{top-}K}$** → **DFP**) produces a substantially larger lift, most pronounced on the long-horizon `cube-triple` and `cube-quadruple` splits.

B.4 Top-K Positive Set Size

Fig. 4 shows the training curves for $K \in \{1, 2, 4, 8\}$ in $\mathcal{L}_{\text{top-}K}$, with all other components fixed.

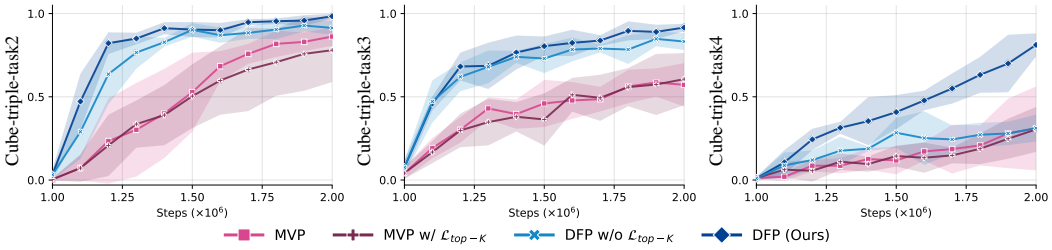


Figure 3: **Online training curves for the backbone $\times$ loss ablation.** Success rate over the online phase, comparing MVP [66], MVP w/ $\mathcal{L}_{\text{top-}K}$, DFP w/o $\mathcal{L}_{\text{top-}K}$, and DFP.

Table 9: Inference and online training cost comparison across baselines. We report the mean wall-clock time (ms) per online training step and per inference step, averaged over 12 tasks.

	BFN	**QC-BFN**	**FQL**	**QC-FQL**	**MVP**	**MVP w/ $\mathcal{L}_{\text{top-}K}$**	**DFP w/o $\mathcal{L}_{\text{top-}K}$**	**DFP (Ours)**
Online Cost (ms)	15.30	16.09	12.00	12.56	13.20	13.48	13.56	13.60
Evaluation Cost (ms)	291.33	295.07	21.90	21.93	25.71	25.71	29.27	29.27

B.5 λ Ablations

Fig. 5 shows the effect of the drift weight $\lambda \in \{0.1, 0.5, 1.0, 5.0\}$ on the cube-quadruple tasks. DFP is robust for $\lambda \geq 0.5$, with all settings yielding near-identical performance.

B.6 Training and Inference Cost

We measure per-step wall-clock cost for online training and inference, averaged across the 12 benchmark tasks (Tab. 9); inference cost is measured on CPU following the protocol of MVP [66]. In our implementation, the measured overhead of $\mathcal{L}_{\text{top-}K}$ is small: DFP (13.60 ms/step) is within 0.04 ms of DFP w/o $\mathcal{L}_{\text{top-}K}$, and the full DFP cost is comparable to other one-step baselines and faster than the multi-step BFN and QC-BFN. At inference, DFP requires a single forward pass of f_θ and stays in the same regime as the other one-step baselines (FQL, QC-FQL, MVP), all roughly an order of magnitude faster than BFN and QC-BFN. The gains reported in Sec. 5 thus come at negligible training overhead and at the inference cost of a standard one-step policy.

