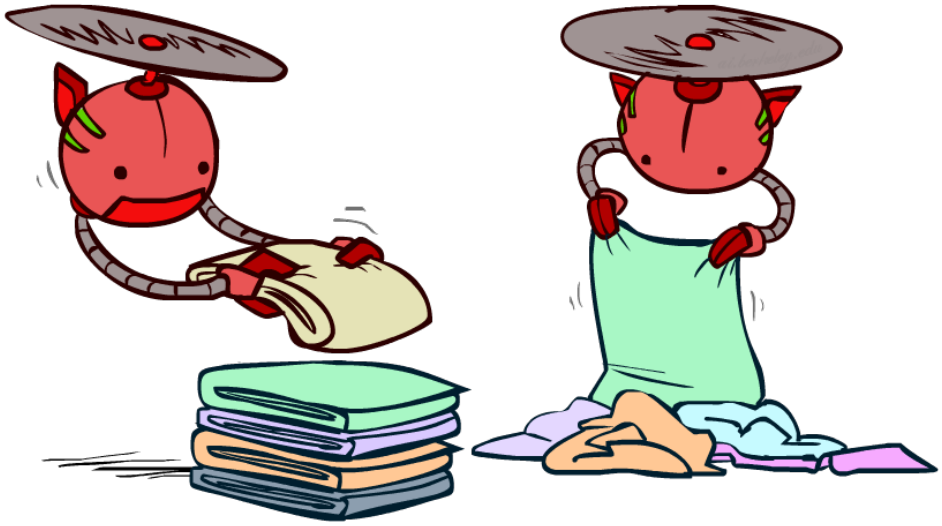
CS 188: Artificial Intelligence

Robotics *



Instructors: Pieter Abbeel & Anca Dragan --- University of California, Berkeley

PR1

[Wyrobek, Berger, Van der Loos, Salisbury, ICRA 2008]

Personal Robotics Hardware

PR2
Willow Garage
$400,000
2009

Baxter
Rethink Robotics
$30,000
2013

Fetch
Fetch Robotics
~$80,000
2015

?

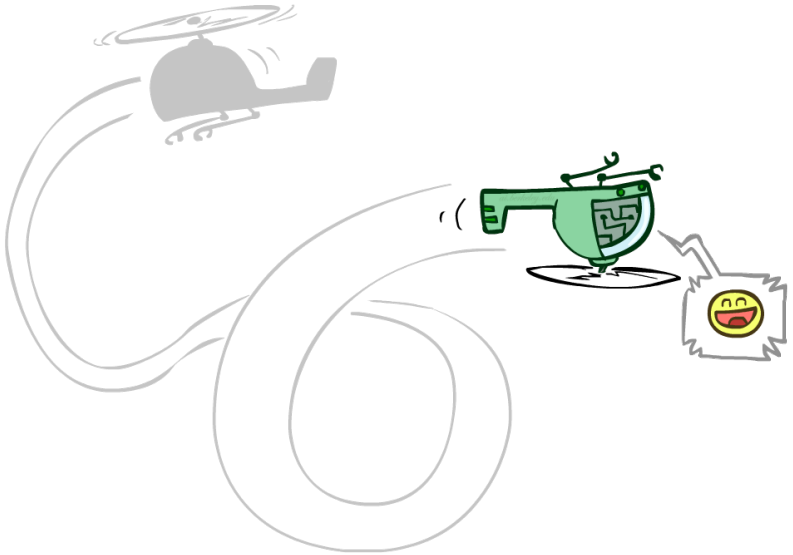
Robot Learning

- Apprenticeship learning
- Reinforcement learning

Robot Learning

- ***Apprenticeship learning***
- Reinforcement learning

Robotic Helicopters



Motivating Example

- How do we execute a task like this?

[VIDEO: tictoc_results.wmv]

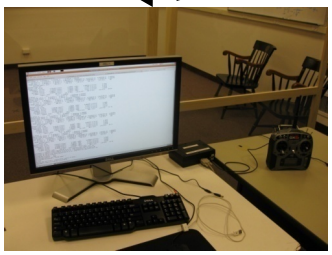
Autonomous Helicopter Flight

- Key challenges:
  - Track helicopter position and orientation during flight
  - Decide on control inputs to send to helicopter

Autonomous Helicopter Setup

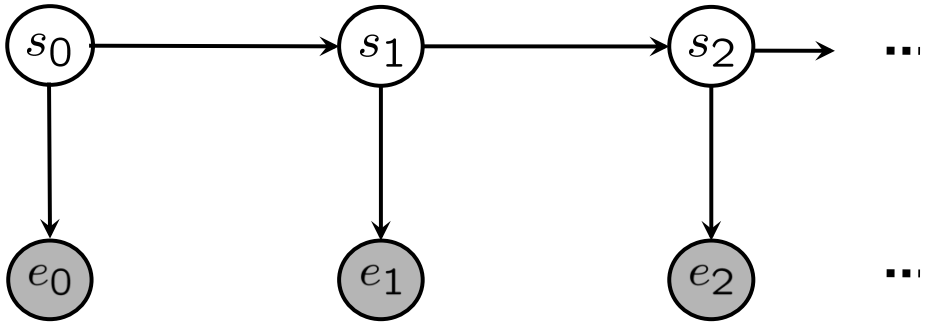
Position

On-board inertial measurement unit (IMU)



Send out controls to helicopter

HMM for Tracking the Helicopter



- State: $s = (x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi})$
- Measurements: [observation update]
  - 3-D coordinates from vision, 3-axis magnetometer, 3-axis gyro, 3-axis accelerometer
- Transitions (dynamics): [time elapse update]
  - $s_{t+1} = f(s_t, a_t) + w_t$ f: encodes helicopter dynamics, w: noise

Helicopter MDP

- State: $s = (x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi})$

- Actions (control inputs):
  - a_{lon} : Main rotor longitudinal cyclic pitch control (affects pitch rate)
  - a_{lat} : Main rotor latitudinal cyclic pitch control (affects roll rate)
  - a_{coll} : Main rotor collective pitch (affects main rotor thrust)
  - a_{rud} : Tail rotor collective pitch (affects tail rotor thrust)

- Transitions (dynamics):
  - $s_{t+1} = f(s_t, a_t) + w_t$

  [f encodes helicopter dynamics]

  [w is a probabilistic noise model]

- Can we solve the MDP yet?

Problem: What's the Reward?

- Reward for hovering:

$$
\begin{aligned}
R(s) \quad = \quad & -\alpha_x (x - x^*)^2 \\
& -\alpha_y (y - y^*)^2 \\
& -\alpha_z (z - z^*)^2 \\
& -\alpha_{\dot{x}} \dot{x}^2 \\
& -\alpha_{\dot{y}} \dot{y}^2 \\
& -\alpha_{\dot{z}} \dot{z}^2
\end{aligned}
$$

[VIDEO: ihover-newencoding09.wmv]

Hover

[Ng et al, 2004]

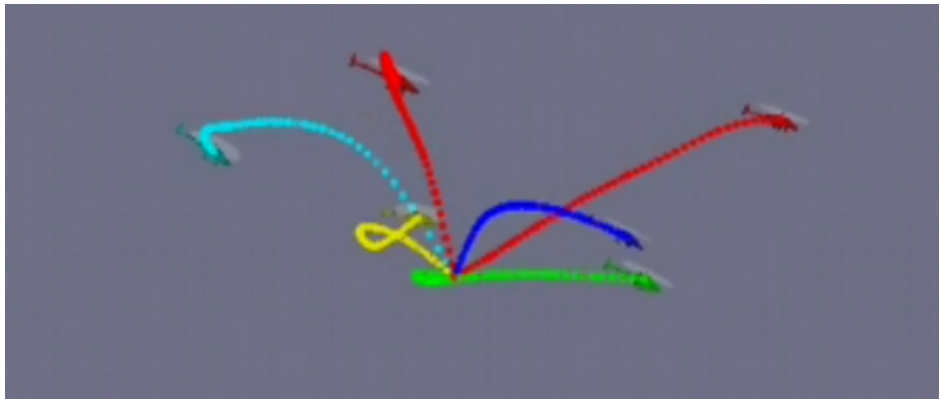
Problem: What's the Reward?

- Rewards for "Flip"?
  - Problem: what's the target trajectory?
  - Just write it down by hand?

[VIDEO: 20061204---bad.wmv]

Flips (?)

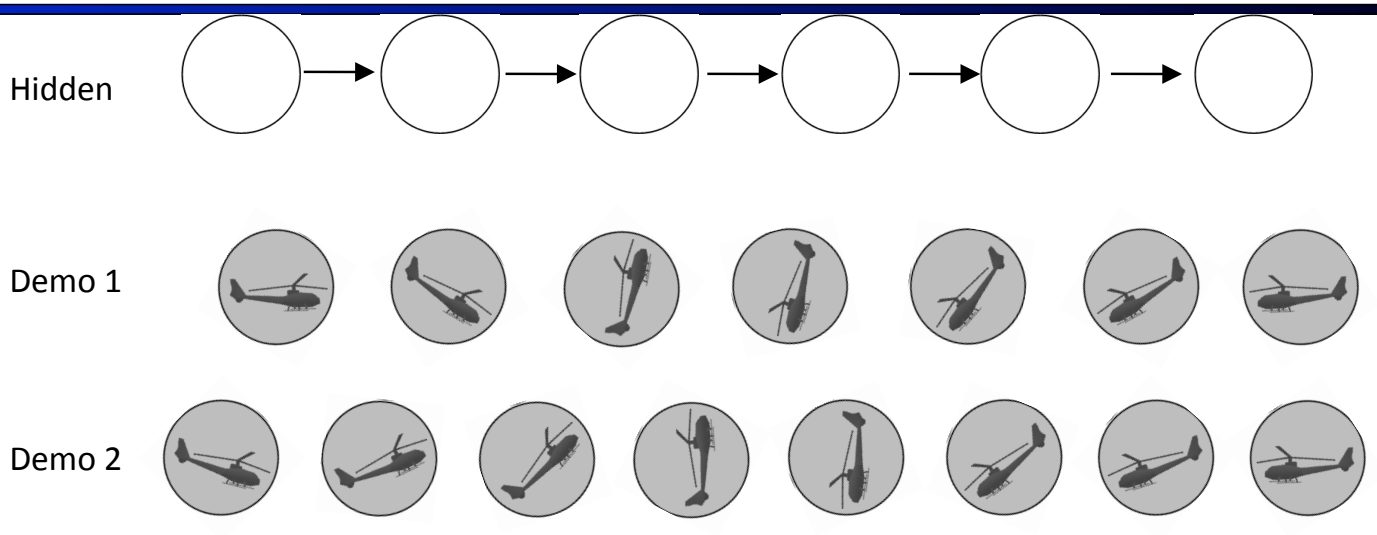
Helicopter Apprenticeship?



[VIDEO: airshow_unaligned.wmv]

Demonstrations

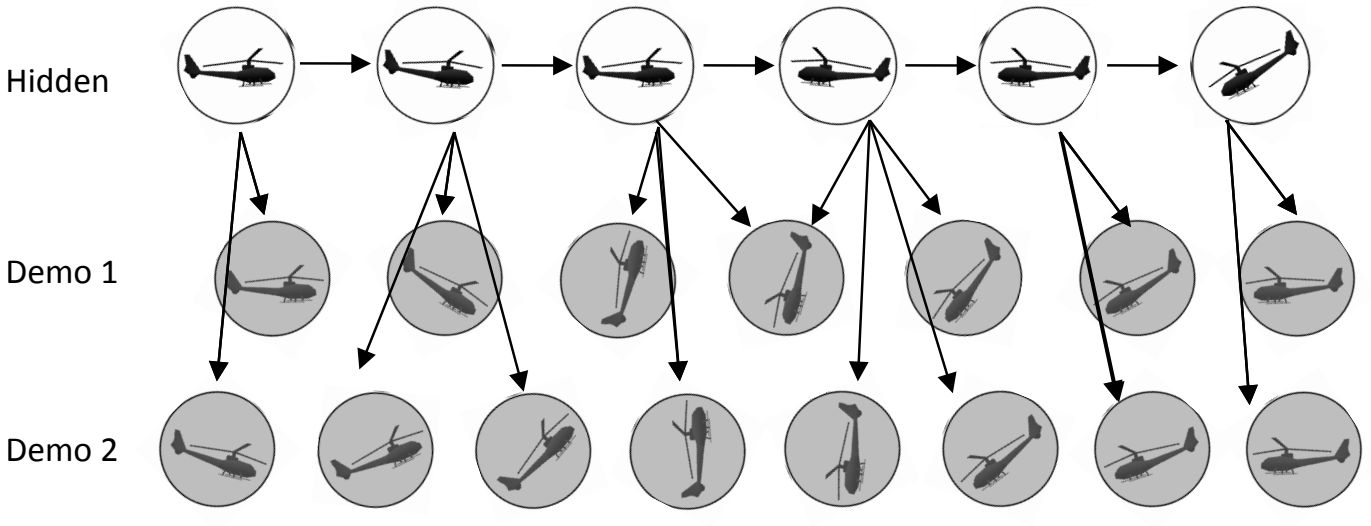
Learning a Trajectory



- HMM-like generative model
  - Dynamics model used as HMM transition model
  - Demos are observations of hidden trajectory
- Problem: how do we align observations to hidden trajectory?

