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

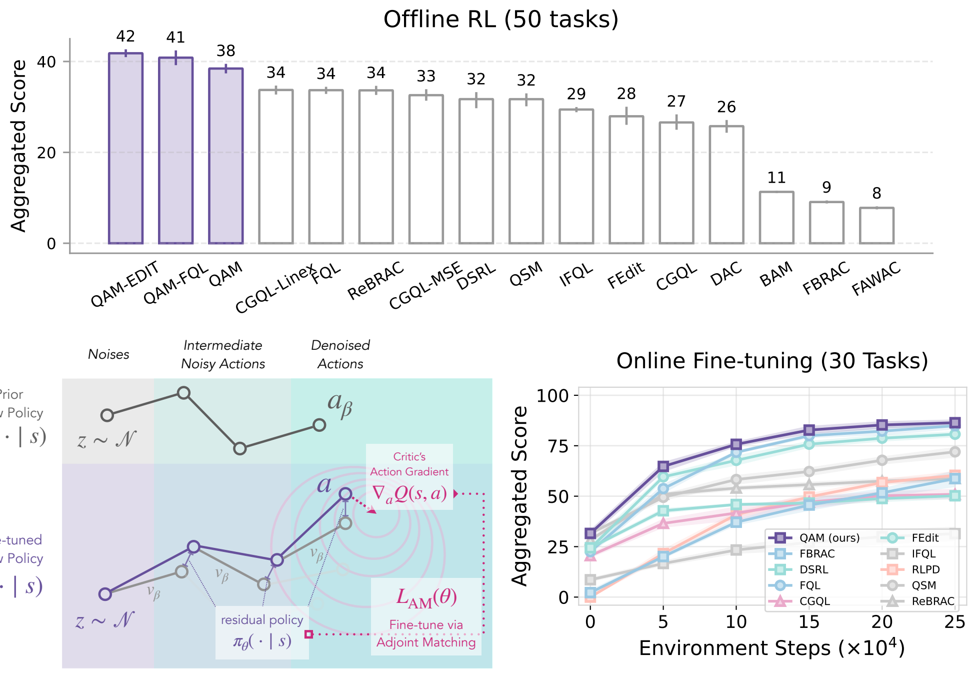

Figure 1: **QAM: Q-learning with Adjoint Matching.** *Bottom-left:* QAM uses adjoint matching objective (Domingo-Enrich et al., 2025) that leverages the critic's action gradient directly to fine-tune a residual flow policy such that the combined policy converges to the optimal prior-constrained policy: $\pi(\cdot \mid s) \propto \pi_\beta(\cdot \mid s)e^{Q(s,\cdot)}$. *Top/Bottom-right:* Aggregated score for offline RL and online fine-tuning (8 seeds).

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., 2025b). 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., 2025b). The former sacrifices learning efficiency and often under-performs methods that use the critic's action gradient (Park et al., 2024b; 2025b), 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 2).

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 refine the action distribution from a base diffusion/flow policy 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.*, rejection sampling for 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) *Backprop-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., 2025b; 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. (2025b), or instead learns a distilled policy (Ding & Jin, 2023; Chen et al., 2024b; Park et al., 2025b; 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 backprop-based 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), these methods often under-perform the methods that leverage explicit offline pre-training, especially on more challenging tasks (Nakamoto et al., 2024; Park et al., 2025b). Our method also operates in this regime where we first perform offline RL pre-training and then perform online fine-tuning from the offline pre-trained initialization. In addition, we follow a common design in prior work where the same offline RL objective is used for both offline pre-training and online fine-tuning (Kostrikov et al., 2021; Fujimoto & Gu, 2021; Tarasov et al., 2023; Park et al., 2025b). 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.*, $\propto \pi_1\pi_2$) 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 D consisting of a set of transitions $\{(s_i, a_i, s_i', r_i)\}_{i=1}^{|D|}$, 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 D that maximizes its expected discounted return,

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

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 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, $X_1 = D$, 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: $X^0 = \mathcal{N}$:

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

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

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

where any optimal velocity field, $v_{\theta^\star}$, results in $\hat{X}_t$ where its marginal distribution $p_f(x_t)$ exactly recovers the marginal distribution of the original denoising process X_t, $p_D(x_t)$, for each $t \in$

$[0, 1]$ (Lipman et al., 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:

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

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 f_β such that it generates the following *tilt* distribution:

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

where $Q : \mathbb{R}^d \to \mathbb{R}$ is any value 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), $\sigma_t = \sqrt{2(1-t)/t}$:

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

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

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

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

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

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

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

which satisfies the following ODE:

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

with the boundary condition $g(\boldsymbol{X}, 1) = -\nabla_{X_1} r(X_1)$. 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:

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

The optimal f_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:

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

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

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

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

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 the optimal solution for f_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:

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

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, $f_\beta : \mathcal{S} \times \mathbb{R}^A \times [0,1] \to \mathbb{R}^A$ that is optimized with the standard flow-matching objective:

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

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

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

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

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

Similar 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 $f_\theta + f_\beta$ and f_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:

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

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

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

Unlike the original SOC objective (Equation (18)) from which calculating the gradient requires back-propagating through an SDE, which suffers from stability challenges, the adjoint matching objective is constructed without backpropagation. Instead, it uses the behavior velocity field f_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.*, $f_\theta + f_\beta$). This is an important distinction because for direct backpropagation, any ill-conditioned action gradient in f_θ (*i.e.*, $\nabla_a f_\theta(s,a,t)$) 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 f_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:

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

where $f(s,\cdot,\cdot) = f_\beta(s,\cdot,\cdot) + f_\theta(s,\cdot,\cdot)$ 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 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

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

We set the boundary condition as $\tilde{g}^1 = -\tau\nabla_{a^1}Q_\phi(s,a^1)$, 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 parameter gradient element-wise by 1 for numerical stability. The backward adjoint state calculation process is then approximated by

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

with $\mathrm{JVP}(\nabla_y b(y), x) = x^\top \nabla_y b(y)$ 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 ϕ^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: $f(s',\cdot,\cdot) = f_\beta(s,\cdot,\cdot) + f_\theta(s,\cdot,\cdot)$. For all our experiments, we do not use a separate training process for f_β and instead training it at the same to as f_θ and Q_ϕ, following Park et al. (2025b); 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 D uniformly. During offline training, D is the offline data. During online fine-tuning, D is combination of the offline and online replay buffer data without any re-weighting.

Theoretical guarantees. As our algorithm builds off from Domingo-Enrich et al. (2025), we can directly extend their theoretical results to our setting as follows (proof in Appendix E):

Proposition 1 (Extension of Proposition 7 in Domingo-Enrich et al. (2025) to Policy Optimization.) Take $L_{\mathrm{AM}}(\theta)$ in Equation (14), there is a unique f_θ such that

$$\frac{\partial}{\partial f_\theta} L_{\mathrm{AM}} = 0, \tag{27}$$

and for all $s \in \mathrm{supp}(D)$,

$$\pi_\theta(\cdot \mid s) \propto \pi_\beta(\cdot \mid s)e^{\tau Q_\phi(s,a)}. \tag{28}$$

The importance of this result is that as long as the loss function L_{AM} is optimized to convergence (*i.e.*, with $\partial L/\partial f_\theta = 0$), the learned policy coincides with the optimal behavior-constrained policy.

Combining with existing flow/diffusion RL methods. In practice, we also find it to be beneficial to combine QAM with existing methods such as FQL (Park et al., 2025b) and edit policies (Dong et al., 2025) to further boost the performance. For the `QAM-FQL` variant, we learn the 1-step noise-conditioned policy from Park et al. (2025b), $\mu_\omega(s,z)$, and optimize the following objective (with α being a tunable BC coefficient),

$$L_{\texttt{QAM-FQL}}(\omega) = \mathbb{E}_{z\sim\mathcal{N}}\left[-Q_\phi(s,\mu_\omega(s,z)) + \alpha\|\mu_\omega(s,z) - \mathrm{ODE}(f(s,\cdot,\cdot),z)\|_2^2\right]. \tag{29}$$

Algorithm 1 Learning procedure in QAM.

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

We then use this 1-step policy both to interact with the environment and to compute value targets (*i.e.*, $\bar{Q}(s', a' = \mu(s, z \sim \mathcal{N}))$). Intuitively, the 1-step policy is optimized to remain close to the QAM-fine-tuned flow policy while also maximizing the action value under the current critic. For the `QAM-EDIT` variant, we optimize a Gaussian edit policy from Dong et al. (2025), $\pi_\omega(\cdot \mid s, \tilde{a})$, to modify the output from the QAM-fine-tuned flow policy (*i.e.*, $\tilde{a}$) for further action refinements with the objective below:

$$L_{\texttt{QAM-EDIT}}(\omega) = \mathbb{E}_{\Delta a \sim \pi_\omega(\cdot|s,\tilde{a}), z\sim\mathcal{N}} \left[-Q_\phi(s, \Delta_a + \tilde{a})\right], \quad \text{where } \tilde{a} := \mathrm{ODE}(f(s, \cdot, \cdot), z) \tag{30}$$

where Δ_a is restricted to be within a L_∞ ball (with a tunable scaling parameter of σ_a). See more details on the implementation on the edit policy and the FQL 1-step policy in Appendix C.

