

Deep Learning for Manipulation and Navigation

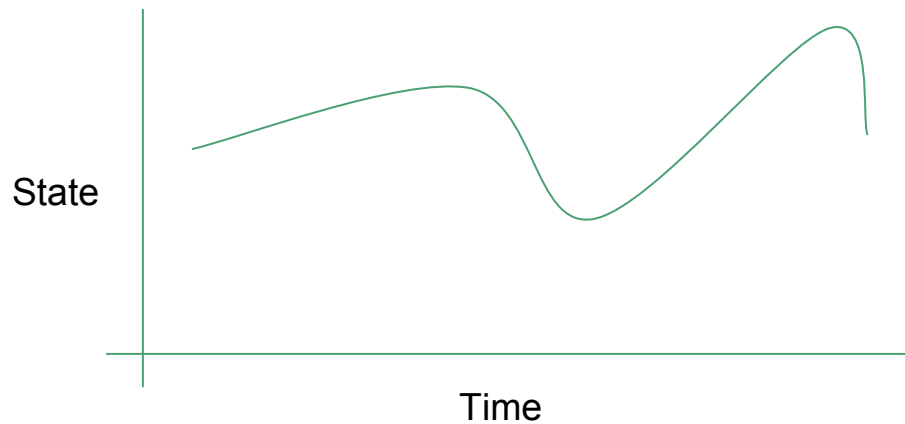
Andrey Zaytsev and Tanmay Gangwani

Outline

- Learning from Demonstrations
 - Imitation Learning
 - Optimal Control and Planning
- Manipulation and Navigation
 - Learning using Physical Interactions
 - Navigation using Auxiliary Supervision

