FLOW ACTOR-CRITIC FOR OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

The dataset distributions in offline reinforcement learning (RL) often exhibit complex and multi-modal distributions, necessitating expressive policies to capture such distributions beyond widely-used Gaussian policies. To handle such complex and multi-modal datasets, in this paper, we propose Flow Actor-Critic, a new actor-critic method for offline RL, based on recent flow policies. The proposed method not only uses the flow model for actor as in previous flow policies but also exploits the expressive flow model for conservative critic acquisition to prevent Q-value explosion in out-of-data regions. To this end, we propose a new form of critic regularizer based on the accurate proxy behavior model obtained as a byproduct of flow-based actor design. Leveraging the flow model in this joint way, we achieve new state-of-the-art performance for test datasets of offline RL including the D4RL and recent OGBench benchmarks.

1 INTRODUCTION

Offline Reinforcement Learning (RL) seeks an optimal policy from a pre-collected dataset without additional environment interactions, which may be costly or unsafe in real-world situations (Lange et al., 2012; Levine et al., 2020). Despite such advantages, offline RL suffers from value overestimation for out-of-distribution (OOD) actions due to its limited dataset coverage, which degrades performance. Many methods have been proposed to deal with this OOD value overestimation based on techniques such as constraining the target policy near the behavior policy of the dataset (Kumar et al., 2019; Fujimoto et al., 2019; Wu et al., 2022; Fujimoto & Gu, 2021; Nair et al., 2020) or suppressing the value function in the OOD region with penalization (Kumar et al., 2020; Mao et al., 2023). These methods have been shown to be effective for many offline RL tasks.

As offline RL datasets accumulate and grow diverse, however, their behavioral distributions become complicated and sometimes multi-modal. Thus, simple behavior policy modeling becomes insufficient and more expressive policies are required to model the behavior policy and its support. For example, diffusion policies adopted from the vision domain have garnered strong attention from the RL community as a candidate for expressive policies (Wang et al., 2022; Chen et al., 2022; Hansen-Estruch et al., 2023; Chen et al., 2023; Zhang et al., 2025). Diffusion policies learn the underlying data distribution using forward and reverse processes and can be trained efficiently by score matching with noise models (Ho et al., 2020; Song et al., 2020). They have shown impressive offline performance. Even with this advantage, diffusion models have a limitation that they can be used to construct an actor capable of sampling actions close to the behavior policy, but the iterative nature of multi-step processing of diffusion policies makes policy optimization computationally heavy.

Therefore, Park et al. (2025) recently adopted the *flow model* (Dinh et al., 2014; Rezende et al., 2014; Dinh et al., 2016) as an alternative for expressive policies. Although their work proposed one-step flow actor and demonstrated promising results, they did not exploit the full advantage of the flow policy. They only considered the actor design to maximize the conventional Q function under a constraint on the distance of the target policy from a flow-based behavior policy estimate, but overlooked the key fact that an accurate behavior density estimate itself is available with the flow model in contrast to diffusion models.

In this paper, we adopt the flow model and fully exploit it in a joint way to optimize an offline RL policy. We not only use the flow behavior policy model as the distance anchor of the target policy but also use it for Q function penalization in the OOD region by identifying the OOD region based on the

flow behavior density. Note that it has long been considered in offline RL that identifying the OOD region is difficult, and many previous works circumvent this difficulty with indirect methods (Kumar et al., 2020; Mao et al., 2023; Wu et al., 2022). However, the highly expressive flow behavior model and accompanying density provide us with a rare opportunity to directly tackle this problem. Our flow actor-critic (FAC) designed in such a way exhibits outstanding performance in many current offline benchmark tasks.

2 PRELIMINARIES

2.1 OFFLINE REINFORCEMENT LEARNING

An RL problem is formulated as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \rho_0, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the $d_{\mathcal{A}}$-dimensional action space, $P(s'|s,a)$ is the transition dynamics, ρ_0 is the initial state distribution, $r(s,a)$ is the reward function, and γ is the discount factor (Sutton et al., 1998). The goal of offline RL is to find a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted return $J(\pi_\theta) = \mathbb{E}_{\rho_0,P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$ from a static dataset $\mathcal{D} = \{(s, a, r, s')\}$ collected by an unknown underlying behavior policy β. For a policy π, the action value function is defined as $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a\right]$ and can be estimated by iterating the Bellman operator $\mathcal{T}^\pi Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a), a' \sim \pi(\cdot|s')}\left[Q(s', a')\right]$.

Support-Constrained Policy Optimization. The main challenge of offline RL is Q-value overestimation in the OOD action region outside the support of β (Kumar et al., 2020; Mao et al., 2023). A typical approach to this problem is policy optimization under a distance constraint between the target policy and the behavior policy (Fujimoto et al., 2019; Peng et al., 2019; Wu et al., 2019; 2022):

$$\max_{\pi} \ \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi}\left[Q(s,a)\right] \quad \text{s.t.} \ \ \mathbb{E}_{s \sim \mathcal{D}}[D(\pi(\cdot|s), \beta(\cdot|s))] < \delta, \tag{1}$$

where D is some distance or divergence measure. The rationale behind eq. (1) is that by restricting the distance of π from β, the action from π does not fall into the OOD region of the estimated Q, avoiding the overestimated region of Q. Practically, the hard constraint in eq. (1) is replaced with a regularization term, yielding the following policy optimization (Kumar et al., 2019; Fujimoto & Gu, 2021):

$$\max_{\pi} \ \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi}\left[Q(s,a)\right] - \lambda \, \mathbb{E}_{s \sim \mathcal{D}}\left[D\left(\pi(\cdot|s), \beta(\cdot|s)\right)\right], \tag{2}$$

where $\lambda > 0$ is a regularization coefficient. Note that eq. (2) can be viewed as the Lagrangian of the problem (1).

2.2 FLOW MODELS AND FLOW-BASED POLICIES

Normalizing flows are a popular framework for modeling complex data distributions (Dinh et al., 2014; Rezende et al., 2014). A continuous normalizing flow (CNF) is defined as a bijective function $\phi_u : \mathbb{R}^d \to \mathbb{R}^d$ (Chen et al., 2018), specified by a time-dependent vector field v_u via the 1st-order ordinary differential equation:

$$\frac{d}{du}\phi_u(x) = v_u(\phi_u(x)), \qquad \phi_0(x) = x, \tag{3}$$

yielding

$$x_u = \phi_u(x_0) = x_0 + \int_0^u v_t(x_t) dt. \tag{4}$$

Given data samples $x_1 \sim p_1(x)$ and a simple base distribution p_0 (e.g., standard normal), we want to learn the mapping ϕ_u such that $\phi_0(x_0) = x_0$ at time $u = 0$ is transported to $\phi_1(x_0) = x_1$ at time $u = 1$. Since $\phi_1(\cdot)$ is a deterministic function, the density of p_1 can easily be computed by the change of variables: $\log p_1(x_1) = \log p_0(x_0) - \log \left|\det\left(\frac{\partial \phi(x_0)}{\partial x_0}\right)\right|$. It is shown that the determinant of the Jacobian can be replaced with the divergence of the vector field v_u (Chen et al., 2018; Ben-Hamu et al., 2022). Thus, the data distribution density can be written as $\log p_1(x_1) = \log p_0(x_0) - \int_0^1 \nabla_{x_u} \cdot v_u(x_u) du$.

A recent efficient way to learn a flow for a set of pairs $\{(x_0, x_1)\}$ is flow matching (FM), not requiring complicated likelihood maximization (Lipman et al., 2022). FM learns the velocity field such that for

a pair (x_0, x_1), the instantaneous movement velocity vector $v_u(x_u)$ matches the overall displacement vector $x_1 - x_0$ along the points on the line connecting x_0 and x_1, i.e., $x_u = (1-u)x_0 + ux_1$. Thus, when the velocity vector field is parameterized with θ, the loss for learning θ is given by

$$\min_{\theta} \mathbb{E}_{u\sim\text{Unif}([0,1]),\, x_1\sim p_1(x),\, x_0\sim p_0(x)} \left[\|v_\theta(x_u; u) - (x_1 - x_0)\|_2^2\right], \tag{5}$$

Once v_θ is trained, samples from p_1 can be generated by solving eq. (4) with an integral solver by setting $u = 1$, and the density of the samples is available as aforementioned.

Flow-based Policy. Constructing a policy from the flow model is straightforward. One can define the flow function from the action space to the action space, depending on both time u and state s. Then, by pairing $z \sim \mathcal{N}(0, I)$ and action a (from the behavior policy), the corresponding velocity vector field parameterized with ψ is learned with the following loss (Lipman et al., 2022; Park et al., 2025):

$$\min_{\psi} \mathbb{E}_{(s,a)\sim\mathcal{D}, z\sim\mathcal{N}(0,I), u\sim\text{Unif}([0,1])} \left[\|v_\psi(\tilde{a}_u; s, u) - (a - z)\|_2^2\right], \tag{6}$$

where $\tilde{a}_u = (1-u)z + ua$. We refer to the so-learned flow policy with $(s, a) \sim \mathcal{D}$ as the *behavior proxy policy* $\hat{\beta}_\psi(\cdot|s)$. Then, the behavior proxy density is given by

$$\log \hat{\beta}_\psi(a|s) = \log p_0(\hat{z}) - \int_0^1 \nabla_{a_u} \cdot v_\psi(a_u; s, u) du. \tag{7}$$

One thing to note is that samples from a flow policy are stochastic samples for given s due to the initial condition $z \sim \mathcal{N}(0, I)$ even though the flow function itself is deterministic. Thus, we will use the notation $a(s, z)$ to show this dependency in the case of flow action, if necessary.

3 Motivation: Strength of Flow Behavior Proxy Density

Now let us investigate how well the flow behavior proxy policy $\hat{\beta}_\psi(\cdot|s)$ tracks the actual behavior policy β in terms of both sampling and density estimation. For this, we conducted an experiment with β being a 2-D Gaussian mixture with four modes, as shown in the leftmost plot in Fig. 1. For comparison, we considered four other behavior cloning (BC) models: simple Gaussian, conditional VAE (Kingma & Welling, 2013), and diffusion models with 10 and 50 denoising steps (Ho et al., 2020; Sohl-Dickstein et al., 2015; Wang et al., 2022). For the flow model, we used the Euler method with 10 steps as an integral solver for both sampling and density evaluation (Lipman et al., 2022; Chen et al., 2018). The results are shown in Fig. 1. In the case of VAE and diffusion models, we show the ELBO since the density is not directly available. (Details of the experiment are in Appendix D.)

