Carnegie Mellon

School of Computer Science

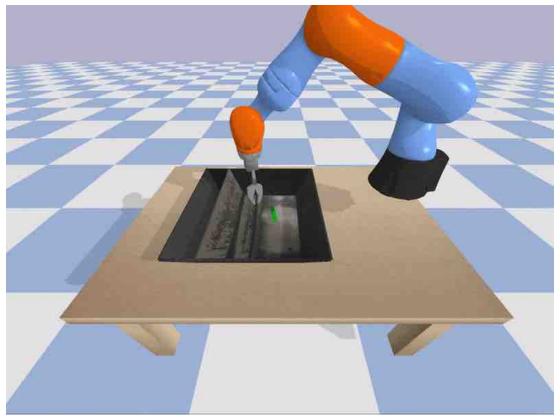
Deep Reinforcement Learning and Control

Multigoal RL

CMU 10-703

Katerina Fragkiadaki

So far we train one policy/value function per task, e.g., win the game of Tetris, win the game of Go, reach to a *particular* location, put the green cube inside the gray bucket, etc.



Universal value function Approximators

$$V(s;\theta) \rightarrow V(s,g;\theta)$$

$$\pi(s;\theta) \rightarrow \pi(s,g;\theta)$$

- All methods we have learnt so far can be used.
- At the beginning of an episode, **we sample not only a start state but also a goal g**, which stays constant throughout the episode
- The experience tuples should contain the goal.

$$(s,a,r,s') \rightarrow \left(s,g,a,r,s'\right)$$

Universal Value Function Approximators, Schaul et al.

Universal value function Approximators

$$V(s, \theta) \rightarrow V(s, g; \theta)$$

$$\pi(s; \theta) \rightarrow \pi(s, g; \theta)$$

What should be my goal representation?

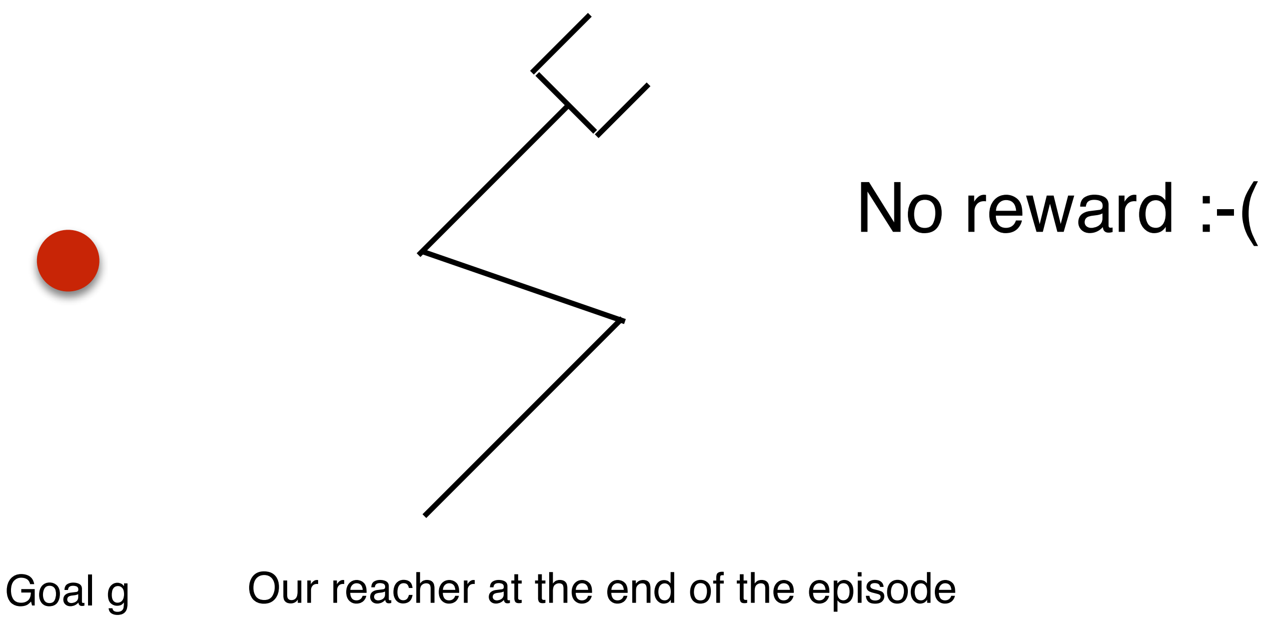
The goal representation is usually the same as your state representation. Usually one of the two:

- **Manual/oracle**: 3d centroids of objects, robot joint angles and velocities, 3d location of the gripper, etc.
- **Learnt**: Some feature encoding over images directly

Hindsight Experience Replay

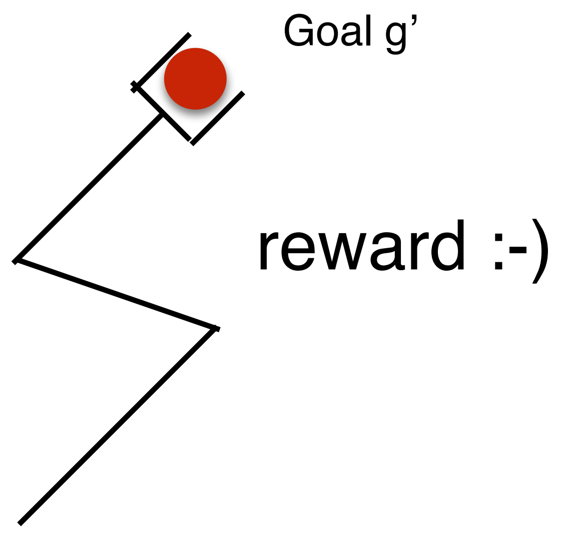
Marcin Andrychowicz[*], Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel[†], Wojciech Zaremba[†]
OpenAI

Main idea: use failed executions under one goal g, as successful executions under an alternative goal g' (which is where we ended at the end of the episode).



Our reacher at the end of the episode

$(s, g, a, 0, s')$

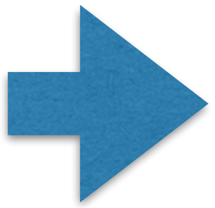


Our reacher at the end of the episode
$(s, g', a, 1, s')$

Hindsight Experience Replay

Marcin Andrychowicz[*], Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel[†], Wojciech Zaremba[†]
OpenAI

Main idea: use failed executions under one goal g, as successful executions under an alternative goal g' (which is where we ended at the end of the episode).