Notations

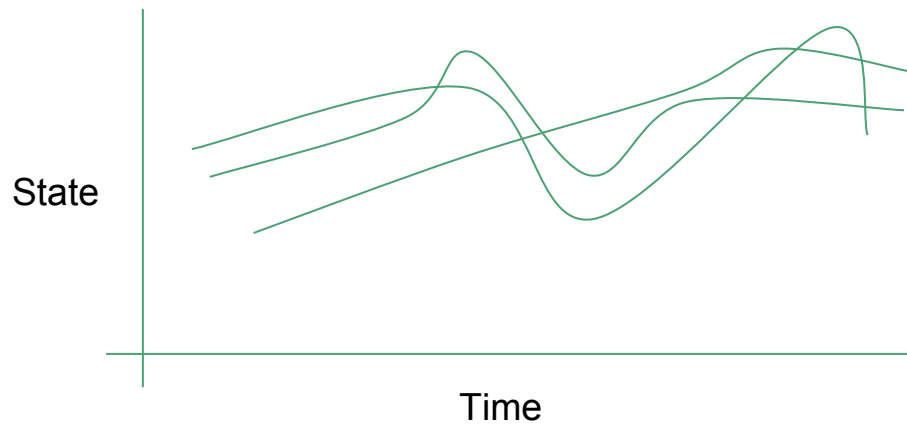
Trajectory



$$\tau = \{x_1, u_1, x_2, u_2, \dots, x_T, u_T\}$$

Notations

Trajectory Distribution

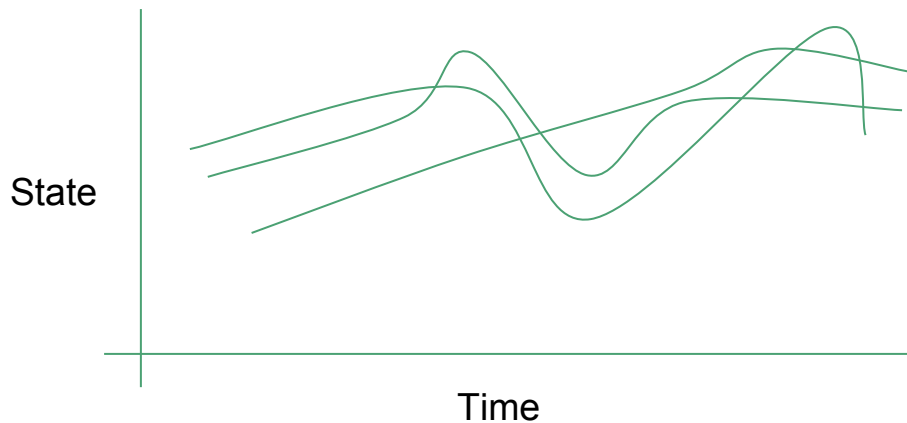


**A few samples from
the distribution**

$$p(\tau) = p(x_1, u_1, x_2, u_2, \dots, x_T, u_T)$$

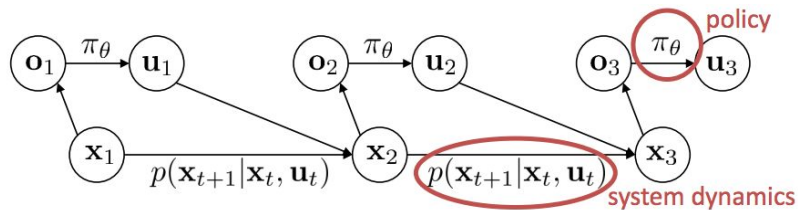
Notations

Trajectory Distribution



$$p(\tau) = p(x_1, u_1, x_2, u_2, \dots, x_T, u_T)$$

MDP

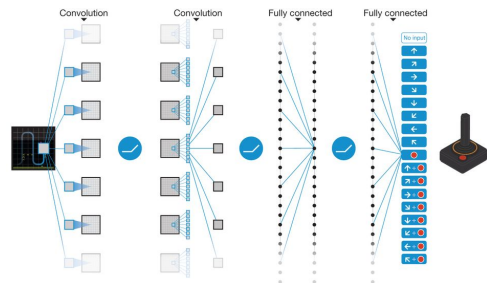


$$p(\tau) = p(x_1) \prod_{t=1}^T \pi_\theta(u_t|x_t) p(x_{t+1}|x_t, u_t)$$

RL with Rewards

- Policy gradient, Q-learning depend on **frequent rewards**

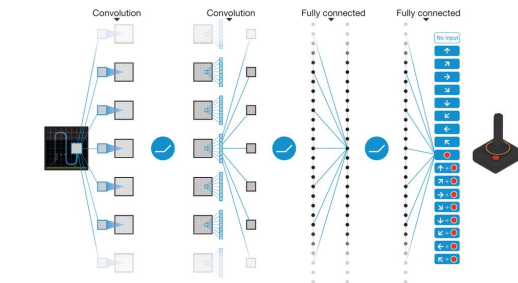
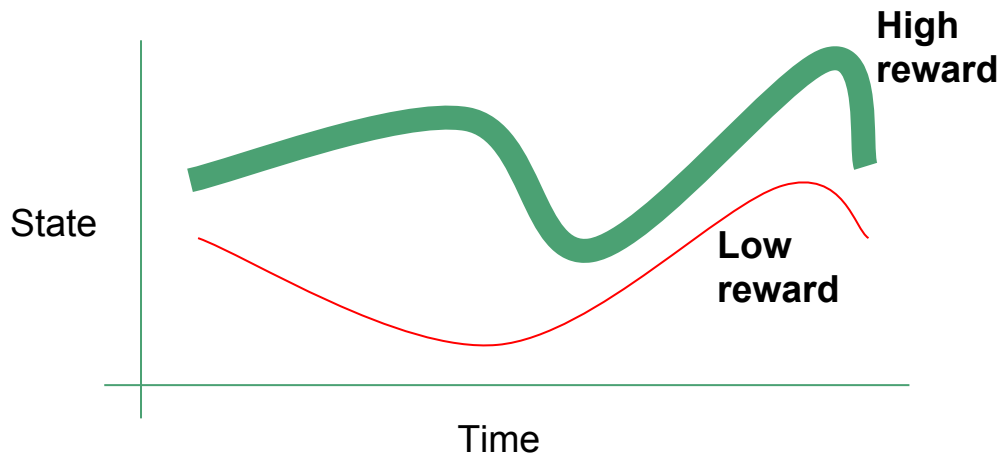
$$\begin{aligned}\nabla_{\theta} J(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(s, a) \nabla_{\theta} \log \pi_{\theta}(s, a) \mathcal{R}_{s,a} \\ &= \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) r]\end{aligned}$$



RL with Rewards

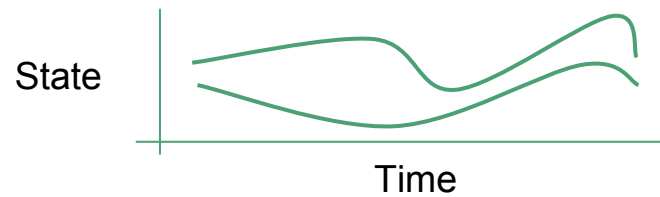
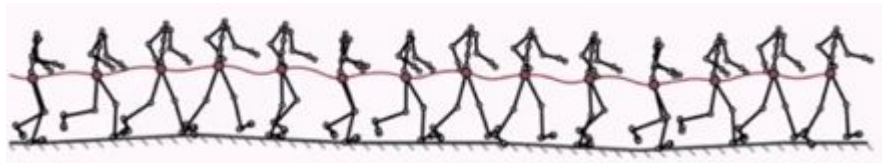
- Policy gradient, Q-learning depend on **frequent rewards**

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(s, a) \nabla_{\theta} \log \pi_{\theta}(s, a) \mathcal{R}_{s,a} \\ &= \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) r]\end{aligned}$$

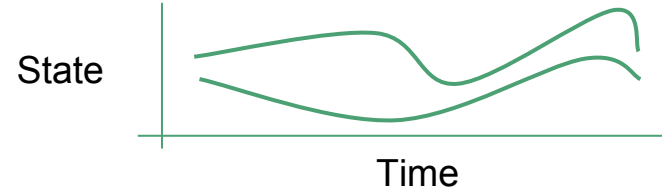


Rollouts, random initially!

Sparse Rewards

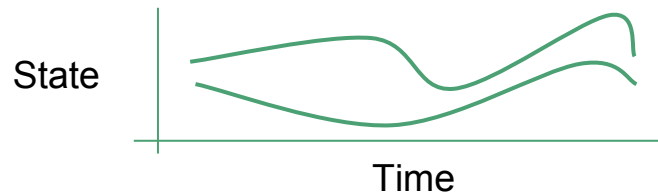
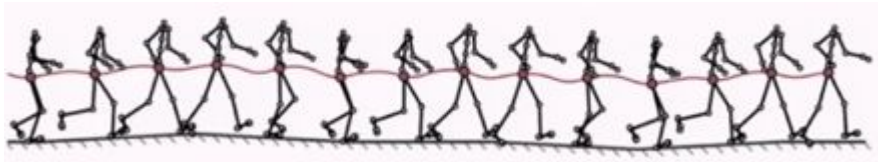


Sparse Rewards



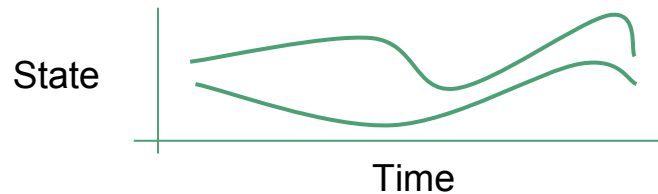
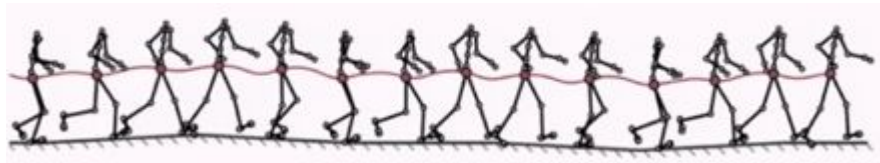
- High dimensional policy
 - Most random trajectories **don't yield positive reward**

Sparse Rewards



- High dimensional policy
 - Most random trajectories **don't yield positive reward**
- May be expensive to evaluate action on physical system
- Failure may not be an option

Sparse Rewards

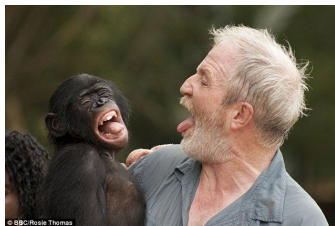


- High dimensional policy
 - Most random trajectories **don't yield**
- May be expensive to evaluate action on ph
- Failure may not be an option

Use an expert (teacher) to guide the learning process!

