Carnegie Mellon

School of Computer Science

Deep Reinforcement Learning and Control

Truly off policy RL

Fall 2020, CMU 10-703

Katerina Fragkiadaki

On policy versus off policy training

- RL on policy: methods that improve a policy that is used to collect the data used for such improvement
- RL off policy: methods that improve a policy that is not the same with the policy that collected the data used for such improvement

Off-policy RL seen so far

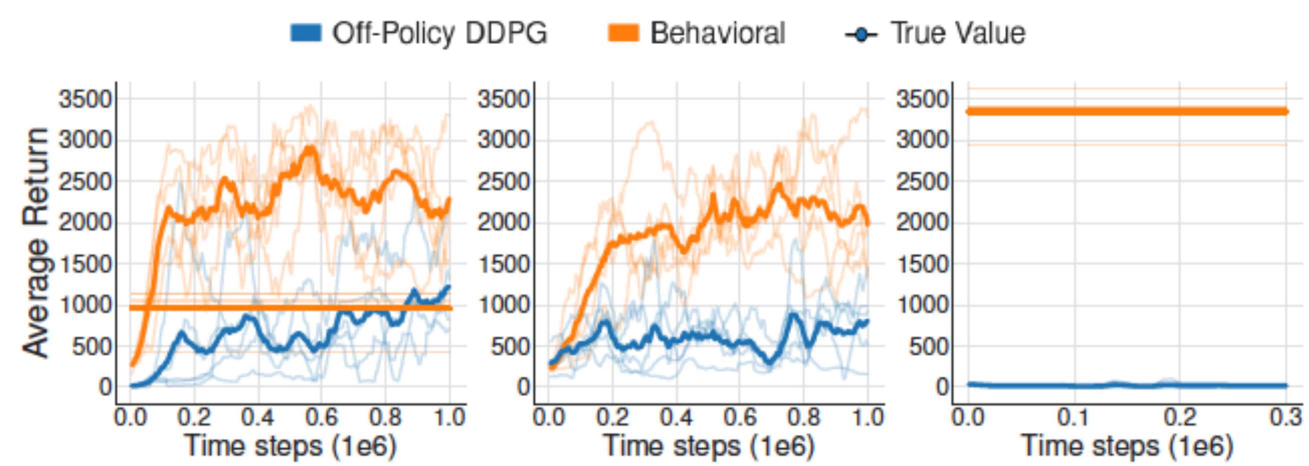
- Off-policy RL learns from data collected under a behavioral policy different than the current policy.
- In what we have seen thus far, “off-policy” transitions are generated from earlier versions of the current policy.
- They are thus heavily correlated to the current policy.
- Not that much *off-policy* after all.

Batch RL

- Batch RL learns from a fixed experience buffer that does not grow with data collected from a near on policy exploratory policy.
- This is truly off-policy RL.
- Q:Who could have provided such an experience buffer?
- A: A set of expert demonstrations for example.

- DDPG (behavioral): (what we have seen in the course) a DDPG policy based on which actions are selected (with small exploration noise) and the experience buffer is populated.
- (Truly) Off-policy DDPG: a DDPG policy that uses experience tuples from the buffer, *it does not influence in any way the data collected in the buffer*

- Final buffer: We train a DDPG agent for 1 million time steps, adding N (0, 0.5) Gaussian noise to actions for high exploration, and store all experienced transitions. This collection procedure creates a dataset with a diverse set of states and actions, with the aim of sufficient coverage.
- Concurrent: We concurrently train the off-policy and behavioral DDPG agents, for 1 million time steps. To ensure sufficient exploration, a standard N (0, 0.1) Gaussian noise is added to actions taken by the behavioral policy. Each transition experienced by the behavioral policy is stored in a buffer replay, which both agents learn from. As a result, both agents are trained with the identical dataset.
- Imitation: A trained DDPG agent acts as an expert, and is used to collect a dataset of 1 million transitions, and populates a buffer, from which the off policy agent learns.



(a) Final buffer performance (b) Concurrent performance (c) Imitation performance

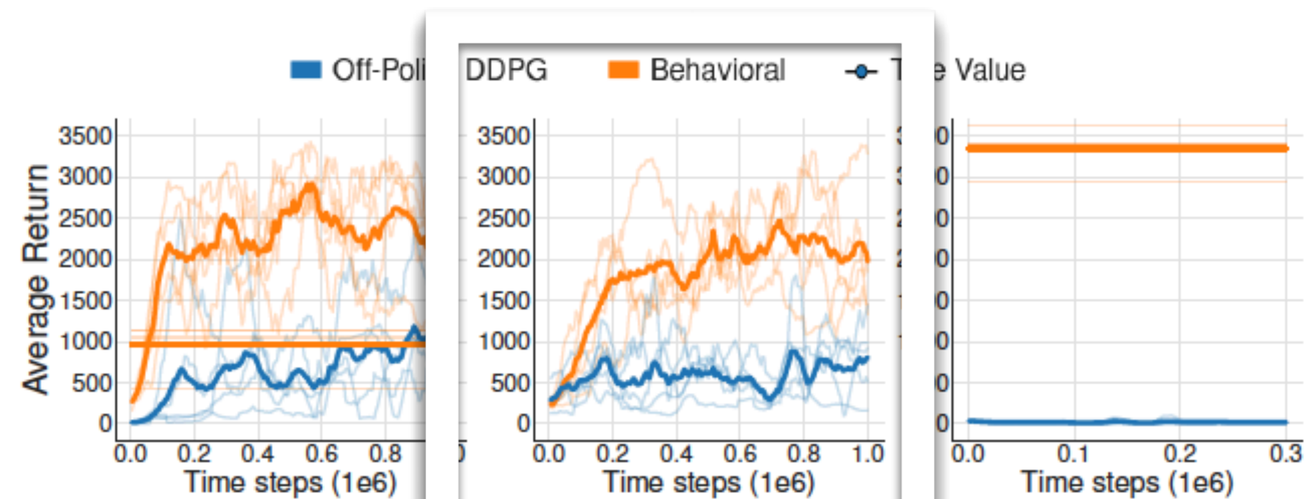
- DDPG (behavioral): (what we have seen in the course) a DDPG policy based on which actions are selected (with small exploration noise) and the experience buffer is populated.

- (Truly) Off-policy DDPG: a DDPG policy that uses experience tuples from the buffer, *it does not influence in any way the data collected in the buffer*

- Final buffer: We train a DDPG agent for 1 million time steps, adding N (0, 0.5) Gaussian noise to actions for high exploration, and store all experienced transitions. This collection procedure creates a dataset with a diverse set of states and actions, with the aim of sufficient coverage.

- Concurrent: We concurrently train the off-policy and behavioral DDPG agents, for 1 million time steps. To ensure sufficient exploration, a standard N (0, 0.1) Gaussian noise is added to actions taken by the behavioral policy. Each transition experienced by the behavioral policy is stored in a buffer replay, which both agents learn from. As a result, both agents are trained with the identical dataset.

- Imitation: A trained DDPG agent acts as an expert, and is used to collect a dataset of 1 million transitions, and populates a buffer, from which the off policy agent learns.

