Showing Your Offline Reinforcement Learning Work: Online Evaluation Budget Matters

Vladislav Kurenkov
Tinkoff
`v.kurenkov@tinkoff.ai`

Sergey Kolesnikov
Tinkoff
`s.s.kolesnikov@tinkoff.ai`

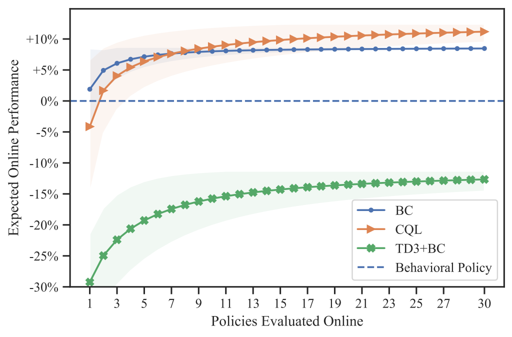
Abstract

Over the recent years, vast progress has been made in Offline Reinforcement Learning (Offline-RL) for various decision-making domains: from finance to robotics. However, comparing and reporting new Offline-RL algorithms has been noted as underdeveloped: (1) use of unlimited online evaluation budget for hyperparameter search (2) sidestepping offline policy selection (3) ad-hoc performance statistics reporting. In this work, we propose an evaluation technique addressing these issues, Expected Online Performance, that provides a performance estimate for a best-found policy given a fixed online evaluation budget. Using our approach, we can estimate the number of online evaluations required to surpass a given behavioral policy performance. Applying it to several Offline-RL baselines, we find that with a limited online evaluation budget, (1) Behavioral Cloning constitutes a strong baseline over various expert levels and data regimes, and (2) offline uniform policy selection is competitive with value-based approaches. We hope the proposed technique will make it into the toolsets of Offline-RL practitioners to help them arrive at informed conclusions when deploying RL in real-world systems.

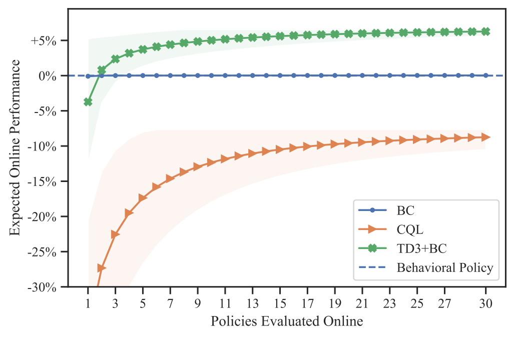
1 Introduction

In recent years, significant success has been achieved in applying Reinforcement Learning (RL) to different real-world scenarios [2, 28, 11]. Nevertheless, RL algorithms are known as being hypersensitive to the choice of hyperparameters [15], which is one of the reasons why their results are usually reported for a narrow range of values. Consequently, they are primarily run in simulated environments before being transferred and deployed to real-world systems. However, beyond the scope of cheap simulated environments, collecting new data in many real-world domains is too tricky and cost-ineffective, making it unrealistic to train an online RL agent for such applications. Fortunately, these systems could already store enough logged data about previously made decisions, making it possible to use other machine learning techniques.

Offline Reinforcement Learning (Offline-RL) is the study of algorithms that could learn a policy from a fixed, previously recorded dataset without the opportunity to interact with the environment during training [20] . However, the evaluation of Offline-RL methods is still mostly done through online performance reports on the best set of hyperparameters, but it is not the only approach. As one of the main goals of Offline-RL is minimizing environment interactions, such evaluation can be confusing for RL practitioners. Recently, authors started approaching algorithm comparison in more nuanced ways. For example, they did so by limiting the number of hyperparameter search trials [1], or reporting winning ratios over the baseline [25]. While we find these steps helpful for comparing Offline-RL algorithms, we believe that there is a need for a unified approach that would make result reporting more consistent and less ad-hoc.



(a) CityLearn



(b) FinRL

Figure 1: Expected Online Performance curve under uniform offline policy selection. Each point on a curve is the expected performance of the best policy deployed online as a function of the budget. Each algorithm was trained using 999 trajectories. The graphs are for medium-level policies.

To overcome the abovementioned challenges, we would like to propose a technique for estimating the performance of a best-found policy given a fixed online evaluation budget. The contributions in this paper are summarized as follows:

- An Offline-RL algorithm comparison technique, named Expected Online Performance (EOP), that simultaneously takes into account offline policy selection method, hyperparameter search, and limited evaluation budget.
- Policy selection methods benchmark using the proposed technique. We found that uniform policy selection is competitive compared to other selection methods under a sufficient evaluation budget for our set of environments.
- Offline-RL algorithms comparison based on EOP. Our results show that preferring one algorithm over another is budget-dependent and that behavioral cloning is a strong baseline under constrained online evaluation budget.

We hope these findings will improve scientific RL performance reporting and help design Offline-RL algorithms for real-world applications. We emphasize that this work is devoted to evaluation methodology, not conducting large-scale benchmarking.

2 Related Work

Evaluation issues related to results reporting have surfaced in many machine learning subareas. Dodge et al. [3] conducted detailed research on an advanced reporting technique, namely Expected Validation Performance (EVP), in the domain of NLP. Mitchell et al. [23] proposed a whole set of prompts for model reporting, including dataset documentation and ethical considerations.

As for the domain of RL, many questions addressing reproducibility and evaluation correctness have started arising in the community after the Duan et al. [4] article. In particular, Henderson et al. [15] investigates challenges posed by reproducibility, proper experimental techniques, and reporting procedures. They also suggest guidelines for making future deep RL results more reproducible, such as using several random seeds, reporting the full hyperparameters grid, and building hyperparameter-agnostic algorithms. Such an approach could help in reducing unfairness from external sources when comparing RL algorithms.

Many works have been published as part of Offline-RL research. To probe novel Offline-RL algorithms [17, 7, 31, 26, 18, 10], a number of diverse offline datasets were introduced recently [5, 12, 25]. At the same time, to establish a common set of benchmarks and evaluation practices, considerable work has been done on offline policy evaluation and selection [6, 30, 16, 24]. Our methodology continues this line of research to propose an all-in-one approach for reporting Offline-RL results and draw a connection to relevant techniques in other areas of deep learning research, such as EVP.

3 Background

3.1 Offline-RL Problem Statement

Reinforcement learning (*RL*) is a framework for solving sequential decision-making problems. It is typically formulated as a Markov Decision Process (MDP) over a 5-tuple (S, A, P, r, γ), with action space A, state space S, transition dynamics P, reward function r, and discount factor γ. The goal of the learning agent is to obtain a policy $\pi(s, a)$ that maximizes the expected discounted return $E_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}]$ through interaction with the MDP.

In *Offline-RL*, also known as Batch-RL, instead of obtaining data and learning via environment interactions, the agent solely relies on a static dataset D that was collected under some unknown behavioral policy (or a mixture of policies) π_{μ}. This setting is considered more challenging, as the agent loses its ability for exploration [20] and is faced with the problem of extrapolation error, not being able to correct its estimation inaccuracies when selected actions are not present in the training dataset [9].

3.2 Offline-RL Training and Evaluation Protocol

Training and evaluating Offline-RL algorithms is still in active development, and various authors approach this in different ways by simplifying the pure offline setting [12]. At the core of the simplifications are two primary issues: (1) unlimited amount of online evaluations available, and therefore (2) sidestepping offline policy selection. For example, it is common to report maximum performance for the best set of hyperparameters. In addition, the number of search trials is not even made explicit in many cases [18].

To eliminate these simplifications, we adhere to a more general setup for training and evaluating Offline-RL algorithms similar to Paine et al. [24] in order to satisfy pure offline constraints.

First, the dataset D is randomly trajectory-wise split into training D_T and validation D_V subsets accordingly. Then a sequence of hyperparameter assignments $(h_1, h_2, ..., h_N)$ is sampled for running an algorithm of interest, resulting in a sequence of policies $(\pi_1, \pi_2, ..., \pi_N)$. Note that at this stage, we do not know how good these policies are.

Then, $B \leq N$ of policies are arbitrarily chosen for online evaluation, which we refer to as an *online evaluation budget*. In the purest Offline-RL setting, $B = 1$. However, generalization to $B > 1$ is justified by the fact the online evaluation budget is conditioned on a decision-making problem at hand and available resources (e.g. few A/B tests are within the budget).

To choose policies for online evaluation, one must resort to *offline policy selection* methods. In specific domains, like recommender systems, one could pick policies based on established offline metrics, e.g. Recall, computed on the validation D_V dataset [32]. Yet, for other domains, general methods can be required.

