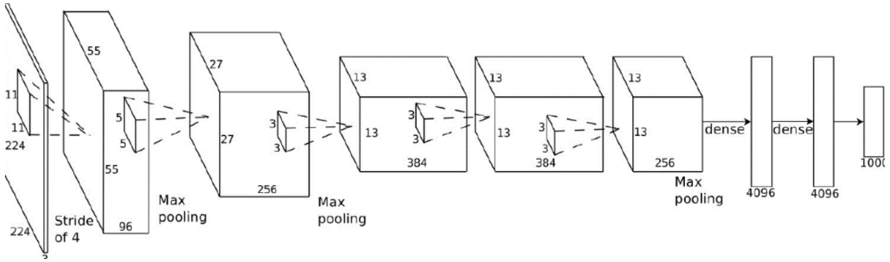
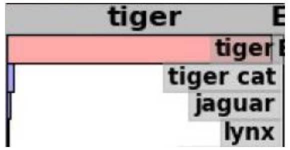
Deep Reinforcement Learning in the Real World

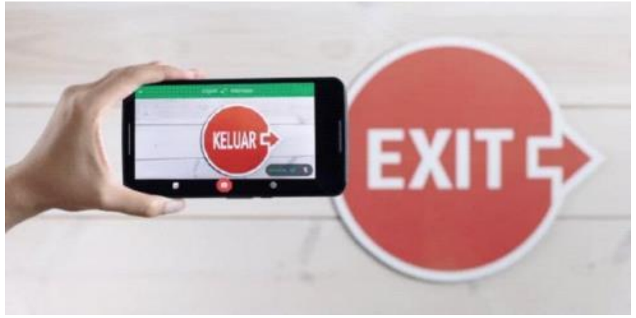
Sergey Levine

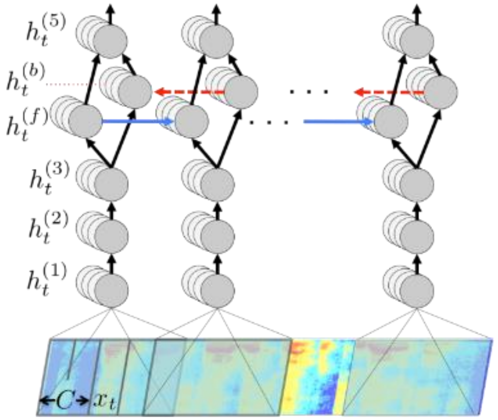
UC Berkeley **Google Brain**

Deep learning helps us handle *unstructured environments*



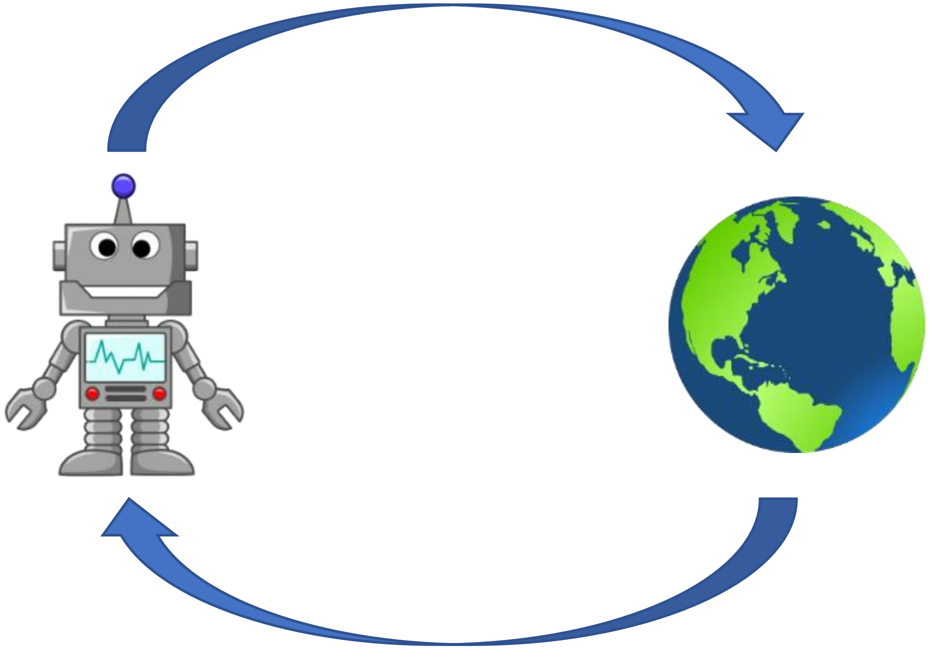






Reinforcement learning provides a formalism for *behavior*

decisions (actions)



consequences
observations
rewards

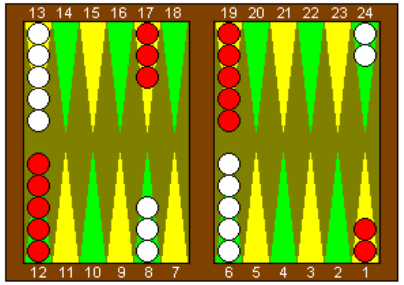
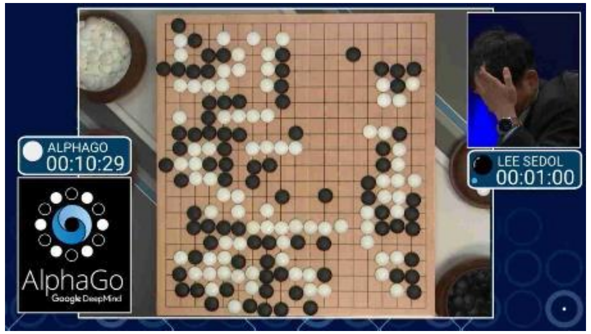
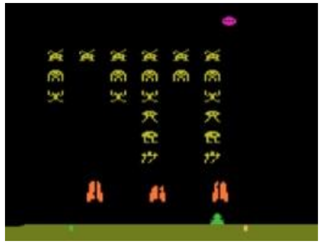


Figure 2. An illustration of the normal opening position in backgammon. TD-Gammon has sparked a near-universal conversion in the way experts play certain opening rolls. For example, with an opening roll of 4-1, most players have now switched from the traditional move of 13-9, 6-5, to TD-Gammon's preference, 13-9, 24-23. TD-Gammon's analysis is given in Table 2.



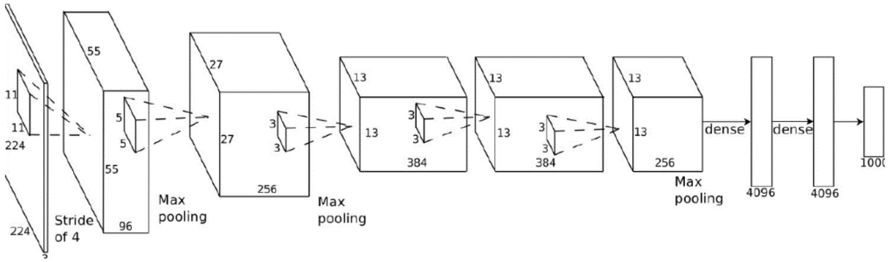


Mnih et al. '13

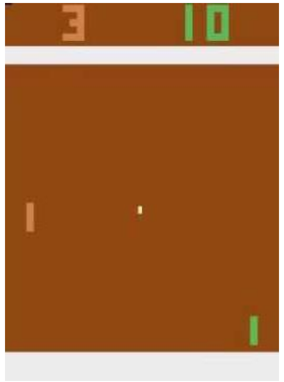
notice something?

But maybe it's not so simple…

deep learning handles unstructured environments using data like this:



how can we possibly hope RL to enable intelligent machines when it trains on data that looks like this:



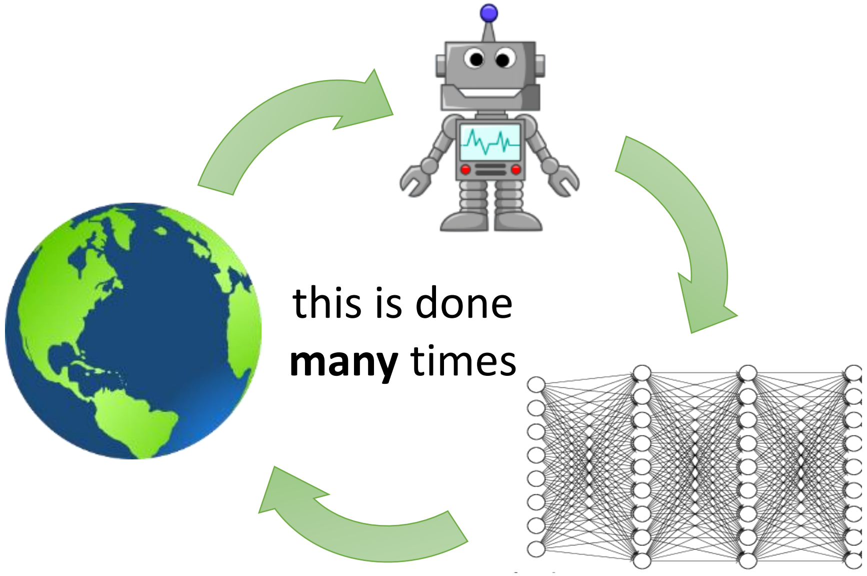
Maybe we learned the wrong lesson from deep learning

It's not about deep nets…

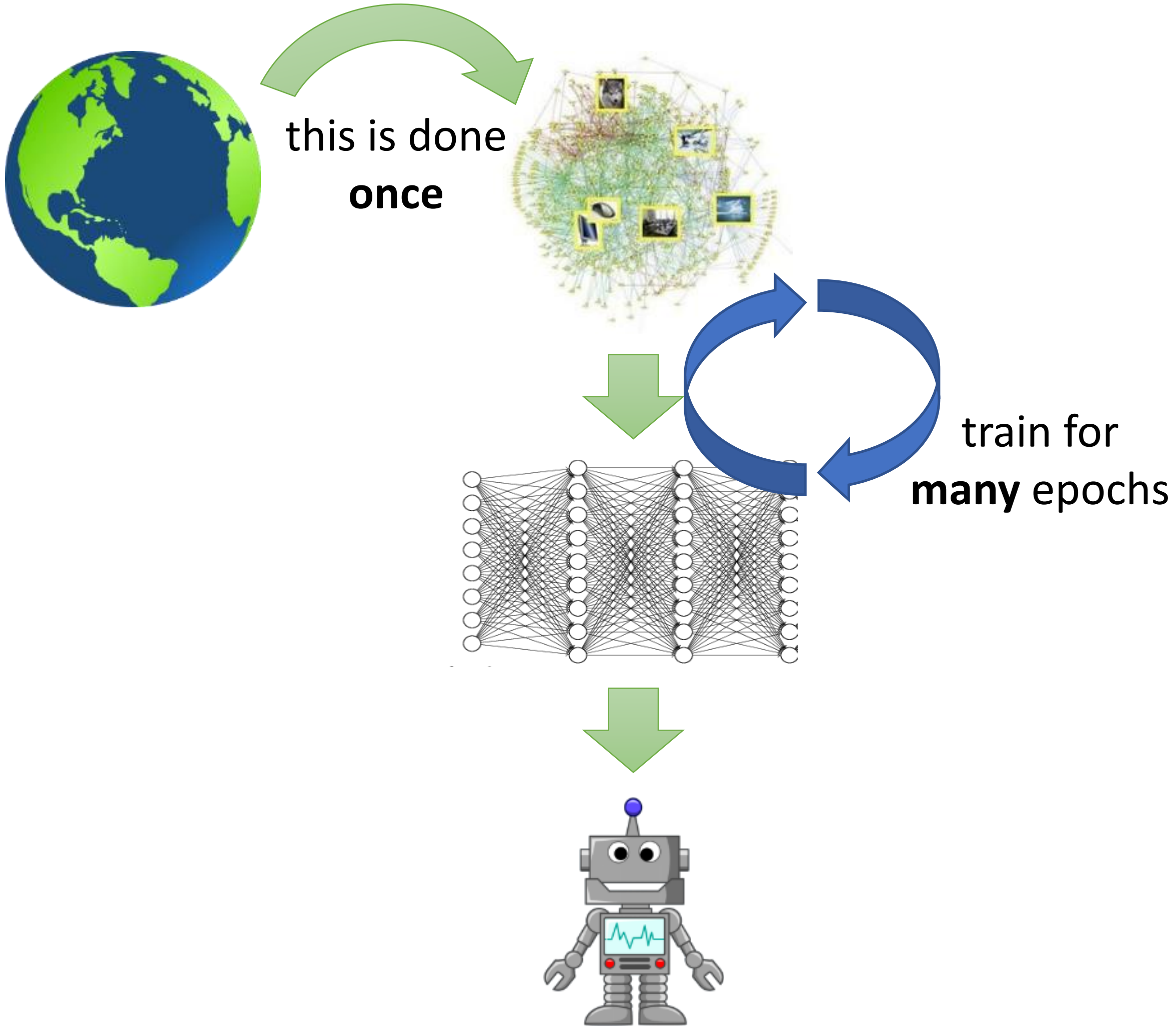
It's about big models + highly varied and diverse data!