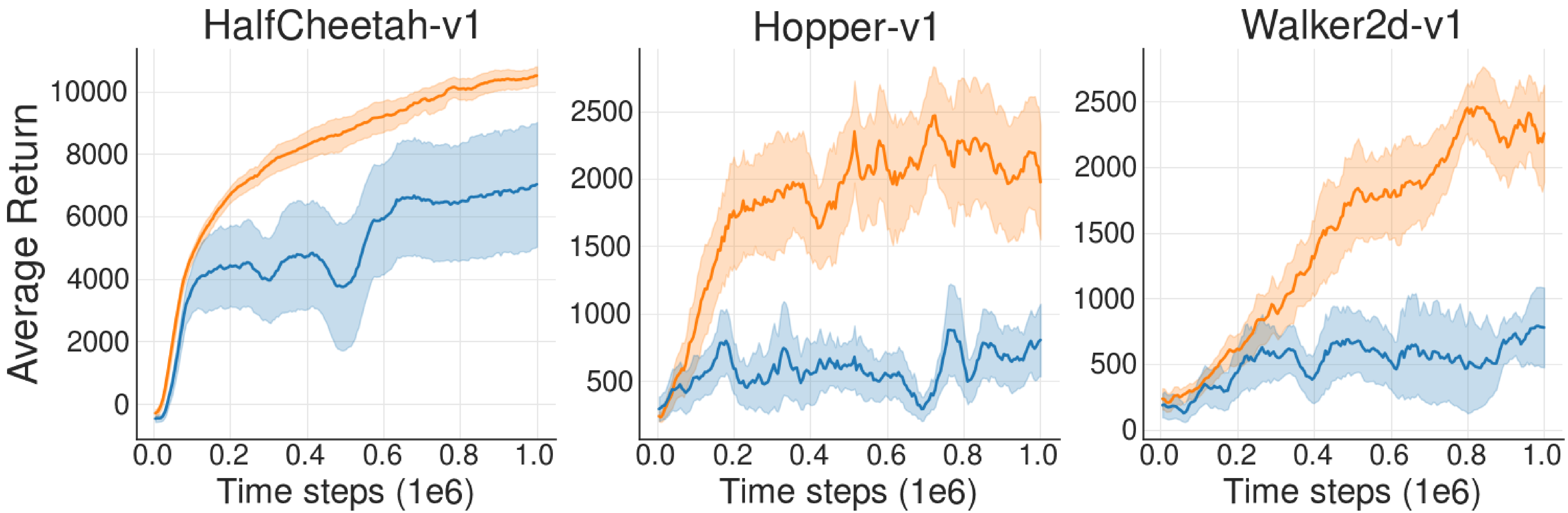


(a) Final buffer performance

(b) Concurrent performance

(c) Imitation performance

Off-Policy Deep Reinforcement Learning without Exploration


HalfCheetah-v1
Hopper-v1
Walker2d-v1
Average Return
Time steps (1e6)
Time steps (1e6)
Time steps (1e6)

Agent orange and agent blue are trained with…

1. The **same off-policy algorithm** (DDPG).

2. The **same dataset.**

The Difference?

1. Agent orange: Interacted with the environment.
    - Standard RL loop.
    - Collect data, store data in buffer, train, repeat.

2. Agent blue: Never interacted with the environment.
    - Trained with data collected by agent orange concurrently.

1. Trained with the same off-policy algorithm.
2. Trained with the same dataset.
3. One interacts with the environment. One doesn't.

Off-policy deep RL fails when **truly off-policy**.

why?

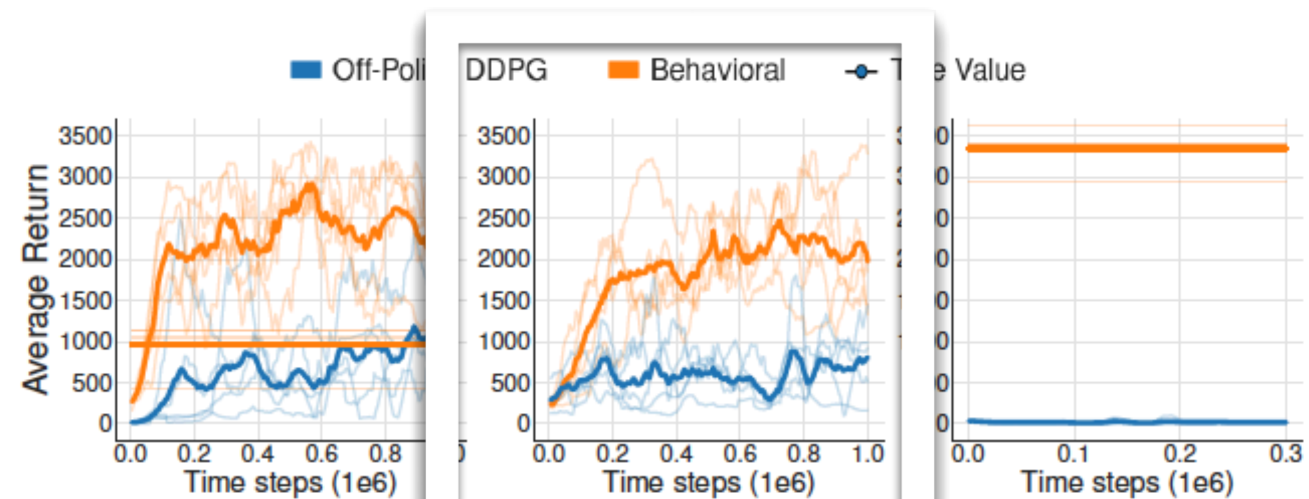
- DDPG (behavioral): (what we have seen in the course) a DDPG policy based on which actions are selected (with small exploration noise) and the experience buffer is populated.

- (Truly) Off-policy DDPG: a DDPG policy that uses experience tuples from the buffer, *it does not influence in any way the data collected in the buffer*

- Final buffer: We train a DDPG agent for 1 million time steps, adding N (0, 0.5) Gaussian noise to actions for high exploration, and store all experienced transitions. This collection procedure creates a dataset with a diverse set of states and actions, with the aim of sufficient coverage.

- Concurrent: We concurrently train the off-policy and behavioral DDPG agents, for 1 million time steps. To ensure sufficient exploration, a standard N (0, 0.1) Gaussian noise is added to actions taken by the behavioral policy. Each transition experienced by the behavioral policy is stored in a buffer replay, which both agents learn from. As a result, both agents are trained with the identical dataset.

- Imitation: A trained DDPG agent acts as an expert, and is used to collect a dataset of 1 million transitions, and populates a buffer, from which the off policy agent learns.



