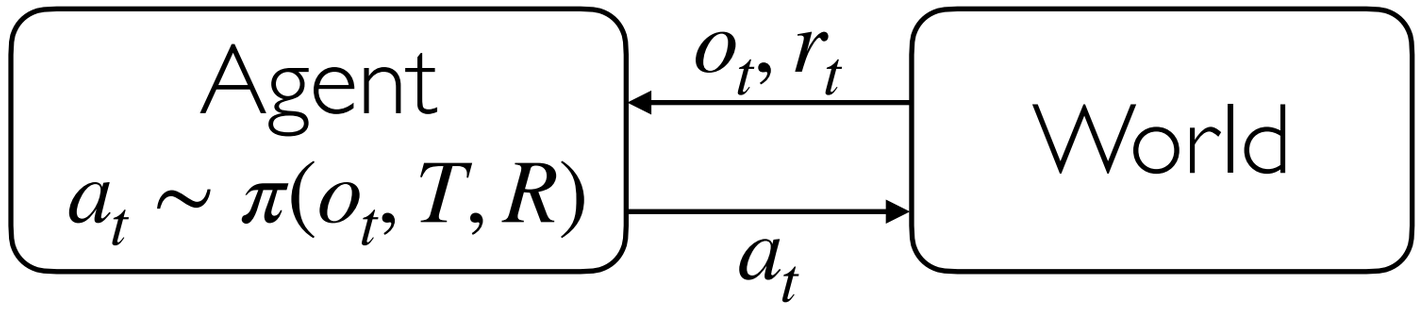
Imitation Learning

Saurabh Gupta

Convert into a Supervised Training Problem

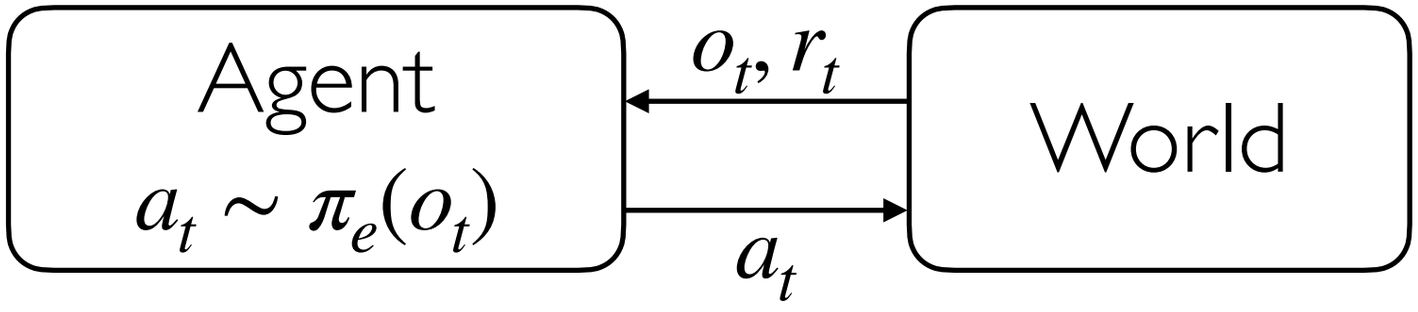
Most General Case



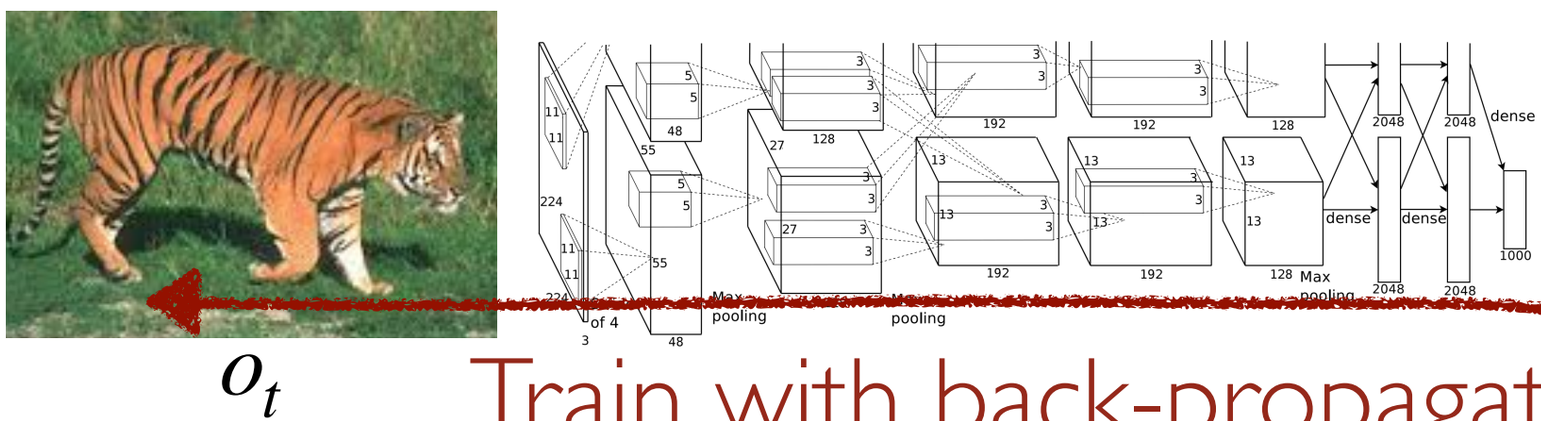
Behavior Cloning

Train Time

Assume an expert e can solve this MDP.

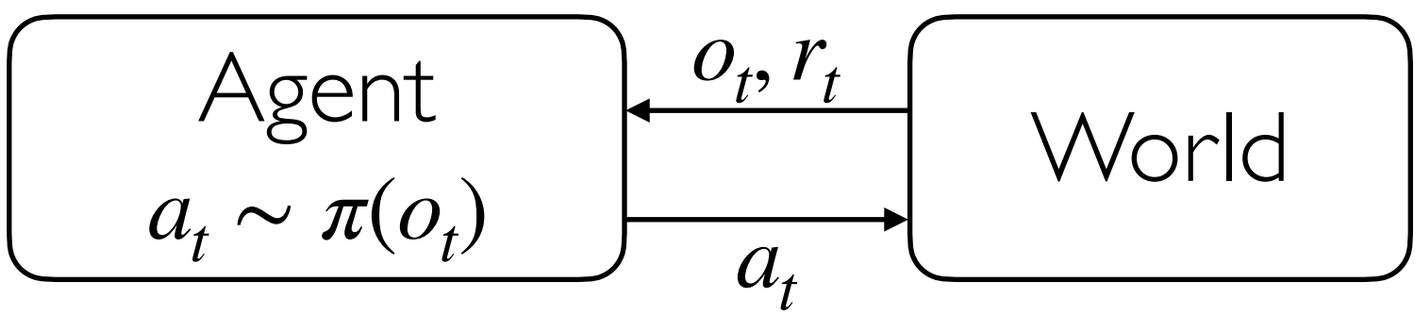


1. *Ask the expert e to solve this MDP.*
2. *Collect labeled dataset D from expert.*
3. *Train a function* $\pi(o_t)$ *that mimics* $\pi_e(o_t)$ *on D.* $\pi(o_t)$ $\pi_e(o_t)$



o_t

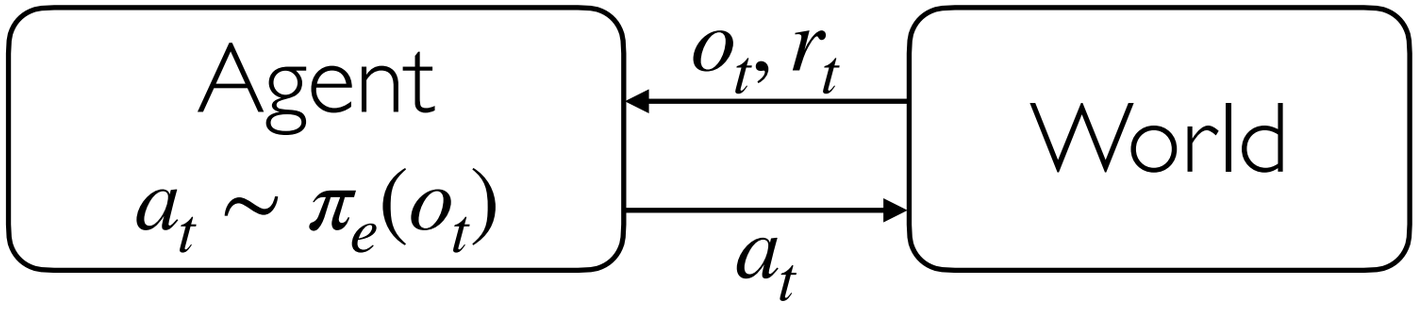
Test Time



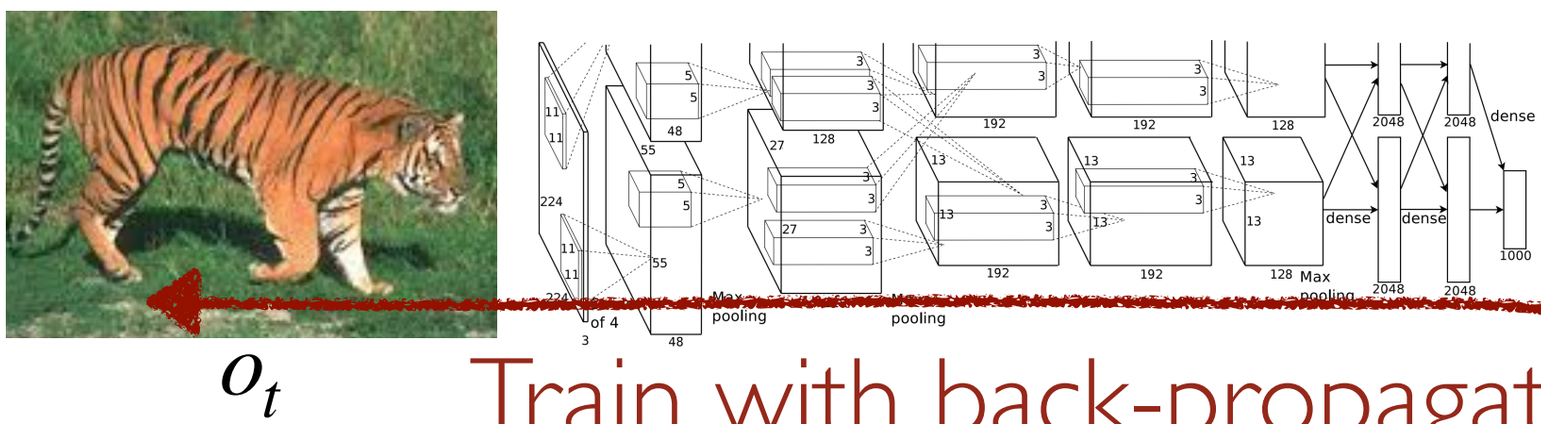
Supervision from Human Expert

Train Time

Assume an expert e can solve this MDP.

