CS 8803 DLM Project Milestone: Consistent Q-Learning

Albert Wilcox **Chetan Reddy** **Atharva Mete**

Abstract: Offline reinforcement learning (RL) aims to learn an optimal policy using a previously collected static dataset. Parameterizing these policies using diffusion models has demonstrated impressive capability in learning from multi-modal data. However, naive parameterization necessitates differentiating through the entire diffusion chain, rendering the process computationally inefficient. Recent works have attempted to address this issue by either approximating the diffusion Markov chain [1] or by employing consistency policies [2], but such formulations compromises their expressivity and limits their scalability to complex domains. To this end we propose Consistent Q-Learning (CoQL), a novel approach to directly evaluate noisy actions throughout the diffusion chain that gives meaningful gradients in one step. We conduct extensive experiments on the D4RL benchmark. The results show that CoQL achieves comparable performance to baselines like Diffusion-QL [3] while significantly reducing the training time, thereby addressing the computational inefficiency inherent in naive diffusion-based policy parameterization.

1 Introduction

Due to the relative expensiveness of collecting and learning from online data, the robotics community has in recent years poured considerable resources into approaches to learn skills and policies using large offline datasets of pre-collected interactions with the world. Two key directions towards addressing this problem setting include behavior cloning and offline RL. Behavior cloning approaches [4, 5, 6] do this by assuming their offline dataset contains optimal or near optimal demonstrations and learning to directly copy the demonstrator actions. The advantage of these approaches is that they are often far more stable and, assuming access to an adequate dataset, can learn to complete a range of difficult tasks. However, they struggle when they are only given access to suboptimal data, since they have no mechanism to differentiate between good and bad actions. This motivates offline RL [7, 8], which instead learns to complete tasks by optimizing a reward function, opening up the possibility of learning policies that outperform suboptimal demonstrators.

Diffusion models [9, 10] are a recently emerging class of generative models which learn to sample from complex, multimodal data distributions through an iterative denoising process known as Langevin Dynamics [11]. Lately roboticists have seen great success using these models to sample from demonstrator datasets for behavior cloning [4, 12, 13]. However, due to the aforementioned limitations of behavior cloning approaches, there is also substantial effort to train diffusion policies using offline RL approaches. While there are numerous lines of research which set out to do this, we focus on a line of work which uses a TD3+BC [14] style update to introduce a policy improvement loss while preventing out-of-distribution errors by regularizing the policy to stay close to offline demonstrations. A seminal work in this direction is Diffusion-QL [3], which simply trains a critic and induces policy improvement losses by differentiating through the full diffusion denoising chain. While this approach shows impressive results in simple environments, its scalability is limited by the computational demands of differentiating through long denoising chains. To that end, others have attempted to address this by introducing approximations to the diffusion process that are more efficient but less scalable to difficult tasks [1, 2]. With these limitations in mind, we set out to design

2nd Conference on CS 8803 DLM (DLM 2024), Mogadishu, Somalia.

an algorithm that is efficient enough to scale to difficult problems without the limiting approximations introduced in other work.

In this project we will investigate a method we call *Consistent Q-Learning (CoQL)*. The idea behind this method is to use the insights from consistency models [15] to train a *consistent critic*, which can be used to evaluate noisy actions at any point in the diffusion denoising process. By combining the consistent critic with a novel score-matching objective, we introduce a method for adding policy improvement objectives to diffusion model training while maintaining the efficient training method introduced in [10].

We evaluate the efficiency and generality of our method on the popular D4RL benchmark [16]. We first benchmark the efficiency of CoQL on gym-locomotion tasks. To show it's efficacy in high dimensional settings we benchmark it performance on Adroid domain, that involves controlling a 24-DoF robotic hand. We show that our method is performs comparably with baseline methods [3, 1, 2]. We also visualize the method on a simple bandit task to illustrate how our formulation allows the consistent critic to elegantly approximates the original critic. Lastly we benchmark our method on navigation environment where we validate the action chunking version of our proposed method.