(a) Final buffer performance (b) Concurrent performance (c) Imitation performance

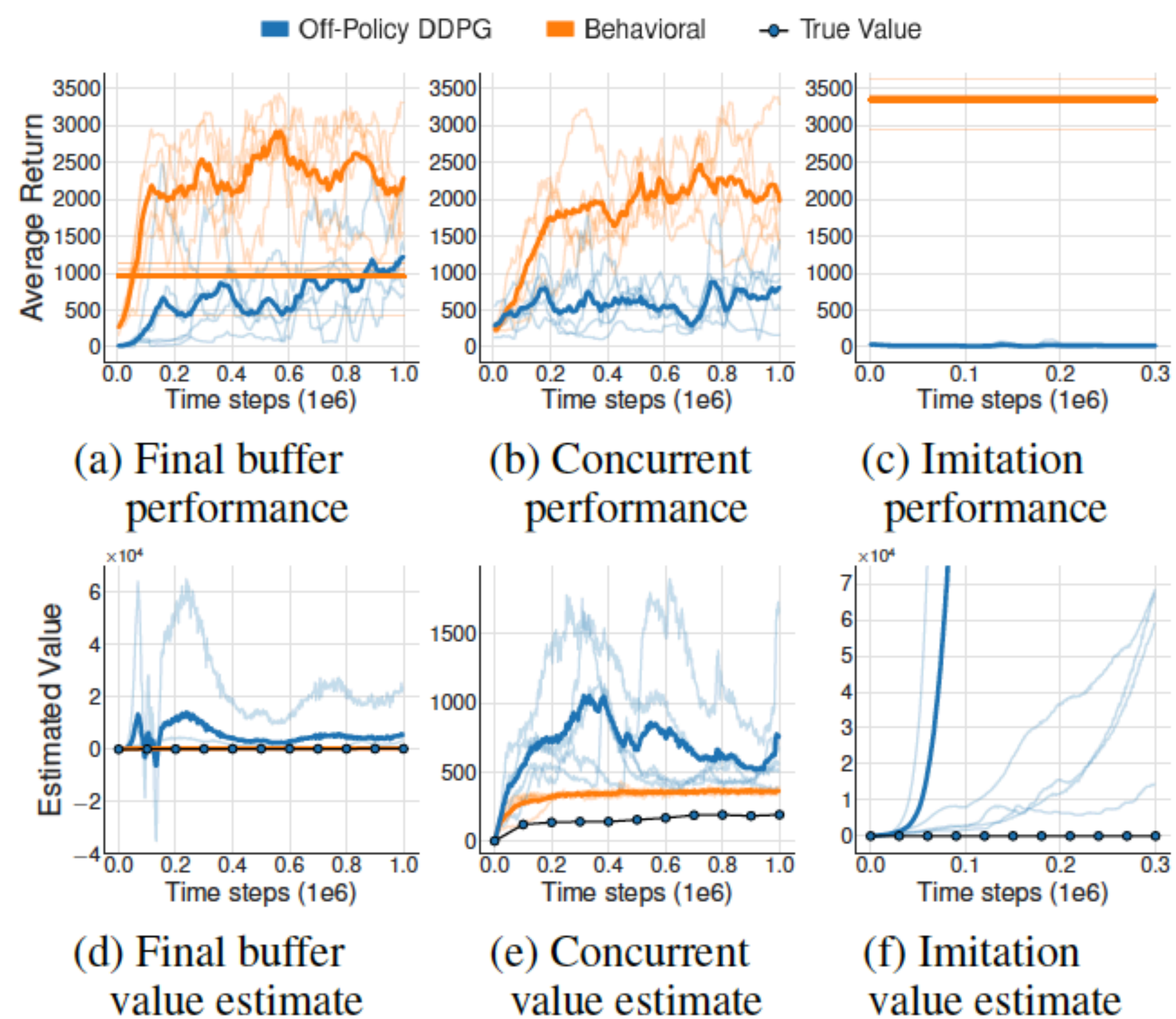
- DDPG (behavioral): (what we have seen in the course) a DDPG policy based on which actions are selected (with small exploration noise) and the experience buffer is populated.

- (Truly) Off-policy DDPG: a DDPG policy that uses experience tuples from the buffer, *it does not influence in any way the data collected in the buffer*

- Final buffer: We train a DDPG agent for 1 million time steps, adding N (0, 0.5) Gaussian noise to actions for high exploration, and store all experienced transitions. This collection procedure creates a dataset with a diverse set of states and actions, with the aim of sufficient coverage.

- Concurrent: We concurrently train the off-policy and behavioral DDPG agents, for 1 million time steps. To ensure sufficient exploration, a standard N (0, 0.1) Gaussian noise is added to actions taken by the behavioral policy. Each transition experienced by the behavioral policy is stored in a buffer replay, which both agents learn from. As a result, both agents are trained with the identical dataset.

- Imitation: A trained DDPG agent acts as an expert, and is used to collect a dataset of 1 million transitions, and populates a buffer, from which the off policy agent learns.



The Q value estimates are higher than their GT values

Off-Policy Deep Reinforcement Learning without Exploration

Why model-free RL does not work with fixed experience buffers?

Extrapolation error:

The Q-function trained from a fixed experience buffer has no way of knowing whether the actions not contained in the buffer are better or worse.

Why model-free RL does not work with fixed experience buffers?

Extrapolation Error

$$Q(s, a) \leftarrow r + \gamma Q(s', a')$$

Extrapolation Error

$$Q(s,a) \leftarrow r + \gamma Q(s',a')$$

GIVEN

GENERATED

Off-Policy Deep Reinforcement Learning without Exploration

Q learning

Extrapolation Error

$$Q(s,a) \leftarrow r + \gamma Q(s',a')$$

1. $(s,a,r,s') \sim Dataset$
2. $a' \sim \pi(s')$

$$a' = \pi(s') = \text{argmax}_a Q_\theta(s',a)$$

Off-Policy Deep Reinforcement Learning without Exploration

Extrapolation Error

$$Q(s,a) \leftarrow r + \gamma Q\boxed{(s',a')}$$

$$\boxed{(s',a') \notin Dataset} \rightarrow Q(s',a') = \mathbf{bad}$$
$$\rightarrow Q(s,a) \quad = \mathbf{bad}$$

Off-Policy Deep Reinforcement Learning without Exploration

Extrapolation Error

$$Q(s,a) \leftarrow r + \gamma \boxed{Q(s',a')}$$

$$(s',a') \notin Dataset \rightarrow \boxed{Q(s',a') = \mathbf{bad}}$$
$$\rightarrow Q(s,a) \quad = \mathbf{bad}$$

Off-Policy Deep Reinforcement Learning without Exploration

Extrapolation Error

$$\boxed{Q(s,a)} \leftarrow r + \gamma Q(s',a')$$

$$(s',a') \notin Dataset \rightarrow Q(s',a') = \mathbf{bad}$$
$$\rightarrow \boxed{Q(s,a) \quad = \mathbf{bad}}$$

Off-Policy Deep Reinforcement Learning without Exploration

Extrapolation Error

Attempting to evaluate π without (sufficient) access to the (s, a) pairs π visits.

Solution: Batch constrained RL

A policy which only traverses *transitions contained in the batch* can be evaluated without error.

BCQ learns a policy with a similar state-action visitation to the data in the batch

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a' \text{s.t.} (s',a') \in \mathcal{B}} Q(s',a')).$$

Solution: Batch constrained RL

BCQ learns a policy with a similar state-action visitation to the data in the batch.

