ICLR

Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization

Tatsuya Matsushima[1*], Hiroki Furuta[1*], Yutaka Matsuo[1], Ofir Nachum[2], Shixiang Shane Gu[2]

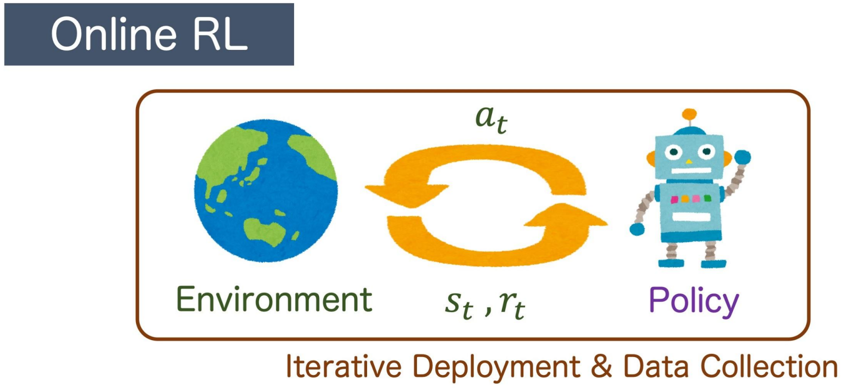
[1]The University of Tokyo, [2]Google Brain (*Contributed Equally)

Contact: matsushima@weblab.t.u-tokyo.ac.jp

Motivation: Reducing Cost & Risks for RL

Impressive success of reinforcement learning (RL) algorithms in sequential decision making

- Depends on frequent data-collection & policy update



Motivation: Reducing Cost & Risks for RL

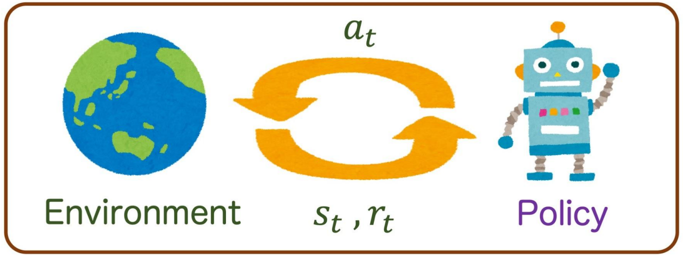
Impressive success of reinforcement learning (RL) algorithms in sequential decision making

- Depends on frequent data-collection & policy update

But, potential risks and costs for deploying new exploratory policies

- e.g. robot control, medicine and education

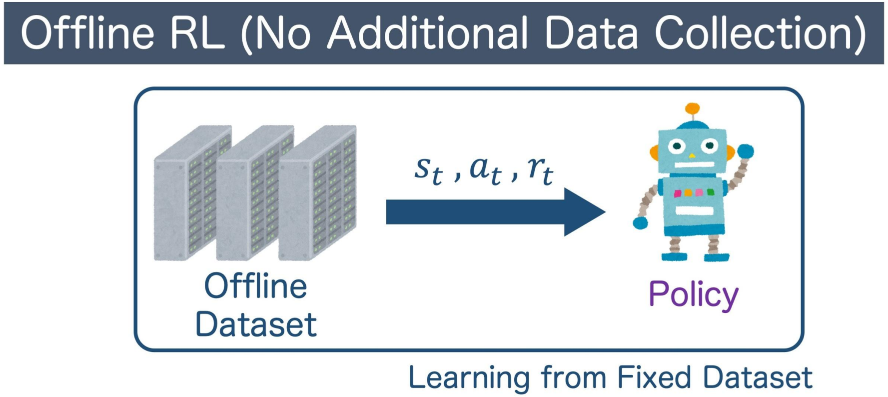
Online RL



Iterative Deployment & Data Collection

Related Framework: Offline RL

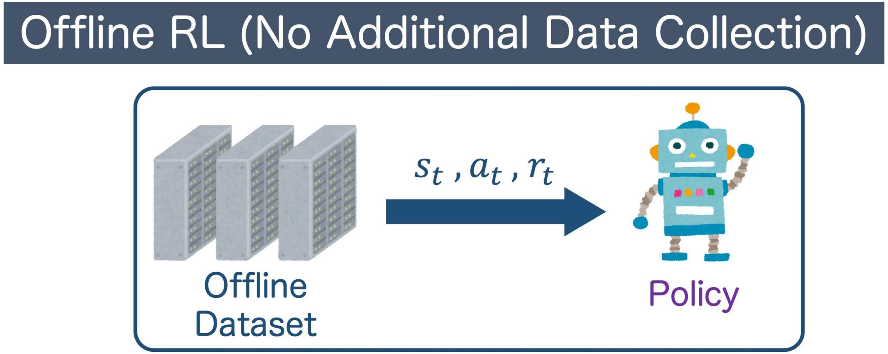
Offline RL learns policies only from a fixed dataset



