A Related Work

Offline RL Many recent methods for offline RL (Fujimoto et al., 2018a; Kumar et al., 2019; Wu et al., 2019; Jaques et al., 2019), where no interactive data collection is allowed during training, mostly rely on constraining the learned policy to stay close to the data collection distribution. Fujimoto et al. (2018a) clip the maximum deviation from actions sampled from a base behavior policy, while Kumar et al. (2019); Wu et al. (2019); Jaques et al. (2019) incorporate additional distributional penalties (such as KL divergence or MMD) for regularizing learned policies to remain close to the base policy. Our work is an instance of this family of approaches for offline RL; however, arguably our method is simpler as it does not involve learning an additional proposal-modifying policy Fujimoto et al. (2018a), or modifying reward functions (Kumar et al., 2019; Jaques et al., 2019).

Finding Maximizing Actions Naïvely, EMaQ can also be seen as just performing approximate search for $\max_a Q(s, a)$ in standard Q-learning operator, which has been studied in various prior works for Q-learning in large scale spaces (e.g. continuous). NAF (Gu et al., 2016b) and ICNN (Amos et al., 2017) directly constrain the function family of Q-functions such that the optimization can be closed-form or tractable. QT-OPT (Kalashnikov et al., 2018b) makes use of two iterations of the Cross-Entropy Method (Rubinstein & Kroese, 2013), while CAQL (Ryu et al., 2019) uses Mixed-Integer Programming to find the exact maximizing action while also introducing faster approximate alternatives. In (Van de Wiele et al., 2020) – the most similar approach to our proposed method EMaQ – throughout training a mixture of uniform and learned proposal distributions are used to sample actions. The sampled actions are then evaluated under the learned Q functions, and the top K maximizing actions are distilled back into the proposal distribution. In contrast to our work, these works assume these are approximate maximization procedures and do not provide extensive analysis for the resulting TD operators. Our theoretical analysis on the family of TD operators described by EMaQ can therefore provide new perspectives on some of these highly successful Q-learning algorithms (Kalashnikov et al., 2018a; Van de Wiele et al., 2020) – particularly on how the proposal distribution affects convergence.

Modified Backup Operators Many prior works study modifications to standard backup operators to achieve different convergence properties for action-value functions or their induced optimal policies. Ψ-learning (Rawlik et al., 2013) proposes a modified operator that corresponds to policy iterations with KL-constrained updates (Kakade, 2002; Peters et al., 2010; Schulman et al., 2015) where the action-value function converges to negative infinity for all sub-optimal actions. Similarly but distinctly, Fox et al. (2015); Jaques et al. (2017); Haarnoja et al. (2018); Nachum et al. (2017) study smoothed TD operators for a modified entropy- or KL-regularized RL objective. Bellemare et al. (2016) derives a family of consistent Bellman operators and shows that they lead to increasing action gaps (Farahmand, 2011) for more stable learning. However, most of these operators have not been studied in offline learning. Our work adds a novel family operators to this rich literature of operators for RL, and provides strong empirical validation on how simple modifications of operators can translate to effective offline RL with function approximations.

B Proofs

All the provided proofs operate under the setting where $\mu(a|s)$ has full support over the action space. When this assumption is not satisfied, the provided proofs can be transferred by assuming we are operating in a new MDP M_μ as defined below.

Given the MDP $M = \langle \mathcal{S}, \mathcal{A}, r, \mathcal{P}, \gamma \rangle$ and $\mu(a|s)$, let us define the new MDP $M_\mu = \langle \mathcal{S}_\mu, \mathcal{A}_\mu, r, \mathcal{P}, \gamma \rangle$, where $\mathcal{S}_\mu$ denotes the set of reachable states by μ, and $\mathcal{A}_\mu$ is $\mathcal{A}$ restricted to the support of $\mu(a|s)$ in each state in $\mathcal{S}_\mu$.

B.1 Contraction Mapping

Theorem 3.1. *In the tabular setting, for any $N \in \mathbb{N}$, $\mathcal{T}_\mu^N$ is a contraction operator in the $\mathcal{L}_\infty$ norm. Hence, with repeated applications of the $\mathcal{T}_\mu^N$, any initial Q function converges to a unique fixed point.*

Proof. Let Q_1 and Q_2 be two arbitrary Q functions.

$$\left\|\mathcal{T}_\mu^N Q_1 - \mathcal{T}_\mu^N Q_2\right\|_\infty = \tag{11}$$

$$\max_{s,a}\left|\left(r(s,a)+\gamma\cdot\mathbb{E}_{s'}\mathbb{E}_{\{a_i\}^N}[\max_{\{a_i\}^N} Q_1(s',a')]\right)-\left(r(s,a)+\gamma\cdot\mathbb{E}_{s'}\mathbb{E}_{\{a_i\}^N}[\max_{\{a_i\}^N} Q_2(s',a')]\right)\right| = \tag{12}$$

$$\gamma\cdot\max_{s,a}\left|\mathbb{E}_{s'}\mathbb{E}_{\{a_i\}^N}\left[\max_{\{a_i\}^N} Q_1(s',a') - \max_{\{a_i\}^N} Q_2(s',a')\right]\right| \leq \tag{13}$$

$$\gamma\cdot\max_{s,a}\mathbb{E}_{s'}\mathbb{E}_{\{a_i\}^N}\left|\max_{\{a_i\}^N} Q_1(s',a') - \max_{\{a_i\}^N} Q_2(s',a')\right| \leq \tag{14}$$

$$\gamma\cdot\max_{s,a}\mathbb{E}_{s'}\mathbb{E}_{\{a_i\}^N}\left\|Q_1-Q_2\right\|_\infty = \tag{15}$$

$$\gamma\cdot\left\|Q_1-Q_2\right\|_\infty \tag{16}$$

where line 15 is due to the following: Let $\hat{a} = \arg\max_{\{a_i\}^N} Q_1(s',a_i)$,

$$\max_{\{a_i\}^N} Q_1(s',a') - \max_{\{a_i\}^N} Q_2(s',a') = Q_1(s',\hat{a}) - \max_{\{a_i\}^N} Q_2(s',a') \tag{17}$$

$$\leq Q_1(s',\hat{a}) - Q_2(s',\hat{a}) \tag{18}$$

$$\leq \left\|Q_1-Q_2\right\|_\infty \tag{19}$$

□

B.2 Limiting Behavior

Theorem 3.3. *Let π_μ^* denote the optimal policy from the class of policies whose actions are restricted to lie within the support of the policy $\mu(a|s)$. Let Q_μ^* denote the Q-value function corresponding to π_μ^*. Furthermore, let Q_μ denote the Q-value function of the policy $\mu(a|s)$. Let $\mu^*(s) := \int_{\textit{Support}(\pi_\mu^*(a|s))}\mu(a|s)$ denote the probability of optimal actions under $\mu(a|s)$. Under the assumption that $\inf_s \mu^*(s) > 0$ and $r(s,a)$, we have that,*

$$Q_\mu^1 = Q_\mu \qquad \textit{and} \qquad \lim_{N\to\infty} Q_\mu^N = Q_\mu^*$$

Let $\mu^*(s) := \int_{\text{Support}(\pi_\mu^*(a|s))}\mu(a|s)$ denote the probability of optimal actions under $\mu(a|s)$. To show $\lim_{N\to\infty} Q_\mu^N = Q_\mu^*$, we also require the additional assumption that $\inf_s \mu^*(s) > 0$.

