Higher-Order Action Supervision Makes A Strong Policy Class

Peng Cheng[1,4]*, **Yunxian Hou**[2]*, **Zhi Zhou**[1,4], **Qian Zhang**[3], **Chang Huang**[3], **Xianyuan Zhan**[1]†
[1] Institute for AI Industry Research (AIR), Tsinghua University
[2] University of Electronic Science and Technology of China
[3] Horizon Robotics [4] Zenfex AI Lab

Abstract

Modern data-driven decision-making methods, such as imitation learning (IL) and reinforcement learning (RL), have achieved great success in solving many complex tasks. However, these methods often suffer from serious control instability and robustness issues when applied in real-world applications such as robotics and autonomous driving, posing notable challenges for their practical deployment. We argue that this instability issue stems largely from their limitations in solely supervising and optimizing zeroth-order actions (i.e., the action labels), failing to account for higher-order action dynamics and temporal consistency. In this paper, we show that simultaneously supervising both zeroth- and first-order actions can dramatically enhance policies' performance and control robustness. To achieve this, we introduce a novel and elegant loss scheme supported by formal theoretical guarantees that can equip any off-the-shelf policy model (e.g., deterministic, stochastic, or flow policies) with the capability for higher-order action supervision, without requiring any structural modifications. Moreover, our proposed method can serve as a lightweight plug-and-play module that seamlessly integrates with a broad spectrum of existing offline RL frameworks. Extensive evaluations on OGBench and D4RL demonstrate that our approach yields substantial performance and robustness improvements across a wide range of continuous control environments. Notably, our method can also enhance policies' out-of-distribution (OOD) generalization capability in the challenging low-data regime, making it an ideal tool in tackling many real-world control problems.

1 Introduction

Learning-based decision-making methods, such as imitation learning (IL) and reinforcement learning (RL), have achieved notable success in many complex tasks, such as games [1, 2], robotics [3, 4], and autonomous driving [5, 6]. However, when these learning-based approaches are deployed in real-world control problems, they are frequently criticized for their control instability and poor robustness. Examples of such issues include lack of smoothness in action outputs, severe jittering across consecutive steps, and poor generalization in offline learning under data sparsity [7, 8, 9, 10].

In contrast to conventional control theories [11] that place heavy emphasis on control stability and higher-order behavior modeling, contemporary learning-based decision-making methods in the AI community predominantly only focus on supervising and optimizing zeroth-order actions (i.e., the actual action labels). In realistic control scenarios, the neglect of higher-order temporal consistency in actions frequently leads to control instability and non-smooth action execution, which constitutes a critical bottleneck for real-world deployment [7]. Under physical hardware

*Equal contribution.
†Corresponding author.

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

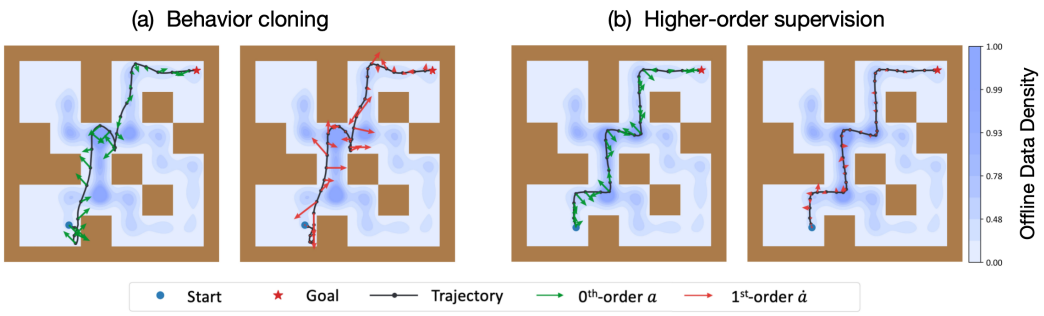

Figure 1: Trajectories and zeroth-/first-order actions of policies learned on the `maze2d` task with 20 successful trajectories and 500k training steps. Green arrows denote the zeroth-order action a (joint torque), while red arrows denote the first-order action $\dot{a}$ (time derivative of joint torque, computed as $a' - a$). (a) Standard behavior cloning (BC) without $\dot{a}$ supervision, which yields a suboptimal trajectory with highly noisy and temporally inconsistent first-order actions. (b) Our method with higher-order action supervision, which yields an optimal navigation trajectory with much smoother and more stable zeroth- and first-order actions.

constraints, such high-frequency action fluctuations can cause excessive mechanical wear and energy consumption, drastically shortening the lifespan of both the system and its constituent parts [8, 12, 13, 14, 15, 16]. These vulnerabilities pose severe safety risks and degrade operational reliability across various domains, including drone control [13, 17], robotic manipulation [18, 19], and autonomous driving [20, 21, 22, 23]. Fundamentally, this issue arises because the policy becomes excessively sensitive to minor state-action perturbations, yielding disproportionately large action variations [8, 21]. We empirically illustrate this vulnerability using a maze navigation task in Figure 1. As shown in Figure 1(a), the policy learned via standard behavior cloning (BC), which lacks higher-order action supervision, produces large and highly oscillatory first-order actions, eventually producing a suboptimal trajectory. By contrast, jointly supervising both zeroth- and first-order actions enforces much stronger behavioral regularization, resulting in significantly smoother and optimal execution (Figure 1(b)).

However, incorporating higher-order action information in policy modeling is not an easy task. As the 0th-order action a and the 1st-order action $\dot{a}$ live in two completely different action spaces, naïvely predicting these two sets of actions not only doubles the action space, causing learning difficulties, but also faces potential conflicting gradient issues that are commonly encountered in multi-task learning [24]. Therefore, existing works on action smoothness improvement primarily focus on introducing action fluctuation penalty regularizations [13, 25, 26, 27], or designing special neural network architectures to constrain the network's Lipschitz constant, e.g., via spectral normalization or specialized modules like LipsNet [8, 12]. However, both streams of work suffer from a number of limitations. Adding smoothness penalties relies on task-specific ad-hoc design and cumbersome tuning efforts, whereas using specialized, heavily regularized neural network architectures loses generality and restricts the model's expressivity to capture complex data distributions.

In this work, we introduce a novel and elegant learning scheme that can achieve simultaneous 0th- and 1st-order action supervision. The best part is, we can easily transform any given policy model into a more powerful policy class without making any structural modification. The secret recipe is to leverage the intrinsic property of the policy model's time derivative for supervision, i.e., $\dot{\pi}(s) = \nabla_s \pi(s) \cdot \dot{s}$. We provide practical instantiations of loss schemes for deterministic, stochastic, and generative flow matching policy classes, supported by formal theoretical guarantees for the IL setting. Moreover, we show that our enhanced policy class is naturally compatible with the continuous-time RL paradigm, like the Hamilton-Jacobi-Bellman (HJB) framework. This leads to a plug-and-play enhancement module that can be seamlessly integrated with a wide spectrum of existing offline RL algorithms with minimal code modifications. We evaluate our method through extensive experiments on diverse tasks from the large-scale OGBench [28] and widely used D4RL [29] benchmarks. Despite its simplicity, our approach achieves consistent and significant performance improvements when added to different offline policy optimization paradigms. Notably, in challenging small-dataset settings, our proposed modification excels by providing robust OOD generalization support, enabling the agent to maintain stable and high performance even on tasks where the base algorithms fail or severely degrade. This makes our method an ideal tool for tackling many real-world control problems.

2 Related work

Smoothness and robustness enhancement for policy learning. Mitigating action fluctuation and trajectory drift remains a critical challenge for deploying policies in physical systems [7, 8]. To enforce action smoothness and robustness, prior studies in continuous control have primarily explored two directions: penalty-based regularizations [13, 25, 26, 27] and architectural enhancements [8, 12]. Penalty-based regularization approaches, such as CAPS [13], L2C2 [25], and SR^{2}L [26, 27], introduce action fluctuation penalty regularizations to their policy optimization objectives. However, they inherently rely on querying or sampling synthetic states to enforce local action consistency. In the offline setting, where interaction with the environment is not permitted, such state-dependent sampling may introduce significant OOD extrapolation errors, and often suffer from significant performance variability across environments [30]. In contrast, architectural enhancement methods explicitly constrain the network's Lipschitz constant, e.g., via spectral normalization or specialized modules like LipsNet [8, 12]. However, imposing extra structural priors can also restrict the model's expressivity. This introduces a fundamental trade-off: while the policy becomes smoother, the restricted capacity commonly leads to underfitting, degraded control performance, and additional computational overhead during inference. Concurrently, there are also a few attempts to incorporate higher-order action derivatives to enhance control smoothness, such as TaSIL [31], which minimizes the gap between the Taylor series expansion of the learned and expert policies. Although promising, it remains more theoretical rather than practical, due to the complex evaluation procedure involved.

Offline RL and continuous-time RL. In this study, we primarily focus on offline policy learning, such as offline IL (e.g., behavior cloning, BC) and offline RL, which directly imitate or optimize over a fixed offline dataset without environment interaction. Compared with IL, offline RL is more complex, which optimizes the cumulative return while being regularized towards the behavior data to avoid extrapolation error accumulation [32, 33]. Depending on how the learned policy is parameterized, existing offline RL algorithms can be classified into three categories. The first category employs deterministic policies and explicitly constrains the learned actions toward the dataset using distance-based behavior cloning penalties [34, 35, 9]. The second and also the more popular category adopts stochastic policies (e.g., Gaussian policies) [36, 33, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47], which maximizes the likelihood of high-return actions under some distribution-based data regularizations. Recently, the third category of offline RL methods has emerged, which leverages expressive generative models, such as diffusion or flow matching policies, to capture the complex, multimodal action distributions in the datasets [48, 49, 50, 51, 52, 53, 54, 55, 56, 57]. However, all of the above methods are restricted to learning policies in the 0th-order action spaces, which suffer from notable control stability and robustness issues. Incorporating higher-order action supervision in offline RL remains entirely unexplored.

Another research avenue closely related to our work is continuous-time RL, which transcends the structural limitations of discrete-time Markov decision processes (MDPs) by modeling state and action evolution continuously over time [58, 59, 60]. The Hamilton-Jacobi-Bellman (HJB) equation serves as the core theoretical foundation for this domain [61, 62, 63, 64]. Recent advancements have successfully utilized HJB-based frameworks to demonstrate how continuous-time structural priors can directly guide stable policy evaluation and control [65, 66, 67]. Although the HJB-based framework naturally requires 1st-order action $\dot{a}$ information to capture Q-value's variation in continuous time, due to the lack of a proper strategy to evaluate $\dot{a}$ from policy π, existing methods often resort to a crude approximation or replace it with a fixed bounding value. This drastically degrades the value estimation quality under the HJB framework and leads to suboptimal performance.

3 Extending Standard Policy Classes for Higher-Order Action Supervision

In this paper, unlike prior methods that mitigate action prediction instability by using specialized network architectures or introducing task-specific ad-hoc designs, we propose an elegant, lightweight, and plug-and-play learning scheme to transform a standard policy model (regardless of deterministic, stochastic, or generative flow policies) into a more powerful policy class. Our proposed method enables simultaneous 0th and 1st-order action supervision for a policy model, maximally utilizing every bit of information in the offline data.

3.1 Notations

We consider the sequential decision-making problem modeled as an MDP, defined by the tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma\}$. $\mathcal{S}$ represents the state space, $\mathcal{A} \subset \mathbb{R}^d$ is the d-dimensional continuous action space, $\mathcal{P}(s'|s,a)$ denotes the stochastic transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho_0(s)$ specifies the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. In the offline setting, the agent cannot interact with the environment and is instead provided with a static dataset $\mathcal{D}$ consisting of offline trajectories $\tau = \{s_0, a_0, r_0, s_1, a_1, \dots\}$. In the offline IL setting, we aim to learn an imitative policy π to clone the expert behaviors specified in the data trajectories; whereas in offline RL, we aim to learn a policy π exclusively from the dataset $\mathcal{D}$ that maximizes the expected cumulative discounted return: $J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. In this paper, we use $\pi(s)$ to denote the deterministic policy and $\pi(\cdot|s)$ to denote the stochastic policy. Unless told otherwise, we use a to denote the 0th-order action and $\dot{a} = \frac{da}{dt}$ to denote the 1st-order action. In the discrete-time MDP setting, we approximate the time derivatives of the state and actions using finite differences, i.e., $\dot{s} \approx (s' - s)/\Delta t$ and $\dot{a} \approx (a' - a)/\Delta t$, with the time interval set to $\Delta t = 1$ by default.

3.2 Supervising Both Zeroth and First-Order Action for Imitation Learning

The core difficulty to achieve simultaneous 0th and 1st-order action supervision lies in how to properly inject $\dot{a}$ information within the original model structure of the policy, since conventional policy models, regardless of deterministic policy $\pi(s)$ or stochastic policy $\pi(\cdot|s)$ are designed for the straightforward mapping between state s and the 0th-order action a. However, if we are allowed to take the time derivative of policy π, then we will notice the following property (using the simplest deterministic-policy case as an example):

$$\dot{a} = \dot{\pi}(s) = \frac{d\pi(s)}{dt} = \frac{d\pi(s)}{ds} \cdot \frac{ds}{dt} = \frac{d\pi(s)}{ds}\dot{s} = \nabla_s \pi(s)\dot{s}. \tag{1}$$

The above equation suggests that we can express the time derivative of policy π as the product of the gradient of policy $\nabla_s \pi(s)$ and 1st-order state $\dot{s}$. As $\dot{s}$ is directly obtainable from the offline dataset during training, this offers us a new perspective to supervise 1st-order action signal directly on the original policy model. In the following, we describe the detailed higher-order action supervision strategies for deterministic, stochastic, and generative flow-matching policies.

Deterministic policy. With Eq. (1), incorporating 1st-order action supervision for deterministic policies under the IL setting is straightforward. We can simply add an extra 1st-order loss term $\mathcal{L}_{\dot{a}}^{\text{det}}$ to the original BC loss:

$$\min_{\pi} \mathbb{E}_{(s,a,s',a')\sim\mathcal{D}}\Big[\, \|\pi(s) - a\|^2 + \underbrace{\|\nabla_s \pi(s)\dot{s} - \dot{a}\|^2}_{\mathcal{L}_{\dot{a}}^{\text{det}}} \Big]. \tag{2}$$

where $\dot{s} = s' - s$ and $\dot{a} = a' - a$ are directly calculated from dataset samples. The second term $\mathcal{L}_{\dot{a}}^{\text{det}}$ regresses $\dot{\pi}(s)$ to the ground truth $\dot{a}$, thereby achieving first-order temporal consistency regularization without the need to make any changes to the original policy network. In practice, we can directly evaluate $\nabla_s \pi(s)\dot{s}$ using Jacobian-vector product (JVP), which is much cheaper to compute than calculating the Jacobian matrix $\nabla_s \pi(s)$, and is natively supported in modern deep learning frameworks like JAX and PyTorch 2.0. A keen reader might notice that we do not introduce any weighting parameter to the two loss terms in Eq. (2), as we find empirically that keeping the same supervision strength of 0th and 1st-order actions normally provides the best performance.

