# A Minimalist Approach to Offline Reinforcement Learning

NIP'21 Spotlight Citation: 42

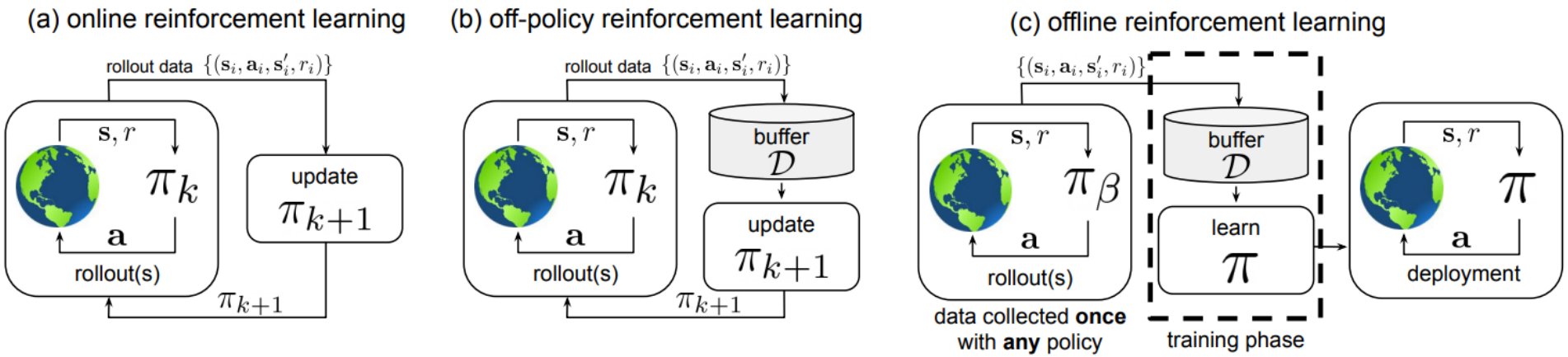
Scott Fujimoto, Shixiang Shane Gu

Mila, McGill University, Google Research, Brain Team

# Motivation

- It's impossible to let RL learn from real environment due to safety consideration.
- The simulator is always different from the real world and it's costly.
- Can we train an agent only from offline dataset without interacting with the environment/simulator?

# Offline RL




We aim to learn a policy $\pi$ from the history of trajectories $\mathcal{D} = \{s_t, a_t, s'_t, r_t\}$ generated by behavioral policy $\pi_\beta$ s.t. the performance $\pi \geq \pi_\beta$(Which means we want to train an agent's policy $\pi$ only from the history of trajectories $\mathcal{D}$)

On the other hand, the imitation learning only mimic the expert policy without reward.

# Offline RL Challenge

- The dataset doesn't cover everything in the environment
- The agent will go crazy(improper Q value) in the unseen state-action.

# Solution

- Let the agent only take the best action that appears in the dataset or close to

Offline RL conquers these issues but yields other issues

- Additional computation cost
- Difficult to implement(includes many minor but matter code-level improvement)
- Instable Performance of Trained Policies

# Issues of Offline RL 1: Additional computation cost

- SAC-N
  - Take the actions that have high Q-value surely, which means avoid action that has high variance Q-value or low Q-value
  - Train $N$ Q-network and update them with the gradient of the most pessimistic network, which is the network gives lowest Q-value.

