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, jiwonch, 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 action generation, DFP achieves state-of-the-art performance on several manipulation tasks across Robomimic and OGBench, outperforming ODE-based policies. Code is available at `https://github.com/KAIST-Visual-AI-Group/DFP`.

1 Introduction

Offline-to-online reinforcement learning (RL) has emerged as a practical paradigm for continuous control [43, 30, 44, 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 [63, 21, 11, 46, 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, 57, 27] that amortize ODE trajectory integration into a single forward pass.

Despite the success of generative policies in offline behavior cloning [4, 24, 49, 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 [51, 35, 11, 41, 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

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

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 show that this surrogate converges to a hard-truncated version of the ideal update, with a bounded bias. 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 [63, 21, 11, 46, 67] 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 intractable target π^+ with a bounded bias relative 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 [40] and OGBench [45] 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 [67], 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 [65, 15, 60] 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, 42] 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, 56, 58, 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, 52]. 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 finetuning 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 push-forward 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.

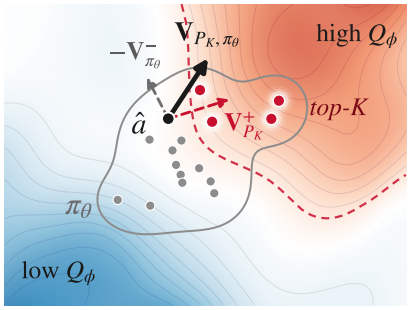


Figure 1: **One DFP update in a 2D action space.** The critic Q_ϕ selects the top-K policy samples as P_K, and $\mathbf{V}_{P_K,\pi_\theta}$ moves $\hat{a}$ toward P_K and away from the other samples of π_θ.

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, 61] and policy search [53, 48, 1, 34, 43, 47]:

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

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*

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

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 its bias with respect to $\mathcal{L}_{\mathrm{PI}}$ is bounded (Proposition 1).

Denoting the resulting positive set by

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

the tractable surrogate is

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

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. Fig. 1 illustrates the resulting update at a single sample.

Proposition 1 (Limit and bias of $\mathcal{L}_{\text{top-}K}$)**.** *Let $\rho := K/N$ and let $q^\rho(s)$ be the $(1-\rho)$-quantile of $Q_\phi(s, A)$ for $A \sim \pi_{\mathrm{old}}(\cdot|s)$. Define the ρ-level-set tilting $\tilde{\pi}^\rho(a|s) := \rho^{-1}\,\mathbb{1}[Q_\phi(s,a) \geq q^\rho(s)]\,\pi_{\mathrm{old}}(a|s)$. Under mild regularity conditions, as $N \to \infty$ with ρ fixed,*

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

for a constant C depending only on the kernel and action-space diameter, where $\overline{\mathrm{TV}}(p,q) := \mathbb{E}_s[\mathrm{TV}(p(\cdot|s), q(\cdot|s))]$. Conditions and full proof in Appendix C.2.

Proposition 1 characterizes $\mathcal{L}_{\text{top-}K}$ as a controlled but biased surrogate of $\mathcal{L}_{\mathrm{PI}}$. The two are drift losses with the same negative source π_θ and differ only in the positive target: π^+ for $\mathcal{L}_{\mathrm{PI}}$ and, as N grows, $\tilde{\pi}^\rho$ for $\mathcal{L}_{\text{top-}K}$. The ratio ρ controls how selective $\tilde{\pi}^\rho$ is, a role analogous to that of the temperature α in π^+.

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 $\mathcal{D}$:

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

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

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

The combined loss $\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{BC}}(\theta) + \lambda\,\mathcal{L}_{\text{top-}K}(\theta)$ 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 [67, 46] 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, 57] 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 [63, 11], including one-step variants [67, 55, 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 [58, 38, 23, 56], and one-step inference (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 $P_K(s^{(b)}) = \left\{\tilde{a}^{(b,j)} : j \in \text{argTopK}_{j\in[N]} Q_\phi(s^{(b)}, \tilde{a}^{(b,j)})\right\}$
  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

NFE) [57, 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 [51, 41, 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, 44], (ii) behavior-cloning regularization on the policy [43, 60], and (iii) replay strategies mixing offline and online transitions [3, 59]. 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 [63, 21, 11]. The field has since shifted toward few-step or one-step backbones for inference efficiency [67, 64, 55, 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, 57, 14] on image generation. Follow-up analyses relate the drift field to score matching [32, 62] 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 [66] and KDP [50] only target offline imitation learning, while DBPO [17] targets offline-to-online fine-tuning but projects the drifting policy onto a unimodal Gaussian for PPO [54] 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

The appendix provides training and evaluation details (Appendix A.3), offline RL experiments (Appendix B.7), full offline-to-online results (Appendix B.1), hyperparameters and the (N, K) ablation (Appendices A.4 and B.6), fully online RL without $\mathcal{L}_{\mathrm{BC}}$ (Appendix B.8), and cost analysis (Appendix B.9).

5.1 Experimental Setup

Benchmarks. We follow the experimental setup of Mean Velocity Policy (MVP) [67], evaluating on 3 tasks from Robomimic [40] (`Lift`, `Can`, `Square`) under the multi-human (MH) datasets and 6 tasks from OGBench [45], 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 [67], 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** [46] 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** [67] 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 compare in Sec. 5.3.

Hyperparameters. For $\mathcal{L}_{\text{top-}K}$, we use $N = 16$ candidate actions, with $K = 2$ on Robomimic [40] and $K = 4$ on OGBench [45], and the loss coefficient $\lambda = 0.5$. The kernel bandwidth is $h = 0.05$ on OGBench, and Robomimic uses the bandwidths $\{0.005, 0.05\}$ jointly. Following prior work in which best-of-N' selection defines the executed policy [19, 21, 36, 67], we execute the highest-Q_ϕ action among N' policy samples both during online data collection and at evaluation, with $N' = 4$ on Robomimic and $N' = 16$ on OGBench. All experiments use these values unless otherwise stated, and full hyperparameter configurations are listed in Appendix A.4.

Training protocol. All methods are pretrained offline for 1M steps and fine-tuned online for 1M steps. During offline pretraining, DFP trains the actor with $\mathcal{L}_{\mathrm{BC}}$ alone and the critic with Eq. (1); online fine-tuning adds $\mathcal{L}_{\text{top-}K}$ as in Algorithm 1.

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 [67], 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

Table 1: **Success rate (%) on Robomimic [40] and OGBench [45] 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 [46]	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 [67]	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**

Table 2: **Ablation on the $\mathcal{L}_{\text{top-}K}$ for MVP [67] and DFP.** Success rate (%) on Robomimic and OGBench tasks under the offline-to-online protocol. Rows without $\mathcal{L}_{\text{top-}K}$ train each backbone with its own BC objective on the same replay-buffer protocol; rows with it add the same top-K targets through that objective. 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	**Backbone**	$\mathcal{L}_{\text{top-}K}$	`lift`	`square`	`can`	`task2`	`task3`	`task4`	`task2`	`task3`	`task4`	`task2`	`task3`	`task4`	**Avg.**
MVP [67]	**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**

over MVP [67] 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. 2, DFP outperforms MVP [67] 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.

