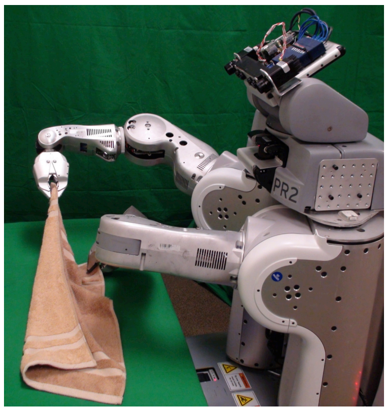
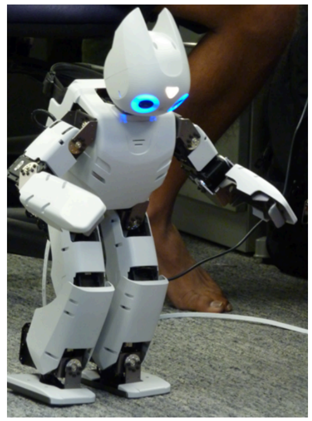
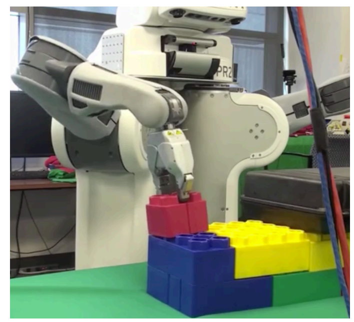






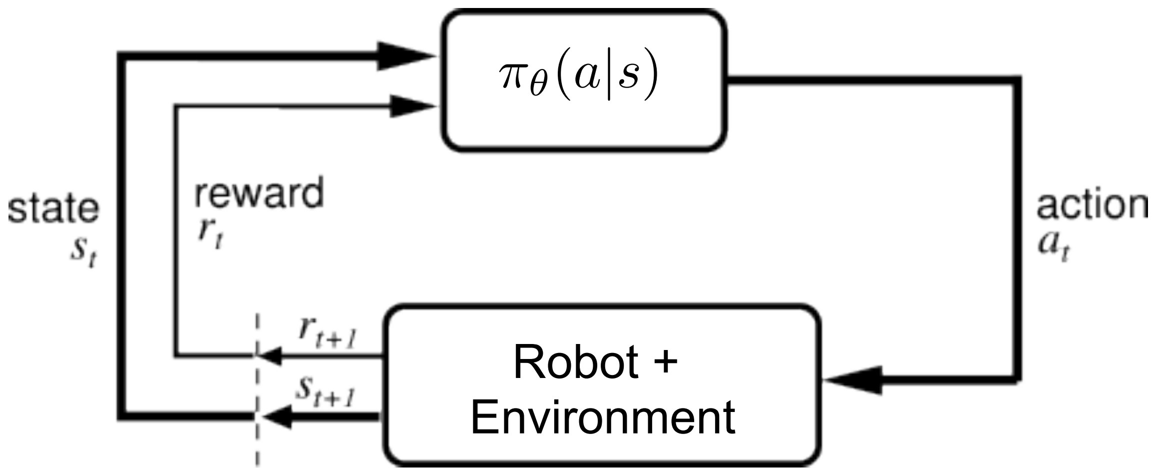
Deep Reinforcement Learning for Robotics

Pieter Abbeel

UC Berkeley / OpenAI / gradescope.com [code: ICML2016]

Deep Reinforcement Learning (RL)

probability of taking action a in state s

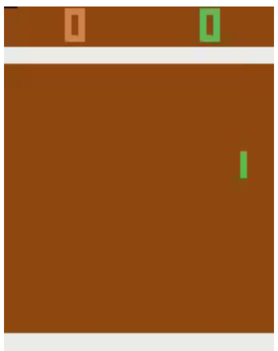


- Goal:

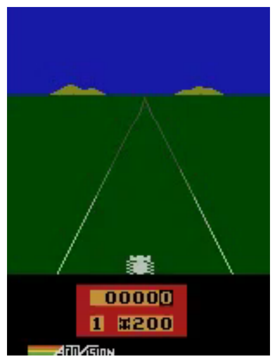
$$\max_\theta \quad \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

- **Additional challenges:**
  - **Stability**
  - **Credit assignment**
  - **Exploration**

From Pixels to Actions?

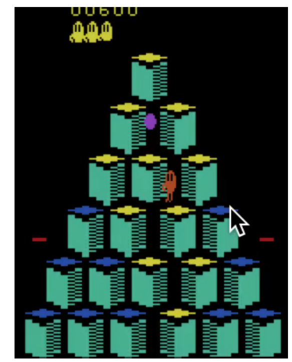


Pong



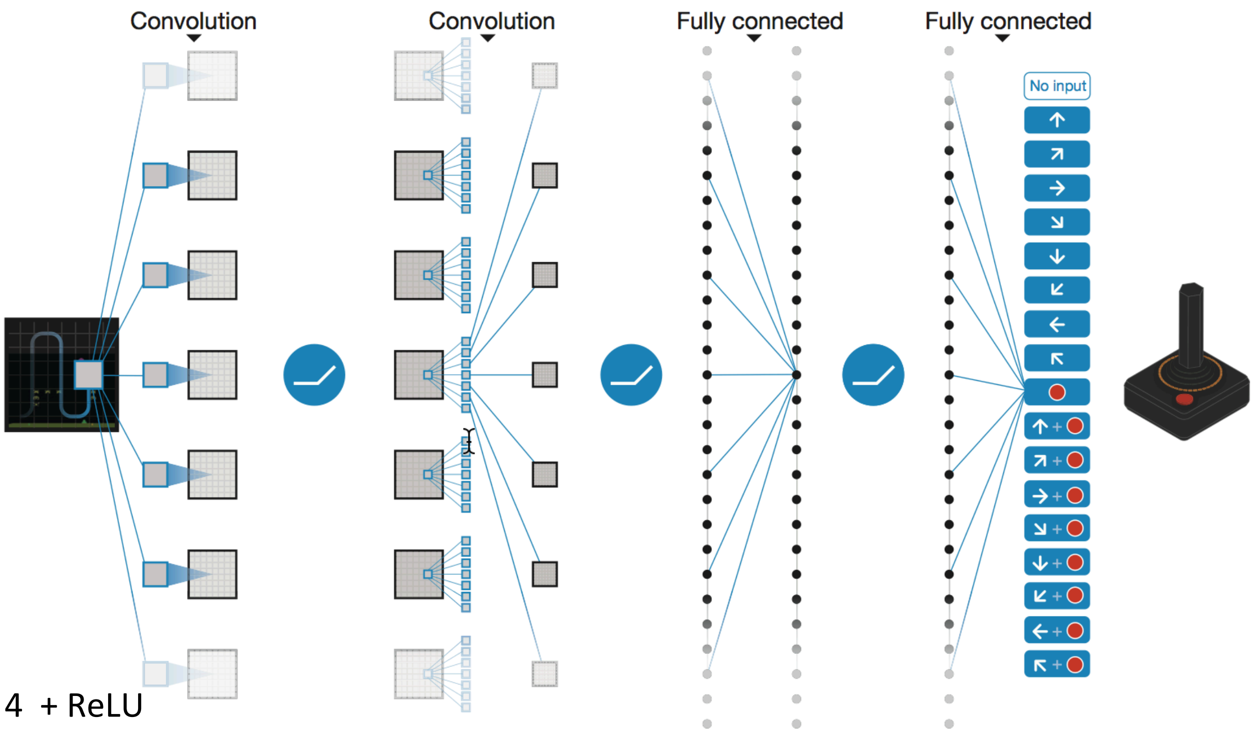
Enduro

Beamrider



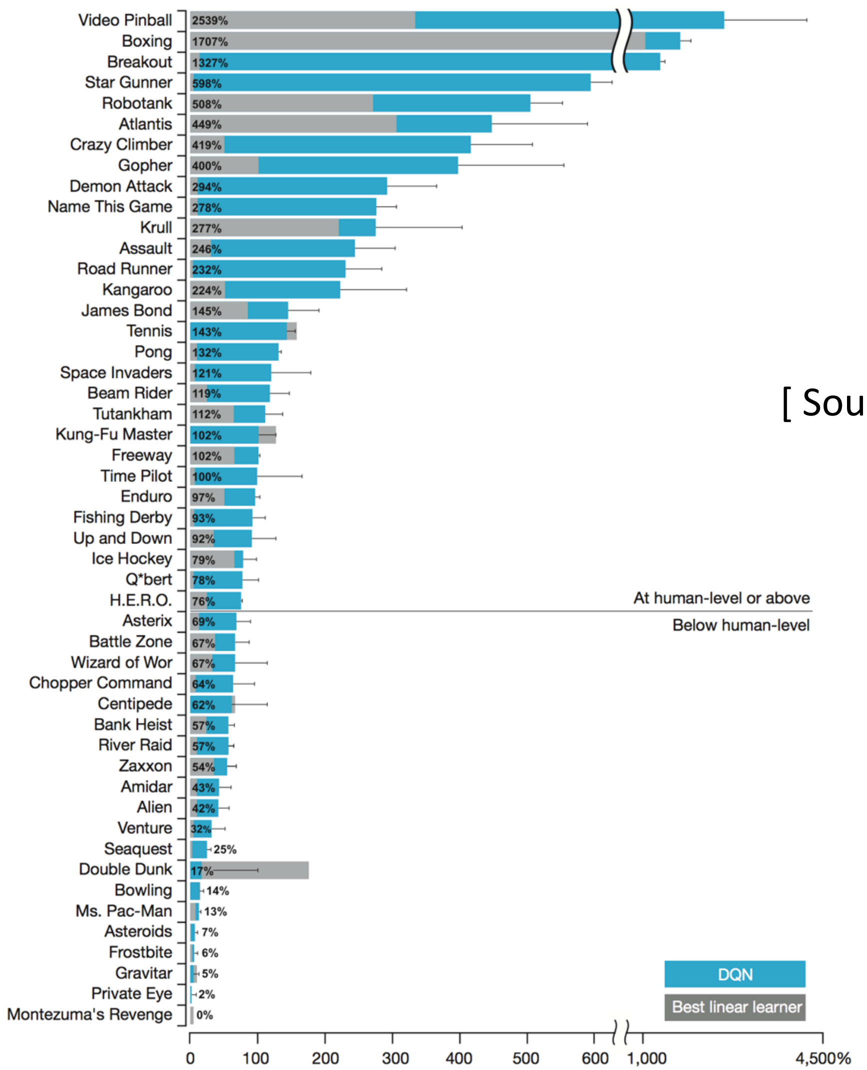
Q*bert

Deep Q-Network (DQN): From Pixels to Joystick Commands



32 8x8 filters with stride 4 + ReLU
64 4x4 filters with stride 2 + ReLU
64 3x3 filters with stride 1 + ReLU
fully connected 512 units + ReLU
fully connected output units, one per action

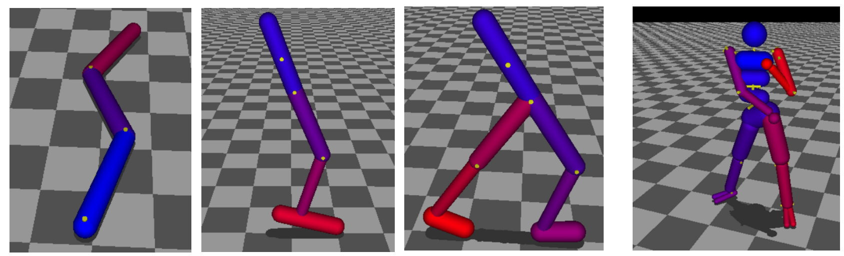
[Source: Mnih et al., Nature 2015 (DeepMind)]



[Source: Mnih et al., Nature 2015 (DeepMind)]

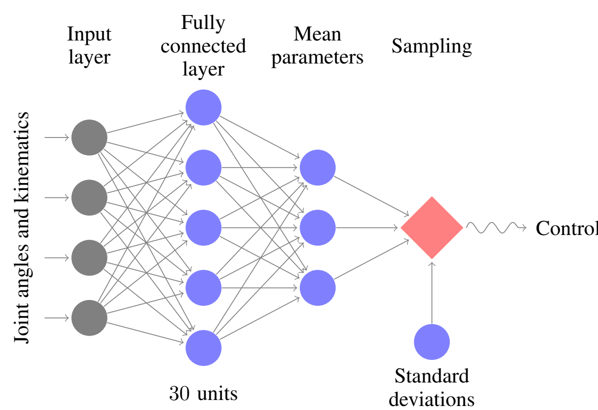
How About Continuous Control, e.g., Locomotion?

