Carnegie Mellon

School of Computer Science

Deep Reinforcement Learning and Control

Off policy RL

Fall 2021, CMU 10-703

Instructors:
Katerina Fragkiadaki
Russ Salakhutdinov

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

Many of these slides were based on Doina Precup's slides

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.

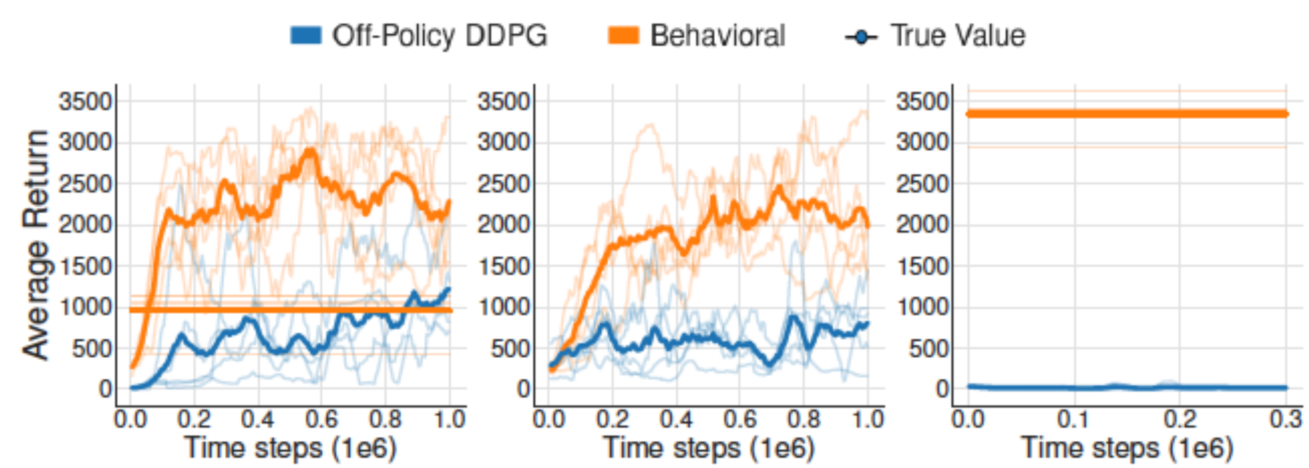
- DDPG (behavioral): (what we have seen in the course) a DDPG policy based on which actions are selected (with small exploration noise) and the experience buffer is populated.

- (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

Off-Policy Deep Reinforcement Learning without Exploration, Fujimoto et al

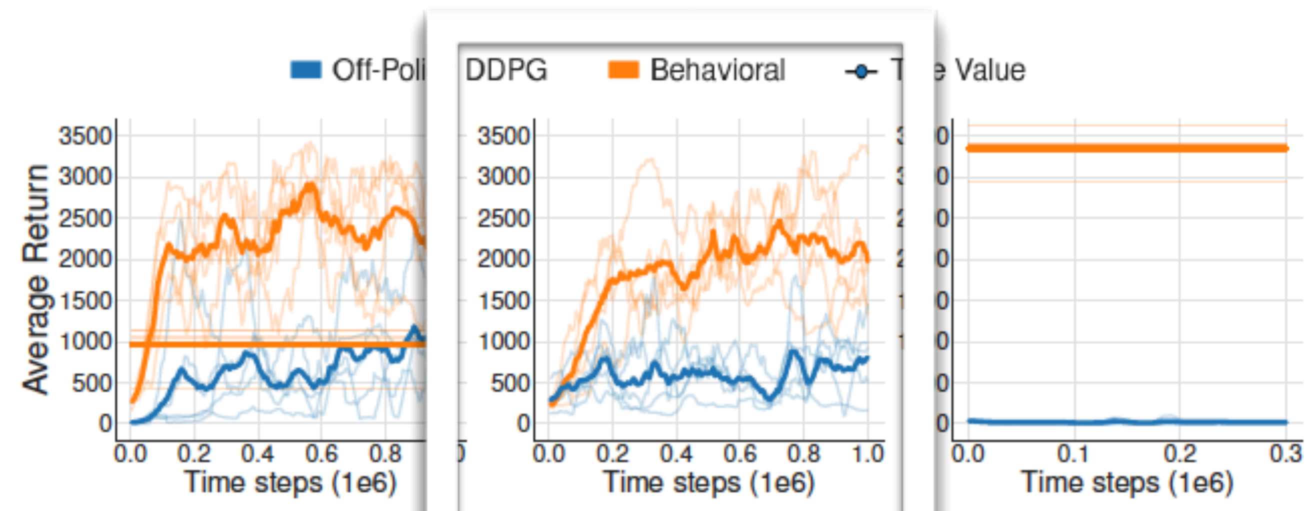
- DDPG (behavioral): (what we have seen in the course) a DDPG policy based on which actions are selected (with small exploration noise) and the experience buffer is populated.

