DIRECT OPTIMAL ACTION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advancements in offline reinforcement learning have leveraged two key innovations: policy extraction from behavior-regularized actor-critic (BRAC) objective and the use of expressive policy architectures, such as diffusion and flow models. However, backpropagation through iterative sampling chains is computationally tricky and often requires policy-specific solutions and careful hyperparameter tuning. We observe that the reparameterized policy gradient of the BRAC objective trains the policy to clone an "optimal" action. Building on this insight, we introduce **Direct Optimal Action Learning (DOAL)**, a novel framework that directly learns this "optimal" action. Then, efficient behavior losses native to the policy's distribution (e.g., flow matching loss) can be used for efficient learning. Furthermore, we demonstrate that the traditional balancing factor between Q-loss and behavior-loss can be reinterpreted as a mechanism for selecting a trust region for the optimal action. The trust region reinterpretation yields a **Batch-Normalizing Optimizer**. This facilitates the hyperparameter search and makes it shareable across policy distributions. Our DOAL framework can be easily integrated with any existing Q-value-based offline RL methods. To control the impact of value estimation, our baseline models use simple behavior clone loss and implicit q-learning. We apply DOAL to Gaussian, Diffusion, and Flow policies. In particular, for Diffusion and Flow policies, we obtained strong baseline models by improving the **MaxQ Action Sampling**. Our results on 15 tasks from the OGBench and D4RL adroit datasets show that DOAL consistently improves performance compared against strong baseline models while simplifying hyperparameter search. Our best models achieved very strong performance. The code is available through Anonymous Github.

1 INTRODUCTION

Offline reinforcement learning (RL) aims at efficiently and effectively extracting policy from experience beyond the simple imitation learning(Lange et al., 2012; Levine et al., 2020). While learning from pre-collected trajectories avoids the costly environment interactions, for offline RL agents to perform better than a simple behavior cloning agent, value estimation and extracting information from the estimated value function is the key(Park et al., 2024). Yet, due to the lack of interaction, agents cannot extrapolate too much to avoid distribution shift. Hence, the success of offline RL depends on balancing the maximization the estimated Q value and behavior cloning (Haarnoja et al., 2018; Wu et al., 2019; Kostrikov et al., 2022; Tarasov et al., 2023).

Meanwhile, as the scale and diversity of offline datasets continue to grow, there is an increasing need for policies capable of modeling highly multi-modal and complex action distributions. Diffusion and flow matching models(Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b; Lipman et al., 2023; Lu et al., 2022; Sun et al., 2025) have recently been applied to Offline RL(Janner et al., 2022; Wang et al., 2023; Hansen-Estruch et al., 2023; Kang et al., 2023; Park et al., 2025c), and improved the modeling capacity of policy distribution.

However, efficient policy extraction requires reparameterized policy gradient through the Q value functionPark et al. (2024), and it is non-trivial for distribution involving an iterative sampling procedural(Park et al., 2025c; Kang et al., 2023; Fujimoto & Gu, 2021; Tarasov et al., 2023). While many solutions have been proposed, they usually involves some computation overhead, requires careful tuning of hyperparameter across environments or not as effective.

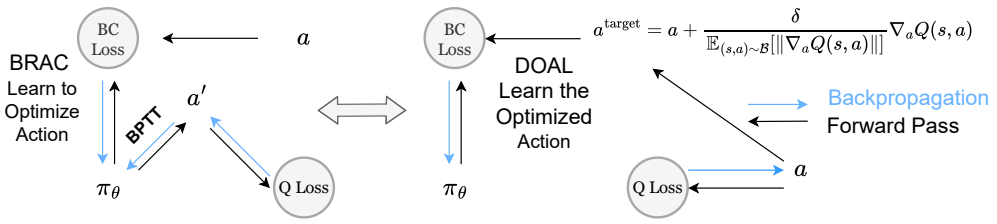

Figure 1: **DOAL Framework.** On the left side, we have the end2end re-parameterized policy gradient from the Q value function and the behavior clone loss. On the right side, we have our DOAL framework, where we extract an optimized a^{target} from Q, and then use efficient behavior clone loss from any policy distribution.

In this work, we provide a simple, effective and universal framework for policy extraction from the Q value function. As shown in Figure 1, we observe that the reparameterized policy gradient w.r.t. *Behavior Regularized Actor Critic (BRAC)* objective can be replaced by simple gradient w.r.t. behavior clone loss of an optimized action.

This observation has two consequences: **Direct Optimal Action Learning (DOAL)** instead of backpropagating end2end, we can optimize the action with q value and behaviour loss, then learn the optimized action with the efficient loss function that is native to the policy distribution (e.g., velocity matching loss); **Batch-Normalizing Optimizer** if learning from Q value function is approaching a better local solution, and Q value estimations are known to be not reliable, we should have a trust region δ for how much the optimized a^{target} can shift away from data distribution. Meanwhile, it is also desirable to the shift distance to be proportional to the gradient of Q value w.r.t. action. This can be realized by normalizing the shift with a batch average of gradient $a^{\text{target}} = a + \frac{\delta}{\mathbb{E}_{(s,a)\sim\mathcal{B}}[\|\nabla_a Q(s,a)\|]}\nabla_a Q(s,a)$, where $\mathcal{B}$ is the data mini-batch.