3.3 Offline Policy Selection

After sampling N hyperparameters, running an Offline-RL algorithm, and obtaining N policies, B of them are selected for online evaluation $\pi_1, ..., \pi_B$. Generally, the probability of being selected for each policy can be defined as $p(\pi_i | h_i, D_v, S)$, where h_i is a tuple of hyperparameters used for training policy π_i, D_v is a validation dataset, and S is a offline policy selection strategy.

There is a plethora of approaches to offline policy evaluation and selection [16, 6, 30]. In this work, we utilize a few already probed selection methods in the robotics domain: those based on value functions [24] and action distributions [22]. In addition, we consider Uniform Selection, a straightforward baseline left unexplored in the context of Offline-RL.

3.3.1 Uniform

Uniform Selection is an extremely simple baseline that corresponds to picking policies completely at random. It ignores all the information regarding the validation dataset. The probability of a policy π_i being selected for online evaluation is simply proportional to the probability of its hyperparameters $p(h_i)$.

$$p(\pi_i | h_i, D_v, S = uniform) \propto p(h_i) \tag{1}$$

As we will further demonstrate, this naive strategy serves as a competitive baseline to other selection methods, has nice computational properties, and is strongly linked with a comparison tool utilized in natural language processing problems [3].

3.3.2 Expected Initial State Value

These selection methods estimate an expected value of the target policy π_i for the initial state distribution $E_{s_0 \sim D_V}[Q(s_0, \pi_i(s_0))]$ and re-rank the policies accordingly. The Q function can be reused from a training process (e.g. a critic from Conservative Q-Learning) – $V(s_0)$ *using Critic*, or be refitted using Fitted-Q Evaluation [19] – we refer to it as $V(s_0)$ *using FQE*. These methods were found to be superior to others in simulated robotic tasks [24].

3.3.3 Temporal Difference Error

This approach uses an average of the temporal difference across all transition tuples $(s, a, r, s^{'})$ in the validation dataset $E_{(s,a,r,s^{'}) \sim D_V}[r + \gamma Q(s^{'}, \pi_i(s^{'})) - Q(s, a)]$. The lower the error for a policy, the higher its rank. Note that this method is rather indicative of the critics' quality and was shown to perform poorly [24]. However, we still include it to obtain more evidence in novel environments (as far as we know, this approach was not tested for our set of environments) and to probe its ability to extract policies that improve over the baseline rather than try to find the best one.

3.3.4 Action Difference

This method measures how good a trained policy π_i matches the behavioral policy π_μ by estimating an average expected difference in action $E_{(s,a) \sim D_V}[(\pi_i(s) - a)^2]$. The lower the discrepancy, the higher the trained policy is ranked. This method is expected to result in finding policies close to the behavioral, with no sudden drop in performance at the expense of lower expected improvement.

4 Comparing Offline-RL Algorithms

Most mainstream Offline-RL literature uses unlimited online evaluation for comparing new algorithms [18, 7, 17]. This approach is primarily justified by the need to demonstrate that successful reinforcement learning from static datasets is achievable.

As considerable progress was attained in this sphere, we argue that there is now a need for a more nuanced comparison method that would address limitations and desiderata coming from real-life setups.

First, real-life setups have varied and limited online evaluation budgets. Therefore, one needs a way to tell how many policies must be deployed to find one achieving a satisfactory performance, as offline policy evaluation is still far from being perfect [25].

Second, one needs to accurately identify what their aim is when applying Offline-RL algorithms. For us, the ultimate goal is to improve over the baseline, i.e. a policy that is usually deployed at the moment. Typically, this is one of the policies that collected the training dataset. Therefore, a satisfactory report method must expose how well-trained policies are with respect to the baseline performance.

4.1 Expected Online Performance

To meet these goals, we reframe Expected Validation Performance (EVP) [3] for an Offline-RL setting, replacing a computational budget with an online evaluation budget (typically, $B \ll N$). We refer to this approach as Expected Online Performance.

Our end goal is to obtain an estimate of an expected maximum performance given that we could deploy B policies out of N trained for online evaluation. The parameters of interest to us are θ_1,...,θ_B, where

$$\theta_b = E[max(V(\pi_1), ..., V(\pi_b))] = E[V_{(b:b)}] \tag{2}$$

for $1 \leq b \leq B$ and V is an random variable (RV) representing the result of online evaluation relative to the baseline.

In other words, θ_b is the expected value of the b^{th} order statistic for a sample of size b. Recall that the i^{th} order statistic $V_{(i:b)}$ is an RV representing the i^{th} smallest value if the RVs were sorted.

4.1.1 Under Uniform Policy Selection

As can be seen in the Equation 1, the probability of a policy being selected for online evaluation under uniform selection is proportional to the probability of its hyperparameters. Virtually, that makes θ_b be based on a sample size b drawn independent and identically distributed.

In this case, the estimator proposed in Dodge et al. [3] can be readily applied. The derivation is similar to Tang et al. [27] as follows:

$$Pr[V_{(b:b)} < v] = Pr[V(\pi_1) \leq v \wedge ... \wedge V(\pi_b) \leq v] \tag{3}$$

$$= Pr[V(\pi) \leq v]^b, \tag{4}$$

which we denote as $F^b(v)$. Then

$$\theta_b = E[V_{(b:b)}] = \int_{-\inf}^{\inf} v dF^b(v). \tag{5}$$

To approximate the Cumulative Distribution Function (CDF), use Empirical Cumulative Distribution Function (ECDF)

$$\hat{F_N^b} = (\frac{1}{N}\sum_{i=1}^{N} I[v_i \leq v])^b \tag{6}$$

To arrive at the final estimator, replace CDF with an ECDF in Equation 5

$$\hat{\theta_b} = \int_{-\inf}^{\inf} v d\hat{F_N^b} \tag{7}$$

which, by definition, evaluates to

$$\hat{\theta_b} = \frac{1}{N}\sum_{i=1}^{N} v(\pi_i)(\hat{F_N^b}(v(\pi_i)) - \hat{F_N^b}(v(\pi_{i-1})) \tag{8}$$

One caveat of this estimator is that it is negatively biased and thus may sometimes lead to false conclusions about how many online evaluations are needed to prefer one method over another [27]. On the other hand, this also provides a side-effect of lower bounding the true value, which is useful in risk-sensitive scenarios.

4.1.2 Under Arbitrary Policy Selection

Since the policies selected under an arbitrary strategy (e.g. FQE) are generally not i.i.d, the plug-in estimator derived above is invalid in this case. However, one can still estimate the parameters of interest using an average estimator.

$$\hat{\theta_b} = \frac{1}{M}\sum_{r=1}^{M} max(V_1^r, ..., V_b^r) \tag{9}$$

where M is a number of hyperparameter searches with offline policy selection runs, V_i^r corresponds to an RV representing the result of online evaluation for the i^{th} selected policy in run r.

This estimator requires multiple runs of hyperparameter search with offline policy selection and is thus more expensive than the estimator in Equation 8 in terms of computing time.

5 Experimental Setup

To demonstrate how the proposed technique could be employed for analysis, we probe several Offline-RL algorithms with different offline policy selection methods in "near real-world" simulated environments, and then report the results using the Expected Online Performance curve. First, we describe our experimental setup, and then we discuss the obtained results.

5.1 Environments

We use a subset of environments from Qin et al. [25], namely: FinRL [21], CityLearn [29], and Industrial Benchmark [14]. Here, we only give a short description of these environments and highlight notable characteristics. More information on the nature of their state and action spaces can be found in the original paper.

FinRL is a trading simulator that mimics the stock market. It is capable of accounting for several market frictions like transaction costs and market liquidity. The reward is defined as a gain in the total assets. Note that this environment exhibits no stochasticity due to its backtesting nature.

CityLearn is an environment aimed at modeling urban-scale energy management and demand response. The agent's goal is to reshape the electricity demand aggregation curve to decrease the overall costs by coordinating the control of water storage in city buildings or electricity consumers. As in Qin et al. [25], we use a centralized single-agent mode.

Industrial Benchmark is an artificial environment that aims to model vital aspects found in industrial problems, such as the control of wind and gas turbines, at a similar level of complexity. This environment is highly stochastic, contains delayed reward patterns, and is partially observable.

5.2 Multi-Level Policies and Dataset Sizes

We replicate a procedure from Qin et al. [25] for obtaining behavioral policies and datasets collection: we run Soft Actor-Critic [13] until convergence and extract from 2 to 3 checkpointed policies with varying levels of expertise: low, medium, and high.