RL has a **big** problem

reinforcement learning

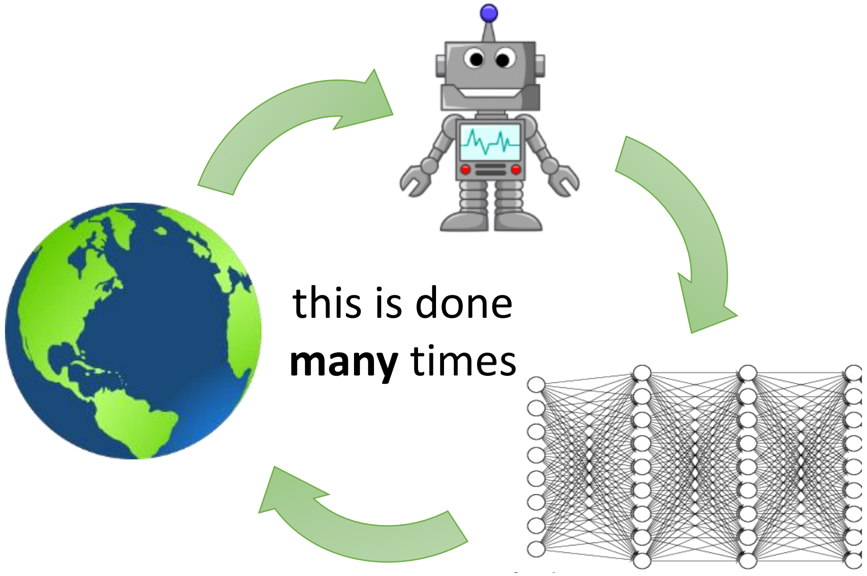


supervised machine learning

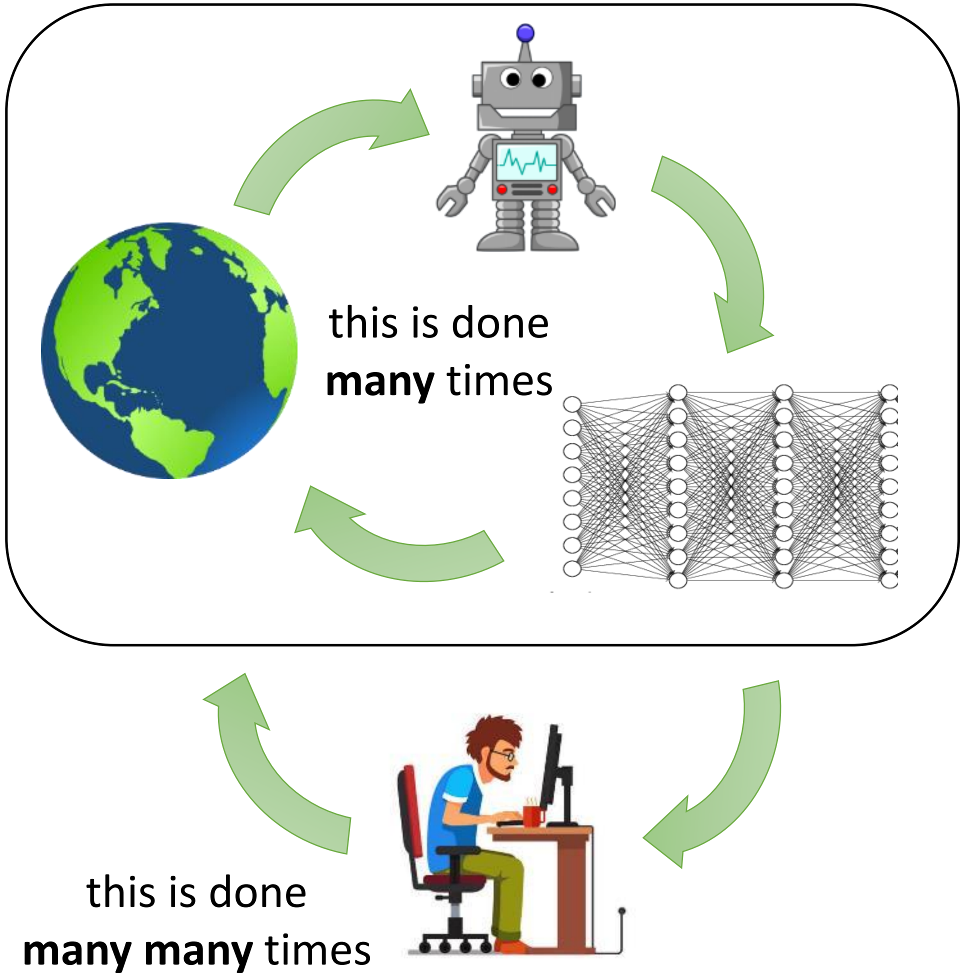


RL has a **big** problem

reinforcement learning

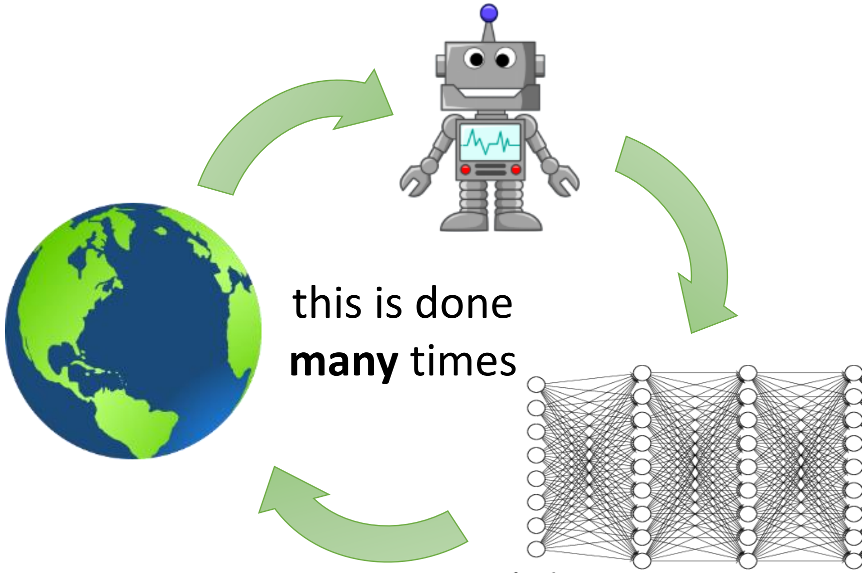


actual reinforcement learning

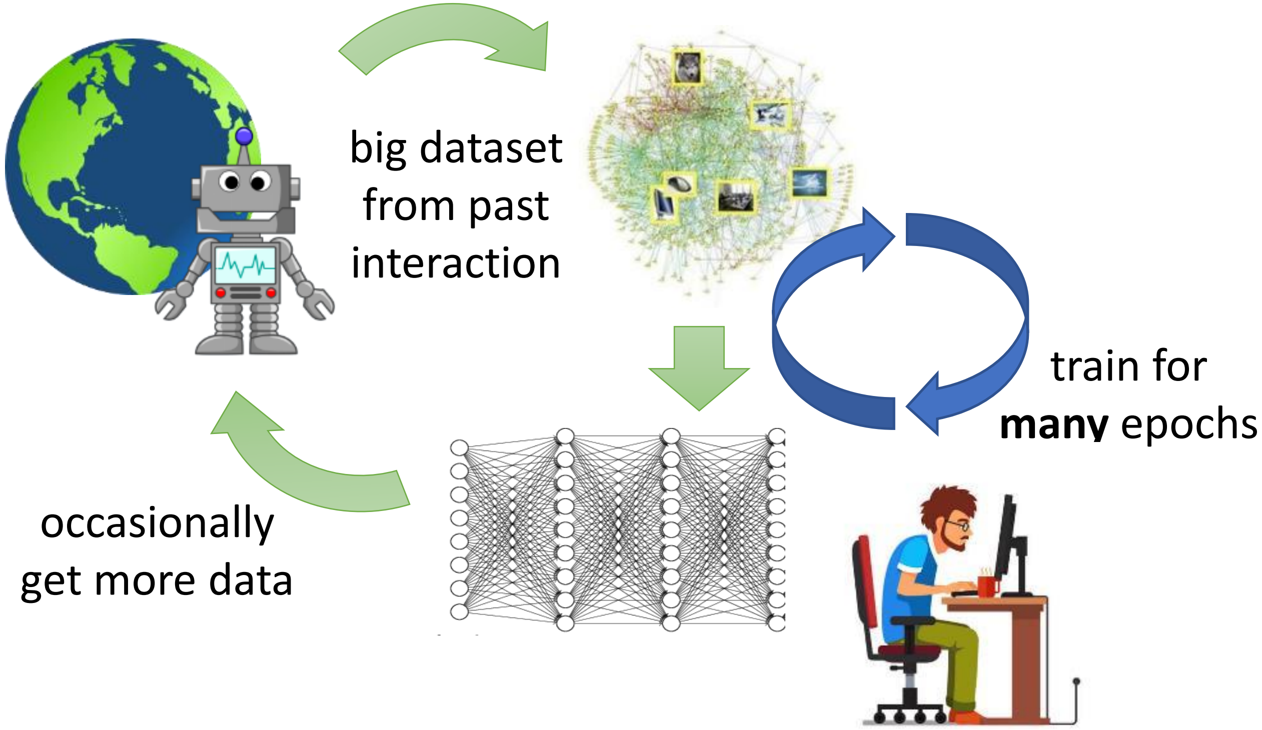


Off-policy RL with large datasets

reinforcement learning

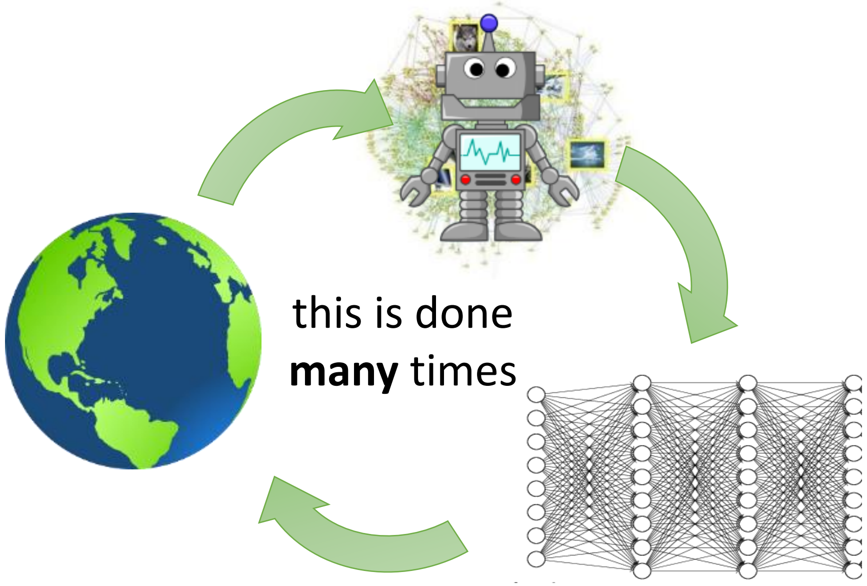


off-policy reinforcement learning

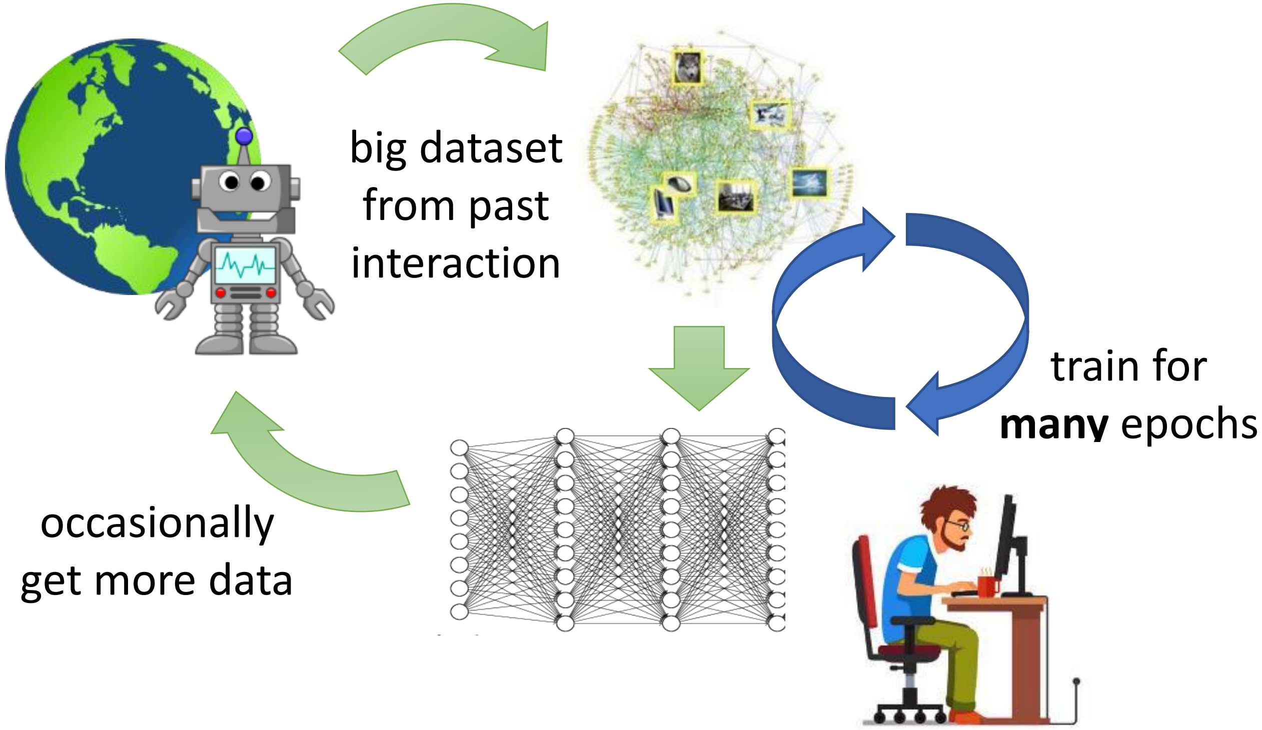


Off-policy RL with large datasets

reinforcement learning



off-policy reinforcement learning

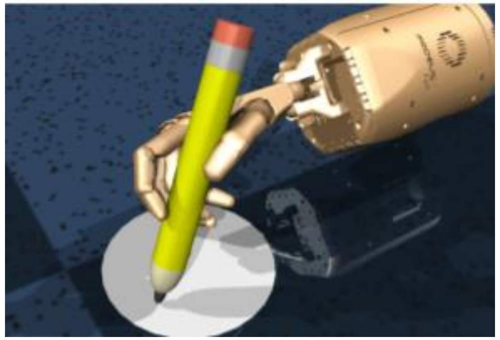


How does off-policy RL work?