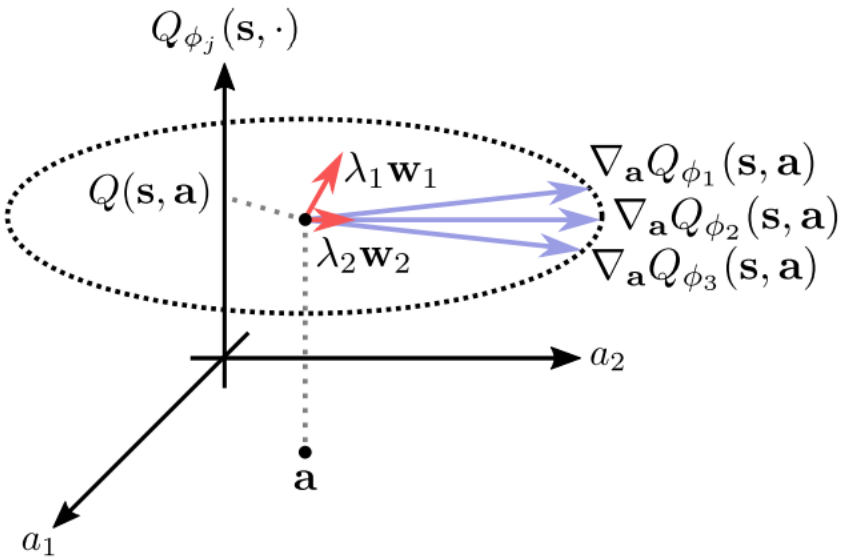
$$\min_{\phi_i} \mathbb{E}_{s,a,s' \sim \mathcal{D}} \left[ \left( Q_{\phi_i}(s,a) - \left( r(s,a) + \gamma \mathbb{E}_{a' \sim \pi_\theta(\cdot|s')} \left[ \min_{j=1,\ldots,N} Q_{\phi'_j}(s',a') - \beta \log \pi_\theta(a' \mid s') \right] \right) \right)^2 \right]$$

$$\max_{\theta} \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_\theta(\cdot|s)} \left[ \min_{j=1,\ldots,N} Q_{\phi_j}(s,a) - \beta \log \pi_\theta(a \mid s) \right]$$

Where $\phi$ is the parameters of the Q-network $Q_\phi$, $\theta$ is the parameters of policy network $\pi_\theta$. The subscript $j$ means the $j$-th network.

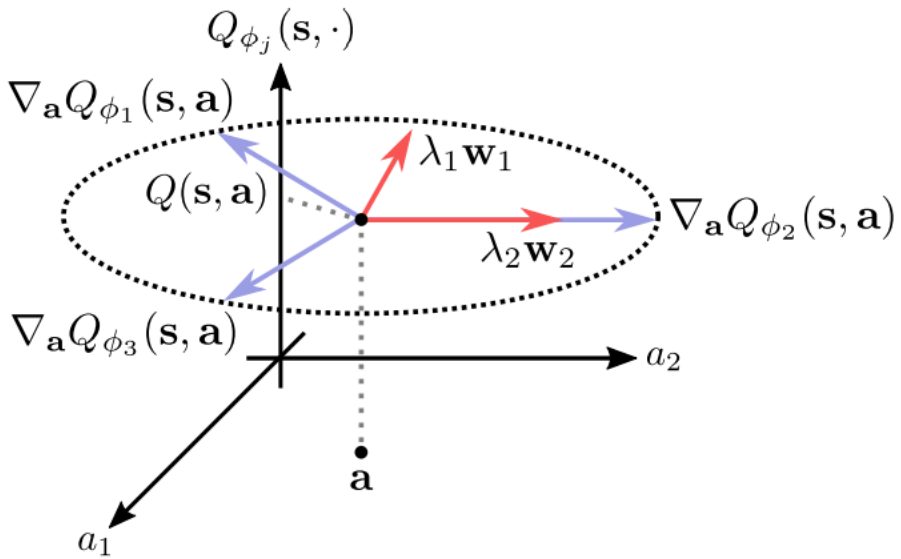
- EDAC
  - But it need hundreds of network. Can we use less network to evaluate the the most pessimistic network.
  - According to the Royston equation, if we enlarge the variance of the $N$ Q-network, we can evaluate the most pessimistic network with less Q-network.
  - Idea: Minimize the similarity of the gradient $\nabla_a Q_{\phi_i}(s,a)$ between $N$ Q-networks $Q_{\phi_i}$.
  - But it still needs tens of Q-networks.

$$\min_{\phi} J_{ES}(Q_\phi) := \mathbb{E}_{s,a\sim\mathcal{D}}\left[\frac{1}{N-1}\sum_{1\le i\neq j\le N}\langle \nabla_a Q_{\phi_i}(s,a), \nabla_a Q_{\phi_j}(s,a)\rangle\right]$$




$\mathrm{Var}(Q_{\phi_j}(\mathbf{s},\mathbf{a}+k\mathbf{w}_2))$ is small so that $\mathbf{a}+k\mathbf{w}_2$ is not sufficiently penalized.