Robot models in physics simulator
(MuJoCo, from Emo Todorov)



Input: joint angles and velocities
Output: joint torques

Neural network architecture:



Challenges with Q-Learning

- How to score every possible action?
- How to ensure monotonic progress?

Policy Optimization

$$\max_{\theta} \quad \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

- Often simpler to represent good policies than good value functions
- True objective of expected cost is optimized (vs. a surrogate like Bellman error)

- Existing work: (natural) policy gradients
  - Challenges: good, large step directions

Trust Region Policy Optimization

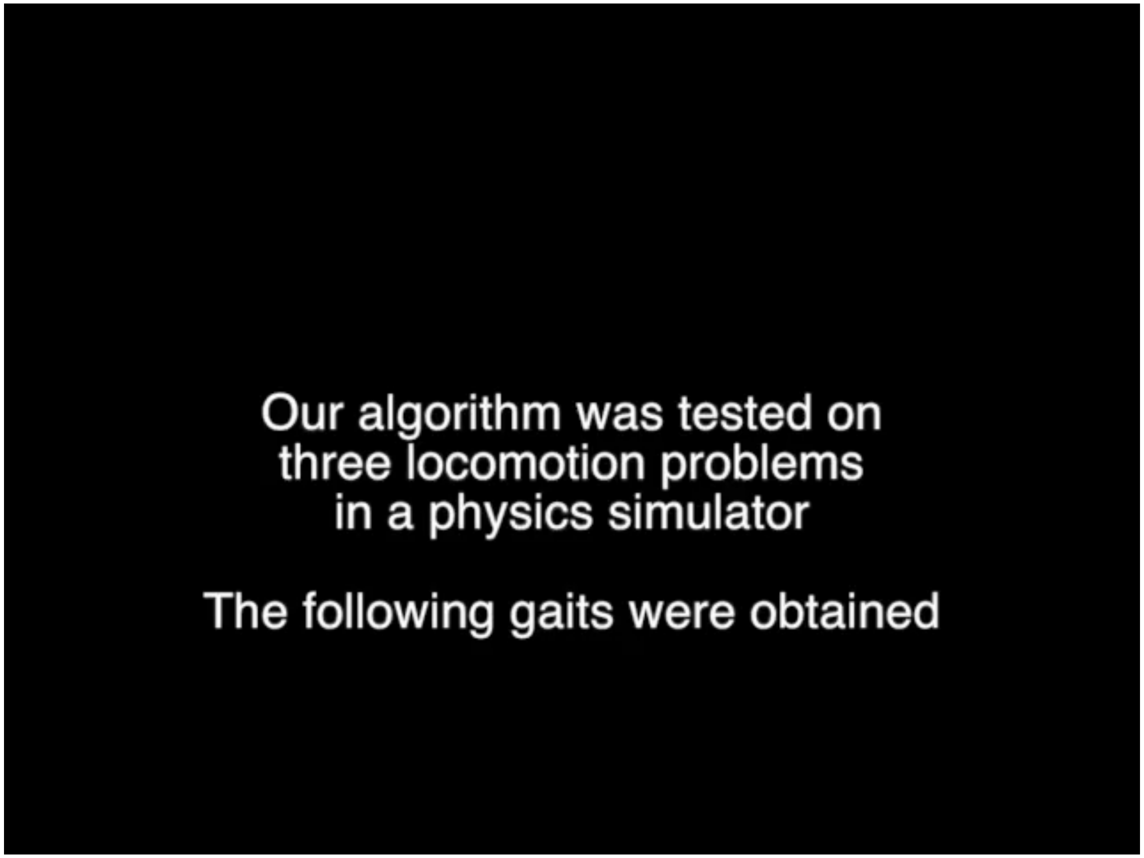
[Schulman, Levine, Moritz, Jordan, Abbeel, 2015]

$$\max_{\theta} \quad \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

$$\begin{aligned} \max_{\delta\theta} \quad & \hat{L}(\theta + \delta\theta) \\ \text{s.t.} \quad & \mathrm{KL}\left(P(\tau;\theta)||P(\tau;\theta+\delta\theta)\right) \leq \varepsilon \end{aligned}$$

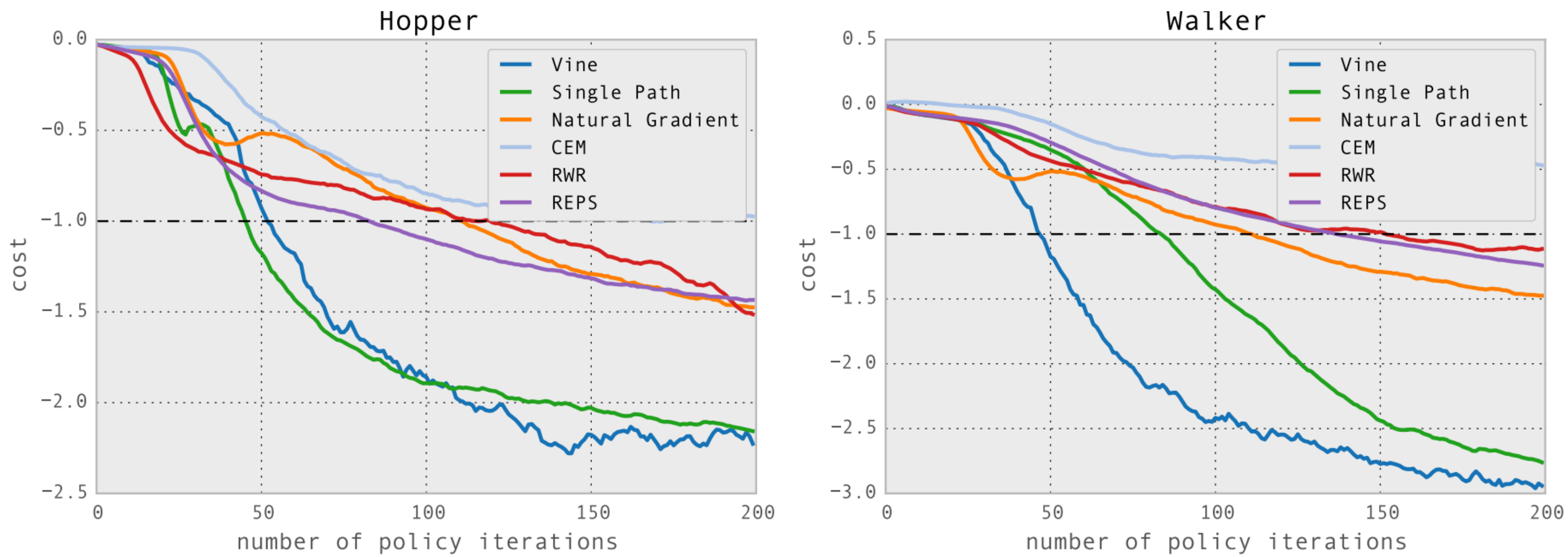
- $\hat{L}$: Surrogate Objective
- KL: Trust region

Experiments in Locomotion



[Schulman, Levine, A.]

Learning Curves -- Comparison



Generalized Advantage Estimation (GAE)

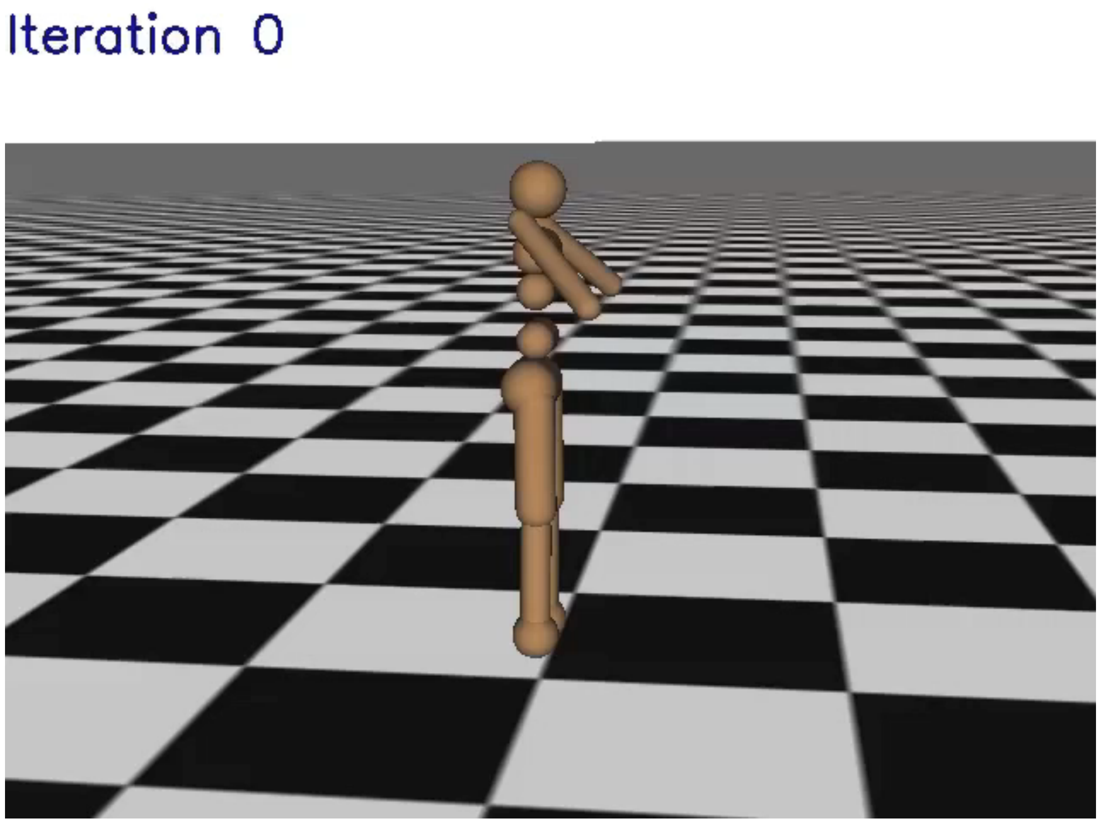
Objective: $$\max_{\theta} \quad \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

Gradient: $$\mathrm{E}[\sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\underbrace{\sum_{k=t}^{H} R(s_k) - V(s_t)}_{\text{single sample estimate of advantage}} \right)]$$

- Generalized Advantage Estimation
  - Exponential interpolation between actor-critic and Monte Carlo estimates
  - Trust region approach to (high-dimensional) value function estimation

[Schulman, Moritz, Levine, Jordan, Abbeel, ICLR 2016]