We are not aiming at producing the most performative offline RL algorithms, but studying the effectiveness of our policy extraction framework (in contrast to the end2end training). To isolate the effects of value estimation from policy extraction, we use the *Implicit Q Learning (IQL)* (Kostrikov et al., 2022). The value estimation in IQL doesn't interact with the policy extraction, making it an ideal choice for our study. Still, the learned Q-value can be used. For Gaussian policy, *Advantage-Weighted Regression (AWR)* can be used (Kostrikov et al., 2022). For flow and diffusion models, without learning from Q function in the training time, *MaxQ Sampling* is used for selecting action with maximal Q value in a sampled candidate sets. In order to have a solid baseline, we identified a key parameters that were previously overlooked hyperparameter: **the number of samples** for action candidates. This parameter directly influences the trade-off between the maximization bias of Q-value and coverage from samples.

Empirically, on 9 default tasks on OGBench and 6 Adroit tasks on D4RL datasets, the DOAL learned policies achieved consistent improvement over the their behavior clone baseline (with max Q sampling 2 or advantaged weighted regression 2) . In particular, for all algorithms in the same environment, the DOAL hyperparameters δ are shared. In all, the DOAL framework is an **efficient, effective** and **versatile** to extract policy distributions from the Q value.

2 Preliminaries

We consider a Markov decision process (MDP) $\mathcal{M}$ $(\mathcal{S}, \mathcal{A}, r, p, \gamma, \rho_0)$(Sutton & Barto, 1998), where $\mathcal{S}$ is the state space, $\mathcal{A} = \mathbb{R}^d$ is a d-dimensional continuous action space, $r(s, a)$ is the reward function, $p(s' \mid s, a)$ is the transition dynamics, $\gamma \in [0, 1)$ is the discount factor, and $\rho_0(s)$ is the initial state distribution.

In offline RL, the agent learns from a fixed dataset $\mathcal{D} = \{(s_i, a_i, s'_i, r_i)\}_{i=1}^N$ consisting of individual transitions rather than complete trajectories. The objective is to learn a policy $\pi_\theta(a \mid s)$ that maximizes the expected discounted return $J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, while avoiding distributional

shift caused by querying actions outside the support of $\mathcal{D}$, as there will be no online interaction to correct overoptimism.

Implicit Q-Learning. IQL (Kostrikov et al., 2022) avoids querying out-of-sample actions through expectile regression. Unlike SARSA-style methods (Brandfonbrener et al., 2021) that learn using the next actions from the dataset, IQL learns the value function of the current policy.

The value function $V_\psi(s)$ and the Q-function Q_ϕ are learned via:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\tau\left(Q_\phi(s,a) - V_\psi(s)\right)\right], \quad \mathcal{L}_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma V_\psi(s') - Q_\phi(s,a)\right)^2\right]. \tag{1}$$

where $L_2^\tau(u) = |\tau - \mathbb{I}(u < 0)|u^2$. For policy extraction, standard IQL uses Advantage-Weighted Regression (AWR) (Peng et al., 2019):

$$\mathcal{L}_{\text{AWR}}(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp\left(\beta(Q_\phi(s,a) - V_\psi(s))\right)\log\pi_\theta(a|s)\right], \tag{2}$$

where β controls the strength on the advantage weighting. All our models use IQL for value estimation.

Flow Matching. Flow Matching (FM) establishes a deterministic path $(p_t)_{t\in[0,1]}$ that continuously transforms a simple source distribution p_1 into the target behavior distribution p_0, with each p_t defined over $\mathbb{R}^d$ (Lipman et al., 2023; 2024). [1]

We adopt the most simple instantiation of FM, utilizing linear interpolation paths with uniform time sampling (Lipman et al., 2024; Park et al., 2025c). For an action $a_0 \sim \pi(a_0)$, the objective is to learn a time-dependent velocity field $v_\theta(a,t) : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ through the following regression loss:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{a_0\sim q(a_0), a_1\sim\mathcal{N}(0,I), t\sim\mathcal{U}[0,1]}\left[\|v_\theta(a_t,t) - (a_0 - a_1)\|^2\right], \quad a_t = (1-t)a_0 + ta_1 \tag{3}$$

This formulation ensures training stability while admitting efficient sampling through explicit Euler discretization of the underlying flow ODE:

$$a_{t-\Delta t} = a_t + \Delta t \cdot v_\theta(a_t, t), \quad \Delta t = \frac{1}{N} \tag{4}$$

where N represents the number of flow steps.

TrigFlow. We adapt the TrigFlow framework (Lu & Song, 2025) to train diffusion policy(Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b), defining the forward diffusion process as follows: given $a_0 \sim p(a_0)$ and $z \sim \mathcal{N}(0, I)$, the noisy sample at time t is given by

$$a_t = \cos(t)a_0 + \sin(t)z, \quad t \in [0, \tfrac{\pi}{2}] \tag{5}$$

with $a_{\pi/2} \sim \mathcal{N}(0, I)$. The model can be trained to minimize the prediction error:

$$\mathcal{L}_{\text{diffusion}}(\theta) = \mathbb{E}_{a_0\sim p(a_0), z\sim\mathcal{N}(0,I), t\sim\mathcal{U}[0,\pi/2]}\left[\|f_\theta(a_t,t) - a_0\|_2^2\right], \tag{6}$$

$$f_\theta(a_t,t) = \cos(t)a_t - \sin(t)\cdot F_\theta(a_t,t), \tag{7}$$

where F_θ is a learned network and the f_θ naturally satisfies the boundary condition $f_\theta(a_0, 0) = a_0$. We implement a first-order DDIM(Song et al., 2021a) sampling scheme for efficient inference:

$$a_t = \cos(k-t)\cdot a_k - \sin(k-t)\cdot F_\theta(a_k, k), \tag{8}$$

where k is the previous time step. In our paper, we divides the steps evenly into 10.