(a) Without ensemble gradient diversification




$\mathrm{Var}(Q_{\phi_j}(\mathbf{s},\mathbf{a}+k\mathbf{w}))$ is large for every direction $\mathbf{w}$ so that all OOD actions are sufficiently penalized.

(b) With ensemble gradient diversification

Figure 3: Illustration of the ensemble gradient diversification. The vector $\lambda_i\mathbf{w}_i$ represents the normalized eigenvector $\mathbf{w}_i$ of $\mathrm{Var}(\nabla_\mathbf{a}Q_{\phi_j}(\mathbf{s},\mathbf{a}))$ multiplied by its eigenvalue $\lambda_i$.

- CQL:
  - Make the expectation of the Q-value of the agent lower
  - Make the expectation of the Q-value of the behavior policy higher

$$\hat{Q}^{k+1} \leftarrow \arg\min_{\hat{Q}^k} \alpha(\mathbb{E}_{s\sim\mathcal{D},a\sim\pi(a|s)}[Q(s,a)] - \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\beta(a|s)}[Q(s,a)]) + \frac{1}{2}\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\left(Q(s,a) - \mathcal{B}\hat{Q}^k(s,a)\right)^2\right]$$

  - The algorithm yields a term $\log\sum_a \exp(Q(s,a))$, which has high variance in the high dimensional action space. It takes time to approximate accurately.

# Issues of Offline RL 2: Difficult to implement

| | CQL [Kumar et al., 2020] | Fisher-BRC [Kostrikov et al., 2021] | TD3+BC (Ours) |
|---|---|---|---|
| Algorithmic Adjustments | Add regularizer to critic†<br>Approximate logsumexp with sampling‡ | Train a generative model†‡<br>Replace critic with offset function<br>Gradient penalty on offset function† | Add a BC term† |
| Implementation Adjustments | Architecture†‡<br>Actor learning rate†<br>Pre-training actor<br>Remove SAC entropy term<br>Max over sampled actions‡ | Architecture†‡<br>Reward bonus†<br>Remove SAC entropy term | Normalize states |

Table 1: Implementation changes offline RL algorithms make to the underlying base RL algorithm. † corresponds to details that add additional hyperparameter(s), and ‡ corresponds to ones that add a computational cost.

Architecture: 2 hidden MLP to 3

Reward Bonus: Additional survival reward

Max over sampled action: Sample 10 actions $a'_1 ... a'_{10}$, and take highest Q-value as target instead of expectation

$$\min_{\phi} \mathbb{E}_{s,a,s' \sim \mathcal{D}} \left[ \left( Q_\phi(s,a) - \left( r(s,a) + \gamma \mathbb{E}_{a' \sim \pi_\theta(\cdot|s')} \left[ Q_\phi\left(s', a'\right) \right] \right) \right)^2 \right]$$