2 Related work

2.1 Diffusion Policies

Our work is most closely related to a recent line of work considering the application of diffusion models to policy learning for decision making tasks. [17] first introduced the idea, applying it to the imitation learning setting by training a diffusion model to sample full trajectories from the demonstrator data distribution. [4] build on this, demonstrating empirically that diffusion models are more successful sampling from multi-modal distributions and scaling the approach to physical robot settings. While these approaches show strong results imitating expert demonstrations, they lack the ability to acquire new behaviors or improve upon suboptimal demonstrations, motivating the use of RL objectives to train these models.

Towards the goal of improving diffusion policies using reward functions, several works have applied diffusion to offline RL settings, for example by conditioning on desired rewards [18, 19], filtering actions according to a Q filter [20], or, most similarly to our project, adding a policy improvement term to a behavior cloning loss [3, 1, 2]. While these methods show strong results in simple tasks, their scalability to more difficult manipulation tasks is limited by their inefficiency in the case of [3] and detrimental approximations in the case of [1, 2]. In this project we hope to develop a method that does away with these limitations to scale to more difficult problems.

2.2 Consistency Models

Consistency models [15] are a class of generative models based on diffusion models, whose goal is to achieve similar sample quality and diversity without the considerable overhead that comes with iterative denoising. The key insight is to learn a consistency function f which directly predicts the sample x_0 given an arbitrary point along the probability flow trajectory x_t. The models does this through two key insights. First, the authors parameterize f such that it is the identity when $t = 0$, so $f(x_0) = x_0$. They do this by parameterizing the model as

$$f_\theta(x, t) = c_{\text{skip}}(t)x + c_{\text{out}}(t)F_\theta(x, t) \tag{1}$$

where $c_{\text{skip}}(0) = 1$, $c_{\text{out}}(0) = 0$ and c_{skip} and c_{out} are both continuous and differentiable. Second, they discretize the trajectory into N segments with boundaries at $t_0, \ldots, t_N$, and train the models using the insight that when properly trained, $f(x_{t_n}) = f(x_{t_{n+1}})$. Using this framework, authors show they can sample reasonable outputs using only a small number of denoising steps. They can be trained as a distillation of a diffusion model or in isolation.

2.3 Diffusion-QL

Diffusion-QL [3] is a recent work using diffusion models to parameterize policies for offline RL. The paper uses a TD3+BC-style approach, training a critic Q_ϕ and updating the policy using a weighted sum of the original denoising loss $\mathcal{L}_\text{D}$ from Equation 4 and a Q-maximization loss,

$$\mathcal{L}_{\text{Diffusion-QL}}(\theta) = \mathcal{L}_\text{D}(\theta) - \eta \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_\theta(s)}\left[Q_\phi(s,a)\right] \tag{2}$$

where $\mathcal{D}$ is the offline dataset, π_θ is the diffusion policy and η is a weighting hyperparameter. While this method achieves impressive results in the environments shown in the paper, this method is not scalable since optimizing Equation 2 requires differentiating through the full diffusion MC. While other approaches used 50 [12] or 100 [4] diffusion steps in training, Diffusion-QL could only use 5.

3 Preliminaries

RL: The environment in RL is typically defined by a Markov decision process (MDP): $M = (S, A, P, R, \gamma, d_0)$, with state space S, action space A, environment dynamics $P(s'|s,a) : S \times S \times A \to [0,1]$, reward function $R : S \times A \to \mathbb{R}$, discount factor $\gamma \in [0,1)$, and initial state distribution d_0 [21]. The goal is to learn policy $\pi_\theta(a|s)$, parameterized by θ, that maximizes the cumulative discounted reward $\mathbb{E}[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)]$. The action-value or Q-value of a policy π is defined as $Q^\pi(s_t,a_t) = \mathbb{E}_{a_{t+1},a_{t+2},\ldots\sim\pi}[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)]$.

In episodic RL, the algorithm is given access to the MDP via trajectory samples for arbitrary π of the algorithm's choosing. Off-policy methods may use experience replay to store these trajectories in a replay buffer $\mathcal{D}$ of transitions (s_t, a_t, s_{t+1}, r_t), and use an off-policy algorithm such as Q-learning [22] to optimize π. However, these methods still iteratively collect additional data, and omitting this collection step can produce poor results. For example, running state-of-the-art off-policy RL algorithms on trajectories collected from an expert policy can result in diverging Q-values [23].