Max Q Sampling. A principled strategy orthogonal to behavior-regularized actor-critic frameworks is a resampling mechanisms(Ghasemipour et al., 2021; Chen et al., 2023; Hansen-Estruch et al., 2023). Instead of training the action policy with Q value, Max Q sampling leverages the estimated Q_ϕ in the inference time. Formally, given a proposal distribution $\pi_\theta(a|s)$ and a target criterion $Q_\phi(s,a)$, the Max Q sampling procedure select the optimal action from a set of samples:

$$a = \underset{a^{(1)},\ldots,a^{(n_{\text{sample}})}}{\arg\max} \; Q_\phi(s, a^{(i)}), \quad a^{(i)} \sim \pi_\theta(s) \tag{9}$$

[1] In the usual, flow matching setting p_1 is the data distribution, but we want to make it consistent with diffusion model notations.

We provide ablation studies in Section 5 to **show that larger n_{sample} values do not necessarily yield better performance.**

Behavior-Regularized Actor-Critic (BRAC). Behavior-regularized actor-critic methods form a family of effective offline RL approaches that combine value function learning with behavior regularization (Haarnoja et al., 2018; Wu et al., 2019; Tarasov et al., 2023). The critic loss follows equation 1. The actor loss, defined in our implementation, integrates policy improvement and behavior regularization:

$$\mathcal{L}^{\text{BRAC}}_{\pi,\text{BCLoss}}(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[-Q_\phi(s,\pi_\theta(s)) + \alpha \cdot \text{BCLoss}(\pi_\theta(s), a)\right], \tag{10}$$

where α is a hyperparameter balancing policy improvement and behavior constraint and BCLoss is a behavior clone loss, e.g. $\|\pi_\theta(s) - a\|^2$ or the velocity matching loss in the previous section. While the BARC objective is almost necessary, it has been shown to be highly sensitive to α (An et al., 2021; Chen et al., 2024b; Fang et al., 2025; Gao et al., 2025), requiring extensive per-task tuning.

The problem for applying Equation 10 to a diffusion/flow-based policy is the computational costly iterative sampling chain(Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b). There exists many works to circumvent this issue, see Section 6 for more discussion. For notational simplicity, we drop the mention of noise in the main text, as that can be easily integrated or we considering a joint state $s' = (s, z)$ for wherever noise is needed. Now, we present our framework for learning from the Q estimator.

3 Direct Optimal Action Learning

In deep learning, there is a doctrine that all we need to train neural networks is a differentiable loss w.r.t. the parameters (Rumelhart et al., 1995). In the context of learning policy with iterative sampling, the computational cost becomes too high. In fact, it is not entirely clear what the learned policy represents aside from it balances behavior regularization and expected return maximization. We present two motivations to derive our Direct Optimal Action Learning framework, and show how it can naturally yields more interpretable and robust hyperparameter choice.

3.1 The Optimal Action

From an intuitive standpoint, an actor policy should learn to produce actions that have high estimated Q-values. When learning from a static dataset, this search for high-value actions should naturally be centered around the actions already present in the data. This leads to a straightforward idea: instead of indirectly learning a policy that optimizes a complex objective, we can directly learn the optimal action itself. Our framework, **Direct Optimal Action Learning (DOAL)**, is founded on this principle.

From a formal perspective, this idea emerges from a re-interpretation of the policy gradient in standard behavior-regularized actor-critic methods.:

Proposition 1 (Equivalence of Policy Gradients)**.** *Let $\pi_\theta(s)$ be a deterministic differentiable policy parameterized by θ and $Q_\phi(s,a)$ be a differentiable action-value function. Consider the behavior-regularized objective:*

$$J_Q(\theta) \triangleq \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[Q_\phi(s,\pi_\theta(s)) - \alpha\|\pi_\theta(s) - a\|_2^2\right] \tag{11}$$

where $\alpha > 0$ is a regularization coefficient. The gradient of this objective with respect to θ is equivalent to the gradient of a simpler squared-error objective $J_{target}(\theta)$:

$$\nabla_\theta J_Q(\theta) = \nabla_\theta J_{target}(\theta) \triangleq \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\nabla_\theta\left(-\alpha\,\|\pi_\theta(s) - a_{target})\|_2^2\right)\right] \tag{12}$$

$$a^{\text{target}} = a + \frac{1}{2\alpha}\nabla_{a'}Q_\phi(s,a')\big|_{a'=\pi_\theta(s)} \tag{13}$$

The proof is a straightforward application of the chain rule (see Appendix A.2). This equivalence reveals that training with the BRAC objective is implicitly minimizing the distance between the policy's output $\pi_\theta(s)$ and a target action a_{target}.

This formulation, however, presents a conceptual inconsistency. The target a^{target} is constructed by taking a gradient ascent step from the data action a, but the gradient $\nabla_{a'} Q_\phi$ is evaluated at the policy's output $\pi_\theta(s)$. This requires sampling from the policy at training time and creates a mismatch between the point of expansion (a) and the point of evaluation ($\pi_\theta(s)$). A more direct approach would be to perform the gradient ascent step using the gradient evaluated at the data action a, which would define a static target for each data point (s, a) and eliminate the need for sampling.