Train a generative model to provide action samples that match the action samples in the batch:

$$\pi(s) = \underset{a_i + \xi_\phi(s, a_i, \Phi)}{\operatorname{argmax}} \; Q_\theta(s, a_i + \xi_\phi(s, a_i, \Phi)),$$

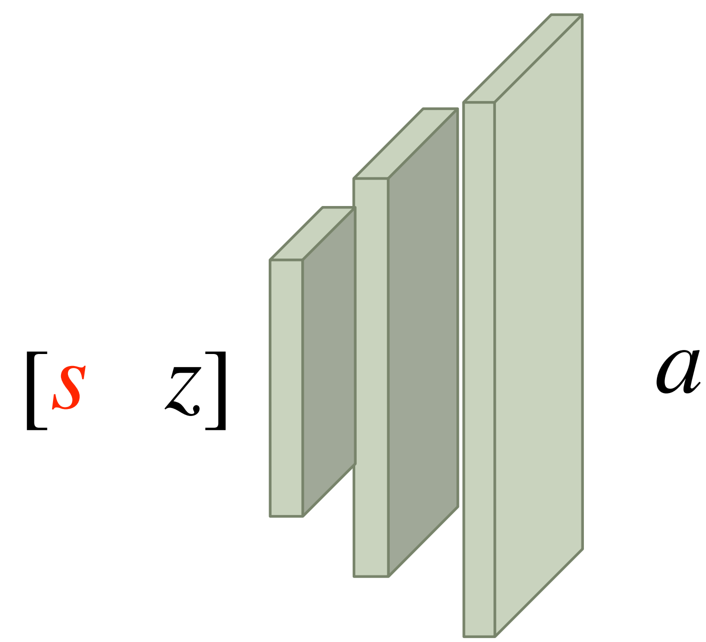
$$\{a_i \sim \boxed{G_\omega(s)}\}_{i=1}^{n}.$$

A state conditioned generative model that predicts actions given a state that are contained in the batch B

Learning stochastic generative models

- As we vary the input noisy samples z, we land in a different plausible action a.

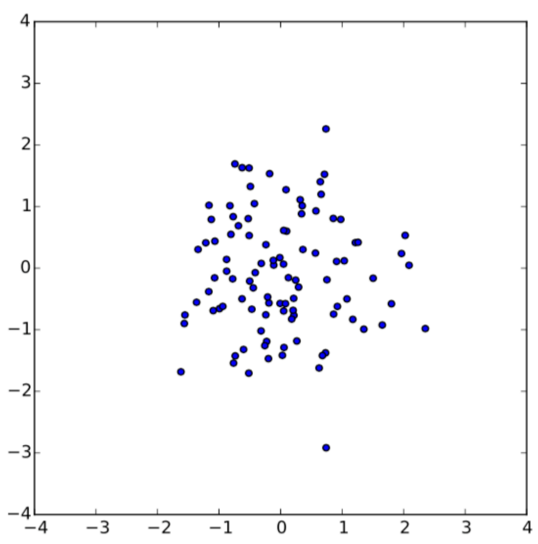
$z \sim \mathcal{N}(\mathbf{0}, I)$



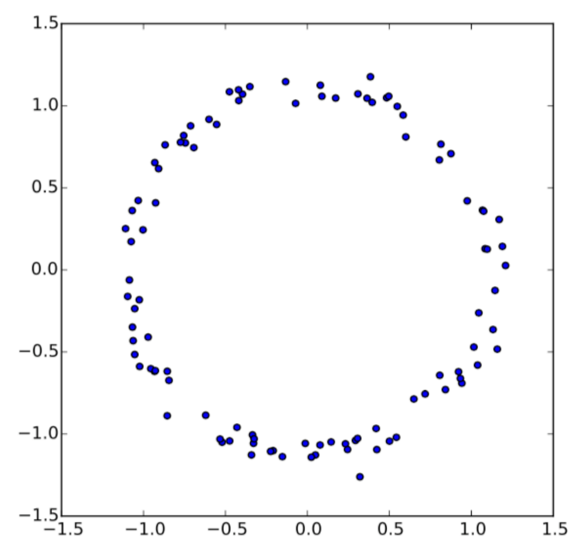
Tutotial on variational Autoencoders, Doersch

Learning stochastic generative models

- Our generative model will transforms the input Gaussian distributions into the desired action distribution.
- Why simple gaussian noise suffices to create complex outputs?
- The neural net will transform it to a complex distribution!

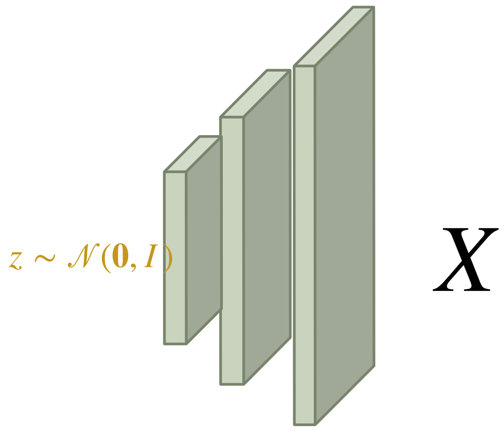


$$z \sim \mathcal{N}(\mathbf{0}, I)$$



$$f(z) = \frac{z}{10} + \frac{z}{||z||}$$

Unconditional generative models



Each sample z should give me a sample from the manifold I am trying to model once it passes through the neural network

We want to learn a mapping from z to the **output X**, usually we assume a Gaussian distribution to sample every coordinate of X from:

$$P(X \mid z; \theta) = \mathcal{N}(X \mid f(z; \theta), \sigma^2 \cdot I)$$

Let's maximize data likelihood. This requires an intractable integral, too many zs..

$$\max_{\theta} . \quad P(X) = \int P(X \mid z; \theta) P(z) dz$$

What if we forget that it is intractable and approximate it with few samples?

(Q: do we know how to take gradients here?)

$$\min_{\theta} . \quad \sum_j -\log P(X_j) = -\sum_j \sum_{z_i \sim \mathcal{N}(\mathbf{0}, I)} \log P(X_j \mid z_i; \theta) = -\sum_j \sum_{z_i \sim \mathcal{N}(\mathbf{0}, I)} \|f(z_i; \theta) - X_j\|^2$$

Motion Prediction Under Multimodality with Conditional Stochastic Networks, Google

Deep Variational Inference

Let's consider sampling z's from an alternative distribution $Q(z)$ and try to minimize the KL between this (variational approximation) and the true posterior, $P(z|X)$. And because I can pick any distribution Q I like, I will also condition it on X to help inform the sampling.

$$D_{KL}(Q(z|X)||P(z|X)) = \int Q(z|X)\log\frac{Q(z|X)}{P(z|X)}dz$$

Deep Variational Inference

Let's consider sampling z's from an alternative distribution $Q(z)$ and try to minimize the KL between this (variational approximation) and the true posterior,$P(z|X)$. And because I can pick any distribution Q I like, I will also condition it on X to help inform the sampling.

$$
\begin{aligned}
D_{KL}(Q(z|X)||P(z|X)) &= \int Q(z|X)\log\frac{Q(z|X)}{P(z|X)}dz \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(z|X)
\end{aligned}
$$