Afterwards, these policies are used to collect datasets of varying sizes. Similar to Qin et al. [25], we use the 99-999-9999 trajectories scheme. However, we do not inject any randomization while collecting the datasets, making the setting closer to more stringent real-life setups. More details on this and the concrete performance numbers from the previous paragraph can be found in the appendix.

5.3 Algorithms

Since our focus is not on extensive benchmarking of a broad set of Offline-RL algorithms but rather on demonstrating the use of the proposed comparison technique, we consider only three algorithms.

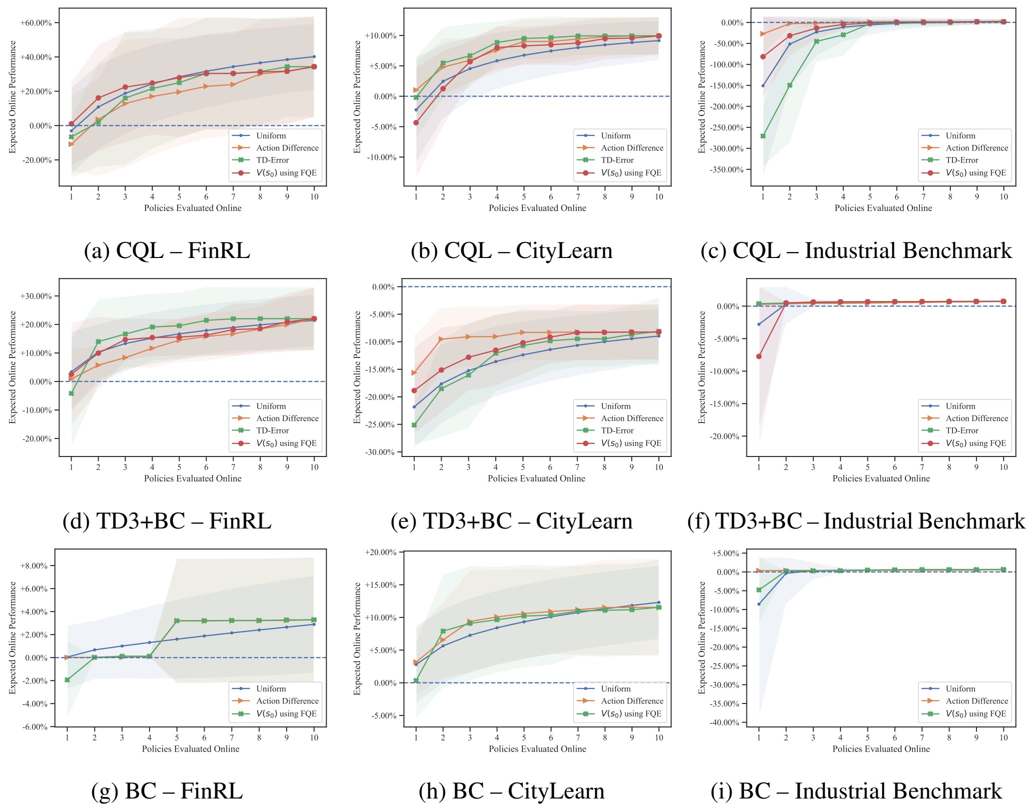
Behavioral Cloning (BC) is a well-studied approach for training an agent by imitating a behavioral policy. The agent is typically trained using supervised learning through trying to map states to actions present in a given dataset and minimizing the discrepancy.

Conservative Q-Learning (CQL) is a state-of-the-art Offline-RL algorithm that learns a conservative Q-function lower-bounding a true value to prevent overestimation [18]. This method was extensively probed in simulated robotic tasks and in our set of environments. However, this algorithm is known to have a high number of hyperparameters that need to be tuned.

Twin-Delayed Deep Deterministic Policy Gradients (TD3) + BC is a recently proposed minimal approach to Offline-RL. It adds a behavioral cloning term to a well-tested off-policy RL algorithm TD3 [8] and suggests whitening the states over the training dataset [7].

$$\pi = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\lambda Q(s, \pi(s)) - (\pi(s) - a)^2\right] \tag{10}$$

This approach overperformed CQL on several simulated robotics tasks while being four times as computationally efficient. For each algorithm, we either use a network architecture utilized in Qin et al. [25] or from an original paper. More details can be found in the appendix.



(a) CQL – FinRL (b) CQL – CityLearn (c) CQL – Industrial Benchmark

(d) TD3+BC – FinRL (e) TD3+BC – CityLearn (f) TD3+BC – Industrial Benchmark

(g) BC – FinRL (h) BC – CityLearn (i) BC – Industrial Benchmark

Figure 2: Expected Online Performance under different offline policy selection methods. In most cases, the resulting curves are hardly distinguishable, suggesting that uniform selection should not be overlooked in research reports and practitioner toolsets. The shaded area represents one standard deviation.

5.4 Training and Evaluation Protocol

We use the protocol described in the background section. Hyperparameters are sampled randomly from a space based on the values previously reported in literature. See the appendix for more discussion on this procedure. We also take only the latest policy from each training run since early stopping in Offline-RL is an open problem.

5.5 Computational Resources

The experiments were run on a computational cluster with 14x NVIDIA Tesla V100, 256GB RAM, and Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz (72 cores) for 13 days.

6 Analysis

6.1 Go Sophisticated or Uniform Selection?

We start our analysis by comparing different offline policy selection methods using the expected online performance curve depicted in Figure 2. Each point is averaged across all available dataset sizes and policy levels over 3 seeds.

Analyzing the results, it is hard to notice any clear winner across all environments or algorithms. This observation suggests that the offline policy selection field is still in its infancy [25], and studied selection methods do not necessarily transfer between decision-making domains.

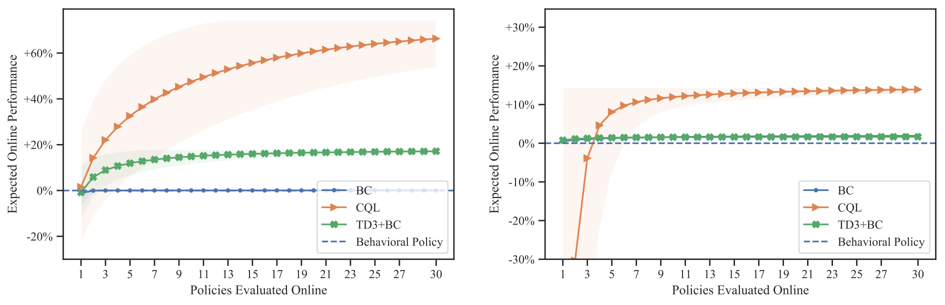
On the other hand, the uniform selection is competitive with other methods while being a clear winner in terms of computing time and ease of adoption (Figure 2). We acknowledge that there is a chance that averaging could be misleading, and we resorted to reporting such graphs due to the space limitations. Nonetheless, we provide an exhaustive set of disentangled graphs in the appendix, which can be used to arrive at the same conclusions.

6.2 Offline-RL Algorithms Comparison

Here, we apply the proposed technique for comparing Offline-RL algorithms between each other. We use uniform policy selection, as we have just observed its competitiveness with other selection methods. It also allows us to consider a higher number of deployed policies as there is no need in averaging across random seeds, and we can use an estimator from Equation 8 to plot the expected online performance curve.

6.2.1 Behavioral Cloning is Strong

Recently, Mandlekar et al. [22] demonstrated the superiority of Behavioral Cloning when learning on human demonstrations against other Offline-RL algorithms. In addition, Qin et al. [25] reported and highlighted the cases in which behavioral cloning could not be beaten. Analyzing BC under varying evaluation budgets, we also find that it is a strong baseline with properties useful under real-life constraints.



(a) FinRL (b) Industrial Benchmark

Figure 3: Expected Online Performance of Offline-RL algorithms under uniform policy selection. The graphs are for medium-level policies using 999 trajectories for training. In cases when Behavioral Cloning cannot improve the behavioral policy, it at least matches its performance.

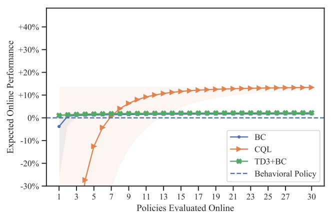
Specifically, BC can improve over the behavioral policy (Figure 1 (a), Figure 4). It is even preferable to CQL under a modest evaluation budget, as around seven CQL policies would need to be deployed to find one with better performance. Moreover, BC always matches the performance of the behavioral policy with a minimum number of policies deployed online (Figure 3). This pattern persists over different policy levels and dataset sizes. We provide more graphs in the appendix.

