Carnegie Mellon

School of Computer Science

Deep Reinforcement Learning and Control

Learning from demonstrations and task rewards, Off policy RL, Adversarial Imitation learning

Spring 2021, CMU 10-403

Katerina Fragkiadaki

Learning from demonstrations

Pros

- Can much accelerate trial-and-error learning by suggesting good actions to try
- Can help us train initial safe policies, to deploy in the real world

Cons

- Time consuming
- May include suboptimal, noise and diverse ways to perform the task
- When you imitate, you cannot surpass the “expert”.

Learning from task rewards

Pros

- Cheap supervision
- Optimizes the right end task, as encoded in the task rewards

Cons

- Super sample inefficient - impossible to have in the real world right now
- Initial policy is random thus unsafe to deploy in the real world

Learning from demonstrations and task rewards

Goals

- More sample efficient that RL alone
- Good/safe initial performance
- Outperform the human expert

Challenges for kinesthetic demonstrations

- Handling expert sub optimality

Additional challenges for learning from video demonstrations

- requires visual perception
- requires handling mismatch between imitator and demonstrator action spaces

Learning from demonstrations and task rewards

Goals

- More sample efficient that RL
- Good/safe initial performance
- Outperform the human expert

Challenges for kinesthetic demonstrations

- Handling expert sub optimality

Additional challenges for learning from video demonstrations

- requires visual perception
- requires handling mismatch between imitator and demonstrator action spaces

Learning from demonstrations and task rewards

- Initialize the replay buffer with demos (which will be later either removed, or kept forever) and start your model-free RL method
- Pre-train the model-free RL method (a policy and a consistent with it value function) with a demonstration only buffer, then fine-tune it.
- Combine imitation and task rewards
- Exploit the temporal structure, and step progressively earlier and earlier in time along a trajectory, to solve progressively longer horizon tasks, as opposed to solving them at once.

Learning from demonstrations and task rewards

- Initialize the replay buffer with demos (which will be later either removed, or kept forever) and start your model-free RL method
- Pre-train the model-free RL method (a policy and a consistent with it value function) with a demonstration only buffer, then fine-tune it.
- Combine imitation and task rewards
- Exploit the temporal structure, and step progressively earlier and earlier in time along a trajectory, to solve progressively longer horizon tasks, as opposed to solving them at once.

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. E.g., that data can come from demonstrations!

Off-policy RL seen so far

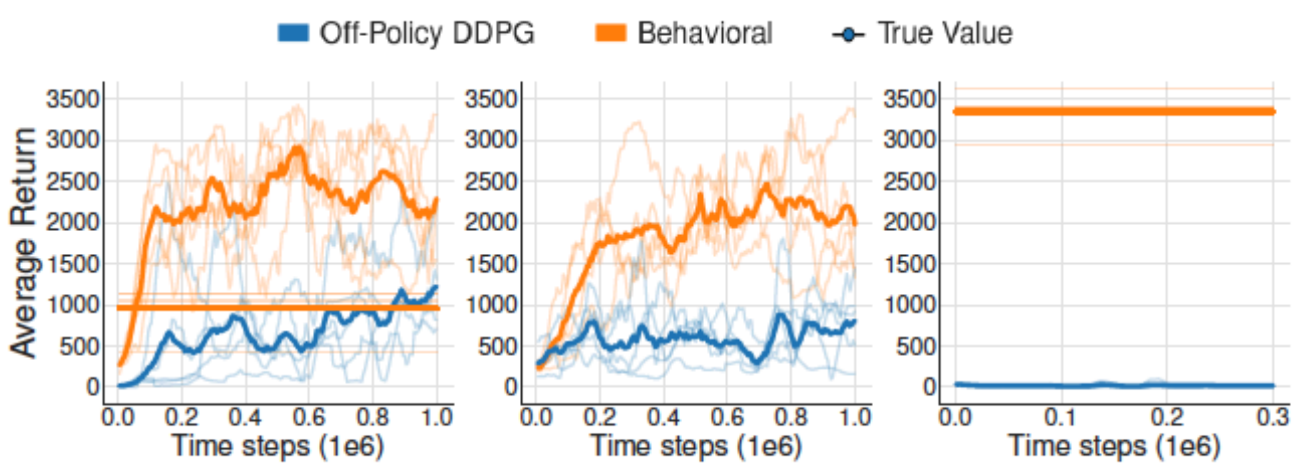
- Off-policy RL learns from data collected under a behavioral policy different than the current policy.
- In what we have seen thus far, “off-policy” transitions are generated from earlier versions of the current policy.
- They are thus heavily correlated to the current policy.
- Not that much *off-policy*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

- 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

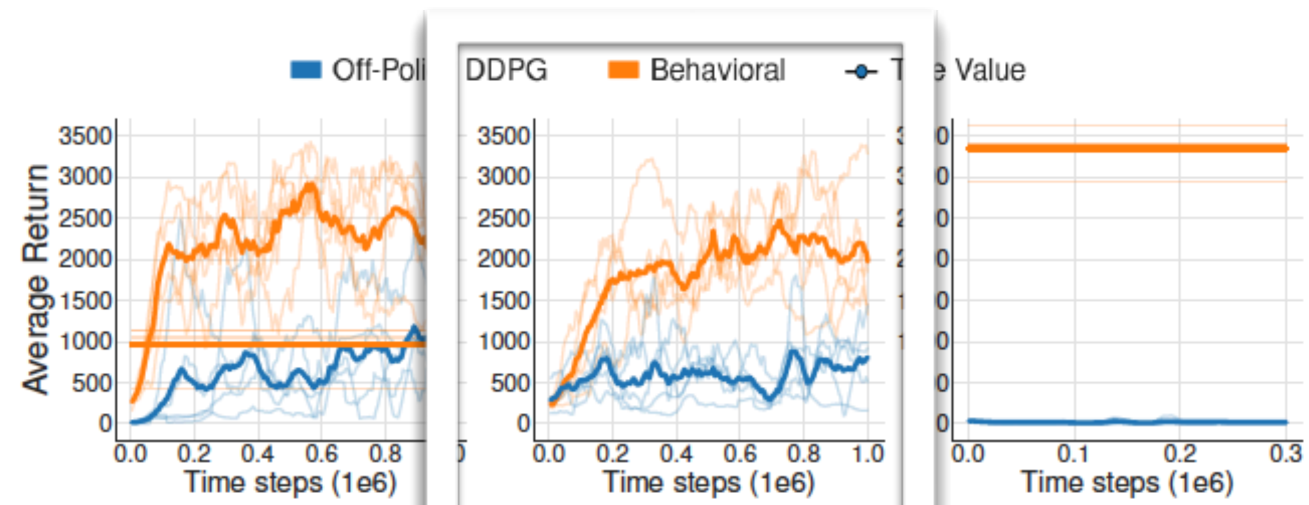
- DDPG (behavioral): (what we have seen in the course) a DDPG policy based on which actions are selected (with small exploration noise) and the experience buffer is populated.

- (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*

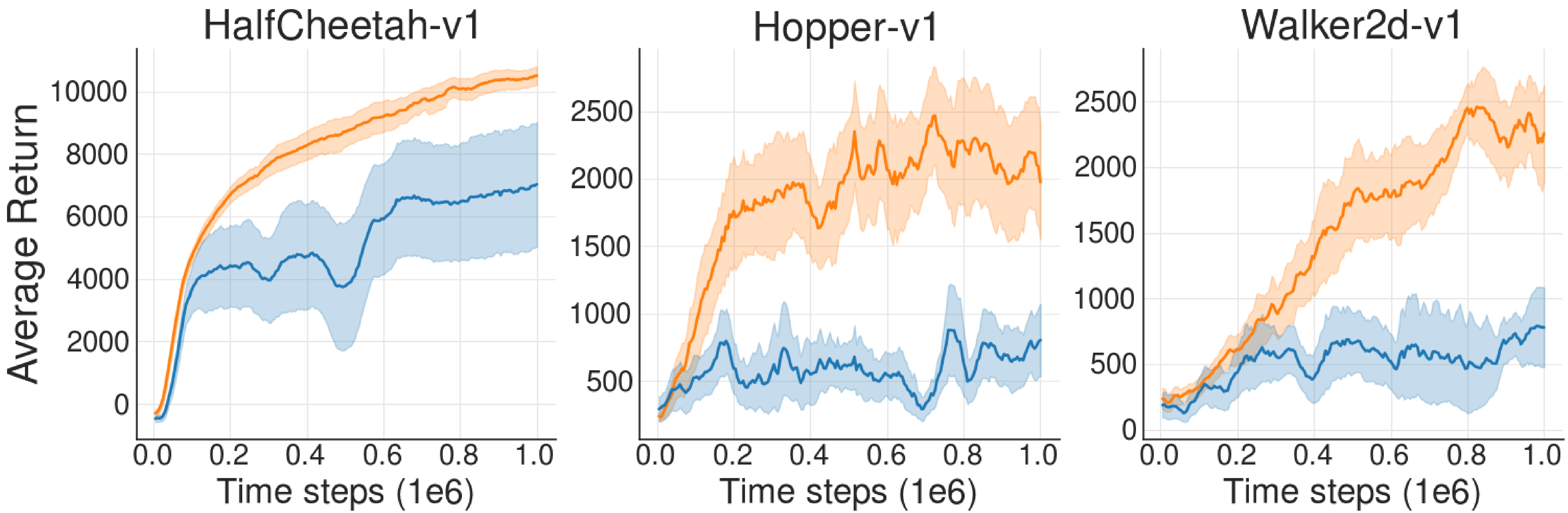
- Final buffer: We train a DDPG agent for 1 million time steps, adding N (0, 0.5) Gaussian noise to actions for high exploration, and store all experienced transitions. This collection procedure creates a dataset with a diverse set of states and actions, with the aim of sufficient coverage.

- 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

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?

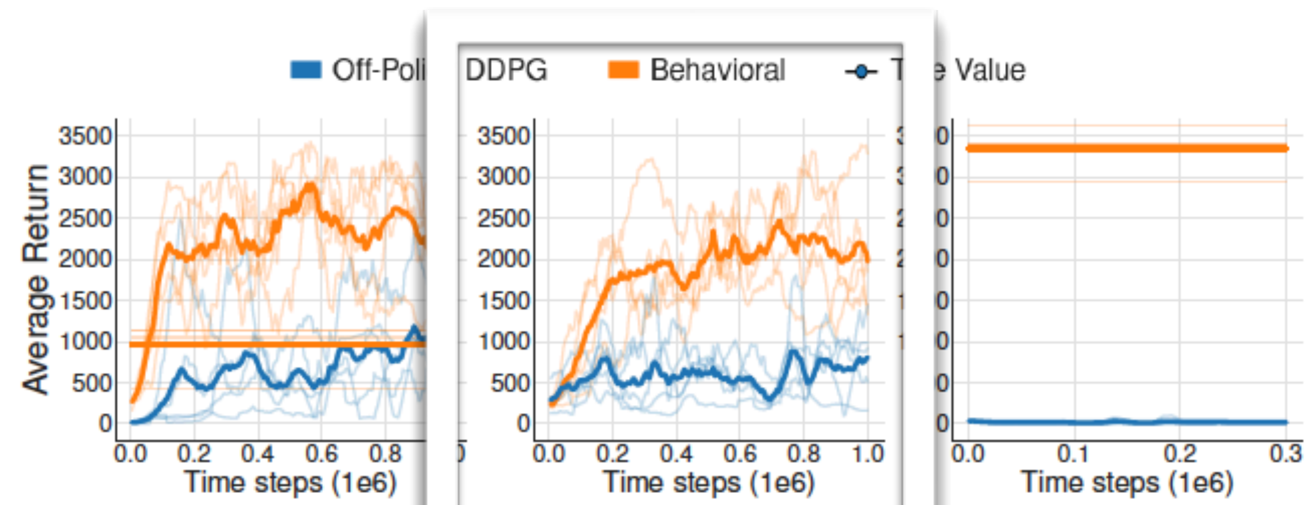
- DDPG (behavioral): (what we have seen in the course) a DDPG policy based on which actions are selected (with small exploration noise) and the experience buffer is populated.