Proof. Given that,

$$\mathcal{T}_\mu^1 Q(s,a) := r(s,a) + \gamma\cdot\mathbb{E}_{s'}\mathbb{E}_{\{a_i\}^N\sim\mu(\cdot|s')}\left[Q(s',a')\right] \tag{20}$$

the unique fixed-point of $\mathcal{T}_\mu^1$ is the Q-value function of the policy $\mu(a|s)$. Hence $Q_\mu^1 = Q_\mu$.

The second part of this theorem will be proven as a Corollary to Theorem 3.5 □

B.3 Increasingly Better Policies

Theorem 3.4. *For all $N, M \in \mathbb{N}$, where $N > M$, we have that $\forall s \in \mathcal{S}, \forall a \in \text{Support}(\mu(\cdot|s))$, $Q_\mu^N(s,a) \geq Q_\mu^M(s,a)$. Hence, $\pi_\mu^N(a|s)$ is at least as good of a policy as $\pi_\mu^M(a|s)$.*

Proof. It is sufficient to show that $\forall s,a, Q_\mu^{N+1}(s,a) \geq Q_\mu^N(s,a)$. We will do so by induction. Let Q^i denote the resulting function after applying $\mathcal{T}_\mu^{N+1}$, i times, starting from Q_μ^N.

Base Case

By definition $Q^0 := Q_\mu^N$. Let $s \in \mathcal{S}$, $a \in \mathcal{A}$.

$$Q^1(s,a) = \mathcal{T}_\mu^{N+1} Q^0(s,a) \tag{21}$$
$$= r(s,a) + \gamma \cdot \mathbb{E}_{s'} \mathbb{E}_{\{a_i\}^{N+1} \sim \mu(a'|s')} [\max_{\{a_i\}^{N+1}} Q^0(s',a')] \tag{22}$$
$$\geq r(s,a) + \gamma \cdot \mathbb{E}_{s'} \mathbb{E}_{\{a_i\}^{N} \sim \mu(a'|s')} [\max_{\{a_i\}^{N}} Q^0(s',a')] \tag{23}$$
$$= r(s,a) + \gamma \cdot \mathbb{E}_{s'} \mathbb{E}_{\{a_i\}^{N} \sim \mu(a'|s')} [\max_{\{a_i\}^{N}} Q_\mu^N(s',a')] \tag{24}$$
$$= Q_\mu^N(s,a) \tag{25}$$
$$= Q^0(s,a) \tag{26}$$

Induction Step

Assume $\forall s, a, Q^i(s,a) \geq Q^{i-1}(s,a)$.

$$Q^{i+1}(s,a) - Q^i(s,a) = \mathcal{T}_\mu^{N+1} Q^i(s,a) - \mathcal{T}_\mu^{N+1} Q^{i-1}(s,a) \tag{27}$$
$$= \gamma \cdot \mathbb{E}_{s'} \mathbb{E}_{\{a_i\}^{N+1} \sim \mu(a'|s')} [\max_{\{a_i\}^{N+1}} Q^i(s',a') - \max_{\{a_i\}^{N+1}} Q^{i-1}(s',a')] \tag{28}$$
$$\geq 0 \tag{29}$$

Hence, by induction we have to $\forall i, j, i > j \implies \forall s, a, Q^i(s,a) \geq Q^j(s,a)$. Since $Q^0 = Q_\mu^N$ and $\lim_{i \to \infty} Q^i = Q_\mu^{N+1}$, we have than $\forall s, a, Q_\mu^{N+1}(s,a) \geq Q_\mu^N(s,a)$. Thus π_μ^{N+1} is a better policy than π_μ^N, and by a simple induction argument, π_μ^N is a better policy than π_μ^M when $N > M$.

□

B.4 BOUNDS

Theorem 3.5. *For $s \in \mathcal{S}$ let,*

$$\Delta(s) = \max_{a \in \text{Support}(\mu(\cdot|s))} Q_\mu^*(s,a) - \mathbb{E}_{\{a_i\}^N \sim \mu(\cdot|s)} [\max_{b \in \{a_i\}^N} Q_\mu^*(s,b)]$$

The suboptimality of Q_μ^N can be upperbounded as follows,

$$\left\| Q_\mu^N - Q_\mu^* \right\|_\infty \leq \frac{\gamma}{1-\gamma} \max_{s,a} \mathbb{E}_{s'} \Big[\Delta(s') \Big] \leq \frac{\gamma}{1-\gamma} \max_s \Delta(s) \tag{30}$$

The same also holds when Q_μ^* is replaced with Q_μ^N in the definition of Δ.

Proof. The two versions where $\Delta(s)$ is defined in terms of Q_μ^N and Q_μ^* have very similar proofs.

Version with Q_μ^N

Let $\mathcal{T}^{QL}$ denote the backup operation in Q-Learning. Let $(\mathcal{T}^{QL})^m = \underbrace{\mathcal{T}^{QL} \circ \mathcal{T}^{QL} \circ ... \circ \mathcal{T}^{QL}}_{m \text{ times}}$. We know the following statements to be true:

$$Q_\mu^N = \mathcal{T}_\mu^N Q_\mu^N = r(s,a) + \gamma \cdot \mathbb{E}_{s'} \mathbb{E}_{\{a_i\}^N \sim \mu(a'|s')} [\max_{\{a_i\}^N} Q_\mu^N(s',a')] \tag{31}$$
$$\mathcal{T}^{QL} Q_\mu^N = r(s,a) + \gamma \cdot \mathbb{E}_{s'} \max_{a'} Q_\mu^N(s',a') \tag{32}$$
$$\lim_{m \to \infty} (\mathcal{T}^{QL})^m Q_\mu^N = Q^* \tag{33}$$
$$\left\| (\mathcal{T}^{QL})^{m+2} Q_\mu^N - (\mathcal{T}^{QL})^{m+1} Q_\mu^N \right\|_\infty \leq \gamma \cdot \left\| (\mathcal{T}^{QL})^{m+1} Q_\mu^N - (\mathcal{T}^{QL})^m Q_\mu^N \right\|_\infty \tag{34}$$
$$\left\| (\mathcal{T}^{QL})^{m+1} Q_\mu^N - (\mathcal{T}^{QL})^m Q_\mu^N \right\|_\infty \leq \gamma^m \cdot \left\| \mathcal{T}^{QL} Q_\mu^N - Q_\mu^N \right\|_\infty \tag{35}$$

Putting these together we have that,

$$\left\|Q_\mu^N - Q^*\right\|_\infty \leq \sum_{m=0}^{\infty} \left\|(\mathcal{T}^{QL})^{m+1} Q_\mu^N - (\mathcal{T}^{QL})^m Q_\mu^N\right\|_\infty \tag{36}$$

$$\leq \sum_{m=0}^{\infty} \gamma^m \cdot \left\|\mathcal{T}^{QL} Q_\mu^N - Q_\mu^N\right\|_\infty \tag{37}$$

$$= \frac{1}{1-\gamma} \left\|\mathcal{T}^{QL} Q_\mu^N - Q_\mu^N\right\|_\infty \tag{38}$$

$$= \frac{1}{1-\gamma} \max_{s,a} \Bigg| \Big(r(s,a) + \gamma \cdot \mathbb{E}_{s'} \max_{a'} Q_\mu^N(s',a') \Big) \tag{39}$$

$$- \Big(r(s,a) + \gamma \cdot \mathbb{E}_{s'} \mathbb{E}_{\{a_i\}^N \sim \mu(a'|s')} [\max_{\{a_i\}^N} Q_\mu^N(s',a')] \Big) \Bigg| \tag{40}$$

$$= \frac{\gamma}{1-\gamma} \max_{s,a} \left| \mathbb{E}_{s'} \Big[\max_{a'} Q_\mu^N(s',a') - \mathbb{E}_{\{a_i\}^N \sim \mu(a'|s')} [\max_{\{a_i\}^N} Q_\mu^N(s',a')] \Big] \right| \tag{41}$$

$$\leq \frac{\gamma}{1-\gamma} \max_{s'} \left| \max_{a'} Q_\mu^N(s',a') - \mathbb{E}_{\{a_i\}^N \sim \mu(a'|s')} [\max_{\{a_i\}^N} Q_\mu^N(s',a')] \right| \tag{42}$$