Related Framework: Offline RL

Offline RL learns policies only from a fixed dataset

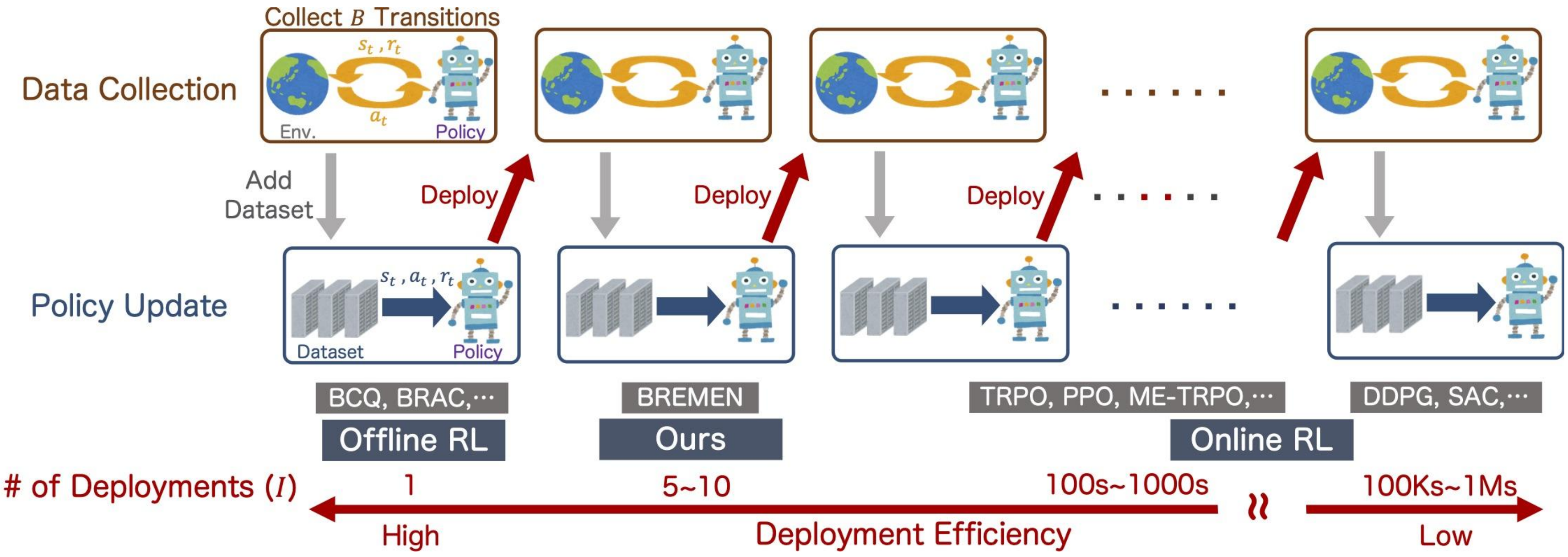
- Assumes we already have some datasets with suboptimal performance
- Usually not learning from scratch (random policy)



Learning from Fixed Dataset

Deployment-Efficiency: A New Metric for RL

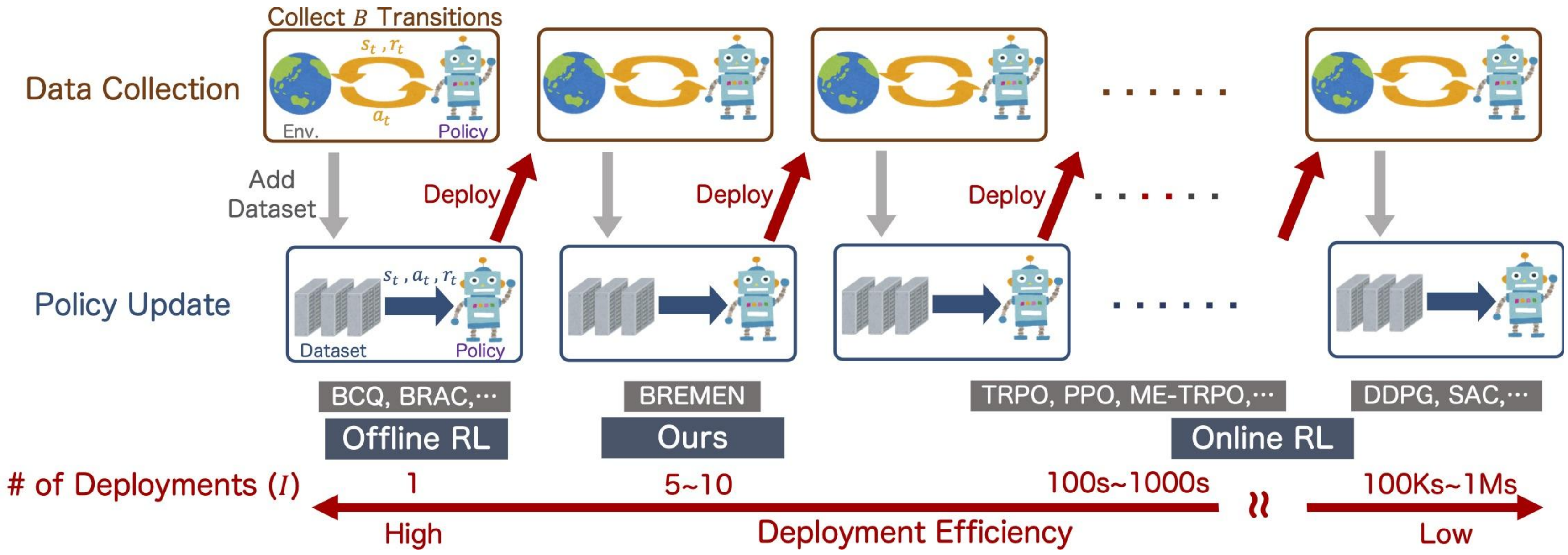
Counts # of “deployments” of the policy



Deployment-Efficiency: A New Metric for RL

Counts # of “deployments” of the policy

- Even algorithms with high sample-efficiency, the deployment-efficiency may be very low e.g. SAC



The Problem & Solution

Problem: Low Sample-Efficiency in previous Offline RL methods

- Simply repeating offline update is not applicable for limited deployment setting (from random policy)

The Problem & Solution

Problem: Low Sample-Efficiency in previous Offline RL methods

- Simply repeating offline update is not applicable for limited deployment setting (from random policy)

Solution: Develop model-based offline RL methods & repeat optimizing it!