- (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

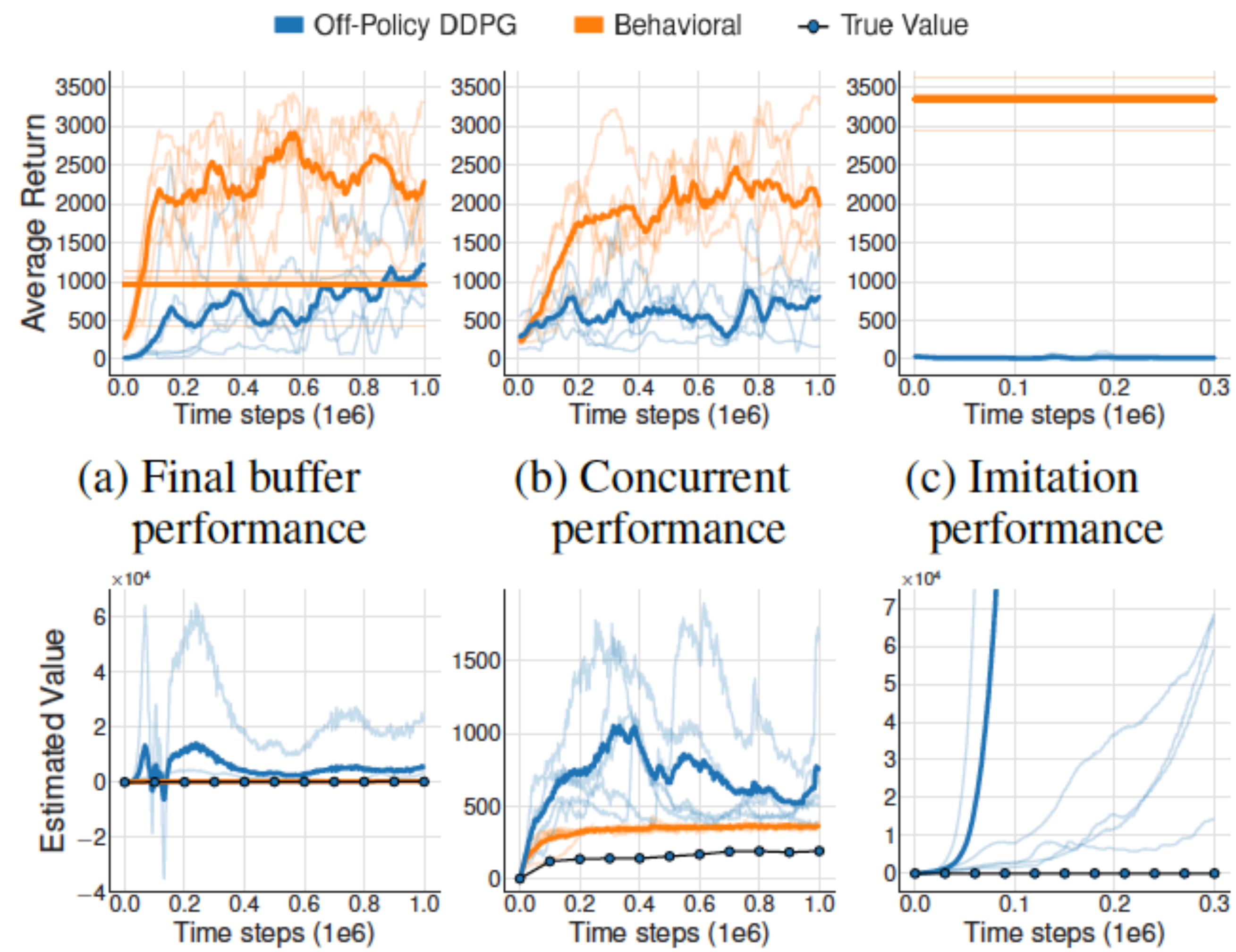
- DDPG (behavioral): (what we have seen in the course) a DDPG policy based on which actions are selected (with small exploration noise) and the experience buffer is populated.

- (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.



(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

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

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

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. $a' \sim \pi(s')$

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

Off-Policy Deep Reinforcement Learning without Exploration

Extrapolation Error

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

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

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

Off-Policy Deep Reinforcement Learning without Exploration

Extrapolation Error

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

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

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

Off-Policy Deep Reinforcement Learning without Exploration

Extrapolation Error

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

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

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:

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

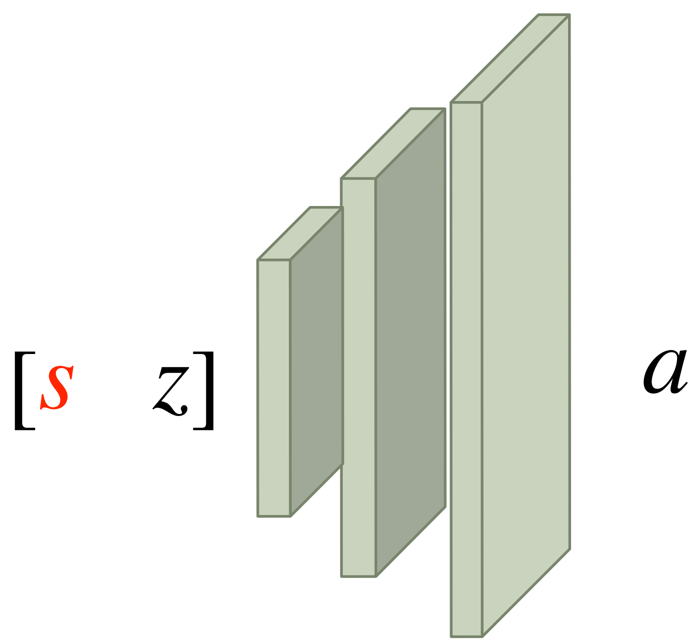
$$\{a_i \sim G_\omega(s)\}_{i=1}^n.$$

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

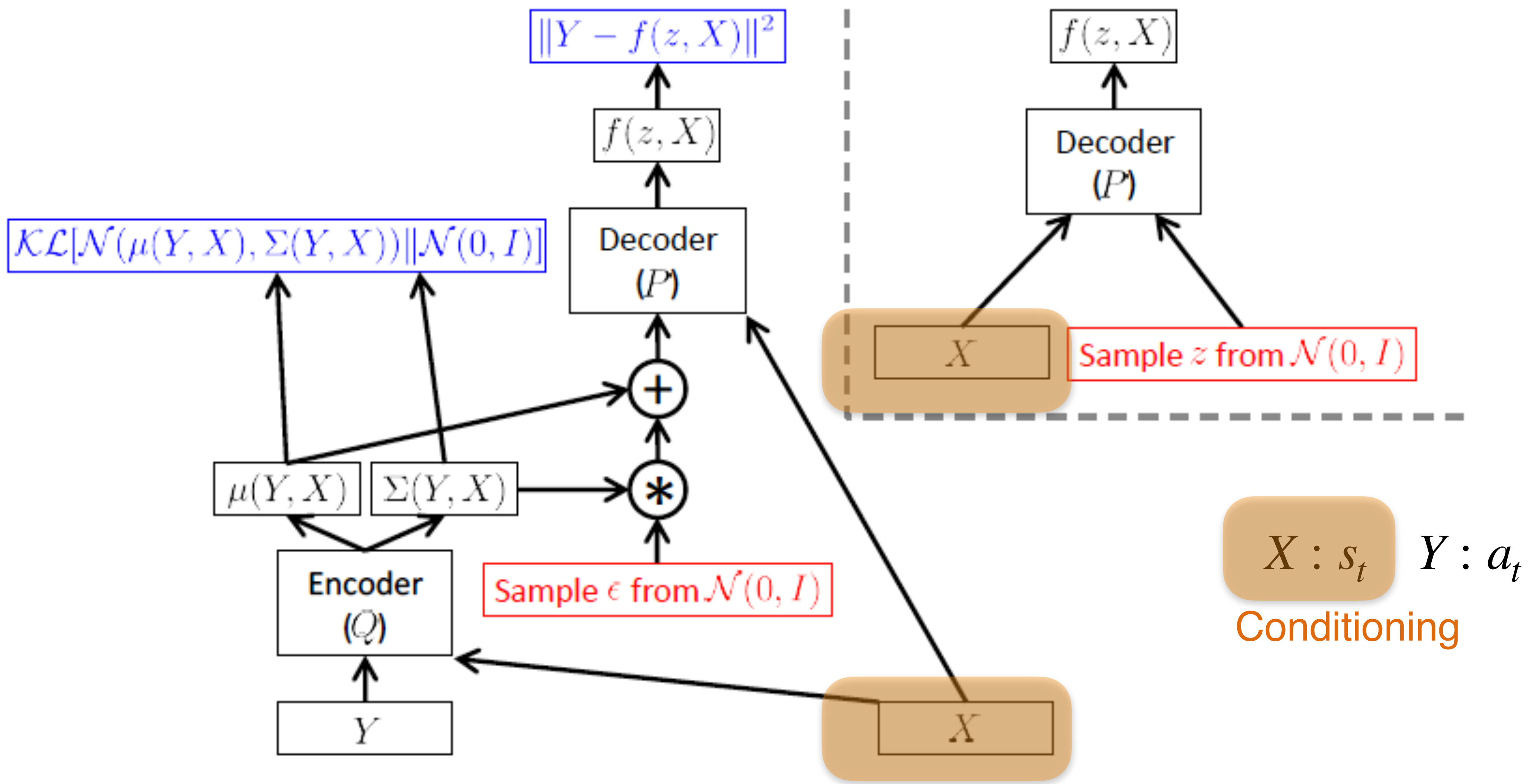
Learning stochastic generative models

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

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



Conditional VAE



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

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)

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

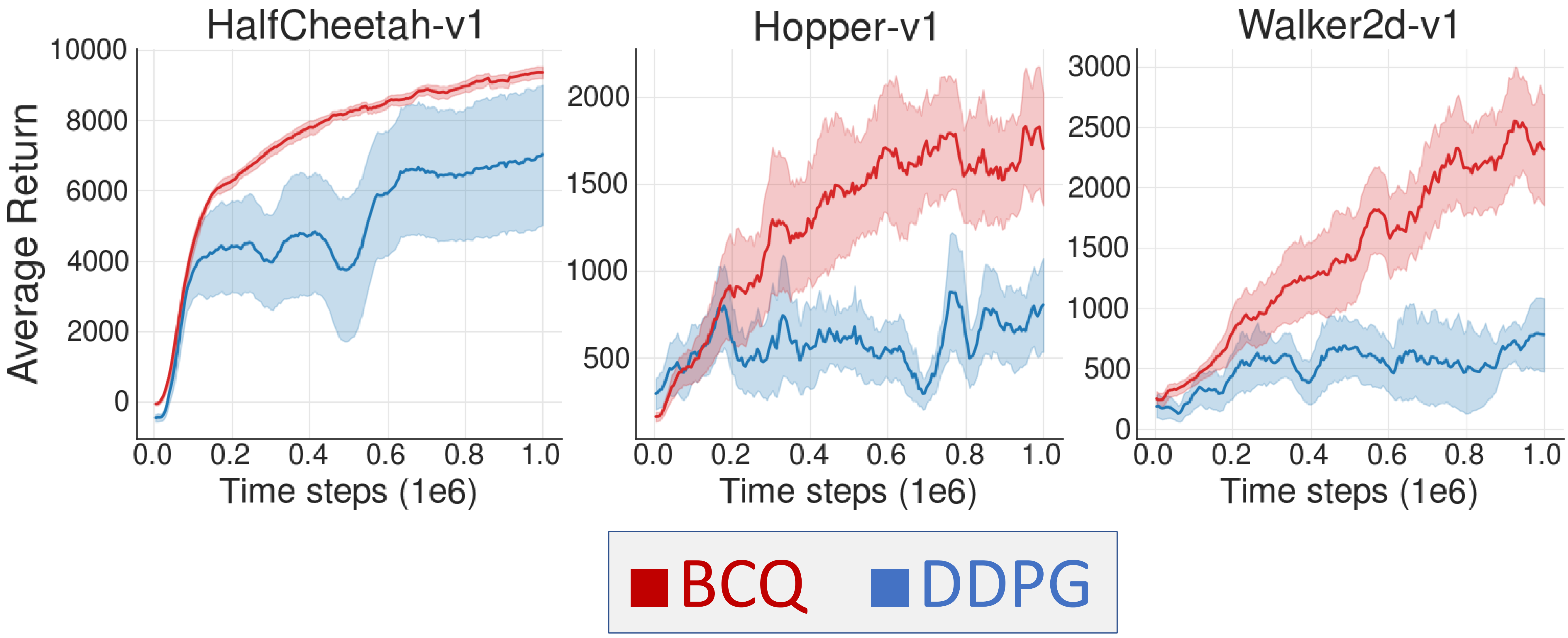
  $\phi \leftarrow \text{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