As expected, simple Gaussian with single mode cannot distinguish the in-distribution (ID) region with its peak at the center of the four true modes on which the actual density is almost zero. VAE finds the four modes but generates noticeable samples outside the dataset support. Furthermore, amortized inference places probability density in low-density areas when the latent variable does not

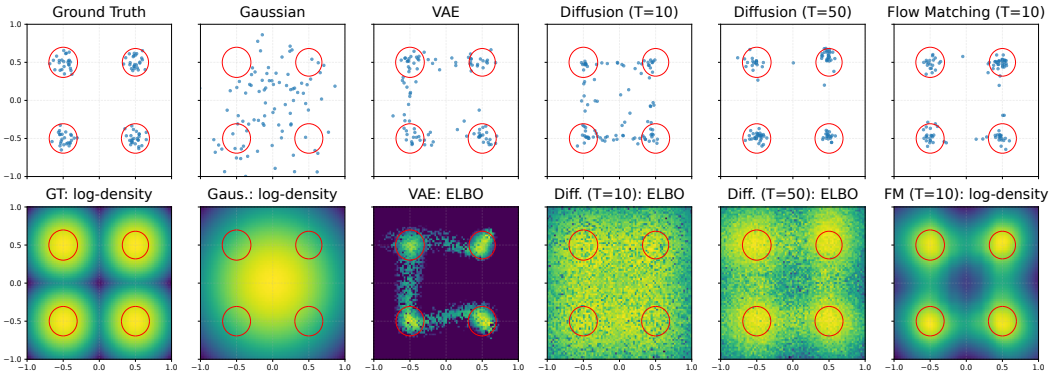


Figure 1: BC models on a synthetic four-component Gaussian mixture dataset: Top row - samples from each BC model. Bottom row - log-density or ELBO plot.

match the true posterior distribution, and hence the obtained ELBO is quite different from the true density as seen in the bottom row of the figure. The diffusion model with 50 denoising steps tends to generate samples within the in-distribution region, while the performance degrades with 10 denoising steps. Note that the obtained ELBO of density is bad, almost not separating the four modes to make it not suited for a density estimator, whereas the sampling performance is satisfactory. In contrast, the flow behavior proxy model shows good sampling performance and, importantly, yields a very strong estimate of the true density. Thus, the flow model can be used not only for an actor but also as a density estimator that can be used to distinguish the ID and OOD regions. In the following section, we present our algorithm to design both actor and critic, exploiting these two facts.

4 Method: Flow Actor-Critic

4.1 Critic Penalization with Flow Behavior Proxy Density

A typical way to prevent overestimation of Q values in the OOD region for offline RL is to learn a penalized Q function. CQL (Kumar et al., 2020) and SVR (Mao et al., 2023), two representative methods, use the following losses:

$$\min_Q \ \alpha \left(\mathbb{E}_{s\sim\mathcal{D},a\sim\pi}\left[Q(s,a)\right) - \mathbb{E}_{s\sim\mathcal{D},a\sim\beta}\left[Q(s,a)\right]\right) + \text{TD Loss}$$

and

$$\min_Q \alpha \left(\mathbb{E}_{s\sim\mathcal{D},a\sim\zeta}[(Q(s,a)-Q_{min})^2] - \mathbb{E}_{s\sim\mathcal{D},a\sim\beta}\left[\frac{\zeta(a|s)}{\beta(a|s)}(Q(s,a)-Q_{min})^2\right]\right) + \text{TD Loss},$$

respectively, where the TD loss is given by $\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\left(Q(s,a)-\mathcal{T}^\pi Q(s,a)\right)^2\right]$, and $\zeta(a|s)$ in SVR is a distribution covering the entire action range. It is known that the CQL penalization does not properly implement the correct Bellman operator in the ID region (Mao et al., 2023). SVR fixes this problem by using the difference of importance sampling (IS) integration when β does not dominate ζ. Although SVR circumvents direct OOD region identification intelligently using the IS technique, the main drawback is simple modeling of β, such as Gaussian density, required to compute the IS ratio. As seen in the previous section, when the true behavior distribution is complicated and multi-modal, Gaussian modeling places high non-zero density on regions with nearly zero actual density. In such cases, the IS ratio blows up incorrectly, and the performance can deteriorate severely.

With the flow behavior proxy density $\hat{\beta}_\psi$, a strong estimator for the true behavior density β at our hands, we can directly identify the ID and OOD regions and control the Q penalization term. To this end, we define the weight $w^{\hat{\beta}_\psi}(s,a)$ as

$$w^{\hat{\beta}_\psi}(s,a) = \max(0,\ 1-(\hat{\beta}_\psi(a|s)/\epsilon)) \quad \text{for some } \epsilon > 0. \tag{8}$$

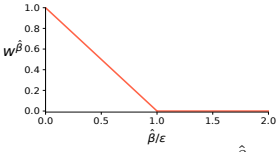


Figure 2: weight $w^{\hat{\beta}}$

The weight $w^{\hat{\beta}_\psi}(s,a)$ vanishes in the well-supported region with $\hat{\beta}_\psi(a|s) \geq \epsilon$, and increases linearly as $\hat{\beta}_\psi$ decreases below the threshold ϵ towards zero. The threshold ϵ is introduced because $\hat{\beta}_\psi$ is not perfect even though it is strong. We want to exclude the weak spurious density from the ID region but gradually. With this weight, we propose the following loss for critic learning:

$$\min_Q \ \alpha\, \mathbb{E}_{s\sim\mathcal{D},\ a\sim\pi}\left[w^{\hat{\beta}_\psi}(s,a)Q(s,a)\right] + \text{TD Loss}, \tag{9}$$

where $\alpha > 0$ controls the strength of penalization. In eq. (9), our penalization term is not in the form of some difference as in CQL or SVR, but directly pinpoints the OOD region based on the strong behavior density estimator $\hat{\beta}_\psi$. When $\hat{\beta}_\psi(a|s) \geq \epsilon$, i.e., $a \in \text{support}(\beta)$ with confidence, the penalization term disappears. However, the penalization term gradually kicks in as the confidence decreases below ϵ.

Proposition 1. *Let β be the underlying behavior policy, $\hat{\beta}$ be our proxy for β, π be the learned actor, and Q be the value function of π. Consider the original Bellman operator $\mathcal{T}^\pi Q(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim P(\cdot|s,a),\ a'\sim\pi(\cdot|s')}\left[Q(s',a')\right]$. Then, in the tabular setting without function approximation, the*

objective (9) yields the following operator:

$$\mathcal{T}^{\pi}_{FAC}Q(s,a) = \begin{cases} \mathcal{T}^{\pi}Q(s,a) & \text{if } \hat{\beta}(a|s) \geq \epsilon,\ \beta(a|s) > 0 \\ \mathcal{T}^{\pi}Q(s,a) - \frac{\alpha}{2}\left(\frac{w^{\hat{\beta}}(s,a)\pi(a|s)}{\beta(a|s)}\right) & \text{if } \hat{\beta}(a|s) < \epsilon,\ \beta(a|s) > 0 \\ -\infty & \text{if } \beta(a|s) = 0. \end{cases} \tag{10}$$

unless $\beta(a|s) = 0$ *and* $w^{\hat{\beta}}(s,a) = 0$ *simultaneously.*

Proof of Proposition 1 is in Appendix E. Note that it is highly unlikely that we have $\beta(a|s) = 0$ and $w^{\hat{\beta}}(s,a) = 0$ simultaneously by the definition of the weight $w^{\hat{\beta}_\psi}(s,a)$. The operator associated with our new penalization maintains the original Bellman operator inside most of the ID support region, while strongly suppressing the Q value in the OOD region. Furthermore, for a weak confidence of the estimator, it gradually suppresses the Q value according to its confidence. Thus, when the proxy $\hat{\beta}_\psi$ is a strong estimator of β, the proposed penalization works properly.

Note that in our critic penalization, determining ϵ is important. We consider two dataset-driven methods for ϵ design: (1) dataset-wide constant threshold $\min_{(s,a)\sim\mathcal{D}} \hat{\beta}_\psi(a|s)$ and (2) batch-adaptive threshold $\hat{\beta}_\psi(a|s)$ using mini-batch samples $\mathcal{B} = \{(s,a)\}$ from $\mathcal{D}$. The dataset-wide threshold is robust when the underlying behavior density is sharp, being suited to datasets with wide state coverage and low action diversity, and the batch-adaptive threshold adapts to local coverage and multi-modality, being suited to datasets with limited state coverage and high action diversity. This ϵ design based on minimum over the dataset ensures that thresholding with ϵ does not exclude some of the actual ID region of the dataset.

4.2 Enhanced Flow Actor Optimization

The typical offline RL actor optimization is done with the objective (2). That is, the objective is to maximize Q value while keeping near the behavior policy β. One can apply this offline actor design principle to flow-based actor (Park et al., 2025). In this case, the flow behavior proxy policy $\hat{\beta}_\psi$ is learned as described in Section 2.2. Then, the objective (2) for the actual target policy π_θ can be implemented as follows (Park et al., 2025):

$$\max_{\pi_\theta} \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\theta}\left[Q_\phi(s,a)\right] - \lambda\,\mathbb{E}_{s\sim\mathcal{D},z\sim\mathcal{N}(0,I)}\left[\|a_\theta(s,z) - a_\psi(s,z)\|_2^2\right], \tag{11}$$

where $a_\theta(s,z)$ and $a_\psi(s,z)$ denote actions generated by π_θ and $\hat{\beta}_\psi$ given (s,z), respectively. Here, $a_\psi(a,z)$ is realized with velocity modeling (6) and integration (4) for full expressibility. However, the target policy π_θ implementation is simplified because the numerical multi-step integration based on velocity modeling makes gradient backpropagation complicated. Thus, Park et al. (2025) used a one-step flow policy, which transports z directly to a via a flow $a_\theta(\,\cdot\,; s, u=1) : \mathcal{A} \to \mathcal{A} : z \mapsto a$. Although this one-step flow policy π_θ has less expressibility than the multistep behavior proxy policy $\hat{\beta}_\psi$, it still has sufficient capability to express multi-modality, as shown in Appendix C. Thus, we adopt this one-step flow policy as our actor too.

In most cases of policy optimization via Q maximization with a distance constraint like (11) including Park et al. (2025), simple $Q_\phi(s,a)$, just trained with the Bellman error $\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[(Q_\phi(s,a) - \mathcal{T}^{\pi_\theta}Q_{\bar{\phi}}(s,a))^2\right]$, is used, where $\bar{\phi}$ is the target Q network parameter. The rationale behind this is that the distance regularization term in (11) attracts the policy to the behavior support although the Q function is overestimated in the OOD region. However, when near-optimal actions are rare compared to sub-optimal ones in the dataset, weak regularization in (11) drives value maximization toward the OOD region on which unpenalized Q_ϕ has overestimated values, yielding wrong actions. On the other hand, strong regularization induces the actor to imitate sub-optimal actions mostly. In such cases, if we use weak distance regularization and a Q function well-penalized at the OOD region in (11), then maximizing such a Q function yields good actions within the ID support. For such an idea to work, we need a Q function estimate precisely penalized at the OOD region. Therefore, we use the critic penalized with highly expressive flow behavior proxy proposed in Section 4.1. We name this new actor-critic structure for offline RL based on the flow model *flow actor-critic (FAC)*. It will be shown that FAC sets new state-of-the-art performance for difficult tasks of the OGBench in the experiment section soon.

4.3 Practical Implementation

The practical implementation of FAC is provided in Algorithm 1. The proposed method comprises three key components: a flow behavior proxy policy, a one-step flow actor, and twin critics to ensure stable Q value estimation (Fujimoto et al., 2018). The training procedure consists of two stages.