- MBRL methods are sample-efficient in online RL

BREMEN (Behaviour-Regularized Model Ensemble)

We propose BREMEN

- A model-based offline RL method
- Achieve high sample- & deployment-efficiency

BREMEN (Behaviour-Regularized Model Ensemble)

We propose BREMEN

- A model-based offline RL method
- Achieve high sample- & deployment-efficiency

Tricks for efficient & stable policy improvement

1. Learning policy with ensembled dynamics model
2. Conservative policy update with behavior policy initialization

Trick 1. Model Ensemble

Learning policy from imaginally rollouts generated by dynamics model ensemble

Trick 1. Model Ensemble

Learning policy from imaginally rollouts generated by dynamics model ensemble

- Prevent policy from exploiting model bias
- Learn K dynamics models with different initialization from dataset with MSE

$$\hat{f}_\phi = \left\{ \hat{f}_{\phi_1}, \ldots, \hat{f}_{\phi_K} \right\} \qquad \min_{\phi_i} \frac{1}{|\mathcal{D}|} \sum_{(s_t, a_t, s_{t+1}) \in \mathcal{D}} \frac{1}{2} \left\| s_{t+1} - \hat{f}_{\phi_i}(s_t, a_t) \right\|_2^2$$

Trick 1. Model Ensemble

Learning policy from imaginally rollouts generated by dynamics model ensemble

- Prevent policy from exploiting model bias
- Learn K dynamics models with different initialization from dataset with MSE

$$\hat{f}_\phi = \left\{ \hat{f}_{\phi_1}, \ldots, \hat{f}_{\phi_K} \right\} \qquad \min_{\phi_i} \frac{1}{|\mathcal{D}|} \sum_{(s_t, a_t, s_{t+1}) \in \mathcal{D}} \frac{1}{2} \left\| s_{t+1} - \hat{f}_{\phi_i}(s_t, a_t) \right\|_2^2$$

- In policy learning, randomly pick up one dynamics model and rollout next state for every step

$$a_t \sim \pi_\theta(\cdot|\hat{s}_t), \quad \hat{s}_{t+1} = \hat{f}_{\phi_i}(\hat{s}_t, a_t) \quad \text{where} \quad i \sim \{1 \cdots K\}$$

Trick 2. Conservative Update with Regularization

Estimate behavior policy from dataset (BC)

$$\min_{\beta} \frac{1}{|\mathcal{D}|} \sum_{(s_t, a_t) \in \mathcal{D}} \frac{1}{2} \left\| a_t - \hat{\pi}_{\beta}\left(s_t\right) \right\|_2^2$$

Trick 2. Conservative Update with Regularization

Estimate behavior policy from dataset (BC)

$$\min_{\beta} \frac{1}{|\mathcal{D}|} \sum_{(s_t, a_t) \in \mathcal{D}} \frac{1}{2} \left\| a_t - \hat{\pi}_{\beta}(s_t) \right\|_2^2$$

KL trust-region policy update initialized with BC policy

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathbb{E}}_{s, a \sim \pi_{\theta_k}, \hat{f}_{\phi_i}} \left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a) \right]$$

$$\text{s.t.} \mathop{\mathbb{E}}_{s \sim \pi_{\theta_k}, \hat{f}_{\phi_i}} \left[D_{\mathrm{KL}} \left(\pi_{\theta}(\cdot|s) \| \pi_{\theta_k}(\cdot|s) \right) \right] \leq \delta, \quad \pi_{\theta_0} = \mathrm{Normal}(\hat{\pi}_{\beta}, 1)$$

Trick 2. Conservative Update with Regularization

Estimate behavior policy from dataset (BC)

$$\min_{\beta} \frac{1}{|\mathcal{D}|} \sum_{(s_t, a_t) \in \mathcal{D}} \frac{1}{2} \left\| a_t - \hat{\pi}_\beta (s_t) \right\|_2^2$$

KL trust-region policy update initialized with BC policy

$$\begin{aligned} \theta_{k+1} = \arg\max_{\theta} & \underset{s, a \sim \pi_{\theta_k}, \hat{f}_{\phi_i}}{\mathbb{E}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a) \right] \\ \text{s.t.} & \underset{s \sim \pi_{\theta_k}, \hat{f}_{\phi_i}}{\mathbb{E}} \left[D_{\mathrm{KL}} \left(\pi_\theta(\cdot|s) \| \pi_{\theta_k}(\cdot|s) \right) \right] \leq \delta, \quad \pi_{\theta_0} = \mathrm{Normal}(\hat{\pi}_\beta, 1) \end{aligned}$$

- Works as an implicit KL regularization, as opposed to explicit penalties to value or immediate reward in previous works

Overview of BREMEN

In limited deployment-setting, recursively apply offline BREMEN procedure

Algorithm 1 BREMEN for Deployment-Efficient RL

Input: Empty dataset $\mathcal{D}_{all}$, $\mathcal{D}$, Initial parameters $\phi = \{\phi_1, \cdots, \phi_K\}$, β, Number of policy optimization T, Number of deployments I.

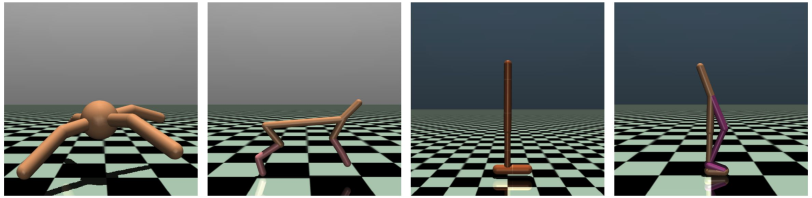
```
Randomly initialize the target policy π_θ.
for deployment i = 1, ··· , I do
    Collect B transitions in the true environment using π_θ and add them to dataset
    D_all ← D_all ∪ {s_t, a_t, r_t, s_{t+1}}, D ← {s_t, a_t, r_t, s_{t+1}}.
    Train K dynamics models f̂_φ using D_all via Equation 1.
    Train estimated behavior policy π̂_β using D by behavior cloning via Equation 3.
    Re-initialize target policy π_θ0 = Normal(π̂_β, 1).        Trick 2. Initialization w/ BC policy
    for policy optimization k = 1, ··· , T do
        Generate imaginary rollout via Equation 2.             Trick 1. Rollout from ensembles
        Optimize target policy π_θ satisfying Equation 4 with the rollout.
```

Experiments

Benchmarking Offline RL

Learn policies from fixed datasets of 1M steps with certain cumulative reward

- Same experimental protocol as the previous work [Wu+19]