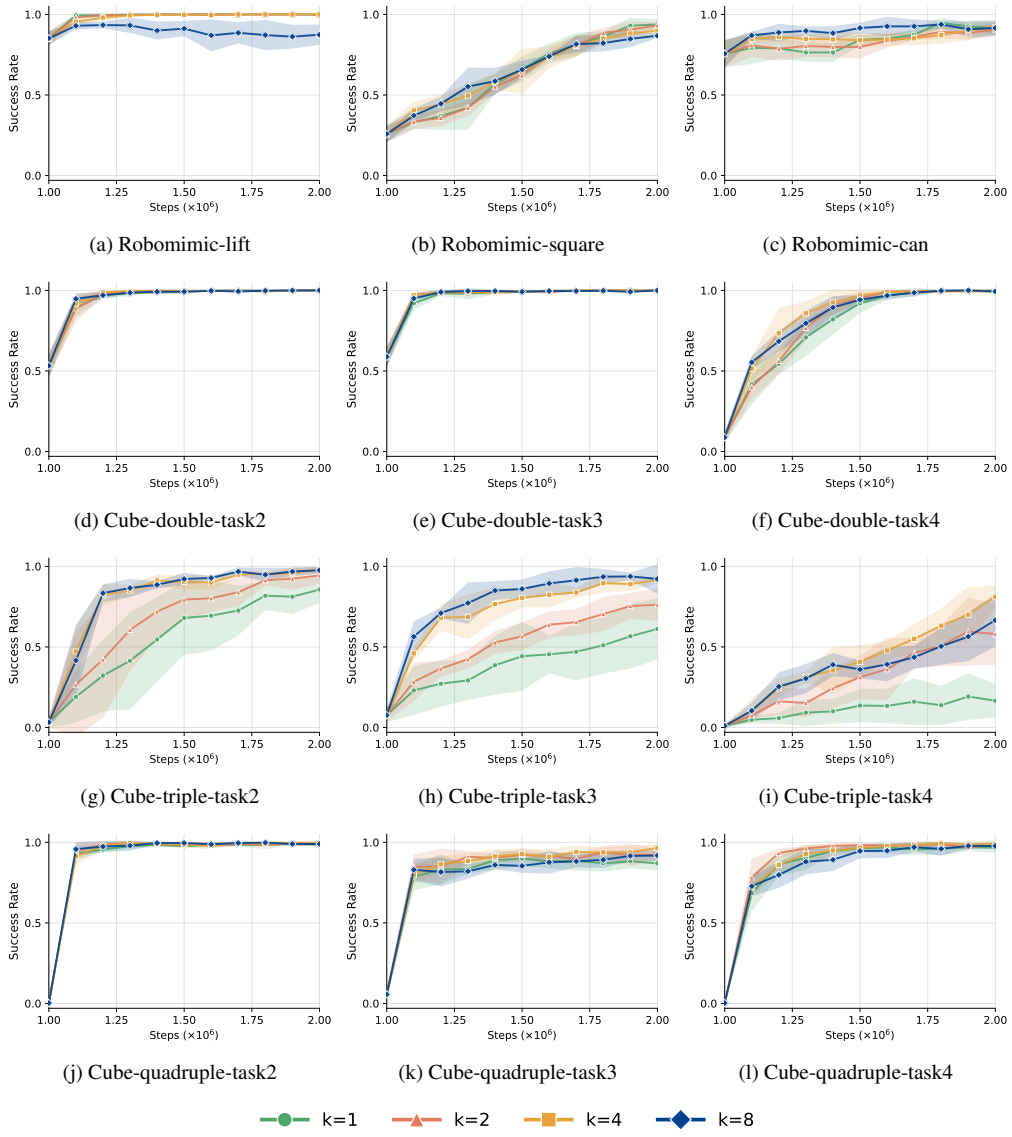


Figure 4: **Online training curves for the top-K ablation.** Success rate over the online phase on Robomimic (top row) and the OGBench.

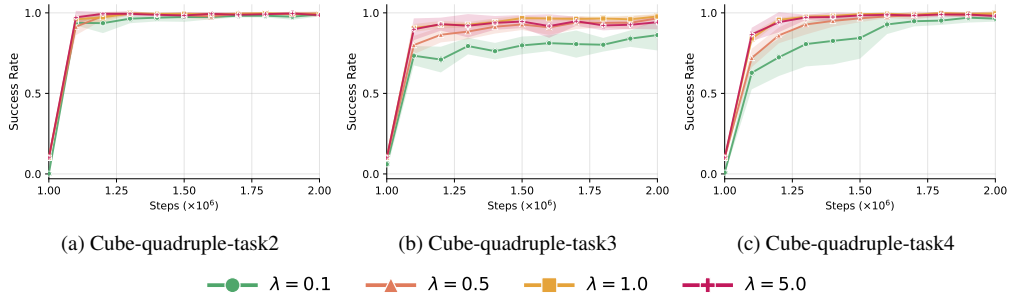


Figure 5: **Online training curves for different λ values.** Success rate over the online phase.