Learning from demonstrations!

Learning from Demonstrations

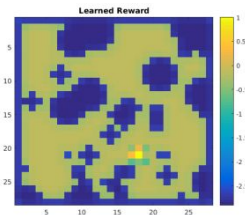
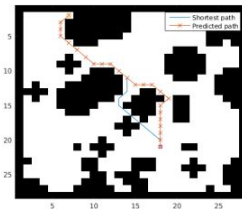


Imitation Learning

- Directly copy the expert
- Supervised learning



This talk!



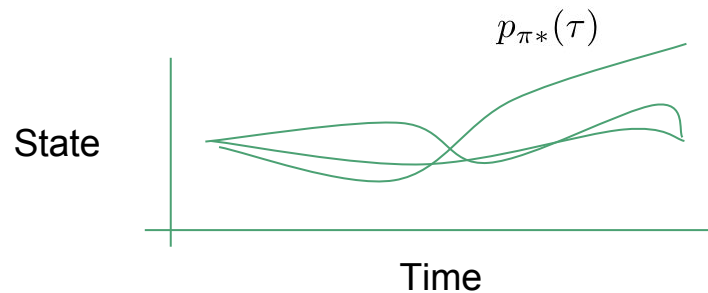
Inverse RL

- Infer the goal of the expert
- Learn the reward function r
- Learn optimal policy under r

Expert Guidance

- Expert provides trajectories from a **good** trajectory distribution

$$p_{\pi_*}(\tau) = \arg \max_{p(\tau)} E_p[r(\tau)]$$



Expert Guidance

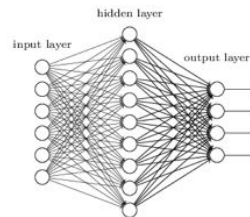
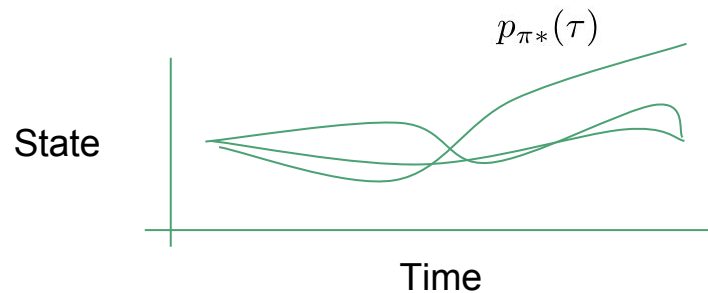
- Expert provides trajectories from a **good** trajectory distribution

$$p_{\pi_*}(\tau) = \arg \max_{p(\tau)} E_p[r(\tau)]$$

- Learner imitates the trajectories - **supervised learning**

$$\tau = \{x_1, u_1, x_2, u_2, \dots, x_T, u_T\}$$

$$L = - \sum_i \log p(\mathbf{u}(x_i) = \pi_*(x_i) | x_i)$$



Expert Guidance

- Expert provides trajectories from a **good** trajectory distribution

$$p_{\pi_*}(\tau) = \arg \max_{p(\tau)} E_p[r(\tau)]$$

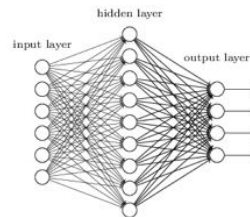
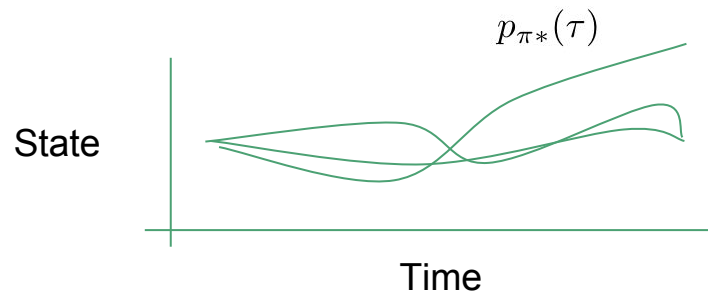
- Learner imitates the trajectories - **supervised learning**