Abbeel, Coates, Ng, IJRR 2010

Probabilistic Alignment using a Bayes' Net



- Dynamic Time Warping

  (Needleman&Wunsch 1970, Sakoe&Chiba, 1978)

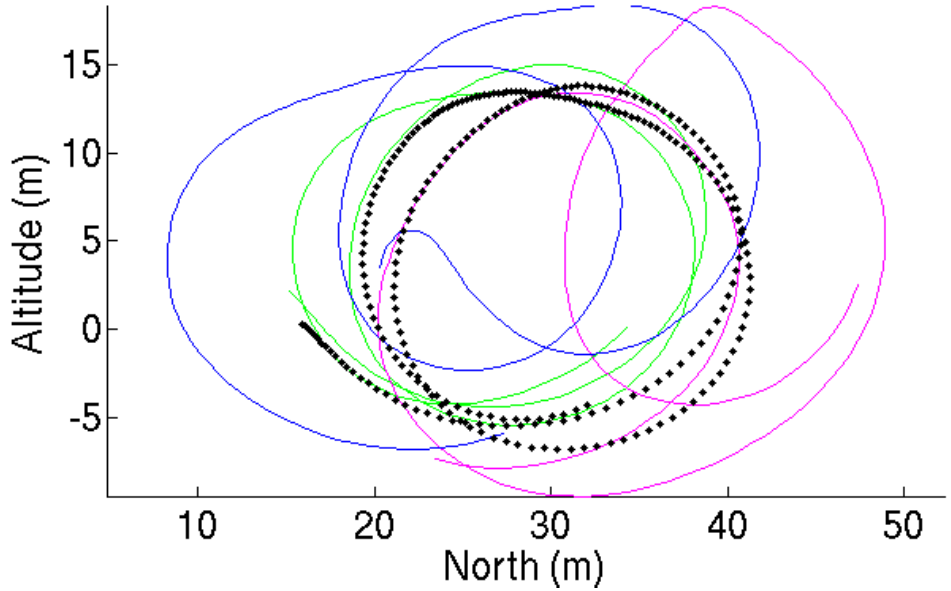
- Extended Kalman filter / smoother

Abbeel, Coates, Ng, IJRR 2010

[VIDEO: airshow_unaligned.wmv]

Aligned Demonstrations

Alignment of Samples



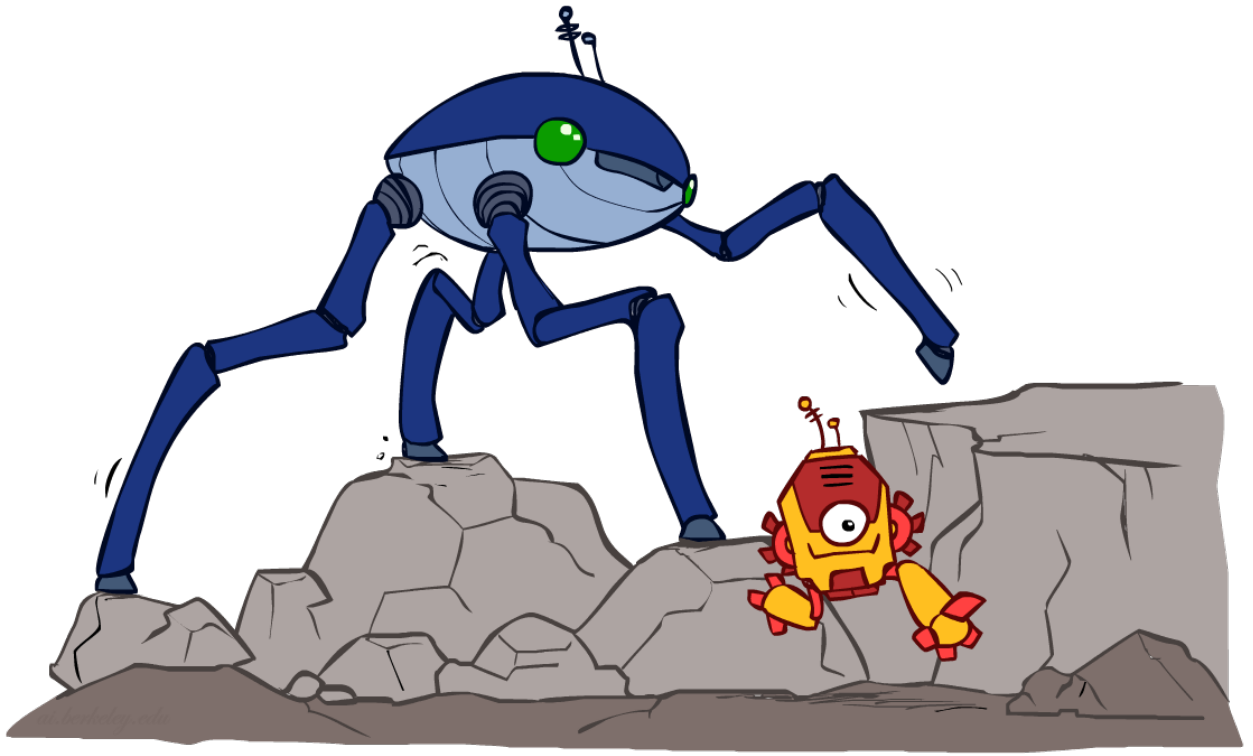
- Result: inferred sequence is much cleaner!

[VIDEO: airshow_trimmed.wmv]

Final Behavior

[Abbeel, Coates, Quigley, Ng, 2010]

Legged Locomotion



Quadruped

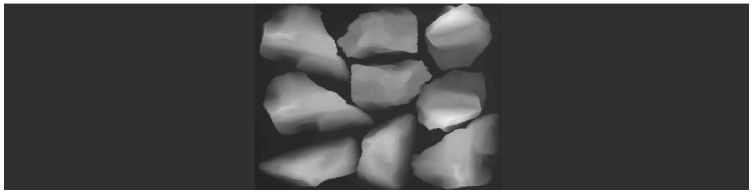
- Low-level control problem: moving a foot into a new location
  → search with successor function ~ moving the motors
- High-level control problem: where should we place the feet?
  - Reward function R(x) = w . f(s) [25 features]

[Kolter, Abbeel & Ng, 2008]

Experimental setup

- Demonstrate path across the "training terrain"

- Run apprenticeship to learn the reward function
- Receive "testing terrain"---height map.



- Find the optimal policy with respect to the *learned reward function* for crossing the testing terrain.

[Kolter, Abbeel & Ng, 2008]

[VIDEO: quad initial.wmv]

Without learning

[VIDEO: quad initial.wmv]

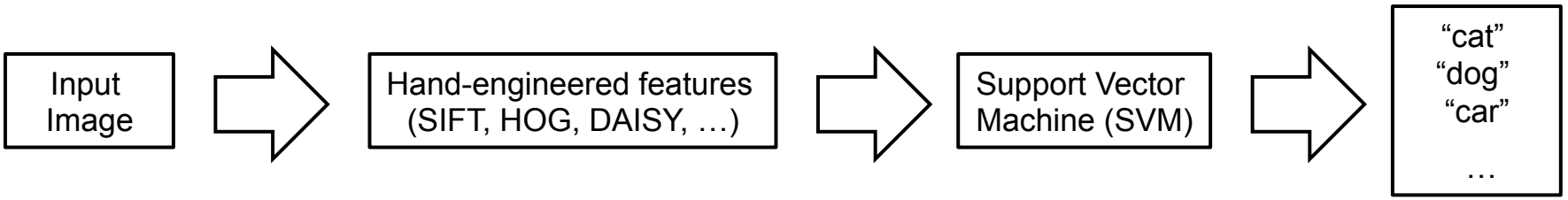
With learned reward function

Robot Learning

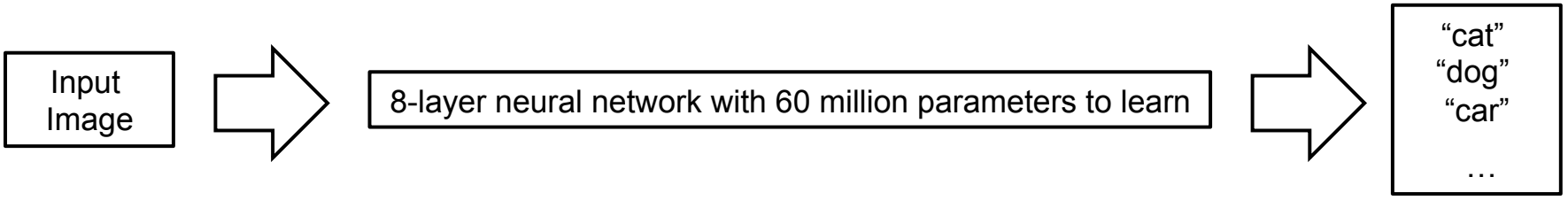
- Apprenticeship learning
- ***Reinforcement learning***

Object Detection in Computer Vision

- State-of-the-art object detection until 2012:

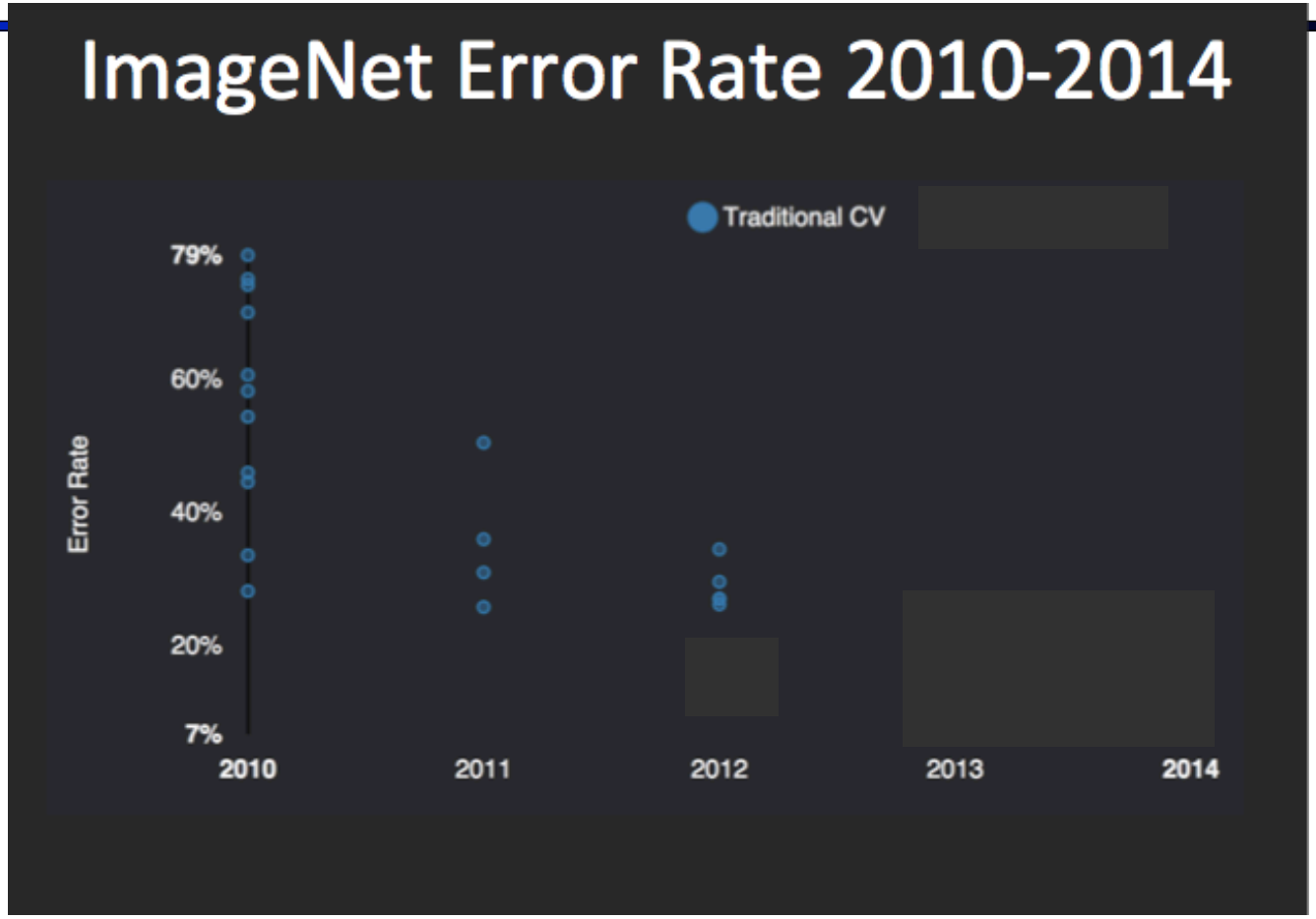


- Deep Supervised Learning (Krizhevsky, Sutskever, Hinton 2012; also LeCun, Bengio, Ng, Darrell, …):