- (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.

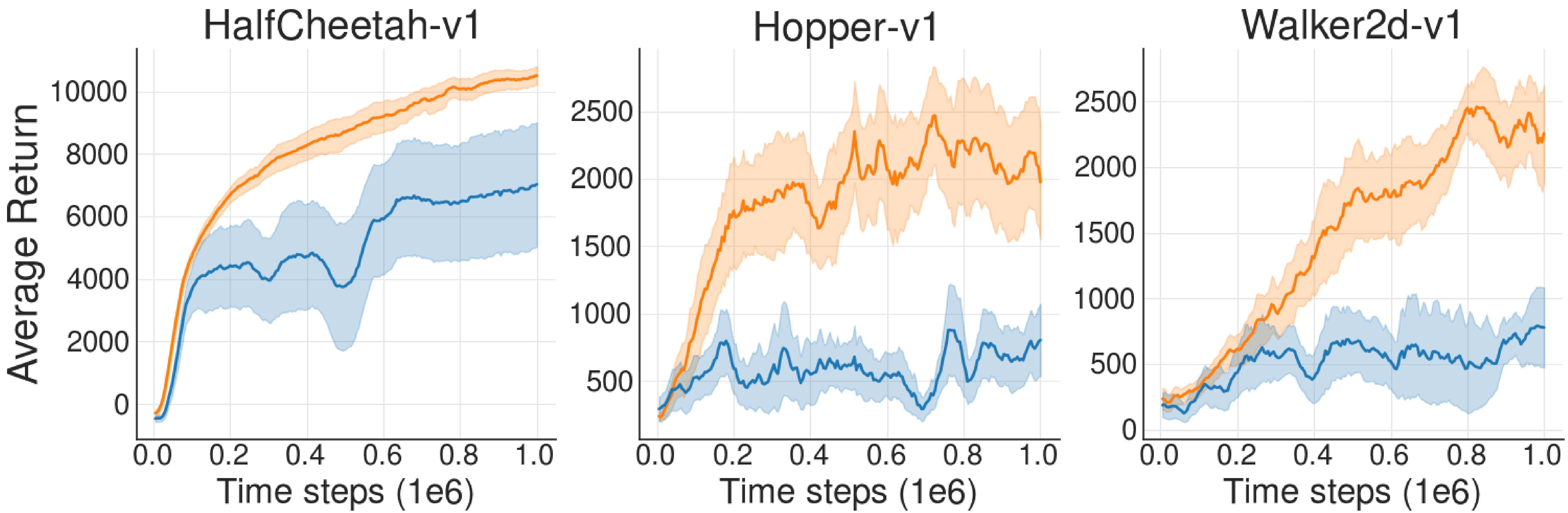


(a) Final buffer performance

(b) Concurrent performance

(c) Imitation performance

Off-Policy Deep Reinforcement Learning without Exploration, Fujimoto et al



Off-Policy Deep Reinforcement Learning without Exploration, Fujimoto et al

Agent orange and agent blue are trained with…

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

2. The **same dataset.**

Off-Policy Deep Reinforcement Learning without Exploration, Fujimoto et al

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.

Off-Policy Deep Reinforcement Learning without Exploration, Fujimoto et al

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 Reinforcement Learning without Exploration, Fujimoto et al

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

why?