Stochastic policy. Stochastic policies are often modelled as a Gaussian policy $\pi(a|s) = \mathcal{N}(a; \mu(s), \text{diag}(\sigma^2(s)))$ in practice, with $\mu(s)$, $\sigma^2(s)$ being implemented as parameterized neural networks outputting mean and dimension-independent variances. They have been widely used in many offline IL and RL algorithms due to their flexibility in capturing dataset distributions. To enable 1st-order action supervision on a stochastic policy, one naïve approach is to supervise the mean policy $\mu(s)$ with an additional loss term $\mathcal{L}_{\dot{a}}^{\text{sto-mean}} = \|\nabla_s \mu(s)\dot{s} - \dot{a}\|$ as in Eq. (2). We refer to this version as "*mean supervision*". Below, we also provide a more rigorous version consistent with the stochastic process framework.

Specifically, if we model the stochastic component of the Gaussian policy using an Ornstein–Uhlenbeck (OU) [68] process, $\mathrm{d}X_t = -\lambda X_t\,\mathrm{d}t + \sqrt{2\lambda}\,\mathrm{d}W_t, X_0 \sim \mathcal{N}(0, I)$, where λ controls

the temporal correlation of the process, and W_t denotes a d-dimensional standard Wiener process. By the stationarity of the OU process, for any t, we have $X_t \sim \mathcal{N}(0, I)$. The action can thus be written as $a = \mu(s) + \sigma(s) \odot X_t$, where $\odot$ denotes element-wise multiplication. By the multi-dimensional Itô formula [69], we obtain the stochastic differential equation (SDE) of the action process,

$$\mathrm{d}a = [\dot{\mu}(s) + (\dot{\sigma}(s) - \lambda\sigma(s)) \odot ((a - \mu(s)) \oslash \sigma(s))]\,\mathrm{d}t + \sqrt{2\lambda}\,\sigma(s) \odot \mathrm{d}W_t, \tag{3}$$

where $\oslash$ denotes element-wise division. The resulting SDE induces a first-order policy distribution over the time derivative of actions (see Appendix C.1 for detailed derivation),

$$\dot{\pi}(\dot{a}|s, a, \dot{s}) = \mathcal{N}\big(\dot{a};\, \dot{\mu}(s) + (\dot{\sigma}(s) - \lambda\sigma(s)) \odot ((a - \mu(s)) \oslash \sigma(s))\,,\, 2\lambda\,\mathrm{diag}(\sigma^2(s))\big)\,, \tag{4}$$

where $\dot{\mu}(s) = \nabla_s \mu(s) \cdot \dot{s}$ and $\dot{\sigma}(s) = \nabla_s \sigma(s) \cdot \dot{s}$ can be evaluated in a similar way as in Eq. (1). The above distribution specifies the likelihood of the time derivative of stochastic actions conditioned on the current state, action, and state derivative. We can thus jointly maximize the likelihood of 0th and 1st-order action for stochastic policies under the IL setting as follows:

$$\min_{\pi} \mathbb{E}_{(s,a,s',a')\sim\mathcal{D}}\Big[-\log \pi(a|s) - \underbrace{\log \dot{\pi}(\dot{a}|s, a, \dot{s})}_{\mathcal{L}^{\text{sto}}_{\dot{a}}} \Big]. \tag{5}$$

Beyond conventional action-level maximum likelihood in the first term, the derivative likelihood $\mathcal{L}^{\text{sto}}_{\dot{a}}$ in Eq.(5) serves as a probabilistic first-order regularizer that aligns the local temporal evolution of sampled actions with the expert trajectory. We provide the full derivation in Appendix C.1.

Flow policy. Flow matching provides an effective and powerful framework for generative modeling [70, 71], and has gained increasing popularity in recent RL and embodied AI models. The central idea is to learn a time-dependent velocity field $v : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ (note this "time" is different from the physical time, which we denote as k) that induces a continuous flow $\psi : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ between a base distribution and the target data distribution, where $\frac{d}{dk}\psi(k, x) = v\big(k, \psi(k, x)\big), \psi(0, x) = x$.

In this paper, we consider the standard linear interpolation path [70, 71], which defines an interpolation process for $k \in [0, 1]$ as $x^k = (1 - k)x^0 + kx^1, v = x^1 - x^0$ between a noise sample x^0 and a data sample x^1. For flow policy [54], let $a^0 \sim \mathcal{N}(0, I)$ denote a noise sample and $a^1 \sim \mathcal{D}$ denote an action. Given the input state s, the generated action is obtained by integrating the learned velocity field along the interpolation path. Note that under the formalism of flow matching, if we differentiate the generated action $a^1 = a^0 + \int_0^1 v(k, s, a^k)\, dk$ with respect to physical time t, we have

$$\dot{a}^1 = \frac{d\int_0^1 v(k, s, a^k)\, dk}{dt} = \int_0^1 \frac{dv(k, s, a^k)}{dt}\, dk = \int_0^1 \frac{\partial v(k, s, a^k)}{\partial s}\dot{s} + \frac{\partial v(k, s, a^k)}{\partial a^k}\dot{a}^k\, dk. \tag{6}$$

We omit $\dot{a}^0$ as the noise sample a^0 is drawn independently of the physical trajectory, hence calculating the time derivative on noise is meaningless. Equation (6) shows that first-order action evolution can be expressed by integrating $\dot{v}(k, s, a^k)$ over flow time, analogous to the original action generation process. This shared structure motivates supervising $\dot{v}(k, s, a^k)$ in a flow-matching manner by regressing it toward the pointwise target $\dot{a}^1 = a' - a^1$. In practice, we approximate $\dot{v}(k, s, a^k)$ with $\frac{\partial v(k,s,a^k)}{\partial s}\dot{s}$. We omit the action-dependent term in Eq. (6) because evaluating it increased computational cost without improving performance, as discussed in Appendix B.2. Therefore, the new objective for flow policies with higher-order supervision can be given as:

$$\min_{\pi} \mathbb{E}_{a^0\sim\mathcal{N}(0,I),\,(s,a^1,s',a')\sim\mathcal{D},\,k\in[0,1]}\Big[\big\|v(k, s, a^k) - (a^1 - a^0)\big\| + \underbrace{\Big\|\frac{\partial v(k, s, a^k)}{\partial s}\dot{s} - \dot{a}^1\Big\|}_{\mathcal{L}^{\text{flow}}_{\dot{a}}} \Big]. \tag{7}$$

Standard flow matching mainly supervises transformation along the probability path at time k, while leaving the physical-time response of the conditional velocity field largely unconstrained. In Eq. (7), the additional term $\mathcal{L}^{\text{flow}}_{\dot{a}}$ completes this missing piece by aligning the state-directional change of the velocity field with the observed action evolution. We provide further discussion in Appendix C.2.

3.3 Theoretical Analysis

With the additional first-order supervision, we can show in the following theorems that this scheme can lead to a tighter error bound for imitation learning under a set of commonly used assumptions (see Appendix D.2). Specifically, consider the learned policy π and expert policy π^* start at the same state $s(0)$ (we denote $s(t)$ as the state in continuous time t), and define the policy error as $e(s) = \pi(s) - \pi^*(s)$. We can show that the policy error can be better bounded with stronger 1st-order supervision (see Appendix D.4 for the proof):

Theorem 1. *Under Assumption 1 in Appendix D.2, the policy error $e(s^*(t))$ satisfies:*

$$\|e(s^*(t))\| \leq \|e(s^*(0))\| + t \max_{0\leq\tau\leq t} \left\| \nabla_s \pi(s^*(\tau))\dot{s}^*(\tau) - \dot{\pi}^*(s^*(\tau)) \right\|.$$

Theorem 1 shows that the growth of policy error $e(s^*)$ on expert trajectory s^* is governed by its initial value and the accumulated 1st-order discrepancy between the learned and expert policies. Note that the standard BC objective does not supervise the second term, hence could lead to substantially increased errors. As Theorem 1 only provides analysis on the per-state policy error, next, we provide a stronger conclusion to bound the state deviation $\delta(t) = s(t) - s^*(t)$ between states from policy rollout s and expert trajectories s^*, which captures the closed-loop long-term error accumulation.

Theorem 2. *Under Assumptions 1–4 in Appendix D.2, for $\delta(0) = 0$, the state deviation $\delta(t)$ satisfies,*

$$\|\delta(t)\| \leq \frac{L_a}{\beta}\left(e^{\beta t} - 1\right) \max_{0\leq\tau\leq t} \|e(s^*(\tau))\|.$$

The detailed proof and exact forms of constants L_a and β are provided in Appendix D.5. Theorem 2 decomposes closed-loop policy rollout state deviation into two parts: the policy mismatch error $e(s^*)$ measured on the expert trajectory s^*, and the amplification gain $\frac{L_a}{\beta}(e^{\beta t} - 1)$ contributed by the environment dynamics, capturing the error amplification due to closed-loop policy-environment interaction. Again, we can see that the long-term state deviation can also be better controlled under higher-order supervision, due to the tighter error bound $\|e(s^*(t))\|$ obtained in Theorem 1.

4 Higher-Order Policy Optimization in Offline RL

The extended policy in Section 3 not only enables higher-order action supervision, but also provides a practical way to evaluate action derivatives $\dot{\pi}_\theta(s)$, a quantity that existing offline RL methods never explicitly use. We now show that it has a natural place in critic learning through the Hamilton-Jacobi-Bellman based continuous-time RL framework, where value learning depends not only on scalar Bellman consistency, but also on how the value changes along the local state-action flow.

4.1 HJB Equation and Value Learning with First-Order Supervision

In the HJB-based continuous-time RL framework, the Q-function is defined by [65]:

$$Q^{\text{HJB}}(s, a) := \max_{a\in\mathcal{A}} \int_0^\infty e^{-\gamma t} r(s(t), a(t)) dt. \tag{8}$$

The dynamic programming equation of the above Q-function is expressed as an *HJB equation*, which is derived by using the Taylor expansion:

$$\gamma Q^{\text{HJB}}(s, a) = \max_{a\in\mathcal{A}} \left[r(s, a) + \nabla_s Q^{\text{HJB}}(s, a)\dot{s} + \nabla_a Q^{\text{HJB}}(s, a)\dot{a} \right] \tag{9}$$

Unlike the typical discrete-time Bellman equation, which compresses value learning into a scalar one-step regression, Eq. (9) reveals what is missing in existing offline RL methods. The value function may match 0th-order Bellman targets, yet its gradient is never required to align with the variation direction of states and actions in time. This makes the learned value landscape numerically plausible but structurally under-specified, especially for later policy improvement. However, with the extended policy class introduced in Section 3, we can easily obtain the time derivative of actions through the

learned policy, i.e., $\dot{a} = \dot{\pi}_\theta(s) = \nabla_s \pi_\theta(s) \cdot \dot{s}$. This allows us to jointly optimize the value function with policy's higher-order information during Bellman update, which leads to the following objective:

$$Q = \arg\min_Q \mathbb{E}_{(s,a,r,s',a')\sim\mathcal{D}}\Bigg[\Big(\mathcal{T}_\pi \hat{Q}(s,a) - Q(s,a)\Big)^2 - \lambda_Q \left(\frac{\nabla_s Q(s,a)\dot{s}}{\|\nabla_s Q(s,a)\|\|\dot{s}\|} + \frac{\nabla_a Q(s,a)\dot{\pi}_\theta(s)}{\|\nabla_a Q(s,a)\|\|\dot{\pi}_\theta(s)\|}\right)\Bigg]. \tag{10}$$

As $\dot{s} = s' - s$ can be directly calculated from the dataset during offline learning, we can actually seamlessly transform the original value learning scheme of an existing offline RL algorithm into a higher-order supervision version by adding the above blue terms, with $\mathcal{T}_\pi$ denoting the specific Bellman operator used in the offline RL algorithm. Moreover, to improve numerical stability and to prevent overly large gradient magnitudes of the first-order terms, we normalize the two HJB directional terms in Eq. (10), similar to existing HJB-based methods [65, 67].

4.2 Higher-Order Policy Learning

With the Q-function learned with 1st-order supervision, we can also transform the policy optimization procedure of an existing offline RL algorithm through a plug-and-play modification. This can be achieved by simply adding an additional 1st-order imitation loss $\mathcal{L}_{\dot{a}}$ as defined in Section 3,

$$\pi = \arg\max_\pi \mathbb{E}_{(s,a,s',a')\sim\mathcal{D},\hat{a}\sim\pi} Q(s,\hat{a}) - \mathcal{R}_\pi(s) - \lambda_{\dot{a}}\mathcal{L}_{\dot{a}}. \tag{11}$$

where we use $Q(s,\hat{a}) - \mathcal{R}_\pi(s)$ to represent the general policy optimization objective of existing offline RL algorithms, which typically optimize a policy by maximizing the Q-function with some behavior regularization term $\mathcal{R}_\pi(s)$. $\mathcal{R}_\pi(s)$ can either be an explicit policy constraint, such as the BC term $\mathcal{R}_\pi(s) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\|\pi(s) - a\|^2$ in TD3+BC [34], or an implicit behavior regularization scheme as adopted in many algorithms [38, 40, 72, 53, 47, 56], which results in an advantage-weighted regression type of learning objective. Depending on the actual policy class used in the original offline RL algorithm, one can flexibly choose $\mathcal{L}_{\dot{a}}$ to be either $\mathcal{L}_{\dot{a}}^{\text{det}}$, $\mathcal{L}_{\dot{a}}^{\text{sto}}$, or $\mathcal{L}_{\dot{a}}^{\text{flow}}$ in Section 3. We introduce a hyperparameter $\lambda_{\dot{a}}$ to balance the regularization strength of the original policy optimization objective and the first-order supervision term. Both the first-order value supervision term and the $\mathcal{L}_{\dot{a}}$ term in policy optimization introduce minimal modifications to the original offline RL algorithm, but as we show in the experiments, they can lead to substantial performance gains, as the policy is taught not only what to do, but how to evolve reasonably over time.

5 Experiments

In this section, we empirically evaluate our proposed policy class. We first integrate it into a range of representative offline RL algorithms and benchmark the resulting variants against their original counterparts. We then conduct ablation studies to isolate the contributions of the first-order policy constraint and the first-order value consistency regularizer.

5.1 Experimental Setup

Benchmarks. To comprehensively evaluate our proposed framework, we conduct experiments on two widely adopted offline RL benchmarks. We first evaluate on OGBench [28], a recent benchmark featuring high-dimensional state spaces, multimodal behaviors, and sparse rewards. We adopt its reward-based "-singletask" variants so that standard reward-maximizing offline RL algorithms can be directly applied. We further conduct experiments on D4RL [29] under a more challenging low-data setting with only 10k transitions per task, which better exposes performance gaps and OOD generalization since recent offline RL methods have largely saturated the D4RL benchmark under the full-dataset setting. The benchmark details and task descriptions are presented in Appendix E.1.