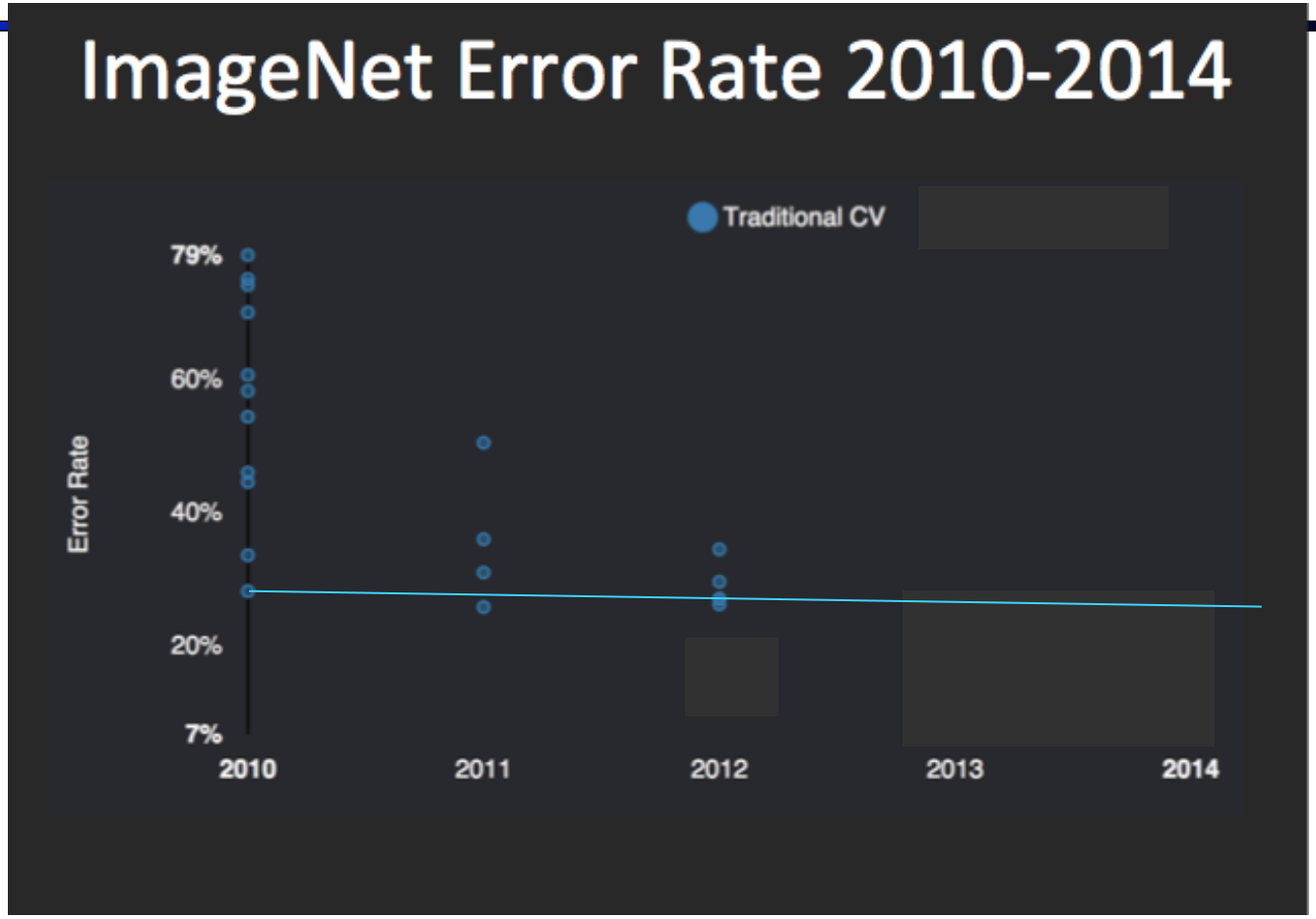
  - ~1.2 million training images from ImageNet [Deng, Dong, Socher, Li, Li, Fei-Fei, 2009]