In the first stage, we train a flow behavior proxy policy $\hat{\beta}_\psi$ by eq. (6). After training the proxy policy, we evaluate behavior proxy density by eq. (7) for all state-action pairs in the dataset and store these evaluations along with the transitions. These stored evaluations as the dataset-driven threshold allow efficient identification of ID and OOD regions in the subsequent training stage.

In the second stage, we train penalized critics and a one-step flow actor concurrently. That is, the critics Q_ϕ are trained by eq. (9) with target critics for stable learning. The proposed critic penalization applies behavior-aware weights derived from the behavior proxy density, both for the actor actions and for the stored evaluations in the dataset. The actor π_θ is trained by eq. (11).

Algorithm 1: Flow Actor-Critic

Initialize one-step flow policy network π_θ, critic networks Q_{ϕ_1}, Q_{ϕ_2}, target networks $Q_{\bar{\phi}_1}, Q_{\bar{\phi}_2}$, and flow behavior proxy network $\hat{\beta}_\psi$

for *each iteration* **do**

- Sample $(s, a) \sim \mathcal{D}$, $z \sim \mathcal{N}(0, I)$, $u \sim \mathrm{Unif}([0, 1])$.
- Update $\hat{\beta}_\psi$ with eq. (6).

Compute $\{\hat{\beta}_\psi(a|s)\}$ for the samples in $\mathcal{D}$ by eq. (7) and store the evaluations to $\mathcal{D}$.

for *each iteration* **do**

- Sample $(s, a, r, s') \sim \mathcal{D}$, $z \sim \mathcal{N}(0, I)$ and compute ϵ.
- Update Q_{ϕ_i} with eq. (9), $\quad i \in \{1, 2\}$.
- Update π_θ with eq. (11).
- $\bar{\phi}_i \leftarrow \rho\phi_i + (1 - \rho)\bar{\phi}_i, \quad i \in \{1, 2\}$ for some ρ.

5 Related Works

Offline RL with Gaussian policies. Most methods with Gaussian policies can be grouped into two complementary approaches: actor regularization (AR) and critic penalization (CP). AR methods constrain the learned policy to the behavior policy in the dataset, such as TD3+BC (Fujimoto & Gu, 2021) imposing a simple behavior cloning regularizer and SPOT (Wu et al., 2022) using VAE-based ELBO to induce high-support actions. Other related AR methods extract the policy through value-weighted regression (Peng et al., 2019; Nair et al., 2020; Lee et al., 2021), such as IQL (Kostrikov et al., 2021). On the other hand, CP methods, such as CQL (Kumar et al., 2020), penalize value estimates for policy actions, but the induced policy becomes overly conservative. To mitigate this, MCQ (Lyu et al., 2022) introduces pseudo target value, EPQ (Yeom et al., 2024) uses state-dependent penalty weights defined by VAE-based ELBO and a current policy, SAC-RND (Nikulin et al., 2023) leverages RND-based anti-exploration bonus (Burda et al., 2018), and SVR (Mao et al., 2023) uses the property of importance sampling. Based on these approaches, ReBRAC (Tarasov et al., 2023) shows that complementary use of AR and CP (Wu et al., 2019) yields strong performance when this joint approach is integrated with modern offline RL design choices.

Diffusion and flow-based policies. Expressive policies have recently emerged as alternatives for modeling complex action distributions. The methods using such expressive policies can be categorized by whether they maximize their Q function explicitly or implicitly. For diffusion-based methods, explicit approaches, such as CAC (Ding & Jin, 2023) and SRPO (Chen et al., 2023), optimize Q-maximizing policies (Wang et al., 2022; He et al., 2023; Zhang et al., 2024). Implicit approaches, such as IDQL (Hansen-Estruch et al., 2023), either use value-weighted regression (Lu et al., 2023; Kang et al., 2023; Ding et al., 2024) or apply value-weighted sampling (Chen et al., 2022; He et al., 2024). For flow-based methods, FQL (Park et al., 2025) trains a Q-maximizing policy using the distance regularizer with a flow behavior proxy policy. Furthermore, Park et al. (2025) has shown that stable policy extraction for multistep flow policies is challenging: value-weighted regression (FAWAC), explicit Q-maximization (FBRAC), and value-weighted sampling (IFQL). Another method, QIPO-OT (Zhang et al., 2025), trains a flow matching policy through value-weighted regression.

Table 1: Evaluation over 50 singletasks of the OGBench. For each task category, we report the final performance averaged across its 5 singletasks, over 8 seeds, with ± indicating the standard deviation. Full evaluation results on the 50 singletasks are in Table 3.

	Gaussian Policies			Diffusion Policies			Flow Policies				
Task Category	**BC**	**IQL**	**ReBRAC**	**IDQL**	**SRPO**	**CAC**	**FAWAC**	**FBRAC**	**IFQL**	**FQL**	**FAC (Ours)**
antmaze-large-navigate	10.6	53.4	80.8	20.8	10.6	32.8	6.4	60.2	28.0	78.6	**92.6**±2.5
antmaze-giant-navigate	0.2	4.0	**26.2**	0.0	0.0	0.0	0.0	3.8	2.6	8.6	23.0±5.1
humanoidmaze-medium-navigate	2.0	32.8	21.8	0.8	1.4	52.8	19.4	38.4	60.4	57.4	**75.6**±3.6
humanoidmaze-large-navigate	0.4	2.4	2.6	0.6	0.2	0.6	0.2	2.2	**11.0**	4.2	8.3±5.5
antsoccer-arena-navigate	1.0	8.4	0.0	11.8	1.0	1.8	12.4	16.0	33.2	60.2	**67.7**±2.9
cube-single-play	5.4	83.0	90.6	94.6	79.6	85.2	81.2	78.6	79.2	95.8	**98.8**±1.4
cube-double-play	1.6	6.4	12.2	14.6	1.4	5.8	5.2	15.0	14.0	28.6	**33.1**±5.7
scene-play	4.6	27.6	40.6	46.2	20.0	39.8	29.8	44.8	30.4	55.8	**71.3**±7.6
puzzle-3x3-play	1.8	9.0	21.6	10.4	17.8	19.4	6.4	14.0	19.0	29.6	**100.0**±0.0
puzzle-4x4-play	0.2	7.4	14.0	29.2	10.6	14.8	0.4	13.2	25.2	17.2	**32.3**±6.5
Average	2.8	23.4	31.0	22.9	14.3	25.3	16.1	28.6	30.3	43.6	**60.3**

6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate FAC on two offline RL benchmarks: recently introduced OGBench (Park et al., 2024) and standard D4RL (Fu et al., 2020). OGBench offers diverse goal-conditioned locomotion and manipulation environments, in which each task category consists of 5 singletask variants, which are relatively more challenging than the D4RL tasks. To align with standard offline RL, we use 50 reward based singletask variants and follow the 10 OGBench task categories. For D4RL, we use 9 MuJoCo tasks, 6 Antmaze tasks, and 8 Adroit tasks. The details are described in Appendix G.1.

Baselines. We evaluate FAC against a total of 17 baselines and categorize them by policy class: Gaussian, diffusion, and flow policies. For OGBench, we compare against Gaussian policy-based baselines BC, IQL (Kostrikov et al., 2021), ReBRAC (Tarasov et al., 2023), diffusion policy-based baselines IDQL (Hansen-Estruch et al., 2023), SRPO (Chen et al., 2023), CAC (Ding & Jin, 2023), and flow policy-based baselines FAWAC, FBRAC, IFQL, FQL (Park et al., 2025). For D4RL, we additionally compare against Gaussian policy-based baselines TD3+BC (Fujimoto & Gu, 2021), CQL (Kumar et al., 2020), MCQ (Lyu et al., 2022), EPQ (Yeom et al., 2024), SAC-RND (Nikulin et al., 2023), SPOT (Wu et al., 2022), and a flow policy-based baseline QIPO-OT (Zhang et al., 2025). The details of the baselines are described in Appendix G.2.

Evaluation and hyperparameters. We report the final performance evaluated after 1M gradient steps. All results are averaged over 8 random seeds and presented as mean ± standard deviation. For evaluation, the one-step flow policy trained by FAC samples one action at each time step. For training FAC, we mainly tune two hyperparameters: the critic penalization coefficient α and the actor regularization coefficient λ. All remaining settings (e.g., dataset-driven threshold schemes) are fixed per task domain. The complete configuration is in Appendix G.3.

6.2 PERFORMANCE ON THE OGBENCH BENCHMARK

The evaluation results aggregated from 50 singletasks of the OGBench are summarized in Table 1, and the full evaluation results are in Table 3 in Appendix F. FAC achieves the highest overall performance across all methods. FAC yields a clear margin, compared to FQL among flow policy-based baselines, CAC among diffusion policy-based baselines, ReBRAC among Gaussian policy-based baselines. In particular, the comparison with Gaussian policy-based baselines highlights the benefit of expressive policies on datasets with multi-modal action distribution, while the gap to diffusion policy-based baselines and multi-step flow policy-based baselines (FAWAC, FBRAC, IFQL) reveals the practical difficulty of optimizing iterative generators. Leveraging a one-step flow actor enables both FAC and FQL to outperform other baselines. However, fusing flow-based critic penalization and actor regularization, FAC improves performance over FQL, which is the actor regularization only.

On navigation tasks, FAC achieves the best performance on 3 out of 5 navigation tasks. Although `antmaze-giant-navigate` remains led by ReBRAC, reflecting that a joint critic penalization

Table 2: Evaluation on 23 tasks of the D4RL. We report the final performance averaged over 8 seeds, with ± indicating the standard deviation. For MuJoCo datasets, we use the following abbreviations: *m* for *medium*, *mr* for *medium-replay*, *me* for *medium-expert*.

