Ensemble-based Offline Reinforcement Learning with Adaptive Behavior Cloning

Danyang Wang
Department of Statistics
Purdue University
West Lafayette, IN 47907
`wang4589@purdue.edu`

Lingsong Zhang
Department of Statistics
Purdue University
West Lafayette, IN 47907
`lingsong@purdue.edu`

Abstract

In this work, we build upon the offline reinforcement learning algorithm, TD3+BC [9], and propose a model-free actor-critic algorithm with an adjustable behavior cloning (BC) term. We employ an ensemble of networks to quantify the uncertainty of the estimated value function, thus addressing the issue of overestimation. Moreover, we introduce a method that is both convenient and intuitively simple for controlling the degree of BC, through a Bernoulli random variable based on the user-specified confidence level for different offline datasets. Our proposed algorithm, named Ensemble-based actor critic with Adaptive Behavior Cloning (EABC), is straightforward to implement, exhibits low variance, and achieves strong performance across all D4RL MuJoCo benchmarks.

1 Introduction

Offline RL, which relies exclusively on pre-collected offline data, has gained popularity in recent years. Similar to traditional supervised learning methods, offline RL does not require online interaction with the environment. Instead, agents learn from a static, pre-collected offline dataset throughout the entire learning process.

However, the lack of interaction with the environment in offline RL can pose challenges. Given a fixed dataset, the agent lacks the means to balance exploration and exploitation, a problem that is widely studied in online RL. As a result, the value functions of actions of inferior quality may be overestimated, leading to unpredictable behavior in the learned policy. Furthermore, when aiming to improve upon the policy that generates the static dataset, referred to as the "behavior policy" and denoted as π_β, one may encounter the dilemma of determining whether actions unseen in offline dataset are better or worse. This problem is typically addressed by online interaction. Another issue hindering offline RL algorithms is their high variance and instability compared to online RL [9], which hampers the generalization of offline RL to real-world environments.

People address these obstacles in offline RL using approaches including importance sampling, explicit policy constraints, and implicit policy constraints [20, 28, 18, 5, 7, 1]. However, many of the existing state-of-the-art offline RL algorithms exhibit long training times, and require sophisticated parameter tuning. Recognizing these challenges, TD3+BC [9] proposes a minimalist approach that makes minor modifications to the established off-policy TD3 algorithm [10], which utilizes twin Q-networks and delayed policy updates to mitigate overestimation bias in off-policy RL settings. TD3+BC achieves state-of-the-art performance when it was proposed, with minimalist modifications from TD3 while maintaining a short running time. However, a major concern with the TD3+BC algorithm is its poor performance on random datasets, as illustrated in Figure 1. This issue stems from a lack of exploration power due to the fixed BC term in TD3+BC, which constrains the learned policy to remain too close to π_β, and thus poses a problem when π_β is inferior.

Keeping the spirit of algorithmic simplicity in mind, we build our algorithm EABC upon TD3+BC. We aim to balance pessimism and optimism in the offline learning process. First, we employ a pessimistic ensemble of Q value estimates. Second, we leverage a weight function with user specified confidence level p to adjust the extent of behavior cloning (BC), taken into account the quality of the underlying dataset. Despite its simplicity, EABC delivers strong and stable performance across all D4RL MuJoCo tasks, while maintaining a short runtime. [1]

2 Related Work

Here, we focus primarily on model-free RL algorithms [25], which are most closely related to our work. Another major area of research is model-based offline RL [31, 13, 21].

Ensemble-based Pessimistic Value Estimation. Employing ensemble techniques in offline RL is a widely observed practice [2, 30, 19, 24]. In Algorithm thm as Ensemble-Diversified Actor Critic (EDAC, [2]), the authors enhance the Soft Actor-Critic (SAC, [12]) algorithm by utilizing disagreement among Q ensembles measured by cosine similarity to learn a pessimistic Q function. In contrast, while part of our work also focuses on uncertainty quantification, we simply using the standard deviation to quantify uncertainty, without imposing additional assumptions on the Q networks, such as normality. Moreover, our work EABC requires only about 10 Q-Networks to achieve superior performance, in contrast to a larger number of ensemble networks required by other approaches (e.g., EDAC requires around 50 Q-Networks). Authors of [3] suggest distinguishing between out-of-distribution and in-distribution actions, and design their algorithm around SAC. In [11], authors argue that ensemble-based methods should train on independent targets. However, we found that training with a shared target yields superior performance in our settings.

Controlling BC. The work most closely related to ours is weighted Policy Constraints (wPC, [23]), which involves learning a separate state value function on top of TD3+BC, and determining the quality of an action based on the advantage function. However, we observe that the results are chaotic and exhibit high variance across most offline RL tasks (see Table 1 and Figure 1). [4] also proposes to adjust the BC term in TD3+BC, a method they refer to as offline refinement with online fine-tuning. However, their approach requires extensive manual parameter adjustment. A recent work [26] adopts a model-based framework, and control the behavior cloning term with a random variable sampled from a Beta distribution, allowing the user to control its parameters. However, it is limited to handling discrete action spaces, while also involves training and post-training processes.

3 Background

In standard online RL setting, agent observes a state (s) from the environment, executes an action (a) following some policy $\pi(a|s)$, and then observes a reward (r), as well as the next state (s'). This sequential decision making process is modeled by a Markov Decision Process (MDP), defined by $(\mathcal{S}, \mathcal{A}, R, P, \rho_0, \gamma)$. Here, $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces; $R : (s, a) \to \mathbb{R}$ is the reward function; $P : (s, a) \to \Delta(\mathcal{S})$ is the transition dynamic; $\rho_0 : \Delta(\mathcal{S})$ is the initial state distribution; and $\gamma \in [0, 1]$ is a discount factor for the rewards. Following a given policy π, the expected return is denoted as $J(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$, where the expectation is taken over the full trajectory distribution involving ρ_0, P, and π. The agent's objective is to find an optimal policy, such that it maximizes the expected return: $\pi^* = \underset{\pi}{\operatorname{argmax}}\, J(\pi)$.