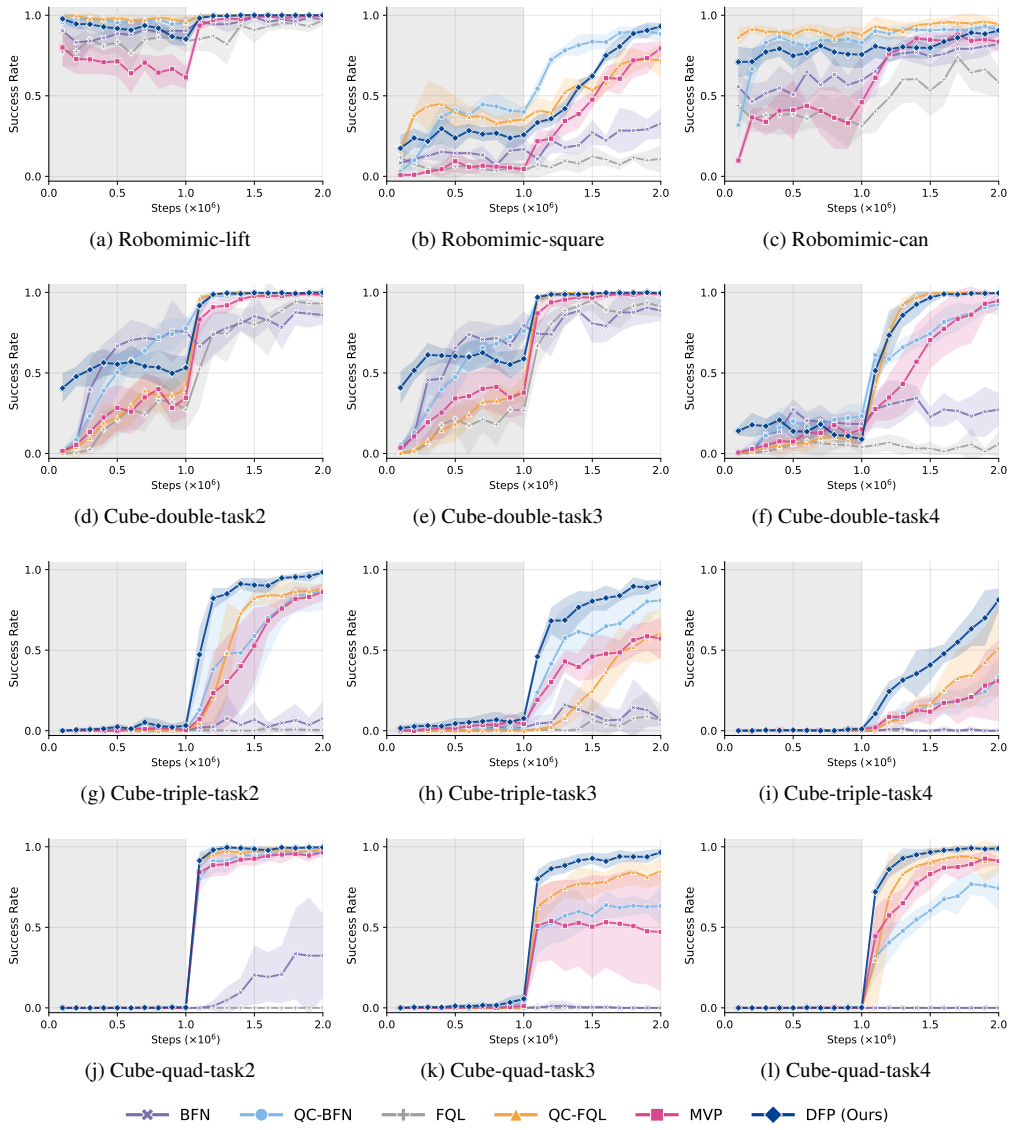
5.3 Analysis: Drifting vs. MeanFlow Backbones

To isolate the contribution of each design choice that distinguishes DFP from MVP [67], we ablate two factors: the policy backbone, MeanFlow [18] or drifting model [10], and the training objective, BC only or BC + $\mathcal{L}_{\text{top-}K}$ (Tab. 2). The two BC-only rows share the same protocol, behavior cloning toward a replay buffer filled with best-of-N' actions, but each keeps its backbone-specific BC objective: the MeanFlow identity and IVC losses for MVP, and $\mathcal{L}_{\text{BC}}$ of Eq. (18) for DFP. The rows with $\mathcal{L}_{\text{top-}K}$ add the same critic-selected top-K targets through each backbone's native objective; for MeanFlow, this applies its identity loss to the top-K candidates. Without $\mathcal{L}_{\text{top-}K}$, DFP yields $+8.1$ pp in average success rate over MVP, indicating that the drifting formulation imitates more effectively. 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 is consistent with the parameterization-level distinction discussed in Sec. 3.2. On the drifting backbone, the top-K supervision is applied directly to generated actions through the pushforward 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, which may explain why the same targets yield a smaller shift in π_θ. See Appendix B.2 for training curve comparisons.

5.4 Ablation Study

Refer to Appendices B.3–B.5 for more results of these ablations.



(a) Robomimic-lift (b) Robomimic-square (c) Robomimic-can

(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-quad-task2 (k) Cube-quad-task3 (l) Cube-quad-task4

Figure 2: **Success rate over training steps across Robomimic [40] and OGBench [45].** 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.

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 for $\lambda \geq 0.5$.

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%). Refer to Appendix B.6 for a joint sweep of N and K.

Effect of the kernel bandwidth h. On the `cube-quadruple` tasks, bandwidths between $h = 0.05$ and $h = 0.1$ perform comparably, whereas a too small $h = 0.01$ fails (Tab. 5).

Effect of the number of best-of-N' candidates at evaluation. To separate the contribution of the learned policy from that of best-of-N' selection at evaluation, we re-evaluate the policies trained

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

λ	task 2	task 3	task 4	Avg.
0.1	99.0 ± 1	86.2 ± 8	96.4 ± 2	93.9
0.5	$\mathbf{99.6 \pm 1}$	$\mathbf{96.6 \pm 2}$	99.0 ± 2	**98.4**
1.0	98.4 ± 1	$\mathbf{96.6 \pm 2}$	$\mathbf{99.2 \pm 0}$	98.1
5.0	99.2 ± 0	94.2 ± 4	98.0 ± 2	97.1

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

K	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

Table 5: **Kernel bandwidth h ablation.** Success rate (%) on cube-quadruple tasks, `mean` and `std` over 5 seeds. **Bold**/underline: best/second-best. Blue: default.

h	task 2	task 3	task 4	Avg.
0.01	0.0 ± 0	0.0 ± 0	0.0 ± 0	0.0
0.05	99.6 ± 1	96.6 ± 2	99.0 ± 2	98.4
0.1	$\mathbf{100.0 \pm 0}$	98.8 ± 1	$\mathbf{100.0 \pm 0}$	**99.6**
$\{0.05, 0.1\}$	99.8 ± 0	$\mathbf{99.2 \pm 0}$	97.5 ± 1	98.8
$\{0.01, 0.025, 0.1\}$	99.6 ± 1	99.0 ± 1	$\mathbf{100.0 \pm 0}$	99.5

Table 6: **Effect of the number of best-of-N' candidates at evaluation.** Average success rate (%) over the nine OGBench tasks, each averaged over 5 seeds. Only the evaluation-time N' changes; online data collection in Algorithm 1 uses $N' = 16$.

N'	1	2	4	8	16	32
MVP [67]	19.7	51.6	68.7	74.4	77.6	78.4
DFP (Ours)	**72.2**	**93.4**	**96.7**	**96.2**	**95.6**	**94.9**
DFP − MVP	+52.6	+41.8	+28.0	+21.8	+18.1	+16.4

with $N' = 16$ at $N' \in \{1, 2, 4, 8, 16, 32\}$ on the nine OGBench tasks (Tab. 6). With $N' = 1$, i.e., executing a single policy sample without critic selection, DFP reaches 72.2% average success versus 19.7% for MVP, so the learned one-step policy itself accounts for a large part of the gap. DFP saturates at 96.7% by $N' = 4$, whereas MVP depends much more on candidate selection, reaching 78.4% at $N' = 32$, still 15.0 pp below DFP at $N' = 2$.

