IDQL: Implicit Q-Learning as an Actor-Critic Method with Diffusion Policies

Philippe Hansen-Estruch
Ilya Kostrikov
Michael Janner
Kuba Grudzien
Sergey Levine

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-62
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-62.html

May 2, 2023

Masters Report

IDQL: Implicit Q-Learning as an Actor-Critic Method with Diffusion Policies

by Philippe Hansen-Estruch

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Sergey Levine
Research Advisor

5/1/23

(Date)

* * * * * * *

Masters Report

Professor Pieter Abbeel
Second Reader

20-April-2023

(Date)

IDQL: Implicit Q-Learning as an Actor-Critic Method with Diffusion Policies

Philippe Hansen-Estruch Ilya Kostrikov Michael Janner
Jakub Grudzien Kuba Sergey Levine
UC Berkeley
{hansenpmeche, kostrikov, janner, kuba}@berkeley.edu
svlevine@eecs.berkeley.edu

Abstract

Effective offline RL methods require properly handling out-of-distribution actions. Implicit Q-learning (IQL) addresses this by training a Q-function using only dataset actions through a modified Bellman backup. However, it is unclear which policy actually attains the values represented by this implicitly trained Q-function. In this paper, we reinterpret IQL as an actor-critic method by generalizing the critic objective and connecting it to a behavior-regularized implicit actor. This generalization shows how the induced actor balances reward maximization and divergence from the behavior policy, with the specific loss choice determining the nature of this tradeoff. Notably, this actor can exhibit complex and multimodal characteristics, suggesting issues with the conditional Gaussian actor fit with advantage weighted regression (AWR) used in prior methods. Instead, we propose using samples from a diffusion parameterized behavior policy and weights computed from the critic to then importance sampled our intended policy. We introduce Implicit Diffusion Q-learning (IDQL), combining our general IQL critic with the policy extraction method. IDQL maintains the ease of implementation of IQL while outperforming prior offline RL methods and demonstrating robustness to hyperparameters. Code is available at github.com/philippe-eecs/IDQL.

1 Introduction

Offline RL holds promise in enabling policies to be learned from static datasets. Value function estimation provides a basic building block for many modern offline RL methods, but such methods need to handle the out-of-distribution actions that arise when evaluating the learned policy, since this policy will deviate from the behavior policy during training. While a variety of methods based on constraints and regularization have been proposed to address this, an intriguing approach is to avoid out-of-sample actions entirely, and instead utilize a loss function that *implicitly* performs maximization over the actions, as in the implicit Q-learning (IQL) method proposed by Kostrikov et al. [26]. This approach avoids the need to query the value for unseen actions, instead, the Q-function is trained with Bellman backups using the state value function as the target value. The state value function is trained via expectile regression onto the Q-values implicitly performing a maximization over actions. While IQL provides a simple and appealing alternative to other methods in offline RL, it remains unclear what policy the learned value function is actually evaluating, and in turn, makes it difficult to understand the bottlenecks in the performance of IQL-style algorithms.

In this paper, we derive a variant of IQL that significantly improves over the performance of the original approach, is relatively insensitive to hyperparameters, and outperforms even more recent offline RL methods. Our key observation is based on a new perspective that reinterprets IQL as an actor-critic method. This is achieved through the generalization of the value optimization problem in IQL to use an arbitrary convex loss, which we then link to an implicit behavior-regularized actor.

Preprint. Under review.

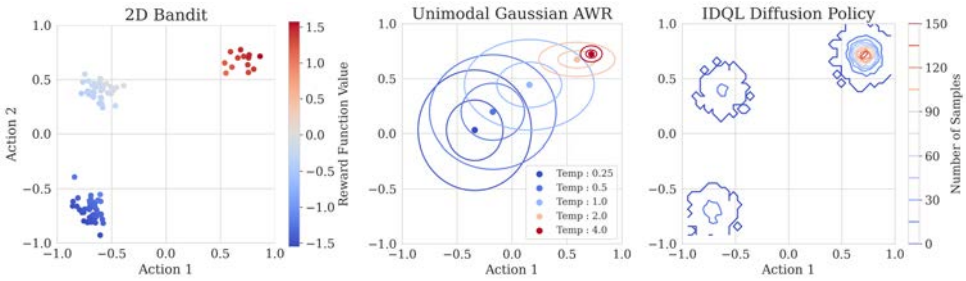


Figure 1: **Comparing unimodal AWR as used by IQL to our IDQL diffusion policy.** (Left) 2D bandit with three actions and reward function $r(a_1, a_2) = a_1 + a_2$. For the 90th expectile of the reward distribution, we compute contour plots for the corresponding unimodal Gaussian policy fitted with AWR (as used by IQL) for various inverse temperatures (center) as well as samples from our proposed diffusion policy extraction method (right). The unimodal actor incorrectly approximates the implicit policy for all temperatures and requires tuning to capture the maximum action, while the diffusion extraction method captures the implicit policy accurately.

This generalization is significant, as it demonstrates how different choices of critic objectives give rise to distinct implicit actor distributions that diverge from the behavior policy to varying extents. Consequently, examining the characteristics of this implicit actor reveals not only the trade-offs made by IQL in critic learning, but also potential avenues for improving the method.

One of the challenges revealed by our analysis is that the implicit actor is complex and potentially multimodal. This suggests an issue with the policy extraction scheme used with IQL [26], in which a unimodal Gaussian actor is trained with advantage weighted regression (AWR) [38, 37, 33]. Such an actor is unlikely to accurately approximate the implicit actor that the Q-function represents (Figure 1). Unlike conventional actor-critic algorithms, where the critic adapts to the *explicit* actor parameterization, in IQL the critic training is fully decoupled from the actor. As a result, any approximation errors in the policy extraction stage are never compensated for by the critic. This decoupling is a major strength of IQL, as it avoids the instability and hyperparameter sensitivity normally associated with coupling between the actor and critic, but it requires policy extraction to approximate the implicit actor accurately. We, therefore, propose a more expressive policy extraction method for IQL that can address this challenge. One helpful observation is the implicit actor induced through critic learning can be expressed as a reweighting of the behavior policy, which can be approximated via importance weights. Thus, for policy extraction, we propose reweighting samples from a diffusion-parameterized behavior model with critic-computed weights, which can then be re-sampled to express both the implicit and greedy policy. Diffusion models [41, 20, 43] are a crucial detail for enabling our policy extraction method to represent multimodal distributions accurately.

We introduce Implicit Diffusion Q-learning (IDQL). Our main contributions are (1) the generalization of IQL to an actor-critic method where the choice of critic loss induces an implicit actor distribution that can be expressed as a reweighting of the behavior policy; (2) a diffusion based policy extraction algorithm that combines samples from an expressive diffusion model with a reweighting scheme for recovering an accurate approximation to the implicit actor, or maximizing the critic value. IDQL outperforms prior methods on the D4RL offline RL benchmarks [11], including methods based on IQL and other methods that use diffusion models. Furthermore, IDQL is very portable, requiring limited tuning of hyperparameters across domains (e.g., antmaze, locomotion) to perform well for both offline RL and online finetuning with offline pretraining.

2 Related Work

While offline RL has been approached with a wide range of algorithmic frameworks [30, 31], recent methods often use value-based algorithms derived from Q-learning and off-policy actor-critic methods. To adapt approaches to the offline setting, it is necessary to avoid overestimated values out-of-distribution actions [27]. To this end, recent methods use implicit divergence constraints [38, 37, 33, 45], explicit density models [47, 13, 27, 15], and supervised learning terms [12]. Some works also directly penalize out-of-distribution action Q-values [25, 28].

Recently, *implicit* TD backups have been developed that avoid the use of out-of-sample actions in Q-function by using asymmetric loss functions with SARSA-style on-policy backups. This approach was proposed in IQL [26], which trains a Q-function with an expectile loss and then extracts a policy with AWR [37]. However, the extraction procedure itself imposes a KL-divergence constraint, which is not consistent with the policy the critic captures. Extreme Q-learning (EQL) [14] modifies the Q-function objective in IQL to estimate a soft value consistent with the entropy regularized AWR policy. Concurrently, Xu et al. [49] provides another generalization of IQL in behavior-regularized MDPs, arriving at two new implicit Q-learning objectives. Our generalized IQL derivation is related, but features a different generalization form. Contrary to these papers, which focus on Q-learning, our primary practical contributions are in the policy extraction step motivated by this generalization, which we show leads to gains in performance and simplify hyperparameter tuning.

Our method leverages expressive generative models to capture the policy. To that end, we review works that use expressive generative models for offline RL. EMaQ [15] defines a policy using an auto-regressive behavioral cloning model, using the argmax action from this policy for backups in critic learning. The Decision Transformer and Trajectory Transformer [8, 21] clone the entire trajectory space using transformers and apply reward conditioning or beam search, respectively, to bias sampled trajectories towards higher rewards. Diffusion models have also been used in behavioral cloning and offline RL. Florence et al. [10] and Pearce et al. [36] use energy-based models and diffusion models, respectively, for behavioral cloning. Janner et al. [22] and Ajay et al. [1] use diffusion to directly model and sample the trajectory space; samples are guided with gradient guidance or reward conditioning. Reuss et al. [40] uses diffusion policies for goal-conditioned imitation learning.