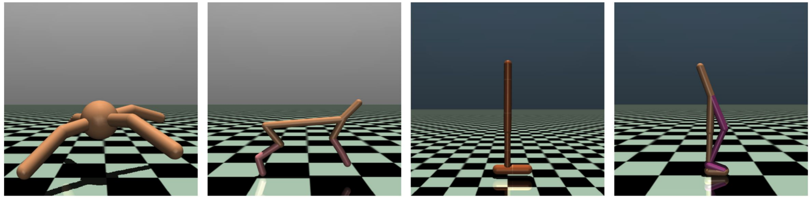
(a) Ant (b) HalfCheetah (c) Hopper (d) Walker2d

Benchmarking Offline RL

Learn policies from fixed datasets of 1M steps with certain cumulative reward

- Same experimental protocol as the previous work [Wu+19]

Achieves competitive with SoTA model-free algorithms in locomotion tasks



(a) Ant (b) HalfCheetah (c) Hopper (d) Walker2d

1,000,000 (1M) transitions				
Method	**Ant**	**HalfCheetah**	**Hopper**	**Walker2d**
Dataset	1191	4126	1128	1376
BC	1321±141	4281±12	1341±161	1421±147
BCQ	2021±31	5783±272	1130±127	2153±753
BRAC	2072±285	7192±115	1422±90	2239±1124
BRAC (max Q)	2369±234	7320±91	1916±343	**2409±1210**
BREMEN (Ours)	**3328±275**	**8055±103**	**2058±852**	2346±230
ME-TRPO (offline)	1258±550	1804±924	518±91	211±154

Benchmarking Offline RL (D4RL)

Works comparably well with other model-free/model-based offline methods in more recent D4RL benchmarks [Fu+20]

Benchmarking Offline RL (D4RL)

Works comparably well with other model-free/model-based offline methods in more recent D4RL benchmarks [Fu+20]

- n.b. Scores are normalized by the expert performance of each datasets

Task Name	BC	BREMEN	MOPO	CQL	BEAR	BRAC-v	AWR	BCQ
halfcheetah-random	2.1	**36.9**	31.9	35.4	25.1	31.2	2.5	2.2
walker2d-random	1.6	3.7	**13.0**	7.0	7.3	1.9	1.5	4.9
hopper-random	9.8	12.2	**13.3**	10.8	11.4	12.2	10.2	10.6
halfcheetah-medium	36.1	**55.0**	40.2	44.4	41.7	46.3	37.4	40.7
walker2d-medium	6.6	59.6	14.0	79.2	59.1	**81.1**	17.4	53.1
hopper-medium	29.0	**69.3**	26.5	58.0	52.1	31.1	35.9	54.5
halfcheetah-medium-replay	38.4	47.2	**54.0**	46.2	38.6	47.7	40.3	38.2
walker2d-medium-replay	11.3	7.6	**42.7**	26.7	19.2	0.9	15.5	15.0
hopper-medium-replay	11.8	24.1	**92.5**	48.6	33.7	0.6	28.4	33.1
halfcheetah-medium-expert	35.8	53.3	57.9	**62.4**	53.4	41.9	52.7	64.7
walker2d-medium-expert	6.4	55.2	55.0	**98.7**	40.1	81.6	53.8	57.5
hopper-medium-expert	**111.9**	64.6	51.7	111.0	96.3	0.8	27.1	110.9

Sample-Efficiency in Offline RL

Works well with 10-20x smaller datasets!

100,000 (100K) transitions				
Method	**Ant**	**HalfCheetah**	**Hopper**	**Walker2d**
Dataset	1191	4066	1128	1376
BC	1330±81	4266±21	1322±109	1426±47
BCQ	1363±199	3915±411	1129±238	**2187±196**
BRAC	-157±383	2505±2501	1310±70	2162±1109
BRAC (max Q)	-226±387	2332±2422	1422±101	2164±1114
BREMEN (Ours)	**1633±127**	**6095±370**	**2191±455**	2132±301
ME-TRPO (offline)	974±4	2±434	307±170	10±61

50,000 (50K) transitions				
Method	**Ant**	**HalfCheetah**	**Hopper**	**Walker2d**
Dataset	1191	4138	1128	1376
BC	1270±65	4230±49	1249±61	1420±194
BCQ	1329±95	1319±626	1178±235	1841±439
BRAC	-878±244	-597±73	1277±102	976±1207
BRAC (max Q)	-843±279	-590±56	1276±225	903±1137
BREMEN (Ours)	**1347±283**	**5823±146**	**1632±796**	**2280±647**
ME-TRPO (offline)	938±32	-73±95	152±13	176±343

Sample-Efficiency in Offline RL

Works well with 10-20x smaller datasets!

- Previous methods are sometimes unstable and don’t exceed even datasets

100,000 (100K) transitions				
Method	**Ant**	**HalfCheetah**	**Hopper**	**Walker2d**
Dataset	1191	4066	1128	1376
BC	1330±81	4266±21	1322±109	1426±47
BCQ	1363±199	3915±411	1129±238	**2187±196**
BRAC	-157±383	2505±2501	1310±70	2162±1109
BRAC (max Q)	-226±387	2332±2422	1422±101	2164±1114
BREMEN (Ours)	**1633±127**	**6095±370**	**2191±455**	2132±301
ME-TRPO (offline)	974±4	2±434	307±170	10±61
50,000 (50K) transitions				
Method	**Ant**	**HalfCheetah**	**Hopper**	**Walker2d**
Dataset	1191	4138	1128	1376
BC	1270±65	4230±49	1249±61	1420±194
BCQ	1329±95	1319±626	1178±235	1841±439
BRAC	-878±244	-597±73	1277±102	976±1207
BRAC (max Q)	-843±279	-590±56	1276±225	903±1137
BREMEN (Ours)	**1347±283**	**5823±146**	**1632±796**	**2280±647**
ME-TRPO (offline)	938±32	-73±95	152±13	176±343

Sample-Efficiency in Offline RL

Works well with 10-20x smaller datasets!

- Previous methods are sometimes unstable and don’t exceed even datasets

BREMEN is a stable & sample-efficient offline RL method!