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 10 domains from OGBench (Park et al., 2024a): `scene`, `puzzle-3x3` (`p33`), `puzzle-4x4` (`p44`), `cube-double` (`c2`), `cube-triple` (`c3`), `cube-quadruple` (`c4`), `humanoidmaze-medium` (`hm`), `humanoidmaze-large` (`hl`), `antmaze-large` (`al`), and `antmaze-giant` (`ag`). For `antmaze-*` and `humanoidmaze-*`, we use the default `navigate` datasets. For `scene`, `puzzle-*`, and `cube-*`, we use the default `play` datasets except for `c4` and `p44` where we use the larger 100M-size dataset from Park et al. (2025a), and use the sparse reward definition for `{p33, p44, scene}`, 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 4) 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-*, scene-*, p33-*, p44-*}` domains, we follow Li et al. (2025) to learn action chunking policies with an action chunking 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—**(1) Gaussian:** `ReBRAC` (Tarasov et al., 2023), **(2) Backprop:** `FBRAC` (Park et al., 2025b) (backprop through the flow policy's denoising step directly), `FQL` (backprop through a 1-step distilled policy) (Park et al., 2025b); **(3) Adv. Weighted:** `FAWAC` (Park et al., 2025b) (advantage weighted actor critic, AWAC (Nair et al., 2020), with flow policy); **(4) Guidance with the critic's action gradient:** `DAC` (Fang et al., 2025), `QSM` (Psenka et al., 2023), and `CGQL`/`CGQL-MSE`/`CGQL-Linex` (three variants of classifier guidance-based methods inspired by Dhariwal & Nichol (2021)); **(5) 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)). For offline-to-online evaluations we additionally compare with `RLPD` (Ball et al., 2023). Finally, we compare with `BAM`, a direct ablation from our method QAM where we use the 'basic' adjoint matching objective in Equation (12) instead of the adjoint matching objective in Equation (14), and keep the rest of the

		al	ag	hm	hl	scene	p33	p44	c2	c3	c4	all
		5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	50 tasks
GAUSSIAN	ReBRAC	[93,95] **94**	[33,43] **38**	[38,46] 42	[6,9] **8**	[55,66] 61	[63,80] 72	[0,0] 0	[7,11] 8	[0,1] 1	[1,4] 2	[33,35] 34
BACKPROP	FBRAC	[0,3] 1	[0,0] 0	[25,32] 28	[0,0] 0	[37,48] 43	[0,0] 0	[13,21] 18	[0,1] 0	[0,0] 0	[0,0] 0	[9,9] 9
	BAM	[68,72] 70	[3,5] 4	[18,22] 20	[1,1] 1	[11,14] 13	[2,3] 2	[0,0] 0	[2,2] 2	[0,1] 1	[0,0] 0	[11,12] 11
	FQL	[63,69] 66	[0,0] 0	[57,62] 59	[2,4] 3	[64,70] 67	[100,100] 100	[4,6] 5	[12,42] 30	[1,2] 2	[0,2] 1	[33,34] 34
ADV. WEIGHTED	FAWAC	[15,20] 17	[0,0] 0	[17,21] 19	[0,0] 0	[35,39] 37	[3,4] 3	[0,0] 0	[0,1] 1	[0,0] 0	[0,0] 0	[7,8] 8
GUIDANCE	CGQL	[64,74] 69	[0,0] 0	[46,55] 51	[4,5] 4	[21,31] 26	[38,48] 43	[23,38] **30**	[28,45] **37**	[4,7] **5**	[0,0] 0	[25,28] 27
	CGQL-MSE	[63,71] 67	[3,12] 8	[39,40] 39	[0,0] 0	[62,70] 66	[100,100] **100**	[0,0] 0	[30,49] **40**	[4,7] **5**	[0,0] 0	[31,34] 33
	CGQL-Linex	[54,62] 58	[0,0] 0	[45,51] 48	[3,4] 3	[82,89] 86	[93,99] 96	[0,0] 0	[30,49] **41**	[4,6] **5**	[0,0] 0	[33,35] 34
	DAC	[73,85] 79	[8,11] 10	[63,73] **68**	[0,0] 0	[54,64] 59	[11,20] 16	[0,0] 0	[18,23] 20	[2,5] **4**	[1,4] 3	[24,27] 26
	QSM	[68,75] 72	[1,8] 4	[63,75] **68**	[5,7] 6	[69,73] 71	[25,41] 34	[0,0] 0	[39,46] **42**	[2,4] **3**	[11,19] **16**	[30,33] 32
POST-PROCESSING	DSRL	[33,42] 37	[0,1] 1	[34,47] 40	[0,1] 0	[100,100] **100**	[70,90] 80	[0,0] 0	[44,71] **59**	[0,1] 0	[0,0] 0	[30,33] 32
	FEdit	[29,44] 37	[0,1] 0	[0,6] 3	[1,3] 2	[64,72] 68	[99,100] 99	[21,38] **29**	[22,34] 30	[1,2] 1	[0,0] 0	[26,30] 28
	IFQL	[19,28] 23	[0,0] 0	[68,71] **70**	[7,13] **10**	[79,84] 82	[100,100] **100**	[0,0] 0	[8,10] 9	[0,0] 0	[0,0] 0	[29,30] 29
ADJOINT MATCHING	QAM	[62,65] 63	[5,18] 11	[58,62] 60	[1,4] 2	[97,98] 98	[94,100] 97	[0,0] 0	[50,60] **57**	[2,7] **5**	[2,4] 3	[37,39] 38
	QAM-FQL	[69,77] 73	[3,14] 8	[52,64] 59	[3,6] 5	[97,99] 98	[100,100] 100	[4,12] 8	[38,56] **49**	[1,2] 1	[6,12] **9**	[39,42] **41**
	QAM-EDIT	[65,70] 67	[0,4] 2	[55,67] 61	[2,5] 3	[97,98] 98	[99,100] 100	[24,35] **29**	[53,60] **57**	[2,4] **4**	[0,3] 2	[41,43] **42**

Table 1: **Offline RL performance at 1M training steps (50 tasks, 8 seeds).** Our method (QAM) and two of its variants, QAM-FQL and QAM-EDIT outperform all prior baselines. We use abbreviations of the domains as follows: al=antmaze-large, ag=antmaze-giant, hm=humanoidmaze-medium, hl=humanoidmaze-large, p33=puzzle-3x3, p44=puzzle-4x4, c2=cube-double, c3=cube-triple, c4=cube-quadruple.

implementation exactly the same. We categorize it as a "backprop" method because its gradient is equivalent to that of backpropagating through the memoryless SDE as we discuss above in Section 3.

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 $\rho = 0.5$ (except on humanoidmaze-large where we find $\rho = 0$ to work better), 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 10 different domains (50 tasks in total). QAM outperforms all prior methods with an aggregated score of 38. Furthermore, combining QAM with FQL and FEdit can push the performance even further. QAM-FQL achieves an aggregated score of 40 and QAM-EDIT achieves an even higher aggregated score of 42.

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

Next, we take the best performing variant of QAM, QAM-EDIT, and evaluate its ability to online fine-tune from its offline RL initialization. Figure 2 shows the sample efficiency curve (with x-axis being the number of environment steps). QAM outperforms all prior methods on cube-triple and is the most robust method across the board. For example, compared to ours, QSM fine-tunes better on al, ag but struggles on all other tasks except on c2 where it performs similar to our method. FQL fine-tunes slightly better on ag and much slower on both p44 and c3.

(Q3) How sensitive is our method to hyperparameters? Finally, we conduct sensitivity analyses for various components of our method including pessimistic backup (ρ), gradient clipping, number of

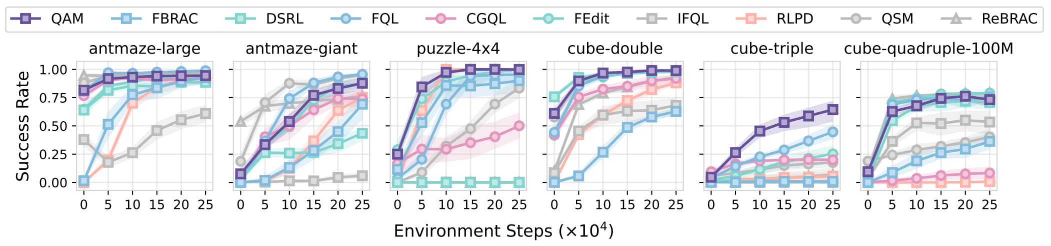

Figure 2: **QAM online fine-tunes more effectively than prior methods (30 tasks, 8 seeds).** For online fine-tuning experiments, we use the `QAM-FEdit` variant for `QAM` and we use `CGQL-Linex` variant for `CGQL` due to their good performance in our offline experiments.