Performance



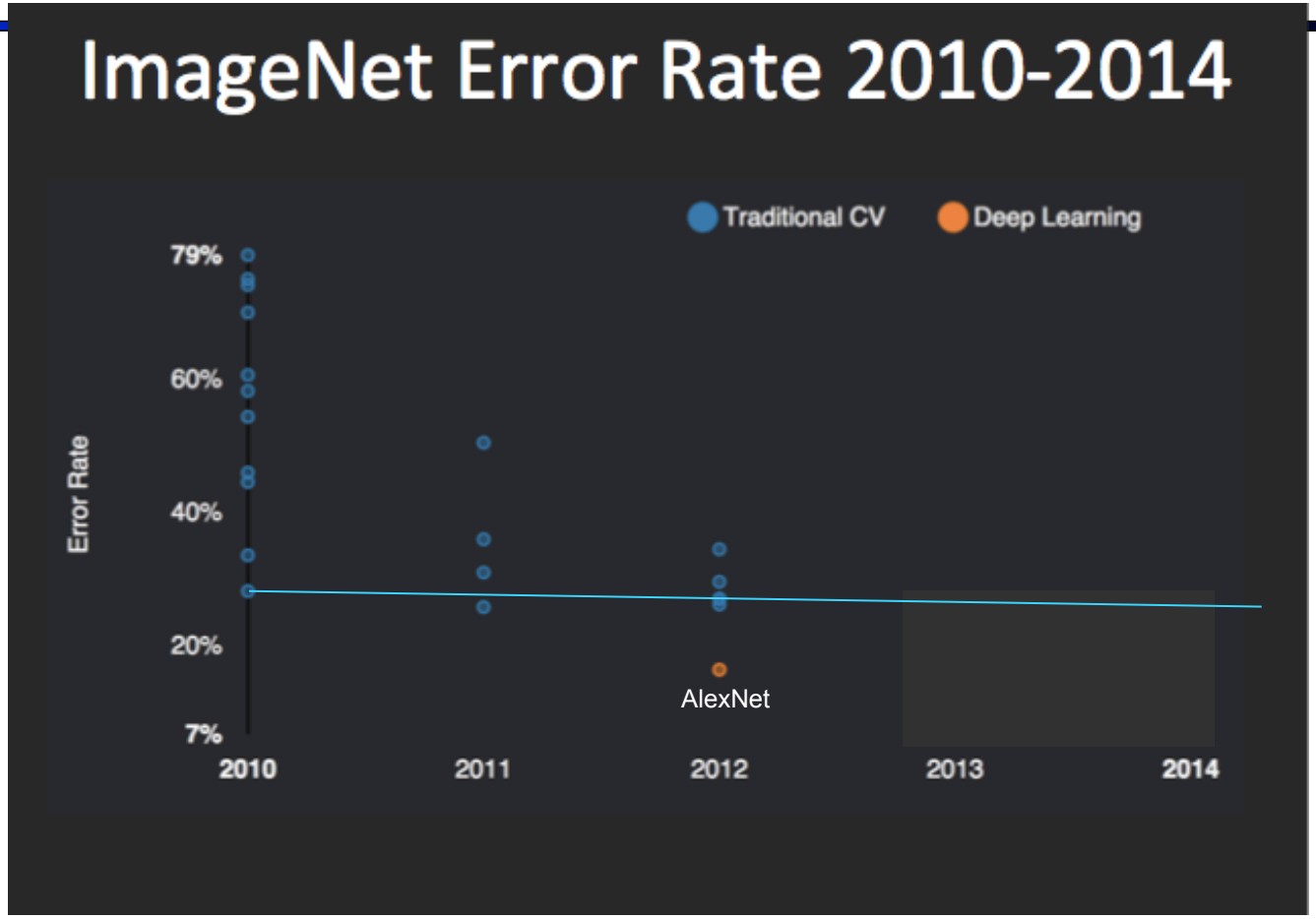
graph credit Matt Zeiler, Clarifai

Performance



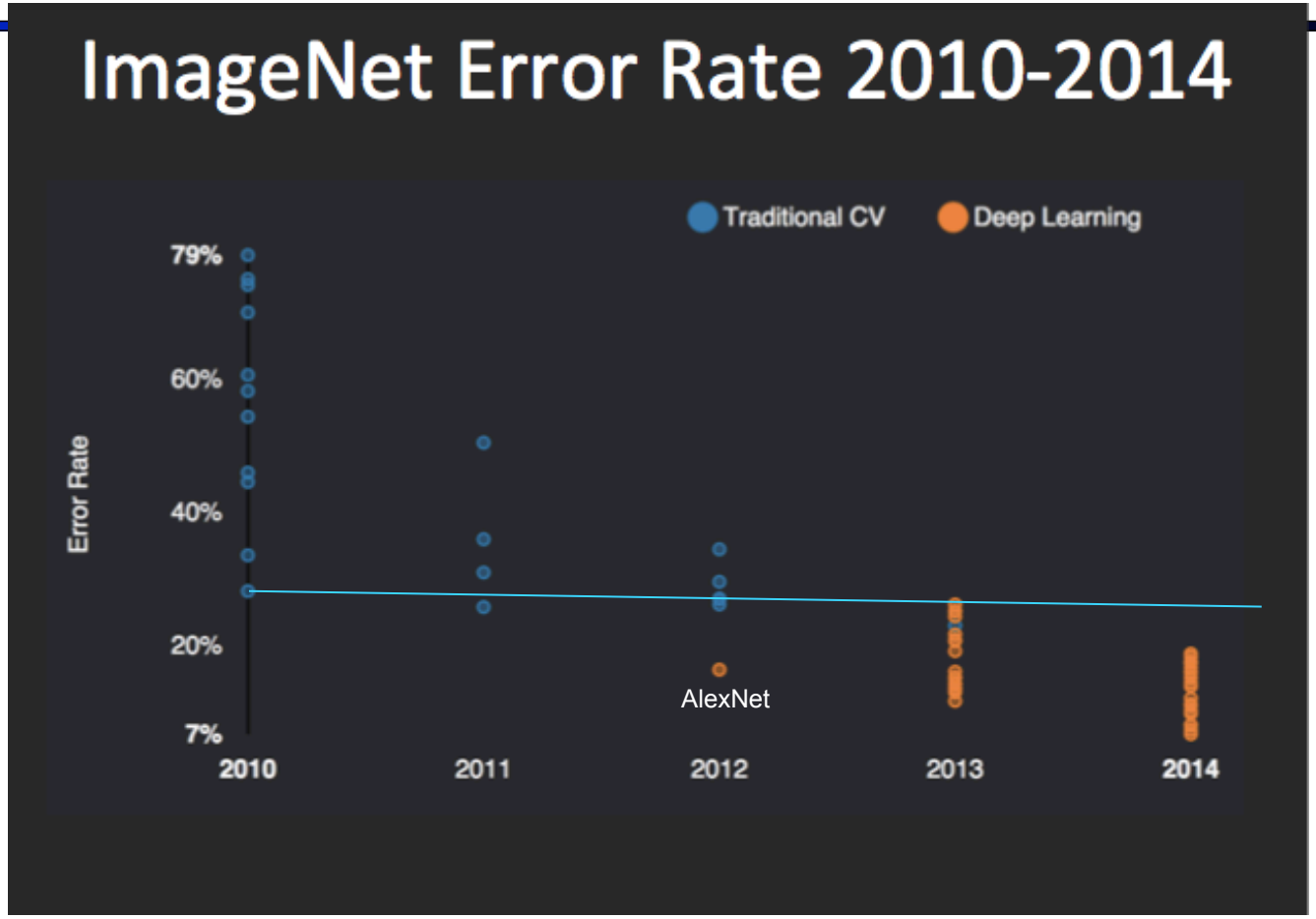
graph credit Matt Zeiler, Clarifai

Performance



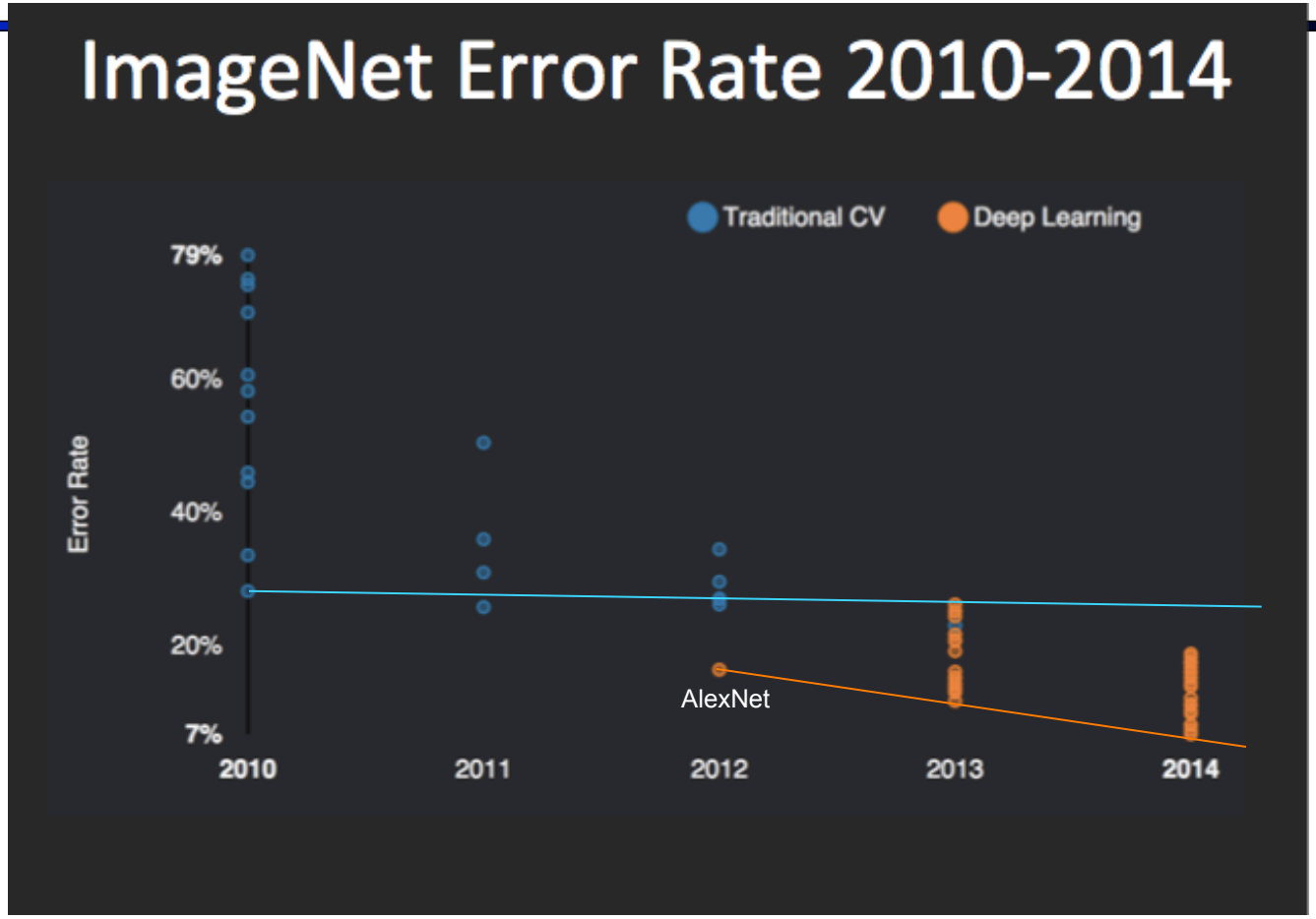
graph credit Matt Zeiler, Clarifai

Performance



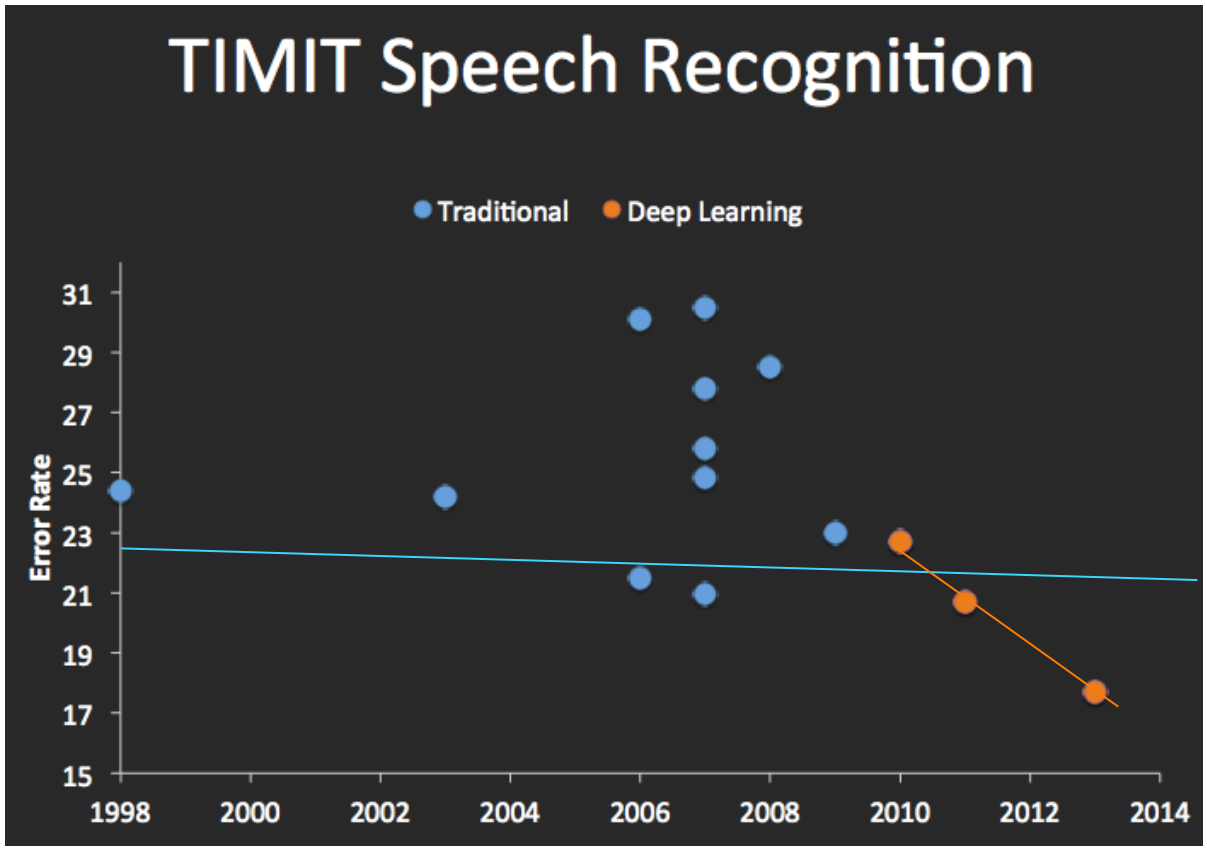
graph credit Matt Zeiler, Clarifai

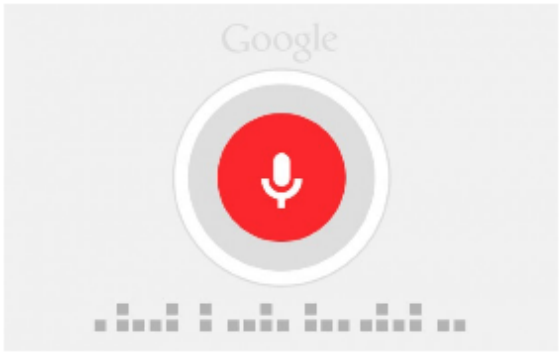
Performance



graph credit Matt Zeiler, Clarifai

Speech Recognition





graph credit Matt Zeiler, Clarifai

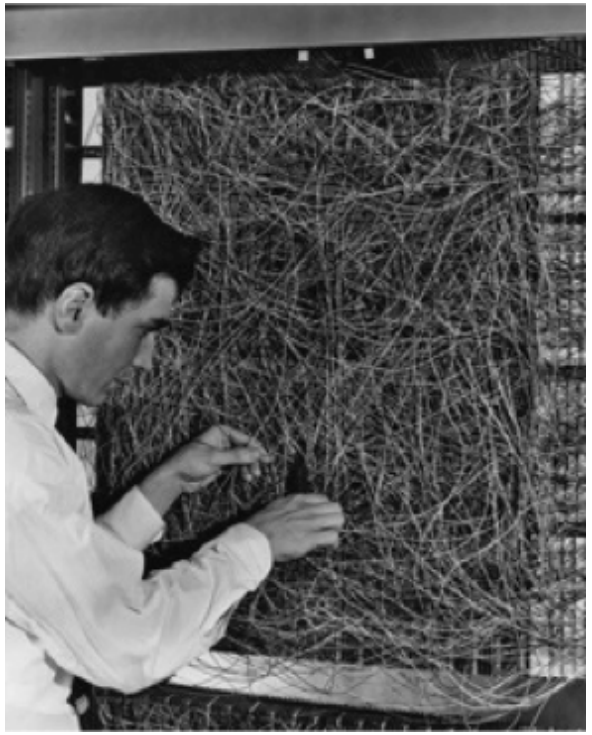
History

1958 Rosenblatt proposed perceptrons
1980 Neocognitron (Fukushima, 1980)
1982 Hopfield network, SOM (Kohonen, 1982), Neural PCA (Oja, 1982)
1985 Boltzmann machines (Ackley et al., 1985)
1986 Multilayer perceptrons and backpropagation (Rumelhart et al., 1986)
1988 RBF networks (Broomhead&Lowe, 1988)
1989 Autoencoders (Baldi&Hornik, 1989), Convolutional network (LeCun, 1989)
1992 Sigmoid belief network (Neal, 1992)
1993 Sparse coding (Field, 1993) (Olshausen, 1996)

2000s Sparse, Probabilistic, and Energy models (Hinton, Bengio, LeCun, Ng)

Is deep learning 3, 30, or 60 years old?

based on history by K. Cho



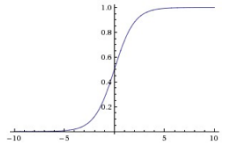
Rosenblatt's Perceptron

What's Changed

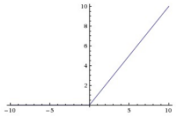
- Data
  - 1.2M training examples
  - 2048 (different crops)
  - 90 (PCA re-colorings)
- Compute power
  - Two NVIDIA GTX 580 GPUs
  - 5-6 days of training time

- Nonlinearity

  Sigmoid

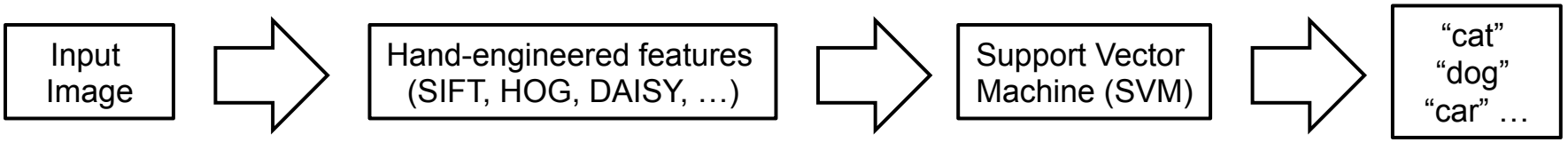
  → ReLU

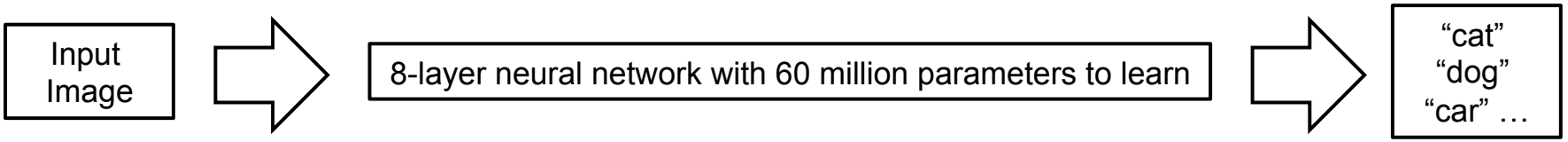
- Regularization
  - Drop-out
- Exploration of model structure
- Optimization know-how

Object Detection in Computer Vision

- State-of-the-art object detection until 2012:



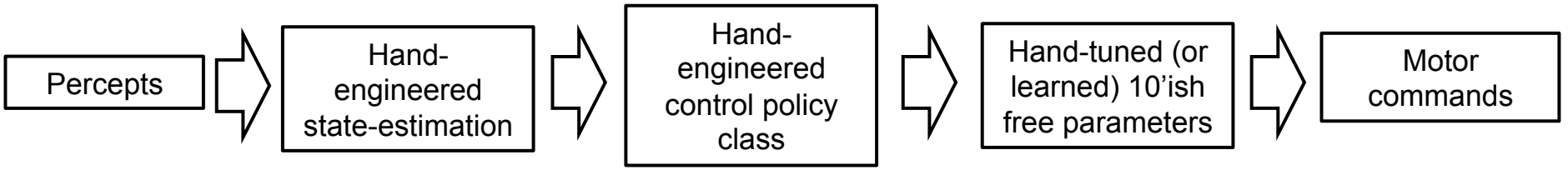
- Deep Supervised Learning (Krizhevsky, Sutskever, Hinton 2012; also LeCun, Bengio, Ng, Darrell, …):



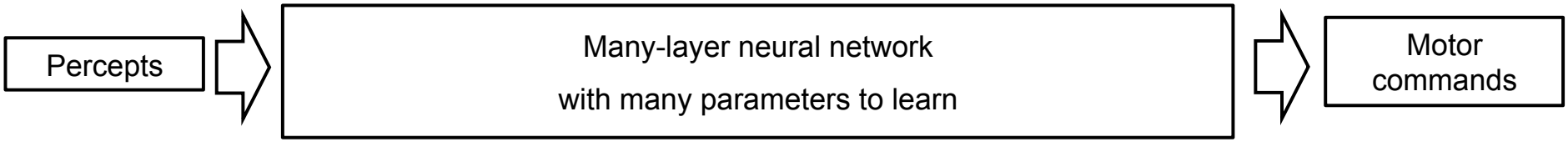
  - ~1.2 million training images from ImageNet [Deng, Dong, Socher, Li, Li, Fei-Fei, 2009]