The state-action value function is defined as the expected return following a policy π, given the starting state and action (s, a): $Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a\right]$. We denote $Q_\theta(s, a)$ as the estimated Q function with modeling parameters θ. Similarly, we denote the learned policy as π_ϕ, with parameters ϕ. Their corresponding target networks are denoted as $Q_{\tilde{\theta}}$ and $\pi_{\tilde{\phi}}$, respectively. In algorithm TD3, given a replay buffer $\mathcal{B}$ with tuples of (s, a, r, s'), the learning process involves

[1]The code implementation of EABC is available at `https://github.com/Penguin0007/EABC`.

minimizing the following two loss functions for critic and actor, respectively,

$$\mathcal{L}_{\mathsf{c}}(\theta_i) = \mathbb{E}_{\mathcal{B}}\left[\left(Q_{\theta_i}(s,a) - [r + \gamma \min_{i=1,2} Q_{\tilde{\theta}_i}(s', \pi_{\tilde{\phi}}(s'))]\right)^2\right], \tag{1}$$

$$\mathcal{L}_{\mathsf{a}}(\phi) = \mathbb{E}_{\mathcal{B}}\left[- Q_{\theta_1}(s, \pi_\phi(s))\right]. \tag{2}$$

Expectations are taken with respect to the replay buffer $\mathcal{B}$. In the context of offline RL, TD3+BC introduces a modification to the actor loss function by adding a behavior cloning (BC) term. This ensures that the learned policies π_ϕ do not deviate significantly from the actions observed in the offline dataset $\mathcal{D}$ for a given state. Specifically, the loss function of TD3+BC is defined as:

$$\mathcal{L}_{\mathsf{a}}(\phi) = \mathbb{E}_{\mathcal{D}}\left[- \lambda Q_{\theta_1}(s, \pi_\phi(s)) + (\pi_\phi(s) - a)^2\right], \tag{3}$$

where λ is the normalization factor that balances the RL term and BC term, and the expectation is taken with respect to the offline dataset $\mathcal{D}$. Intuitively, however, based on the underlying quality of the behavior policy π_β, we probably want to avoid keeping too close to the underlying π_β. Instead, we want the learned policy to keep exploring more of other actions given an inferior π_β.

4 Ensemble-based Offline RL Algorithm

Uncertainty Penalized Value Estimate. Denote the true underlying Q value of a policy π_t learned at time step t as Q^{π_t}, and its estimators as $\{Q_{\theta_i}\}_{i=1}^K$, parameterized by θ_i, where K represents the number of estimators, e.g., the number of Neural Networks. Assume that each Q_{θ_i} is identically distributed, we model the distribution of the Q ensemble using its mean and standard deviation, defined as $Q_\theta := \mathbb{E}\left[Q_{\theta_i}\right]$ and $\sigma := \sigma(Q_{\theta_i})$. Without explicitly assuming the distribution that Q_{θ_i} follows (e.g., Gaussian), we consider the Q-ensemble as potentially following any distribution, a more realistic situation in practice. We approximate Q_θ and σ by $\bar{Q}_\theta := \frac{1}{K}\left(\sum_{i=1}^K Q_{\theta_i}\right)$ and $\hat{\sigma} := \sqrt{\frac{1}{K-1}\sum_{i=1}^K (Q_{\theta_i} - \bar{Q}_\theta)^2}$. Due to the presence of the overestimation bias, Q_θ (and its approximation $\bar{Q}_\theta$) still tends to overestimate Q^{π_t}. Common practices of obtaining a pessimistic estimator from ensemble include: 1) $\min_i (Q_{\theta_i})$, 2) $\min_{\text{Sample } 2 \in \{i\}} (Q_{\theta_i})$, or 3) $\bar{Q}_\theta - \beta\hat{\sigma}$. Where β represents a pre-specified real number. We choose

$$\texttt{pess}\left(Q_\theta(s,a)\right) = \bar{Q}_\theta(s,a) - \hat{\sigma}, \tag{4}$$

as the target for EABC. This penalization approach effectively penalizes the Q values without the risk of over-penalization that might occur when subtracting 2 or 3 times σ. We also evaluate the performance of other choices of 1) and 2) in Section 5.2, and found that these alternatives do not perform as well as $\bar{Q}_\theta - \hat{\sigma}$.

Adaptive Behavior Cloning. The BC term in the actor loss 3 implies $\hat{\phi} = \operatorname{argmin} \sum_{\mathcal{D}} (\pi_\phi(s) - a)^2$, which is minimized when $\hat{\pi}_\phi(s) = \mathbb{E}_{\mathcal{D}}(a|s)$, therefore maintaining a safe policy update without deviating excessively from π_β. However, π_β might not always be optimal.

To efficiently adjust the strength of behavior cloning, we propose to multiply the BC term by a weight function $w(s,a)$, while keeping λ unchanged from the TD3+BC. Given a user specified confidence level $p \in [0,1]$, we adjust the extent of BC through a Bernoulli random variable. Letting the weight $w(s,a)$ take values from $\mathcal{W} = \{0,1\}$, we determine $w(s,a)$ by randomly sampling:

$$w(s,a) = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p. \end{cases} \tag{5}$$

Through this straightforward random sampling approach, we can effectively control the extent of BC based on different quality of offline datasets. By adjusting the p value, we can decide the percentage of the offline dataset that contributes to the behavior cloning term, thereby controlling the degree to which we want to mimic the behavior policy π_β that was used to collect the offline dataset. When $p = 0$, the algorithm becomes an ensemble version of TD3, whereas $p = 1$ results in an ensemble version of TD3+BC. Following convention in offline RL, we use $d = 1$ to denote the completion of an offline trajectory collection, while $d = 0$ indicates otherwise. The EABC algorithm is summarized in Algorithm 1, with main differences from TD3+BC highlighted in red.

