Q-LEARNING WITH ADJOINT MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose Q-learning with Adjoint Matching (QAM), a novel TD-based reinforcement learning (RL) algorithm that tackles a long-standing challenge in continuous-action RL: efficient optimization of an expressive diffusion or flow-matching policy with respect to a parameterized value function (*i.e.*, the critic $Q_\phi(s, a)$). Effective optimization requires exploiting the first-order information of the critic (*i.e.*, the action gradient, $\nabla_a Q_\phi(s, a)$), but it is challenging to do so for flow or diffusion policies because direct gradient-based optimization via backpropagation through their multi-step denoising process is numerically unstable. Existing methods work around this either by only using the value and discarding the gradient information, or by relying on approximations that sacrifice policy expressivity or bias the learned policy. QAM sidesteps both of these challenges by leveraging adjoint matching, a recently proposed technique in generative modeling, which transforms the critic's action gradient to form a step-wise objective function that is free from unstable backpropagation, while providing an unbiased, expressive policy at the optimum. Combined with temporal-difference (TD) backup for critic learning, QAM consistently outperforms prior approaches across challenging, sparse reward tasks in both offline and offline-to-online RL settings.

1 INTRODUCTION

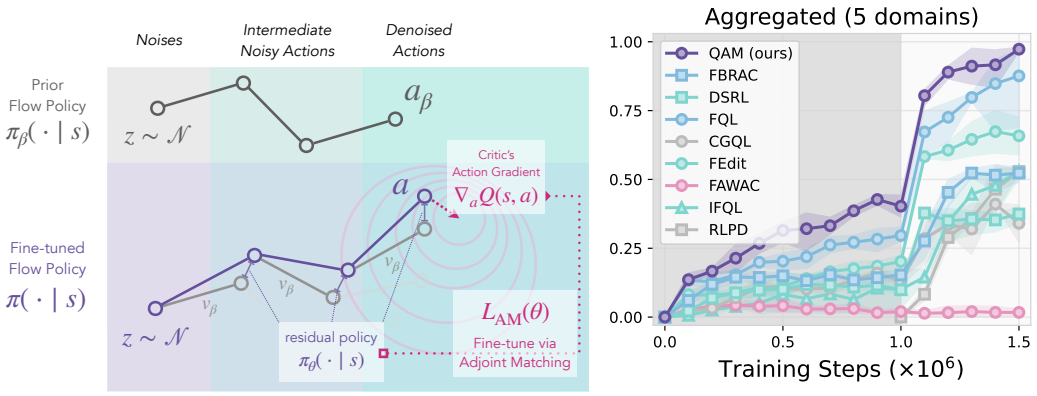

Figure 1: **QAM: Q-learning with Adjoint Matching.** *Left:* QAM uses adjoint matching objective (Domingo-Enrich et al., 2025) that leverages the critic's action gradient directly to fine-tune a residual flow policy such that the combined policy, π converges to the optimal prior-constrained policy: $\pi(\cdot \mid s) \propto \pi_\beta(\cdot \mid s)e^{Q(s,\cdot)}$. *Right:* Aggregated performance of QAM compared to prior methods for offline and offline-to-online RL over 5 OGBench domains. The gray area (first 1M steps) is the offline training and the white area (subsequent 500K steps) is the online training.

A long-standing tension in continuous-action reinforcement learning (RL) especially in the offline/offline-to-online setting is between policy expressivity and optimization tractability with respect to a critic (*i.e.*, $Q(s, a)$). Simple policies, such as single-step Gaussian policies, are easy to train, since they can directly leverage the critic's action gradient (*i.e.*, $\nabla_a Q(s, a)$) via the reparameterization trick (Haarnoja et al., 2018). This optimization tractability, however, often comes at the

cost of expressivity. Some of the most expressive policy classes today, such as flow policies, generate actions through a multi-step denoising process. While this allows flow policies to represent complex, multi-modal action distributions, leveraging the action gradient requires backpropagation through the entire denoising process, which often leads to instability (Park et al., 2025). Prior work has therefore resorted to either (1) discarding the critic's action gradient entirely and only using its value (Ren et al., 2024; Zhang et al., 2025; McAllister et al., 2025), or (2) distilling expressive, multi-step flow policies into one-step noise-conditioned approximations (Park et al., 2025). The former sacrifices learning efficiency and often under-performs methods that use the critic's action gradient (Park et al., 2024b; 2025), while the latter compromises expressivity. This raises a question: can we somehow keep the full expressivity of flow policies while incorporating the critic's action gradient directly into the denoising process without backpropagation instability?

One might be tempted to directly apply the critic's action gradient to intermediate noisy actions within the denoising process, as in diffusion classifier guidance (with the critic function being the classifier) (Dhariwal & Nichol, 2021). Intuitively, this blends two generative process together: one that generates a behavior action distribution, and another that hill-climbs the critic to maximize action value. While this approach bypasses the backpropagation instability and retains full policy expressivity, it relies on the assumption that the critic's gradient at a noisy action is a good proxy for its gradient at the corresponding denoised action. In practice, this assumption often breaks down: when the offline dataset has limited action coverage, the critic is well-trained only on a narrow distribution of noiseless actions, rendering its gradients unreliable for intermediate noisy actions that are out of distribution. As what we will show in our experiments, methods that use the gradients at intermediate noisy actions underperform (CGQL in Section 5, Figure 3).

We propose **Q-learning with Adjoint Matching (QAM)**, a novel RL algorithm that leverages adjoint matching (Domingo-Enrich et al., 2025), a recently developed technique in generative modeling, to effectively use the critic's action gradient for training flow policies to maximize returns subject to a prior constraint (*e.g.*, behavior or entropy constraint) (Figure 1). In general, such a constrained optimization problem on a flow model can be formulated as a stochastic optimal control (SOC) objective, which can be solved by using the continuous adjoint method (Pontryagin et al., 1962). However, this standard formulation has the same loss landscape as directly backpropagating through the SOC objective, causing instability. Instead, we leverage a modified objective from Domingo-Enrich et al. (2025) that admits the same optimal solution, but does not suffer from the instability challenge. At a high level, the critic's gradient at noiseless actions is directly transformed by a flow model constructed from the prior, independent from the possibly ill-conditioned flow model that is being optimized, to construct unbiased gradient estimates for optimizing the state-conditioned velocity field at intermediate denoising steps. This allows the flow policy's velocity field to align directly with the optimal state-conditioned velocity field implied by the critic and the prior, without direct and potentially unstable backpropagation, while preserving the full expressivity of multi-step flow models. By combining this policy extraction procedure with a standard temporal-difference (TD) backup for critic learning, QAM enables the flow policy to efficiently converge to the optimal policy subject to the prior constraint. In contrast, approximation methods that rely on the critic's gradients at noisy intermediate actions lack such convergence guarantees.

Our main contribution is a novel TD-based RL algorithm that leverages adjoint matching to perform policy extraction effectively on a critic function. Unlikely prior Q-learning methods with flow-matching that rely on approximations or throwing away the action gradient of the critic altogether, our algorithm directly uses the gradient to form an objective that at convergence recovers the optimal behavior-regularized policy. We conduct a comprehensive empirical study comparing policy extraction methods for flow/diffusion policies, including recent approaches and new baselines, and show that QAM consistently achieves strong performance across both offline RL and offline-to-online RL benchmarks.

2 Related Work

RL with diffusion and flow policies. Diffusion and flow policies have been explored in both policy gradient methods (Ren et al., 2024) and actor-critic methods (Fang et al., 2025; Kang et al., 2023; Chen et al., 2024c;a; Lu et al., 2023b; Ding et al., 2024b; Wang et al., 2023; He et al., 2023a; Ding & Jin, 2024; Ada et al., 2024; Zhang et al., 2024; Hansen-Estruch et al., 2023). The key challenge