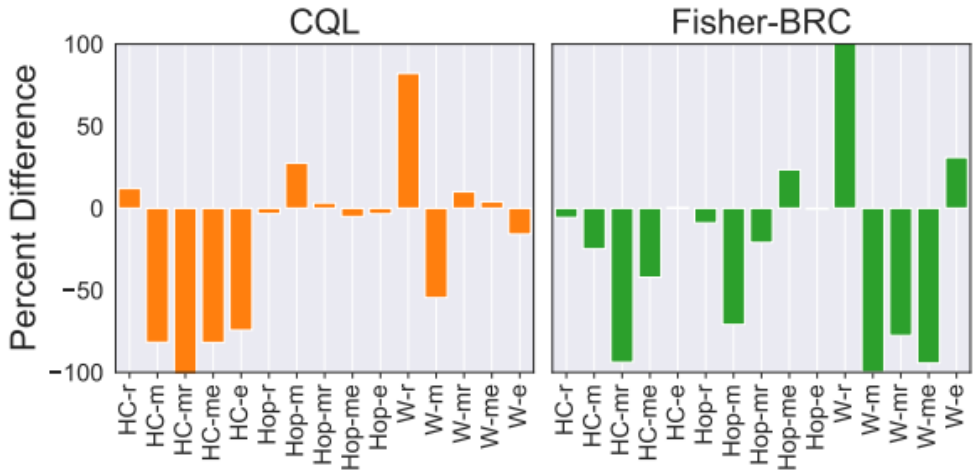



Figure 1: Percent difference of performance of offline RL algorithms and their simplified versions which remove implementation adjustments to their underlying algorithm. HC = HalfCheetah, Hop = Hopper, W = Walker, r = random, m = medium, mr = medium-replay, me = medium-expert, e = expert. Huge drops in performances show that the implementation complexities are crucial for achieving the best results in these prior algorithms.

Performance difference between with and without implementation changes

# Issues of Offline RL 3: Instable Performance of Trained Policies

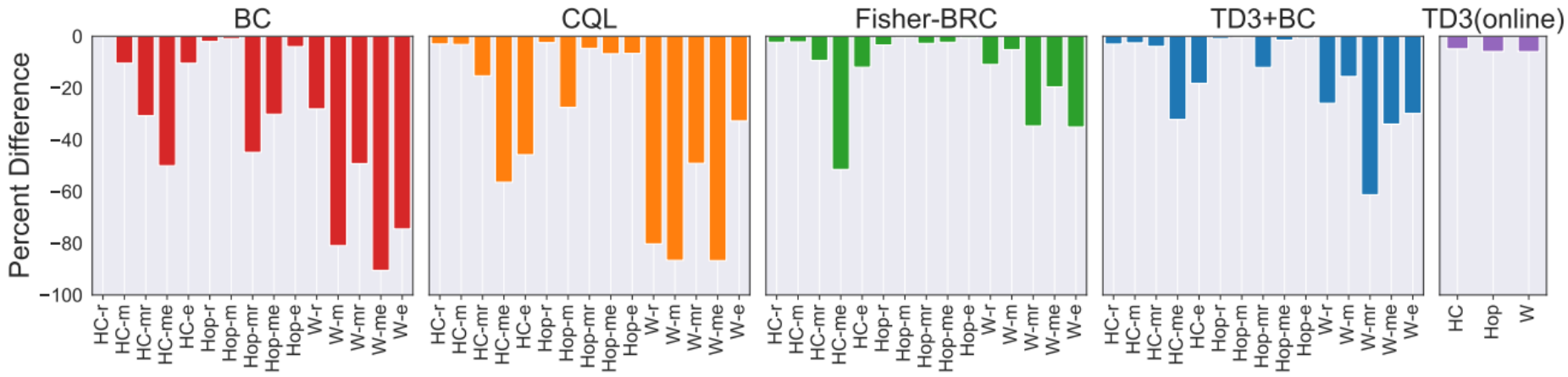



Figure 2: Percent difference of the worst episode during the 10 evaluation episodes at the last evaluation. This measures the deviations in performance at single point in time. HC = HalfCheetah, Hop = Hopper, W = Walker, r = random, m = medium, mr = medium-replay, me = medium-expert, e = expert. While online algorithms (TD3) typically have small episode variances per trained policy (as they should at *convergence*), all offline algorithms have surprisingly high episodic variances for *trained* policies.