Offline RL: In this setting, the algorithm no longer has access to the MDP, and is instead presented with a fixed static dataset of transitions $\mathcal{D} = \{(s,a,r,s')\}$. The (unknown) policy that generated this data is referred to as a behavior policy π_b. Effective offline RL algorithms must handle distribution shift, as well as data collected via processes that may not be representable by the chosen policy class. [24] provide a comprehensive discussion of the problems affecting offline RL.

In this work we consider the setting of Offline RL.

3.1 Score-Based Generative Models

Diffusion models are a class of generative models which learn to sample high-quality outputs from high-dimensional, multimodal outputs through an iterative denoising process. More specifically, they define a Markov chain where the forward process $x_t \to x_{t+1}$ iteratively adds Gaussian noise to a sample $x_0 \sim p_\text{data}(x_0)$ until reaching a final state $x_T \sim \mathcal{N}(0, I)$, which they reverse using a learned Langevin dynamics [11] step

$$x_{t-1} = \alpha_t(x_t + \gamma_t \epsilon_\theta(x_t, t) + \mathcal{N}(0, \sigma_t^2 I)) \tag{3}$$

where ϵ_θ is trained to predict noise added to samples and $\alpha_t, \gamma_t, \sigma_t$ are hyperparameters. [10] show that this ϵ_θ can be learned by sampling $x_0 \sim p_\text{data}, t \sim \text{Uniform}(\{1,2,\ldots,T\}), \epsilon \sim \mathcal{N}(0,I)$ and minimizing the loss

$$\mathcal{L}(\theta) = \left\|\epsilon - \epsilon_\theta(x_0 + \sigma_t\epsilon, t)\right\|^2. \tag{4}$$

3.2 Implicit Q-Learning

A key problem in offline reinforcement learning is to prevent agents from taking out of distribution actions and prevent policies from optimizing around them and their corresponding (potentially overestimated) out of distribution critic values. Implicit Q-Learning [8] proposes to address this by training the critic to estimate the expectile τ over the distribution of actions, removing the

requirement to query the critic on potentially out-of-distribution policy actions. Specifically, they learn a parameterized critic Q_θ, target critic $Q_{\hat{\theta}}$ and value function V_φ. Using the asymmetric squared error objective

$$L_2^\tau(u) = |\tau - \mathbb{1}(u < 0)|u^2, \tag{5}$$

the value function is updated according to the following objective:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\tau(Q_{\hat{\phi}}(s,a) - V_\psi(s))\right]. \tag{6}$$

This value function is then used to update the critic using the following objective:

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[(r(s,a) + \gamma V_\psi(s') - Q_\phi(s,a))^2\right]. \tag{7}$$

Notably, this formulation learns the value functions without any need to query the policy. [8] argue this is useful for offline RL, since it prevents optimizing around erroneous critic values from out of distribution actions. Hansen-Estruch et al. [20] show it is particularly useful when dealing with diffusion policies, which are particularly expensive to sample from.

Once the critic is learned, IQL optimizes the policy using advantage weighted regression:

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp(\alpha(Q_{\hat{\phi}}(s,a) - V_\psi(s))\log\pi_\theta(a|s)\right] \tag{8}$$

where $\alpha \in [0, \infty]$ is a temperature hyerparameter.

4 Method

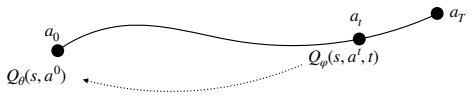


Figure 1: The consistent critic learns, based on any noised version of an action along the diffusion MC, to predict the output of the original critic on the original action.

In this section we outline the method we are implementing to address this problem. Prior work [3, 1, 25] induces policy gradients by learning a traditional Q function, and using it to backpropagate through the entire diffusion MC. While this method achieves strong initial results, backpropagating through the diffusion MC is limited as it presents a difficult tradeoff between computational speed and sample quality. We note that the method in [10] goes to great lengths to avoid doing this. The key insight to address this issue is to use the idea from consistency models to learn a consistent Q function, which can accurately predict Q values at any point in the diffusion chain. With this, we can optimize the policy using an advantage weighted version of Equation 4. We refer to this algorithm as Consistent Q Learning (CoQL).