Comparison methods. We integrate our proposed policy class into representative offline RL algorithms spanning three policy families (deterministic, stochastic, and flow-matching), and compare each base algorithm with its first-order-supervised variant. For deterministic policies, we adopt TD3+BC [34] as the base algorithm. For stochastic policies, we instantiate the two strategies introduced in Section 3.2: a simplified *mean supervision* variant, which applies first-order supervision only to the Gaussian mean and is built upon ReBRAC [73] (denoted "mean sup." in Table 1), and a *standard* variant based on the likelihood objective in Eq. 5 and built upon IQL [38] (denoted

Table 1: Average normalized scores on 10 OGBench task categories, each aggregating 5 sub-tasks over 5 random seeds. Values after ± denote standard deviations. The ↑ marks indicate improvements greater than 1.0, whereas ↑ denotes minor improvements. Full per-task results are presented in Table 4.

	Det. Policy		Sto. Policy (mean sup.)		Sto. Policy (standard)		Flow Policy	
Task Category (Avg. of 5 tasks)	TD3+BC	TD3+BC+ours	ReBRAC	ReBRAC+ours	IQL	IQL+ours	IFQL	IFQL+ours
antmaze-large-navigate	83±4	98±3(↑15)	81±5	98±2(↑17)	53±3	68±5(↑15)	28±13	36±14(↑8)
antmaze-giant-navigate	42±8	65±10(↑23)	26±8	61±12(↑35)	4±1	9±1(↑5)	3±2	4±3(↑1)
humanoidmaze-medium-navigate	34±8	42±5(↑8)	22±8	55±10(↑33)	33±2	36±3(↑3)	60±16	64±13(↑4)
humanoidmaze-large-navigate	2±1	4±1(↑2)	2±1	4±2(↑2)	2±1	3±1(↑1)	11±2	12±3(↑1)
antsoccer-arena-navigate	13±3	16±3(↑3)	0±0	16±4(↑16)	8±2	23±3(↑15)	33±6	40±13(↑7)
cube-single-play	88±5	91±4(↑3)	91±2	93±2(↑2)	83±3	92±3(↑9)	79±2	95±7(↑16)
cube-double-play	7±1	12±1(↑5)	12±1	24±3(↑12)	7±1	8±2(↑1)	14±3	42±15(↑28)
scene-play	44±3	47±2(↑3)	41±3	47±4(↑6)	28±1	33±3(↑5)	30±3	56±7(↑26)
puzzle-3x3-play	27±3	29±1(↑2)	21±1	24±2(↑3)	9±1	12±4(↑3)	19±1	43±15(↑24)
puzzle-4x4-play	6±1	9±1(↑3)	14±1	17±3(↑3)	7±1	13±2(↑6)	25±5	40±12(↑15)
Average Score	35	41(↑6)	31	44(↑13)	23	30(↑7)	30	43(↑13)

Table 2: Average normalized scores on D4RL MuJoCo locomotion tasks with reduced-size (10k) datasets. Standard deviations are reported after the ± sign. The ↑ symbol indicates a performance increase greater than 1.0, whereas ↑ denotes minor improvements. All evaluation results are averaged over 5 random seeds.

	Det. Policy		Sto. Policy (mean sup.)		Sto. Policy (standard)		Flow Policy	
Task Category	TD3+BC	TD3+BC+ours	ReBRAC	ReBRAC+ours	IQL	IQL+ours	IFQL	IFQL+ours
Hopper-m	40.1±18.6	44.1±12.7(↑4.0)	9.8±3.1	32.4±11.0(↑22.6)	47.7±1.0	48.5±2.5(↑0.8)	50.8±2.3	51.3±2.9(↑0.5)
Hopper-mr	7.3±6.1	22.3±3.3(↑15.0)	19.0±1.0	24.9±2.8(↑5.9)	19.3±3.0	19.9±2.0(↑0.6)	23.1±1.4	28.2±0.4(↑5.1)
Hopper-me	17.8±7.9	42.3±10.8(↑24.5)	36.2±3.1	43.0±3.3(↑6.8)	34.3±8.7	42.3±3.3(↑8.0)	45.7±1.9	47.9±1.7(↑2.2)
Hopper-e	23.2±18.2	93.5±13.6(↑70.3)	59.2±3.5	97.2±5.9(↑38.0)	55.7±22.5	99.5±3.2(↑43.8)	78.5±13.6	96.9±4.9(↑18.4)
Halfcheetah-m	16.4±10.2	31.5±2.4(↑15.1)	37.1±1.3	38.9±0.5(↑1.8)	38.2±1.4	39.1±0.1(↑0.9)	38.9±1.4	39.5±1.5(↑0.6)
Halfcheetah-mr	17.9±9.5	21.2±7.1(↑3.3)	13.0±5.7	15.1±4.1(↑2.1)	28.2±1.3	29.5±0.6(↑1.3)	29.6±1.0	31.1±1.2(↑1.5)
Halfcheetah-me	15.4±10.7	35.6±2.4(↑20.2)	12.2±6.7	19.7±1.7(↑7.5)	34.4±2.2	36.8±1.5(↑2.4)	40.2±0.7	40.4±1.3(↑0.2)
Halfcheetah-e	1.7±3.3	3.9±2.1(↑2.2)	2.3±0.9	4.0±2.3(↑1.7)	1.2±1.0	6.7±2.6(↑5.5)	18.8±7.8	35.9±6.2(↑17.1)
Walker2d-m	7.4±13.1	41.1±10.8(↑33.7)	21.2±4.8	56.9±4.3(↑35.7)	34.6±9.5	45.0±4.8(↑10.4)	50.6±6.8	56.7±4.7(↑6.1)
Walker2d-mr	5.7±5.8	23.2±5.7(↑17.5)	10.0±2.2	22.8±5.5(↑12.8)	10.7±11.9	17.6±4.0(↑6.9)	17.5±5.4	29.0±2.8(↑11.5)
Walker2d-me	7.9±9.1	37.8±12.3(↑29.9)	17.6±8.6	58.9±11.8(↑41.3)	29.1±16.9	41.2±9.2(↑12.1)	38.8±3.7	57.6±10.4(↑18.8)
Walker2d-e	2.7±3.1	82.3±10.8(↑79.5)	23.6±9.4	83.9±6.8(↑60.3)	43.1±6.0	84.8±13.9(↑41.7)	95.2±11.8	104.9±3.7(↑9.7)
Average Score	13.6	39.9(↑26.3)	21.8	41.5(↑19.7)	31.4	42.6(↑10.2)	44.0	51.6(↑7.6)

"standard"). For flow-matching policies, we adopt IFQL, an algorithmic variant of FQL [54] that enables our first-order supervision to be directly applied to the flow policy without any additional policy-distillation step. Experimental details are provided in Appendix E.2.

5.2 Evaluation Results

Evaluation results on OGBench tasks. Table 1 reports the average normalized scores on 50 diverse OGBench tasks, with complete per-task results provided in Appendix B.1. Our first-order action supervision yields consistent performance gains across all three policy paradigms. On control-sensitive navigation tasks, these gains underline the importance of temporally consistent actions, since small action fluctuations can accumulate over time and easily lead to collisions or failed navigation. For instance, applying our scheme to ReBRAC raises the score from 26 to 61 on `antmaze-giant-navigate` and from 22 to 55 on `humanoidmaze-medium-navigate`, both revealing over 100% relative improvement.

The performance gains are even more pronounced beyond navigation. On object-centric play tasks, applying our proposed scheme to IFQL improves `cube-double-play` from 14 to 42 (a 200% relative improvement) and `scene-play` from 30 to 56 (an 87% relative improvement). In particular, on several challenging sub-tasks where the IFQL baseline completely fails with a score of exactly 0, equipping it with our scheme lifts the score to clearly non-trivial levels, e.g., from 0 to 88 on `scene-play-singletask-task2-v0`, expanding the set of tasks on which offline policy learning is feasible at all (see Table 4 in Appendix B.1 for full per-task results).

Improved OOD generalization under small-dataset settings. Our first-order action supervision is particularly beneficial in small-dataset regimes, where standard offline RL baselines often struggle to recover the expert's robust control behavior from sparse state-action samples, even though such samples still encode rich local variations along expert trajectories. Table 2 reports the D4RL locomotion results on 10k datasets. By supervising the policy's first-order action derivative, our scheme enables the local dynamics and control structure of the expert behavior to be effectively recovered from an extremely limited dataset, leading to substantial performance gains. For instance, on

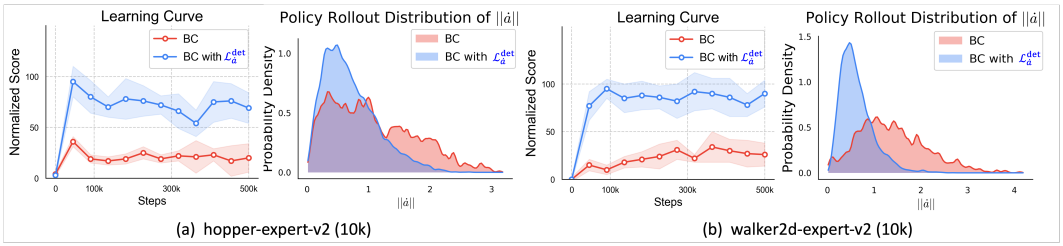

Figure 2: Comparison of vanilla deterministic BC and our 1st-order action supervision on small-sample locomotion tasks. Scores are averaged over 5 random seeds. The policy rollout distributions of action variation $||\dot{a}|| = ||a' - a||$ (a is joint torque) are aggregated from 10 trajectories.

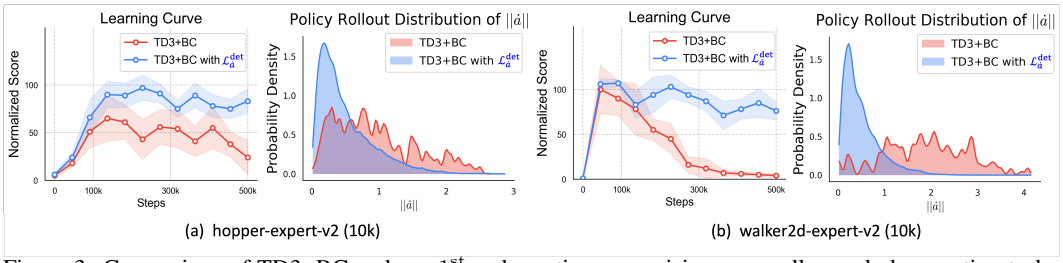

Figure 3: Comparison of TD3+BC and our 1st-order action supervision on small-sample locomotion tasks. Scores are averaged over 5 random seeds. The policy rollout distributions of action variation $||\dot{a}|| = ||a' - a||$ (a is joint torque) are aggregated from 10 trajectories.

`hopper-expert-v2`, IQL improves from 55.7 to 99.5, nearly recovering its full-dataset performance. More notably, on `walker2d-expert-v2`, TD3+BC collapses to a score of only 2.7, indicating severe extrapolation failure under such a tiny-data regime; equipping it with our scheme recovers the score to 82.3, an absolute gain of 79.5 points. In addition to the score improvements, our scheme also yields noticeably smaller standard deviations across most tasks, indicating more stable and reproducible policy learning under small-dataset settings. Additional limited-data imitation learning results, as well as full-data D4RL results, are provided in Appendix B.1.

Benefits of utilizing first-order supervision. Figure 2 provides a mechanistic view of how our first-order action supervision improves performance. With only 10k transitions, deterministic BC produces a dispersed distribution of $||\dot{a}||$, reflecting erratic action changes during execution. In contrast, our scheme yields a compact distribution concentrated around small $||\dot{a}||$, indicating smoother and temporally consistent action evolution. This pattern is even more pronounced for TD3+BC in Figure 3, where its plain MSE penalty leaves the policy vulnerable to action jittering, whereas our scheme directly aligns action changes with the local expert dynamics. Overall, by constraining how actions evolve along trajectories, first-order supervision yields noticeably smoother and more robust policies. We conduct additional analysis results in Appendix B.1.

5.3 Ablation experiments

Comparing with alternative action-consistency approaches. To further benchmark our method, we compare against two categories of representative alternatives, using TD3+BC as the shared backbone. The first category enforces action smoothness, including the penalty-based methods CAPS [13] and L2C2 [25], which encourage local action consistency on perturbed or interpolated states, and the architectural method LipsNet [8], which explicitly bounds the network's Lipschitz constant. The second category explicitly predicts first-order actions, for which we construct an auxiliary-head variant of TD3+BC, denoted as +Extra $\dot{a}$ head, where the actor is equipped with a dedicated prediction head, i.e., learning a policy $\pi(s) = (a, \dot{a})$, and the predicted $\dot{a}$ is also utilized during inference. All baselines follow their original configurations or exactly the same training configurations as our method.

For action-smoothness approaches, as reported in Table 3 (+CAPS/+L2C2/+LipsNet columns), our method achieves consistent improvements across all tasks. Penalty-based methods were originally designed for online settings; in offline RL, enforcing smoothness on synthetic, unobserved states risks OOD extrapolation errors, and this degradation is particularly pronounced on high-quality (*-e*) datasets, whose narrow and near-deterministic trajectories conflict with isotropic perturbations or

Table 3: Comparison with action-smoothness approaches and explicit first-order action prediction on D4RL MuJoCo locomotion tasks with reduced-size (10k) datasets. All methods use TD3+BC as the shared backbone. Arrows show score differences relative to TD3+BC on the same task: ↑ indicates a gain of at least 1.0, ↑ a smaller positive gain, and ↓ a decrease. No arrow indicates no change.

Task	TD3+BC	+CAPS	+L2C2	+LipsNet	+Extra $\dot{a}$ head	TD3+BC+Ours
Hopper-m	40.1±18.6	42.6±5.1(↑2.5)	42.7±7.5(↑2.6)	43.2±3.6(↑3.1)	41.5±7.8(↑1.4)	44.1±12.7(↑**4.0**)
Hopper-me	17.8±7.9	39.6±8.2(↑21.8)	39.8±8.3(↑22.0)	32.8±18.5(↑15.0)	26.6±8.6(↑8.8)	42.3±10.8(↑**24.5**)
Hopper-e	23.2±18.2	57.1±29.7(↑33.9)	55.2±34.5(↑32.0)	16.9±2.8(↓6.3)	30.6±15.1(↑7.4)	93.5±13.6(↑**70.3**)
HalfCheetah-m	16.4±10.2	29.6±5.6(↑13.2)	30.3±7.1(↑13.9)	6.6±3.0(↓9.8)	28.1±4.5(↑11.7)	31.5±2.4(↑**15.1**)
HalfCheetah-me	15.4±10.7	19.8±3.4(↑4.4)	28.6±3.8(↑13.2)	3.3±2.2(↓12.1)	20.6±3.5(↑5.2)	35.6±2.4(↑**20.2**)
HalfCheetah-e	1.7±3.3	1.7±1.6	2.4±1.9(↑0.7)	0.8±0.6(↓0.9)	0.2±1.8(↓1.5)	3.9±2.1(↑**2.2**)
Walker2d-m	7.4±13.1	24.5±3.6(↑17.1)	20.4±4.1(↑13.0)	8.9±2.8(↑1.5)	8.6±6.5(↑1.2)	41.1±10.8(↑**33.7**)
Walker2d-me	7.9±9.1	17.8±10.1(↑9.9)	23.3±9.8(↑15.4)	5.9±3.6(↓2.0)	8.3±6.1(↑0.4)	37.8±12.3(↑**29.9**)
Walker2d-e	2.7±3.1	3.9±2.1(↑1.2)	7.4±3.6(↑4.7)	13.8±6.8(↑11.1)	3.9±1.6(↑1.2)	82.3±10.8(↑**79.5**)
Average Score	14.7	26.3(↑11.6)	27.8(↑13.1)	14.7	18.7(↑4.0)	45.8(↑**31.1**)

uniform interpolations. Meanwhile, strictly constraining the network architecture (LipsNet) limits model expressivity, making it difficult to fit offline data from limited samples.