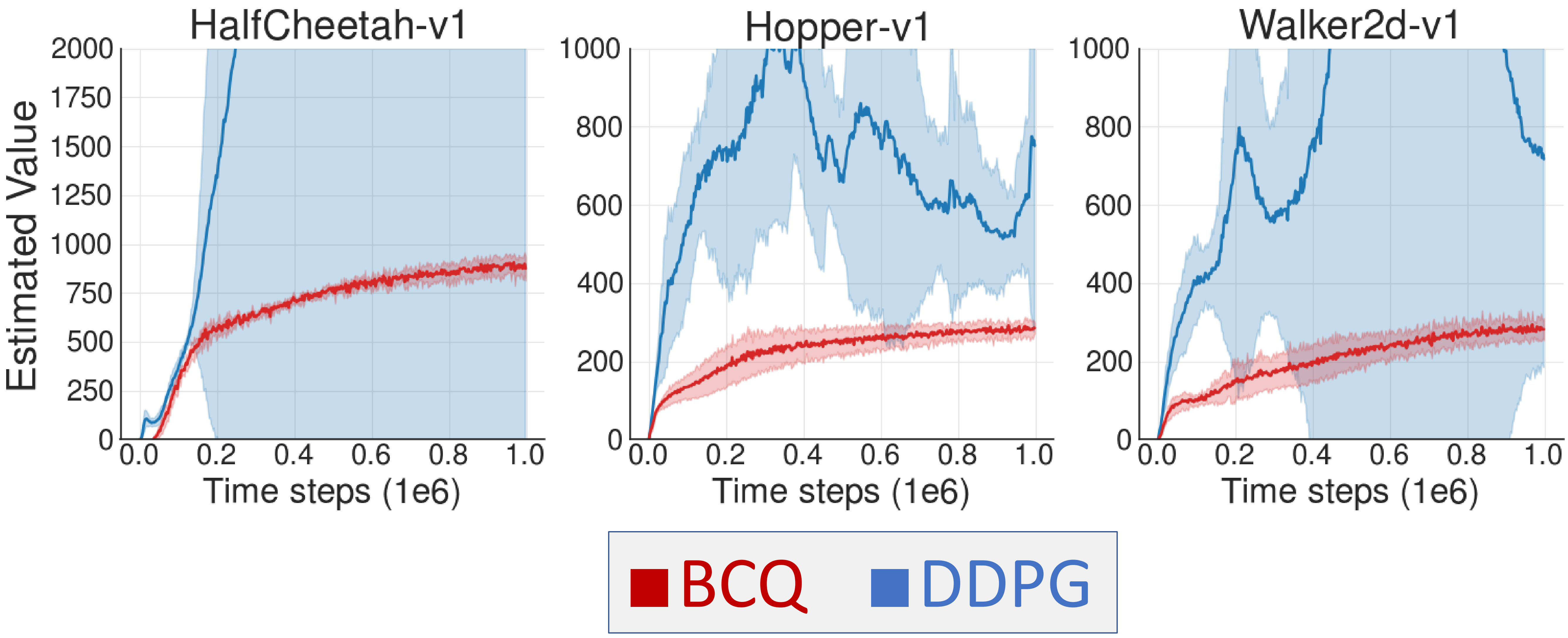
  Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$

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

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
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

Learning from demonstrations and task rewards

- Initialize the replay buffer with demos (which will be later either removed, or kept forever) and start your model-free RL method
- Pre-train the model-free RL method (a policy and a consistent with it value function) with a demonstration only buffer, then fine-tune it.
- Combine imitation and task rewards
- Exploit the temporal structure, and step progressively earlier and earlier in time along a trajectory, to solve progressively longer horizon tasks, as opposed to solving them at once.

What should be our imitation reward?

- The trained policy should match the actions of the expert on the demonstration states
- The trained policy should visit the same state distribution as the demonstration trajectories.

What should be our imitation reward?

- The trained policy should match the actions of the expert on the demonstration states
- The trained policy should visit the same state distribution as the demonstration trajectories.

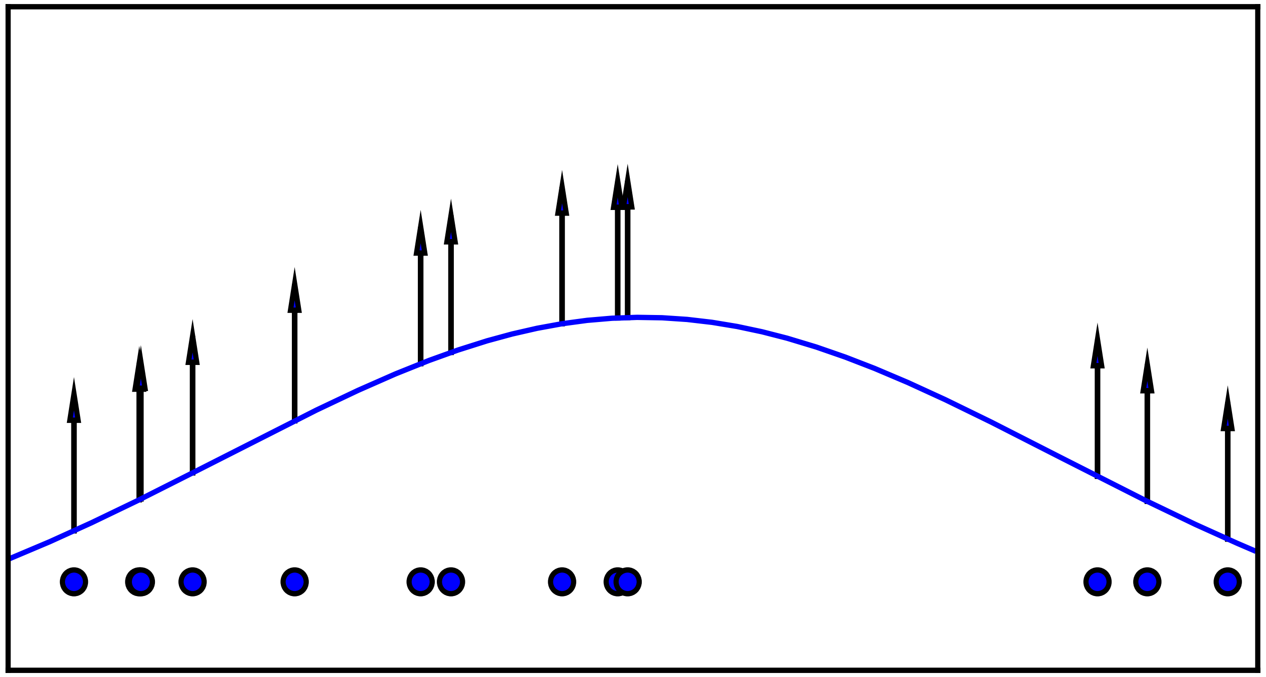
We will use generative adversarial network for state distribution matching!

State-action distribution matching objective

- The state-action distribution from the expert trajectories and the state-action distribution that the agent visits by deploying the policy in the environment need to match.

  - New solution to the compounding error problem of BC!
  - Let's see how we can optimize this distribution matching objective!

BC Maximizes Conditional Likelihood



$$\theta^* = \arg\max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} \log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$$

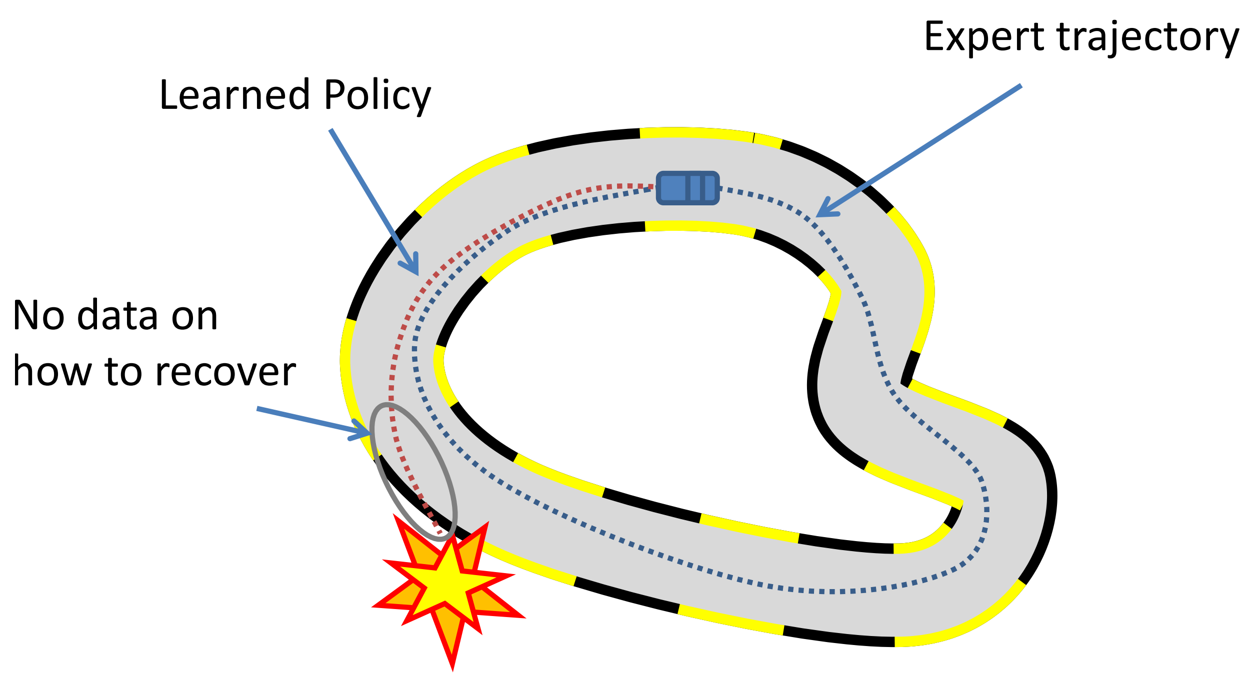
explicit density

extra conditioning information

$$\mathscr{L}_{BC}(\theta, \mathscr{T}) = \mathbb{E}_{(s_t^j, a_t^j) \sim \mathscr{T}} \left[\| a_t^j - \pi_\theta(s_t^j) \|_2^2 \right]$$

BC Maximizes Conditional Likelihood

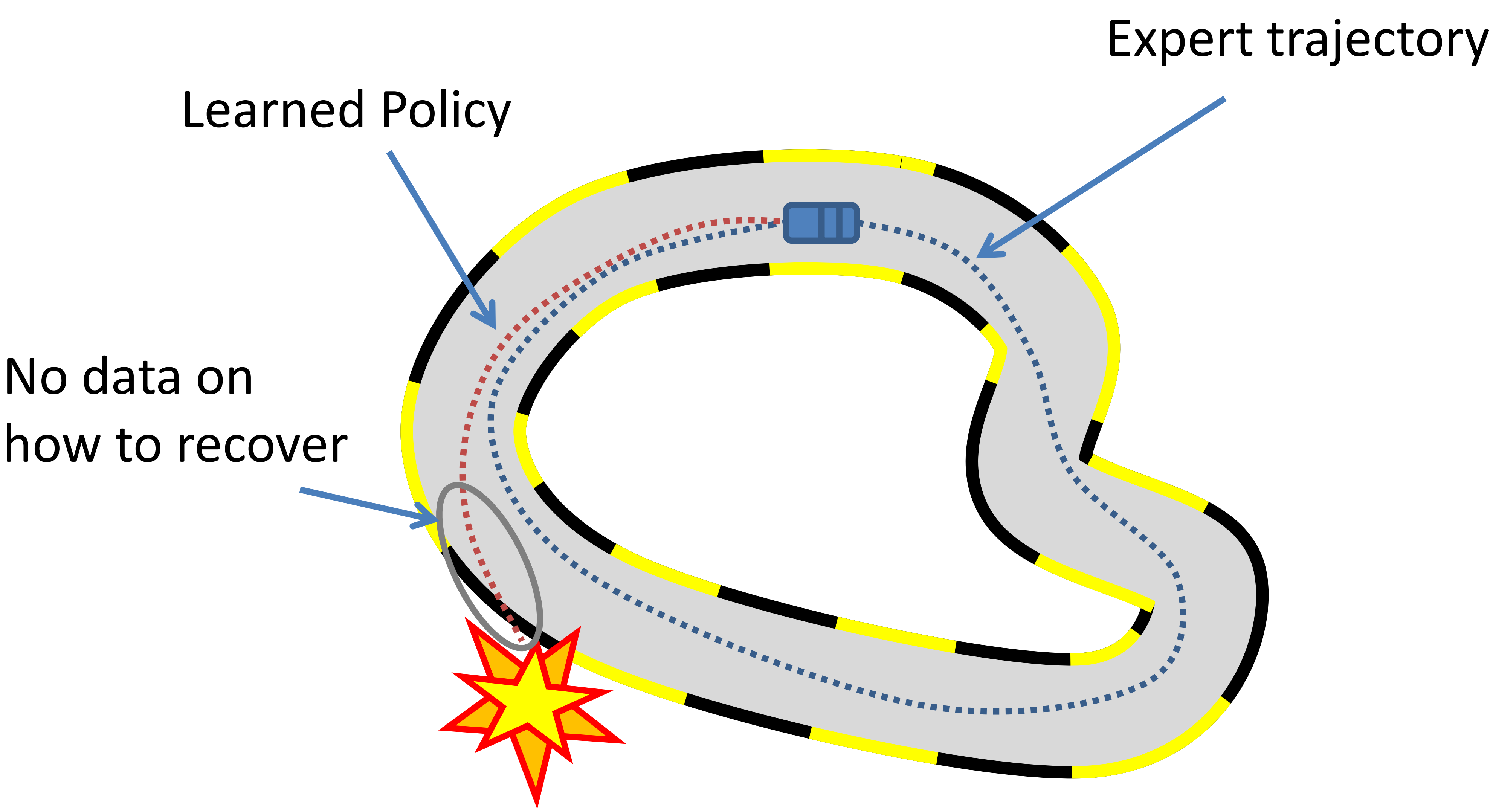
$$\mathscr{L}_{BC}(\theta, \mathscr{T}) = \mathbb{E}_{(s_t^j, a_t^j) \sim \mathscr{T}} \left[\| a_t^j - \pi_\theta(s_t^j) \|_2^2 \right]$$