4.1 Consistent Q Functions

Given a learned critic Q_ϕ, we learn a consistent Q function $\tilde{Q}_\varphi$. The goal is to distill the original critic into a new critic that can predict useful values given noisy inputs. Similarly with consistency models, which learn to predict the final denoised x_0 given a noisy version x_t, we learn to predict the value for the final denoised action $Q(s, a_0)$ based on the state and noisy action a_t. Similarly with [15] in Equation 1, we parameterize this model in order to maintain the property that it should equal the original critic Q_ϕ when $t = 0$, so

$$\tilde{Q}_\varphi(s, a_t, t) = c_{\text{skip}}(t)(a_t)Q_\phi(s, a_t, t) + c_{\text{out}}(t)F_\varphi(a_t) \tag{9}$$

where F_φ is a neural network and c_{skip} and c_{out} have the same properties as discussed in Section 3. Note that t here is the timestep in the diffusion MC, not the RL timestep. Then, we discretize the probability flow trajectory, sample $n \sim \text{Uniform}(1, \ldots, N)$ and train with the loss

$$\mathcal{L}(\varphi) = d(\tilde{Q}_\varphi(s, a_{t_n}, t), \tilde{Q}_\varphi(s, a_{t_{n-1}}, t)) \tag{10}$$

where d is some distance metric such as ℓ_2 distance. This training procedure trains $\tilde{Q}_\varphi$ to match the output of Q_ϕ when presented with noisy versions of actions, meaning it can be used to evaluate the policy at any point in the denoising process. The resulting function is illustrated in Figure 1.

4.2 Policy Learning

Our policy is parameterized as a conditional DDPM, which denoises from a prior noise distribution into actions conditioned on state values. To train the denoising network we use an advantage weighted version of the original denoising loss from Equation 4. Specifically, we sample a state action pair $s, a_0 \sim \mathcal{D}$ and follow the process from Section 3.1 to sample a diffusion timestep t, noise ϵ, and noisy action a_t. Then, we use our denoising network ϵ_θ to compute a slightly denoised action a_{t-1}. Finally, we apply our consistent critic to a_{t-1} to compute an advantage-weighted denoising loss:

$$\mathcal{L}_\pi(\theta) = \left\| \exp\left(\alpha \left(\tilde{Q}_\varphi(s, a_{t-1}, t-1) - V_\psi(s)\right)\right)(\epsilon - \epsilon_\theta(a_t, t)) \right\|^2. \tag{11}$$

Intuitively, this loss function uses the consistent critic to denoise in a way that prioritizes high advantage actions.

4.3 Adapting to Action Chunking

As discussed in Chi et al. [4], diffusion policies work best when used to output chunks of actions rather than single actions, as we have considered so far. Thus, we here present a method for adapting the methods discussed above to allow us to train action chunking diffusion policies with the RL objective in Equation 11.

The main change that needs to be made is to develop a critic which can condition on sequences of actions, so that the consistent critic can be used to reason about noisy sequences of actions. To do this, we sample a length H sequence of transitions $\tau = (s_i, a_i, r_i), \ldots, (s_{i+H}, a_{i+H}, r_{i+H}), s_{i+H+1}$ from the replay buffer, and compute a long-horizon Bellman backup, updating the critic according to the following objective:

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{\tau \sim \mathcal{D}} \left[\left(\sum_{j=0}^{H} \gamma^j r_{i+j} + \gamma^{H+1} V_\psi(s_{i+H+1}) - Q_\phi(s_i, a_{i:i+H}) \right)^2 \right]. \tag{12}$$

From here, we can use the action chunking critic wherever we would have used the original critic in the original algorithm.