Off-policy RL = prediction

- What do we predict?
  - Future observations: understand how the world works
  - Future rewards: understand the consequences of your actions
- They're not as different as you think – more on this later

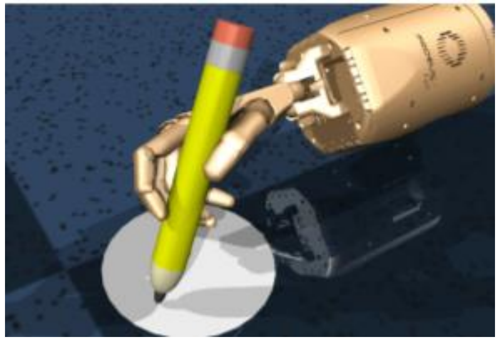
Off-policy model-free RL algorithms



Off-policy model-based RL algorithms

Why these are actually the same thing

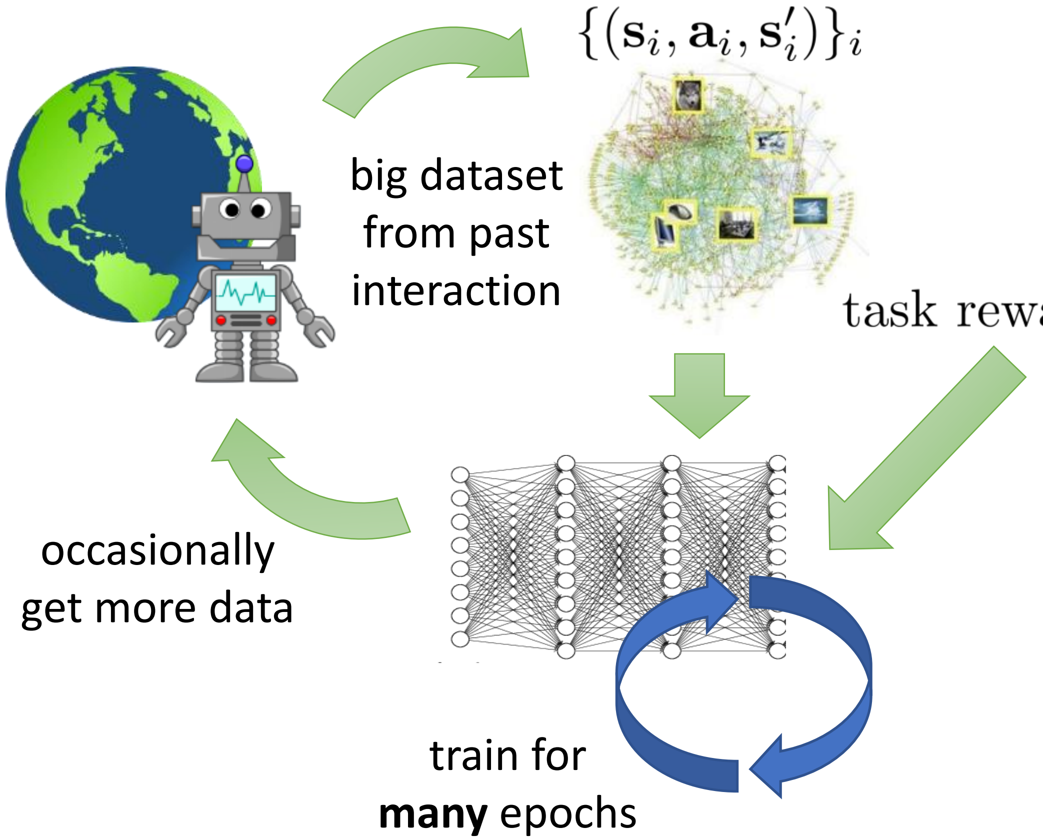
Off-policy model-free RL algorithms



Off-policy model-based RL algorithms

Why these are actually the same thing

Off-policy model-free learning



task reward $r(\mathbf{s}, \mathbf{a})$

RL objective: $\max_\pi \sum_{t=1}^{T} E_{\mathbf{s}_t, \mathbf{a}_t \sim \pi}[r(\mathbf{s}_t, \mathbf{a}_t)]$

Q-function: $Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\mathbf{s}_{t'}, \mathbf{a}_{t'} \sim \pi}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$

$\pi(\mathbf{a}|\mathbf{s}) = 1 \text{ if } \mathbf{a} = \arg\max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$

$$Q^\star(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \max_{\mathbf{a}'} Q^\star(\mathbf{s}', \mathbf{a}')$$

enforce this equation on all off-policy data

How to solve for the Q-function?

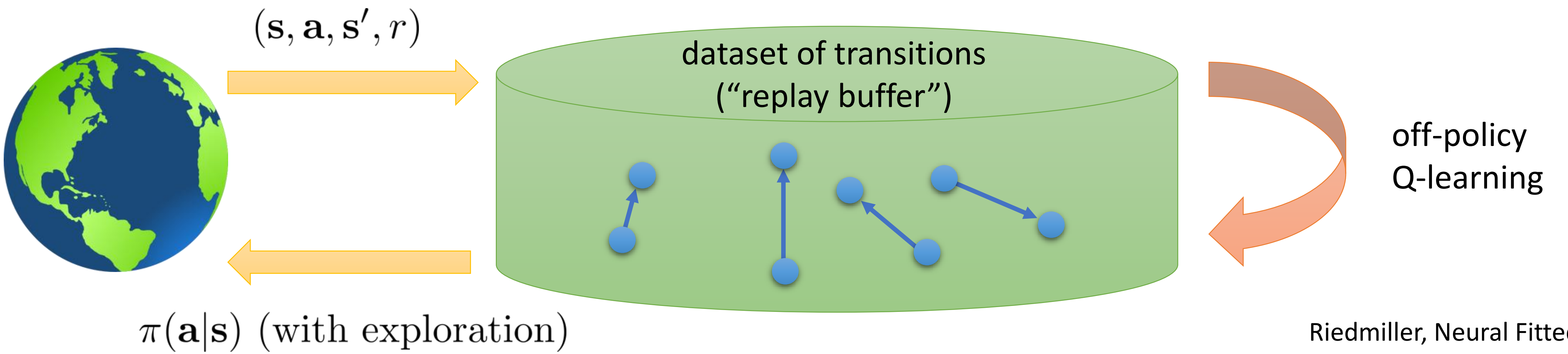
$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}')$$ don't need on-policy data for this!

off-policy Q-learning:

1. collect dataset $\{(\mathbf{s}_i,\mathbf{a}_i,\mathbf{s}'_i,r_i)\}$ using some policy, add it to $\mathcal{B}$

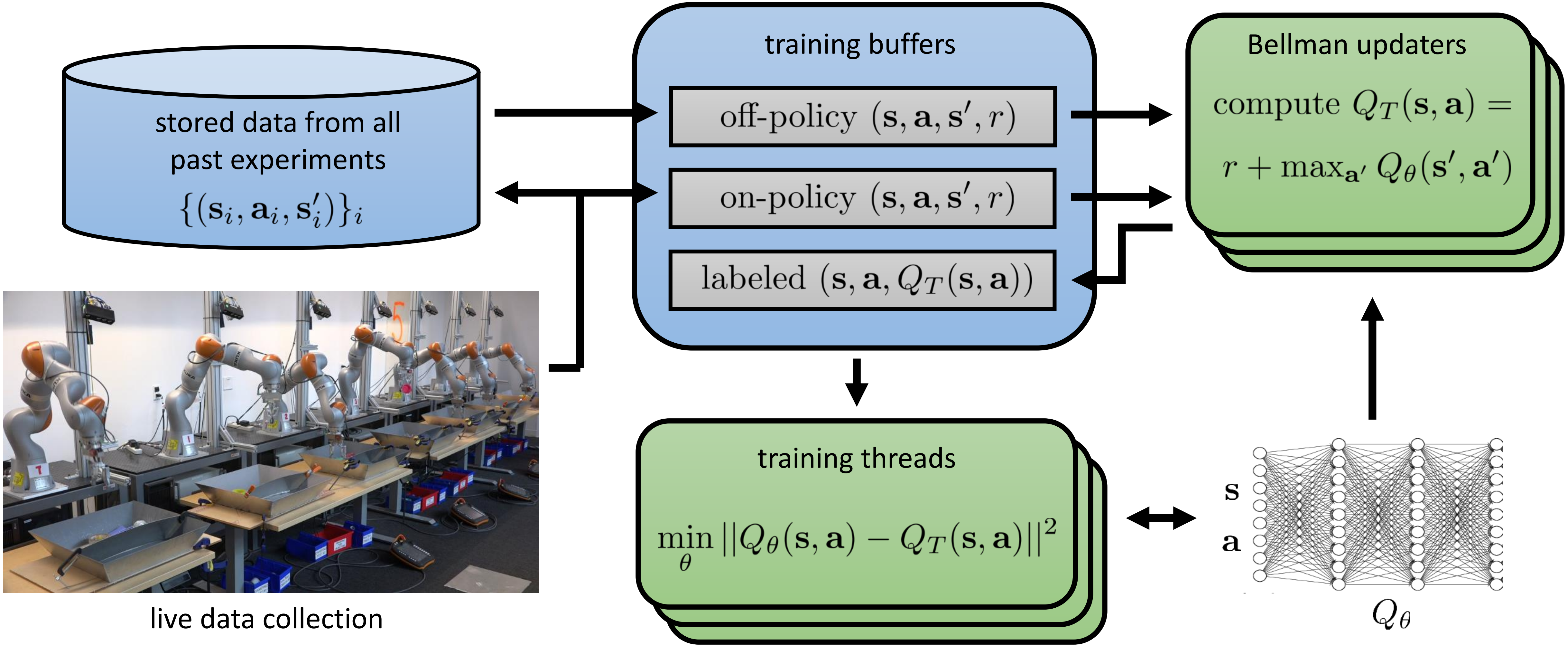
$K\times$

2. sample a batch $(\mathbf{s}_i,\mathbf{a}_i,\mathbf{s}'_i,r_i)$ from $\mathcal{B}$
3. minimize $\sum_i (Q(\mathbf{s}_i,\mathbf{a}_i) - [r(\mathbf{s}_i,\mathbf{a}_i) + \max_{\mathbf{a}'_i} Q(\mathbf{s}'_i,\mathbf{a}'_i)])^2$



See, e.g.
Riedmiller, Neural Fitted Q-Iteration '05
Ernst et al., Tree-Based Batch Mode RL '05

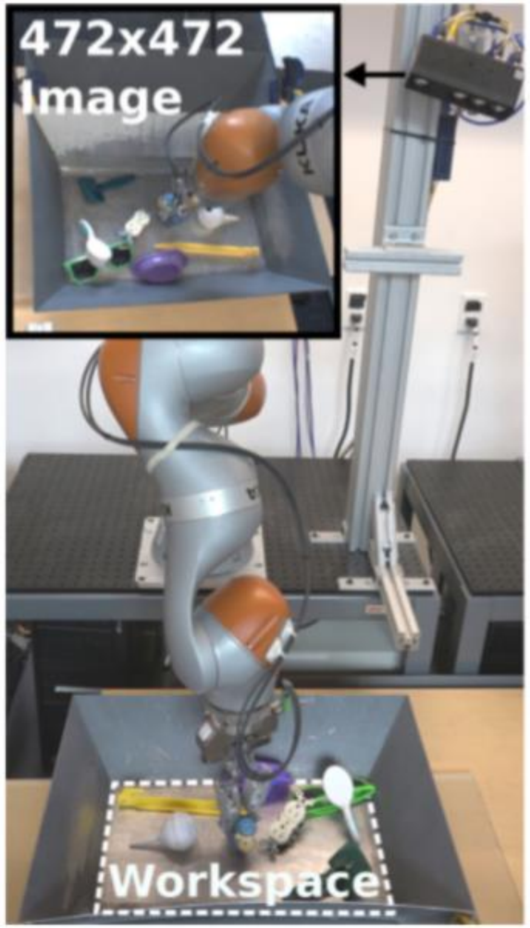
QT-Opt: off-policy Q-learning at scale

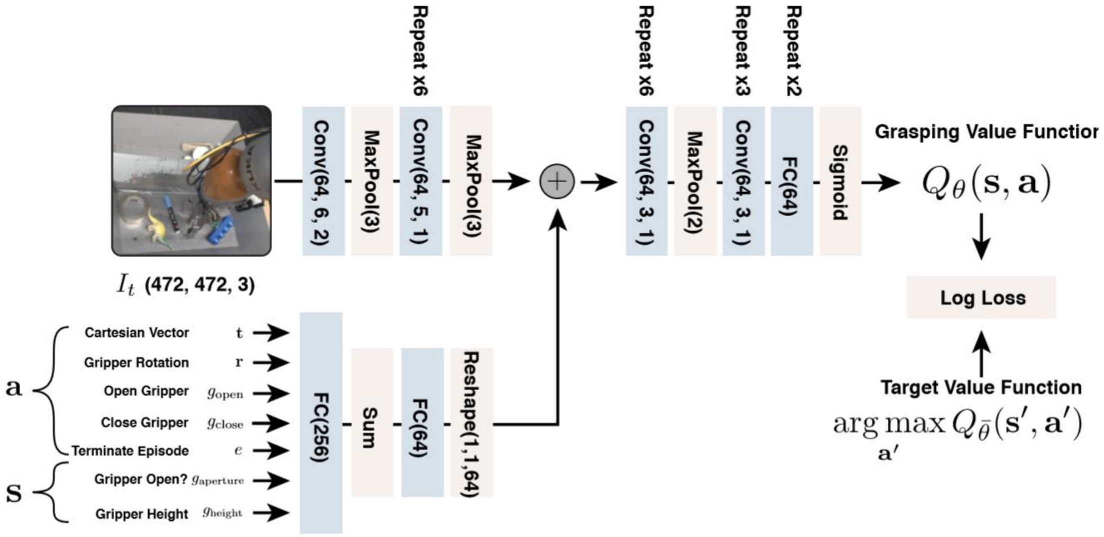


Kalashnikov, Irpan, Pastor, Ibarz, Herzong, Jang, Quillen, Holly, Kalakrishnan, Vanhoucke, Levine. **QT-Opt: Scalable Deep Reinforcement Learning of Vision-Based Robotic Manipulation Skills**

$$\text{minimize} \sum_i (Q(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \max_{\mathbf{a}'_i} Q(\mathbf{s}'_i, \mathbf{a}'_i)])^2$$