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 [40] and OGBench [45] 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.

Acknowledgments and Disclosure of Funding

This work was supported by the NRF grant (RS-2026-25486000); IITP grants (RS-2024-00399817, RS-2025-25441313, RS-2025-25443318, RS-2026-25526850, RS-2026-25617720) funded by MSIT, Korea; the Industrial Technology Innovation Program grant (RS-2025-02317326) funded by MOTIE, Korea; the InnoCORE program of MSIT (AI Meta-Scientist, N10260110); and the DRB-KAIST SketchTheFuture Research Center.

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. In *Robotics: Science and Systems*, 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. In *Robotics: Science and Systems*, 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] H. Ma, T. Chen, K. Wang, N. Li, and B. Dai. Efficient online reinforcement learning for diffusion policy. In *International Conference on Machine Learning*, 2025.

[40] 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.

[41] 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.

[42] 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.

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

[44] 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.

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

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

[47] 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.

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

[49] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Robotics: Science and Systems*, 2024.

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

[51] 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.

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

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

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

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

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

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

[58] 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.

[59] 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.

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

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

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

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

[64] 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.

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

[66] 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.

[67] 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.

Appendix Contents

A Experimental Details **16**

A.1 Environment Descriptions 16

A.2 Baseline Method Details 18

A.3 Training and Evaluation Details 18

A.4 Hyperparameters 19

B Additional Results **21**

B.1 More Results of Baseline Comparison 21

B.2 More Results of Backbone $\times$ $\mathcal{L}_{\text{top-}K}$ Ablation 21

B.3 More Results of λ Ablation 21

B.4 More Results of K Ablation 21

B.5 More Results of N' Ablation 23

B.6 (N, K) Ablation 23

B.7 Adaptation to Offline RL 24

B.8 Fully Online RL without $\mathcal{L}_{\text{BC}}$ 24

B.9 Training and Inference Cost 25

C Proofs **26**

C.1 Derivations for Sec. 3.1 26

C.2 Proof of Proposition 1 27

D Computation Costs **29**

E Societal Impact **29**

A Experimental Details

A.1 Environment Descriptions

We evaluate on 12 manipulation tasks drawn from the Robomimic benchmark [40] and the OGBench manipulation suite [45]. 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. 3 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. 3a) – Grasp a randomly placed cube and raise it above a height threshold. Tests basic grasping.
- **Can** (Fig. 3b) – Transport a coke can from one bin to a smaller target bin, requiring coordinated reach, grasp, and release.
- **Square** (Fig. 3c) – 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

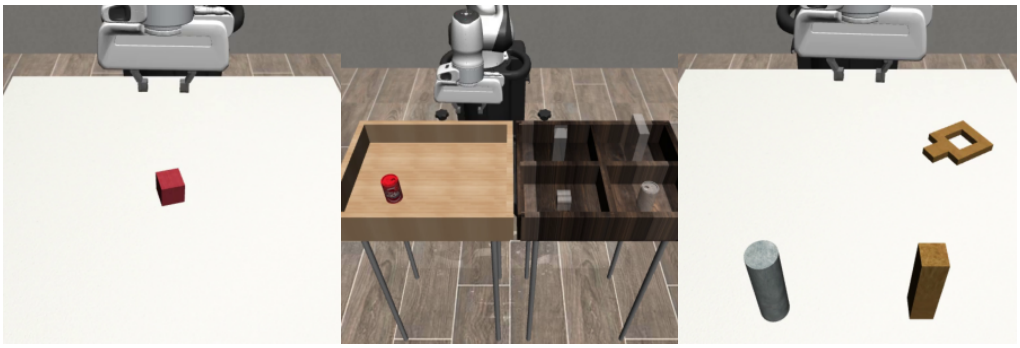
Common setup. The `cube-double`, `cube-triple`, and `cube-quadruple` environments share an identical 6-DoF UR5e arm with a Robotiq 2F-85 gripper but vary the number of cubes $N \in \{2, 3, 4\}$. We use the `play-singletask-task[N]-v0` variants, which fix an evaluation goal and relabel the unstructured `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. 3d–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.

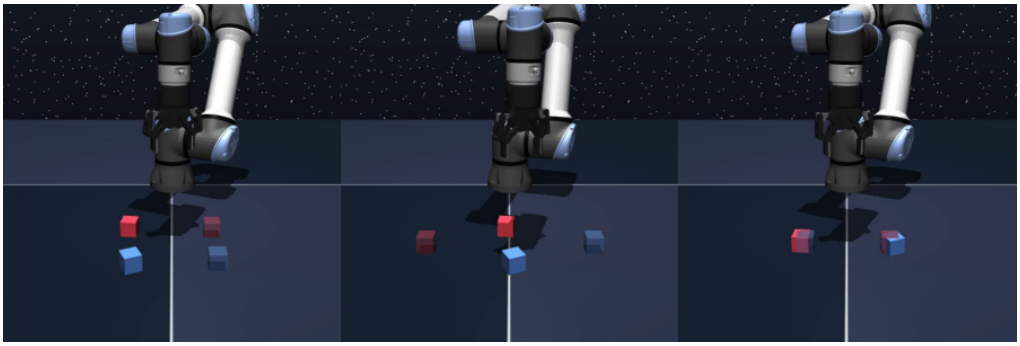
cube-triple (Fig. 3g–i) requires composing pick-and-place primitives in non-trivial sequences: *triple-pnp* rearrangement of three cubes (`task2`), *pnp-from-stack* which involves manipulating cubes from a stacked configuration (`task3`), and *cyclic permutation* of three cubes (`task4`).

cube-quadruple (Fig. 3j–l) amplifies long-horizon coordination, requiring up to four sequential pick-and-place subtasks per goal (Table 1 of [45]): *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.

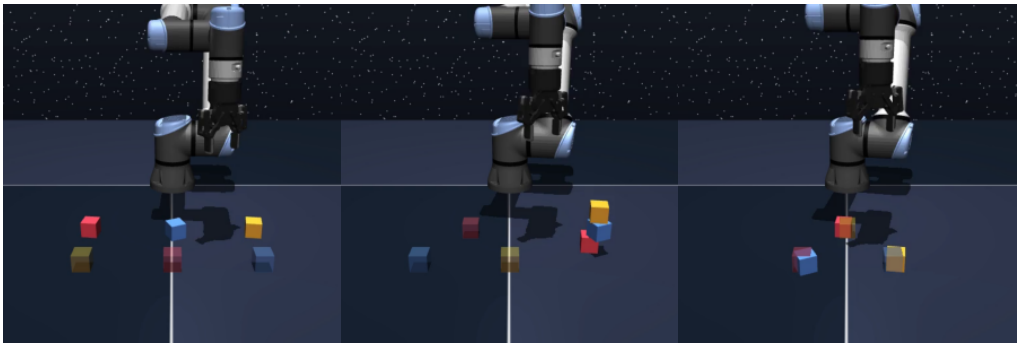
The progression from `cube-double` to `cube-quadruple` is intentionally combinatorial: the same primitives (pick-and-place, swap, cycle) recur, but the number of required subtasks scales with N (up to 1, 2, 4 atomic behaviors for $N = 2, 3, 4$ respectively, per OGBench Table 1), with the longest evaluation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