of leveraging diffusion/flow policies in TD-based RL methods is to optimize these policies against the critic function (*i.e.*, $Q(s, a)$). Prior work can be largely put into three categories based on how the value function is used: 1) *post-processing* approaches, where the action distribution from a base diffusion/flow policy is refined with rejection sampling based on the critic value (Hansen-Estruch et al., 2023; Mark et al., 2024; Li et al., 2025; Dong et al., 2025), or using additional gradient steps to hill climb the critic (Mark et al., 2024) (*i.e.*, $a_t \leftarrow a_t + \nabla_a Q(s, a)$). These approaches often reliably improve the quality of extracted policy but at the expense of additional computation during evaluation or even training (*i.e.*, if a similar rejection sampling procedure is used for computing the value backup target (Li et al., 2025; Dong et al., 2025)). Alternatively, one may train a residual policy that modifies a base behavior policy in either the noise space (Singh et al., 2020; Wagenmaker et al., 2025) or in the action space directly (Yuan et al., 2024; Dong et al., 2025) 2) *DDPG-based* approaches perform direct backpropagation through both the critic and the policy (Wang et al., 2023; He et al., 2023b; Ding & Jin, 2023; Zhang et al., 2024; Park et al., 2025; Espinosa-Dice et al., 2025; Chen et al., 2025). While this is the most-straightforward implementation-wise, it requires backpropagation through the diffusion/flow policy's denoising process which has been observed to be unstable Park et al. (2025), or instead learns a distilled policy (Ding & Jin, 2023; Chen et al., 2024b; Park et al., 2025; Espinosa-Dice et al., 2025; Chen et al., 2025), in the expense of policy expressivity. 3) *intermediate fine-tuning* approaches, which our method also belongs to, mitigate the need of the stability/expressivity trade-off in DDPG-base approaches by leveraging the critic to construct an objective that provides direct step-wise supervision to the intermediate denoising process (Psenka et al., 2023; Fang et al., 2025; Ding et al., 2024a; Li et al., 2024b; Frans et al., 2025; Zhang et al., 2025; Ma et al., 2025; Koirala & Fleming, 2025). While these approaches remove the need for backpropagation through the denoising process completely, the challenge lies in carefully crafting the step-wise objective that does not introduce additional biases and learning instability.Compared to prior methods that either rely on approximations (Lu et al., 2023a; Fang et al., 2025) that do not provide theoretical guarantees (see more discussions in Appendix A) or directly throwing away the critic's action gradient (and use its value instead) (Ding et al., 2024a; Zhang et al., 2025; Ma et al., 2025; Koirala & Fleming, 2025), we leverage adjoint matching (Domingo-Enrich et al., 2025) which allows us to use the critic's action gradient directly to construct an direct step-wise objective for our flow policy that recovers the optimal prior regularized policy at the optimum of the objective.

Offline-to-online reinforcement learning methods focus on leveraging offline RL to first pretrain on an offline dataset, and then use the pretrained policy and value function(s) as initialization to accelerate online RL (Xie et al., 2021; Song et al., 2023; Lee et al., 2022; Agarwal et al., 2022; Zhang et al., 2023; Zheng et al., 2023; Ball et al., 2023; Nakamoto et al., 2024; Li et al., 2024a; Wilcoxson et al., 2024; Zhou et al., 2025). While it is possible to skip the offline pre-training phase altogether and use online RL methods directly by treating the offline dataset as additional off-policy data that is pre-loaded into the replay buffer (Lee et al., 2022; Song et al., 2023; Ball et al., 2023), it is often more convenient to directly use the same offline RL objective for both offline pre-training and online fine-tuning (Kostrikov et al., 2021; Fujimoto & Gu, 2021; Tarasov et al., 2023; Park et al., 2025) directly. Our method also operates in this regime where the same RL objective is used for both offline pre-training and online fine-tuning. While we focus on evaluating our method in the offline-to-online RL setting, the idea of using adjoint matching to train an expressive flow policy can be applied to other settings such as online RL.

Diffusion and flow-matching with guidance. Diffusion and flow-matching models have been used for generating data with different modalities ranging from images (Rombach et al., 2022), videos (Ho et al., 2022), and text (Lou et al., 2023). In most applications, the generative models trained on large-scale unlabeled data do not provide high-quality samples when conditioned on some context (*e.g.*, language description), and a common practice is to augment the sampling process with classifier guidance/classifier-free guidance (Dhariwal & Nichol, 2021; Ho & Salimans, 2022), with the goal of aligning the sampling distribution better with the posterior distribution conditioned on the context. However, most of the guidance methods suffer from a bias problem that is tricky to tackle, stemming from the fact that simply adding or interpolating two diffusion/flow-matching sampling processes (*i.e.*, $v^1(\cdot, t) + v^2(\cdot, t)$ for flow or $\log p_t^1 + \log p_t^2$ for diffusion) does not lead to the correct composite distribution (*i.e.*, $\propto \pi_1 \pi_2$) distribution in general (Du et al., 2023; Bradley & Nakkiran, 2024). One solution is to use Langevin dynamics sampling approaches (Song & Ermon, 2019) where only the score function for the noise-free distribution is required, but they have been known to under-perform diffusion/flow models due to the challenge of accurately estimating the score functions in low-density regions (Song & Ermon, 2020). Since then, a line of work has proposed solutions to generate

the correct composite distribution. Du et al. (2023), Phillips et al. (2024), Thornton et al. (2025), Singhal et al. (2025) and Skreta et al. (2025) propose to use Sequential Monte Carlo (SMC) that uses resampling procedures to leverage additional test-time compute to correct such bias. Rather than correcting the distribution at test-time, Domingo-Enrich et al. (2025) and Havens et al. (2025) take a different perspective by formulating it as a stochastic optimal control (SOC) objective that can be efficiently optimized as a fine-tuning process while providing guarantee that the model converges to the correct distribution at the optimum. The flow/diffusion policy optimization problem in actor-critic RL methods shares a similarity to the aforementioned classifier/classifier-free guidance problem in generative modeling literature where the critic function serves as the guidance to the generative policy model. This allows our method builds directly on top of the algorithm developed by Domingo-Enrich et al. (2025) while enjoying the guarantee that our policy converges to the optimal prior regularized solution (*i.e.*, $\pi \propto \pi_\beta \exp(Q(s,a))$).

3 PRELIMINARIES

Reinforcement learning and problem setup. We consider a Markov Decision Process (MDP), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma, R, \mu)$, where $\mathcal{S}$ is the state space, $\mathcal{A} = \mathbb{R}^A$ ($A \in \mathbb{Z}^+$) is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ is the transition function, $\gamma \in [0, 1)$ is the discount factor, $R : \mathcal{S} \times \mathbb{R}^A \to \mathbb{R}$ is the reward function, and $\mu \in \Delta_{\mathcal{A}}$ is the initial state distribution. We have access to a dataset $\mathcal{D}$ consisting of a set of transitions $\{(s_i, a_i, s'_i, r_i)\}_{i=1}^{|\mathcal{D}|}$, where $s' \sim P(\cdot \mid s, a)$ and $r = R(s, a)$. Our first goal (*offline RL*) is to learn a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ from $\mathcal{D}$ that maximizes its expected discounted return,

$$\eta_\pi = \mathbb{E}_{s_0 \sim \mu, s_{t+1} \sim P(\cdot|s_t,a_t), a_t \sim \pi(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]. \tag{1}$$

The second goal (*offline-to-online RL*) is to fine-tune the offline pre-trained policy π_θ by continuously interacting with the MDP through trajectory episodes with a task/environment dependent maximum episode length of H (*i.e.*, the maximum number of time steps before the agent is reset to μ). The central challenge of offline-to-online RL is to maximally leverage the behavior prior π_β in $\mathcal{D}$ to learn as sample-efficiently as possible online.

Flow-matching generative model. A flow model uses a time-variant velocity field $v : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ to estimate the marginal distribution of a denoising process from noise, $X_0 = \mathcal{N}(0, I_d)$, to data, $X_1 = \mathcal{D}$, at each intermediate time $t \in [0, 1]$:

$$X_t = (1 - t)X_0 + tX_1. \tag{2}$$

In particular, the flow model approximates the intermediate X_t via an ordinary differential equation (ODE) starting from the noise: $X^0 = \mathcal{N}$:

$$\mathrm{d}\hat{X}_t = v(\hat{X}_t, t)\mathrm{d}t. \tag{3}$$

Flow models are typically trained with a *flow matching* objective (Liu et al., 2022):

$$L_{\mathrm{fm}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}[0,1], x_0 \sim \mathcal{N}, x_1 \sim \mathcal{D}} \left[\|v_\theta((1 - t)x_0 + tx_1, t) - x_1 + x_0\|_2^2 \right], \tag{4}$$

where any optimal velocity field, $v_{\theta^\star}$, results in $\hat{X}_t$ where its marginal distribution $p_v(x_t)$ exactly recovers the marginal distribution of the original denoising process X_t, $p_{\mathcal{D}}(x_t)$, for each $t \in [0, 1]$ (lip, 2024). Furthermore, one may use the Fokker-Planck equations to construct a family of stochastic differential equations (SDE) that admits the same marginals as well:

$$\mathrm{d}\hat{X}_t = \left(v(\hat{X}_t, t) + \frac{\sigma_t^2 t}{2(1 - t)} \left(v(\hat{X}_t, t) + X_t/t \right) \right) \mathrm{d}t + \sigma_t \mathrm{d}B_t \tag{5}$$

with B_t being a Brownian motion and $\sigma_t > 0$ being any noise schedule.