Grasping with QT-Opt





- About 1000 training objects
- About 600k training grasp attempts
- Q-function network with 1.2M parameters
- The only grasp-specific feature is the reward (1 if grasped)

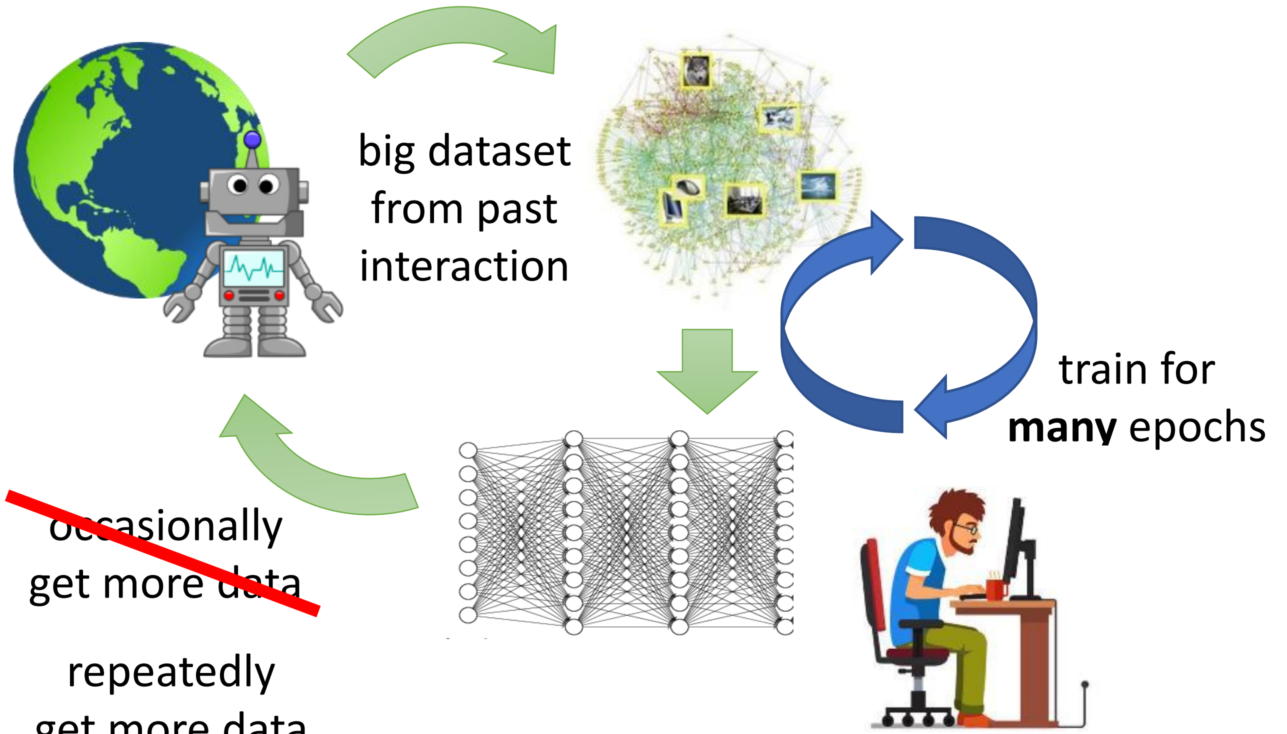
Kalashnikov, Irpan, Pastor, Ibarz, Herzong, Jang, Quillen, Holly, Kalakrishnan, Vanhoucke, Levine. **QT-Opt: Scalable Deep Reinforcement Learning of Vision-Based Robotic Manipulation Skills**

Emergent grasping strategies

96%

So far...

off-policy reinforcement learning



live data collection

So what's the problem?

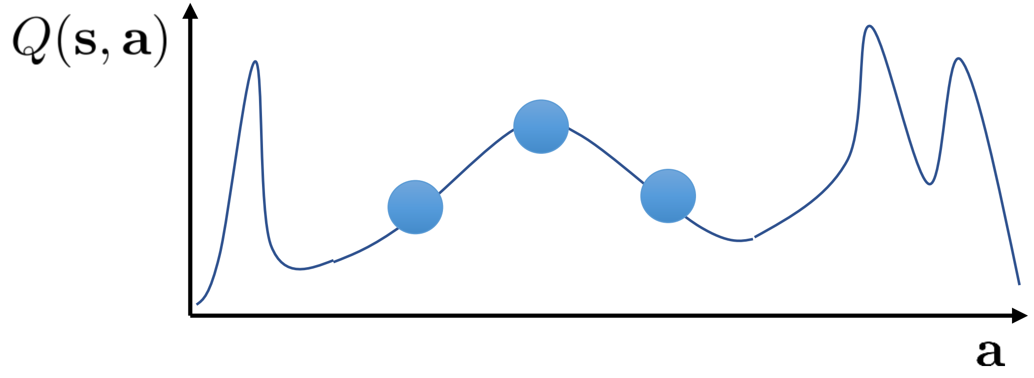
$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}')$$

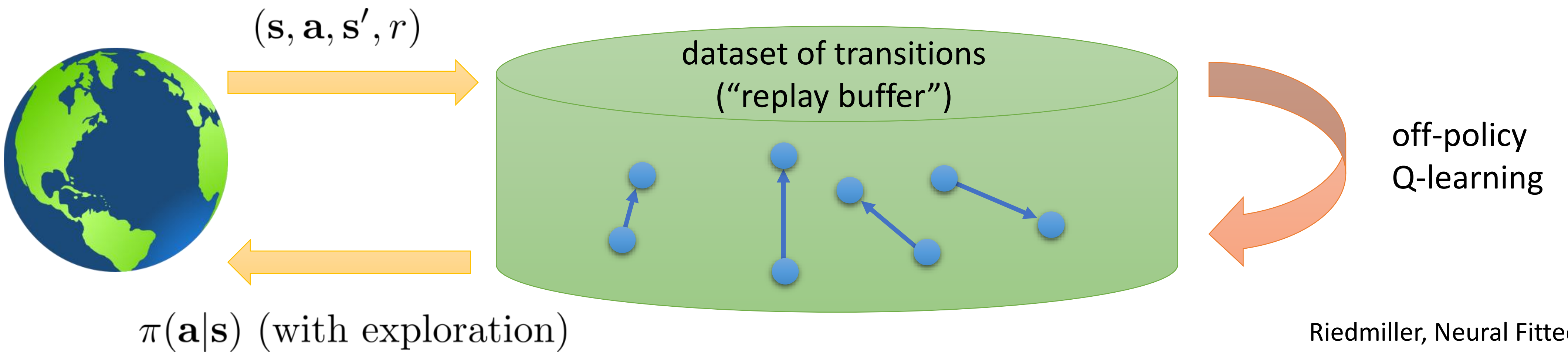
what action will this pick?

if $\mathbf{a}^\star = \arg\max_{\mathbf{a}} Q(\mathbf{s},\mathbf{a})$ makes $\beta(\mathbf{s},\mathbf{a}^\star)$ small

we end up training on garbage!

$Q(\mathbf{s},\mathbf{a})$ is trained on $(\mathbf{s},\mathbf{a}) \sim \beta(\mathbf{s},\mathbf{a})$





See, e.g.
Riedmiller, Neural Fitted Q-Iteration '05
Ernst et al., Tree-Based Batch Mode RL '05

How to stop training on garbage?

$$\cancel{Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}')}$$

$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + E_{\mathbf{a}'\sim\pi_{\text{new}}}[Q(\mathbf{s}',\mathbf{a}')]$$

how to pick $\pi_{\text{new}}(\mathbf{a}|\mathbf{s})$?

option 1: stay *close* to β

e.g. $D_{\text{KL}}(\pi_{\text{new}}(\cdot|\mathbf{s})\|\beta(\cdot|\mathbf{s})) \leq \epsilon$

issue 1: we don't know β

issue 2: this is *way* too conservative

key idea: constrain to *support* of β

$$\max_{\pi\in\Delta_{|S|}} \mathbb{E}_{a\sim\pi(\cdot|s)}[\hat{Q}_k(s,a)] - \lambda\sqrt{\text{var}_k\,\mathbb{E}_{a\sim\pi(\cdot|s)}[\hat{Q}_k(s,a)]}$$

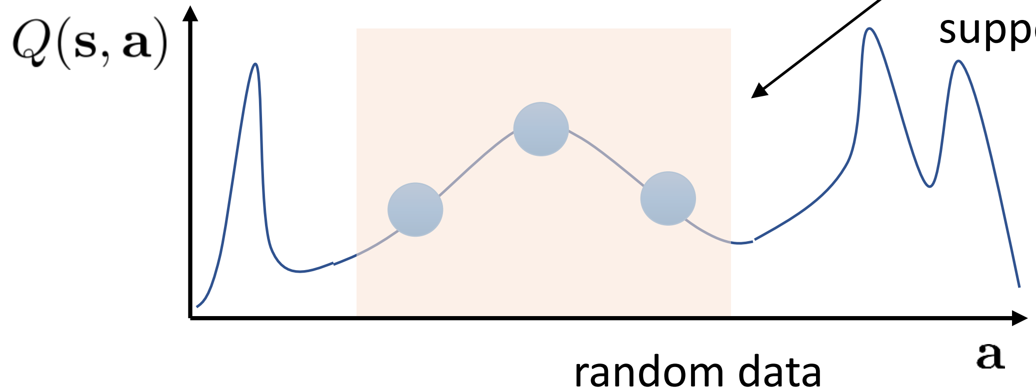
pessimistic w.r.t. epistemic uncertainty

$$\text{s.t. } \mathbb{E}_{s\sim\mathcal{D}}[\text{MMD}(\mathcal{D}(s),\pi(\cdot|s))] \leq \varepsilon$$

support constraint

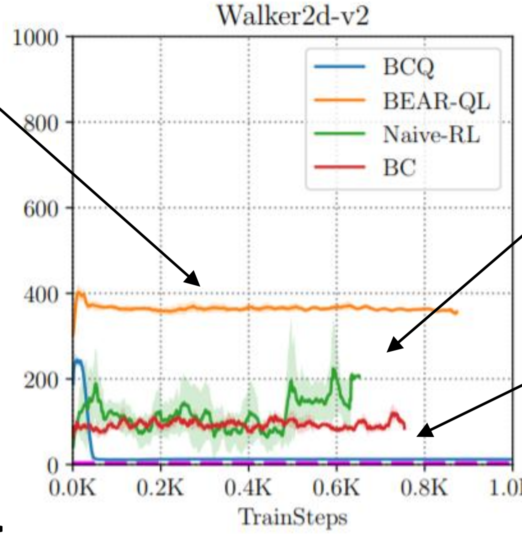
$Q(\mathbf{s},\mathbf{a})$ is trained on $(\mathbf{s},\mathbf{a}) \sim \beta(\mathbf{s},\mathbf{a})$

only use values inside support region



random data

our method



naïve RL

distrib. matching (Fujimoto)

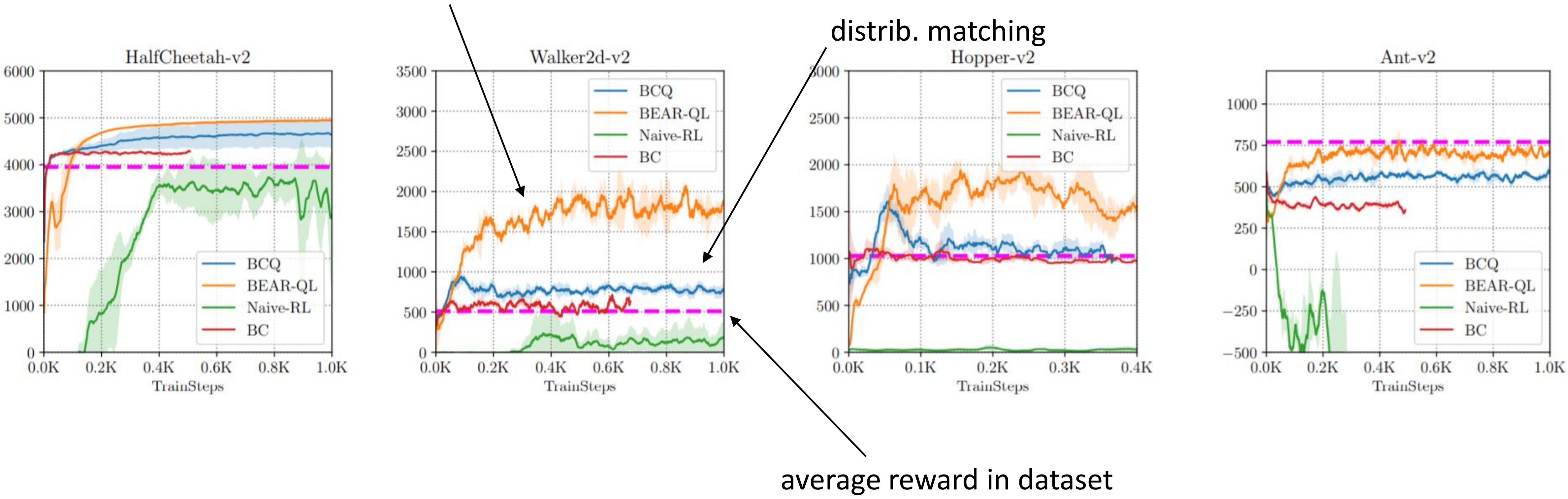
Kumar, Fu, Tucker, Levine. **Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction.**

See also: Fujimoto, Meger, Precup. **Off-Policy Deep Reinforcement Learning without Exploration.**

How well does it work?