Off-Policy Deep Reinforcement Learning without Exploration, Fujimoto et al

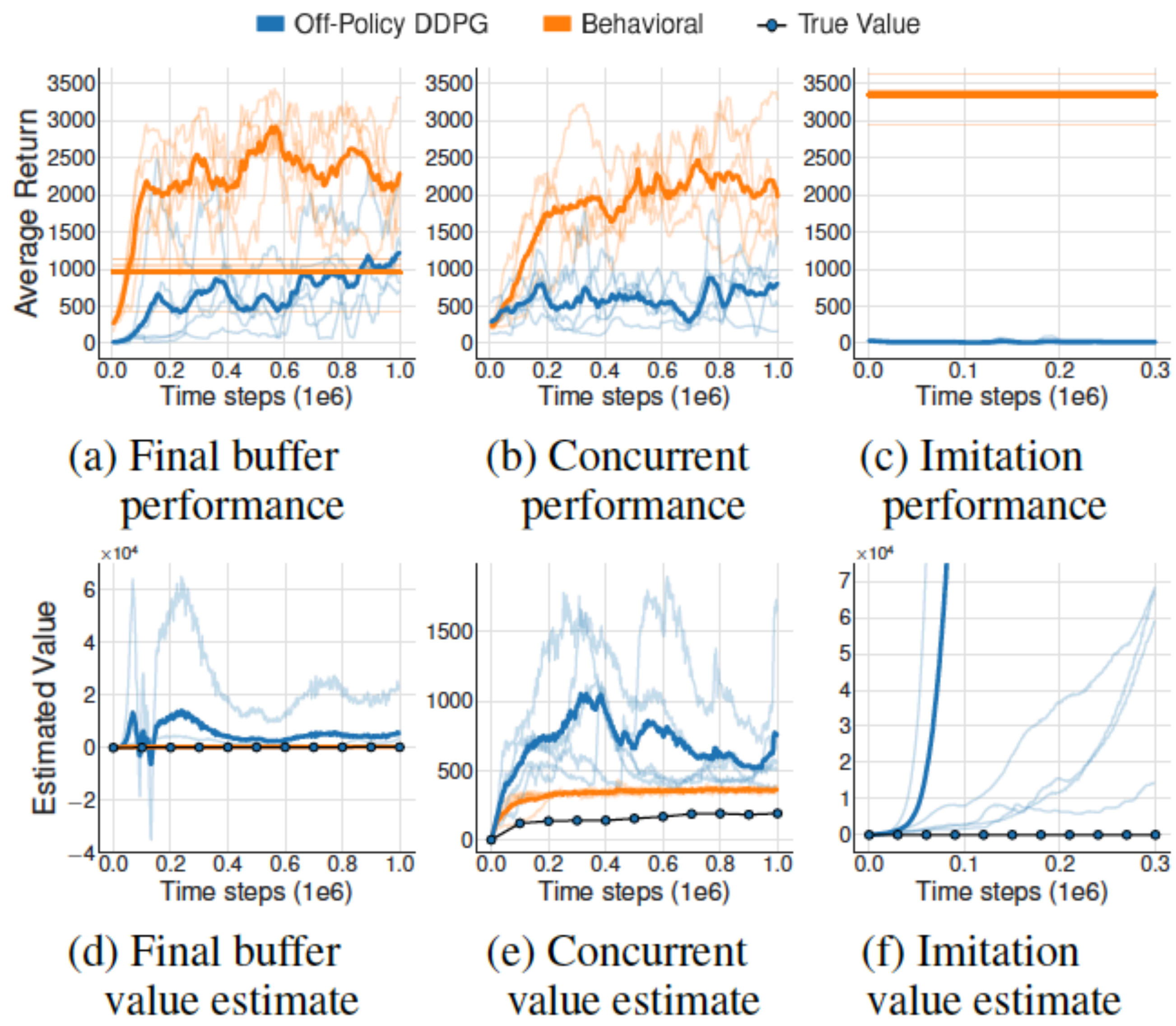
- DDPG (behavioral): (what we have seen in the course) a DDPG policy based on which actions are selected (with small exploration noise) and the experience buffer is populated.

- (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

(d) Final buffer value estimate (e) Concurrent value estimate (f) Imitation value estimate

The Q value estimates are higher than their GT values

Off-Policy Deep Reinforcement Learning without Exploration, Fujimoto et al

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.

Off-Policy Deep Reinforcement Learning without Exploration, Fujimoto et al

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

Extrapolation Error

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

Off-Policy Deep Reinforcement Learning without Exploration, Fujimoto et al

Extrapolation Error

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

GIVEN

GENERATED

Off-Policy Deep Reinforcement Learning without Exploration, Fujimoto et al

Q learning

Extrapolation Error

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

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

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

Off-Policy Deep Reinforcement Learning without Exploratio

Extrapolation Error

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

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

$$\rightarrow Q(s,a) \quad = \mathbf{bad}$$

Extrapolation Error

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

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

Extrapolation Error

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

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

$$\rightarrow \boxed{Q(s,a) \quad = \textbf{bad}}$$

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:

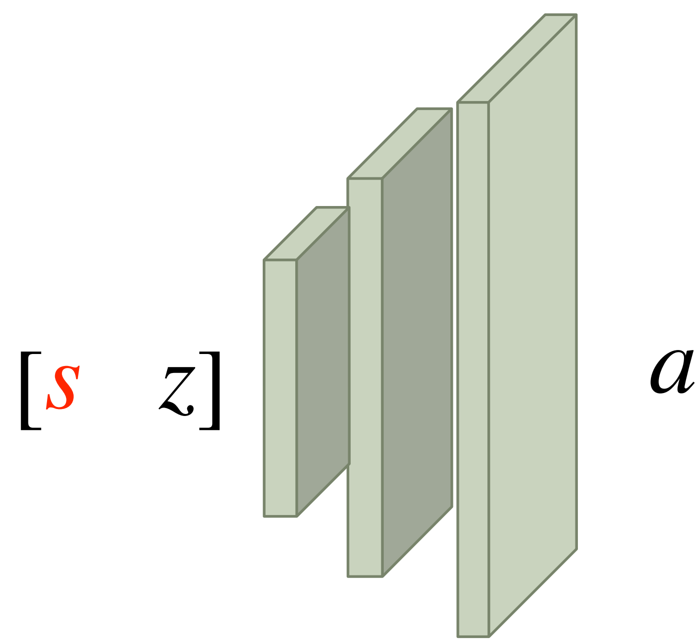
$$\pi(s) = \underset{a_i + \xi_\phi(s, a_i, \Phi)}{\operatorname{argmax}} \; Q_\theta(s, a_i + \xi_\phi(s, a_i, \Phi)),$$
$$\{a_i \sim \boxed{G_\omega(s)}\}_{i=1}^{n}.$$

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.

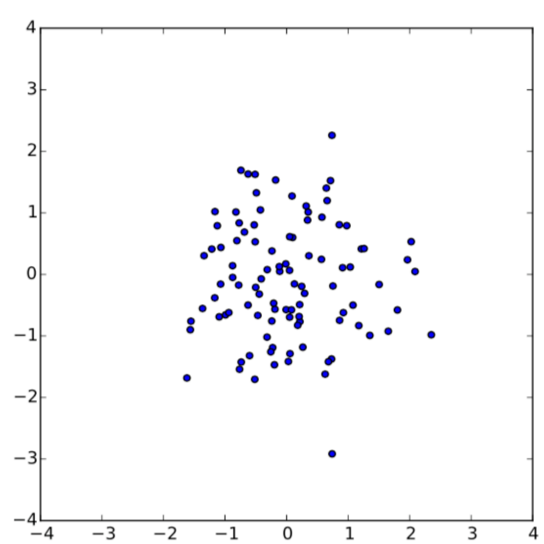
$z \sim \mathcal{N}(\mathbf{0}, I)$



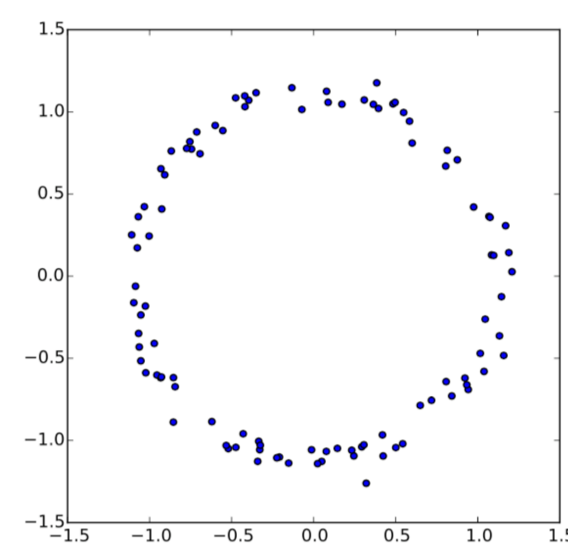
Tutotial on variational Autoencoders, Doersch

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!



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

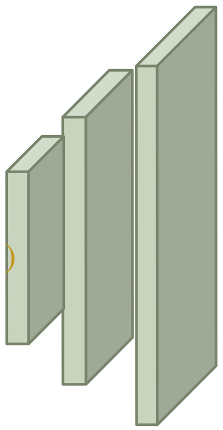


$$f(z) = \frac{z}{10} + \frac{z}{\|z\|}$$

Tutotial on variational Autoencoders, Doersch

Unconditional generative models

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



X

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?

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

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

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.

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

$$= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(z|X)$$

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.

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

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.