For explicit first-order action prediction, the auxiliary-head variant brings only marginal improvements over the backbone and substantially underperforms our method (+Extra $\dot{a}$ head column). We attribute this gap to two factors. First, predicting two distinct action outputs converts policy optimization into a multi-task learning problem, which introduces conflicting gradient directions between the zeroth-order and first-order action objectives and requires extra hyperparameter tuning to balance their trade-off. Second, the dedicated head doubles the output dimension and requires structural modification of the base network, losing compatibility with off-the-shelf policy architectures. In contrast, our method requires zero architectural changes, acting as a lightweight, plug-and-play module that preserves temporal consistency without sacrificing network capacity.

Effectiveness of value and policy constraints. Figure 4 isolates the respective contributions of the two components in our IFQL instantiation on OGBench by sweeping one weight while strictly setting the other to zero. The weight λ_Q controls the first-order value-consistency regularizer; sweeping it over $\{0, 0.5, 1\}$ reveals steady performance gains, indicating that aligning the value landscape with the local state-action flow provides a highly beneficial training signal.

Similarly, increasing the first-order action supervision weight $\lambda_{\dot{a}}$ from the same grid $\{0, 0.5, 1\}$ consistently improves performance over standard IFQL. This confirms that first-order action supervision offers crucial guidance beyond mere zeroth-order action fitting, even when the value-side regularizer is completely disabled. Additional ablations are deferred to Appendix B.2.

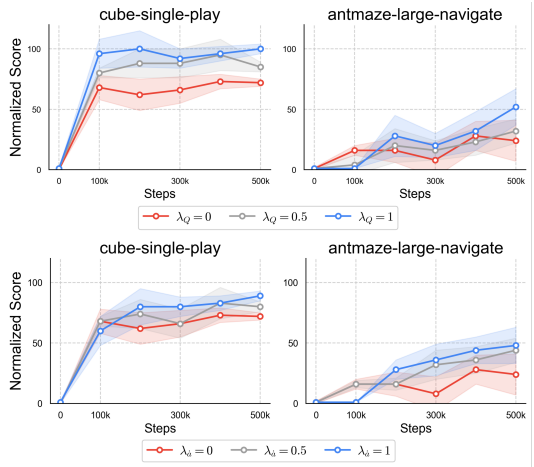

Figure 4: Ablation study on value regularization weight λ_Q and 1st-order action supervision weight $\lambda_{\dot{a}}$ of IFQL.

6 Conclusion

In this paper, we identified the neglect of higher-order action dynamics as a fundamental source of control instability and poor robustness in contemporary offline RL and IL methods. To bridge this gap, we proposed an elegant, plug-and-play learning scheme that enables simultaneous zeroth- and first-order action supervision for any off-the-shelf policy model, including deterministic, stochastic, and generative flow policies, without requiring any structural modifications. Supported by formal theoretical guarantees, our enhanced policy class also integrates seamlessly into the continuous-time HJB framework, providing additional first-order regularization on both value estimation and policy optimization. Extensive empirical evaluations confirm that our scheme yields substantial improvements in control smoothness, robustness, and out-of-distribution generalization, particularly under challenging small-dataset settings, making it a promising tool for real-world control problems.

Acknowledgment

This work was supported by the Wuxi Research Institute of Applied Technologies at Tsinghua University under the Grant 20242001120, Xiongan AI Institute, and Horizon Robotics.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[2] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[3] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

[4] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

[5] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 international conference on robotics and automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[6] Yinan Zheng, Ruiming Liang, Kexin ZHENG, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Diffusion-based planning for autonomous driving with flexible guidance. In *The Thirteenth International Conference on Learning Representations*, 2025.

[7] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

[8] Xujie Song, Jingliang Duan, Wenxuan Wang, Shengbo Eben Li, Chen Chen, Bo Cheng, Bo Zhang, Junqing Wei, and Xiaoming Simon Wang. Lipsnet: A smooth and robust neural network with adaptive lipschitz constant for high accuracy optimal control. In *International Conference on Machine Learning*, pages 32253–32272. PMLR, 2023.

[9] Peng Cheng, Xianyuan Zhan, Zhihao Wu, Wenjia Zhang, Youfang Lin, Han Wang, Li Jiang, et al. Look beneath the surface: Exploiting fundamental symmetry for sample-efficient offline rl. *Advances in Neural Information Processing Systems*, 36, 2023.

[10] Peng Cheng, Zhihao Wu, Jianxiong Li, Ziteng He, Haoran Xu, Wei Sun, Youfang Lin, Yunxin Liu, and Xianyuan Zhan. Sample efficient offline rl via t-symmetry enforced latent state-stitching. In *International Conference on Learning Representations*, volume 2026, pages 139752–139781, 2026.

[11] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.

[12] Xujie Song, Liangfa Chen, Tong Liu, Wenxuan Wang, Yinuo Wang, Shentao Qin, Yinsong Ma, Jingliang Duan, and Shengbo Eben Li. Lipsnet++: Unifying filter and controller into a policy network. In *Forty-second International Conference on Machine Learning*, 2025.

[13] Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. Regularizing action policies for smooth control with reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1810–1816. IEEE, 2021.

[14] Román de La Presilla, Sebastian Wandel, Matthias Stammler, Markus Grebe, Gerhard Poll, and Sergei Glavatskih. Oscillating rolling element bearings: A review of tribotesting and analysis approaches. *Tribology International*, 188:108805, 2023.

[15] Xianyuan Zhan, Haoran Xu, Yue Zhang, Xiangyu Zhu, Honglei Yin, and Yu Zheng. Deepthermal: Combustion optimization for thermal power generating units using offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4680–4688, 2022.

[16] Xianyuan Zhan, Xiangyu Zhu, Peng Cheng, Xiao Hu, Ziteng He, Hanfei Geng, Jichao Leng, Huiwen Zheng, Chenhui Liu, Tianshun Hong, et al. Data center cooling system optimization using offline reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pages 47735–47757, 2025.

[17] Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 international conference on robotics and automation (icra)*, pages 9784–9790. IEEE, 2019.

[18] Haonan Yu, Wei Xu, and Haichao Zhang. Taac: Temporally abstract actor-critic for continuous control. *Advances in neural information processing systems*, 34:29021–29033, 2021.

[19] Yang Zhang, Buqing Nie, and Yue Gao. Robust locomotion policy with adaptive lipschitz constraint for legged robots. *IEEE Robotics and Automation Letters*, 10(1):272–279, 2024.

[20] Peide Cai, Xiaodong Mei, Lei Tai, Yuxiang Sun, and Ming Liu. High-speed autonomous drifting with deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):1247–1254, 2020.

[21] Chen Chen, Hongyao Tang, Jianye Hao, Wulong Liu, and Zhaopeng Meng. Addressing action oscillations through learning policy inertia. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7020–7027, 2021.

[22] Asanka Wasala, Donal Byrne, Philip Miesbauer, Joseph O'Hanlon, Paul Heraty, and Peter Barry. Trajectory based lateral control: A reinforcement learning case study. *Engineering Applications of Artificial Intelligence*, 94:103799, 2020.

[23] I Lee, Hoang-Giang Cao, Cong-Tinh Dao, Yu-Cheng Chen, and I-Chen Wu. Gradient-based regularization for action smoothness in robotic control with reinforcement learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 603–610. IEEE, 2024.

[24] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020.

[25] Taisuke Kobayashi. L2c2: Locally lipschitz continuous constraint towards stable and smooth reinforcement learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4032–4039. IEEE, 2022.

[26] Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, pages 8707–8718. PMLR, 2020.

[27] Zhigen Zhao, Simiao Zuo, Tuo Zhao, and Ye Zhao. Adversarially regularized policy learning guided by trajectory optimization. In *Learning for Dynamics and Control Conference*, pages 844–857. PMLR, 2022.

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

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

[30] Guilherme Christmann, Ying-Sheng Luo, Hanjaya Mandala, and Wei-Chao Chen. Benchmarking smoothness and reducing high-frequency oscillations in continuous control policies. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 627–634. IEEE, 2024.

[31] Daniel Pfrommer, Thomas Zhang, Stephen Tu, and Nikolai Matni. Tasil: Taylor series imitation learning. *Advances in Neural Information Processing Systems*, 35:20162–20174, 2022.

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

[33] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

[34] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[35] Jianxiong Li, Xianyuan Zhan, Haoran Xu, Xiangyu Zhu, Jingjing Liu, and Ya-Qin Zhang. When data geometry meets deep function: Generalizing offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

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

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

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

[39] Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1711–1724, 2022.

[40] 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 *The Eleventh International Conference on Learning Representations*, 2023.

[41] Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. Offline multi-agent reinforcement learning with implicit global-to-local value regularization. In *Advances in Neural Information Processing Systems*, 2023.

[42] Haoran Xu, Li Jiang, Jianxiong Li, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

[43] Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[44] Haoyi Niu, Shubham Sharma, Yiwen Qiu, Ming Li, Guyue Zhou, Jianming HU, and Xianyuan Zhan. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

[45] Jianxiong Li, Xiao Hu, Haoran Xu, Jingjing Liu, Xianyuan Zhan, Qing-Shan Jia, and Ya-Qin Zhang. Mind the gap: Offline policy optimization for imperfect rewards. In *The Eleventh International Conference on Learning Representations*, 2023.

[46] Liyuan Mao, Haoran Xu, Weinan Zhang, and Xianyuan Zhan. Odice: Revealing the mystery of distribution correction estimation via orthogonal-gradient update. In *The Twelfth International Conference on Learning Representations*, 2024.

[47] Haoran Xu, Liyuan Mao, Hui Jin, Weinan Zhang, Xianyuan Zhan, and Amy Zhang. Uni-rl: Unifying online and offline rl via implicit value regularization. *Advances in Neural Information Processing Systems*, 38:133162–133187, 2026.

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

[49] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

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

[51] Guanghe Li, Yixiang Shan, Zhengbang Zhu, Ting Long, and Weinan Zhang. Diffstitch: Boosting offline reinforcement learning with diffusion-based trajectory stitching. In *Forty-first International Conference on Machine Learning*, pages 28597 – 28609. PMLR, 2024.

[52] Liyuan Mao, Haoran Xu, Xianyuan Zhan, Weinan Zhang, and Amy Zhang. Diffusion-dice: In-sample diffusion guidance for offline reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[53] Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. In *The Twelfth International Conference on Learning Representations*, 2024.

[54] Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning*, 2025.

[55] Kexin Zheng, Lauriane Teyssier, Yinan Zheng, Yu Luo, and Xianyuan Zhan. Towards robust zero-shot reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[56] Ruiming Liang, Yinan Zheng, Kexin Zheng, Tianyi Tan, Jianxiong Li, Liyuan Mao, Zhihao Wang, Guang Chen, Hangjun Ye, Jingjing Liu, Jinqiao Wang, and Xianyuan Zhan. Dichotomous diffusion policy optimization. In *International Conference on Learning Representations*, 2026.

[57] Kexin Zheng, Xianyuan Zhan, and Xintao Yan. Horizon adaptive offline policy learning via value stitching. In *The Fortieth Annual Conference on Neural Information Processing Systems*, 2026.

[58] Kenji Doya. Reinforcement learning in continuous time and space. *Neural Computation*, 12(1):219–245, 2000.

[59] Haoran Wang, Thaleia Zariphopoulou, and Xun Yu Zhou. Reinforcement learning in continuous time and space: A stochastic control approach. *Journal of Machine Learning Research*, 21(198):1–34, 2020.

[60] Rémi Munos. Policy gradient in continuous time. *Journal of Machine Learning Research*, 7:771–791, 2006.

[61] Jaeyoung Lee and Richard S. Sutton. Policy iterations for reinforcement learning problems in continuous time and space—fundamental theory and methods. *Automatica*, 126:109421, 2021.

[62] Kyriakos G. Vamvoudakis and Frank L. Lewis. Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, 46(5):878–888, 2010.

[63] Shubhendu Bhasin, Rushikesh Kamalapurkar, Michael Johnson, Kyriakos G. Vamvoudakis, Frank L. Lewis, and Warren E. Dixon. A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica*, 49(1):82–92, 2013.

[64] Kyriakos G. Vamvoudakis, Draguna Vrabie, and Frank L. Lewis. Online adaptive algorithm for optimal control with integral reinforcement learning. *International Journal of Robust and Nonlinear Control*, 24(17):2686–2710, 2014.

[65] Jeongho Kim, Jaeuk Shin, and Insoon Yang. Hamilton–jacobi deep q-learning for deterministic continuous-time systems with lipschitz continuous controls. *Journal of Machine Learning Research*, 22(206):1–34, 2021.

[66] Yanwei Jia and Xun Yu Zhou. Policy evaluation and temporal-difference learning in continuous time and space: A martingale approach. *Journal of Machine Learning Research*, 23(154):1–55, 2022.

[67] Yun-Hsuan Lien, Ping-Chun Hsieh, Tzu-Mao Li, and Yu-Shuen Wang. Enhancing value function estimation through first-order state-action dynamics in offline reinforcement learning. In *Forty-first International Conference on Machine Learning*, 2024.

[68] George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

[69] Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

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

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

[72] Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. Offline multi-agent reinforcement learning with implicit global-to-local value regularization. *Advances in Neural Information Processing Systems*, 36, 2024.

[73] Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:11592–11620, 2023.

[74] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[75] Carl de Boor. *A Practical Guide to Splines*. Applied Mathematical Sciences. Springer, New York, NY, 2001.

[76] Abraham Savitzky and Marcel J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[77] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018.

[78] Etienne Emmrich. Discrete versions of gronwall's lemma and their application to the numerical analysis of parabolic problems. Preprint, Technische Universität Berlin, Fachbereich Mathematik, 1999.