"mediocre" data

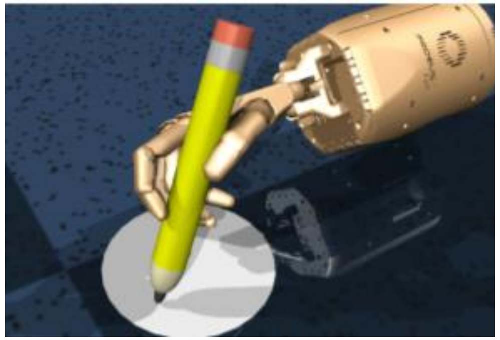
our method



Kumar, Fu, Tucker, Levine. **Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction.**

See also: Fujimoto, Meger, Precup**. Off-Policy Deep Reinforcement Learning without Exploration.**

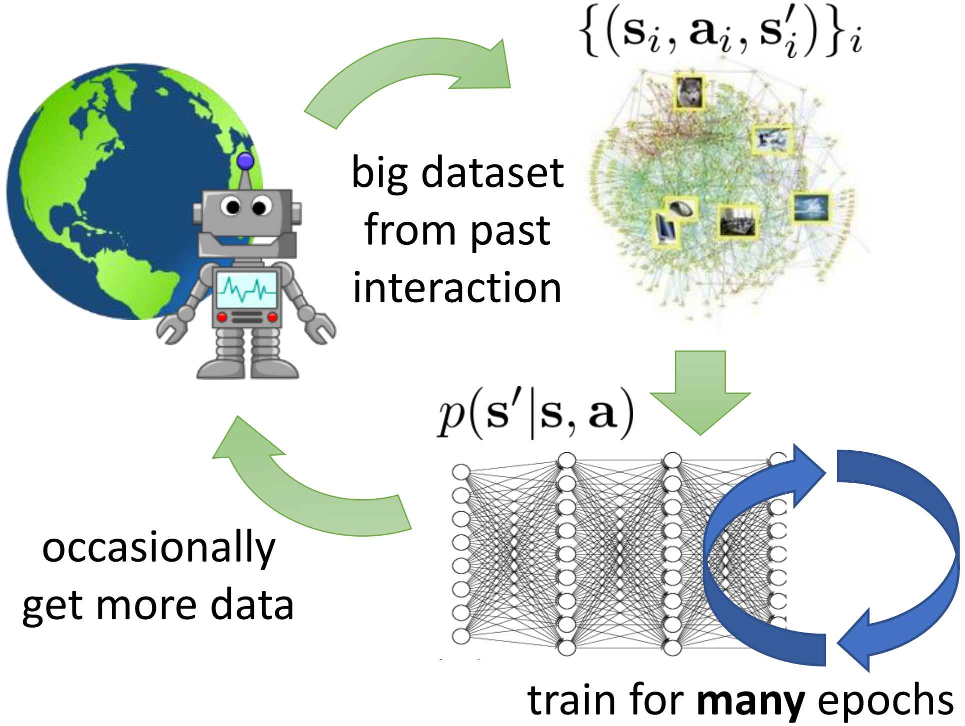
Off-policy model-free RL algorithms



Off-policy model-based RL algorithms

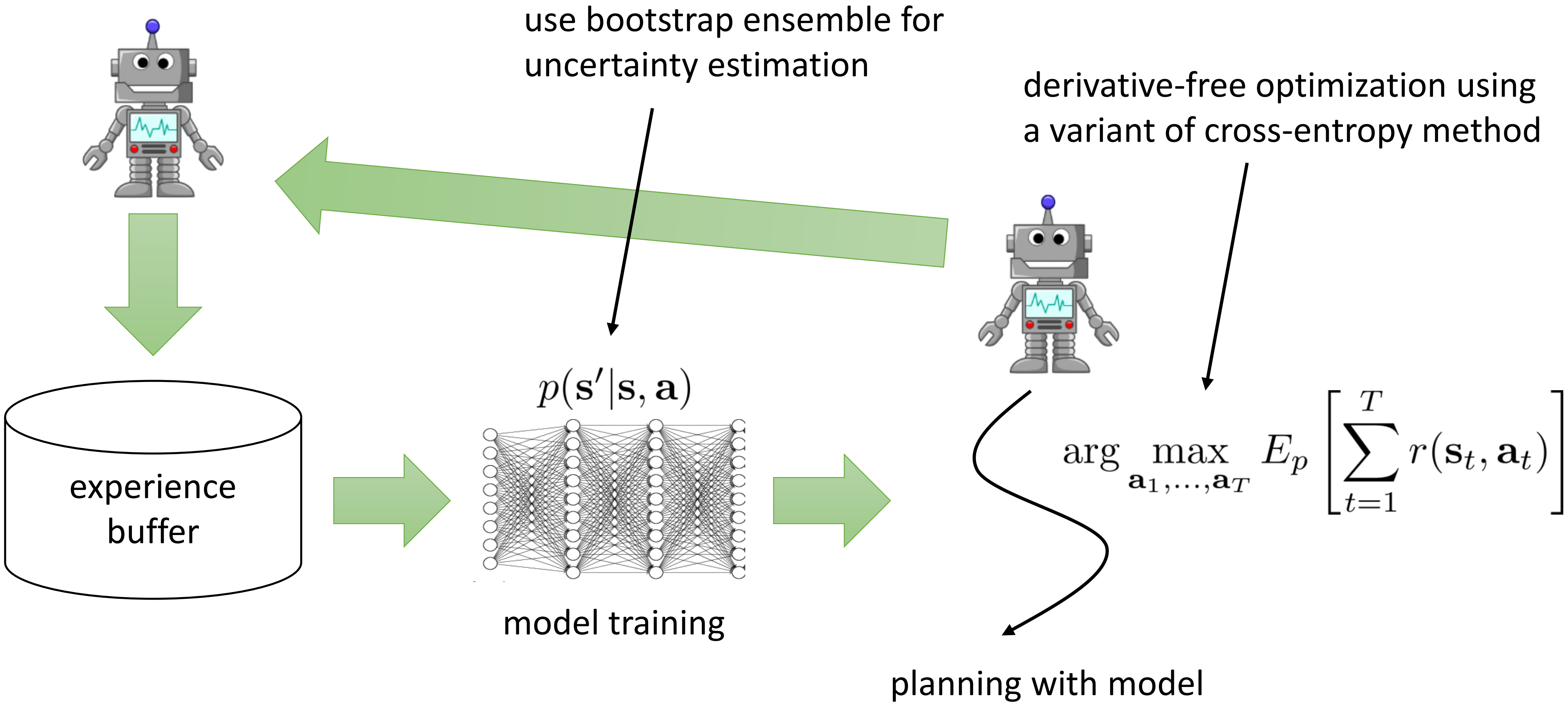
Why these are actually the same thing

Off-policy model-based reinforcement learning



- *Comparatively simple* ***supervised*** *learning problem*
- *Natural and straightforward to apply to* ***multi-task*** *settings (all tasks have the same physics)*
- *Models can be repurposed to new tasks without any additional learning*
- *Typically worse final performance than model-free RL – is that always true?*

High-level algorithm outline



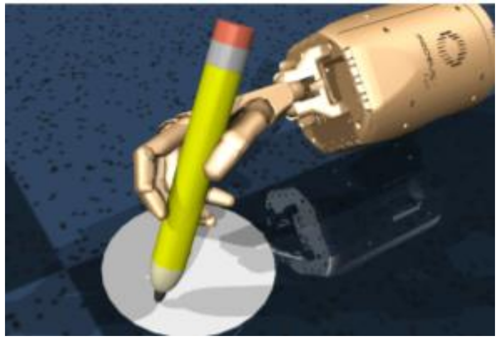
Nagabandi, Konolige, Levine, Kumar. **Deep Dynamics Models for Learning Dexterous Manipulation.** CoRL 2019.

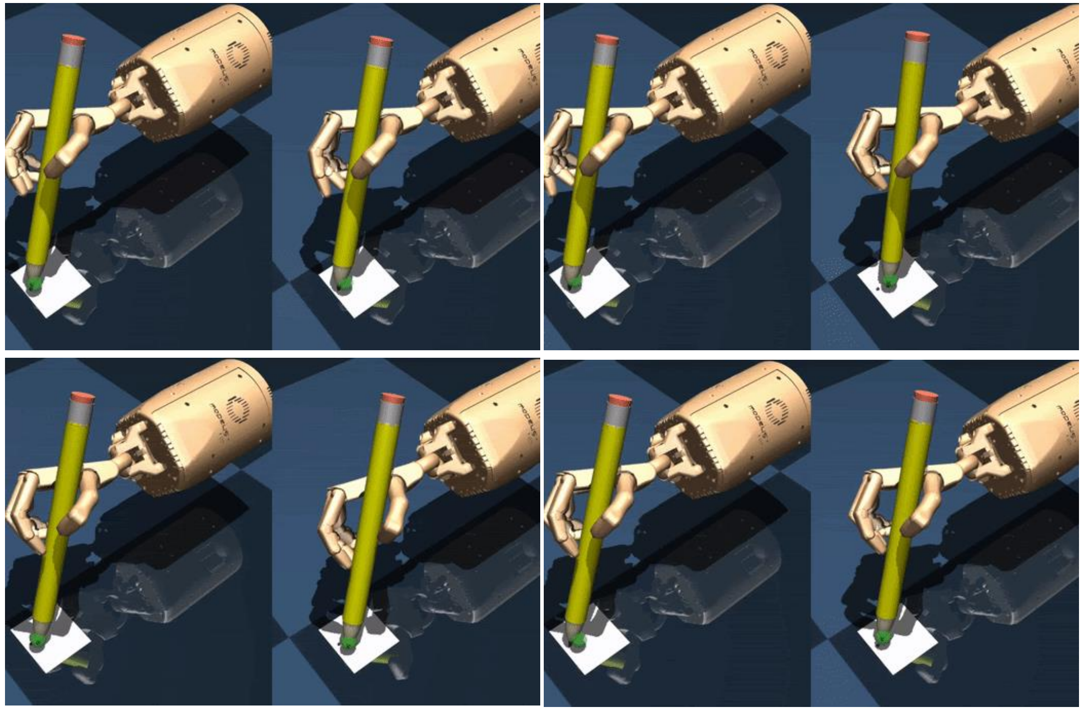
Model-based RL for dexterous manipulation

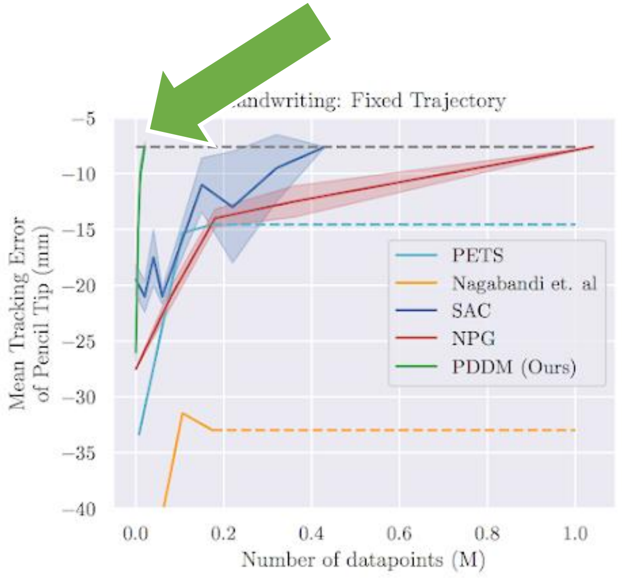
When should we **prefer** model-based RL?