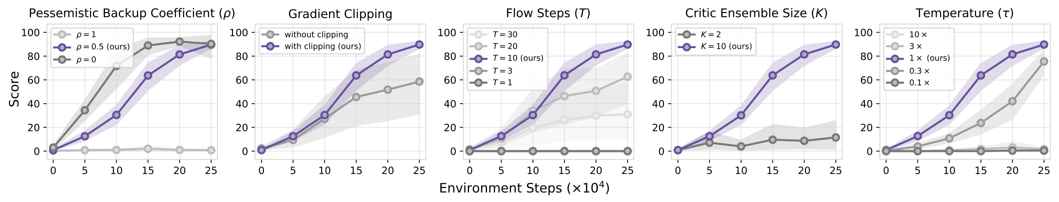

Figure 3: **Sensitivity analysis on `cube-triple-task2` (8 seeds).** *Pessimistic Backup Coefficient (ρ)*: this parameter controls how the standard deviation multiplier in the TD backup target $\bar{Q}_{\text{mean}} - \rho\bar{Q}_{\text{std}}$ (Equation (26)); *Gradient Clipping*: whether to use gradient clipping in our optimizer; *Flow Steps (T)*: this parameter indicates the number of numerical integration steps that we use for the flow model. *Critic Ensemble Size (K)*: number of critic network in the ensemble; *Temperature (τ)*: the parameter that modulates the influence of the prior. We rerun our method with $0.1\times, 0.3\times, 3\times$, and $10\times$ the temperature value we obtain from our tuning runs.

flow steps (T), critic ensemble size (K) and the temperature coefficient (τ). As we show in Figure 3, all of these components contribute to QAM's effectiveness. Among them, the pessimistic backup coefficient, and the temperature parameter (τ) have the biggest impact on QAM's performance and need to be tuned. For the other components, we find enabling gradient clipping and having a large critic ensemble size ($K = 10$) always helps. We also find setting the number of flow steps to $T = 10$ works the best empirically.

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

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.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

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=ldVkAO09Km.

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.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

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, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

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.

Vivek Myers, Bill Chunyuan Zheng, Benjamin Eysenbach, and Sergey Levine. Offline goal-conditioned reinforcement learning with quasimetric representations. *arXiv preprint arXiv:2509.20478*, 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, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *arXiv preprint arXiv:2506.04168*, 2025a.

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

Ahmad Parsian and SNUA Kirmani. Estimation under linex loss function. In *Handbook of applied econometrics and statistical inference*, pp. 75–98. CRC Press, 2002.

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.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *International Conference on Machine Learning (ICML)*, 2024.

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.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

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.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

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.

$$\text{CEP (diffusion):} \quad \log \pi^t(s, a^t, t) \leftarrow \alpha \log \pi^t_\beta(s, a^t, t) + (1-\alpha)Q^t(s, a^t, t) \tag{31}$$

$$\text{CFGRL (flow):} \quad v(s, a^t, t) \leftarrow \alpha v_\beta(s, a^t, t) + (1-\alpha)v_{o=1}(s, a^t, t) \tag{32}$$

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 $\pi \propto \pi^\alpha_\beta e^{(1-\alpha)Q(s,a)}$ whereas CFGRL aims at approximating $\pi \propto \pi^\alpha_\beta \pi^{(1-\alpha)}_{o=1}$.

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:

$$\nabla_{a^t} \log \pi^t(a^t \mid s) \neq \nabla_{a^t} \log \pi^t_\beta(a^t \mid s) + \tau \nabla_{a^t} Q^t(s, a^t). \tag{33}$$

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

$$\nabla_{a^u} \log p^u(a^u \mid s) \approx \nabla_{a^u} \log p^u_\beta(a^u \mid s) + \tau \nabla_{a^u} Q^u(s, a^u), \tag{34}$$

and additionally

$$\nabla_{a^u} Q^u(s, a^u) \approx \nabla_{a^u} Q(s, a^u). \tag{35}$$

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 10 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 8 seeds. All plots and tables report the means with 95% confidence intervals computed via bootstrapping. All our experiments are run on NVIDIA RTX-A5000 GPU and our code is written in JAX (Bradbury et al., 2018). We use NVIDIA-A5000 GPU to run all our experiments. Each complete offline-to-online experiment run takes around 3 hours. To reproduce all our results in Table 1 and Figure 2, we estimate that it would take around $\underbrace{3}_{\text{hours per single run}} \times \underbrace{17}_{\text{\# of methods}} \times \underbrace{50}_{\text{\# of tasks}} \times \underbrace{8}_{\text{\# of seeds}} = 20\,400$ GPU hours.

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
humanoidmaze-medium-*	2M	2000	21
humanoidmaze-large-*	2M	2000	21
scene-sparse-*	1M	750	5
puzzle-3x3-sparse-*	1M	500	5
puzzle-4x4-100M-sparse-*	100M	500	5

Table 2: **Domain metadata.**

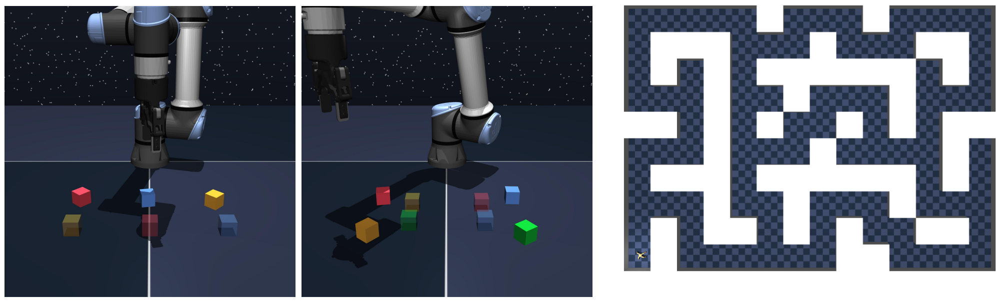

Figure 4: **OGBench domains**. In this paper, we primarily focus on evaluating our method on some of the hardest domains on OGBench (Park et al., 2024a). `cube-{triple, quadruple}` (left) requires a robot arm to manipulate up to 3/4 cubes from an initial arrangement to a goal arrangement. `antmaze-giant` (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.

C Baselines

In this section, we describe in details how each of our baselines are implemented. In all the loss functions below, unless specified otherwise, s, a, r, s' are assumed to be sampled from D and as part of the expectation even when it is not explicitly written under the expectation (*i.e.*, $\mathbb{E}_{z\sim\mathcal{N}}[\cdot] := \mathbb{E}_{(s,a,r,s')\sim D, z\sim\mathcal{N}}[\cdot]$).

C.1 Baseline implementation details

1. Backprop-based.

FBRAC is a baseline considered in FQL (Park et al., 2025b) 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:

$$L_{\text{FBRAC}}(\theta) = \alpha L_{\text{FM}}(\theta) + L_{\text{BPTT}}(\theta), \tag{36}$$

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

$$L_{\text{BPTT}}(\theta) = -\mathbb{E}_{a^0=z\sim\mathcal{N}(0,I_A)}\left[Q_\phi\left(s, \text{Clip}\left[z + h\sum_{k=0}^{T-1} v_\theta(s, a^{k-1}, kh)\right]_{-1}^{1}\right)\right], \tag{37}$$

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 $h = 1/T$:

$$a^{i+1} := a^i + hv_\theta(s, a^{i-1}, ih), \forall i \in \{0, 1, \cdots, T\}. \tag{38}$$

We use

$$\text{ODE}(v_\theta(s,\cdot,\cdot), z) := \text{Clip}\left[z + h\sum_{i=0}^{T-1} v_\theta(s, a^{i-1}, ih)\right]_{-1}^{1} \tag{39}$$

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:

$$L(\phi) = \mathbb{E}_{z\sim\mathcal{N}}\left[(Q_\phi(s,a) - r - Q_{\bar\phi}(s', \text{ODE}(v_\theta(s',\cdot,\cdot), z))^2\right] \tag{40}$$

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., 2025b) 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.*, $\Omega_\omega : \mathcal{S} \times \mathbb{R}^A \to \mathbb{R}^A$):

$$L_{\text{FQL}}(\theta, \omega) = L_{\text{FM}}(\theta) + L_{\text{onestep}}(\omega) \tag{41}$$

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

$$L_{\text{onestep}}(\omega) = \mathbb{E}_{z\sim\mathcal{Z}}\left[\alpha\|\Omega_\omega(s, z) - \text{ODE}(v_\theta(s', \cdot, \cdot), z)\|_2^2 - Q(s, \Omega_\omega(s, z))\right] \tag{42}$$

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:

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

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_{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.

QSM (Psenka et al., 2024) uses the action gradient of the critic to train a diffusion model such that it can reconstruct the policy that follows the Boltzmann distribution of the Q-function:

$$\pi(\cdot \mid s) \propto \tau Q_\phi(s, \cdot). \tag{44}$$

To approximate the score of the intermediate actions, $\nabla_{a^i} \log \pi^i(a^i \mid s)$, QSM directly uses the critic evaluated at a^i:

$$\nabla_{a^i} \log \pi^i(a^i \mid s) \approx \tau \nabla_{a^i} Q_\phi(s, a^i) \tag{45}$$

where $a^i = \sqrt{\alpha^i}a + \sqrt{1 - \alpha^i}\varepsilon$ with $\varepsilon \sim \mathcal{N}$ and $\{\alpha^0, \cdots, \alpha^{T-1}\}$ being any diffusion schedule with the number of diffusion steps of T. For all diffusion methods in this paper, we follow Hansen-Estruch et al. (2023) to use the variance preserving diffusion schedule:

$$\beta^i = 1 - \exp\left(-\frac{b_{\min}}{T} - \frac{(b_{\max} - b_{\min})(2i + 1)}{2T^2}\right), \tag{46}$$

$$\alpha^i = \prod_{j=0}^{i}(1 - \beta^j), \tag{47}$$

for all $i \in \{0, 1, \cdots, T - 1\}$ with $b_{\min} = 0.1, b_{\max} = 10$ and T being the number of diffusion steps.