Robotics

- **Current state-of-the-art robotics**

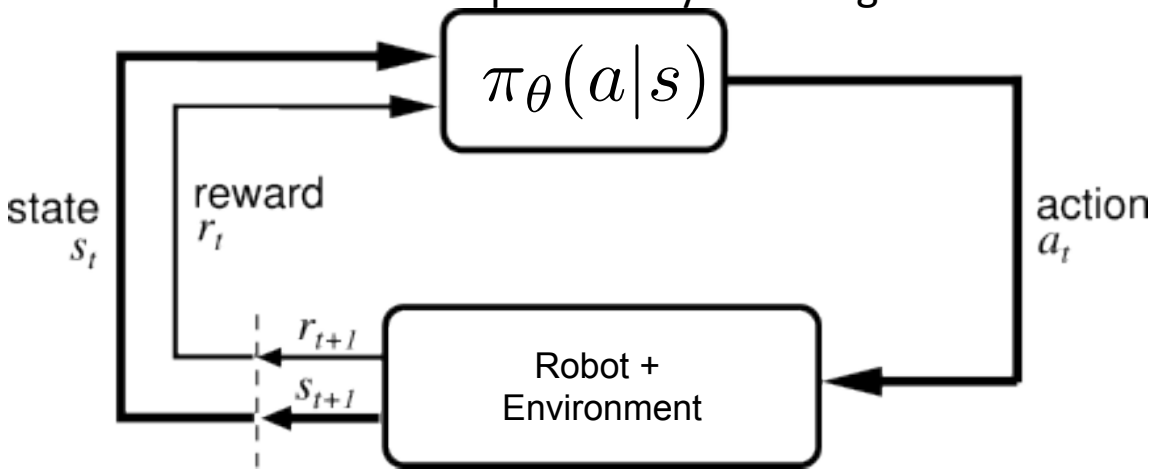


- **Deep reinforcement learning**



Reinforcement Learning (RL)

probability of taking action a in state s

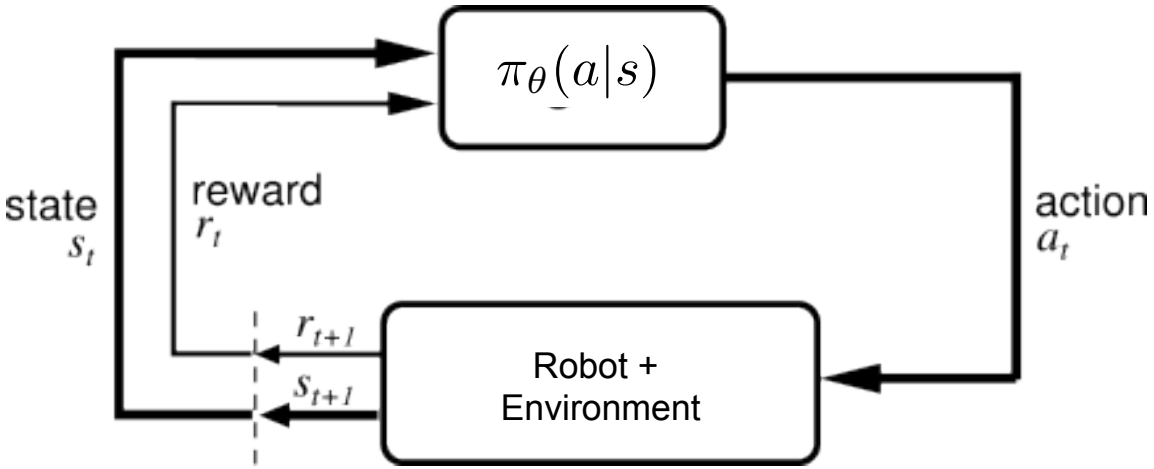


- Robotics
- Marketing / Advertising
- Dialogue
- Optimizing operations / logistics
- Queue management
- …

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

Reinforcement Learning

probability of taking action a in state s

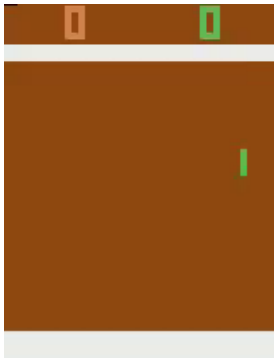


- Goal:

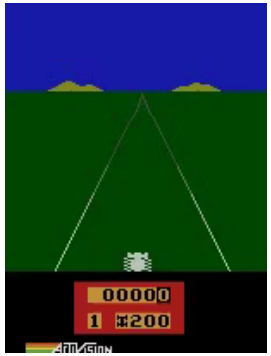
$$\max_\theta \quad \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

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

From Pixels to Actions?

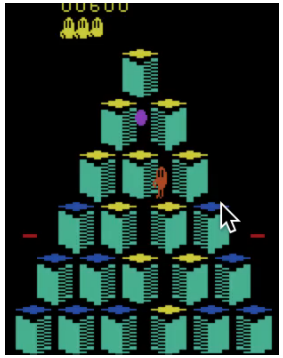


Pong



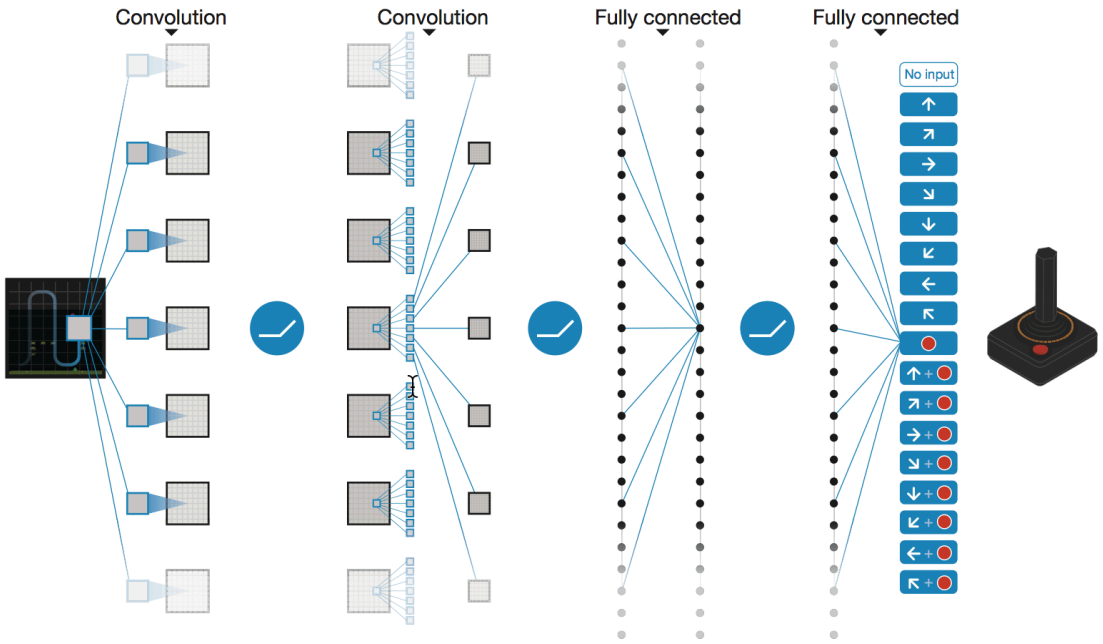
Enduro

Beamrider



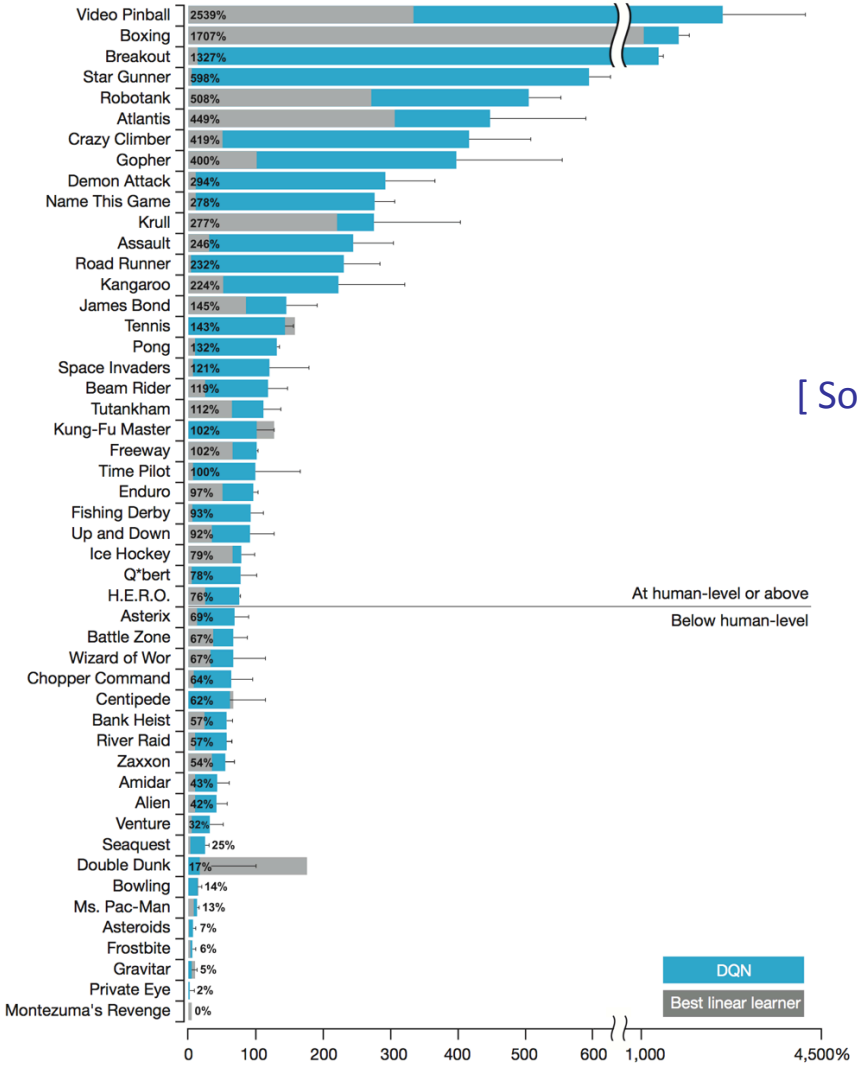
Q*bert

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)]



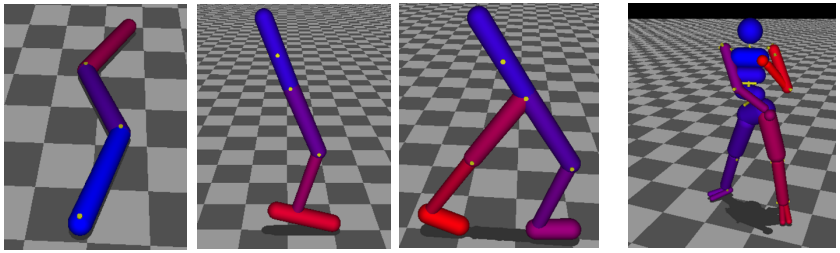
[Source: Mnih et al., Nature 2015 (DeepMind)]

Deep Q-Network (DQN)

- Approach:
  - Q-learning with e-greedy and deep network as function approximator
- Key idea 1: stabilizing Q-learning
  - Mini-batches of size 32 (vs. single sample updates)
  - Q-values used to compute temporal difference only updated every 10,000 updates
- Key idea 2: lots of data / compute
  - trained for a total of 50 million frames (=***38 days of game experience***) and use a replay memory of one million most recent frames

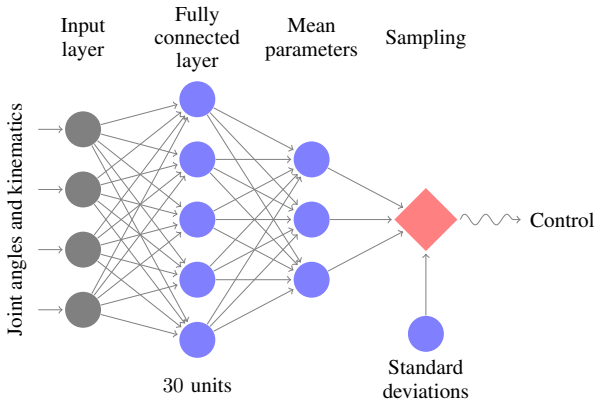
How About Continuous Control, e.g., Locomotion?