Closest to our work are prior methods that represent the actor with a diffusion model in offline RL. Diffusion Q-learning (DQL) [44] incorporates diffusion to parameterize the actor in a TD3+BC-style algorithm [12]. Select from Behavior Candidates (SfBC) [7] uses importance reweighting from a diffusion behavior model to define the policy and then trains a critic with value iteration using samples from the policy. Action-Restricted Q-learning (ARQ) [16] performs a batch constrained critic update with a diffusion behavior model and then uses AWR to extract the policy. Our method differs from these methods as the diffusion behavior model is kept separate from the critic learning process entirely, only interacting with the critic at evaluation time. This makes it comparatively computationally efficient and simple to tune hyperparameters, as we illustrate in our "1-hyperparameter" experiment, which shows significant gains over prior methods (including the best performing method, DQL). Furthermore, to our knowledge, we are the first method to apply diffusion to online finetuning.

3 Preliminaries

RL is formulated in the context of a Markov decision process (MDP), which is defined as a tuple $(\mathcal{S}, \mathcal{A}, p_0(s), p_M(s'|s,a), r(s,a), \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, initial state distribution $p_0(s)$, transition dynamics $p_M(s'|s,a)$, reward function $r(s,a)$, and discount γ. The goal is to recover a policy $\pi(a|s)$ that maximizes the discounted sum of rewards or return in the MDP. In offline reinforcement learning, the algorithm is given access to a fixed dataset of transitions $\mathcal{D} = \{s, a, r, s'\}$ from which it has to train a policy.

Implicit Q-learning. Instead of constraining the policy or regularizing the critic, Kostrikov et al. [26] proposed to approximate the expectile τ over the distribution of actions. For a parameterized critic $Q_\theta(s,a)$, target critic $Q_{\hat{\theta}}(s,a)$, and value network $V_\psi(s)$ the value objective is

$$\begin{aligned} \mathcal{L}_V(\psi) &= \mathbb{E}_{(s,a)\sim\mathcal{D}}[L_2^\tau(Q_{\hat{\theta}}(s,a) - V_\psi(s))] \\ \text{where } L_2^\tau(u) &= |\tau - \mathbb{1}(u<0)|u^2. \end{aligned} \tag{1}$$

Expectile regression uses a simple asymmetric squared error and requires only sampling actions from the datasets without any explicit policy. Then, this value function is used to update the Q-function:

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a))^2]. \tag{2}$$

The Q-function is induced under the implicit policy distribution defined by the expectile. For policy extraction, IQL uses AWR [38, 37, 33], which trains the policy via weighted regression by minimizing

$$\mathcal{L}_\pi(\phi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[\exp(\alpha(Q_{\hat{\theta}}(s,a) - V_\psi(s)))\log\pi_\phi(a|s)]. \tag{3}$$

The temperature parameter, $\alpha \in [0, \infty]$, serves to balance critic exploitation with behavior cloning.

Diffusion models. We briefly review diffusion for behavior cloning as it is the basis of our policy extraction algorithm. Diffusion models [41, 20, 43] are latent variable models that use a Markovian noising and denoising process that can be used to model a parameterized behavior distribution $\mu_\phi(a_0|s) = \int \mu_\phi(a_{0:T}|s)\, da_{1:T}$ for the latent variables $a_1, \cdots, a_T$. The forward noising process follows a fixed variance schedule $\beta_1, \cdots, \beta_T$ that follows the distribution

$$q(a_t|a_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}a_{t-1}, \beta_t I).$$

Following DDPMs [20], our practical implementation involves parameterizing the score network directly $\mu_\phi(a_{t-1}|a_t, s, t)$ to recover the behavior cloning objective

$$\mathcal{L}_\mu(\phi) = \mathbb{E}_{t\sim\mathcal{U}(1,T),\epsilon\sim\mathcal{N}(0,I),s,a\sim\mathcal{D}}[||\epsilon - \mu_\phi(\sqrt{\hat{\alpha}_t}a + \sqrt{1-\hat{\alpha}_t}\epsilon, s, t)||]. \tag{4}$$

To sample from $\mu_\phi(a_0|s)$, we use Langevin sampling or reverse diffusion where $a_T \sim \mathcal{N}(0, I)$ and $\epsilon \sim \mathcal{N}(0, I)$ gets resampled every step

$$a_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(a_t - \frac{\beta_t}{\sqrt{1-\hat{\alpha}_t}}\mu_\phi(a_t|s,t)\right) + \sqrt{\beta_t}\epsilon, \text{ for } t = \{T, \cdots, 1\} \tag{5}$$

4 Implicit Q-Learning as an Actor-Critic Method

In this section, we will present a generalization of IQL that will not only provide a more complete conceptual understanding of implicit Q-learning, but also help us to understand how this method could be improved. Kostrikov et al. [26] show that IQL recovers Q-learning in the limit as τ approaches 1 in Equation 1, but this does not describe what policy is captured by the Q-function in practice, when $0.5 < \tau < 1$. We can better understand the real behavior of IQL by reinterpreting it as an actor-critic method, where critic learning induces an implicit behavioral regularized actor $\pi_{\text{imp}}(a|s)$. This generalization will help us to understand how IQL can be improved, and the tradeoff that is captured by IQL's hyperparameters and the form of its loss function.

4.1 Generalized Implicit Q-Learning

To rederive IQL as an actor-critic method, we first generalize the value loss in Equation 1 to use an arbitrary convex loss f on the difference $Q(s,a) - V(s)$. For a given $Q(s,a)$, the general IQL critic update can be defined as

$$V^*(s) = \arg\min_{V(s)} \mathbb{E}_{a\sim\mu(a|s)}[f(Q(s,a) - V(s))] = \arg\min_{V(s)} \mathcal{L}_V^f(V(s)). \tag{6}$$

We recover IQL when $f(u) = L_2^\tau(u)$, as in Equation 1, but we will also consider other asymmetric convex losses below. To define the implicit actor, we follow the conventional definition of value functions in actor-critic methods, where

$$V(s) = \mathbb{E}_{a\sim\pi_{\text{imp}}(a|s)}[Q(s,a)]. \tag{7}$$

We use $f' = \frac{\partial f}{\partial V(s)}$ as shorthand for the derivative of f with respect to $V(s)$.

Theorem 4.1. *For every state s and convex loss function f where $f'(0) = 0$, the solution to the optimization problem defined in Equation 6 is also a solution to the optimization problem*

$$\arg\min_{V(s)} \mathbb{E}_{a\sim\pi_{imp}(a|s)}[(Q(s,a) - V(s))^2],$$

where $\pi_{imp}(a|s) \propto \frac{\mu(a|s)|f'(Q(s,a)-V^*(s))|}{|Q(s,a)-V^*(s)|}$.

Proof. See Appendix B. □

Theorem 4.1 provides us with a relationship between the (generalized) IQL loss function f and the corresponding implicit actor $\pi_{\text{imp}}(a|s)$, thus indicating that IQL is an actor-critic method. To make it more apparent how the implicit actor relates to the behavior policy $\mu(a|s)$, we can define an importance weight

$$w(s,a) = \frac{|f'(Q(s,a) - V^*(s))|}{|Q(s,a) - V^*(s)|}, \tag{8}$$

which yields an expression for the implicit actor as $\pi_{\text{imp}}(a|s) \propto \mu(a|s)w(s,a)$. The form of f affects how $\pi_{\text{imp}}(a|s)$ deviates from $\mu(a|s)$. Since IQL can recover the value function $V^*(s)$ without constructing the policy explicitly, the implicit actor only needs to be recovered at the end in order to select the actions at evaluation. This should be a major strength of the method since decoupling the critic from the actor removes the complex interactions that might contribute to instability and hyperparameter sensitivity. However, the complex form of the weight in Equation 8 also suggests a challenge, since policy extraction with a non-expressive policy class (e.g., conditional Gaussian policy) is likely to lead to a poor approximation of this implicit actor. We will return to this point in the next section when we discuss our proposed policy extraction method, but first we will provide three examples of potential functions f and derive their corresponding implicit policies.

Expectiles. The expectile of a distribution corresponds to the conditional mean if points above the expectile are sampled more frequently than in the standard distribution. For a given τ, the expectile objective corresponds to $f(u) = L_2^\tau(u)$ (from Equation 1) and solution $V_\tau^2(s)$. From Theorem 4.1,

$$w_2^\tau(s,a) = |\tau - \mathbb{1}(Q(s,a) < V_\tau^2(s))|. \tag{9}$$

Increasing τ for expectiles directly increases the deviation from the behavior policy.

Quantiles. The quantile statistic measures the top $\tau\%$ of the behavior policy performance. This is analogous to the mean behavior performance (SARSA) and Q-learning trade-off that occurs with expectiles, but instead balances median behavior performance with Q-learning. For a given τ, the quantile objective corresponds to $f(u) = |\tau - \mathbb{1}(u < 0)||u|$ and solution $V_\tau^1(s)$. From Theorem 4.1,

$$w_1^\tau(s,a) = \frac{|\tau - \mathbb{1}(Q(s,a) < V_1^\tau(s))|}{|Q(s,a) - V_\tau^1(s)|}. \tag{10}$$

As with expectiles, increasing τ directly influences the level of extrapolation from the behavior policy. Points should cluster around the quantile for this implicit policy.

