Moment-Constrained Latent Steering for Flow Policy

Wenxin Zhao[1,†], Letian Tao[1,†], Zhilong Zheng[1], Guojian Zhan[1], Yujie Yang[1]
Likun Wang[1], Yinuo Wang[1], Feihong Zhang[1], Tianze Zhu[1], Jingliang Duan[2]
Yang Guan[1,*], Shengbo Eben Li[1,*]

[1] College of AI & School of Vehicle and Mobility, Tsinghua University
[2] School of Mechanical Engineering, University of Science & Technology Beijing

[†] Equal contribution. [*] Corresponding authors.

Abstract

Fine-tuning continuous-time generative models, such as flow models, via reinforcement learning (RL) is a promising approach for complex continuous control, but directly updating the network weights often leads to catastrophic forgetting of the pre-trained prior. Latent steering method mitigates this by freezing the generative backbone and optimizing a latent policy to generate the initial noise. However, existing methods typically enforce hard support boundaries to prevent out-of-distribution (OOD) shifts in the latent space, inducing a fundamental trade-off between prior-density compatibility and policy expressiveness. To resolve this dilemma, we propose Moment Constraint Parameterization, which regulates the latent distribution through a statistical budget. By decoupling this budget into inter-statistic constraints and intra-statistic flexibility, our method acts as a probabilistic safeguard against OOD shifts while providing the degrees of freedom to retain policy expressiveness. Furthermore, while moment constraints define a flexible search space, effectively optimizing the latent policy within it requires accurate gradient signals. To address the optimization lag and bias caused by proxy latent critics in existing methods, we introduce Adjoint Q-Gradient Propagation. By exploiting the white-box structure of flow models, this technique backpropagates exact analytical gradients from the action-space critic directly to the initial noise. Integrating these two mechanisms, we present Moment Constraint Flow Steering (MCFS). Empirical results on D4RL and OGBench show that MCFS consistently delivers stronger online adaptation performance than prior latent steering and flow-policy baselines, with particularly pronounced gains on challenging long-horizon compositional manipulation tasks.

1 Introduction

The integration of high-capacity continuous-time generative models, such as diffusion models and flow matching models, has significantly advanced continuous control by providing multimodal action policies [1, 2, 3, 4, 5]. Because pre-training captures diverse offline behaviors, fine-tuning these models is a promising paradigm for adapting them to complex downstream tasks [6]. These models generate actions by mapping standard Gaussian noise through a learned vector field. However, directly updating the network weights via reinforcement learning (RL) distorts their previously calibrated vector field, leading to catastrophic forgetting of the original generative prior [7]. To bypass this issue, latent space steering has become an effective alternative [8, 9]. By freezing the generative backbone and optimizing a lightweight latent policy to modulate the initial noise, this approach treats

*Corresponding authors: yguan@tsinghua.edu.cn; lishbo@tsinghua.edu.cn.

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

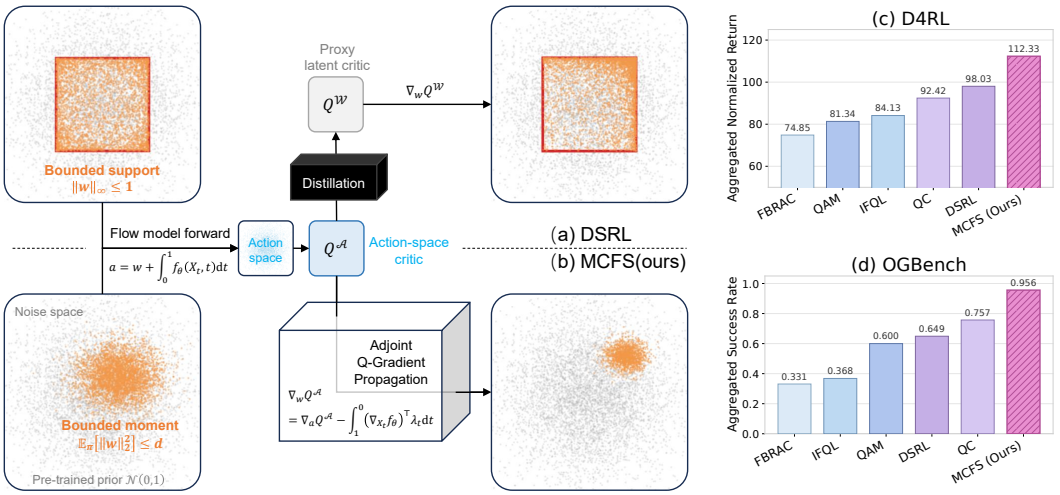


Figure 1: **Comparison of latent space steering frameworks.** (a) Prior work DSRL relies on rigid bounded support ($||w||_\infty \leq 1$) and a distilled proxy latent critic (Q^w), which restricts policy expressiveness and introduces optimization bias. (b) Our proposed MCFS applies a bounded moment constraint ($\mathbb{E}_\pi[||w||_2^2] \leq d$) for soft probabilistic containment, and utilizes Adjoint Q-Gradient Propagation to directly backpropagate exact gradients from the action-space critic ($Q^{\mathcal{A}}$). (c-d) Aggregate performance comparisons on D4RL and OGBench. Results demonstrate that MCFS outperforms all evaluated baselines across both domains.

the pre-trained model as a fixed dynamical system. The primary objective is to achieve downstream task alignment while fundamentally retaining the structural integrity of the backbone.

However, realizing the full potential of latent steering requires prior-density compatibility. This compatibility entails statistical control of extreme excursions into low-prior-density regions while retaining flexibility for policy optimization. Existing methods [8, 9] typically rely on hard support constraints in an attempt to keep the generated noise away from extreme low-prior-density regions. This rigid geometric intervention inevitably induces a trade-off between **prior-density compatibility** and **policy expressiveness**. If the support boundaries are too large, the latent variables fall into extreme low-prior-density regions of the original latent prior, triggering severe out-of-distribution (OOD) shifts in the generative backbone [10]. Conversely, if the bound is too small, it leads to restricted noise support. This creates an expressiveness bottleneck, limiting the policy's capacity to explore and model diverse behaviors. This dilemma raises a fundamental question: *How can we prevent OOD shifts from the pre-trained prior without sacrificing the expressiveness of the latent policy?*

To address this dilemma, we propose Moment Constraint Parameterization for the latent policy. Rather than imposing hard support boundaries that truncate the noise space, our method operates as a probabilistic budget bounding the second moment of the latent policy. Specifically, this budget is allocated across two levels regarding the policy's primary statistics (i.e., mean and variance): **inter-statistic** and **intra-statistic**. To prevent OOD shifts from the pre-trained prior, the inter-statistic level coordinates the distribution between the two statistics, balancing the overall mean shift against the variance scale. Meanwhile, to maintain policy expressiveness, the intra-statistic level flexibly allocates the specific budget of the mean or variance across different action dimensions. This dual-level allocation addresses the aforementioned dilemma: the inter-statistic constraint facilitates prior-density compatibility, while the intra-statistic flexibility retains policy expressiveness.

However, defining a flexible search space via moment constraints addresses only the representation problem; effectively navigating this space requires accurate gradient signals. A critical optimization bottleneck in latent steering is the lack of an efficient gradient path from the action-space critic $Q^{\mathcal{A}}$ through the frozen generative backbone. Existing methods treat the generative backbone as a black box, distilling a proxy latent critic $Q^{\mathcal{W}}$ that inevitably introduces approximation bias and an optimization lag, limiting sample efficiency [8]. To bypass this, we leverage the differentiable, white-box ordinary differential equation (ODE) structure of flow models. We introduce Adjoint

Q-Gradient Propagation, which utilizes the adjoint state method [11, 12] to backpropagate analytical gradients from the action-space critic $Q^{\mathcal{A}}$ directly to the initial noise, eliminating proxy bottlenecks. By replacing the biased proxy with the exact action-space critic signal, our method improves the optimization efficiency and performance of latent steering.

Our contributions are as follows:

1. We propose Moment Constraint Parameterization to address the trade-off between prior-density compatibility and policy expressiveness in latent steering. By decoupling a second-moment budget into inter-statistic constraints and intra-statistic flexibility, this method bounds the tail probability of OOD shifts while providing the degrees of freedom to freely shape the latent distribution.
2. We introduce Adjoint Q-Gradient Propagation, a differentiable latent policy optimization scheme that exploits the white-box ODE structure of flow models. By backpropagating analytical gradients from the action-space critic directly to the initial noise, it eliminates the reliance on biased proxy latent critics, yielding an exact and memory-efficient optimization pipeline without unrolling the computation graph.
3. Integrating the aforementioned mechanisms, we present Moment Constraint Flow Steering (MCFS). Empirical results on D4RL and OGBench demonstrate that MCFS outperforms both prior hard-support latent steering methods and flow policy fine-tuning baselines. Specifically, it achieves superior average returns and faster early-stage improvement, validating the efficiency of our proposed framework.

2 Preliminaries

Reinforcement Learning. We model the continuous control problem as a standard Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$, with continuous states $\mathcal{S}$, d-dimensional actions $\mathcal{A} \subset \mathbb{R}^d$, transition dynamics $\mathcal{P}(s'|s,a)$, reward $r(s,a)$ and discount factor $\gamma \in (0,1)$. The RL objective is to maximize the expected cumulative return under policy π: $\max J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. To evaluate the expected return of taking an action a in state s, RL algorithms typically learn an action-space critic, denoted as $Q^{\mathcal{A}}(s,a)$.

Flow-Based Policies. Flow matching models formulate the generative process as a continuous-time transformation from a simple tractable prior distribution to a complex empirical data distribution via an ODE [13]. A pre-trained flow-based policy generates an action $a \in \mathcal{A}$ conditioned on a state $s \in \mathcal{S}$ by simulating a vector field. Let the base noise variable be $w \in \mathbb{R}^d$, which follows a standard Gaussian prior distribution $p(w) = \mathcal{N}(0, I_d)$. A neural network $f_\theta(X_t, t; s)$ parameterized by θ is trained to approximate the target vector field. During the action generation (inference) phase, the model solves the following initial value problem (IVP):

$$\frac{dX_t}{dt} = f_\theta(X_t, t; s), \quad t \in [0,1], \tag{1}$$

subject to the initial condition $X_0 = w \sim p(w)$. The terminal state of this ODE trajectory constitutes the generated action, i.e., $a = X_1$. Due to the high capacity of the neural vector field, flow-based policies have demonstrated state-of-the-art expressiveness in modeling complex action distributions.