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

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

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$

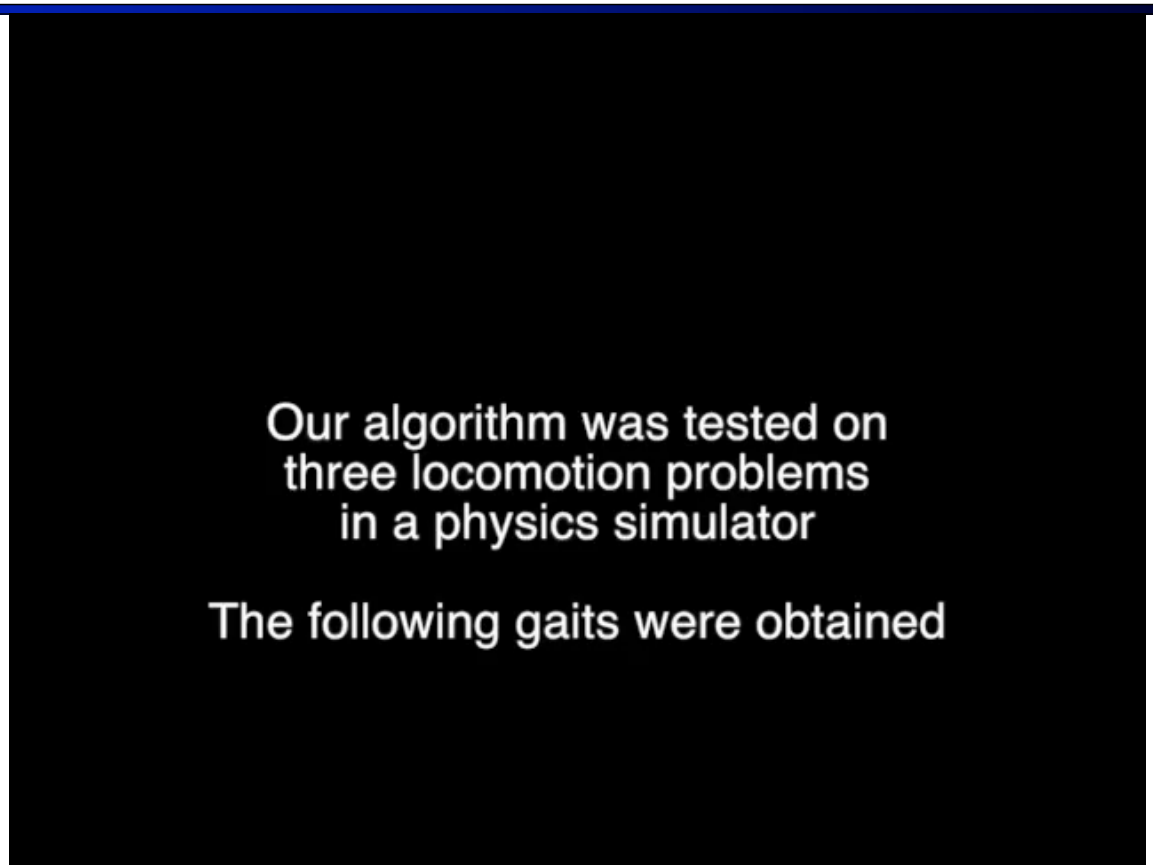
$$\text{s.t.} \quad KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

Stability

- Trust Region:
  - Sampled evaluation of gradient
  - Gradient only locally a good approximation
  - Change in policy changes state-action visitation frequencies

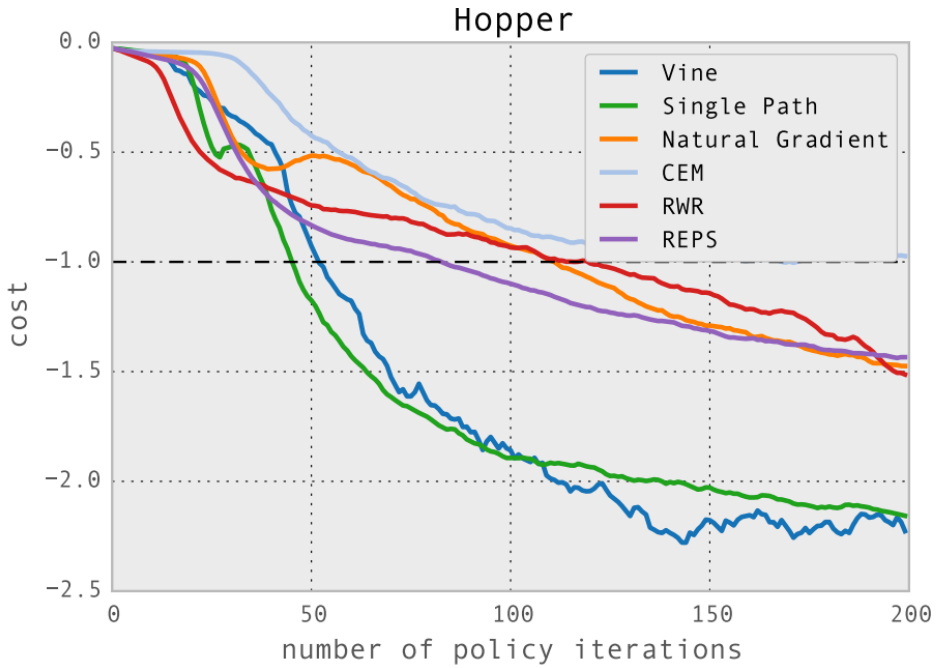
[Schulman, Levine, Moritz, Jordan, Abbeel, ICML 2015]

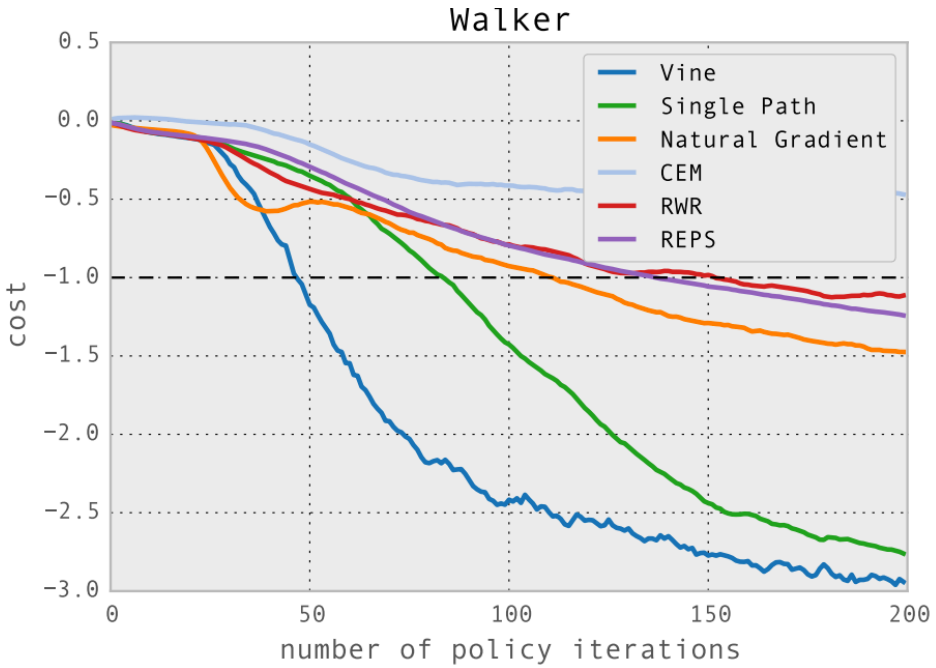
Experiments in Locomotion



[Schulman, Levine, A.]

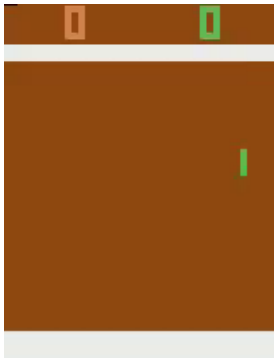
Learning Curves -- Comparison



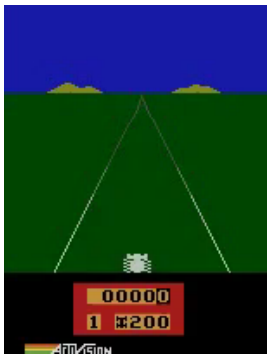


Atari Games

- 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]

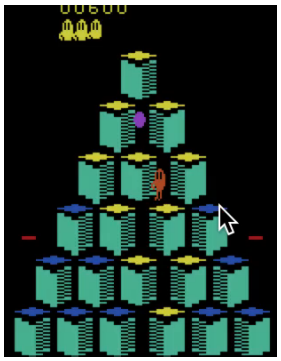


Pong



Enduro

Beamrider



Q*bert

Revisiting the Gradient Estimate

Trust Region Policy Optimization (TRPO)

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$

$$\text{s.t.} \quad KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

Generalized Advantage Estimation (GAE)

$$\hat{g} = \sum_i \sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{k=t}^{H} R(s_k) - \hat{V}_\phi(s_t) \right)$$

Key ideas:

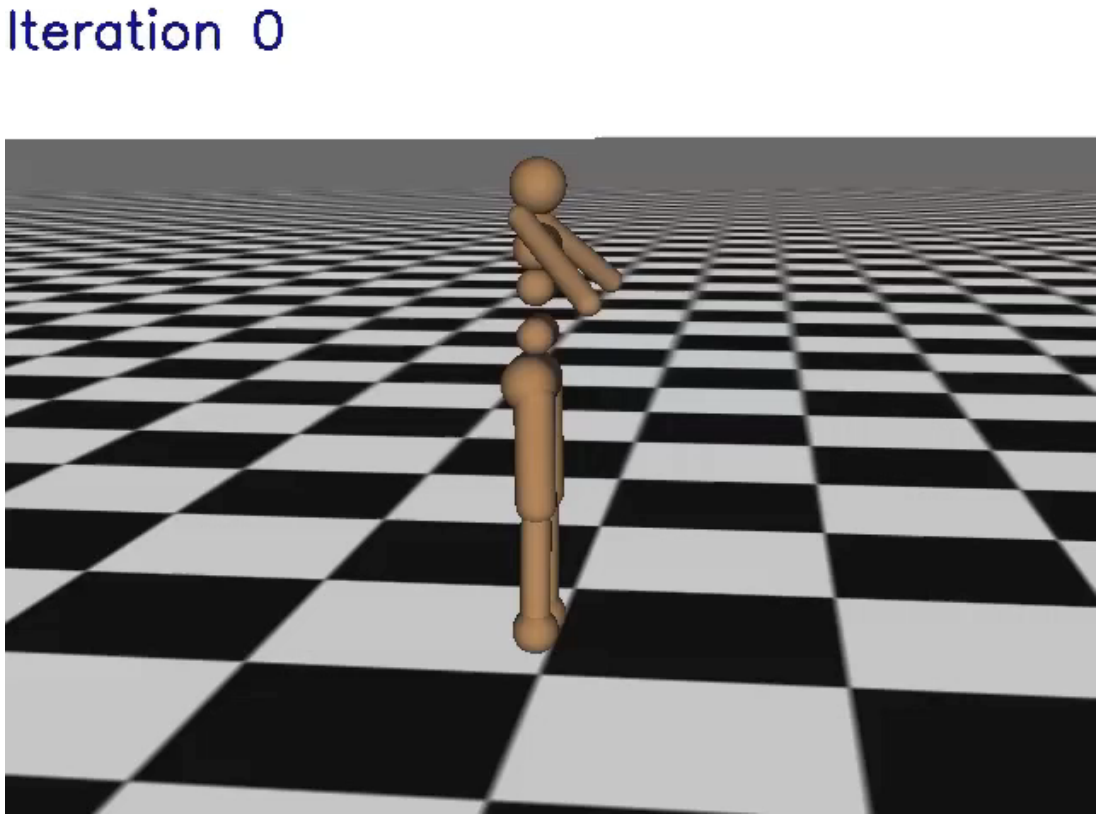
Credit Assignment

$\hat{Q}_\phi(s_t)$

- Deep neural net
- Trust-region optimization

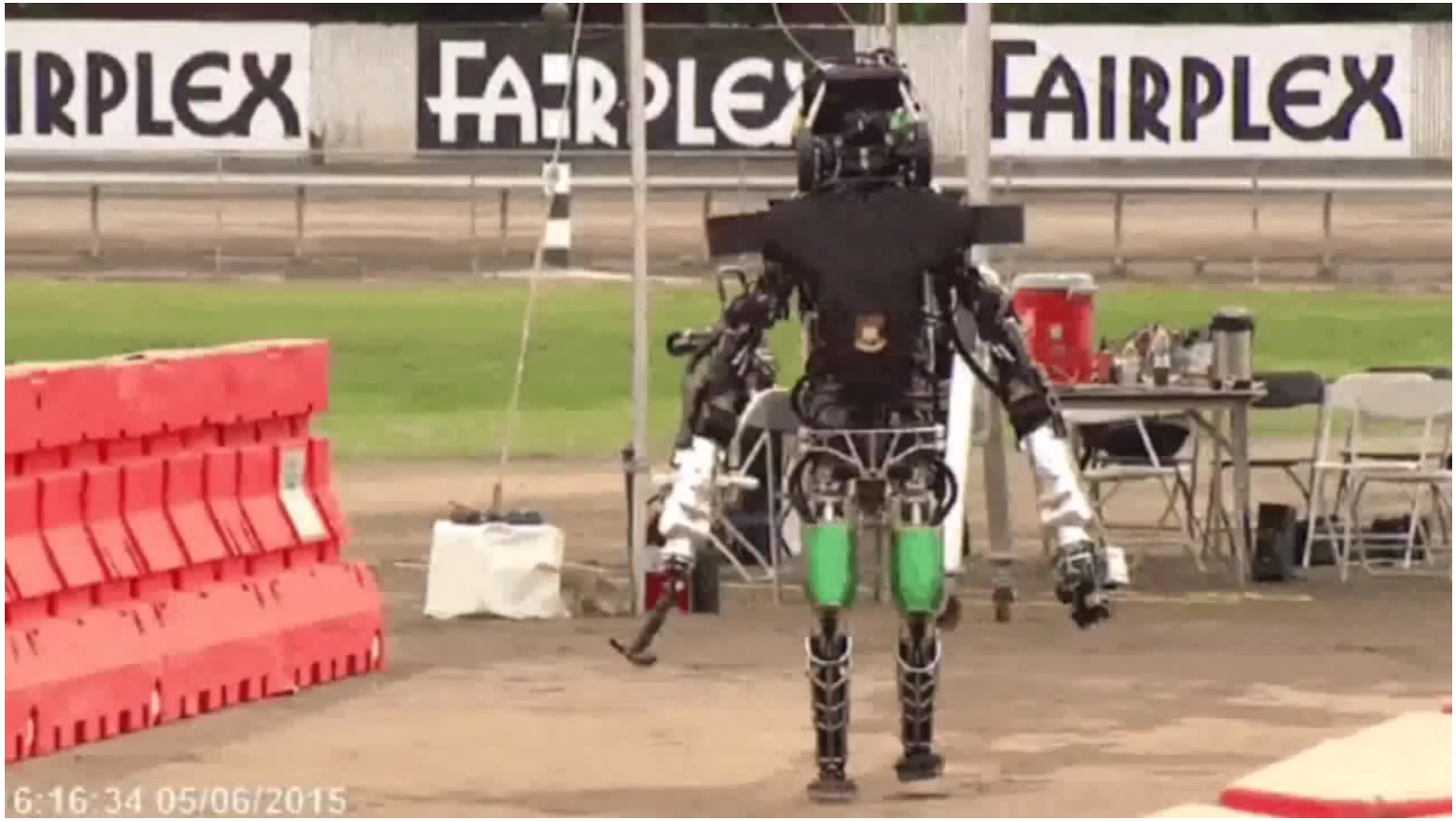
[Schulman, Moritz, Levine, Jordan, Abbeel, 2015 (under review ICLR)]