This insight is the cornerstone of DOAL. By defining the target action directly from the data and Q value, we decouple the target computation from the policy being trained. The major practical benefit is that we are no longer constrained by the need to sample a full action during training. Consequently, we can leverage more powerful generative modeling techniques to learn the action policy, such as training flow-based models with flow matching losses or diffusion models with their diverse loss functions.

3.2 Batch-Normalizing Optimizer

A challenge with BRAC-style methods is the sensitivity of the regularization coefficient α, which often requires careful tuning across several orders of magnitude (Park et al., 2025c; Kumar et al., 2020). Our re-interpretation of the objective as learning a target action raises a critical question: what is the appropriate magnitude for the update from the data action a to the target action a_{target}?

In offline reinforcement learning, the learned Q-function is merely an estimator, and it is crucial to remain conservative to avoid distribution shift to out-of-support actions where the Q-function is unreliable (Kostrikov et al., 2022; Kumar et al., 2020; Tarasov et al., 2023). Therefore, instead of an arbitrary step size dictated by α, we should define a **statistical trust region** for our action optimization. We can achieve this by setting a fixed expected magnitude for the update vector $g(s, a) = a_{\text{target}} - a$. Specifically, we desire two conditions for the update vector $g(s, a)$:

1. The update should be in the direction of the Q-function's gradient at the data action: $g(s, a) \propto \nabla_a Q_\phi(s, a)$.
2. The expected squared magnitude of the update over the dataset should be a constant, which we denote as δ: $\mathbb{E}_{(s,a)\sim\mathcal{D}}[\|g(s, a)\|_2] = \delta$.

These two conditions uniquely determine the update vector, as shown in the following proposition.

Proposition 2 (Batch Normalized Update)**.** *To satisfy the conditions $g(s, a) \propto \nabla_a Q_\phi(s, a)$ and $\mathbb{E}_{(s,a)\sim\mathcal{D}}[\|g(s, a)\|_2] = \delta$, the action update vector $g(s, a)$ is defined as:*

$$g(s, a) = \frac{\delta}{\mathbb{E}_{(s',a')\sim\mathcal{D}}\left[\|\nabla_{a'} Q_\phi(s', a')\|_2\right]} \cdot \nabla_a Q_\phi(s, a) \tag{14}$$

where $\delta > 0$ is a hyperparameter representing the desired expected squared magnitude of the action update, and the expectation in the denominator is computed over a batch of data.

This formulation replaces the obscure hyperparameter α with an interpretable one, δ, which directly controls the expected squared magnitude of the action update. In practice, we use the batch statistics as estimator, so we have $\mathbb{E}_{(s',a')\sim\mathcal{B}}\left[\|\nabla_{a'} Q_\phi(s', a')\|_2^2\right]$, where $\mathcal{B}$ is the current mini-batch. This "batch-normalization" of the action gradients provides a more stable and robust training target, alleviating the need for extensive hyperparameter sweeps.

Why α varies so much? On a flip-side, tuning α in BRAC is similar to finding our statistical trust region. Therefore, as the reward function varies across environments, the gradients of Q varies a lot. This factor means even actions all live in a box, the statistical trust regions are of similar size, the optimal α could still vary hugely.

3.3 The DOAL Objectives

In this paper, we use IQL for value estimation. However, this is not an necessity, we make this choice for better controlled study of policy extraction. We only alter the actor loss :

$$J_{\text{DOAL}}(\theta) = \alpha \cdot \mathbb{E}_{(s,a)\sim\mathcal{D}} \text{BCLoss}(\pi_\theta(s), a^{\text{target}}) \tag{15}$$

$$a^{\text{target}} := a + \frac{\delta}{\mathbb{E}_{(s',a')\sim\mathcal{B}}\left[\|\nabla_{a'} Q_\phi(s', a')\|_2\right]} \cdot \nabla_a Q_\phi(s, a) \tag{16}$$

Algorithm 1 Direct Optimal Action Learning (DOAL) with IQL

Require: Dataset $\mathcal{D}$, policy parameters θ, Q-function parameters ϕ, , Value-function parameters ψ

1: **repeat**
2: Update ϕ, ψ using Implicit Q Learning as inEquation 1
3: Update θ using Equation 15 for BCLoss of Choice
4: **until** convergence

where BCLoss can be any distribution matching losses. In terms of computational overhead, as the gradient $\nabla_a Q_\phi(s,a)$ will be called explicit or implicitly at least once to extract first order structure information from Q_ϕ. The DOAL objective is at least as efficient as any algorithms that utilize $\nabla_a Q_\phi(s,a)$. Notice that we still keep the α parameter, because we believe this parameter also controls the learning rate of actor. We copy this parameter from Park et al. (2025c) for all our experiments. In summary, the overall algorithm for DOAL is given in Algorithm 1.

4 MaxQ Sampling needs Balancing

As our DOAL framework learn from optimized action based on $\nabla_a Q(s,a)$, we consider models without learning from $\nabla_a Q(s,a)$ as our baseline models. In such case, $Q(s,a)$ can still be used for weighting the behavior clone loss and performing actions selection, as we discussed Section 2 and Section2. To have a solid baseline for comparison, we argue that n_{sample} a crucial hyper-parameter that were previously overlooked.

