CoDrift: Compositional Drifting for Offline Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Offline reinforcement learning is intrinsically multi-objective: a policy must remain compatible with the behavioral support of a fixed dataset while preferentially selecting high-value actions. We recast these objectives in a common form by viewing each as an action-space motion field that specifies how generated actions should move. This perspective enables heterogeneous learning objectives to be combined directly through field composition. Inspired by drifting models, we propose **CoDrift**, a compositional framework for one-step generative policy learning. CoDrift combines three objective-level fields into a unified policy field. The conditional field preserves state-dependent behavioral structure, while the marginal field pools actions across states to provide a more stable generative signal in the single-positive-sample regime of continuous-control offline RL. The value field moves generated actions toward higher-value regions. The composed field is absorbed into a stochastic generator that produces an action with a single forward pass at deployment. We evaluate CoDrift on 73 tasks from OGBench and D4RL in both offline and offline-to-online settings. CoDrift compares favorably with state-of-the-art methods and achieves the best average rank in both settings. Code will be released.

1 Introduction

Offline reinforcement learning (RL) aims to learn effective policies entirely from previously collected experience, without further interaction with the environment (Levine et al., 2020). A central challenge is that policy optimization must satisfy multiple objectives simultaneously. On the one hand, the learned policy should remain compatible with the behavioral support of the offline dataset, since actions far outside the data distribution may lead to unreliable value estimates and poor decisions. On the other hand, merely reproducing the behavior policy is insufficient: the policy must preferentially select high-value actions in order to improve return. Offline RL can therefore be viewed as an intrinsically multi-objective policy-learning problem, balancing behavioral fidelity with value maximization.

Generative policies provide a powerful way to model the behavioral side of this problem. In particular, diffusion-based policies have demonstrated good capability to represent complex and multimodal action distributions (Wang et al., 2023; Hansen-Estruch et al., 2023), while flow-matching approaches provide an even stronger generative policy class (Park et al., 2025b). These successes suggest that expressive generative modeling is a natural foundation for offline policy learning. We build on this perspective, but take a different view of how the multiple objectives of offline RL should be incorporated into a generative policy.

Our starting point is simple: different learning objectives can all be interpreted as specifying how a generated action should move in action space. A behavioral objective moves generated actions toward regions supported by the offline data, whereas a value objective moves them toward directions of higher predicted return. Once expressed as action-space displacements, these heterogeneous objectives share a common representation and can thereby be composed naturally. This motivates the central principle of our approach: offline policy learning can be formulated as the *composition of action-space motion fields*, with each field corresponding to a distinct learning objective.

Our formulation is inspired by drifting models (Deng et al., 2026), a recent class of generative models that learns stochastic generators from displacement fields acting directly on generated samples.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In the original drifting formulation, a displacement field is constructed from two elementary components: attraction toward positive data samples and repulsion from negative generated samples. For our setting, it suggests a broader principle: while the original attraction and repulsion components jointly realize a single distribution-matching objective, different learning objectives can themselves be represented as displacement fields and composed directly in action space.

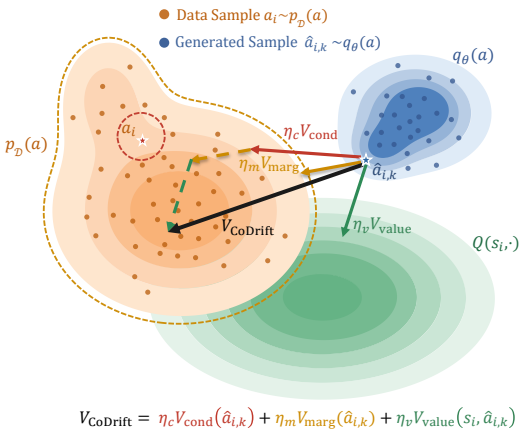

Figure 1: **Overview of CoDrift.** Conditional and marginal behavioral fields keep generated actions within the support of the offline data at two statistical scales, while the value field moves them toward higher-return regions. The three are composed additively into a single policy field V_{CoDrift}.

Based on this principle, we propose **CoDrift**, a compositional drifting framework for offline RL. CoDrift instantiates three complementary action-space fields as illustrated in Figure 1. First, a **conditional behavioral field** preserves state-dependent behavioral support by attracting candidate actions toward actions observed at the corresponding states while maintaining stochasticity through repulsion among generated actions. Second, a **marginal behavioral field** captures global structure in the action distribution by pooling data and generated actions across states within a minibatch. This marginal field is particularly useful in continuous-control offline RL, where essentially every state is paired with only a single observed action. Consequently, the conditional field is estimated from a single Monte Carlo sample at each state, whereas marginal field can exploit multiple actions across the batch, yielding a more stable, lower-variance generative signal. Finally, a **value field** pushes generated actions toward higher-value regions using action gradients from learned critics. These fields play complementary roles and are combined additively into a single generative policy field.

Our contributions are summarized as follows:

- We introduce CoDrift, a compositional drifting framework for generative policy learning in offline RL, where heterogeneous learning objectives are represented as additive action-space fields. CoDrift achieves expressive stochastic generation while requiring only a single forward pass at deployment.
- We identify the single-positive-sample problem in conditional drifting for offline RL. We address this issue with marginal drifting, which pools actions across states to provide a more stable training signal.
- We conduct extensive experiments across 73 tasks from OGBench and D4RL in both offline and offline-to-online settings. CoDrift compares favorably with state-of-the-art approaches, achieving the best average rank in both settings with low runtime cost.