Latent Policy Steering. Directly fine-tuning the vector field θ via RL often disrupts the calibrated ODE trajectories, causing catastrophic forgetting of the pre-trained prior. Latent steering circumvents this by freezing f_θ and optimizing a learnable, state-conditioned latent policy $\pi_\phi(w|s)$ to replace the fixed Gaussian prior $p(w)$. Under this paradigm, actions are generated by integrating the frozen ODE starting from $X_0 = w \sim \pi_\phi(\cdot|s)$:

$$a = X_1 = w + \int_0^1 f_\theta(X_t, t; s) \mathrm{d}t. \tag{2}$$

The optimization objective is to update ϕ to maximize the expected critic value: $\max_\phi \mathbb{E}_{w \sim \pi_\phi(\cdot|s)}[Q^{\mathcal{A}}(s,a)]$. Existing methods typically distill $Q^{\mathcal{A}}(s,a)$ into a proxy latent critic $Q^{\mathcal{W}}(s,w)$, which inevitably introduces approximation bias.



(a) Inter-statistic Constraint (b) Intra-statistic Flexibility

Figure 2: **Latent Policy Distribution induced by the Moment Constraint Parameterization.** The background gray points represent the unconstrained standard Gaussian prior, while the orange points show the steered latent policy bounded by the second-moment budget. **(a) Inter-statistic constraint:** As the policy mean μ shifts toward the boundary, the available variance contracts to facilitate prior-density compatibility and prevent OOD shift. **(b) Intra-statistic flexibility:** For a fixed mean shift ($\|\mu\|_2^2 = 1.10$), the residual variance budget can be flexibly distributed across different dimensions. This retains expressiveness by allowing the policy to freely shape its distribution within the budget.

3 Method

In this section, we propose MCFS. First, we introduce a Moment Constraint Parameterization to control distribution shifts via a statistical budget while preserving policy expressiveness. Second, we present Adjoint Q-Gradient Propagation, which leverages the white-box ODE structure to backpropagate exact critic gradients. Finally, we summarize the complete algorithm.

3.1 Moment Constraint Parameterization for Prior-Density-Compatible Steering

Since hard support boundaries in latent steering act as a rigid wall that truncates the noise space, they induce a trade-off between prior-density compatibility and policy expressiveness. In practice, enforcing such bounds by squashing Gaussian samples through a $\tanh$ mapping further bottlenecks expressiveness due to gradient vanishing at the limits. To decouple these objectives, we introduce the *Moment Constraint Parameterization*. Instead of enforcing rigid geometric limits, our method regulates the latent policy by anchoring its overall probability mass within the prior's second moment. Specifically, we operationalize this statistical budget across the policy's primary statistics (i.e., mean and variance) at two distinct levels: inter-statistic and intra-statistic. This formulation provides the degrees of freedom to sample high-magnitude latent variables.

Let the pre-trained latent prior be the standard Gaussian $w \sim p(w) = \mathcal{N}(0, I_d)$, where d denotes the action dimension. For any latent policy $\pi(w|s)$ conditioned on state s, we enforce the following second-moment constraint:

$$\mathbb{E}_{w\sim\pi(\cdot|s)}[\|w\|_2^2] \leq \mathbb{E}_{w\sim p(w)}[\|w\|_2^2] = d. \tag{3}$$

For a general latent policy $\pi(w|s)$ with mean μ and covariance Σ, the second-moment constraint can be exactly decomposed as $\mathbb{E}_\pi[\|w\|_2^2] = \|\mu\|_2^2 + \mathrm{Tr}(\Sigma) \leq d$. This decomposition governs the aforementioned dual-level allocation:

- **Inter-statistic Constraint:** The constraint dynamically coordinates the budget between the mean and the variance. As μ shifts toward the boundary (i.e., $\|\mu\|_2^2$ increases), the overall variance $\mathrm{Tr}(\Sigma)$ must inherently contract. This coupling facilitates prior-density compatibility by preventing severe OOD shifts (Fig. 2a).
- **Intra-statistic Flexibility:** The policy flexibly allocates the specific budget of the mean (via $\sum_{i=1}^d \mu_i^2$) or the variance (via $\sum_{i=1}^d \Sigma_{ii}$) across different latent dimensions. This retains the policy expressiveness by providing the degrees of freedom to freely shape its distribution within the constrained space (Fig. 2b).

Through this adaptive parameterization, our method mitigates the expressiveness bottlenecks of hard support bounds while remaining anchored within the prior's second moment.Next, we analyze how

this constraint promotes prior-density compatibility by (1) bounding worst-case tail probabilities and (2) preserving the expected prior likelihood of the latent policy.

(1) Bounding Worst-Case Tail Probabilities. By Markov's inequality, the moment constraint bounds the probability of sampled latents escaping into extrapolated noise regions.

Proposition 1 (Tail Probability Bound) *Consider an open hyper-ball $\mathcal{B}_\epsilon = \{w \in \mathbb{R}^d : \|w\|_2 < \epsilon\}$ with radius $\epsilon > \sqrt{d}$. Under the moment constraint $\mathbb{E}_{\pi(\cdot|s)}[\|w\|_2^2] \leq d$, the tail probability of the steered latent w escaping $\mathcal{B}_\epsilon$ is bounded by: $\mathbb{P}_{w\sim\pi(\cdot|s)}(\|w\|_2 \geq \epsilon) \leq \frac{d}{\epsilon^2}$.*

Proof 1 *We apply Markov's inequality to the non-negative random variable $\|w\|_2^2$. Since $\epsilon > 0$, we have $\mathbb{P}(\|w\|_2 \geq \epsilon) = \mathbb{P}(\|w\|_2^2 \geq \epsilon^2) \leq \mathbb{E}_{\pi(\cdot|s)}[\|w\|_2^2]/\epsilon^2$. Substituting the moment constraint $\mathbb{E}_{\pi(\cdot|s)}[\|w\|_2^2] \leq d$ yields the final bound.*

Proposition 1 establishes an $\mathcal{O}(1/\epsilon^2)$ polynomial bound on the tail probabilities. Unlike hard support boundaries that permanently block access to distal latent spaces, this controlled decay rate ensures the latent space remains topologically open for smooth gradient flow, while providing a probabilistic safety net against extreme noise inflation.

(2) Bounding Expected Prior Likelihood. While Proposition 1 restricts the worst-case probability of extreme outliers, Proposition 2 prevents catastrophic global OOD shift by ensuring the expected prior log-likelihood of the steered latents does not collapse.

Proposition 2 (Expected Prior Likelihood Bound) *For any arbitrary steered distribution $\pi(\cdot|s)$ satisfying the moment constraint $\mathbb{E}_\pi[\|w\|_2^2] \leq d$, its expected log-likelihood under the prior is lower-bounded by that of the prior itself:*

$$\mathbb{E}_{w\sim\pi(\cdot|s)}[\log p(w)] \geq \mathbb{E}_{w\sim p}[\log p(w)] = -\frac{d}{2}\big(\log(2\pi) + 1\big). \tag{4}$$

Proof 2 *For $p(w) = \mathcal{N}(0, I_d)$, the log-likelihood is $\log p(w) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\|w\|_2^2$. Taking the expectation over $\pi(\cdot|s)$ yields $\mathbb{E}_{\pi(\cdot|s)}[\log p(w)] = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\mathbb{E}_{\pi(\cdot|s)}[\|w\|_2^2]$. Substituting the constraint $\mathbb{E}_{\pi(\cdot|s)}[\|w\|_2^2] \leq d$ directly completes the proof.*

This lower bound guarantees that the latent policy $\pi(\cdot|s)$ is, in expectation, more concentrated within the high-probability regions of the prior than the prior itself. By statistically anchoring the probability mass, the moment constraint prevents catastrophic OOD shift. Therefore, this constraint acts as a global statistical anchor, ensuring the steered policy remains concentrated within the prior's high-probability manifold.

3.2 Adjoint Q-Gradient Propagation for Efficient Latent Steering

The moment constraint establishes a flexible search space with prior-density compatibility, but optimizing the latent policy within it requires high-fidelity gradient signals. A critical challenge in latent steering is the absence of a gradient flow from the action-space critic $Q^{\mathcal{A}}(s, a)$ back to the initial noise w. Existing methods typically treat the generative backbone as an intractable black box. Consequently, they have to introduce a proxy latent critic $Q^{\mathcal{W}}(s, w)$ as an intermediary and distill the action-space critic $Q^{\mathcal{A}}(s, a)$ into latent space. However, this distillation inevitably injects approximation bias and limits sample efficiency. In contrast, we exploit the fact that the flow-based policy is a differentiable ODE system. By leveraging the white-box structure, we bypass proxy approximations and derive the exact latent gradient directly from the action-space critic $Q^{\mathcal{A}}(s, a)$.

Consider the objective of maximizing the expected return $J(\phi) = \mathbb{E}_{w\sim\pi_\phi(\cdot|s)}[Q_\psi^{\mathcal{A}}(s, a)]$, where $Q_\psi^{\mathcal{A}}$ is the learned action-space critic network parameterized by ψ, and the action $a = X_1$ is the terminal state of a Neural ODE defined by the frozen pre-trained vector field f_θ:

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = f_\theta(X_t, t; s), \quad X_0 = w. \tag{5}$$

We employ the adjoint state method [11] to derive exact latent policy gradients through the flow model. We define the adjoint state as the gradient of the action-space critic with respect to the intermediate state $\lambda_t := \nabla_{X_t} Q_\psi^{\mathcal{A}}(s, X_1)$. The dynamics of λ_t are governed by the adjoint ODE:

$$\frac{\mathrm{d}\lambda_t}{\mathrm{d}t} = -\left(\nabla_{X_t} f_\theta(X_t, t; s)\right)^\top \lambda_t, \tag{6}$$

subject to the terminal boundary condition $\lambda_1 = \nabla_a Q_\psi^{\mathcal{A}}(s, a)$. By solving this adjoint ODE backward from $t = 1$ to $t = 0$, we explicitly recover the adjoint state at the initial time, $\lambda_0 = \nabla_w Q_\psi^{\mathcal{A}}(s, a)$, which provides the exact gradient of the return with respect to the initial latent noise.

To formalize the policy update under this latent gradient, we parameterize the latent policy as a Gaussian distribution $\pi_\phi(w|s) = \mathcal{N}(\mu_\phi(s), \mathrm{diag}(\sigma_\phi(s)^2))$. Using the reparameterization trick, we draw the latent noise $w_\phi(s, \epsilon) = \mu_\phi(s) + \sigma_\phi(s) \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I_d)$ and $\odot$ denotes the element-wise product. By the chain rule, the gradient of the objective $J(\phi)$ is given by:

$$\nabla_\phi J(\phi) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I_d)} \left[\nabla_w Q_\psi^{\mathcal{A}}(s, a)^\top \nabla_\phi w_\phi(s, \epsilon) \right] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I_d)} \left[\lambda_0^\top \nabla_\phi w_\phi(s, \epsilon) \right]. \tag{7}$$

To integrate this exact gradient into standard auto-differentiation pipelines while enabling explicit gradient manipulation, we construct a surrogate loss $\mathcal{L}(\phi)$ whose auto-differentiation matches Eq. (7):

$$\mathcal{L}(\phi) = -\mathbb{E}_{\epsilon \sim \mathcal{N}(0, I_d)} \left[\mathrm{sg}(\lambda_0)^\top w_\phi(s, \epsilon) \right], \tag{8}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. This surrogate loss explicitly separates the policy's parameter update from the backward pass through the ODE flow. By extracting the adjoint state as a detached signal, this formulation effectively avoids the prohibitive memory overhead of caching intermediate network activations during standard backpropagation.