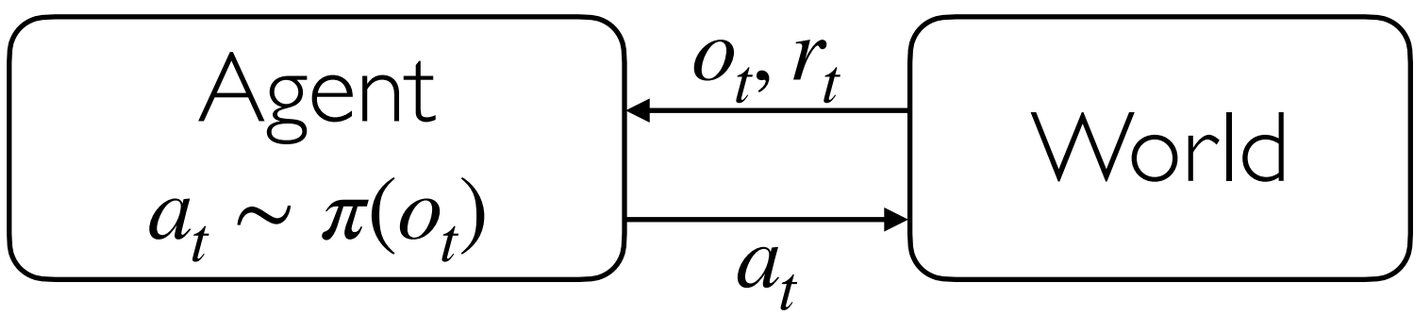


1. *Ask the expert e to solve this MDP.*
2. *Collect labeled dataset D from expert.*
3. *Train a function* $\pi(o_t)$ *that mimics* $\pi_e(o_t)$ *on D.* $\pi(o_t)$ $\pi_e(o_t)$