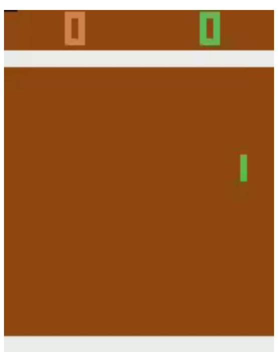
Learning Locomotion through Trust Region Policy Optimization (TRPO)



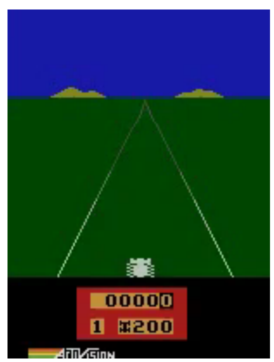
[Schulman, Moritz, Levine, Jordan, Abbeel, ICLR 2016]

Atari Games

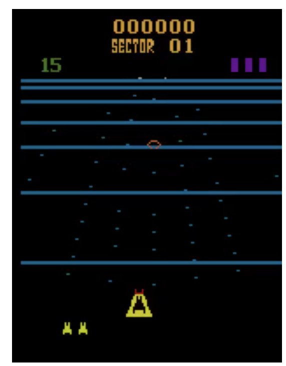
- Deep Q-Network (DQN) [Mnih et al, 2013/2015]
- Dagger with Monte Carlo Tree Search [Xiao-Xiao et al, 2014]
- Trust Region Policy Optimization [Schulman, Levine, Moritz, Jordan, Abbeel, 2015]
- A3C [Mnih et al., 2016]



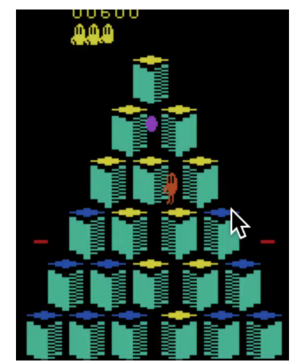
Pong



Enduro



Beamrider



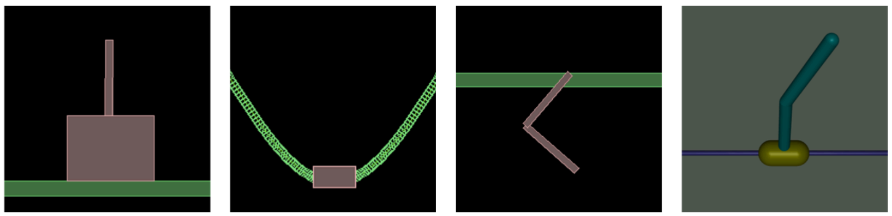
Q*bert

Deep RL Benchmarking

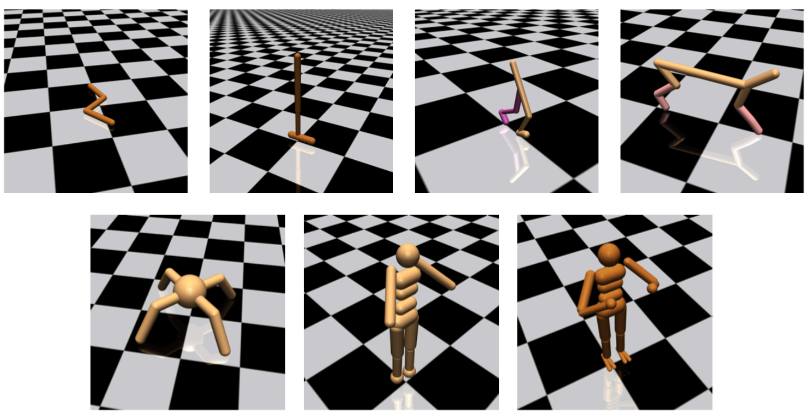
- Tasks
- Algorithms
- Experimental setup

Deep RL Benchmarking -- Tasks

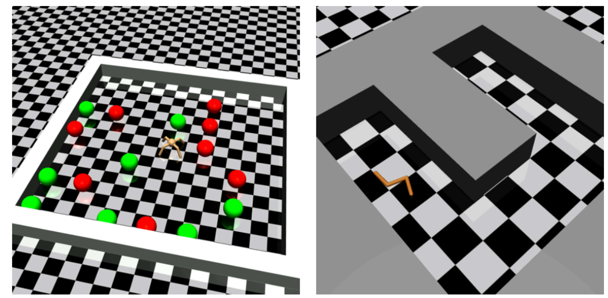
1. Basic tasks



2. Locomotion



3. Hierarchical



4. Partially observable

sensing, delayed action, sysID

5. Driving…

Deep RL Benchmarking -- Algorithms

- Reinforce
- Truncated Natural Policy Gradient
- Reward-Weighted Regression (RWR)
- Relative Entropy Policy Search (REPS)
- Trust-Region Policy Optimization (TRPO)
- Cross-Entropy Method (CEM)
- Covariance Matrix Adaptation Evolution Strategy (CMA-ES)
- Deep Deterministic Policy Gradients (DDPG)
- …

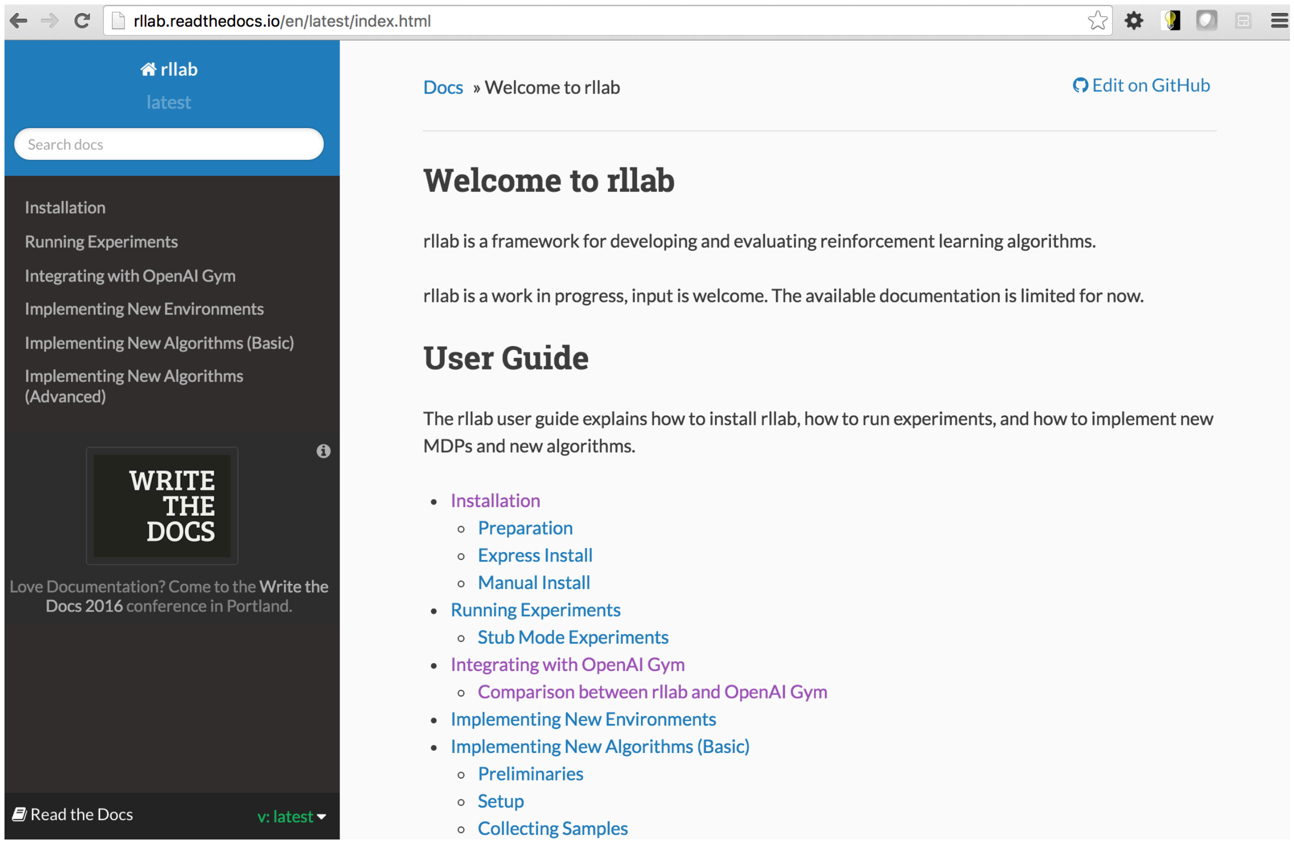
Benchmarking [Duan et al, ICML 2016]

Table 1. Performance of the implemented algorithms in terms of average return over all training iterations for five different random seeds (same across all algorithms). The results of the best-performing algorithm on each task are highlighted in boldface. In the tasks column, the partially observable variants of the tasks are annotated as follows: LS stands for limited sensors, NO for noisy observations and delayed actions, and SI for system identifications. The notation N/A denotes that an algorithm has failed on the task at hand, e.g., CMA-ES leading to out-of-memory errors in the Full Humanoid task.