Adjoint matching is a technique developed by Domingo-Enrich et al. (2025) with the goal of modifying a base flow generative model v_β such that it generates the following *tilt* distribution:

$$p^\star(x_1) \propto p_\beta(x_1) e^{r(x_1)} \tag{6}$$

where $r : \mathbb{R}^d \to \mathbb{R}$ is any tilt function that up-weights or down-weights the probability of each example in the domain $\mathbb{R}^d$. Domingo-Enrich et al. (2025) uses a marginal-preserving SDE with a 'memoryless' noise schedule (*i.e.*, X_0 and X_1 are independent), $\sigma_t = \sqrt{2(1-t)/t}$:

$$\mathrm{d}X^t = (2v(X_t, t) - X_t/t)\,\mathrm{d}t + \sqrt{2(1-t)/t}\mathrm{d}B_t, \tag{7}$$

because solving the following stochastic optimal control equation (with X^t sampling from the joint distribution defined by the SDE in Equation (7)),

$$L(\theta) = \mathbb{E}_{\boldsymbol{X}=\{X_t\}_t}\left[\int_0^1 \left(\frac{1}{2}\|v_\theta(X_t,t) - v_\beta(X_t,t)\|_2^2\right) + r(X_1)\right] \tag{8}$$

gives the correct marginal *tilt* distribution for X^1:

$$p(X_1) \propto p_\beta(X_1)e^{r(X_1)}. \tag{9}$$

Let the adjoint state be the gradient of the tilt function applied at the denoised X_1:

$$g(\boldsymbol{X}, t) = \nabla_{X_t}\left[\int_t^1 \|v_\theta(X_{t'}, t') - v_\beta(X_t, t)\|_2^2 \mathrm{d}t' + r(X_1)\right], \tag{10}$$

which satisfies the following ODE:

$$\mathrm{d}g(\boldsymbol{X}, t) = -g(\boldsymbol{X}, t)^\top \nabla_{X_t}\left[2v_\theta(X_t, t) - X_t/t\right] + \nabla_{X_t}\|v_\theta(X_t,t) - v_\beta(X_t,t)\|_2^2)/(2\sigma_t^2)\mathrm{d}t \tag{11}$$

with the boundary condition $a(\boldsymbol{X}, 1) = -\nabla_{X_1} r(X_1)$. We can compute the adjoint states by stepping through the reverse ODE (which can be effcieintly computed with the Jacobian-vector product (JVP) in most modern deep learning frameworks). Then, it can be shown that it equivalently optimizes the 'basic' adjoint matching objective below:

$$L_{\mathrm{BAM}}(\theta) = \mathbb{E}_{\boldsymbol{X}}\left[\int_0^1 \|2(v_\theta(X_t,t) - v_\beta(X_t,t))/\sigma_t + \sigma_t g(\boldsymbol{X}, t)\|_2^2 \mathrm{d}t\right]. \tag{12}$$

The optimal v_θ coincides with the optimal solution in the original SOC equation (Equation (8)), which gives the correct marginal distribution of X^1 as a result. However, the objective is equivalent to the objective used in the continuous adjoint method (Pontryagin et al., 1962) with its gradient equivalent to that of backpropagation through the denoising process.

Instead, Domingo-Enrich et al. (2025) derive the 'lean' adjoint state where all the terms in the adjoint state that are zero at the optimum are removed from the state. The 'lean' adjoint state satisfies the following ODE:

$$\mathrm{d}\tilde{g}(\boldsymbol{X}, t) = -\tilde{g}(\boldsymbol{X}, t)^\top \nabla_{X_t}\left[2v_\beta(X_t, t) - X_t/t\right]\mathrm{d}t, \tag{13}$$

with the same boundary condition $\tilde{a}(\mathbf{X}, 1) = -\nabla_{X_1} r(X_1)$.

Note that computing the 'lean' adjoint state only requires the base flow model $v_\beta(X_t, t)$ and no longer needs to use $v_\theta(X_t, t)$ as needed in either the basic adjoint matching objective (Equation (12)) or naive backpropagation through the denoising process. The resulting adjoint matching objective is

$$L_{\mathrm{AM}}(\theta) = \mathbb{E}_{\boldsymbol{X}}\left[\int_0^1 \|2(v_\theta(X_t,t) - v_\beta(X_t,t))/\sigma_t + \sigma_t \tilde{g}(\boldsymbol{X}, t)\|_2^2 \mathrm{d}t\right], \tag{14}$$

where again $\boldsymbol{X}$ is sampled from the marginal preserving SDE in Equation (7). Because the terms omitted in the 'lean' adjoint state are zero at the optimum, and thus do not change optimal solution for v_θ. Thus, the optimal solution for the adjoint matching gives the correct tilt distribution.

4 Q-LEARNING WITH ADJOINT MATCHING (QAM)

In this section, we describe in details how our method leverages adjoint matching to directly align the flow policy to prior regularized optimal policy without suffering from backpropagation instability.

To start with, we first define the optimal policy that we want to learn as the solution of the best policy the under the standard KL behavior constraint:

$$\arg\max_\pi \mathbb{E}_{a\sim\pi(\cdot|s)}[Q(s,a)] \quad \text{s.t.} \quad D_{\mathrm{KL}}(\pi_\beta \parallel \pi) \le \epsilon(s). \tag{15}$$

or equivalently, for an appropriate $\tau(s)$,

$$\pi^\star(\cdot \mid s) \propto \pi_\beta(\cdot \mid s) e^{\tau(s) Q_\phi(s,a)} \tag{16}$$

where $\tau : \mathcal{S} \to \mathbb{R}^+$ is the inverse temperature coefficient that controls the strength of the behavior constraint at each state.

We approximate the behavior policy using a flow-matching behavior policy, $v_\beta : \mathcal{S} \times \mathbb{R}^A \times [0,1] \to \mathbb{R}^A$ that is optimized with the standard flow-matching objective:

$$L_{\mathrm{FM}}(\beta) = \mathbb{E}_{(s,a)\sim\mathcal{D}, u\sim[0,1], z\sim\mathcal{N}} \left[\| v_\beta(s, (1-u)z + uz, t) - a + z \|_2^2 \right] \tag{17}$$

We then parameterize our approximation of the optimal policy as a sum of the behavior flow model v_β and a residual flow model $v_\theta : \mathcal{S} \times \mathbb{R}^A \times [0,1] \to \mathbb{R}^A$ and solve the following SOC equation:

$$L(\theta) = \mathbb{E}_{s\sim\mathcal{D}, a^u} \left[\int_0^1 \| v_\theta(s, a^u, t) \|_2^2 + \tau(s) Q_\phi(s, a^1) \mathrm{d}u \right], \tag{18}$$

where a^u is defined by the following 'memoryless' SDE (*e.g.*, a^0 is independent from a^1):

$$\mathrm{d}a^u = (2v_\theta(s, a^u, u) + 2v_\beta(s, a^u, u) - a^u/u)\mathrm{d}u + \sqrt{2(1-u)/u}\mathrm{d}B_u. \tag{19}$$

Similarly to the derivation by Domingo-Enrich et al. (2025), the memoryless property allows us to directly conclude that the SOC equation has the optimum at

$$\pi_\theta(\cdot \mid s) \propto \pi_\beta(\cdot \mid s) e^{\tau(s) Q_\phi(s,a)} \tag{20}$$

where $\pi_\theta(\cdot \mid s)$ and $\pi_\beta(\cdot \mid s)$ are the corresponding action distributions defined by $v_\theta + v_\beta$ and v_β.

However, directly solving the SOC equation involves backpropagation through time that introduces additional stability. To circumvent this issue, we use the adjoint matching objective proposed by Domingo-Enrich et al. (2025) (Equation (14)) to construct a similar objective for policy optimization in our case:

$$L_{\mathrm{AM}}(\theta) = \mathbb{E}_{s\sim\mathcal{D}, \{a^u\}_u} \left[\int_0^1 \| 2v_\theta(s, a^u, u)/\sigma_u + \sigma_u \tilde{g}^u \|_2^2 \mathrm{d}u \right] \tag{21}$$

where $\tilde{g}^u$ is the 'lean' adjoint state defined by a reverse ODE constructed from a^u defined by the forward SDE:

$$\mathrm{d}\tilde{g}^u = -\tilde{g}^{u\top} \nabla_{a^u} \left[2v_\beta(s, a^u, u) - a^u/u \right] \mathrm{d}u. \tag{22}$$

Unlike the original SOC objective (Equation (18)) from which calculating the gradient requires backpropagating through an SDE, which suffers from stability challenges, the adjoint matching objective is constructed without backpropagation. Instead, it uses the behavior velocity field v_β to calculate the 'lean' adjoint states $\{\tilde{g}^u\}_u$ through a series of JVPs for every SDE trajectory $\{a^u\}_u$, which are then used to form a squared loss in the adjoint matching objective. Mathematically, backpropagation can also be interpreted as calculating the adjoint states through a series of JVPs, with the key distinction that the JVPs are computed under the flow model that is being optimized (*i.e.*, $v_\theta + v_\beta$). This is an important distinction because for direct backpropagation, any ill-conditioned action gradient in v_θ (*i.e.*, $\nabla_a v_\theta(s, a, t)$) would compound over the entire denoising process, contributing to the 'ill-condition-ness' of the overall gradient to the parameter θ, which can in turn destabilize the whole optimization process. In contrast, in adjoint matching, action gradient of v_θ has no contribution to the overall gradient to θ, which allows the optimization to be much more stable.