# Problem Formulation

**What's the minimalist adjustment to build an offline RL algorithm?**

# Idea

Modify from TD3 algorithm, which is modified from DDPG

**1. Add a *behavior cloning* regularizer**

$$\pi = \arg\max_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\lambda Q(s, \pi(s)) - (\pi(s) - a)^2\right]$$

Where $\lambda$ is a hyperparameter to trade off between optimality and conservativeness. It is determined by

$$\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(s_i,a_i)} |Q(s_i, a_i)|}$$

Where $\alpha = 2.5$ is a constant.

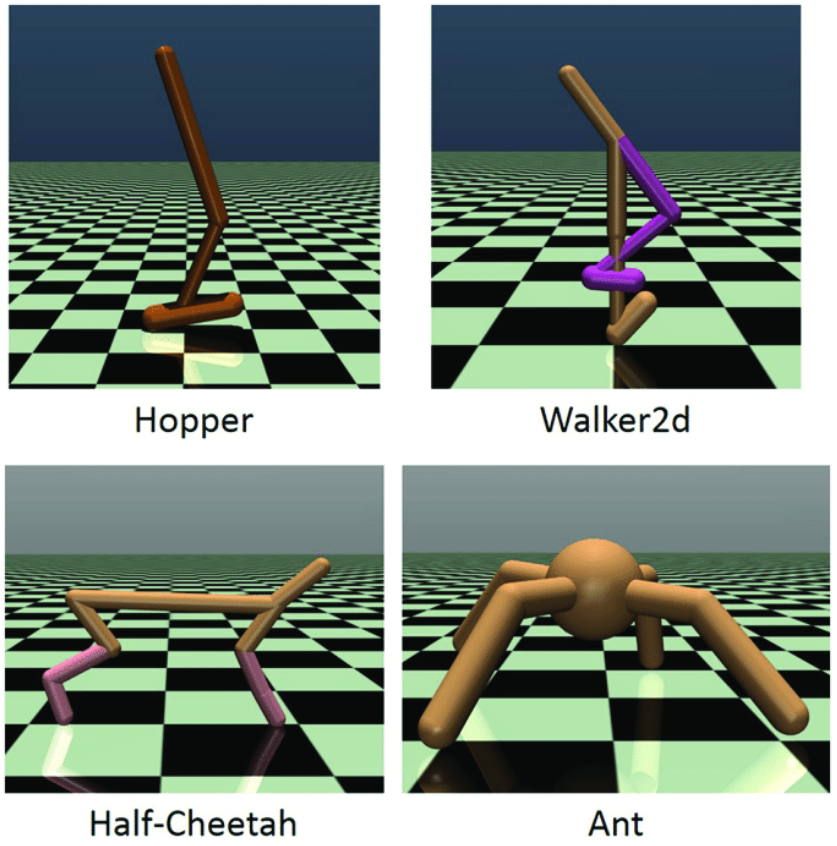
## 2. Normalize the state

$$s_i = \frac{s_i - \mu_i}{\sigma_i + \epsilon}$$

Where $\epsilon$ is a small normalization constant $10^{-3}$.

# Experiment - D4RL Dataset

- expert: a fully trained online expert
- medium: a suboptimal policy with approximately 1/3 the performance of the expert
- medium-expert: a mixture of medium and expert policies
- medium-replay: the replay buffer of a policy trained up to the performance of the medium agent
- full-replay: the final replay buffer of the expert policy
- 1M transitions