4.4 Energy-Based Action Selection

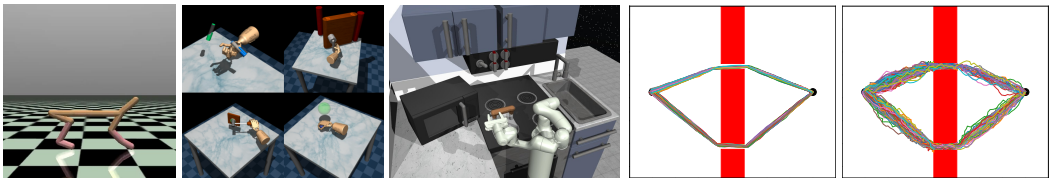
While many other RL algorithms with stochastic policies, such as SAC [26], use the mode of the policy at test time, this is intractable to compute with a diffusion policy. In order to mitigate this effect, as in [3, 1, 20] we sample N actions $a_1, \ldots, a_N$ at test time and sample a final action from this set from a categorical distribution with weights according to a softmax over $Q_\phi(s, a_i)$.

5 Experiments

We plan to evaluate CoQL in the offline RL setting, using a subset of the popular D4RL benchmark [16] as well as RoboMimic [27]. Our main baseline is Diffusion-QL and we will compare to a key ablation where we do not use a consistent critic.

We aim to test the following hypotheses:

1. CoQL is comparable with or improves upon baseline results
2. CoQL is better than the ablation without a consistent critic, is it provides more accurate Q value estimations at noisy actions



(a) D4RL (b) Navigation

Figure 2: The domains where we experiment. (a) D4RL tasks evaluate the agent's ability to learn in a variety of settings with a variety of data distributions. (b) The navigation environment tests the agent's ability to learn to complete long horizon tasks and improve from suboptimal demonstrations. (Left) One dataset (`nav-opt`) includes optimal demonstrations while other datasets (Right) force the algorithm to learn from suboptimal demonstrations.

3. CoQL performs better than baselines on high-dimensional tasks, as we don't have to approximate the diffusion process as [1, 2] did, and we can use a larger number of diffusion timesteps.

5.1 Datasets

As shown in Figure 2, we experiment on two task suites. The D4RL benchmark [16] is a widely used task suite which we use to evaluate the singlet action version of our algorithm. The navigation suite requires precise long horizon reasoning and learning from suboptimal multimodal demonstrations, and we use it to evaluate the action-chunking version of our algorithm.

5.1.1 D4RL

We test our method on 3 domains of tasks within the D4RL suite [16], Gym locomotion, Adroit, and Kitchen as shown in Figure 2a. We choose these tasks, as they are a common benchmark for many offline reinforcement learning algorithms, and our main baseline, diffusion-QL reports their results on these tasks. All of these tasks are run in simulation using the Open AI gym API and the Mujoco simulator. We directly use the state information as our input modality rather than images, as our focus is on the reinforcement learning algorithm and not perception.

For each of the gym locomotion tasks, `halfcheetah`, `hopper`, and `walker2d`, there are three different datasets we run experiments on, `medium`, `medium-replay`, and `medium-expert`. The medium datasets were collected by training a soft actor-critic online, early stopping, and running rollouts using this partially trained policy. The medium-replay datasets use the data collected in the replay buffer throughout the training of a policy until it reaches a "medium" level performance. The medium-expert datasets contains a mixture of expert demonstrations and sub-optimal rollouts. These tasks are low-dimensional with suboptimal data, and provide a baseline set of tasks to prove our approach is comparable to the state-of-the-art.

We also test on the adroit task `pen` which is the task of twirling a pen to reach a goal position. Within this, there are two variations: `human` and `cloned`. The `human` dataset consists of 25 human demonstrations. To collect the `cloned` dataset, they trained an imitation policy on the demonstrations and mixed rollouts produced by the imitation policy with the demonstrations. This task is high-dimensional, which we believe is a better environment to demonstrate our approach.

The `kitchen` task requires completing multiple subtasks to reach a goal state. The `complete` dataset consists of trajectories completing the desired tasks in order. The `partial` dataset has some trajectories where the robot completes the tasks in order and others where it does not. The `mixed` dataset does not contain any trajectories that complete the desired tasks in order and requires the agent to learn how to stitch trajectories. This is another complex high-dimensional task, which will test our belief that CoQL is more scalable in comparison to previous diffusion approaches.