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

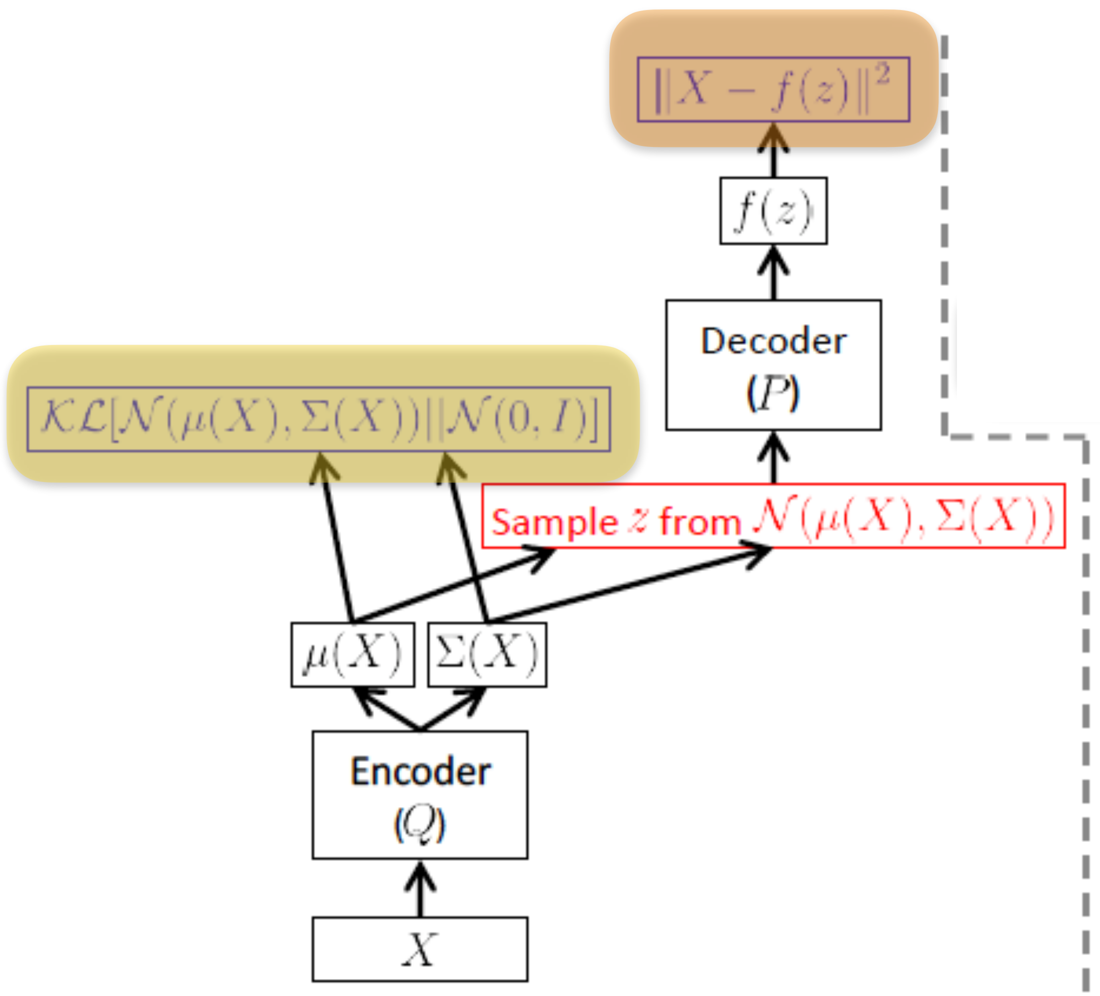
Deep Variational Inference

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, $P(z \mid X)$. And because I can pick any distribution Q I like, I will also condition it on X to help inform the sampling.