$$\tau = \{x_1, u_1, x_2, u_2, \dots, x_T, u_T\}$$

$$L = - \sum_i \log p(\mathbf{u}(x_i) = \pi_*(x_i) | x_i)$$

- Policy should produce trajectory distribution close to expert's

$$D_{KL}(p_{\pi_*}(\tau) || p_{\pi_\theta}(\tau)) < \epsilon$$

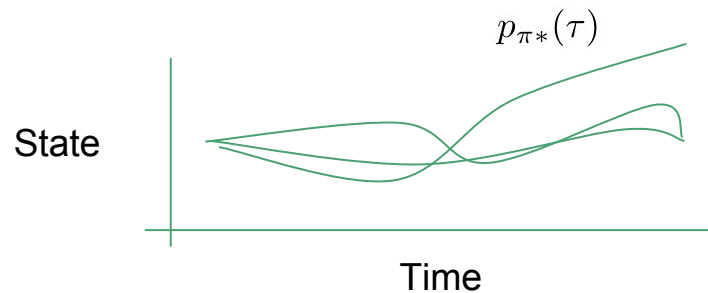


Expert Guidance

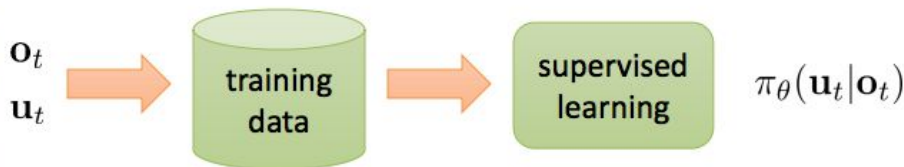
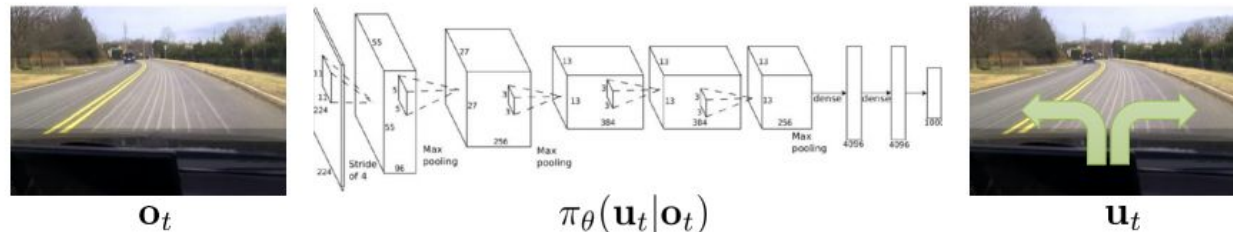
- Expert provides trajectories from a **good** trajectory distribution

$$p_{\pi_*}(\tau) = \arg \max_{p(\tau)} E_p[r(\tau)]$$

- Who's an expert ??
 - Clone demonstrations shown by humans
 - Machine provides demonstrations
 - Optimal control / planning / trajectory optimization



Imitation Learning for Driving

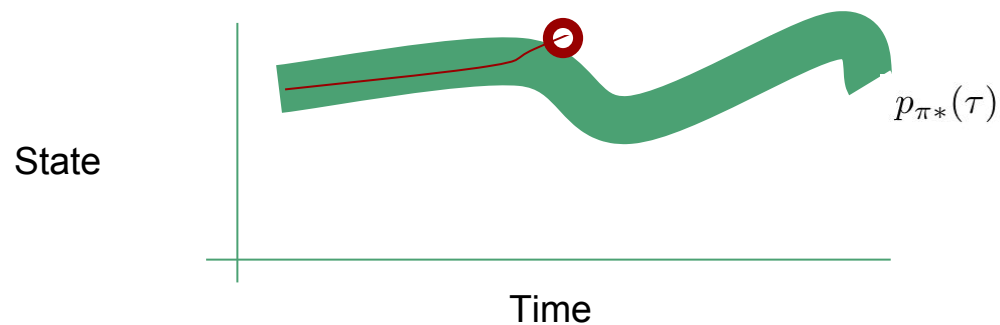


Stochastic policy for predicting steering wheel angle from observations

Missing Supervision

$p_{\pi_*}(\tau)$: On-road trajectories by the expert

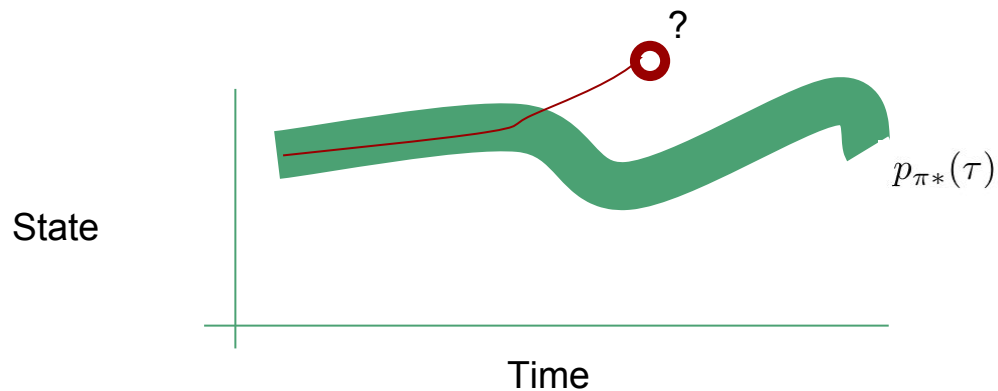
$p_{\pi_\theta}(\tau)$: Trajectories by the autonomous vehicle



Missing Supervision

$p_{\pi_*}(\tau)$: On-road trajectories by the expert

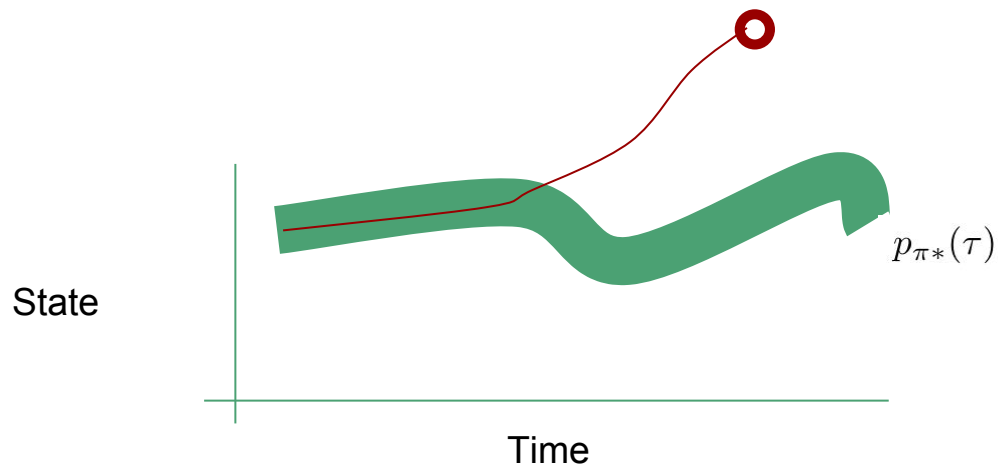
$p_{\pi_\theta}(\tau)$: Trajectories by the autonomous vehicle