Algorithm 1 Ensemble-based Actor Critic with Adaptive Behavior Cloning (EABC)

```
Input: offline dataset $\mathcal{D}$, number of Q-ensembles $K$, confidence level $p \in [0,1]$. Initialize critic
network ensemble $Q_{\theta_i}$ for $i = 1, ...K$, and actor network $\pi_\phi$, with random parameters $\theta_i$'s, $\phi$.
Initialize target networks $\tilde{\theta}_i \leftarrow \theta_i$; $\tilde{\phi} \leftarrow \phi$.
for $i = 1$ to $T$ do
    Sample batch of $N$ transitions $\{(s, a, r, s', d)\}$ from $\mathcal{D}$.
    $\tilde{a}' \leftarrow \pi_{\tilde{\phi}}(s') + \epsilon, \quad \epsilon \sim clip(\mathcal{N}(0, \tilde{\sigma}), -c, c)$.
    Compute $\texttt{pess}(Q_{\tilde{\theta}}(s', \tilde{a}'))$ based on 4.
    $y = r + \gamma(1-d)\texttt{pess}(Q_{\tilde{\theta}}(s', \tilde{a}'))$.
    Update critics: $\theta_i \leftarrow \underset{\theta_i}{\text{argmin}} N^{-1} \sum (Q_{\theta_i}(s,a) - y)^2$.
    if $t$ % policy update frequency $== 0$ then
        $\tilde{a} \leftarrow \pi_\phi(s)$.
        Compute $\texttt{pess}(Q_\theta(s, \tilde{a}))$ based on 4.
        $\lambda = \frac{\alpha}{N^{-1}\sum |\texttt{pess}(Q_\theta(s,\tilde{a}))|}$.
        Sample $w(s,a) \sim Bernoulli(p)$.
        Update actor: $\phi \leftarrow \underset{\phi}{\text{argmin}} N^{-1} \sum \big[ -\lambda \texttt{pess}(Q_\theta(s,\tilde{a})) + w(s,a)(\pi_\phi(s) - a)^2 \big]$.
        Update target networks: $\tilde{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\tilde{\theta}_i$; $\tilde{\phi} \leftarrow \tau\phi + (1-\tau)\tilde{\phi}$.
    end if
end for
```

5 Experiment Results

In this section we demonstrate experimental efficacy of EABC on the D4RL MuJoCo benchmarks [8], and briefly summarize our results from ablation study.

5.1 D4RL MuJoCo Benchmark

The performance of EABC is reported in Table 1. We compare it with several algorithms similar to ours [27, 10, 9, 23], as well as some other representative model-free offline RL algorithms [17, 15]. For algorithms run locally, we train the algorithm for 1 million time steps using 5 seeds, and evaluate the learned policy through online interaction for 10 episodes, in every 5000 steps. In Section A.5, we further compare EABC with more recent state-of-the-art offline RL algorithms, including several model-based algorithms.

As highlighted in Table 1, EABC consistently exhibits low variance across all tasks, achieving remarkable stability with an average standard deviation of 1.4. A clear convergence pattern is also shown in Figure 1, showing reduced variance and improved performance across almost all tasks compared to its predecessors. Besides, EABC exhibits a comparatively shorter runtime (Appendix A.7), yet still achieves results comparable to the state-of-the-art algorithms.

5.2 Ablation Study

Effect of K. We experimented with K values taken among $K \in \{3, 5, 7, 10, 20, 30\}$. The full results are presented in Figure 2. We found that overall, a value of $K \geq 10$ gives the algorithm a stable performance, which is a relatively small number among the ensemble-based offline RL algorithms.

Other Choices of Pessimistic Targets. As mentioned in Section 4, there are 3 commonly used options for pessimistically estimating the target Q with ensemble networks, when we use 1), we denote the algorithm as `EABC_min`; when we use 2), we denote it as `EABC_min(2)`. Overall, we observe superior and more stable performance across all tasks when using Equation 4 (Figure 4).

Choosing Appropriate p. To effectively analyze the effect of p, we select p values from $\{0.0, 0.05, 0.25, 0.5, 0.75, 0.95, 1.0\}$, which covers representative quantile numbers of $p \sim \mathcal{U}(0,1)$, where $\mathcal{U}$ represents a continuous uniform distribution. The optimal p values for each offline dataset are summarized in Table 3, and full learning curves are shown in Figure 3. We find that in general, except for the task where EABC completely failed to learn ("Walker2d-Random"), higher-quality offline datasets require higher p values.

Table 1: Average D4RL normalized score over the final 10 evaluations on 5 seeds, ± standard deviation over seeds. TD3 and wPC are based on our own implementation, where experiments are run on D4RL MuJoCo environment "v2". Results for BC, TD3+BC, CQL, and IQL are based on their original paper, or re-implementations following author's recommended parameter settings. We highlight the best average, and those close to the best average with comparably smaller variance.