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

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

Variational Autoencoder

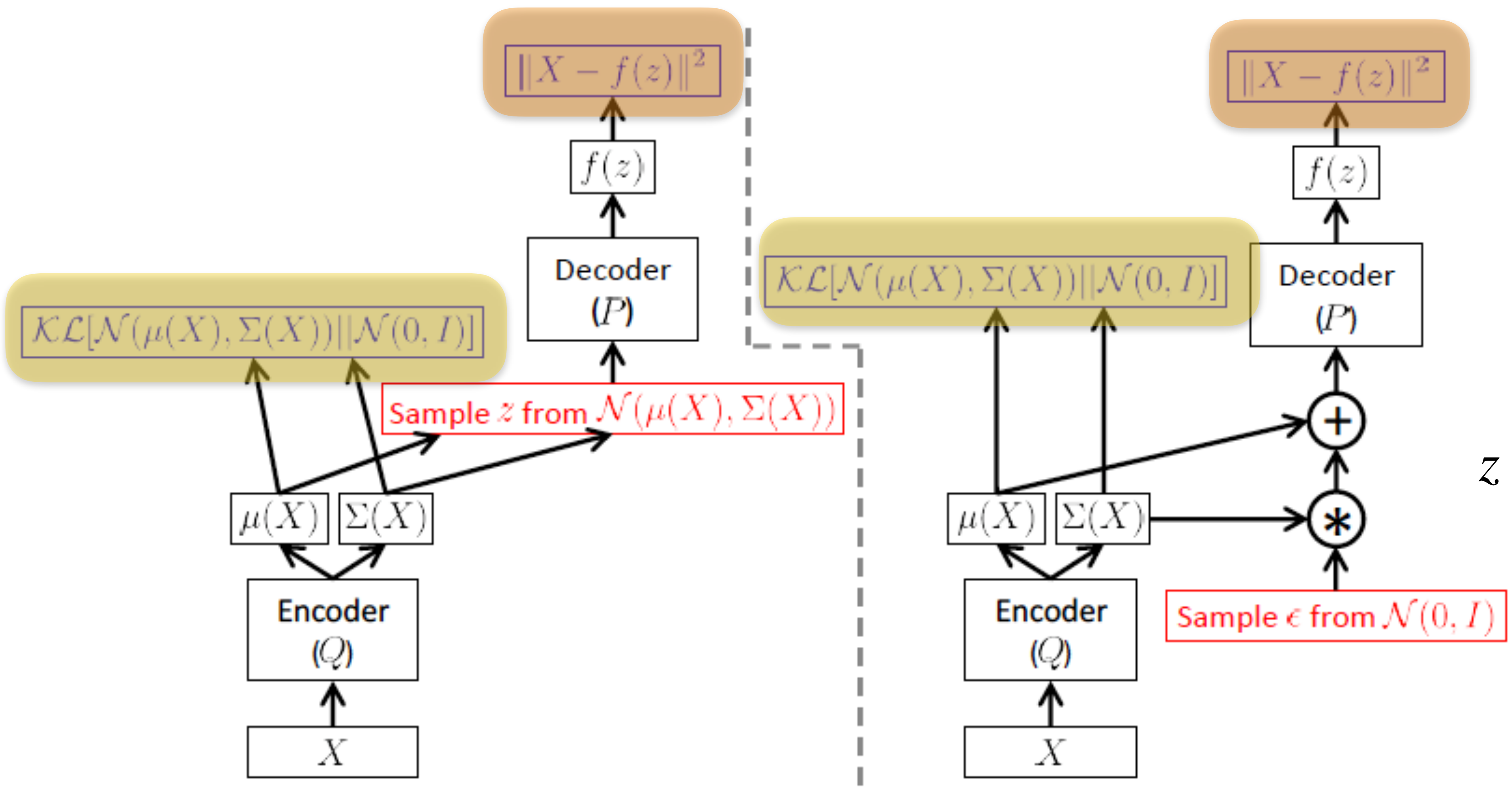


$$\min_{\phi,\theta} . \quad D_{KL}(\underbrace{Q(z \mid X; \phi)}_{\text{encoder}} \| P(z)) - \mathbb{E}_Q \log \underbrace{P(X \mid z; \theta)}_{\text{decoder}}$$

Tutotial on variational Autoencoders, Doersch

Variational Autoencoder

From left to right: re-parametrization trick!