100,000 (100K) transitions				
Method	**Ant**	**HalfCheetah**	**Hopper**	**Walker2d**
Dataset	1191	4066	1128	1376
BC	1330±81	4266±21	1322±109	1426±47
BCQ	1363±199	3915±411	1129±238	**2187±196**
BRAC	-157±383	2505±2501	1310±70	2162±1109
BRAC (max Q)	-226±387	2332±2422	1422±101	2164±1114
BREMEN (Ours)	**1633±127**	**6095±370**	**2191±455**	2132±301
ME-TRPO (offline)	974±4	2±434	307±170	10±61
50,000 (50K) transitions				
Method	**Ant**	**HalfCheetah**	**Hopper**	**Walker2d**
Dataset	1191	4138	1128	1376
BC	1270±65	4230±49	1249±61	1420±194
BCQ	1329±95	1319±626	1178±235	1841±439
BRAC	-878±244	-597±73	1277±102	976±1207
BRAC (max Q)	-843±279	-590±56	1276±225	903±1137
BREMEN (Ours)	**1347±283**	**5823±146**	**1632±796**	**2280±647**
ME-TRPO (offline)	938±32	-73±95	152±13	176±343

Deployment-Efficiency

Recursively applying offline RL method

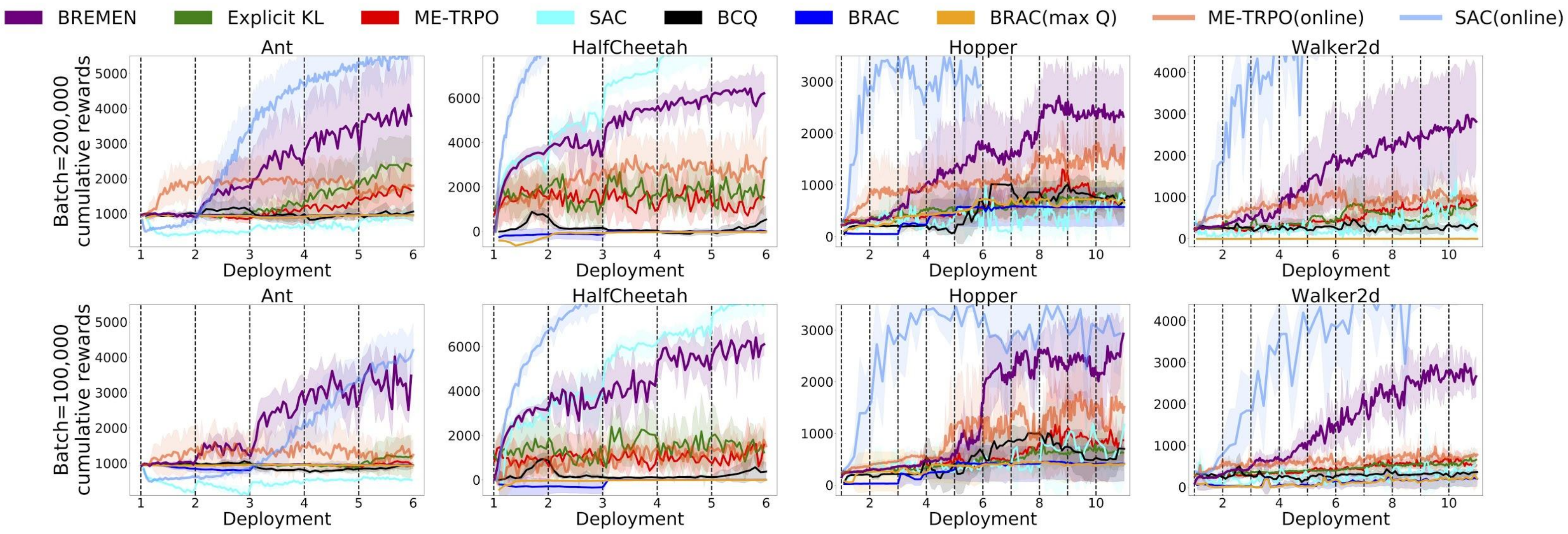
- Online learning from random dataset with deployment limits

Deployment-Efficiency

Recursively applying offline RL method

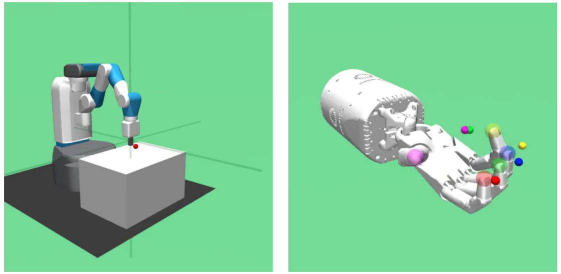
- Online learning from random dataset with deployment limits

