A PROOFS

A.1 PROOF OF LEMMA 2

Proof. We can rewrite $V_{\tau_1}(s)$ as

$$
\begin{aligned}
V_{\tau_1}(s) &= \mathbb{E}^{\tau_1}_{a\sim\mu(\cdot|s)}[r(s,a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)}[V_{\tau_1}(s')]] \\
&\leq \mathbb{E}^{\tau_2}_{a\sim\mu(\cdot|s)}[r(s,a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)}[V_{\tau_1}(s')]] \\
&= \mathbb{E}^{\tau_2}_{a\sim\mu(\cdot|s)}[r(s,a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)}\mathbb{E}^{\tau_1}_{a'\sim\mu(\cdot|s')}[r(s',a') + \gamma \mathbb{E}_{s''\sim p(\cdot|s',a')}[V_{\tau_1}(s'')]] \\
&\leq \mathbb{E}^{\tau_2}_{a\sim\mu(\cdot|s)}[r(s,a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)}\mathbb{E}^{\tau_2}_{a'\sim\mu(\cdot|s')}[r(s',a') + \gamma \mathbb{E}_{s''\sim p(\cdot|s',a')}[V_{\tau_1}(s'')]] \\
&= \mathbb{E}^{\tau_2}_{a\sim\mu(\cdot|s)}[r(s,a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)}\mathbb{E}^{\tau_2}_{a'\sim\mu(\cdot|s')}[r(s',a') + \gamma \mathbb{E}_{s''\sim p(\cdot|s',a')}\mathbb{E}^{\tau_1}_{a''\sim\mu(\cdot|s'')}[r(s'',a'') + \ldots]] \\
&\vdots \\
&\leq V_{\tau_2}(s)
\end{aligned}
$$

□

B EXPERIMENTAL DETAILS

Experimental details. For the MuJoCo locomotion tasks, we average mean returns over 10 evaluation trajectories and 10 random seeds. For the Ant Maze tasks, we average over 100 evaluation trajectories. We standardize MuJoCo locomotion task rewards by dividing by the difference of returns of the best and worst trajectories in each dataset. Following the suggestions of the authors of the dataset, we subtract 1 from rewards for the Ant Maze datasets. We use $\tau = 0.9$ and $\beta = 10.0$ for Ant Maze tasks and $\tau = 0.7$ and $\beta = 3.0$ for MuJoCo locomotion tasks. We use Adam optimizer (Kingma & Ba, 2014) with a learning rate $3 \cdot 10^{-4}$ and 2 layer MLP with ReLU activations and 256 hidden units for all networks. We use cosine schedule for the actor learning rate. We parameterize the policy as a Gaussian distribution with a state-independent standard deviation. We update the target network with soft updates with parameter $\alpha = 0.005$. And following Brandfonbrener et al. (2021) we clip exponentiated advantages to $(-\infty, 100]$. We implemented our method in the JAX (Bradbury et al., 2018) framework using the Flax (Heek et al., 2020) neural networks library.

Extended results on Locomotion and Ant Maze tasks. We present learning curves for MuJoCo locomotion tasks in Fig. 4. We also present results on Locomotion and Ant Maze for different values of τ in Fig. 5 and Table 3. We want to emphasize that $\tau = 0.5$ corresponds to using the mean squared error instead of expectile regression.

Results on Franca Kitchen and Adoit tasks. For Franca Kitchen and Adroit tasks we use $\tau = 0.7$ and the inverse temperature $\beta = 0.5$. Due to the size of the dataset, we also apply Dropout (Srivastava et al., 2014) with dropout rate of 0.1 to regularize the policy network. See complete results in Table 4.

Table 3: Effect of τ. Fitting $V(s)$ with mean squared error ($\tau = 0.5$) is not sufficient to propagate the signal through recursion and fails to solve more challenging medium and large tasks.