Version with Q_μ^*

Very similarly we have,

$$\left\|Q_\mu^N - Q^*\right\|_\infty \leq \sum_{m=0}^{\infty} \left\|(\mathcal{T}_\mu^N)^{m+1} Q^* - (\mathcal{T}_\mu^N)^m Q^*\right\|_\infty \tag{43}$$

$$\leq \sum_{m=0}^{\infty} \gamma^m \cdot \left\|\mathcal{T}_\mu^N Q^* - Q^*\right\|_\infty \tag{44}$$

$$= \frac{1}{1-\gamma} \left\|Q^* - \mathcal{T}_\mu^N Q^*\right\|_\infty \tag{45}$$

$$= \frac{1}{1-\gamma} \max_{s,a} \Bigg| \Big(r(s,a) + \gamma \cdot \mathbb{E}_{s'} \max_{a'} Q^*(s',a') \Big) \tag{46}$$

$$- \Big(r(s,a) + \gamma \cdot \mathbb{E}_{s'} \mathbb{E}_{\{a_i\}^N \sim \mu(a'|s')} [\max_{\{a_i\}^N} Q^*(s',a')] \Big) \Bigg| \tag{47}$$

$$= \frac{\gamma}{1-\gamma} \max_{s,a} \left| \mathbb{E}_{s'} \Big[\max_{a'} Q^*(s',a') - \mathbb{E}_{\{a_i\}^N \sim \mu(a'|s')} [\max_{\{a_i\}^N} Q^*(s',a')] \Big] \right| \tag{48}$$

$$\leq \frac{\gamma}{1-\gamma} \max_{s'} \left| \max_{a'} Q^*(s',a') - \mathbb{E}_{\{a_i\}^N \sim \mu(a'|s')} [\max_{\{a_i\}^N} Q^*(s',a')] \right| \tag{49}$$

□

Corollary B.1. *Let V_μ, Q_μ, A_μ denote the value, Q, and advantage functions of μ respectively. When $N = 1$ we have that,*

$$\|Q_\mu - Q^*\|_\infty \leq \frac{\gamma}{1-\gamma} \max_{s'} \left| \max_{a'} Q_\mu(s',a') - \mathbb{E}_{a' \sim \mu(a'|s')} [Q_\mu(s',a')] \right| \tag{50}$$

$$= \frac{\gamma}{1-\gamma} \max_{s'} \left| \max_{a'} Q_\mu(s',a') - V_\mu(s') \right| \tag{51}$$

$$= \frac{\gamma}{1-\gamma} \max_{s',a'} A_\mu(s',a') \tag{52}$$

It is interesting how the sub-optimality can be upper-bounded in terms of a policy's own advantage function.

Corollary B.2. ***(Proof for second part of Theorem 3.3)***

Proof. We want to show $\lim_{N\to\infty} Q_\mu^N = Q^*$. More exactly, what we seek to show is the following,

$$\lim_{N\to\infty} \left\|Q_\mu^N - Q^*\right\|_\infty = 0 \tag{53}$$

or,

$$\forall \epsilon > 0, \exists N, \text{ s.t. } \forall M \geq N, \left\|Q_\mu^N - Q^*\right\|_\infty < \epsilon \tag{54}$$

Let $\epsilon > 0$. Recall,

$$\Delta(s) = \max_{a \in \text{Support}(\mu(\cdot|s))} Q_\mu^*(s,a) - \mathbb{E}_{\{a_i\}^N \sim \mu(\cdot|s)}\big[\max_{b \in \{a_i\}^N} Q_\mu^*(s,b)\big] \tag{55}$$

Let $\inf_s \mu^*(s) = p > 0$. Let the lower and upper bounds of rewards be ℓ and L, and let $\alpha = \frac{1}{1-\gamma}\ell$ and $\beta = \frac{1}{1-\gamma}L$. We have that,

$$\mathbb{E}_{\{a_i\}^N \sim \mu(\cdot|s)}\big[\max_{b \in \{a_i\}^N} Q_\mu^*(s,b)\big] \geq (1-p)^N \cdot \alpha + (1-(1-p)^N) \cdot \max_{a \in \text{Support}(\mu(\cdot|s))} Q_\mu^*(s,a) \tag{56}$$

Hence $\forall s$,

$$\Delta(s) \leq (1-p)^N \cdot \max_{a \in \text{Support}(\mu(\cdot|s))} Q_\mu^*(s,a) - (1-p)^N \cdot \alpha \tag{57}$$

$$= (1-p)^N \cdot \Big(\max_{a \in \text{Support}(\mu(\cdot|s))} Q_\mu^*(s,a) - \alpha\Big) \tag{58}$$

$$\leq (1-p)^N \cdot \Big(\beta - \alpha\Big) \tag{59}$$

Thus, for large enough N we have that,

$$\left\|Q_\mu^N - Q_\mu^*\right\|_\infty \leq \frac{\gamma}{1-\gamma} \max_s \Delta(s) < \epsilon \tag{60}$$

concluding the proof. □

C AUTOREGRESSIVE GENERATIVE MODEL

The architecture for our autoregressive generative model is inspired by the works of (Metz et al., 2017; Van de Wiele et al., 2020; Germain et al., 2015). Given a state-action pair from the dataset (s, a), first an MLP produces a d-dimensional embedding for s, which we will denote by h. Below, we use the notation a_i to denote the i^{th} index of a, and $a_{[:i]}$ to represent a slice from first up to and **not including** the i^{th} index, where indexing begins at 0. We use a discretization in each action dimension. Thus, we discretize the range of each action dimension into N uniformly sized bins, and represent a by the labels of the bins. Let ℓ_i denote the label of the i^{th} action index.

Training We use separate MLPs per action dimension. Each MLP takes in the d-dimensional state embedding and ground-truth actions before that index, and outputs N logits for the choice over bins. The probability of a given index's label is given by,

$$p(\ell_i | s, a[:i]) = \text{SoftMax}\Big(\text{MLP}_i(d, a[:i])\Big)[\ell_i] \tag{61}$$

We use standard maximum-likelihood training (i.e. cross-entropy loss).