Task	Random	VPG	TNPG	RWR	REPS	TRPO	CEM	CMA-ES
Cart-Pole Balancing	77.1 ± 0.0	4693.7 ± 14.0	3986.4 ± 748.9	4861.5 ± 12.3	565.6 ± 137.6	**4869.8 ± 37.6**	4815.4 ± 4.8	2440.4 ± 568.3
Inverted Pendulum*	−153.4 ± 0.2	13.4 ± 18.0	209.7 ± 55.5	84.7 ± 13.8	−113.3 ± 4.6	**247.2 ± 76.1**	38.2 ± 25.7	−40.1 ± 5.7
Mountain Car	−415.4 ± 0.0	−67.1 ± 1.0	−66.5 ± 4.5	−79.4 ± 1.1	−275.6 ± 166.3	**−61.7 ± 0.9**	−66.0 ± 2.4	−85.0 ± 7.7
Acrobot	−1904.5 ± 1.0	−508.1 ± 91.0	−395.8 ± 121.2	−352.7 ± 35.9	−1001.5 ± 10.8	**−326.0 ± 24.4**	−436.8 ± 14.7	−785.6 ± 13.1
Double Inverted Pendulum*	149.7 ± 0.1	4116.5 ± 65.2	**4455.4 ± 37.6**	3614.8 ± 368.1	446.7 ± 114.8	4412.4 ± 50.4	2566.2 ± 178.9	1576.1 ± 51.3
Swimmer*	−1.7 ± 0.1	92.3 ± 0.1	**96.0 ± 0.2**	60.7 ± 5.5	3.8 ± 3.3	**96.0 ± 0.2**	68.8 ± 2.4	64.9 ± 1.4
Hopper	8.4 ± 0.0	714.0 ± 29.3	1155.1 ± 57.9	553.2 ± 71.0	86.7 ± 17.6	**1183.3 ± 150.0**	63.1 ± 7.8	20.3 ± 14.3
2D Walker	−1.7 ± 0.0	506.5 ± 78.8	**1382.6 ± 108.2**	136.0 ± 15.9	−37.0 ± 38.1	1353.8 ± 85.0	84.5 ± 19.2	77.1 ± 24.3
Half-Cheetah	−90.8 ± 0.3	1183.1 ± 69.2	1729.5 ± 184.6	376.1 ± 28.2	34.5 ± 38.0	**1914.0 ± 120.1**	330.4 ± 274.8	441.3 ± 107.6
Ant*	13.4 ± 0.7	548.3 ± 55.5	706.0 ± 127.7	37.6 ± 3.1	39.0 ± 9.8	**730.2 ± 61.3**	49.2 ± 5.9	17.8 ± 15.5
Simple Humanoid	41.5 ± 0.2	87.6 ± 2.3	276.6 ± 31.7	56.7 ± 3.8	36.8 ± 4.0	**493.9 ± 75.5**	98.8 ± 11.5	115.9 ± 2.7
Full Humanoid	13.2 ± 0.1	214.4 ± 35.0	282.9 ± 19.8	42.3 ± 3.4	45.5 ± 4.1	**455.9 ± 22.8**	104.0 ± 14.5	N/A ± N/A
Cart-Pole Balancing (LS)*	77.1 ± 0.0	420.9 ± 265.5	945.1 ± 27.8	68.9 ± 1.5	898.1 ± 22.1	**960.2 ± 46.0**	227.0 ± 223.0	68.0 ± 1.6
Inverted Pendulum (LS)	−122.1 ± 0.1	−13.4 ± 3.2	0.7 ± 6.1	−107.4 ± 0.2	−87.2 ± 8.0	**4.5 ± 4.1**	−81.2 ± 33.2	−62.4 ± 3.4
Mountain Car (LS)	−83.0 ± 0.0	−81.2 ± 0.6	−65.7 ± 9.0	−81.7 ± 0.1	−82.6 ± 0.4	**−64.2 ± 9.5**	−68.9 ± 1.3	−73.2 ± 0.6
Acrobot (LS)*	−393.2 ± 0.0	−128.9 ± 11.6	−84.6 ± 2.9	−235.9 ± 5.3	−379.5 ± 1.4	**−83.3 ± 9.9**	−149.5 ± 15.3	−159.9 ± 7.5
Cart-Pole Balancing (NO)*	101.4 ± 0.1	616.0 ± 210.8	**916.3 ± 23.0**	93.8 ± 1.2	99.6 ± 7.2	606.2 ± 122.2	181.4 ± 32.1	104.4 ± 16.0
Inverted Pendulum (NO)	−122.2 ± 0.1	6.5 ± 1.1	**11.5 ± 0.5**	−110.0 ± 1.4	−119.3 ± 4.2	10.4 ± 2.2	−55.6 ± 16.7	−80.3 ± 2.8
Mountain Car (NO)	−83.0 ± 0.0	−74.7 ± 7.8	−64.5 ± 8.6	−81.7 ± 0.1	−82.9 ± 0.1	**−60.2 ± 2.0**	−67.4 ± 1.4	−73.5 ± 0.5
Acrobot (NO)*	−393.5 ± 0.0	−186.7 ± 31.3	−164.5 ± 13.4	−233.1 ± 0.4	−258.5 ± 14.0	**−149.6 ± 8.6**	−213.4 ± 6.3	−236.6 ± 6.2
Cart-Pole Balancing (SI)*	76.3 ± 0.1	431.7 ± 274.1	**980.5 ± 7.3**	69.0 ± 2.8	702.4 ± 196.4	980.3 ± 5.1	746.6 ± 93.2	71.6 ± 2.9
Inverted Pendulum (SI)	−121.8 ± 0.2	−5.3 ± 5.6	**14.8 ± 1.7**	−108.7 ± 4.7	−92.8 ± 23.9	14.1 ± 0.9	−51.8 ± 10.6	−63.1 ± 4.8
Mountain Car (SI)	−82.7 ± 0.0	−63.9 ± 0.2	−61.8 ± 0.4	−81.4 ± 0.1	−80.7 ± 2.3	**−61.6 ± 0.4**	−63.9 ± 1.0	−66.9 ± 0.6
Acrobot (SI)*	−387.8 ± 1.0	−169.1 ± 32.3	**−156.6 ± 38.9**	−233.2 ± 2.6	−216.1 ± 7.7	−170.9 ± 40.3	−250.2 ± 13.7	−245.0 ± 5.5
Swimmer + Gathering	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Ant + Gathering	−5.8 ± 5.0	**−0.1 ± 0.1**	−0.4 ± 0.1	−5.5 ± 0.5	−6.7 ± 0.7	−0.4 ± 0.0	−4.7 ± 0.7	N/A ± N/A
Swimmer + Maze	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Ant + Maze	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N/A ± N/A

rllab

[Duan et al, ICML 2016]



rllab.readthedocs.io/en/latest/index.html

rllab
latest

Search docs

- Installation
- Running Experiments
- Integrating with OpenAI Gym
- Implementing New Environments
- Implementing New Algorithms (Basic)
- Implementing New Algorithms (Advanced)

WRITE THE DOCS

Love Documentation? Come to the Write the Docs 2016 conference in Portland.

Read the Docs v: latest

Docs » Welcome to rllab

Edit on GitHub

Welcome to rllab

rllab is a framework for developing and evaluating reinforcement learning algorithms.

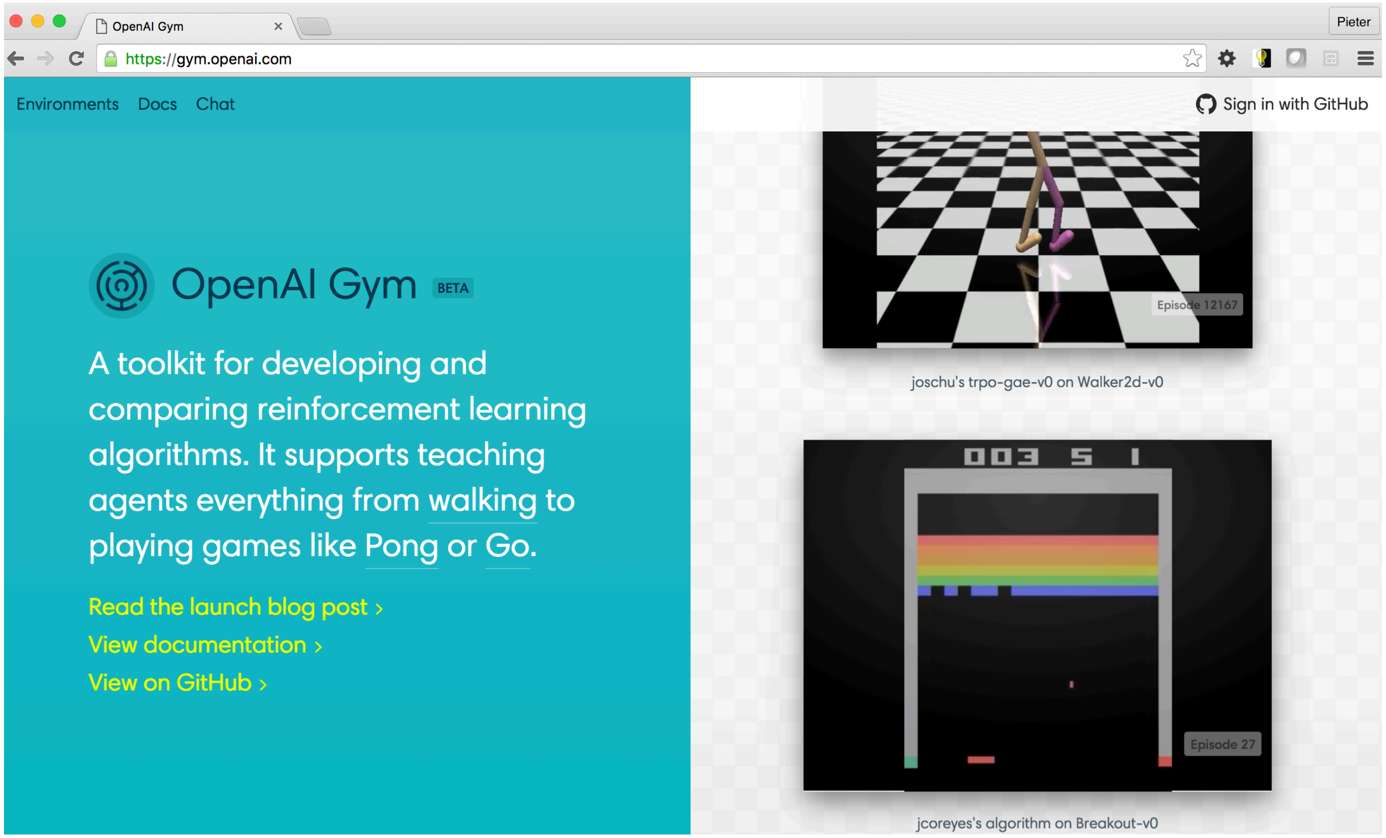
rllab is a work in progress, input is welcome. The available documentation is limited for now.

User Guide

The rllab user guide explains how to install rllab, how to run experiments, and how to implement new MDPs and new algorithms.

- Installation
  - Preparation
  - Express Install
  - Manual Install
- Running Experiments
  - Stub Mode Experiments
- Integrating with OpenAI Gym
  - Comparison between rllab and OpenAI Gym
- Implementing New Environments
- Implementing New Algorithms (Basic)
  - Preliminaries
  - Setup
  - Collecting Samples

Open AI Gym



OpenAI Gym
https://gym.openai.com
Pieter
Environments Docs Chat
Sign in with GitHub
OpenAI Gym BETA
A toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Go.
Read the launch blog post ›
View documentation ›
View on GitHub ›
Episode 12167
joschu's trpo-gae-v0 on Walker2d-v0
Episode 27
jcoreyes's algorithm on Breakout-v0

Curiosity-driven Exploration

[Houthooft, Chen, Duan, Schulman, Turck, Abbeel, 2016]

$$r'(s_t, a_t, s_{t+1}) = r(s_t, a_t) + \eta D_{\mathrm{KL}}[p(\theta|\xi_t, a_t, s_{t+1}) \,\|\, p(\theta|\xi_t)]$$

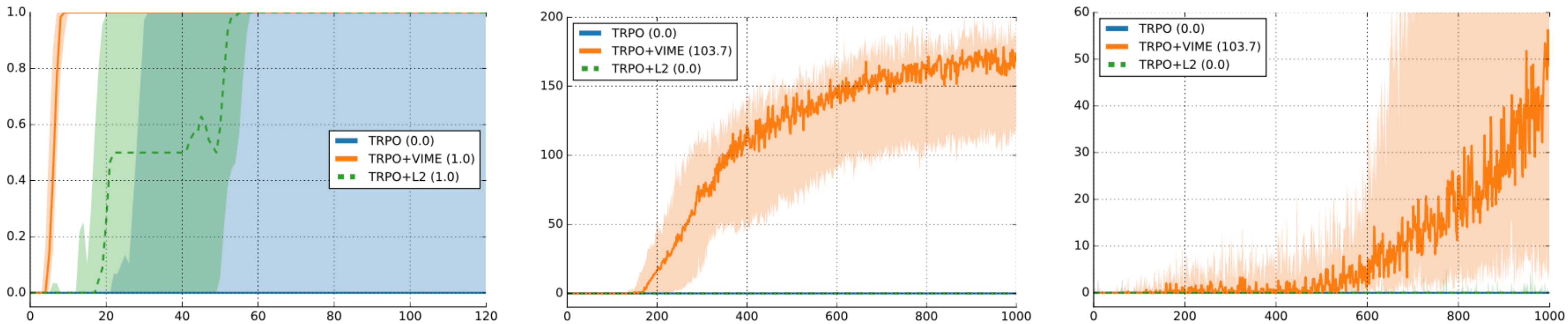
Building on:

Curiosity: Schmidhuber , 1991; Sun, Gomez, Schmidhuber, 2011; Schmidhuber, 2010

Bayesian neural nets: Blundell, Cornebise, Kavukcuoglu, Wierstra, 2015

Curiosity-driven Exploration

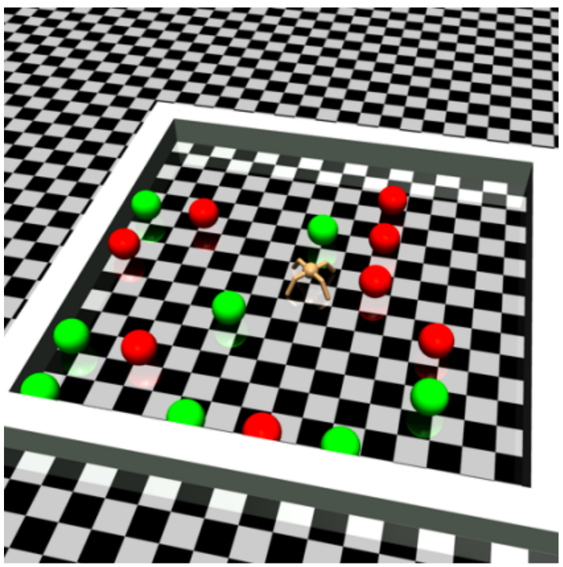
[Houthooft, Chen, Duan, Schulman, Turck, Abbeel, 2016]

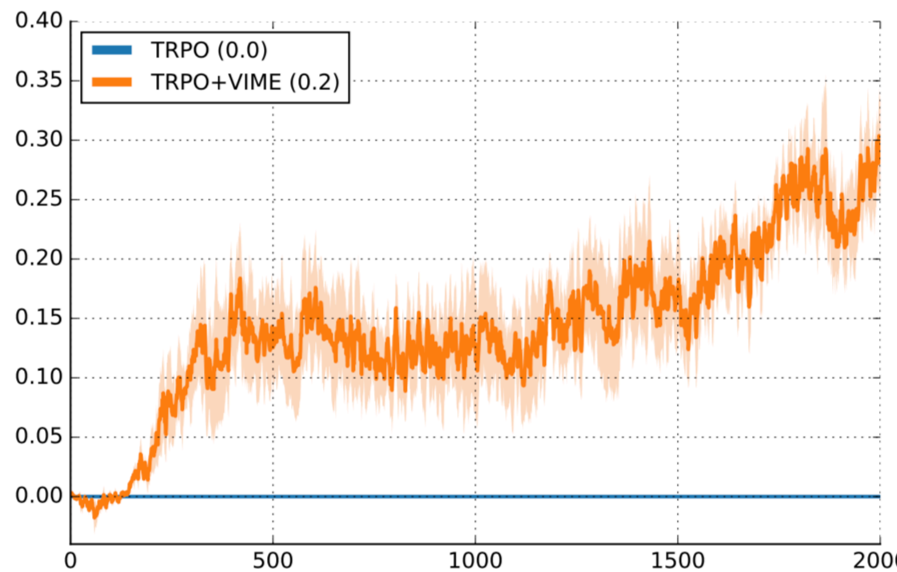


(a) MountainCar (b) CartPoleSwingup (c) HalfCheetah

Curiosity-driven Exploration

[Houthooft, Chen, Duan, Schulman, Turck, Abbeel, 2016]

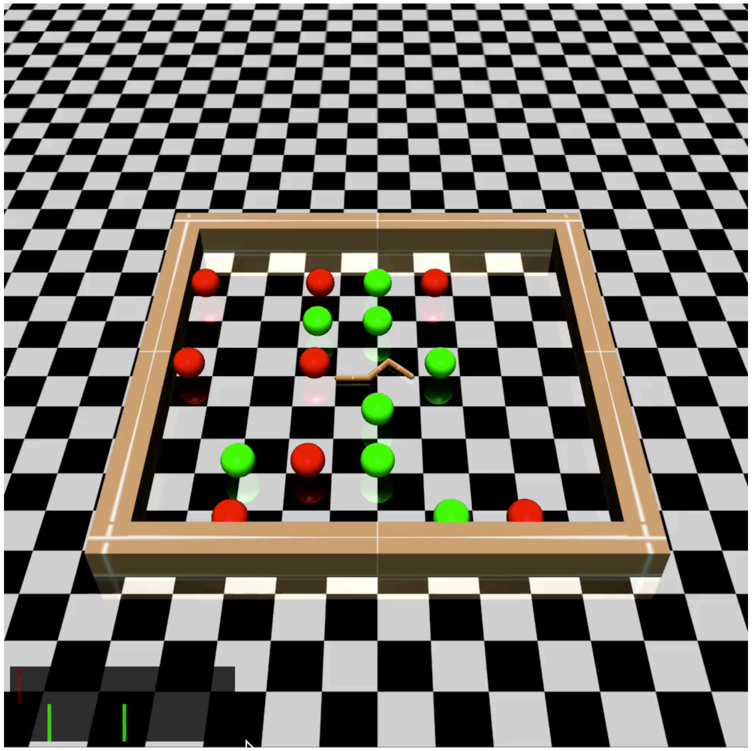




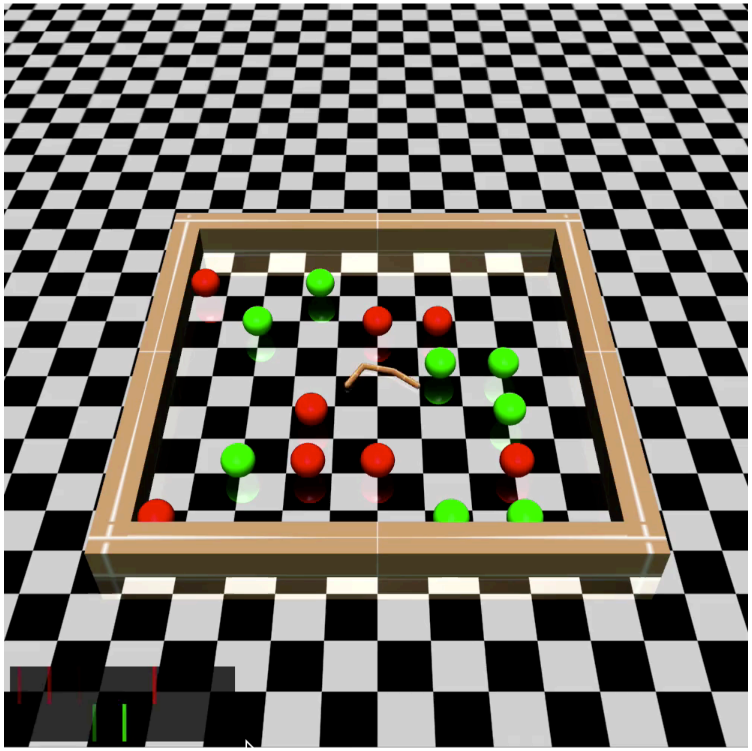
Swimmer + Food Collection

Curiosity-driven Exploration

[Houthooft, Chen, Duan, Schulman, Turck, Abbeel, 2016]



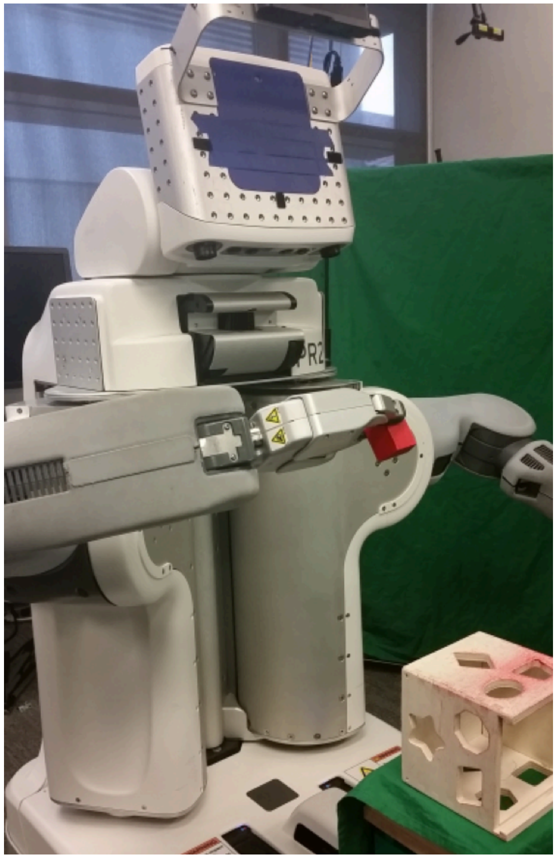
TRPO



TRPO + VIME

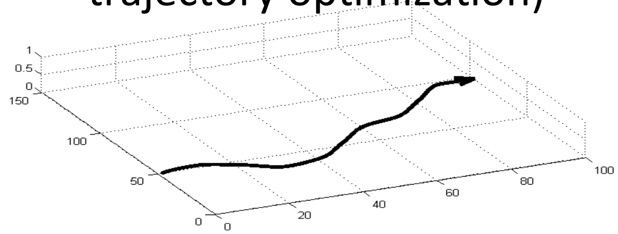
Swimmer + Food Collection

How About Real Robotic Visuo-Motor Skills?



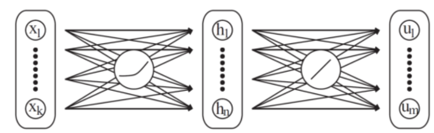
Guided Policy Search

Model-Based RL (through trajectory optimization)