Hindsight Experience Replay

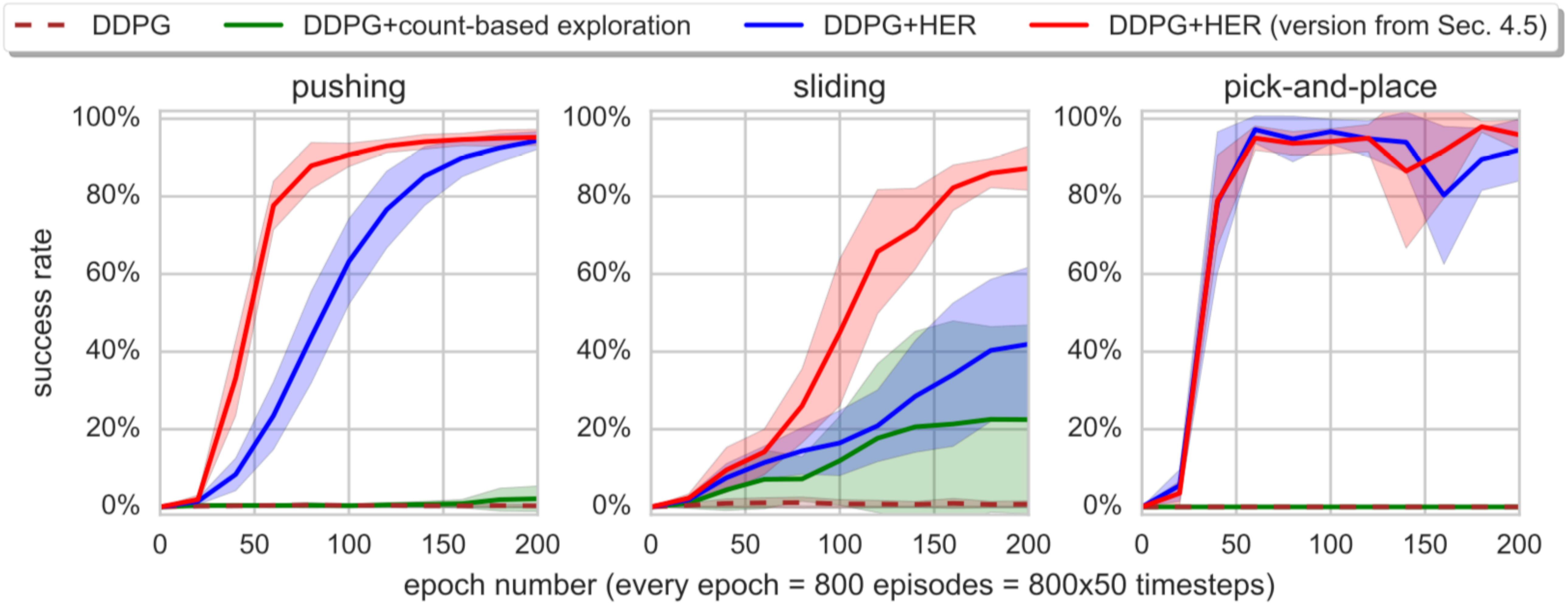
Algorithm 1 Hindsight Experience Replay (HER)

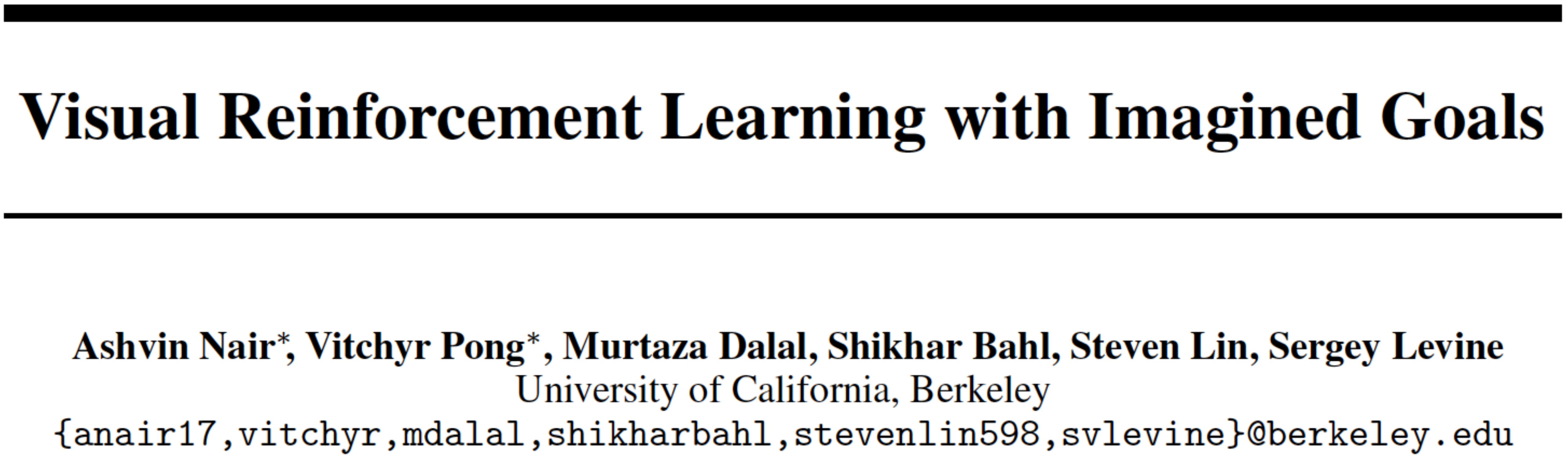
```
Given:
  • an off-policy RL algorithm $\mathbb{A}$,                    ▷ e.g. DQN, DDPG, NAF, SDQN
  • a strategy $\mathbb{S}$ for sampling goals for replay,       ▷ e.g. $\mathbb{S}(s_0, \ldots, s_T) = m(s_T)$
  • a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$.   ▷ e.g. $r(s, a, g) = -[f_g(s) = 0]$
Initialize $\mathbb{A}$                                          ▷ e.g. initialize neural networks
Initialize replay buffer $R$
for episode = $1, M$ do
    Sample a goal $g$ and an initial state $s_0$.
    for $t = 0, T-1$ do
        Sample an action $a_t$ using the behavioral policy from $\mathbb{A}$:
                $a_t \leftarrow \pi_b(s_t||g)$                   ▷ || denotes concatenation
        Execute the action $a_t$ and observe a new state $s_{t+1}$
    end for
    for $t = 0, T-1$ do
        $r_t := r(s_t, a_t, g)$
        Store the transition $(s_t||g, a_t, r_t, s_{t+1}||g)$ in $R$   ▷ standard experience replay
        Sample a set of additional goals for replay $G := \mathbb{S}($current episode$)$
        for $g' \in G$ do
            $r' := r(s_t, a_t, g')$
            Store the transition $(s_t||g', a_t, r', s_{t+1}||g')$ in $R$   ▷ HER
        end for
    end for
    for $t = 1, N$ do
        Sample a minibatch $B$ from the replay buffer $R$
        Perform one step of optimization using $\mathbb{A}$ and minibatch $B$
    end for
end for
```