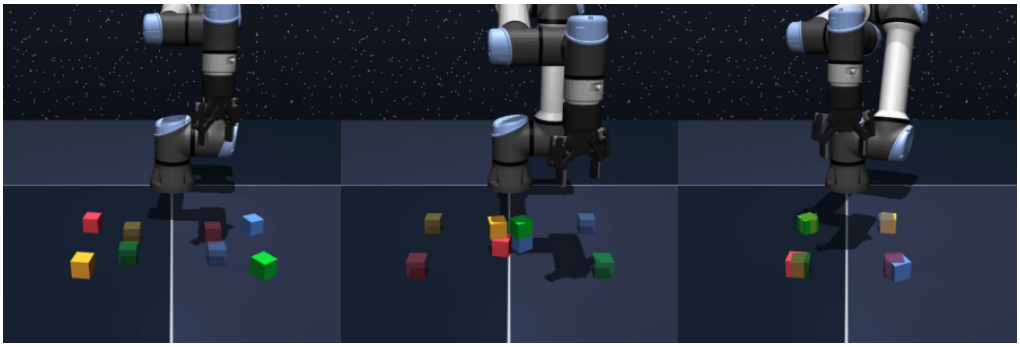
(a) Lift (b) Can (c) Square



(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 3: **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.4.

FQL [46]. 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 [67]. 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 $P_K(s)$ 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 Training and Evaluation Details

Offline pretraining. During the 1M-step offline phase, the replay buffer $\mathcal{D}$ contains only the offline dataset. We sample state-action pairs (s, a) uniformly from $\mathcal{D}$ and train the actor with $\mathcal{L}_{\text{BC}}$ alone, Eq. (18): the dataset action is the positive sample, the actions generated by the current policy form the negative distribution, and the actor is updated by the stop-gradient fixed-point regression of Eq. (4). The term $\mathcal{L}_{\text{top-}K}$ is not used in this phase. The critic is trained on the same batches with the Bellman objective of Eq. (1). Both phases use the batch size, optimizer, and learning rate in Tab. 7.

Online fine-tuning. At the start of online fine-tuning, π_{old} is initialized from the pretrained actor, and new transitions collected with best-of-N' action selection are appended to $\mathcal{D}$. The actor is then trained with $\mathcal{L}_{\text{BC}} + \lambda\,\mathcal{L}_{\text{top-}K}$ as in Algorithm 1. In contrast, the pure offline RL experiments in Appendix B.7 apply both losses from the start of training.

Evaluation. At evaluation, DFP and MVP use the same best-of-N' action selection as during online data collection: N' candidate action chunks are generated in one batched forward pass of the policy, scored by Q_ϕ, and the highest-valued chunk is executed. FQL and QC-FQL execute a single sample from their one-step policy, and BFN and QC-BFN apply best-of-N to their multi-step BC policy. Sec. 5.4 and Appendix B.5 report results for different values of N' at evaluation.

Bounded actions. Actions are normalized to $\mathcal{A} = [-1, 1]^d$, and, following FQL [46], we clip the output of f_θ to $\mathcal{A}$. The kernel mean shifts in Eq. (3) can be written as $\mathbf{V}_p^+(x) = \bar{a}_p(x) - x$ and $\mathbf{V}_q^-(x) = \bar{a}_q(x) - x$, where $\bar{a}_p(x)$ and $\bar{a}_q(x)$ are kernel-weighted means of the positive and negative samples. Since the weights are nonnegative and normalized, both means lie in the convex set $\mathcal{A}$, so $\mathbf{V}_{p,q}(x) = \bar{a}_p(x) - \bar{a}_q(x)$ satisfies $\|\mathbf{V}_{p,q}(x)\|_2 \leq \mathrm{diam}(\mathcal{A}) = 2\sqrt{d}$, and the drifting field cannot produce arbitrarily large updates. We did not observe boundary effects of the kernel, and Tab. 5 reports the effect of the bandwidth h.

One positive action per state in behavior cloning. Standard offline datasets provide one action for each continuous state. Thus, the single positive in $\mathcal{L}_{\mathrm{BC}}$ reflects the supervision available in the data rather than subsampling one of several available actions, as for diffusion and flow policies trained on the same data. Multimodal behavior must instead be learned through amortization across nearby states, while the repulsion from generated samples discourages collapse. This setting differs from image generation with drifting models [10], where several positives can be drawn for the same condition, so the effect of the number of positives reported there does not transfer directly. During online fine-tuning, $\mathcal{L}_{\text{top-}K}$ uses K positives per state.

A.4 Hyperparameters

Tab. 7 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 [46, 19, 36, 67].

Table 7: **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
Top-K loss weight (λ)	0.5
Actor EMA smoothing (τ_{EMA})	1×10^{-4}
Number of candidates for best-of-N' execution (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`).

B Additional Results

Appendices B.1–B.5 report more comprehensive results of the experiments in Sec. 5, and Appendices B.6–B.9 report additional experiments.