Finally, we combine the policy optimization with the standard critic learning objective in TD-based RL algorithms:

$$L(\phi) = \mathbb{E}_{s,a,s',r\sim\mathcal{D}} \left[(Q(s,a) - r - \gamma Q_{\bar{\phi}}(s', a') \right], \quad a' \leftarrow \mathrm{ODE}(v(s,\cdot,\cdot), a^0 \sim \mathcal{N}) \tag{23}$$

where $v(s,\cdot,\cdot) = v_\beta(s,\cdot,\cdot) + v_\theta(s,\cdot,\cdot)$ is the summation of the behavior velocity field and the residual velocity field and $\bar{\phi}$ is the exponential moving average of ϕ with a time-constant of $\lambda = 0.005$ (*i.e.*, $\bar{\phi}_{i+1} \leftarrow (1-\lambda)\bar{\phi}_i + \lambda\phi_i$ for each training step i).

Practical considerations. In practice, following Domingo-Enrich et al. (2025), we solve both the SDE and the reverse ODE with discrete approximation and a fixed step size of $h = 1/T$, where T is

Algorithm 1 Learning procedure in QAM.

Input: (s, a, s', r): off-policy transition tuple, v_β: behavior velocity field, v_θ residual velocity field, Q_ϕ: critic function.
$v(s, \cdot) \leftarrow v_\theta(s, \cdot, \cdot) + v_\beta(s, \cdot, \cdot)$
$\boldsymbol{a} = \{a^0, a^h, \cdots, a^1\} \leftarrow \mathrm{SDE}_{\mathrm{am}}(v(s, \cdot, \cdot))$ ▷ *Memoryless SDE (Equation (24))*
$\tilde{g}^1 \leftarrow -\mathrm{Clip}[\tau \nabla_{a^1} Q_\phi(s, a^1)]_{-c}^{c}$ ▷ *Computing the critic's action gradient*
$\tilde{g}^0, \tilde{g}^h, \cdots, \tilde{g}^{1-h} \leftarrow \mathrm{LeanAdj}_{\mathrm{am}}(v_\beta(s, \cdot, \cdot), \tilde{g}^1, \boldsymbol{a})$ ▷ *Lean adjoint states (Equation (25))*
Optimize θ w.r.t $L(\theta) = \sum_u \|2v_\theta(s, a^u, u)/\sigma_u + \sigma_u \tilde{g}^u\|_2^2$ ▷ *Adjoint matching (Equation (21))*
$a' \leftarrow \mathrm{ODE}(v(s, \cdot, \cdot), z \sim \mathcal{N}(0, I_A))$
Optimize ϕ w.r.t $L(\phi) = (Q_\phi(s, a) - r - \gamma Q_{\bar{\phi}}(s', a'))^2$
Output: v_θ, Q_ϕ

the number of discretization steps. In particular, with $a^0 \sim \mathcal{N}$ and $z^u \sim \mathcal{N}, \forall u \in \{0, h, \cdots (T-1)h\}$, the forward SDE process is approximated by

$$a^{u+h} \leftarrow a^u + h \cdot (2v_\theta(s, a^u, u) + 2v_\beta(s, a^u, u) - a^u/u) + \sqrt{2h(1-u)/u} z^u. \tag{24}$$

We set the boundary condition as $\tilde{g}^1 = -\mathrm{Clip}[\tau \nabla_{a^1} Q_\phi(s, a^1)]_{-c}^{c}$, where we use a state *independent* inverse temperature coefficient τ to modulate the influence of the prior π_β and we additionally clip the magnitude of the action gradient element-wise by c for numerical stability. The backward adjoint state calculation process is then approximated by

$$\tilde{g}^{u-h} \leftarrow \tilde{g}^u + h \cdot \mathrm{JVP}(\nabla_{a^u}(2v_\beta(s, a^u, u) - a^u/u), \tilde{g}^u), \tag{25}$$

with $\mathrm{JVP}(\nabla_y f(y), x) = x^\top \nabla_y f(y)$ being the Jacobian-vector product and it can be practically implemented by carrying the 'gradient' x with backpropagation through f. For the critic, we use an ensemble of $K = 10$ critic functions $\phi^1, \cdots, \phi^K$ and use the pessimistic target value backup with a coefficient of $\rho = 0.5$ (Ghasemipour et al., 2022). The loss function for each $\phi^j, j \in \{1, 2, \cdots, K\}$ is

$$L(\phi^j) = \left(Q_{\phi^j}(s, a) - r - \gamma \left[\bar{Q}_{\mathrm{mean}}(s', a') - \rho \bar{Q}_{\mathrm{std}}(s', a')\right]\right)^2, \tag{26}$$

where $\bar{Q}_{\mathrm{mean}}(s', a') := \frac{1}{K} \sum_k Q_{\bar{\phi}^k}(s', a'), \bar{Q}_{\mathrm{std}}(s', a') = \sqrt{\sum_k (Q_{\bar{\phi}^k}(s', a') - \bar{Q}_{\mathrm{mean}}(s', a'))^2}$, and a' is the action sampled from the combined flow model: $v(s', \cdot, \cdot) = v_\beta(s, \cdot, \cdot) + v_\theta(s, \cdot, \cdot)$. For all our experiments, we do not use a separate training process for v_β and instead training it at the same to as v_θ and Q_ϕ, following Park et al. (2025); Li et al. (2025), using the standard flow-matching objective described in Equation (17). For all our loss functions, the transition tuple (s, a, s', r) is drawn from $\mathcal{D}$ uniformly. During offline training, $\mathcal{D}$ is the offline data. During online fine-tuning, $\mathcal{D}$ is combination of the offline and online replay buffer data without any re-weighting.

5 Experiments

We conduct experiments to evaluate the effectiveness of our method on a range of long-horizon, sparse-reward domains and compare it against a set of representative baselines.

Domains and datasets. We consider five domains from OGBench (Park et al., 2024a), `cube-double`, `cube-triple`, `cube-quadruple`, `antmaze-large`, `antmaze-giant`. For `antmaze-*`, we use the default `navigate` datasets. For `cube-*`, we use the default `play` datasets except for `cube-quadruple` where we use a large 100M-size dataset, following Li et al. (2025). All of these domains require the RL agent to solve long-horizon tasks from diverse offline behavior data that can only be accurately captured by expressive policies like flow/diffusion policies. Furthermore, the harder domains (Figure 2) are difficult to solve from offline data alone, making these benchmarks great for evaluating the online fine-tuning effectiveness of our approach. In addition, for all `cube-*` domains, we follow Li et al. (2025) to learn action chunking policies with an action chunk size of $h = 5$. Action chunking policies output high-dimensional actions that exhibits a much more complex behavior distribution, where the policy extraction becomes critical. Since our approach primarily focuses on the policy extraction aspect, these domains make an ideal testbed us to compare our method to prior work. See Appendix B for more details on these domains.

Comparisons. To provide a comprehensive empirical evaluation of our method, we carefully select 8 representative, strong baselines that can be roughly categorized into the following 5 categories

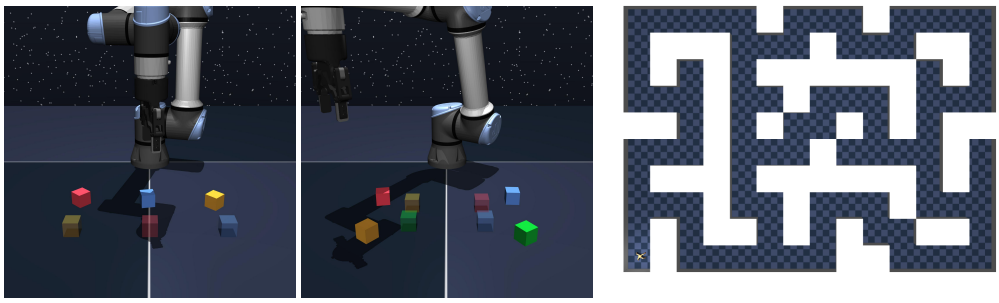

Figure 2: **OGBench domains**. In this paper, we primarily focus on evaluating our method on some of the hardest domains on OGBench (Park et al., 2024a). `cube-{triple, quadruple}` (left) requires a robot arm to manipulate up to 3/4 cubes from an initial arrangement to a goal arrangement. `antmaze-giant` (right) requires an ant robot agent to navigate from one location to another location. All of these domains are long-horizon by design and are difficult to solve from offline data alone. The offline dataset in these domains contain diverse multi-modal behaviors that can only be accurately captured by expressive generative models like flow/diffusion policies.