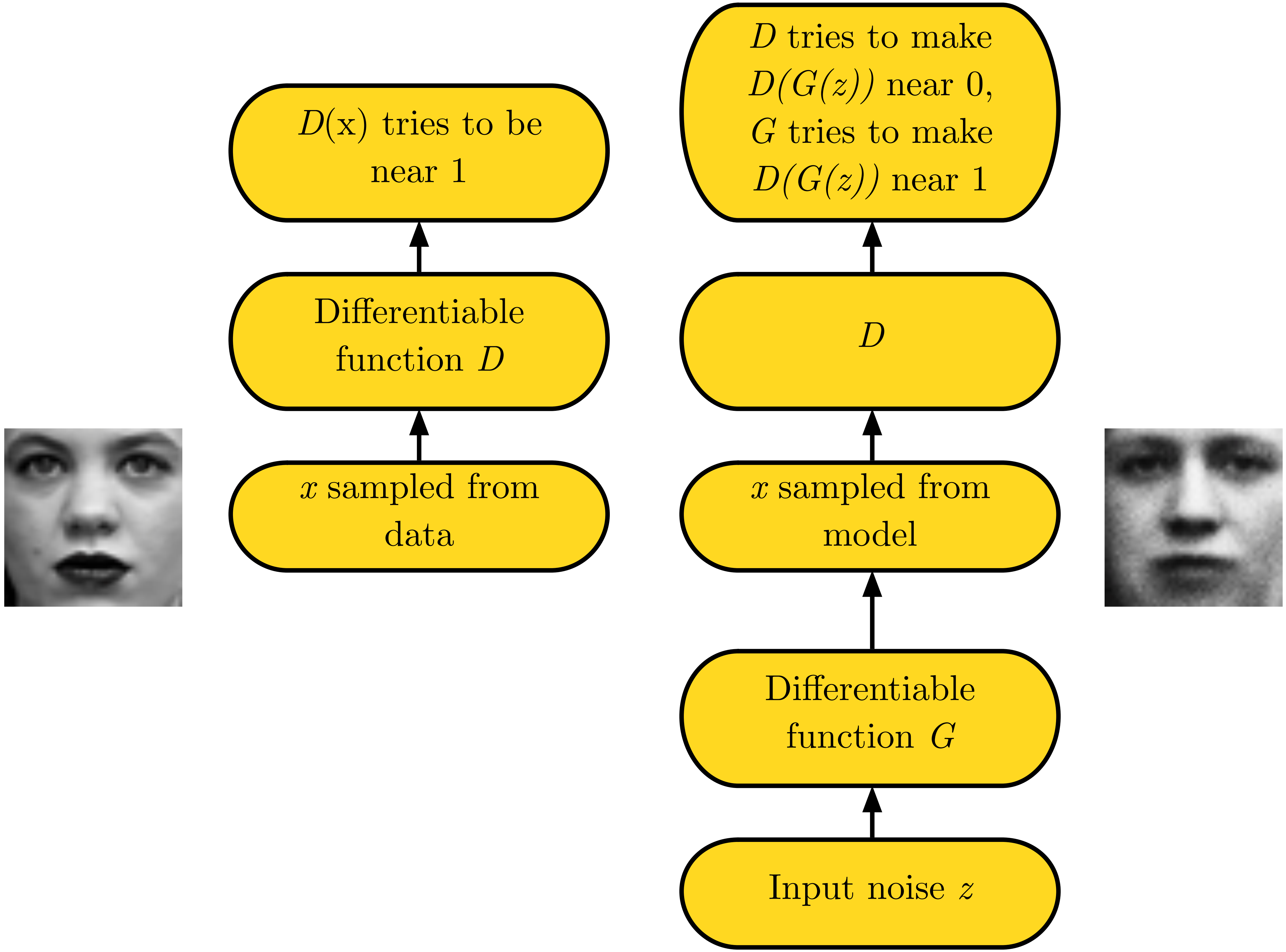
- Makes the expert actions most likely in the states of the expert trajectories.
- But what about the states not on the expert trajectories? There the actions are unconstrained!

Distribution mismatch (distribution shift)

$$P_{\pi^*}(\mathbf{o}_t) \neq P_{\pi_\theta}(\mathbf{o}_t)$$

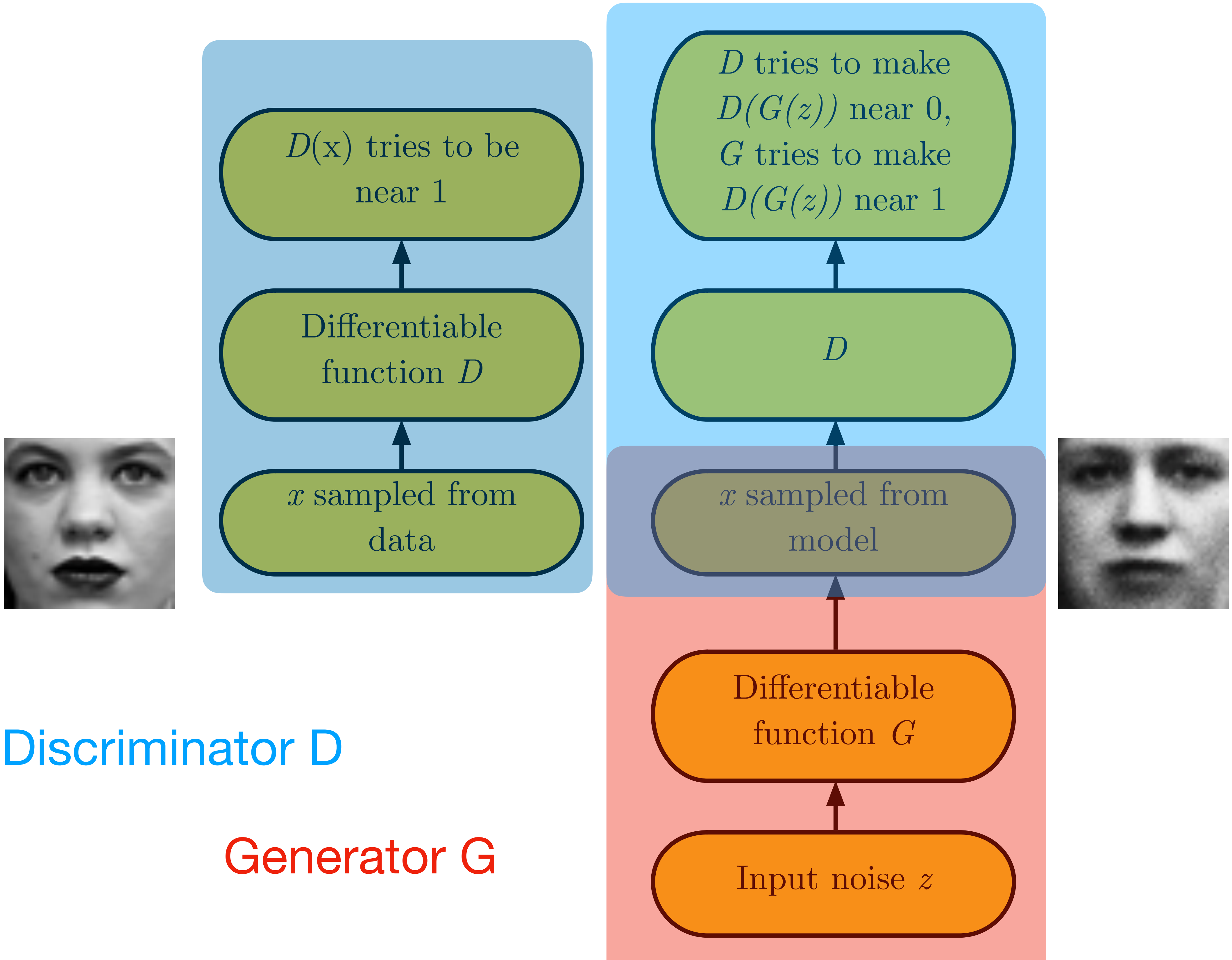


Adversarial Nets Framework



(Goodfellow 2016)

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$



(Goodfellow 2016)

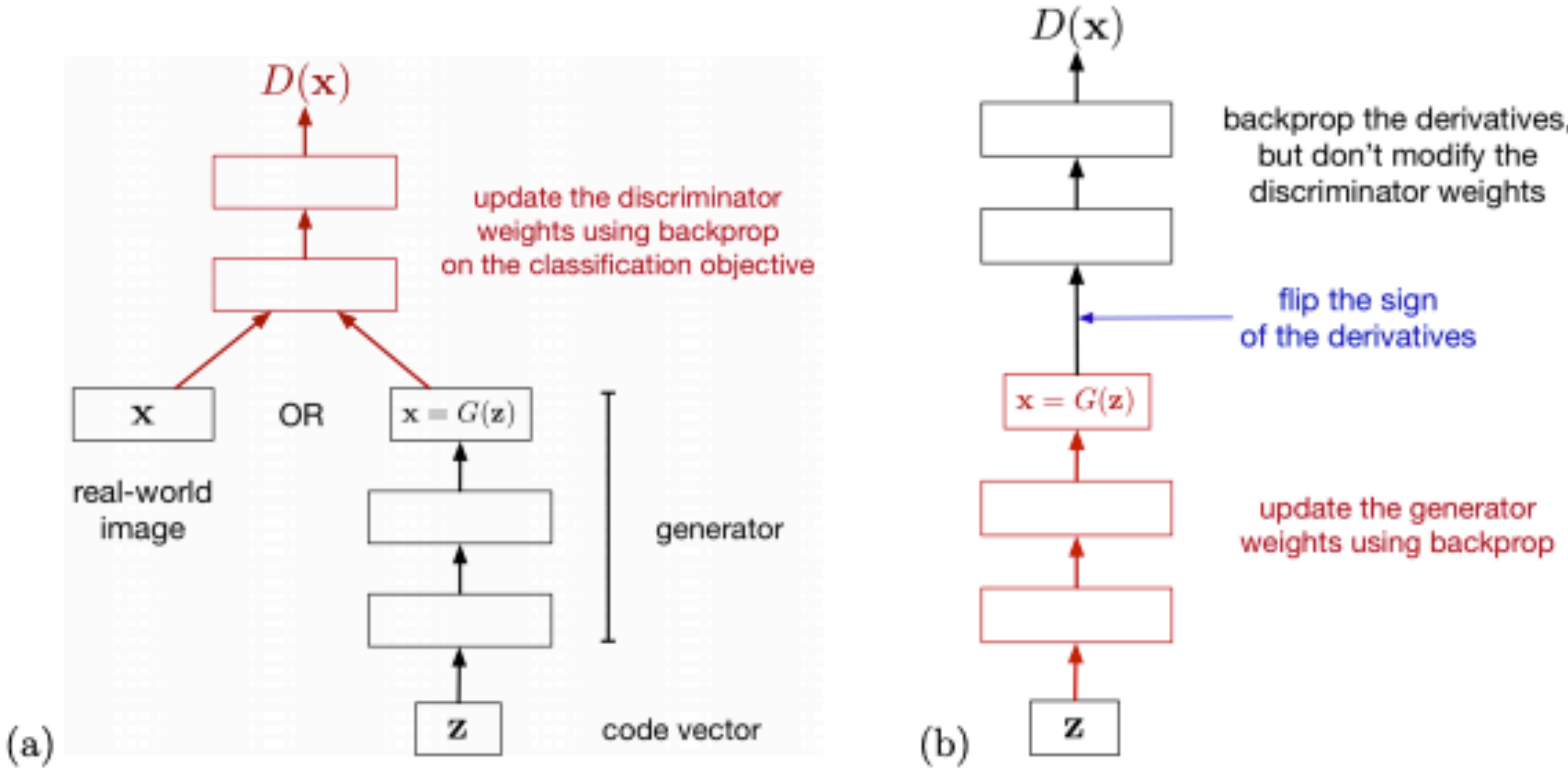
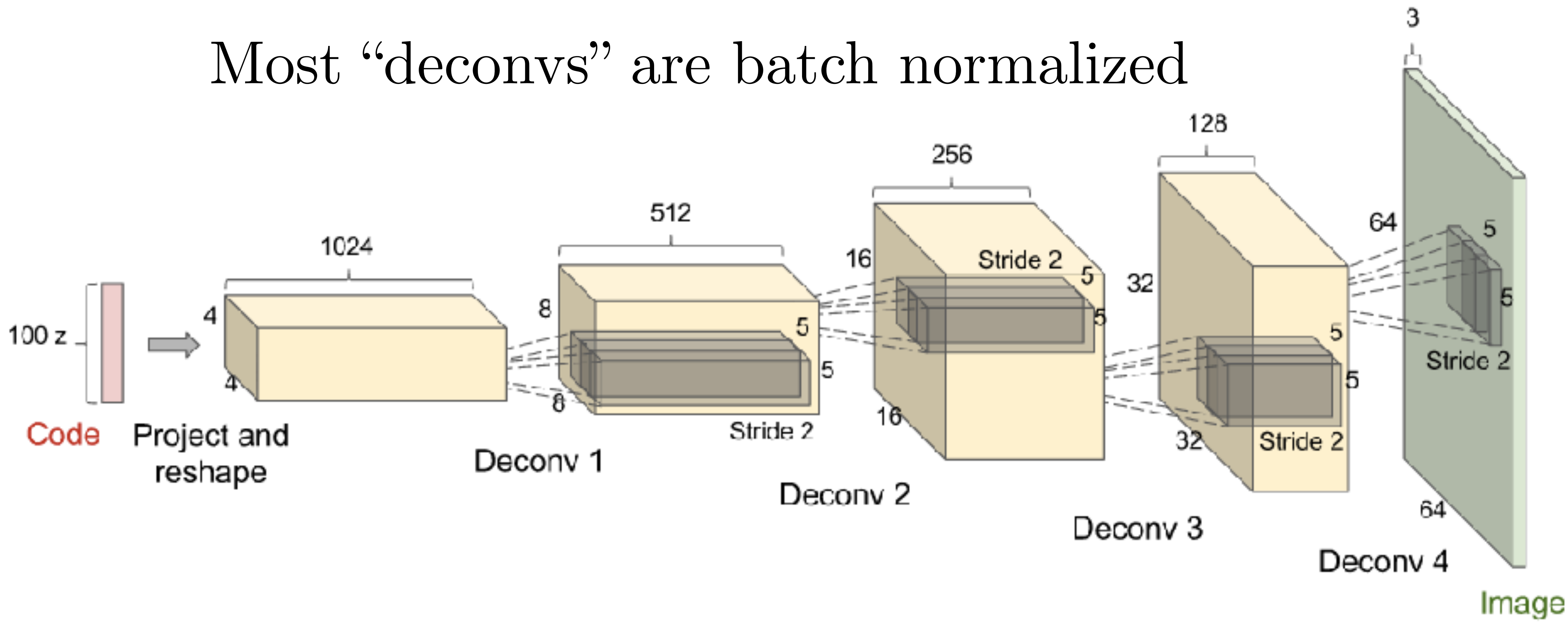