5.1.2 Navigation

The navigation environment, shown in Figure 2b, is used to test the ability to perform long horizon reasoning after learning from suboptimal demonstrations. In the environment, the agent must navigate from a fixed start location through one of two small slits to a fixed goal location. The agent receives a reward of -1 for each timestep where it is out of the goal and the episode terminates upon task completion or collision with the boundary. Boundary collisions lead to a large negative reward of -100.

We include four dataset variations for the environment. The `nav-opt` dataset provides optimal demonstrations, while the `nav-noisy` dataset injects Gaussian noise at every action. The `nav-slow` and `nav-slow-noisy` datasets sample a velocity from the set $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ at the beginning of each trajectory and scales the action from the `nav-opt` and `nav-noisy` demonstrator respectively by the sampled velocity.

The `nav-opt` environment is used to determine whether the agent can learn to match optimal demonstrations. The noisy environments are used to determine whether the agent can learn to outperform them by outputting more smooth action sequences. The slow environment is used to determine whether the agent can learn to filter out the slower trajectories and output the faster ones.

5.2 Baselines

Due to the different availability of algorithms concerning singlet and chunked actions, we consider a different set of baselines for each version of the algorithm.

For the singlet action version of the algorithm we compare to the following baselines:

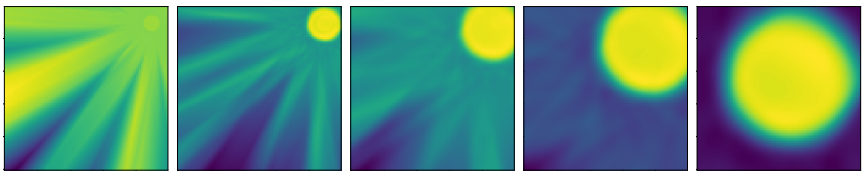
- **DQL**: Diffusion Q Learning which we discuss in 2.3. We report the results from their paper.
- **EDP**: Efficient Diffusion Policy approximately constructs actions with a one stage denoising process. This approach can be plugged into a variety of offline reinforcement learning approaches including TD3+BC, CRR, and IQL. We report the best results from their paper across their various models. It mostly follows that TD3+BC was the best for the gym-locomotion tasks and IQL for the pen and kitchen tasks.
- **No Consistent Critic**: This is an ablation of our approach where we do not train a consistent critic, and we instead use the regular critic to evaluate the noisy actions.

For the action chunking version we compare to the following baselines:

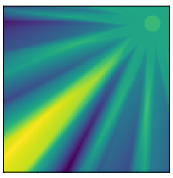
- **BC**: The behavior cloning baseline uses a diffusion policy to output a sequence of actions, similarly to Chi et al. [4].
- **IDQL**: The IDQL baseline from Hansen-Estruch et al. [20] learns a standard diffusion policy as well as a chunk-based IQL-style critic as described in Section 4.3. At inference time, it samples numerous action sequences from the diffusion policy and samples a final action sequence from a softmax categorical distribution over these action sequences according to the values from the critic, also known as energy-based action selection as described in Section 4.4. Note that since our pipeline also performs energy-based action selection, this baseline also ablates our policy update.
- **No Consistent**: Same as described above.

5.3 Results

In this section we present the results from the experiments described above as well as a didactic bandit experiment to visualize the outputs of the consistent critic.



(a) Consistent critic outputs at time $t = \{0, 2, 5, 10, 20\}$.



(b) Original critic output.

Figure 3: Consistent critic outputs at various diffusion timesteps compared with the original critic output. We see that the consistent critic successfully smooths out the critic outputs for out of distribution actions.