2 Preliminaries

2.1 Offline Reinforcement Learning

We consider a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho, \gamma)$ (Sutton et al., 1998), where $\mathcal{S}$ is the state space, $\mathcal{A} = [-1, 1]^d$ is the continuous action space, P is the transition dynamics, r is the reward function, ρ is the initial-state distribution, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi_\theta(\cdot \mid s)$ is evaluated by its expected discounted return

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]. \tag{1}$$

Offline RL seeks to maximize Eq. 1 using only a fixed dataset $\mathcal{D} = \{(s, a, r, s')\}$ collected by an unknown behavior policy, without further interaction with the environment. Because value

estimates can be unreliable for actions outside the support of the offline data, unconstrained value maximization may drive the policy toward out-of-distribution actions (Levine et al., 2020). Offline policy learning therefore involves two competing objectives: the policy should remain compatible with the behavioral support of the data while favoring actions with high predicted value. We refer to these two requirements as *behavioral fidelity* and *value maximization*, respectively. CoDrift represents these objectives as action-space displacement fields and combines them directly within a generative policy.

2.2 Drifting Models

Drifting models (Deng et al., 2026) are one-step generative models that learn to transform a prior distribution into a target distribution through displacement fields. Let p denote a target distribution on $\mathbb{R}^d$ and $p_\varepsilon = \mathcal{N}(0, I)$ a prior distribution. A generator f_θ maps $\varepsilon \sim p_\varepsilon$ to $x = f_\theta(\varepsilon)$, inducing the pushforward distribution $q = (f_\theta)_{\#} p_\varepsilon$. The goal is to learn f_θ such that q matches p. Unlike diffusion and flow models, which transport samples through multiple intermediate states at inference time, drifting models perform this transport during training, while generation requires only a single forward pass.

The transport direction is specified by a drifting field:

$$V_{p,q}(x) = V_p(x) - V_q(x), \tag{2}$$

where, for a reference distribution $\texttt{ref} \in \{p, q\}$,

$$V_{\texttt{ref}}(x) = \frac{\mathbb{E}_{u\sim\texttt{ref}}\left[\kappa_\tau(x,u)(u-x)\right]}{\mathbb{E}_{u\sim\texttt{ref}}\left[\kappa_\tau(x,u)\right]}. \tag{3}$$

Here $\kappa_\tau(\cdot,\cdot)$ is a positive similarity kernel; throughout this work, we use the Laplace kernel $\kappa_\tau(x,u) = \exp(-\|x-u\|_2/\tau)$. V_p attracts generated samples x toward the target distribution, whereas subtracting V_q induces repulsion among generated samples. When $p = q$, the two terms in Eq. 2 coincide, and the drifting field vanishes.

Training uses the drifting field to construct a displaced target: for $x = f_\theta(\varepsilon)$, define $\tilde{x} = x + V_{p,q}(x)$. Treating $\tilde{x}$ as fixed, the generator is trained by

$$\mathcal{L}_{\text{drift}}(\theta; p, q) = \mathbb{E}_{\varepsilon\sim p_\varepsilon}\left[\|x - \text{sg}[\tilde{x}]\|_2^2\right], \tag{4}$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operator. Repeated regression toward these displaced targets progressively moves q toward p during training.

In practice, the population fields in Eq. 3 are estimated from minibatch samples. For a reference set $\mathcal{Y} = \{u_1, \ldots, u_N\}$, we define

$$V(x; \mathcal{Y}) = \sum_{i=1}^{N} w_i(u_i - x), \qquad w_i = \frac{\kappa_\tau(x, u_i)}{\sum_{j=1}^{N}\kappa_\tau(x, u_j)}. \tag{5}$$

This is the Monte Carlo counterpart of $V_{\texttt{ref}}(x)$. Given positive and negative reference sets $\mathcal{Y}$ and $\mathcal{Z}$, respectively, the minibatch drifting field is therefore written explicitly as $V(x;\mathcal{Y}) - V(x;\mathcal{Z})$ following Eq. 2.

3 Method

We introduce **CoDrift**, a compositional drifting framework for offline reinforcement learning. CoDrift represents different learning objectives as displacement fields acting directly on generated actions and composes them into a single policy field. Specifically, CoDrift contains three objective-level fields: a *conditional behavioral field*, a *marginal behavioral field*, and a *value field*. The first two preserve behavioral structure at complementary statistical scales, while the third pushes generated actions toward higher-value regions. Figure 1 provides an overview.

We parameterize the policy as a one-step stochastic generator f_θ. Given a state s and noise $\varepsilon \sim p_\varepsilon = \mathcal{N}(0, I)$,

$$\hat{a} = f_\theta(s, \varepsilon), \quad \pi_\theta(\cdot \mid s) = \big(f_\theta(s, \cdot)\big)_{\#} p_\varepsilon, \tag{6}$$

where f_θ uses $\tanh$ to ensure that $\hat{a} \in [-1, 1]^d$. Sampling from π_θ therefore requires only one noise draw and one forward pass.

3.1 CONDITIONAL BEHAVIORAL FIELD

The conditional behavioral field preserves the state-dependent action distribution of the offline data. At the population level, it instantiates the drifting field of Sec. 2.2 between the behavioral conditional distribution $p_{\mathcal{D}}(a \mid s)$ and the policy $\pi_\theta(a \mid s)$, where $p_{\mathcal{D}}(a \mid s)$ plays the role of the target distribution p, while $\pi_\theta(a \mid s)$ plays the role of the generated distribution q.