	IQL w/ $\tau = 0.5$ (MSE)	IQL w/ $\tau = 0.7$	IQL w/ $\tau = 0.9$
antmaze-umaze-v0	44.2±7.2	87.0±2.3	87.5±2.6
antmaze-umaze-diverse-v0	53.6 ±12.7	57.2±11.9	62.2±13.8
antmaze-medium-play-v0	0.0±0.0	4.0±2.0	71.2 ±7.3
antmaze-medium-diverse-v0	0.0 ±0.0	2.6±1.4	70.0±10.9
antmaze-large-play-v0	0.0±0.0	0.2±0.4	39.6 ±5.8
antmaze-large-diverse-v0	0.0 ±0.0	1.2±1.6	47.5±9.5
total	97.8±19.9	152.2±19.6	378.0±49.9

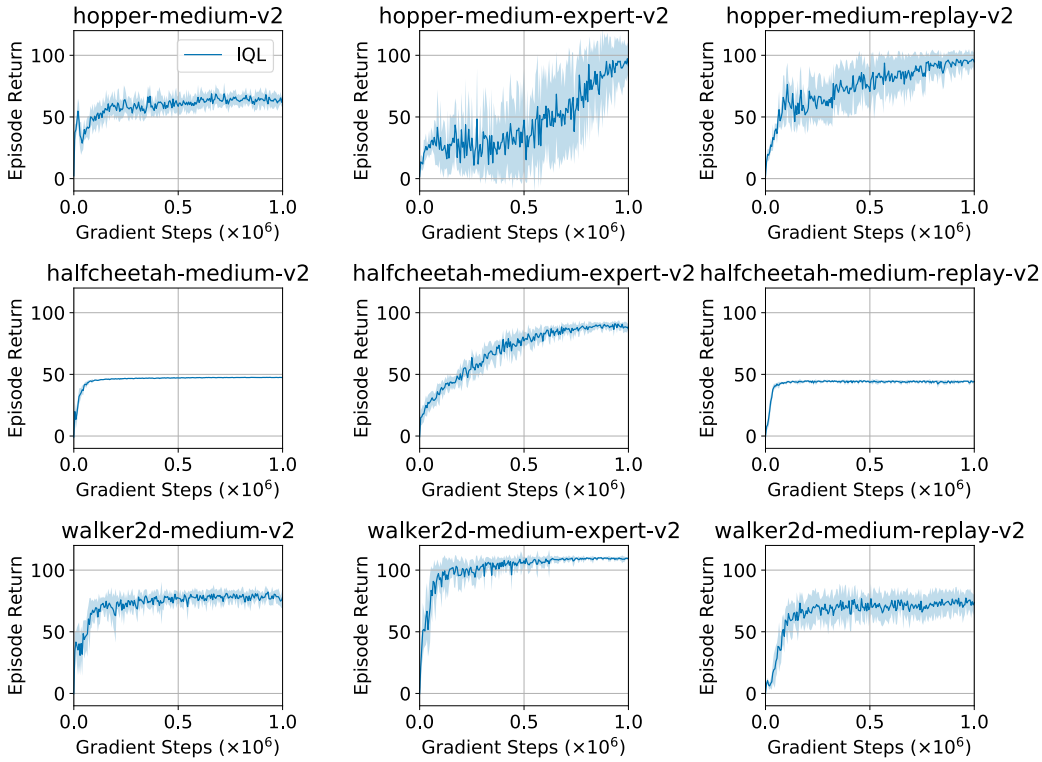

Figure 4: Learning curves for MuJoCo locomotion tasks.