Environment	DQL	EDP	No Consistent	CoQL
halfcheetah-medium-v2	0.508 ± 0.005	0.521	0.545 ± 0.003	$\mathbf{0.608 \pm 0.006}$
hopper-medium-v2	$\mathbf{0.822 \pm 0.030}$	0.819	0.690 ± 0.032	0.685 ± 0.095
walker2d-medium-v2	$\mathbf{0.871 \pm 0.006}$	0.869	0.432 ± 0.198	0.863 ± 0.014
halfcheetah-medium-replay-v2	0.474 ± 0.001	0.494	0.512 ± 0.004	$\mathbf{0.532 \pm 0.013}$
hopper-medium-replay-v2	1.004 ± 0.003	1.010	$\mathbf{1.025 \pm 0.003}$	0.898 ± 0.135
walker2d-medium-replay-v2	0.871 ± 0.071	**0.949**	0.902 ± 0.002	0.833 ± 0.051
halfcheetah-medium-expert-v2	0.953 ± 0.011	0.955	$\mathbf{0.958 \pm 0.024}$	0.734 ± 0.202
hopper-medium-expert-v2	$\mathbf{1.061 \pm 0.067}$	0.974	0.177 ± 0.052	0.792 ± 0.391
walker2d-medium-expert-v2	1.099 ± 0.001	**1.102**	1.012 ± 0.082	1.029 ± 0.017
Average	0.853	**0.869**	0.717	0.793
pen-human-v1	0.590 ± 0.106	**0.727**	0.712 ± 0.156	0.657 ± 0.213
pen-cloned-v1	0.452 ± 0.130	**0.700**	0.558 ± 0.057	0.66 ± 0.163
Average	0.521	**0.714**	0.635	0.659
kitchen-complete-v0	$\mathbf{0.775 \pm 0.088}$	0.755	0.742 ± 0.155	0.725 ± 0.235
kitchen-partial-v0	0.528 ± 0.078	0.528	0.625 ± 0.074	$\mathbf{0.717 \pm 0.024}$
kitchen-mixed-v0	0.519 ± 0.047	0.608	$\mathbf{0.692 \pm 0.042}$	$\mathbf{0.692 \pm 0.012}$
Average	0.607	0.630	0.686	**0.711**

Table 1: Results on the D4RL benchmark. We see that CoQL achieves impressive results in some locomotion environments and does particularly well in the manipulation environments, outperforming DQL across the board and outperforming EDP in the kitchen environments.

5.3.1 Bandit Experiment

For this experiment, we designed a bandit problem where the agent receives a binary reward for outputting actions within some radius r of a goal location g. We sample actions according to a Gaussian distribution $a_i \sim \mathcal{N}(g, rI)$. We visualize the output of the consistent critic and original critic by running all actions in the grid through the critic, and the outputs are shown in Figure 3. We see that the consistent critic effectively leads to reasonable outputs for noisy actions that are out of the original action distribution while still providing a sharp boundary for noiseless actions, while the original critic suffers for out of distribution actions.

5.3.2 D4RL Experiments

In Table 1 we present results from the D4RL set of tasks, comparing against the baselines DQL, EDP, and CoQL without the consistent critic.

For the locomotion environments, we find that our model is comparable but slightly worse than our baselines DQL and EDP. We also find that the no consistent critic ablation ends up producing similar results to our model, and is actually better in some instances.

For the adroit environments we see that, as expected, our method as well as EDP, which are designed to scale better to higher-dimensional tasks, outperform DQL which does not scale well. While our

Environment	Demos	BC	IDQL	No Consistent	CoQL
nav-opt	1.0	$\mathbf{0.933 \pm 0.047}$	0.8 ± 0.082	$\mathbf{0.933 \pm 0.047}$	$\mathbf{0.933 \pm 0.094}$
nav-slow	0.95	0.767 ± 0.047	$\mathbf{0.967 \pm 0.047}$	0.9 ± 0.082	0.833 ± 0.125
nav-noisy	0.245	0.3 ± 0.082	$\mathbf{0.667 \pm 0.047}$	0.6 ± 0.082	0.633 ± 0.094
nav-slow-noisy	0.33	0.467 ± 0.047	$\mathbf{0.767 \pm 0.047}$	0.6 ± 0.082	0.6 ± 0.0
Average		0.617	0.800	0.758	0.750

Table 2: Success rates for various algorithms on the navigation environment.