Task Name	BC	TD3	TD3+BC	CQL	IQL	wPC	EABC (ours)
halfcheetah-r	2.2±0.0	32.0±2.2	11.0±1.1	17.5±1.5	13.1±1.3	19.7±0.8	**32.4**±0.7
hopper-r	3.7±0.6	26.8±5.1	8.5±0.6	7.9±0.4	7.9±0.2	20.9±9.4	**31.5**±0.4
walker2d-r	1.3±0.1	-0.1±0.2	1.6±1.7	5.1±1.3	**5.4**±1.2	1.3±2.3	1.7±1.7
halfcheetah-m	43.2±0.6	33.8±11.8	48.3±0.3	47.0±0.5	47.4±0.2	53.2±0.3	**67.3**±0.9
hopper-m	54.1±3.8	0.7±0.0	59.3±4.2	53.0±28.5	66.2±5.7	79.4±2.0	**92.4**±3.9
walker2d-m	70.9±11.0	0.6±1.0	83.7±2.1	73.3±17.7	78.3±8.7	71.0±31.6	**89.0**±0.6
halfcheetah-m-r	37.6±2.1	42.3±7.8	44.6±0.5	45.5±0.7	44.2±1.2	48.1±0.4	**61.4**±1.6
hopper-m-r	16.6±4.8	44.4±23.8	60.9±18.8	88.7±12.9	94.7±8.6	94.5±3.8	**102.6**±1.4
walker2d-m-r	20.3±9.8	31.0±14.2	81.8±5.5	81.8±2.7	73.8±7.1	84.0±11.0	**93.2**±2.9
halfcheetah-m-e	44.0±1.6	6.2±7.1	90.7±4.3	75.6±25.7	86.7±5.3	63.7±10.8	**92.9**±1.9
hopper-m-e	53.9±4.7	0.7±0.1	98.0±9.4	**105.6**±12.9	91.5±14.3	64.7±29.1	**104.0**±3.6
walker2d-m-e	90.1±13.2	0.7±1.1	110.1±0.5	107.9±1.6	109.6±1.0	91.4±39.1	**112.0**±0.3
halfcheetah-e	91.8±1.5	-2.7±0.3	96.7±1.1	96.3±1.3	95.0±0.5	64.9±13.0	**97.6**±0.2
hopper-e	107.7±0.7	1.3±0.5	107.8±7	96.5±28.0	109.4±0.5	44.4±49.2	**111.2**±0.3
walker2d-e	106.7±0.2	1.8±0.3	110.2±0.3	108.5±0.5	109.9±1.2	68.1±53.9	**110.8**±0.1
Average	49.6±3.6	14.6±5.0	67.5±3.8	67.3±9.1	68.9±3.8	58.0±17.1	**80.0±1.4**

Since p is a continuous variable, it is, in many respects, robust to perturbations. This is in particular evident from Figure 3, where a broad range of p values often yield comparable performances. A general guideline is to use $p \leq 0.25$ for offline data collected with inferior policies, and $p \geq 0.5$ for more sophisticated policies. The complete results of ablation study are presented in Appendix A.6.

6 Conclusion

In this work, building upon TD3+BC, we introduce an ensemble-based actor-critic algorithm with an adjustable behavior cloning term. By employing Q ensembles, we construct a pessimistic Q estimate. More importantly, we provide a convenient and intuitively interpretable way to control the level of behavior cloning, by introducing randomness through a Bernoulli random variable with a user-specified confidence level p. Meanwhile, the proposed algorithm is simple to implement, adding around 10 lines of code to the base algorithm TD3+BC. Despite its simplicity and reduced computational time compared to many state-of-the-art offline RL algorithms, EABC demonstrates surprisingly strong experimental performance, comparable to the best on D4RL MuJoCo benchmarks. It also exhibits low episodic variance during evaluations, benefiting from the power of ensemble.

A major limitation of EABC is the requirement to preset p. A user-determined p might not always capture the full potential of EABC. However, we note that this is essentially a hyperparameter fine-tuning issue, a common challenge in almost all algorithms. With limited hyperparameters to tune, and a set of five numbers covering representative scenarios of p, our method remains appealing. One potential extension of this work could involve expanding the weights to a multi-armed bandit approach, or automating the determination of p.

The essence of EABC bears similarities to supervised learning, notably, the utilization of pre-known expert information can be immensely valuable. By leveraging such supervised information (confidence level p regarding the offline dataset), it is feasible to circumvent extensive parameter fine-tuning and complex training strategies. We aspire for our approach to encourage further development of algorithms that maintain a simple structure, employing straightforward statistical methods, and leveraging the pre-existing knowledge of offline datasets. This work can possibly spark increased interest in simpler algorithm designs that place greater emphasis on the pre-known characteristics of the offline dataset.

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[2] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

[3] Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhihong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. *arXiv preprint arXiv:2202.11566*, 2022.

[4] Alex Beeson and Giovanni Montana. Improving td3-bc: Relaxed policy constraint for offline learning and stable online fine-tuning. *arXiv preprint arXiv:2211.11802*, 2022.

[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] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[7] Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

[8] 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.

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

[10] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[11] Kamyar Ghasemipour, Shixiang Shane Gu, and Ofir Nachum. Why so pessimistic? estimating uncertainties for offline rl through ensembles, and why their independence matters. *Advances in Neural Information Processing Systems*, 35:18267–18281, 2022.

[12] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[13] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

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

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

[16] 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.

[17] 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.

[18] Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International conference on machine learning*, pages 3652–3661. PMLR, 2019.

[19] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022.

[20] 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.

[21] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

[22] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8024–8035, 2019.

[23] Zhiyong Peng, Changlin Han, Yadong Liu, and Zongtan Zhou. Weighted policy constraints for offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9435–9443, 2023.

[24] Jordi Smit, Canmanie T Ponnambalam, Matthijs TJ Spaan, and Frans A Oliehoek. Pebl: Pessimistic ensembles for offline deep reinforcement learning. In *Robust and Reliable Autonomy in the Wild Workshop at the 30th International Joint Conference of Artificial Intelligence*, 2021.

[25] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[26] Phillip Swazinna, Steffen Udluft, and Thomas Runkler. User-interactive offline reinforcement learning. *arXiv preprint arXiv:2205.10629*, 2022.

[27] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[28] 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.

[29] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*, 2021.

[30] Rui Yang, Chenjia Bai, Xiaoteng Ma, Zhaoran Wang, Chongjie Zhang, and Lei Han. Rorl: Robust offline reinforcement learning via conservative smoothing. *Advances in Neural Information Processing Systems*, 35:23851–23866, 2022.