Exponential. Another interesting choice for f is the linex function $f(u) = \exp(u) - u$, or

$$V_{\exp}(s) = \arg\min_{V(s)} \mathbb{E}_{a\sim\mu(a|s)}[\exp(\alpha(Q(s,a) - V(s)) - \alpha(Q(s,a) - V(s))], \tag{11}$$

where the temperature $\alpha \in [0, \infty]$ serves to balance matching the behavior policy and optimizing the critic. The solution for this objective is

$$V_{\exp}(s) = \frac{1}{\alpha} \log \sum_a \exp(\alpha Q(s,a) + \log \mu(a|s)). \tag{12}$$

We derive the solution to this objective in Appendix C. The derivation of $V_{\exp}(s)$ indicates that it is a normalizer for the AWR policy (Equation 3), where $\pi_{\text{awr}}(a|s) \propto \exp(\alpha Q(s,a) + \log \mu(a|s))$. Effectively, this loss applies a KL-divergence constraint. This objective matches the critic objectives used by Garg et al. [14] and Xu et al. [49], and it is also similar to the soft policy definition used by Haarnoja et al. [17]. From Theorem 4.1,

$$w_{\exp}(s,a) = \frac{\alpha|\exp\left(\alpha(Q(s,a) - V_{\exp}(s))\right) - 1|}{|Q(s,a) - V_{\exp}(s)|}. \tag{13}$$

This weighting gives most of the weight to the actions with the highest Q-value. If IQL is used with the standard AWR policy extraction proposed by Kostrikov et al. [26], then this choice of f would provide the correct corresponding critic, though we will see later that in practice this loss does not tend to lead to the best performance.

Comparing different loss functions. Theorem 4.1 demonstrates that IQL can be generalized to an actor-critic method with the choice of loss function f influencing the implicit policy. In each case, there is a hyperparameter that controls how much the implicit actor deviates from the behavior policy. In a simple bandit problem (Figure 2), the expectile increases smoothly with τ, the quantile corresponds to the cumulative distribution function, and the exponential aligns with the maximum. The implicit policy distributions also differ for each objective: the expectile actor covers a broad range of outcomes, the quantile actor is tightly focused around the mean, and the exponential actor has some coverage around the maximum. Although it seems that the exponential critic learns the most optimal policy, in practice we find that it can be unstable, as we will show in Section 6.3.

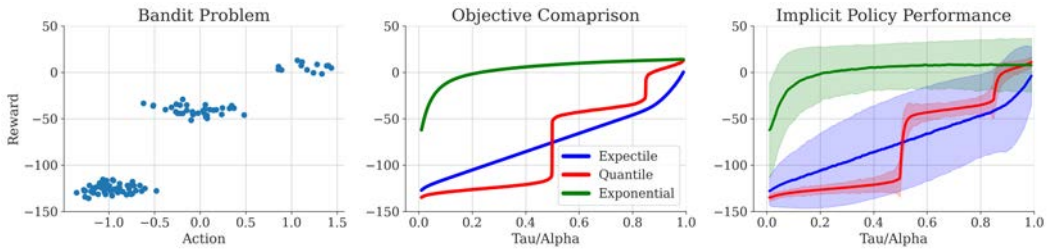


Figure 2: **Comparison of different loss functions for generalized IQL.** We compare the different losses on a simple didactic bandit task. The bandit's continuous action space has three clusters corresponding to low, medium, and high reward, with additive noise (left). We compute values from the different objectives in Section 4 over choices of hyperparameter on the bandit distribution (center). Finally, we show the mean and standard deviation performance of samples from the implicit actor (from Theorem 4.1) induced by each objective (right). Each objective captures different characteristics from the distribution of rewards as its hyperparameter is increased.

4.2 Policy Extraction and General Algorithm

Algorithm 1 General IQL Training

Hyperparameters: LR λ, EMA η
Initialize: θ, $\hat{\theta}$, ψ, and ϕ

while training not converged **do**
  $\psi \leftarrow \psi - \lambda \nabla_\psi \mathcal{L}_V^f(\psi)$ (Equation 6)
  $\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}_Q(\theta)$ (Equation 2)
  $\hat{\theta} \leftarrow (1-\eta)\hat{\theta} + \eta\theta$
  $\phi \leftarrow \phi - \lambda \nabla_\phi \mathcal{L}_\mu(\phi)$ (Equation 4)
end while

Algorithm 2 General IQL Policy Extraction

Hyperparameters: Samples per state N
Pretraining: $Q_{\hat{\theta}}(s,a)$, $V_\psi(s)$, and $\mu_\phi(a|s)$

while not done with episode **do**
  Observe current state s
  Sample $a_i \sim \mu_\phi(a|s)$, $i = 1, \ldots, N$
  Compute $w(s, a_i)$ using Eqns. 9, 10, 13, or 14
  Normalize: $p_i = \frac{w(s,a_i)}{\sum_j w(s,a_j)}$
  Select a_{taken} as a categorical from p_i
end while

Section 4 shows that the implicit actors corresponding to critics trained with IQL can be complex and multimodal. However, standard IQL approximates these complex implicit actors with a standard unimodal conditional Gaussian policy [26], and since the critic is unaware of this approximation, it does not adapt to it (in contrast to a standard actor-critic method where the critic adjusts to the limitations of the actor). While decoupling the actor from the critic should make the method less sensitive to hyperparameters, we hypothesize that in practice this benefit can only be realized if the final explicit actor is powerful enough to capture the complex implicit actor distribution.

One approach to train a more expressive actor would be to use a more powerful conditional density model, such as a diffusion model or normalizing flow, and combine it with the same AWR-style importance weighted objective as in standard IQL. However, it is known in the literature that using highly expressive models with importance weighted objectives can be problematic, as such models can increase the likelihood of all training points regardless of their weight [6, 48]. We find using AWR in the DDPM objective to not help performance (Appendix F). In order to capture these policies without requiring extensive hyperparameter tuning, we instead take a different approach: we train a highly expressive policy to represent the behavior policy, without any importance weighting, and then reweight the samples from this behavior policy model. This is similar to the policy extraction proposed by Chen et al. [7].

Denoting our learned behavior policy model as $\mu_\phi(a|s)$, we can generate samples from this model, and then use the critic to reweight these actions, ultimately forming the intended policy when resampled. This approach is summarized in Algorithm 2, and provides samples from the correct implicit actor distribution. In practice, we also found that simply taking the action with the highest Q-value tends to yield better performance at evaluation time. This approach is analogous to how stochastic actor methods typically use a stochastic actor for critic learning and a deterministic actor at

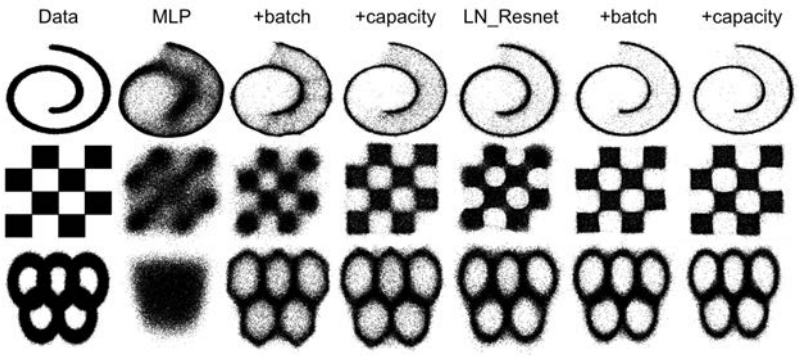


Figure 3: Samples from configurations of DDPMs for toy 2D continuous datasets. MLP uses a batch size of 256 and two hidden layers size 256. +Batch increases the batch size to 4096 and + capacity uses a batch size of 4096 and 3 hidden layers. LN_Resnet follows the same pattern, but uses 2 hidden blocks originally and then 3 hidden blocks for + capacity. Code is provided at https://github.com/philippe-eecs/JaxDDPM

.

evaluation time, [5, 17]. This yields the deterministic policy

$$\pi(s) = \underset{a_i \sim \mu(a|s),\ i=1\ldots N}{\arg\max} Q(s, a_i). \tag{14}$$

This corresponds to selecting $w(s, a)$ to be a one-hot over the sampled actions.

General algorithm summary. For choice of loss f and generative model parameterization, our complete algorithm is summarized in the training phase (Algorithm 1) and inference phase (Algorithm 2). Theoretically, any parameterization of generative model could be used for behavior cloning in our method. However, we find that diffusion models yield the best results for our practical approach, as we discuss in the subsequent section.

Online finetuning procedure. Since general IQL extracts policies only during evaluation, finetuning the behavioral distribution may not be strictly necessary for situations where exploration is not a significant requirement. We outline two possible methods for fine-tuning using general IQL: (1) freezing the behavior policy $\mu_\phi(a|s)$, sampling the argmax action for exploration, and only finetuning $Q_\theta(s, a)$ and $V_\psi(s)$ and, (2) sampling $\pi_{\text{imp}}(a|s)$ for exploration, and finetuning $Q_\theta(s, a)$, $V_\psi(s)$, and $\mu_\phi(a|s)$.

5 Implicit Diffusion Q-Learning

As argued in Section 4.2, the policy extraction algorithm requires an expressive behavior distribution to model the implicit actor accurately. Diffusion models are a good fit here as they have been used to model complex distributions in images [20] as well as in continuous action spaces [44, 36] prior. Therefore, for the practical implementation of our general IQL algorithm depicted in Algorithm 1 and Algorithm 2, we use a diffusion model parameterization for $\mu_\phi(a|s)$ and the DDPM objective (Equation 4). We dub this method Implicit Diffusion Q-learning (IDQL).