Deep Variational Inference

Let's consider sampling z's from an alternative distribution $Q(z)$ and try to minimize the KL between this (variational approximation) and the true posterior, $P(z \mid X)$. And because I can pick any distribution Q I like, I will also condition it on X to help inform the sampling.

$$
\begin{aligned}
D_{KL}(Q(z \mid X) \,||\, P(z \mid X)) &= \int Q(z \mid X) \log \frac{Q(z \mid X)}{P(z \mid X)} dz \\
&= \mathbb{E}_Q \log Q(z \mid X) - \mathbb{E}_Q \log P(z \mid X) \\
&= \mathbb{E}_Q \log Q(z \mid X) - \mathbb{E}_Q \log \frac{P(X \mid z) P(z)}{P(X)}
\end{aligned}
$$

Deep Variational Inference

Let's consider sampling z's from an alternative distribution $Q(z)$ and try to minimize the KL between this (variational approximation) and the true posterior,$P(z\,|\,X)$. And because I can pick any distribution Q I like, I will also condition it on X to help inform the sampling.

$$
\begin{aligned}
D_{KL}(Q(z\,|\,X)\,||\,P(z\,|\,X)) &= \int Q(z\,|\,X)\log\frac{Q(z\,|\,X)}{P(z\,|\,X)}dz \\
&= \mathbb{E}_Q \log Q(z\,|\,X) - \mathbb{E}_Q \log P(z\,|\,X) \\
&= \mathbb{E}_Q \log Q(z\,|\,X) - \mathbb{E}_Q \log \frac{P(X\,|\,z)P(z)}{P(X)} \\
&= \mathbb{E}_Q \log Q(z\,|\,X) - \mathbb{E}_Q \log P(X\,|\,z) - \mathbb{E}_Q \log P(z) + \log P(X)
\end{aligned}
$$

Deep Variational Inference

Let's consider sampling z's from an alternative distribution $Q(z)$ and try to minimize the KL between this (variational approximation) and the true posterior, $P(z|X)$. And because I can pick any distribution Q I like, I will also condition it on X to help inform the sampling.

$$
\begin{aligned}
D_{KL}(Q(z|X)||P(z|X)) &= \int Q(z|X) \log \frac{Q(z|X)}{P(z|X)} dz \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(z|X) \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log \frac{P(X|z)P(z)}{P(X)} \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(X|z) - \mathbb{E}_Q \log P(z) + \log P(X) \\
&= D_{KL}(Q(z|X)|P(z)) - \mathbb{E}_Q \log P(X|z) + \log P(X)
\end{aligned}
$$

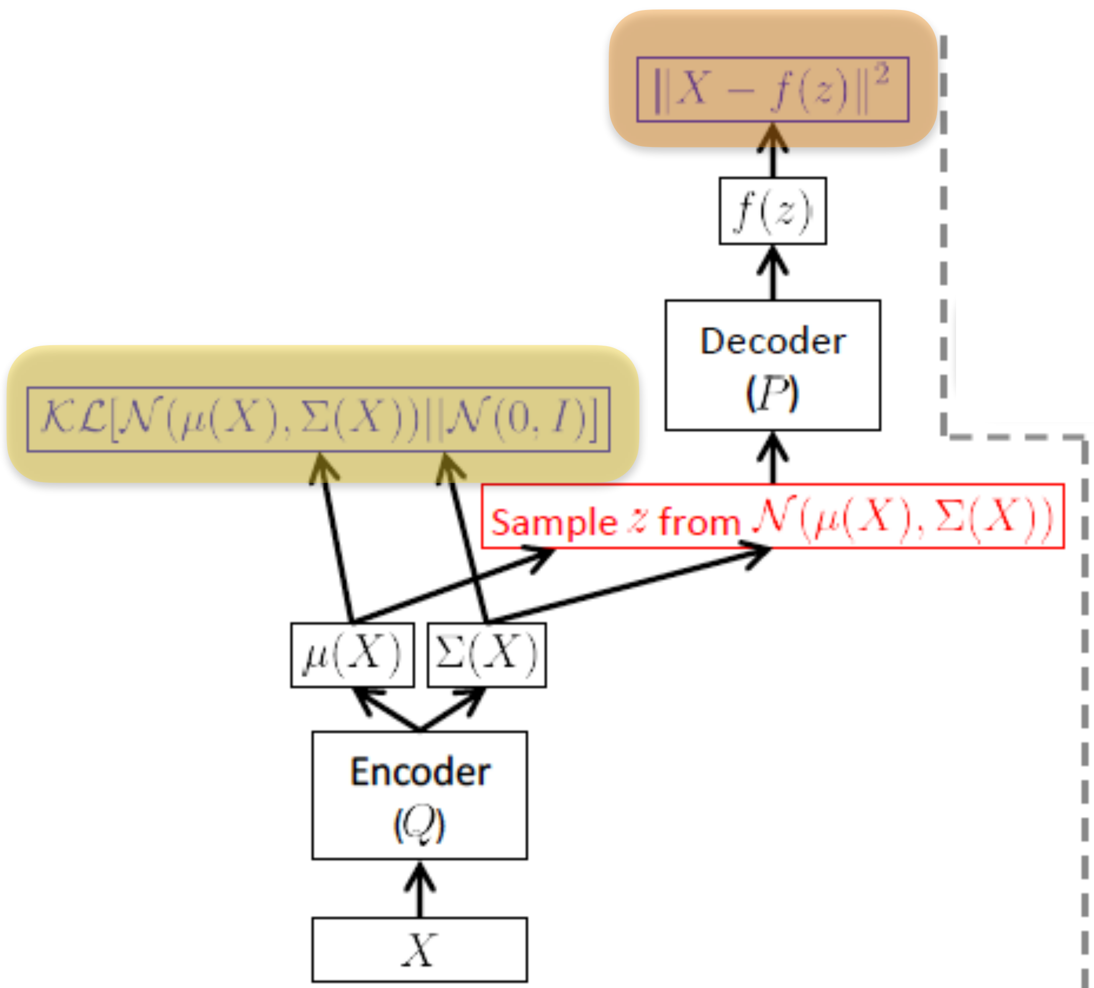
Deep Variational Inference

Let's consider sampling z's from an alternative distribution $Q(z)$ and try to minimize the KL between this (variational approximation) and the true posterior, $P(z|X)$. And because I can pick any distribution Q I like, I will also condition it on X to help inform the sampling.

$$
\begin{aligned}
D_{KL}(Q(z|X)||P(z|X)) &= \int Q(z|X)\log\frac{Q(z|X)}{P(z|X)}dz \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(z|X) \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log \frac{P(X|z)P(z)}{P(X)} \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(X|z) - \mathbb{E}_Q \log P(z) + \log P(X) \\
&= D_{KL}(Q(z|X)|P(z)) - \mathbb{E}_Q \log P(X|z) + \log P(X)
\end{aligned}
$$

$$
\min_{\phi,\theta}.\quad D_{KL}(\underbrace{Q(z|X;\phi)}_{\text{encoder}}||P(z)) - \mathbb{E}_Q \log \underbrace{P(X|z;\theta)}_{\text{decoder}}
$$