# D4RL Benchmark

| | | BC | BRAC-p | AWAC | CQL | Fisher-BRC | TD3+BC |
|---|---|---|---|---|---|---|---|
| Random | HalfCheetah | 2.0 ±0.1 | 23.5 | 2.2 | 21.7 ±0.9 | 32.2 ±2.2 | 10.2 ±1.3 |
| | Hopper | 9.5 ±0.1 | 11.1 | 9.6 | 10.7 ±0.1 | 11.4 ±0.2 | 11.0 ±0.1 |
| | Walker2d | 1.2 ±0.2 | 0.8 | 5.1 | 2.7 ±1.2 | 0.6 ±0.6 | 1.4 ±1.6 |
| Medium | HalfCheetah | 36.6 ±0.6 | 44.0 | 37.4 | 37.2 ±0.3 | 41.3 ±0.5 | 42.8 ±0.3 |
| | Hopper | 30.0 ±0.5 | 31.2 | 72.0 | 44.2 ±10.8 | 99.4 ±0.4 | 99.5 ±1.0 |
| | Walker2d | 11.4 ±6.3 | 72.7 | 30.1 | 57.5 ±8.3 | 79.5 ±1.0 | 79.7 ±1.8 |
| Medium Replay | HalfCheetah | 34.7 ±1.8 | 45.6 | - | 41.9 ±1.1 | 43.3 ±0.9 | 43.3 ±0.5 |
| | Hopper | 19.7 ±5.9 | 0.7 | - | 28.6 ±0.9 | 35.6 ±2.5 | 31.4 ±3.0 |
| | Walker2d | 8.3 ±1.5 | -0.3 | - | 15.8 ±2.6 | 42.6 ±7.0 | 25.2 ±5.1 |
| Medium Expert | HalfCheetah | 67.6 ±13.2 | 43.8 | 36.8 | 27.1 ±3.9 | 96.1 ±9.5 | 97.9 ±4.4 |
| | Hopper | 89.6 ±27.6 | 1.1 | 80.9 | 111.4 ±1.2 | 90.6 ±43.3 | 112.2 ±0.2 |
| | Walker2d | 12.0 ±5.8 | -0.3 | 42.7 | 68.1 ±13.1 | 103.6 ±4.6 | 101.1 ±9.3 |
| Expert | HalfCheetah | 105.2 ±1.7 | 3.8 | 78.5 | 82.4 ±7.4 | 106.8 ±3.0 | 105.7 ±1.9 |
| | Hopper | 111.5 ±1.3 | 6.6 | 85.2 | 111.2 ±2.1 | 112.3 ±0.2 | 112.2 ±0.2 |
| | Walker2d | 56.0 ±24.9 | -0.2 | 57.0 | 103.8 ±7.6 | 79.9 ±32.4 | 105.7 ±2.7 |
| | Total | 595.3 ±91.5 | 284.1 | - | 764.3 ±61.5 | 974.6 ±108.3 | 979.3 ±33.4 |

Table 2: Average normalized score over the final 10 evaluations and 5 seeds. The highest performing scores are highlighted. CQL and Fisher-BRC are re-run using author-provided implementations to ensure an identical evaluation process, while BRAC and AWAC use previously reported results. ± captures the standard deviation over seeds. TD3+BC achieves effectively the same performances as the state-of-the-art Fisher-BRC, despite being much simpler to implement and tune and more than halving the computation cost.