Task	IFQL	FAWAC	CGQL	FEdit	FQL	DSRL	FBRAC	QAM (Ours)
`antmaze-large`	$41_{[13,62]}$	$10_{[6,17]}$	$25_{[21,30]}$	$73_{[64,85]}$	$\mathbf{89}_{[\mathbf{86,92}]}$	$0_{[0,2]}$	$76_{[70,83]}$	$\mathbf{72}_{[\mathbf{43,94}]}$
`antmaze-giant`	$0_{[0,0]}$	$0_{[0,0]}$	$4_{[0,12]}$	$0_{[0,0]}$	$0_{[0,0]}$	$0_{[0,0]}$	$0_{[0,0]}$	$5_{[0,15]}$
`cube-double`	$12_{[8,16]}$	$0_{[0,0]}$	$41_{[35,46]}$	$27_{[16,39]}$	$49_{[41,56]}$	$49_{[40,60]}$	$0_{[0,0]}$	$\mathbf{84}_{[\mathbf{77,91}]}$
`cube-triple`	$0_{[0,0]}$	$0_{[0,0]}$	$0_{[0,0]}$	$0_{[0,0]}$	$0_{[0,0]}$	$0_{[0,0]}$	$0_{[0,0]}$	$2_{[0,4]}$
`cube-quadruple-100M`	$0_{[0,0]}$	$0_{[0,0]}$	$0_{[0,0]}$	$0_{[0,0]}$	$\mathbf{10}_{[\mathbf{0,32}]}$	$0_{[0,2]}$	$0_{[0,0]}$	$\mathbf{39}_{[\mathbf{27,57}]}$

Table 1: **Offline RL at 1M training steps.** Our method (QAM) outperforms prior methods `cube-double`, and achieves non-trivial success rate on both `antmaze-giant` and `cube-triple` which none of the prior method could achieve. We report the means and the 95% bootstrapped confidence intervals are computed over 4 seeds.

— **(1) DDPG-based:** FBRAC (Park et al., 2025) (backpropagation through flow denoising step directly), FQL (1-step distillation) (Park et al., 2025); **(2) Using critic's value only:** FAWAC (Park et al., 2025) (AWAC (Nair et al., 2020) with flow policy); **(3) Using critic's action gradient with approximations:** CGQL (a novel baseline that is based on classifier guidance); **(4) Post-processing-based:** DSRL (Wagenmaker et al., 2025), FEdit (flow + Gaussian edit policy from Dong et al. (2025)), and IFQL (flow counterpart of IDQL (Hansen-Estruch et al., 2023)); **(5) Gaussian:** RLPD (Ball et al., 2023). Among them, RLPD does not employ any behavior constraint, so we directly train them from scratch online with 50/50 offline/online sampling (*i.e.*, half of the training batch comes from offline and half of the training batch comes from the online replay buffer). To make the comparison fair, we use $K = 10$ critic networks, pessimistic value backup with $\rho = 0.5$, no best-of-N sampling (*i.e.*, $N = 1$) for both our method and all our baselines except for IFQL where best-of-N is used for policy extraction. We refer the reader to Appendix C for detailed description and implementation detail for each of the baselines. We also include the domain-specific hyperparameters for each baseline in Appendix D.

6 Results

In this section, we present our experimental results to answer the following three questions:

(Q1) How effective is our method for offline RL?

Table 1 reports the offline RL performance across five different domains. QAM performs on par or better than all prior methods. In particular, QAM is able to achieve non-trivial success rates on both `cube-triple` and `antmaze-giant` offline (not achievable by any prior methods).

(Q2) How effective is our method for offline-to-online fine-tuning?

Next, we evaluate QAM's ability to online fine-tune from its offline RL initialization. Figure 3 shows the sample efficiency curve (with x-axis being the number of environment steps) where QAM either performs on par or outperforms all baselines. In particular, on `cube-triple`, none of the baselines other than FQL is able to achieve non-zero success rate throughout online training.

(Q3) How sensitive is our method to hyperparameters?

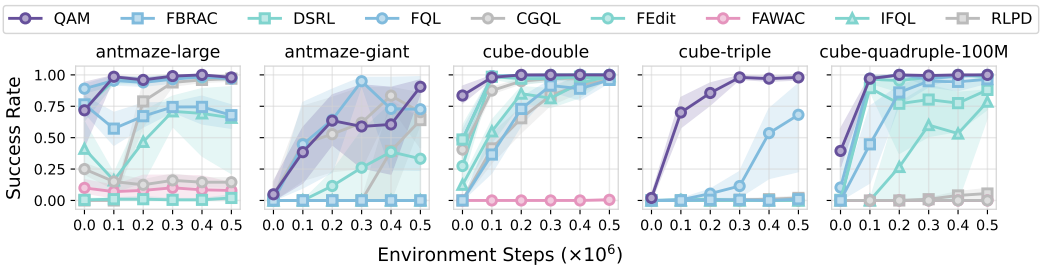

Figure 3: **QAM online fine-tunes more effectively than prior methods.** Our method (QAM) performs on par of outperforms prior methods. On `cube-triple`, QAM solves the task in around 200K-300K environment steps whereas none of the baselines other than FQL is able to achieve non-zero success rate.

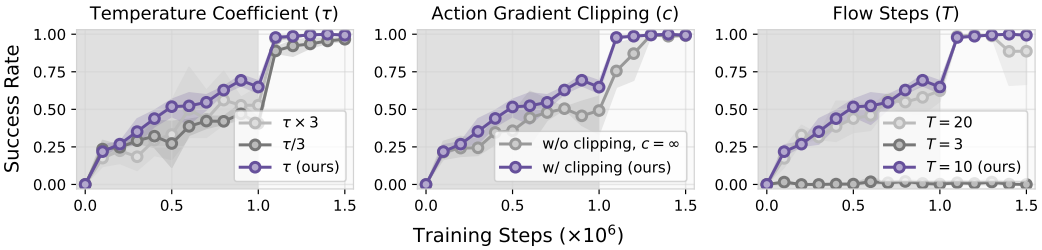

Figure 4: **Sensitivity analysis.** We report the success rate aggregated over three domains: `antmaze-large`, `cube-double` and `cube-quadruple`. *Action gradient clipping (c)*: this parameter controls the clipping range of the action gradient, $\tau\nabla_a Q(s,a)$. The clipped gradient is then used as the boundary/initial condition for the lean adjoint calculation (Equation (25)); *Temperature coefficient (τ)*: this parameter controls how strong the prior is, where a higher τ leads to a weaker prior and a lower τ leads to a stronger prior; *Flow steps (T)*: this parameter indicates the number of numerical integration steps that we use for the flow model. Gray: offline; White: online.

Finally, we perform a sensitivity analysis for three main hyperparameters in our method with the result reported in Figure 4. The first parameter, τ, is the most important hyperparameter that controls how closely/conservative the learned policy should be following the behavior policy. Different parameter value directly contributes to the difference in performance. The second parameter, c, is mainly for numerical stability. In practice, the 'lean' adjoint state computed in the adjoint matching objective can blow up in value if the initial boundary condition (*i.e.*, the action gradient of the critic) is too large. Clipping the boundary condition often leads to better learning stability. Finally, the number of flow steps also affects the performance. Having too few steps often destroys the expressivity of flow policy entirely. Setting $T > 10$ (*e.g.*, $T = 20$) does not lead to better performance.

7 Discussion

We present Q-learning with Adjoint Matching (QAM), a novel TD-based RL method that effectively leverages the critic's action gradient to extract an optimal prior-constrained policy while circumventing common limitations of prior approaches (*e.g.*, approximations that do not guarantee to converge to the desired optimal solution, learning instability, or reduced expressivity from distillation). Our empirical results suggest that QAM is an effective policy extraction method in both the offline RL setting and the offline-to-online RL setting, performing on par or better than prior methods. There are still practical challenges associated with QAM. While QAM's effectiveness can be largely attributed to how well it is able to leverage the critic's action gradient, this can be a double-edge sword—for cases where the critic function is ill-conditioned, it could lead to optimization stability issue. Gradient clipping (as done in our method) can alleviate this issue, but a more principled method that combines both value and gradient information could further improve robustness and performance. Another possible extension is to apply QAM in real-world robotic settings with action chunking policies. Our initial success (especially in the manipulation domains where we also leverage action chunking policies) may suggest that our method might work more effectively in complex real-world scenarios.

Reproducibility Statement

We include our source code as part of the supplementary materials (including installation instructions and example scripts for running our method and all our baselines). We describe our evaluation domains in Appendix B, hyperparameters in Appendix D, and implementation details for each of our baselines in Appendix C.

References

Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Suzan Ece Ada, Erhan Oztop, and Emre Ugur. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *IEEE Robotics and Automation Letters (RA-L)*, 9:3116–3123, 2024.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 28955–28971. Curran Associates, Inc., 2022.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Arwen Bradley and Preetum Nakkiran. Classifier-free guidance is a predictor-corrector. *arXiv preprint arXiv:2408.09000*, 2024.

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

Tianyi Chen, Haitong Ma, Na Li, Kai Wang, and Bo Dai. One-step flow policy mirror descent. *arXiv preprint arXiv:2507.23675*, 2025.

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, 2024b.

Tianyu Chen, Zhendong Wang, and Mingyuan Zhou. Diffusion policies creating a trust region for offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2024c.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024a.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. In *Neural Information Processing Systems (NeurIPS)*, 2024b.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. *arXiv preprint arXiv:2309.16984*, 2023.

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

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xQBRrtQM8u.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies. *arXiv preprint arXiv:2507.07986*, 2025.

Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International conference on machine learning*, pp. 8489–8510. PMLR, 2023.

Nicolas Espinosa-Dice, Yiyi Zhang, Yiding Chen, Bradley Guo, Owen Oertell, Gokul Swamy, Kiante Brantley, and Wen Sun. Scaling offline rl via efficient and expressive shortcut models. *arXiv preprint arXiv:2505.22866*, 2025.

Linjiajie Fang, Ruoxue Liu, Jing Zhang, Wenjia Wang, and Bingyi Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ldVkAOO9Km.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *arXiv preprint arXiv:2505.23458*, 2025.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Kamyar Ghasemipour, Shixiang Shane Gu, and Ofir Nachum. Why so pessimistic? estimating uncertainties for offline RL through ensembles, and why their independence matters. *Advances in Neural Information Processing Systems*, 35:18267–18281, 2022.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

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.

Aaron Havens, Benjamin Kurt Miller, Bing Yan, Carles Domingo-Enrich, Anuroop Sriram, Brandon Wood, Daniel Levine, Bin Hu, Brandon Amos, Brian Karrer, et al. Adjoint sampling: Highly scalable diffusion samplers via adjoint matching. *arXiv preprint arXiv:2504.11713*, 2025.

Longxiang He, Li Shen, Linrui Zhang, Junbo Tan, and Xueqian Wang. Diffcps: Diffusion model based constrained policy search for offline reinforcement learning. *ArXiv*, abs/2310.05333, 2023a.

Longxiang He, Li Shen, Linrui Zhang, Junbo Tan, and Xueqian Wang. Diffcps: Diffusion model based constrained policy search for offline reinforcement learning. *arXiv preprint arXiv:2310.05333*, 2023b.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Prajwal Koirala and Cody Fleming. Flow-based single-step completion for efficient and expressive policy learning. *arXiv preprint arXiv:2506.21427*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic Q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022.

Qiyang Li, Jason Zhang, Dibya Ghosh, Amy Zhang, and Sergey Levine. Accelerating exploration with unlabeled prior data. *Advances in Neural Information Processing Systems*, 36, 2024a.

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

Steven Li, Rickmer Krohn, Tao Chen, Anurag Ajay, Pulkit Agrawal, and Georgia Chalvatzaki. Learning multimodal behaviors from scratch with diffusion policy gradient. *Advances in Neural Information Processing Systems*, 37:38456–38479, 2024b.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pp. 22825–22855. PMLR, 2023a.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023b.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. *arXiv preprint arXiv:2502.00361*, 2025.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024.

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.

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

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. *ArXiv*, 2024a.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline RL? *Advances in Neural Information Processing Systems*, 37:79029–79056, 2024b.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

Angus Phillips, Hai-Dang Dau, Michael John Hutchinson, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. Particle denoising diffusion sampler. *arXiv preprint arXiv:2402.06320*, 2024.

Lev Semenovich Pontryagin, V G Boltyanskii, R V Gamkrelidze, and E F Mishchenko. *The Mathematical Theory of Optimal Processes*. Wiley, 1962.

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.

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.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025.

Marta Skreta, Tara Akhound-Sadegh, Viktor Ohanesian, Roberto Bondesan, Alán Aspuru-Guzik, Arnaud Doucet, Rob Brekelmans, Alexander Tong, and Kirill Neklyudov. Feynman-kac correctors in diffusion: Annealing, guidance, and product of experts. *arXiv preprint arXiv:2503.02819*, 2025.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

Yuda Song, Yifei Zhou, Ayush Sekhari, Drew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid RL: Using both offline and online data can make RL efficient. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=yyBis80iUuU.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vqGWslLeEw.

James Thornton, Louis Béthune, Ruixiang Zhang, Arwen Bradley, Preetum Nakkiran, and Shuangfei Zhai. Composition and control with distilled energy diffusion models and sequential monte carlo. *arXiv preprint arXiv:2502.12786*, 2025.

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.

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

Max Wilcoxson, Qiyang Li, Kevin Frans, and Sergey Levine. Leveraging skills from unlabeled prior data for efficient online exploration. In *Arxiv*, 2024. URL https://arxiv.org/abs/2410.18076.

Tengyang Xie, Nan Jiang, Huan Wang, Caiming Xiong, and Yu Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *Advances in neural information processing systems*, 34:27395–27407, 2021.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. *arXiv preprint arXiv:2412.13630*, 2024.

Haichao Zhang, Wei Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=-Y34L45JR6z.

Ruoqi Zhang, Ziwei Luo, Jens Sjölund, Thomas B Schön, and Per Mattsson. Entropy-regularized diffusion policy with Q-ensembles for offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HA0oLUvuGI.

Han Zheng, Xufang Luo, Pengfei Wei, Xuan Song, Dongsheng Li, and Jing Jiang. Adaptive policy learning for offline-to-online reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11372–11380, 2023.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HN0CYZbAPw.

A Additional Discussions for Related Work

CEP (Lu et al., 2023a) and CFGRL (Frans et al., 2025). Both of them build off from the idea of classifier/classifier-free guidance, which combines the denoising step of a base diffusion/flow policy to the denoising step of for a tilt distribution.

$$\text{CEP (diffusion):} \quad \log \pi^t(s, a^t, t) \leftarrow \alpha \log \pi^t_\beta(s, a^t, t) + (1-\alpha) Q^t(s, a^t, t) \tag{27}$$

$$\text{CFGRL (flow):} \quad v(s, a^t, t) \leftarrow \alpha v_\beta(s, a^t, t) + (1-\alpha) v_{o=1}(s, a^t, t) \tag{28}$$

where $Q^t(s, a^t, t) \leftarrow \log \mathbb{E}_{a^t|a}\left[e^{Q(s,a)}\right]$ is the score of the Boltzmann distribution (i.e., $\propto e^{Q(s,a)}$) at denoising time t and v_o is the velocity field of the policy that is conditioned on a optimality variable, a binary indicator of whether the policy is 'optimal' ($o = 1$ means it is). CEP aims at approximating $\pi \propto \pi_\beta^\alpha e^{(1-\alpha)Q(s,a)}$ whereas CFGRL aims at approximating $\pi \propto \pi_\beta^\alpha \pi_{o=1}^{(1-\alpha)}$.

However, as discussed in many prior work (Du et al., 2023; Bradley & Nakkiran, 2024), even when both the denoising steps are exact ($\log \pi^t$ for diffusion and $v(\cdot, \cdot, t)$ for flow), the denoising process that uses a summation of them do not lead to the correct distribution:

$$\nabla_{a^t} \log \pi^t(a^t \mid s) \neq \nabla_{a^t} \log \pi^t_\beta(a^t \mid s) + \tau \nabla_{a^t} Q^t(s, a^t). \tag{29}$$

DAC (Fang et al., 2025). Diffusion actor critic uses the diffusion formulation where the goal is to find a policy that satisfies $\pi(\cdot \mid s) \propto \pi_\beta(\cdot \mid s) e^{Q(s,\cdot)}$. However, their training objective is derived based on the assumption that

$$\nabla_{a^u} \log p^u(a^u \mid s) \approx \nabla_{a^u} \log p^u_\beta(a^u \mid s) + \tau \nabla_{a^u} Q^u(s, a^u), \tag{30}$$

and additionally

$$\nabla_{a^u} Q^u(s, a^u) \approx \nabla_{a^u} Q(s, a^u). \tag{31}$$

While these assumptions provide a convenient approximation of the objective function, it does not provide guarantees on where policy converges to at the optimum.

B Domain and Experiment Details

We consider 5 domains in our experiments. The dataset size, episode length, and the action dimension for each domain is available in Table 2. For each method and each task, we run four seeds. All the plots and tables are reported the mean of the four seeds (or more for aggregated results) with a 95% confidence interval computed via bootstrapping. All our experiments are run on RTX5000 GPU and our code is written in JAX (Bradbury et al., 2018).

Tasks	Dataset Size	Episode Length	Action Dimension (A)
`cube-double-*`	1M	500	5
`cube-triple-*`	3M	1000	5
`cube-quadruple-100M-*`	100M	1000	5
`antmaze-large-*`	1M	1000	8
`antmaze-giant-*`	1M	1000	8

Table 2: **Domain metadata.**

C Baselines

1. DDPG-based.

FBRAC is a baseline considered in FQL (Park et al., 2025) as a flow counterpart of diffusion Q-learning (DQL) (Wang et al., 2023), where the multi-step flow policy is directly optimized against the Q-function with backpropagation through time (BPTT). In addition to maximizing the Q-value, FBRAC also has a behavior cloning term where the flow policy is trained with the standard flow-matching objective on the dataset action with a BC coefficient α.