3.3 Moment Constraint Flow Steering

By integrating the moment constraint parameterization and adjoint Q-gradient propagation, we propose MCFS, an online RL algorithm for fine-tuning generative flow policies.

Implementation of the Moment Constraint Parameterization. In practice, we implement the budget $\|\mu_\phi\|_2^2 + \|\sigma_\phi\|_2^2 \leq d$ by first ensuring a minimum variance floor. We introduce a per-dimension variance floor $\sigma_{\min}$, which defines a maximum mean magnitude $\mu_{\max} = \sqrt{d - d\sigma_{\min}^2}$. During the forward pass, we scale the predicted mean $\|\mu_\phi\|_2$ to $\mu_{\max}$ and subsequently scale the standard deviations σ_ϕ to satisfy the overall constraint $\|\sigma_\phi\|_2^2 \leq d - \|\mu_\phi\|_2^2$. This procedure ensures the statistical budget is met while preserving the relative variance distribution across dimensions.

Critic Optimization. We train the action-space critic $Q_\psi^{\mathcal{A}}$ using standard temporal difference (TD) learning. Following common practices in generative RL [14], the Bellman target is computed by sampling candidate actions via the frozen flow model and selecting the maximum Q-value.

Actor Optimization. To maximize expected returns while maintaining sufficient exploration, we augment the adjoint surrogate loss with an entropy term $\mathcal{H}(\pi_\phi)$. This regularization encourages the policy to utilize its residual variance budget for stochastic exploration, preventing premature collapse into deterministic solutions:

$$\mathcal{L}_{\text{Actor}}(\phi) = \mathbb{E}_{\mathcal{B}} \left[-\mathbb{E}_{\epsilon \sim \mathcal{N}(0, I_d)} \left[\mathrm{sg}(\lambda_0)^\top w_\phi(s, \epsilon) \right] - \beta \mathcal{H}\left(\pi_\phi(\cdot|s)\right) \right], \tag{9}$$

where $\lambda_0 = \nabla_w Q_\psi^{\mathcal{A}}(s, a)$ is the analytical latent gradient, β is the entropy temperature, and $\mathcal{B}$ is the replay data. The complete MCFS is summarized in Algorithm 1.

4 Experiments

4.1 Experimental Setup

Settings All experiments are conducted under an offline-to-online (O2O) adaptation setting. For a fair comparison, all methods are initialized from the same pretrained checkpoints and then adapted during the online phase. Detailed experimental settings are provided in Appendix A.2.

Algorithm 1 Moment Constraint Flow Steering (MCFS)

Input: Frozen pre-trained flow model f_θ, action dimension d, variance floor $\sigma_{\min}$.
Initialize: Critic $Q_\psi^{\mathcal{A}}$, Latent policy π_ϕ, Replay buffer $\mathcal{D}$.

// Latent Policy Forward with Moment Constraint Parameterization
function CONSTRAINEDLATENT(s)
  Predict unconstrained parameters $\mu, \sigma \leftarrow \mu_\phi(s), \sigma_\phi(s)$.
  $\mu \leftarrow \mu \cdot \min\left(1, \frac{\sqrt{d - d\sigma_{\min}^2}}{\|\mu\|_2}\right)$
  $\sigma \leftarrow \sigma \cdot \min\left(1, \sqrt{\frac{d - \|\mu\|_2^2}{\|\sigma\|_2^2}}\right)$
  Sample $w \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$
  return w, μ, σ
end function
for each environment step **do**
  // 1. Data Sample
  Obtain N latents $\{w^i\}_{i=1}^N \leftarrow$ CONSTRAINEDLATENT(s), integrate Eq.(5) to get $\{a^i\}_{i=1}^N$.
  Execute $a^* = \arg\max_{a^i} Q_\psi^{\mathcal{A}}(s, a^i)$, receive r, s', and store transition in $\mathcal{D}$.
  // 2. Critic Update
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$.
  Obtain next latents $\{w^i\}_{i=1}^N \leftarrow$ CONSTRAINEDLATENT(s'), generate $\{a^i\}$ via Eq.(5).
  Compute target $y = r + \gamma \max_i\{Q_{\bar{\psi}}^{\mathcal{A}}(s', a^i)\}$ and stop gradient.
  $\mathcal{L}_{\text{critic}}(\psi) = \mathbb{E}_{\mathcal{B}}\left[(Q_\psi^{\mathcal{A}}(s, a) - y)^2\right]$; Update $Q_\psi^{\mathcal{A}}$ by minimizing $\mathcal{L}_{\text{critic}}(\psi)$.
  // 3. Actor Update
  Obtain latents $w, \mu, \sigma \leftarrow$ CONSTRAINEDLATENT(s), integrate Eq.(5) to get a.
  Solve Adjoint ODE in Eq. (6) from $t = 1$ to 0 to obtain latent gradient $\lambda_0 = \nabla_w Q_\psi^{\mathcal{A}}(s, a)$.
  Compute $\mathcal{L}_{\text{Actor}}(\phi)$ in Eq. (9); Update π_ϕ by minimizing $\mathcal{L}_{\text{Actor}}(\phi)$.
end for

Benchmarks We evaluate our method on a comprehensive suite of high-dimensional, contact-rich robotic manipulation benchmarks from **OGBench** [15] and **D4RL Adroit** [16]. For OGBench, we consider the `cube-double-play`, `cube-triple-play`, and `cube-quadruple-play` families, using Tasks 1–4 in each family, as well as Task 4 from `scene-play` and Tasks 2 and 4 from `puzzle-4x4-play`, which use sparse rewards. For D4RL Adroit, we evaluate on `pen`, `door`, and `hammer`. Additional details on the environments and datasets are provided in Appendix A.1.

Baselines and our method We compare against five reinforcement learning baselines from two algorithmic categories: (1) **End-to-End Generative Fine-Tuning**, including **QC** (Q-Learning with Action Chunking) [14], **QAM** (Q-Learning with Adjoint Matching) [17], **FBRAC** (flow counterpart of DQL[1])[4], and **IFQL**, the flow instantiation of IDQL [18]; and (2) **Latent Space Steering**, represented by **DSRL** (Diffusion Steering RL) [8] and our proposed **MCFS**. These methods differ in how RL signals enter the policy update: QC and IFQL mainly retain online flow-matching training and use the critic primarily for action evaluation or selection; FBRAC directly updates the flow policy by backpropagating the action-space Q signal through the full flow model with BPTT; QAM further injects critic gradients into a dedicated fast flow through adjoint matching; and DSRL together with our method instead uses Q value information to steer a separate noise policy while keeping the pre-trained generative backbone frozen. This categorization allows us to evaluate both the overall benefit of frozen-flow fine-tuning and the method-specific advantage of our design within latent steering framework.

4.2 Main Results

Following standard practices, we report normalized return for D4RL and success rate for OGBench. Our proposed MCFS steers noise under the moment constraint. It achieves the best overall results on both benchmark families, improving asymptotic performance and sample efficiency over the latent-steering baseline DSRL [8] as well as the end-to-end flow fine-tuning baselines. Main training

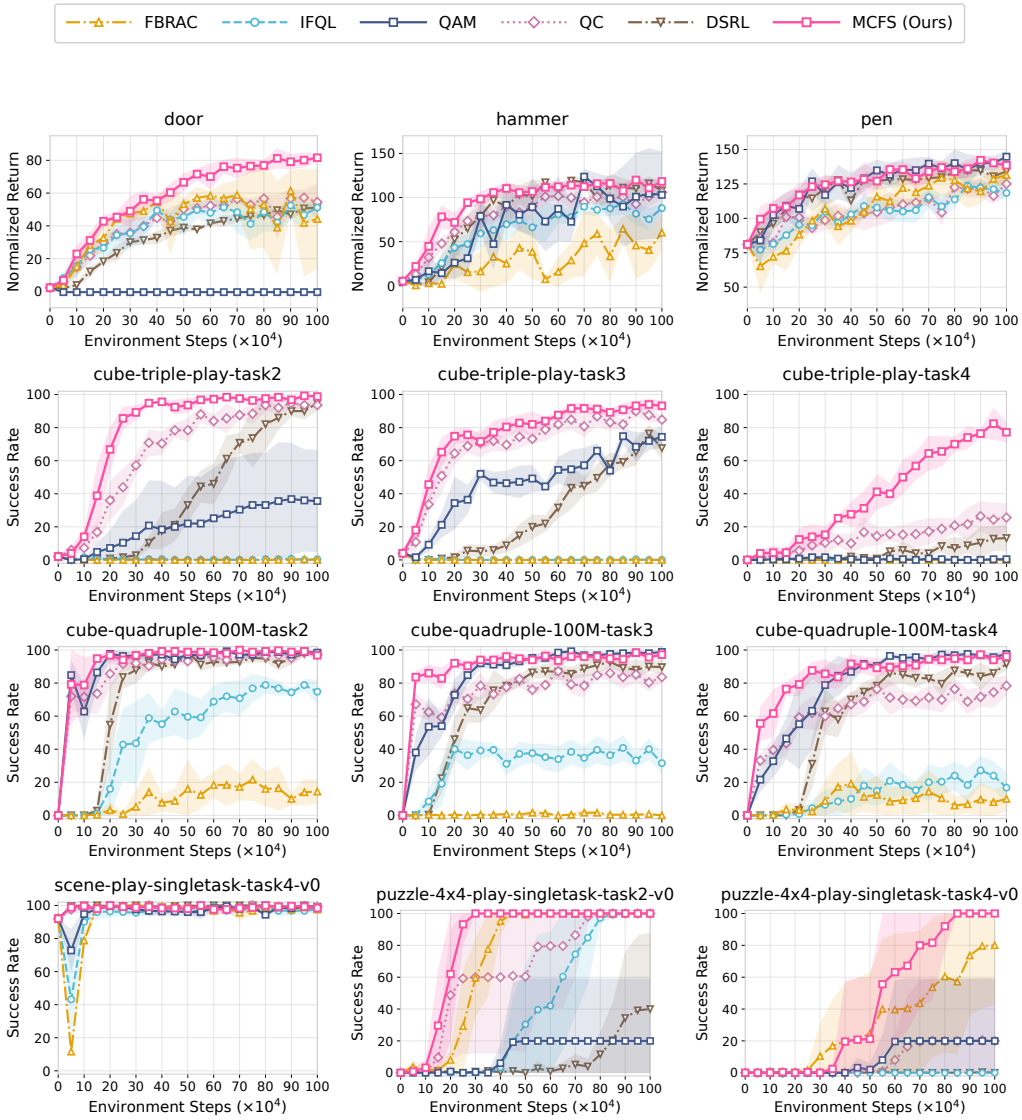


Figure 3: Online adaptation curves on D4RL and OGBench benchmarks. The solid lines denote the mean performance over seeds and the shaded regions show the corresponding 95% confidence intervals. Full offline-to-online results are deferred to Appendix A.4.