To train the diffusion model, QSM uses the following loss function:

$$L_{\text{qsm}}(\theta) = \mathbb{E}_{\varepsilon\sim\mathcal{N}, i\sim\mathcal{U}\{0,\cdots,T-1\}}\left[\| - \tau\nabla_a Q_\phi(s, a^i) - f_\theta(s, a^i, i)\|_2^2\right], \tag{48}$$

where again $a^i = \sqrt{\alpha^i}a + \sqrt{1 - \alpha^i}\varepsilon$. In practice, we find that using $\nabla_a Q_{\bar{\phi}}(s, a^i)$ (the action gradient of the critic network) helps learning stability, so we use that instead.

To adopt QSM into the offline-to-online RL setting, we additionally augment the loss function with the standard diffusion loss (Ho et al., 2020) as the behavior regularization:

$$L_{\text{ddpm}}(\theta) = \mathbb{E}_{\varepsilon\sim\mathcal{N}, i\sim\mathcal{U}\{0,\cdots,T-1\}}\left[\|\varepsilon - f_\theta(s, a^i, i)\|_2^2\right]. \tag{49}$$

The overall actor loss is thus

$$L(\theta) = L_{\text{qsm}}(\theta) + \eta L_{\text{ddpm}}(\theta). \tag{50}$$

QSM then uses the standard diffusion denoising procedure (with the clipping to make sure generated actions are within $[-1, 1]^{|A|}$):

$$a^{i-1} \leftarrow \text{Clip}\left[\frac{1}{\sqrt{1 - \beta^i}}\left(a^i - \frac{\beta^i}{\sqrt{1 - \alpha^i}} f_\theta(s, a^i, i)\right) + \sqrt{\beta^i}\varepsilon^i\right]_{-1}^{1}, \quad \varepsilon^i \sim \mathcal{N}, \tag{51}$$

where $a^T \sim \mathcal{N}$. We denote $\text{Diff}(f_\theta(s,\cdot,\cdot))$ as the resulting a^0 after going through the diffusion procedure above and $\pi_{f_\theta} : s \mapsto \text{Diff}(f_\theta(s,\cdot,\cdot))$ as the policy that generates actions using f_θ.

The critic loss can now be defined as follows:

$$L(\phi) = \mathbb{E}_{z\sim\mathcal{N}}\left[(Q_\phi(s,a) - r - Q_{\bar{\phi}}(s', a' \sim \pi_{f_\theta}(\cdot \mid s')))^2\right] \tag{52}$$

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_{f_θ}.

DAC (Fang et al., 2025) is another method that uses a similar approximation with a behavior prior (*i.e.*, $\nabla_{a^i} \log \pi^i(a^i \mid s) \approx \nabla_{a^i} \log \pi^i_\beta(a^i \mid s) + \tau \nabla_{a^i} Q_\phi(s, a^i)$). To train the diffusion model, DAC matches f_θ to a linear combination of ε (behavior cloning) and $\nabla_{a^i} Q(s, a^i)$ (Q-maximization).

$$\begin{aligned} L_{\text{dac}}(\theta) &= \mathbb{E}_{\varepsilon\sim\mathcal{N}, i\sim\mathcal{U}\{0,\cdots,T-1\}}\left[\|\varepsilon - \tau\nabla_{a^i} Q_\phi(s,a^i) - f_\theta(s,a^i,i)\|_2^2\right] \\ &= \mathbb{E}_{\varepsilon\sim\mathcal{N}, i\sim\mathcal{U}\{0,\cdots,T-1\}}\left[\|\varepsilon - f_\theta(s,a^i,i)\|_2^2 + \eta^{-1}(f_\theta(s,a^i,i)\cdot\nabla_{a^i} Q_\phi(s,a^i))\right] + C, \end{aligned} \tag{53}$$

with $\eta = 2/\tau$. In practice, it is implemented as a combination of

$$L(\theta) = \mathbb{E}_{\varepsilon\sim\mathcal{N}, i\sim\mathcal{U}\{0,\cdots,T-1\}}\left[\|(f_\theta(s,a^i,i)\cdot\nabla_{a^i} Q_\phi(s,a^i))\right] + \eta L_{\text{ddpm}}(\theta) \tag{54}$$

DAC uses a slightly different way to clip the actions:

$$\hat{a}^{i-1} \leftarrow \text{Clip}\left[\frac{1}{\sqrt{1-\beta^i}}\left(a^i - \frac{\beta^i}{\sqrt{1-\alpha^i}} f_\theta(s,a^i,i)\right)\right]_{-1}^{1} \tag{55}$$

$$a^{i-1} \leftarrow \frac{\beta^i\sqrt{\alpha^{i-1}}}{1-\alpha^i}\hat{a}^{i-1} + \sqrt{1-\beta^i}(1-\alpha^{i-1})a^i + \sqrt{\beta^i}\varepsilon^i, \quad \varepsilon^i \sim \mathcal{N} \tag{56}$$

where $a^T \sim \mathcal{N}$ and $\hat{a}^{i-1}$ can be interpreted as an approximation of the denoised action (*e.g.*, in the DDIM sampler (Song et al., 2020)). This approximated denoised action is first clipped to be within $[-1,1]^{|A|}$ and then used to reconstruct a^{i-1} using the closed-form Gaussian conditional probability distribution of $p(a^{i-1} \mid a^0 = \hat{a}^{i-1}, a^T = a)$ (*e.g.*, Eq. (7) in Ho et al. (2020)). In practice, DAC also uses $\sqrt{\beta}^i$ as standard deviation for this conditional distribution instead of the correct one $\tilde{\beta}^i = \frac{\sqrt{1-\alpha^{i-1}}}{\sqrt{1-\alpha^i}}\beta^i$ as an approximation. This is why in the expression above the multiplier before ε^i is $\sqrt{\beta^i}$.

Similar to QSM, the critic loss can now be defined as follows:

$$L(\phi) = \mathbb{E}_{z\sim\mathcal{N}}\left[(Q_\phi(s,a) - r - Q_{\bar{\phi}}(s', a' \sim \pi_{f_\theta}(\cdot \mid s')))^2\right], \tag{57}$$

where π_{f_θ} is the policy that generates actions using the procedure above in Equation (55) and Equation (56). 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_{f_θ}.

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 Lipman et al. (2024)):

$$v_\phi(s,a,u) := \frac{(1-u)\tau\nabla_a Q^u_\phi(s,a) + a}{u}, \tag{58}$$

where

$$Q^u_\phi(s,a) := \frac{1}{\tau}\log \mathbb{E}_{z\sim\mathcal{N}(0,I_A)}\left[e^{\tau Q_\phi(s,(1-u)z+ua)}\right], \tag{59}$$

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 (Dhariwal & Nichol, 2021). In our setting this translates to

$$\hat{v}_\phi(s,a^u,u) := \frac{(1-u)\tau\nabla_a Q_\phi(s,a^u) + a^u}{u}. \tag{60}$$

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 $u \in [0,1]$. Finally, we add the velocity field defined by Q_ϕ directly to the behavior cloning velocity field to form our policy:

$$v = v_\beta + \vartheta \hat{v}_\phi, \tag{61}$$

where v_β is trained with the standard flow-matching loss (i.e., $L(\beta) = L_{\text{FM}}(\beta)$) and ϑ is coefficient that modulates influence of the guidance that we find to be helpful (*e.g.*, $\vartheta < 1$ often works better than $\vartheta = 1$). The critic loss uses π_v to backup the target Q-value:

$$L(\phi) = \mathbb{E}_{z\sim\mathcal{N}}\left[(Q_\phi(s,a) - r - Q_{\bar\phi}(s', \text{ODE}(v(s',\cdot,\cdot),z))^2\right]. \tag{62}$$

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 $\hat{v}_\phi$.