	Gaussian Policies							Diffusion Policies			Flow Policies		
MuJoCo Tasks	**TD3+BC**	**IQL**	**CQL**	**MCQ**	**EPQ**	**SPOT**	**ReBRAC**	**IDQL**	**SRPO**	**CAC**	**QIPO-OT**	**FQL**	**FAC (Ours)**
halfcheetah-m	48.3	47.4	44.0	64.3	67.3	58.4	65.6	51.0	60.4	**69.1**	54.2	60.3±1.1	65.0±1.5
hopper-m	59.3	66.3	58.5	78.4	101.3	86.0	**102.0**	65.4	95.5	80.7	94.1	68.1±3.4	91.9±3.9
walker2d-m	83.7	78.3	72.5	**91.0**	87.8	86.4	82.5	82.5	84.4	83.1	87.6	77.2±2.5	85.2±0.9
halfcheetah-mr	44.6	44.2	45.5	56.8	**62.0**	52.2	51.0	45.9	51.4	58.7	48.0	49.3±0.5	55.4±2.7
hopper-mr	60.9	94.7	95.0	**101.6**	97.8	100.2	98.1	92.1	101.2	99.7	101.3	49.8±7.2	99.1±0.9
walker2d-mr	81.8	73.9	77.2	91.3	85.3	**91.6**	77.3	85.1	84.6	79.5	78.6	53.1±7.9	83.0±5.8
halfcheetah-me	90.7	86.7	91.6	87.5	95.7	86.9	101.1	95.9	92.2	84.3	94.5	99.6±6.5	**101.9**±5.6
hopper-me	98.0	91.5	105.4	**111.2**	108.8	99.3	107.0	108.6	100.1	100.4	108.0	83.1±17.0	104.2±4.6
walker2d-me	110.1	109.6	108.8	**114.2**	112.0	112.0	111.6	112.7	114.0	110.4	110.9	106.1±1.8	108.4±0.6
Average	75.3	77.0	77.6	88.5	**90.9**	85.9	88.5	82.1	87.1	85.1	86.4	71.8	88.2

	Gaussian Policies							Diffusion Policies			Flow Policies		
Antmaze Tasks	**TD3+BC**	**IQL**	**CQL**	**MCQ**	**EPQ**	**SAC-RND**	**ReBRAC**	**IDQL**	**SRPO**	**CAC**	**QIPO-OT**	**FQL**	**FAC (Ours)**
umaze	78.6	87.5	74.0	98.3	**99.4**	97.0	97.8	94.0	97.1	75.8	93.6	96.0	98.5±3.0
umaze-diverse	71.4	62.2	84.0	80.0	78.3	66.0	88.3	80.2	82.1	77.6	76.1	89.0	**93.5**±6.0
medium-play	10.6	71.2	61.2	52.5	85.0	38.5	84.0	84.5	80.7	56.8	80.0	78.0	**88.0**±9.6
medium-diverse	3.0	70.0	53.7	37.5	**86.7**	74.7	76.3	84.8	75.0	0.0	86.4	71.0	85.0±7.3
large-play	0.2	39.6	15.8	2.5	40.0	43.9	60.4	63.5	53.6	0.0	55.5	84.0	**90.0**±4.3
large-diverse	0.0	47.5	14.9	7.5	36.7	45.7	54.4	67.9	53.6	0.0	32.1	83.0	**88.0**±6.0
Average	27.3	63.0	50.6	46.4	71.0	61.0	76.9	79.2	73.7	35.0	70.6	83.5	**90.5**

	Gaussian Policies							Diffusion Policies			Flow Policies	
Adroit Tasks	**TD3+BC**	**IQL**	**CQL**	**MCQ**	**EPQ**	**SAC-RND**	**ReBRAC**	**IDQL**	**SRPO**	**CAC**	**FQL**	**FAC (Ours)**
pen-human	81.8	81.5	37.5	68.5	83.9	5.6	**103.5**	76.0	69.0	64.0	53.0	73.9±14.7
pen-cloned	61.4	77.2	39.2	49.4	91.8	2.5	91.8	64.0	61.0	56.0	74.0	**103.2**±11.1
door-human	-0.1	3.1	9.9	2.3	**13.2**	0.0	0.0	6.0	3.0	5.0	0.0	5.5±3.3
door-cloned	0.1	0.8	0.4	1.3	**5.8**	0.2	1.1	0.0	0.0	1.0	2.0	4.1±3.9
hammer-human	0.4	2.5	4.4	0.3	3.9	-0.1	0.2	2.0	1.0	2.0	1.0	**8.6**±5.4
hammer-cloned	0.8	1.1	2.1	1.4	**22.8**	0.1	6.7	2.0	2.0	1.0	11.0	11.1±11.2
relocate-human	-0.2	0.1	0.2	0.1	0.3	0.0	0.0	0.0	0.0	0.0	0.0	**0.6**±0.5
relocate-cloned	-0.1	0.2	-0.1	0.0	0.1	0.0	**0.9**	0.0	0.0	0.0	0.0	0.5±0.4
Average	18.0	20.8	11.7	15.4	**27.7**	1.0	25.5	18.8	17.0	16.1	17.6	25.9

and actor regularization approach can be competitive even with Gaussian policies. For higher-dimensional `humanoid` tasks, FAC outperforms ReBRAC, suggesting that the expressive actor contributes in complex action spaces. In `humanoidmaze-large-navigate`, FAC is slightly below IFQL on average but matches or exceeds this baseline on 4 of the 5 singletasks (refer to Table 3 in the appendix for details).

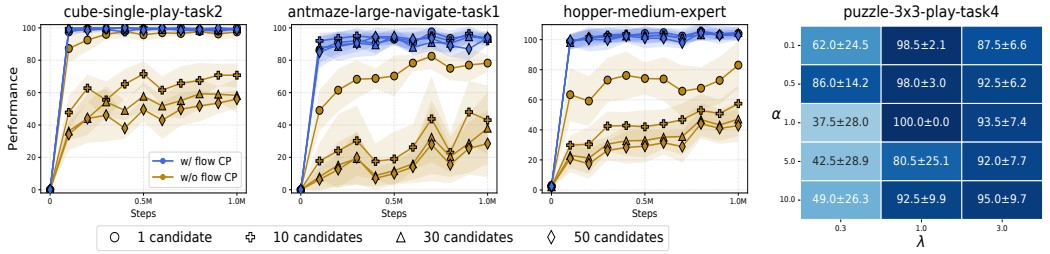
On manipulation tasks, FAC outperforms all baselines across the board and achieves perfect success on `puzzle-3x3-play`. In particular, the `puzzle-play` tasks for solving lights-out puzzle are designed to test combinatorial reasoning and trajectory stitching, on these tasks, FAC attains 238% and 11% performance improvements over the strongest baselines (FQL and IDQL) on `puzzle-3x3-play` and `puzzle-4x4-play`, respectively.

6.3 Performance on the D4RL Benchmark

The evaluation results on 23 tasks of the D4RL are summarized in Table 2. Overall, FAC is competitive or superior across all three domains (MuJoCo, Antmaze, Adroit), indicating that the benefit of jointly penalizing the critic and regularizing the actor with the expressive behavior proxy policy extends beyond the OGBench and holds under diverse reward functions and state-action distributions. Among methods with expressive policies, FAC attains the best overall performance across all three domains.

On MuJoCo tasks with dense reward functions, it is susceptible to value overestimation for OOD actions. It is seen that FQL exhibits sensitivity to the value overestimation bias, while FAC matches strong Gaussian policy-based baselines (MCQ, EPQ, ReBRAC). This demonstrates that the flow-based critic penalization in FAC successfully mitigates the value overestimation.

On Antmaze tasks, as task difficulty increases from `umaze` to `medium` to `large`, many baselines including the strong baselines on the MuJoCo domain, diffusion policy-based baselines, and the



(a) $\arg\max_a Q(s, a)$ evaluation with N candidate actions. (b) Performance across α and λ

Figure 3: Ablation studies. (a) Effect of flow-based critic penalization (CP) on performance across three offline RL tasks under N candidate actions. (b) Performances on the default singletask of `puzzle-3x3-play` across actor regularization coefficient λ and critic penalization coefficient α.

multistep flow policy-based baseline (QIPO-OT) exhibit pronounced performance degradation. In contrast, FAC maintains robust performance against difficulty and consistently improves over FQL.

On Adroit tasks that are notoriously challenging due to the high-dimensional action space and the sparse dataset support, FAC is comparable to the strong Gaussian policy-based baselines (EPQ and ReBRAC) in overall performance. In particular, FAC attains substantial improvements on `pen-cloned` and `hammer-human` tasks.

6.4 Ablation Studies

Effect of the critic penalization. We investigate the impact of the proposed flow-based critic penalization (CP) of FAC in terms of OOD action sampling by replacing our flow-based penalized critic with an unpenalized conventional Q estimator (i.e., reducing to FQL). For this, we sample $N \in \{1, 10, 30, 50\}$ candidate actions $a(s, z)$ from the one-step flow actor with N i.i.d. z for given s, and execute $\arg\max_a Q_\phi(s, a)$ at each step for each method, with λ set to its best-performing value for fair comparison. Note that as N increases, the chance of selecting an OOD action increases and $\arg\max_a Q_\phi(s, a)$ with the unpenalized Q can check this. As shown in Fig. 3a, as N increases, FQL exhibits significant performance degradation, showing that OOD actions are drawn even with distance regularization. In contrast, our method maintains nearly-invariant performance with increasing N, demonstrating our flow-based CP together with actor regularization is very powerful to handle possibly-occurring OOD actions.

Sensitivity to α and λ. FAC substantially improves performance on `puzzle-3x3-play-v0`. Fig. 3b shows a performance heatmap across critic penalization coefficient α and actor regularization coefficient λ on this default singletask, with each cell presenting the mean $\pm$ standard deviation over 8 seeds. A broad range of high performance appears when both α and λ are jointly active and balanced. With adequate $\lambda \geq 1$, performance is robust over α, whereas with too small $\lambda = 0.3$, performance becomes sensitive to α. These results demonstrate that jointly applying flow-based critic penalization and flow actor optimization produces a synergistic effect, enabling reliable support-aware flow policy optimization in the offline RL setting.

7 Conclusion

We have proposed *flow actor-critic* (FAC), which leverages a flow behavior proxy policy jointly for critic penalization and actor regularization. The behavior proxy policy provides tractable behavior densities that yield confidence weights for critic penalization so that the resulting operator preserves the original Bellman operator in confident in-distribution region and penalizes value overestimation for out-of-distribution actions in complex and multimodal datasets. The same proxy policy anchors the flow actor near the dataset support while keeping it sufficiently expressive to concentrate on high-value actions. This joint approach enables flow-based actor-critic optimization that is reliably constrained within well-supported regions. Empirically, FAC achieves consistently strong performance across tasks of the OGBench and D4RL benchmarks, effectively handling out-of-distribution actions and maintaining high true returns under diverse reward structures and state-action distributions.

REFERENCES

Ryan P Adams, Jeffrey Pennington, Matthew J Johnson, Jamie Smith, Yaniv Ovadia, Brian Patton, and James Saunderson. Estimating the spectral density of large implicit matrices. *arXiv preprint arXiv:1802.03451*, 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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.

Heli Ben-Hamu, Samuel Cohen, Joey Bose, Brandon Amos, Aditya Grover, Maximilian Nickel, Ricky TQ Chen, and Yaron Lipman. Matching normalizing flows and probability paths on manifolds. *arXiv preprint arXiv:2207.04711*, 2022.

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`.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

Huayu Chen, Cheng Lu, Zhengyi Wang, Hang Su, and Jun Zhu. Score regularized policy optimization through diffusion behavior. *arXiv preprint arXiv:2310.07297*, 2023.

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

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, 2024.

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

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

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

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

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 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.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 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.

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*, 2023.

Longxiang He, Li Shen, Junbo Tan, and Xueqian Wang. Aligniql: Policy alignment in implicit q-learning through constrained optimization. *arXiv preprint arXiv:2405.18187*, 2024.

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

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

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

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

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pp. 45–73. Springer, 2012.

Jongmin Lee, Wonseok Jeon, Byungjun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In *International Conference on Machine Learning*, pp. 6120–6130. PMLR, 2021.

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

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

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.

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, 2023.

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

Yixiu Mao, Hongchang Zhang, Chen Chen, Yi Xu, and Xiangyang Ji. Supported value regularization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36: 40587–40609, 2023.