# D4RL Learning Curve

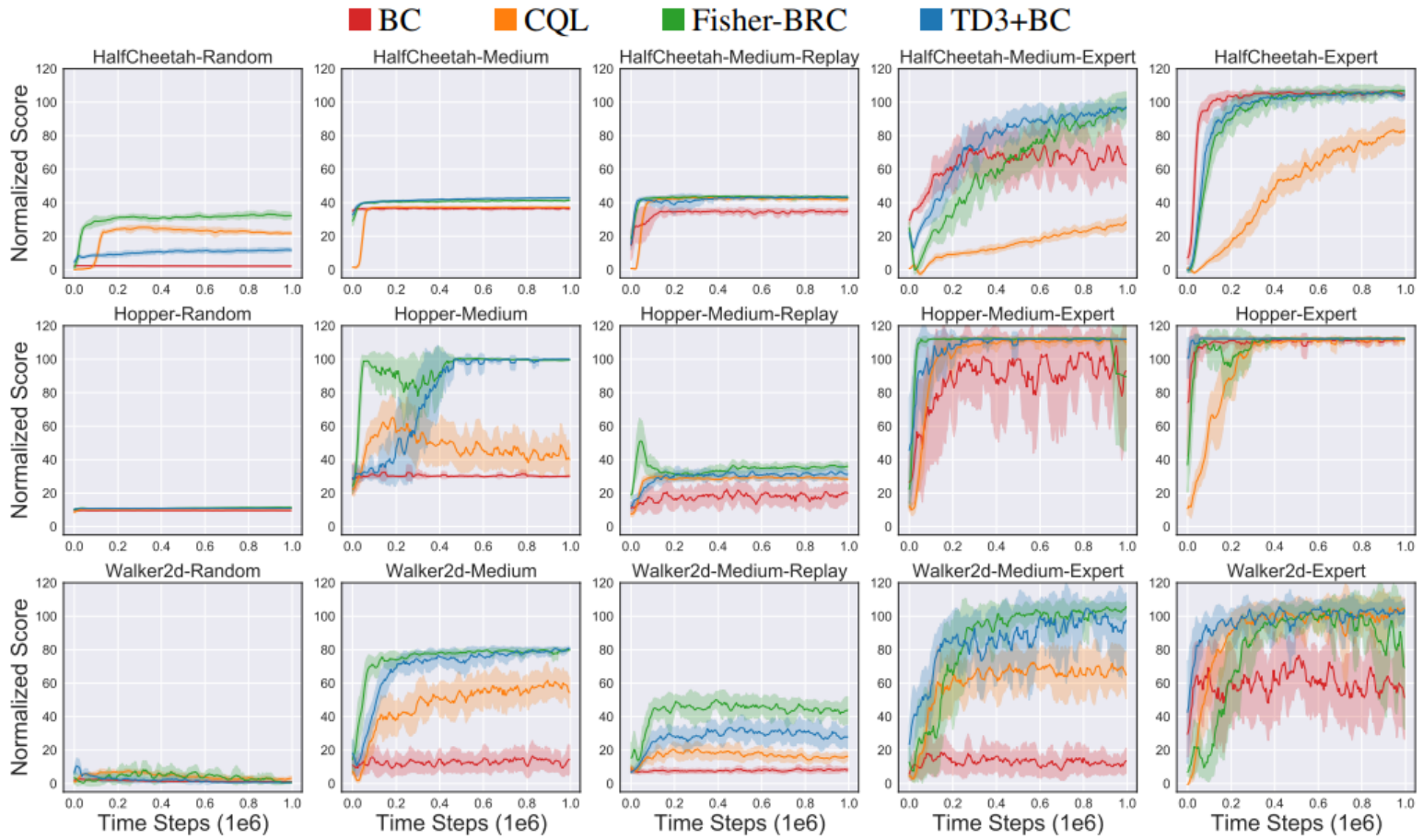



Figure 4: Learning curves comparing the performance of TD3+BC against offline RL baselines in the D4RL datasets. Curves are averaged over 5 seeds, with the shaded area representing the standard deviation across seeds. TD3+BC exhibits a similar learning speed and final performance as the state-of-the-art Fisher-BRC, without the need of pre-training a generative model.

# D4RL 1M Training Steps Runtime

| | CQL (GitHub) | Fisher-BRC (GitHub) | Fisher-BRC (Ours) | TD3+BC (Ours) |
|---|---|---|---|---|
| Implementation | 25m | 39m | 15m | < 1s |
| Algorithmic | 1h 29m | 33m | 58m | < 5s |
| Total | 4h 11m | 2h 12m | 2h 8m | 39m |