Given a minibatch $\mathcal{B} = \{(s_i, a_i)\}_{i=1}^{B}$, we draw K independent noise samples for each state and generate actions:

$$\varepsilon_{i,k} \sim \mathcal{N}(0, I), \qquad \hat{a}_{i,k} = f_\theta(s_i, \varepsilon_{i,k}), \qquad k = 1, \dots, K.$$

For a generated action $\hat{a}_{i,k}$, the positive and generated reference sets are

$$\mathcal{Y}^{\text{cond}} = \{a_i\}, \qquad \mathcal{Z}^{\text{cond}} = \{\hat{a}_{i,l} : l = 1, \dots, K, \quad l \neq k\}.$$

The conditional field is

$$V_{\text{cond}}(\hat{a}_{i,k}) = V\big(\hat{a}_{i,k}; \mathcal{Y}^{\text{cond}}\big) - V\big(\hat{a}_{i,k}; \mathcal{Z}^{\text{cond}}\big), \tag{7}$$

where V has been defined in Eq. 5. As the set $\mathcal{Y}^{\text{cond}}$ has only one element, the first term reduces to

$$V(\hat{a}_{i,k}, \mathcal{Y}^{\text{cond}}) = a_i - \hat{a}_{i,k}, \tag{8}$$

which attracts the generated action toward the observed action paired with s_i. The second term of Eq. 7 induces repulsion from other generated actions at the same state and can be written as:

$$V(\hat{a}_{i,k}, \mathcal{Z}^{\text{cond}}) = \sum_{l \neq k} w_{kl}^{\text{cond}} (\hat{a}_{i,l} - \hat{a}_{i,k}),$$

where the normalized weights $w_{kl}^{\text{cond}} = \frac{\kappa_\tau(\hat{a}_{i,k}, \hat{a}_{i,l})}{\sum_{j \neq k} \kappa_\tau(\hat{a}_{i,k}, \hat{a}_{i,j})}$.

A distinctive issue arises in continuous-control offline RL. Because the state space is continuous, an exact state is typically observed only once in the offline dataset. Hence $p_{\mathcal{D}}(a \mid s_i)$ is represented by only one observed action. Consequently, the attractive component in Eq. 8 is effectively a one-sample Monte Carlo estimate of the population conditional drifting field, which often has high variance. This motivates a complementary field that can exploit distributional information pooled across states.

3.2 MARGINAL BEHAVIORAL FIELD

The same stochastic generator also induces a marginal action distribution when states are drawn from the offline state distribution. Recall from Eq. 6 that, for a fixed state s, $f_\theta(s, \cdot)$ pushes the noise distribution p_ε forward to the conditional policy distribution $\pi_\theta(\cdot \mid s)$. If we additionally draw $s \sim p_{\mathcal{D}}(s)$, then the joint mapping $(s, \varepsilon) \mapsto f_\theta(s, \varepsilon)$ pushes the product distribution $p_{\mathcal{D}}(s)p_\varepsilon(\varepsilon)$ forward to the policy-induced marginal action distribution

$$q_\theta = (f_\theta)_{\#}\big(p_{\mathcal{D}} \otimes p_\varepsilon\big),$$

where $\otimes$ denotes the product distribution. Equivalently, $q_\theta(a) = \int p_{\mathcal{D}}(s)\, \pi_\theta(a \mid s)\, ds$. The corresponding marginal action distribution in the offline data is $p_{\mathcal{D}}(a) = \int p_{\mathcal{D}}(s)\, p_{\mathcal{D}}(a \mid s)\, ds$.

Marginal matching is naturally connected to conditional behavioral matching. If $\pi_\theta(a \mid s) = p_{\mathcal{D}}(a \mid s)$, then marginalizing over states gives $q_\theta(a) = p_{\mathcal{D}}(a)$. Therefore, matching the marginal action distribution is a necessary condition for matching the full conditional behavioral distribution.

Unlike the conditional distribution, however, the marginal action distribution is much better sampled within a minibatch. For each generated action $\hat{a}_{i,k}$, we pool offline actions and generated actions across states:

$$\mathcal{Y}^{\text{marg}} = \{a_1, \dots, a_B\}, \qquad \mathcal{Z}^{\text{marg}} = \{\hat{a}_{j,k} : j = 1, \dots, B, \quad j \neq i\}.$$

Then the marginal behavioral field acting on $\hat{a}_{i,k}$ is

$$V_{\text{marg}}(\hat{a}_{i,k}) = V(\hat{a}_{i,k}; \mathcal{Y}^{\text{marg}}) - V(\hat{a}_{i,k}; \mathcal{Z}^{\text{marg}}). \tag{9}$$

The first term uses all B offline actions in the minibatch as positive references, while the second uses generated actions from other states as negative references.

The key difference from conditional drifting is that state–action pairing is deliberately ignored at the marginal level. States used to generate policy actions are sampled from $p_{\mathcal{D}}(s)$, and the actions in the same minibatch are pooled as positive references for marginal matching, irrespective of their associated states. Therefore, instead of estimating a conditional expectation from the single action observed at one state ($\mathcal{Y}^{\text{cond}}$ in Eq. 7), marginal drifting is able to estimate the corresponding distributional field using B positive samples in $\mathcal{Y}^{\text{marg}}$. Consequently, it provides a more stable, lower-variance generative signal while constraining a distribution that must also match whenever the full conditional behavior is matched. The conditional and marginal fields are thus complementary: the former preserves state–action correspondence, whereas the latter supplies population-level action-distribution information.