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

[80] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

A Limitations

In this section, we summarize the limitations of our proposed method and outline potential directions for future research. First, applying first-order supervision directly in raw pixel space can be computationally expensive and semantically noisy. The issue is not that higher-order supervision is incompatible with visual inputs, but that pixels are a poor coordinate system for measuring dynamics and control evolution. A practical remedy is to apply the same supervision after mapping observations into a compact state representation [74], where derivatives reflect task-relevant changes rather than appearance variation. Pixel-level higher-order supervision therefore requires a good representation. Second, our higher-order supervision relies on finite-difference estimates of local derivatives, i.e., $\dot{s} = s' - s$. With coarse sampling, these estimated tangents may provide poor approximations of the true local dynamics. This issue can be alleviated by reconstructing smoother trajectories via interpolation [75] or filtering [76] before applying supervision.

B Additional Results

In this section, we first present the detailed evaluation results on OGBench in Table 4, followed by the performance scores on the D4RL benchmark under the full-dataset setting in Table 5. Additionally, we report the imitation learning results in Table 6. Finally, Section B.2 provides more ablation studies.

B.1 Additional Evaluation Results

Full Evaluation results on OGBench. We present the comprehensive per-task results on OGBench in Table 4. Note that the baseline performance of TD3+BC [34] is evaluated using the hyperparameters provided in Appendix E.2, while the baseline results for ReBRAC, IQL, and IFQL are directly sourced from FQL [54].

Table 4: Average normalized scores on 50 OGBench tasks over 5 random seeds. Values after ± denote standard deviations. The ↑ marks indicate improvements greater than 1.0, whereas ↑ denotes minor improvements.

	Det. Policy		Sto. Policy (mean sup.)		Sto. Policy (standard)		Flow Policy	
Task Category	TD3+BC	TD3+BC+ours	ReBRAC	ReBRAC+ours	IQL	IQL+ours	IFQL	IFQL+ours
antmaze-large-navigate-singletask-task1-v0 (∗)	92±8	99±2 (↑7)	91±10	100±1 (↑9)	48±9	80±10 (↑32)	24±17	56±14 (↑32)
antmaze-large-navigate-singletask-task2-v0	85±3	91±4 (↑6)	88±4	92±4 (↑4)	42±6	42±15	8±3	16±11 (↑8)
antmaze-large-navigate-singletask-task3-v0	72±10	100±0 (↑28)	51±18	100±1 (↑49)	72±7	90±6 (↑18)	52±17	72±10 (↑20)
antmaze-large-navigate-singletask-task4-v0	85±5	99±1 (↑14)	84±7	100±1 (↑16)	51±9	52±22 (↑1)	18±8	40±17 (↑22)
antmaze-large-navigate-singletask-task5-v0	82±10	99±0 (↑17)	90±2	99±1 (↑9)	54±22	78±9 (↑24)	38±18	48±19 (↑10)
antmaze-giant-navigate-singletask-task1-v0 (∗)	52±20	57±19 (↑5)	27±22	39±21 (↑12)	0±0	0±0	0±0	0±0
antmaze-giant-navigate-singletask-task2-v0	47±30	71±36 (↑24)	16±17	70±26 (↑54)	1±1	2±1 (↑1)	0±0	0±0
antmaze-giant-navigate-singletask-task3-v0	44±13	44±12	34±22	40±15 (↑6)	0±0	0±0	0±0	0±0
antmaze-giant-navigate-singletask-task4-v0	3±4	63±31 (↑60)	5±12	67±33 (↑62)	0±0	1±1 (↑1)	0±0	0±0
antmaze-giant-navigate-singletask-task5-v0	65±7	92±11 (↑27)	49±22	96±3 (↑47)	19±7	41±6 (↑22)	13±9	21±17 (↑8)
humanoidmaze-medium-navigate-singletask-task1-v0 (∗)	24±11	26±4 (↑2)	16±9	36±19 (↑20)	32±7	34±5 (↑2)	69±19	71±16 (↑2)
humanoidmaze-medium-navigate-singletask-task2-v0	45±13	49±9 (↑4)	18±16	68±24 (↑50)	41±9	51±5 (↑10)	85±11	91±17 (↑6)
humanoidmaze-medium-navigate-singletask-task3-v0	40±26	60±19 (↑20)	36±13	66±22 (↑30)	25±5	27±6 (↑2)	49±49	56±32 (↑7)
humanoidmaze-medium-navigate-singletask-task4-v0	30±20	44±9 (↑14)	15±16	30±21 (↑15)	0±1	0±0	1±1	1±1
humanoidmaze-medium-navigate-singletask-task5-v0	31±13	31±7	24±20	78±16 (↑54)	66±4	66±4	98±2	100±1 (↑2)
humanoidmaze-large-navigate-singletask-task1-v0 (∗)	3±3	3±2	2±1	7±7 (↑5)	3±1	4±1 (↑1)	6±2	8±4 (↑2)
humanoidmaze-large-navigate-singletask-task2-v0	0±0	0±0	0±0	0±0	0±0	0±0	0±0	0±0
humanoidmaze-large-navigate-singletask-task3-v0	8±4	17±6 (↑9)	8±4	11±6 (↑3)	7±3	7±1	48±10	52±16 (↑4)
humanoidmaze-large-navigate-singletask-task4-v0	1±1	2±1 (↑1)	1±1	2±1 (↑1)	1±0	2±1 (↑1)	1±1	1±1
humanoidmaze-large-navigate-singletask-task5-v0	0±0	1±1 (↑1)	2±2	2±1	1±1	3±1 (↑2)	0±0	0±0
antsoccer-arena-navigate-singletask-task1-v0	9±3	11±5 (↑2)	0±0	13±6 (↑13)	14±5	36±10 (↑22)	61±26	68±21 (↑7)
antsoccer-arena-navigate-singletask-task2-v0	19±8	19±2	0±1	13±4 (↑13)	17±7	37±5 (↑20)	75±3	83±12 (↑8)
antsoccer-arena-navigate-singletask-task3-v0	25±8	27±10 (↑2)	0±0	32±11 (↑32)	6±4	19±7 (↑13)	14±22	22±18 (↑8)
antsoccer-arena-navigate-singletask-task4-v0 (∗)	1±1	1±1	0±0	20±4 (↑20)	3±2	9±3 (↑6)	16±9	16±8
antsoccer-arena-navigate-singletask-task5-v0	12±7	20±9 (↑8)	0±0	6±5 (↑6)	2±2	11±7 (↑9)	0±1	8±7 (↑8)
cube-single-play-singletask-task1-v0	89±3	94±2 (↑5)	89±5	94±2 (↑5)	88±3	90±5 (↑2)	79±4	96±9 (↑17)
cube-single-play-singletask-task2-v0 (∗)	87±5	88±2 (↑1)	92±4	93±4 (↑1)	85±8	93±3 (↑8)	73±3	100±1 (↑27)
cube-single-play-singletask-task3-v0	93±3	96±3 (↑3)	93±3	94±2 (↑1)	91±5	95±5 (↑4)	88±4	100±1 (↑12)
cube-single-play-singletask-task4-v0	87±4	89±3 (↑2)	92±3	92±4	73±6	90±5 (↑17)	79±6	88±16 (↑9)
cube-single-play-singletask-task5-v0	85±5	85±1	87±8	90±4 (↑3)	78±9	90±6 (↑12)	77±7	92±10 (↑15)
cube-double-play-singletask-task1-v0	30±4	46±6 (↑16)	45±6	45±6	27±5	33±7 (↑6)	35±9	76±8 (↑41)
cube-double-play-singletask-task2-v0 (∗)	6±1	9±1 (↑3)	7±3	32±7 (↑25)	1±1	2±1 (↑1)	9±5	40±22 (↑31)
cube-double-play-singletask-task3-v0	1±1	2±1 (↑1)	4±1	17±5 (↑13)	0±0	1±1 (↑1)	8±5	25±13 (↑17)
cube-double-play-singletask-task4-v0	0±1	1±1 (↑1)	1±1	3±1 (↑2)	0±0	0±0	1±1	12±10 (↑11)
cube-double-play-singletask-task5-v0	1±1	1±1	4±2	22±7 (↑18)	4±3	4±2	17±6	55±21 (↑38)
scene-play-singletask-task1-v0	95±3	96±1 (↑1)	95±2	95±3	94±3	95±2 (↑1)	98±3	99±2 (↑1)
scene-play-singletask-task2-v0 (∗)	54±10	58±4 (↑4)	50±13	60±13 (↑10)	12±3	17±6 (↑5)	0±0	88±17 (↑88)
scene-play-singletask-task3-v0	68±10	78±7 (↑10)	55±16	75±10 (↑20)	32±7	52±9 (↑20)	54±19	92±17 (↑38)
scene-play-singletask-task4-v0	4±3	4±2	3±3	5±3 (↑2)	0±1	1±1 (↑1)	0±0	0±0
scene-play-singletask-task5-v0	0±0	0±0	0±0	0±0	0±0	0±0	0±0	0±0
puzzle-3x3-play-singletask-task1-v0	96±3	98±3 (↑2)	97±4	98±3 (↑1)	33±6	39±17 (↑6)	94±3	100±1 (↑6)
puzzle-3x3-play-singletask-task2-v0	13±9	14±4 (↑1)	1±1	4±2 (↑3)	4±3	9±6 (↑5)	1±2	60±32 (↑59)
puzzle-3x3-play-singletask-task3-v0	5±6	6±1 (↑1)	3±1	4±4 (↑1)	3±2	3±1	0±0	28±17 (↑28)
puzzle-3x3-play-singletask-task4-v0 (∗)	7±4	11±4 (↑4)	2±1	4±2 (↑2)	2±1	2±1	0±0	16±12 (↑16)
puzzle-3x3-play-singletask-task5-v0	14±7	15±2 (↑1)	5±3	11±6 (↑6)	3±2	4±2 (↑1)	0±0	12±11 (↑12)
puzzle-4x4-play-singletask-task1-v0	1±2	14±2 (↑13)	26±4	33±9 (↑7)	12±2	20±6 (↑8)	49±9	53±6 (↑4)
puzzle-4x4-play-singletask-task2-v0	7±3	9±2 (↑2)	12±4	13±7 (↑1)	7±4	10±3 (↑3)	4±4	12±9 (↑8)
puzzle-4x4-play-singletask-task3-v0	3±2	8±2 (↑5)	15±3	19±6 (↑4)	9±3	14±6 (↑5)	50±14	84±16 (↑34)
puzzle-4x4-play-singletask-task4-v0 (∗)	5±2	7±2 (↑2)	10±3	10±4	5±2	10±2 (↑5)	21±11	30±17 (↑9)
puzzle-4x4-play-singletask-task5-v0	6±1	6±1	7±3	8±4 (↑1)	4±1	8±5 (↑4)	2±2	21±14 (↑19)

Table 5: Average normalized scores on D4RL MuJoCo locomotion tasks under full-datasize. Standard deviations are reported after the $\pm$ sign. The $\uparrow$ symbol indicates a performance increase greater than 1.0, whereas $\uparrow$ denotes minor improvements. All results and standard deviations ($\pm$) are averaged across 5 random seeds.

	Det. Policy		Sto. Policy (mean sup.)		Sto. Policy (standard)		Flow Policy	
Task Category	TD3+BC	TD3+BC+ours	ReBRAC	ReBRAC+ours	IQL	IQL+ours	IFQL	IFQL+ours
Hopper-m	55.5±7.7	58.9±7.4(↑3.4)	84.9±2.6	85.7±1.3(↑0.8)	67.3±5.7	68.8±5.5(↑1.5)	89.5±5.7	94.2±7.2(↑4.7)
Hopper-mr	57.9±17.6	58.4±9.2(↑0.5)	81.2±25.1	94.3±10.1(↑13.1)	91.7±8.6	94.2±5.5(↑2.5)	96.2±4.7	96.3±2.3(↑0.1)
Hopper-me	89.8±15.2	89.9±8.4(↑0.1)	106.4±10.7	106.6±5.4(↑0.2)	91.5±14.3	109.2±9.8(↑17.7)	77.1±25.5	89.8±8.7(↑12.7)
Hopper-e	108.3±6.7	110.5±0.2(↑2.2)	108.5±4.1	108.6±3.8(↑0.1)	99.3±9.2	106.5±5.1(↑7.2)	96.8±12.2	106.2±8.6(↑9.4)
Halfcheetah-m	47.8±0.4	47.9±0.4(↑0.1)	55.4±1.0	56.9±1.1(↑1.5)	47.4±0.4	48.6±0.2(↑1.2)	51.1±0.6	52.8±0.3(↑1.7)
Halfcheetah-mr	41.6±0.3	42.3±0.2(↑0.7)	46.6±0.3	46.7±0.4(↑0.1)	41.3±2.3	41.4±2.0(↑0.1)	42.5±0.4	43.6±1.3(↑1.1)
Halfcheetah-me	88.1±5.1	89.5±4.7(↑1.4)	96.2±1.6	97.4±1.4(↑1.2)	86.7±3.4	87.4±2.6(↑0.7)	88.4±2.2	89.5±1.6(↑1.1)
Halfcheetah-e	90.1±0.3	93.4±0.4(↑3.3)	91.3±1.0	91.7±0.3(↑0.4)	89.6±7.8	91.4±1.2(↑1.8)	91.2±1.5	91.6±1.7(↑0.4)
Walker2d-m	79.7±4.1	80.1±1.6(↑0.4)	76.7±9.9	82.6±2.2(↑5.9)	71.5±10.8	78.8±5.0(↑7.3)	79.5±4.1	82.6±4.4(↑3.1)
Walker2d-mr	67.2±16.4	73.3±4.1(↑6.1)	70.2±11.5	83.6±7.1(↑13.4)	71.5±8.2	73.9±5.5(↑2.4)	72.5±19.1	83.1±7.1(↑10.6)
Walker2d-me	108.8±0.5	108.9±0.3(↑0.1)	110.3±1.4	110.4±0.1(↑0.1)	108.4±0.7	108.5±0.5(↑0.1)	106.1±1.0	107.5±0.7(↑1.4)
Walker2d-e	108.6±0.2	109.3±0.1(↑0.7)	102.2±1.5	109.7±0.5(↑7.5)	110.3±1.7	110.4±0.5(↑0.1)	102.7±11.5	106.4±0.5(↑3.7)

Table 6: Imitation learning performance on D4RL MuJoCo locomotion tasks with 10k datasets. Average normalized scores across deterministic, stochastic, and flow-based policy paradigms. The $\uparrow$ indicates a performance increase > 1.0, whereas $\uparrow$ represents marginal gains. All results and standard deviations ($\pm$) are averaged across 5 random seeds.