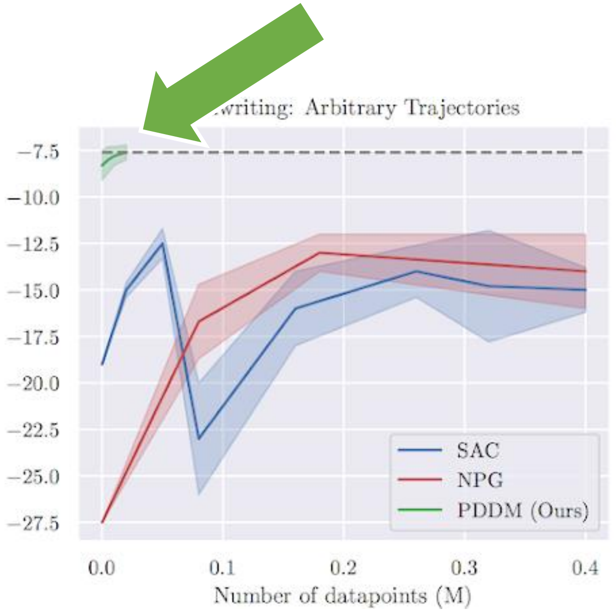
- *On **narrow** tasks, model-free RL tends to do very well*

- *On **broad and diverse** tasks, model-based RL has a major advantage!*







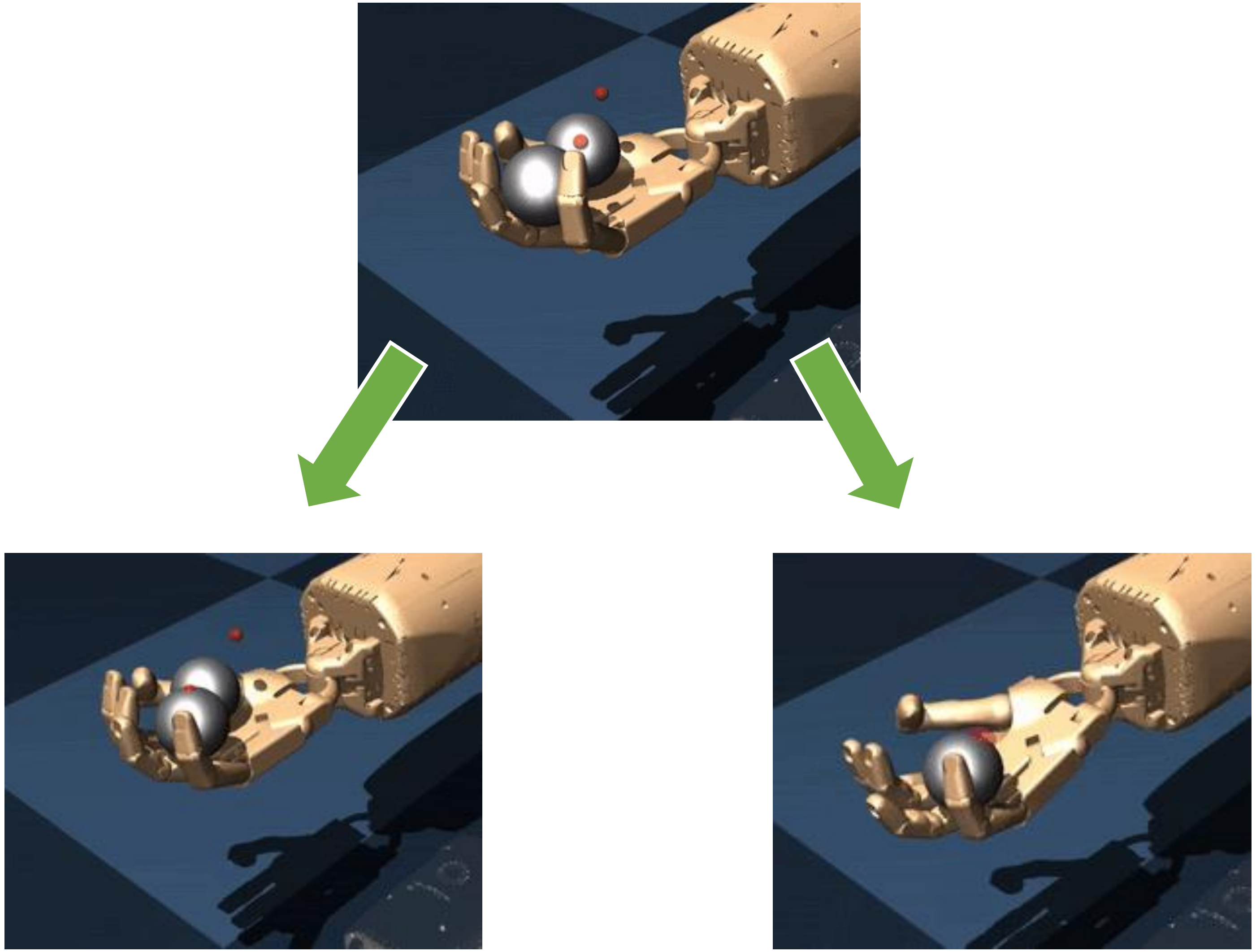


Nagabandi, Konolige, Levine, Kumar. **Deep Dynamics Models for Learning Dexterous Manipulation.** CoRL 2019.

Model-based RL for dexterous manipulation

When should we **prefer** model-based RL?

- *When we want to transfer the same dynamics model to perform multiple tasks*

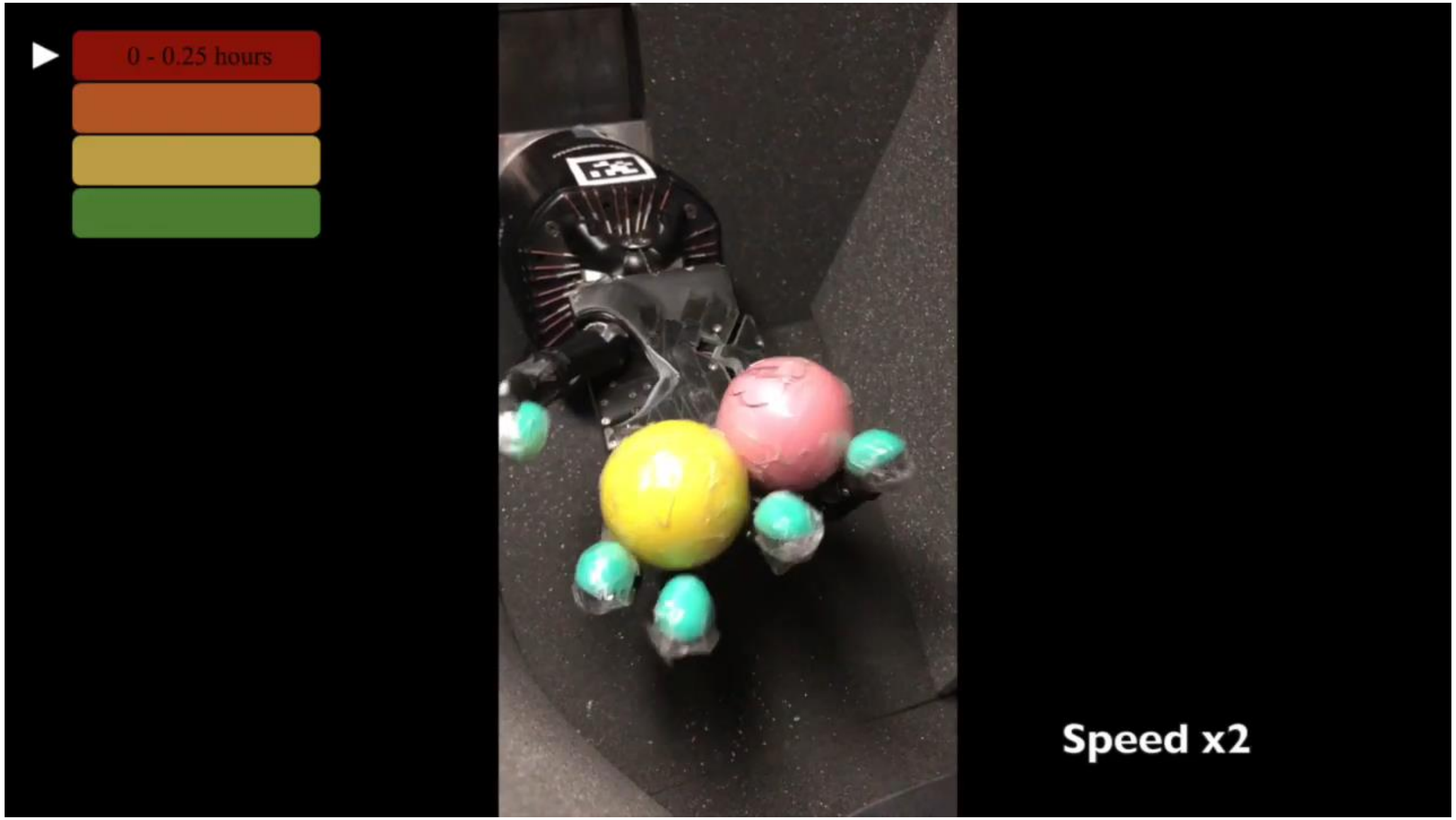


Nagabandi, Konolige, Levine, Kumar. **Deep Dynamics Models for Learning Dexterous Manipulation.** CoRL 2019.

Model-based RL for dexterous manipulation

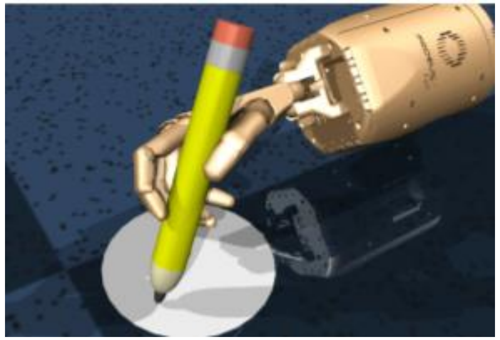
When should we **prefer** model-based RL?

- *When we want to learn in the real world with just a few hours of interaction*



Nagabandi, Konolige, Levine, Kumar. **Deep Dynamics Models for Learning Dexterous Manipulation.** CoRL 2019.

Off-policy model-free RL algorithms

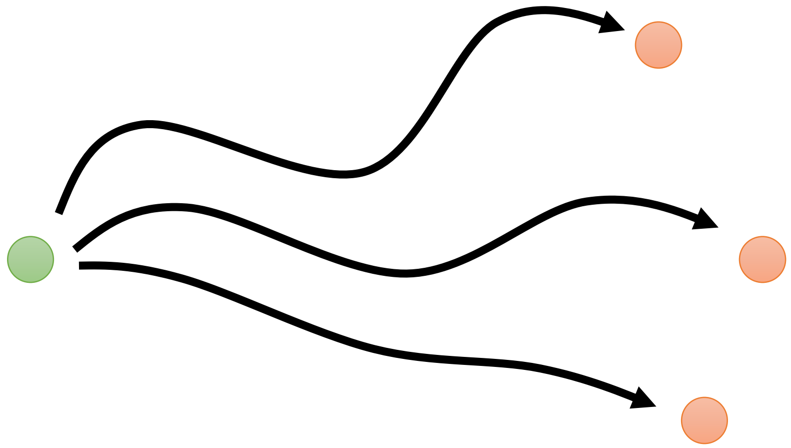


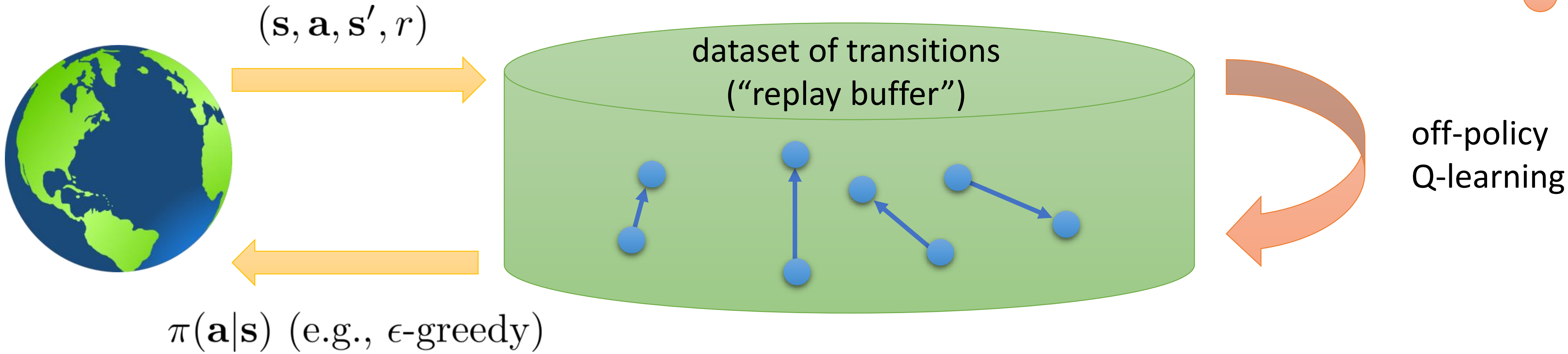
Off-policy model-based RL algorithms

Why these are actually the same thing

Q-Functions (can) learn models

$$Q^\star(\mathbf{s},\mathbf{a},\mathbf{g}) = r(\mathbf{s},\mathbf{a},\mathbf{g}) + \max_{\mathbf{a}'} Q^\star(\mathbf{s}',\mathbf{a}',\mathbf{g})$$





1. sample a batch $(\mathbf{s}_i,\mathbf{a}_i,\mathbf{s}'_i,\mathbf{g}_i,r_i)$ from $\mathcal{B}$
2. minimize $\sum_i (Q(\mathbf{s}_i,\mathbf{a}_i,\mathbf{g}_i) - [r(\mathbf{s}_i,\mathbf{a}_i,\mathbf{g}_i) + \max_{\mathbf{a}'_i} Q(\mathbf{s}'_i,\mathbf{a}'_i,\mathbf{g}_i)])^2$

Q-Functions (can) learn models

Before:

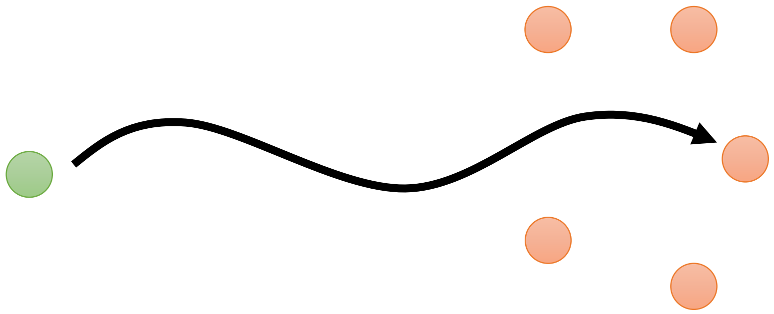
1. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, \mathbf{g}_i, r_i)$ from $\mathcal{B}$
2. minimize $\sum_i (Q(\mathbf{s}_i, \mathbf{a}_i, \mathbf{g}_i) - [r(\mathbf{s}_i, \mathbf{a}_i, \mathbf{g}_i) + \max_{\mathbf{a}'_i} Q(\mathbf{s}'_i, \mathbf{a}'_i, \mathbf{g}_i)])^2$

Kaelbling et al.
"Learning to Reach Goals"
Andrychowicz et al.
"Hindsight Experience Replay"

Now:

1. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{B}$
2. sample $\mathbf{g}_i$ from some distribution $p(\mathbf{g})$
3. minimize $\sum_i (Q(\mathbf{s}_i, \mathbf{a}_i, \mathbf{g}_i) - [r(\mathbf{s}_i, \mathbf{a}_i, \mathbf{g}_i) + \max_{\mathbf{a}'_i} Q(\mathbf{s}'_i, \mathbf{a}'_i, \mathbf{g}_i)])^2$

special case that samples terminal state

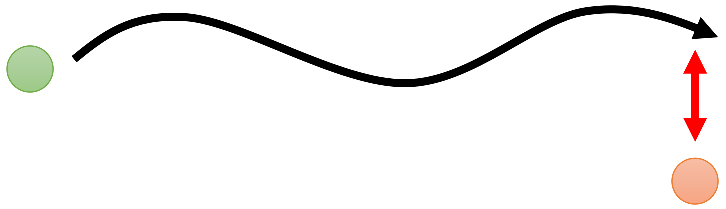


This results in **much** faster learning

Why??