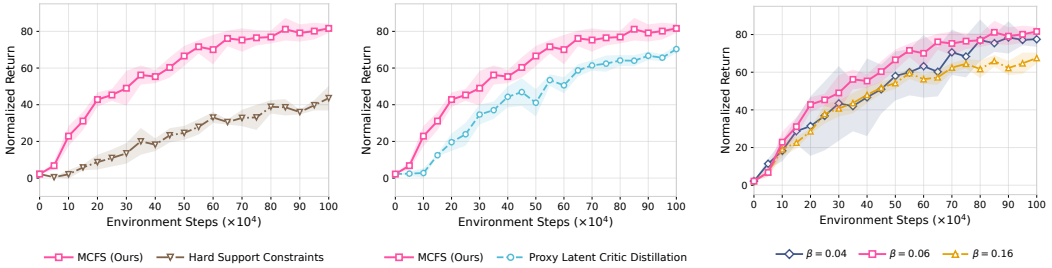
curves are summarized in Figure 3, with the corresponding quantitative results reported in Table 1. For OGBench, we report selected OGBench tasks here; full offline-to-online visualizations and the complete OGBench results are deferred to Appendix A.4.

Asymptotic Performance MCFS yields top-tier performance across the twelve environments, achieving either the best or second-best result in every case. It also attains the best average performance on both benchmark families, reaching **112.33** on D4RL and **0.956** on the exhibited OGBench tasks, improving over the second-best by **+14.6%** and **+26.4%**, respectively. The most prominent advantage of MCFS is observed in `cube-triple-play-task4`, one of the hardest OGBench settings. MCFS reaches a success rate of **0.787**, which is **+208.6%** higher than the second-best result.

Sample Efficiency In Figure 3, MCFS shows a clear early advantage and converges faster than the baselines. This likely reflects two features of our method: steering the latent noise with the moment

Table 1: Main quantitative results. D4RL and OGBench rows report their corresponding metrics averaged over the final 10% of online training steps, both summarized as Mean ± Std over seeds. **Bold** indicates the best result in each row and underline indicates the second-best.

Task	FBRAC	IFQL	QAM	QC	DSRL	MCFS (ours)
door	49.151 ± 22.019	50.113 ± 7.308	-0.457 ± 0.075	56.186 ± 4.012	49.603 ± 2.155	**80.296 ± 2.972**
hammer	48.757 ± 19.083	81.686 ± 4.637	102.749 ± 57.101	99.876 ± 6.479	111.376 ± 6.654	**116.313 ± 5.253**
pen	126.633 ± 6.028	120.587 ± 3.903	**141.717 ± 4.234**	121.198 ± 6.629	133.103 ± 5.578	140.367 ± 1.916
D4RL Average	74.847 ± 6.581	84.129 ± 2.953	81.336 ± 20.233	92.420 ± 3.654	98.027 ± 3.106	**112.325 ± 1.832**
cube-triple-play-task2	0.001 ± 0.003	0.004 ± 0.006	0.361 ± 0.363	0.940 ± 0.017	0.916 ± 0.044	**0.983 ± 0.006**
cube-triple-play-task3	0.000 ± 0.000	0.001 ± 0.003	0.716 ± 0.065	0.875 ± 0.024	0.697 ± 0.055	**0.935 ± 0.020**
cube-triple-play-task4	0.000 ± 0.000	0.000 ± 0.000	0.003 ± 0.004	0.255 ± 0.092	0.121 ± 0.090	**0.787 ± 0.032**
cube-quadruple-100M-task2	0.128 ± 0.060	0.760 ± 0.039	0.981 ± 0.010	0.969 ± 0.025	0.968 ± 0.017	**0.983 ± 0.011**
cube-quadruple-100M-task3	0.004 ± 0.004	0.349 ± 0.056	**0.984 ± 0.010**	0.831 ± 0.039	0.892 ± 0.038	0.971 ± 0.004
cube-quadruple-100M-task4	0.092 ± 0.095	0.227 ± 0.078	**0.967 ± 0.018**	0.748 ± 0.066	0.872 ± 0.034	0.957 ± 0.016
scene-play-task4	0.980 ± 0.005	0.971 ± 0.026	0.987 ± 0.007	0.991 ± 0.004	**0.995 ± 0.006**	0.991 ± 0.011
puzzle-4x4-play-task2	**1.000 ± 0.000**	0.999 ± 0.003	0.200 ± 0.447	**1.000 ± 0.000**	0.379 ± 0.519	**1.000 ± 0.000**
puzzle-4x4-play-task4	0.777 ± 0.430	0.000 ± 0.000	0.200 ± 0.447	0.201 ± 0.446	0.001 ± 0.003	**1.000 ± 0.000**
Selected OGBench Average	0.331 ± 0.042	0.368 ± 0.019	0.600 ± 0.078	0.757 ± 0.060	0.649 ± 0.069	**0.956 ± 0.003**



(a) Moment Constraint

(b) Adjoint Q-value Backpropagation

(c) Entropy Sensitivity

Figure 4: Ablation studies on `door-cloned-v1`. The first two subplots compare the full MCFS method against targeted mechanism variants, isolating the effect of *Moment Constraint* and *Adjoint Q-value Backpropagation*. The third subplot studies sensitivity to the latent-policy entropy coefficient β. Curves show mean normalized return over seeds with 95% confidence intervals.

constraint is more amenable to optimization than updating the full flow policy; backpropagating the exact action-space critic gradient, rather than relying on a distilled proxy critic, provides a more direct and informative learning signal.

4.3 Ablation Studies

We conduct three ablation studies on `door-cloned-v1`, shown in Figure 4, to isolate the roles of *Moment Constraint*, *Adjoint Q-value Backpropagation*, and the latent-policy entropy coefficient β.

Moment Constraint We replace *Moment Constraint* with *Hard Support Boundaries* to test whether the second-moment formulation is necessary. As shown in Figure 4(a), this change causes a substantial degradation throughout training. This suggests that rigid support clipping over-constrains latent adaptation, whereas the moment-based formulation allows more flexible and effective steering under a shared statistical budget.

Adjoint Q-value Backpropagation We replace *Adjoint Q-value Backpropagation* with *Proxy Latent Critic Distillation* to assess the value of direct gradient propagation through the continuous-time dynamics. Figure 4(b) shows that the distilled variant remains viable, but its learning curve is consistently weaker than that of the full method. This supports the view that direct action-space value gradients provide a more effective updating signal than an auxiliary latent-space critic. We also observe a substantially smaller memory footprint than direct BPTT, with detailed resource comparisons deferred to Appendix A.5.

Entropy Coefficient Sensitivity to the entropy coefficient β is studied to assess latent-policy exploration. Figure 4(c) indicates that performance is best in an intermediate regime: weaker entropy regularization still yields useful steering but with less stable behavior, whereas excessive entropy leads to clear degradation. This suggests that moderate stochasticity best balances exploration and value-guided exploitation.

5 Conclusion

In this paper, we studied how to adapt pre-trained flow policies with online RL while retaining the generative backbone learned during offline pre-training. We proposed MCFS, which combines a moment-constrained latent policy parameterization with adjoint Q-gradient propagation. The moment constraint provides a soft statistical mechanism for limiting OOD latent shifts while preserving policy expressiveness, and the adjoint formulation enables exact critic-gradient propagation through frozen flow dynamics without relying on a proxy latent critic. Empirically, MCFS achieves the strongest overall performance on both D4RL and OGBench, with especially clear gains on difficult long-horizon compositional manipulation tasks, including those with sparse rewards. The results also show improved early-stage sample efficiency, supporting our central claim that promoting prior-density compatibility and maintaining expressive online adaptation are not inherently conflicting objectives. Our work shows that latent-space constraints can enable efficient and effective RL fine-tuning of generative models.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 92582205), the Beijing Natural Science Foundation (Grant No. L257002), and the Tsinghua University (State Key Laboratory of Intelligent Green Vehicle and Mobility)–Xiaomi EV Technology Co., Ltd Joint Research Center for Autonomous Driving Safety and Intelligence. The authors declare no competing interests.

References

[1] 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.

[2] Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, et al. Diffusion actor-critic with entropy regulator. *Advances in Neural Information Processing Systems*, 37:54183–54204, 2024.

[3] Yinuo Wang, Likun Wang, Mining Tan, Wenjun Zou, Xujie Song, Wenxuan Wang, Tong Liu, Guojian Zhan, Tianze Zhu, Shiqi Liu, et al. Enhanced dacer algorithm with high diffusion efficiency. *arXiv preprint arXiv:2505.23426*, 2025.

[4] Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning, 2025.

[5] Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. *arXiv preprint arXiv:2506.12811*, 2025.

[6] Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

[7] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

[8] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

[9] Hokyun Im, Andrey Kolobov, Jianlong Fu, and Youngwoon Lee. Latent policy steering through one-step flow policies. *arXiv preprint arXiv:2603.05296*, 2026.

[10] Bram Wallace, Akash Gokul, Stefano Ermon, and Nikhil Naik. End-to-end diffusion latent optimization improves classifier guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7280–7290, 2023.

[11] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[12] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. Routledge, 2018.

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

[14] Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. *arXiv preprint arXiv:2507.07969*, 2025.

[15] Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. *arXiv preprint arXiv:2410.20092*, 2024.

[16] 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.

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

[18] 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.

[19] Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *arXiv preprint arXiv:2506.04168*, 2025.

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

[21] Xuanran Zhai, Qianyou Zhao, Qiaojun Yu, and Ce Hao. Vfp: Variational flow-matching policy for multi-modal robot manipulation. *arXiv preprint arXiv:2508.01622*, 2025.

[22] Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.

[23] Haoran Ding, Noémie Jaquier, Jan Peters, and Leonel Rozo. Fast and robust visuomotor riemannian flow matching policy. *IEEE Transactions on Robotics*, 2025.

[24] Zibin Dong, Yifu Yuan, Jianye Hao, Fei Ni, Yao Mu, Yan Zheng, Yujing Hu, Tangjie Lv, Changjie Fan, and Zhipeng Hu. Aligndiff: Aligning diverse human preferences via behavior-customisable diffusion model. *arXiv preprint arXiv:2310.02054*, 2023.

[25] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

[26] Samuel Pfrommer, Yixiao Huang, and Somayeh Sojoudi. Reinforcement learning for flow-matching policies. *arXiv preprint arXiv:2507.15073*, 2025.

[27] David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 2025.

[28] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

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

[30] Tianyu Chen, Zhendong Wang, and Mingyuan Zhou. Diffusion policies creating a trust region for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:50098–50125, 2024.

[31] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

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

[33] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

[34] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. Pmlr, 2017.