Variational Autoencoder

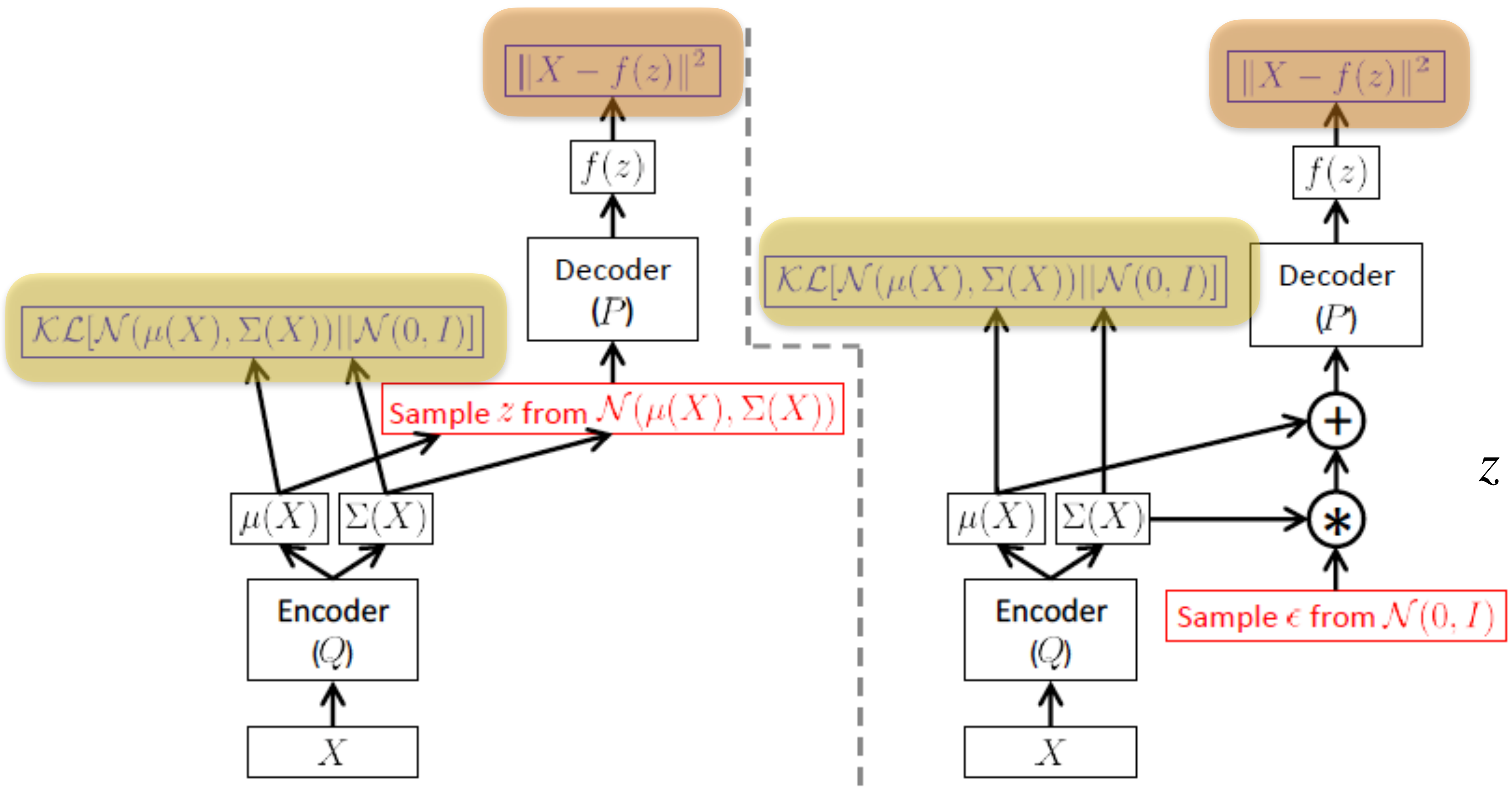


$$\min_{\phi,\theta} . \quad D_{KL}(\underbrace{Q(z \mid X; \phi)}_{\text{encoder}} \,||\, P(z)) - \mathbb{E}_Q \log \underbrace{P(X \mid z; \theta)}_{\text{decoder}}$$

T*utotial on variational Autoencoders,* Doersch

Variational Autoencoder

From left to right: re-parametrization trick!



$$z = \mu + \epsilon \cdot \sigma$$

$$\min_{\phi,\theta} . \quad D_{KL}(Q(z\,|\,X;\phi)\,||\,P(z)) - \mathbb{E}_Q \log P(X\,|\,z;\theta)$$

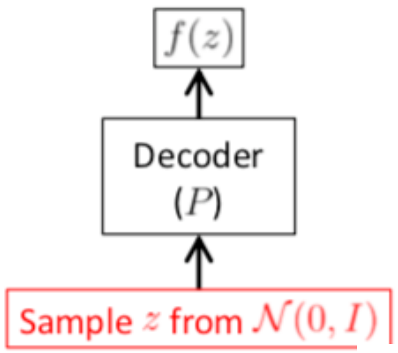
decoder

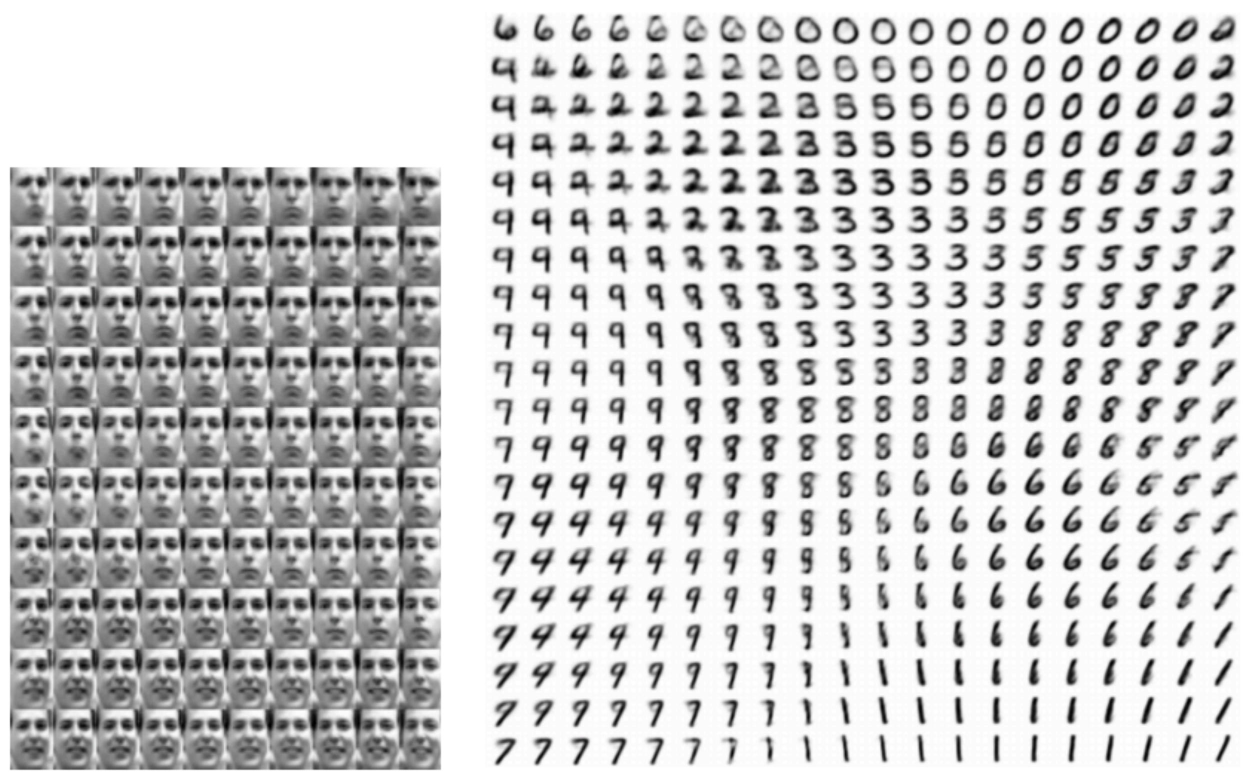
$$D_{\mathrm{KL}}(\mathcal{N}(\mu,\sigma)||\mathcal{N}(0,1)) = -\frac{1}{2}\sum_{j=1}^{J}(1+\log(\sigma_j^2)-\mu_j^2-\sigma_j^2)$$