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

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

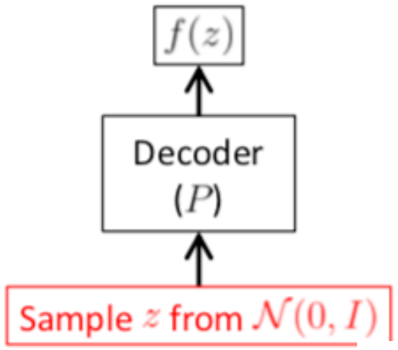
decoder

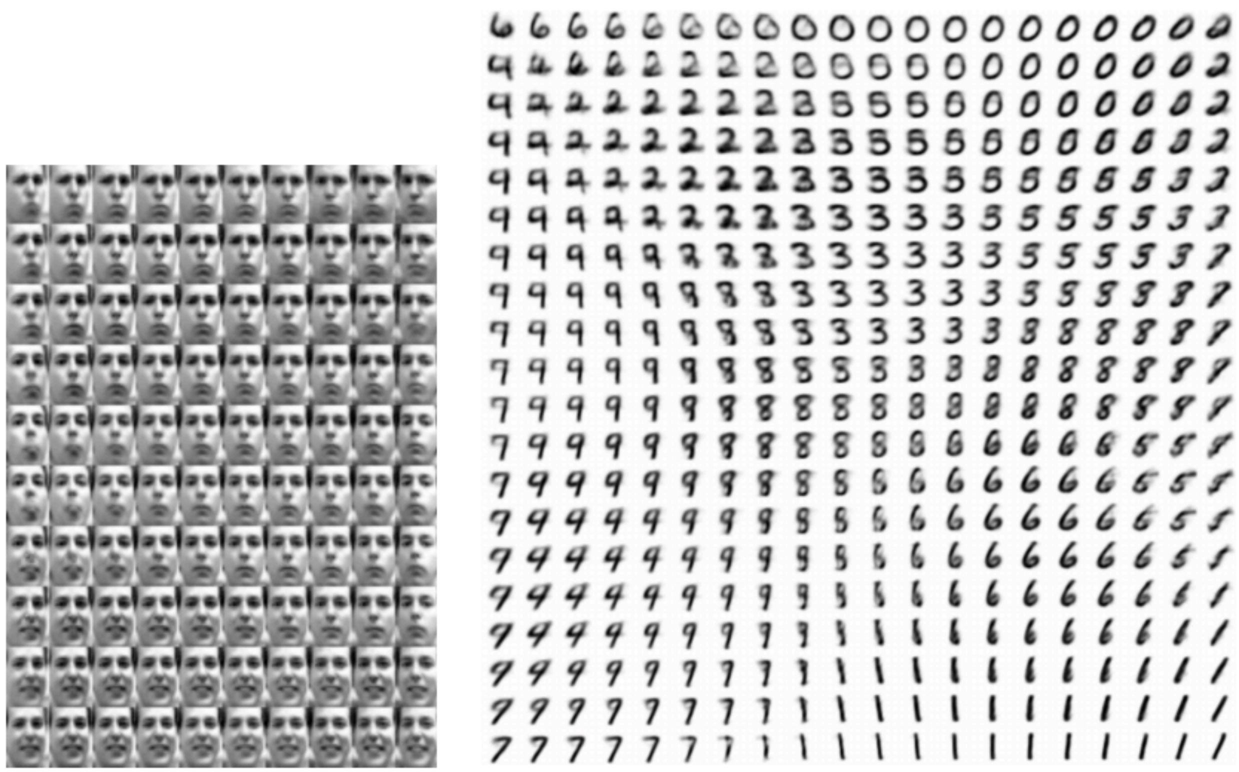
$$D_{\mathrm{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1)) = -\frac{1}{2} \sum_{j=1}^{J} (1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)$$