3.3 Value Field

Behavioral matching alone does not solve offline RL. Among actions consistent with the offline data, the policy should favor those with higher expected return. We therefore represent value maximization as an additional action-space field.

We maintain two critics, Q_{φ_1} and Q_{φ_2}, together with target networks $Q_{\bar{\varphi}_1}$ and $Q_{\bar{\varphi}_2}$ obtained through Polyak averaging (Fujimoto et al., 2018). For a transition $(s, a, r, s') \sim \mathcal{D}$, we sample $a' = f_\theta(s', \varepsilon')$, $\varepsilon' \sim \mathcal{N}(0, I)$, and construct the Bellman target

$$y = r + \gamma \frac{Q_{\bar{\varphi}_1}(s', a') + Q_{\bar{\varphi}_2}(s', a')}{2}.$$

The critics are trained by minimizing

$$\mathcal{L}_{\text{critic}}(\varphi_1, \varphi_2) = \mathbb{E}\left[\frac{1}{2}\sum_{n=1}^{2}\left(Q_{\varphi_n}(s, a) - y\right)^2\right].$$

When updating the actor, the critics are held fixed. The value field is simply the action gradient of the critics, $V_{\text{value}}(s, a) = \left(\nabla_a Q_{\varphi_1}(s, a) + \nabla_a Q_{\varphi_2}(s, a)\right)/2$, which points locally toward actions with higher predicted value.

Unlike the conditional and marginal fields, which arise from distribution matching, the value field is not associated with a reference distribution. Nevertheless, all three objects are displacement fields in the same action space and can therefore be composed directly.

3.4 Compositional Drifting

We now combine the three objective-level fields into a joint policy field. For every generated action $\hat{a}_{i,k}$, we define

$$V_{\text{CoDrift}}(s_i, \hat{a}_{i,k}) = \eta_c V_{\text{cond}}(\hat{a}_{i,k}) + \eta_m V_{\text{marg}}(\hat{a}_{i,k}) + \eta_v V_{\text{value}}(s_i, \hat{a}_{i,k}), \tag{10}$$

where $\eta_c, \eta_m, \eta_v \geq 0$ control the relative contributions of conditional behavioral matching, marginal behavioral matching, and value maximization.

Similar to the training of drifting models in Eq. 4, the joint field directly defines a displaced target for every generated action:

$$\tilde{a}_{i,k} = \hat{a}_{i,k} + V_{\text{CoDrift}}(s_i, \hat{a}_{i,k}).$$

The actor is finally trained with

$$\mathcal{L}_{\text{CoDrift}}(\theta) = \mathbb{E}\left[\frac{1}{BK}\sum_{i=1}^{B}\sum_{k=1}^{K}\|\hat{a}_{i,k} - \text{sg}[\tilde{a}_{i,k}]\|_2^2\right],$$

which learns the one-step stochastic policy by moving toward the composed target.

In our implementation, we parameterize the field weights as

$$\eta_v = 1, \qquad \eta_c = \alpha\lambda, \qquad \eta_m = \alpha(1 - \lambda),$$

where α controls the overall strength of behavioral regularization and $\lambda \in [0, 1]$ balances conditional and marginal behavioral matching.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks. We follow the benchmark suite and evaluation setting of existing works (Park et al., 2025b; Mu, 2026). Specifically, we evaluate on OGBench (Park et al., 2025a), which contains diverse long-horizon continuous-control tasks spanning navigation, locomotion, and manipulation. The datasets of OGBench are collected by task-agnostic policies, and we use their standard single-task variants, where a fixed evaluation goal is specified, and semi-sparse task rewards are relabeled from the original trajectories. We consider ten state-based datasets across six domains: AntMaze, HumanoidMaze, AntSoccer, Cube, Scene, and Puzzle, with five evaluation tasks per dataset, together with five pixel-based visual manipulation tasks. We additionally evaluate on D4RL (Fu et al., 2020), including six AntMaze navigation tasks and twelve Adroit manipulation tasks. Altogether, the offline evaluation comprises 73 tasks spanning different state and action dimensions, observation modalities, reward structures, and control problems. For offline-to-online RL, we follow the same 15-task protocol used by FQL and DeFlow (Park et al., 2025b; Mu, 2026), including five OGBench tasks, six D4RL AntMaze tasks, and four D4RL Adroit tasks. The policy is first trained offline and subsequently fine-tuned using online interactions without changing the learning objective.

Baselines. We compare CoDrift against eleven representative offline RL methods spanning several policy classes. Gaussian-policy baselines include behavior cloning (BC), Implicit Q-Learning (IQL; Kostrikov et al., 2022), and ReBRAC (Tarasov et al., 2023). Diffusion-based baselines include IDQL (Hansen-Estruch et al., 2023), SRPO (Chen et al., 2024), and Consistency Actor-Critic (CAC; Ding & Jin, 2024). Flow-based baselines include FQL (Park et al., 2025b), DeFlow (Mu, 2026), and three flow-policy variants introduced as baselines by Park et al. (2025b): FAWAC, FBRAC, and IFQL, which are flow counterparts of AWAC (Nair et al., 2020), Diffusion-QL (Wang et al., 2023), and IDQL (Hansen-Estruch et al., 2023), respectively. In the offline-to-online setting, we additionally compare with Cal-QL (Nakamoto et al., 2023) and RLPD (Ball et al., 2023). Baseline results are taken from prior work under the same benchmark protocols.