G : the states of the current episode

Usually as additional goal we pick the goal that this episode achieved, and the reward becomes non zero

Hindsight Experience Replay





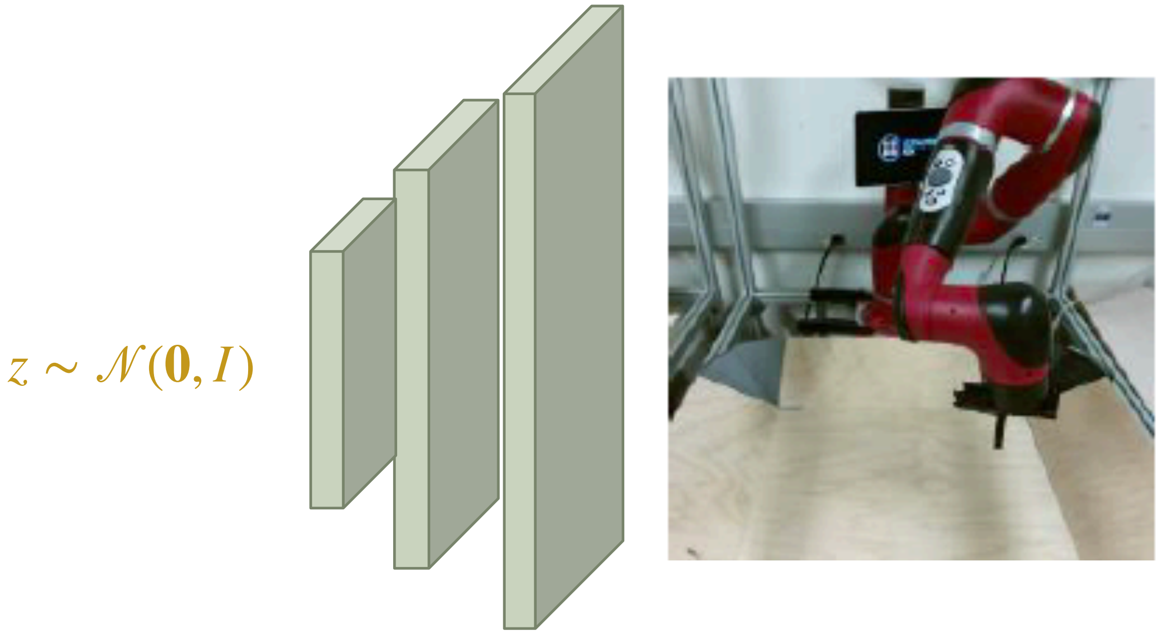
Visual Reinforcement Learning with Imagined Goals

Ashvin Nair*, Vitchyr Pong*, Murtaza Dalal, Shikhar Bahl, Steven Lin, Sergey Levine
University of California, Berkeley
{anair17,vitchyr,mdalal,shikharbahl,stevenlin598,svlevine}@berkeley.edu

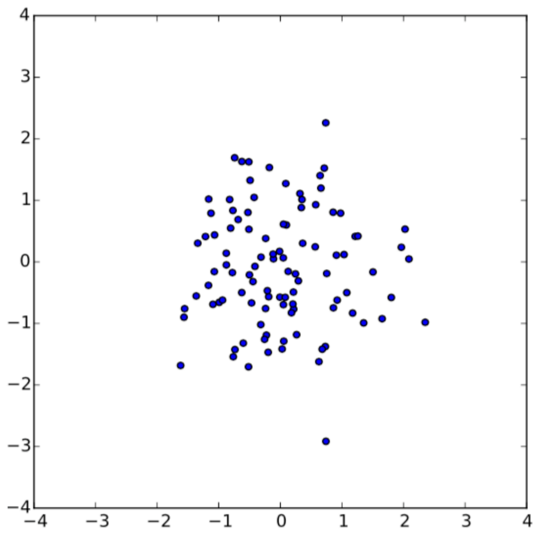
Main ideas:

- Train a generative model of images in an unsupervised manner (they used a VAE). The trained model can geenrate images by sampling latent codes from a gaussian distribution.
- Use that latent code as the state and goal representation
- Use L2 distance over latent codes as the (inverse of) reward function.
- Sample goals from that generative model for goal relabelling (augmenting experience)
- Retrain the generative model as the policy changes and the agent visits different parts of the state space

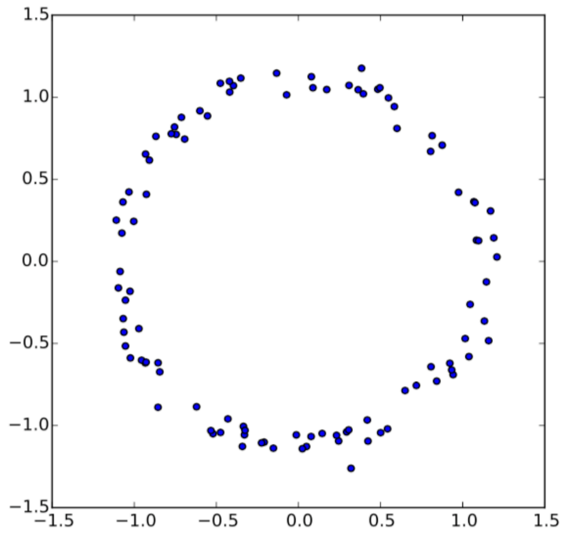
Learning Generative models of images



- Why simple gaussian noise suffices to create complex outputs?
- The neural net will transform it to a complex distribution!