When taking $n_{\text{sample}} = 1$, we clearly recovered the actor distribution and completely lost access to information from Q_ψ. On the other extreme, Some earlier work suggested the bigger n_{sample}, the better (Ghasemipour et al., 2021). However, while a^* maximize the Q value estimator Q_ψ as $n_{\text{sample}} \to +\infty$, there exists maximization bias such that the maximum Q value will be overestimated due to noise in the estimator. While we are interested in the maximizer a^* not the maximized Q value, this still is problematic.

Proposition 3 (Informal)**.** *Consider countably many actions $a_1, a_2, \ldots$. For each i, the Q-estimator is independent Gaussian: $\widehat{Q}(a_i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$, with bounded means $\sup_i \mu_i < \infty$ and infinitely many actions having nontrivial noise ($\sigma_i \geq c > 0$). Draw n i.i.d. actions from any policy with full support and pick the one with the largest observed $\widehat{Q}$. As $n \to \infty$: (i) the selected action eventually leaves every fixed finite set (its index drifts to infinity); (ii) the selected $\widehat{Q}$ value diverges to $+\infty$ (driven by extreme positive noise); (iii) the probability of picking any true-mean maximizer tends to* 0*.*

Proof Intuition. The maximum of m i.i.d. Gaussian draws grows like $\mu_i + \sigma_i\sqrt{2\log m}$. With infinitely many actions having $\sigma_i > 0$, some action will realize an extremely large positive fluctuation and dominate the max, regardless of the bounded means. Thus, making n_{sample} very large pushes selection toward noise outliers rather than toward actions with the highest μ_i, even when the estimator is unbiased.

The above analysis shows that increasing n_{sample} can exacerbate maximization bias: as $n_{\text{sample}} \to \infty$, $\max$-selection over noisy Q-estimates systematically prefers actions with large positive noise realizations, independent of the learned Q.

While it is hard to precisely characterize the resulting distribution from MaxQ sampling, the more samples we have the more data coverage we have. In the case where action policy is value agnostic, having multiple samples is necessary to ensure mode coverage so that Q function can provide a selection. Yet, if we sample too much, the stochasticity in Q_ψ dominates and we might get "good" actions due to overestimation rather than being reliable.

Pragmatically, n_{sample} balances the conflict between the in distribution behaviour and Q value estimator in the inference time. To our knowledge, this is an overlooked issue in the literature, where the consensus is that the n_{sample} is trading off between computation and accuracy. . As we will show in the experiments, this indeed improved our baseline models significantly.

Table 1: **Full offline RL results.** We present the full results on the OGBench and D4RL tasks. They are the default single-task in each environment. The results are averaged over 4 seeds on D4Rl and 3 seeds on OGBench. The ReBRAC*, IQL*, FQL*, IFQL* results are collected from Flow Q learning(Park et al., 2025c).We treat each return statement as an independent output and compute the standard deviation across all such outputs, rather than only considering the last three return values as in the original approach.

	Gaussian Policies			Flow Policies				Diffusion Policies		
Task	ReBRAC*	IQL*	DIQL	FQL*	IFQL*	IFQL	DIFQL	Trigflow	TrigflowQL	DTrigflow
antmaze-large-navigate	**91** ±10	48 ±9	66 ±10	80 ±8	24 ±17	66 ±26	77 ±17	51 ±28	81 ±10	77 ±24
antmaze-giant-navigate	**27** ±22	0 ±0	0 ±0	4 ±5	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0
humanoidmaze-medium-navigate	16 ±9	32 ±7	52 ±9	19 ±12	**69** ±19	67 ±32	56 ±7	63 ±8	67 ±5	65 ±7
humanoidmaze-large-navigate	2 ±1	3 ±1	8 ±3	7 ±6	6 ±2	7 ±10	9 ±4	6 ±5	6 ±6	6 ±6
antsoccer-arena-navigate	0 ±0	3 ±2	13 ±8	39 ±6	16 ±9	38 ±12	31 ±16	63 ±18	34 ±6	35 ±18
cube-single-play	92 ±4	85 ±8	80 ±3	97 ±2	73 ±3	11 ±27	18 ±4	89 ±4	87 ±4	19 ±25
cube-double-play	7 ±3	1 ±1	6 ±25	36 ±6	9 ±5	11 ±27	18 ±4	14 ±26	15 ±25	19 ±25
scene-play	50 ±13	12 ±3	28 ±7	76 ±9	0 ±0	22 ±12	58 ±26	45 ±18	58 ±12	61 ±18
puzzle-3x3-play	2 ±1	2 ±1	0 ±0	16 ±5	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0
puzzle-4x4-play	10 ±3	5 ±2	10 ±1	11 ±3	21 ±11	25 ±1	20 ±7	20 ±6	4 ±3	24 ±5
pen-human-v1	103	78	50 ±14	53 ±6	71 ±12	71 ±8	70 ±1	68±13	68 ±12	64 ±8
pen-cloned-v1	103	83	59 ±3	74 ±11	80 ±11	76 ±13	85 ±12	65 ±4	60 ±4	60 ±6
pen-expert-v1	152	128	125 ±7	142 ±6	139 ±5	138 ±2	136 ±5	135 ±3	135 ±4	141 ±5
door-expert-v1	106	107	103 ±4	104 ±1	104 ±2	104 ±2	102 ±2	104 ±2	104 ±2	102 ±2
hammer-expert-v1	134	129	106 ±44	125 ±3	117 ±9	97 ±17	120 ±6	93 ±16	100 ±12	97 ±18
relocate-expert-v1	108	106	100 ±4	107 ±1	104 ±3	103 ±5	103 ±7	104 ±2	103 ±2	103 ±3

5 Experiments & Discussion

In this section, we empirically examine the effectiveness of our DOAL framework for extracting information from $\nabla_a Q_\phi(s, a)$. Our baseline models still benefit from $Q_\phi(s, a)$ either through advantage weighted regression or max-Q sampling. We also compare against efficient diffusion policy learning Kang et al. (2023).