B.1 More Results of Baseline Comparison

Table 8: **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 [46]	QC-FQL [36]	MVP [67]	MVP w/ $\mathcal{L}_{\text{top-}K}$	**DFP w/o $\mathcal{L}_{\text{top-}K}$**	**DFP (Ours)**
Robomimic	lift	90.4$_{\pm 7.1}$→97.6$_{\pm 2.2}$	95.2$_{\pm 3.4}$→99.6$_{\pm 0.5}$	84.0$_{\pm 5.7}$→96.8$_{\pm 2.3}$	**95.8**$_{\pm 3.3}$→**100.0**$_{\pm 0.0}$	61.4$_{\pm 5.7}$→99.8$_{\pm 0.4}$	61.4$_{\pm 5.7}$→100.0$_{\pm 0.0}$	85.2$_{\pm 3.4}$→**100.0**$_{\pm 0.0}$	85.2$_{\pm 3.4}$→**100.0**$_{\pm 0.0}$
	square	16.8$_{\pm 5.2}$→32.8$_{\pm 7.6}$	40.0$_{\pm 2.0}$→88.4$_{\pm 3.6}$	3.6$_{\pm 3.6}$→10.8$_{\pm 6.7}$	35.4$_{\pm 6.7}$→72.0$_{\pm 8.7}$	4.6$_{\pm 2.5}$→79.4$_{\pm 3.9}$	4.6$_{\pm 2.5}$→81.6$_{\pm 5.0}$	**77.6**$_{\pm 6.9}$→88.6$_{\pm 1.5}$	**77.6**$_{\pm 6.9}$→**93.2**$_{\pm 1.6}$
	can	59.6$_{\pm 12.8}$→82.0$_{\pm 2.4}$	83.0$_{\pm 2.7}$→90.6$_{\pm 3.2}$	31.2$_{\pm 4.1}$→58.4$_{\pm 7.5}$	**88.0**$_{\pm 3.3}$→**94.4**$_{\pm 1.8}$	46.0$_{\pm 7.1}$→83.6$_{\pm 5.2}$	46.0$_{\pm 7.1}$→86.2$_{\pm 5.6}$	25.8$_{\pm 4.1}$→90.4$_{\pm 4.5}$	25.8$_{\pm 4.1}$→90.6$_{\pm 3.2}$
Cube-double	task2	75.6$_{\pm 7.4}$→86.0$_{\pm 4.7}$	**77.6**$_{\pm 4.7}$→99.8$_{\pm 0.4}$	27.2$_{\pm 10.4}$→93.2$_{\pm 7.8}$	39.6$_{\pm 9.7}$→**100.0**$_{\pm 0.0}$	34.6$_{\pm 9.8}$→98.4$_{\pm 1.3}$	34.6$_{\pm 9.8}$→99.6$_{\pm 0.5}$	53.2$_{\pm 5.8}$→99.2$_{\pm 0.8}$	53.2$_{\pm 5.8}$→**100.0**$_{\pm 0.0}$
	task3	**79.6**$_{\pm 6.8}$→88.8$_{\pm 5.2}$	76.0$_{\pm 4.7}$→**99.8**$_{\pm 0.4}$	26.8$_{\pm 9.3}$→91.2$_{\pm 4.8}$	40.2$_{\pm 6.2}$→**99.8**$_{\pm 0.4}$	37.8$_{\pm 10.8}$→98.6$_{\pm 1.1}$	37.8$_{\pm 10.8}$→98.8$_{\pm 0.8}$	58.8$_{\pm 5.0}$→99.6$_{\pm 0.9}$	58.8$_{\pm 5.0}$→99.6$_{\pm 0.5}$
	task4	18.4$_{\pm 5.0}$→27.2$_{\pm 8.2}$	**23.2**$_{\pm 5.8}$→92.6$_{\pm 5.7}$	4.0$_{\pm 3.2}$→6.0$_{\pm 6.3}$	9.4$_{\pm 3.6}$→**99.8**$_{\pm 0.4}$	15.0$_{\pm 7.0}$→94.8$_{\pm 4.3}$	15.0$_{\pm 7.0}$→96.6$_{\pm 0.5}$	8.8$_{\pm 2.6}$→96.0$_{\pm 2.8}$	8.8$_{\pm 2.6}$→99.6$_{\pm 0.5}$
Cube-triple	task2	0.4$_{\pm 0.9}$→7.6$_{\pm 8.5}$	0.6$_{\pm 0.9}$→87.4$_{\pm 9.8}$	0.4$_{\pm 0.9}$→0.4$_{\pm 0.9}$	0.0$_{\pm 0.0}$→88.2$_{\pm 2.2}$	0.4$_{\pm 0.5}$→86.2$_{\pm 4.4}$	0.4$_{\pm 0.5}$→78.0$_{\pm 15.4}$	**3.2**$_{\pm 2.5}$→91.4$_{\pm 3.4}$	**3.2**$_{\pm 2.5}$→**98.4**$_{\pm 1.1}$
	task3	2.0$_{\pm 2.0}$→6.8$_{\pm 3.0}$	0.8$_{\pm 0.8}$→80.8$_{\pm 3.8}$	0.4$_{\pm 0.9}$→6.4$_{\pm 8.2}$	0.0$_{\pm 0.0}$→60.4$_{\pm 12.3}$	4.2$_{\pm 2.9}$→57.2$_{\pm 10.5}$	4.2$_{\pm 2.9}$→60.6$_{\pm 12.6}$	**7.6**$_{\pm 3.5}$→83.2$_{\pm 4.1}$	**7.6**$_{\pm 3.5}$→**91.6**$_{\pm 1.8}$
	task4	0.0$_{\pm 0.0}$→0.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$→33.4$_{\pm 9.4}$	0.0$_{\pm 0.0}$→0.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$→51.4$_{\pm 24.2}$	**1.2**$_{\pm 0.8}$→31.0$_{\pm 20.3}$	**1.2**$_{\pm 0.8}$→30.4$_{\pm 10.9}$	1.0$_{\pm 0.7}$→31.2$_{\pm 6.5}$	1.0$_{\pm 0.7}$→**81.2**$_{\pm 5.6}$
Cube-quad	task2	0.0$_{\pm 0.0}$→32.4$_{\pm 21.0}$	0.0$_{\pm 0.0}$→95.8$_{\pm 2.3}$	0.0$_{\pm 0.0}$→0.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$→98.0$_{\pm 2.0}$	0.0$_{\pm 0.0}$→96.6$_{\pm 1.5}$	0.0$_{\pm 0.0}$→97.2$_{\pm 1.5}$	**0.2**$_{\pm 0.4}$→97.6$_{\pm 1.3}$	**0.2**$_{\pm 0.4}$→**99.6**$_{\pm 0.9}$
	task3	0.0$_{\pm 0.0}$→0.0$_{\pm 0.0}$	1.6$_{\pm 1.8}$→63.2$_{\pm 9.7}$	0.0$_{\pm 0.0}$→0.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$→85.0$_{\pm 6.9}$	1.2$_{\pm 2.2}$→47.2$_{\pm 29.8}$	1.2$_{\pm 2.2}$→69.2$_{\pm 14.8}$	**5.6**$_{\pm 2.5}$→88.8$_{\pm 3.2}$	**5.6**$_{\pm 2.5}$→**96.6**$_{\pm 1.9}$
	task4	0.0$_{\pm 0.0}$→0.0$_{\pm 0.0}$	**0.4**$_{\pm 0.5}$→74.2$_{\pm 10.9}$	0.0$_{\pm 0.0}$→0.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$→92.2$_{\pm 7.0}$	0.0$_{\pm 0.0}$→91.2$_{\pm 2.4}$	0.0$_{\pm 0.0}$→94.2$_{\pm 4.1}$	0.2$_{\pm 0.4}$→95.2$_{\pm 3.2}$	0.2$_{\pm 0.4}$→**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**

Tab. 8 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 explicit, isolating how effectively each parameterization absorbs buffer updates during fine-tuning. Consistent with the analysis in Sec. 5, **DFP** shows 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 More Results of Backbone × $\mathcal{L}_{\text{top-}K}$ Ablation

Figure 4 shows the training curves of the backbone × loss ablation on the `cube-triple` tasks. 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 on these long-horizon tasks.

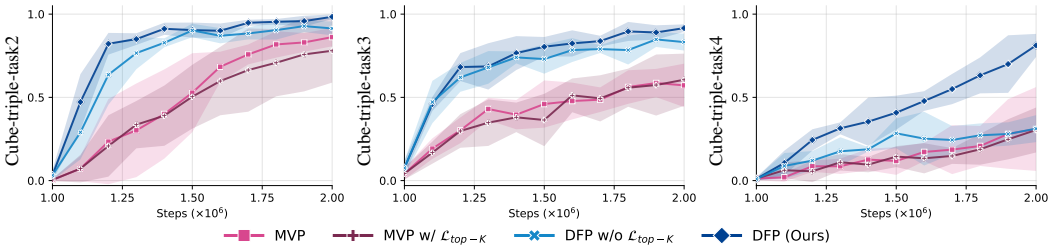


Figure 4: **Online training curves for the backbone × loss ablation.** Success rate over the online phase, comparing MVP [67], MVP w/ $\mathcal{L}_{\text{top-}K}$, DFP w/o $\mathcal{L}_{\text{top-}K}$, and DFP.

B.3 More Results of λ Ablation

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

B.4 More Results of K Ablation

Fig. 6 shows the training curves for $K \in \{1, 2, 4, 8\}$ in $\mathcal{L}_{\text{top-}K}$, with all other components fixed.

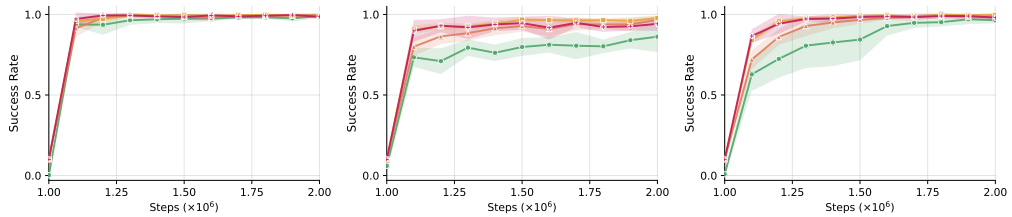


Figure 5: **Online training curves for different λ values.** Success rate over the online phase.