BREMEN (purple) achieves remarkable performance in limited-deployment settings

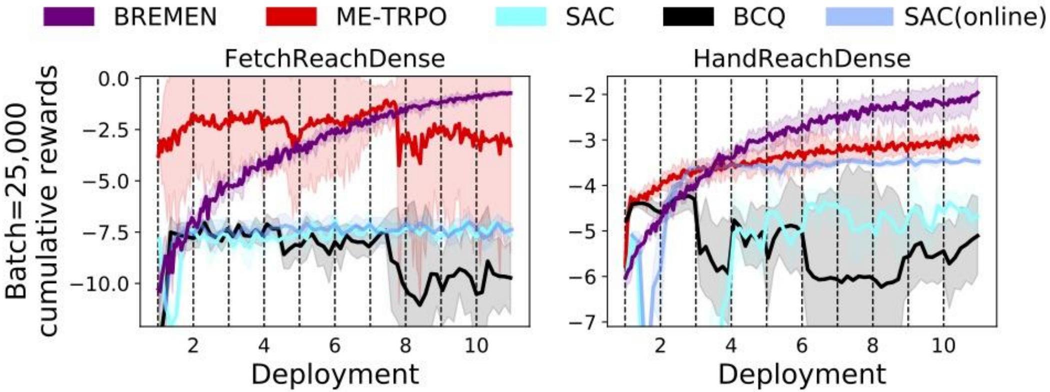


Deployment-Efficiency

BREMEN stably improves the policy also in manipulation tasks



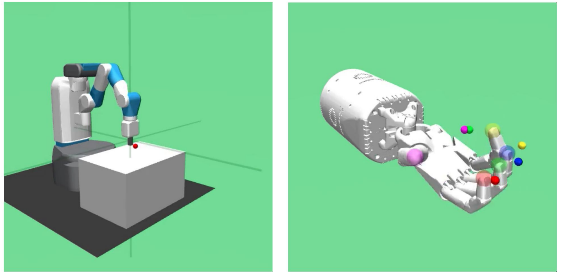
(a) FetchReach (b) HandReach



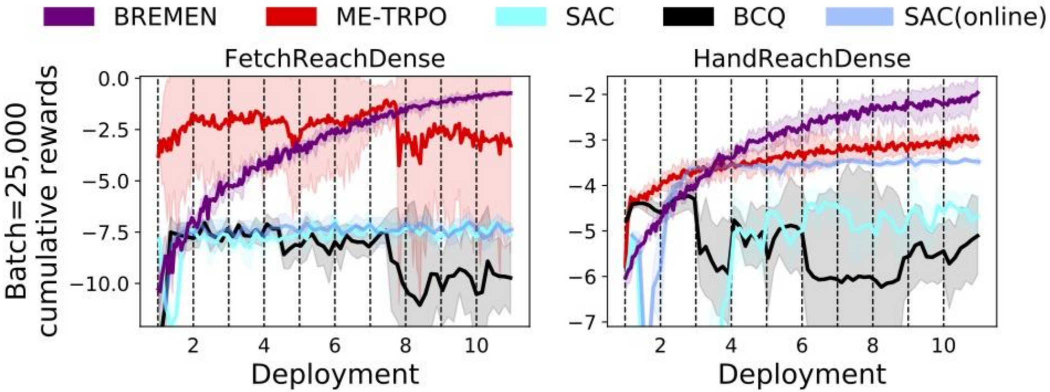
Deployment-Efficiency

BREMEN stably improves the policy also in manipulation tasks

- Satisfies practical requirements in robotics, sample- & deployment-efficiency

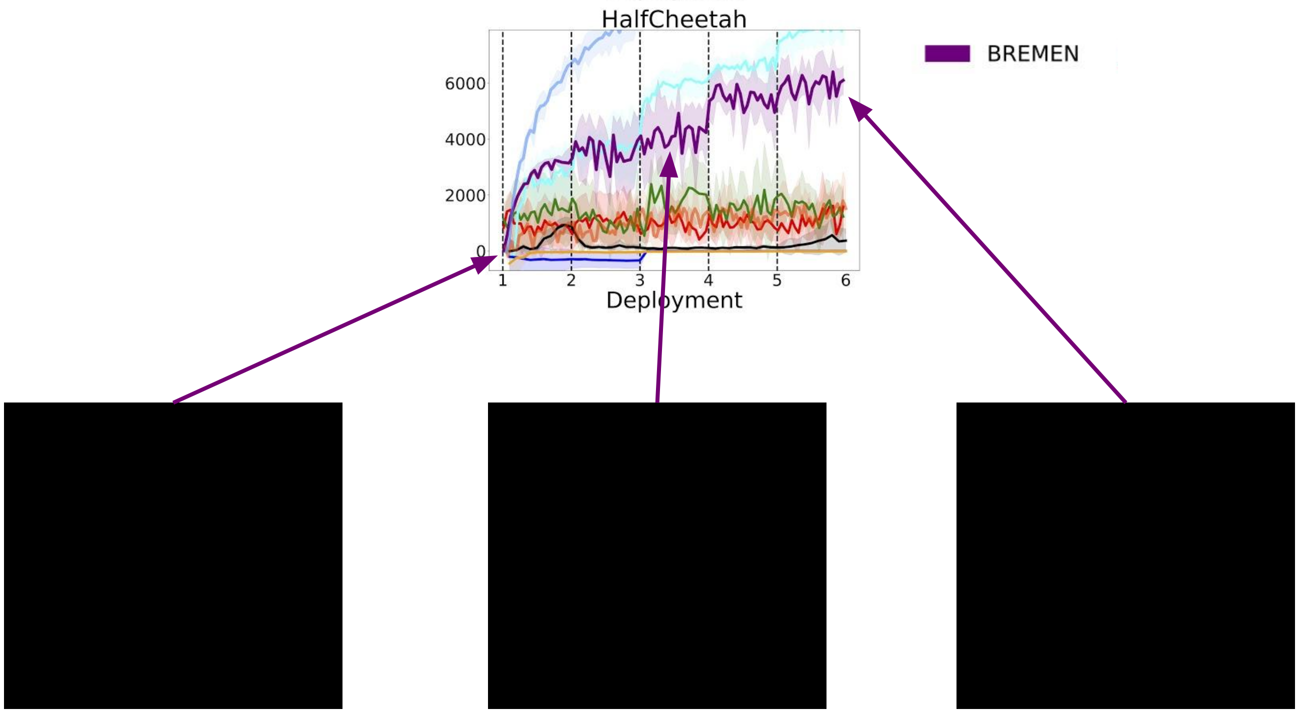


(a) FetchReach (b) HandReach



Qualitative Results

Locomotion (HalfCheetah)