Issues in continuous space expression with diffusion. A naïve DDPM implementation on continuous spaces can have issues with outputting outliers and expressing the distribution accurately. As an example, we test a simple implementation of DDPMs on 2D continuous datasets. In Figure 3, the simple MLP architecture fails to capture the data distributions and produces many out-of-distribution samples. The outliers are particularly problematic because they might be out-of-distribution for a trained Q-function and as a result, might receive erroneously high Q values.

As demonstrated with diffusion on images [35], we find that increasing the batch size and capacity of the MLP network fits the distribution better, but many outliers remain. Inspired by architectures

that work well for transformers [39], we find that a more ideal network for diffusion should have high capacity while being well-regularized. We use a residual network [18] with layer normalization [2] (LN_resnet) as our score network parameterization (depicted in Appendix G). Figure 3 shows our architecture choice producing higher quality samples with fewer outliers compared to a standard MLP architecture. We demonstrate in Section 6.4 that this architecture choice is crucial for strong performance and reduced sensitivity to N. This section provides contributions on how batch size, capacity, and architecture choices affect the modeling of continuous spaces with diffusion. Though, similar architectures for action modeling with diffusion have been proposed before [7, 36, 40].

6 Experimental Evaluation

In order to assess the performance, scalability, and robustness of our approach, we compare it to prior works using two protocols: (1) a direct comparison on D4RL benchmarks [11] with reported numbers, and (2) a "one hyperparameter evaluation" where we rerun top-performing methods with one hyperparameter setting tuned per domain. We also analyze our method's performance in online finetuning and on offline performance over the critic objectives mentioned in Section 4. We include additional ablations and comparisons in Appendix F.

6.1 Offline RL Results

A major appeal of offline RL methods is their ability to produce effective policies without any online interaction. The less tuning that is required for a given method, the easier it will be to use in the real world. Therefore, in this section, we evaluate select methods both with any number of hyperparameter allowed (or reported results) and in a regime where only one hyperparameter can be tuned per domain. We select Conservative Q-learning (CQL) [28], Implicit Q-learning (IQL) [26], and Diffusion Q-learning (DQL) [44] to focus on because of their strong performance in the standard offline RL setting. We refer "-A" as reported results that allow any amount of tuning and "-1" as results that only allow one hyperparameter to be tuned between domains (for IDQL, $\tau = 0.7$ for all locomotion tasks and $\tau = 0.9$ for all antmaze tasks). Results are in Table 1. We also include comparisons to %BC, Decision Transformers (DT) [8], TD3+BC (TD3) [12], extreme Q-learning (EQL) [14], and Selecting from Behavior Candidates (SfBC) [7] in Table 2 (full table in Appendix F).

In the standard evaluation protocol, IDQL performs competitively to the best prior methods on the locomotion tasks while outperforming prior methods on the antmaze tasks. In the one hyperparameter regime, the performance of IDQL degrades only slightly from the results in Table 1, while the other prior methods suffer considerably more, particularly on the more challenging antmaze domain. Thus, with limited tuning, IDQL outperforms the prior methods by a very significant margin. This is specifically apparent in the antmaze domain, where our one hyperparameter results outperfrom the best method (IQL) by +70 points. For fairness, we train all "-1" methods for 2M gradient steps, which is why IDQL's training time is reduced from its "-A" variant which is trained for 3M gradient steps.

Furthermore, we compare training time between the different methods. IDQL only has a small increase in training time over IQL, while CQL, DQL, and SfBC are slower overall. In particular, IDQL is much faster than the other two diffusion methods Chen et al. [7] and Wang et al. [44]. Although IDQL uses a much more expressive policy model, the computationally expensive critic training is completely separated from this model, leading to IDQL's computational efficiency.

6.2 Online Finetuning

After offline training, policies can be improved with online interactions. We test the procedure of freezing the behavior policy and finetuning the value networks only, as well as finetuning all networks, as described in Section 4.2. We compare to Cal-QL [34], RLPD [3], and IQL [26]. Results are presented in Table 3. We see large improvement in both pre-training and final fine-tuning performance compared to IQL. IDQL also remains competitive with RLPD and Cal-QL in finetuning, while having stronger pre-training results. Most of the gains come from improvements in the hardest antmaze-large environments. To compare the sample efficiency and effectiveness of our policy extraction method, we compare IDQL to IQL in antmaze-large environments in Figure 4. The training curves show a performance gain and sample efficiency gain over IQL, indicating the strength of IDQL in finetuning.

Dataset	CQL-A	IQL-A	DQL-A	IDQL-A	CQL-1	IQL-1	DQL-1	IDQL-1
halfcheetah-med	44.0	47.4	**51.1**	**51.0**	46.4	47.6	**50.6**	**49.7**
hopper-med	58.5	66.3	**90.5**	65.4	64.4	63.7	**75.2**	63.1
walker2d-med	72.5	78.3	**87.0**	**82.5**	**81.6**	**81.9**	**83.4**	**80.2**
halfcheetah-med-rep	**45.5**	44.2	**47.8**	**45.9**	**45.4**	43.1	**45.8**	**45.1**
hopper-med-rep	**95.0**	**94.7**	**101.3**	**92.1**	88.5	42.5	**94.5**	82.4
walker2d-med-rep	77.2	73.9	**95.5**	85.1	74.5	**78.4**	**86.7**	**79.8**
halfcheetah-med-exp	**91.6**	86.7	**96.8**	**95.9**	64.6	88.1	**93.3**	**94.4**
hopper-med-exp	**105.4**	91.5	**111.1**	**108.6**	99.3	73.7	**102.1**	**105.3**
walker2d-med-exp	**108.8**	**109.6**	**110.1**	**112.7**	**109.6**	**110.5**	**109.6**	**111.6**
locomotion-v2 total	698.5	692.4	**791.2**	**739.2**	674.3	629.5	**741.2**	**711.6**
antmaze-umaze	74.0	87.5	**93.4**	**94.0**	65.0	86.4	47.6	**93.8**
antmaze-umaze-div	**84.0**	62.2	66.2	**80.2**	41.3	**62.4**	35.8	**62.0**
antmaze-med-play	61.2	71.2	76.6	**84.5**	31.4	76.0	42.5	**86.6**
antmaze-med-div	53.7	70.0	78.6	**84.8**	25.8	74.8	46.3	**83.5**
antmaze-large-play	15.8	39.6	46.4	**63.5**	8.5	31.6	19.0	**57.0**
antmaze-large-div	14.9	47.5	57.3	**67.9**	7.0	36.4	25.2	**56.4**
antmaze-v0 total	303.6	378.0	418.5	**474.6**	180.0	368.4	216.4	**439.3**
total	1002.1	1070.4	**1209.7**	**1213.8**	854.3	997.9	957.4	**1150.9**
training time	80m	20m	240m	60m	80m	20m	240m	40m

Table 1: **Focused offline RL comparison.** IDQL performs on par or better than other SOTA offline RL methods. "-A" refers to any number of hyperparameters allowed and "-1" allows only one hyperparameter. Results for our method are averaged over 10 seeds. Additional details and comparisons with other offline RL methods are in Appendix D and Appendix F repectively.

Dataset	%BC	DT	TD3	CQL	IQL	EQL	SfBC	DQL	IDQL
locomotion-v2 total	666.2	672.6	677.4	698.5	692.4	**725.3**	680.4	**791.2**	**739.2**
antmaze-v0 total	134.2	112.2	163.8	303.6	378.0	386	**445.2**	418.5	**474.6**
total	800.4	784.8	841.2	1002.1	1070.4	1111.3	1125.6	**1209.7**	**1213.8**
training time	10m	960m	20m	80m	20m	20m	785m	240m	60m

Table 2: **Full comparison offline RL.** We compare IDQL-A to other prior offline RL methods. IDQL outperforms all methods in total score and receives the strongest antmaze results.

Dataset	Cal-QL	RLPD	IQL	IDQL-Max	IDQL-Imp
antmaze-umaze	$- \rightarrow -$	$0.0 \rightarrow$ **99.0**	**86.7** $\rightarrow$ **96.0**	**92.0** $\rightarrow$ **99.0**	**93.5** $\rightarrow$ **99.5**
antmaze-umaze-diverse	$- \rightarrow -$	$0.0 \rightarrow$ **99.0**	**75.0** $\rightarrow$ 84.0	**78.7** $\rightarrow$ 85.6	**78.4** $\rightarrow$ 73.0
antmaze-medium-play	54.0 $\rightarrow$ **98.0**	$0.0 \rightarrow$ **99.5**	72.0 $\rightarrow$ **95.0**	**83.3** $\rightarrow$ **97.8**	**84.4** $\rightarrow$ **94.0**
antmaze-medium-diverse	73.0 $\rightarrow$ **98.0**	$0.0 \rightarrow$ **98.0**	**68.3** $\rightarrow$ **92.0**	**84.7** $\rightarrow$ **98.0**	**84.0** $\rightarrow$ **98.7**
antmaze-large-play	28.0 $\rightarrow$ **90.0**	$0.0 \rightarrow$ **88.0**	25.5 $\rightarrow$ 46.0	**60.1** $\rightarrow$ **88.0**	**60.3** $\rightarrow$ **90.0**
antmaze-large-diverse	32.0 $\rightarrow$ **94.0**	$0.0 \rightarrow$ **87.5**	42.6 $\rightarrow$ 60.7	**61.1** $\rightarrow$ **90.7**	**61.4** $\rightarrow$ **93.0**
total	$- \rightarrow -$	$0.0 \rightarrow$ **571.0**	408.2 $\rightarrow$ 473.7	**459.9** $\rightarrow$ **559.1**	**462** $\rightarrow$ **548.2**

Table 3: **Online finetuning results.** We compare finetuning performance with IDQL using two approaches: "-Max" refers to freezing the behavior policy and finetuning the value networks and "-Imp" refers to finetuning all networks as described in Section 4.2.