Sampling Given a state s, to sample an action we again embed the state, and sample the action indices one-by-one.

$$p(\ell_0 | s) = \text{SoftMax}\Big(\text{MLP}_i(d)\Big)[\ell_0] \tag{62}$$

$$\ell_0 \sim p(\ell_0|s), a_0 \sim \text{Uniform(Bin corresponding to } \ell_0) \tag{63}$$

$$p(\ell_i | s) = \text{SoftMax}\Big(\text{MLP}_i(d, a[:i])\Big)[\ell_i] \tag{64}$$

$$\ell_i \sim p(\ell_i|s, a[:i]), a_i \sim \text{Uniform(Bin corresponding to } \ell_i) \tag{65}$$

Algorithm 2: Full EMaQ Training Algorithm

Offline dataset $\mathcal{D}$, Pretrain $\mu(a|s)$ on $\mathcal{D}$
Initialize K Q functions with parameters θ_i, and K target Q functions with parameters θ_i^{target}
`Ensemble` parameter λ, Exponential moving average parameter α

Function `Ensemble(`*values*`)`**:**
  return $\lambda \cdot \min(values) + (1-\lambda) \cdot \max(values)$
Function $y_{\text{target}}(s, a, s', r, t)$**:**
  $\{a'_i\}^N \sim \mu(a'|s')$
  $Qvalues \leftarrow [\ \ ]$
  for $k \leftarrow 1$ **to** N **do**
    `/* Estimate the value of action` a'_k `*/`
    $Qvalues$.`append`$\Big($`Ensemble`$\big([Q_i^{target}(s', a'_k) \text{ for all } i]\big)\Big)$
  return $r + (1-t) \cdot \gamma \max(Qvalues)$

while *not converged* **do**
  Sample a batch $\{(s_m, a_m, s'_m, r_m, t_m)\}^M \sim \mathcal{D}$
  for $i = 1, ..., K$ **do**
    $\mathcal{L}(\theta_i) = \sum_m \Big(Q_i(s_m, a_m) - y_{\text{target}}(s_m, a_m, s'_m, r_m, t_m)\Big)^2$
    $\theta_i \leftarrow \theta_i -$ `AdamUpdate`$\Big(\mathcal{L}(\theta_i), \theta_i\Big)$
    $\theta_i^{\text{target}} \leftarrow \alpha \cdot \theta_i^{\text{target}} + (1-\alpha) \cdot \theta_i$

D ALGORITHM BOX

E INCONCLUSIVE EXPERIMENTS

E.1 UPDATING THE PROPOSAL DISTRIBUTION

Akin to the work of (Van de Wiele et al., 2020), we considered maintaining a second proposal distribution $\tilde{\mu}$ that is updated to distill $\arg\max_{\{a_i\}^N} Q(s, a)$, and sampling from the mixture of μ and $\tilde{\mu}$. In our experiments however, we did not observe noticeabel gains. This may potentially be due to the relative simplicity of the Mujoco benchmark domains, and may become more important in more challenging domains with more uniformly distributed $\mu(a|s)$.

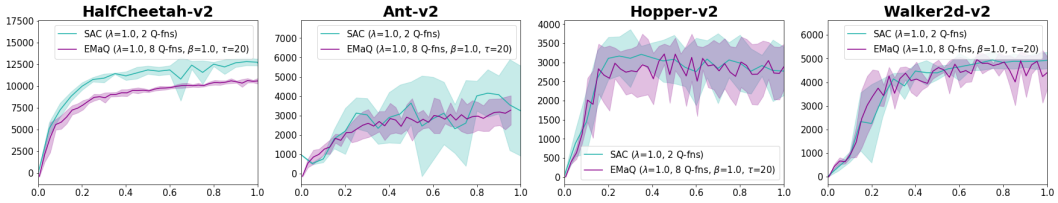
F LAUNDRY LIST

- Autoregressive models are slow to generate samples from and EMaQ needs to take many samples, so it was slower to train than the alternative methods. However, this may be addressed by better generative models and engineering effort.

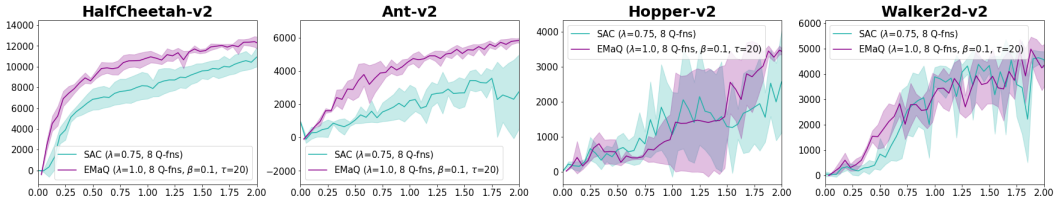
G ONLINE RL

EMaQ is also applicable to online RL setting. Combining strong offline RL methods with good exploration policies has the potential for producing highly sample-efficient online RL algorithms. Concretely, we refer to online RL as the setting where iteratively, a batch of M environment steps with an exploration policy are interleaved with M RL updates (Levine et al., 2020; Matsushima et al., 2020).

EMaQ is designed to remain within the support of the provided training distribution. This however, is problematic for online RL which requires good exploration interleaved with RL updates. To this end, first, we modify our autoregressive proposal distribution $\mu(a|s)$ by dividing the logits of all



(a) SAC vs. EMaQ, Trajectory Batch Size 1: For easier visual interpretration we plot a single hyperparameter setting of EMaQ that tended to perform well across the 4 domains considered. The hyperparameters considered were $N = 200$, $\lambda = 1.0$, $\beta = 1.0$, $\tau \in \{1, 5, 10, 20\}$. SAC performed worse when using 8 Q-functions as in EMaQ. x-axis unit is 1 million environment steps.



(b) SAC vs. EMaQ, Trajectory Batch Size 50K: For easier visual interpretration we plot a single hyperparameter setting of EMaQ that tended to perform well across the 4 domains considered. The hyperparameters considered were $N = 200$, $\lambda \in \{0.75, 1.0\}$, $\beta \in \{0.1, 1.0\}$, $\tau \in \{1, 5, 10, 20\}$. x-axis unit is 1 million environment steps.

Figure 3: Online RL results under different trajectory batch sizes.

softmaxes by $\tau > 1$. This has the effect of smoothing the $\mu(a|s)$ distribution, and increasing the probability of sampling actions from the low-density regions and the boundaries of the support. Given this online proposal distribution, a criteria is required by which to choose amongst sampled actions. While there exists a rich literature on how to design effective RL exploration policies (Weng, 2020), in this work we used a simple UCB-style exploration criterion (Chen et al., 2017) as follows:

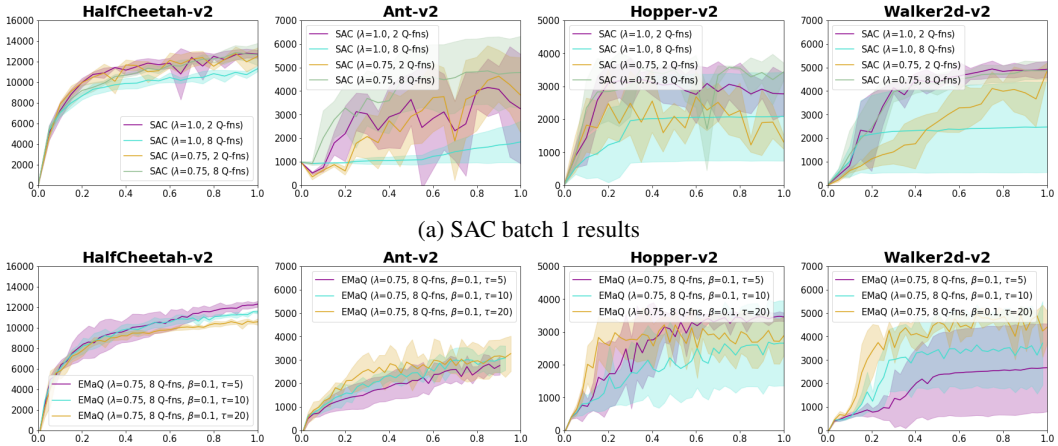
$$Q^{\text{explore}}(s, a) = \text{mean}\Big(\{Q_i(s, a)\}_K\Big) + \beta \cdot \text{std}\Big(\{Q_i(s, a)\}_K\Big) \tag{66}$$

Given N sampled actions from the modified proposal distribution, we take the action with highest Q^{explore}.