o_t

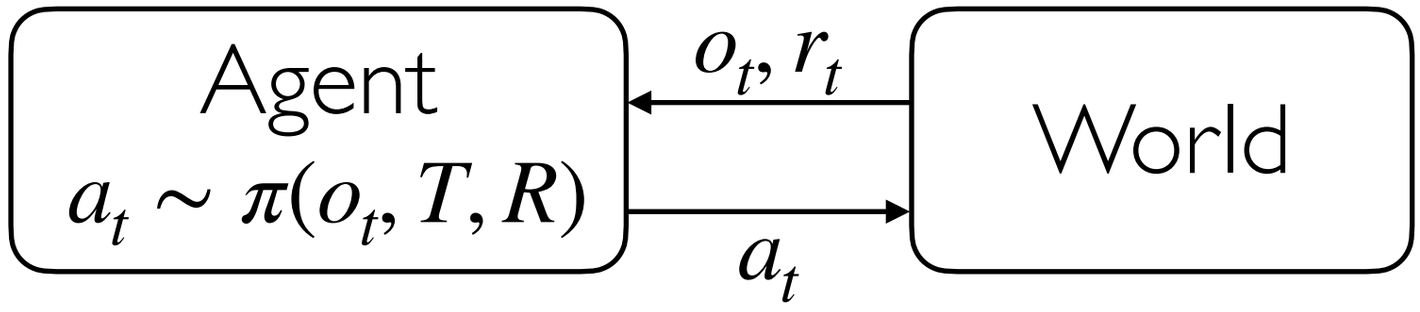
Test Time



Supervision from ~~Human~~ Algorithmic Expert

Train Time

1. Instrument the environment such that it becomes a known MDP.



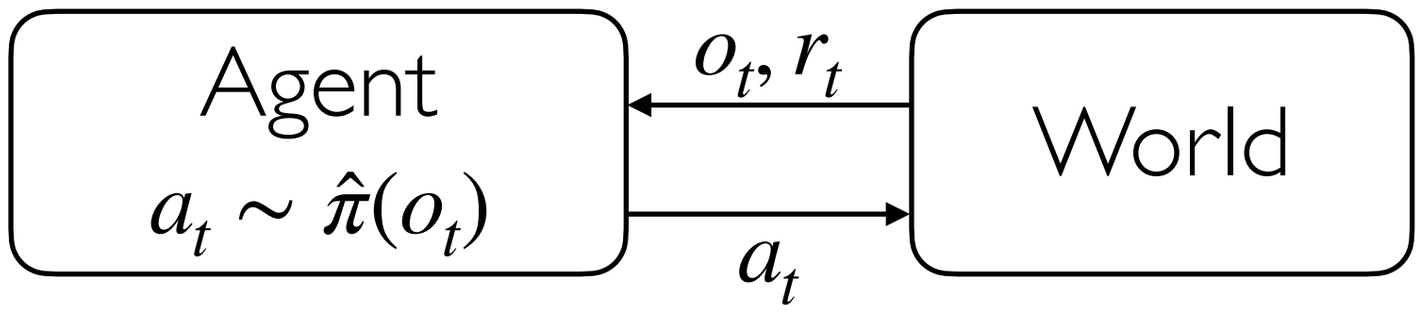
Fully Observed System $o_t = s_t$

Known or Learned Transition Function $s_{t+1} \sim \hat{T}(s_t, a_t)$

Known Reward Function $R(s_{t+1}, s_t, a_t)$

2. Train a function $\hat{\pi}(o_t)$ that mimics $\pi(o_t, T, R)$

Test Time

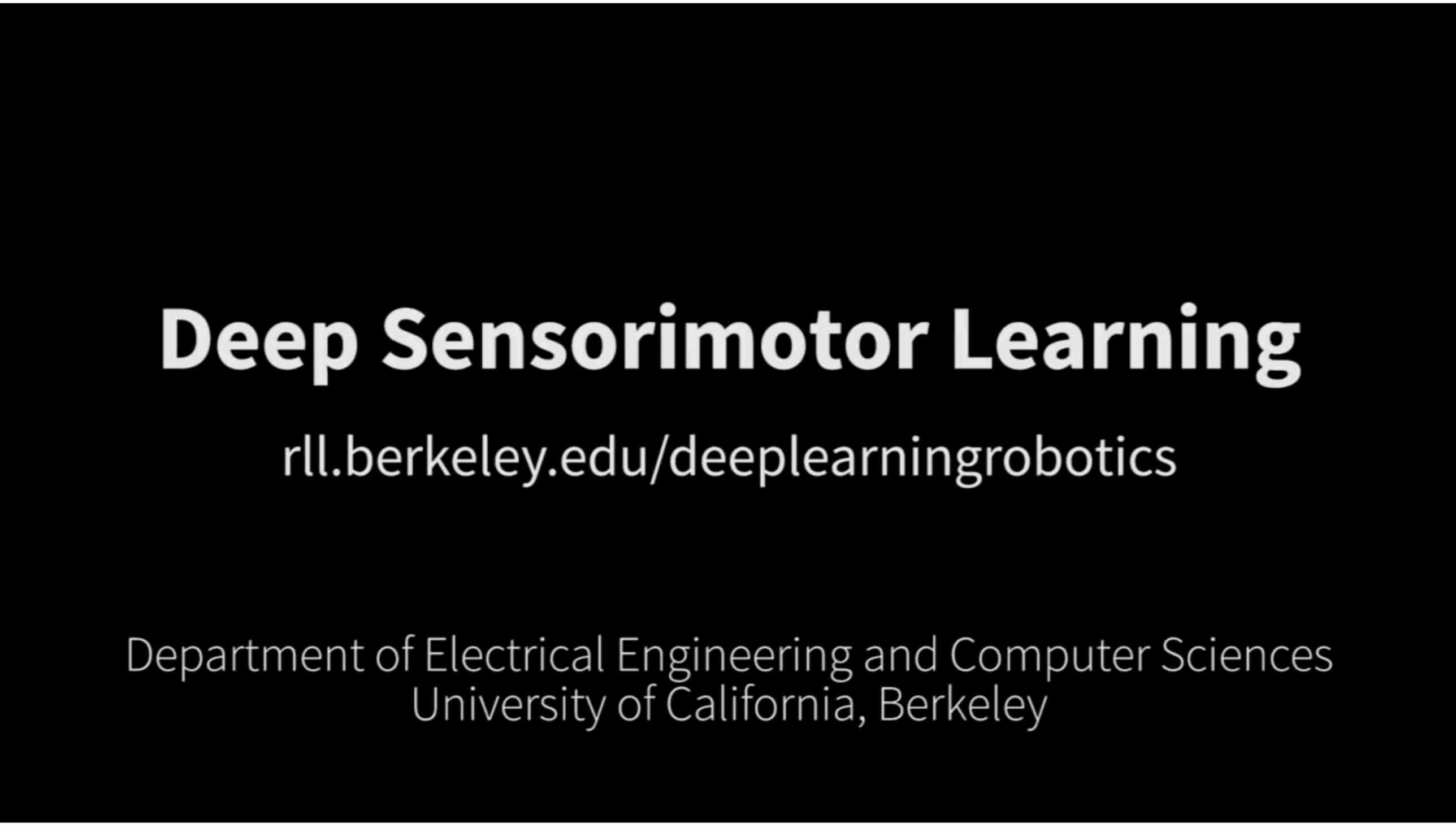


Fully Observed System $o_t = s_t$

Known or Learned Transition Function $s_{t+1} \sim \hat{T}(s_t, a_t)$

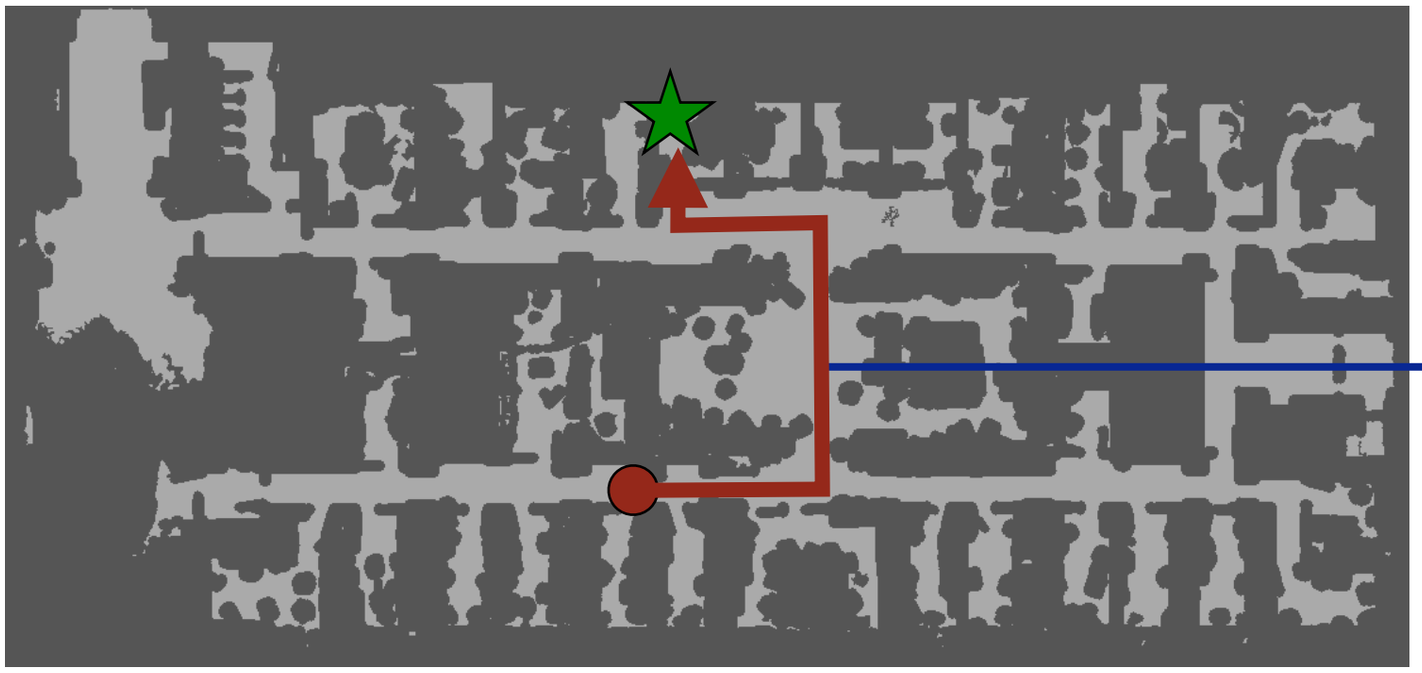
Known Reward Function $R(s_{t+1}, s_t, a_t)$

Supervision from ~~Human~~ Algorithmic Expert



S. Levine et al. **A End-to-End Training of Deep Visuomotor Policies.** JMLR 2016.

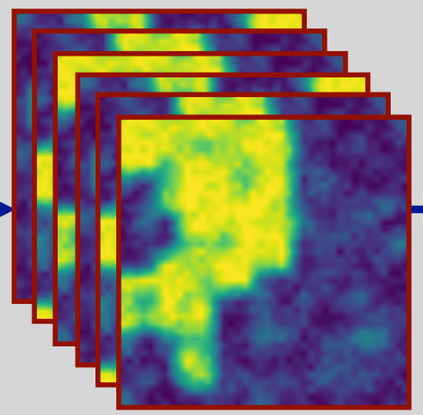
Supervision from ~~Human~~ Algorithmic Expert



Optimal Action

=

Differentiable Mapper



Differentiable Planner

Predicted Action

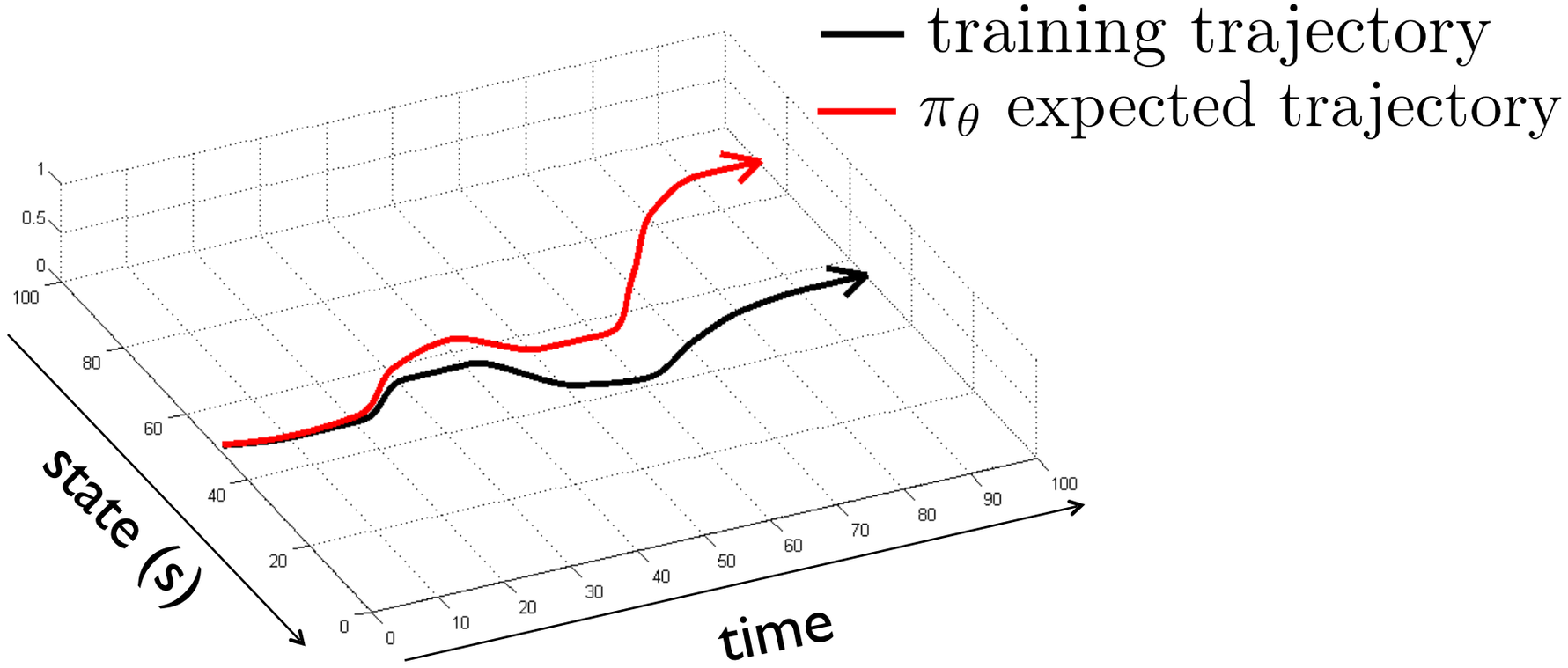
Train with back-propagation

S. Gupta et al. **Cognitive Mapping and Planning for Visual Navigation.** CVPR 2017.

Behavior Cloning

Does it always work?

No, data mis-match problem

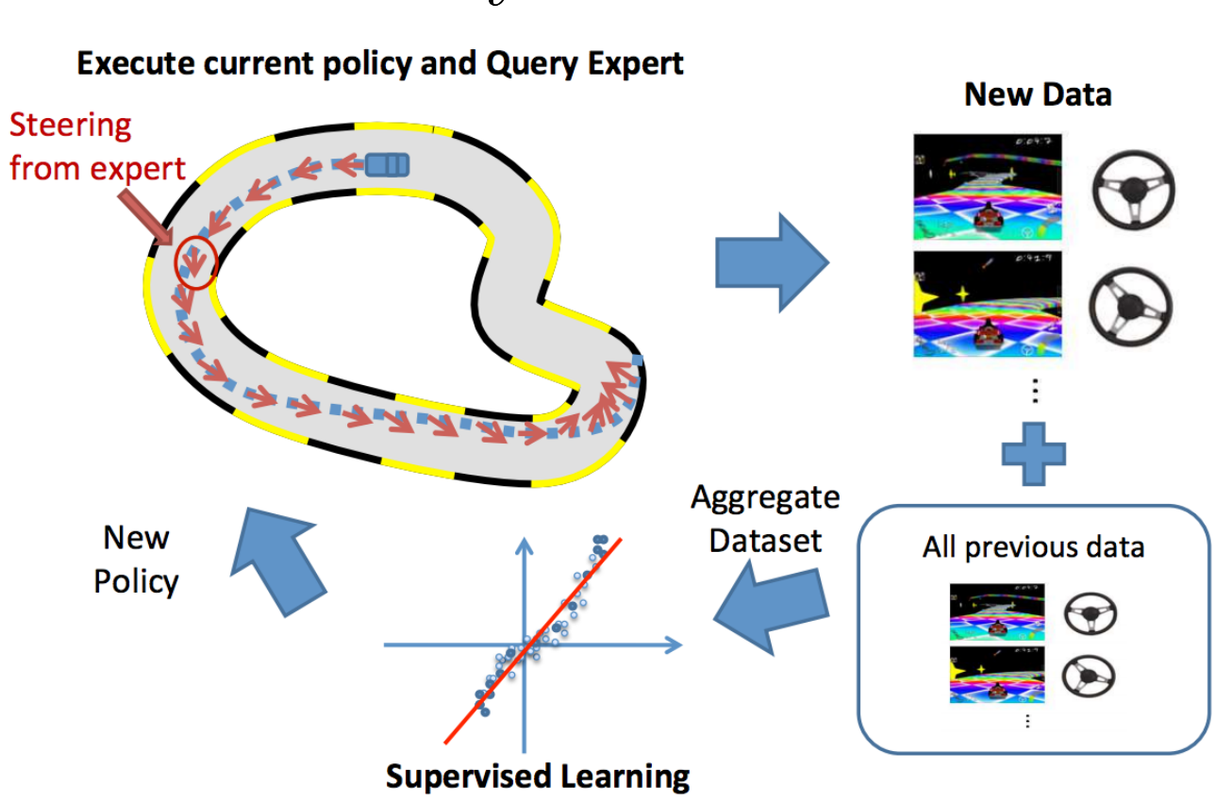


Fix Data Mis-Match Problem

DAgger: Dataset Aggregation

Collect labels on states visited by $\pi(o_t)$ *instead of* $\pi_e(o_t)$.