Q-Functions (can) learn models

$$Q^\star(\mathbf{s},\mathbf{a},\mathbf{g},\tau) = r(\mathbf{s},\mathbf{a},\mathbf{g},\tau) + \max_{\mathbf{a}'} Q^\star(\mathbf{s}',\mathbf{a}',\mathbf{g},\tau-1)$$

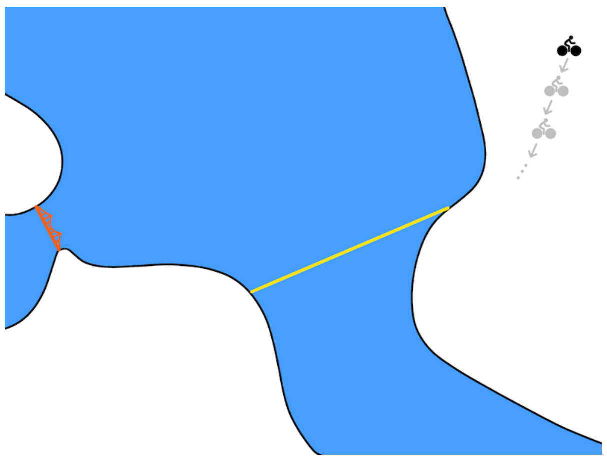


$$r(\mathbf{s},\mathbf{a},\mathbf{g},\tau) = \begin{cases} 0 \text{ if } \tau > 0 \\ -||\mathbf{s}-\mathbf{g}||^2 \text{ otherwise} \end{cases}$$

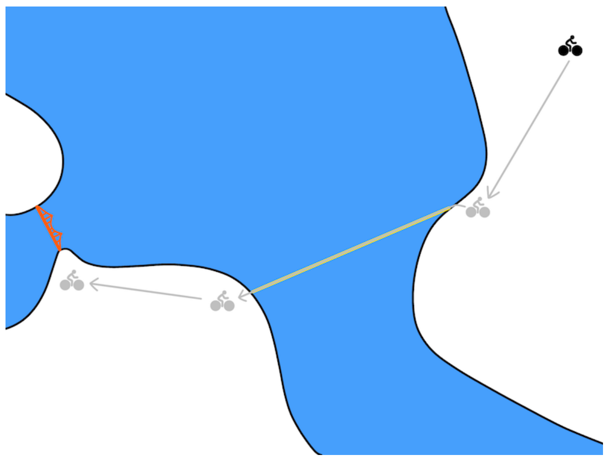
"if I'm trying to reach $\mathbf{g}$ in τ steps, how close will I get?"

this is a model, and can be used like one!

$$\max_{\mathbf{a}_{1:T},\mathbf{s}_{2:T}} \sum_t r(\mathbf{s}_t,\mathbf{a}_t) \quad \text{s.t. } Q(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1},1) = 0$$



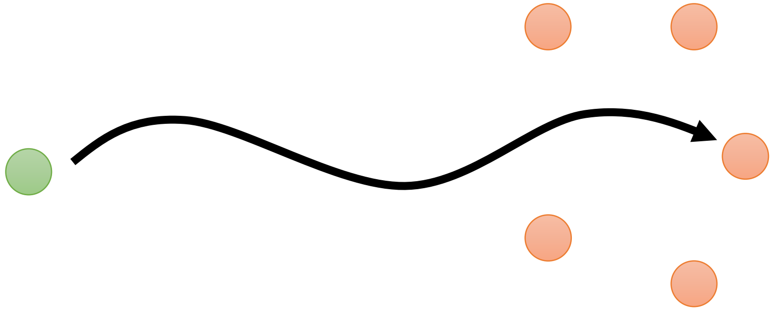
$$\max_{\mathbf{a}_{1:K:T},\mathbf{s}_{2:K:T}} r(\mathbf{s}_T) \quad \text{s.t. } Q(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+K},K) = 0$$

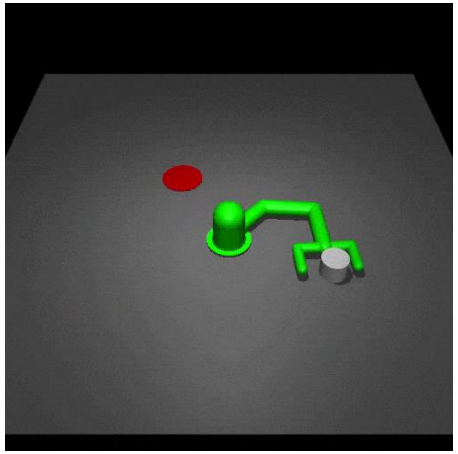


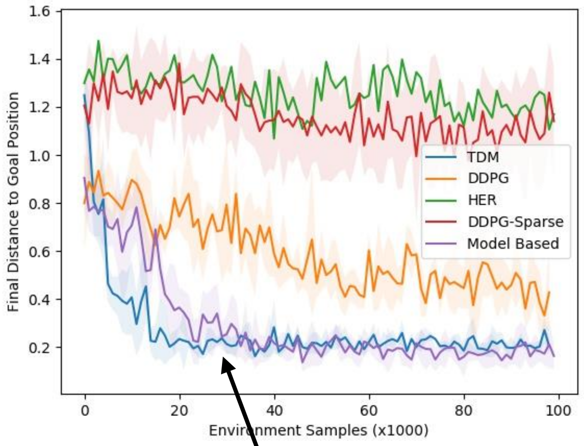
Temporal difference models

$$Q^\star(\mathbf{s},\mathbf{a},\mathbf{g},\tau) = r(\mathbf{s},\mathbf{a},\mathbf{g},\tau) + \max_{\mathbf{a}'} Q^\star(\mathbf{s}',\mathbf{a}',\mathbf{g},\tau-1)$$

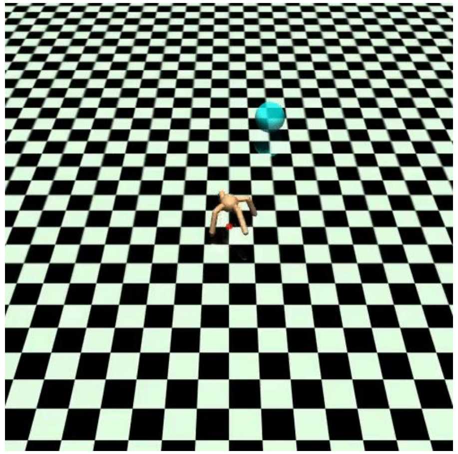
$$r(\mathbf{s},\mathbf{a},\mathbf{g},\tau) = \begin{cases} 0 \text{ if } \tau > 0 \\ -||\mathbf{s}-\mathbf{g}||^2 \text{ otherwise} \end{cases}$$

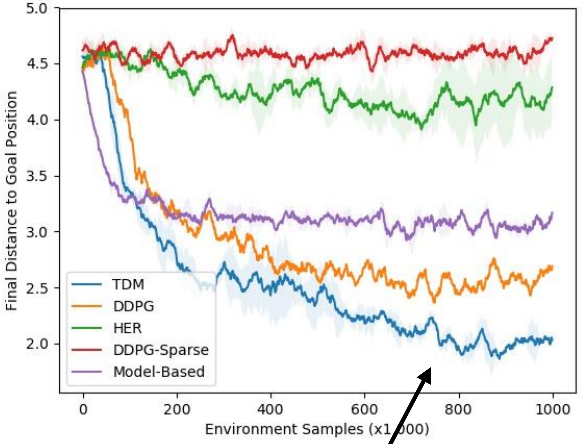






as fast as model-based learning





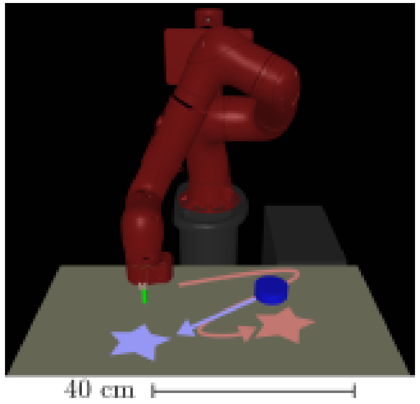
asymptotic performance as good (or better) vs model-free

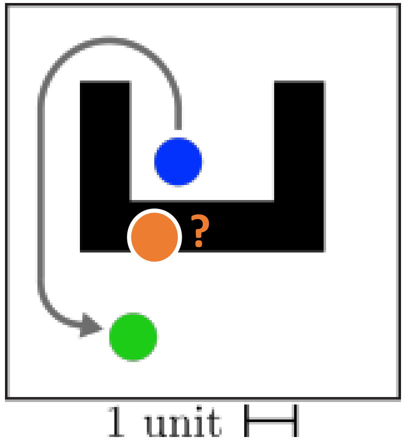
Temporal Difference Models.
Pong*, Gu*, Dalal, L.

Planning with TDMs

$$\max_{\mathbf{a}_{1:K:T}, \mathbf{s}_{2:K:T}} r(\mathbf{s}_T) \quad \text{s.t.} \quad Q(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+K}, K) = 0$$

optimization over state variables

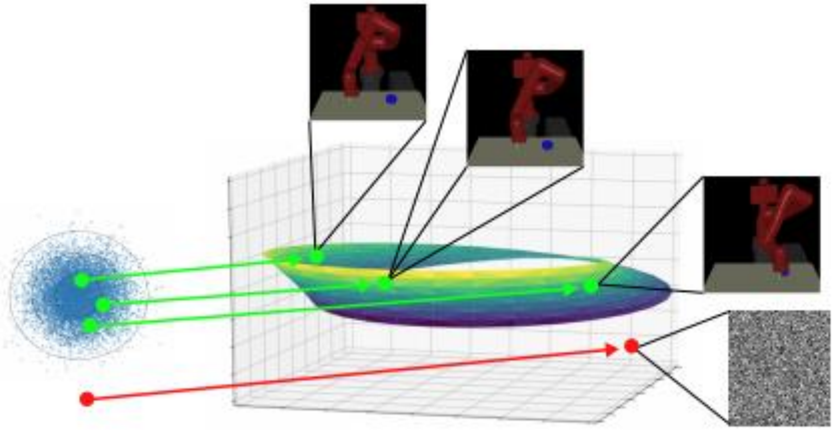




invalid states should be unreachable

but invalid states are *out of distribution*!

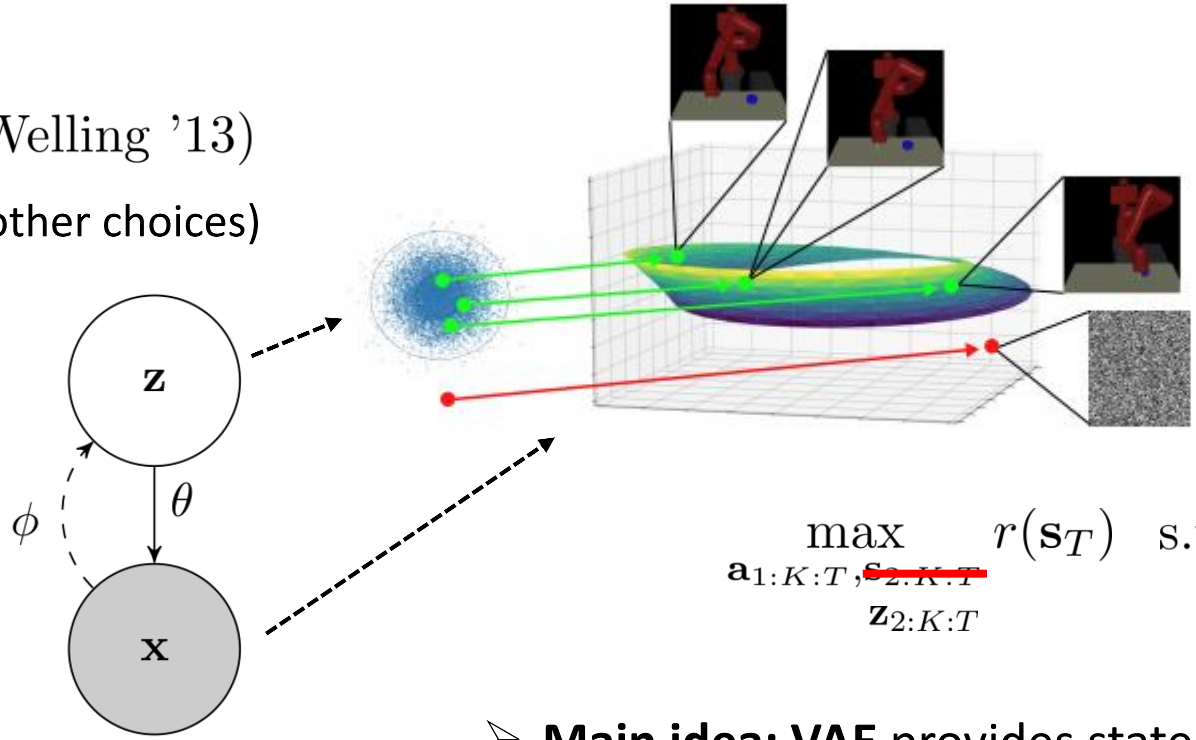
cannot make good predictions for OOD states!