T*utotial on variational Autoencoders,* Doersch

Variational Autoencoder

At test time

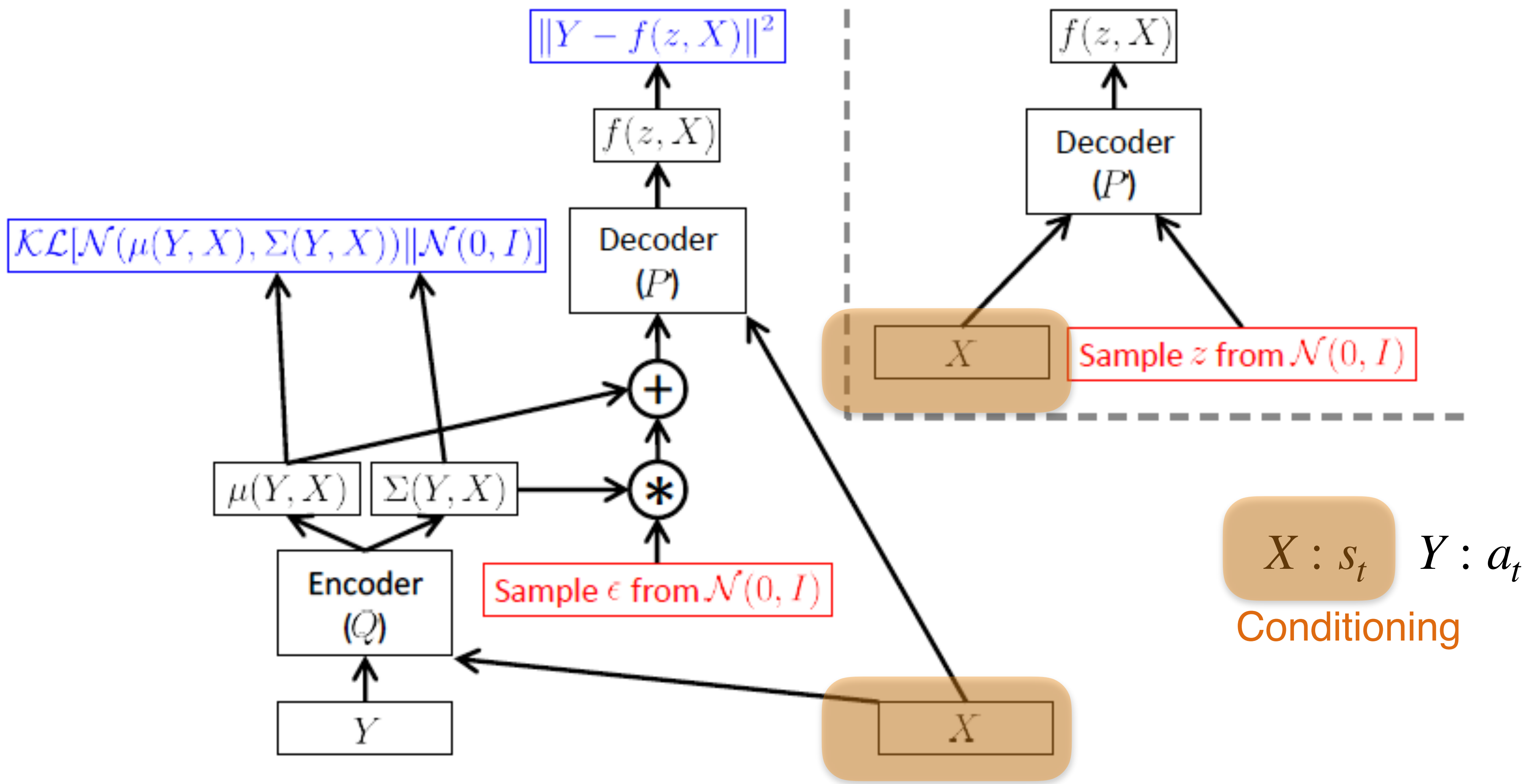




(a) Learned Frey Face manifold (b) Learned MNIST manifold

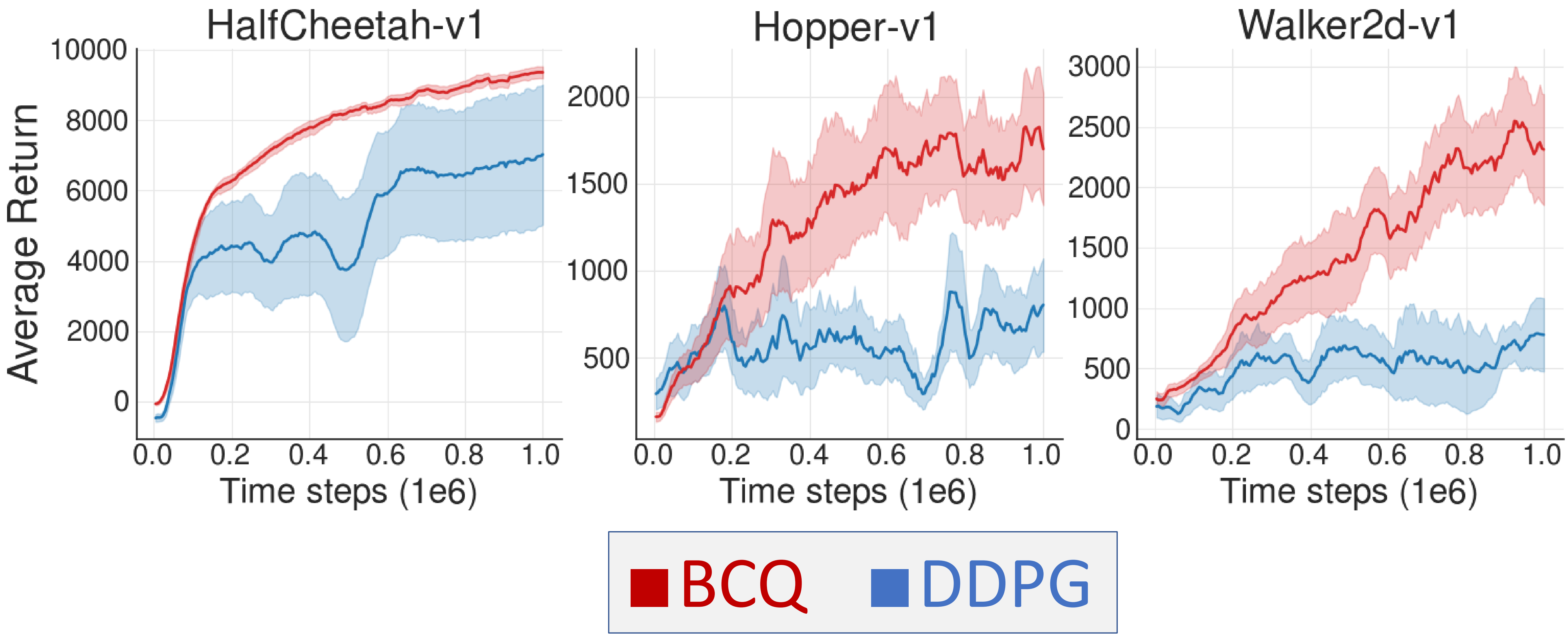
Auto-Encoding Variational Bayes, Kingma and Welling

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


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