[35] Zuoshuai Wang, Yao Lyu, Congsheng Zhang, Zhe Xin, Jiaxin Gao, Beiyan Jiang, and Shengbo Eben Li. Feasible reinforcement learning for safety ensurance with stabilized training. *Automotive Innovation*, 9(2):340–353, 2026.

[36] Amir Gholami, Kurt Keutzer, and George Biros. Anode: Unconditionally accurate memory-efficient gradients for neural odes. *arXiv preprint arXiv:1902.10298*, 2019.

[37] Yinuo Wang, Wenxuan Wang, Xujie Song, Tong Liu, Yuming Yin, Liangfa Chen, Likun Wang, Jingliang Duan, and Shengbo Eben Li. Ode-based smoothing neural network for reinforcement learning tasks. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[38] Chen Sun, Ruihe Zhang, Ahmad Reza Alghooneh, Minghao Ning, Pouya Panahandeh, Steven Tuer, and Amir Khajepour. Cascaded safety analysis and test scenario generation techniques for autonomous driving: A case study with watonobus. *Automotive Innovation*, 8(2):252–263, 2025.

A Experimental Details

A.1 Environments and Datasets

We consider two benchmark families in this work: the Adroit hand-manipulation tasks from D4RL[16] and the Cube, Scene, and Puzzle manipulation tasks from OGBench[15]. D4RL provides dexterous manipulation benchmarks built on the Adroit platform, while OGBench provides multi-object tabletop manipulation benchmarks built on a UR5e arm with a Robotiq gripper.

D4RL Adroit. We use the three Adroit manipulation tasks `pen`, `door`, and `hammer`. The `pen` task uses a 45-dimensional state and a 24-dimensional action, and requires reorienting the pen to match a target pose. Its dense reward combines the pen-to-target distance, the orientation similarity between the pen and the target, bonuses when the pose is within the official distance-and-orientation tolerances, and a penalty for dropping the pen. The `door` task uses a 39-dimensional state and a 28-dimensional action, and requires the agent to unlatch and swing the door open. Its official dense reward combines a hand-to-handle shaping term, a door-angle term centered at the open configuration, a small velocity penalty, and success bonuses when the door hinge exceeds 0.2, 1.0, and 1.35 radians. The `hammer` task uses a 46-dimensional state and a 26-dimensional action, and asks the agent to grasp the hammer and drive the nail into the board. Its dense reward includes shaping for reaching the hammer, aligning the hammer head with the nail, pushing the nail toward its goal, a velocity penalty, a bonus for lifting the hammer, and additional bonuses when the nail is within 0.02 and 0.01 meters of the goal.

For each Adroit task, D4RL provides three offline dataset variants: `human`, `cloned`, and `expert`. In the official dataset documentation, the `human` split consists of 25 human demonstration episodes, the `expert` split contains 5000 trajectories from a fine-tuned RL policy, and the `cloned` split is constructed by training an imitation policy on the human and expert demonstrations and then mixing its rollouts with the demonstrations at a 50–50 ratio. In this paper, we use the `*-cloned-v1` datasets for all three Adroit tasks. This choice matches recent Adroit evaluations of flow-policy offline-to-online RL methods, and is also a reasonable compromise between the two extremes of the benchmark: it is substantially broader than the very small `human` split, while remaining less idealized than the pure `expert` split.

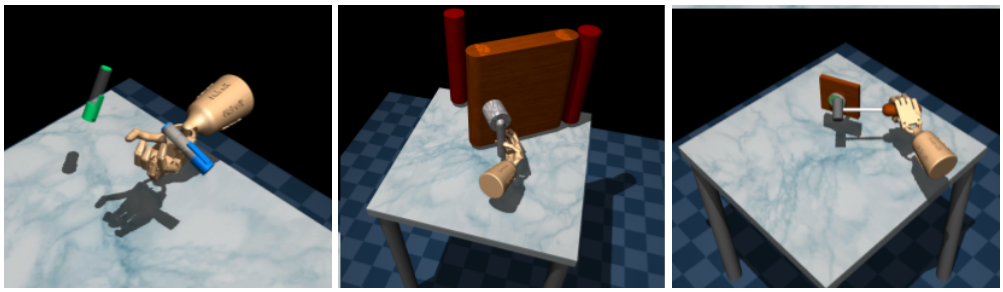


Figure 5: Official task illustrations for the D4RL Adroit benchmarks used in this paper: `pen`, `door`, and `hammer`.

Table 2: D4RL Adroit tasks used in our experiments.

Task	State Dim.	Action Dim.	Metric	Task Goal	Reward Setting
`pen`	45	24	Normalized return	Reorient the pen to match the target pose while keeping it in hand.	Official dense reward with pen-to-target distance, orientation similarity, bonuses for satisfying the official pose tolerances, and a dropping penalty when the pen height falls below the threshold.
`door`	39	28	Normalized return	Undo the latch and swing the door fully open.	Official dense reward with reaching-to-handle shaping, door-angle shaping, a small velocity penalty, and bonuses at hinge angles > 0.2, > 1.0, and > 1.35.
`hammer`	46	26	Normalized return	Pick up the hammer and drive the nail into the board.	Official dense reward with hammer-reaching, hammer-head-to-nail alignment, nail-to-goal shaping, a velocity penalty, a lifting bonus, and bonuses when the nail is within 0.02 and 0.01 m of the goal.

OGBench. We also evaluate on the official OGBench manipulation suite, using the `cube-double-play`, `cube-triple-play`, `cube-quadruple-play`, `scene-play`, and `puzzle-4x4-play` families with single-task evaluation goals. These environments are built on a UR5e arm with a Robotiq 2F-85 gripper and a common 5-dimensional action space corresponding to relative end-effector translation in (x, y, z), relative end-effector yaw, and gripper opening. For the Cube families, the state observation concatenates robot proprioception and per-cube pose information. This yields 37, 46, and 55 state dimensions for the double-, triple-, and quadruple-cube variants, respectively. For the Cube families in single-task RL mode, the official implementation computes the reward as the number of cubes already within the 0.04 m success threshold minus the total number of cubes, so the reward reaches 0 only when all cubes satisfy their target constraints.

The specific tasks used in this paper are the official pre-defined single-task goals. Following the main text, we evaluate Tasks 1–4 for each Cube family. Concretely, `cube-double-play` contains a single-cube relocation task in which one cube remains fixed while the other is moved across the table, followed by moving both cubes to upper-row targets, redistributing two center-line cubes to separated side targets, and swapping the two cube locations. `cube-triple-play` begins with a partial relocation task that preserves two cubes and moves the third to the opposite side, then includes transferring a three-cube column to distinct targets, unstacking a vertical pile into tabletop targets, and performing a cyclic permutation over three cube locations. `cube-quadruple-100M` starts from a partial two-cube rearrangement that pushes the outer cubes farther apart while leaving the inner pair unchanged, and then includes rearranging four cubes into a compact central pattern, dispersing a square stack into separated targets, and performing a four-way cyclic permutation.

Beyond Cube, we evaluate Task 4 from `scene-play` and Tasks 2 and 4 from `puzzle-4x4-play`. Scene contains a cube, two buttons, a drawer, and a window; the buttons toggle the locks of the drawer and window. Task 4 (`task4_put_in_drawer`) requires unlocking and opening the drawer, placing the cube inside, and closing the drawer, with the drawer unlocked and the window closed and locked in the target configuration. Its 40-dimensional state comprises robot proprioception, cube pose, button states and joint positions/velocities, and drawer/window positions and velocities. Official success requires the cube to be within 0.04 m of its target, both button states to match their targets, and the drawer and window positions to be within 0.04 m of their targets.

Puzzle implements a robotic "Lights Out" task: pressing a button toggles its binary color state and those of its orthogonally adjacent neighbors. We use the 4×4 grid with 16 buttons and an 83-dimensional state comprising robot proprioception and each button's one-hot color state, joint position, and joint velocity. Task 2 starts with all buttons blue and targets a red central 2×2 square surrounded by blue buttons; Task 4 starts with blue outer columns and red inner columns and targets an all-red grid. Success requires all 16 button states to match the target pattern. Both Scene and Puzzle retain the same 5-dimensional end-effector action interface as Cube.

The official single-task reward is the number of satisfied object conditions minus five for Scene and the number of correctly configured buttons minus 16 for Puzzle. In our Scene and Puzzle experiments, we apply the same sparse-reward transformation in offline pretraining and online adaptation: $r = 0$ when all target conditions are satisfied and $r = -1$ otherwise.

Regarding offline data, OGBench provides two manipulation dataset types, `play` and `noisy`, where `play` is the standard dataset type collected by a scripted policy with temporally correlated noise and `noisy` is a higher-coverage variant with larger uncorrelated Gaussian perturbations. For `cube-double-play` and `cube-triple-play`, we use the default `play` datasets loaded by the official OGBench API, i.e., the standard datasets that are automatically downloaded for these tasks. For the quadruple-cube setting, we follow the protocol used by QAM [17] and replace the default-scale dataset with the larger manually downloaded 100M-transition `play` dataset, namely `cube-quadruple-play-100m-v0` as provided by Park et al. [19]. For Scene, we use the default `scene-play` dataset; For both Puzzle tasks, we use the `puzzle-4x4-play-100m-v0` dataset specified in our experiment configurations, with a dataset replacement interval of 1000 steps in both training stages.

A.2 Experimental Settings

All experiments are conducted under an offline-to-online (O2O) adaptation setting that aligns with the general paradigm of flow model pre-training followed by reinforcement learning fine-tuning. The

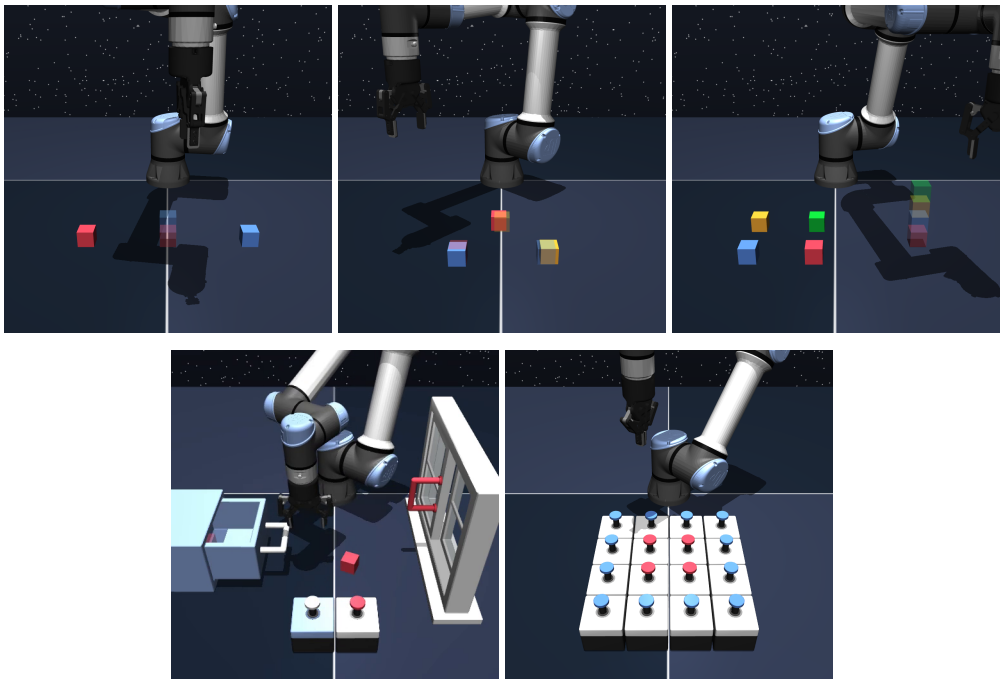


Figure 6: Illustrations of the OGBench manipulation environments used in this paper. Top row, from left to right: `cube-double-play`, `cube-triple-play`, and `cube-quadruple-100M`. Bottom row, from left to right: `scene-play` and `puzzle-4x4-play`.