Figure 3: **(a)** Updating the discriminator. **(b)** Updating the generator.

A Generator network (DCGAN)

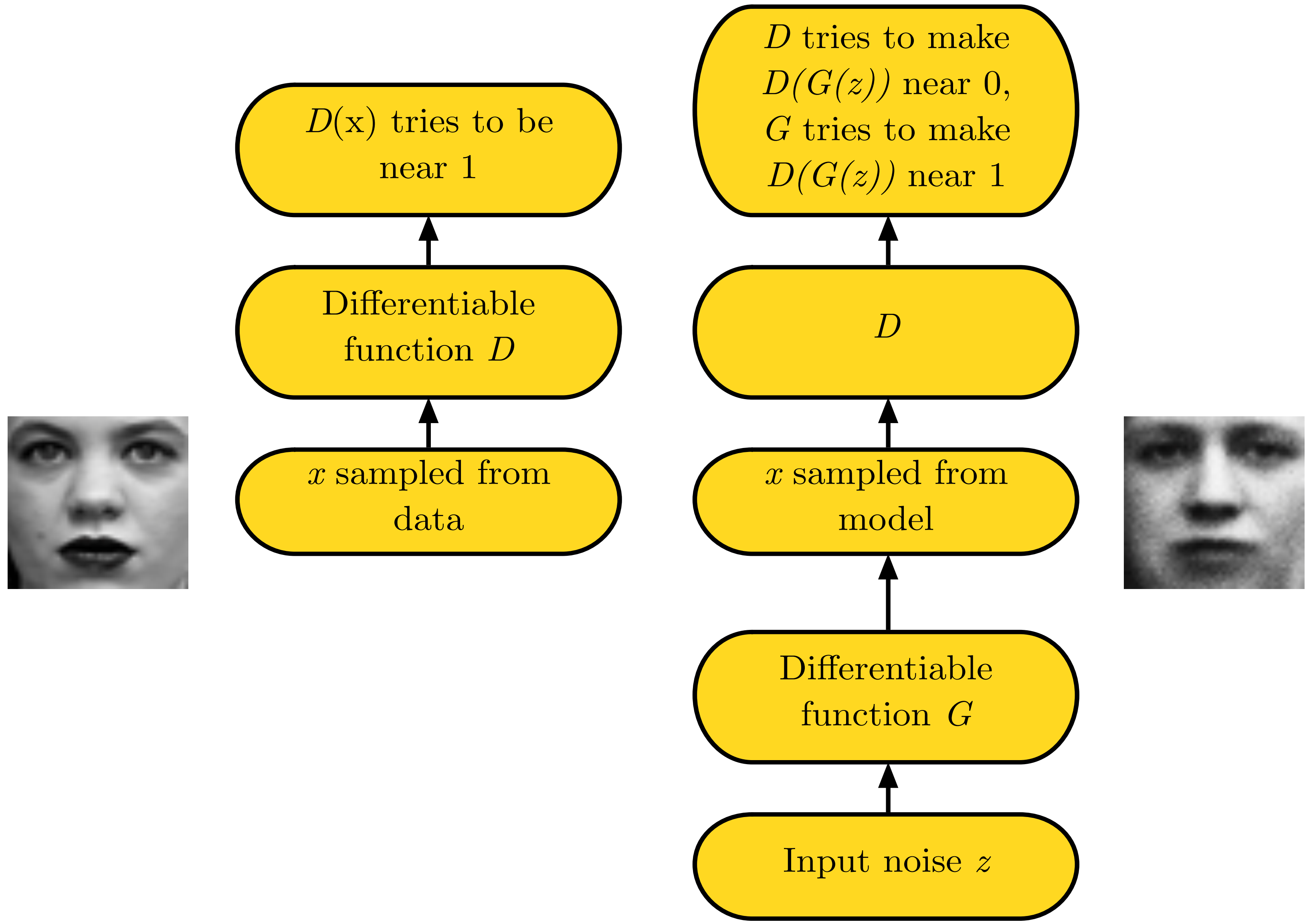


(Radford et al 2015)

Training Procedure

- Use SGD-like algorithm of choice (Adam) on two minibatches simultaneously:
  - A minibatch of training examples
  - A minibatch of generated samples
- Optional: run k steps of one player for every step of the other player.

(Goodfellow 2016)



(Goodfellow 2016)

Questions:

What if the generator maps all noise vectors to a single super photorealistic image?

What if we train the discriminator till convergence (it is just a supervised classifier…) and becomes perfect in distinguishing real from generated images?

A minimax game

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Optimal discriminator strategy

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

$$V(D,G) = \int_x p_{\text{data}}(x) \log D(x) dx + \int_z p_z(z) \log(1 - D(G(z))) dz$$

Optimal discriminator strategy

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

$$V(D, G) = \int_x p_{\text{data}}(x) \log D(x) dx + \int_z p_z(z) \log(1 - D(G(z))) dz$$

$$\int_x p_{\text{data}}(x) \log D(x) dx + \int_x p_G(x) \log(1 - D(x)) dx$$

Optimal discriminator strategy

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

$$V(D, G) = \int_x p_{\text{data}}(x) \log D(x) dx + \int_z p_z(z) \log(1 - D(G(z))) dz$$

$$= \int_x p_{\text{data}}(x) \log D(x) dx + \int_x p_G(x) \log(1 - D(x)) dx$$

$$= \int_x p_{\text{data}}(x) \log D(x) + p_G(x) \log(1 - D(x)) dx$$

Optimal discriminator strategy

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x))dx$$

The discriminator assigns values D(x) to each image x. Let's take the derivative to see where the optimum is attained.

Optimal discriminator strategy

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1 - D(x))dx$$

$$\frac{d}{dD(x)}\left(p_{\text{data}}(x)\log D(x) + p_G(x)\log(1 - D(x))\right) = 0$$

Optimal discriminator strategy

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1 - D(x))dx$$

$$\frac{d}{dD(x)}\big(p_{\text{data}}(x)\log D(x) + p_G(x)\log(1 - D(x))\big) = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} - p_G(x)\frac{1}{1 - D(x)} = 0$$

Optimal discriminator strategy

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1 - D(x))dx$$

$$\frac{d}{dD(x)}\left(p_{\text{data}}(x)\log D(x) + p_G(x)\log(1 - D(x))\right) = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} - p_G(x)\frac{1}{1 - D(x)} = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} = p_G(x)\frac{1}{1 - D(x)}$$

Optimal discriminator strategy

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1 - D(x))dx$$

$$\frac{d}{dD(x)}\left(p_{\text{data}}(x)\log D(x) + p_G(x)\log(1 - D(x))\right) = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} - p_G(x)\frac{1}{1 - D(x)} = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} = p_G(x)\frac{1}{1 - D(x)}$$

$$\Leftrightarrow p_{\text{data}}(x)(1 - D(x)) = p_G(x)D(x)$$

Optimal discriminator strategy

$$V(D, G) = \int_x p_{\text{data}}(x) \log D(x) + p_G(x) \log(1 - D(x))dx$$

$$\frac{d}{dD(x)} \left(p_{\text{data}}(x) \log D(x) + p_G(x) \log(1 - D(x))\right) = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} - p_G(x)\frac{1}{1 - D(x)} = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} = p_G(x)\frac{1}{1 - D(x)}$$

$$\Leftrightarrow p_{\text{data}}(x)(1 - D(x)) = p_G(x)D(x)$$

$$\Leftrightarrow D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$$

Optimal generator strategy

$$C(G) = \max_D V(G, D)$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))]
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)]
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D^*_G(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D^*_G(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D^*_G(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D^*_G(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})]
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D^*_G(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D^*_G(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D^*_G(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D^*_G(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})]
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 + \log 4
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 + \log 4 \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{2p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{2p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 + \log 4 \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{2p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{2p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}}] + \mathbb{E}_{x \sim p_G(x)}[\log \frac{p_{\text{G}}(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}}] - \log 4
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x\sim p_G(x)}[\log(1 - D_G^*(x)] \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x\sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x\sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x\sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 + \log 4 \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log \frac{2p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x\sim p_G(x)}[\log(\frac{2p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}}] + \mathbb{E}_{x\sim p_G(x)}[\log \frac{p_{\text{G}}(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}}] - \log 4 \\
&= \mathrm{D}_{\mathrm{KL}}\left(p_{data}(x)\,||\,\frac{p_{\text{data}}(x) + p_G(x)}{2}\right) + \mathrm{D}_{\mathrm{KL}}\left(p_G(x)\|\frac{p_{\text{data}}(x) + p_G(x)}{2}\right) - \log 4
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 + \log 4 \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{2p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{2p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}}] + \mathbb{E}_{x \sim p_G(x)}[\log \frac{p_{\text{G}}(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}}] - \log 4 \\
&= \mathrm{D}_{\mathrm{KL}}\left(p_{data}(x) \,||\, \frac{p_{\text{data}}(x) + p_G(x)}{2}\right) + \mathrm{D}_{\mathrm{KL}}\left(p_G(x) \| \frac{p_{\text{data}}(x) + p_G(x)}{2}\right) - \log 4 \\
&= 2\mathrm{D}_{\mathrm{JSD}}\left(p_{data}(x) \,||\, p_G(x)\right) - \log 4
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x\sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= 2\text{D}_{\text{JSD}}\left(p_{data}(x) || p_G(x)\right) - \log 4
\end{aligned}
$$