Missing Supervision

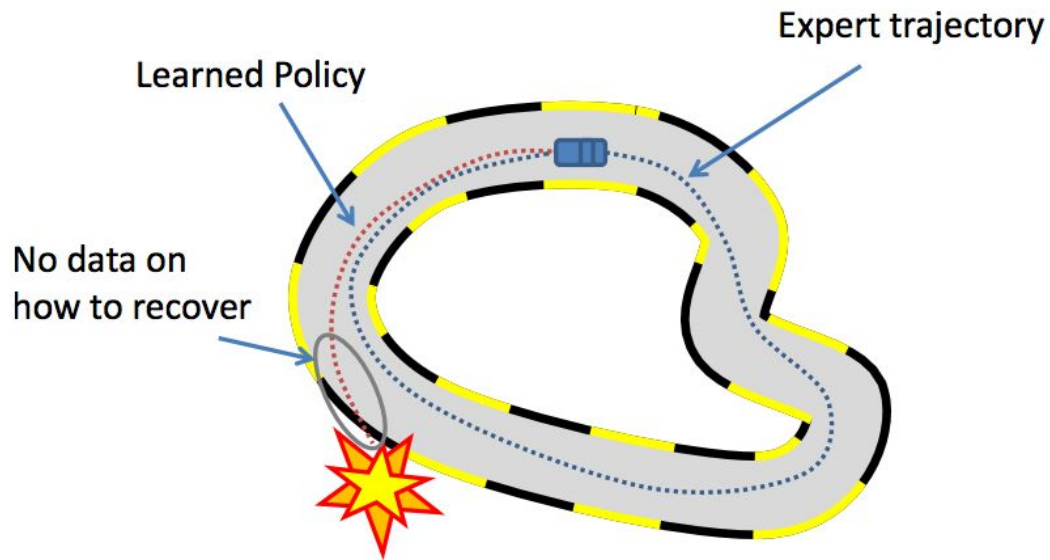
$p_{\pi_*}(\tau)$: On-road trajectories by the expert

$p_{\pi_\theta}(\tau)$: Trajectories by the autonomous vehicle



Compounding Errors!

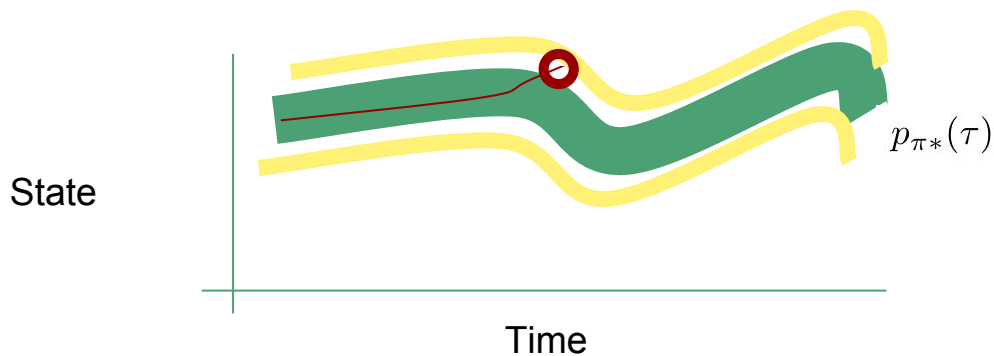
Trajectory Distribution Mismatch



$$p_{\pi^*}(\tau) \neq p_{\pi_\theta}(\tau)$$

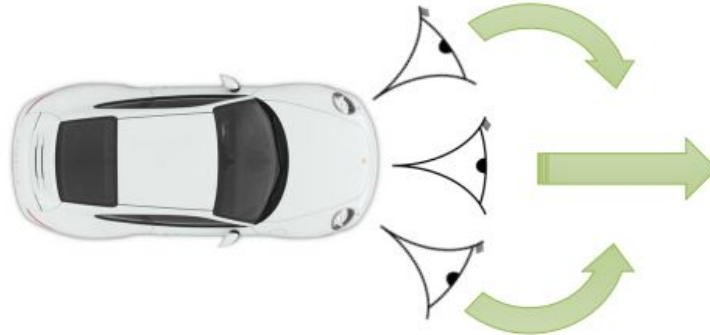
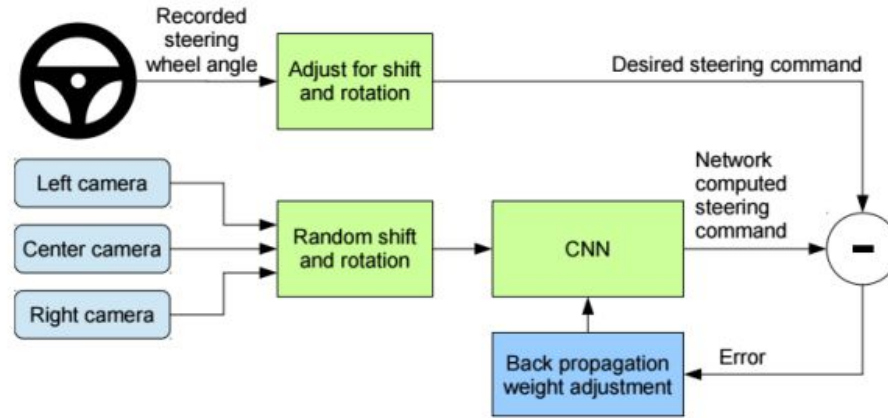
A Hacky Solution

Demonstration Augmentation - **Include extra supervision** in expert trajectories for states that the policy is likely to visit during test time