We implement this baseline by training a flow-matching policy with the following loss:

$$L_{\text{FBRAC}}(\theta) = \alpha L_{\text{FM}}(\theta) + L_{\text{BPTT}}(\theta), \tag{32}$$

where L_{FM} is the standard flow-matching loss that clones the data behavior (*i.e.*, Equation (17)) and

$$L_{\text{BPTT}}(\theta) = -\mathbb{E}_{a^0=z\sim\mathcal{N}(0,I_A)}\left[Q_\phi\left(s, \text{Clip}\left[z + h\sum_{k=0}^{T-1} v_\theta(s, a^{k-1}, kh)\right]_{-1}^{1}\right)\right], \tag{33}$$

where $\text{Clip}[\cdot]_a^b$ is an element-wise clipping function that makes sure the actions generated from the flow model v_θ are within the valid range $[-1, 1]$ and $\{a^i\}_i$ is the discrete approximation of the ODE trajectory using Euler's method with a step size of $h = 1/T$:

$$a^{i+1} := a^i + h v_\theta(s, a^{i-1}, ih), \forall i \in \{0, 1, \cdots, T\}. \tag{34}$$

We use

$$\text{ODE}(v_\theta(s, \cdot, \cdot), z) := \text{Clip}\left[z + h\sum_{i=0}^{T-1} v_\theta(s, a^{i-1}, ih)\right]_{-1}^{1} \tag{35}$$

as the abbreviation for the rest of this section, and additionally use π_{v_θ} to denote the distribution of actions generated by v_θ.

The critic loss is the standard TD backup:

$$L(\phi) = \mathbb{E}_{z\sim\mathcal{N}}\left[(Q_\phi(s, a) - r - Q_{\bar{\phi}}(s', \text{ODE}(v_\theta(s', \cdot, \cdot), z))^2\right] \tag{36}$$

In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (26) and the policy we use to interact with the environment is π_{v_θ}.

FQL (Park et al., 2025) distill a multi-step flow policy into a one-step distillation to avoid BPTT.

This baseline is implemented by training a behavior cloning flow-matching policy (*i.e.*, $v_\theta : \mathcal{S} \times \mathbb{R}^A \times [0, 1] \to \mathbb{R}^A$), and a 1-step distilled noise-conditioned policy (*i.e.*, $\Omega_\omega : \mathcal{S} \times \mathbb{R}^A \to \mathbb{R}^A$):

$$L_{\text{FQL}}(\theta, \omega) = L_{\text{FM}}(\theta) + L_{\text{onestep}}(\omega) \tag{37}$$

where $L_{\text{FM}}(\theta)$ is the standard flow-matching loss (i.e., Equation (17)) and

$$L_{\text{onestep}}(\omega) = \mathbb{E}_{z\sim\mathcal{Z}}\left[\alpha\|\Omega_\omega(s, z) - \text{ODE}(v_\theta(s', \cdot, \cdot), z)\|_2^2 - Q(s, \Omega_\omega(s, z))\right] \tag{38}$$

where α is the BC coeffient that controls how close the 1-step distilled policy should be relative to the BC policy π_{v_θ}. Finally, the critic loss is the standard TD backup:

$$L(\phi) = \mathbb{E}_{z\sim\mathcal{N}}\left[(Q_\phi(s, a) - r - Q_{\bar{\phi}}(s', \Omega_\omega(s', z))^2\right] \tag{39}$$

In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (26) and the policy we use to interact with the environment is π_{Ω_ω} where the action is sampled by first drawing a Gaussian noise $z \sim \mathcal{N}$ and then obtain the action by running through the one-step distilled model: $a = \Omega_\omega(s, z)$.

2. Directly using the action gradient of the critic (i.e., $\nabla_a Q(s, a)$) with approximations.

CGQL is a novel baseline built on top of the idea of classifier guidance (Dhariwal & Nichol, 2021). In particular, we combine the velocity field of a behavior cloning flow policy and the gradient field of the Q-function to form a new velocity field that approximates the velocity field that generates the optimal behavior-constrained action distribution.

More specifically, we implement this baseline by interpreting $Q_\phi(s, \cdot)$ as the score of the optimal entropy-regularized distribution $\log \pi^\star(\cdot \mid s)$ (where $\pi^\star(\cdot \mid s) \propto e^{\tau Q_\phi(s,\cdot)}$). The corresponding velocity field that generates this distribution of actions can be obtained through a simple conversion (e.g., following Equation 4.79 from lip (2024)):

$$v_\phi(s, a, u) := \frac{(1 - t)Q_\phi^u(s, a) + a}{u}, \tag{40}$$

where $Q^u_\phi(s,a) = \log \mathbb{E}_{a^u=(1-u)a+uz, z\sim\mathcal{N}(0,I_A)}\left[e^{Q_\phi(s,a)}\right]$ is the score of the distribution over the noisy intermediate actions. In the classifier guidance literature, the score of the noisy examples is approximated by the score of the noise-free examples at the noisy examples (). In our setting this translates to

$$\hat{v}_\phi(s,a,u) := \frac{(1-u)Q_\phi(s,a)+a}{t}. \tag{41}$$

Empirically, both versions (v_ϕ and $\hat{v}_\phi$) perform similarly and we opt for a simpler design $\hat{v}$ as it does not require learning or approximating $Q^u_\phi(s,a)$ for all t's. Finally, we add the velocity field defined by Q_ϕ directly to the behavior cloning velocity field to form our policy:

$$v = v_\beta + \hat{v}_\phi, \tag{42}$$

where v_β is trained with the standard flow-matching loss (i.e., $L(\beta) = L_{\text{FM}}(\beta)$). The critic loss uses π_v to backup the target Q-value:

$$L(\phi) = \mathbb{E}_{z\sim\mathcal{N}}\left[\left(Q_\phi(s,a) - r - Q_{\bar{\phi}}(s', \text{ODE}(v(s',\cdot,\cdot),z))^2\right]\right. . \tag{43}$$

In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (26) and the policy we use to interact with the environment is π_v (generated from the summation of v_β and $\hat{v}_\phi$.

3. Directly using the critic value (i.e., $Q(s,a)$).

FAWAC is a baseline considered in FQL (Park et al., 2025) where it uses AWR to train the flow policy similar to QIPO (Zhang et al., 2025).

We implement it by training a flow-matching policy with the weighted flow-matching loss:

$$\begin{aligned} L_{\text{FAWAC}}(\theta) &= \tilde{w}(s,a)L_{\text{FM}}(\theta) && (44)\\ &= \tilde{w}(s,a)\mathbb{E}_{u\sim\mathcal{U}[0,1],z\sim\mathcal{N}}\left[\|v_\theta(s,(1-u)z+ua,u) - z + a\|_2^2\right] && (45)\end{aligned}$$

where $\tilde{w}(s,a) = \min\left(e^{\tau(Q_\phi(s,a)-V_\xi(s))}, 100.0\right)$. The inverse temperature parameter τ controls how sharp the prior regularized optimal policy distribution is.

The critic function $Q_\phi(s,a)$ is trained with the standard TD backup and the value function $V_\xi(s)$ regresses to the same target:

$$\begin{aligned} L(\phi) &= \mathbb{E}_{z\sim\mathcal{N}}\left[\left(Q_\phi(s,a) - r - Q_{\bar{\phi}}(s', \text{ODE}(v(s',\cdot,\cdot),z))^2\right]\right. && (46)\\ L(\xi) &= \mathbb{E}_{z\sim\mathcal{N}}\left[\left(V_\xi(s) - r - Q_{\bar{\phi}}(s', \text{ODE}(v(s',\cdot,\cdot),z))^2\right]\right. && (47)\end{aligned}$$

The second line can also be alternatively implemented by regressing to the critic function $Q(s,a)$ directly. We implement in this particular way because we can re-use the Q-target computed.

In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (26) and the policy we use to interact with the environment is π_{v_θ}.

4. Post-processing-based.

FEdit is a baseline that uses the policy edit from a recent offline-to-online RL method conceptually similar to EXPO (Dong et al., 2025). We implement a Gaussian edit policy on top of a BC flow policy rather than a diffusion policy used in EXPO. EXPO also uses the standard sample-and-rank trick where it samples multiple actions and rank them based on the value. To keep computational cost down and comparisons fair to other methods, we only use a single edited action for both value backup and evaluation.

We implement this baseline by training a flow-matching policy (*i.e.*, $v_\theta : \mathcal{S}\times\mathbb{R}^A\times[0,1]\to\mathbb{R}^A$), and a 1-step Gaussian edit policy (*i.e.*, $\pi_\omega : \mathcal{S}\times\mathcal{A}\to\Delta_{\mathcal{A}}$) implemented with an entropy regularized SAC policy (Haarnoja et al., 2018). The loss function can be described as follows:

$$L_{\text{FEdit}}(\theta,\omega) = L_{\text{FM}}(\theta) + L_{\text{Gaussian}}(\omega), \quad \text{s.t.} \quad \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{H}(\pi_\omega(\cdot\mid s))\right] \geq H_{\text{target}} \tag{48}$$

where L_{FM} is the standard flow-matching loss that clones the data behavior (*i.e.*, Equation (17)), H_{target} is the target entropy that the Gaussian policy is constrained to be above of, and

$$L_{\text{Gaussian}}(\omega) = \mathbb{E}_{\Delta a\sim\pi_\omega(\cdot|s,\tilde{a}),z\sim\mathcal{N}}\left[-Q_\phi(s,\text{Clip}\left[\sigma_a\cdot\Delta a+\tilde{a},z\right])\right]^1_{-1}\Big) \tag{49}$$

where $\tilde{a} = \text{ODE}(v_\theta(s, \cdot, \cdot), z)$ and $\text{Clip}[\cdot]_a^b$ is an element-wise clipping function that makes sure the actions are within the valid range $[-1, 1]$.