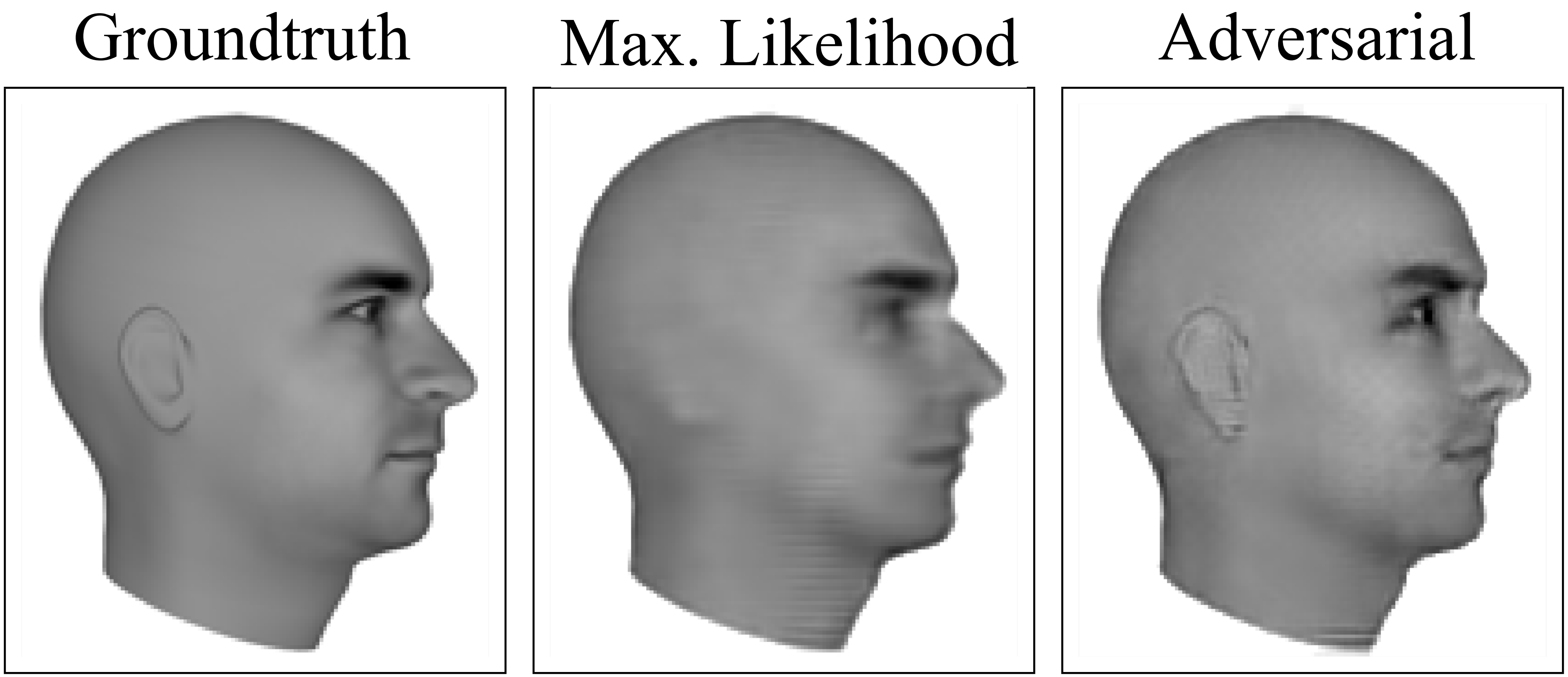
Since $\text{D}_{\text{JSD}} \geq 0, \quad C(G) \geq -\log 4$

By setting $P_G(x) = \text{p}_{\text{data}}(\text{x})$ in the equation above, we get:

$$
C(G) = \mathbb{E}_{x\sim \text{p}_{\text{data}}(x)} \log \frac{1}{2} + \mathbb{E}_{x\sim p_G(x)} \log \frac{1}{2} = -\log 4
$$

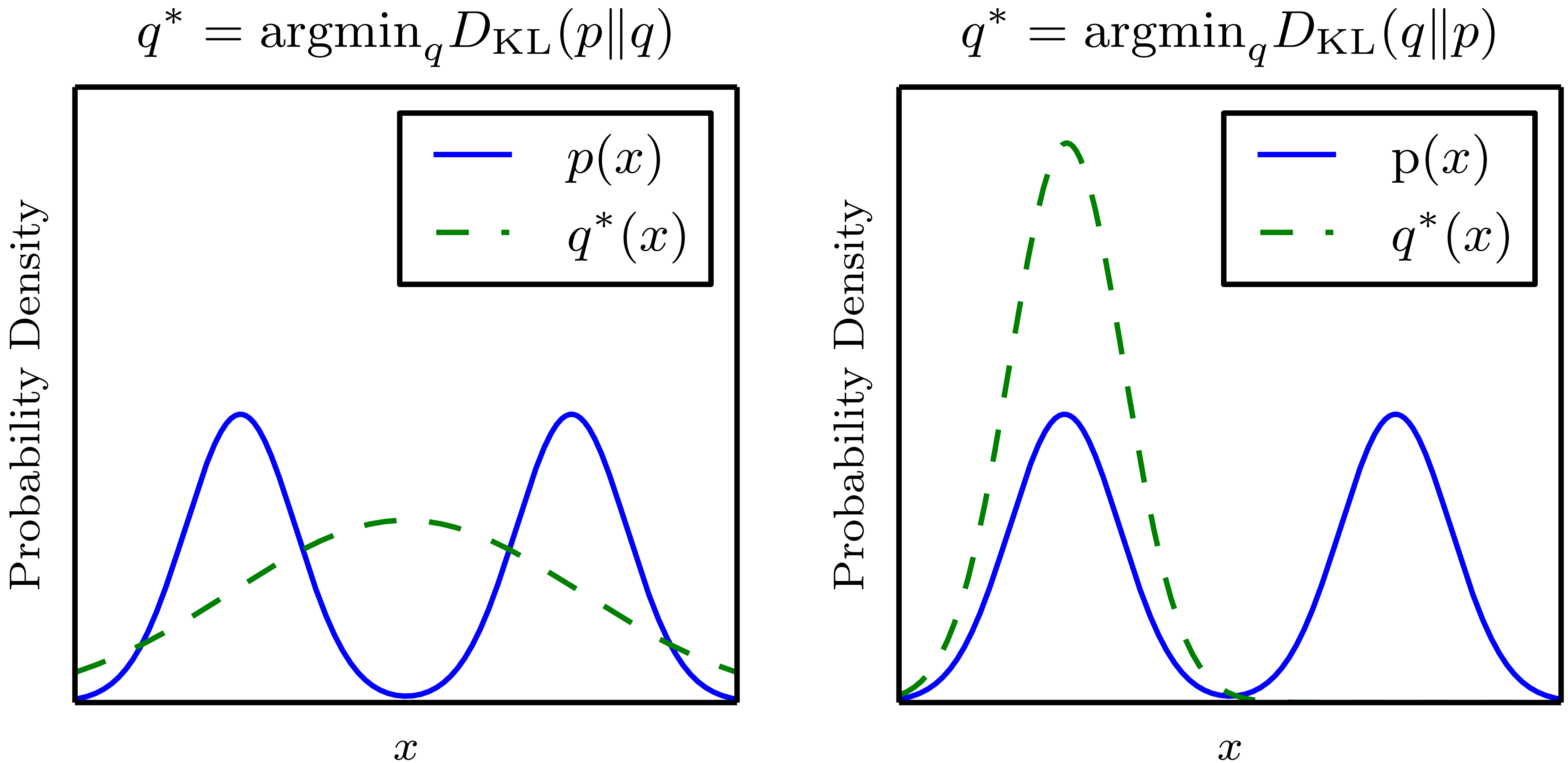
Thus generator achieves the optimum when $P_G(x) = \text{p}_{\text{data}}(\text{x})$.

Next Video Frame Prediction



(Lotter et al 2016)

Maybe an explanation of why GANs work



Maximum likelihood

Reverse KL

Generative Adversarial Imitation learning

The policy network will be our generator, that conditions on the state:

$$\pi_\theta(s) \rightarrow a$$

Generative Adversarial Imitation learning

Find a policy π_θ that makes it impossible for a discriminator network to distinguish between state-actions from the expert demonstrations and state-action pairs visited by the agent's policy π_θ:

$$\min_{\pi_\theta} \quad \mathbb{E}_{(s,a)\sim\pi_\theta}[-\log(D_\phi(s,a))]$$

$$\min_{D_\phi} \quad \mathbb{E}_{(s,a)\sim \text{Demo}}[\log(1-D_\phi(s,a))] + \mathbb{E}_{(s,a)\sim\pi_\theta}[\log(D_\phi(s,a))]$$

The reward for the policy optimization is how well I matched the demonstrator's trajectory distribution, else, how well I confused the discriminator.

$$r(s,a) = \log D_\phi(s,a), (s,a) \sim \pi_\theta$$

Generative Adversarial Imitation learning

Input: Expert trajectories , initial policy parameters θ_0 and initial discriminator weights ϕ_0.

For i=0,1,2,3… **do**

1. Sample agent trajectories $\tau_i \sim \pi_{\theta_i}$
2. Update the discriminator parameters with the gradient:

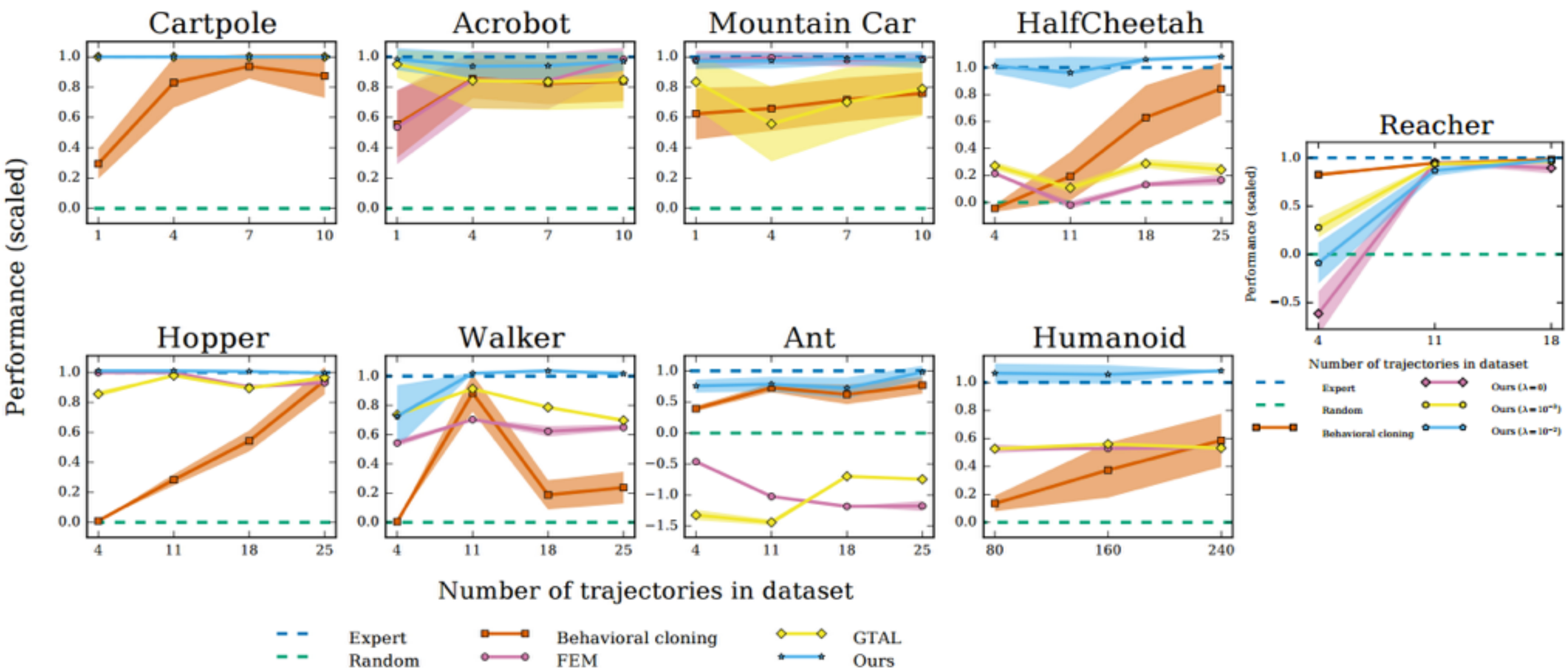
$$\mathbb{E}_{(s,a)\sim \text{Demo}}[\nabla_\phi \log(1 - D_\phi(s,a))] + \mathbb{E}_{(s,a)\in\tau_i}[\nabla_\phi \log(D_\phi(s,a))]$$

3. Update the policy using a policy gradient computed with the rewards, e.g., the REINFORCE policy gradient would be:

$$\mathbb{E}_{(s,a)\in\tau_i}[\nabla_\theta \log \pi_\theta \log D_{\phi_{i+1}}(s,a)]$$

end for

Generative Adversarial Imitation learning



- GAIL: a reinforcement learning method with a reward based on trajectory distribution matching between the agent and an expert.
- BC: reduces imitation learning to supervised learning for individual actions.
- GAIL performs better than behaviour cloning but it requires MORE interactions with the environment.
- Q:Can BC or GAIL outperform the expert?