C Proofs

We provide the formal derivations of the equations in Sec. 3.1 in Appendix C.1 and the proof of Proposition 1 in Appendix C.2.

C.1 Derivations for Sec. 3.1

Lagrangian derivation of π^+ (Eq. (10)). The closed form follows from the standard Lagrangian derivation of the KL-regularized Q-maximization update [33, 20]; we leave the derivation here for completeness. The per-iteration regularized optimal policy is defined as the maximizer of the KL-regularized policy improvement objective,

$$\pi^+(\cdot|s) = \arg\max_{\pi} \;\; \mathbb{E}_{a\sim\pi}[Q_\phi(s,a)] \;-\; \alpha\,\mathrm{KL}\big(\pi(\cdot|s)\,\big\|\,\pi_{\mathrm{old}}(\cdot|s)\big) \quad \text{s.t.} \int \pi(a|s)\,da = 1. \tag{19}$$

Forming the Lagrangian (state s fixed, with multiplier λ for normalization),

$$\mathcal{L}(\pi,\lambda) = \int \pi(a)\,Q_\phi(s,a)\,da - \alpha \int \pi(a)\,\log\frac{\pi(a)}{\pi_{\mathrm{old}}(a)}\,da + \lambda\Big(1 - \int \pi(a)\,da\Big), \tag{20}$$

and setting $\delta\mathcal{L}/\delta\pi(a) = 0$,

$$Q_\phi(s,a) - \alpha\Big(\log\frac{\pi(a)}{\pi_{\mathrm{old}}(a)} + 1\Big) - \lambda = 0. \tag{21}$$

Solving yields $\pi(a) = \pi_{\mathrm{old}}(a)\,\exp\big((Q_\phi(s,a) - \alpha - \lambda)/\alpha\big)$, and absorbing the a-independent factor into the partition function,

$$\pi^+(a|s) = \frac{\pi_{\mathrm{old}}(a|s)\,\exp(Q_\phi(s,a)/\alpha)}{Z(s)}, \qquad Z(s) = \int \pi_{\mathrm{old}}(a'|s)\,\exp(Q_\phi(s,a')/\alpha)\,da'. \quad \square \tag{22}$$

Lemma 1 (W_2 gradient flow velocity of $\mathrm{KL}(q\|p)$)**.** *For a fixed reference $p \in \mathcal{P}_2(\mathbb{R}^d)$ with absolutely continuous, strictly positive density, the W_2 gradient flow particle velocity of the KL functional $\mathcal{F}(q) = \mathrm{KL}(q\|p)$ is the score difference $v_t(x) = \nabla_x \log p(x) - \nabla_x \log q_t(x)$ (Eq. (6)).*

Proof. Writing $\mathcal{F}(q) = \int q(x)\log\big(q(x)/p(x)\big)\,dx$ and computing the first variation,

$$\frac{\delta\mathcal{F}}{\delta q}(x) = \log\frac{q(x)}{p(x)} + 1, \tag{23}$$

so that

$$\nabla_x \frac{\delta\mathcal{F}}{\delta q}(x) = \nabla_x \log q(x) - \nabla_x \log p(x). \tag{24}$$

The W_2 gradient flow particle velocity is the steepest-descent direction $v_t(x) = -\nabla_x \frac{\delta\mathcal{F}}{\delta q_t}(x)$ [25, 2], which yields the score-difference form. $\square$