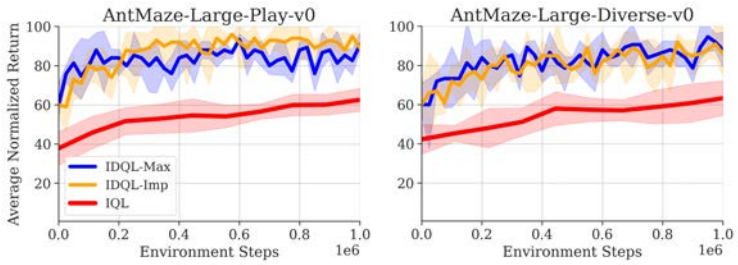


Figure 4: **Online finetuning results for antmaze-large tasks.** Finetuning training curves for various sampling and finetuning strategies for our method on antmaze-large tasks. The frozen actor requires only 100k samples to reach peak performance.

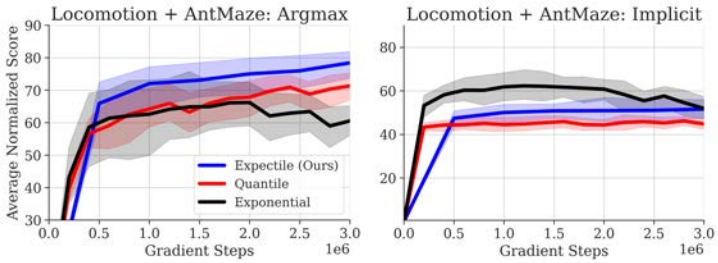


Figure 5: **IQL Objective Comparison.** We compare the IQL objectives on all offline D4RL tasks using argmax extraction (left) and implicit policy extraction (right). As expected, deterministic argmax extraction attains better performance, though the stochastic implicit actor still attains good results. The original expectile loss leads to the best overall results with the argmax.

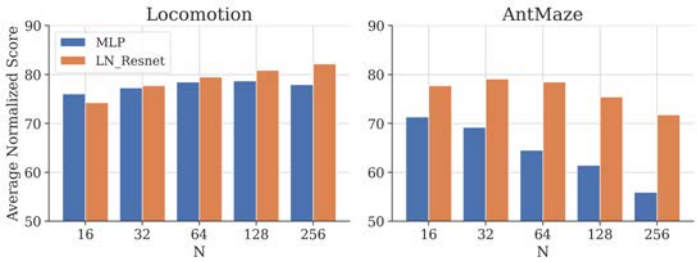


Figure 6: **Sensitivity to N ablation.** Architecture comparison on offline RL for D4RL domains over various N. The LN_Resnet architecture remains less sensitive to N across domains. Results are averaged over 10 seeds.

6.3 IQL Objective Ablations

In Section 4, we discussed different objectives and the different implicit policies they induce. We compare the critic losses on the D4RL benchmarks in Figure 5. Both expectiles and quantiles perform well with argmax extraction, but the exponential loss is unstable and performs worse. This indicates that the exponential critic proposed by Garg et al. [14] and Xu et al. [49] require more tuning and tricks to perform well, though these works also report instability issues with the objective. The implicit policy distributions also do not perform as well as the argmax extraction, but this is most likely because these policies have a non-zero probability of taking a poor action. This follows similar findings that stochastic policies work best for critic learning, but deterministic policies work best for evaluation [17, 5]. Overall, the expectile objective performs the strongest with greedy extraction.

6.4 Evaluating Diffusion Architecture Choices

In Section 5, we introduced design choices that were important for reducing outliers and increasing the expressivity of diffusion models. To confirm that our architecture aids in performance and sensitivity to N, we compare our LN_Resnet architecture to a three-layer MLP which contains a similar number of activation layers. We sweep over N and measure the total final score. Results are in Figure 6. Using an LN_resnet is crucial for a reduction in sensitivity to N: for locomotion results, a higher N leads to stronger performance, and for antmaze results increasing N has a small effect on results. For the MLP architecture, increasing N decreases performance, especially in antmaze.

7 Discussion and Future Work

In this work, we generalize IQL into an actor-critic method, where choices of convex asymmetric critic loss induces a behavior regularized implicit actor (Theorem 4.1). The implicit policy is shown to be an importance weighted behavior distribution, but the form of this distribution is quite complex. This suggests that policy extraction methods based on simple Gaussian policies, of the sort employed in prior work, might not perform well in IQL. We confirm this hypothesis by proposing a new policy

extraction approach based on expressive diffusion models, describe a number of architecture design decisions that make such policies work well in practice, and present state-of-the-art results across offline RL benchmarks. Our method performs particularly well when the amount of hyperparameter tuning is severely restricted, which is important for practical applications of offline RL where tuning is often difficult or impossible.

Our analysis shows that, although we can generalize IQL to use a variety of loss functions, the original expectile loss ends up performing the best on current tasks. However, we believe that this generalization still has considerable value in informing future research. First, it illuminates how IQL is actually an actor-critic method, and provides for an entire family of implicit actor-critic methods. Future work might investigate new and more effective loss functions, or loss functions that are more amenable to tractable policy extraction. Our work also illustrates how critical the choice of policy extraction method is for implicit Q-learning methods, in contrast to explicit actor-critic methods where the critic can adapt to the capacity limitations of the actor. Finally, our work provides an effective, easy to implement, computationally efficient, and relatively hyperparameter insensitive approach for integrating diffusion models into offline RL. While a number of recent works have proposed to use diffusion models for imitation learning and reinforcement learning, we hope that our work will provide a simple recipe for how more effective behavior modeling can translate directly into more effective RL.

8 Acknowledgements

This research was supported by the Office of Naval Research and AFOSR FA9550-22-1-0273, with compute support from Berkeley Research Computing. We thank Manan Tomar for help with early drafts of the paper.

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

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

[3] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

[4] 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/google/jax.

[5] David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Advances in neural information processing systems*, 34:4933–4946, 2021.

[6] Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *International conference on machine learning*, pages 872–881. PMLR, 2019.

[7] 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.

[8] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[9] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[10] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=rif3a5NAxU6.

[11] 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.

[12] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

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

[14] Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. *arXiv preprint arXiv:2301.02328*, 2023.

[15] 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 *International Conference on Machine Learning*, pages 3682–3691. PMLR, 2021.

[16] Wonjoon Goo and Scott Niekum. Know your boundaries: The necessity of explicit behavioral cloning in offline rl. *arXiv preprint arXiv:2206.00695*, 2022.

[17] 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*, pages 1861–1870. PMLR, 2018.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023. URL http://github.com/google/flax.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[21] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[22] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

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

[24] Ilya Kostrikov. JAXRL: Implementations of Reinforcement Learning algorithms in JAX, 10 2021. URL https://github.com/ikostrikov/jaxrl.

[25] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021.

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

[27] 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.

[28] 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.

[29] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pages 5556–5566. PMLR, 2020.

[30] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. *Reinforcement learning: State-of-the-art*, pages 45–73, 2012.

[31] 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.

[32] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

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

[34] Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *arXiv preprint arXiv:2303.05479*, 2023.

[35] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[36] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Pv1GPQzRrC8.

[37] 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.

[38] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, volume 227 of *ACM International Conference Proceeding Series*, pages 745–750. ACM, 2007. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273590.

[39] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[40] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

[41] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[42] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

[43] 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.

[44] 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.

[45] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020.

[46] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

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

[48] Da Xu, Yuting Ye, and Chuanwei Ruan. Understanding the role of importance weighting for deep learning. *arXiv preprint arXiv:2103.15209*, 2021.

[49] Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. *arXiv preprint arXiv:2303.15810*, 2023.

A Reinforcement Learning Definitions

RL is formulated in the context of a Markov decision process (MDP), which is defined as a tuple $(\mathcal{S}, \mathcal{A}, p_0(s), p_M(s'|s,a), r(s,a), \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, initial state distribution $p_0(s)$, transition dynamics $p_M(s'|s,a)$, reward function $r(s,a)$, and discount γ. The goal is to recover a policy $\pi(a|s)$ that maximizes the discounted sum of rewards or return in the MDP,

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\rho(\pi)}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)],$$

where $\rho(\pi)$ describes the distribution over trajectories, $\{s_t, a_t \sim \pi(a|s_t), s'_t \sim p_M(s'|s_t, a_t)\}_{t=0}^{\infty}$, induced by a given policy π. A widely used framework for off-policy RL is Q-learning [46], which involves fitting a Q-function $Q(s,a)$ to match the discounted returns starting at state s and action a and following the best current policy at the next state s' ($\arg\max_{a'} Q(s', a')$).