CGQL-MSE/Linex. Alternatively, we can approximate Q^u_ϕ in Equation (59) more closely with a training objective. In particular, we explore the following two regression objectives:

$$\textbf{MSE:}\quad L_{\text{MSE}}(\zeta) = \mathbb{E}_{u\sim\mathcal{U}[0,1],z\sim\mathcal{N}}\left[\left(\hat{Q}^u_\zeta(s,a^u) - Q_\phi(s,a)\right)^2\right] \tag{63}$$

$$\textbf{Linex:}\quad L_{\text{Linex}}(\zeta) = \mathbb{E}_{u\sim\mathcal{U}[0,1],z\sim\mathcal{N}}\left[\exp(\tau(Q_\phi(s,a) - \hat{Q}^u_\xi(s,a^u))) + \tau\hat{Q}^u_\xi(s,a^u)\right] \tag{64}$$

where $a^u := (1-u)z + ua$. The optimal solution of $\hat{Q}^u_\zeta$ in Equation (63) is not exactly the same as the desired Q^u_ϕ in Equation (59) but constitutes a lower-bound due to Jensen's inequality:

$$Q^{u\star}_{\text{mse}} = \mathbb{E}_{z,u}\left[Q_\phi(s,a^u)\right] \le \frac{1}{\tau}\log \mathbb{E}_{z\sim\mathcal{N}(0,I_A)}\left[e^{\tau Q_\phi(s,a^u)}\right]. \tag{65}$$

The second objective resembles the classic Linex objective (Parsian & Kirmani, 2002) where the optimal solution of $\hat{Q}^u_\zeta$ in Equation (64) is the same as the desired Q^u_ϕ in Equation (59):

$$Q^{u\star}_{\text{linex}} = \frac{1}{\tau}\log \mathbb{E}_{z\sim\mathcal{N}(0,I_A)}\left[e^{\tau Q_\phi(s,(1-u)z+ua)}\right]. \tag{66}$$

A discussion on this can be found in Myers et al. (2025). For completeness, we show this below. Without loss of generality, we just need to show that $\exp(y) = \mathbb{E}[\exp(x)] = \int p(x)\exp(x)\mathrm{d}x$ is the unique minimum for the following loss function:

$$L(y) = \int p(x)\left[\exp(x-y) + y\right]\mathrm{d}x \tag{67}$$

Taking the first and second derivatives with respect to y gives

$$\frac{\mathrm{d}L}{\mathrm{d}y} = -\exp(-y)\mathbb{E}[\exp(x)] + 1 \tag{68}$$

$$\frac{\mathrm{d}^2L}{\mathrm{d}y^2} = \exp(-y)\mathbb{E}[\exp(x)] \tag{69}$$

Setting Equation (68) gives $y = \log\mathbb{E}[\exp(x)]$ at which the second derivative is 1 (*i.e.*, $\frac{\mathrm{d}^2L}{\mathrm{d}y^2} = 1$). Furthermore, to prevent the exponential blow up of $\exp(Q_\phi(s,a) - \hat{Q}^u_\xi(s,a^u))$, we follow Myers et al. (2025) to use a Huber-style loss that locally behaves like a Linex loss but with a linear penalty when the exponential term is too large. In particular, we use the following loss:

$$L_{\text{Linex}^+}(\zeta) = \begin{cases} \mathbb{E}\left[\exp(\Delta) + \tau\hat{Q}^u_\xi(s,a^u)\right], & \Delta > 5 \\ \mathbb{E}\left[\Delta\right], & \Delta \le 5 \end{cases} \tag{70}$$

where $\Delta = \tau(Q_\phi(s,a) - \hat{Q}^u_\xi(s,a^u))$.

In practice, however, both the MSE and Linex objectives ($L_{\text{Linex}^+}, L_{\text{MSE}^+}$) can still be unstable and exhibit high variances. Instead of learning $\hat{Q}^u_\zeta$ and Q_ϕ separately, we find it is often better to directly learn both of them in a single network $Q_\phi(s,a,u) : \mathcal{S}\times\mathcal{A}\times[0,1] \to \mathbb{R}$ as follows:

$$\begin{aligned} L(\phi) =& \mathbb{E}_{z\sim\mathcal{N}}\left[(Q_\phi(s,a,1) - r - Q_{\bar\phi}(s', \text{ODE}(v(s',\cdot,\cdot),z))^2\right] + \\ & \varrho\mathbb{E}_{u\in\mathcal{U}[0,1],z\sim\mathcal{N}}\left[(Q_\phi(s,(1-u)z+ua,u) - Q_{\bar\phi}(s,a,1))^2\right], \end{aligned} \tag{71}$$

where ϱ is the coefficient that balances the TD backup for $Q_\phi(s,a,1)$ and the noisy target regression for $Q_\phi(s,a,u<1)$. With $Q_\phi(s,\cdot,u)$, we can define our velocity field (which is also used as the TD backup above) as

$$v := v_\beta + \vartheta \hat{v}_\phi, \quad \text{where } \hat{v}_\phi(s,a^u,u) := \frac{(1-u)\tau\nabla_a Q_\phi(s,a^u,u) + a^u}{u}, \tag{72}$$

and ϑ again modulates the guidance strength.

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

FAWAC is a baseline considered in FQL (Park et al., 2025b) 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:

$$L_{\text{FAWAC}}(\theta) = \tilde{w}(s,a)L_{\text{FM}}(\theta) \tag{73}$$
$$= \tilde{w}(s,a)\mathbb{E}_{u\sim\mathcal{U}[0,1],z\sim\mathcal{N}}\left[\|v_\theta(s,(1-u)z+ua,u) - z + a\|_2^2\right] \tag{74}$$

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:

$$L(\phi) = \mathbb{E}_{z\sim\mathcal{N}}\left[(Q_\phi(s,a) - r - Q_{\bar{\phi}}(s', \text{ODE}(v(s',\cdot,\cdot),z))^2\right] \tag{75}$$
$$L(\xi) = \mathbb{E}_{z\sim\mathcal{N}}\left[(V_\xi(s) - r - Q_{\bar{\phi}}(s', \text{ODE}(v(s',\cdot,\cdot),z))^2\right] \tag{76}$$

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:

$$L_{\text{FEdit}}(\theta,\omega) = L_{\text{FM}}(\theta) + L_{\text{Gaussian}}(\omega), \quad \text{s.t.} \quad \mathbb{E}_{s\sim D}\left[\mathbb{H}(\pi_\omega(\cdot\mid s))\right] \geq H_{\text{target}} \tag{77}$$

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

$$L_{\text{Gaussian}}(\omega) = \mathbb{E}_{\Delta a\sim\pi_\omega(\cdot|s,\tilde{a}),z\sim\mathcal{N}}\left[-Q_\phi(s,\text{Clip}\left[\sigma_a\cdot\Delta a + \tilde{a}\right]_{-1}^{1})\right] \tag{78}$$

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:

$$L(\phi) = \mathbb{E}_{z\sim\mathcal{N},\Delta a'\sim\pi_\omega(\cdot|s',\tilde{a}')}\left[(Q_\phi(s,a) - r - Q_{\bar{\phi}}(s',\text{Clip}\left[\tilde{a}' + \sigma_a\cdot\Delta a'\right]_{-1}^{1})^2\right] \tag{79}$$

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 $\tilde{a} \leftarrow \mathrm{ODE}(v_\theta(s, \cdot, \cdot), z)$ 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:

$$L_{\mathrm{DSRL}}(\theta, \omega) = L_{\mathrm{FM}}(\theta) + L_{\mathrm{LatentGaussian}}(\omega), \quad \text{s.t.} \quad \mathbb{E}_{s\sim D}\left[\mathbb{H}(\pi_\omega(\cdot \mid s))\right] \geq H_{\mathrm{target}} \tag{80}$$

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

$$L_{\mathrm{LatentGaussian}}(\omega) = \mathbb{E}_{z\sim\pi_\omega(\cdot|s)}\left[-Q^z_\psi(s, z)\right] \tag{81}$$

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

$$L(\psi) = \mathbb{E}_{z\sim\mathcal{N}}\left[(Q^z_\psi(s, z) - Q_\phi(s, \mathrm{ODE}(v_{\bar{\theta}}(s, \cdot, \cdot), z))^2\right]. \tag{82}$$

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

$$L(\phi) = \mathbb{E}_{z\sim\pi_\omega(\cdot|s')}\left[(Q_\phi(s, a) - r - Q_{\bar{\phi}}(s', \mathrm{ODE}(v_{\bar{\theta}}(s', \cdot, \cdot), z))^2\right] \tag{83}$$

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 $a \leftarrow \mathrm{ODE}(v_\theta(s, \cdot, \cdot), z)$. 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., 2025b) 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:

$$L(\phi) = (Q_\phi(s, a) - r - V_\xi(s'))^2 \tag{84}$$

$$L(\xi) = f^\kappa_{\exp}(Q_{\bar{\phi}}(s, a) - V_\xi(s)) \tag{85}$$

where $f^\kappa_{\exp}(u) = |\kappa - \mathbb{I}_{u<0}|u^2$ 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:

$$a^\star \leftarrow \arg\max_{a_1,\cdots,a_N} Q(s, a_i), \quad a_1, \cdots, a_N \sim \pi_{v_\beta}(\cdot \mid s). \tag{86}$$

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).

ReBRAC (Tarasov et al., 2024) is a strong offline RL method that trains a TD3 (Fujimoto et al., 2018) agent with behavior cloning loss. In practice, we find two hyperparameters to impact performance the most. The first hyperparameter is α which controls the strength of the behavior cloning loss. The second hyperparameter is σ which controls the magnitude of the Gaussian noise added in the TD3 policy. We keep the action noise clip to be 0.5 and an actor delay of 2, following the original paper.