1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$



S. Ross et al. **A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning.** AISTAT 2011.

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning

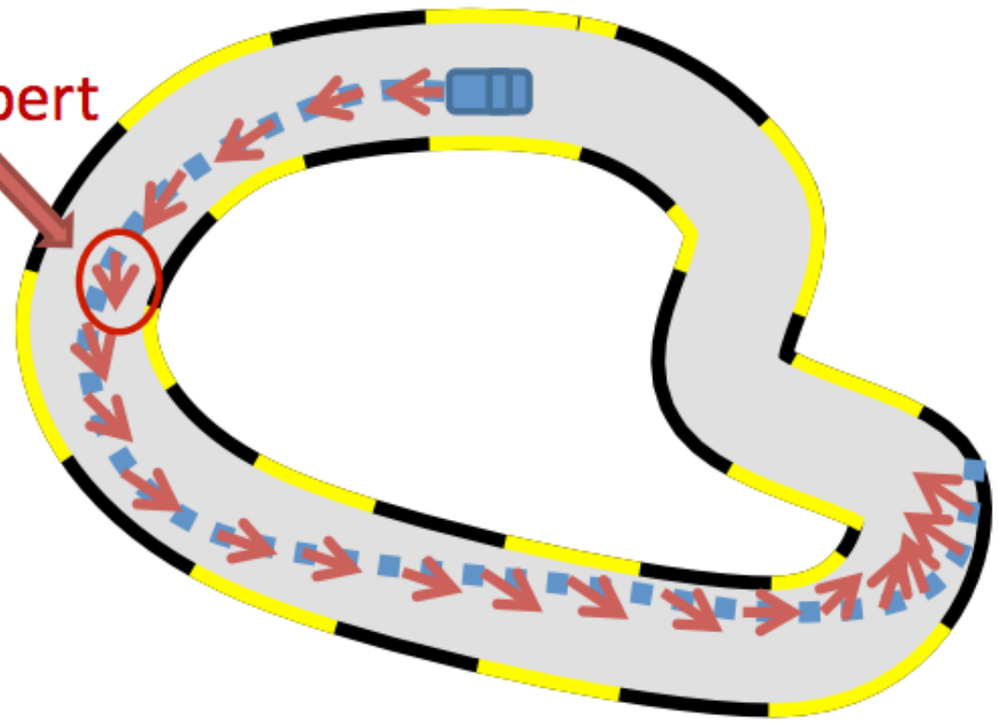
Stéphane Ross
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
stephaneross@cmu.edu

Geoffrey J. Gordon
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ggordon@cs.cmu.edu

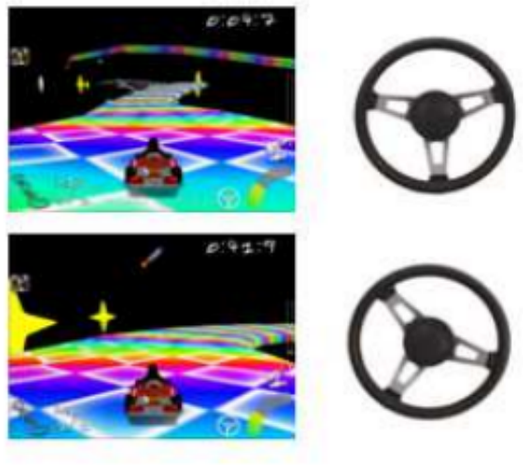
J. Andrew Bagnell
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dbagnell@ri.cmu.edu

Execute current policy and Query Expert

Steering
from expert

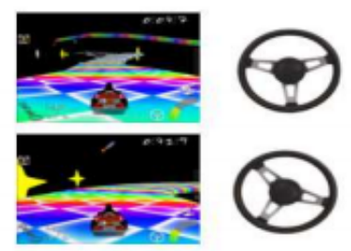


New Data

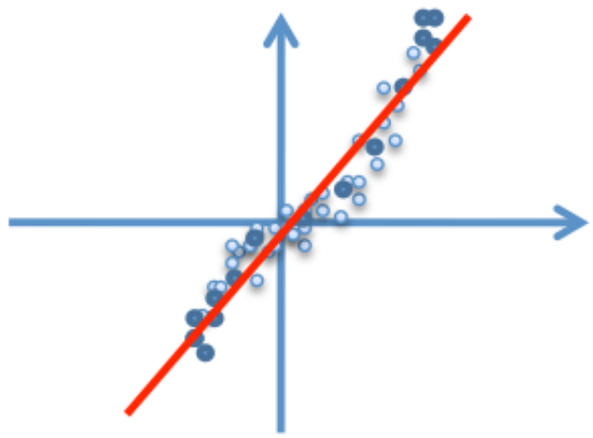


Aggregate
Dataset

All previous data



New
Policy



Supervised Learning

Forward Training and SMILe Algorithm

Initialize $\pi_1^0, \ldots, \pi_T^0$ to query and execute π^*.
for $i = 1$ **to** T **do**
  Sample T-step trajectories by following π^{i-1}.
  Get dataset $\mathcal{D} = \{(s_i, \pi^*(s_i))\}$ of states, actions taken by expert at step i.
  Train classifier $\pi_i^i = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim \mathcal{D}}(e_\pi(s))$.
  $\pi_j^i = \pi_j^{i-1}$ for all $j \neq i$
end for
Return $\pi_1^T, \ldots, \pi_T^T$

Algorithm 3.1: Forward Training Algorithm.

Initialize $\pi^0 \leftarrow \pi^*$ to query and execute expert.
for $i = 1$ **to** N **do**
  Execute π^{i-1} to get $\mathcal{D} = \{(s, \pi^*(s))\}$.
  Train classifier $\hat{\pi}^{*i} = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim \mathcal{D}}(e_\pi(s))$.
  $\pi^i = (1-\alpha)^i \pi^* + \alpha \sum_{j=1}^{i} (1-\alpha)^{j-1} \hat{\pi}^{*j}$.
end for
Remove expert queries: $\tilde{\pi}^N = \frac{\pi^N - (1-\alpha)^N \pi^*}{1-(1-\alpha)^N}$
Return $\tilde{\pi}^N$

Algorithm 4.1: The SMILe Algorithm.

DAgger

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

Algorithm 3.1: DAGGER Algorithm.

Super Tux Kart

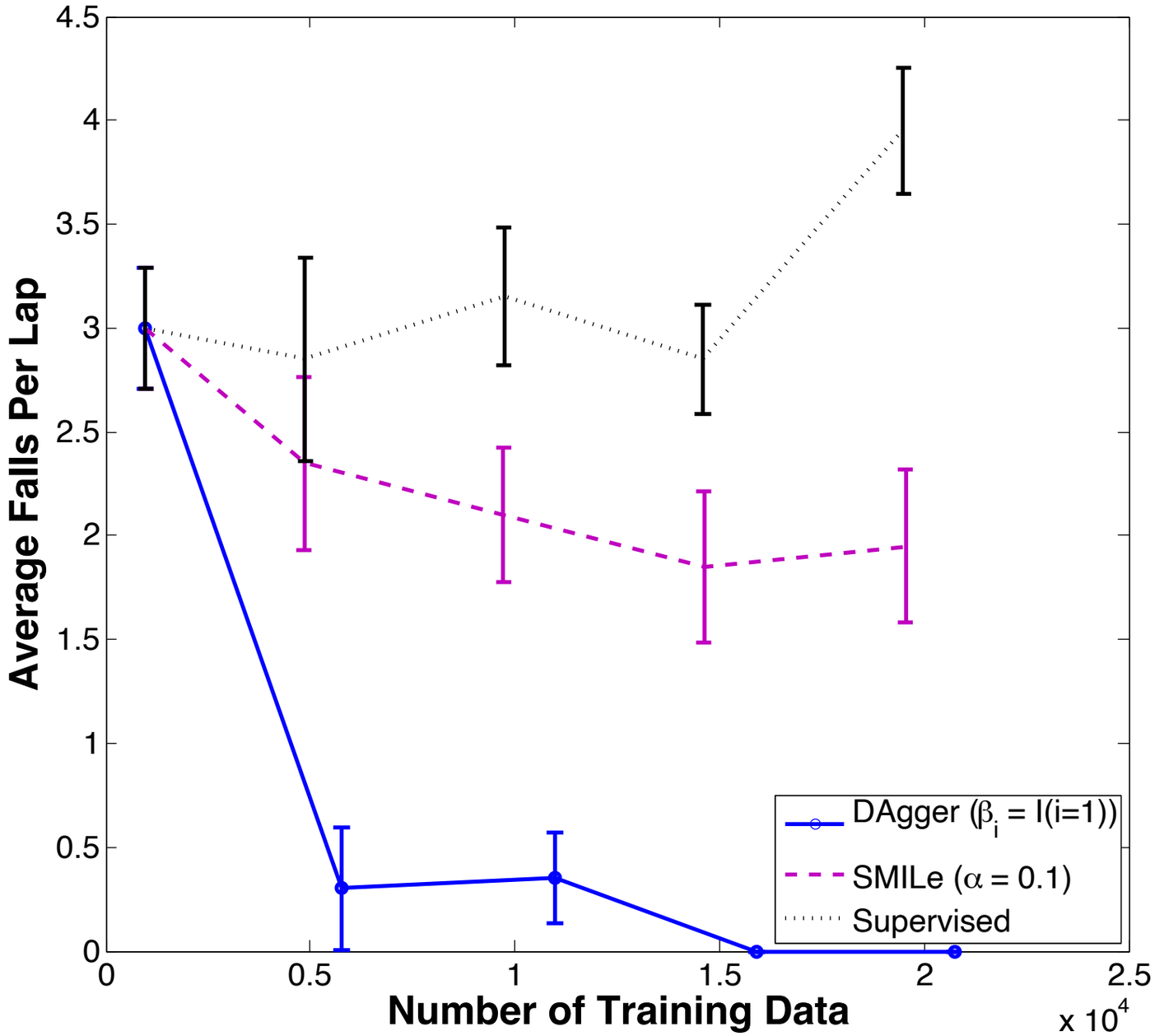


Figure 2: Average falls/lap as a function of training data.

Super Mario Bros.