Optimizing over *valid* states

VAE (Kingma & Welling '13)

(but there are many other choices)

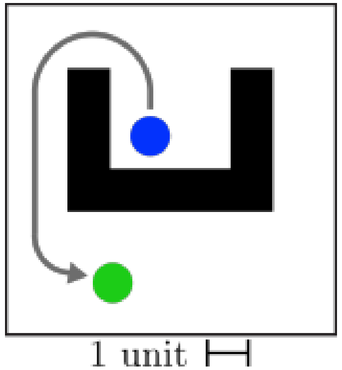


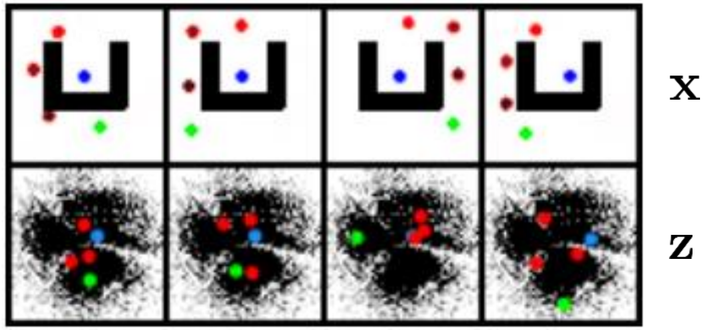
$$\max_{\substack{\mathbf{a}_{1:K:T},\cancel{\mathbf{s}_{2:K:T}} \\ \mathbf{z}_{2:K:T}}} r(\mathbf{s}_T) \quad \text{s.t.} \quad Q(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+K}, K) = 0$$

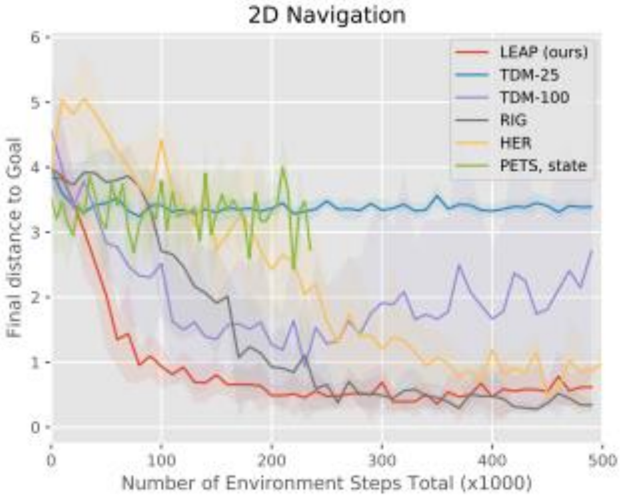
- **Main idea: VAE** provides state abstraction, **TDM** provides temporal abstraction
- **Details (not covered here) matter:** how to turn this into unconstrained optimization, how to optimize, etc. (see paper)

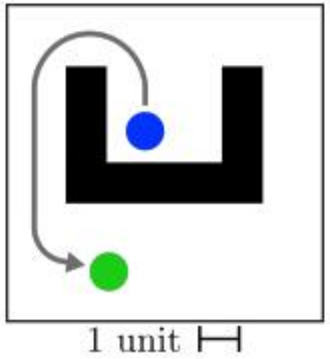
Planning with Goal-Conditioned Policies.
Nasiriany*, Pong*, Lin, L.

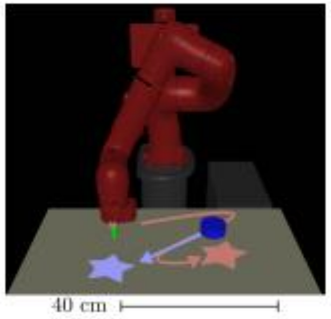
LEAP: Latent Embeddings for Abstracted Planning







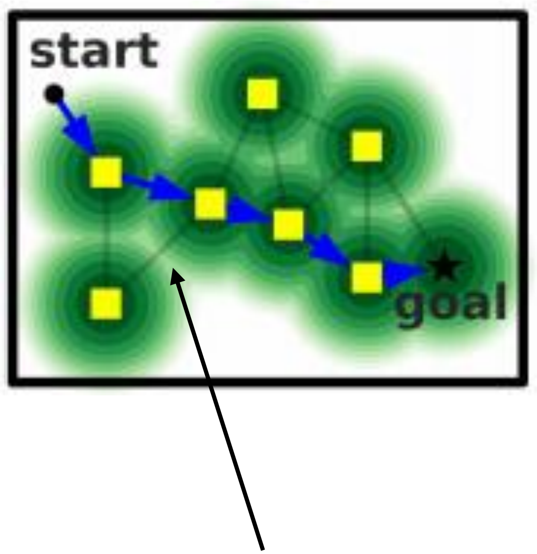




Planning with Goal-Conditioned Policies.

Nasiriany*, Pong*, Lin, L.

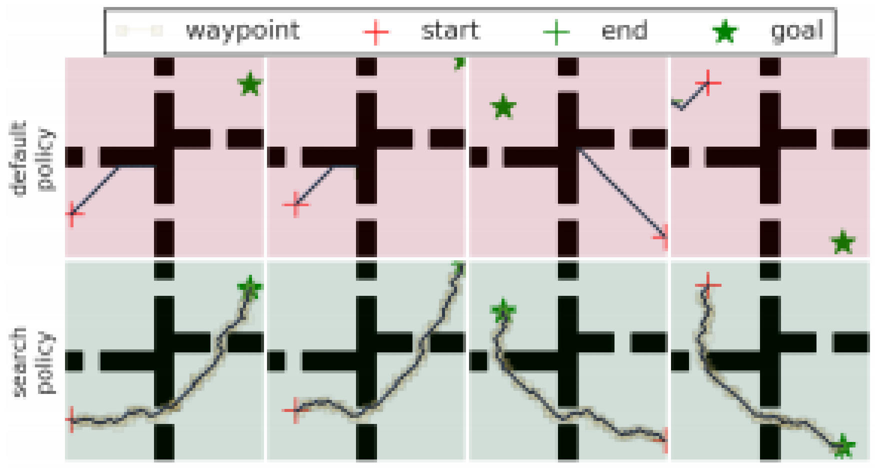
Graph search with goal-conditioned values



$$c(\mathbf{s}_i, \mathbf{s}_j) = -V(\mathbf{s}_i, \mathbf{s}_j)$$

$$r(\mathbf{s}, \mathbf{a}, \mathbf{g}) = \begin{cases} -1 \text{ if } \mathbf{s} \neq \mathbf{g} \\ 0 \text{ otherwise} \end{cases}$$

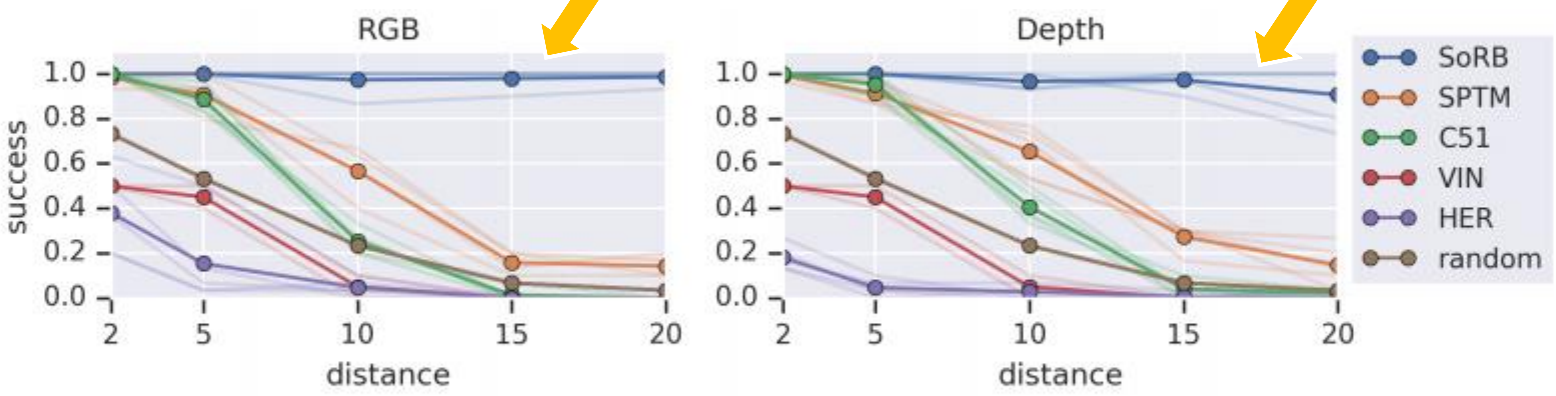
- Can search through graph using **any** graph search algorithm
- Where do we get the graph nodes?
- Use the entire replay buffer!



Search on the Replay Buffer: Bridging Planning and Reinforcement Learning

Eysenbach, Salakhutdinov, L.

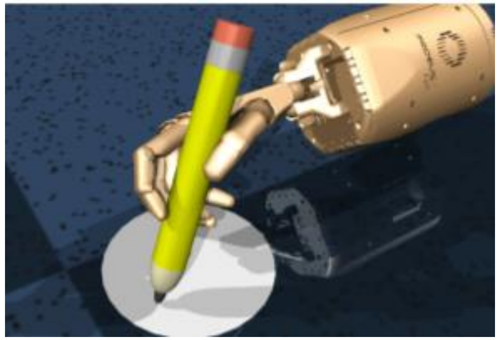
Search on the Replay Buffer (SoRB)



Search on the Replay Buffer: Bridging Planning and Reinforcement Learning

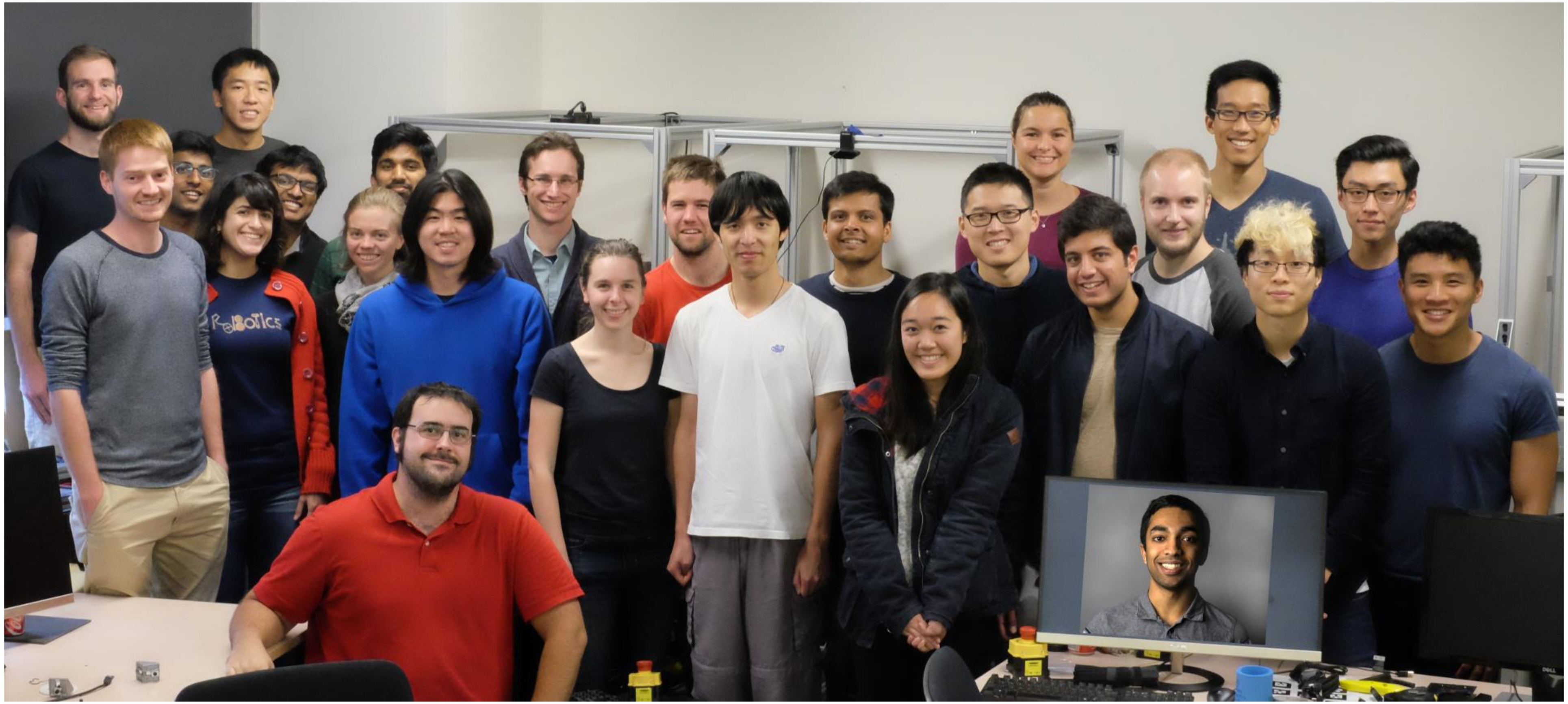
Eysenbach, Salakhutdinov, L.

Off-policy model-free RL algorithms



Off-policy model-based RL algorithms

Why these are actually the same thing



RAIL
Robotic AI & Learning Lab

website: http://rail.eecs.berkeley.edu
source code: http://rail.eecs.berkeley.edu/code.html