We compare the online variant of EMaQ with entropy-constrained Soft Actor Critic (SAC) with automatic tuning of the temperature parameter (Haarnoja et al., 2018). For EMaQ we swept the temperatures and used a fixed bin size of 40, 8 Q-function ensembles and $N = 200$. For fairness of comparisons, we also ran SAC with similar sweeps over different collection batch sizes and number of Q-function ensembles. In the fully online setting (trajectory batch size 1, Figure 3a), EMaQ is already competitive with SAC, and more excitingly, in the deployment-efficient setting[3] (trajectory batch size 50K, Figure 3b), EMaQ can outperform SAC[4]. Figures 4 and 5 present the results for all hyperparameter settings, for SAC and EMaQ, in the batch size 1 and batch size $50K$ settings respectively. **In the fully online setting, EMaQ is already competitive with SAC, and more excitingly, in the deployment-efficient setting, EMaQ can outperform SAC.**

[3]By deployment-efficient we mean that less number of different policies need to be executed in the environment, which may have substantial benefits for safety and otherwise constrained domains (Matsushima et al., 2020).

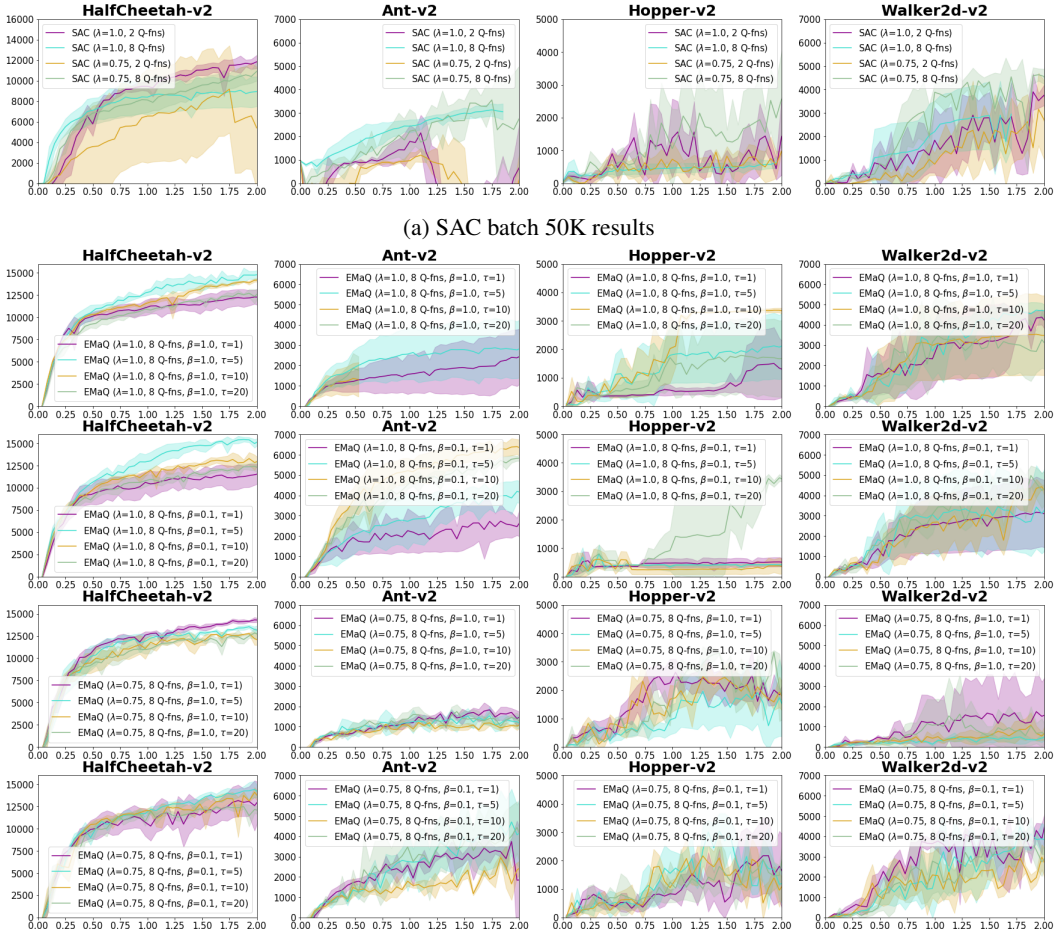
[4]It must be noted that the online variant of EMaQ has more hyperparameters to tune, and the relative performance is dependent on these hyperparameters, while SAC with ensembles has the one extra ensemble mixing parameter λ to tune.



(a) SAC batch 1 results

(b) EMaQ batch 1 results

Figure 4: All results for batch size 1



(a) SAC batch 50K results

(b) EMaQ batch 50K results

Figure 5: All results for batch size 50K

H OFFLINE RL EXPERIMENTAL DETAILS

For each environment and data setting, we train an autoregressive model – as described above – on the provided data with 2 random seeds. These generative models are then frozen, and used by the downstream algorithms (EMaQ, BEAR, and BCQ) as the base behavior policy ($\mu(a|s)$ in EMaQ)[5].

H.1 COMPARING OFFLINE RL METHODS

Following the bechmarking efforts of (Wu et al., 2019), the range of clipping factor considered for BCQ was $\Phi \in \{0.005, 0.015, 0.05, 0.15, 0.5\}$, and the range of target divergence value considered for BEAR was $\epsilon \in \{0.015, 0.05, 0.15, 0.5, 1.5\}$. For both methods, the larger the value of the hyperparameter is, the more the learned policy is allowed to deviate from the $\mu(a|s)$.

The rest of the hyperparameters use can be found in Table 1. The autoregressive models have the following architecture sizes (refer to Appendix C for description of the models used). The state embedding MLP consists of 2 hidden layers of dimension 750 with relu activations, followed by a linear embedding into a 750 dimensional state representation. The individual MLP for each action dimension consist of 3 hidden layers of dimension 256 with relu activations. Each action dimension is discretized into 40 equally sized bins.

Shared Hyperparameters	
λ	1.0
Batch Size	256
Num Updates	1e6
Num Q Functions	8
Q Architecture	MLP, 3 layers, 750 hid dim, relu
μ lr	5e-4
α	0.995
EMaQ Hyperparameters	
Q lr	1e-4
BEAR Hyperparameters	
π Architecture	MLP, 3 layers, 750 hid dim, relu
Q lr	1e-3
π lr	3e-5
BCQ Hyperparameters	
π Architecture	MLP, 3 layers, 750 hid dim, relu
Q lr	1e-4
π lr	5e-4

Table 1: Hyperparameters for Mujoco Experiments

H.2 EMAQ ABLATION EXPERIMENT

Hyperparameters are identical to those in Table 1, except batch size is 100 and number of updates is 500K.

H.3 DETAILS FOR TABLE ?? EXPERIMENTS

Generative Model The generative models used are almost identical to the description in Appendix C, with a slight modification that $\text{MLP}_i(d, a[:i])$ is replace with $\text{MLP}_i(d, \text{Lin}_i(a[:i]))$ where Lin_i is a linear transformation. This change was not necessary for good performance; it was as architectural detail that we experimented with and did not revert prior generating Table **??**. The model dimensions for each domain are shown in 2 in the following format (state embedding MLP hidden size, state embedding MLP number of layers, action MLP hidden size, action MLP number of layers, Ouput size of Lin_i, number of bins for action discretization). Increasing the number of discretization bins

[5]While in the original presentation of BCQ and BEAR the behvior policy is learned online, there is technically no reason for this to be the case, and in theory both methods should benefit from this pretraining

from 40 (value for standard Mujoco experiments) to 80 was the most important change. Output dimension of state-embedding MLP is the same as the hidden size.