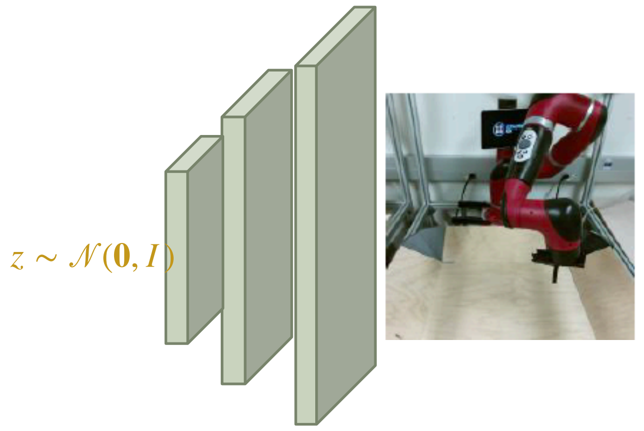
$z \sim \mathcal{N}(\mathbf{0}, I)$



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

Tutotial on variational Autoencoders, Doersch

Training Networks with Stochastic Units



Each sample z should give me a realistic image X once it passes through the neural network

We want to learn a mapping from z to the **output image X**, usually we assume a Gaussian distribution to sample every pixel 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?

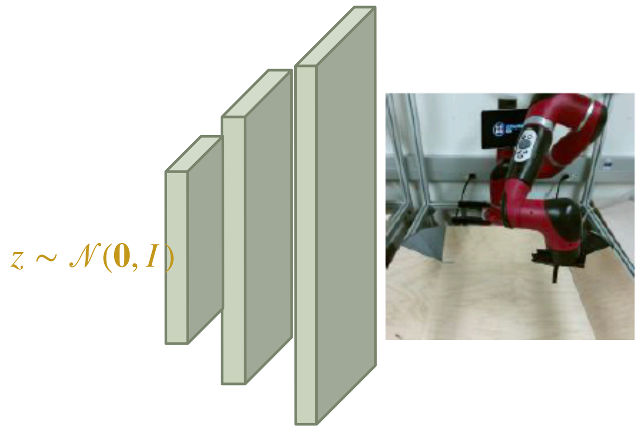
$$\min_{\theta} . \quad \sum_j -\log P(X_j) = -\sum_j \sum_{z_i \sim \mathcal{N}(\mathbf{0}, I)} \log P(X_j \mid z; \theta) = -\sum_j \sum_{z_i \sim \mathcal{N}(\mathbf{0}, I)} \|f(z_i; \theta) - X_j\|^2$$

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

This is a bad approximation, except if we have a very large number of zs. Only few zs would produce after training reasonable X. How will we find the zs that produce good X?

Motion Prediction Under Multimodality with Conditional Stochastic Networks, Google

Training Networks with Stochastic Units



Each sample z should give me a realistic image X once it passes through the neural network

We want to learn a mapping from z to the **output image X**, usually we assume a Gaussian distribution to sample every pixel 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?

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

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

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

K-best loss

Motion Prediction Under Multimodality with Conditional Stochastic Networks, Google, IMLE, Li et al. 2018

Deep Variational Inference

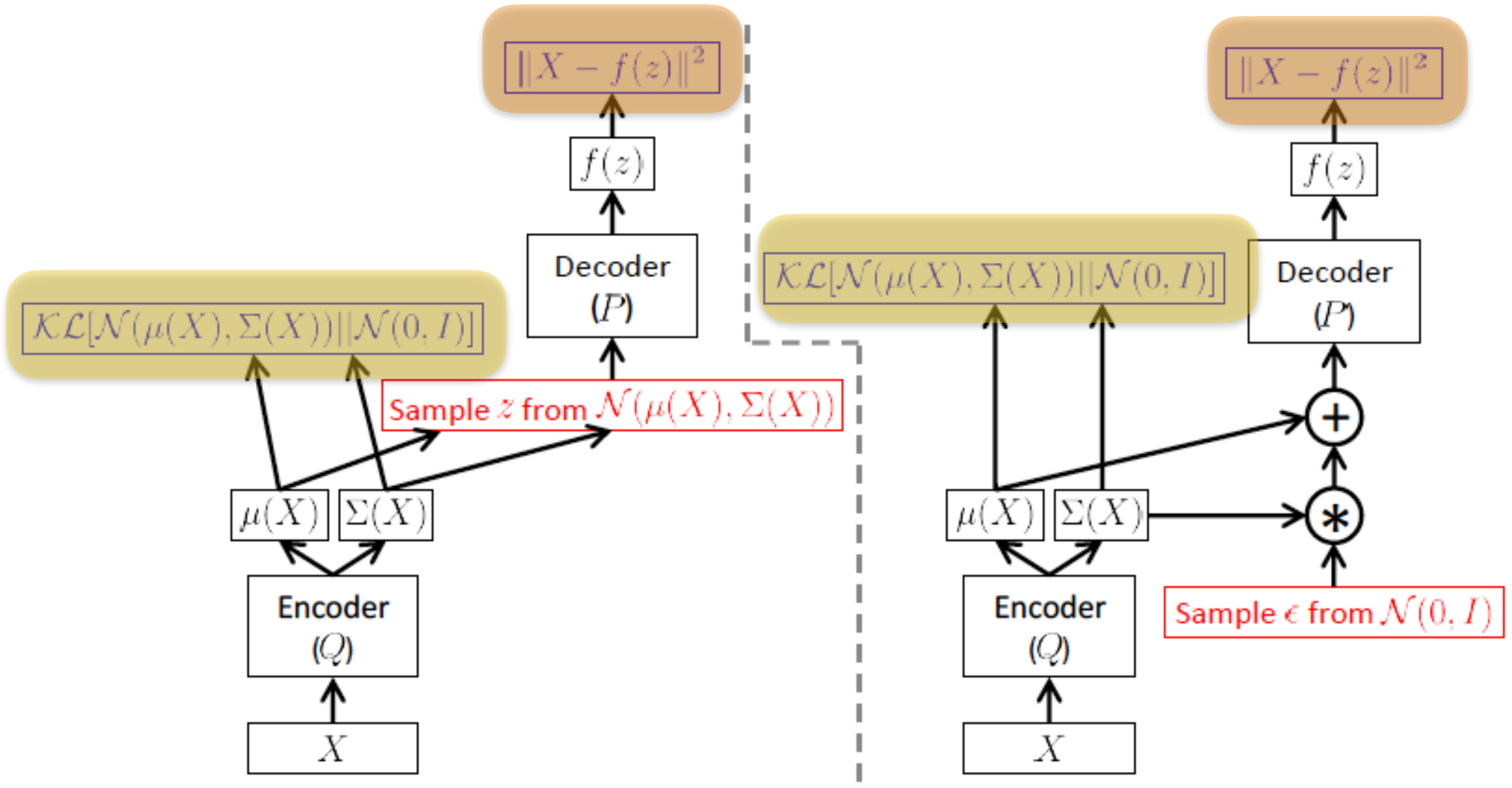
Let's consider sampling zs from an alternative distribution Q(z) and try to minimize the KL between this (variational approximation) and the true posterior, P(zlX). And because I can pick any distribution Q I like, I will also condition it on X to help inform the sampling.