B Proof of Theoretical Results

B.1 Proof of Theorem 4.1

Proof. We write the objective in Equation 6.

$$\arg\min_{V(s)} \mathbb{E}_{a\sim\mu(a|s)}[f(Q(s,a) - V(s))]$$

Note that the objective function is convex with respect to $V(s)$

$$\begin{aligned} 0 &= \frac{\partial}{\partial V(s)} \mathbb{E}_{a\sim\mu(a|s)}[f(Q(s,a) - V(s))]\Big|_{V=V^*} \\ &= -\mathbb{E}_{a\sim\mu(a|s)}[f'(Q(s,a) - V^*(s))] \end{aligned}$$

Due to convexity of f and the assumption $f'(0) = 0$, $f'(x) = |f'(x)| \cdot \text{sign}(x) = |f'(x)| \frac{x}{|x|}$

$$= \mathbb{E}_{a\sim\mu(a|s)}\left[\frac{|f'(Q(s,a) - V^*(s))|(Q(s,a) - V^*(s))}{|Q(s,a) - V^*(s)|}\right].$$

We then define the implicit policy to be $\pi_{\text{imp}}(a|s) = \frac{\mu(a|s)|f'(Q(s,a)-V^*(s))|}{Z_{\text{imp}}|Q(s,a)-V^*(s)|}$, where Z_{imp} is a normalization constant, and rewrite the above expression as

$$= \mathbb{E}_{a\sim\pi_{\text{imp}}(a|s)}[(Q(s,a) - V^*(s))]$$

$$= \frac{\partial}{\partial V(s)} - \frac{1}{2} \cdot \mathbb{E}_{a\sim\pi_{\text{imp}}(a|s)}[(Q(s,a) - V(s))^2]\Big|_{V=V^*} = 0$$

This means that $V^*(s)$ is a solution for the optimization problem

$$\arg\min_{V(s)} \mathbb{E}_{a\sim\pi_{\text{imp}}(a|s)}[(Q(s,a) - V(s))^2]$$

□

C Additional Derivations

C.1 Proof of Optimal Solution to Equation 11

Proof. We can rewrite the objective to remove irrelevent terms

$$\begin{aligned} &\arg\min_{V(s)} \mathbb{E}_{a\sim\mu(a|s)}[\exp(\alpha(Q(s,a) - V(s))) - \alpha(Q(s,a) - V(s))] \\ &= \arg\min_{V(s)} \mathbb{E}_{a\sim\mu(a|s)}[\exp(\alpha(Q(s,a) - V(s))) + \alpha V(s)] \end{aligned}$$

We expand the expectation for the objective. Assume $\mu(a|s) > 0$

$$= \arg\min_{V(s)} \sum_a \mu(a|s)\Big(\exp(\alpha(Q(s,a) - V(s))) + \alpha V(s)\Big)$$

Note that the objective function is convex with respect to $V(s)$.

$$
\begin{aligned}
0 &= \frac{\partial}{\partial V(s)}\Big|_{V=V^*} \sum_a \mu(a|s)\Big(\exp(\alpha(Q(s,a)-V(s))) + \alpha V(s)\Big) \\
&= \sum_a \mu(a|s)\Big(-\alpha\exp(\alpha(Q(s,a)-V^*(s))) + \alpha\Big) \\
&= \alpha\sum_a \mu(a|s)\Big(-\exp(\alpha(Q(s,a)-V^*(s)))\Big) + \alpha\sum_a \mu(a|s) \\
&= \alpha\sum_a \mu(a|s)\Big(-\exp(\alpha(Q(s,a)-V^*(s)))\Big) + \alpha \\
&= \alpha\exp(-\alpha V^*(s))\sum_a\Big(-\exp(\alpha Q(s,a)+\log\mu(a|s))\Big) + \alpha
\end{aligned}
$$

We can now easily solve for $V^*(s)$.

$$
\begin{aligned}
1 &= \exp(-\alpha V^*(s))\sum_a\Big(\exp(\alpha Q(s,a)+\log\mu(a|s))\Big) \\
\frac{1}{\sum_a\Big(\exp(\alpha Q(s,a)+\log\mu(a|s))\Big)} &= \exp(-\alpha V^*(s)) \\
\frac{1}{\alpha}\log\sum_a\Big(\exp(\alpha Q(s,a)+\log\mu(a|s))\Big) &= V^*(s) = V_{\exp}(s)
\end{aligned}
$$

□

C.2 Kullback–Leibler Divergence between Exponential Implicit Policy and behavior Distribution

We look to compute the KL divergence between the implicit policy of the exponential distribution and the behavior policy.

$$
\begin{aligned}
D_{KL}(\mu(a|s)\|\pi_{\exp}(a|s)) &= \sum_a \mu(a|s)\log\big(\frac{\mu(a|s)}{\pi_{\exp}(a|s)}\big) \\
&= \sum_a \mu(a|s)\log\big(\frac{\exp(\log\mu(a|s))}{\exp(\alpha(Q(s,a)-V(s))+\log\mu(a|s))}\big) \\
&= \sum_a \mu(a|s)\log\big(\frac{1}{\exp(\alpha(Q(s,a)-V(s)))}\big) \\
&= \sum_a \mu(a|s)\alpha(V(s)-Q(s,a)) \\
&= \mathbb{E}_{(s,a)\sim\mathcal{D}}[\alpha(V(s)-Q(s,a))]
\end{aligned}
$$

This shows that the divergence of the behavior policy with the implicit policy is related to the advantage as well as the temperature hyperparameter.

D Experimental Details

Our implementation is based from the jaxrl repo [24] which uses the JAX [4] framework using Flax [19].

D.1 Standard Offline RL

For the standard offline RL benchmark, we train the critic with 1.5 million gradient updates and the diffusion behavior policy with 3 million gradient updates. We found critic learning to be slightly unstable with more updates than 2 million. All reported results ("-A") for Table 1 are taken from Table 1 in Kostrikov et al. [26] except for EQL, SfBC, and DQL which are taken directly from their main offline RL results table. As in IQL, we standardize the rewards for the locomotion tasks and we subtract rewards by one for the antmaze tasks. This is also done for fairness in the one hyperparameter reruns of prior methods ("-1"). For training time, we take numbers also from Kostrikov et al. [26] and from Chen et al. [7].

D.2 One-Hyperparameter Offline RL

For the one hyper parameter experiment, we re-implemented IQL from the IQL repo., we reran CQL from the CQL repo., and we reran DiffusionQL from the DQL repo. We only sweep over the main-hyper parameter mention in the paper and select the best performing one per domain; all other hyper parameters are left constant. For CQL, we sweep over $\lambda \in \{1.0, 2.0, 5.0, 10.0\}$ or the weight of the CQL term, for DQL we swept over the $\eta \in \{1.0, 2.0, 2.5, 3.0\}$ or weight of the Q maximization objective, and for IQL and IDQL, we swept over the expectile $\tau \in \{0.6, 0.7, 0.8, 0.9\}$. For each domain (locomotion and antmaze), we select the best-performing hyperparameter (e.g. $\tau = 0.7$ for locomotion and $\tau = 0.9$ for antmaze).

While this may not fairly represent each algorithm, our main intention was to show how our method can perform well out of the box (i.e. the posted GitHub implementation) without the need for excessive tuning. We find that many non-IQL methods tend to overtune their ant maze results to include many more changes than used in locomotion tasks. For example, CQL requires many changes from their locomotion configuration linked here to get strong antmaze results. While not bad per se, it does indicate that these methods require more careful tuning to work well across domains. Furthermore, some methods do not fairly present their results as they either tune their method per environment or take the max of their evaluation curve. As a result, another intention of ours was to clearly state a protocol to fairly compare algorithms side by side.

D.3 Finetuning

For fine-tuning, we pretrain the critic for 1 million steps and the diffusion BC actor for 2 million steps. During online finetuning, we take gradient step on the critic for each environment step. If the actor is fine-tuned as well, we take 2 gradient steps per environment step. We found fine-tuning failed to improve from pre-training on the adroit tasks, but this is not unexpected. The adroit fine-tuning tasks require significant exploration to achieve strong returns, so having behavior regularization can be harmful. We leave this as future work.

E Table Hyperparameters

The critic and value network follow the same parameterization as in IQL [26] (2 Layer MLP with hidden size 256 and ReLU activations). Other details mentioned about architecture are for the diffusion model.

LR (For all networks)	3e-4
Critc Batch Size	256
Actor Batch Size	1024
τ **Expectiles**	0.7 (locomotion), 0.9 (antmaze)
τ **Quantiles**	0.6 (locomotion), 0.8 (antmaze)
α **Exponential**	1.0 (locomotion), 0.5 (antmaze)
Critic Grad Steps	1.5e6 ("-A"), 1e6 ("-1")
Actor Grad Steps	3e6 ("-A"), 2e6 ("-1")
Critic Pre-Training Steps	1e6 (Figure 3)
Actor Pre-Training Steps	2e6 (Figure 3)
Target Critic EMA	0.005
T	5
N	32 (antmaze "-A"), 128 (loco "-A"), 64 ("-1")
Beta schedule	Variance Preserving [43]
Dropout Rate [42]	0.1
Number Residual Blocks	3
Actor Cosine Decay [32]	Number of Actor Grad Steps
Optimizer	Adam [23]

F Additional Experiments and Results

F.1 Other Prior Offline RL Work Results

We compare our method to a number of recent offline RL algorithms discussed in the related work section: %BC, Decision Transformers (DT) [8], TD3+BC (TD3) [12], conservative Q-learning (CQL) [28], implicit