Drifting field as KDE-WGF velocity on policy space (Eq. (13)). The identification of the drifting field $\mathbf{V}_{p,q}$ with the KDE-approximated W_2 gradient flow velocity of $\mathrm{KL}(q\|p)$ is established by [6]; we reproduce the specialization to policy space here for completeness. By Lemma 1 applied to $\mathcal{F}(\pi_t) = \mathrm{KL}(\pi_t\|\pi^+)$ with $p = \pi^+, q_t = \pi_t$, the W_2 gradient flow particle velocity on policy space is

$$v_t(a|s) = \nabla_a \log \pi^+(a|s) - \nabla_a \log \pi_t(a|s). \tag{25}$$

Applying the KDE gradient identity Eq. (7) to both score functions with kernel bandwidth h,

$$h^2\,\nabla_a \log \pi^+_{\mathrm{kde}}(a|s) = \mathbf{V}^+_{\pi^+(\cdot|s)}(a), \qquad h^2\,\nabla_a \log \pi_{t,\mathrm{kde}}(a|s) = \mathbf{V}^-_{\pi_t(\cdot|s)}(a), \tag{26}$$

where $\mathbf{V}^+, \mathbf{V}^-$ are the kernel mean-shift forms in Sec. 2.2. Multiplying Eq. (25) by h^2 and substituting the KDE-approximated scores,

$$h^2\,v_t^{\mathrm{kde}}(a|s) = \mathbf{V}^+_{\pi^+(\cdot|s)}(a) - \mathbf{V}^-_{\pi_t(\cdot|s)}(a) = \mathbf{V}_{\pi^+,\pi_t}(a|s). \tag{27}$$

Specializing $\pi_t = \pi_\theta$ gives Eq. (13). $\square$

Decomposition of the drifting field (Eq. (14)). Taking the a-gradient of $\log \pi^+(a|s)$ from the closed form $\pi^+(a|s) = \pi_{\text{old}}(a|s)\exp(Q_\phi(s,a)/\alpha)/Z(s)$,

$$\nabla_a \log \pi^+(a|s) = \nabla_a \log \pi_{\text{old}}(a|s) + \frac{1}{\alpha}\nabla_a Q_\phi(s,a), \tag{28}$$

since $\nabla_a \log Z(s) = 0$. Eq. (13) expresses $\mathbf{V}_{\pi^+,\pi_\theta}$ in terms of KDE-smoothed log densities; under the small-bandwidth limit $\log p_{\text{kde}} \to \log p$, substituting the Boltzmann gradient above yields

$$\begin{aligned}
\mathbf{V}_{\pi^+,\pi_\theta}(a|s) &\simeq h^2\big[\nabla_a \log \pi^+(a|s) - \nabla_a \log \pi_\theta(a|s)\big] \\
&= h^2\big[\nabla_a \log \pi_{\text{old}}(a|s) + \tfrac{1}{\alpha}\nabla_a Q_\phi(s,a) - \nabla_a \log \pi_\theta(a|s)\big] \\
&= \frac{h^2}{\alpha}\nabla_a Q_\phi(s,a) + h^2\left(\nabla_a \log \pi_{\text{old}}(a|s) - \nabla_a \log \pi_\theta(a|s)\right),
\end{aligned}$$

which is Eq. (14). □

C.2 Proof of Proposition 1

We prove the two claims of Proposition 1: (i) convergence of $\mathcal{L}_{\text{top-}K}$ to the level-set drift loss, and (ii) the total-variation bias bound to $\mathcal{L}_{\text{PI}}$.

Setup. Let $a^{(1)},\dots,a^{(N)} \overset{\text{i.i.d.}}{\sim} \pi_{\text{old}}(\cdot|s)$ and let $P_K(s) := \text{TopK}_j Q_\phi(s,a^{(j)})$ be the empirical top-K set with $K/N = \rho$. Denote by F_s the cumulative distribution function of $Q_\phi(s,A)$ for $A \sim \pi_{\text{old}}(\cdot|s)$, and by $q^\rho(s) := F_s^{-1}(1-\rho)$ the population $(1-\rho)$-quantile. Under the density assumption (strictly positive density of F_s at $q^\rho(s)$), the empirical $(1-\rho)$-quantile $\hat{q}_N^\rho(s)$ satisfies $\hat{q}_N^\rho(s) \to q^\rho(s)$ a.s. by the Bahadur representation (or directly by Glivenko-Cantelli applied to F_s).