6.2.2 Different Algorithms are Preferred Under Varied Online Budgets

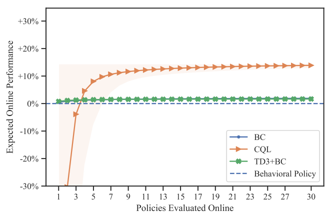
One of the key insights Expected Validation Performance [3] provided was a clear demonstration of the dependence between the available computational budget and the final best performance, and therefore, preference between methods. We observe a similar dependence for Offline-RL algorithms and online evaluation budget. In particular, we see in Figures 1 and 3 that there are several cases where CQL becomes preferable to TD3+BC and BC when having access to a higher number of online evaluations. This is particularly crucial in the industrial benchmark environment. More examples are provided in the appendix.

6.3 How Many Policies to Deploy?

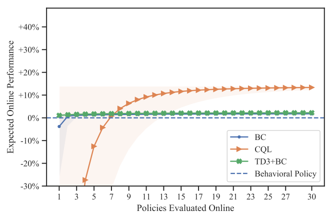
As an interesting side effect of the proposed technique, we can estimate the average number of policies for online deployment to find at least one that is performing satisfactorily. As we can observe in Figures 4 and 1, this number can vary greatly from environment to environment under different dataset conditions. We find this property of expected online performance especially interesting, as it



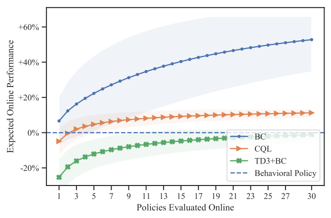
(a) Industrial Benchmark, 9999



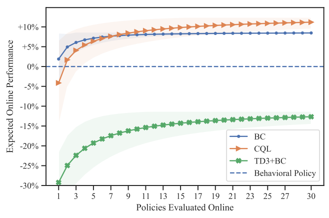
(b) Industrial Benchmark, 999



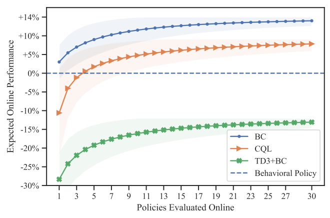
(c) Industrial Benchmark, 99



(d) CityLearn, 9999



(e) CityLearn, 999



(f) CityLearn, 99

Figure 4: Expected Online Performance of Offline-RL algorithms under uniform policy selection over different dataset sizes. The graphs are for medium-level policies. In most cases, when the Offline-RL algorithm is successful, it is expected to deploy more than one policy online to find one which improves over the baseline.

clearly demonstrates that with current state-of-the-art algorithms, selecting just one policy after a hyperparameter search is not sufficient, and one should try more policies. In addition, using the EOP for results reporting can help a practitioner to forecast a potential number of A/B tests to run when deploying RL methods.

7 Conclusion

In this article, we proposed an Offline-RL algorithm comparison technique named Expected Online Performance to estimate the performance for a best-found policy given a fixed online evaluation budget. Using the proposed approach, we benchmarked different offline policy selection techniques with varying online evaluation budgets, and found that uniform policy selection is a competitive baseline compared to other selection methods. We also conducted an analysis of Offline-RL algorithms and found behavioral cloning to be a strong baseline under realistic offline constraints. In addition, we observed that preferring one algorithm over another is budget-dependent.

References

[1] D. Brandfonbrener, W. F. Whitney, R. Ranganath, and J. Bruna. Offline RL Without Off-Policy Evaluation. *arXiv:2106.08909 [cs, stat]*, June 2021.

[2] M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, and E. H. Chi. Top-K Off-Policy Correction for a REINFORCE Recommender System. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 456–464, New York, NY, USA, Jan. 2019. Association for Computing Machinery. ISBN 978-1-4503-5940-5. doi: 10.1145/3289600.3290999.

[3] J. Dodge, S. Gururangan, D. Card, R. Schwartz, and N. A. Smith. Show Your Work: Improved Reporting of Experimental Results. *arXiv:1909.03004 [cs, stat]*, Sept. 2019.

[4] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1329–1338, New York, NY, USA, June 2016. JMLR.org.

[5] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *arXiv:2004.07219 [cs, stat]*, Feb. 2021.

[6] J. Fu, M. Norouzi, O. Nachum, G. Tucker, Z. Wang, A. Novikov, M. Yang, M. R. Zhang, Y. Chen, A. Kumar, C. Paduraru, S. Levine, and T. L. Paine. Benchmarks for Deep Off-Policy Evaluation. *arXiv:2103.16596 [cs, stat]*, Mar. 2021.

[7] S. Fujimoto and S. S. Gu. A Minimalist Approach to Offline Reinforcement Learning. *arXiv:2106.06860 [cs, stat]*, June 2021.

[8] S. Fujimoto, H. Hoof, and D. Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, July 2018.

[9] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau. Benchmarking Batch Deep Reinforcement Learning Algorithms. *arXiv:1910.01708 [cs, stat]*, Oct. 2019.

[10] S. Fujimoto, D. Meger, and D. Precup. Off-Policy Deep Reinforcement Learning without Exploration. *arXiv:1812.02900 [cs, stat]*, Aug. 2019.

[11] J. Gauci, E. Conti, Y. Liang, K. Virochsiri, Y. He, Z. Kaden, V. Narayanan, X. Ye, Z. Chen, and S. Fujimoto. Horizon: Facebook's Open Source Applied Reinforcement Learning Platform. *arXiv:1811.00260 [cs, stat]*, Sept. 2019.

[12] C. Gulcehre, Z. Wang, A. Novikov, T. L. Paine, S. G. Colmenarejo, K. Zolna, R. Agarwal, J. Merel, D. Mankowitz, C. Paduraru, G. Dulac-Arnold, J. Li, M. Norouzi, M. Hoffman, O. Nachum, G. Tucker, N. Heess, and N. de Freitas. RL Unplugged: A Suite of Benchmarks for Offline Reinforcement Learning. *arXiv:2006.13888 [cs, stat]*, Feb. 2021.

[13] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, July 2018.

[14] D. Hein, S. Depeweg, M. Tokic, S. Udluft, A. Hentschel, T. A. Runkler, and V. Sterzing. A benchmark environment motivated by industrial control problems. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Nov. 2017. doi: 10.1109/SSCI.2017.8280935.

[15] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep Reinforcement Learning That Matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, Apr. 2018.

[16] A. Irpan, K. Rao, K. Bousmalis, C. Harris, J. Ibarz, and S. Levine. Off-Policy Evaluation via Off-Policy Classification. *arXiv:1906.01624 [cs, stat]*, Nov. 2019.

[17] I. Kostrikov, J. Tompson, R. Fergus, and O. Nachum. Offline Reinforcement Learning with Fisher Divergence Critic Regularization. *arXiv:2103.08050 [cs]*, Mar. 2021.

[18] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative Q-Learning for Offline Reinforcement Learning. *arXiv:2006.04779 [cs, stat]*, Aug. 2020.

[19] H. M. Le, C. Voloshin, and Y. Yue. Batch Policy Learning under Constraints. *arXiv:1903.08738 [cs, math, stat]*, Mar. 2019.

[20] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv:2005.01643 [cs, stat]*, Nov. 2020.

[21] X.-Y. Liu, H. Yang, Q. Chen, R. Zhang, L. Yang, B. Xiao, and C. D. Wang. FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance. *arXiv:2011.09607 [cs, q-fin]*, Nov. 2020.

[22] 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. *arXiv:2108.03298 [cs]*, Aug. 2021.

[23] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru. Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 220–229, New York, NY, USA, Jan. 2019. Association for Computing Machinery. ISBN 978-1-4503-6125-5. doi: 10.1145/3287560.3287596.

[24] T. L. Paine, C. Paduraru, A. Michi, C. Gulcehre, K. Zolna, A. Novikov, Z. Wang, and N. de Freitas. Hyperparameter Selection for Offline Reinforcement Learning. *arXiv:2007.09055 [cs, stat]*, July 2020.

[25] R. Qin, S. Gao, X. Zhang, Z. Xu, S. Huang, Z. Li, W. Zhang, and Y. Yu. NeoRL: A Near Real-World Benchmark for Offline Reinforcement Learning. *arXiv:2102.00714 [cs]*, Feb. 2021.