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

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

Variational Autoencoder

From left to right: re-parametrization trick!



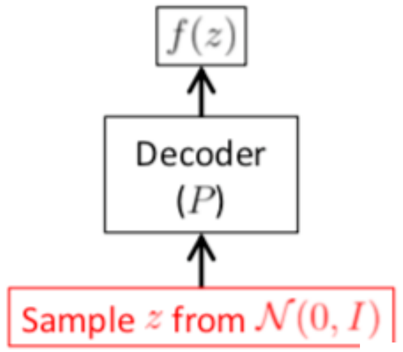
$\|X - f(z)\|^2$
$f(z)$
Decoder (P)
$\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X)) \| \mathcal{N}(0, I)]$
Sample z from $\mathcal{N}(\mu(X), \Sigma(X))$
$\mu(X)$
$\Sigma(X)$
Encoder (Q)
X
$\|X - f(z)\|^2$
$f(z)$
$\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X)) \| \mathcal{N}(0, I)]$
Decoder (P)
$+$
$*$
$\mu(X)$
$\Sigma(X)$
Encoder (Q)
Sample ϵ from $\mathcal{N}(0, I)$
X

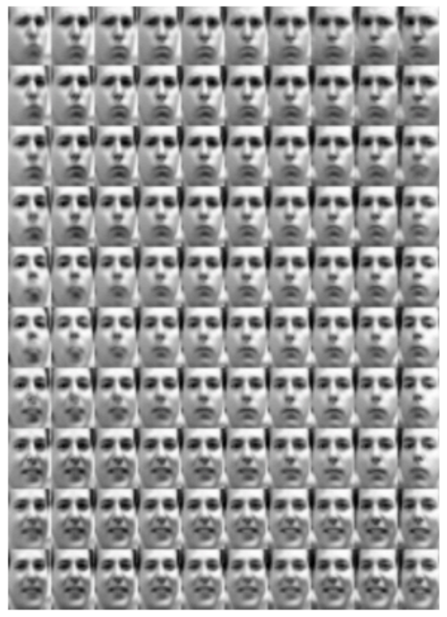
$$\min_{\phi,\theta} . \quad D_{KL}(\underbrace{Q(z \mid X; \phi)}_{\text{encoder}} \,||\, P(z)) - \mathbb{E}_Q \log \underbrace{P(X \mid z; \theta)}_{\text{decoder}}$$

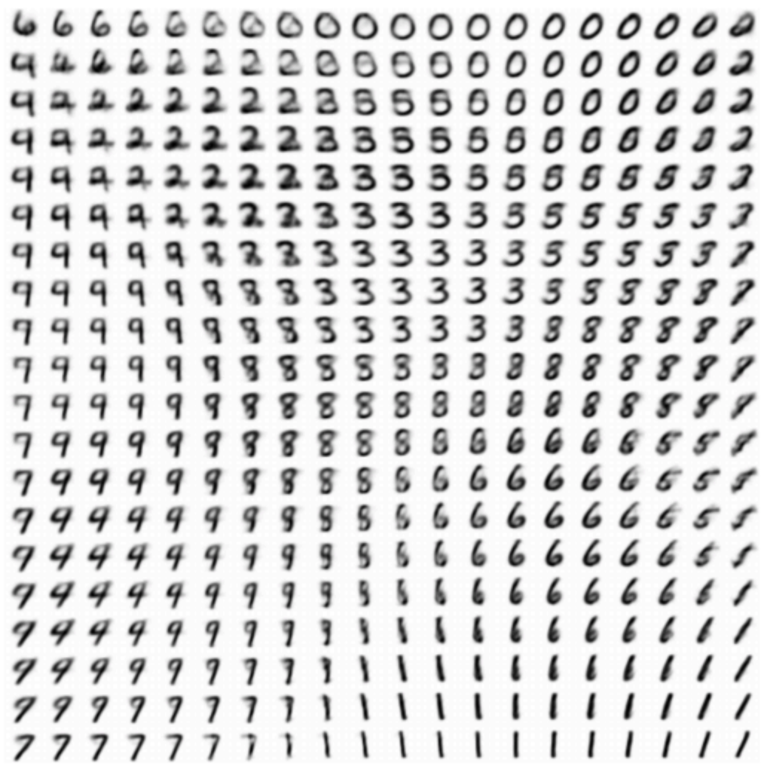
T*utotial on variational Autoencoders,* Doersch