Hyperparameters Table 2 shows the hyperparameters used for the experiments in Table **??**.

Shared Hyperparameters	
λ	1.0
Batch Size	128
Num Updates	1e6
Num Q Functions	16
Q Architecture	MLP, 4 layers, 256 hid dim, relu
α	0.995
μ lr	5e-4
Kitchen μ Arch Params	$(256, 4, 128, 1, 128, 80)$
Antmaze μ Arch Params	$(256, 4, 128, 1, 128, 80)$
Adroit μ Arch Params	$(256, 4, 128, 1, 128, 80)$
EMaQ Hyperparameters	
Q lr	1e-4
Kitchen N's Searched	$\{4, 8, 16, 32, 64\}$
Antmaze N's Searched	$\{50, 100, 150, 200\}$
Adroit N's Searched	$\{16, 32, 64, 128\}$
BEAR Hyperparameters	
π Architecture	MLP, 4 layers, 256 hid dim, relu
Q lr	1e-4
π lr	5e-4
BCQ Hyperparameters	
π Architecture	MLP, 4 layers, 256 hid dim, relu
Q lr	1e-4
π lr	5e-4

Table 2: Hyperparameters for Table **??** Experiments

Full Results Table Due to space limitations, we were unable to include the full table in the main text. Table 3 presents the full set of results.

I VAE RESULTS

I.1 IMPLEMENTATION

We also ran experiments with VAE parameterizations for $\mu(a|s)$. To be approximately matched in parameter count with our autoregressive models, the encoder and decoder both have 3 hidden layers of size 1024 with relu activations. The dimension of the latent space was twice the number of action dimensions. The decoder outputs a vector v which, and the decoder action distribution is defined to be $\mathcal{N}(\text{Tanh}(v), I)$. When sampling from the VAE, following prior work, samples from the VAE prior (spherical normal distribution) were clipped to the range $[-0.5, 0.5]$ and mean of the decoder distibution was used (i.e. the decoder distribution was not sampled from). The KL divergence loss term was weighted by 0.5. This VAE implementation was the one used in the benchmarking codebase of (Wu et al., 2019), so we did not modify it.

I.2 RESULTS

As can be seen in Figure 6, EMaQ has a harder time improving upon $\mu(a|s)$ when using the VAE architecture described above. However, as can be seen in Figure 7, BCQ and BEAR do show some variability as well when switching to the VAEs. Since as an algorithm EMaQ is much more reliant on $\mu(a|s)$, our hypothesis is that if it is true that the autoregressive models better captured the action distribution, letting EMaQ not make poor generalizations to out-of-distribution actions. Figures 8 and 9 show autoregressive and VAE results side-by-side for easier comparison.

Setting	BC	BCQ	BEAR	EMaQ	EMaQ N
kitchen-complete	27.2 ± 3.2	26.5 ± 4.8	—	**36.9 ± 3.7**	64
kitchen-partial	46.2 ± 2.8	69.3 ± 5.2	—	**74.6 ± 0.6**	8
kitchen-mixed	52.5 ± 3.8	65.5 ± 1.8	—	**70.8 ± 2.3**	8
antmaze-umaze	59.0 ± 5.5	25.5 ± 20.0	56.3 ± 28.8	**91.0 ± 4.6**	100
antmaze-umaze-diverse	58.8 ± 9.5	68.0 ± 19.0	57.5 ± 39.2	**94.0 ± 2.4**	50
antmaze-medium-play	**0.7 ± 1.0**	**3.5 ± 6.1**	**0.2 ± 0.4**	0.0 ± 0.0	—
antmaze-medium-diverse	**0.4 ± 0.8**	**0.5 ± 0.9**	**0.2 ± 0.4**	0.0 ± 0.0	—
antmaze-large-play	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	—
antmaze-large-diverse	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	—
door-cloned	**0.0 ± 0.0**	**0.2 ± 0.4**	**0.0 ± 0.0**	**0.2 ± 0.3**	64
hammer-cloned	**1.2 ± 0.6**	**1.3 ± 0.5**	**0.3 ± 0.0**	**1.0 ± 0.7**	64
pen-cloned	24.5 ± 10.2	**43.8 ± 6.4**	-3.1 ± 0.2	27.9 ± 3.7	128
relocate-cloned	**-0.2 ± 0.0**	**-0.2 ± 0.0**	**0.0 ± 0.0**	**-0.2 ± 0.2**	16

Table 3: Results on a series of other environments and data settings from the D4RL benchmark (Fu et al., 2020a). Results are normalized to the range $[0, 100]$, per the D4RL normalization scheme. For each method, for each environment and data setting the results of the best hyperparameter setting are reported. The last column indicates the best value of N in EMaQ amongst the considered hyperparameters (for the larger `antmaze` domains, we do not report this value since no value of N obtains nonzero returns). All the domains below the blue double-line are effectively unsolved by all methods. We have technical difficulties in evaluating BEAR on the kitchen domains. This manuscript will be updated upon obtaining these results. Additional details can be found in Appendix H.3.

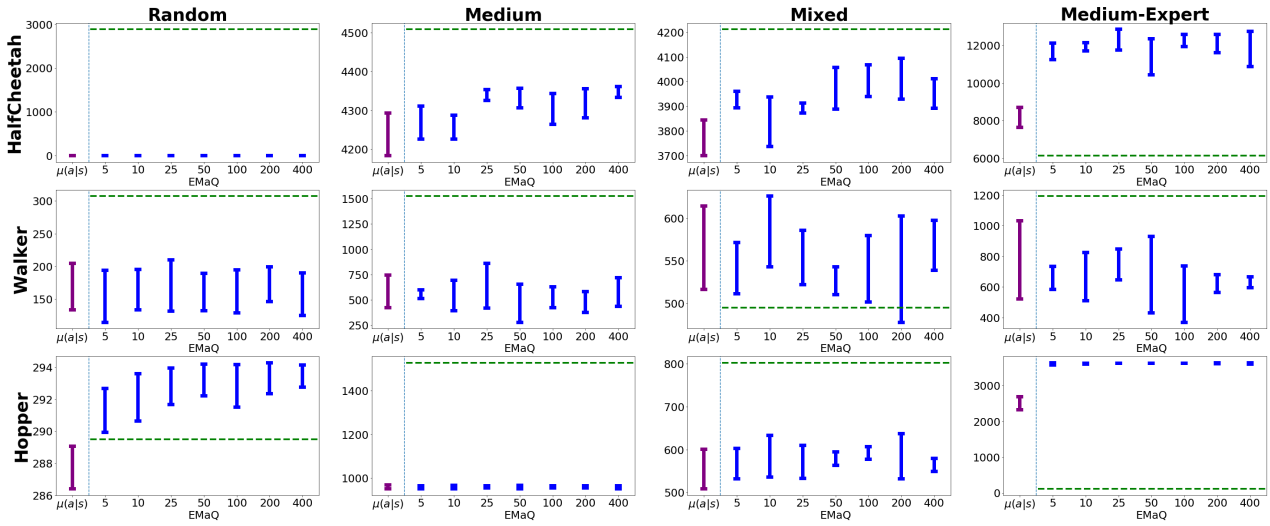


Figure 6: Results for evaluating EMaQ on D4RL (Fu et al., 2020b) benchmark domains when using the described VAE implementation, with $N \in \{5, 10, 25, 50, 100, 200, 400\}$. Values above $\mu(a|s)$ represent the result of evaluating the base behavior policies. Horizontal green lines represent the reported performance of BEAR in the D4RL benchmark (apples to apples comparisons in Figure 7).