A Hacky Solution

Labelled data from left and right camera to recover from mistakes



Imitation Learning for Driving



End to End Learning for Self-Driving Cars

A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots

Alessandro Giusti¹, Jérôme Guzzi¹, Dan C. Cireşan¹, Fang-Lin He¹, Juan P. Rodríguez¹
Flavio Fontana², Matthias Faessler², Christian Forster²
Jürgen Schmidhuber¹, Gianni Di Caro¹, Davide Scaramuzza², Luca M. Gambardella¹



DAgger

Solving missing supervision or the trajectory distribution mismatch problem using on-policy data


Algorithm:

1. train $\pi_{\theta}(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \dots, \mathbf{o}_N, \mathbf{u}_N\}$

DAgger

Solving missing supervision or the trajectory distribution mismatch problem **using on-policy data**

Algorithm:

1. train $\pi_{\theta}(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \dots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\pi_{\theta}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_{\pi} = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$  **Throw out the actions!**

Dagger

Solving missing supervision or the trajectory distribution mismatch problem **using on-policy data**


Algorithm:

1. train $\pi_{\theta}(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \dots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\pi_{\theta}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_{\pi} = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask human to label \mathcal{D}_{π} with actions \mathbf{u}_t

DAgger

Solving missing supervision or the trajectory distribution mismatch problem **using on-policy data**

Algorithm:

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

DAgger in Practice

Predict a steering angle, given RGB images from drone camera



[Learning Monocular Reactive UAV Control in Cluttered Natural Environments](#)

Imitation with Human Expert

Who's an expert ??

- Clone demonstrations shown by humans
 - Unnatural / hard in some cases (e.g. steering angle from images)
 - Not scalable - continuous improvements not possible



Imitation with Human Expert

Who's an expert ??

- Clone demonstrations shown by humans
 - Unnatural / hard in some cases (e.g. steering angle from images)
 - Not scalable - continuous improvements not possible
- **Machine provides demonstrations**
 - **Optimal control / planning / trajectory optimization**



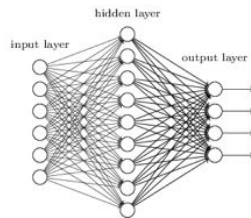
Imitating Optimal Control

- Expert provides trajectories from a **good** trajectory distribution

$$p_{\pi_*}(\tau) = \arg \max_{p(\tau)} E_p[r(\tau)]$$



Supervised Learning



$$\pi_{\theta}(\mathbf{u}_t | \mathbf{o}_t)$$

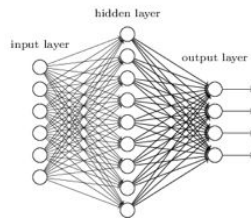
Imitating Optimal Control

- Expert provides trajectories from a **good** trajectory distribution

$$p_{\pi^*}(\tau) = \arg \max_{p(\tau)} E_p[r(\tau)]$$



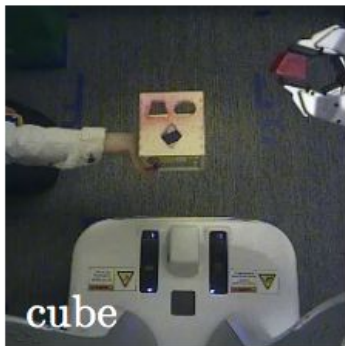
Supervised Learning



$$\pi_{\theta}(\mathbf{u}_t | \mathbf{o}_t)$$

Imitating Optimal Control

- Learn policies for various robotics tasks using only camera images
- Use **Guided Policy Search** as expert