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

Alexander Nikulin, Vladislav Kurenkov, Denis Tarasov, and Sergey Kolesnikov. Anti-exploration by random network distillation. In *International conference on machine learning*, pp. 26228–26244. PMLR, 2023.

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

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

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

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

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

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

Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 31278–31291, 2022.

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

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

Junghyuk Yeom, Yonghyeon Jo, Jeongmo Kim, Sanghyeon Lee, and Seungyul Han. Exclusively penalized q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:113405–113435, 2024.

Ruoqi Zhang, Ziwei Luo, Jens Sjölund, Thomas Schön, and Per Mattsson. Entropy-regularized diffusion policy with q-ensembles for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:98871–98897, 2024.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. *arXiv preprint arXiv:2503.04975*, 2025.

A LIMITATIONS

The proposed method heavily relies on the fidelity and accuracy of density evaluation of the flow behavior proxy policy for the underlying behavior policies. When the proxy policy fails to capture the multi-modal dataset distribution, the critic penalization with the flow behavior proxy density can suppress values in inappropriate regions, and the distance regularization for flow actor optimization can attract the flow actor toward the regions. Moreover, FAC provides no mechanism for beneficial exploration, and therefore its applicability to online RL or offline-to-online RL remains uncertain. These limitations suggest future extensions: robust learning of the behavior proxy (Liu et al., 2022) to enable reliable support-aware policy improvement, and efficient exploitation strategies (Haarnoja et al., 2018) for flow policies.

B THE USE OF LARGE LANGUAGE MODELS

We used large language models to refine and polish our writing.

C FLOW BASED CRITIC PENALIZATION AND ACTOR REGULARIZATION ON A DIDACTIC EXAMPLE

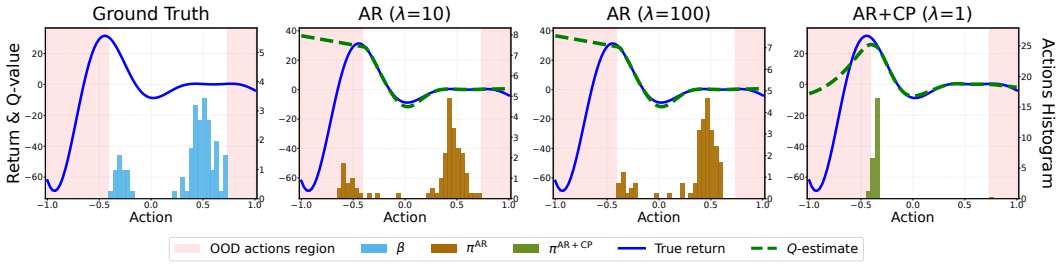


Figure 4: Didactic comparison of AR and AR+CP on a Gaussian mixture dataset. Leftmost: action histogram (blue bars) in the dataset overlaid with ground-truth return (blue line). Middle: AR variants, each with the learned Q estimate (green dashed line). Rightmost: AR+CP. Red shading marks OOD region.

In this appendix, we investigate whether the one-step flow actor has sufficient expressibility and whether the proposed flow-based critic penalization is effective for the flow actor. For this, we follow the regression approach from Ball et al. (2023) to construct a didactic multi-modal dataset and compare two variants: AR+CP, which uses both flow-based critic penalization (9) and flow actor regularization (11), and AR, which ablates the critic penalization.

Fig. 4 visualizes, for each variant, sampled actions and learned critic values against the ground-truth return, with out-of-distribution (OOD) regions shaded. The one-step flow actor captures the multi-modal action distribution in both variants, confirming sufficient expressibility on this task. The key difference appears in critic learning. AR tends to overestimate in the OOD region, which leads to OOD action sampling under weak regularization and to over-imitation of sub-optimal actions under strong regularization. In contrast, AR+CP preserves accurate values on the in-distribution (ID) region and gradually suppresses values in the OOD region, thus the actor concentrates on the well-supported high-value region even with weaker actor regularization. These observations demonstrate how flow-based critic penalization complements flow actor regularization by aligning critic learning with dataset support while retaining multi-modal expressibility.

C.1 IMPLEMENTATION

Dataset. We construct a dataset by generating a single 2-dimensional state, $s = [0.5, -0.5]$, and 1-dimensional actions from a Gaussian mixture with two modes, concretely $a \sim \frac{1}{5}\mathcal{N}(\mu = -0.3, \sigma = 0.004) + \frac{4}{5}\mathcal{N}(\mu = 0.5, \sigma = 0.01)$. Since the dataset contains only a single state, instead of training the Q functions by minimizing the Bellman error, we directly provide true returns for the dataset actions, $y(a) = 30(0.6 - a)^2 \sin(2\pi \times a + 1.3\pi)$, enabling regression.

Network architecture. We use a [256, 256, 256]-sized MLP for all neural networks. The critic network Q_ϕ takes a state-action pair (s, a) as input and outputs a scalar estimate of the true return. The behavior proxy policy $\hat{\beta}_\psi$ takes the state s, Gaussian noise $z \in \mathbb{R}^{d_\mathcal{A}}$, and flow time $u \in [0, 1]$ as inputs and then outputs the velocity field v_u. The one-step flow policy π_θ takes the state s and Gaussian noise z and then outputs a single action a.

C.2 ALGORITHMS

We train the AR+CP and AR variants by adapting the task in which the Q_ϕ regresses to the ground-truth returns. For the variants, we use a behavior flow proxy policy trained with the flow matching objective (6). Here, we restate the objective.

$$\min_\psi \mathbb{E}_{(s,a)\sim\mathcal{D}, z\sim\mathcal{N}(0,I), u\sim\text{Unif}([0,1])} \left[\|v_\psi(\tilde{a}_u; s, u) - (a - z)\|_2^2\right]$$

We use the Euler method with 10 steps as an integral (ODE) solver for action sampling and evaluating behavior proxy density from $\hat{\beta}_\psi$. The detailed objectives for the variants are presented below.

Actor Regularization and Critic Penalization (AR+CP). In this regression case, AR+CP is a variant of the full method using both flow-based critic penalization and actor regularization. The objectives are given by

$$\max_{\pi_\theta} \ \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\pi_\theta}\left[Q_\phi(s,a)\right] - \lambda\,\mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[\|a_\theta(s,z)-a_\psi(s,z)\|_2^2\right]\right]$$

$$\min_{Q_\phi} \ \alpha\,\mathbb{E}_{s\sim\mathcal{D},\ a\sim\pi}\left[w^{\hat{\beta}_\psi}(s,a)\cdot Q(s,a)\right] + \mathbb{E}_{s,a\sim\mathcal{D}}\left[\left(Q(s,a)-y(a)\right)^2\right]$$

Actor Regularization (AR). In this regression case, AR is a variant without the flow-based critic penalization. The objectives are given by

$$\max_{\pi_\theta} \ \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\pi_\theta}\left[Q_\phi(s,a)\right] - \lambda\,\mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[\|a_\theta(s,z)-a_\psi(s,z)\|_2^2\right]\right]$$

$$\min_{Q_\phi} \ \mathbb{E}_{s,a\sim\mathcal{D}}\left[\left(Q(s,a)-y(a)\right)^2\right]$$

D BEHAVIOR CLONING MODELS

In Fig. 1, we compare the flow behavior proxy model against other behavior cloning (BC) models. Specifically, we investigate data sampling as well as log-density or Evidence Lower Bound (ELBO) evaluations when employing a flow matching model (Lipman et al., 2022), a simple Gaussian probability model, a conditional VAE (Kingma & Welling, 2013; Sohn et al., 2015), and diffusion models with $T = \{10, 50\}$ denoising steps (Ho et al., 2020).

D.1 IMPLEMENTATION

Dataset. We construct a dataset by generating a single state, $s = [0.5, -0.5, 0.5, -0.5]$, and 2-dimensional actions from a Gaussian mixture with four modes, $a \sim \frac{1}{4}\sum_{i=1}^{4}\mathcal{N}(\mu_i, \Sigma = 0.008I)$, where $\mu_i \in \{[-0.5, -0.5], [-0.5, 0.5], [0.5, -0.5], [0.5, 0.5]\}$.

Network architecture. We use a [512, 512, 512, 512]-sized MLP for all neural networks. For the flow matching model, we implement one neural network: velocity prediction network that takes the state s, Gaussian noise $z \in \mathbb{R}^{d_\mathcal{A}}$, and flow time $u \in [0, 1]$ and then outputs velocity estimate $v_u \in \mathbb{R}^{d_\mathcal{A}}$. For the Gaussian model, we use one neural network: Gaussian probability model, $\mathcal{N}(\cdot|\mu_\theta, \sigma_\theta)$. For the conditional VAE, we implement two neural networks: (1) Encoder network that takes the state s and one action a then outputs latent variable z, and (2) Decoder network that takes the state s and latent variable z then outputs an action estimate. For the diffusion models, we use one neural network: conditional ϵ prediction network that takes the state s, diffusion time t, and latent variable $x \in \mathbb{R}^{d_\mathcal{A}}$ and then outputs ϵ-estimate.

D.2 TRAINING OBJECTIVE AND ACTION SAMPLING

Flow Matching model. We train a flow matching model with the flow matching objective (6) and restate it:

$$\min_{\psi} \mathbb{E}_{(s,a)\sim\mathcal{D}, z\sim\mathcal{N}(0,I), u\sim\text{Unif}([0,1])}\left[\|v_\psi(\tilde{a}_u; s, u) - (a - z)\|_2^2\right]$$

For action sampling, we solve the following integral equation using the Euler method with 10 steps:

$$a = z + \int_0^1 v_u du = z + \int_0^1 v_\psi(a_u; s, u)du,$$

where $a_u = \int_0^u v_t dt$, and $z \sim \mathcal{N}(0, I)$.

Gaussian model. A Gaussian model is trained by maximizing log-likelihood:

$$\max_{\theta} \mathbb{E}_{s,a\sim\mathcal{D}}\left[\log p(a|s)\right] = \mathbb{E}_{s,a\sim\mathcal{D}}\left[\log\mathcal{N}(a|\mu_\theta(s), \sigma_\theta(s))\right]$$

We can easily sample actions from the Gaussian model.

Conditional VAE model. We train Encoder and Decoder models, $q_\phi(z|s, a)$ and $p_\theta(a|s, z)$, respectively, by minimizing the ELBO of log-likelihood (Kingma & Welling, 2013). The objective is given by

$$\max_{\theta,\phi} \mathbb{E}_{s,a\sim\mathcal{D}}\left[\log p(a|s)\right] \geq \mathbb{E}_{s,a\sim\mathcal{D}}\left[\mathbb{E}_{z\sim q_\phi}\left[\log p_\theta(a|s, z)\right] - D_{\text{KL}}(q_\phi(z|s, a)||p(z))\right],$$

where the prior distribution $p(z)$ is set to $\mathcal{N}(0, I)$.

For action sampling, we sample $z \sim \mathcal{N}(0, I)$, and feed the state s and z to the Decoder model.