[26] N. Y. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller. Keep Doing What Worked: Behavioral Modelling Priors for Offline Reinforcement Learning. *arXiv:2002.08396 [cs, stat]*, June 2020.

[27] R. Tang, J. Lee, J. Xin, X. Liu, Y. Yu, and J. Lin. Showing Your Work Doesn't Always Work. *arXiv:2004.13705 [cs]*, Apr. 2020.

[28] X. Tang, Z. T. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye. A Deep Value-network Based Approach for Multi-Driver Order Dispatching. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 1780–1790, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330724.

[29] J. R. Vázquez-Canteli, J. Kämpf, G. Henze, and Z. Nagy. CityLearn v1.0: An OpenAI Gym Environment for Demand Response with Deep Reinforcement Learning. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '19, pages 356–357, New York, NY, USA, Nov. 2019. Association for Computing Machinery. ISBN 978-1-4503-7005-9. doi: 10.1145/3360322.3360998.

[30] C. Voloshin, H. M. Le, N. Jiang, and Y. Yue. Empirical Study of Off-Policy Policy Evaluation for Reinforcement Learning. *arXiv:1911.06854 [cs, stat]*, Feb. 2020.

[31] Z. Wang, A. Novikov, K. Zolna, J. T. Springenberg, S. Reed, B. Shahriari, N. Siegel, J. Merel, C. Gulcehre, N. Heess, and N. de Freitas. Critic Regularized Regression. *arXiv:2006.15134 [cs, stat]*, Sept. 2020.

[32] X. Xin, A. Karatzoglou, I. Arapakis, and J. M. Jose. Self-Supervised Reinforcement Learning for Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 931–940. Association for Computing Machinery, New York, NY, USA, July 2020. ISBN 978-1-4503-8016-4.

8 Appendix

8.1 Hyperparameters

To define the space of hyperparameters, we mimicked a procedure common among deep learning practitioners. For a given algorithm, we scan through relevant papers and extract values specified there. However, instead of taking specific hyperparameter assignments, we randomize across the extracted values, as the optimal choice is largely problem-dependent. The exact hyperparameter space can be found in Table 1.

8.1.1 CQL

We use hyperparameter values mentioned in Kumar et al. [18] and Qin et al. [25]. Actor and critic networks are the same as in Qin et al. [25]: two separate MLPs with 2 hidden layers and 256 units per layer. The implementation can be found in the attached source code.

8.1.2 TD3+BC

For this algorithm, we rely on hyperparameters specified in Fujimoto and Gu [7] and also extend it with learning rates from Kumar et al. [18]. Actor and critic networks are the same as in Fujimoto and Gu [7]: two separate MLPs with 2 hidden layers and 256 units per layer. The implementation can be found in the attached source code.

8.1.3 BC

The search space for Behavioral Cloning is taken from Mandlekar et al. [22] and Qin et al. [25]. The network architecture is the same as in the algorithms above. Note that we do not use early stopping as in Qin et al. [25], but rather fix the number of gradient steps.

Algorithms	**Search Space**
CQL	variant $\in \{\mathcal{H}, \rho\}$
	$\alpha \in \{5, 10\}$
	$\tau \in \{-1, 2, 5, 10\}$
	approximate-max backup $\in$ {True, False}
	actor alpha tuning $\in$ {True, False}
	actor learning rate $\in \{3e-5, 3e-4, 1e-4, 1e-3\}$
	critic learning rate $\in \{3e-4, 1e-4, 1e-3\}$
	tau $\in \{5e-3, 1e-2\}$
	batch size $\in \{256, 512\}$
	$\gamma \in [0.9, 1.0]$
	number of gradient steps $\in \{3e5\}$
TD3+BC	$\alpha \in \{1.0, 2.0, 2.5, 3.0, 4.0\}$
	actor learning rate $\in \{3e-5, 3e-4, 1e-4, 1e-3\}$
	critic learning rate $\in \{3e-4, 1e-4, 1e-3\}$
	tau $\in \{5e-3, 1e-2\}$
	batch size $\in \{256, 512\}$
	$\gamma \in [0.9, 1.0]$
	number of gradient steps $\in \{3e5\}$
BC	num modes in gaussian mixture model $\in \{1, 5, 10, 100\}$
	learning rate $\in \{1e-3, 3e-4, 1e-4\}$
	batch size $\in \{256, 512\}$
	number of gradient steps $\in \{2e5\}$

Table 1: Hyperparameters search space.

8.2 Environments, Baselines, and Datasets

For a more precise picture of the high-dimensionality of the problems, below we include the environment configurations table from Qin et al. [25].

Environment	Observation Shape	Action Shape	Have Done	Max Timesteps
IB	182	3	False	1000
FinRL	181	30	False	2516
CL	74	14	False	1000

Table 2: Environment configurations. Taken from Qin et al. [25].

As we discussed in the main text, we reproduced the procedure from Qin et al. [25] to train behavioral policies and generate datasets. We omitted the CityLearn high-level policy since we could not match the performance reported in Qin et al. [25] in a reasonable amount of time. Online performance for the policies used in the paper is in Table 3.

Environment	Low Level	Medium Level	High Level
Industrial Benchmark	-323856	-276379	-234101
CityLearn	27797	32498	-
FinRL	221	294	446

Table 3: Behavioral policies.

8.3 Expected Online Performance for Robotic Tasks

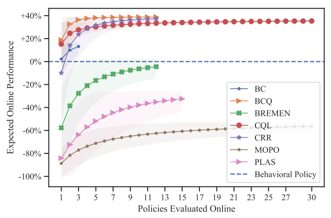
Recently, Qin et al. [25] released an exhaustive set of results that includes online and offline evaluations for different hyperparameter assignments studied in their paper. Although they relied on grid search with varying sizes of the grid depending on the algorithm, we can still build an expected online performance curve under uniform selection. We do so for their robotic set of environments. The resulting curves can be found below in Figures 5, 6, and 7.

In this case, we also find that observations made in the main text hold plausibility: (1) Behavioral Cloning can outperform baseline policy, (2) the best algorithm depends on online evaluation budget, environment, and dataset conditions.

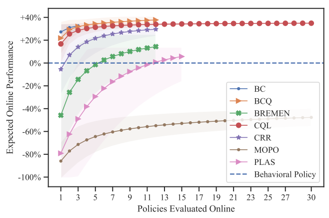
In addition, in this domain, Conservative Q-Learning looks quite dominant if one aims to find a policy that just improves over the baseline with the least possible online deployments.

8.4 Expected Online Performance Under Different Selection Methods

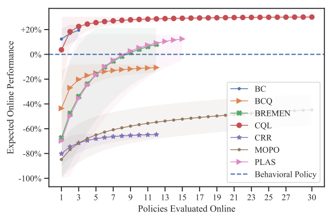
Evaluating each graph separately (Figures 8 - 16), we see that there is almost no certain preference within considered offline policy selection methods due to high variance. But we note that if one would compare mean values only, which we do not consider reliable because of the observed standard deviations, they could conclude that using the Action Difference selection method leads to better policies more often than when using other methods.