[31] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

A Appendix

A.1 Broader Impact

Our algorithm is under the broader category of offline RL algorithms. These algorithms have natural advantage in application scenarios where ample offline data are available, such as in robotics and autonomous driving; or in situations where collecting data online using a learned policy during the training process may be dangerous or unethical, as in medical care and, once again, autonomous driving. Our proposed algorithm, EABC exhibits simplicity and efficiency, offering lower runtime compared to most current state-of-the-art algorithms, while still delivering stable and cutting-edge learning outcomes. We anticipate that our work will have a broader impact of inspiring research interest towards simpler modifications of existing algorithms, and a minimalistic approach to hyper-parameter tuning. However, it is important to note that our work does not fundamentally change the potential scope or breadth of applications for offline RL algorithms.

A.2 Experimental Details

Software

We use the following software versions: Python 3.9; Pytorch 2.1.1 [22]; Gym 0.23.1 [6]; MuJoCo 3.0.1; mujoco-py; D4RL [8]. For all D4RL datasets, we use the "v2" version, where a normalized D4RL score is provided. Formally,

$$\text{D4RL Score} = 100 \times \frac{\text{score} - \text{random score}}{\text{expert score} - \text{random score}}.$$

The code for our algorithm implementation is in `https://github.com/Penguin0007/EABC`. Due to limited resources of hardware, our experiments are run on Two AMD Epyc 7763 "Milan" CPUs @ 2.2GHz, without GPU. Average wall clock run time of EABC for each MuJoCo experiment is around 12.5 hours, while TD3+BC takes around 4 hours.

Hyperparameters

Table 2: EABC hyperparameters. The hyperparameters of TD3 and TD3+BC are kept the same from official implementation.

	Hyperparameter	Value
TD3 Hyperparameters	Optimizer	Adam [14]
	Critic learning rate	3e-4
	Actor learning rate	3e-4
	Mini-batch size	256
	Discount factor	0.99
	Target update rate	5e-3
	Policy noise	0.2
	Policy noise clipping	(-0.5, 0.5)
	Policy update frequency	2
TD3+BC Hyperparameters	α	2.5
EABC Hyperparameters	K	10
	p	See Table 3

Table 3: Best confidence level p for different D4RL MuJoCo datasets.

p	Halfcheetah	Hopper	Walker2d
Expert	0.5	1.0	0.5
Medium-Expert	1.0	0.5	0.25
Medium	0.0	0.25	0.25
Medium-Replay	0.0	0.0	0.0
Random	0.0	0.0	0.95

Neural Network Architecture

Table 4: Neural network architectures for EABC (same as TD3+BC).

Critic hidden dim	256
Critic hidden layers	2
Critic activation function	ReLU
Actor hidden dim	256
Actor hidden layers	2
Actor activation function	ReLU

A.3 Learning Curves

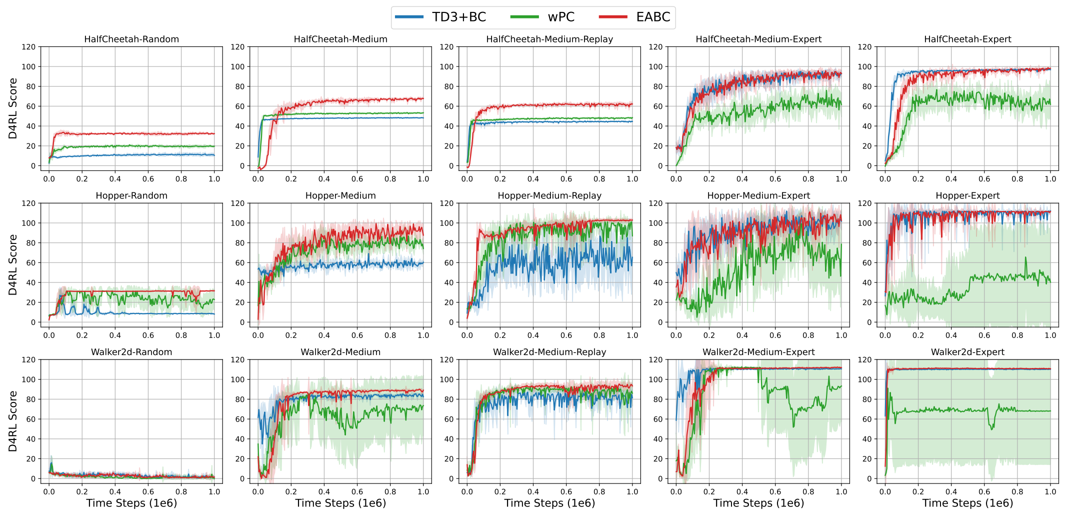

Figure 1: Learning curves of EABC, compared with TD3+BC and wPC. Curves are averaged over 5 seeds, where the shaded areas represent ± one standard deviation across seeds.

A.4 Other offline datasets

For AntMaze datasets, we find adaptively adjusting BC doesn't help in performance (Table A.4). The task that EABC significantly outperforms TD3+BC is AntMaze-Umaze, where the increase of performance all credits from ensemble. This is potentially due to the nature of the games, that AntMaze is designed to evaluate navigation and goal-reaching capabilities, whereas MuJoCo is aimed to achieve forward locomotion while maintaining balance and avoiding falling over, making adjusted BC more suitable for dataset collected with different π_β for MuJoCo tasks.

Table 5: Average D4RL normalized score over the final 10 evaluations on 5 seeds, ± standard deviation over seeds. TD3+BC is based on our own implementation with author's original code. All experiments are run on D4RL Antmaze environments "v2". We highlight the best average.