Benchmarks We conduct a comprehensive evaluation of our method on two challenging offline RL benchmarks: 1) The D4RL 6 Gymnasium tasks, focusing on the complex pen, hammer, relocate, and door environments with expert, human, and cloned dataset qualities. 2) 9 tasks from the more diverse and demanding OGBench suite. To ensure a fair comparison with reward-maximizing algorithms, we utilize the variant of OGBench.The selection of this 6+9 tasks are due to the fact, no current simple algorithms can work well.

Models We have three representative and strong baseline algorithms with our method: IQL, A popular implicit constraint algorithm; IFQL, A flow-based policy learning method; TrigFlow, our proposed efficient generative policy. For each one of them, we have the DOAL versions DIQL, DIFQL, and DTrigflow (see Appendix A.3 for details). Furthermore, we have TrigflowQL that uses the BRAC objective but uses one-step sampling $a' = f_\theta(a_t, t) = \cos(t)a_t - \sin(t) \cdot F_\theta(a_t, t)$, from corrupted a_t.

All our models use the same IQL hyperparameters for training Q value function, therefore we are only comparing policy extraction . Environment dependent hyperparams $\delta, n_{\text{sample}}$ are searched based on Trigflow and DTrigflow, but shared across algorithms $\delta, \alpha, n_{\text{sample}}$.

Evaluation Following Park et al. (2025c) average last three for OGbench 1m steps, the last one for adroit 500k last. For each environment and algorithm, we report the normalized score as defined by the respective benchmark. We train multiple seeds for each run, but the seed is shared across algorithms, we aim at providing a stable and representative measure of relative performance.

5.1 Main Results

The first observation we can made from Table 1 is that we made really strong baseline models of IFQL and Trigflow. In particular, our number of samples tuned IFQL improved over the original paper results significantly on OGBench. On the D4RL, the results are less clear. As those tasks are highly random in its' own. In our prelimnary experiments. We re-run testing with the same model, and we observed over 20 std across different runs. The adroit benchmark needs more investigation to useful for algorithm evaluation.

There is no clear advantage between our models Dtrigflow and TrigflowQL. As for comparing against other models, while we are not aiming at pushing the most performative model, [2] we ob-

[2]There are many recent works that train flow/diffusion with additional consistency constraints that could further stabilize the training.

tained a very strong results. It is still clear that Rebrac model has advantage in many tasks. However, as we discussed we pick IQL for controlled study, it is possible to adopt Rebarc in the future. In particular, the behaviour regularized target could be similarly approched by DOAL framework. While it is not shown here, many runs actually achieved much higher performance during the training then collapsed. Yet, we are not able to identify a conjecture for explanation yet. We will publish the wandb log with associated code after the anonymous period.

5.2 Hyperparameter Search

We use a single set of hyperparameters (including the untuned regularization coefficient α that serves as learning rate controller for our purpose) for each environment across all augmented algorithms (DIQL, DFQL, DTrigFlow). To ensure fairness, we also apply this same set of environment-specific hyperparameters to the original base algorithms (IQL, FQL, TrigFlow) when run on that environment. This design choice demonstrates that our performance gains are not from extensive hyperparameter tuning but from the stable optimization dynamics of the DOAL framework itself. All other network architecture and training details (e.g., learning rates) are kept consistent with the original implementations of the base algorithms.

Max Q sampling n^{sample} To identifying the best number of samples, we run Trigflow for three seeds. Then only do the re-testing with the last epoch saved model with different n^{sample}. We attempted (1,2,4,8,16,32,63,128) but very rarely, we have over 32 as the optimal. In Table 2, we have three selected environments where mid-range number of n^{sample} is preferred. In fact, in Ghasemipour et al. (2021) , their experiments also find cases where larger n^{sample} hurts.

Envs	1	2	4	8	16	32	64	128
cube-double-play	0	1	9	15	17	13	13	13
scene-play-v0	1	10	43	47	57	54	45	40
puzzle-3x3-play	0	2	5	9	5	6	2	1

Table 2: Success Rates of Sample Checkpoint with Different n^{sample}

Choose Statistical Trust Region δ For choosing a candidates for δ. In OGBench Park et al. (2025a), actions are bounded in [-1,1] box of varying dimensionality. The statistical trust region should be related to how reliable are the Q value estimation at data point and how well can neural network generalize. As we are using IQL for value estimation and the same value net in all our experiments, the only thing varying is the datasets. For OGBench experiments, we choose δ from $(0.03, 0.1, 0.3)$, and for d4rl experiments, we choose from $(0.003, 0.01, 0.03)$ as we see from α in the FQL paper Park et al. (2025c) tends to be extremely large. See the Anonymous Github for the exact hyperparameter. In Table 3, we present a few representative environments and their average $||\nabla_a Q(s,a)||$,

Envs	pen-cloned-v1	pen-expert-v1	scene-play	antmaze-large-navigate
$\|\|\nabla_a Q(s,a)\|\|$	279.99	64.99	15.58	0.55
α	10000	3000	300	10
δ	0.03	0.003	0.03	0.1

Table 3: Sample Environments with the Observed Mean $||\nabla_a Q(s,a)||$, α, and δ.

optimal α Park et al. (2025c) and δ we selected. As you can see the larger the gradient, the larger the selected α and it ranges across three orders of magnitude. Meanwhile, our hyperparameter δ is relatively stable and easier to search for.