Evaluation protocol. We follow the evaluation protocol of Park et al. (2025b) to ensure direct comparability with previously reported results. Following Kurenkov & Kolesnikov (2022), we do not select the best-performing evaluation checkpoint, which can introduce selection bias. For OGBench, we report the average over the final three evaluation epochs and measure task success rate. For D4RL, we report the final evaluation epoch, using success rate for AntMaze and normalized return for Adroit (Fu et al., 2020). CoDrift results are averaged over eight random seeds, except for pixel-based tasks, which use four seeds. For the offline-to-online experiments, we report performance after offline pretraining at 10^6 steps and after the subsequent online fine-tuning phase at 2×10^6 total steps.

Following prior work (Park et al., 2025b; Mu, 2026), values within 95% of the best result in each task category are highlighted as near-best. For average rank, where lower is better, only the best result is highlighted.

4.2 COMPARISON WITH THE STATE OF THE ARTS

Offline RL performance. Table 1 summarizes the offline RL results. Across the full benchmark suite, CoDrift achieves the best overall average rank among the compared methods, indicating strong performance across diverse task domains. CoDrift performs particularly strongly on OGBench tasks, while remaining competitive on D4RL. These results show that the proposed compositional drifting formulation provides a strong alternative to existing Gaussian-, diffusion-, and flow-based policy classes across a diverse collection of offline RL problems.

Offline-to-online performance. Table 2 reports the offline-to-online results on the 15 fine-tuning tasks. CoDrift can be fine-tuned online without introducing a separate online-stage objective: newly collected transitions are added to the replay buffer, and training proceeds using the same approach as in the offline phase. Across the 15 tasks, CoDrift attains the best overall average rank among the compared methods. These results show that the same compositional drifting formulation remains effective when transitioning from purely offline training to online policy improvement, without requiring an algorithmic change between the two phases.

Table 1: **Offline RL results.** CoDrift is evaluated against representative Gaussian-, diffusion-, and flow-based offline RL methods across OGBench and D4RL. Results for CoDrift are averaged over 8 seeds (4 for pixel-based tasks), and baseline values are taken from prior work (Tarasov et al., 2023; Hansen-Estruch et al., 2023; Chen et al., 2024; Park et al., 2025b; Mu, 2026). The pixel-based row is excluded from ranking as several baselines are unavailable. Lower average rank is better. `M`, `L`, and `G` denote medium, large, and giant, respectively.

	Gaussian			Diffusion			Flow					Drift
Task Category	BC	IQL	ReBRAC	IDQL	SRPO	CAC	FAWAC	FBRAC	IFQL	FQL	DeFlow	CoDrift
OGBench antmaze-L	11 ±1	53 ±3	**81** ±5	21 ±5	11 ±4	33 ±4	6 ±1	60 ±6	28 ±5	79 ±3	**81** ±3	**84** ±3
OGBench antmaze-G	0 ±0	4 ±1	26 ±8	0 ±0	0 ±0	0 ±0	0 ±0	4 ±4	3 ±2	9 ±6	12 ±5	**46** ±22
OGBench humanoidmaze-M	2 ±1	33 ±2	22 ±8	1 ±0	1 ±1	53 ±8	19 ±1	38 ±5	60 ±14	58 ±5	48 ±4	**83** ±5
OGBench humanoidmaze-L	1 ±0	2 ±1	2 ±1	1 ±0	0 ±0	0 ±0	0 ±0	2 ±0	11 ±2	4 ±2	5 ±2	**15** ±5
OGBench antsoccer-arena	1 ±0	8 ±2	0 ±0	12 ±4	1 ±0	2 ±4	12 ±0	16 ±1	33 ±6	60 ±2	**67** ±3	**66** ±5
OGBench visual manipulation	—	42 ±4	60 ±2	—	—	—	—	22 ±2	50 ±5	65 ±2	**69** ±2	61 ±7
OGBench puzzle-3x3	2 ±0	9 ±1	21 ±1	10 ±2	18 ±1	19 ±0	6 ±2	14 ±4	19 ±1	30 ±1	43 ±4	**82** ±13
OGBench puzzle-4x4	0 ±0	7 ±1	14 ±1	**29** ±3	10 ±3	15 ±3	1 ±0	13 ±1	25 ±5	17 ±2	11 ±2	22 ±4
OGBench cube-single	5 ±1	83 ±3	91 ±2	**95** ±2	80 ±5	85 ±9	81 ±4	79 ±7	79 ±2	**96** ±1	**96** ±2	**92** ±3
OGBench cube-double	2 ±1	7 ±1	12 ±1	15 ±6	2 ±1	6 ±2	5 ±2	15 ±3	14 ±3	29 ±2	40 ±5	**49** ±10
OGBench scene	5 ±1	28 ±1	41 ±3	46 ±3	20 ±1	40 ±7	30 ±3	45 ±5	30 ±3	56 ±2	51 ±3	**59** ±2
D4RL antmaze	17	57	78	79	74	30 ±3	44 ±3	64 ±7	65 ±7	**84** ±3	**83** ±2	73 ±6
D4RL adroit	48	53	**59**	52 ±1	51 ±1	43 ±2	48 ±1	50 ±2	52 ±1	52 ±1	52 ±1	50 ±3
Average rank (↓)	10.96	7.38	5.29	6.21	9.42	8.04	9.79	6.58	5.58	3.08	3.13	**2.54**

Table 2: **Offline-to-online RL results.** Performance before and after online fine-tuning on the 15-task benchmark suite.