Dataset	%BC	DT	TD3	CQL	IQL	EQL	SfBC	DQL	IDQL
halfcheetah-m	48.4	42.6	48.3	44.0	47.4	47.7	45.9	**51.1**	**51.0**
hopper-m	56.9	67.6	59.3	58.5	66.3	71.1	57.1	**90.5**	65.4
walker2d-m	75.0	74.0	**83.7**	72.5	78.3	77.9	**81.5**	**87.0**	**82.5**
halfcheetah-mr	40.6	36.6	44.6	**45.5**	44.2	44.8	37.1	**47.8**	**45.9**
hopper-mr	75.9	82.7	60.9	**95.0**	**94.7**	**97.3**	86.2	**101.3**	**92.1**
walker2d-mr	62.5	66.6	81.8	77.2	73.9	75.9	65.1	**95.5**	85.1
halfcheetah-me	**92.9**	86.8	**90.7**	**91.6**	86.7	89.8	**92.6**	**96.8**	**95.9**
hopper-me	**110.9**	**110.9**	98.0	**105.4**	91.5	**107.1**	**108.6**	**111.1**	**108.6**
walker2d-me	**109.0**	**108.1**	**110.1**	**108.8**	**109.6**	**109.6**	**109.8**	**110.1**	**112.7**
locomotion-v2 total	666.2	672.6	677.4	698.5	692.4	**725.3**	680.4	**791.2**	**739.2**
antmaze-u	62.8	59.2	78.6	74.0	87.5	87.2	**92.0**	**93.4**	**94.0**
antmaze-ud	50.2	53.0	71.4	**84.0**	62.2	69.2	**85.3**	66.2	**80.2**
antmaze-mp	5.4	0.0	10.6	61.2	71.2	73.5	**81.3**	76.6	**84.2**
antmaze-md	9.8	0.0	3.0	53.7	70.0	71.2	**82.0**	78.6	**84.8**
antmaze-lp	0.0	0.0	0.2	15.8	39.6	41	**59.3**	46.4	**63.5**
antmaze-ld	6.0	0.0	0.0	14.9	47.5	47.3	45.5	57.3	**67.9**
antmaze-v0 total	134.2	112.2	163.8	303.6	378.0	386	**445.2**	418.5	**474.6**
total	800.4	784.8	841.2	1002.1	1070.4	1111.3	1125.6	**1209.7**	**1213.8**
training time	10m	960m	20m	80m	20m	20m	785m	240m	60m

Table 4: **Standard evaluation for offline RL.** IDQL performs on par or better than other SOTA offline RL methods. Algorithm names are shortened to save space. Results for our method are averaged over 10 seeds.

Dataset	IQL	Diffuser	IDQL
umaze	47.4	**113.9**	57.9
medium	34.9	**121.5**	89.5
large	58.6	**123.0**	90.1
total	94	**358.5**	237.5

Table 5: **Maze2D results.** We include maze2d results compared to IQL and Diffuser. Our method outperforms IQL and remains close to diffusers performance.

Q-learning (IQL) [26], extreme Q-learning (EQL) [14], and Selecting from Behavior Candidates(SfBC) [7]. We don't include all comparisons in the main paper to save space and since our "-1" method outperforms all the reported methods other than DQL. Some of these prior works tune the hyperparameters coarsely, for example, Wang et al. [44] uses per-task tuning on both locomotion and antmaze tasks and Garg et al. [14] takes the max evaluation during training. This generally leads to better performance but implies an assumption of being able to do per-task online hyperparameter selection. Results are reported in Table 4. IDQL remains competitive with all prior methods on the locomotion tasks and outperforms all prior methods on the antmaze tasks.

F.2 Maze2D Results

IQL performs quite poorly in the maze2d environment compare to other offline RL methods. Specifically, Diffuser [22] performs very strongly when compared to IQL and utilizes diffusion. We look to see if our method aids the performance of IQL. Results are in Table 5. Our method outperforms IQL consistently but comes short against Diffuser. We suspect that model based planning generalizes better in the Maze2d environments.

F.3 Diffusion Steps T and Beta Schedule

The choice of number of diffusion steps (T) was also pretty crucial to good performance on D4RL. Strangely, the T needed for quality samples of the simple 2D continuous datasets was much larger than required for D4RL. We found T = 50 to be best for the 2D datasets, but T = 5 to be best for D4RL. This might be because control datasets are more deterministic than the continuous datasets presented (which include lots of added noise on top of a simple signal), but this remains an open question. We include ablations of various T on D4RL in Figure 7. Increasing T has a small but negative effect on performance. One thing to note is that for T > 20, evaluation is quite slow due to needing to resample the entire chain every step of the Markov process.

Also, the choice of beta schedule has a strong impact in performance. We ablate over a linear [20], cosine [35], and variance preserving schedule [43] in Figure 7. We found the vp to work the best for D4RL, but cosine also worked very well with small T. The linear schedule required far too large of T to perform well and ended up

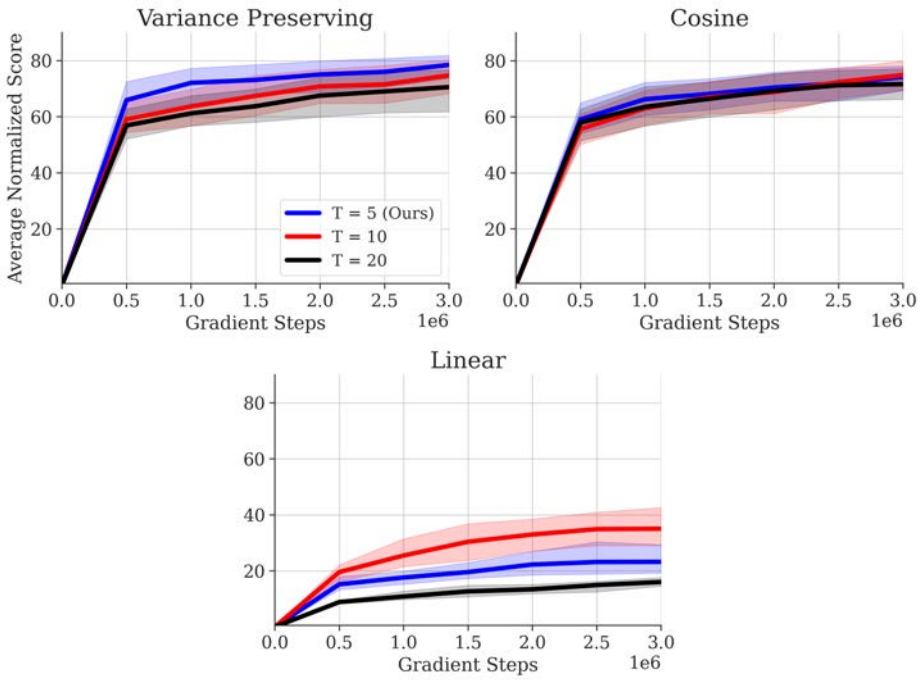


Figure 7: Sweeps over different schedules and T. The variance preserving schedule with low T generally works the best.

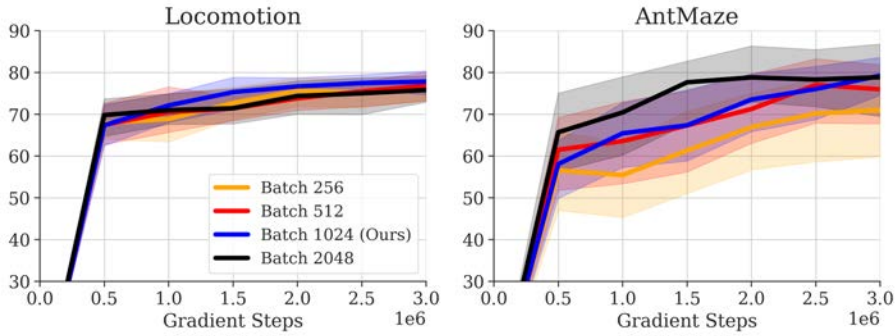


Figure 8: Finetuning training curves for various sampling and fine-tuning strategies for our method on antmaze-large tasks. The frozen actor requires only 100k samples to reach peak performance. Results are averaged over 10 seeds.

being a poor choice for D4RL tasks. We suspect that the signal to noise ratio induced by the schedules is very important in proper expression. In particular, the first noise step is a crucial part of the noising process. In summary, we recommend to sweep over different T and schedule for the best possible modeling of the action space.

F.4 Batch Size

As discussed in Section 5, batch size was important for reducing outliers samples. We ablate over different batch sizes on the DDPM loss in the D4RL benchmarks. Results are in Figure 8. While batch size has a small effect on the locomotion tasks, having a larger batch size leads to stronger performance on the antmaze tasks. Though, the batch size is not as crucial as the choice of architecture or capacity of the model.

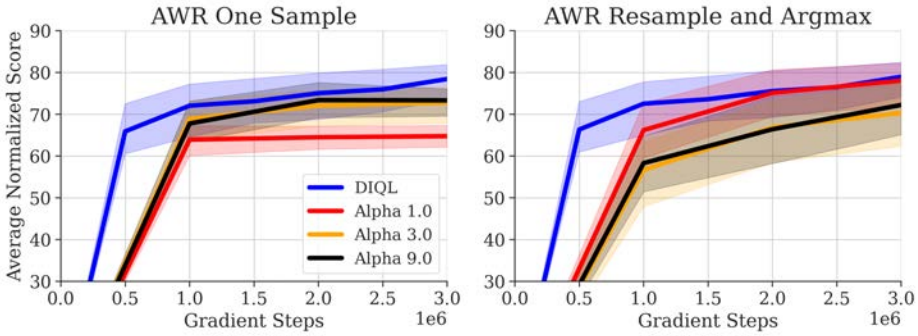


Figure 9: Weighted DDPM loss ablations using AWR. We try both one sample (Left) and multiple samples plus argmax (Right). Results are averaged over 10 seeds.