Diffusion model. We train an ϵ-prediction model implemented as in DDPM (Ho et al., 2020).

$$\min_{\epsilon_\theta} \mathbb{E}_{t\sim\text{Unif}(\{1,\ldots,T\}), \epsilon\sim\mathcal{N}(0,I), (s,a)\sim\mathcal{D}}\left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}a + \sqrt{1-\bar{\alpha}_t}\epsilon, s, t)\|^2\right], \tag{12}$$

where $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{s=1}^{t}\alpha_s$. Following (Xiao et al., 2021) as in (Wang et al., 2022), we use the beta schedule $\beta_t = 1 - \exp\left(-\frac{\beta_{\text{min}}}{N} - \frac{(\beta_{\text{max}}-\beta_{\text{min}})(2t-1)}{2N^2}\right)$.

For action sampling, we first sample a noise $a_T \sim \mathcal{N}(0, I)$, and then iteratively generate actions through the reverse process of the diffusion model:

$$a_{t-1}|a_t = \frac{a_t}{\sqrt{\alpha_t}} - \frac{\beta_t}{\sqrt{\alpha_t(1-\bar{\alpha}_t)}}\epsilon_\theta(s, a_t, t) + \sqrt{\beta_t}\epsilon,$$

where $\epsilon \sim \mathcal{N}(0, I)$. Note that in the diffusion model, the denoising time $t = 0$ corresponds to the data index a_0, while $t = T$ corresponds to the noise index a_T, which is the opposite of the indexing convention in flow-based models.

D.3 Log-Density and ELBO for Arbitrary Actions

Flow Matching model. We can directly evaluate the log-density using the Instantaneous Change of Variables (Chen et al., 2018), which is a property of Continuous Normalizing Flows.

$$\log p(a|s) = \log p_0(\hat{z}) + \int_0^1 \frac{d}{du} \log p_u(a|s) du = \log p_0(\hat{z}) - \int_0^1 \nabla_{a_u} \cdot v_\psi(a_u; s, u) du,$$

where $a_u = a - \int_u^1 v_t dt$, and the base distribution p_0 is set to normal distribution. We also use the 10 step count of the Euler method. Note that $\hat{z}$ is obtained from $a = a_1$ due to the bijective property of Continuous Normalizing Flows.

Gaussian model. We can easily compute log-density.

$$\log p(a|s) = \log \mathcal{N}(a|\mu_\theta(s), \sigma_\theta(s))$$

Conditional VAE model. Since the VAE model cannot compute log-density exactly, we instead evaluate ELBO, which is a lower bound of its log-density. Using a normal distribution as the prior $p(z)$ and implementing both the encoder q_ϕ and decoder p_θ as Gaussian probability models, ELBO can be computed in a straightforward manner:

$$\log p(a|s) \geq \mathbb{E}_{z\sim q_\phi}\left[\log p_\theta(a|s, z)\right] - D_{\text{KL}}(q_\phi(z|s, a)||p(z))$$

Diffusion model. Diffusion model (Sohl-Dickstein et al., 2015) defines a forward process that gradually perturbs data to noise through a Markov chain, and a reverse process that learns to reconstruct data by denoising procedure that inverts the forward process. In the diffusion model, its log-likelihood is optimized via a variational lower bound (ELBO).

DDPM (Ho et al., 2020) simplifies the ELBO by reparameterizing the reverse process as the ϵ-prediction objective (12), which allows the ELBO to be expressed in a tractable form while retaining its variational interpretation.

In the RL setting, we define the forward process $q(a_t|a_{t-1}, s) = \mathcal{N}(a_t; \sqrt{1-\beta_t}a_{t-1}, \beta_t I)$ and the reverse process $p_\theta(a_{t-1}|a_t, s) = \mathcal{N}(a_{t-1}; \mu_\theta(s, a_t, t), \Sigma_\theta(s, a_t, t))$, following (Sohl-Dickstein et al., 2015). We therefore compute the ELBO as follows:

$$\begin{aligned}
\log p(a|s) &\geq \mathbb{E}_q\left[\log \frac{p_\theta(a_{0:T}|s)}{q(a_{1:T}|a_0, s)}\right] \\
&= \mathbb{E}_q\big[\log p_\theta(a_0|a_1, s) - \sum_{t>1} D_{\text{KL}}(q(a_{t-1}|a_t, a_0, s)\|p_\theta(a_{t-1}|a_t, s)) \\
&\qquad - D_{\text{KL}}(q(a_T|a_0, s)\|p(a_T))\big], \qquad (13)
\end{aligned}$$

where $p(a_T)$ is a normal distribution.

We already have the predefined beta schedule $\{\beta_t\}$ and the trained ϵ-prediction model, we can choose the forward process as in DDPM (Ho et al., 2020):

$$p_\theta(a_{t-1}|a_t, s) = \mathcal{N}(a_{t-1}; \mu_\theta(s, a_t, t), \tilde{\beta}_t I),$$

where $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$.

Using notable properties of the diffusion model (Sohl-Dickstein et al., 2015)

$$\begin{aligned}
q(a_t|a_0, s) &= \mathcal{N}\left(a_t; \sqrt{\bar{\alpha}_t}a_0, (1-\bar{\alpha}_t)I\right) \\
q(a_{t-1}|a_t, a_0, s) &= \mathcal{N}\left(a_{t-1}; \tilde{\mu}_t(a_t, a_0), \tilde{\beta}_t I\right),
\end{aligned}$$

where $\tilde{\mu}_t(a_t, a_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} a_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} a_t$, and the KL-divergence between two multivariate Gaussian distributions, $D_{\mathrm{KL}}(\mathcal{N}(\mu_1, \Sigma_1) \| \mathcal{N}(\mu_2, \Sigma_2))$,

$$D_{\mathrm{KL}}(\mathcal{N}_1 \| \mathcal{N}_2) = \frac{1}{2} \left\{ \mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + (\mu_2 - \mu_1)^\top \Sigma_2^{-1} (\mu_2 - \mu_1) - d_{\mathcal{A}} + \log \frac{\det(\Sigma_2)}{\det(\Sigma_1)} \right\},$$

where $d_{\mathcal{A}}$ denotes the dimension of the action space, each term of the ELBO (13) can be computed as

$$\log p_\theta(a_0 | a_1, s) = -\frac{d_{\mathcal{A}}}{2} \log(2\pi\tilde{\beta}_1) - \frac{1}{2\tilde{\beta}_1} \| a_0 - \mu_\theta(s, a_1, 1) \|^2$$

$$D_{\mathrm{KL}}(q(a_{t-1} | a_t, a_0, s) \| p_\theta(a_{t-1} | a_t, s)) = \frac{1}{2\tilde{\beta}_t} \| \tilde{\mu}_t(a_t, a_0) - \mu_\theta(s, a_t, t) \|^2,$$

$$D_{\mathrm{KL}}(q(a_T | a_0, s) \| p(a_T)) = \frac{1}{2} \left(\bar{\alpha}_T a_0^\top a_0 - d_{\mathcal{A}} \left(\bar{\alpha}_T + \log(1 - \bar{\alpha}_T) \right) \right)$$

where $\mu_\theta(s, a_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(a_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(s, a_t, t) \right)$.

E PROOF

Proposition 1. *Let β be the underlying behavior policy, $\hat{\beta}$ be our proxy for β, π be the learned actor, and Q be the value function of π. Consider the original Bellman operator $\mathcal{T}^\pi Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a),\, a' \sim \pi(\cdot|s')}\left[Q(s',a')\right]$. Then, in the tabular setting without function approximation, the objective (9) yields the following operator:*

$$\mathcal{T}^\pi_{FAC} Q(s,a) = \begin{cases} \mathcal{T}^\pi Q(s,a) & \text{if } \hat{\beta}(a|s) \geq \epsilon,\ \beta(a|s) > 0 \\ \mathcal{T}^\pi Q(s,a) - \frac{\alpha}{2}\left(\frac{w^{\hat{\beta}}(s,a)\pi(a|s)}{\beta(a|s)}\right) & \text{if } \hat{\beta}(a|s) < \epsilon,\ \beta(a|s) > 0 \\ -\infty & \text{if } \beta(a|s) = 0. \end{cases} \tag{10}$$

unless $\beta(a|s) = 0$ and $w^{\hat{\beta}}(s,a) = 0$ simultaneously.

Proof. In the tabular setting, we can replace $a \sim \mathcal{D}$ with $a \sim \beta$. Then, the objective (9) can be rewritten as

$$\begin{aligned} &\alpha\, \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi}\left[w^{\hat{\beta}}(s,a) \cdot Q(s,a)\right] + \mathbb{E}_{s\sim\mathcal{D},\, a\sim\beta}\left[\left(Q(s,a) - \mathcal{T}^\pi Q(s,a)\right)^2\right] \\ &= \mathbb{E}_{s\sim\mathcal{D}, a\sim\beta}\left[\alpha\, w^{\hat{\beta}}(s,a)\frac{\pi(a|s)}{\beta(a|s)}\, Q(s,a) + \left(Q(s,a) - \mathcal{T}^\pi Q(s,a)\right)^2\right] \end{aligned} \tag{14}$$

We set the derivative of (14) to zero:

$$\alpha \frac{w^{\hat{\beta}}(s,a)\pi(a|s)}{\beta(a|s)} + 2\left(Q(s,a) - \mathcal{T}^\pi Q(s,a)\right) = 0,$$

leading to the solution $Q^*(s,a)$, given by

$$Q^*(s,a) = \mathcal{T}^\pi_{\text{FAC}} Q(s,a) = \mathcal{T}^\pi Q(s,a) - \frac{\alpha}{2}\left(\frac{w^{\hat{\beta}}(s,a)\pi(a|s)}{\beta(a|s)}\right).$$

Now consider three cases:

i) When $\hat{\beta}(a|s) \geq \epsilon$ and $\beta(a|s) > 0$, $w^{\hat{\beta}}(s,a) = 0$. So, $\mathcal{T}^\pi_{\text{FAC}} Q(s,a) = \mathcal{T}^\pi Q(s,a)$.

ii) When $\hat{\beta}(a|s) < \epsilon$ and $\beta(a|s) > 0$, $w^{\hat{\beta}}(s,a) > 0$. So, $\mathcal{T}^\pi_{\text{FAC}} Q(s,a) = \mathcal{T}^\pi Q(s,a) - \frac{\alpha}{2}\left(\frac{w^{\hat{\beta}}(s,a)\pi(a|s)}{\beta(a|s)}\right)$.

iii) When $\beta(a|s) = 0$, the ratio $\pi(a|s)/\beta(a|s) = \infty$ for any a with $w^{\hat{\beta}}(s,a)\pi(a|s) > 0$. Then, its negative becomes $-\infty$.

The only case requiring special care is that $\pi(a|s) > 0$ and $\beta(a|s) = 0$ and $w^{\hat{\beta}}(s,a) = 0$ simultaneously. The ratio $w^{\hat{\beta}}(s,a)/\beta(a|s)$ may cancel out for $\beta(a|s) = 0$ and $w^{\hat{\beta}}(s,a) = 0$. But, it is highly unlikely that we have $\beta(a|s) = 0$ and $w^{\hat{\beta}}(s,a) = 0$ simultaneously by the definition of the weight $w^{\hat{\beta}}(s,a)$.