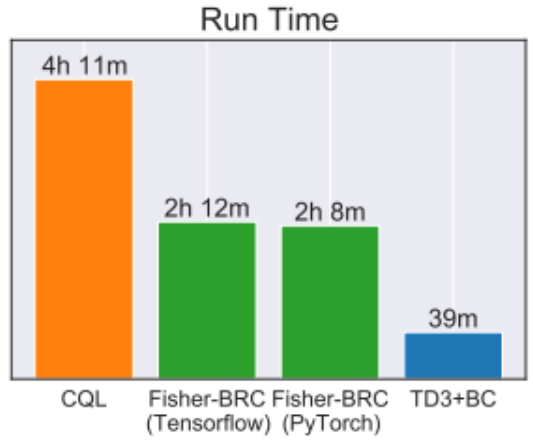



Table 3: Run time comparison of training each offline RL algorithm (does not include evaluation costs). (Left) Breakdown of the cost of the added implementation details (mainly architecture changes) and the algorithmic details by each method. (Right) Total training time of each algorithm. While CQL and Fisher-BRC have significantly increased computational costs over their base online RL algorithm due to various added complexities (e.g. see Table 1), TD3+BC has effectively no increase. This results in less than half of the computational cost of these prior state-of-the-art algorithms.

- GeForce GTX 1080 GPU and an Intel Core i7-6700K CPU at 4.00GHz.
- Train 1M steps
- CQL, TD3+BC, and Fisher-BRC (Ours) are implemented in Pytorch

# D4RL TD3+BC Remove State Normalization

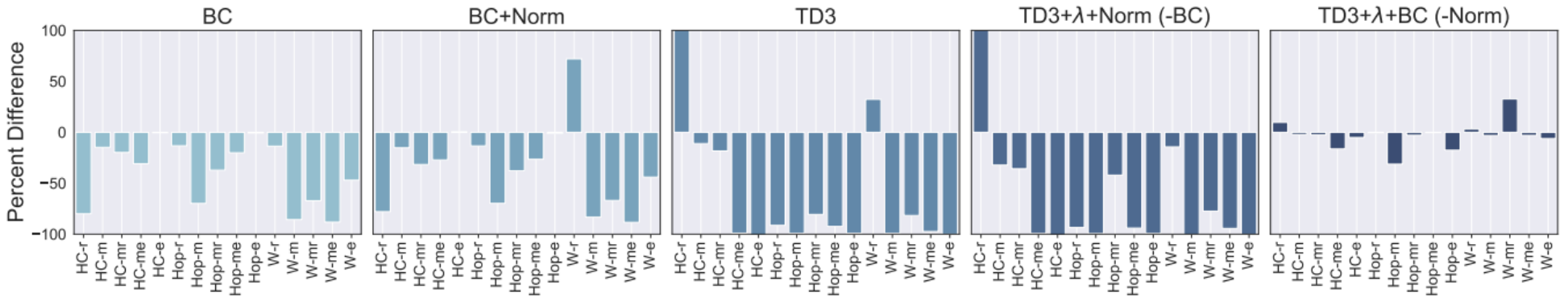



Figure 5: Percent difference of the performance of an ablation of our proposed approach, compared to the full algorithm. TD3+$\lambda$+BC+Norm refers to the complete algorithm, where Norm refers to the state feature normalization. HC = HalfCheetah, Hop = Hopper, W = Walker, r = random, m = medium, mr = medium-replay, me = medium-expert, e = expert. As expected, both BC and TD3 are necessary components to achieve a strong performance. While removing state normalization is not devastating to the performance of the algorithm, we remark it provides a boost in performance across many tasks, while being a straightforward addition.

# Thank You For Listening

Note:

1. High-level Fisher-BRC?
2. What's extrapolation error?
3. What's the max over sampled actions?
4. What's TD3?

# Reference


TD3

- 李宏毅 - Double DQN
- 【强化学习算法 21】TD3
- 强化学习基础 XIII: Twin Delayed DDPG TD3原理与实现
- Addressing Function Approximation Error in Actor-Critic Methods

Fisher-BRC

- [Offline RL]Fisher Divergence Critic Regularization
- Offline Reinforcement Learning with Fisher Divergence Critic Regularization

## CQL

- Offline RL(3): CQL
- 【论文笔记 5】Conservative Q-Learning
- Conservative Q-Learning for Offline Reinforcement Learning

## A Minimalist Approach to Offline Reinforcement Learning

- 离线强化学习(Offline RL)系列3: (算法篇) TD3+BC 算法详解与实现（经验篇）
- A Minimalist Approach to Offline Reinforcement Learning[TD3+BC]阅读笔记