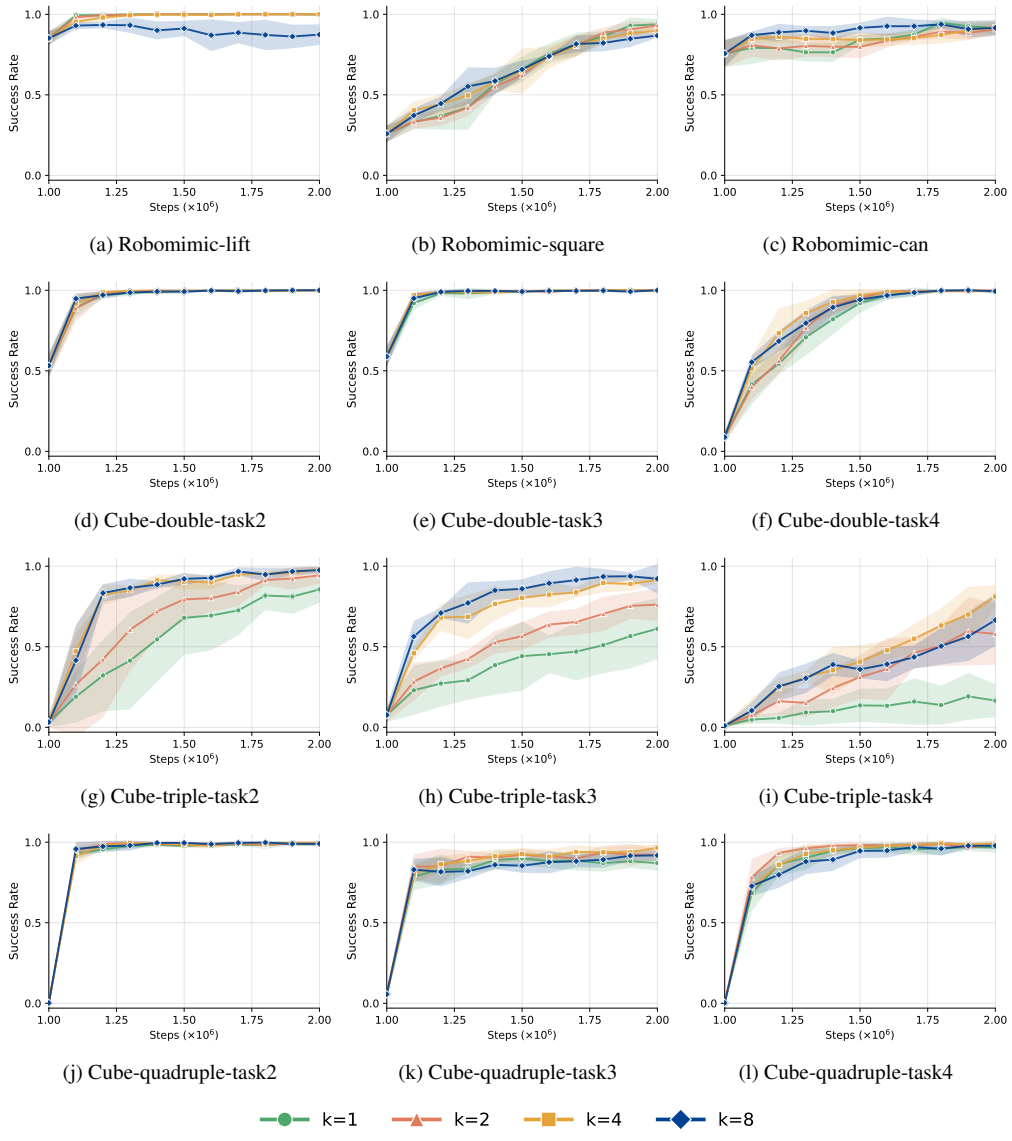


Figure 6: **Online training curves for the top-K ablation.** Success rate over the online phase on Robomimic (top row) and the OGBench.

B.5 More Results of N' Ablation

Tab. 9 reports the per-task results summarized in Tab. 6. At evaluation with $N' = 1$, DFP outperforms MVP on all nine tasks, and MVP stays at or below 20.8% on all six `cube-triple` and `cube-quadruple` tasks.

Table 9: **Per-task results for the number of best-of-N' candidates at evaluation.** Success rate (%) on the OGBench tasks, `mean` $\pm$ `std` over 5 seeds. Only the evaluation-time N' changes; online data collection in Algorithm 1 uses $N' = 16$.

N'	1	2	4	8	16	32
DFP (Ours)						
`cube-double-task2`	99.0 ± 1	99.0 ± 0	100.0 ± 0	100.0 ± 0	100.0 ± 0	100.0 ± 0
`cube-double-task3`	98.6 ± 2	99.8 ± 0	100.0 ± 0	100.0 ± 0	100.0 ± 0	100.0 ± 0
`cube-double-task4`	95.2 ± 2	99.6 ± 0	99.6 ± 0	100.0 ± 0	100.0 ± 0	99.8 ± 0
`cube-triple-task2`	79.0 ± 7	95.8 ± 3	97.6 ± 1	97.2 ± 3	97.8 ± 1	95.8 ± 2
`cube-triple-task3`	47.8 ± 6	84.0 ± 6	93.4 ± 4	89.0 ± 4	89.2 ± 4	89.6 ± 2
`cube-triple-task4`	29.0 ± 10	72.8 ± 8	83.6 ± 7	83.2 ± 3	79.8 ± 3	76.8 ± 6
`cube-quadruple-task2`	74.0 ± 7	98.8 ± 1	99.4 ± 0	99.4 ± 0	99.4 ± 1	99.4 ± 0
`cube-quadruple-task3`	53.0 ± 1	93.0 ± 1	97.6 ± 2	97.2 ± 2	95.8 ± 2	94.6 ± 2
`cube-quadruple-task4`	74.4 ± 5	97.6 ± 1	99.0 ± 1	99.8 ± 0	98.8 ± 1	97.8 ± 2
Avg.	**72.2**	**93.4**	**96.7**	**96.2**	**95.6**	**94.9**
MVP [67]						
`cube-double-task2`	64.6 ± 4	93.0 ± 3	97.2 ± 2	99.0 ± 1	98.4 ± 1	99.0 ± 1
`cube-double-task3`	70.2 ± 6	94.4 ± 2	98.0 ± 2	100.0 ± 0	99.0 ± 1	99.4 ± 0
`cube-double-task4`	10.6 ± 9	64.2 ± 17	87.4 ± 8	92.4 ± 4	92.8 ± 5	94.2 ± 2
`cube-triple-task2`	4.2 ± 3	49.0 ± 7	78.6 ± 7	84.0 ± 2	86.8 ± 2	82.8 ± 3
`cube-triple-task3`	0.4 ± 0	15.6 ± 8	41.0 ± 9	43.8 ± 6	61.0 ± 8	58.8 ± 5
`cube-triple-task4`	0.0 ± 0	1.6 ± 2	11.6 ± 12	21.4 ± 13	25.8 ± 17	27.2 ± 15
`cube-quadruple-task2`	20.8 ± 3	74.4 ± 3	92.2 ± 3	94.0 ± 3	97.0 ± 1	97.4 ± 2
`cube-quadruple-task3`	0.4 ± 1	15.2 ± 16	33.6 ± 23	47.4 ± 25	46.8 ± 25	54.6 ± 24
`cube-quadruple-task4`	5.8 ± 3	56.8 ± 8	78.6 ± 6	87.8 ± 5	90.4 ± 4	92.4 ± 3
Avg.	19.7	51.6	68.7	74.4	77.6	78.4

B.6 (N, K) Ablation

We vary the number of candidates N for $\mathcal{L}_{\text{top-}K}$ jointly with K on the `cube-quadruple` tasks, keeping all other hyperparameters at their defaults (Tab. 10). Across the (N, K) pairs, the average success rate stays between 92.1% and 98.7%, compared with 98.4% for the default $(16, 4)$.