Variational Autoencoder

At test time

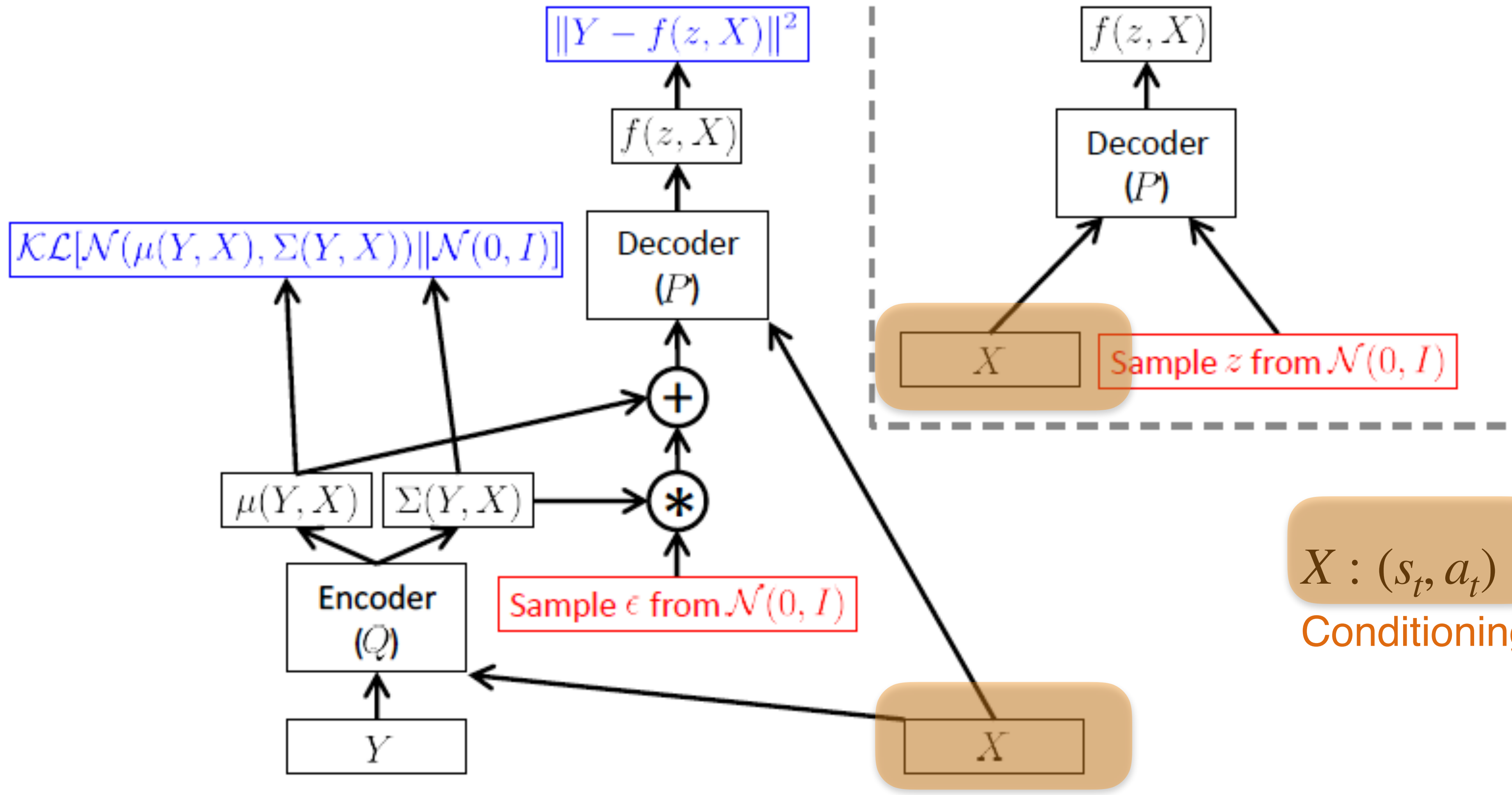



(a) Learned Frey Face manifold


(b) Learned MNIST manifold

Auto-Encoding Variational Bayes, Kingma and Welling

Conditional VAE



$\|Y - f(z, X)\|^2$
$f(z, X)$
Decoder (P)
$\mathcal{KL}[\mathcal{N}(\mu(Y,X), \Sigma(Y,X)) \| \mathcal{N}(0, I)]$
+
$\mu(Y, X)$
$\Sigma(Y, X)$
*
Encoder (Q)
Sample ϵ from $\mathcal{N}(0, I)$
Y
X
$f(z, X)$
Decoder (P)
X
Sample z from $\mathcal{N}(0, I)$