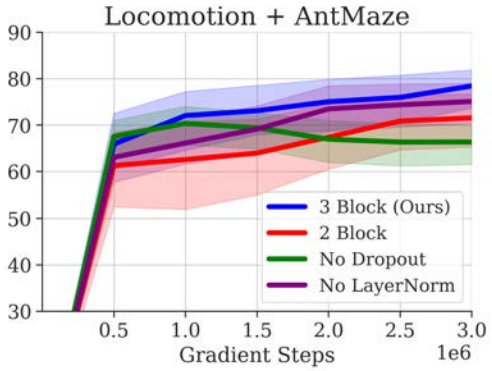


Figure 10: Ablation of capacity, dropout, and layer norm on offline results.

F.5 AWR weighted DDPM Ablation.

To show that our method performs better than directly training a policy, we ablate using AWR to weigh the DDPM loss over a batch. We try two versions: in one we sample the policy only once and in the other we sample $N = 64$ times and take the sample that maximizes the critic. Results are in Figure 9. In the case of the one sample method, the AWR weighted objective usually performs worse than IDQL overall and at best performs on par with the multi-sample setup. As a result, AWR is not necessary for achieving strong performance and therefore adds an unnecessary training parameter.

F.6 Other Important Architecture Details

There are other important architecture details for strong performance on D4RL. We ablate over capacity (number of resent blocks), layer norm, and dropout to see the effect in Figure 10. As shown in Section 5, having a larger capacity and well regularized network is crucial for strong performance. Dropout has a large benefit in reducing overfitting and layer norm has a small benefit in overall performance.

F.7 Full Learning Curves

Figure 11 and Figure 12 are the learning curves for our one hyperparameter ("-1") variant of IDQL.

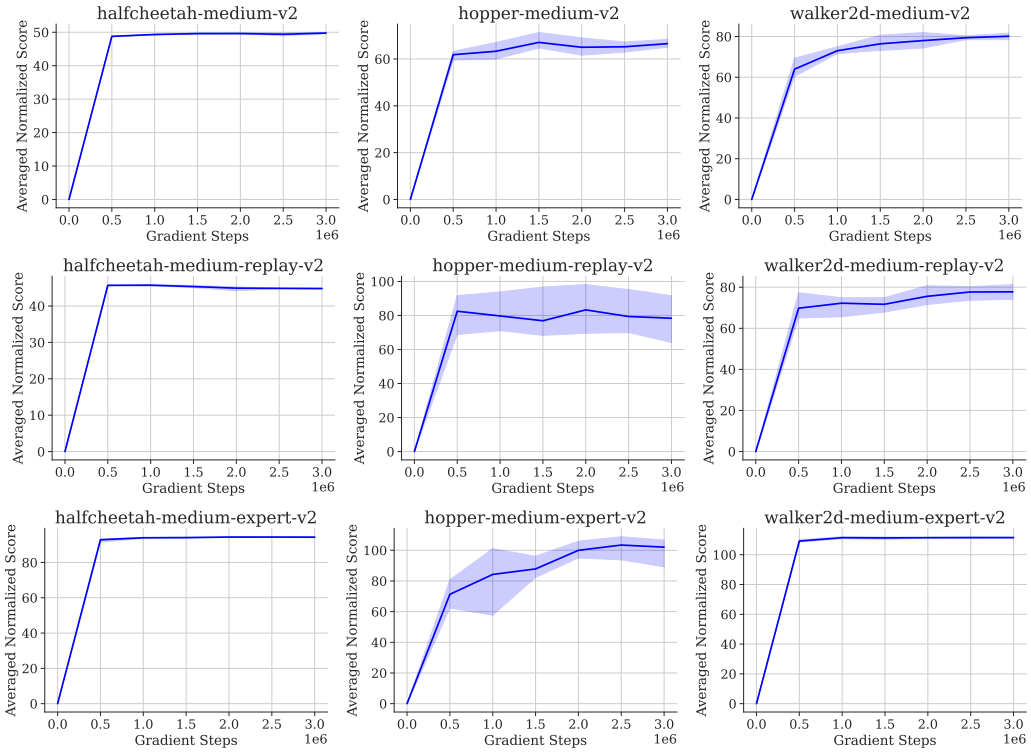


Figure 11: Training curves for locomotion tasks for one-hyperparameter variant of IDQL.

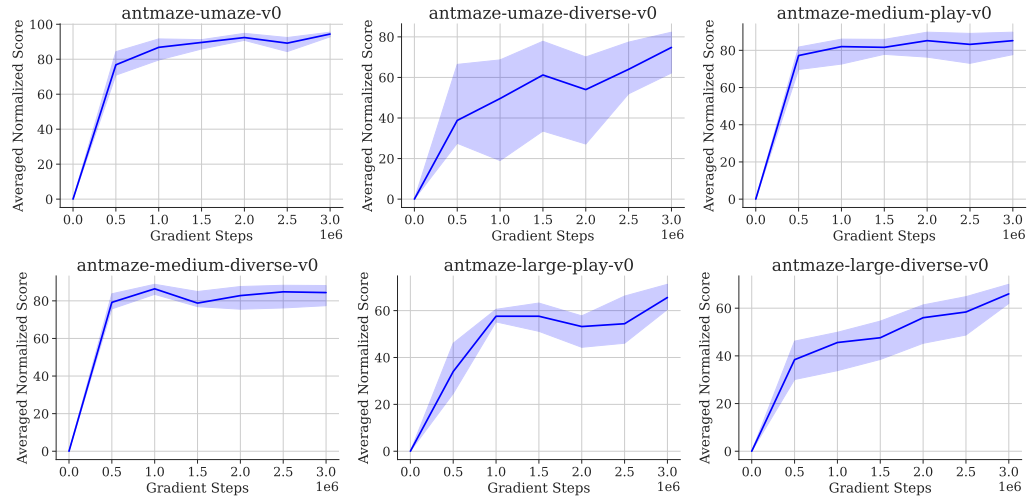
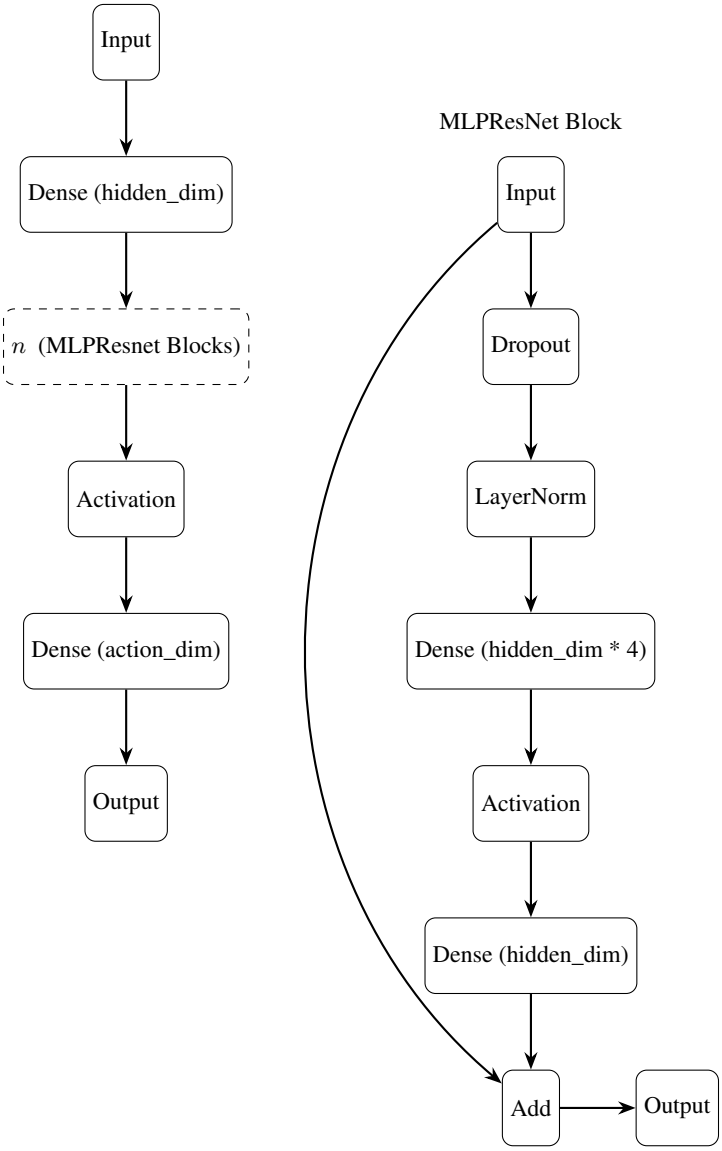


Figure 12: Training curves for antmaze tasks for one-hyperparameter variant of IDQL.

G LN_Resnet Architecture

We depict the architecture used in the LN_Resnet described in Section 5. We use hidden dim size 256 and $n = 3$ blocks for our practical implementation.



H Extra Related Work

Other Offline RL Methods. Brandfonbrener et al. [5] use a SARSA critic objective (equivalent to expectile $\tau = 0.5$) for critic learning and then a greedy extraction via AWR.

Measuring Statistic for RL. In the family of algorithms, we present quantiles as a potential statistic to measure that induces an implicit policy distribution. Quantile statistics have also been used in RL prior for estimating distributions [9, 29] over Q-functions. Though, just like IQL, we avoid querying the Q-function on out of distribution actions.