Learning Locomotion

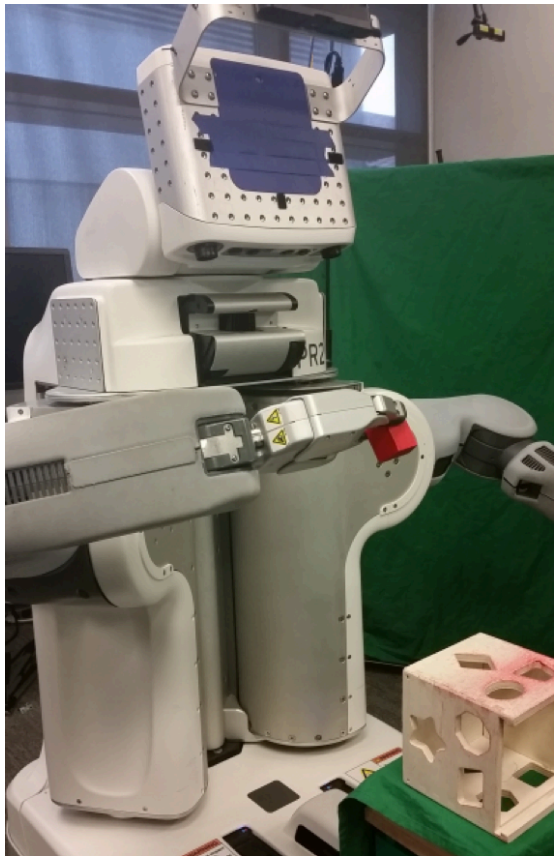


[Schulman, Moritz, Levine, Jordan, Abbeel, 2015]

In Contrast: Darpa Robotics Challenge

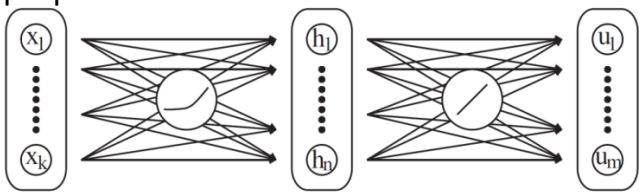


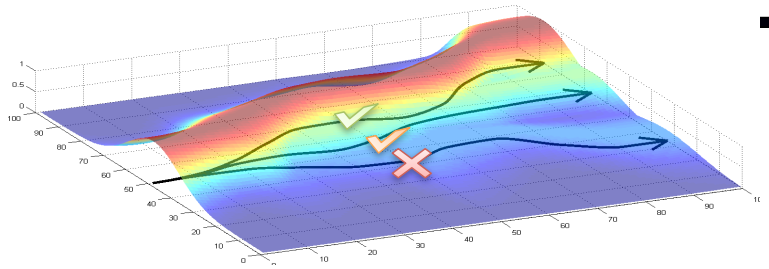
How About Real Robotic Visuo-Motor Skills?



Guided Policy Search

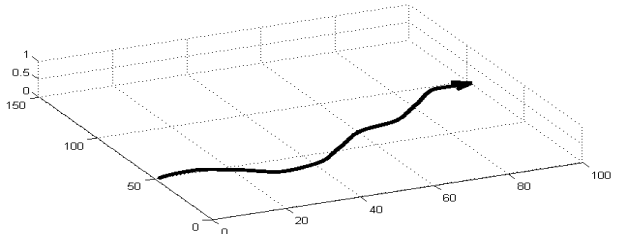
general-purpose neural network controller



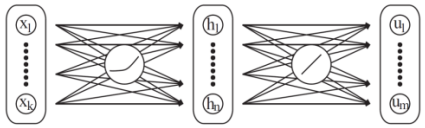


policy search (RL)	complex dynamics	complex policy	HARD
supervised learning	~~complex dynamics~~	complex policy	EASY
trajectory optimization	complex dynamics	~~complex policy~~	EASY

trajectory optimization

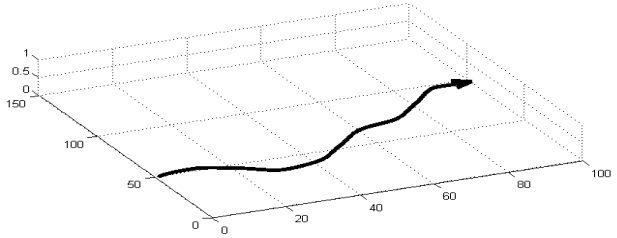


Supervised learning

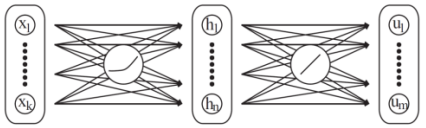


Guided Policy Search

trajectory optimization



Supervised learning



- Issue with just optimal trajectories
  - Don't learn how to steer back onto optimal trajectories

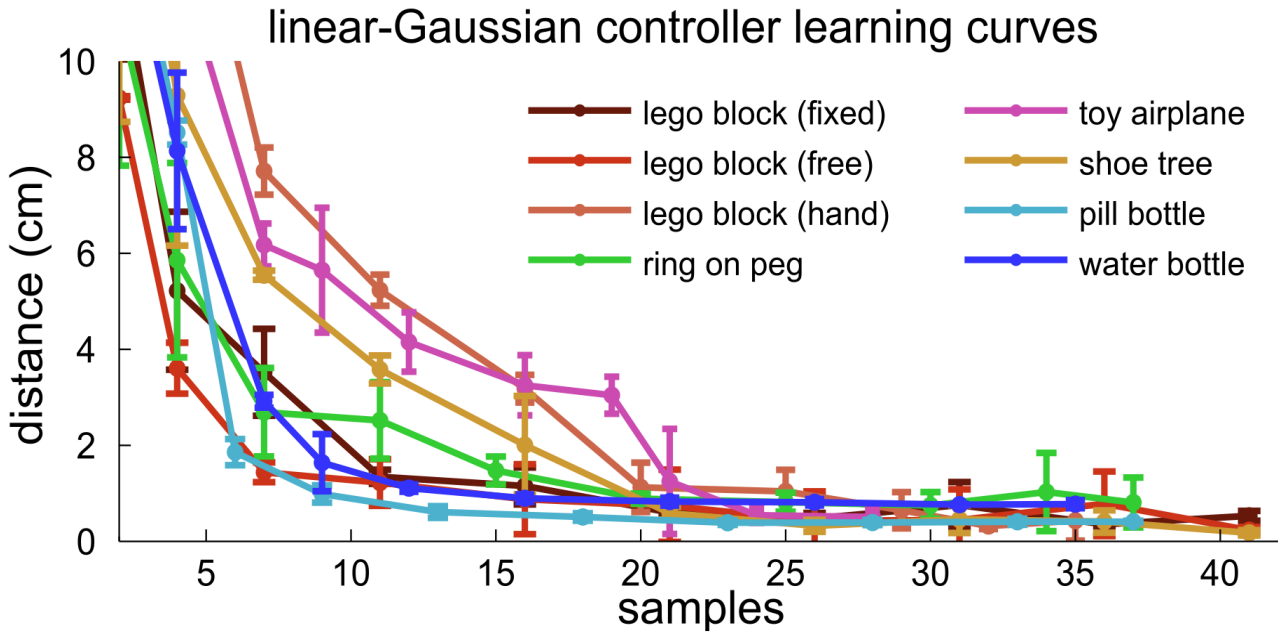
  → Distribution over near-optimal trajectories
- Issue with two-phase pipeline
  - Representational mismatch trajectory distribution vs. neural net

  → Joint optimization

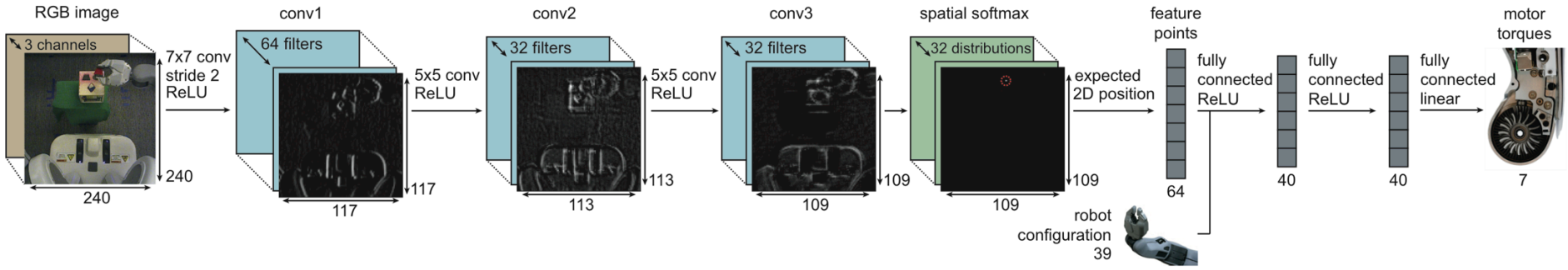
$$\max_{\{\pi^{(i)}\},\theta} \sum_i \mathrm{E}[\sum_{t=0}^{H} R(s_t^{(i)}) \mid \pi^{(i)}] - \lambda \sum_i \|\pi^{(i)} - \pi_\theta\|$$

Block Stacking – Learning the Controller for a Single Instance

Linear-Gaussian Controller Learning Curves

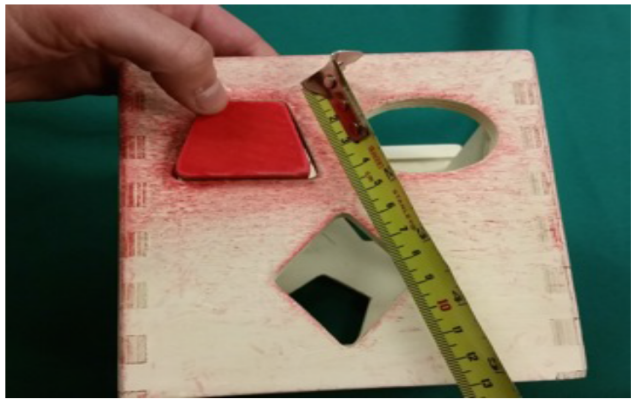


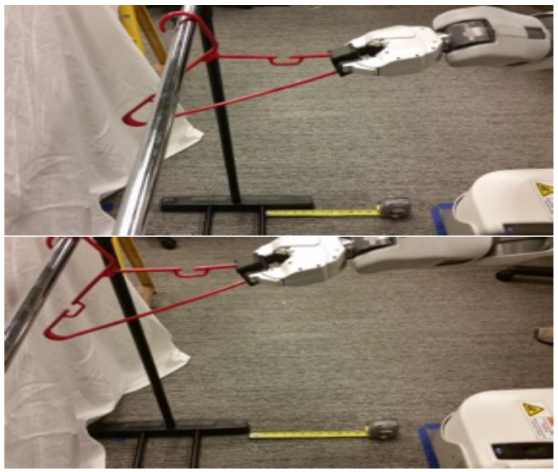
Architecture (92,000 parameters)



[Levine*, Finn*, Darrell, Abbeel, 2015, TR at: rll.berkeley.edu/deeplearningrobotics]