Supervised learning

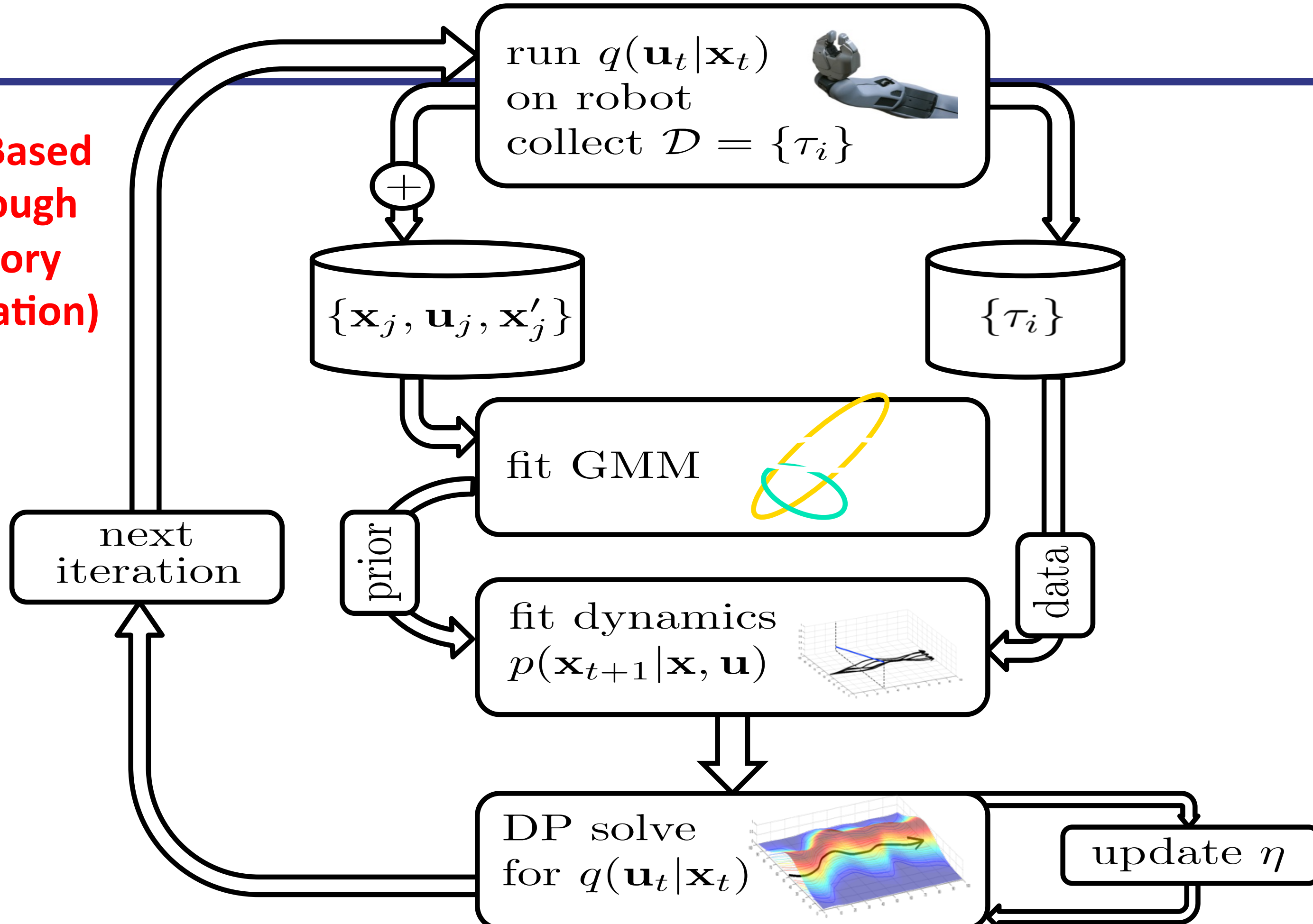
General Neural Net Policy



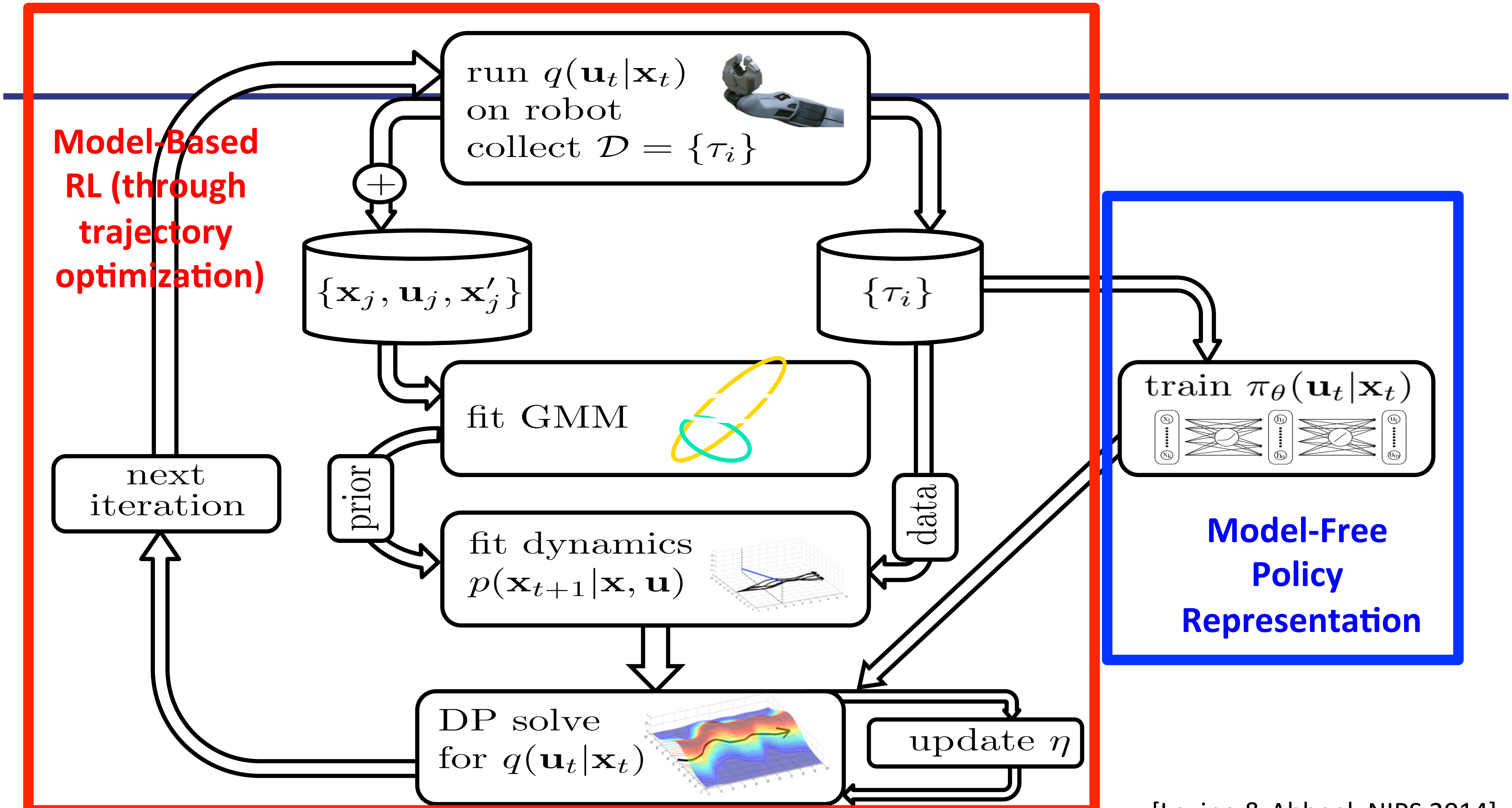
- Issue with two-phase pipeline
  - Representational mismatch trajectory distribution vs. neural net

→ Joint optimization $$\max_{\{\pi^{(i)}\},\theta} \sum_i \mathrm{E}[\sum_{t=0}^{H} R(s_t^{(i)}) \mid \pi^{(i)}] - \lambda \sum_i \|\pi^{(i)} - \pi_\theta\|$$

Model-Based RL (through trajectory optimization)

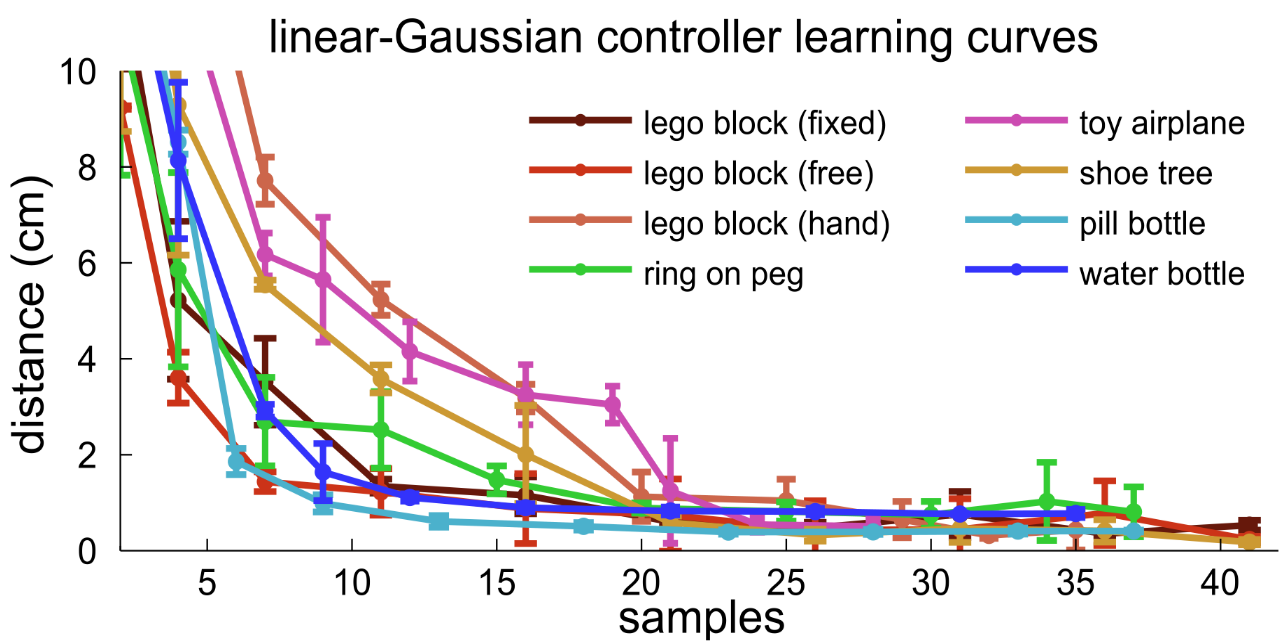


[Levine & Abbeel, NIPS 2014]



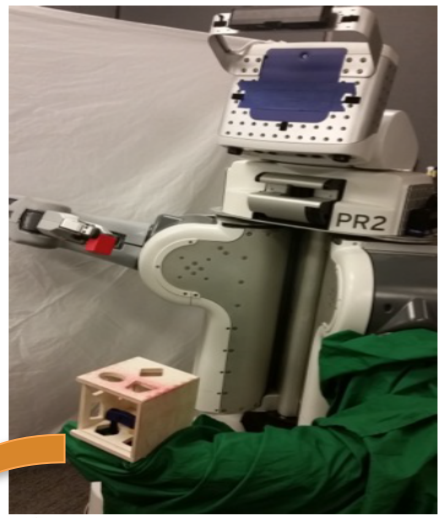
[Levine & Abbeel, NIPS 2014]

Linear-Gaussian Controller Learning Curves

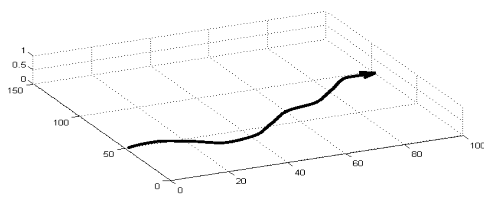


Instrumented Training

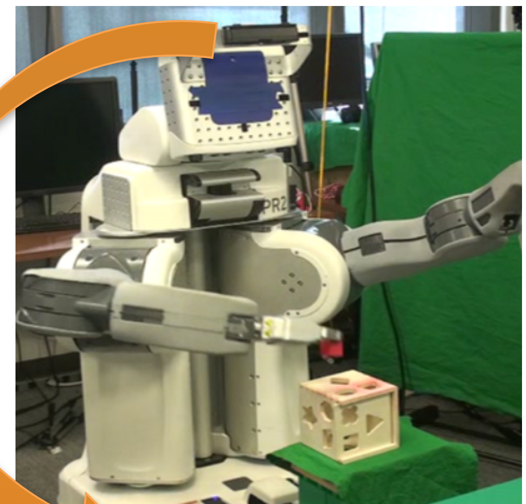
training time



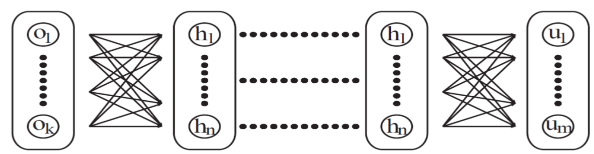
$$\mathbf{x}_t \longrightarrow \mathbf{u}_t$$