Combining imitation and task rewards

$$r(s, a) = \lambda r_{GAIL}(s, a) + (1 - \lambda) r_{task}(s, a), \quad \lambda \in [0,1] .$$

Combining imitation and task rewards

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

$$r(s, a) = \lambda r_{GAIL}(s, a) + (1 - \lambda) r_{task}(s, a), \quad \lambda \in [0,1] \, .$$

$$r_{GAIL}(s, a) = -\log(1 - D(s, a))$$

Reinforcement and Imitation Learning for Diverse Visuomotor Skills

Yuke Zhu[†] Ziyu Wang[‡] Josh Merel[‡] Andrei Rusu[‡] Tom Erez[‡] Serkan Cabi[‡]
Saran Tunyasuvunakool[‡] János Kramár[‡] Raia Hadsell[‡] Nando de Freitas[‡] Nicolas Heess[‡]
[†]Computer Science Department, Stanford University, USA
[‡]DeepMind, London, UK

- Combine imitation and task rewards.

- Start episodes by setting the world in states of the demonstration trajectories. This means we can reset the world however we like, and that we have full state information to be able to set our simulator to it. (Have we done this earlier?)

- Asymmetric actor-critic: the value network takes as input the low-dim state of the system and the policy is trained from pixels.

- Only scene state info to the discriminator

- Co-train the policy CNN with auxiliary task

- Sim2REAL via domain randomization.

Reinforcement and Imitation Learning for Diverse Visuomotor Skills

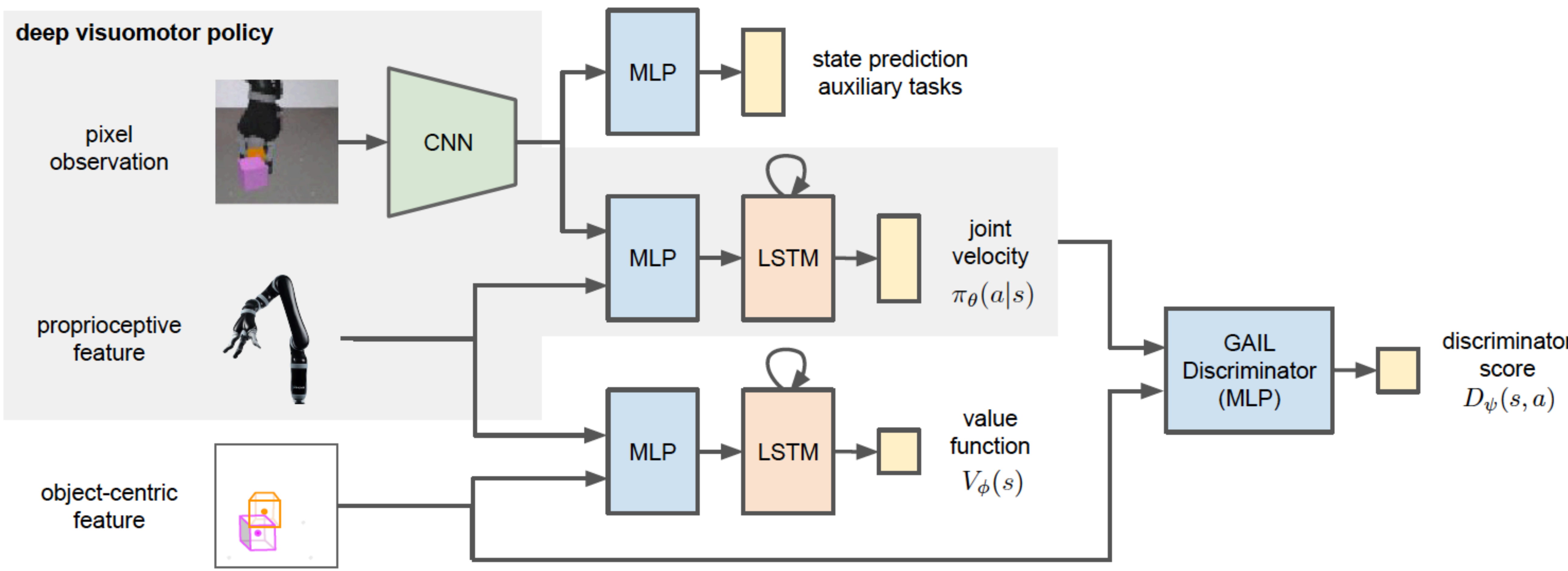
Yuke Zhu† Ziyu Wang‡ Josh Merel‡ Andrei Rusu‡ Tom Erez‡ Serkan Cabi‡
Saran Tunyasuvunakool‡ János Kramár‡ Raia Hadsell‡ Nando de Freitas‡ Nicolas Heess‡
†Computer Science Department, Stanford University, USA
‡DeepMind, London, UK

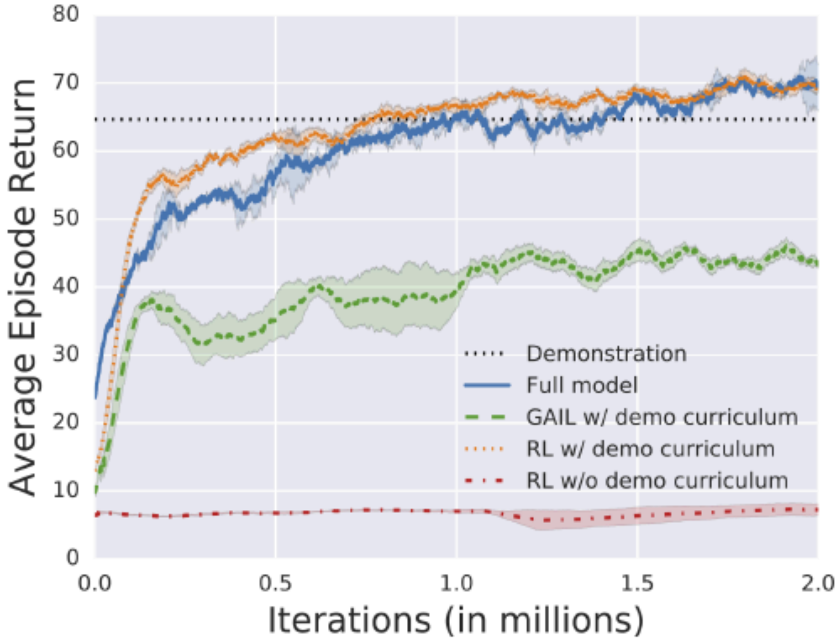
- Input: video demonstrations (without rewards Combine imitation and task rewards.
- Start episods by setting the world in states of the demonstration trajectories. This means we can reset the world however we like, and that we have full state information to be able to set our simulator to it. (Have we done this earlier?)
- Asymetric actor-critic: the value network takes as input the low-dim state of the system (3D object location and velocities and relative distances between objects and the gripper) and the policy is trained from pixels directly. This means we need to have access to such state information at training time, but not at test time.
- Only scene state info to the discriminator
- Co-train the policy CNN with auxiliary task
- Sim2REAL via domain randomization.

Reinforcement and Imitation Learning for Diverse Visuomotor Skills

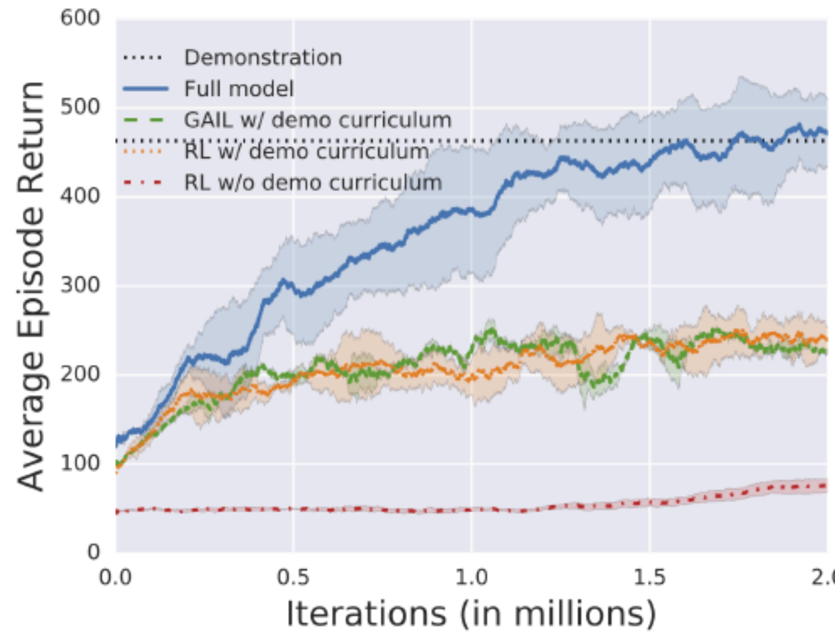
Yuke Zhu[†] Ziyu Wang[‡] Josh Merel[‡] Andrei Rusu[‡] Tom Erez[‡] Serkan Cabi[‡]
Saran Tunyasuvunakool[‡] János Kramár[‡] Raia Hadsell[‡] Nando de Freitas[‡] Nicolas Heess[‡]
[†]Computer Science Department, Stanford University, USA
[‡]DeepMind, London, UK

- Combine imitation and task rewards.
- Start episods by setting the world in states of the demonstration trajectories.
- Assymetric actor-critic: the value network takes as input the low-dim state of the system and the policy is trained from pixels.
- Only scene state info to the discriminator
- Co-train the policy CNN with auxiliary task: map images to object locationswith regression and minimize L2 loss. Any object detection/semantic labelling task would work, e.g., learning to detect the robot's gripper is also a useful auxiliary task for training the visual features.
- Sim2REAL via domain randomization.

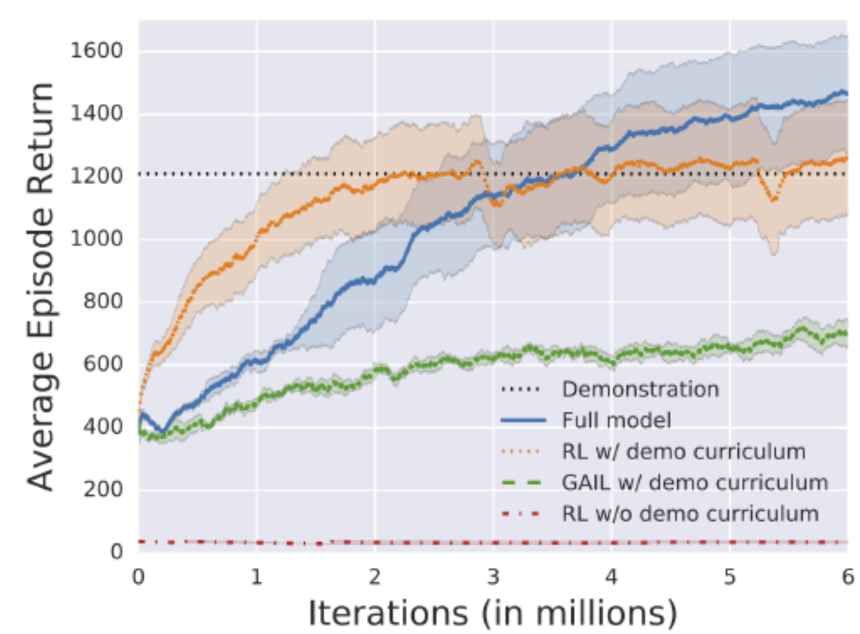

deep visuomotor policy
pixel observation
CNN
MLP
state prediction auxiliary tasks
MLP
LSTM
joint velocity
$\pi_\theta(a|s)$
proprioceptive feature
MLP
LSTM
value function
$V_\phi(s)$
object-centric feature
GAIL Discriminator (MLP)
discriminator score
$D_\psi(s,a)$