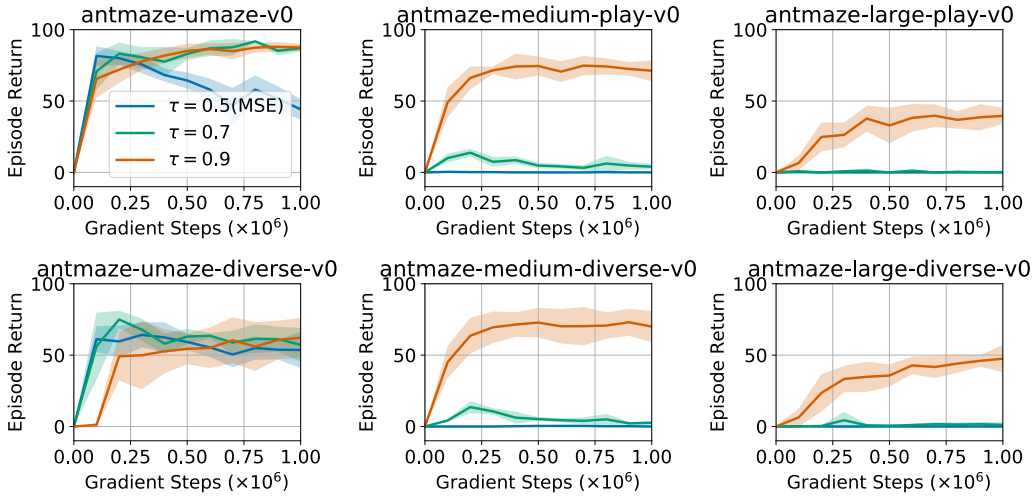

Figure 5: Results on Ant Maze for different values of τ. Note that $\tau = 0.5$ corresponds to using the mean squared error instead of expectile regression.

Table 4: Evaluation on Franca Kitchen and Adroit tasks from D4RL

dataset	BC	BRAC-p	BEAR	Onestep RL	CQL	Ours
kitchen-complete-v0	**65.0**	0.0	0.0	-	43.8	**62.5**
kitchen-partial-v0	38.0	0.0	0.0	-	**49.8**	**46.3**
kitchen-mixed-v0	**51.5**	0.0	0.0	-	**51.0**	**51.0**
kitchen-v0 total	**154.5**	0.0	0.0	-	144.6	**159.8**
pen-human-v0	63.9	8.1	-1.0	-	37.5	**71.5**
hammer-human-v0	1.2	0.3	0.3	-	**4.4**	1.4
door-human-v0	2	-0.3	-0.3	-	**9.9**	4.3
relocate-human-v0	0.1	-0.3	-0.3	-	0.2	0.1
pen-cloned-v0	37	1.6	26.5	**60.0**	39.2	37.3
hammer-cloned-v0	0.6	0.3	0.3	**2.1**	**2.1**	**2.1**
door-cloned-v0	0.0	-0.1	-0.1	0.4	0.4	**1.6**
relocate-cloned-v0	-0.3	-0.3	-0.3	-0.1	-0.1	-0.2
adroit-v0 total	104.5	9.3	25.1	-	93.6	**118.1**
total	259	9.3	25.1	-	238.2	**277.9**

C FINETUNING EXPERIMENTAL DETAILS

For finetuning experiments, we first run offline RL for 1M gradient steps. Then we continue training while collecting data actively in the environment and adding that data to the replay buffer, running 1 gradient update / environment step. All other training details are kept the same between the offline RL phase and the online RL phase. For dextrous manipulation environments (Rajeswaran et al., 2018), we use $\tau = 0.8$ and $\beta = 3.0$, 25000 offline training steps, and add Gaussian noise with standard deviation $\sigma = 0.03$ to the policy for exploration.

For baselines we compare to the original implementations of AWAC (Nair et al., 2020) and CQL (Kumar et al., 2020). For AWAC we used `https://github.com/rail-berkeley/rlkit/tree/master/rlkit`. We found AWAC to overfit heavily with too many offline gradient steps, and instead used 25000 offline gradient steps as in the original paper. For the dextrous manipulation results, we report average return normalized from 0 to 100 for consistency, instead of success rate at the final timestep, as reported in Nair et al. (2020). For CQL, we used `https://github.com/aviralkumar2907/CQL`. Our reproduced results offline are worse than the reported results, particularly on medium and large antmaze environments. We were not able to improve these results after checking for discrepancies with the CQL paper authors and running CQL with an alternative implementation (`https://github.com/tensorflow/agents`). Thus, although for offline experiments (Table 1) we report results from the original paper, for finetuning experiments we did not have this option and report our own results running CQL in Table 5.