Table 3: OGBench manipulation families used in our experiments.

Environment Family	State Dim.	Action Dim.	Metric	Tasks Used in This Paper	Reward Setting
`cube-double-play`	37	5	Success rate	`task1_single_pnp`, `task2_double_pnp1`, `task3_double_pnp2`, `task4_swap`.	Single-task official reward: $\sum_i \mathbb{1}[\|p_i - p_i^*\| \leq 0.04] - 2$, with success achieved only when both cubes satisfy their targets.
`cube-triple-play`	46	5	Success rate	`task1_single_pnp`, `task2_triple_pnp`, `task3_pnp_from_stack`, `task4_cycle`.	Single-task official reward: $\sum_i \mathbb{1}[\|p_i - p_i^*\| \leq 0.04] - 3$, with success achieved only when all three cubes satisfy their targets.
`cube-quadruple-100M`	55	5	Success rate	`task1_double_pnp`, `task2_quadruple_pnp`, `task3_pnp_from_square`, `task4_cycle`.	Single-task official reward: $\sum_i \mathbb{1}[\|p_i - p_i^*\| \leq 0.04] - 4$, with success achieved only when all four cubes satisfy their targets.
`scene-play`	40	5	Success rate	Task 4: `task4_put_in_drawer`; place the cube inside the drawer and close it with the prescribed lock states.	Sparse reward used in our experiments: 0 if all five object conditions are satisfied, and -1 otherwise; cube and drawer/window position tolerances are 0.04 m, and button states must match exactly.
`puzzle-4x4-play`	83	5	Success rate	Tasks 2 and 4: form the prescribed central-square pattern and the all-red pattern, respectively.	Sparse reward used in our experiments: 0 if all 16 button states match the target, and -1 otherwise.

overall training pipeline is decomposed into two stages. In the *offline pre-training* stage, we jointly train a conditional flow policy and an action-space Q-function on offline demonstration datasets, adopting the same action-chunking practice as in [14]. Concretely, although the pretraining policy operates on action chunks, we write the objective below using the compact symbol a for the chunked action output. The flow model f_θ is trained with the flow-matching objective

$$\begin{aligned}\mathcal{L}_{\text{flow}}(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D},\, t\sim\mathcal{U}[0,1],\, z\sim\mathcal{N}(0,I)} \\ \|f_\theta(t\,a + (1-t)z,\, t;\, s) - (a - z)\|_2^2\,,\end{aligned} \tag{10}$$

while the critic is trained with the chunked temporal-difference loss

$$\mathcal{L}_{\text{critic}}(\psi) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left(Q_\psi(s,a) - r^{(h)} - \gamma^h Q_{\bar{\psi}}(s', a^*)\right)^2, \tag{11}$$

where $r^{(h)} = \sum_{i=0}^{h-1} \gamma^i r_{t+i}$ and a^* is obtained by sampling N candidate chunked actions from the policy at s' and choosing one by the standard *Best-of-N* practice. These offline objectives produce the

Table 4: Pretraining hyperparameters.

Parameter	Value
Offline training steps	1.0×10^6
Online environment steps	0
Replay buffer size	1.0×10^6
Batch size	256
Optimizer	Adam
Learning rate	3×10^{-4}
Discount factor γ	0.99
UTD ratio	1
Flow rollout steps	10
Horizon length	5
Critic ensemble size	2
Target update rate τ	5×10^{-3}
Best-of-N	16
Q aggregation	mean
Actor/Value MLPs	$[512, 512, 512, 512]$
Action chunking	True

shared pretrained initialization used for subsequent online adaptation. In the *online adaptation* stage, all evaluated methods are initialized from the same pretrained checkpoints to ensure a fair comparison, while algorithm-specific auxiliary components, such as latent noise policies, are initialized separately when required.

To make the comparison more demanding, our method and other strict latent-space steering baselines (e.g., DSRL [8]) keep the base flow policy frozen during the online phase and update only the lightweight noise policy and the critic. End-to-end generative fine-tuning methods, by contrast, continue updating the entire flow policy during online interaction. This setup makes online adaptation substantially more challenging for latent-space steering methods, thereby providing a stricter evaluation of their practical effectiveness.

A.3 Hyperparameters

We summarize the main hyperparameter settings used in our experiments in Tables 4–6. Each method is evaluated under the same protocol with five random seeds. Table 4 reports the shared offline pretraining setup used to produce the initialization checkpoints. Table 5 lists the hyperparameters shared by the online adaptation runs. Table 6 reports the task-specific algorithm hyperparameters for all 18 environments used in the experiments. Besides, for `cube-quadruple-100M` datasets, we use a dataset replace interval of 1000 steps. For QAM, we use the original formulation rather than its QAM-FQL or QAM-EDIT variants, since those variants introduce additional auxiliary policies on top of the core adjoint-matching objective; using the original QAM therefore provides the cleanest comparison for isolating the effect of adjoint-based guidance itself.

A.4 Supplementary Results for Main Experiments

To visualize the entire training process rather than only the online stage, Figures 7 and 8 present continuous offline-to-online training curves. In these plots, the gray-shaded interval $[0, 1.0] \times 10^6$ corresponds to the shared offline pre-training phase. Since all methods start from the same pre-trained backbone, this region is represented by a single shared pre-training curve. The online curves trace the algorithm-specific online fine-tuning dynamics over the following 1.0×10^6 environment steps. This visualization separates the common prior acquired during offline representation learning from the gains obtained during downstream online adaptation. Please refer to Table 1 in the main text and Table 7 for the final quantitative results corresponding to these curves. Note that average rows in these tables are computed by first averaging the corresponding late-stage metric across the included tasks for each seed.

Table 5: Common hyperparameters for online adaptation.

Parameter	Value
Online environment steps	1.0×10^6
Replay buffer size	1.0×10^6
Batch size	256
Optimizer	Adam
Learning rate	3×10^{-4}
Discount factor γ	0.99
UTD ratio	1
Flow rollout steps	10
Horizon length	5
Critic ensemble size	2
Target update rate τ	5×10^{-3}
Actor/value MLPs	$[512, 512, 512, 512]$
Action chunking	True

Table 6: Task-specific algorithm hyperparameters. Column symbols: N, number of samples used in *Best-of-N*; τ_{inv}, inverse temperature in QAM; κ, IFQL expectile; α, FBRAC coefficient; σ, noise scale in DSRL; η, target-entropy multiplier in DSRL; and β, policy entropy coefficient of MCFS.

Environment	QC N	QAM (τ_{inv}, N)	IFQL (κ, N)	FBRAC α	DSRL (σ, η)	MCFS (β, N)
`door-cloned-v1`	16	$(0.3, 16)$	$(0.9, 16)$	10000	$(1.0, 0.5)$	$(0.06, 16)$
`hammer-cloned-v1`	16	$(0.3, 16)$	$(0.9, 16)$	10000	$(1.0, 0.5)$	$(0.08, 16)$
`pen-cloned-v1`	16	$(0.3, 16)$	$(0.9, 16)$	10000	$(1.0, 0.5)$	$(0.08, 16)$
`cube-double-play-task1`	16	$(1.0, 16)$	$(0.9, 16)$	0.1	$(1.0, 0.5)$	$(0.04, 16)$
`cube-double-play-task2`	16	$(1.0, 16)$	$(0.9, 16)$	0.1	$(1.0, 0.5)$	$(0.04, 16)$
`cube-double-play-task3`	16	$(1.0, 16)$	$(0.9, 16)$	0.1	$(1.0, 0.5)$	$(0.04, 16)$
`cube-double-play-task4`	16	$(1.0, 16)$	$(0.9, 16)$	0.1	$(1.0, 0.5)$	$(0.04, 16)$
`cube-triple-play-task1`	16	$(3.0, 16)$	$(0.95, 16)$	0.03	$(1.0, 0.5)$	$(0.04, 16)$
`cube-triple-play-task2`	16	$(3.0, 16)$	$(0.95, 16)$	0.03	$(1.0, 0.5)$	$(0.04, 16)$
`cube-triple-play-task3`	16	$(3.0, 16)$	$(0.95, 16)$	0.03	$(1.0, 0.5)$	$(0.04, 16)$
`cube-triple-play-task4`	16	$(3.0, 16)$	$(0.95, 16)$	0.03	$(1.0, 0.5)$	$(0.04, 16)$
`cube-quadruple-100M-task1`	16	$(1.0, 16)$	$(0.95, 16)$	1.0	$(1.0, 0.5)$	$(0.04, 16)$
`cube-quadruple-100M-task2`	16	$(1.0, 16)$	$(0.95, 16)$	1.0	$(1.0, 0.5)$	$(0.04, 16)$
`cube-quadruple-100M-task3`	16	$(1.0, 16)$	$(0.95, 16)$	1.0	$(1.0, 0.5)$	$(0.04, 16)$
`cube-quadruple-100M-task4`	16	$(1.0, 16)$	$(0.95, 16)$	1.0	$(1.0, 0.5)$	$(0.04, 16)$
`scene-play-task4`	16	$(1.0, 16)$	$(0.9, 16)$	100	$(1.0, 0.5)$	$(0.04, 16)$
`puzzle-4x4-play-task2`	16	$(30.0, 16)$	$(0.9, 16)$	0.3	$(1.0, 0.5)$	$(0.04, 16)$
`puzzle-4x4-play-task4`	16	$(30.0, 16)$	$(0.9, 16)$	0.3	$(1.0, 0.5)$	$(0.04, 16)$

A.5 Supplementary Results for Ablation Studies

To complement the mechanism and sensitivity ablation in Section 4.3, we first summarize the late-stage quantitative results in Table 8.

We further compare the device-memory cost of actor-gradient evaluation for Adjoint and BPTT. All measurements are taken under the same JAX/XLA backend, actor architecture, and dummy input construction, using the observation and action dimensions of `cube-triple-play-singletask-task3-v0`. We report four groups of experiments that vary the flow rollout steps, hidden width, hidden depth, and actor-gradient batch size, respectively.