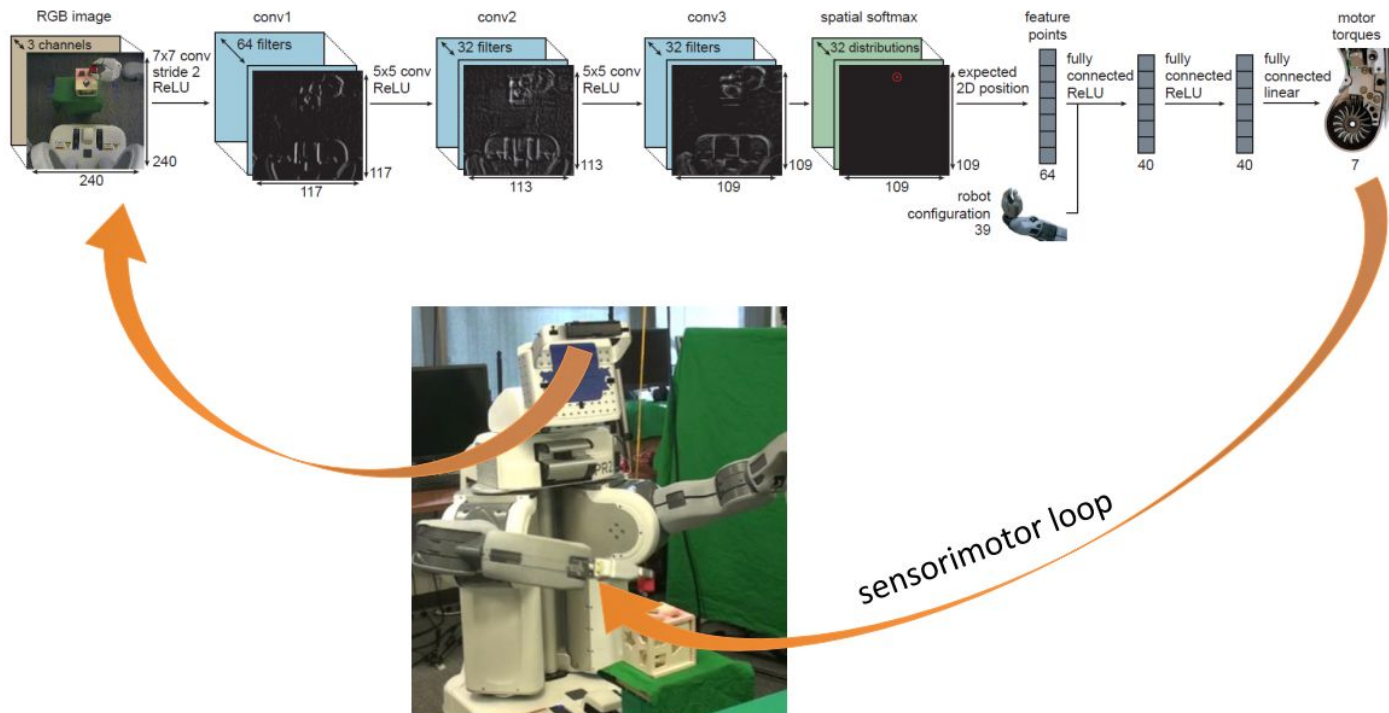


Imitating GPS



End-to-End Training of Deep Visuomotor Policies

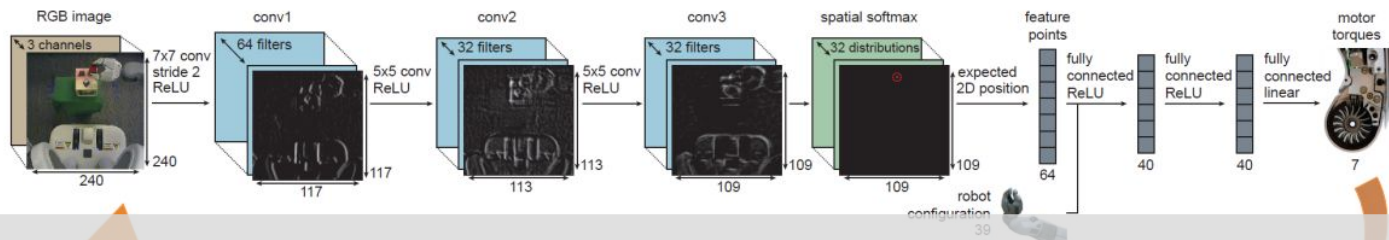
Neural Net Architecture



End-to-End Training of Deep Visuomotor Policies

Vision layers to localize target and end-effector

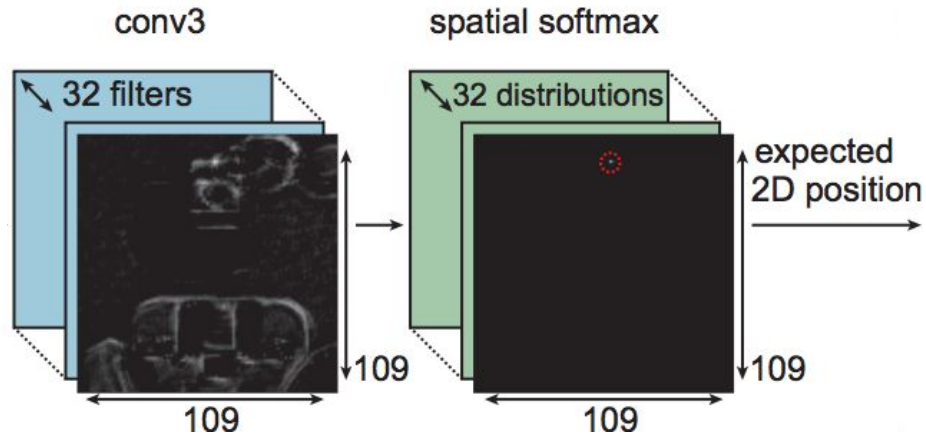
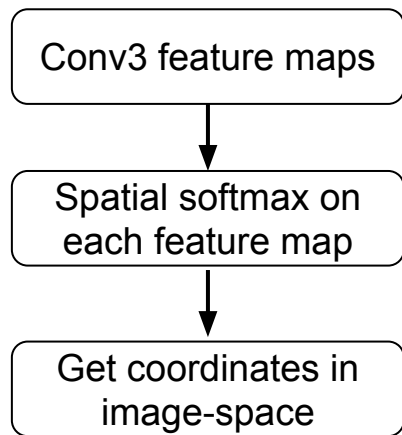
Policy layers with expert supervision



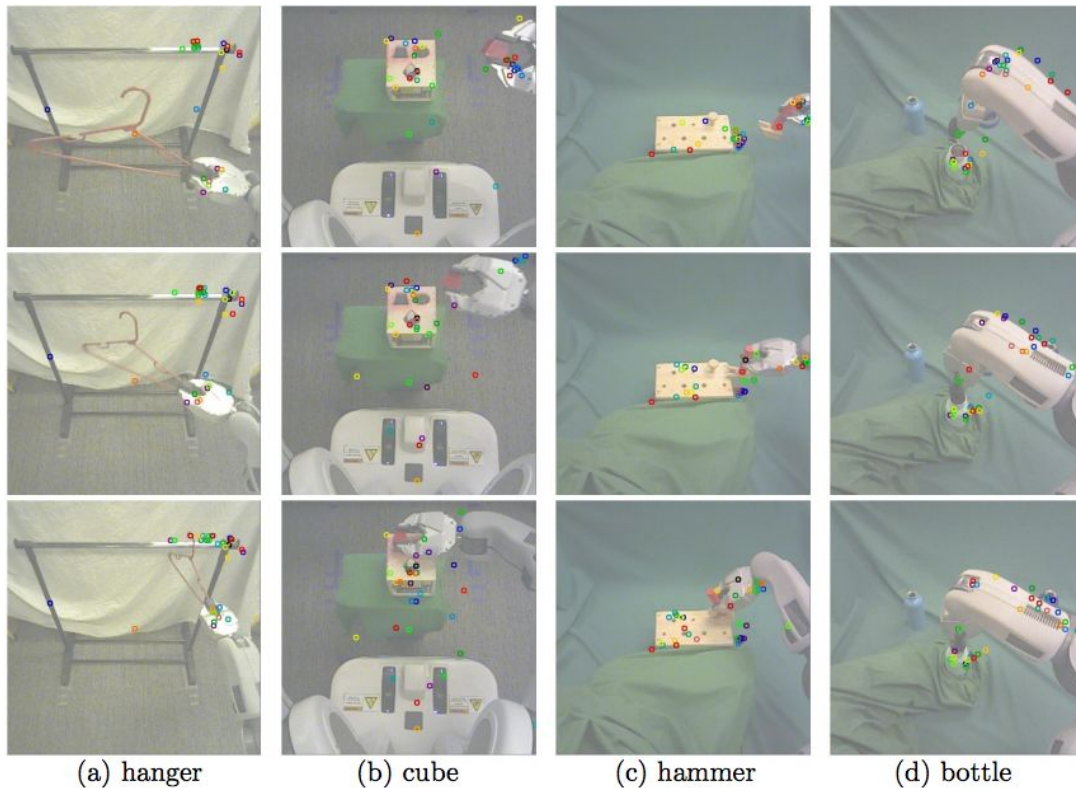
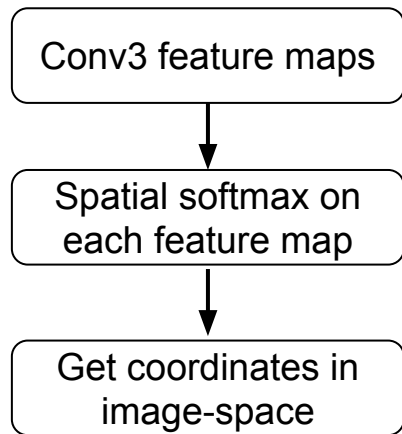
sensorimotor loop