Dataset	Offline	Online
antmaze-umaze-v0	70.1 → **99.4**	**86.7** → **96.0**
antmaze-umaze-diverse-v0	31.1 → **99.4**	**75.0** → 84.0
antmaze-medium-play-v0	23.0 → 0.0	**72.0** → **95.0**
antmaze-medium-diverse-v0	23.0 → 32.3	**68.3** → **92.0**
antmaze-large-play-v0	1.0 → 0.0	**25.5** → **46.0**
antmaze-large-diverse-v0	1.0 → 0.0	**42.6** → **60.7**
pen-binary-v0	46.2 ± 6.3	63.3 ± 1.9
door-binary-v0	1.3 ± 0.7	42.0 ± 3.2
relocate-binary-v0	0.3 ± 0.4	23.3 ± 14.5

Table 5: Error bars for fine-tuning experiments with 20 seeds, showing one standard deviation.

D CONNECTIONS TO PRIOR WORK

In this section, we discuss how our approach is related to prior work on offline reinforcement learning. In particular, we discuss connections to BCQ Fujimoto et al. (2019).

Our batch constrained optimization objective is similar to BCQ (Fujimoto et al., 2019). In particular, the authors of BCQ build on the Q-learning framework and define the policy as

$$\pi(s) = \arg\max_{\substack{a \\ s.t.(s,a)\in\mathcal{D}}} Q(s,a). \tag{8}$$

Note that in contrast to the standard Q-learning, maximization in Eqn. (8) is performed only over the state-action pairs that appear in the dataset. In Fujimoto et al. (2019), these constraints are implemented via fitting a generative model $\mu(\cdot|s)$ on the dataset, sampling several candidate actions

from this generative model, and taking an argmax over these actions:

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

However, this generative model can still produce out-of-dataset actions that will lead to querying undefined Q-values. Thus, our work introduces an alternative way to optimize this objective without requiring an additional density model. Our approach avoids this issue by enforcing the hard constraints via estimating expectiles. Also, it is worth mentioning that a number of sampled actions N in BCQ has similar properties to choosing a particular expectile τ in our approach.

Note that our algorithm for optimal value approximation does not require an explicit policy, in contrast to other algorithms for offline reinforcement learning for continuous action spaces (Fujimoto et al., 2019; Fujimoto & Gu, 2021; Wu et al., 2019; Kostrikov et al., 2021; Kumar et al., 2019; 2020). Thus, we do not need to alternate between actor and critic updates, though with continuous actions, we must still extract an actor at the end once the critic converges.

E Different Estimators of $V(s)$

We also evaluate different ways to estimate the value function $V(s)$ (Table 6). We compare $V(s)$ learned with expectile regression as in IQL with $V(s)$ estimated with several samples from the learned policy as in ABM (Siegel et al., 2020). In particular, we use $N = 20$ to estimate the value function.

Table 6: Different estimators of $V(s)$

	IQL	$V(s) = \sum_{i=1}^{N} Q(s, a_i)/N$
hopper-medium-v2	66.2±5.7	69.5±3.9
hopper-medium-expert-v2	91.5±14.3	75.8±37.8
hopper-medium-replay-v2	94.7±8.6	64.7±22.6
halfcheetah-medium-v2	47.4±0.2	47.2±0.2
halfcheetah-medium-expert-v2	86.7±5.3	93.0±0.9
halfcheetah-medium-replay-v2	44.2±1.2	45.1±0.3
walker2d-medium-v2	78.3±8.7	72.0±24.6
walker2d-medium-expert-v2	109.6±1.0	110.7±0.4
walker2d-medium-replay-v2	73.8±7.1	83.3±3.0
locomotion total	692.4±52.1	661.4±93.7
antmaze-umaze-v0	87.5±2.6	96.4±1.8
antmaze-medium-play-v0	71.2±7.3	0.0±0.0
antmaze-large-play-v0	39.6±5.8	0.0±0.0
antmaze-umaze-diverse-v0	62.2±13.8	57.5±6.3
antmaze-medium-diverse-v0	70.0±10.9	0.0±0.0
antmaze-large-diverse-v0	47.5±9.5	0.0±0.0
antmaze total	378.0±49.9	153.9±8.1