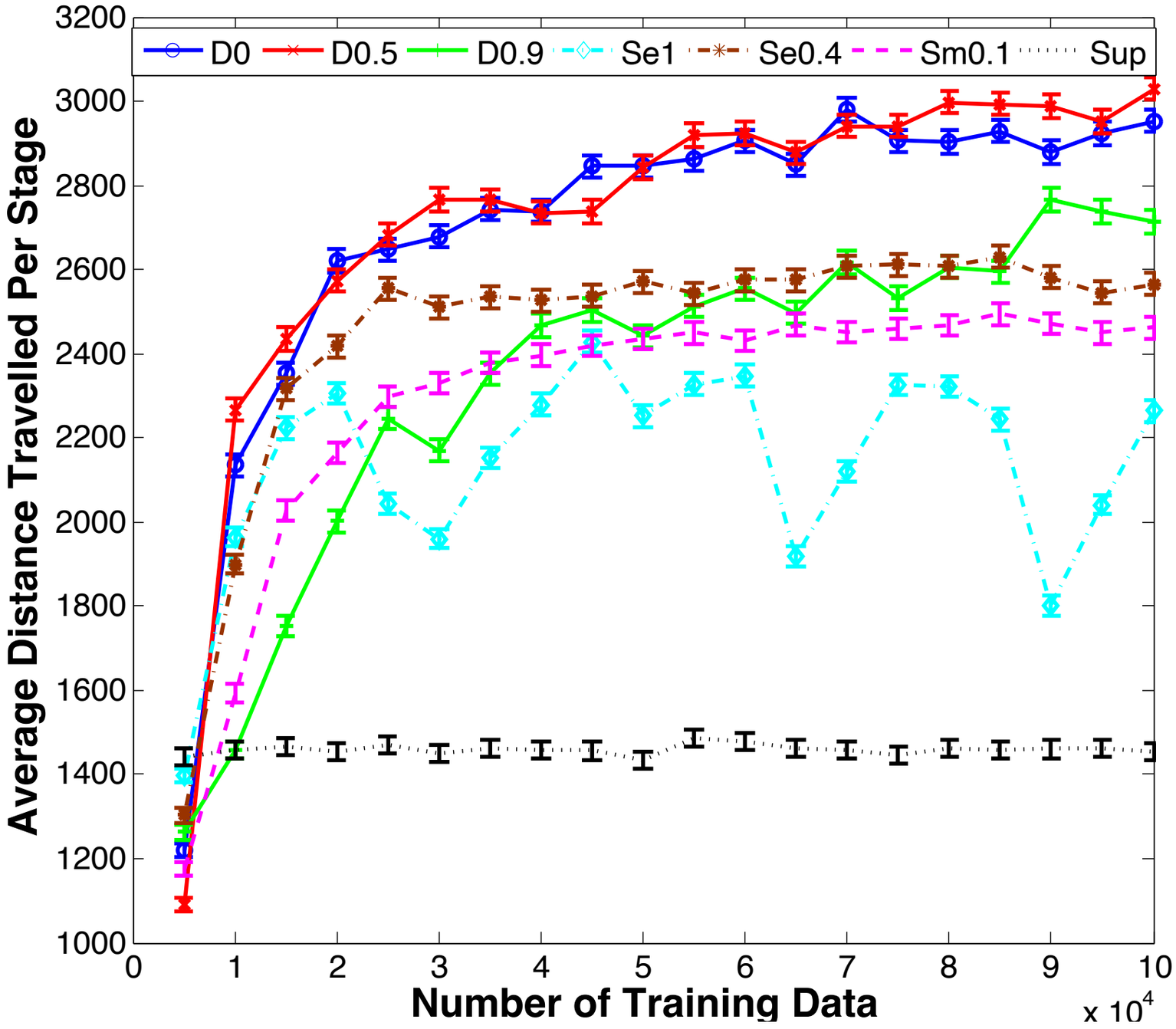


Figure 4: Average distance/stage as a function of data.

Structured Prediction

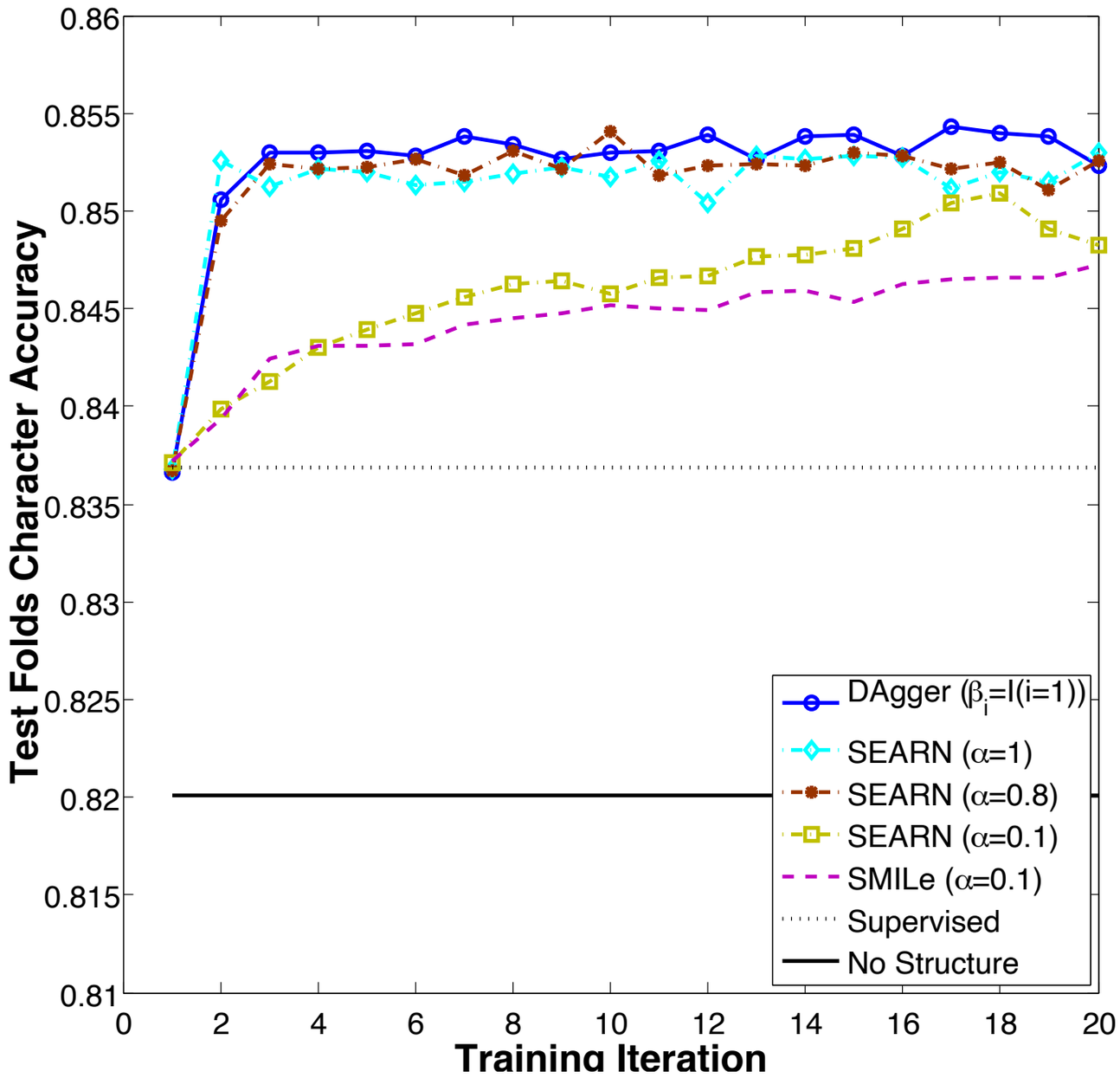


Figure 5: Character accuracy as a function of iteration.

Limitations?

- Need an interactive expert
- Exploration?

End-to-End Training of Deep Visuomotor Policies

Sergey Levine[†] SVLEVINE@EECS.BERKELEY.EDU
Chelsea Finn[†] CBFINN@EECS.BERKELEY.EDU
Trevor Darrell TREVOR@EECS.BERKELEY.EDU
Pieter Abbeel PABBEEL@EECS.BERKELEY.EDU
Division of Computer Science
University of California
Berkeley, CA 94720-1776, USA
[†]These authors contributed equally.

Summary

- Demonstrates end-to-end learning leads to higher performance than modularized approaches
- Design of a sample efficient way of training end-to-end policies
- Design of neural network architectures for visuo-motor tasks
- Key-trick: assume full-state information at training time