C.2 CHARACTERIZATION BY BEHAVIOR CONSTRAINT SHAPE

Reversed KL. The reversed KL constraint is the most commonly used constraint in offline RL and offline-to-online RL. It amounts to the following constraint on the learned policy π:

$$D_{\mathrm{KL}}(\pi(\cdot \mid s) \parallel \pi_\beta(\cdot \mid s)) = \int \pi_\theta(a \mid s)(\log \pi_\beta(a \mid s) - \log \pi(a \mid s))\mathrm{d}a \tag{87}$$

When combined with the Q-maximization objective,

$$L(\pi) = -\mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q(s,a)\right] + \beta(s) D_{\mathrm{KL}}(\pi \parallel \pi_\beta), \tag{88}$$

the optimal policy admits the following closed-form solution:

$$\pi^\star(\cdot \mid s) \propto \pi_\beta(\cdot \mid s)\exp(\tau(s)Q(s,\cdot)) \tag{89}$$

where $\tau(s)$ depends on $\beta(s)$ and the magnitude of Q.

In practice, we often use a state-independent τ for simplicity:

$$\pi^\star(\cdot \mid s) \propto \pi_\beta(\cdot \mid s)\exp(\tau Q(s,\cdot)). \tag{90}$$

Our method, `QAM`, and many of our baselines, `FAWAC`, `CGQL*`, `DAC`, `QSM` approximate $\pi^\star$ in different ways. `FAWAC` uses advantage-weighted behavior cloning (flow-matching loss since we are using flow policies) like the following:

$$L_{\mathrm{FAWAC}}(\pi) = \mathbb{E}_{a\sim\mathcal{D}}\left[\exp(Q(s,a) - V(s))L^{\pi}_{\mathrm{FM}}(s,a)\right] \tag{91}$$

such that the policy converges to the desired optimal policy at the optimum:

$$\pi^\star(\cdot \mid s) \propto \pi_\beta(\cdot \mid s)\exp(\tau(Q(s,\cdot) - V(s))) \propto \pi_\beta(\cdot \mid s)\exp(\tau Q(s,\cdot)). \tag{92}$$

The value function does not change the optimum but is important to reduce the learning variance.

`CGQL*`, `DAC`, `QSM` are based on the observation that

$$\nabla_a \log \pi^\star(a \mid s) = \nabla_a \log \pi_\beta(a \mid s) + \tau\nabla_a Q(s,a) \tag{93}$$

make the approximation that

$$\nabla_{a^u} \log \pi^{u\star}(a^u \mid s) \approx \nabla_{a^u} \log \pi^u_\beta(a^u \mid s) + \tau\nabla_{a^u} Q(s,a^u) \tag{94}$$

where a^u is the intermediate noisy action at some denoising time u and $\pi^u, \pi^{u\star}$ are the distributions of the noisy actions from the behavior and optimal respectively from the corresponding time u.

In contrast to these prior methods, `QAM` does not rely on approximations while leveraging $\nabla_a Q(s,a)$ to align the flow-matching policy directly to $\pi^\star$.

In general, reversed KL behavior constraint respects the support of the behavior distribution π_β the best because at the optimum, $\pi^\star$ cannot assign non-zero probability to any action a that is outside the support of π_β (*i.e.*, $\pi_\beta(a \mid s) = 0$). The geometry-agnostic nature of the KL divergence, in principle, allows these methods to deal with any behavior distribution and value function landscapes. This is in contrast to geometry-aware methods which we will discuss below.

Wasserstein distance. Another rising class of behavior constraint is the Wasserstein distance constraint. It amounts to the following constraint:

$$W_p(\pi, \pi_\beta) = \min_{P\in\mathcal{C}(\pi,\pi_\beta)} \mathbb{E}_{(a^\pi, a^\beta\sim P)}\left[d(a^\pi, a^\beta)^p\right]^{1/p} \tag{95}$$

where $d : \mathcal{A}\times\mathcal{A}$ is some distance metric that is commonly picked to be L_2 or L_∞.

With $d(a, a') = \|a - a'\|_2$ and $p = 2$, `FQL` (Park et al., 2025b) optimizes the following objective

$$L(\pi) = -\mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q(s, a)\right] + \alpha W_2(\pi_\beta, \pi)^2 \tag{96}$$

where α is the behavior regularization strength.

Unlike reversed KL behavior constraint, the Wasserstein behavior constraint does not necessarily respect the support constraint and can assign high probability mass to an action as long as it is close to some behavior actions in terms of d.

Loosely speaking, `FEdit` also imposes the behavior constraint in a similar spirit but with a non-metric d. In particular, it can be interpreted as using

$$d(a^\pi, a^\beta) = \mathbb{I}\left[\|a^\pi - a^\beta\|_\infty > \sigma_e\right] \tag{97}$$

with $\alpha \to \infty$. This is equivalent to parameterizing $\pi(\cdot \mid s)$ as $\tilde{a} \leftarrow \pi_{\text{edit}}(s, a \sim \pi_\beta(\cdot \mid s))$ where a residual edit policy $\pi_{\text{edit}} : \mathcal{S} \times \mathcal{A} \to \mathcal{A}$ is used to modify the original behavior policy up to σ_e (in terms of the infinity norm of the action difference):

$$L(\pi) = -\mathbb{E}_{a\sim\pi_\beta(\cdot|s)}\left[Q(s, \pi_{\text{edit}}(s, a))\right], \quad \text{s.t.} \quad \|\pi_{\text{edit}}(s, a) - a\|_\infty \le \sigma_e \tag{98}$$

Though strictly speaking it is no longer a Wasserstein distance as d is no longer a metric, it is still a valid behavior constraint that shares a similar geometry property as the Wasserstein behavior constraint–it does not respect the support of π_β and can assign large probability mass that is close to the behavior action in terms of d.

While not respecting the support constraint might seem to be a bad thing, these methods, conceptually, may attain a better performance compared to the reversed KL methods, especially when the Q-function generalizes well beyond the behavior support.

Behavior support constraint. The behavior constraint shape in our discussion is the support constraint. This is similar to the reversed KL constraint but with a more uniform weight over the actions in the support:

$$D_{\text{supp}}(\pi \parallel \pi_\beta) = \int \pi_\theta(a \mid s)\mathbb{I}\left[\pi_\beta(a \mid s) > 0\right] \mathrm{d}a. \tag{99}$$

Compared to the reversed KL constraint, which up-weights actions that have higher probability in the behavior distribution, the behavior support constraint puts a uniform weight over all actions that are in the support. While achieving this exactly is difficult, a line of work (`DSRL` (Wagenmaker et al., 2025) and Singh et al. (2020)) can be seen as an approximation of this by learning a policy in the noise-space of a diffusion/flow policy. To approximate the support of the diffusion/flow policy, it uses a L_∞-bounded action space in the noise space $a_z \in [-\sigma_z, \sigma_z]$ and then optimize the following objective:

$$L(\pi) = -\mathbb{E}_{a_z\sim\pi(\cdot|s)}\left[Q(s, a = F_\beta(s, a_z))\right], \tag{100}$$

where $F_\beta : \mathcal{S} \times \mathcal{Z} \to \mathcal{A}$ is the behavior policy that maps from a noise space to the action space. There are many choices of F_β (*e.g.*, Singh et al. (2020) use normalizing flow (Dinh et al., 2016)) and in our experiments, we use `DSRL`, which uses a flow-matching policy. One practical trick that makes DSRL work is to use a distilled Q-network that directly estimates the action value in the noise space. We refer the reader to the original paper for more details. We describe the implementation details regarding the distilled Q-network in Appendix C.1.

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. To pick the hyperparmaeter for each domain for each method, we first run a sweep over all hyperparameters in the range (specified by Table 5), or all combinations of them if there are multiple hyperparameters involved. The hyperparameter tuning runs use 4 seeds for each method for each hyperparameter configuration for each of the two tasks per domain. For locomotion domains, we use task 1 (the default task) and task 4. For manipulation domains, we use task 2 (the default task) and task 4. We use task 4 in addition to the default task recommended by OGBench because we often find the combination of these two cover the characteristics of each domain better.

We then use the combined performance of the two tuning tasks per domain per method to pick the hyperparameter configuration. We include them in Table 4. Finally, for all our main results, we run all methods on all five tasks for each domain on 8 *new* seeds (different from the tuning seeds). We pick the hyperparmeter range such that the total number of tuning runs are similar across methods. To achieve this, we use the following strategy: For methods where more than one hyperparameter needs to be tuned, we use a coarser hyperparmeter range. For methods where there is only one hyperparmeter, we use a more fine-grained sweep. The only exception is our method `QAM` where we find 4 tuning hyperparameters (*i.e.*, $\{1, 3, 10, 30\}$) are enough to outperform all prior methods.

Parameter	**Value**
Batch size	256
Discount factor (γ)	0.99 for `{puzzle/scene/cube/antmaze-large}-*` 0.995 for `{humanoidmaze/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 for `{puzzle/scene/cube/antmaze}-*` 0 for `humanoidmaze-*`
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
Optimizer Gradient clipping	1 for `QSM`, `DAC` and `QAM*`. No clipping for others.