Task	IQL	ReBRAC	Cal-QL	RLPD	IFQL	FQL	DeFlow	CoDrift
humanoidmaze-M-nav	21 ±13 → 16 ±8	16 ±20 → 1 ±1	0 ±0 → 0 ±0	0 ±0 → 8 ±10	56 ±35 → 82 ±20	12 ±7 → 22 ±12	13 ±5 → 65 ±12	94 ±5 → **99** ±1
antsoccer-arena-nav	2 ±1 → 0 ±0	0 ±0 → 0 ±0	0 ±0 → 0 ±0	0 ±0 → 0 ±0	26 ±15 → 39 ±10	28 ±8 → **86** ±5	44 ±8 → **86** ±3	52 ±7 → **86** ±5
cube-double-play	0 ±1 → 0 ±0	6 ±5 → 28 ±28	0 ±0 → 0 ±0	0 ±0 → 0 ±0	12 ±9 → 40 ±5	40 ±11 → 92 ±3	48 ±13 → 93 ±4	65 ±10 → **100** ±0
scene-play	14 ±11 → 10 ±9	55 ±10 → **100** ±0	1 ±2 → 50 ±53	0 ±0 → **100** ±0	0 ±1 → 60 ±39	82 ±11 → **100** ±1	58 ±14 → **100** ±1	92 ±6 → **100** ±0
puzzle-4x4-play	5 ±2 → 1 ±1	8 ±4 → 14 ±35	0 ±0 → 0 ±0	0 ±0 → **100** ±1	23 ±6 → 19 ±33	8 ±3 → 38 ±52	4 ±4 → **100** ±0	12 ±5 → 88 ±32
antmaze-umaze-v2	77 → **96**	98 → 75	77 → **100**	0 ±0 → **98** ±3	94 ±5 → **96** ±2	97 ±2 → **99** ±1	96 ±3 → **99** ±2	90 ±5 → **99** ±1
antmaze-umaze-div-v2	60 → 64	74 → **98**	32 → **98**	0 ±0 → 94 ±5	69 ±20 → 93 ±5	79 ±16 → **100** ±1	87 ±6 → **99** ±1	66 ±13 → **98** ±2
antmaze-M-play-v2	72 → 90	88 → **98**	72 → **99**	0 ±0 → **98** ±2	52 ±19 → 93 ±2	77 ±7 → **97** ±2	76 ±5 → **97** ±2	60 ±27 → **98** ±2
antmaze-M-div-v2	64 → 92	85 → **99**	62 → **98**	0 ±0 → **97** ±2	44 ±26 → 89 ±4	55 ±19 → **97** ±3	61 ±9 → **98** ±2	59 ±15 → **97** ±2
antmaze-L-play-v2	38 → 64	68 → 32	32 → **97**	0 ±0 → **93** ±5	64 ±14 → 80 ±5	66 ±40 → 84 ±30	76 ±6 → **95** ±4	41 ±17 → 92 ±3
antmaze-L-div-v2	27 → 64	67 → 72	44 → **92**	0 ±0 → **94** ±3	69 ±6 → 86 ±5	75 ±24 → **94** ±3	76 ±9 → **96** ±2	7 ±19 → **92** ±4
pen-cloned-v1	84 → 102	74 → 138	−3 → −3	3 ±2 → 120 ±10	77 ±7 → 107 ±10	53 ±14 → **149** ±6	64 ±13 → **142** ±7	50 ±6 → **143** ±4
door-cloned-v1	1 → 20	0 → 102	−0 → −0	0 ±0 → 102 ±7	3 ±2 → 50 ±15	0 ±0 → 102 ±5	0 ±0 → 99 ±2	0 ±0 → **108** ±4
hammer-cloned-v1	1 → 57	7 → 125	0 → 0	0 ±0 → 128 ±29	4 ±2 → 60 ±14	0 ±0 → 127 ±17	7 ±3 → 106 ±10	0 ±0 → **139** ±4
relocate-cloned-v1	0 → 0	1 → 7	−0 → −0	0 ±0 → 2 ±2	−0 ±0 → 5 ±3	0 ±1 → **62** ±8	1 ±0 → 35 ±8	0 ±0 → 23 ±15
Average rank (↓)	7.10	4.90	5.47	4.30	5.63	3.07	2.83	**2.70**

4.3 Effectiveness of Marginal Drifting

As discussed in Sec. 3.2, conditional drifting faces a single-positive-sample problem in continuous-control offline RL: each state is typically observed only once and is therefore paired with only one action. The conditional drifting field must consequently estimate its positive component from a single Monte Carlo sample. Our marginal drifting addresses this issue by pooling actions across states, providing a better-sampled distributional signal while preserving a necessary condition for conditional distribution matching. Next, we examine its effect in a controlled conditional-generation experiment and through offline RL ablations.

2D toy experiment. To simulate the single-positive-sample setting, we construct a continuous conditional 2D distribution. We first sample noisy points from a 2D spiral. Each point is then transformed according to a continuous scalar condition $c \in [1, 2]$. The condtion controls scale, twist, and radial deformation of the spiral, and changing c nonlinearly alters the geometry of the conditional distribution. Training conditions are sampled continuously from $[1, 2]$, so observations at exactly the same c occur with probability zero. Consequently, each training condition provides only a single Monte Carlo sample from its corresponding conditional distribution. For evaluation, we exclude an interval around $c = 1.5$ from training and evaluate the learned generator at $c = 1.5$.