$X : (s_t, a_t)$ $\quad Y : s_{t+1}$

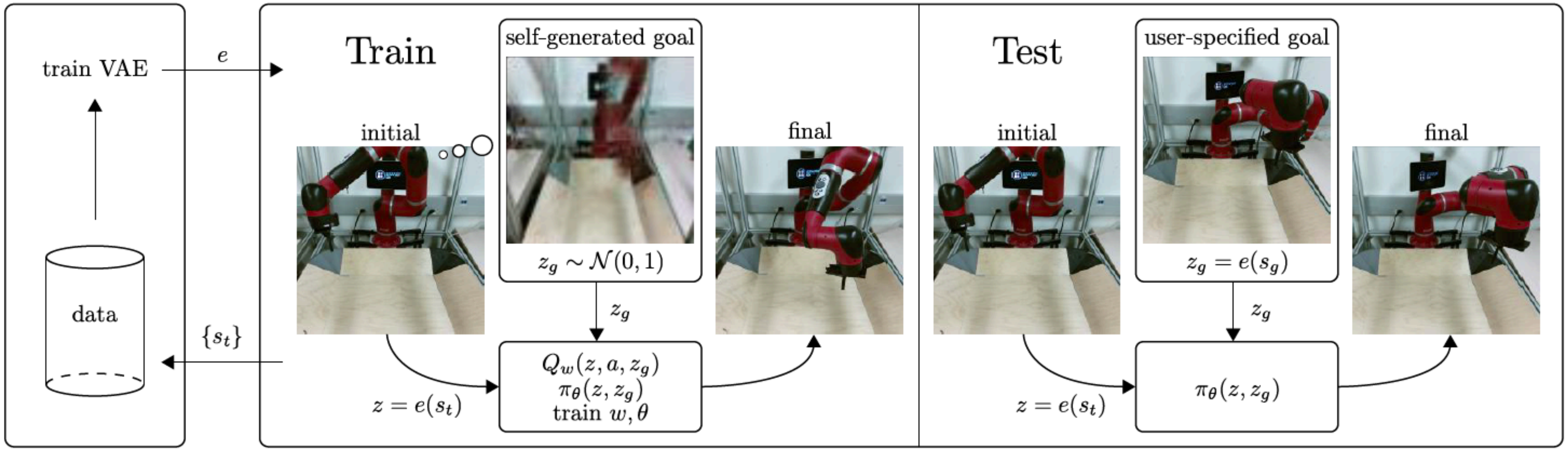
Conditioning

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

Tutotial on variational Autoencoders, Doersch

Visual Reinforcement Learning with Imagined Goals

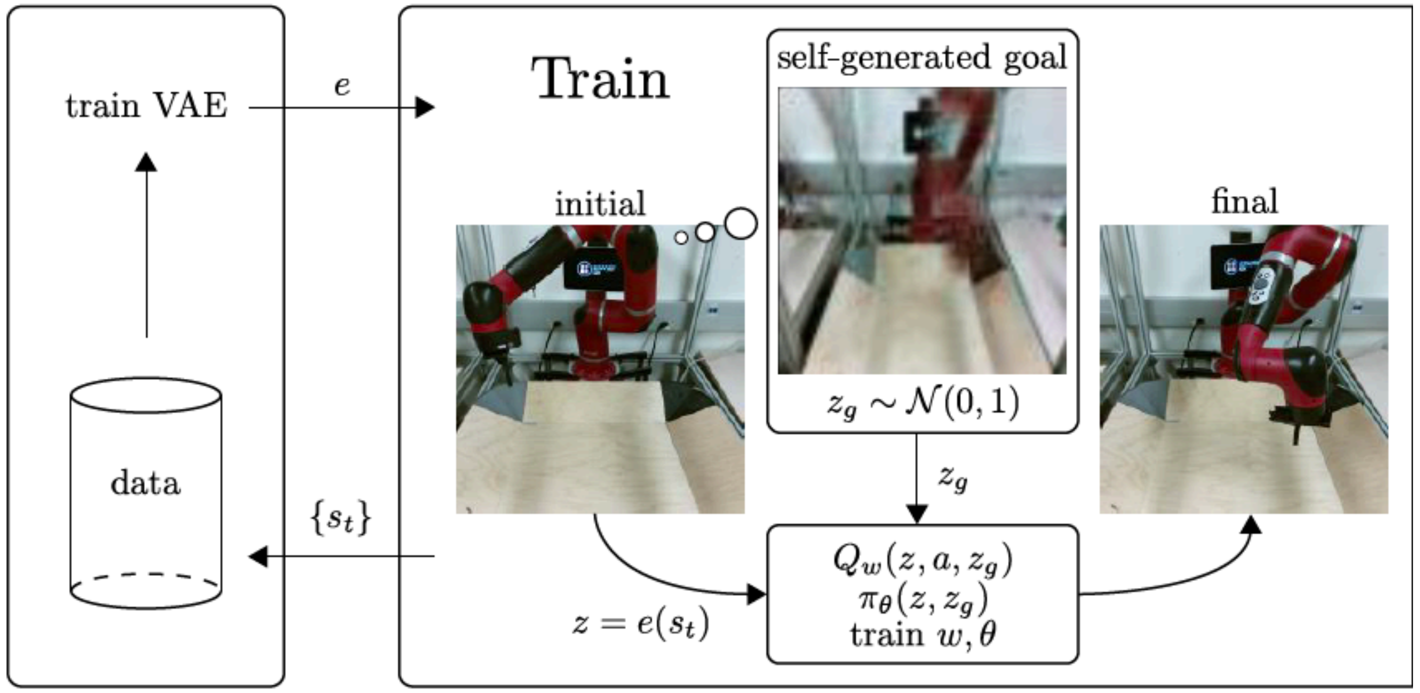
Ashvin Nair[*], Vitchyr Pong[*], Murtaza Dalal, Shikhar Bahl, Steven Lin, Sergey Levine
University of California, Berkeley
`{anair17,vitchyr,mdalal,shikharbahl,stevenlin598,svlevine}@berkeley.edu`



Visual Reinforcement Learning with Imagined Goals

Ashvin Nair*, Vitchyr Pong*, Murtaza Dalal, Shikhar Bahl, Steven Lin, Sergey Levine
University of California, Berkeley
`{anair17,vitchyr,mdalal,shikharbahl,stevenlin598,svlevine}@berkeley.edu`

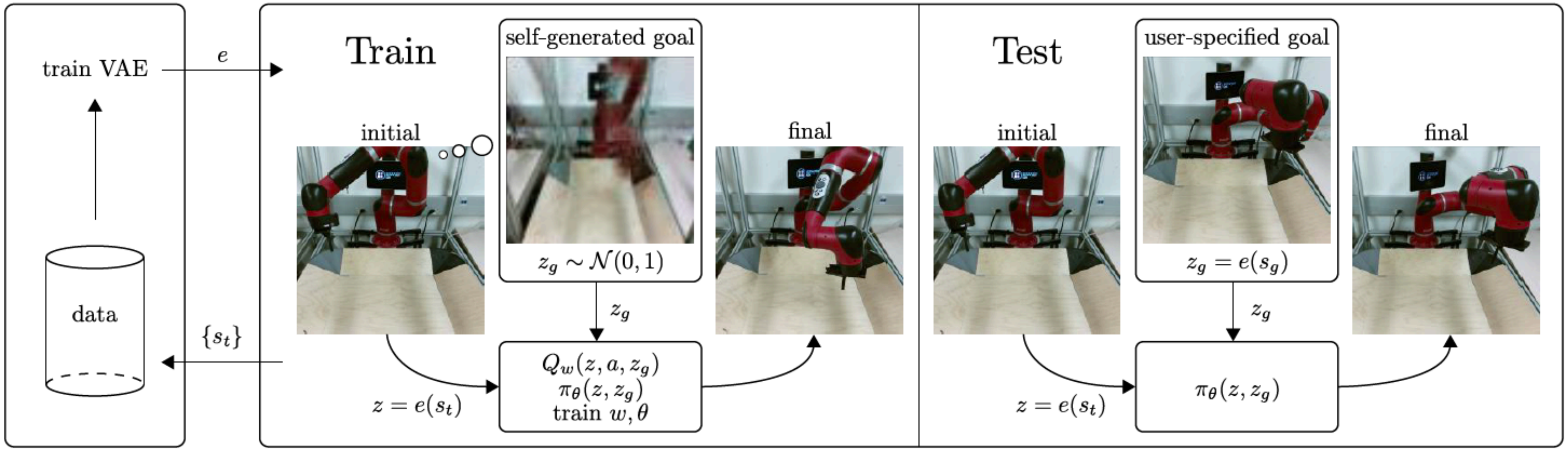
At training time the agent imagines goals to reach by simply sampling codes (vectors) from the latent space.