6 Related Work

The training of multi-step generative policies (e.g., diffusion or flow models) in offline reinforcement learning presents significant computational challenges, predominantly stemming from the need for backpropagation through time (BPTT) during policy gradient estimation. Recent work has demonstrated the efficacy of diffusion models as expressive policy classes (Frans et al., 2025). We now critically examine existing methodological paradigms to delineate the fundamental limitations imposed

by BPTT, thereby contextualizing the conceptual novelty of our Direct Optimal Action Learning (DOAL) framework.

Long Horizon Challenge Problems For challenging tasks that most algorithm cannot address, like antmaze-giant-navigate, (Park et al., 2025b) looked at the the horizon reducation issue. While this is orthogonal to the our problem, but such methods might be necessary for offline RL to work in challenging environments.

Accelerated Sampling Techniques. A prominent research direction focuses on accelerating the sampling process of pre-trained generative policies. Methods such as Efficient Diffusion Policy (EDP) (Kang et al., 2023) reformulate the reverse denoising process to estimate the target action a_0 in a single step. While these approaches achieve notable improvements in sampling efficiency, they remain inherently approximate, as they seek to mimic the output of a computationally intensive multi-step generative procedure. In contrast, DOAL fundamentally circumvents this approximation by directly regressing toward the analytically computed optimal action a^*. Our approach not only matches the sampling efficiency of accelerated methods but also provides a more stable and targeted learning signal, being grounded in a principled one-step optimization objective derived from regularized policy improvement.

Value Guidance Methods. A substantial body of work in offline RL leverages diffusion models to approximate the behavior policy underlying the dataset. One prevalent strategy involves using the gradient of Q-value functions to guide the action generation process. This includes techniques such as Q gradient guidance oor energy-based guidance paradigms, exemplified by QGPO (Wang et al., 2023), SFBC (Chen et al., 2023), EDA (Chen et al., 2024a), and QVPO (Ding et al., 2024). An alternative, more straightforward approach modulates the policy by re-weighting transition samples based on their estimated values, as seen in advantage-weighted regression (AWR). A common drawback of these guidance-based techniques is their reliance on delicate hyperparameter tuning (e.g., guidance scales) to balance the often competing objectives of data fidelity and value maximization, resulting in a complex and sometimes unstable optimization landscape.

DOAL presents a fundamentally simpler and more cohesive alternative. It collapses the aforementioned complexity into a single, unified objective derived from a closed-form regularized policy improvement step. The key hyperparameter admits a clear interpretation as a trust-region radius, enabling a stable and interpretable interpolation between conservative imitation and aggressive value maximization without resorting to ad-hoc guidance heuristics.

Policy Distillation and Actor-Critic Methods. A third category of approaches focuses on distilling a more efficient policy from value estimates or planning results. This lineage distill the outcomes of planning procedures using value ensembles (An et al., 2021; Hansen-Estruch et al., 2023). Contemporary methods like DAC (Fang et al., 2025) and BDPO (Gao et al., 2025) further integrate diffusion models within an actor-critic framework.

DOAL distinguishes itself through its conceptual and computational directness. The target action a^{target} is computed analytically in a single step, eliminating the need for iterative distillation or complex optimization. This not only reduces computational burden but also provides a cleaner and more direct learning signal by precisely targeting the optimal action prescribed by the current policy and value function. Our DOAL framework presents a fourth, fundamentally distinct approach by completely decoupling the optimal action computation from policy learning.

7 Conclusion

In this work, we present Direct Optimal Action Learning, a framework that enables efficient and effective learning from $\nabla_a Q(s, a)$ for any policy distribution. We provide strong baselines by re-examining the importance of n^{sample} in MaxQ sampling, then we are able to show our models improved over baseline in most cases. Our experiment set up is mostly aiming at controlled study, so we rely on IQL. In the future, uncertainty aware Q estimation should be important to explore, as it might further improve the statistical trust region identification.

REFERENCES

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Offline RL without off-policy evaluation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Huayu Chen, Kaiwen Zheng, Hang Su, and Jun Zhu. Aligning diffusion behaviors with q-functions for efficient continuous control. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

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

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 *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

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 (ICLR)*, 2025.

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

Scott Fujimoto and Shixiang Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Chen-Xiao Gao, Chenyang Wu, Mingjun Cao, Chenjun Xiao, Yang Yu, and Zongzhang Zhang. Behavior-regularized diffusion policy optimization for offline reinforcement learning. In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

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 (ICML)*, 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*, abs/2304.10573, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems (NeurIPS)*, 2020.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022.

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

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

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems (NeurIPS)*, 2020.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. *Batch Reinforcement Learning*, pp. 45–73. Springer Berlin Heidelberg, 2012.

Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

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

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems (NeurIPS)*, 2022.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline RL? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025a.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable, 2025b.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning (ICML)*, 2025c.