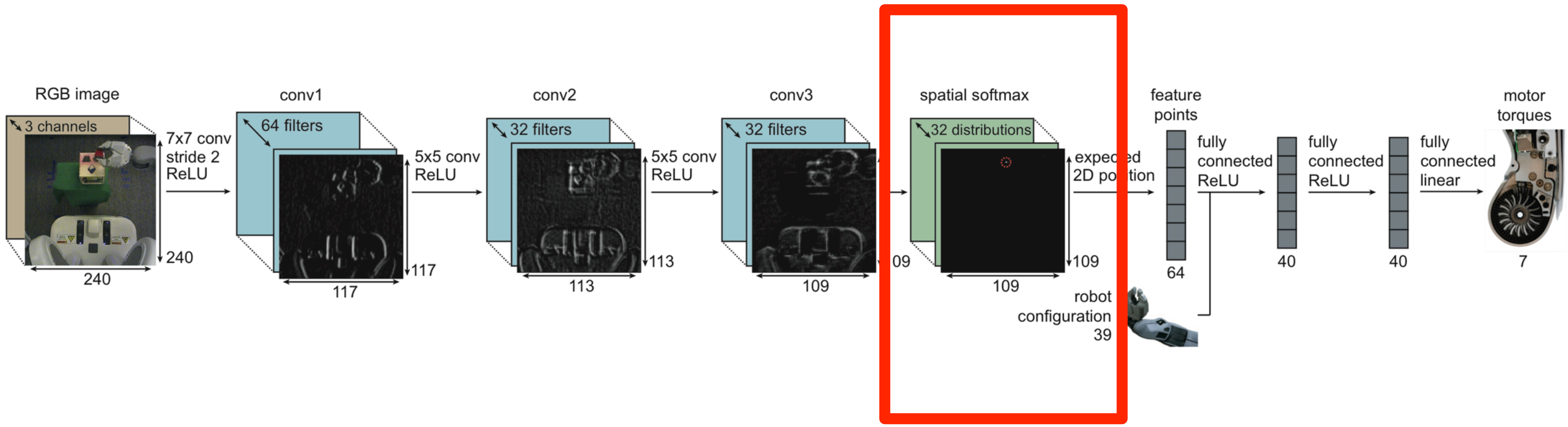
test time



$$\mathbf{o}_t \longrightarrow \mathbf{u}_t$$



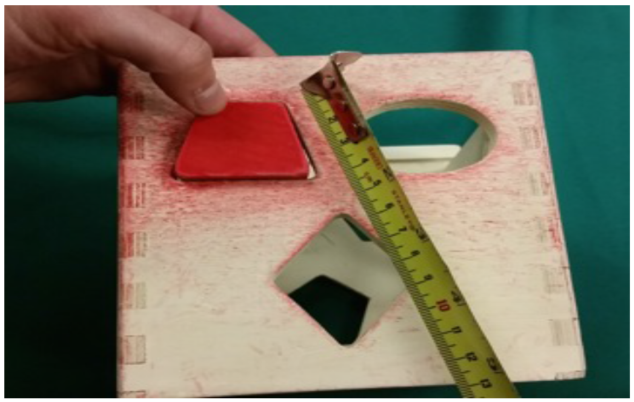
π_θ Deep Spatial Neural Net Architecture

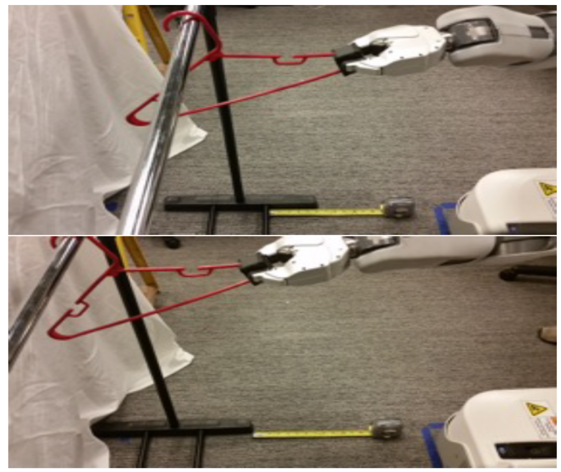


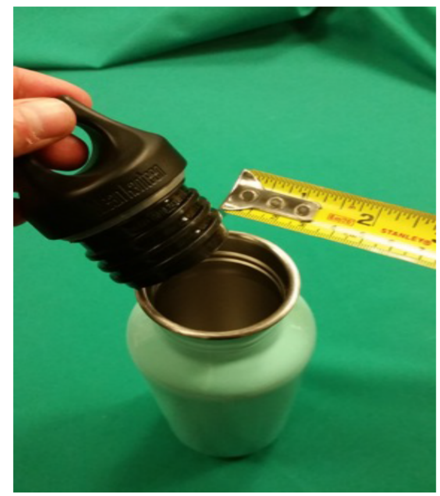
(92,000 parameters)

[Levine*, Finn*, Darrell, Abbeel, JMLR 2016]

Experimental Tasks





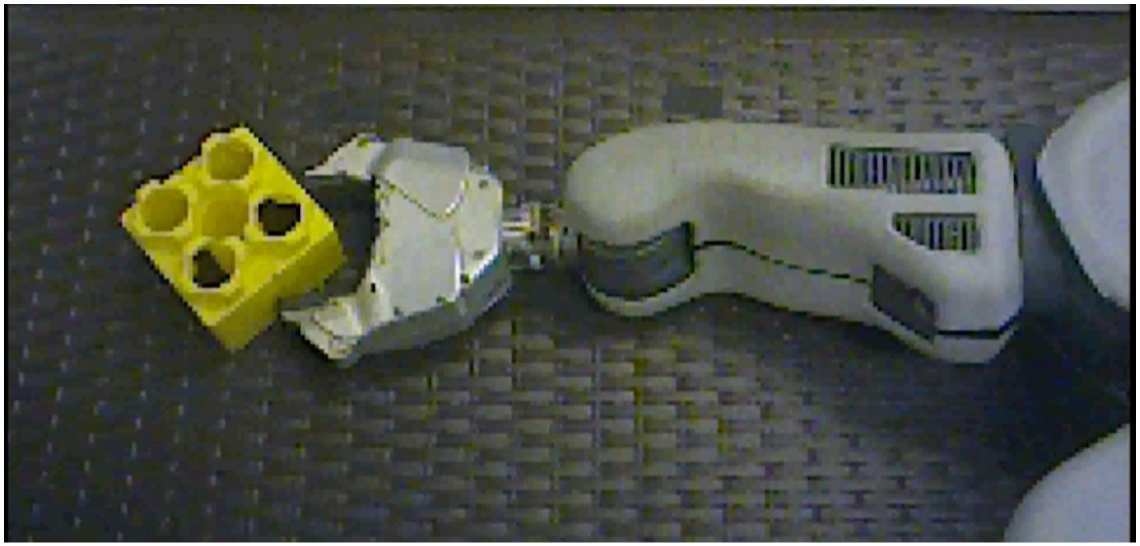


[Levine*, Finn*, Darrell, Abbeel, JMLR 2016]

Learning

[Levine*, Finn*, Darrell, Abbeel, JMLR 2016]

Visuomotor Learning Directly in Visual Space



[Finn, Tan, Duan, Darrell, Levine, Abbeel, ICRA 2016]

Related work: Embed to Control [Wattenberg, Springenberg, Boedecker, Riedmiller, 2015]

Visuomotor Learning Directly in Visual Space

[Finn, Tan, Duan, Darrell, Levine, Abbeel, 2015]

Visuomotor Cost Function Learning

- Learning from goal image can be great
- But:
  - Often other objects in environment --- don't actually expect to perfectly match example goal image
  - Goal image might not reveal much about how to get there

Visuomotor Cost Function Learning

→ infer cost function from demonstrations

Challenges:

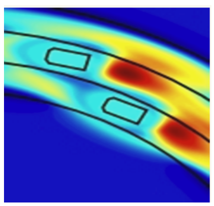
- underdefined problem
- difficult to evaluate learned cost
- large perceptual input spaces

Prior Approaches

repeatedly solve MDP

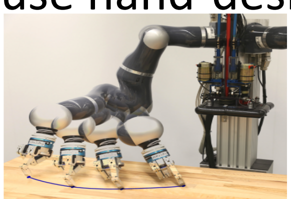
Abbeel & Ng ’04
Ziebart et al. ’08
Ratliff et al. ‘09