Task Name	TD3+BC	EABC	Best p for EABC
AntMaze-Umaze	59.2±45.1	**89.2±1.6**	1.0
AntMaze-Umaze-Diverse	**53.8±39.0**	32.3±32.3	0.5
AntMaze-Medium-Diverse	**1.6±2.3**	0.2±0.4	0.75
AntMaze-Medium-Play	**2.0±2.1**	0.0±0.0	1.0
AntMaze-Large-Diverse	0.0±0.0	0.0±0.0	1.0
AntMaze-Large-Play	0.0±0.0	0.0±0.0	1.0

A.5 More Baseline Comparison

We also extend our comparative analysis to include additional up-to-date most state-of-the-art baseline methods on the D4RL MuJoCo datasets. Our expanded evaluation now encompasses notable offline RL algorithms such as BEAR, UWAC and EDAC, as well as model-based approaches like MOPO and MOReL ([16, 29, 2, 31, 13]). The results are obtained from the original paper or implemented based on the author's official implementation for each method, with the normalized average returns presented in Table 6.

Table 6: Average D4RL normalized score over the final 10 evaluations on 5 seeds, ± standard deviation over seeds. Experiment results for other algorithms are based on their original paper, or re-implements with author's recommended parameters. We highlight the best average, and the ones that are close to the best average, but have comparably smaller variance.

Task Name	BEAR	MOReL	MOPO	UWAC	EDAC	EDAC-10	EABC (ours)
halfcheetah-r	2.3±0.0	25.6	**35.9**±2.9	14.5±3.3	28.4±1.0	13.4±1.1	**32.4**±0.7
hopper-r	3.9±2.3	**53.6**	16.7±12.2	22.4±12.1	25.3±10.4	16.9±10.1	31.5±0.4
walker2d-r	12.8±10.2	**37.3**	4.2±5.7	15.5±11.7	16.6±7.0	6.7±8.8	1.7±1.7
halfcheetah-m	43.0±0.2	42.1	**73.1**±2.4	46.5±2.5	65.9±0.6	64.1±1.1	67.3±0.9
hopper-m	51.8±4.0	95.4	38.3±34.9	88.9±12.2	101.6±0.6	**103.6**±0.2	92.4±3.9
walker2d-m	-0.2±0.1	77.8	41.2±30.8	57.5±7.8	**92.5**±0.8	87.6±11	89.0±0.6
halfcheetah-m-r	36.3±3.1	40.2	**69.2**±1.1	46.8±3.0	61.3±1.9	60.1±0.3	61.4±1.6
hopper-m-r	52.2±19.3	93.6	32.7±9.4	39.4±6.1	101.0±0.5	**102.8**±0.3	102.6±1.4
walker2d-m-r	7.0±7.8	49.8	73.7±9.4	27.0±6.3	87.1±2.3	**94.0**±1.2	93.2±2.9
halfcheetah-m-e	46.0±4.7	53.3	70.3±21.9	**127.4**±3.7	106.3±1.9	107.2±1.0	92.9±1.9
hopper-m-e	50.6±25.3	108.7	60.6±32.5	**134.7**±21.1	110.7±0.1	58.1±22.3	104.0±3.6
walker2d-m-e	22.1±44.9	95.6	77.4±27.9	99.7±12.2	114.7±0.9	**115.4**±0.5	112.0±0.3
halfcheetah-e	92.7±0.6	-	81.3±21.8	**128.6**±2.9	106.8±3.4	104.0±0.8	97.6±0.2
hopper-e	54.6±21.0	-	62.5±29.0	**135.0**±14.1	110.1±0.1	77.0±43.9	111.2±0.3
walker2d-e	106.6±6.8	-	62.4±3.2	**121.1**±22.4	115.1±1.9	57.8±55.7	110.8±0.1
Total	38.78±10.0	64.4	53.3±16.34	73.7±9.4	**82.9**±2.2	71.2±10.6	**80.0±1.4**

The results demonstrate that overall, EABC achieves better or comparable performance relative to most baseline methods, and closely matches the state-of-the-art EDAC method, but with a smaller standard deviation. Notably, the EDAC algorithm also employs an ensemble approach, but defaults to using 50 Neural Networks (NNs). We also compare against EDAC with 10 NNs as presented in Table 6 ([30]), where EABC outperforms EDAC-10 in most cases with the same number of NNs. Remarkably, we attain these results with a straightforward algorithm structure and minimal hyperparameters added to the base TD3+BC algorithm, with a comparatively short runtime.

A.6 Ablation Study Results

In the next few pages, we provide complete ablation study results for hyperparameters K, p, and different choices of $\texttt{pess}(Q)$. The detailed setting and choices of the hyperparameters are described in Section 5.2.