J EMaQ Medium-Expert Setting Results

In HalfCheetah, increasing N significantly slows down the convergence rate of the training curves; while large Ns continue to improve, we were unable to train them long enough for convergence. In Walker, for EMaQ, BCQ, and most hyperparameter settings of BEAR, training curves have a prototypical shape of a hump, where performance improves up to a certain high value, and then continues to fall very low. In Hopper, for higher values of N in EMaQ we observed that increasing batch size from 100 to 256 largely resolved the poor performance, but for consistency we did not alter Figure 1 with these values.

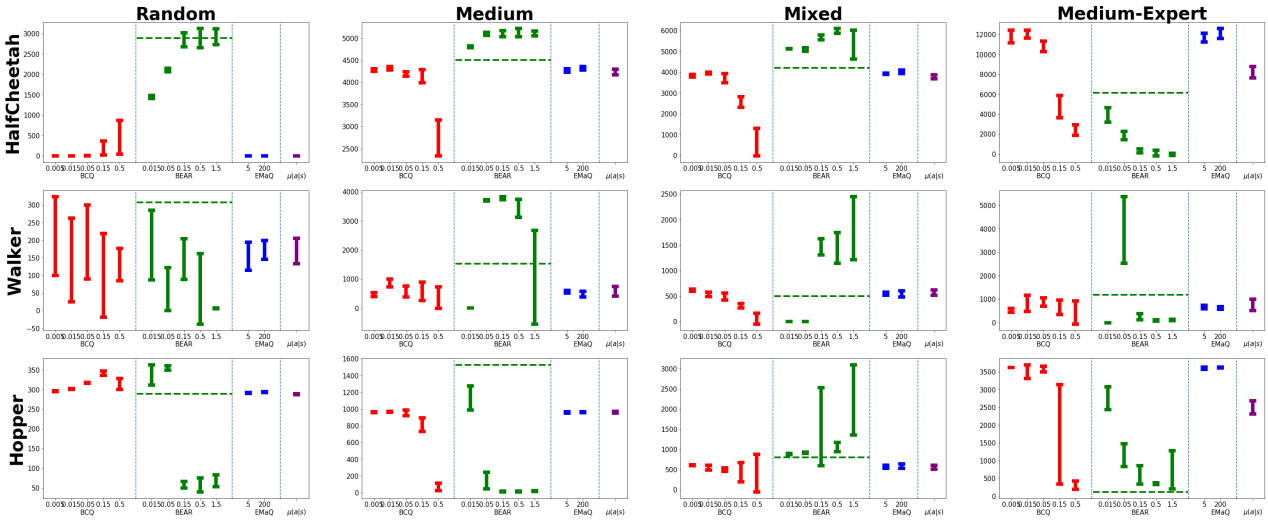


Figure 7: Comparison of EMaQ, BCQ, and BEAR on D4RL (Fu et al., 2020b) benchmark domains when using when using the described VAE implementation for $\mu(a|s)$. For both BCQ and BEAR, from left to right the allowed deviation from $\mu(a|s)$ increases. Horizontal green lines represent the reported performance of BEAR in the D4RL benchmark.

K LARGER PLOTS FOR VISIBILITY

Due to larger size of plots, each plot is shown on a separate page below. For ablation results, see Figure 10. For MuJoCo results, see Figure 11.

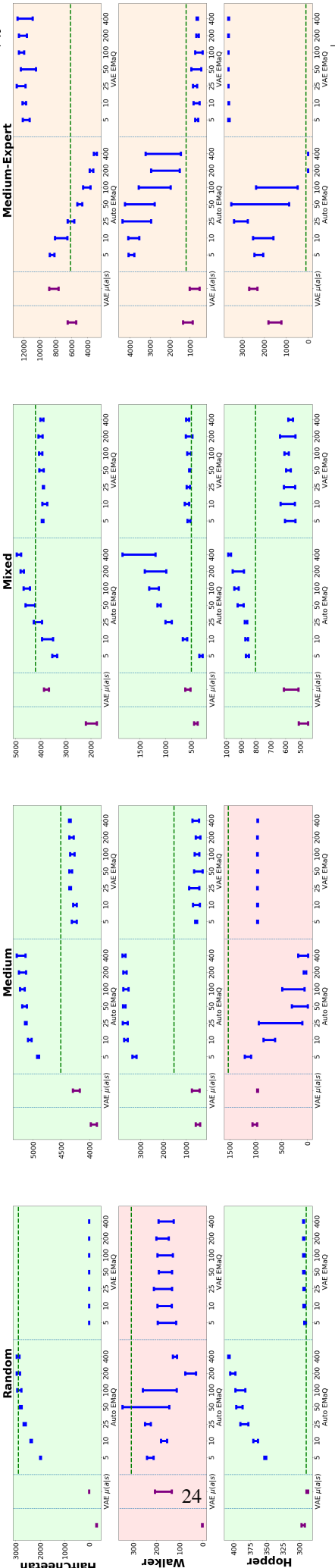


Figure 8: Results with both autoregressive and VAE models in one plot for easier comparison.

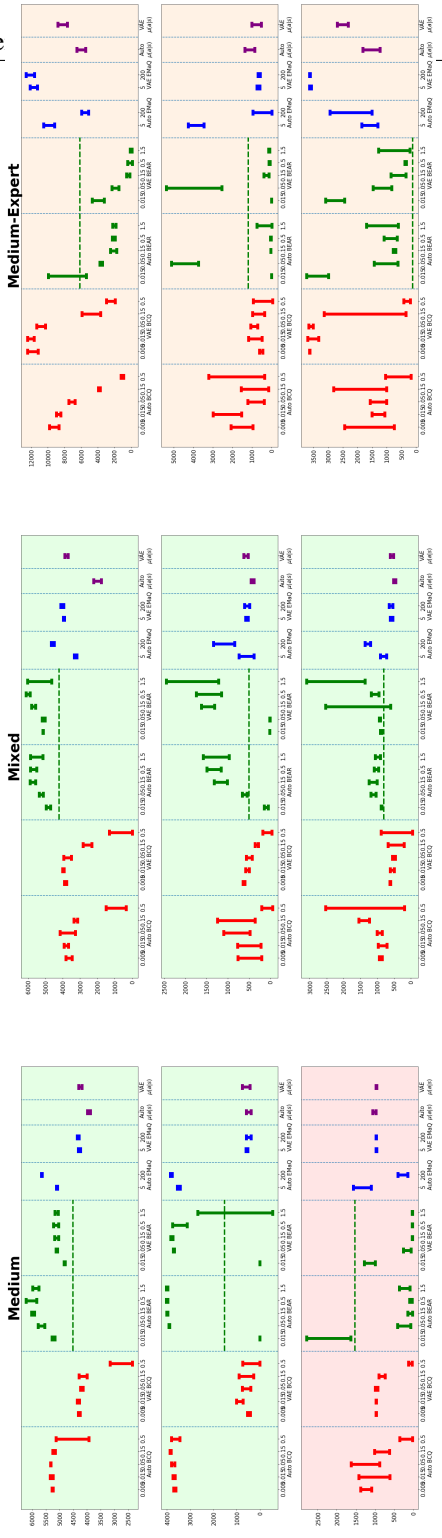


Figure 9: Results with both autoregressive and VAE models in one plot for easier comparison.