Experimental Tasks



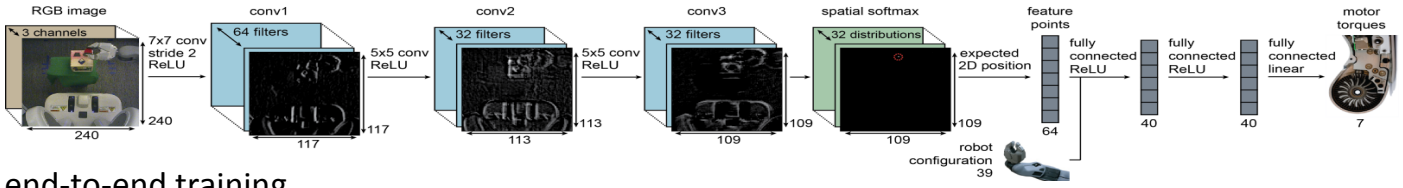


Learning

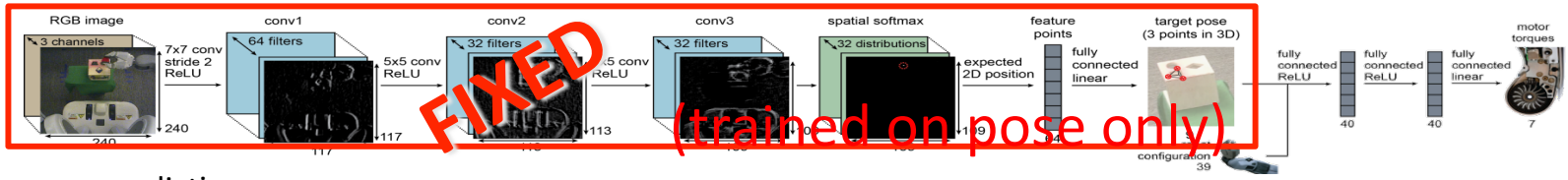
Learned Skills

[Levine*, Finn*, Darrell, Abbeel, 2015, TR at: rll.berkeley.edu/deeplearningrobotics]

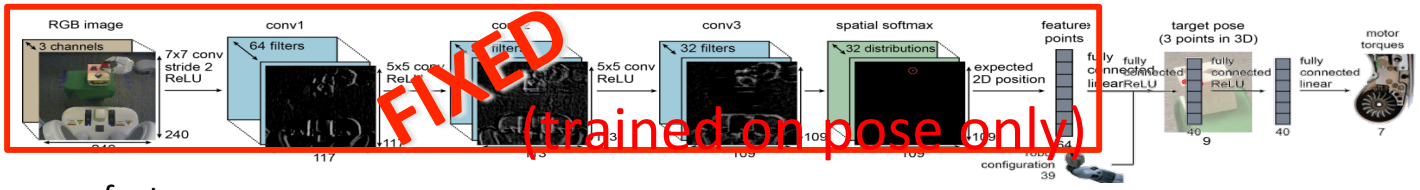
Comparisons



end-to-end training



pose prediction



pose features

Comparisons

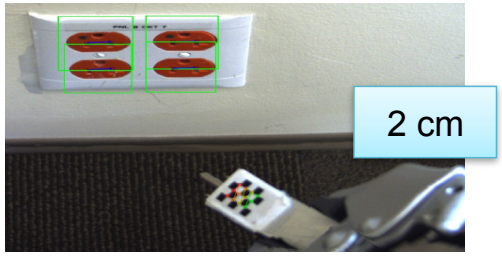
coat hanger	success rate
pose prediction	55.6%
pose features	88.9%
end-to-end training	**100%**

shape sorting cube	success rate
pose prediction	0%
pose features	70.4%
end-to-end training	**96.3%**

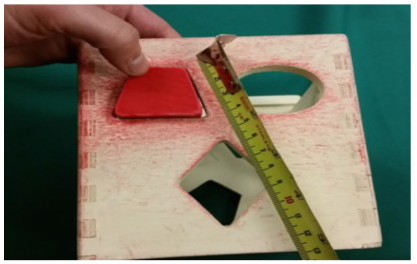
toy claw hammer	success rate
pose prediction	8.9%
pose features	62.2%
end-to-end training	**91.1%**

bottle cap	success rate
pose prediction	n/a
pose features	55.6%
end-to-end training	**88.9%**

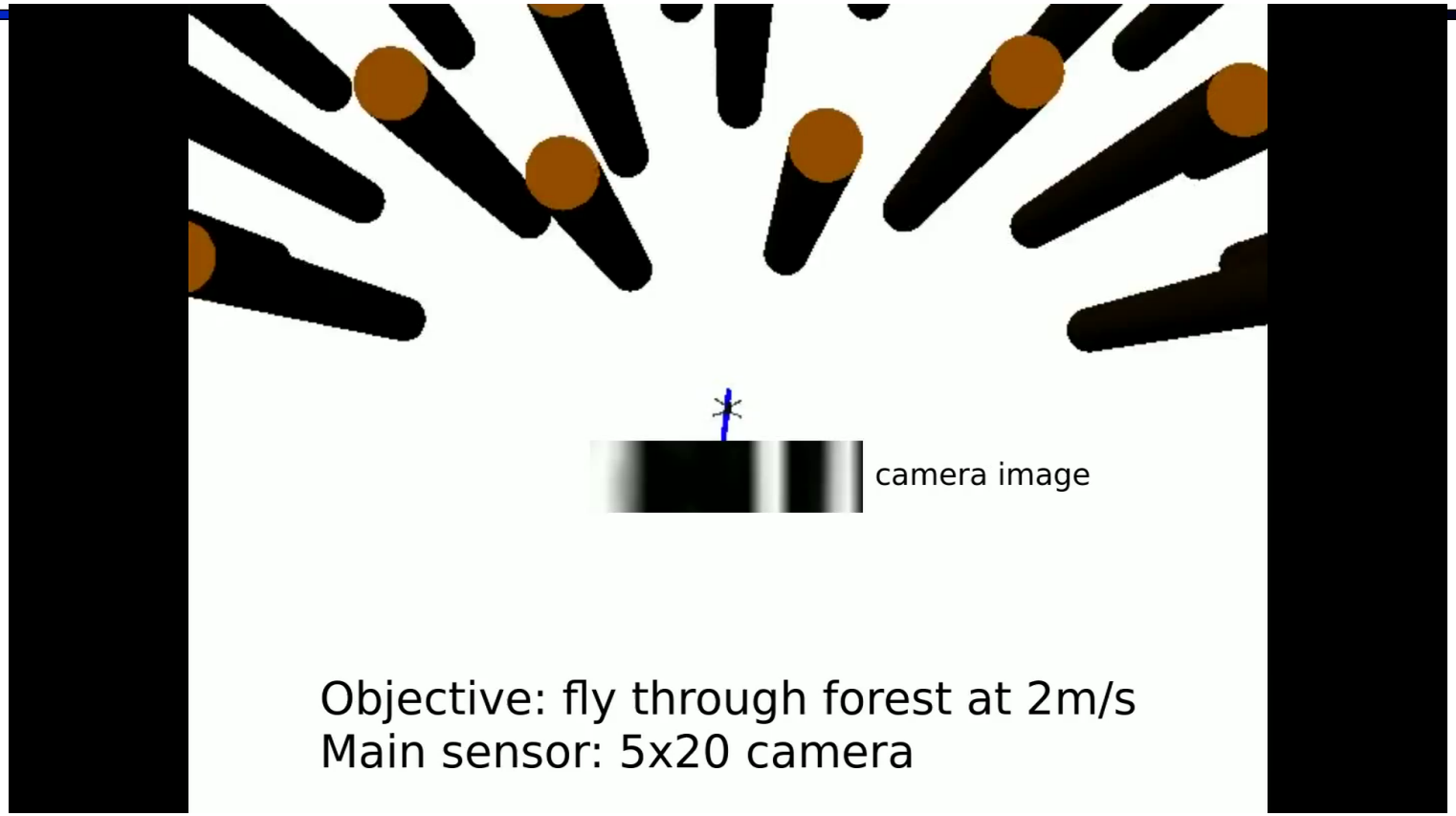
network architecture	test error (cm)
softmax + feature points (**ours**)	**1.30 ± 0.73**
softmax + fully connected layer	2.59 ± 1.19
fully connected layer	4.75 ± 2.29
max-pooling + fully connected	3.71 ± 1.73



Meeussen et al. (Willow Garage)



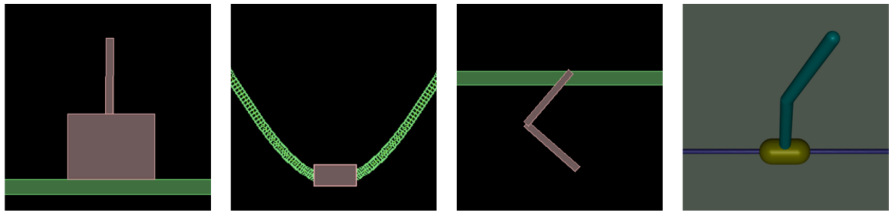
Experiments: Learned Neural Network Policy



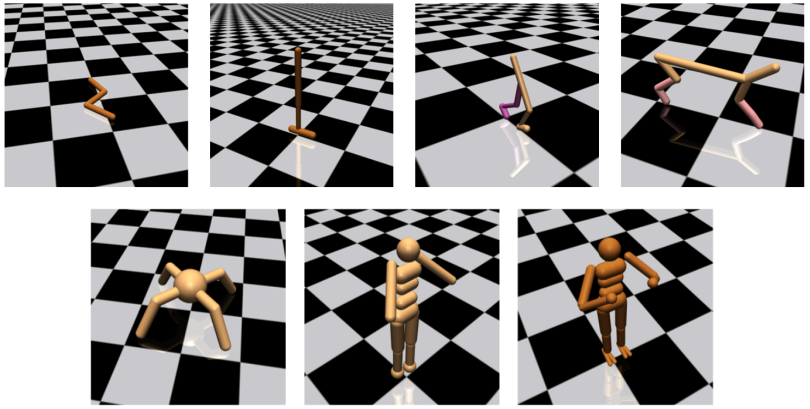
[Khan, Zhang, Levine, Abbeel 2016]

Deep RL Benchmarking -- Tasks

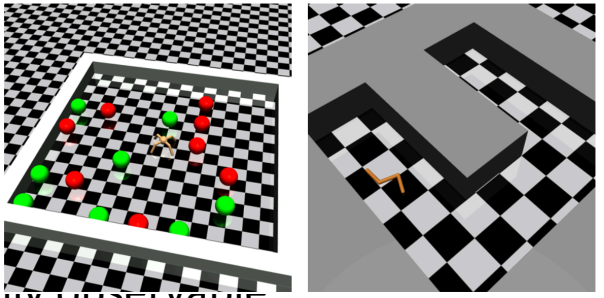
1. Basic tasks



2. Locomotion



3. Hierarchical



4. Partially observable

   sensing, delayed action, sysID

5. Driving…

Deep RL Benchmarking -- Algorithms

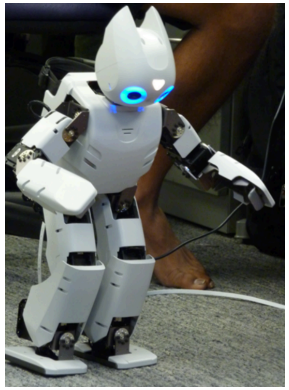
- Reinforce
- Truncated Natural Policy Gradient
- Reward-Weighted Regression (RWR)
- Relative Entropy Policy Search (REPS)
- Trust-Region Policy Optimization (TRPO)
- Cross-Entropy Method (CEM)
- Covariance Matrix Adaptation Evolution Strategy (CMA-ES)
- Deep Deterministic Policy Gradients (DDPG)
- …

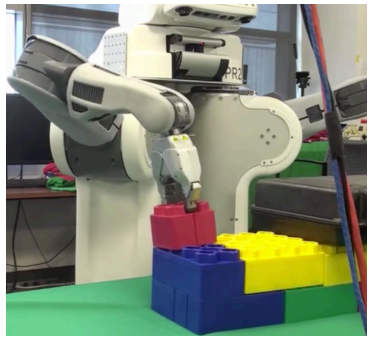
Started benchmarking (on arXiv soon...)

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

Frontiers: Applications





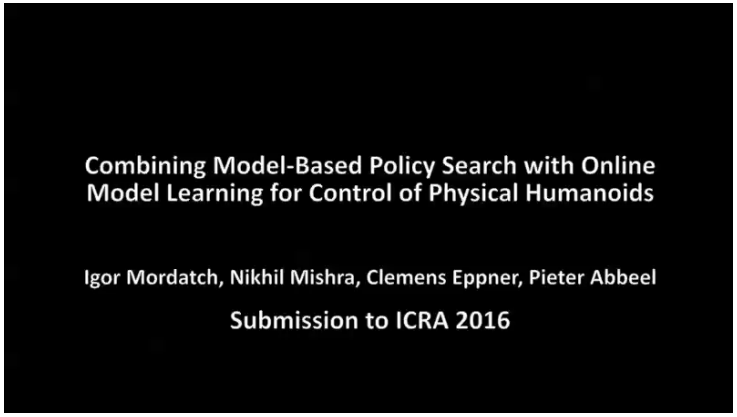
Frontiers

- Shared and transfer learning





- Transfer simulation -> real world



[Mordatch, Mishra, Eppner, Abbeel ICRA 2016]

- Memory
  - Estimation
  - Unsupervised + RL

Next Time: Natural Language Processing!