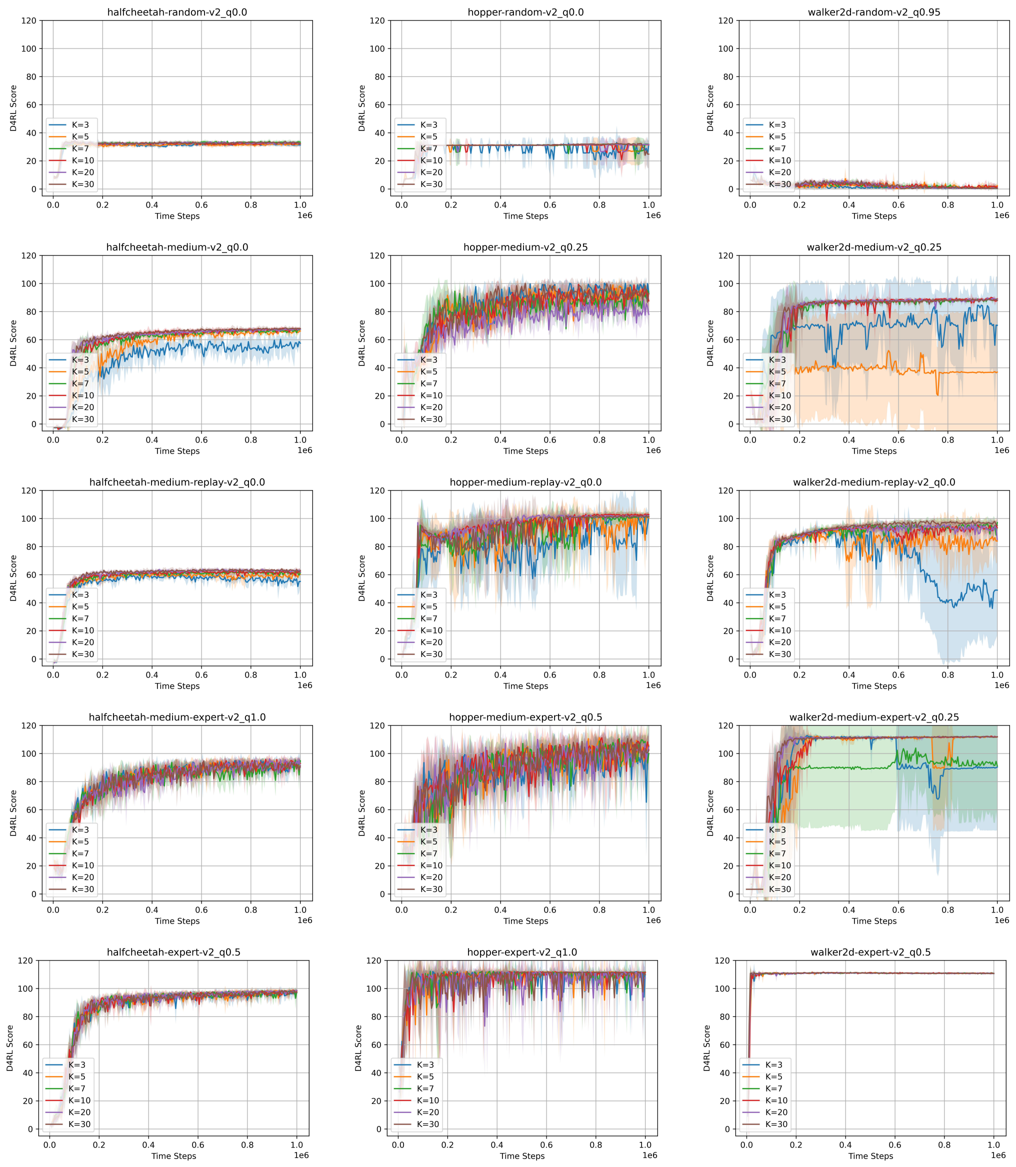

Figure 2: Ablation study of EABC for K, where K's are chosen from set $\{3, 5, 7, 10, 20, 30\}$. The p value is set at the corresponding optimal values for each tasks.

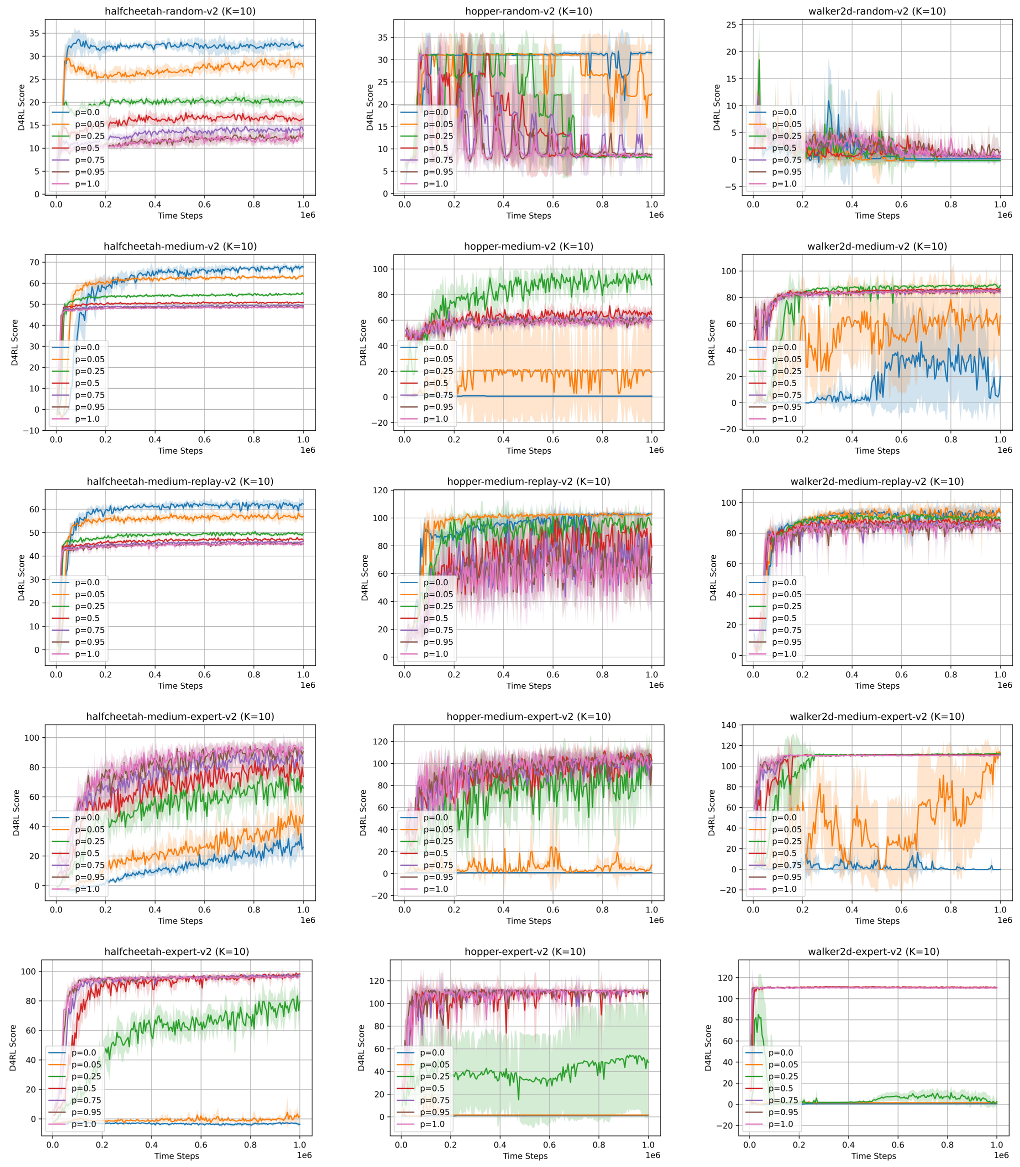

Figure 3: Ablation study of p for EABC ($K = 10$), where the choices of p's are described in 5.2.