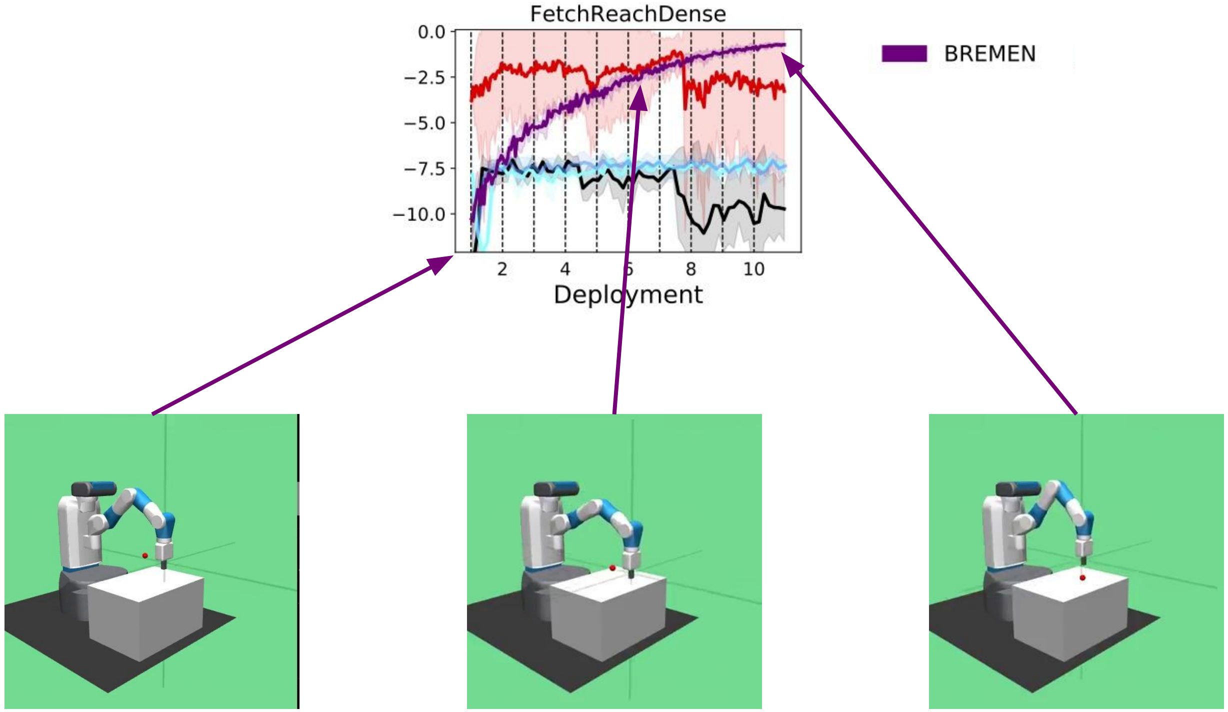
1st Deployment

3rd

5th

Qualitative Results

Manipulation (FetchReach)