Table 10: (N, K) **ablation.** Success rate (%) on the `cube-quadruple` tasks, `mean` $\pm$ `std` over 5 seeds; **Avg.** is the mean over the three tasks. N is the number of candidates drawn from π_{old} for $\mathcal{L}_{\text{top-}K}$; all other hyperparameters are at their defaults. The default $(16, 4)$ row is from Tab. 1. **Bold**: best per column.

(N, K)	`task2`	`task3`	`task4`	**Avg.**
$(2, 1)$	99.0 ± 1	93.8 ± 3	96.0 ± 2	96.3
$(4, 1)$	98.8 ± 0	93.2 ± 2	96.8 ± 2	96.3
$(4, 2)$	99.0 ± 1	91.0 ± 3	96.8 ± 2	95.6
$(8, 1)$	99.2 ± 1	94.8 ± 1	98.0 ± 1	97.3
$(8, 2)$	99.2 ± 1	94.6 ± 2	98.4 ± 1	97.4
$(8, 4)$	99.0 ± 1	92.4 ± 2	98.0 ± 1	96.5
$(16, 4)$, default	99.6 ± 1	**96.6 ± 2**	99.0 ± 2	98.4
$(32, 1)$	96.2 ± 3	84.8 ± 3	95.4 ± 2	92.1
$(32, 2)$	98.4 ± 2	89.8 ± 3	96.4 ± 2	94.9
$(32, 4)$	99.6 ± 0	96.2 ± 3	99.2 ± 1	98.3
$(32, 8)$	**100.0 ± 0**	96.4 ± 1	**99.6 ± 1**	**98.7**

B.7 Adaptation to Offline RL

While our main experiments in Sec. 5 follow the offline-to-online RL setup of MVP [67], 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 $\mathcal{L}_{\mathrm{BC}}(\theta) + \lambda\, \mathcal{L}_{\text{top-}K}(\theta)$ is applied throughout training.

Offline RL Benchmarks. Following prior offline RL work [46], we evaluate on the 5 robot-manipulation environments from OGBench [45], 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 [46].

Offline RL Baselines. We compare against three categories of baselines: (i) *Gaussian policy*: Behavior Cloning (BC), Implicit Q-Learning (IQL) [30], and ReBRAC [60]; (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 [43] and BRAC [65] respectively, and Flow Q-Learning with its iterative form (FQL, IFQL) [46].

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. 11 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 11: **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	$3_{\pm 1}$	$85_{\pm 8}$	$92_{\pm 4}$	$96_{\pm 2}$	$82_{\pm 16}$	$80_{\pm 30}$	$81_{\pm 9}$	$83_{\pm 13}$	$73_{\pm 3}$	**$97_{\pm 2}$**	$95_{\pm 3}$
cube-double-task2	$0_{\pm 0}$	$1_{\pm 1}$	$7_{\pm 3}$	$16_{\pm 10}$	$0_{\pm 0}$	$2_{\pm 2}$	$2_{\pm 1}$	$22_{\pm 12}$	$9_{\pm 5}$	$36_{\pm 6}$	**$41_{\pm 4}$**
scene-task2	$1_{\pm 1}$	$12_{\pm 3}$	$50_{\pm 13}$	$33_{\pm 14}$	$2_{\pm 2}$	$50_{\pm 40}$	$18_{\pm 8}$	$46_{\pm 10}$	$0_{\pm 0}$	$76_{\pm 9}$	**$93_{\pm 4}$**
puzzle-3x3-task4	$1_{\pm 1}$	$2_{\pm 1}$	$2_{\pm 1}$	$0_{\pm 0}$	$0_{\pm 0}$	$0_{\pm 0}$	$1_{\pm 1}$	$2_{\pm 2}$	$0_{\pm 0}$	**$16_{\pm 5}$**	$3_{\pm 2}$
puzzle-4x4-task4	$0_{\pm 0}$	$4_{\pm 1}$	$10_{\pm 3}$	**$26_{\pm 6}$**	$7_{\pm 4}$	$1_{\pm 1}$	$0_{\pm 0}$	$5_{\pm 1}$	$21_{\pm 11}$	$11_{\pm 3}$	$20_{\pm 2}$

Hyperparameters. We list the offline-specific hyperparameters in Tab. 12. 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. 7

Table 12: **Task-specific hyperparameters for DFP offline training.**

Hyperparameter	cube-single task2	cube-double task2	scene task2	puzzle-3x3 task4	puzzle-4x4 task4
Top-K loss 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.8 Fully Online RL without $\mathcal{L}_{\mathrm{BC}}$

To test whether DFP remains viable without the behavior-cloning anchor, we remove $\mathcal{L}_{\mathrm{BC}}$ and train DFP from scratch in the fully online setting on three MuJoCo tasks, following the protocol and codebase of Ma et al. [39]: 1M environment steps with 200K gradient steps. The candidates for $\mathcal{L}_{\text{top-}K}$ are drawn from the slowly updated π_{old}, which keeps successive targets within the support of the preceding policy. Unless stated in Tab. 13, we use $N = 16$, $K = 2$, $N_{\mathrm{gen}} = 8$, $\tau_{\mathrm{EMA}} = 10^{-3}$, Gaussian action noise with scale 0.1, and a reward scale of 0.2. DFP exceeds SDAC on Hopper and lies between SAC and SDAC on Walker2d. These results show that DFP can learn without $\mathcal{L}_{\mathrm{BC}}$; a systematic study of the fully online regime is left to future work.

Table 13: **Fully online RL without $\mathcal{L}_{\mathrm{BC}}$ on MuJoCo.** Return after 1M environment steps, `mean` $\pm$ `std`. DFP uses $h = 0.1$ for HalfCheetah, $h = 0.05$ for Hopper, and $h = 0.15$ for Walker2d. All baseline results are taken from Ma et al. [39]. **Bold**: best per column.

	Method	HalfCheetah-v4	Hopper-v4	Walker2d-v4
Classic RL	PPO [54]	4852 ± 732	3227 ± 164	4114 ± 806
	TD3 [16]	8149 ± 688	1934 ± 1079	2476 ± 1357
	SAC [20]	8981 ± 370	3197 ± 294	3233 ± 871
Diffusion policy	QSM	10740 ± 444	2804 ± 466	2523 ± 872
	DIPO	9063 ± 654	1191 ± 770	1961 ± 1509
	DACER	11203 ± 246	3212 ± 86	3194 ± 1822
	QVPO [11]	7321 ± 1087	2873 ± 607	2337 ± 1215
	DPPO [51]	1173 ± 392	2175 ± 556	1130 ± 686
	DPMD [39]	11924 ± 609	$\mathbf{3275 \pm 55}$	$\mathbf{4365 \pm 266}$
	SDAC [39]	$\mathbf{12210 \pm 964}$	2955 ± 370	3995 ± 498
Drifting policy	**DFP**	10224 ± 987	3121 ± 112	3832 ± 139

B.9 Training and Inference Cost

We measure per-step wall-clock cost for online training and inference, averaged across the 12 benchmark tasks (Tab. 14); inference cost is measured on CPU following the protocol of MVP [67]. 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 evaluation, DFP generates its N' candidates in one batched forward pass of f_θ and selects one with the critic, which keeps its cost in the same regime as the other one-step baselines FQL, QC-FQL, and MVP, all roughly an order of magnitude faster than BFN and QC-BFN. The evaluation cost is measured with each method's evaluation-time action-selection code, including batch-level best-of-N selection. The gains reported in Sec. 5 thus come at negligible training overhead and at an evaluation cost comparable to that of the other one-step policies.