Table 3: **Common hyperparameters.**

Domains	`ReBRAC`	`QSM`	`DAC`	`FBRAC`	`FQL`	`DSRL`	`FEdit`	`IFQL`	`CGQL`	`CGQL-MSE`	`CGQL-Linex`	`QAM & BAM`	`QAM-Edit`	`QAM-FQL`
	(α, σ)	(τ, η)	η	α	α	σ_z	σ_a	τ	(ϑ, τ)	$(\vartheta, \varrho, \tau)$	$(\vartheta, \varrho, \tau)$	τ	(τ, σ_a)	(τ, α)
`scene-sparse-*`	$(0.03, 0)$	$(3, 30)$	1	100	300	0.4	0.2	0.9	$(10, 0.1)$	$(10, 0.1, 0.1)$	$(10, 0.1, 0.1)$	1	$(1, 0)$	$(1, \infty)$
`puzzle-3x3-sparse-*`	$(0.1, 0)$	$(10, 30)$	1	0.3	300	1	0.2	0.95	$(10, 0.1)$	$(10, 0.1, 0.1)$	$(10, 0.001, 0.1)$	3	$(3, 0.1)$	$(3, \infty)$
`puzzle-4x4-100M-sparse-*`	$(0.01, 0.2)$	$(10, 1)$	1	0.3	1	1	0.8	0.9	$(10, 0.1)$	$(10, 0.001, 1)$	$(10, 0.001, 1)$	30	$(3, 0.9)$	$(1, 3)$
`cube-double-*`	$(0.01, 0)$	$(1, 10)$	3	0.1	300	1	0.2	0.9	$(10, 0.01)$	$(10, 0.001, 0.01)$	$(10, 0.001, 0.01)$	1	$(1, 0)$	$(1, 0)$
`cube-triple-*`	$(0.01, 0.2)$	$(10, 10)$	0.3	0.03	30	1.4	0.3	0.95	$(10, 0.1)$	$(10, 0.001, 0.1)$	$(10, 0.001, 0.1)$	10	$(10, 0.1)$	$(10, 30)$
`cube-quadruple-100M-*`	$(0.01, 0.2)$	$(1, 10)$	1	1	100	1.4	0.4	0.95	$(10, 0.1)$	$(10, 0.1, 0.01)$	$(10, 0.1, 0.01)$	1	$(3, 0.5)$	$(1, 30)$
`antmaze-large-*`	$(0.01, 0)$	$(10, 10)$	0.3	0.1	3	0.8	0.2	0.9	$(10, 0.1)$	$(10, 0.1, 0.1)$	$(10, 0.001, 0.1)$	10	$(1, 0.1)$	$(3, 30)$
`antmaze-giant-*`	$(0.01, 0)$	$(10, 10)$	0.3	0.1	3	1.2	0.3	0.8	$(10, 0.1)$	$(10, 0.001, 0.1)$	$(10, 0.001, 0.1)$	3	$(3, 0.1)$	$(3, 300)$
`humanoidmaze-medium-*`	$(0.01, 0)$	$(10, 30)$	1	30	30	0.6	0.5	0.7	$(10, 0.01)$	$(10, 0.1, 0.1)$	$(10, 0.1, 0.1)$	3	$(1, 0.1)$	$(3, \infty)$
`humanoidmaze-large-*`	$(0.01, 0.1)$	$(10, 30)$	0.3	10	30	0.8	0.1	0.8	$(10, 0.01)$	$(10, 0.1, 0.1)$	$(10, 0.1, 0.1)$	3	$(1, 0.1)$	$(1, 30)$

Table 4: **Domain-specific hyperparameters.** The best hyperparameter configuration obtained from our tuning runs. We use the same hyperparameter configuration for all tasks in each domain.

E PROOF OF PROPOSITION 1

Proposition 1 (Extension of Proposition 7 in Domingo-Enrich et al. (2025) to Policy Optimization.) Take $L_{\text{AM}}(\theta)$ in Equation (14), there is a unique f_θ such that

$$\frac{\partial}{\partial f_\theta} L_{\text{AM}} = 0, \tag{27}$$

and for all $s \in \text{supp}(D)$,

$$\pi_\theta(\cdot \mid s) \propto \pi_\beta(\cdot \mid s) e^{\tau Q_\phi(s,a)}. \tag{28}$$

Proof. Our proof mainly rewrites the assumptions and the statement of Proposition 7 of Domingo-Enrich et al. (2025) in our notations with a simple extra step that extends it to the state-conditioned version (*e.g.*, for policy optimization).

Method	Hyperparameter(s)	Sweep Range
ReBRAC	(α, σ)	$(\{0.01, 0.03, 0.1, 0.3, 1\}, \{0, 0.1, 0.2\})$
FBRAC	α	$\{0.03, 0.1, 0.3, 1.0, 3.0, 10.0, 30.0, 100.0\}$
CGQL	(ϑ, τ)	$(\{0.01, 0.1, 1\}, \{0.1, 1, 10\})$
CGQL-{MSE, Linex}	$(\vartheta, \tau, \varrho)$	$(\{0.01, 0.1, 1\}, \{0.1, 1, 10\}, \{0.001, 0.1\})$
FQL	α	$\{0.3, 1, 3, 10, 30, 100, 300, 1000\}$
DSRL	σ_z	$\{0.1, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4\}$
FEdit	σ_a	$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$
FAWAC	τ	$\{0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 12.8\}$
IFQL	κ	$\{0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$
DAC	η	$\{0.1, 0.3, 1, 3.0, 10, 30, 100, 300\}$
QSM	(τ, η)	$(\{1.0, 3.0, 10.0, 30\}, \{1, 3, 10, 30\})$
QAM	τ	$\{1, 3, 10, 30\}$
QAM-Edit	(τ, σ_a)	$(\{1, 3, 10\}, \{0, 0.1, 0.5, 0.9\})$
QAM-FQL	(τ, α)	$(\{1, 3, 10\}, \{3, 30, 300, \infty\})$

Table 5: **Domain-specific hyperparameter tuning range.** For QAM-FQL, $\alpha = \infty$ is implemented by QAM. Similarly, for QAM-FQL, $\sigma_a = 0$ is implemented by QAM. For methods with more than one parameter, we tune all possible combinations within the sweep range. For example, for CGQL, we sweep over all $3 \times 3 = 9$ configurations with 3 possible values of ϑ and 3 possible values of τ.

We first define (from Equation 27 in Domingo-Enrich et al. (2025)) the residual

$$v_{\text{res}}(s, a^u, u) := \sqrt{\frac{2}{\beta_u(\beta_u \dot{\alpha}_u/\alpha_u - \dot{\beta}_u)}}(v_\theta(s, a^u, u)). \tag{101}$$

While the original result showed for a more general case with a family of α, β (and later on σ), in our work we assume

$$\alpha(u) = u \tag{102}$$
$$\beta(u) = 1 - u \tag{103}$$
$$\sigma(u) = \sqrt{2(1-u)/u} \tag{104}$$

This allows us to simplify the expression for v_{res} as

$$v_{\text{res}}(s, a^u, u) = \sqrt{\frac{2u}{1-u}} v_\theta(s, a^u, u) \tag{105}$$

Then, we restate the definition of b (from Equation 27 in Domingo-Enrich et al. (2025)):

$$\begin{aligned} b(s, a^u, u) &= 2(v_\beta(s, a^u, u) + v_\theta(s, a^u, u)) - \frac{\dot{\alpha}_u}{\alpha_u} a^u - \sigma(u) v_{\text{res}}(s, a^u, u) && (106) \\ &= 2(v_\beta(s, a^u, u) + v_\theta(s, a^u, u)) - a^u/u - 2v_\theta(s, a^u, u) && (107) \\ &= 2v_\beta(s, a^u, u) - a^u/u && (108) \end{aligned}$$

We can now rewrite our 'lean' adjoint state definition as

$$\mathrm{d}\tilde{g}^u = -\tilde{g}^{u\top} \nabla_{a^u} [b(s, a^u, u)], \quad \tilde{g}^1 = -\tau \nabla_{a^1} Q_\phi(s, a^1) \tag{109}$$

which coincides with the definition in Equation 38 in Domingo-Enrich et al. (2025), with $f = 0$ and $g(\cdot) = -\tau Q_\phi(s, \cdot)$). Now, we can rewrite our adjoint matching objective as

$$L_{\text{AM}}(\theta) := \mathbb{E}_{s \sim D, \{a^u\}_u} \left[\tilde{L}_{\text{AM}}(v_\theta, \{a^u\}_u) \right] \tag{110}$$

where

$$\tilde{L}_{\mathrm{AM}}(s, v_\theta, \{a^u\}_u) := \int_0^1 \|v_{\mathrm{res}}(s, a^u, u) + \sigma(u)\tilde{g}^u\| \,\mathrm{d}u \tag{111}$$

$$= \frac{1}{2}\int_0^1 \left\| \sqrt{\frac{2u}{1-u}} v_\theta(s, a^u, u) + \sigma(u)\tilde{g}^u \right\|_2^2 \mathrm{d}u \tag{112}$$

$$= \frac{1}{2}\int_0^1 \|2v_\theta(s, a^u, u)/\sigma(u) + \sigma(u)\tilde{g}^u\|_2^2 \,\mathrm{d}u \tag{113}$$

Comparing $\tilde{L}_{\mathrm{AM}}(s, v_\theta, \{a^u\}_u)$ to the definition in Equation 37 in Domingo-Enrich et al. (2025), they are different by only a factor of 2 *conditioned on a fixed* s. Thus, their critical points are the same.