Training Process

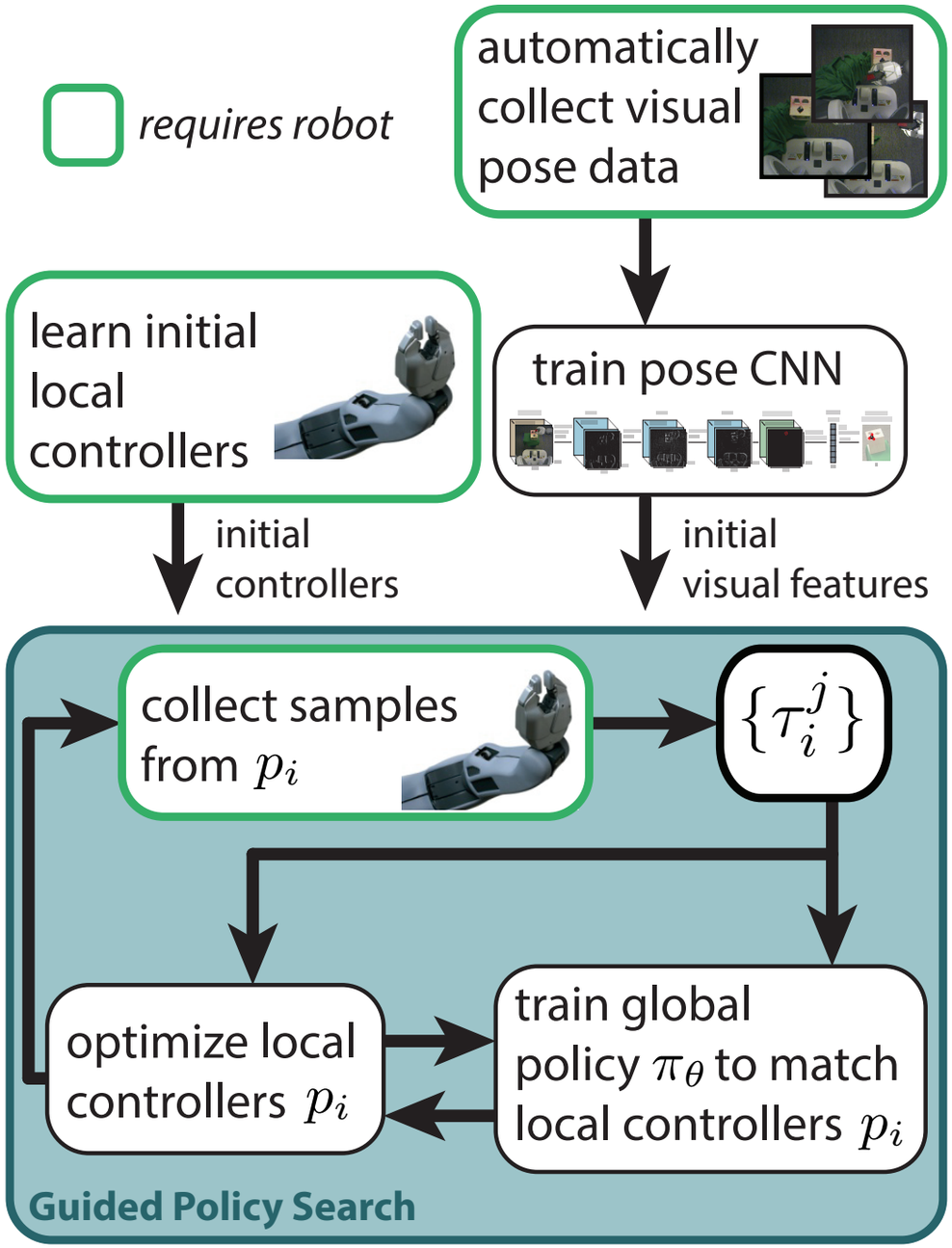
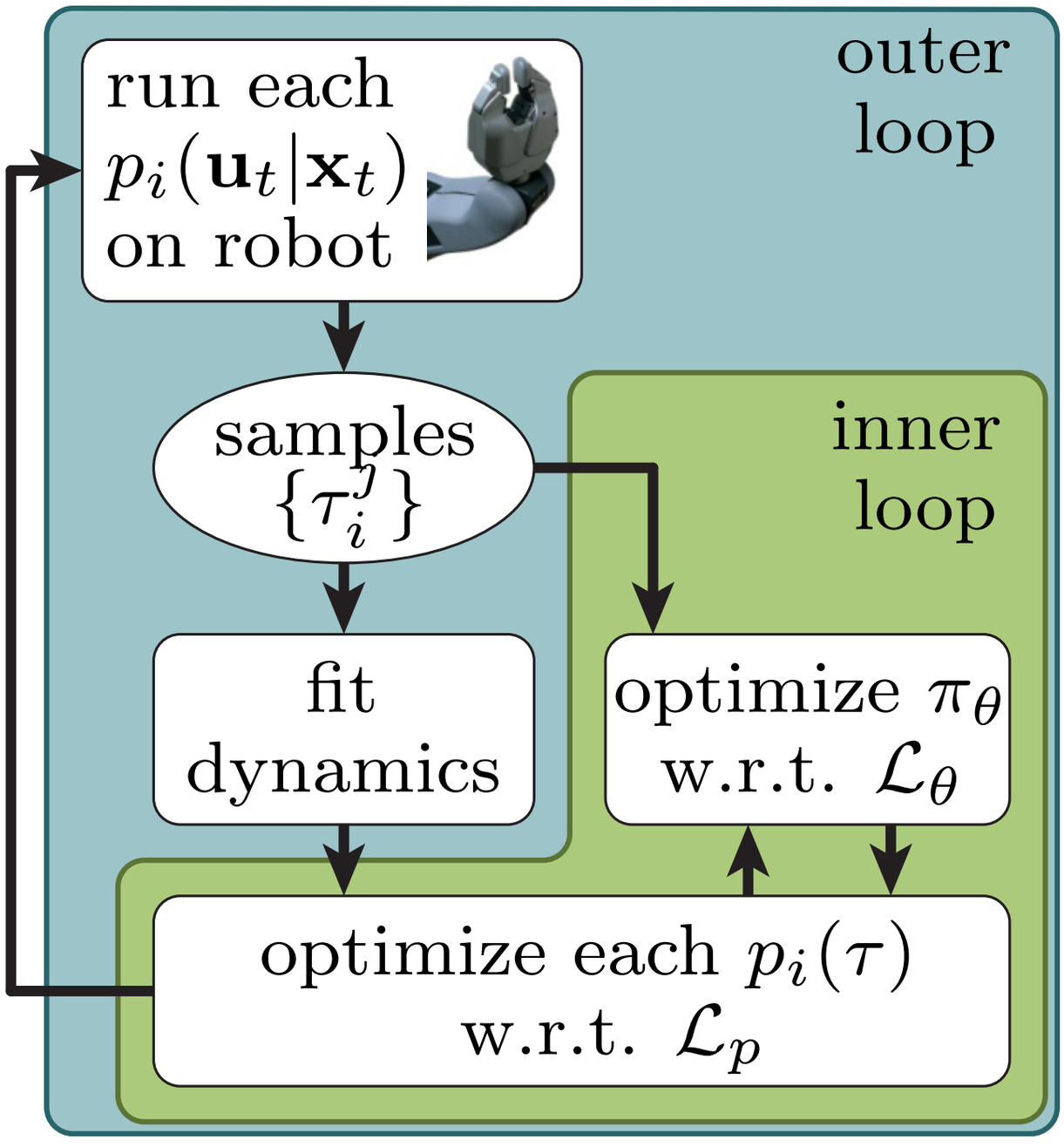
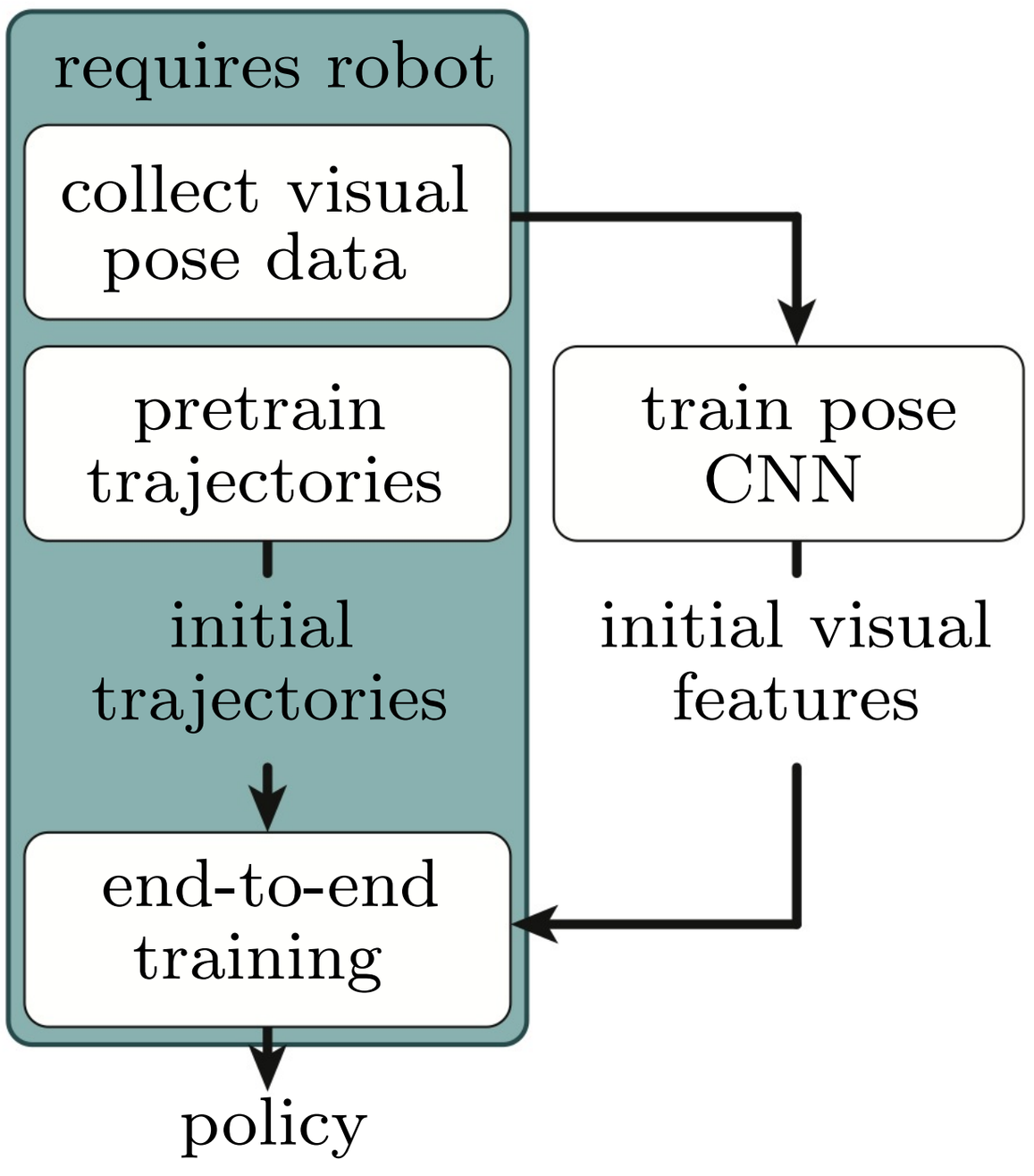


Figure 3: Diagram of our approach, including the main guided policy search phase and initialization phases.

Training Process





Training

- Full objective can be broken down into sub-problems
- $$\min_{p,\pi_\theta} E_p[\ell(\tau)] \text{ s.t. } p(\mathbf{u}_t|\mathbf{x}_t) = \pi_\theta(\mathbf{u}_t|\mathbf{x}_t) \; \forall \, \mathbf{x}_t, \mathbf{u}_t, t,$$
- Trajectory Optimization under Unknown Dynamics
  - Fit linear gaussians to dynamics
  - Constrain new trajectory to be close to current trajectory
- $$\min_{p(\tau)\in\mathcal{N}(\tau)} \mathcal{L}_p(p,\theta) \text{ s.t. } D_{\mathrm{KL}}(p(\tau)\|\hat{p}(\tau)) \leq \epsilon.$$
- Supervised Policy Optimization
  - Minimize KL divergence
- $$\mathcal{L}_\theta(\theta,p) = \frac{1}{2N}\sum_{i=1}^{N}\sum_{t=1}^{T} E_{p_i(\mathbf{x}_t,\mathbf{o}_t)}\left[\mathrm{tr}[\mathbf{C}_{ti}^{-1}\Sigma^\pi(\mathbf{o}_t)] - \log|\Sigma^\pi(\mathbf{o}_t)|\right.$$
$$\left. + (\mu^\pi(\mathbf{o}_t) - \mu_{ti}^p(\mathbf{x}_t))\mathbf{C}_{ti}^{-1}(\mu^\pi(\mathbf{o}_t) - \mu_{ti}^p(\mathbf{x}_t)) + 2\lambda_{\mu t}^{\mathrm{T}}\mu^\pi(\mathbf{o}_t)\right],$$

Tasks (from states)