(i) Convergence of $\mathcal{L}_{\text{top-}K}$ to the level-set drift loss. The empirical distribution $P_K(s)$ is the conditional empirical measure of $\{a^{(j)}\}$ given $Q_\phi(s,a^{(j)}) \geq \hat{q}_N^\rho(s)$. By a standard truncation argument, $P_K \xrightarrow{d} \tilde{\pi}^\rho$ in W_2 as $N \to \infty$. The kernel mean shift $\mathbf{V}_p^+(x)$, defined by Eq. (7), is continuous in p in total variation. Writing $\mathbf{V}_p^+(x) = N(p)/D(p)$ with $N(p) := \int k(x,y)(y-x)\,dp(y)$ and $D(p) := \int k(x,y)\,dp(y)$, and using $\left|\int f\,(dp-dq)\right| \leq \|f\|_\infty \,\text{TV}(p,q)$ on both with $\|k(x,\cdot)(\cdot - x)\|_\infty \leq K_{\max}\,\text{diam}(\mathcal{A})$ and $\|k(x,\cdot)\|_\infty \leq K_{\max}$, the standard quotient identity yields

$$\left|\mathbf{V}_p^+(x) - \mathbf{V}_q^+(x)\right| \;\leq\; L_V\,\text{TV}(p,q), \quad L_V \;=\; \mathcal{O}\!\left(\frac{K_{\max}^2\,\text{diam}(\mathcal{A})}{k_{\min}^2}\right), \tag{29}$$

where $k_{\min}$ is a positive lower bound on $D(\cdot)$ (positive when p,q have local full support). Combining with $P_K \to \tilde{\pi}^\rho$ in TV gives $\mathbf{V}_{P_K}^+ \to \mathbf{V}_{\tilde{\pi}^\rho}^+$ pointwise. The negative side $\mathbf{V}_{\pi_\theta}^-$ is unchanged (population), so the drift field $\mathbf{V}_{P_K,\pi_\theta} \to \mathbf{V}_{\tilde{\pi}^\rho,\pi_\theta}$. The squared-norm form of $\mathcal{L}_{\text{drift}}$ (Eq. (4)) and dominated convergence yield $\mathcal{L}_{\text{top-}K}(\theta) \to \mathcal{L}_{\text{drift}}(\theta;\,\tilde{\pi}^\rho,\pi_\theta)$.

(ii) Bias bound to $\mathcal{L}_{\text{PI}}$. Both $\mathcal{L}_{\text{drift}}(\theta;\,\tilde{\pi}^\rho,\pi_\theta)$ and $\mathcal{L}_{\text{PI}}(\theta) = \mathcal{L}_{\text{drift}}(\theta;\,\pi^+,\pi_\theta)$ share the same negative side $\mathbf{V}_{\pi_\theta}^-$, so by Eq. (29),

$$\left\|\mathbf{V}_{\tilde{\pi}^\rho}^+(\hat{a}|s) - \mathbf{V}_{\pi^+}^+(\hat{a}|s)\right\| \;\leq\; L_V\,\text{TV}\big(\tilde{\pi}^\rho(\cdot|s),\pi^+(\cdot|s)\big). \tag{30}$$

Expanding the squared norm in Eq. (4) $\mathcal{L}_{\text{drift}}(\theta;p,q) = \mathbb{E}\big[\|\mathbf{V}_p^+ - \mathbf{V}_q^-\|^2\big]$. The difference factorizes as

$$\mathcal{L}_{\text{drift}}(\theta;\,\tilde{\pi}^\rho,\pi_\theta) - \mathcal{L}_{\text{PI}}(\theta) = \mathbb{E}\big[(\mathbf{V}_{\tilde{\pi}^\rho}^+ - \mathbf{V}_{\pi^+}^+)\cdot(\mathbf{V}_{\tilde{\pi}^\rho}^+ + \mathbf{V}_{\pi^+}^+ - 2\mathbf{V}_{\pi_\theta}^-)\big]. \tag{31}$$