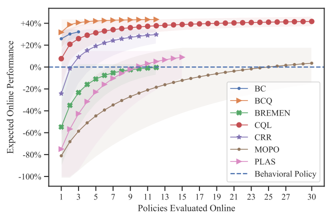
(a) High-Level Policy, 10000



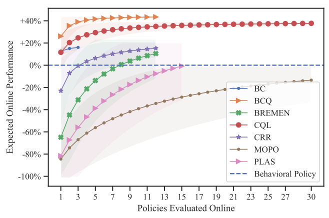
(b) High-Level Policy, 1000



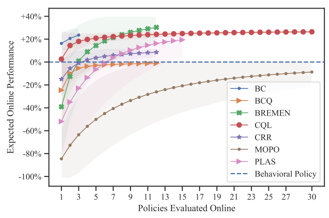
(c) High-Level Policy, 100



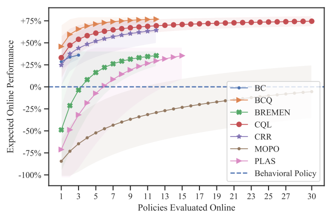
(d) Medium-Level Policy, 10000



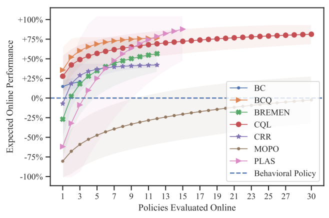
(e) Medium-Level Policy, 1000



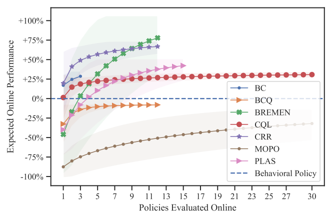
(f) Medium-Level Policy, 100



(g) Low-Level Policy, 10000

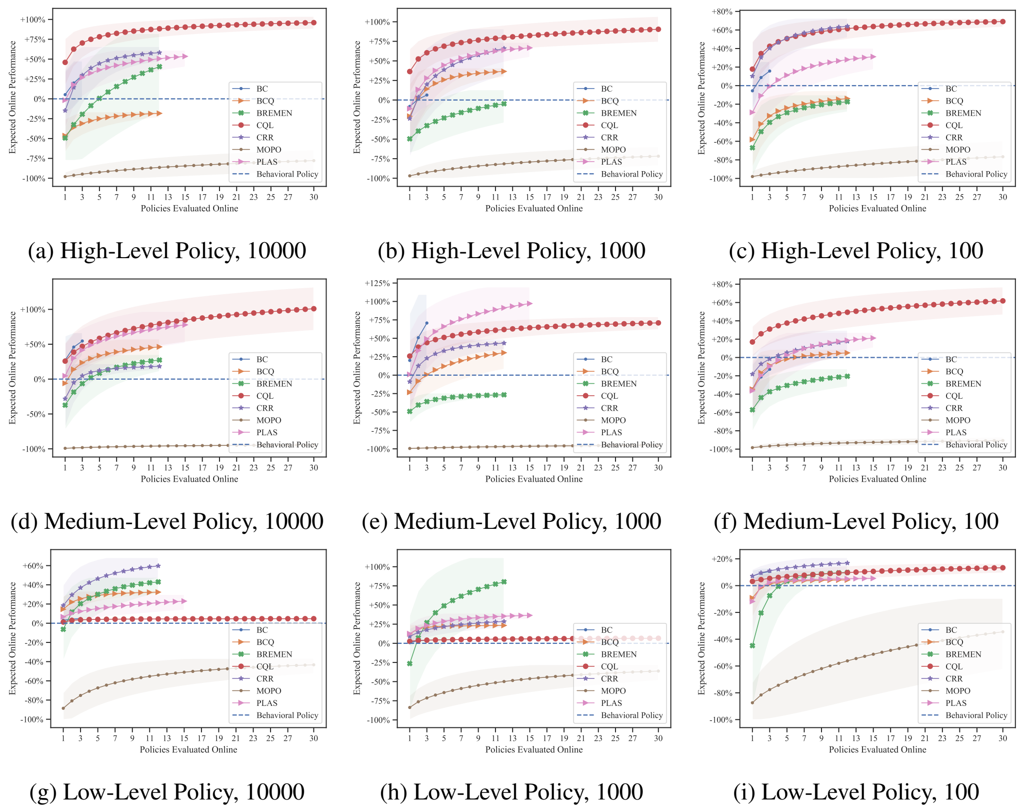


(h) Low-Level Policy, 1000



(i) Low-Level Policy, 100

Figure 5: Expected Online Performance under uniform policy selection. Walker2d-v3, a robotic task from Qin et al. [25], the graphs are computed using their open-sourced online evaluations for different hyperparameter assignments.

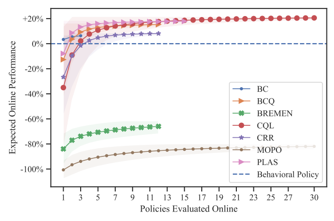


(a) High-Level Policy, 10000 (b) High-Level Policy, 1000 (c) High-Level Policy, 100

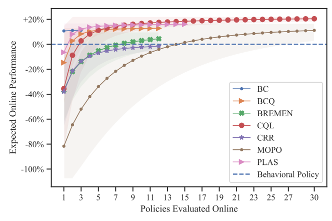
(d) Medium-Level Policy, 10000 (e) Medium-Level Policy, 1000 (f) Medium-Level Policy, 100

(g) Low-Level Policy, 10000 (h) Low-Level Policy, 1000 (i) Low-Level Policy, 100

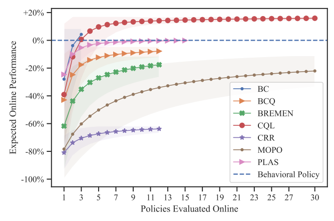
Figure 6: Expected Online Performance under uniform policy selection. Hopper-v3, a robotic task from Qin et al. [25], the graphs are computed using their open-sourced online evaluations for different hyperparameter assignments.



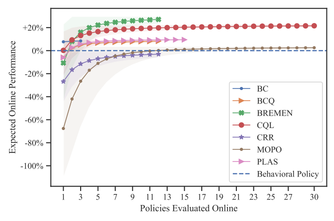
(a) High-Level Policy, 10000



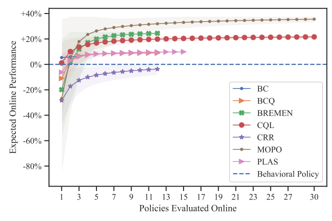
(b) High-Level Policy, 1000



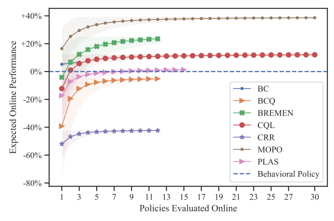
(c) High-Level Policy, 100



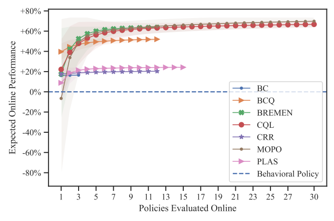
(d) Medium-Level Policy, 10000



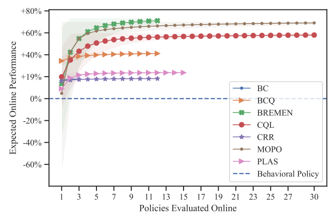
(e) Medium-Level Policy, 1000



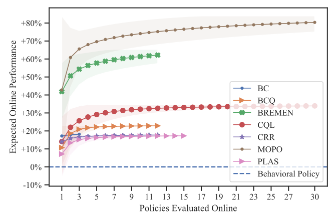
(f) Medium-Level Policy, 100



(g) Low-Level Policy, 10000

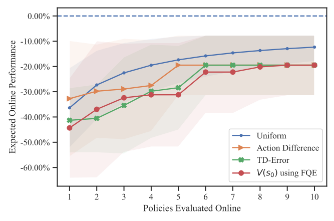


(h) Low-Level Policy, 1000

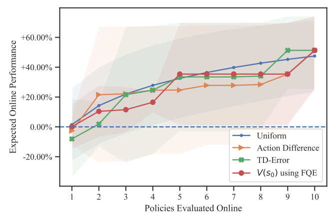


(i) Low-Level Policy, 100

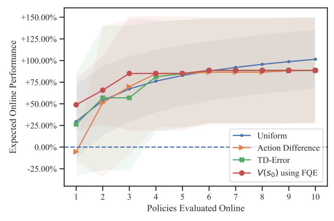
Figure 7: Expected Online Performance under uniform policy selection. HalfCheetah-v3, a robotic task from Qin et al. [25], the graphs are computed using their open-sourced online evaluations for different hyperparameter assignments.