	Det. Policy		Sto. Policy (standard)		Flow Policy	
Task Category	base	base + $\mathcal{L}_{\dot{a}}^{\text{det}}$	base	base + $\mathcal{L}_{\dot{a}}^{\text{sto}}$	base	base + $\mathcal{L}_{\dot{a}}^{\text{flow}}$
Hopper-e	21.5±13.5	67.3±14.3(↑45.8)	57.5±15.2	93.1±10.7(↑35.6)	72.2±9.2	97.2±12.3(↑25.0)
Halfcheetah-e	2.4±1.2	5.7±1.5(↑3.3)	2.8±1.6	16.2±10.1(↑13.4)	11.4±4.5	17.9±6.9(↑6.5)
Walker2d-e	26.7±12.2	91.1±14.7(↑64.4)	46.4±15.2	91.6±9.5(↑45.2)	89.8±14.6	91.5±6.3(↑1.7)

The empirical results show that integrating our first-order action supervision yields consistent performance improvements across a diverse range of offline policy optimization paradigms. Notably, our scheme can improve policy learning in cases where baseline algorithms fundamentally fail. On several complex, sparse-reward tasks, baselines collapse and obtain a score of exactly 0, indicating that they cannot extract any meaningful learning signal from the offline dataset. Equipping the same algorithm with our first-order supervision lifts these scores from 0 to clearly non-trivial levels. For example, on `scene-play-singletask-task2-v0`, IFQL completely fails (0), whereas IFQL with our first-order supervision reaches 88, a near-expert level. The same breakthrough is observed on the `puzzle-3x3-play` domain (e.g., `task3-v0` and `task4-v0`), where the baselines obtain no useful learning signal (scoring 0), while the augmented policies acquire the temporal consistency required to achieve non-trivial success rates. Beyond raising scores on already-learnable tasks, our scheme thus expands the set of tasks on which offline policy learning is feasible at all.

The improvements are also consistent across all three policy paradigms. On high-dimensional dexterous manipulation tasks such as `cube-double-play-singletask-task2-v0`, our scheme raises the score of IFQL from 9 to 40. On locomotion and navigation tasks such as `antmaze-giant-navigate-singletask-task5-v0`, integrating our scheme into ReBRAC further raises the score from 49 to 96, approaching expert-level proficiency with noticeably reduced seed-level variance. Together, these results highlight the broad applicability of our scheme and indicate that supervising higher-order action dynamics is crucial for stable offline policy learning and reliable closed-loop execution.

Evaluation results on D4RL under the full-dataset setting. To further evaluate the effectiveness of our scheme, we additionally conduct experiments on the D4RL MuJoCo locomotion tasks under the standard full-size dataset setting. As reported in Table 5, integrating our first-order action supervision yields consistent performance improvements across all four base algorithms, which further confirms the broad applicability of our scheme, showing it reliably stabilizes and improves offline policy learning across varying dataset sizes.

Evaluation results of imitation learning methods on the D4RL 10k setting. Within each policy paradigm, "base" denotes the standard behavior cloning baseline natively implemented under the corresponding framework, while "base + $\mathcal{L}_{\dot{a}}^{(\cdot)}$" denotes the same baseline equipped with our first-order action supervision loss. As shown in Table 6, our first-order supervision consistently improves all three behavior cloning baselines, demonstrating its standalone effectiveness even without any reward signal or RL-based optimization objective. In particular, on the deterministic `Walker2d-e` task, adding $\mathcal{L}_{\dot{a}}^{\text{det}}$ raises the standard BC score from 26.7 to 91.1, an absolute improvement of 64.4.

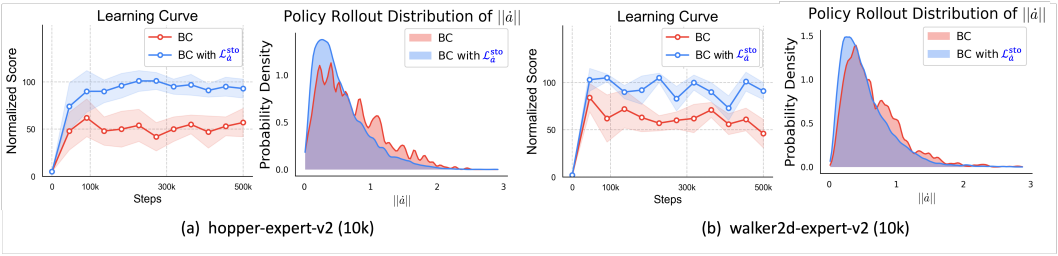

Figure 5: Comparison of vanilla stochastic BC and our 1st-order action supervision on small-sample locomotion tasks. Scores are averaged over 5 random seeds. The policy rollout distributions of action variation $||\dot{a}|| = ||a' - a||$ (a is joint torque) are aggregated from 10 trajectories.

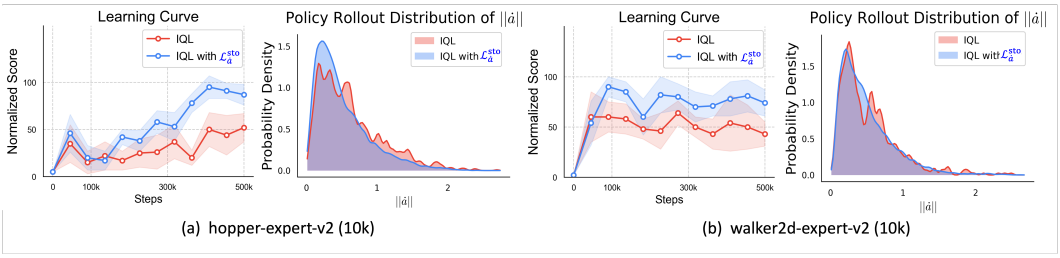

Figure 6: Comparison of vanilla IQL and our 1st-order action supervision on small-sample locomotion tasks. Scores are averaged over 5 random seeds. The policy rollout distributions of action variation $||\dot{a}|| = ||a' - a||$ (a is joint torque) are aggregated from 10 trajectories.

Such a substantial gain reveals that, under small-dataset settings, standard BC is highly susceptible to compounding errors along the rollout trajectory. By directly supervising higher-order action dynamics, our scheme provides the missing temporal regularization that aligns rollout actions with expert dynamics and effectively recovers the expert behavior. More detailed ablation studies that further validate the mechanism of first-order supervision are provided in the following subsections.

Learning curves and policy rollout distributions of $||\dot{a}||$. We additionally present learning curves and rollout distributions of $||\dot{a}||$ for several baseline policies and their first-order action supervised variants, covering both imitation learning and offline RL settings. These results extend the analysis in Figure 2 and Figure 3 to additional algorithms and tasks, and they consistently show that adding our first-order supervision yields smoother learning curves and more concentrated $||\dot{a}||$ distributions, confirming that our scheme produces temporally consistent action evolution beyond the cases highlighted in the main text.

Training time comparison. Table 7 compares the training time of each base algorithm against its variant equipped with our first-order action supervision on `hopper-medium-v2` (D4RL) and `cube-double` (OGBench). Across all four policy parameterizations, integrating our scheme only moderately increases the training time, indicating that our scheme is computationally efficient and practical for real-world use. All algorithms are trained on a workstation equipped with an Intel Xeon Platinum 8336C CPU @ 2.30 GHz and NVIDIA A100-SXM4-80GB GPUs, running Ubuntu 22.04.5 LTS 64-bit.

Table 7: Training time comparison between each base algorithm and its variant equipped with our first-order action supervision. All runs use the same number of training steps and are conducted on a single GPU.

	Det. Policy		Sto. Policy (mean sup.)		Sto. Policy (standard)		Flow Policy	
Task Category	TD3+BC	TD3+BC+ours	ReBRAC	ReBRAC+ours	IQL	IQL+ours	IFQL	IFQL+ours
Training time (D4RL)	28 min 30 s	33 min 6 s	25 min 8 s	30 min 34 s	34 min 44 s	39 min 56 s	30 min 49 s	35 min 52 s
Training time (OGBench)	28 min 23 s	38 min 56 s	25 min 11 s	31 min 54 s	35 min 7 s	42 min 5 s	32 min 0 s	36 min 51 s

Generalizing to online RL. To evaluate the versatility of our approach beyond the offline setting, we integrate both the first-order action supervision and the continuous-time HJB value regularization into the standard SAC [77] algorithm, and evaluate on three MuJoCo benchmarks over 1M training steps. We keep SAC's original hyperparameters intact, and set $\lambda_{\dot{a}} = \lambda_Q = 0.1$ for Hopper-v2 and Walker2d-v2, and $\lambda_{\dot{a}} = 0.01, \lambda_Q = 0.1$ for HalfCheetah-v2.

Table 8: Performance snapshots of SAC and SAC+Ours on online MuJoCo benchmarks over 1M training steps. Results are averaged over 5 random seeds, with standard deviations reported after the ± sign. **Bold** indicates the better result at each snapshot.

Environment	Method	100k	300k	500k	700k	1M
Hopper-v2	SAC	1767.1±1005.9	2399.1±1391.8	3076.8±405.9	**3389.6±162.2**	3530.4±32.6
	SAC + Ours	**2375.8±1036.4**	**3273.9±111.6**	**3226.4±101.1**	3359.2±116.4	**3531.8±33.2**
HalfCheetah-v2	SAC	5659.2±676.4	8564.9±1699.7	9750.7±1228.6	10805.1±804.7	11557.4±493.4
	SAC + Ours	**5790.9±553.5**	**8614.4±1423.9**	**9954.8±1126.3**	**10893.5±612.1**	**11858.2±414.1**
Walker2d-v2	SAC	702.9±343.5	2413.7±1150.3	2989.8±1820.9	4291.2±950.5	4768.9±393.1
	SAC + Ours	**869.9±310.3**	**3300.3±879.6**	**4141.4±572.9**	**4805.8±184.5**	**4991.1±264.5**

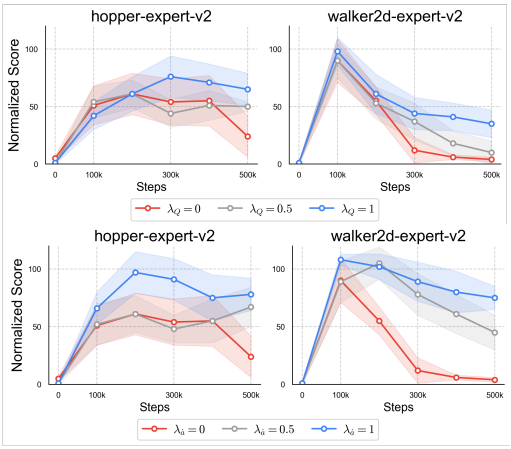

Figure 7: Ablation study on value regularization weight λ_Q and 1st-order action supervision weight $\lambda_{\dot{a}}$ of TD3+BC.

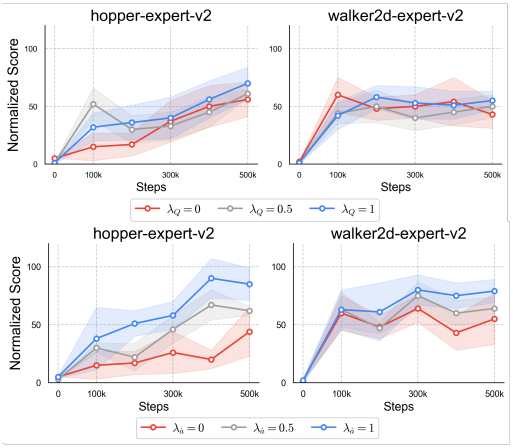

Figure 8: Ablation study on value regularization weight λ_Q and 1st-order action supervision weight $\lambda_{\dot{a}}$ of IQL

As reported in Table 8, incorporating our first-order supervision leads to noticeably faster convergence, which is particularly evident at 300k steps, alongside reduced performance variance across seeds. These results confirm that our lightweight, plug-and-play module generalizes effectively to the online RL paradigm, improving sample efficiency and training stability without modifying the base algorithm.

B.2 Additional Ablation Results

To verify that the benefits of our scheme are not specific to a single algorithmic instantiation, we conduct ablation studies on TD3+BC (deterministic policy) and IQL (stochastic policy), evaluated on the D4RL `hopper-expert-v2` and `walker2d-expert-v2` tasks under the 10k dataset setting. We sweep one of the two weights over $\{0, 0.5, 1\}$ while strictly setting the other to zero, so that the contribution of each component can be attributed independently. All results are averaged over 5 random seeds.

As shown in Figures 7 and 8, both algorithms exhibit the same trend: increasing either the first-order action supervision weight $\lambda_{\dot{a}}$ or the value regularization weight λ_Q from 0 to 1 yields steady performance gains. This confirms that the policy-side and value-side first-order regularizers are robustly beneficial across both deterministic and stochastic policy classes.

To evaluate the action-dependent term omitted from Eq. (7), we compare Ours with an ablation denoted as AK. AK adds $\frac{\partial v(k,s,a^k)}{\partial a^k}\dot{a}^k$ from Eq. (6) to the state-directional term, uses the same objective as Eq. (7), replacing $\mathcal{L}_{\dot{a}}^{\text{flow}}$ with

$$\mathcal{L}_{\dot{a}}^{\text{AK}} = \left\| \frac{\partial v(k,s,a^k)}{\partial s}\dot{s} + \frac{\partial v(k,s,a^k)}{\partial a^k}\dot{a}^k - \dot{a}^1 \right\|. \tag{12}$$

Here $\dot{a}^k = k\dot{a}^1$ on the linear interpolation path.

Table 9: Performance on OGBench environments. Results are reported as mean ± standard deviation. **Bold** indicates the highest mean in each environment.

Environment (OGBENCH)	IFQL	+Ours	+AK ($\lambda_a = 0.5$)	+AK ($\lambda_a = 0.1$)	+AK ($\lambda_a = 1$)
humanoidmaze-medium-navigate-singletask-task1-v0	69±19	71±16	**73.2±7.0**	51.2±32.4	67.6±7.1
humanoidmaze-medium-navigate-singletask-task4-v0	**1±1**	**1±1**	0.0±0.0	0.4±0.8	0.0±0.0
antsoccer-arena-navigate-singletask-task2-v0	75±3	**83±12**	66.8±15.3	40.4±33.6	35.2±20.4
antsoccer-arena-navigate-singletask-task4-v0	16±9	16±8	13.6±9.6	**20.4±20.8**	6.4±7.3
cube-single-play-singletask-task2-v0	73±3	**100±1**	90.0±7.0	27.2±33.5	94.0±2.8
cube-single-play-singletask-task4-v0	79±6	88±16	**93.6±2.3**	26.8±34.2	92.0±4.0
cube-double-play-singletask-task1-v0	35±9	**76±8**	23.6±7.5	29.6±5.6	25.6±10.8
cube-double-play-singletask-task4-v0	1±1	**12±10**	1.2±1.0	1.2±1.0	1.2±1.0
puzzle-4x4-play-singletask-task2-v0	4±4	**12±9**	2.8±5.6	0.0±0.0	8.4±4.3
puzzle-4x4-play-singletask-task3-v0	50±14	**84±16**	5.6±10.2	0.0±0.0	37.2±7.8

Table 10: Computational cost and GPU memory usage of Ours and AK. Additional overhead is measured relative to Ours.

Metric	Ours	AK	Additional overhead
Computational cost (per update)	2.498 GFLOPs	4.978 GFLOPs	**+2.480 GFLOPs (+99.3%)**
GPU memory	1.141 GiB	1.266 GiB	**+0.125 GiB (+11.0%)**

Overall, Table 9 shows that Ours performs more consistently across OGBench environments. It matches or improves upon IFQL on every task and generally outperforms AK across the tested values of λ_a. Although AK benefits a few tasks, it produces substantial drops on others, particularly `cube-double` and `puzzle-4x4`. Thus, including the action-dependent term does not yield a reliable performance benefit.