□

F ADDITIONAL RESULTS

Table 3 presents evaluation results on the 50 singletask variants in the OGBench. For each task category, we select the best hyperparameters on its default task and evaluate the remaining tasks with the same ones. The detailed experimental settings are provided in Appendix G.3.

Table 3: Full evaluation results on the OGBench. (∗) indicates the default task in each task category. We report the final performance averaged over 8 seeds, with ± indicating the standard deviation.

	Gaussian Policies			Diffusion Policies			Flow Policies				
Task	**BC**	**IQL**	**ReBRAC**	**IDQL**	**SRPO**	**CAC**	**FAWAC**	**FBRAC**	**IFQL**	**FQL**	**FAC (Ours)**
antmaze-large-navigate-singletask-task1-v0 (∗)	0	48	91	0	0	42	1	70	24	80	**94.0**±3.0
antmaze-large-navigate-singletask-task2-v0	6	42	**88**	14	4	1	0	35	8	57	86.0±5.7
antmaze-large-navigate-singletask-task3-v0	29	72	51	26	3	49	12	83	52	93	**97.5**±2.1
antmaze-large-navigate-singletask-task4-v0	8	51	84	62	45	17	10	37	18	80	**89.5**±7.7
antmaze-large-navigate-singletask-task5-v0	10	54	90	2	1	55	9	76	38	83	**96.0**±4.8
antmaze-giant-navigate-singletask-task1-v0 (∗)	0	0	**27**	0	0	0	0	0	0	4	6.5±5.2
antmaze-giant-navigate-singletask-task2-v0	0	1	16	0	0	0	0	4	0	9	**37.5**±15.0
antmaze-giant-navigate-singletask-task3-v0	0	0	**34**	0	0	0	0	0	0	0	0.5±1.4
antmaze-giant-navigate-singletask-task4-v0	0	0	5	0	0	0	0	9	0	14	**20.0**±17.9
antmaze-giant-navigate-singletask-task5-v0	1	19	49	0	0	0	0	6	13	16	**50.5**±29.4
humanoidmaze-medium-navigate-singletask-task1-v0 (∗)	1	32	16	1	0	38	6	25	69	19	**71.5**±14.7
humanoidmaze-medium-navigate-singletask-task2-v0	1	41	18	1	1	47	40	76	85	**94**	88.0±12.1
humanoidmaze-medium-navigate-singletask-task3-v0	6	25	36	0	2	83	19	27	49	74	**95.5**±3.3
humanoidmaze-medium-navigate-singletask-task4-v0	0	0	15	1	1	5	1	1	1	3	**25.0**±9.0
humanoidmaze-medium-navigate-singletask-task5-v0	2	66	24	1	3	91	31	63	**98**	97	**98.0**±2.1
humanoidmaze-large-navigate-singletask-task1-v0 (∗)	0	3	2	0	0	1	0	0	6	7	**15.0**±19.8
humanoidmaze-large-navigate-singletask-task2-v0	0	0	0	0	0	0	0	0	0	0	0.0±0.0
humanoidmaze-large-navigate-singletask-task3-v0	1	7	8	3	1	2	1	10	**48**	11	20.5±18.4
humanoidmaze-large-navigate-singletask-task4-v0	1	1	1	0	0	0	0	0	1	2	**4.5**±11.2
humanoidmaze-large-navigate-singletask-task5-v0	0	1	**2**	0	0	0	0	1	0	1	1.5±2.1
antsoccer-arena-navigate-singletask-task1-v0	2	14	0	44	2	1	22	17	61	77	**82.0**±4.8
antsoccer-arena-navigate-singletask-task2-v0	2	17	0	15	3	0	8	8	75	88	**93.5**±4.2
antsoccer-arena-navigate-singletask-task3-v0	0	6	0	0	0	8	11	16	14	61	**62.5**±11.5
antsoccer-arena-navigate-singletask-task4-v0 (∗)	1	3	0	0	0	0	12	24	16	39	**53.0**±5.6
antsoccer-arena-navigate-singletask-task5-v0	0	2	0	0	0	0	9	15	0	36	**47.5**±15.3
cube-single-play-singletask-task1-v0	10	88	89	95	89	77	81	73	79	97	**99.0**±1.9
cube-single-play-singletask-task2-v0 (∗)	3	85	92	96	82	80	81	83	73	97	**100.0**±0.0
cube-single-play-singletask-task3-v0	9	91	93	99	96	98	87	82	88	98	**100.0**±0.0
cube-single-play-singletask-task4-v0	2	73	92	93	70	91	79	79	79	94	**98.5**±3.0
cube-single-play-singletask-task5-v0	3	78	87	90	61	80	78	76	77	93	**96.5**±3.3
cube-double-play-singletask-task1-v0	8	27	45	39	7	21	21	47	35	**61**	60.0±11.3
cube-double-play-singletask-task2-v0 (∗)	0	1	7	16	0	2	2	22	9	36	**37.5**±10.0
cube-double-play-singletask-task3-v0	0	0	4	17	0	3	1	4	8	22	**31.5**±10.8
cube-double-play-singletask-task4-v0	0	0	1	0	0	0	0	0	1	**5**	4.0±3.7
cube-double-play-singletask-task5-v0	0	4	4	1	0	3	2	2	17	19	**32.5**±6.2
scene-play-singletask-task1-v0	19	94	95	**100**	94	**100**	87	96	98	**100**	**100.0**±0.0
scene-play-singletask-task2-v0 (∗)	1	12	50	33	2	50	18	46	0	76	**100.0**±0.0
scene-play-singletask-task3-v0	1	32	55	94	4	49	38	78	54	**98**	97.0±2.8
scene-play-singletask-task4-v0	2	0	3	4	0	0	6	4	0	5	**58.0**±38.4
scene-play-singletask-task5-v0	0	0	0	0	0	0	0	0	0	0	**1.5**±4.2
puzzle-3x3-play-singletask-task1-v0	5	33	97	52	89	97	25	63	94	90	**100.0**±0.0
puzzle-3x3-play-singletask-task2-v0	1	4	1	0	0	0	4	2	1	16	**100.0**±0.0
puzzle-3x3-play-singletask-task3-v0	1	3	3	0	0	0	1	1	0	10	**100.0**±0.0
puzzle-3x3-play-singletask-task4-v0 (∗)	1	2	2	0	0	0	1	2	0	16	**100.0**±0.0
puzzle-3x3-play-singletask-task5-v0	1	3	5	0	0	0	1	2	0	16	**100.0**±0.0
puzzle-4x4-play-singletask-task1-v0	1	12	26	48	24	44	1	32	49	34	**52.0**±25.1
puzzle-4x4-play-singletask-task2-v0	0	7	12	14	0	0	0	5	4	**16**	7.5±7.5
puzzle-4x4-play-singletask-task3-v0	0	9	15	34	21	29	1	20	50	18	**62.0**±13.4
puzzle-4x4-play-singletask-task4-v0 (∗)	0	5	10	26	7	1	0	5	21	11	**35.0**±12.4
puzzle-4x4-play-singletask-task5-v0	0	4	7	**24**	1	0	0	4	2	7	5.0±5.1

G Experimental Details

We implement the proposed algorithm in JAX (Bradbury et al., 2018) on top of an official implementation of Park et al. (2025).

G.1 Benchmarks

We evaluate our proposed method on 50 singletasks of OGBench (Park et al., 2024) and 23 tasks of D4RL (Fu et al., 2020).

OGBench. We evaluate FAC across the following tasks from 10 navigation and manipulation environments:

- `antmaze-large-navigate-singletask-task{1,2,3,4,5}-v0`
- `antmaze-giant-navigate-singletask-task{1,2,3,4,5}-v0`
- `humanoidmaze-medium-navigate-singletask-task{1,2,3,4,5}-v0`
- `humanoidmaze-large-navigate-singletask-task{1,2,3,4,5}-v0`
- `antsoccer-arena-navigate-singletask-task{1,2,3,4,5}-v0`
- `cube-single-play-singletask-task{1,2,3,4,5}-v0`
- `cube-double-play-singletask-task{1,2,3,4,5}-v0`
- `scene-play-singletask-task{1,2,3,4,5}-v0`
- `puzzle-3x3-play-singletask-task{1,2,3,4,5}-v0`
- `puzzle-4x4-play-singletask-task{1,2,3,4,5}-v0`

OGBench provides a suite of environments and datasets for offline goal-conditioned RL. We use datasets from 5 navigation environments (`antmaze-large`, `antmaze-giant`, `humanoidmaze-medium`, `humanoidmaze-large`, `antsoccer-arena`) and 5 manipulation environments (`cube-single`, `cube-double`, `scene`, `puzzle-3x3`, `puzzle-4x4`). Each dataset offers 5 singletask variants. For offline RL evaluation, we use the reward-based singletask variants whose rewards are relabeled to align with the singletask specification. In navigation tasks, a semi-sparse reward function is used, the agent receives a reward of 0 on successfully reaching the goal position and -1 otherwise. In manipulation tasks, each singletask is composed of multiple subtasks, and the agent receives the negative of the number of unsuccessful subtasks as its reward.

The navigation environments include `antmaze` and `humanoidmaze`, where a quadrupedal agent and a humanoid one are required to reach goal positions in a given maze, and `antsoccer`, where a quadrupedal agent dribbles a ball to a goal position. The manipulation environments include `cube`, where a robot arm is required to pick and place colored cubes, and `scene`, where a robot arm executes a sequence of subtasks, and `puzzle`, where a robot arm solves a lights-out puzzle.

D4RL. We evaluate FAC across the following tasks from the MuJoCo, Antmaze, Adroit domains:

- `halfcheetah-medium-v2`
- `hopper-medium-v2`
- `walker2d-medium-v2`
- `halfcheetah-medium-replay-v2`
- `hopper-medium-replay-v2`
- `walker2d-medium-replay-v2`
- `halfcheetah-medium-expert-v2`
- `hopper-medium-expert-v2`
- `walker2d-medium-expert-v2`
- `antmaze-umaze-v2`
- `antmaze-umaze-diverse-v2`

- `antmaze-medium-play-v2`
- `antmaze-medium-diverse-v2`
- `antmaze-large-play-v2`
- `antmaze-large-diverse-v2`
- `pen-human-v1`
- `pen-cloned-v1`
- `door-human-v1`
- `door-cloned-v1`
- `hammer-human-v1`
- `hammer-cloned-v1`
- `relocate-human-v1`
- `relocate-cloned-v1`

D4RL provides standard offline RL datasets for locomotion (MuJoCo domain), navigation (Antmaze domain), and dexterous manipulation (Adroit domain). In the MuJoCo domain, we evaluate on `halfcheetah`, `hopper`, `walker2d` with three datasets `medium`, `medium-replay`, `medium-expert`. `medium` dataset consists of transition rollouts from a partially trained behavior policy, and `medium-replay` dataset is the full replay buffer of the behavior policy, and `medium-expert` dataset mixes trajectories from the medium and expert-level behavior policies. In the Antmaze domain, we use {`umaze`, `medium`, `large`}-sized mazes with two datasets `play` and `diverse`. The two datasets contain transitions to reach from start positions to goal positions. In the Adroit domain, we use `pen`, `door`, `hammer`, `relocate` tasks with two datasets `human` and `cloned`. The `human` dataset consists of tele-operated demonstrations, and the `cloned` dataset contains transition rollouts from a behavior-cloned policy trained on the `human` dataset.