By triggering Proposition 7 in Domingo-Enrich et al. (2025), we can conclude that for a fixed s, the only critical point of the following loss function,

$$\mathbb{E}_{\{a^u\}_u}\left[\tilde{L}_{\mathrm{AM}}(s, v_\theta, \{a^u\}_u)\right], \tag{114}$$

is $v^\star_{\mathrm{res}}(s, a^u, u)$, the velocity field that generates the following distribution,

$$\pi^\star(\cdot \mid s) \propto \pi_\beta(\cdot \mid s)\exp(\tau Q_\phi(s, \cdot)). \tag{115}$$

Finally, since the $L_{\mathrm{AM}}(\theta)$ is a linear combination of $\mathbb{E}_{\{a^u\}_u}\left[\tilde{L}_{\mathrm{AM}}(s, v_\theta, \{a^u\}_u)\right]$ over different s, the critic point of $L_{\mathrm{AM}}(v_\theta)$ is simply the cartesian product of over the critic points for each $s \in \mathrm{supp}(D)$. Since there is only one critical point for each $\mathbb{E}_{\{a^u\}_u}\left[\tilde{L}_{\mathrm{AM}}(s, v_\theta, \{a^u\}_u)\right]$, $L_{\mathrm{AM}}(v_\theta)$ also has only one critical point and coincides with $v^\star_{\mathrm{res}}(s, a^u, u)$. This concludes that the only critical point of $L_{\mathrm{AM}}(v_\theta)$ results in velocity fields that satisfy

$$\pi^\star(\cdot \mid s) \propto \pi_\beta(\cdot \mid s)\exp(\tau Q_\phi(s, \cdot)), \quad \forall s \in \mathrm{supp}(D). \tag{116}$$

□

Method	Training Speed (milliseconds/step)	Parameter Count
`ReBRAC`	2.97	18 238 534
`FBRAC`	4.54	17 426 989
`CGQL`	9.62	18 275 398
`CGQL-{MSE, Linex}`	10.40	18 275 398
`FQL`	3.58	18 264 646
`DSRL`	5.63	27 397 251
`FEdit`	3.77	18 277 472
`FAWAC`	3.44	18 243 630
`IFQL`	2.66	18 243 630
`DAC`	4.44	18 509 901
`QSM`	4.44	18 509 901
`QAM`	5.83	19 941 496
`QAM-Edit`	6.61	20 791 979
`QAM-FQL`	6.36	20 779 153

Table 6: **Training speed and parameter count for each method on `cube-triple`.**

F Additional results

In this section, we analyze the performance of our method, QAM, when subject to data of different equalities. Figure 5 reports the performance of our method on `cube-quadruple-task2` with different dataset sizes. Figure 6 reports the comparison of our method with representative baselines (FQL, ReBRAC and FEdit) on `cube-double-noisy-task4` and `cube-triple-noisy-task4`, two of the hardest tasks in the benchmark. Figure 7 contains additional results that aggregate over all 5 tasks for each domain.

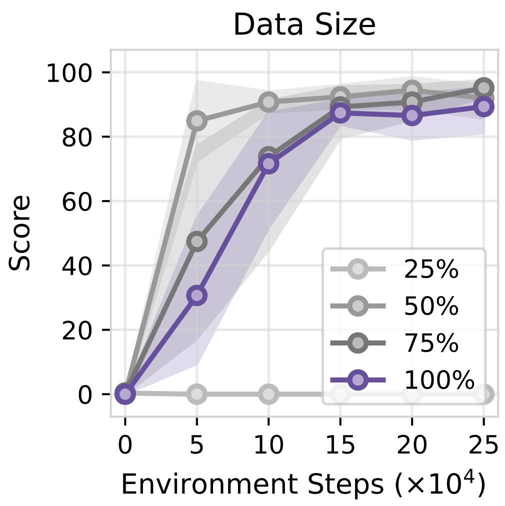

Figure 5: **Dataset size analysis on `cube-quadruple-task2` (6 seeds).** 100%, 75%, 50%, 25% corresponds to a dataset size of 100M, 75M, 50M, and 25M. Our method can also work on dataset with smaller size until 25M-size dataset where it completely fails.

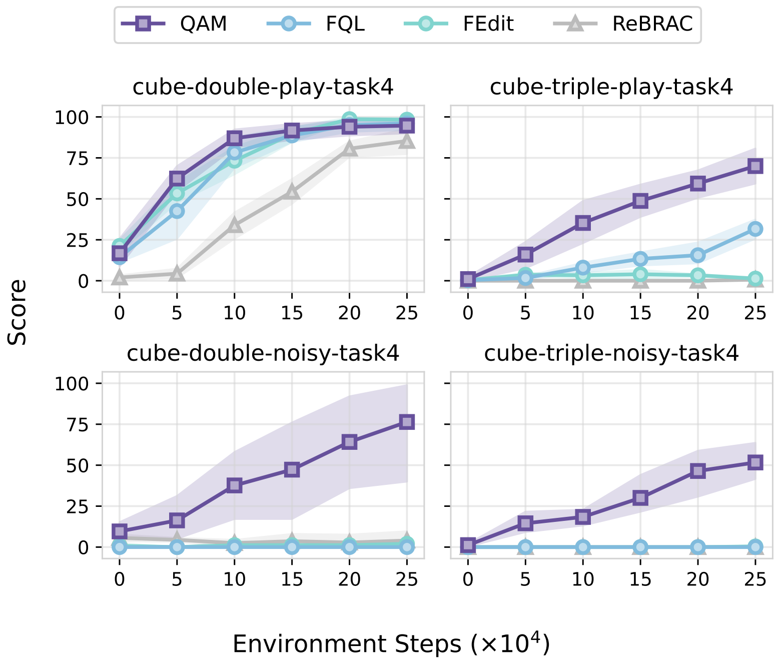

Figure 6: **Experiments on `cube-double-task4` and `cube-triple-task4` on play and noisy datasets (6 seeds).** *Top:* the original `play`-style datasets we use in our main experiments; *Bottom:* the `noisy`-style datasets that were collected by expert policies with large uncorrelated, Gaussian action noises. We use the `QAM-EDIT` variant as it is the best-performing variant in our offline RL experiments. We use the same hyperparameters for experiments on `noisy`-style and `play`-style datasets. Our method exhibits strong robustness against action noises. While all our baselines fail completely on the `noisy` datasets, our method only suffers minor performance drop.

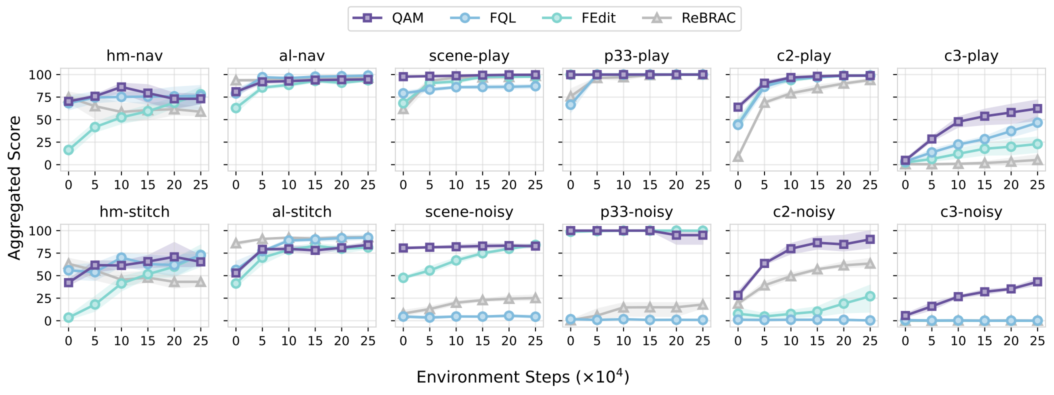

Figure 7: **Data quality analysis (4 seeds).** *Top:* performance on the original `{navigate/play}`-style datasets we use in our main experiments; *Bottom:* performance on the `{stitch/noisy}`-style datasets. `stitch`-style datasets were collected the same way as `navigate`-style datasets but with much shorter trajectory segments. `noisy`-style datasets were collected by expert policies with large uncorrelated, Gaussian action noises. We use the `QAM-EDIT` variant as it is the best-performing variant in our offline RL experiments. We use the same hyperparameters from our main experiments. Each subplot reports the aggregated score over 5 tasks in each domain. For locomotion tasks (*e.g.*, `humanoidmaze-medium` and `antmaze-large`), we observe almost no performance difference when switching from `navigate`-style to `stitch`-style dataset. For manipulation tasks, our method exhibits strong robustness against action noises. In particular, while all our baselines fail completely on the `cube-triple-noisy` environments, our method only suffers minor performance drop.