Table 10 further shows that AK nearly doubles computation per update (+99.3%) and increases GPU memory usage by 11.0% relative to Ours. Together, these results support our empirical choice of the state-directional approximation.

C Derivation of the Higher-Order Supervision

This section provides the derivations of the higher-order policy as demonstrated in section 3. Specifically, Appendix C.1 derives the action differential for stochastic Gaussian policies, while Appendix C.2 derives the time derivative of the generated action for flow-based policies.

C.1 First-Order Stochastic Gaussian Policies Derivation in Eq. (3)

Consider the Gaussian policy with diagonal covariance

$$\pi(a|s) = \mathcal{N}\big(a; \mu(s), \mathrm{diag}(\sigma^2(s))\big)\,, \tag{13}$$

where $a, \mu(s), \sigma(s) \in \mathbb{R}^d$, and $\sigma^2(s)$ denotes the element-wise variance. We use $\sigma(s)$ to denote the element-wise standard deviation, i.e., the element-wise square root of the variance output.

We represent the stochastic component of the sampled action by

$$a = \mu(s) + \sigma(s) \odot X_t, \tag{14}$$

where $\odot$ denotes element-wise multiplication, and $X_t \in \mathbb{R}^d$ is assumed to follow the stationary vector OU process [68]

$$\mathrm{d}X_t = -\lambda X_t\,\mathrm{d}t + \sqrt{2\lambda}\,\mathrm{d}W_t, \qquad X_0 \sim \mathcal{N}(0, I). \tag{15}$$

Here W_t is a d-dimensional standard Wiener process. By stationarity, $X_t \sim \mathcal{N}(0, I)$ for all t. Under the diagonal covariance assumption, the stochastic perturbation is applied independently across action dimensions. Therefore,

$$a = \mu(s) + \sigma(s) \odot X_t, \qquad X_t \sim \mathcal{N}(0, I) \quad \Longrightarrow \quad a \sim \mathcal{N}\big(\mu(s), \mathrm{diag}(\sigma^2(s))\big)\,.$$

We condition on $(s, \dot{s})$ and treat $s = s(t)$ i.e., $\mathrm{d}s = \dot{s}\,\mathrm{d}t$ as a differentiable trajectory. For notational simplicity, we omit the explicit dependence of s on t. Define

$$F(t, X_t) = \mu(s) + \sigma(s) \odot X_t, \qquad a = F(t, X_t).$$

For each action dimension i, define

$$a_i = F_i(t, X_t) = \mu_i(s) + \sigma_i(s) X_{t,i}. \tag{16}$$

Since X_t is a vector Itô process, applying the multi-dimensional Itô formula [69] to $F_i(t, X_t)$ and substituting the OU dynamics of X_t gives,

$$\begin{aligned}
\mathrm{d}a_i = &\left[\frac{\partial F_i}{\partial t} + \nabla_X F_i(t, X_t)^\top (-\lambda X_t) + \frac{1}{2}\,\mathrm{Tr}\left((\sqrt{2\lambda} I)^\top H_X F_i(t, X_t)(\sqrt{2\lambda} I)\right)\right]\mathrm{d}t \\
&+ \nabla_X F_i(t, X_t)^\top \sqrt{2\lambda} I\,\mathrm{d}W_t \\
= &\left[\frac{\partial F_i}{\partial t} - \lambda \nabla_X F_i(t, X_t)^\top X_t + \lambda\,\mathrm{Tr}\left(H_X F_i(t, X_t)\right)\right]\mathrm{d}t \\
&+ \sqrt{2\lambda}\,\nabla_X F_i(t, X_t)^\top \mathrm{d}W_t.
\end{aligned} \tag{17}$$

From Eq. (16), F_i is linear with respect to X_t.

$$H_X F_i(t, X_t) = 0. \tag{18}$$

Moreover,

$$\frac{\partial F_i}{\partial t} = \dot{\mu}_i(s) + \dot{\sigma}_i(s) X_{t,i}, \qquad \nabla_X F_i(t, X_t) = \sigma_i(s) e_i, \tag{19}$$

where e_i is the i-th standard basis vector. Substituting these identities into the previous expression gives

$$\mathrm{d}a_i = \left[\dot{\mu}_i(s) + \left(\dot{\sigma}_i(s) - \lambda \sigma_i(s)\right) X_{t,i}\right]\mathrm{d}t + \sqrt{2\lambda}\,\sigma_i(s)\mathrm{d}W_{t,i}. \tag{20}$$

Stacking all action dimensions yields

$$\mathrm{d}a = \left[\dot{\mu}(s) + \left(\dot{\sigma}(s) - \lambda \sigma(s)\right) \odot X_t\right]\mathrm{d}t + \sqrt{2\lambda}\,\sigma(s) \odot \mathrm{d}W_t. \tag{21}$$

Using Eq. (14), we obtain

$$X_t = (a - \mu(s)) \oslash \sigma(s), \tag{22}$$

where $\oslash$ denotes element-wise division. Hence

$$\mathrm{d}a = \left[\dot{\mu}(s) + \left(\dot{\sigma}(s) - \lambda \sigma(s)\right) \odot \left((a - \mu(s)) \oslash \sigma(s)\right)\right]\mathrm{d}t + \sqrt{2\lambda}\,\sigma(s) \odot \mathrm{d}W_t. \tag{23}$$

Since the infinitesimal Wiener increment satisfies $\mathrm{d}W_t \sim \mathcal{N}(0, I\mathrm{d}t)$, Eq. (23) implies that the infinitesimal action increment conditioned on $(s, a, \dot{s})$ follows

$$\mathrm{d}a \sim \mathcal{N}\left(\left[\dot{\mu}(s) + \left(\dot{\sigma}(s) - \lambda \sigma(s)\right) \odot \left((a - \mu(s)) \oslash \sigma(s)\right)\right]\mathrm{d}t,\ 2\lambda\,\mathrm{diag}(\sigma^2(s))\mathrm{d}t\right). \tag{24}$$

As mentioned in Section 3.1, in our discrete-time implementation, we use the finite-difference approximation $\dot{a} \approx (a' - a)/\Delta t$ with $\Delta t = 1$. Under this unit-step finite-difference approximation, the induced first-order policy distribution over the action derivative is

$$\dot{\pi}(\dot{a}|s, a, \dot{s}) = \mathcal{N}\left(\dot{a};\ \dot{\mu}(s) + \left(\dot{\sigma}(s) - \lambda \sigma(s)\right) \odot \left((a - \mu(s)) \oslash \sigma(s)\right),\ 2\lambda\,\mathrm{diag}(\sigma^2(s))\right). \tag{25}$$

C.2 First-Order Flow Policies Derivation in Eq. (6)

Given the state s and initial noise sample a^0, the generated action is obtained by solving

$$\frac{da^k}{dk} = v(k, s, a^k), \qquad a^{k=0} = a^0, \tag{26}$$

and taking $a^1 = \pi(s)$,

$$a^1 = a^0 + \int_0^1 v(k, s, a^k)\,dk. \tag{27}$$

Assuming sufficient smoothness, differentiating a^1 with respect to physical time t and applying the Leibniz integral rule gives

$$\begin{aligned}\dot{a}^1 &= \dot{a}^0 + \frac{d}{dt}\int_0^1 v(k, s, a^k)\, dk \\ &= \dot{a}^0 + \int_0^1 \frac{d}{dt} v(k, s, a^k)\, dk. \end{aligned} \tag{28}$$

Since $v(k, s, a^k)$ depends on t through both s and a^k, the chain rule gives

$$\frac{d}{dt} v(k, s, a^k) = \frac{\partial v(k, s, a^k)}{\partial s}\dot{s} + \frac{\partial v(k, s, a^k)}{\partial a^k}\dot{a}^k. \tag{29}$$

Conditioning on the sampled noise a^0 when taking the physical-time derivative gives $\dot{a}^0 = 0$. Combining Eqs. (28) and (29) yields

$$\dot{a}^1 = \int_0^1 \dot{v}(k, s, a^k)\, dk = \int_0^1 \left(\frac{\partial v(k, s, a^k)}{\partial s}\dot{s} + \frac{\partial v(k, s, a^k)}{\partial a^k}\dot{a}^k \right) dk. \tag{30}$$

We consider the linear interpolation path used in flow matching,

$$a^k = (1 - k)a^0 + ka^1, \qquad k \in [0, 1]. \tag{31}$$

Taking its physical-time derivative while holding a^0 fixed gives

$$\dot{a}^k = (1 - k)\dot{a}^0 + k\dot{a}^1 = k\dot{a}^1. \tag{32}$$

Equation (30) therefore takes the following form along this path,

$$\dot{a}^1 = \int_0^1 \left(\frac{\partial v(k, s, a^k)}{\partial s}\dot{s} + k\frac{\partial v(k, s, a^k)}{\partial a^k}\dot{a}^1 \right) dk. \tag{33}$$

As in the zeroth-order action generation process in Eq. (27), the first-order action is obtained by integrating a vector field over flow time k. For the linear path, the sample-wise flow-matching target is $\frac{da^k}{dk} = a^1 - a^0$. Taking its physical-time derivative while holding a^0 fixed gives $\frac{d}{dt}(a^1 - a^0) = \dot{a}^1$. Thus, $\dot{a}^1 = a' - a^1$ provides a pointwise target for $\dot{v}(k, s, a^k)$. The shared integral structure motivates supervising this first-order field in a flow-matching manner.

Based on the computational cost and performance results in Appendix B.2, we approximate $\dot{v}(k, s, a^k)$ by $\frac{\partial v(k,s,a^k)}{\partial s}\dot{s}$. Accordingly, Eq. (7) supervises this approximation at sampled values of k; the integral describes action generation rather than the training loss.

D Theoretical Analysis

We first introduce the formal notation, assumptions, and auxiliary lemmas used in the analysis. Then we provide the proofs of Theorems 1 and 2.

D.1 Notation

We briefly recap the formal notation used in Section 3.3, with the time dependence made explicit for the analysis below. Let π and π^* denote the learned and expert policies, and let $s(t)$ and $s^*(t)$ denote their corresponding trajectories. The closed-loop dynamics are $\dot{s}(t) = f(s(t), \pi(s(t)))$ and $\dot{s}^*(t) = f(s^*(t), \pi^*(s^*(t)))$. The policy error and state deviation are denoted by $e(s) = \pi(s) - \pi^*(s)$ and $\delta(t) = s(t) - s^*(t)$, respectively. For each time t, define $T(t) = \{s^*(t) + \tau(s(t) - s^*(t)) : \tau \in [0, 1]\}$.

D.2 Assumptions

Assumption 1 (Domain and regularity)**.** *There exists a convex compact set $\Omega \subset \mathbb{R}^d$ such that $s(t), s^*(t) \in \Omega$ for all t under consideration. The policies π and π^* are continuously differentiable on Ω, and the policy error e admits the Jacobian $\nabla_s e$ on Ω.*

Assumption 2 (Lipschitz dynamics)**.** *For any $s_1, s_2 \in \Omega$ and any $a \in \mathbb{R}^m$, the dynamics f is Lipschitz continuous in the state variable with constant L_s, i.e.,*

$$\|f(s_1, a) - f(s_2, a)\| \leq L_s \|s_1 - s_2\|.$$

For any $s \in \Omega$ and any $a_1, a_2 \in \mathbb{R}^m$, the dynamics f is Lipschitz continuous in the action variable with constant L_a, i.e.,

$$\|f(s, a_1) - f(s, a_2)\| \leq L_a \|a_1 - a_2\|.$$

Assumption 3 (Lipschitz expert policy)**.** *The expert policy π^* is Lipschitz continuous with constant L_{π^*} on Ω, i.e., for any $s_1, s_2 \in \Omega$,*

$$\|\pi^*(s_1) - \pi^*(s_2)\| \leq L_{\pi^*} \|s_1 - s_2\|.$$

Assumption 4 (Bounded $\nabla_s e$)**.** *For all t under consideration, the Jacobian of the policy error is bounded on $T(t)$ by L_e. That is, for any $z \in T(t)$,*

$$\|\nabla_s e(z)\| \leq L_e.$$

D.3 Auxiliary Lemmas

Lemma 1. *Under Assumption 1, for any $s_1, s_2 \in \Omega$, letting $\Delta = s_1 - s_2$ and $\gamma(\tau) = s_2 + \tau\Delta$ for $\tau \in [0, 1]$, we have*

$$e(s_1) - e(s_2) = \int_0^1 \nabla_s e(\gamma(\tau))\, \Delta \, d\tau.$$

Consequently, for any time t, along the segment $T(t)$, we have

$$\|e(s(t))\| \leq \|e(s^*(t))\| + \sup_{z \in T(t)} \|\nabla_s e(z)\| \, \|s(t) - s^*(t)\|.$$

Proof. Fix $s_1, s_2 \in \Omega$, and define $\Delta = s_1 - s_2$ and $\gamma(\tau) = s_2 + \tau\Delta$. By Assumption 1, $\gamma(\tau) \in \Omega$ for all $\tau \in [0, 1]$. Let $g(\tau) = e(\gamma(\tau))$. Since e is continuously differentiable by Assumption 1, the chain rule gives

$$g'(\tau) = \nabla_s e(\gamma(\tau))\, \Delta.$$

By the fundamental theorem of calculus,

$$e(s_1) - e(s_2) = g(1) - g(0) = \int_0^1 g'(\tau)\, d\tau = \int_0^1 \nabla_s e(\gamma(\tau))\, \Delta \, d\tau.$$

Now take $s_1 = s(t)$ and $s_2 = s^*(t)$. By the definitions of $\delta(t)$ and $T(t)$, we have $\Delta = \delta(t)$ and $\gamma(\tau) \in T(t)$. Therefore,

$$e(s(t)) - e(s^*(t)) = \int_0^1 \nabla_s e(\gamma(\tau))\, \delta(t)\, d\tau.$$

Taking norms yields

$$\begin{aligned}
\|e(s(t))\| &\leq \|e(s^*(t))\| + \int_0^1 \|\nabla_s e(\gamma(\tau))\, \delta(t)\| \, d\tau \\
&\leq \|e(s^*(t))\| + \int_0^1 \|\nabla_s e(\gamma(\tau))\| \, \|\delta(t)\| \, d\tau \\
&\leq \|e(s^*(t))\| + \sup_{z \in T(t)} \|\nabla_s e(z)\| \, \|\delta(t)\|.
\end{aligned}$$

Using $\delta(t) = s(t) - s^*(t)$, we obtain

$$\|e(s(t))\| \leq \|e(s^*(t))\| + \sup_{z \in T(t)} \|\nabla_s e(z)\| \, \|s(t) - s^*(t)\|.$$

This completes the proof. □

Lemma 2 (Gronwall's lemma)**.** *[78] Let $T \in (0, \infty)$. Assume u is absolutely continuous on $[0, T)$ and satisfies the differential inequality*

$$\dot{u}(t) \leq a(t)\, u(t) + b(t), \qquad \text{for a.e. } t \in (0, T),$$

for some $a, b \in L^1(0, T)$. Then u satisfies the pointwise estimate

$$u(t) \leq e^{A(t)} u(0) + \int_0^t b(s)\, e^{A(t)-A(s)}\, ds, \qquad \forall\, t \in [0, T),$$

where we define

$$A(t) = \int_0^t a(s)\, ds.$$

This is the differential form of Gronwall's inequality, see Proposition 2.2 of [78] for a proof.