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

David E. Rumelhart, Richard Durbin, Richard M. Golden, and Yves Chauvin. Backpropagation: the basic theory. 1995. URL `https://api.semanticscholar.org/CorpusID:60753175`.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021b.

Qiao Sun, Zhicheng Jiang, Hanhong Zhao, and Kaiming He. Is noise conditioning necessary for denoising generative models? In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Yifan Wu, G. Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *ArXiv*, abs/1911.11361, 2019.

A Appendix

A.1 The Use of Large Language Models

Large language models are used for polishing the texts, including equations and literature reviews.

A.2 Proof of equation 11

Proof. The proof follows from applying the chain rule. First, we compute the gradient of the regularized Q-objective in equation 11:

$$\begin{aligned}\nabla_\theta J_Q(\theta) &= \nabla_\theta Q(s, \pi_\theta(s)) - \nabla_\theta \left(\alpha \|\pi_\theta(s) - a\|_2^2\right) \\ &= \left(\nabla_a Q(s, a')\big|_{a'=\pi_\theta(s)}\right)^\top \nabla_\theta \pi_\theta(s) - 2\alpha(\pi_\theta(s) - a)^\top \nabla_\theta \pi_\theta(s) \\ &= \left[\nabla_a Q(s, a')\big|_{a'=\pi_\theta(s)} - 2\alpha(\pi_\theta(s) - a)\right]^\top \nabla_\theta \pi_\theta(s)\end{aligned}$$

Next, let the target action be $a^* \triangleq a + \frac{1}{2\alpha}\nabla_a Q(s, a')\big|_{a'=\pi_\theta(s)}$. We compute the gradient of the target-matching objective $J_{\text{target}}(\theta) = -\alpha\|\pi_\theta(s) - a^*\|_2^2$:

$$\begin{aligned}\nabla_\theta J_{\text{target}}(\theta) &= -\nabla_\theta \left(\alpha \|\pi_\theta(s) - a^*\|_2^2\right) \\ &= -2\alpha(\pi_\theta(s) - a^*)^\top \nabla_\theta \pi_\theta(s) \\ &= -2\alpha \left(\pi_\theta(s) - \left(a + \frac{1}{2\alpha}\nabla_a Q(s, a')\big|_{a'=\pi_\theta(s)}\right)\right)^\top \nabla_\theta \pi_\theta(s) \\ &= \left[-2\alpha(\pi_\theta(s) - a) + \nabla_a Q(s, a')\big|_{a'=\pi_\theta(s)}\right]^\top \nabla_\theta \pi_\theta(s)\end{aligned}$$

The resulting gradients are identical. □

A.3 DOAL Objectives

A.3.1 Directed Implicit Q-Learning (DIQL)

DIQL extends the implicit Q-learning framework by introducing a direct optimal action learning approach that explicitly guides policy optimization through value-aware action adjustments. The key innovation lies in replacing the dataset action a with the optimized target action a^* in the policy loss, thereby directly steering the policy towards high-value regions.

$$\mathcal{L}_\pi^{\text{DIQL}}(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp\left(\alpha_{\text{actor}}(Q_\phi(s, a) - V_\psi(s))\right) \log \pi_\theta(a^{\text{target}}|s)\right], \tag{17}$$

Notice the weighting is using the original $Q(s, a)$ as we want to reduce computational overhead.

A.3.2 Direct Implicit Flow-Q-Learning (DIFQL)

Implicit Flow-Q-Learning (DIFQL) (Park et al., 2025c) builds upon the framework established by Implicit Diffusion Q-learning (Hansen-Estruch et al., 2023). While the updates for the Q-function

and value function remain consistent with IQL (see Equation 1), IFQL distinguishes itself by employing a flow-matching objective for policy optimization, as defined by the behavior cloning loss in Equation 3.

The policy's behavior cloning loss in DIFQL is formulated as:

$$\mathcal{L}_\pi^{\text{DIFQL}}(\theta) = \alpha \cdot \mathbb{E}_{a_1 \sim \mathcal{N}(0,I), t \sim \mathcal{U}[0,1]} \left[\| v_\theta(a_t, t) - (a^{\text{target}} - a_1) \|^2 \right], \quad a_t = (1-t)a^{\text{target}} + t a_1 \tag{18}$$

Actions are subsequently generated by sampling from the learned flow model (see Equation 4) and the final action selection is determined by maximizing the learned Q-value function (see Equation 9).

A.3.3 Directed Implicit TrigFlow-Q-Learning (DTrigflow)

Following Equation 6 and adopt DOAL, we have

$$\mathcal{L}^{\text{DTrigflow}}(\theta) = \mathbb{E}_{a_0 \sim p(a_0), z \sim \mathcal{N}(0,I), t \sim \mathcal{U}[0, \pi/2]} \left[\| f_\theta(a_t, t) - a_0^{\text{target}} \|_2^2 \right], \quad a_t = \cos(t) a^{\text{target}} + \sin(t) z \tag{19}$$

A.3.4 TrigFlow-Q-Learning (TrigflowQL)

For TrigflowQL, we have the following:

$$\mathcal{L}_\pi^{\text{TrigflowQL}}(\theta) = \mathbb{E}_{(s, a_0, t, z) \sim \mathcal{D}} \left[-Q_\phi(s, f_\theta(a_t, t)) + \alpha \cdot \left[\| f_\theta(a_t, t) - a_0 \|_2^2 \right] \right] \quad a_t = \cos(t) a_0 + \sin(t) z \tag{20}$$