1st Deployment

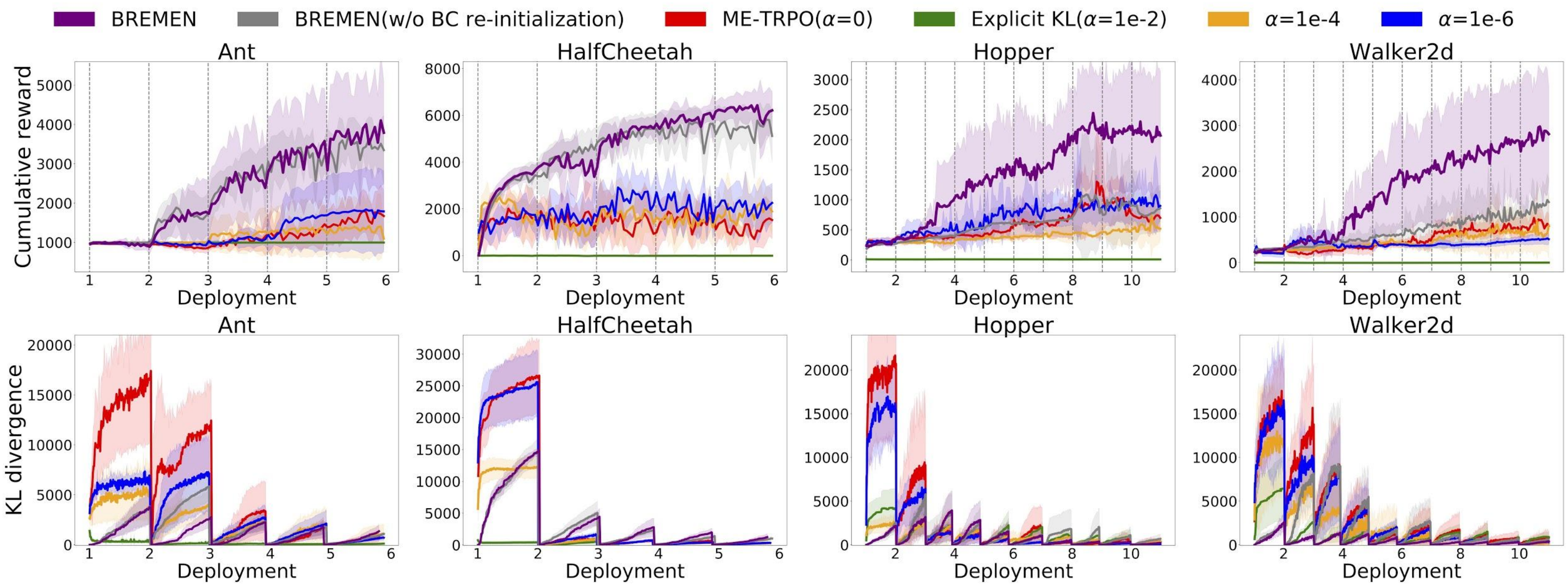
6th

10th

Ablations: Effectiveness of Implicit KL Control

Explicit KL penalty w/o BC initialization moves farther away from last policy

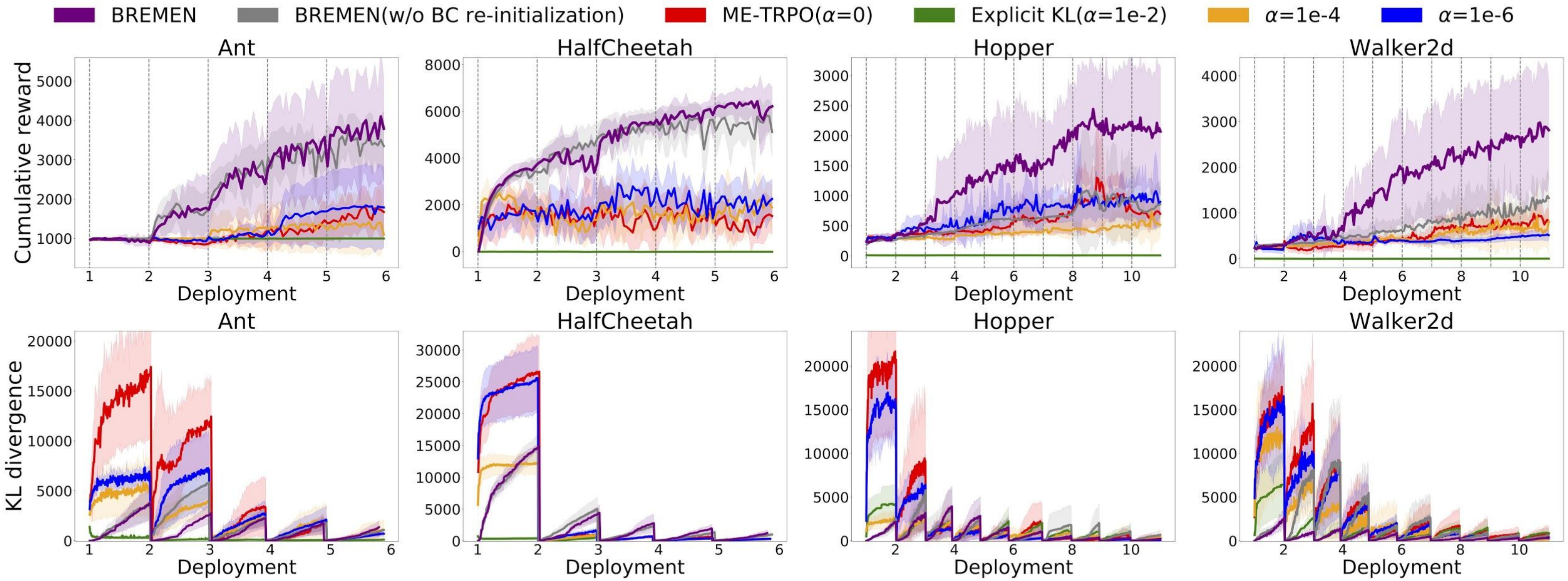
- α is coefficient for explicit KL penalty to the value
- Suggesting the implicit regularization is more effective as conservative update



Ablations: Effectiveness of Implicit KL Control

Explicit KL penalty w/o BC initialization moves farther away from last policy

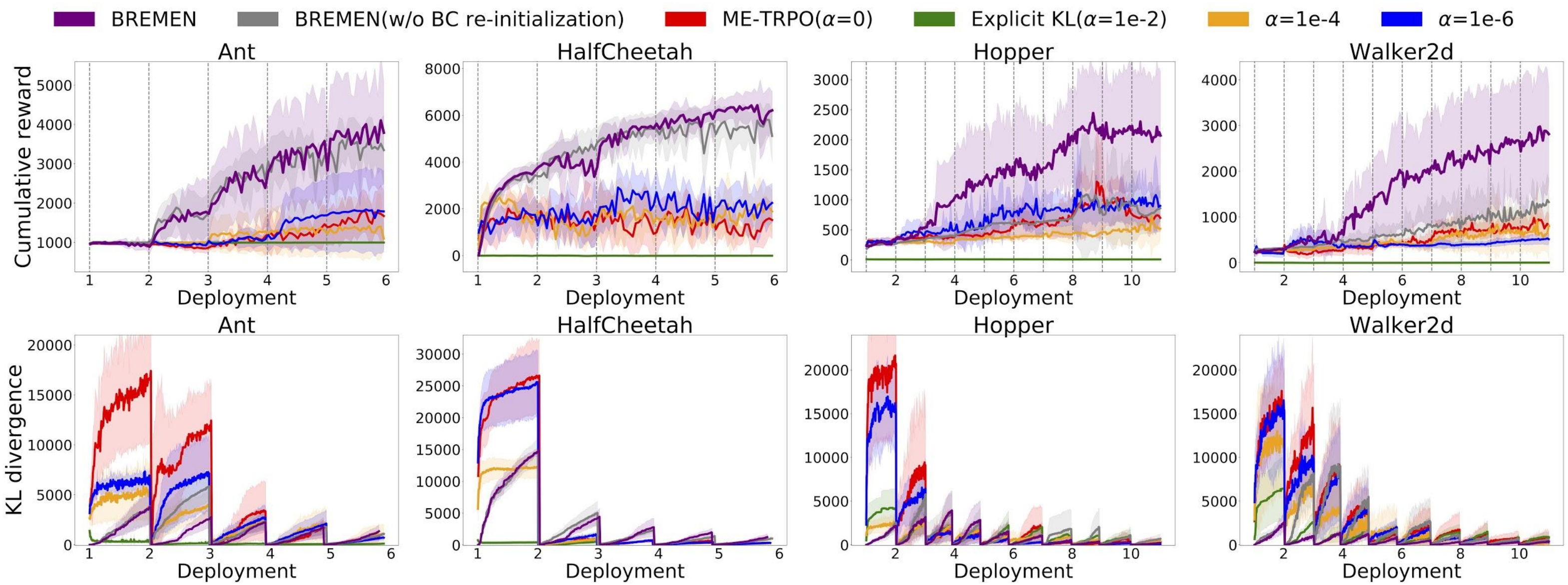
- α is coefficient for explicit KL penalty to the value
- Suggesting the implicit regularization is more effective as conservative update



Ablations: Effectiveness of Implicit KL Control

Explicit KL penalty w/o BC initialization moves farther away from last policy

- α is coefficient for explicit KL penalty to the value
- Suggesting the implicit regularization is more effective as conservative update



Summary

We propose BREMEN

- Model-based offline RL algorithm with high sample-efficiency
- Also achieves high deployment-efficiency

Future directions

- Policy verification for safe & efficient data-collection
- Applying to real robots

Code & pretrained-model