Intuitively, the Gaussian SAC policy edits the behavior flow policy by modifying its output action where σ_a is the hyperparameter that controls how much the origianl behavior actions can be edited.

The critic loss is the standard TD backup:

$$L(\phi) = \mathbb{E}_{z\sim\mathcal{N},\Delta a'\sim\pi_\omega(\cdot|s',\tilde{a}')}\left[(Q_\phi(s,a) - r - Q_{\bar{\phi}}(s', \text{Clip}\left[\tilde{a}' + \sigma_a \cdot \Delta a'\right]_{-1}^{1})^2\right] \tag{50}$$

where again $\tilde{a}' = \text{ODE}(v_\theta(s', \cdot, \cdot), z)$. In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (26) and the policy we use to interact with the environment is a combination of the BC policy π_{v_θ} and the Gaussian edit policy. We first sample $z \sim \mathcal{N}$ and then run it through the BC flow policy to obtain an initial action $\tilde{a} \leftarrow \text{ODE}(v_\theta(s, \cdot, \cdot), z)$ and then both the initial action and the state is fed into the edit policy to generate the final action $a \leftarrow \tilde{a} + \sigma_a \cdot \Delta a$ where $\Delta a \sim \pi_\omega(\cdot \mid s, \tilde{a})$.

DSRL (Wagenmaker et al., 2025) is a recently proposed method that performs RL directly in the noise-space of a pre-trained expressive BC policy (flow or diffusion). We use the flow-matching version of DSRL as our method is also based on flow-matching policies. The original DSRL implementation does not fine-tune the BC policy during online learning while all our baselines do fine-tune the BC policy online. To make the comparison fair, we modify the DSRL implementation such that it also fine-tunes the BC policy. One additional implementation trick that allows this modification to work well is the use of target policy network for the noise-space policy. In general, we find that fine-tuning the BC policy yields better online performance, so we adopt this new design of DSRL in our experiments.

More specifically, we train a flow-matching policy (*i.e.*, $v_\theta : \mathcal{S} \times \mathbb{R}^A \times [0, 1] \to \mathbb{R}^A$), and a 1-step Gaussian edit policy (*i.e.*, $\pi_\omega : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{A}}$) implemented with an entropy regularized SAC policy (Haarnoja et al., 2018). The loss function can be described as follows:

$$L_{\text{DSRL}}(\theta, \omega) = L_{\text{FM}}(\theta) + L_{\text{LatentGaussian}}(\omega), \quad \text{s.t.} \quad \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{H}(\pi_\omega(\cdot \mid s))\right] \geq H_{\text{target}} \tag{51}$$

where L_{FM} is the standard flow-matching loss that clones the data behavior (*i.e.*, Equation (17)), H_{target} is the target entropy that the Gaussian policy is constrained to be above of, and

$$L_{\text{LatentGaussian}}(\omega) = \mathbb{E}_{z\sim\pi_\omega(\cdot|s)}\left[-Q_\psi^z(s, z)\right] \tag{52}$$

where $Q_\psi^z(s, z)$ is a distilled critic function in the noise space that is regressed to the original critic function, $Q_\phi(s, a)$:

$$L(\psi) = \mathbb{E}_{z\sim\mathcal{N}}\left[(Q_\psi^z(s, z) - Q_\phi(s, \text{ODE}(v_{\bar{\theta}}(s, \cdot, \cdot), z))^2\right]. \tag{53}$$

Intuitively, DSRL directly learns a policy in the noise space by hill-climbing a distilled critic that also operates in the noise space. Finally, the critic loss for the original critic function in the action space is

$$L(\phi) = \mathbb{E}_{z\sim\pi_\omega(\cdot|s')}\left[(Q_\phi(s, a) - r - Q_{\bar{\phi}}(s', \text{ODE}(v_{\bar{\theta}}(s', \cdot, \cdot), z))^2\right] \tag{54}$$

We use $K = 10$ critic functions and pessimistic target value backup as described in Equation (26). The policy we use to interact with the environment is a combination of the BC policy π_{v_θ} and the Gaussian noise-space policy. We first sample $z \sim \pi_\omega(\cdot \mid s)$ and then run it through the BC flow policy to obtain the final action $a \leftarrow \text{ODE}(v_\theta(s, \cdot, \cdot), z)$. One important implementation detail for stability in the offline-to-online setting is to use the target network for the BC flow policy $v_{\bar{\theta}}$ (instead of v_θ. Without it DSRL can become unstable sometimes when the BC flow policy changes too fast online.

IFQL is a baseline considered in FQL (Park et al., 2025) as a flow counterpart of implicit diffusion Q-learning (IDQL) (Hansen-Estruch et al., 2023), where IQL (Kostrikov et al., 2021) is used for value learning and the policy extraction is done by sampling multiple actions from a behavior cloning diffusion policy and select the one that maximizes the Q-function value.

More specifically, we train a critic function $Q_\phi(s, a)$ and a value function $V_\xi(s)$ with implicit value backup:

$$L(\phi) = (Q_\phi(s, a) - r - V_\xi(s'))^2 \tag{55}$$

$$L(\xi) = f_{\text{exp}}^\tau(Q_{\bar{\phi}}(s, a) - V_\xi(s)) \tag{56}$$

where $f^{\tau}_{\text{exp}}(u) = |u - \mathbb{I}_{u<0}|u^2$ is the expectile regression loss function.

On top of that, we also $K = 10$ critic functions and pessimistic target value backup as described in Equation (26) for training the value function $V_\xi(s)$. To extract a policy from $Q_\phi(s, a)$, IFQL uses rejection sampling with a base behavior cloning flow policy that is trained with the standard flow-matching objective. In particular, the output action $a^\star$ for s is selected as the following:

$$a^\star \leftarrow \arg\max_{a_1,\cdots,a_N} Q(s, a_i), \quad a_1, \cdots, a_N \sim \pi_{v_\beta}(\cdot \mid s). \tag{57}$$

5. Gaussian.

RLPD (Ball et al., 2023) is a strong offline-to-online RL method that trains a SAC agent from scratch online with a 50/50 sampling scheme (*i.e.*, 50% of training examples in a batch comes from the offline dataset wheras the other 50% of training examples comes from the online replay buffer).

D HYPERPARAMETERS

While most methods share a common set of hyperparameters (Table 3 for a fair comparison, most methods need to be tuned for each domain. We include the domain-specific in Table 4 and the tuning range of them in Table 5.

Parameter	Value
Batch size	256
Discount factor (γ)	0.99 for `cube-*/antmaze-large-*` 0.999 for `antmaze-giant-*`
Optimizer	Adam
Learning rate	3×10^{-4}
Target network update rate (λ)	5×10^{-3}
Critic ensemble size (K)	10
Critic target pessimistic coefficient (ρ)	0.5
UTD Ratio	1
Number of flow steps (T)	10
Number of offline training steps	10^6; except RLPD (0)
Number of online environment steps	0.5×10^6
Network width	512
Network depth	4 hidden layers

Table 3: **Common hyperparameters.**

Domains	FBRAC	CGQL	FQL	DSRL	FEdit	FAWAC	IFQL	QAM
	α	τ	α	σ_z	σ_a	τ	N	(τ, c)
`cube-double-*`	0.1	0.1	300.0	0.5	0.25	10.0	32	$(1.0, 10.0)$
`cube-triple-*`	0.01	0.1	100.0	1.0	0.5	10.0	32	$(10.0, 100.0)$
`cube-quadruple-100M-*`	1.0	0.1	30.0	1.25	0.5	10.0	32	$(3.0, 10.0)$
`antmaze-large-*`	10.0	0.1	10.0	0.25	0.25	10.0	32	$(10.0, 100.0)$
`antmaze-giant-*`	10.0	0.1	10.0	0.25	0.25	10.0	32	$(3.0, 100.0)$

Table 4: **Domain-specific hyperparameters for each method.**

Method	Hyperparameter(s)	Sweep Range
FBRAC	α	$\{0.01, 0.1, 1.0, 10, 100\}$
CGQL	τ	$\{0.1, 0.3, 1, 3, 10\}$
FQL	α	$\{3, 10, 30, 100, 300\}$
DSRL	σ_z	$\{0.25, 0.5, 0.75, 1, 1.25\}$
FEdit	σ_a	$\{0.1, 0.25, 0.5, 1.0\}$
FAWAC	τ	$\{1.0, 3.0, 10.0\}$
QAM	(τ, c)	$(\{1.0, 3.0, 10.0\}, \{10.0, 100.0\})$

Table 5: **Domain-specific hyperparameter tuning range for each method.**