As shown in Figure 9, the resulting trends consistently favor the step-wise VJP-based adjoint backpropagation used in MCFS. The gap is modest in the easiest settings, but grows substantially as the rollout becomes longer or the actor becomes wider, deeper, or more heavily batched. For example, at 128 flow steps, BPTT reaches 2948 MB while Adjoint uses 900 MB, and at width 8192, BPTT reaches 8852 MB while Adjoint uses 4754 MB. This difference is consistent with

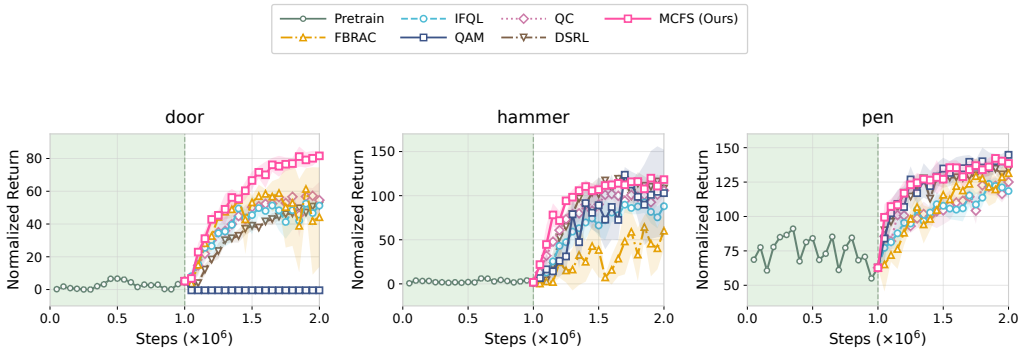


Figure 7: Continuous offline-to-online training curves on D4RL. The gray-shaded region denotes the shared offline pre-training phase of the pretrained backbone. The solid colored curves then show how each method fine-tunes from the same final pre-training checkpoint during the online stage.

Table 7: Full OGBench quantitative results. We report success rate averaged over the final 10% of online training steps, summarized as Mean ± Std over seeds. The average row is computed by first averaging the corresponding late-stage metric across all included tasks for each seed. **Bold** indicates the best result in each row and underline indicates the second-best.

Task	FBRAC	IFQL	QAM	QC	DSRL	MCFS (ours)
cube-double-play-task1	0.919 ± 0.031	0.992 ± 0.006	0.995 ± 0.006	0.999 ± 0.003	**1.000 ± 0.000**	0.999 ± 0.003
cube-double-play-task2	0.731 ± 0.067	0.965 ± 0.028	0.999 ± 0.003	0.996 ± 0.006	0.999 ± 0.003	**1.000 ± 0.000**
cube-double-play-task3	0.803 ± 0.040	0.984 ± 0.012	0.999 ± 0.003	0.999 ± 0.003	**1.000 ± 0.000**	0.999 ± 0.003
cube-double-play-task4	0.189 ± 0.026	0.020 ± 0.020	**0.999 ± 0.003**	0.973 ± 0.034	**0.999 ± 0.003**	0.996 ± 0.004
cube-triple-play-task1	0.000 ± 0.000	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	0.997 ± 0.004	**1.000 ± 0.000**
cube-triple-play-task2	0.001 ± 0.003	0.004 ± 0.006	0.361 ± 0.363	0.940 ± 0.017	0.916 ± 0.044	**0.983 ± 0.006**
cube-triple-play-task3	0.000 ± 0.000	0.001 ± 0.003	0.716 ± 0.065	0.875 ± 0.024	0.697 ± 0.055	**0.935 ± 0.020**
cube-triple-play-task4	0.000 ± 0.000	0.000 ± 0.000	0.003 ± 0.004	0.255 ± 0.092	0.121 ± 0.090	**0.787 ± 0.032**
cube-quadruple-100M-task1	0.705 ± 0.404	0.951 ± 0.029	**1.000 ± 0.000**	0.997 ± 0.004	0.999 ± 0.003	**1.000 ± 0.000**
cube-quadruple-100M-task2	0.128 ± 0.060	0.760 ± 0.039	0.981 ± 0.010	0.969 ± 0.025	0.968 ± 0.017	**0.983 ± 0.011**
cube-quadruple-100M-task3	0.004 ± 0.004	0.349 ± 0.056	**0.984 ± 0.010**	0.831 ± 0.039	0.892 ± 0.038	0.971 ± 0.004
cube-quadruple-100M-task4	0.092 ± 0.095	0.227 ± 0.078	**0.967 ± 0.018**	0.748 ± 0.066	0.872 ± 0.034	0.957 ± 0.016
scene-play-task4	0.980 ± 0.005	0.971 ± 0.026	0.987 ± 0.007	0.991 ± 0.004	**0.995 ± 0.006**	0.991 ± 0.011
puzzle-4x4-play-task2	**1.000 ± 0.000**	0.999 ± 0.003	0.200 ± 0.447	**1.000 ± 0.000**	0.379 ± 0.519	**1.000 ± 0.000**
puzzle-4x4-play-task4	0.777 ± 0.430	0.000 ± 0.000	0.200 ± 0.447	0.201 ± 0.446	0.001 ± 0.003	**1.000 ± 0.000**
OGBench Average	0.422 ± 0.029	0.548 ± 0.013	0.759 ± 0.046	0.852 ± 0.038	0.789 ± 0.042	**0.973 ± 0.002**

the underlying computation. BPTT differentiates through the full unrolled rollout and therefore accumulates memory over intermediate activations as the rollout and network scale increase. By contrast, the adjoint implementation still stores the forward trajectory needed for backward integration, but avoids retaining the full BPTT graph through the rollout, which yields a much milder growth in device-memory usage. Overall, Adjoint shows a smoother memory profile across these settings and is the more practical choice when memory efficiency matters.

A.6 Compute Resources

All experiments were conducted on compute resources equipped with Intel Xeon Platinum 8468V CPUs and NVIDIA H20 GPUs. The main offline pretraining and online adaptation experiments were run on a single GPU per run. In our setup, the reported training required several hours for most of the runs, depending on the environment and method.

A.7 External Code Assets and Licenses

Our implementation was developed on top of two public research codebases released by their original authors: `https://github.com/ColinQiyangLi/qc` and `https://github.com/ColinQiyangLi/qam`. Both repositories are distributed under the MIT License. We credit these original assets and use them in accordance with their stated licensing terms.

Table 8: Ablation summary on `door-cloned-v1`. We report normalized return averaged over the final 10% of online training steps, summarized as Mean $\pm$ Std over seeds.

Study	Setting	Return
Mechanism	MCFS (ours)	80.296 ± 2.972
Mechanism	Hard Support Boundaries	39.763 ± 3.675
Mechanism	Proxy Latent Critic Distillation	67.559 ± 1.798
Sensitivity	$\beta = 0.04$	77.720 ± 5.008
Sensitivity	$\beta = 0.06$	80.296 ± 2.972
Sensitivity	$\beta = 0.16$	64.824 ± 3.192

B Additional Related Work

RL Fine-Tuning for Generative Models. Continuous-time generative models, such as diffusion models and flow matching, have demonstrated remarkable success in modeling complex, multimodal action distributions for continuous control [1, 4, 20, 21, 22, 23]. While these models are typically pre-trained via behavioral cloning on offline datasets, adapting them to maximize returns in downstream tasks or new environments requires RL fine-tuning [6, 24, 25, 26, 27]. Standard fine-tuning methods typically update the neural network weights directly using policy gradient [6, 25, 27] or value-based actor-critic objectives [18, 14, 17, 1, 28, 29, 30]. Related efforts have also emphasized stabilizing policy improvement through explicit update budgets, such as trust-region or constrained optimization objectives [31, 32, 33, 34, 30, 35]. However, for continuous-time ordinary differential equation (ODE) models, directly modifying the weights can distort the calibrated vector fields. This often leads to catastrophic forgetting, degrading the behavioral prior learned during pre-training. Consequently, exploring fine-tuning methods that preserve the structural integrity of the pre-trained backbone has become an active area of research.

Latent Policy Steering. To fine-tune policies without altering the pre-trained weights, recent works have explored latent space steering. By freezing the generative backbone, these methods optimize the initial noise distribution to guide the generation process. In robotic control, DSRL [8] effectively steers diffusion policies by training a latent noise policy via online RL. However, DSRL treats the pre-trained model as a black box and relies on distilling the action-space critic into a proxy latent critic, which introduces approximation bias and reduces sample efficiency. Furthermore, it employs hard support boundaries to prevent OOD queries, which restricts the exploration space and expressiveness of the policy. To bypass the reliance on proxy critics, LPS [9] proposes backpropagating Q-value gradients through the generative model. Due to the computational cost and numerical sensitivity of unrolling deep continuous trajectories, issues that have also been highlighted in the Neural ODE gradient literature [36, 37], LPS is restricted to customized one-step flow architectures, limiting its applicability to standard, high-capacity flow models. Unlike these approaches, our proposed MCFS algorithm addresses both challenges simultaneously. It replaces restrictive hard boundaries with a moment constraint parameterization to control distribution shifts without sacrificing policy expressiveness. Moreover, by leveraging the adjoint state method and a gradient interception mechanism, MCFS enables exact and stable Q-gradient backpropagation through deep, arbitrary-step flow models.

C Limitations

Although MCFS shows strong performance, it still has two main limitations. First, the latent policy is modeled as a diagonal Gaussian, which may limit its ability to represent richer exploration patterns. Second, our current adjoint Q-gradient propagation does not yet exploit more specialized iterative acceleration schemes, leaving room for further efficiency gains.

D Positive and Negative Societal Impacts

This work is primarily methodological and does not introduce an immediate direct societal impact. In the positive direction, MCFS may improve the sample efficiency and reliability of learning-based control systems, which could reduce training cost and support more robust adaptation in robotics, autonomous driving and other continuous-control applications. Potential negative impacts are indirect:

as with other reinforcement-learning methods, improved policy optimization could be misused in unsafe or poorly validated autonomous systems, or deployed outside the conditions under which they were evaluated. Careful testing, monitoring, and deployment remain necessary in any safety-critical setting. Such considerations are emphasized in areas such as autonomous driving safety research [38]. Given its potential applications in autonomous driving and robotics, MCFS warrants similar care in evaluation and deployment.