Table 14: **Training and inference cost.** Mean wall-clock time (ms) per online training step and per evaluation step, averaged over the 12 tasks. Evaluation is timed on CPU following MVP [67] and includes batch-level best-of-N selection, from generating the candidates to scoring them with the critic.

	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

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. We assume that the action space $\mathcal{A}$ is compact and that the kernel k of Sec. 2.2 is bounded and Lipschitz, as is the Gaussian kernel. 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. We further assume that $Q_\phi(s,A)$ has a strictly positive density at $q^\rho(s)$. Then 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 $Q_\phi(s,a^{(j)}) \ge \hat{q}_N^\rho(s)$. Since $\hat{q}_N^\rho(s) \to q^\rho(s)$, $Q_\phi(s,A)$ has no atom at $q^\rho(s)$, and the level set $\{a : Q_\phi(s,a) \ge q^\rho(s)\}$ has π_{old}-mass ρ, a standard truncation argument gives $P_K(s) \to \tilde{\pi}^\rho(\cdot|s)$ weakly, and hence in W_2 on the bounded action space, almost surely as $N \to \infty$. In general, this convergence does not hold in total variation: whenever $\tilde{\pi}^\rho(\cdot|s)$ has no atoms, e.g., when $\pi_{\text{old}}(\cdot|s)$ has a density, $\text{TV}(P_K(s),\tilde{\pi}^\rho(\cdot|s)) = 1$ for every N because $P_K(s)$ is discrete. We therefore pass to the limit after applying the kernel mean shift. Write $\mathbf{V}_p^+(x) = N_x(p)/D_x(p)$ with $N_x(p) := \int k(x,y)(y-x)\,dp(y)$ and $D_x(p) := \int k(x,y)\,dp(y)$. For fixed $x \in \mathcal{A}$, the integrands $y \mapsto k(x,y)(y-x)$ and $y \mapsto k(x,y)$ are bounded and continuous on $\mathcal{A}$, so weak convergence gives $N_x(P_K) \to N_x(\tilde{\pi}^\rho)$ and $D_x(P_K) \to D_x(\tilde{\pi}^\rho)$. Because k is positive and continuous on the compact set $\mathcal{A}\times\mathcal{A}$, it is bounded below by $k_{\min} := \min_{x,y\in\mathcal{A}} k(x,y) > 0$, so $D_x(p) \ge k_{\min}$ for every probability measure p on $\mathcal{A}$. Hence $\mathbf{V}^+_{P_K}(x) \to \mathbf{V}^+_{\tilde{\pi}^\rho}(x)$ for every $x \in \mathcal{A}$. The negative side $\mathbf{V}^-_{\pi_\theta}$ is unchanged, so $\mathbf{V}_{P_K,\pi_\theta} \to \mathbf{V}_{\tilde{\pi}^\rho,\pi_\theta}$ pointwise. Every kernel mean shift is a weighted average of $y - x$ with $x, y \in \mathcal{A}$ and is thus bounded by $\text{diam}(\mathcal{A})$, so the integrand of $\mathcal{L}_{\text{drift}}$ in Eq. (4) is bounded, and dominated convergence yields $\mathcal{L}_{\text{top-}K}(\theta) \to \mathcal{L}_{\text{drift}}(\theta;\,\tilde{\pi}^\rho,\pi_\theta)$.

For part (ii), we use the Lipschitz continuity of the kernel mean shift in total variation, which applies between the population distributions $\tilde{\pi}^\rho$ and π^+. Using $\left\|\int f\,(dp-dq)\right\| \le 2\|f\|_\infty \,\text{TV}(p,q)$ for N_x and D_x, with $\|k(x,\cdot)(\cdot - x)\|_\infty \le K_{\max}\,\text{diam}(\mathcal{A})$, $\|k(x,\cdot)\|_\infty \le K_{\max}$, and $D_x(\cdot) \ge k_{\min}$, the standard quotient identity yields

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

(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),

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

Since the stop-gradient target in Eq. (4) differs from $\hat{a}$ by $\mathbf{V}_{p,q}(\hat{a}|s)$, the value of the loss is $\mathcal{L}_{\text{drift}}(\theta;p,q) = \mathbb{E}\big[\|\mathbf{V}_p^+ - \mathbf{V}_q^-\|^2\big]$, and 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 ($\|\mathbf{V}_p^+\|, \|\mathbf{V}_q^-\| \le M$ with $M = \mathrm{diam}(\mathcal{A})$, since the mean shift is a weighted average of $(y - x)$ with $y, x \in \mathcal{A}$),

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

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

Tightness of the bound. The gap $\overline{\mathrm{TV}}(\tilde{\pi}^\rho, \pi^+)$ depends on both ρ and α and reflects the structural mismatch between hard ρ-quantile truncation and soft Boltzmann tilting $\exp(Q_\phi/\alpha)$. For fixed ρ and α, the two distributions differ in general, so the gap does not vanish and $\mathcal{L}_{\text{top-}K}$ remains a biased surrogate of $\mathcal{L}_{\mathrm{PI}}$. The truncation level ρ plays a role analogous to that of α, since both control how strongly the target concentrates on high-Q_ϕ actions; for a given α, choosing ρ to match α reduces the gap, but the gap is not zero in general. □

Magnitude of C. The constant $C = \mathcal{O}\big(K^2_{\max}\,\mathrm{diam}(\mathcal{A})^2/k^2_{\min}\big)$ is finite for any bounded action space, but it is not small in general: for the Gaussian kernel, $k_{\min} = \exp\big(-\mathrm{diam}(\mathcal{A})^2/(2h^2)\big)$ decays exponentially in $\mathrm{diam}(\mathcal{A})^2/h^2$, so C grows rapidly for small bandwidths and high-dimensional action chunks. Proposition 1 therefore establishes that the bias of $\mathcal{L}_{\text{top-}K}$ is bounded and attributes it to the mismatch between $\tilde{\pi}^\rho$ and π^+, rather than giving a numerically tight estimate of its size. The empirical effect of K and h is reported in Sec. 5.4.

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 15 reports the GPU-hours used for each experiment.

Table 15: **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
Top-K loss weight λ ablation (Table 3)	500h
Offline RL evaluation (Sec. B.7)	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.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Both abstract and introduction reflect contributions in the main paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations discussed in the conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Assumptions stated; full proofs in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the implementation details and hyperparameters for reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code, including the baseline implementations and task configurations, is available at `https://github.com/KAIST-Visual-AI-Group/DFP`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experiment setups and implementation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Means and standard deviations are reported over 5 seeds, except for the per-task MuJoCo configurations in Appendix B.8, which use 3 seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources described in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work fully conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Discussed in the appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No high-risk data or models released.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All used assets are properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: No new assets are released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or human subjects involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects research conducted.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LMs not used in core methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.