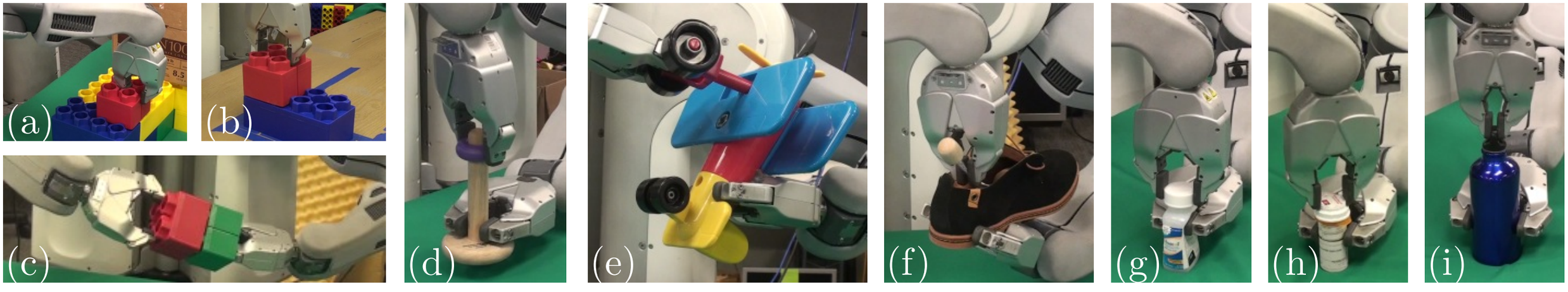
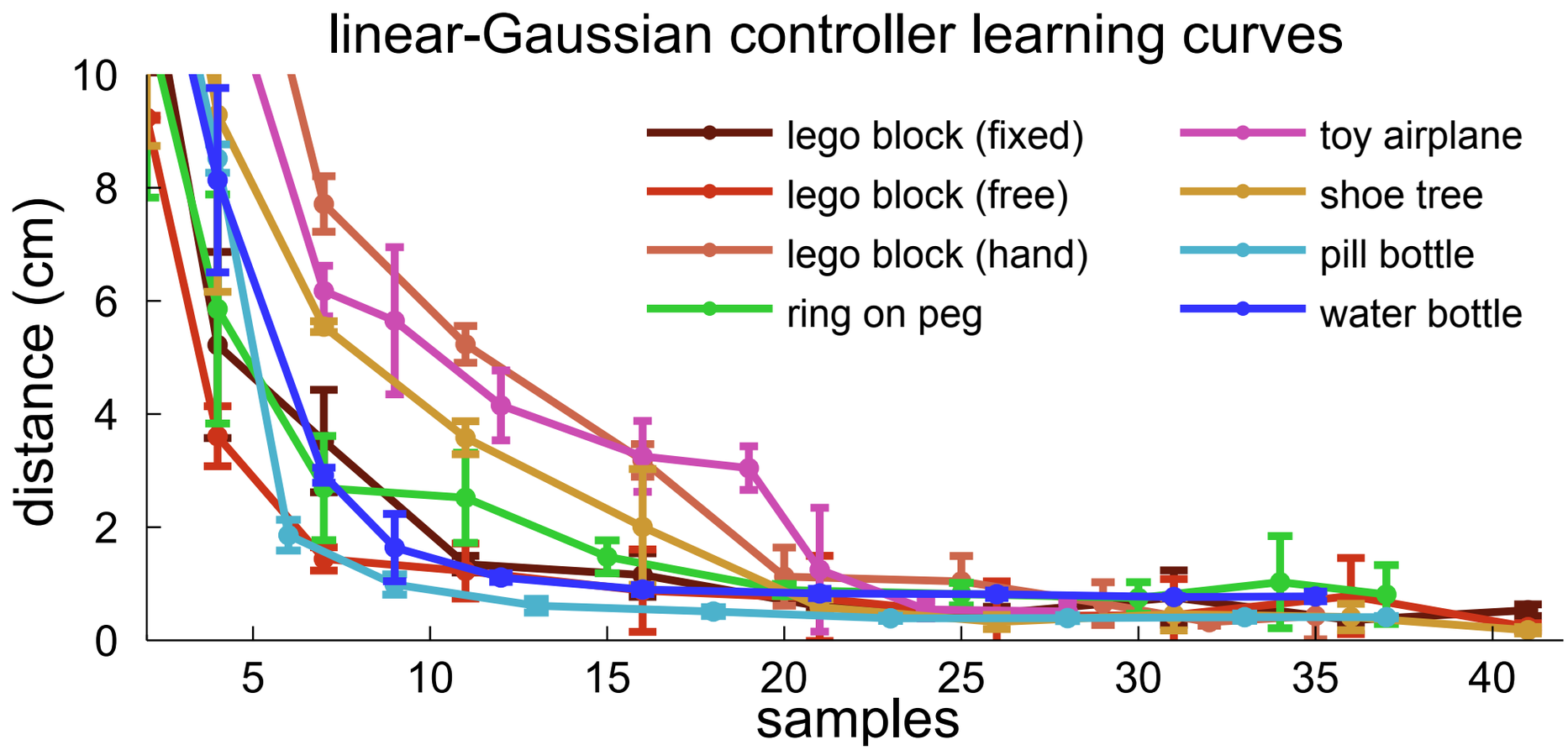


Figure 6: Tasks for linear-Gaussian controller evaluation: (a) stacking lego blocks on a fixed base, (b) onto a free-standing block, (c) held in both gripper; (d) threading wooden rings onto a peg; (e) attaching the wheels to a toy airplane; (f) inserting a shoe tree into a shoe; (g,h) screwing caps onto pill bottles and (i) onto a water bottle.



Visuo-motor control from pixels

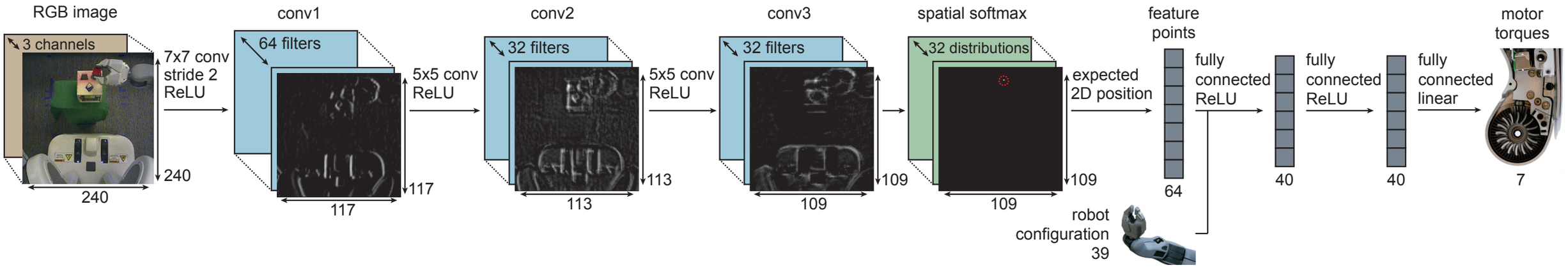
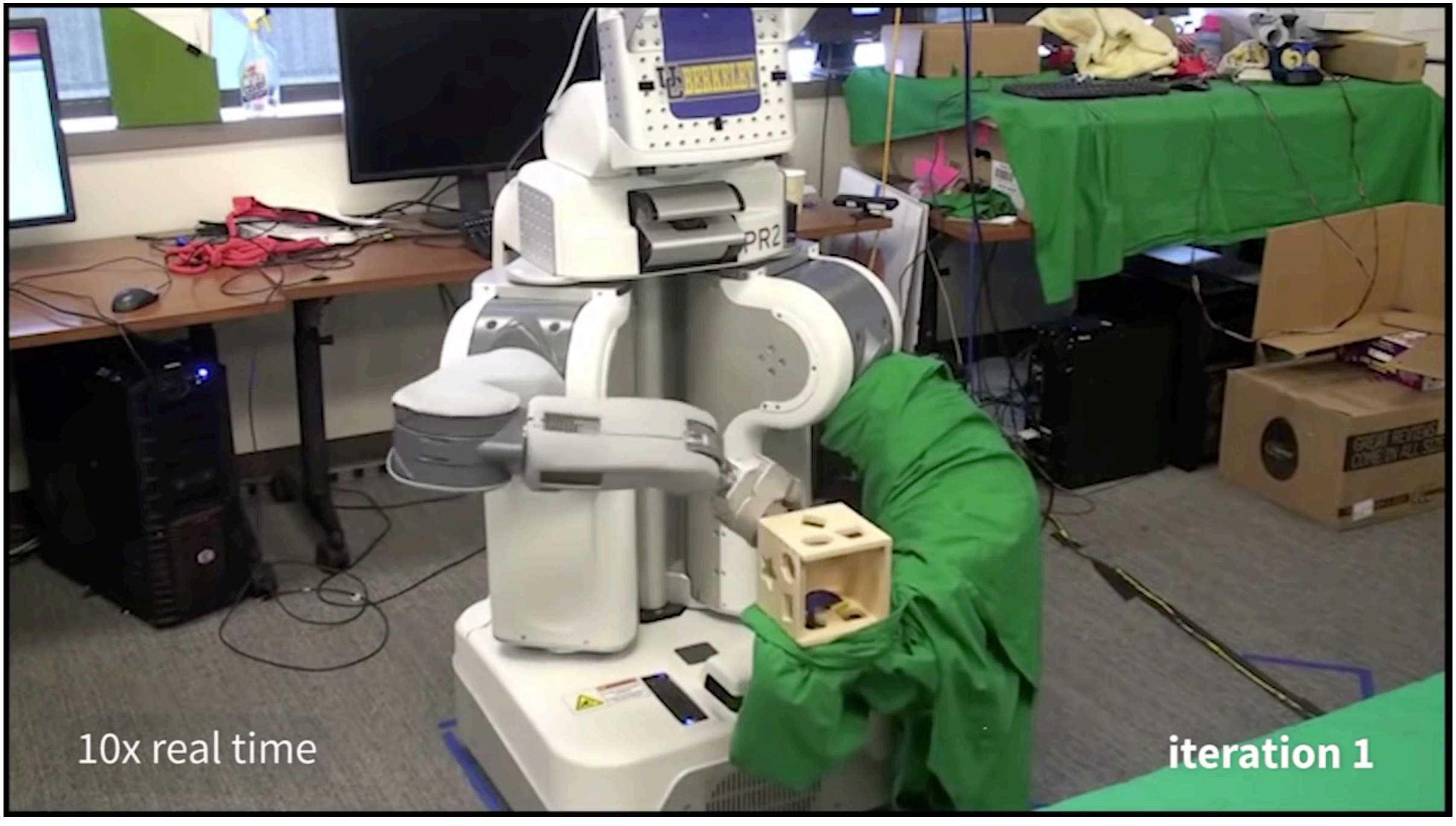


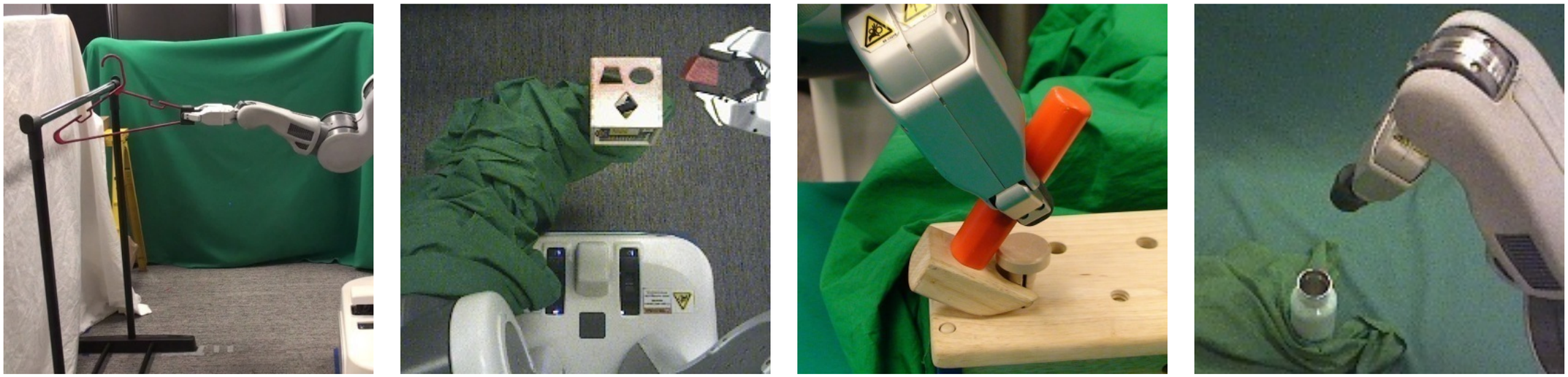
Figure 2: Visuomotor policy architecture. The network contains three convolutional layers, followed by a spatial softmax and an expected position layer that converts pixel-wise features to feature points, which are better suited for spatial computations. The points are concatenated with the robot configuration, then passed through three fully connected layers to produce the torques.