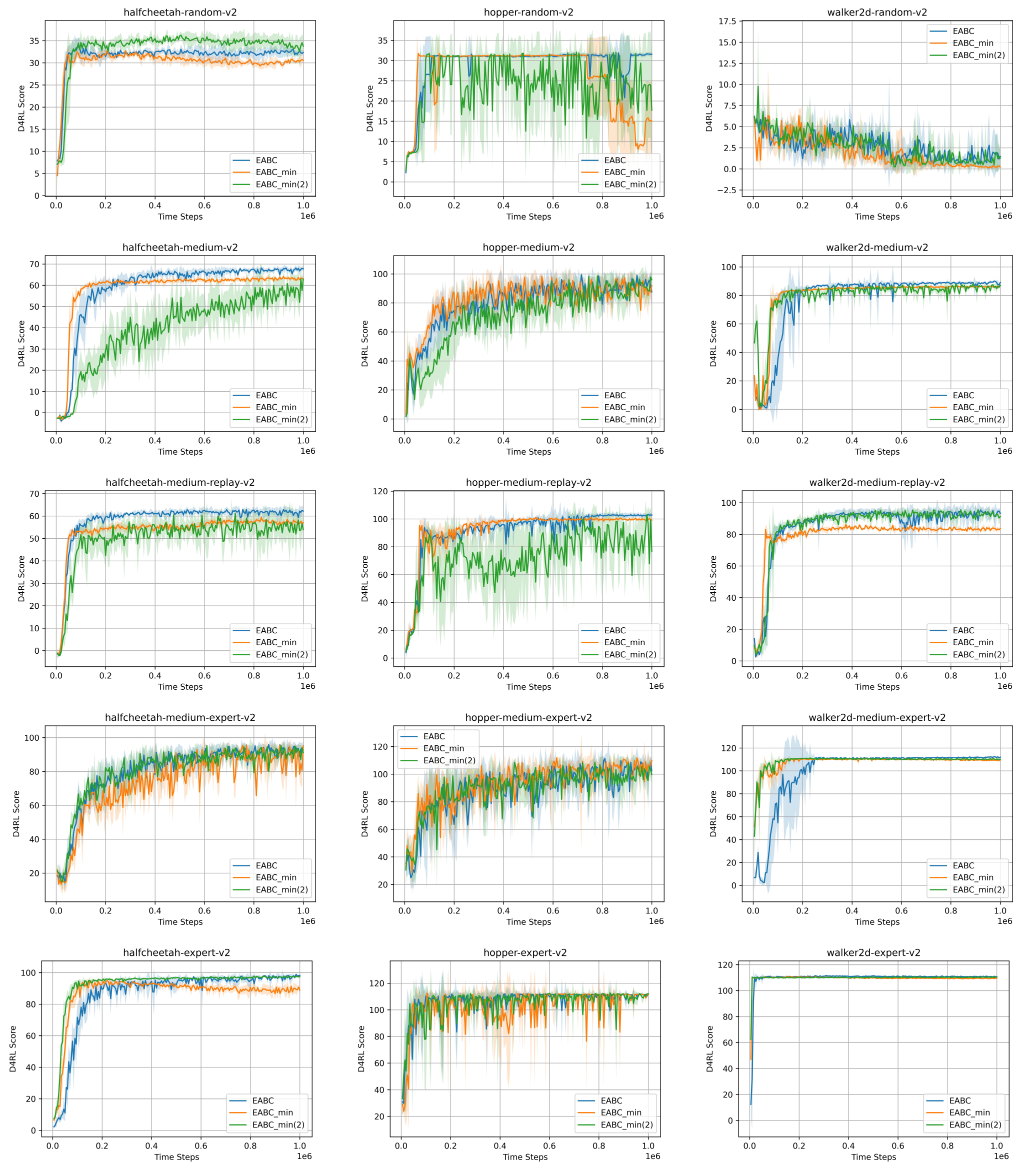

Figure 4: Ablation study results of $\texttt{pess}(Q)$. All the experiments are performed at $K = 10$, and the curves shown are corresponding to their optimistic p value for each task.

A.7 Runtime

The runtime of EABC ($K = 10$) for 1 million time steps, on average, is approximately 3.17 times that of TD3+BC, which amounts to around 2 hours of wall-clock time using the same hardware as TD3+BC. The longer runtime compared to the TD3+BC algorithm is unsurprising, considering TD3+BC employs 2 NNs, whereas EABC by default uses 10 NNs. Nevertheless, EABC's runtime is roughly half that of some more sophisticated algorithms, such as the CQL algorithm. See 5 for a visual comparison.

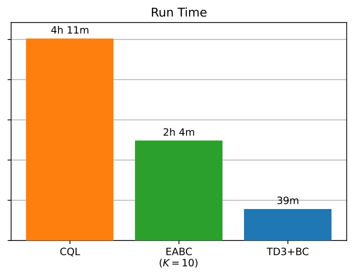

Figure 5: Runtime comparison of EABC with CQL and TD3+BC. Average wall-clock run time of EABC is about 3.17 times of TD3+BC. We therefore use the runtime of TD3+BC as a baseline, where from TD3+BC paper, all experiments are run on a single GeForce GTX 1080 GPU and an Intel Core i7-6700K CPU at 4.00GHz [9].