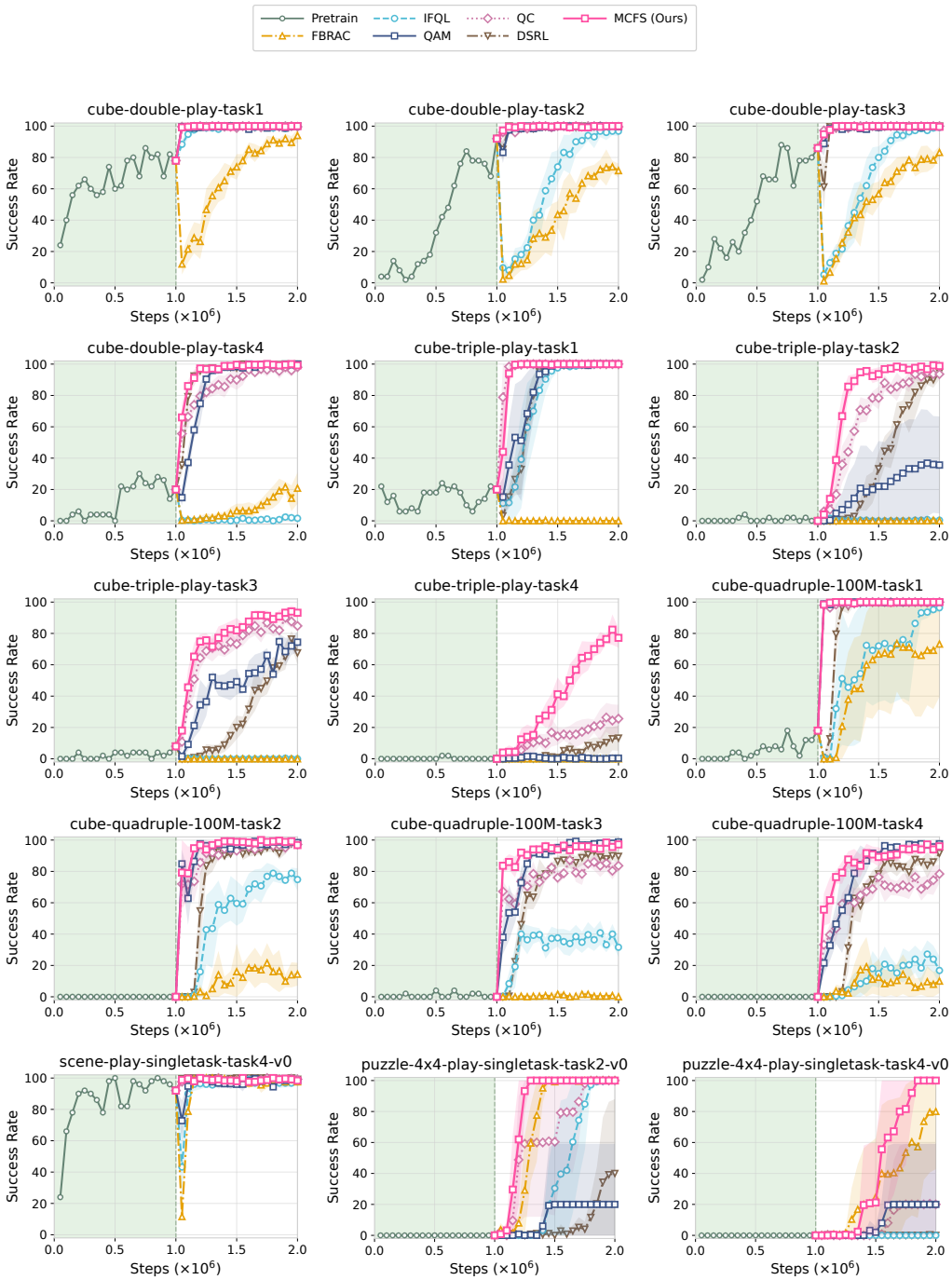
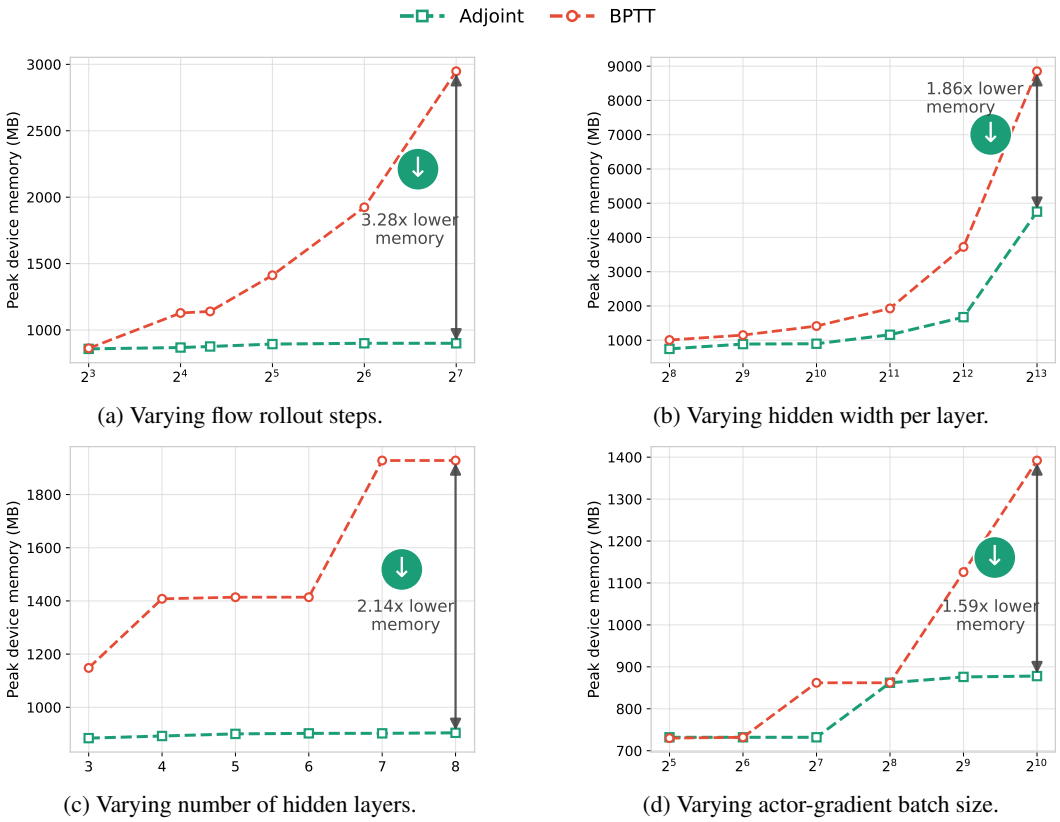


Figure 8: Continuous offline-to-online training curves on the full OGBench suite. The gray-shaded region denotes the shared offline pre-training phase of the pretrained backbone, while the colored trajectories show method-specific online fine-tuning from the same final pre-training checkpoint.



(a) Varying flow rollout steps.

(b) Varying hidden width per layer.

(c) Varying number of hidden layers.

(d) Varying actor-gradient batch size.

Figure 9: Peak device memory during actor-gradient evaluation for *Adjoint Q-Value Backpropagation* versus *direct BPTT through the full flow rollout*. The benchmark uses the observation and action dimensions of `cube-triple-play-singletask-task3-v0`. The four panels vary (a) flow rollout steps, (b) hidden width per layer, (c) number of hidden layers, and (d) actor-gradient batch size, while keeping the remaining configuration fixed.

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: The abstract and introduction state that the paper proposes MCFS, which combines moment-constrained latent steering with adjoint Q-value backpropagation for RL fine-tuning of flow policies, and these claims are matched by the method and empirical sections. In particular, the theoretical analysis in Sections 3 and the experimental results in Section 4 support the paper's scope, namely improved online adaptation performance together with a more memory-efficient actor-gradient propagation procedure, without claiming broader generalization beyond the evaluated settings.

   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: The limitations are addressed directly in Section C. The discussion identifies two concrete weaknesses of the present method: the latent policy currently relies on a diagonal Gaussian parameterization, which restricts the expressiveness of the prior distribution, and the adjoint computation has not yet incorporated stronger acceleration strategies for the underlying iterative procedure.

   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: The paper states the assumptions for its formal theoretical results and provides complete proofs for them in Section 3, including Proposition 1 and Proposition 2. The remaining theoretical components are presented as standard derivations tied directly to the ODE and adjoint formulations used by the method.

   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: The paper provides the information needed to reproduce the main experimental results. Section 4.1 specifies the offline-to-online adaptation protocol, benchmark suites, comparison baselines, and evaluation metrics, while Appendix A.1 and Appendix A.2 detail the environments, datasets, and training procedure. Appendix A.3 reports the shared pretraining setup, common online hyperparameters, task-specific algorithm settings for all environments, and the use of five random seeds under a shared initialization protocol. The paper also reports the quantitative summaries and training curves used for the main claims in Section 4.2, with full offline-to-online curves and additional quantitative results provided in Appendix A.4.

   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: As described above, we have disclosed the hyperparameters, training protocol, and experimental settings needed to reproduce our results. In addition, we will include an anonymized code archive as supplemental material at submission time. And we will release the full codebase publicly after acceptance.

   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: The paper specifies the overall offline-to-online training protocol, benchmark suite, dataset splits, evaluation metrics, and baseline comparison setup in Section 4.1, with further environment and dataset details in Appendix A.1 and the shared training protocol in Appendix A.2. The detailed optimizer and model hyperparameter choices are referenced to Appendix A.3, which is reserved for the full hyperparameter listing.

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: The paper reports uncertainty information for the experiments supporting the main claims. In particular, the main training curves in Figure 3 and the ablation plots in Figure 4 state that the shaded regions or error bars denote 95% confidence intervals across random seeds. In addition, Table 1 and the supplementary quantitative tables report results as Mean $\pm$ Std over seeds, making clear both the source of variation and the summary statistic used. The paper also states that each method is evaluated with five random seeds in Appendix A.3.

   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: The paper provides the main compute information needed to contextualize and reproduce the experiments. In particular, Appendix A.6 specifies the CPU and GPU hardware used for the reported runs and gives the approximate wall-clock budget for the main training jobs.

   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 confirm that we have reviewed the NeurIPS Code of Ethics and that our research conforms to it in every respect.

   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: The paper discusses both positive and negative societal impacts in Section D. It explains that the method may have positive impact by improving the sample efficiency and reliability of learning-based control systems in robotics and related continuous-control applications. It also notes potential negative impacts, namely that stronger reinforcement-learning-based policy optimization could be misused in unsafe or insufficiently validated autonomous systems or deployed outside the settings in which it was evaluated, and it emphasizes the need for careful testing, monitoring, and deployment in safety-critical use cases.

    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: Our work poses no such risks.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The paper properly credits the existing code assets used in this work and explicitly states their URLs and licenses in Appendix A.7. These assets are cited and used in a manner consistent with their stated terms of use.

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: [Yes]

Justification: The new code asset associated with this paper will be documented and provided alongside the submission. Specifically, we will include an anonymized code archive as supplemental material at submission time. After acceptance, we will release the code publicly in non-anonymized form.

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: Our paper does not involve crowdsourcing nor research with human subjects.

    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: Our paper does not involve crowdsourcing nor research with human subjects.

    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: LLMs do not play an important role in our research. We only use LLMs for writing and other auxiliary purposes.

    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.