T*utotial on variational Autoencoders,* Doersch

Variational Autoencoder

At test time

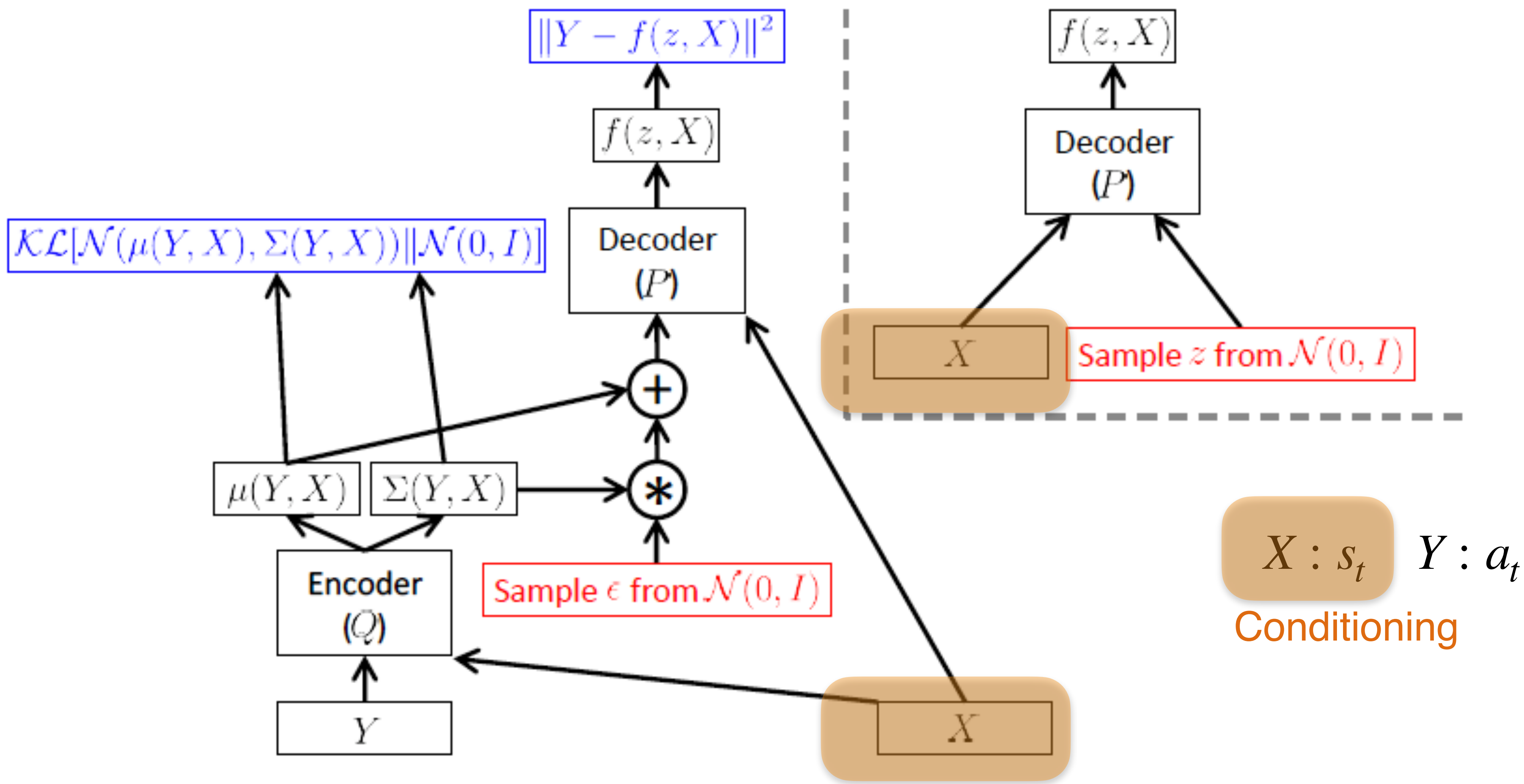




(a) Learned Frey Face manifold (b) Learned MNIST manifold

Auto-Encoding Variational Bayes, *Kingma and Welling*

Conditional VAE



$$\min_{\phi}. \quad D_{KL}(Q(z \mid X, Y) \,||\, P(z \mid \mathscr{D}) = \min_{\phi}. \quad D_{KL}(Q(z \mid X, Y) \mid P(z)) - \mathbb{E}_Q \log P(\mathscr{D} \mid z)$$

Tutotial on variational Autoencoders, Doersch

Algorithm 1 BCQ

Input: Batch $\mathcal{B}$, horizon T, target network update rate τ, mini-batch size N, max perturbation Φ, number of sampled actions n, minimum weighting λ.