D.4 Proof of Theorem 1

Theorem 1. *Under Assumption 1, the expert-trajectory policy error $e(s^*(t))$ satisfies:*

$$\|e(s^*(t))\| \leq \|e(s^*(0))\| + t \max_{0 \leq \tau \leq t} \left\| \nabla_s \pi(s^*(\tau)) \dot{s}^*(\tau) - \dot{\pi}^*(s^*(\tau)) \right\|.$$

Proof. Recall the definition of the expert-trajectory policy error:

$$e(s^*(t)) = \pi(s^*(t)) - \pi^*(s^*(t)).$$

Define the expert action derivative along the expert trajectory as

$$\dot{\pi}^*(s^*(t)) = \frac{d}{dt} \pi^*(s^*(t)) = \nabla_s \pi^*(s^*(t))\, \dot{s}^*(t).$$

By Assumption 1 and the chain rule, the time derivative of the expert-trajectory policy error is

$$\frac{d}{dt} e(s^*(t)) = \nabla_s \pi(s^*(t))\, \dot{s}^*(t) - \dot{\pi}^*(s^*(t)).$$

Integrating over time yields:

$$e(s^*(t)) = e(s^*(0)) + \int_0^t \left(\nabla_s \pi(s^*(\tau))\, \dot{s}^*(\tau) - \dot{\pi}^*(s^*(\tau)) \right) d\tau.$$

Taking norms:

$$\|e(s^*(t))\| \leq \|e(s^*(0))\| + \int_0^t \left\| \nabla_s \pi(s^*(\tau))\, \dot{s}^*(\tau) - \dot{\pi}^*(s^*(\tau)) \right\| d\tau.$$

Therefore,

$$\|e(s^*(t))\| \leq \|e(s^*(0))\| + t \max_{0 \leq \tau \leq t} \left\| \nabla_s \pi(s^*(\tau))\, \dot{s}^*(\tau) - \dot{\pi}^*(s^*(\tau)) \right\|.$$

This completes the proof. □

D.5 Proof of Theorem 2

Theorem 2. *Under Assumptions 1–4, for $\delta(0) = 0$, the state deviation $\delta(t) = s(t) - s^*(t)$ satisfies*

$$\|\delta(t)\| \leq \frac{L_a}{\beta} \left(e^{\beta t} - 1 \right) \max_{0 \leq \tau \leq t} \|e(s^*(\tau))\|, \quad \textit{where} \quad \beta = L_s + L_a L_{\pi^*} + L_a L_e.$$

Proof. From Lemma 1 and Assumption 4, we have:

$$\|e(s(t))\| \leq \|e(s^*(t))\| + L_e \|s(t) - s^*(t)\|. \tag{34}$$

The time derivative of the state deviation is:

$$\dot{\delta}(t) = f(s(t), \pi(s(t))) - f(s^*(t), \pi^*(s^*(t))).$$

By Assumptions 2 and 3, we have

$$\begin{aligned}
\|\dot{\delta}(t)\| &= \|f(s(t), \pi(s(t))) - f(s^*(t), \pi^*(s^*(t)))\| \\
&\leq \|f(s(t), \pi(s(t))) - f(s(t), \pi^*(s(t)))\| + \|f(s(t), \pi^*(s(t))) - f(s^*(t), \pi^*(s^*(t)))\| \\
&\leq L_a \|\pi(s(t)) - \pi^*(s(t))\| + \|f(s(t), \pi^*(s(t))) - f(s^*(t), \pi^*(s(t)))\| \\
&\quad + \|f(s^*(t), \pi^*(s(t))) - f(s^*(t), \pi^*(s^*(t)))\| \\
&\leq L_a \|e(s(t))\| + L_s \|s(t) - s^*(t)\| + L_a \|\pi^*(s(t)) - \pi^*(s^*(t))\| \\
&\leq L_a \|e(s(t))\| + L_s \|\delta(t)\| + L_a L_{\pi^*} \|\delta(t)\| \\
&= L_a \|e(s(t))\| + \big(L_s + L_a L_{\pi^*}\big) \|\delta(t)\|.
\end{aligned}$$

Using the bound in Eq. (34), we get:

$$\|\dot{\delta}(t)\| \leq L_a \|e(s^*(t))\| + \big(L_s + L_a L_{\pi^*} + L_a L_e\big) \|\delta(t)\|.$$

Let $\beta = L_s + L_a L_{\pi^*} + L_a L_e$. Then:

$$\|\dot{\delta}(t)\| \leq \beta \|\delta(t)\| + L_a \|e(s^*(t))\|.$$

Let $u(t) = \|\delta(t)\|$. Under Assumptions 1 and 2, the trajectories $s(t)$ and $s^*(t)$ are absolutely continuous. Hence, $\delta(t) = s(t) - s^*(t)$ is absolutely continuous. Since the norm is Lipschitz continuous, $u(t)$ is also absolutely continuous. Moreover, for almost all $t \in (0, T)$, we have

$$\dot{u}(t) \leq \|\dot{\delta}(t)\|.$$

Therefore,

$$\dot{u}(t) \leq \beta u(t) + L_a \|e(s^*(t))\|, \qquad \text{for a.e. } t \in (0, T). \tag{35}$$

For Eq. (35), applying Lemma 2 with $a(t) = \beta$ and $b(t) = L_a \|e(s^*(t))\|$ yields

$$\|\delta(t)\| \leq e^{\beta t} \|\delta(0)\| + L_a \int_0^t e^{\beta(t-\tau)} \|e(s^*(\tau))\| \, d\tau.$$

This can be further bounded as:

$$\|\delta(t)\| \leq e^{\beta t} \|\delta(0)\| + \frac{L_a}{\beta} \big(e^{\beta t} - 1\big) \max_{0 \leq \tau \leq t} \|e(s^*(\tau))\|.$$

Assuming $\delta(0) = 0$, we get:

$$\|\delta(t)\| \leq \frac{L_a}{\beta} \big(e^{\beta t} - 1\big) \max_{0 \leq \tau \leq t} \|e(s^*(\tau))\|.$$

This completes the proof. □

E Experimental Details

We evaluate our method on two widely used offline RL benchmarks: OGBench [28] and D4RL [29]. OGBench provides high-dimensional, long-horizon, and structurally diverse tasks that go well beyond standard locomotion, while D4RL offers a well-established testbed of MuJoCo locomotion tasks with datasets of varying behavioral quality. Together, they cover a broad spectrum of state and action dimensionalities, task structures, and dataset qualities, providing a comprehensive evaluation of our scheme. All methods are trained on the offline datasets only, with no online interaction.

E.1 Benchmarks and Tasks

OGBench. Since OGBench was originally designed for goal-conditioned RL, we adopt its reward-based single-task variants (denoted by the `-singletask` suffix) to benchmark standard reward-maximizing offline RL methods. In total, we evaluate on 50 state-based tasks drawn from 10 environments (5 tasks per environment), covering long-horizon locomotion (`antmaze`, `humanoidmaze`), object interaction (`antsoccer`), and robotic manipulation (`cube`, `scene`, `puzzle`), where `scene` additionally requires long-horizon multi-object manipulation and `puzzle` involves more combinatorial reasoning.

These tasks pose substantial challenges in high-dimensional continuous control, with state dimensions ranging from 28 to 83 and action dimensions from 5 to 21. For instance, `humanoidmaze` has a 69-dimensional state space and a 21-dimensional action space, while `cube-double`, `scene`, and `puzzle-4x4` have 37-, 40-, and 83-dimensional state spaces, respectively. We use the standard reward-based datasets provided by OGBench, which are highly suboptimal as they aggregate trajectories from policies performing diverse random tasks (e.g., reaching different goals or manipulating different objects). This makes it particularly difficult for offline RL methods to extract task-relevant behaviors from such heterogeneous data.

D4RL. We further evaluate our method on the MuJoCo locomotion tasks from D4RL, including `halfcheetah`, `hopper`, and `walker2d`, with the standard `medium`, `medium-replay`, `medium-expert`, and `expert` dataset types collected from behavior policies of varying quality. The full-size datasets contain about 1M transitions for the `medium` and `expert` settings, 2M transitions for `medium-expert`, and around 200k–400k transitions for `medium-replay`. Beyond the standard full-data setting, we additionally consider a small-dataset setting to study the data efficiency of our scheme. Specifically, for each task we first split the original dataset into individual trajectories, and then sample a small set of complete trajectories until the accumulated transitions reach 10k, rather than randomly sampling individual transitions. This trajectory-level sampling preserves the temporal continuity and dynamical consistency of the original demonstrations, which is essential for our first-order action supervision since it relies on consecutive transitions (s, s', a, a') to compute the local state and action derivatives $(\dot{s}, \dot{a})$. To ensure a fair comparison, all methods use the same subsampled dataset under the same random seed.

E.2 Methods and Hyperparameters

We instantiate our first-order action supervision on four representative offline RL algorithms covering deterministic, stochastic, and flow-based policies: TD3+BC [34], ReBRAC [73], IQL [38], and IFQL [54]. For a fair comparison, all algorithms are implemented on top of the FQL [54] codebase and share the same network architectures and default hyperparameters unless otherwise specified. For OGBench tasks, we follow the hyperparameter settings reported in FQL [54] for ReBRAC, IQL, and IFQL. After integrating our scheme, we keep all baseline-specific hyperparameters unchanged, and only specify the two additional weights λ_Q (the value-side regularization weight) and $\lambda_{\dot{a}}$ (the policy-side first-order supervision weight) introduced by our scheme. The basic shared hyperparameters are summarized in Table 11.

TD3+BC. We re-implement TD3+BC on top of the FQL [54] codebase. On OGBench, we sweep the Q-value scaling coefficient α over $\{1, 3, 10, 30, 100, 300\}$ and select the best value per environment. Intuitively, a smaller α is preferable in complex environments where the Q-function is hard to learn reliably, while a larger α works better on simpler ones where the learned Q-function offers stronger policy improvement signals. We also observe Q-learning instability on several challenging OGBench tasks, and apply layer normalization to the Q-network to mitigate this issue. On D4RL, we follow the hyperparameter settings of the original TD3+BC paper [34], and use the default values for all other hyperparameters (e.g., noise standard deviation, noise clipping threshold). For our scheme, we set $\lambda_Q = 0.1$ on all OGBench tasks, $\lambda_{\dot{a}} = 1.0$ on `antmaze` and $\lambda_{\dot{a}} = 0.1$ on all other OGBench tasks; on D4RL, we set $\lambda_Q = 0.1$ and $\lambda_{\dot{a}} = 1.0$.

ReBRAC. We follow the hyperparameter settings reported in the original ReBRAC paper [73] on D4RL, including the actor and critic BC coefficients (α_a, α_c). For our scheme, we set $\lambda_Q = 0.1$ on all OGBench tasks, $\lambda_{\dot{a}} = 1.0\,\alpha_a$ on `antmaze`, `humanoidmaze`, and `antsoccer`, and $\lambda_{\dot{a}} = 0.1\,\alpha_a$ on `cube`, `scene`, and `puzzle`; on D4RL, we set $\lambda_Q = 0.1$ and $\lambda_{\dot{a}} = 1.0\,\alpha_a$.

IQL. We follow the hyperparameter settings reported in the original IQL paper [38] on D4RL, including the expectile value and the AWR inverse temperature. For our scheme on OGBench, we set $\lambda_Q = 0.5$ on `scene` and $\lambda_Q = 0.1$ on all other tasks, and $\lambda_{\dot{a}} = 1.0$ on `antmaze` and `scene` and $\lambda_{\dot{a}} = 0.1$ on all other tasks; on D4RL, we set $\lambda_Q = 0.1$ and $\lambda_{\dot{a}} = 1.0$.

IFQL. For IFQL, we use $N = 32$ action samples on D4RL tasks, and follow all other settings from FQL [54]. For our scheme, we set $\lambda_Q = 1.0$ and $\lambda_{\dot{a}} = 1.0$ on all OGBench and D4RL tasks.

Table 11: Shared hyperparameters used across all four offline RL algorithms.

Hyperparameter	**Value**
Learning rate	3×10^{-4}
Optimizer	Adam [79]
Gradient steps	1,000,000
Minibatch size	256
MLP dimensions	$[512, 512, 512, 512]$ (default); $[256, 256, 256, 256]$ (IQL)
Nonlinearity	GELU [80]
Target network smoothing coefficient	0.005
Discount factor γ	0.99 (default); 0.995 (`antmaze-giant`, `humanoidmaze`, `antsoccer`)
Temporal correlation coefficient λ	1.0

F Broader Impact

Modern learning-based decision-making methods, such as imitation learning (IL) and reinforcement learning (RL), often suffer from control instability and poor robustness when deployed in real-world systems, due to their sole reliance on zeroth-order action supervision. To address this, we propose a lightweight and plug-and-play learning scheme that simultaneously supervises zeroth- and first-order actions, transforming any standard policy model into a more powerful policy class without structural modifications. Our method substantially improves action smoothness, control stability, and out-of-distribution generalization, especially in challenging low-data regimes, making it well-suited for safety-critical domains such as robotics, drone control, and autonomous driving, where high-frequency action fluctuations can cause mechanical wear, energy waste, and safety risks. By bridging classical control principles with modern policy learning, our work offers a promising pathway toward more reliable and deployable learning-based controllers. A potential limitation is that, like other offline methods, the learned policy may still inherit unsafe or suboptimal behaviors present in the historical dataset.

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: We introduce the contributions and scope in the abstract and introduction.

   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: We discussed the limitations of our work in Appendix A.

   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: We presented the utilized assumptions in Section 3 and in Appendix D.2, and provided complete proof for the theory in Appendix D.

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 explained our problem settings in Section 5 and implementation details in Appendix E.2.

   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: For anonymity reasons, we have not made our code public. Upon acceptance, we will release our code on GitHub.

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: All the training and model details are specified in Appendix E.2 and the experiment setup details are presented in Appendix E.

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: We have provided the necessary information in the table caption to clarify how the results and standard deviations were calculated.

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: In Table 7, we report the computational training time and the hardware resources we used to obtain the experimental results.

   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: We read and followed 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: We discuss societal impacts of the work in Appendix F.

    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: N/A

    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: [N/A]

    Justification: [N/A]

    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: [N/A]

    Justification: N/A

    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: N/A

    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: N/A

    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: N/A

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.