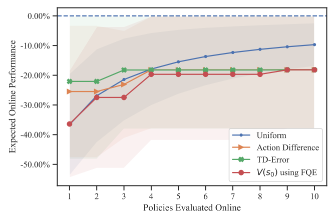
(a) High-Level Policy, 999



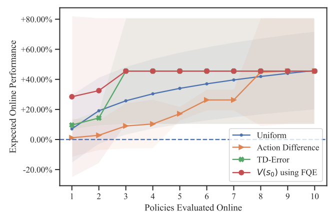
(b) Medium-Level Policy, 999



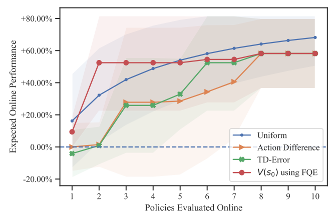
(c) Low-Level Policy, 999



(d) High-Level Policy, 99

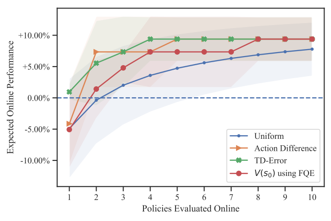


(e) Medium-Level Policy, 99

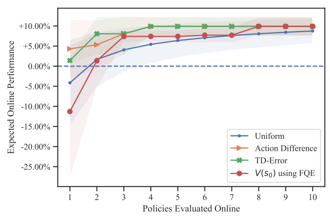


(f) Low-Level Policy, 99

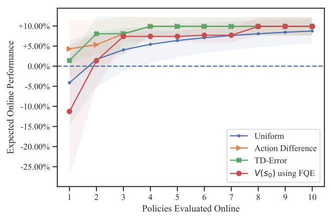
Figure 8: CQL, FinRL. Expected Online Performance under different offline policy selection methods. The shaded area represents one standard deviation



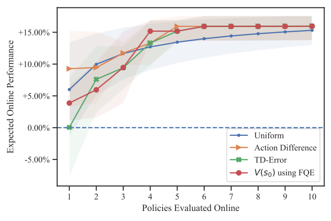
(a) Medium-Level Policy, 9999



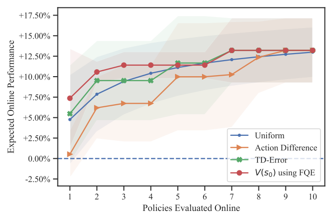
(b) Medium-Level Policy, 999



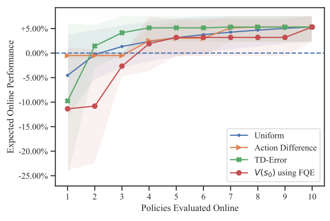
(c) Medium-Level Policy, 99



(d) Low-Level Policy, 9999

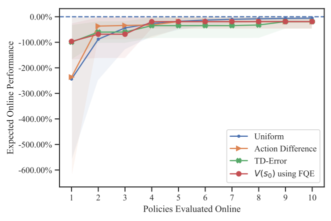


(e) Low-Level Policy, 999

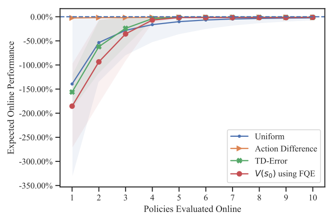


(f) Low-Level Policy, 99

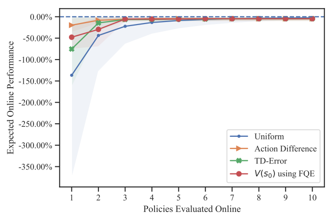
Figure 9: CQL, CityLearn. Expected Online Performance under different offline policy selection methods. The shaded area represents one standard deviation



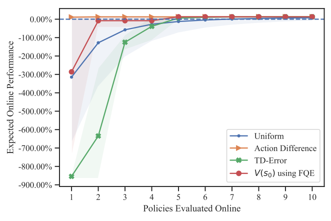
(a) High-Level Policy, 9999



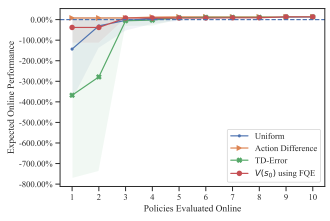
(b) High-Level Policy, 999



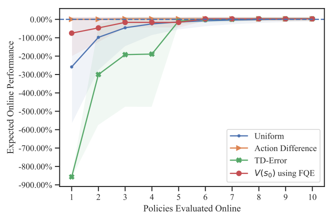
(c) High-Level Policy, 99



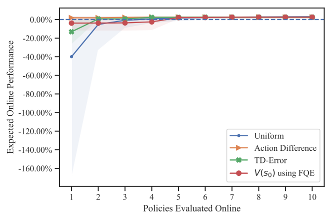
(d) Medium-Level Policy, 9999



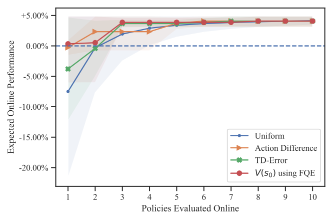
(e) Medium-Level Policy, 999



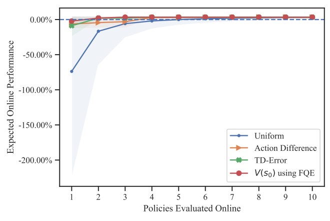
(f) Medium-Level Policy, 99



(g) Low-Level Policy, 9999

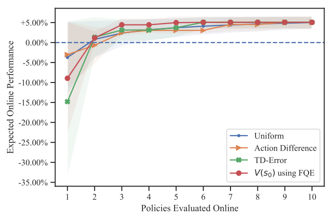


(h) Low-Level Policy, 999

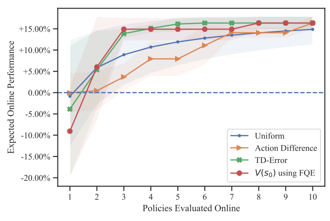


(i) Low-Level Policy, 99

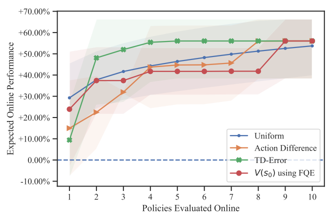
Figure 10: CQL, Industrial Benchmark. Expected Online Performance under different offline policy selection methods. The shaded area represents one standard deviation



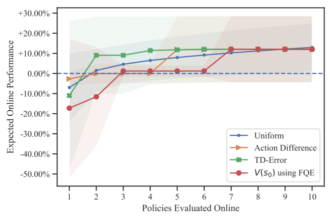
(a) High-Level Policy, 999



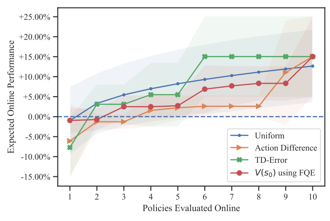
(b) Medium-Level Policy, 999



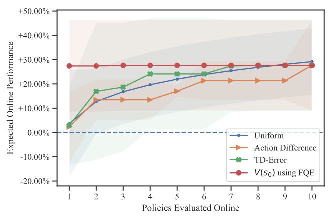
(c) Low-Level Policy, 999



(d) High-Level Policy, 99

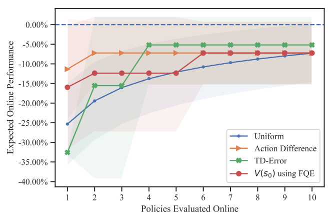


(e) Medium-Level Policy, 99

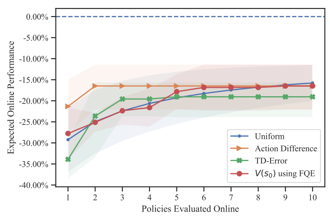


(f) Low-Level Policy, 99

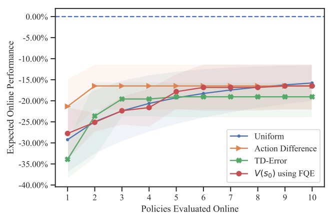
Figure 11: TD3+BC, FinRL. Expected Online Performance under different offline policy selection methods. The shaded area represents one standard deviation



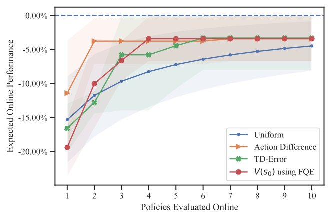
(a) Medium-Level Policy, 9999



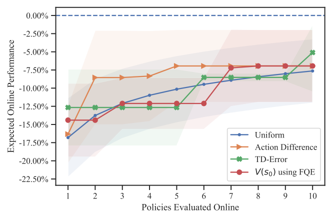
(b) Medium-Level Policy, 999



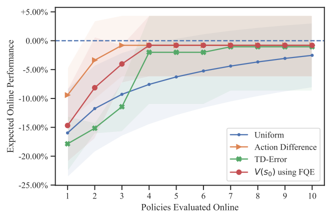
(c) Medium-Level Policy, 99



(d) Low-Level Policy, 9999

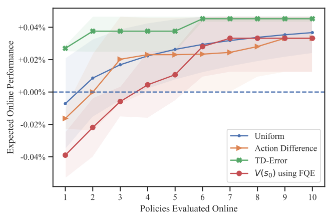


(e) Low-Level Policy, 999

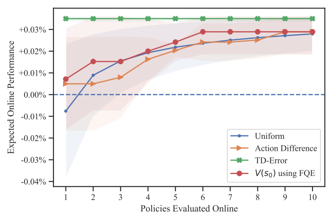


(f) Low-Level Policy, 99

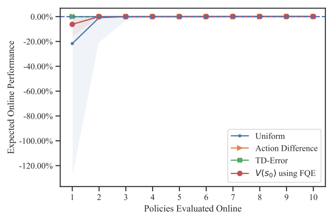
Figure 12: TD3+BC, CityLearn. Expected Online Performance under different offline policy selection methods. The shaded area represents one standard deviation