use known dynamics



Todorov ’06
Levine et al. ‘12
Dragan et al. ’12

use hand-designed features



Boularias et al. ‘11
Kalakrishnan et al. ’13
Doerr et al. ‘15

Desiderata

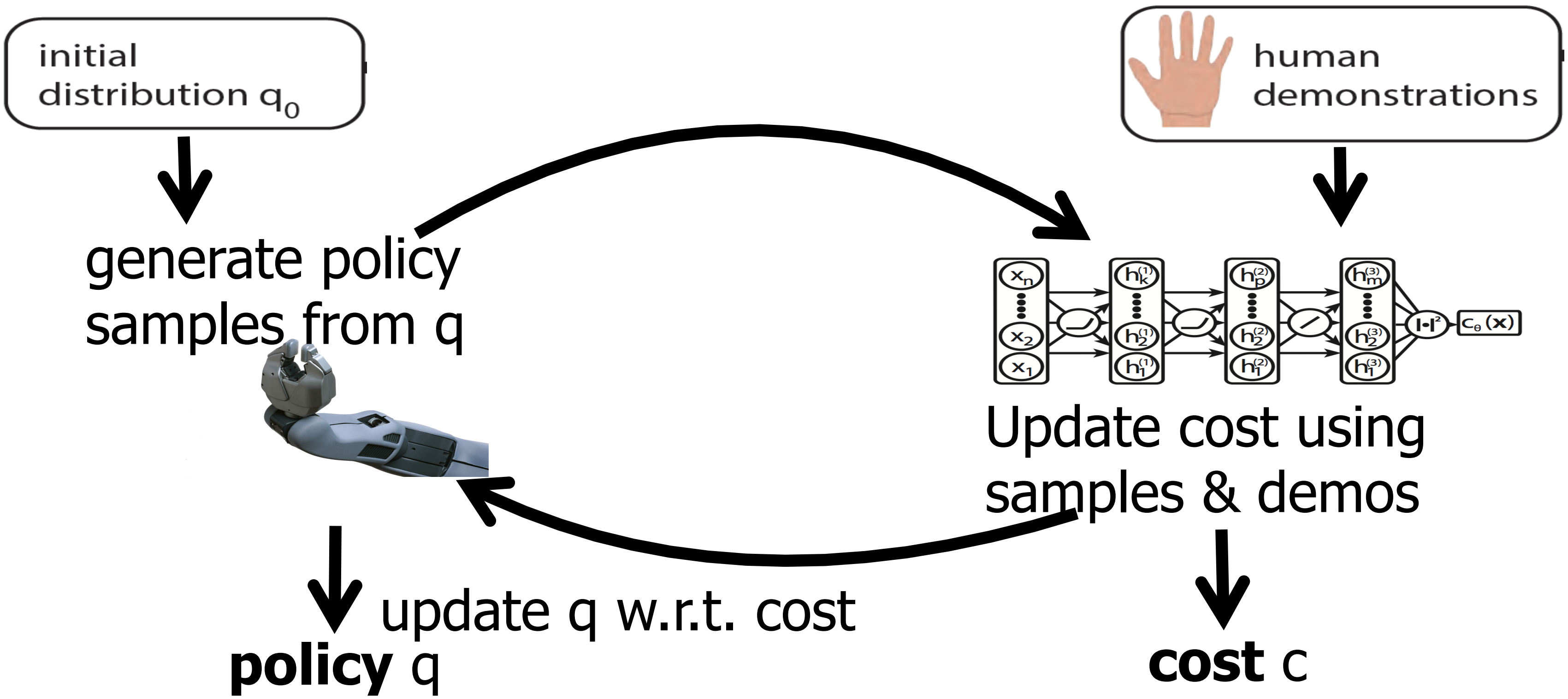
avoid (repeatedly) solving the MDP

handle unknown dynamics

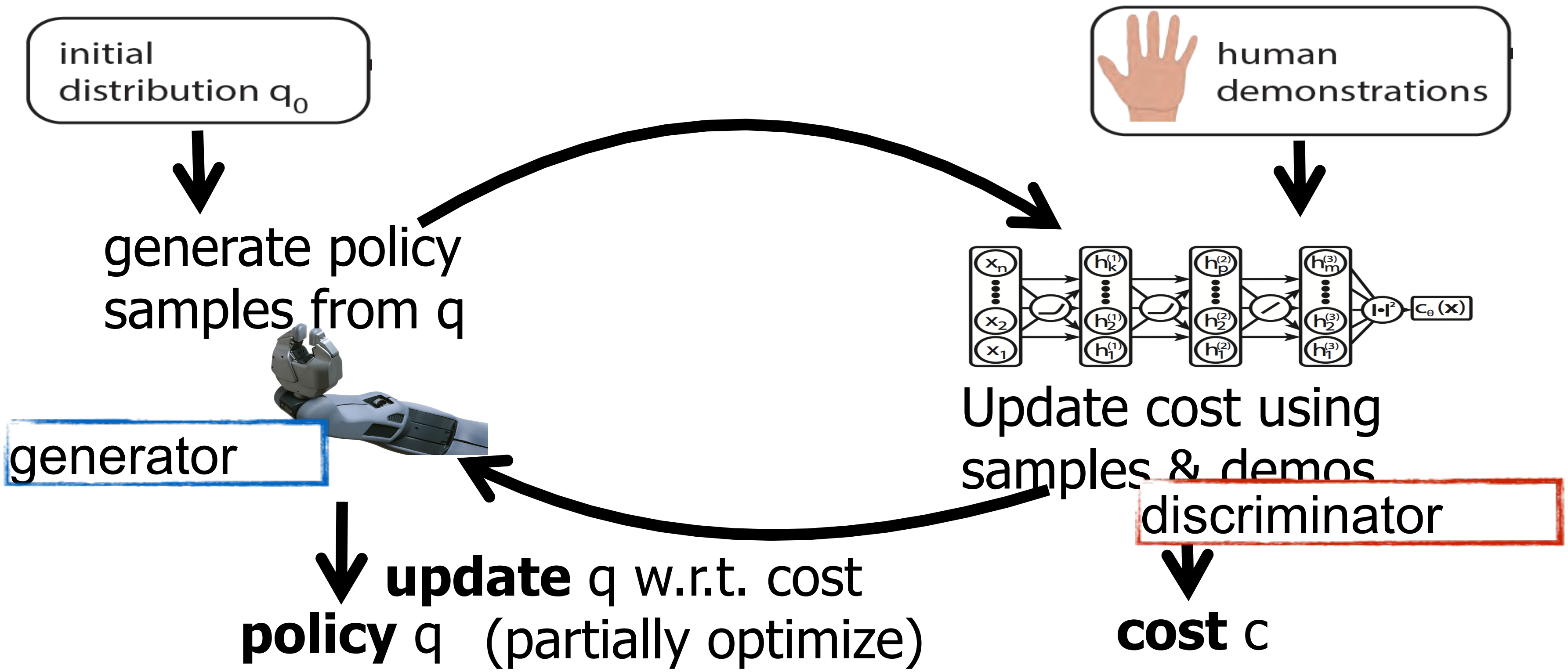
learn features with flexible, nonlinear cost parametrization

\+ sample efficiency

Guided Cost Learning

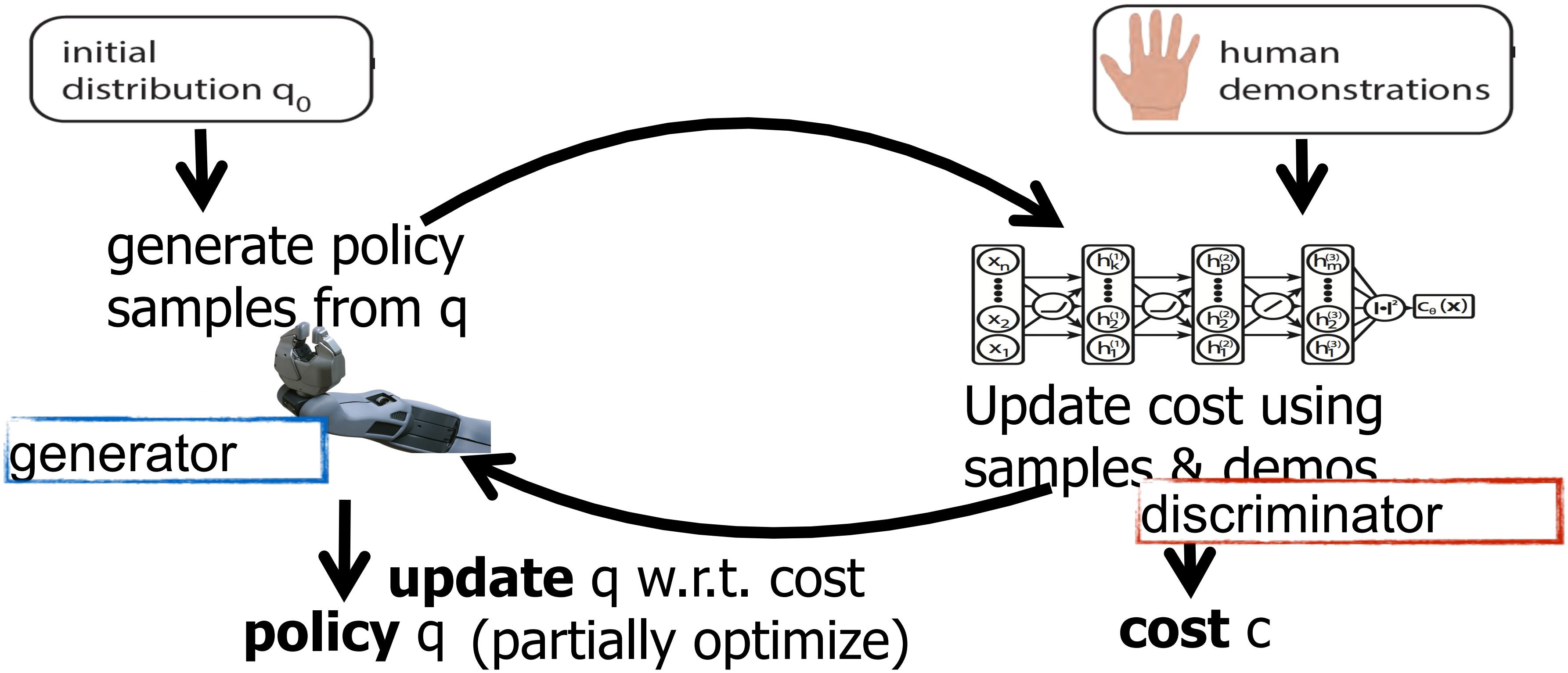


Guided Cost Learning



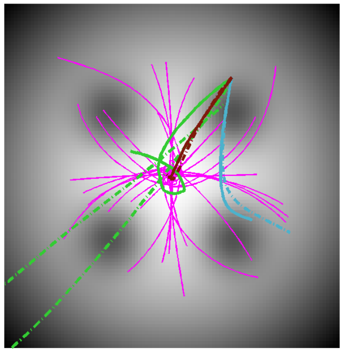
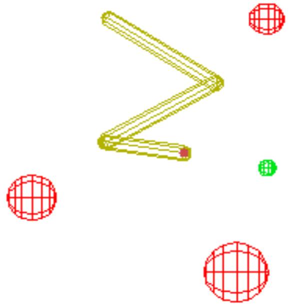
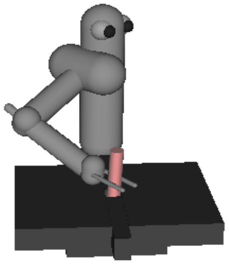
update cost in inner loop of policy optimization

Guided Cost Learning



Ho et al., ICML '16, arXiv '16
Kim & Bengio, arXiv '16

Experiments

2D navigation	2D reaching	peg insertion
		

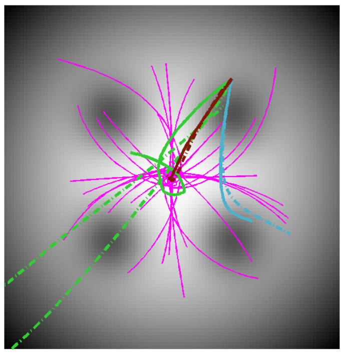
high-dimensional continuous states & actions

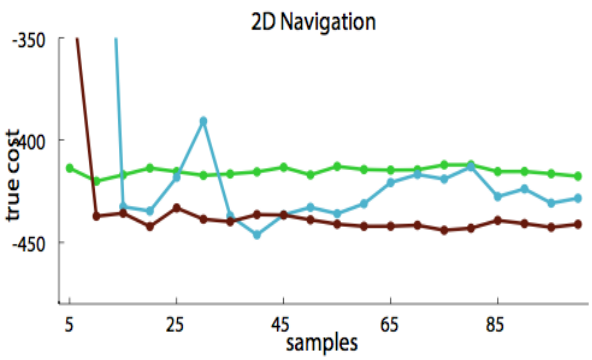
direct torque control

complex contact dynamics

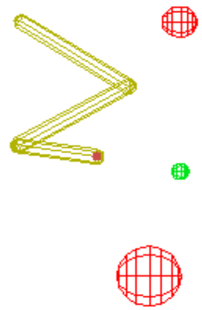
Experiments

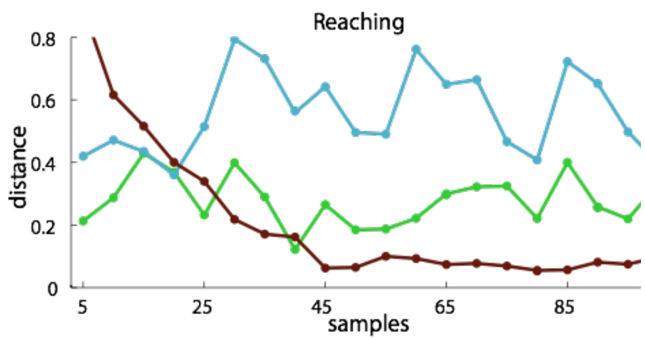
2D navigation



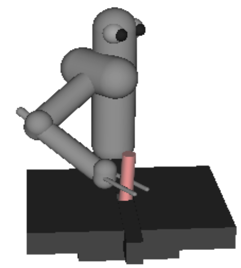


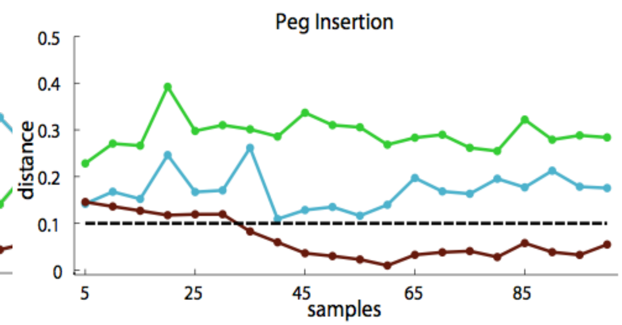
2D reaching





peg insertion

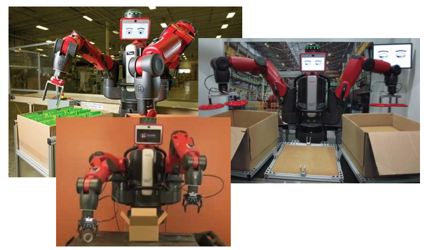


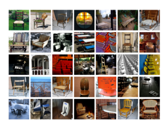


Frontiers

- Shared and transfer learning

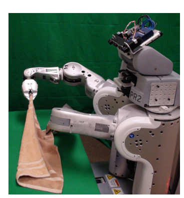






- Memory
  - Estimation
  - Temporal hierarchy / goal setting

- Applications




Thank you