By Cauchy-Schwarz and the bounded norm of kernel mean shift ($\|\mathbf{V}_p^+\|, \|\mathbf{V}_q^-\| \leq M$ with $M = \text{diam}(\mathcal{A})$, since the mean shift is a weighted average of $(y-x)$ with $y,x \in \mathcal{A}$),

$$\left|\mathcal{L}_{\text{drift}}(\theta;\,\tilde{\pi}^\rho,\pi_\theta) - \mathcal{L}_{\text{PI}}(\theta)\right| \;\leq\; 4M\,\mathbb{E}\big[\|\mathbf{V}_{\tilde{\pi}^\rho}^+ - \mathbf{V}_{\pi^+}^+\|\big] \;\leq\; 4ML_V\,\text{TV}(\tilde{\pi}^\rho,\pi^+), \tag{32}$$

which establishes the bound with constant $C := 4ML_V = \mathcal{O}\big(K_{\max}^2\,\text{diam}(\mathcal{A})^2/k_{\min}^2\big)$ depending only on the kernel and action-space diameter.

Tightness of the bound. The TV gap $\overline{\mathrm{TV}}(\tilde{\pi}^\rho, \pi^+)$ is a function of both ρ and α, reflecting the structural mismatch between hard ρ-quantile truncation and soft Boltzmann tilting $\exp(Q_\phi/\alpha)$. In the joint sharp limit $\rho, \alpha \to 0$, both distributions collapse to $\delta_{a^\star(s)}$ at the argmax and the gap vanishes. More relevantly for our finite-parameter setting, for each soft temperature α, there exists a matched truncation level $\rho^*(\alpha) := \arg\min_\rho \overline{\mathrm{TV}}(\tilde{\pi}^\rho, \pi^+(\alpha))$ at which the gap is minimized; equivalently, our choice of $\rho = K/N$ implicitly selects a matched $\alpha^*(\rho)$ at which the bound is tight, with the residual gap small under mild regularity of the Q-distribution under π_{old}. □

C in RL settings. The constant $C = \mathcal{O}\big(K_{\max}^2 \operatorname{diam}(\mathcal{A})^2 / k_{\min}^2\big)$ depends on the kernel $(K_{\max}, k_{\min})$ and quadratically on the action-space diameter. Standard continuous-control RL benchmarks clip the action space to $\mathcal{A} = [-1, 1]^d$ with low dimension d (e.g., Robomimic [39] and OGBench [44]), so $\operatorname{diam}(\mathcal{A}) = 2\sqrt{d}$ remains small and C is moderate in practice, in contrast to the high-dimensional image-generation setting where drifting models were originally introduced [10].

D Computation Costs

In this section, we report the computational resources used in our experiments. All experiments were conducted on Nvidia RTX 3090 GPUs with Intel Xeon Gold 6342 CPUs (96 cores). Table 10 reports the GPU-hours used for each experiment.

Table 10: Computational resources for each experiment in this paper. We report the total GPU-hours aggregated across all tasks and seeds, measured on Nvidia RTX 3090 GPUs.

Experiment	**GPU-hours**
Main offline-to-online (Table 1)	3,500h
Backbone $\times$ $\mathcal{L}_{\text{top-}K}$ ablation (Table 2)	900h
Top-K size ablation (Table 4)	1,500h
Drift weight λ ablation (Table 3)	500h
Offline RL evaluation (Sec. B.2)	100h
Total	6,500h

E Societal Impact

Our work introduces a non-ODE generative policy paradigm for offline-to-online reinforcement learning, with potential downstream applications in robotics, automated manufacturing, and other settings requiring learning from limited demonstrations followed by online refinement. As with reinforcement learning methods broadly, deployed policies inherit biases in the reward specification and demonstration data and may behave unpredictably under distribution shift; physical deployment requires additional safety mechanisms beyond what our simulation experiments evaluate. We do not foresee specific misuse pathways unique to this work beyond those associated with reinforcement learning for continuous control.