Initialize Q-networks $Q_{\theta_1}, Q_{\theta_2}$, perturbation network ξ_ϕ, and VAE $G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$, with random parameters θ_1, θ_2, ϕ, ω, and target networks $Q_{\theta'_1}, Q_{\theta'_2}$, $\xi_{\phi'}$ with $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$.

for $t = 1$ **to** T **do**

  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$

  $\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$

  $\omega \leftarrow \text{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

  Sample n actions: $\{a_i \sim G_\omega(s')\}_{i=1}^n$

  Perturb each action: $\{a_i = a_i + \xi_\phi(s', a_i, \Phi)\}_{i=1}^n$

  Set value target y (Eqn. 13)

  $\theta \leftarrow \text{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

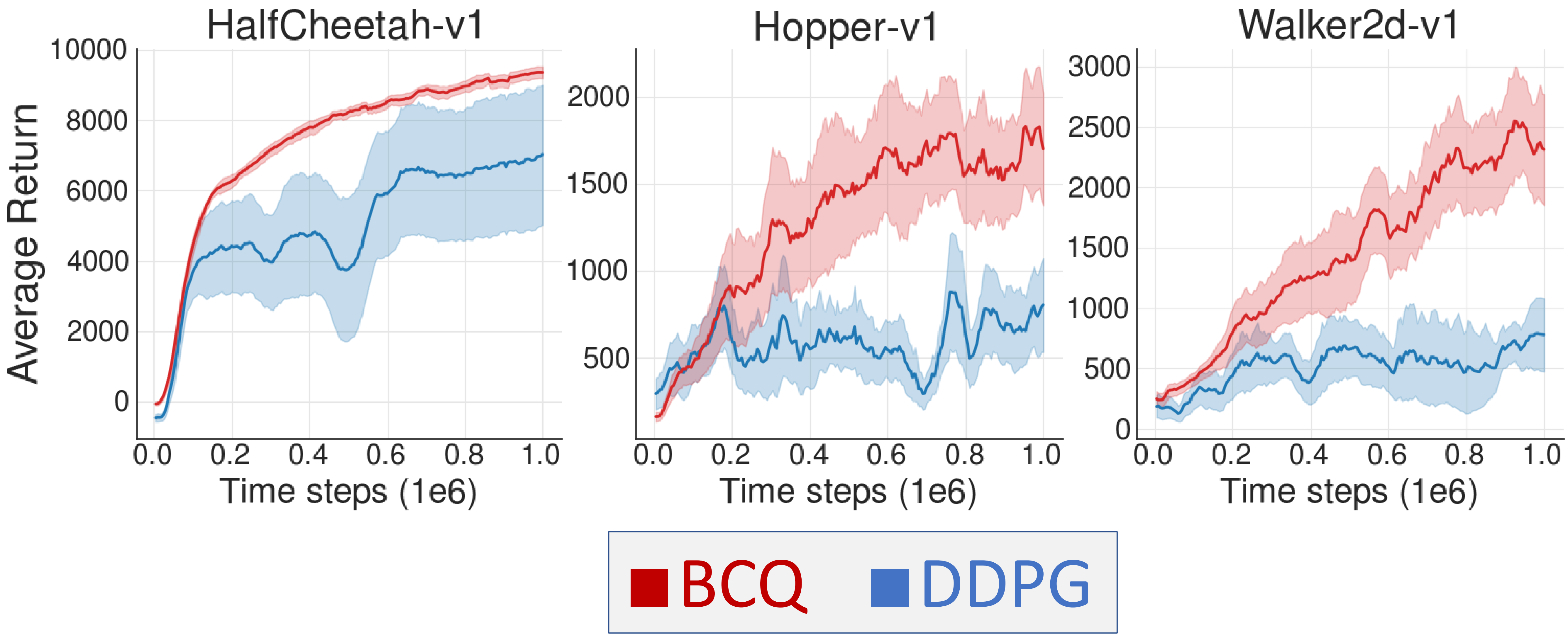
  $\phi \leftarrow \text{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

  Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$

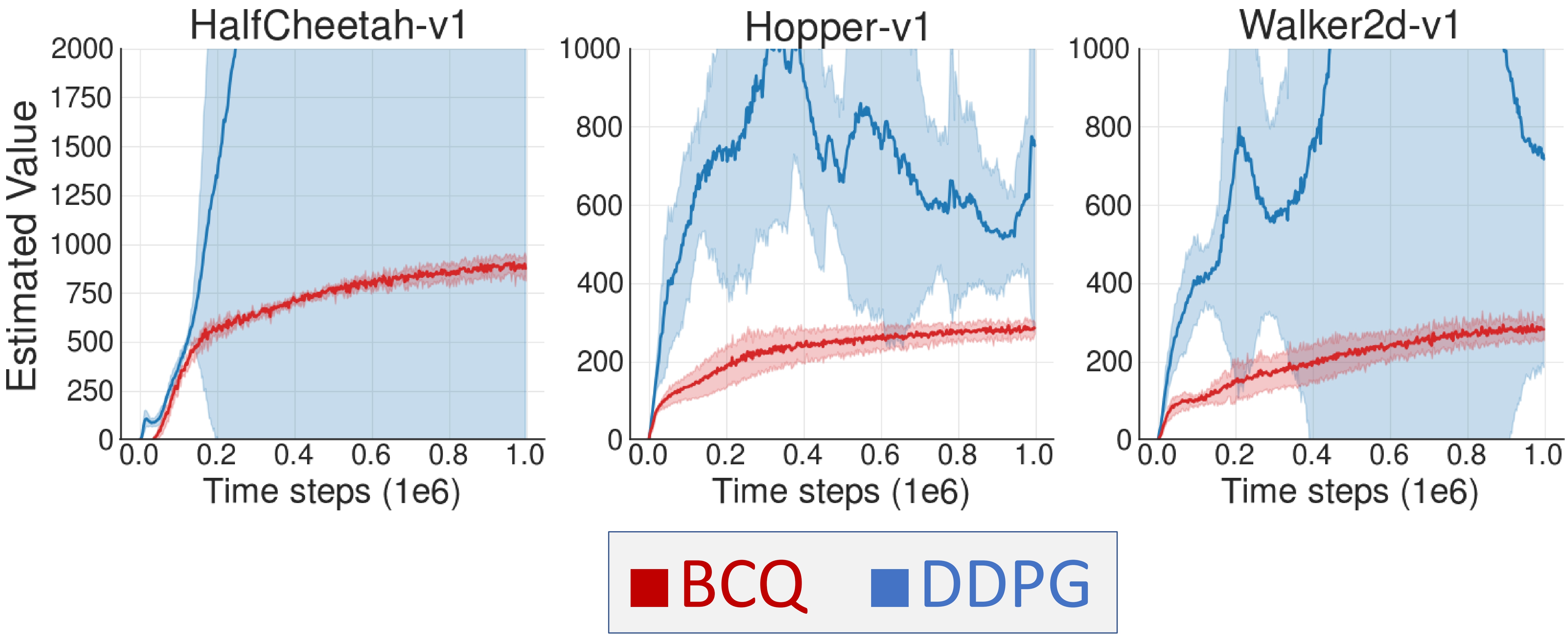
  $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end for

$$r + \gamma \max_{a_i} \left[\lambda \min_{j=1,2} Q_{\theta'_j}(s', a_i) + (1 - \lambda) \max_{j=1,2} Q_{\theta'_j}(s', a_i) \right]$$



HalfCheetah-v1
Hopper-v1
Walker2d-v1
Average Return
10000
8000
6000
4000
2000
0
2000
1500
1000
500
3000
2500
2000
1500
1000
500
0
0.0 0.2 0.4 0.6 0.8 1.0
Time steps (1e6)
Time steps (1e6)
Time steps (1e6)
BCQ
DDPG


HalfCheetah-v1
Hopper-v1
Walker2d-v1
Estimated Value
2000
1750
1500
1250
1000
750
500
250
0
800
600
400
200
0.0
0.2
0.4
0.6
0.8
1.0
Time steps (1e6)
BCQ
DDPG