G.2 Baselines

We primarily present the official reported performance from each baseline paper for benchmark datasets. For datasets that are not reported in the baseline papers, we refer to performances from other papers reporting performance for those datasets. For datasets without reported performances from other papers, we reproduced the results using an official implementation of baselines, tuning hyperparameters.

Specifically, all results for our method and the reproduced baselines are reported as the mean $\pm$ standard deviation over 8 random seeds.

For OGBench, we obtain the reported performances of BC, IQL, ReBRAC, IDQL, SRPO, CAC, FAWAC, FBRAC, IFQL, FQL from Park et al. (2025).

For D4RL-MuJoCo domain, we obtain the reported performance of TD3+BC, CQL from Kostrikov et al. (2021). Additionally, we reproduce FQL and tune a broader range of hyperparameters than the ones recommended in the original paper. The hyperparameters and training settings for FQL are summarized in Table 4 & 5. In particular, we consider normalizing Q-values in the actor objective, following TD3+BC (Fujimoto & Gu, 2021). Empirically, enabling or disabling this Q-normalization exhibits almost similar performance. To maintain consistency with default hyperparameter settings of FQL, we therefore report results obtained without the Q-normalization in the actor objective.

For D4RL-Antmaze domain, we obtain the reported performance of TD3+BC from Kostrikov et al. (2021), the reported performance of MCQ from Yeom et al. (2024), ones of SAC-RND from (Tarasov et al., 2023), and the reported performance of CAC from the original paper and Park et al. (2025).

For D4RL-Adroit domain, we obtain the reported performance of TD3+BC, IQL, SAC-RND from Tarasov et al. (2023) and the reported performance of IDQL, SRPO, CAC from Park et al. (2025).

Table 4: Hyperparameters for FQL on the D4RL-MuJoCo domain.

Hyperparameters	**Value**
Learning rate	0.0003
Optimizer	Adam (Kingma & Ba, 2014)
Gradient Steps	1000000
Minibatch size	256
MLP dimension	[512, 512, 512, 512]
Nonlinearity	GELU (Hendrycks & Gimpel, 2016)
Target network smoothing coefficient	0.005
Discount factor γ	0.99
Flow steps	10
Flow time sampling distribution	Unif($[0, 1]$)
Clipped double Q-learning	True
BC coefficient λ	Table 5
Q-normalization in actor loss	False

Table 5: BC Coefficient λ for FQL on the D4RL-MuJoCo domain.

Tasks (Datasets)	**BC coefficient λ**
halfcheetah-medium-v2	3.0
hopper-medium-v2	100.0
walker2d-medium-v2	300.0
halfcheetah-medium-replay-v2	30.0
hopper-medium-replay-v2	100.0
walker2d-medium-replay-v2	300.0
halfcheetah-medium-expert-v2	30.0
hopper-medium-expert-v2	100.0
walker2d-medium-expert-v2	300.0

G.3 Our Algorithm

G.3.1 Implementation Details

We describe the implementation details for flow actor-critic (FAC).

Network architecture. We use a [512, 512, 512, 512]-sized MLP for all neural networks. We apply layer normalization (Ba et al., 2016) to critic networks. The critic network Q_ϕ takes a state-action pair (s, a) as input and outputs a scalar estimate of the expected return under the one-step flow policy π_θ. The behavior proxy policy $\hat{\beta}_\psi$ takes state s, Gaussian noise $z \in \mathbb{R}^{d_\mathcal{A}}$, and flow time $u \in [0, 1]$ as inputs and outputs the velocity field v_u. The one-step flow policy π_θ takes state s and Gaussian noise z and then outputs a single action $a_\theta(s, z)$.

Flow matching proxy. We use the flow matching objective (6) (Lipman et al., 2022; Park et al., 2025). We use the Euler method with 10 steps to sample actions $a_\psi(s, z)$ and evaluate behavior proxy density $\hat{\beta}_\psi(a|s)$.

Specifically, the proxy density can be evaluated exactly via the Instantaneous Change of Variables (Chen et al., 2018), which requires the divergence of the velocity field $\nabla_{a_u} \cdot v_u(a_u; s, u)$. This is the trace of a Jacobian with respect to a_u, and it scales as $\mathcal{O}((d_\mathcal{A})^2)$ (Grathwohl et al., 2018), which may be prohibitive for high-dimensional action spaces. To reduce computational cost without introducing bias, we can approximate the divergence using Hutchinson's unbiased estimator (Hutchinson, 1989; Adams et al., 2018), lowering the complexity to $\mathcal{O}(d_\mathcal{A})$. In practice, we adopt this estimator for OGBench-Humanoidmaze and D4RL-Adroit domains, which have action space dimension $d_\mathcal{A} > 8$, and using the estimator exhibits performance comparable to exact evaluation.

Critic learning. We use twin critic networks to improve stability (Fujimoto et al., 2018). For the empirical Bellman operator in the critic loss, we can aggregate them either by the arithmetic mean of the two Q values or by their minimization. We present the aggregation choices in Table 6.

One-step flow policy learning. Following (Fujimoto & Gu, 2021), we apply the Q-value normalization in the one-step flow actor loss to balance the Q-maximizing term and the distance regularization term, i.e., $\max_{\pi_\theta} \mathbb{E}_{a\sim\pi_\theta}\left[\frac{Q_\phi(s,a)}{|Q_\phi|}\right] + \lambda\, D(\pi_\theta, \hat{\beta}_\psi)$, where $|Q_\phi| = \frac{1}{M}\sum_m |Q_\phi(s_m, a)|$ with mini-batch $\{s_m\}_{m=1}^M$ and actions $a \sim \pi_\theta(\cdot|s_m)$. In practice, this normalization reduces the sensitivity to the actor regularization coefficient λ and enables a common tuning range across offline RL tasks.

Training and Evaluation. We train FAC for 1M gradient steps on all the tasks of OGBench and D4RL. We evaluate it every 100K updates using 25 episodes. At evaluation, the actor samples exactly one action at each time step. One thing to note is that we report the last performance measured after 1M gradient steps.

G.3.2 HYPERPARAMETERS

All hyperparameters are summarized in Table 6 & 7. The critic penalization coefficient α and actor regularization coefficient λ are important in our method. The entire set of the two hyperparameters $(\alpha,\ \lambda)$ used across all tasks is $\alpha \in \{0.05, 0.1, 0.5, 1.0, 5.0, 10.0\}$ and $\lambda \in \{0.0003, 0.001, 0.003, 0.1, 0.3, 1.0, 3.0, 10.0\}$. For task-specific hyperparameters and a narrow range for tuning hyperparameters, please refer to Table 7.

Another important hyperparameter is the type of dataset-driven threshold ϵ in the weight $f^{\hat{\beta}_\psi}(s, a)$. As described, for datasets where the state coverage is broad but the action diversity is limited, we use the dataset-wide constant threshold (OGBench: Antmaze, Antsoccer tasks / D4RL: Antmaze tasks). For all remaining tasks, we use the batch-adaptive threshold.

Other hyperparameter is the use of clipped double Q-learning. With twin critics, the empirical Bellman target can be computed using either the minimization or the mean of the two Q values for state-action pairs. We use the minimization on the D4RL tasks, and the mean on OGBench tasks other than Antmaze tasks.

Table 6: Hyperparameters for our method

Hyperparameters	**Value**
Learning rate	0.0003
Optimizer	Adam (Kingma & Ba, 2014)
Gradient Steps	1000000
Minibatch size	256
Epochs for flow proxy model	250 (default), 125 (`D4RL:MuJoCo`)
MLP dimension	[512, 512, 512, 512]
Nonlinearity	GELU (Hendrycks & Gimpel, 2016)
Target network smoothing coefficient ρ	0.005
Discount factor γ	0.995
Step count of the Euler method	10
Flow time sampling distribution	Unif($[0, 1]$)
Q-normalization in actor loss	True
Clipped double Q-learning	Min (`D4RL`: default / `OGBench:Antmaze`) Mean (`D4RL`: Null / `OGBench`: default)
Estimator for evaluating $\hat{\beta}_\psi(a\|s)$	Exact (`D4RL`: default / `OGBench`: default) Hutchinson's estimator with 8 probes (`D4RL:Adroit` / `OGBench:Humanoidmaze`)
Dataset-driven threshold ϵ	Batch-adaptive (`D4RL`: default / `OGBench`: default) Dataset-wide constant (`D4RL:Antmaze` / `OGBench:Antmaze,Antsoccer`)
Critic penalization coefficient α	Table 7
Actor regularization coefficient λ	Table 7

Table 7: Critic penalization coefficient α and actor regularization coefficient λ of FAC

(a) OGBench

Tasks (Datasets)	α	λ
antmaze-large-navigate-singletask-task{1,2,3,4,5}-v0	0.5	0.1
antmaze-giant-navigate-singletask-task{1,2,3,4,5}-v0	1.0	0.1
humanoidmaze-medium-navigate-singletask-task{1,2,3,4,5}-v0	0.5	0.3
humanoidmaze-large-navigate-singletask-task{1,2,3,4,5}-v0	0.5	0.1
antsoccer-arena-navigate-singletask-task{1,2,3,4,5}-v0	1.0	0.1
cube-single-play-singletask-task{1,2,3,4,5}-v0	0.5	10.0
cube-double-play-singletask-task{1,2,3,4,5}-v0	1.0	1.0
scene-play-singletask-task{1,2,3,4,5}-v0	1.0	1.0
puzzle-3x3-play-singletask-task{1,2,3,4,5}-v0	1.0	1.0
puzzle-4x4-play-singletask-task{1,2,3,4,5}-v0	5.0	0.3

(b) D4RL: MuJoCo domain

Tasks (Datasets)	α	λ
halfcheetah-medium-v2	0.05	0.0003
hopper-medium-v2	5.0	0.1
walker2d-medium-v2	5.0	0.03
halfcheetah-medium-replay-v2	0.05	0.0003
hopper-medium-replay-v2	5.0	0.1
walker2d-medium-replay-v2	5.0	0.1
halfcheetah-medium-expert-v2	0.5	0.003
hopper-medium-expert-v2	5.0	0.3
walker2d-medium-expert-v2	5.0	0.03

(c) D4RL: Antmaze domain

Tasks (Datasets)	α	λ
antmaze-umaze-v2	1.0	0.1
antmaze-umaze-diverse-v2	1.0	0.1
antmaze-medium-play-v2	0.5	0.03
antmaze-medium-diverse-v2	5.0	0.1
antmaze-large-play-v2	5.0	0.03
antmaze-large-diverse-v2	1.0	0.03

(d) D4RL: Adroit domain

Tasks (Datasets)	α	λ
pen-human-v1	10.0	0.3
pen-cloned-v1	1.0	0.1
door-human-v1	1.0	3.0
door-cloned-v1	5.0	10.0
hammer-human-v1	0.5	0.03
hammer-cloned-v1	0.5	1.0
relocate-human-v1	5.0	10.0
relocate-cloned-v1	5.0	10.0