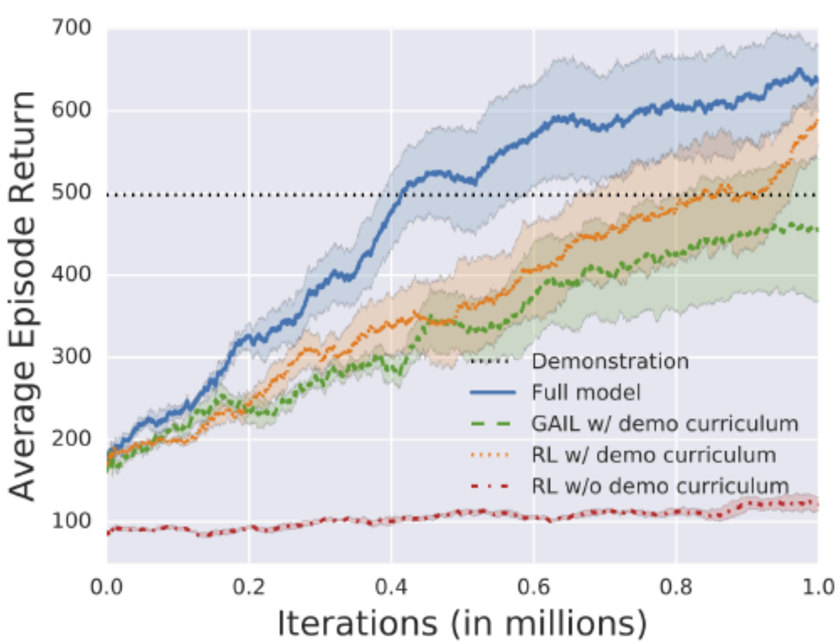
(a) Block lifting



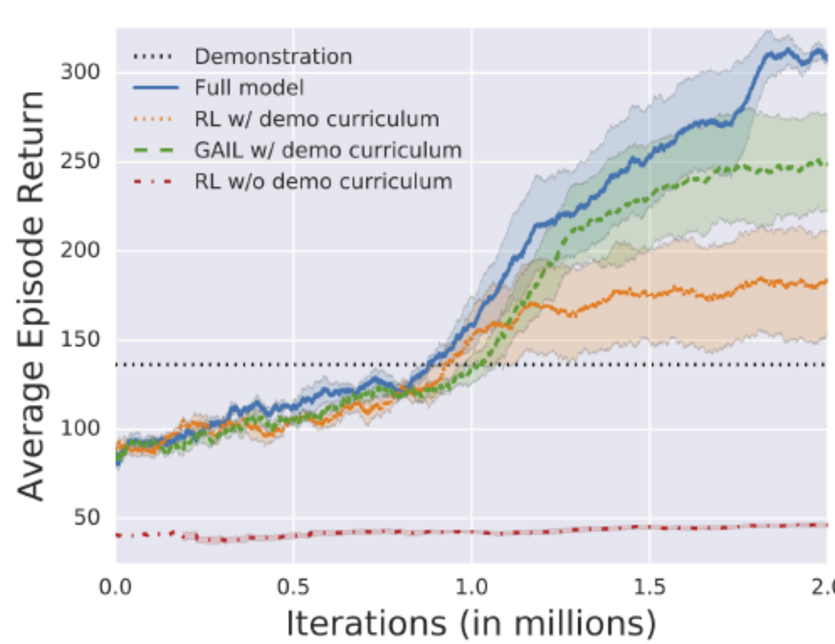
(b) Block stacking



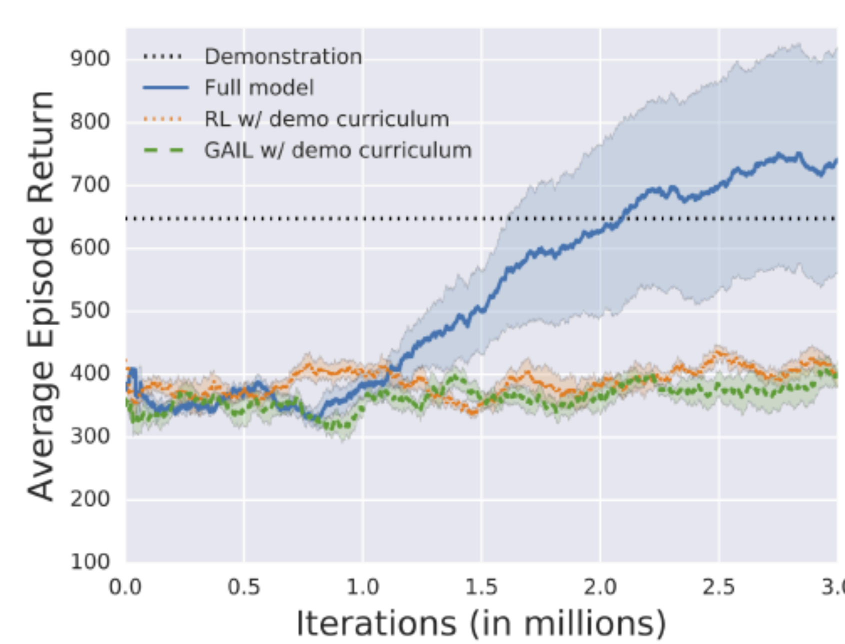
(c) Clearing table with blocks



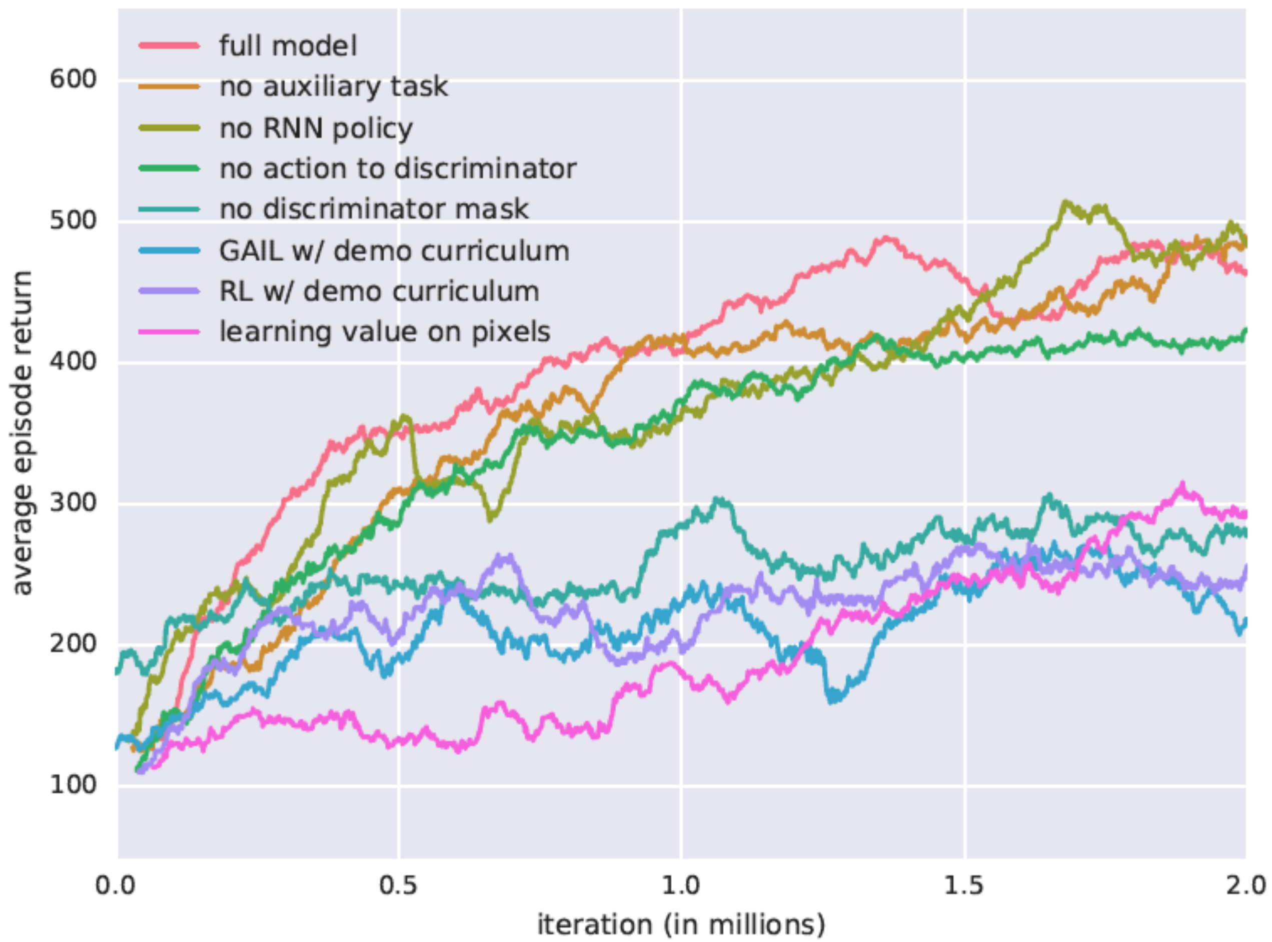
(d) Clearing table with a box



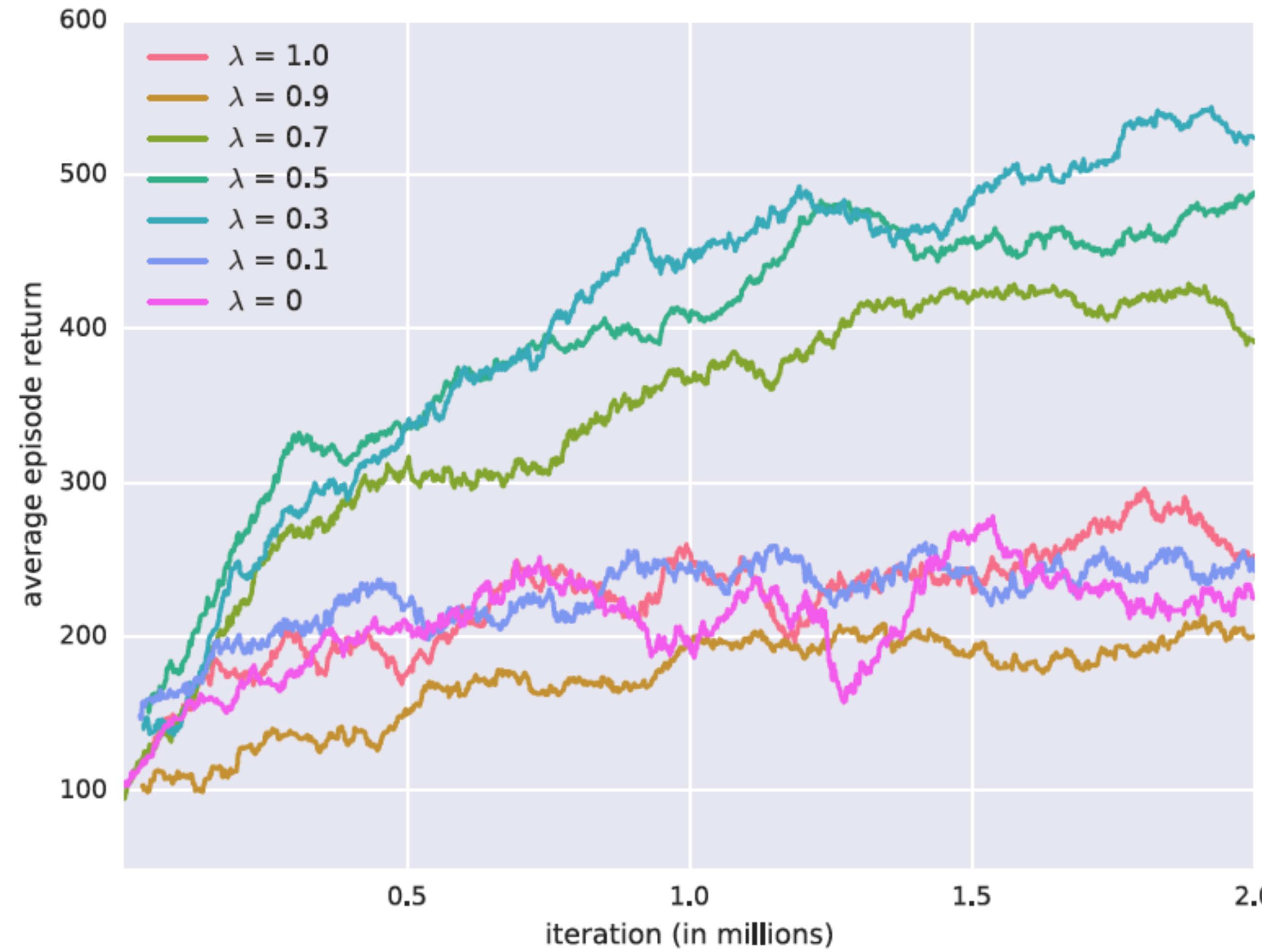
(e) Pouring liquid



(f) Order fulfillment



(a) Ablation study of model components



(b) Model sensitivity to λ values

- Learning value function from pixels directly is slow
- Not using the GAIL imitation reward but rather using demos just to start episodes in demo states is slow
- No task reward (just imitation) seems not to work. Why?
- No RNN policy: no problem, RNNs are not great way to integrate info over visual frames.
- No auxiliary task: not big problem.
- Not masking arm info from the discriminator creates problems