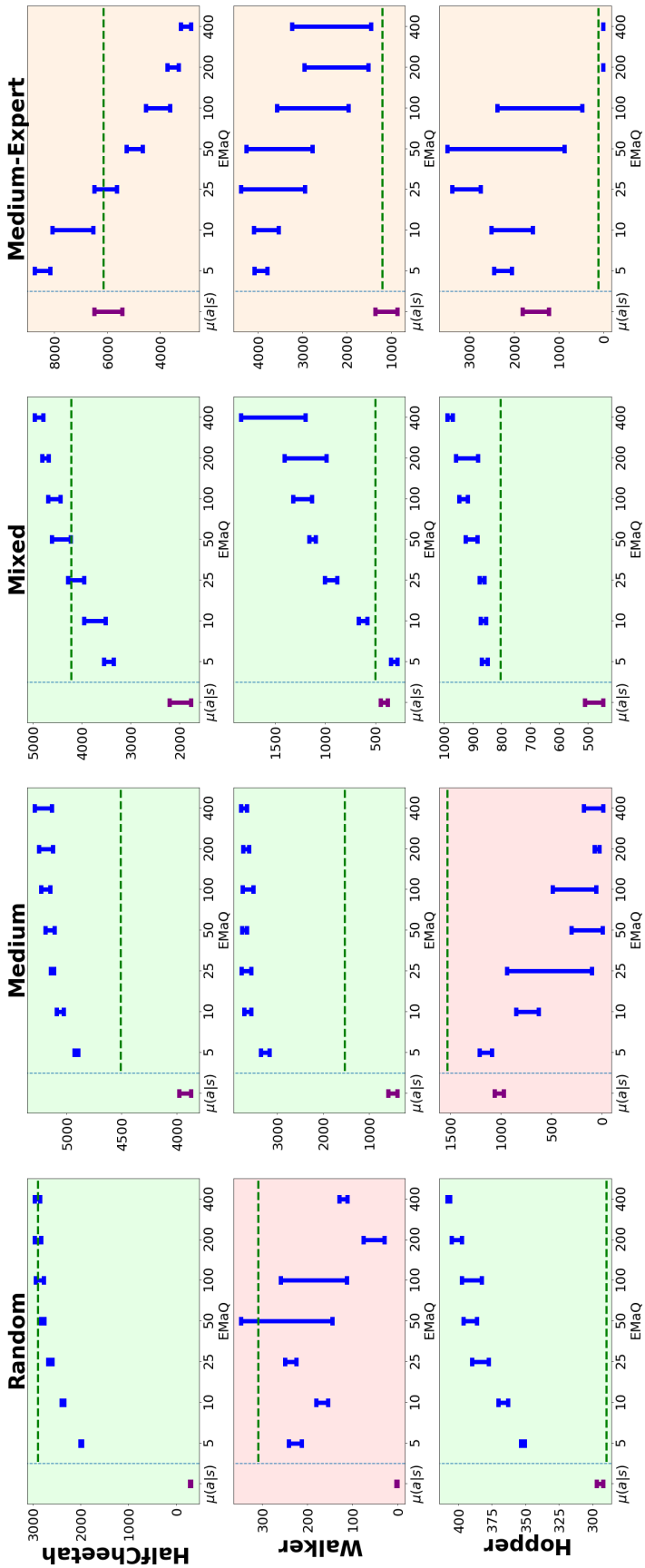


Figure 10: Results for evaluating EMaQ on D4RL (Fu et al., 2020b) benchmark domains, with $N \in \{5, 10, 25, 50, 100, 200, 400\}$. Values above $\mu(a|s)$ represent the result of evaluating the base behavior policies. Horizontal green lines represent the reported performance of BEAR in the D4RL benchmark (apples to apples comparisons in Figure 2). Refer to main text (Section 5.1) for description of color-coding.

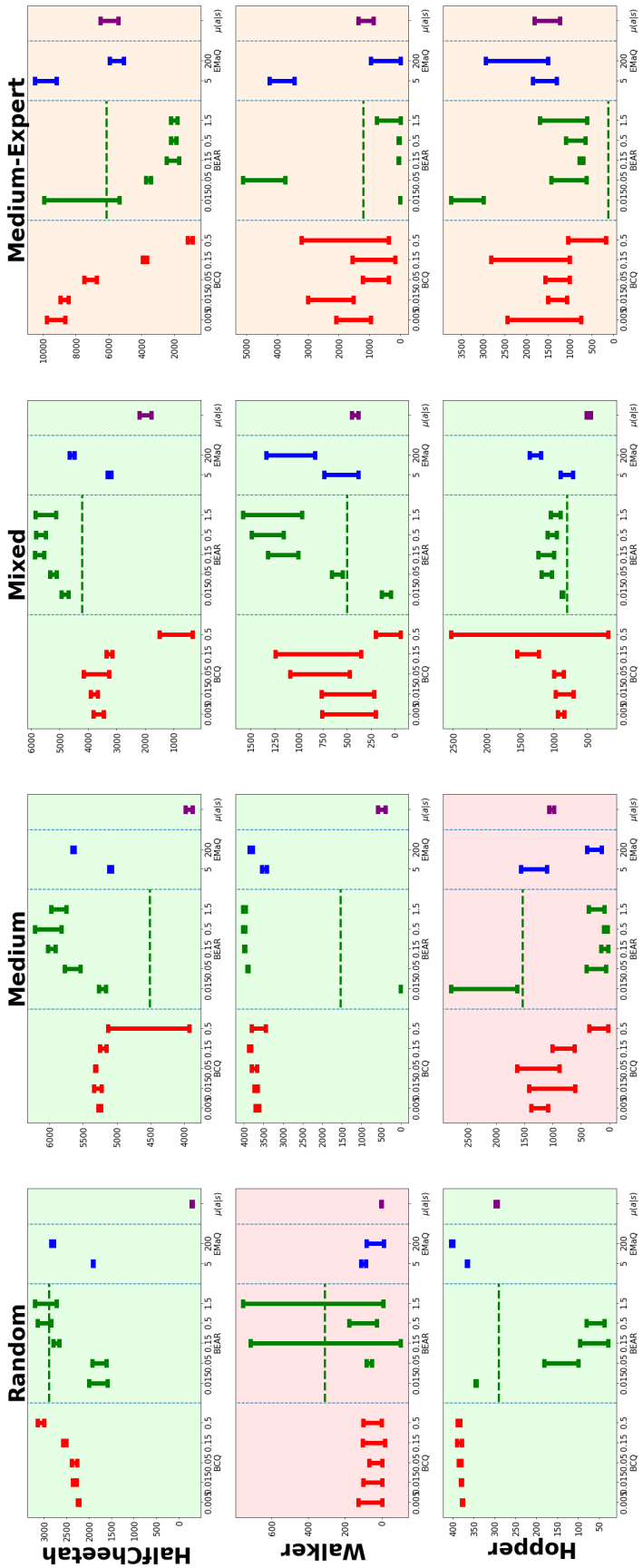


Figure 11: Comparison of EMaQ, BCQ, and BEAR on D4RL (Fu et al., 2020b) benchmark domains when using our proposed autoregressive $\mu(a|s)$. For both BCQ and BEAR, from left to right the allowed deviation from $\mu(a|s)$ increases. Horizontal green lines represent the reported performance of BEAR in the D4RL benchmark. Color-coding follows Figure 1.