We compare two variants: 1) the composition of conditional and marginal fields, *i.e.*, $\lambda V_{\text{cond}} + (1 - \lambda)V_{\text{marg}}$ with $\lambda = 0.9$; 2) conditional drifting alone, *i.e.*, $\lambda = 1$. Both variants receive the same

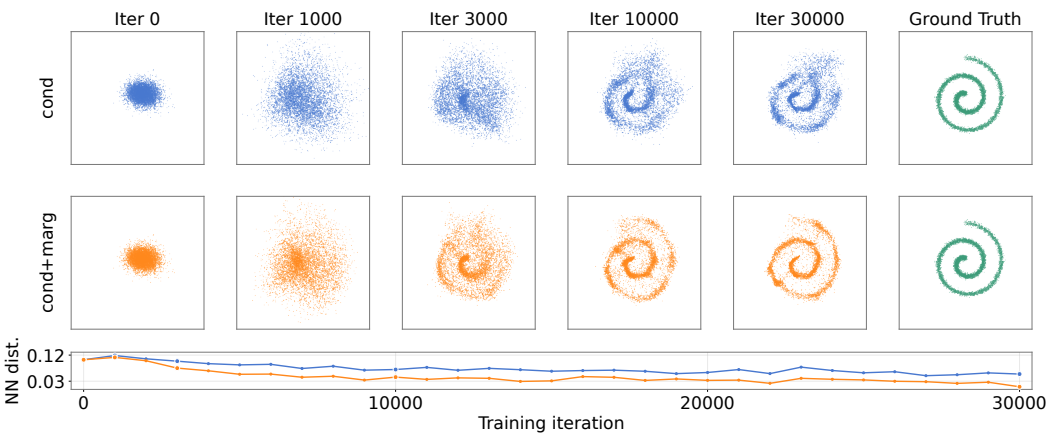

Figure 2: **Conditional generation under single-positive regime. Middle row**: compositional drifting $\lambda V_{\text{cond}} + (1-\lambda)V_{\text{marg}}$ with $\lambda = 0.9$. **Top row**: conditional drifting only ($\lambda = 1$). Each snapshot shows 6,000 generated samples at the held-out condition $c = 1.5$; the rightmost panels show samples from the corresponding target distribution in green. **Bottom row**: the curves report the generated-to-target nearest-neighbor (NN) distance, defined as the Euclidean distance from each generated sample to its nearest target sample. Lower is better; the y-axis is logarithmic.

Table 3: **Quantitative results of 2D conditional generation.** We test at the held-out condition $c = 1.5$. p and q denote the target and generated distributions, respectively. Negative Log-Likelihood (NLL) and KL divergence are estimated using KDE with bandwidth 0.05. The Wasserstein-1 (W1) distance uses Euclidean transport cost with 4,000 samples from each distribution.

Method	NLL ↓	KL($p\|\|q$) ↓	KL($q\|\|p$) ↓	W1 ↓
Conditional only	1.3483	0.6541	1.6837	0.1141
Conditional + marginal	**1.0938**	**0.3996**	**0.7070**	**0.0981**

one-positive supervision, and the only difference is that marginal drifting additionally pools positive samples across conditions.

Figure 2 shows the evolution of the generated distribution during training. Conditional drifting alone gradually recovers the global shape of the target but produces visibly more dispersed samples around the spiral structure. Adding marginal drifting yields a cleaner approximation of the target geometry and consistently lowers the nearest-neighbor distance during training. We further quantify distributional matching at the held-out condition $c = 1.5$. As shown in Table 3, marginal drifting improves all four distributional metrics, which further illustrates the effectiveness of marginal drifting.

Offline RL ablation. We next evaluate whether the same effect translates to offline RL. We compare the full CoDrift model with a variant that removes the marginal field by setting $\lambda = 1$, thereby assigning all behavioral regularization to conditional drifting. All other components of the algorithm are kept unchanged. Table 4 reports results on four representative tasks from OGBench similar to (Park et al., 2025b). Marginal drifting improves success rates on all four tasks by 6–8 percentage points, with an average gain of 6.8 points. The results further support the use of marginal drifting as a complementary behavioral field in the single-positive regime of continuous-control offline RL.

4.4 Role of Other Fields

The conditional and value fields play fundamental roles. Removing the conditional field eliminates the paired state–action supervision, while removing the value field reduces CoDrift to behavioral distribution matching without value-directed optimization. In both cases, performance degrades substantially: the average success rate on the four representative tasks drops from 85.3 to 8.8 without the conditional field and to 3.8 without the value field. Full results are provided in the supplement.

Table 4: **Effectiveness of marginal drifting.** Success rates on four representative tasks, averaged over 8 seeds. `w/o Marg` denotes the variant without the marginal field. Δ denotes the resulting gain.

Task	CoDrift	w/o Marg	Δ
antmaze-L	**84** ± 3	77 ± 6	+7
humanoidmaze-M	**83** ± 5	77 ± 5	+6
cube-single	**92** ± 3	86 ± 3	+6
puzzle-3x3	**82** ± 13	74 ± 4	+8
Average	**85.3**	78.5	+6.8

Table 5: **Runtime cost.** Average over 20 repeats on `cube-double`. `Latency` is in ms/action. `Speedup` is relative to CoDrift.