Visual Reinforcement Learning with Imagined Goals

Ashvin Nair*, Vitchyr Pong*, Murtaza Dalal, Shikhar Bahl, Steven Lin, Sergey Levine
University of California, Berkeley
{anair17,vitchyr,mdalal,shikharbahl,stevenlin598,svlevine}@berkeley.edu

At test time, the human supplies a goal image which is encoded into a latent code by the trained encoder.



Algorithm 1 RIG: Reinforcement learning with imagined goals

```
Require: VAE encoder $q_\phi$, VAE decoder $p_\psi$, policy
    $\pi_\theta$, goal-conditioned value function $Q_w$.
 1: Collect $\mathcal{D} = \{s^{(i)}\}$ using exploration policy.
 2: Train $\beta$-VAE on $\mathcal{D}$ by optimizing (2).
 3: for $n = 0, ..., N-1$ episodes do
 4:     Sample latent goal from prior $z_g \sim p(z)$.
 5:     Sample initial state $s_0 \sim E$.
 6:     for $t = 0, ..., H-1$ steps do
 7:         Get action $a_t = \pi_\theta(e(s_t), z_g) +$ noise.
 8:         Get next state $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.
 9:         Store $(s_t, a_t, s_{t+1}, z_g)$ into replay buffer $\mathcal{R}$.
10:         Sample transition $(s, a, s', z_g) \sim \mathcal{R}$.
11:         Encode $z' = e(s')$.
12:         (Probability $0.5$) replace $z_g$ with $z'_g \sim p(z)$.
13:         Compute new reward $r = -||z' - z_g||$.
14:         Minimize (1) using $(z, a, z', z_g, r)$.
15:     end for
16:     Fine-tune $\beta$-VAE every $K$ episodes on mixture
    of $\mathcal{D}$ and $\mathcal{R}$.
17: end for
```

$$\mathcal{E}(w) = \frac{1}{2}||Q_w(s,a,g) - (r + \gamma \max_{a'} Q_{\bar{w}}(s', a', g))||^2$$

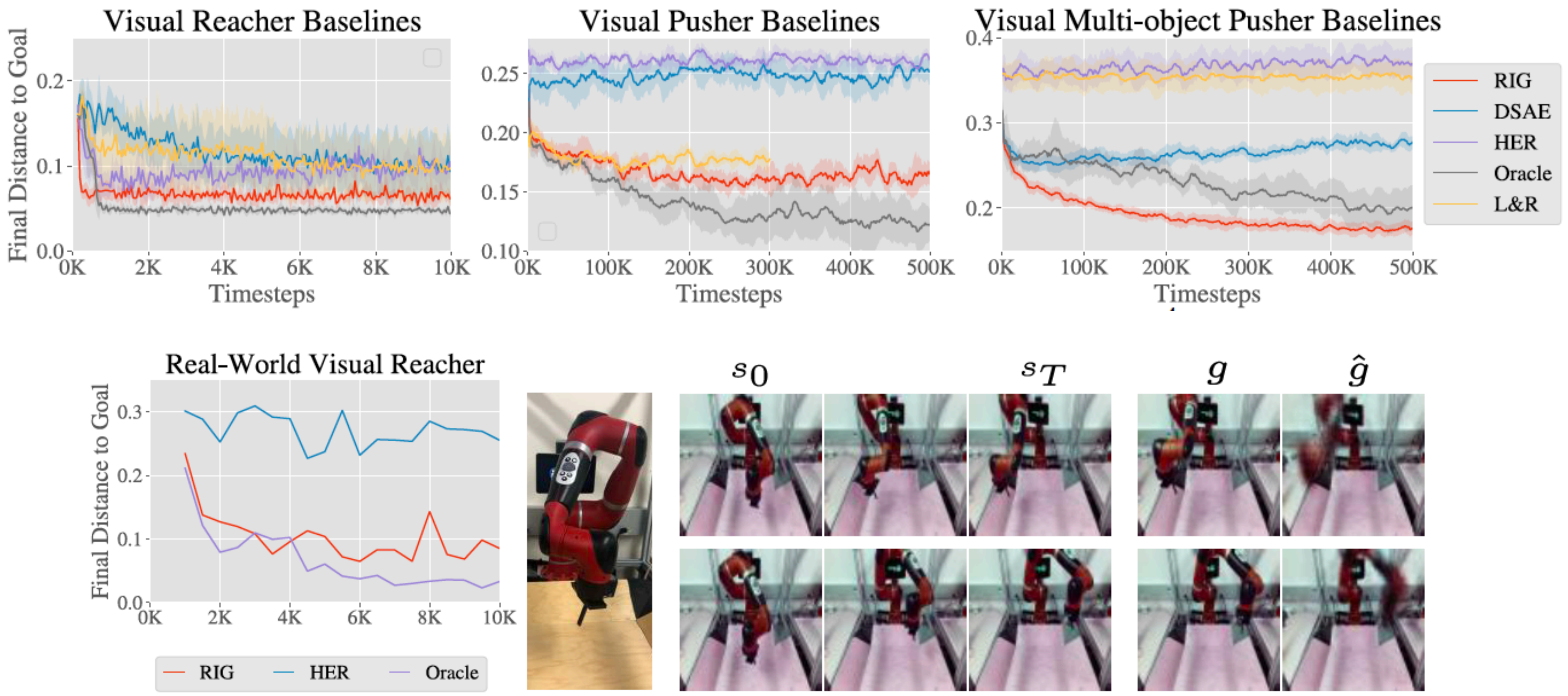


Figure 7: (Left) Our method compared to the HER baseline and oracle on a real-world visual reaching task. (Middle) Our robot setup is pictured. (Right) Test rollouts of our learned policy.

HER here is using L2 over images, that’s a terrible (inverse of) reward function