Environment	Demos	BC	IDQL	No Consistent	CoQL
nav-opt	-53	-57.8 ± 2.2	-63.4 ± 5.9	$\mathbf{-53.8 \pm 3.4}$	-65.9 ± 14.8
nav-slow	-108	-150.4 ± 11.1	$\mathbf{-56.6 \pm 2.5}$	-63.7 ± 6.3	-68.5 ± 8.7
nav-noisy	-115	-110.2 ± 5.5	$\mathbf{-86.4 \pm 4.2}$	-103.3 ± 13.0	-89.5 ± 6.9
nav-slow-noisy	-154	-203.7 ± 18.2	$\mathbf{-181.5 \pm 7.4}$	-186.4 ± 4.8	-199.8 ± 9.8
Average		-130.5	-97.0	-101.8	-105.9

Table 3: Cumulative rewards for various algorith,s on the navigation environment.

method outperforms DQL, it fails to outperform EDP and the no consistent ablation. This suggests that the extra algorithmic burden learning the consistent critic is not useful for this task.

For the kitchen environment, we see that our algorithm achieves SOTA results learning from the partial and mixed datasets. This suggests that the consistent critic is useful in these settings where action stitching is important.

5.3.3 Navigation Experiments

For the navigation experiments, we use the action chunking version of our policy. Thus, these experiments serve to show whether CoQL is useful for improving action chunking policies.

In Tables 3 and 2 we present cumulative rewards and success rates respectively for CoQL, the baselines and the offline demonstrations.

Overall, we see that the algorithms which use RL lead to substantial improvement over the demonstrations and behavior cloning baseline. However, overall the experiments do not demonstrate that CoQL is any better than the baselines. In fact, we see that IDQL outperforms CoQL more or less across the board. This suggests that the CoQL policy improvement objective as it is now does not provide much benefit for action chunking policies.

6 Conclusion

In this work, we present Consistent Q-Learning (CoQL), a consistency-based formulation for learning consistent critic to evaluate noisy actions from diffusion Markov Chain. We observe the CoQL performs comparably with the selected baselines in most D4RL environments. It outperforms DQL in high-dimensional dexterous settings proving its scalability and achieves SOTA results on kitchen environment that requires trajectory stitching. We also observe that no-consistent ablation also performs decently well suggesting that the original critic also gives "good" gradient estimates on noisy actions. We suppose this is due to the fact that adding noise smooths out the overall gradient landscape and in-turn helps the learning process. In future we plan to explore other policy improvement formulations that may work better than the advantage weighting objective and experiment with more complicated domains where the gains from consistent critic are more pronounced.

References

[1] B. Kang, X. Ma, C. Du, T. Pang, and S. Yan. Efficient diffusion policies for offline reinforcement learning, 2023.

[2] Z. Ding and C. Jin. Consistency models as a rich and efficient policy class for reinforcement learning, 2023.

[3] Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

[4] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

[6] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. *CoRL*, 2023.

[7] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/0d2b2061826a5df3221116a5085a6052-Paper.pdf.

[8] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

[9] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[10] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[11] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning*, 2011. URL https://api.semanticscholar.org/CorpusID:2178983.

[12] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Proceedings of the 2023 Conference on Robot Learning*, 2023.

[13] Z. Xian, N. Gkanatsios, T. Gervet, T.-W. Ke, and K. Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=W0zgY2mBTA8.

[14] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[15] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models, 2023.

[16] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[17] M. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[18] H. He, C. Bai, K. Xu, Z. Yang, W. Zhang, D. Wang, B. Zhao, and X. Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. 2023.

[19] H. Yuan, K. Huang, C. Ni, M. Chen, and M. Wang. Reward-directed conditional diffusion: Provable distribution estimation and reward improvement. 2023.

[20] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. 2023.

[21] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[22] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3–4):293–321, 1992. URL http://www.cs.ualberta.ca/~sutton/lin-92.pdf.

[23] A. Kumar, J. Fu, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction, 2019.

[24] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

[25] Y. Chen, H. Li, and D. Zhao. Boosting continuous control with consistency policy, 2023.

[26] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[27] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=JrsfBJtDFdI.

7 Team Contributions

Team Member	Contribution Summary
Albert	Developed the core algorithm for Consistent Q-Learning (CoQL). Proposed the advantage weighted regression formulation. Implemented the navigation environment and baselines. Performed extensive D4RL experiments.
Atharva	Conducted experiments and ablations on D4RL and PushT environment for best hyperparameters. Implemented some baselines like Extreme Q-learning.
Chetan	Performed the didactic bandit experiments and validated the consistency formulation. Improved the consistency formulation following recent developments in consistency models.