Policy	Latency↓	Speedup
IFQL	3.681	0.10×
DeFlow	3.087	0.12×
FQL	0.360	1.04×
CoDrift	0.376	1.00×

4.5 Runtime

In Table 5, we evaluate the runtime cost of CoDrift on `cube-double`. As CoDrift requires only a single forward pass at deployment, it matches the speed of the one-step FQL policy and is roughly an order of magnitude faster than the multi-step policies IFQL and DeFlow. All methods are timed under identical conditions, including hardware and warmup protocol.

5 Related Work

Offline RL and behavior regularization. Offline RL seeks to maximize return using a fixed dataset while avoiding extrapolation errors caused by out-of-distribution actions (Levine et al., 2020; Fujimoto et al., 2019). Existing approaches include value regularization (Kumar et al., 2020; An et al., 2021), in-sample learning (Kostrikov et al., 2022; Xu et al., 2023), and policy regularization. The latter retains an actor–critic formulation while constraining the learned policy toward the behavior distribution, either through explicit penalties (Wu et al., 2019; Fujimoto & Gu, 2021; Tarasov et al., 2023) or weighted regression (Peng et al., 2019; Nair et al., 2020). CoDrift follows the same general principle of balancing behavioral fidelity and value maximization, but represents these objectives as action-space displacement fields. In particular, it constrains behavior at both the conditional and marginal levels and composes these behavioral fields directly with a critic-guided value field.

Generative policies. Expressive generative models provide a natural policy class for representing complex and multimodal action distributions. Existing generative policies have explored several modeling paradigms, including autoregressive models (Kim et al., 2024; Wang et al., 2026a), generative adversarial networks (Vuong et al., 2022), diffusion models (Wang et al., 2023; Chi et al., 2023), and flow matching (Park et al., 2025b; Chang et al., 2026; Wang et al., 2026b; Espinosa-Dice et al., 2025). CoDrift takes a different approach based on drifting models (Deng et al., 2026). Drifting models learn a stochastic generator from displacement fields acting directly on generated samples and support native one-step generation. More importantly for offline RL, their displacement-based formulation provides a natural interface for combining heterogeneous learning objectives: once different objectives are expressed as action-space fields, their effects can be composed additively before training the generator. Concurrent with our work, Koo et al. (2026) also applies drifting models to one-step policy learning, but uses a single drifting field toward a value-reweighted target. In contrast, CoDrift explicitly composes multiple objective-level fields within a unified one-step generative framework.

6 Concluding Remarks

We introduced CoDrift, a compositional drifting framework for offline reinforcement learning. CoDrift represents behavioral fidelity and value maximization as action-space fields and combines conditional behavioral, marginal behavioral, and value fields into a single update direction for a one-step stochastic policy. Across OGBench and D4RL tasks, CoDrift achieves strong performance in both offline and offline-to-online settings with low runtime cost. More broadly, CoDrift highlights the compositionality of drifting models, allowing heterogeneous objectives and constraints to be combined within a single generative policy.

AI USE STATEMENT

We used generative AI tools to improve the English wording and readability of the manuscript and assist with LaTeX formatting and layout adjustments. We also used these tools for general code completion and debugging. We did not use generative AI tools for any other tasks requiring disclosure, and the remaining tasks requiring disclosure were not applicable to this work. All AI-assisted text and code were carefully reviewed by the authors, and all AI-generated code was tested. The authors take full responsibility for the final manuscript, code, and reported research results.

ETHICS STATEMENT

This work studies offline and offline-to-online reinforcement learning using the OGBench and D4RL benchmarks, together with a synthetic two-dimensional experiment. We did not recruit human participants or collect personal or sensitive data for this study. The authors have no conflicts of interest to disclose. To the best of our knowledge, this work does not raise specific ethical concerns that require further discussion.

REPRODUCIBILITY STATEMENT

We provide an anonymized implementation of CoDrift in the supplementary material to facilitate reproducibility. The package includes training and evaluation code, dependency specifications, and a README with example commands. The accompanying supplementary material lists relevant hyperparameter settings for the offline and offline-to-online experiments on OGBench and D4RL.

REFERENCES

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning (ICML)*, 2023.

Jianlei Chang, Ruofeng Mei, Wei Ke, and Xiangyu Xu. Efficientflow: Efficient equivariant flow policy learning for embodied ai. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

Huayu Chen, Cheng Lu, Zhengyi Wang, Hang Su, and Jun Zhu. Score regularized policy optimization through diffusion behavior. In *International Conference on Learning Representations (ICLR)*, 2024.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

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

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

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

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, 2019.

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

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv:2406.09246*, 2024.

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

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Vladislav Kurenkov and Sergey Kolesnikov. Showing your offline reinforcement learning work: Online evaluation budget matters. In *International Conference on Machine Learning (ICML)*, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv:2005.01643*, 2020.

Zhancun Mu. DeFlow: Decoupling manifold modeling and value maximization for offline policy extraction. *arXiv:2601.10471*, 2026.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv:2006.09359*, 2020.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning (ICML)*, 2025b.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv:1910.00177*, 2019.

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

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Quan Vuong, Aviral Kumar, Sergey Levine, and Yevgen Chebotar. Dasco: Dual-generator adversarial support constrained offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Ruiheng Wang, Shuanghao Bai, Haoran Zhang, Badong Chen, and Xiangyu Xu. Blockvla: Accelerating autoregressive vla via block diffusion finetuning. *arXiv:2605.13382*, 2026a.

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

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv:1911.11361*, 2019.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. In *International Conference on Learning Representations (ICLR)*, 2023.