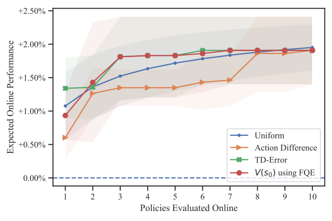
(a) High-Level Policy, 9999



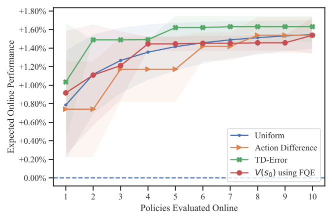
(b) High-Level Policy, 999



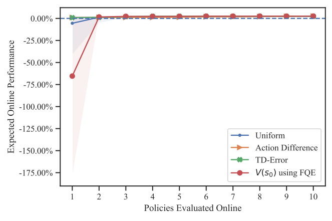
(c) High-Level Policy, 99



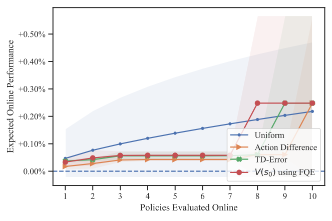
(d) Medium-Level Policy, 9999



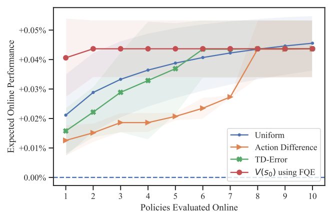
(e) Medium-Level Policy, 999



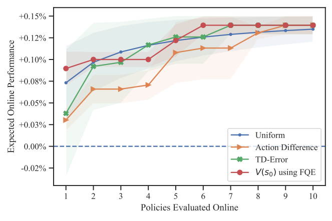
(f) Medium-Level Policy, 99



(g) Low-Level Policy, 9999

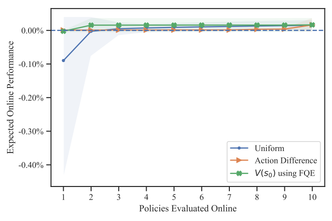


(h) Low-Level Policy, 999

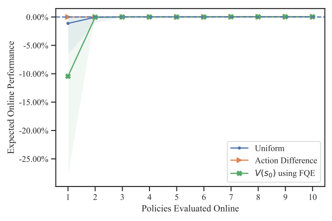


(i) Low-Level Policy, 99

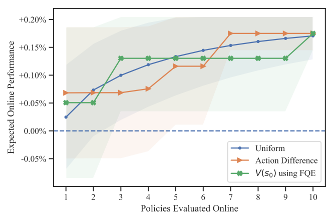
Figure 13: TD3+BC, Industrial Benchmark. Expected Online Performance under different offline policy selection methods. The shaded area represents one standard deviation



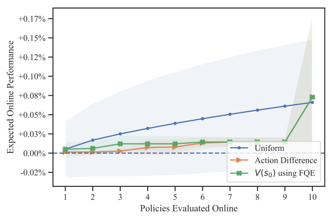
(a) High-Level Policy, 999



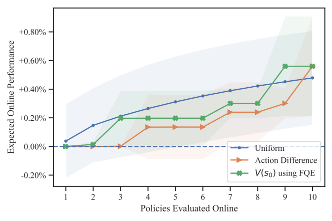
(b) Medium-Level Policy, 999



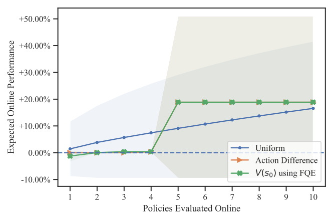
(c) Low-Level Policy, 999



(d) High-Level Policy, 99

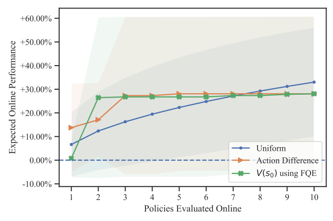


(e) Medium-Level Policy, 99

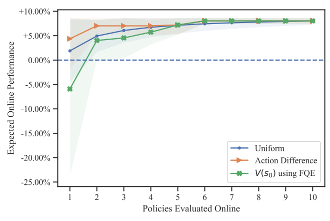


(f) Low-Level Policy, 99

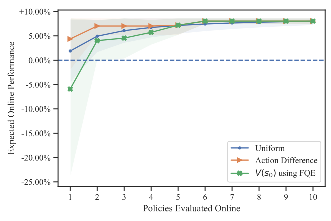
Figure 14: BC, FinRL. Expected Online Performance under different offline policy selection methods. The shaded area represents one standard deviation



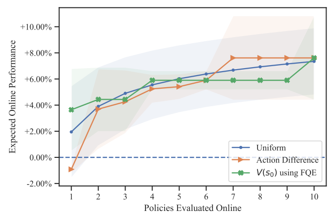
(a) Medium-Level Policy, 9999



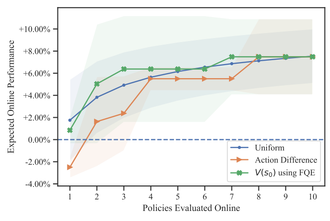
(b) Medium-Level Policy, 999



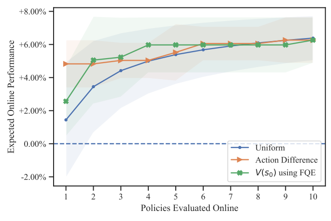
(c) Medium-Level Policy, 99



(d) Low-Level Policy, 9999

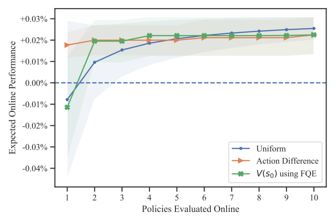


(e) Low-Level Policy, 999

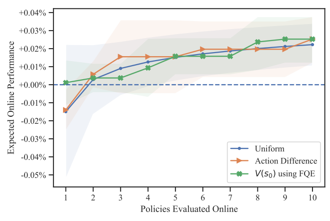


(f) Low-Level Policy, 99

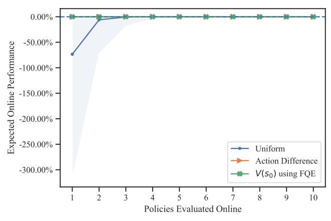
Figure 15: BC, CityLearn. Expected Online Performance under different offline policy selection methods. The shaded area represents one standard deviation



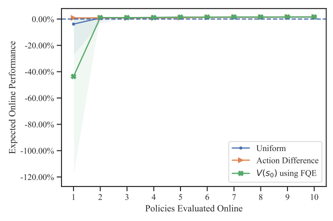
(a) High-Level Policy, 9999



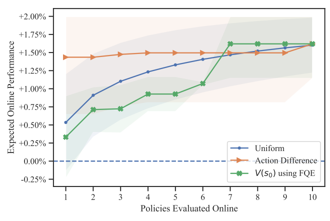
(b) High-Level Policy, 999



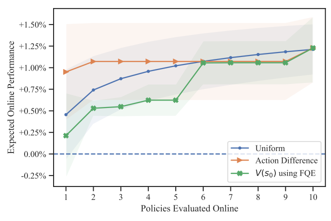
(c) High-Level Policy, 99



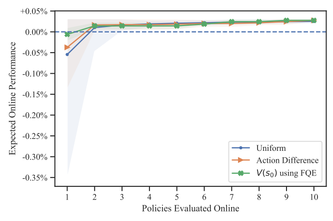
(d) Medium-Level Policy, 9999



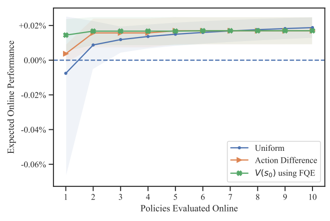
(e) Medium-Level Policy, 999



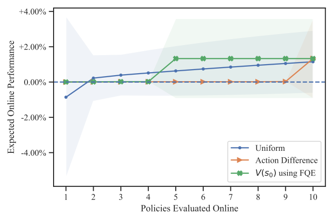
(f) Medium-Level Policy, 99



(g) Low-Level Policy, 9999



(h) Low-Level Policy, 999



(i) Low-Level Policy, 99

Figure 16: BC, Industrial Benchmark. Expected Online Performance under different offline policy selection methods. The shaded area represents one standard deviation