- Directly outputting torques
- Spatial softmax layer

Tasks (from vision)

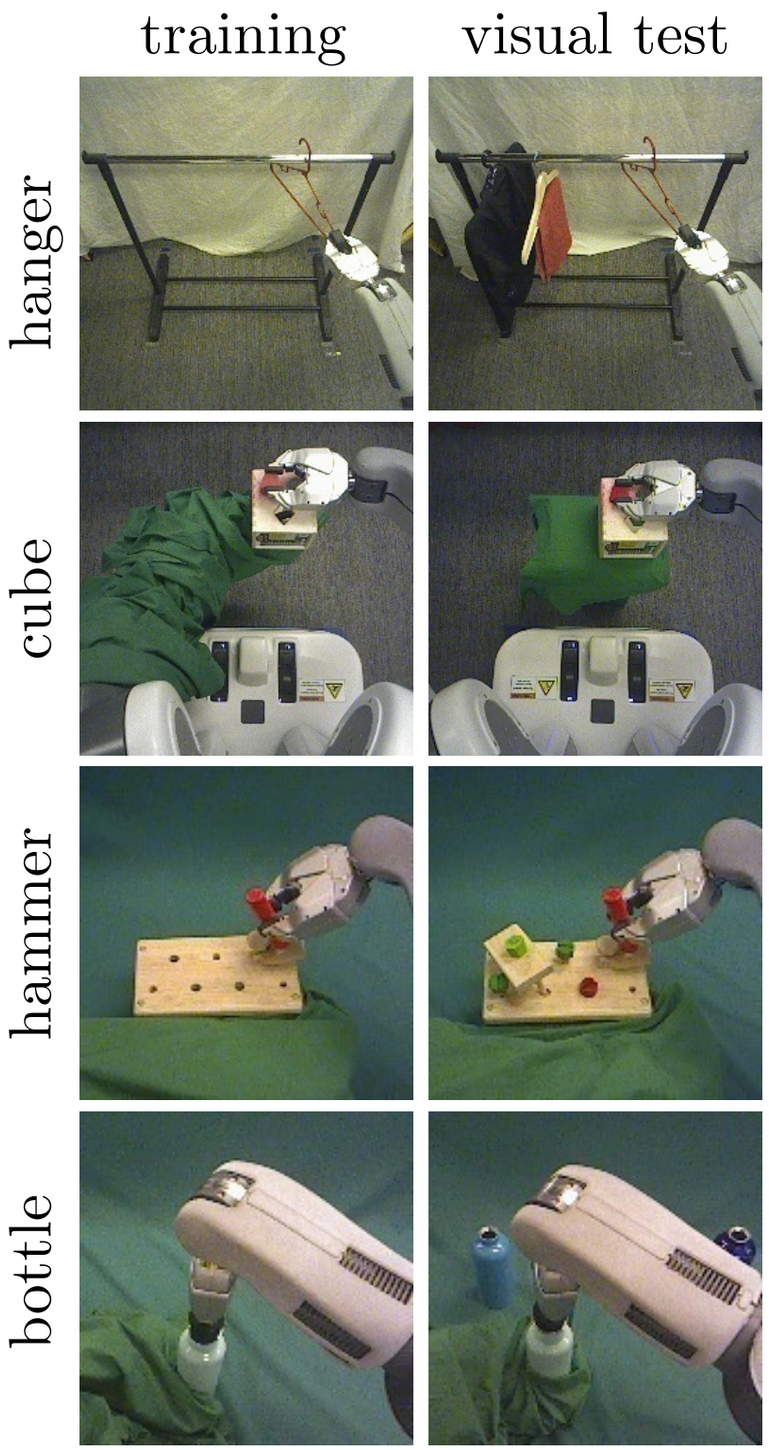


Tasks (from vision)



(a) hanger (b) cube (c) hammer (d) bottle

Performance



coat hanger	training (18)	spatial test (24)	visual test (18)
end-to-end	**100%**	**100%**	**100%**
pose features	88.9%	87.5%	83.3%
pose prediction	55.6%	58.3%	66.7%

shape cube	training (27)	spatial test (36)	visual test (40)
end-to-end	**96.3%**	**91.7%**	**87.5%**
pose features	70.4%	83.3%	40%
pose prediction	0%	0%	n/a

toy hammer	training (45)	spatial test (60)	visual test (60)
end-to-end	**91.1%**	**86.7%**	**78.3%**
pose features	62.2%	75.0%	53.3%
pose prediction	8.9%	18.3%	n/a

bottle cap	training (27)	spatial test (12)	visual test (40)
end-to-end	**88.9%**	**83.3%**	**62.5%**
pose features	55.6%	58.3%	27.5%

Success rates on training positions, on novel test positions, and in the presence of visual distractors. The number of trials per test is shown in parentheses.

Figure 9: Training and visual test scenes as seen by the policy (left), and experimental results (right). The hammer and bottle images were cropped for visualization only.

network architecture	test error (cm)
softmax + feature points (**ours**)	**1.30 ± 0.73**

Sample Efficiency

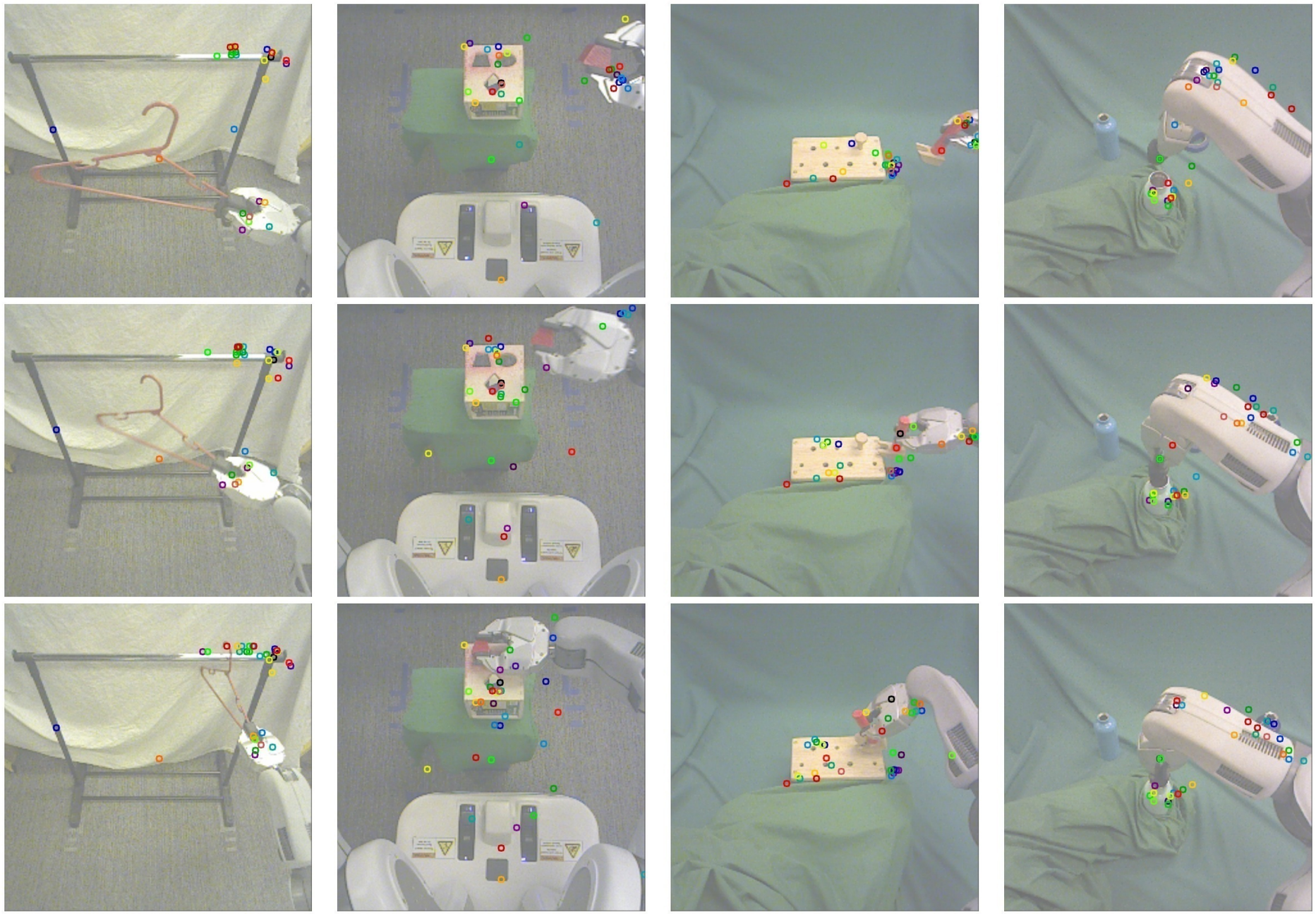
	number of trials		
task	trajectory pretraining	end-to-end training	total
coat hanger	120	36	156
shape cube	90	81	171
toy hammer	150	90	240
bottle cap	180	108	288

Table 4: Total number of trials used for learning each visuomotor policy.

Network Design Ablation

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

Network Visualization



(a) hanger (b) cube (c) hammer (d) bottle

Figure 10: Feature points tracked by the policy during task execution for each of the four tasks. Each feature point is displayed in a different random color, with consistent coloring across images. The policy finds features on the target object and the robot gripper and arm. In the bottle cap task, note that the policy correctly ignores the distractor bottle in the background, even though it was not present during training.

Thank you