End-to-End Training of Deep Visuomotor Policies

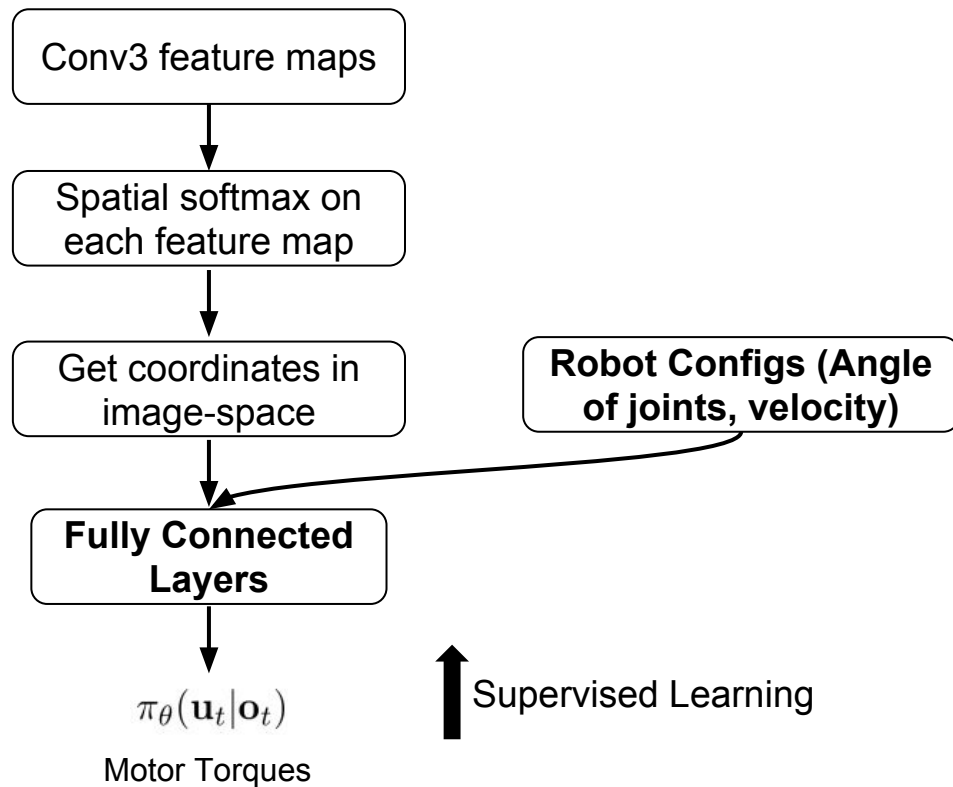
Localize Target and End-effector



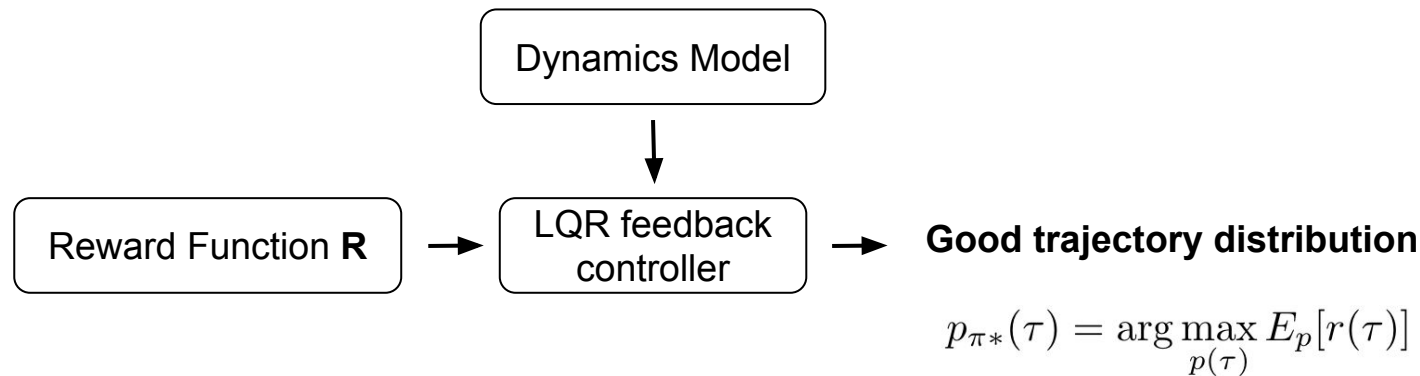
Localize Target and End-effector



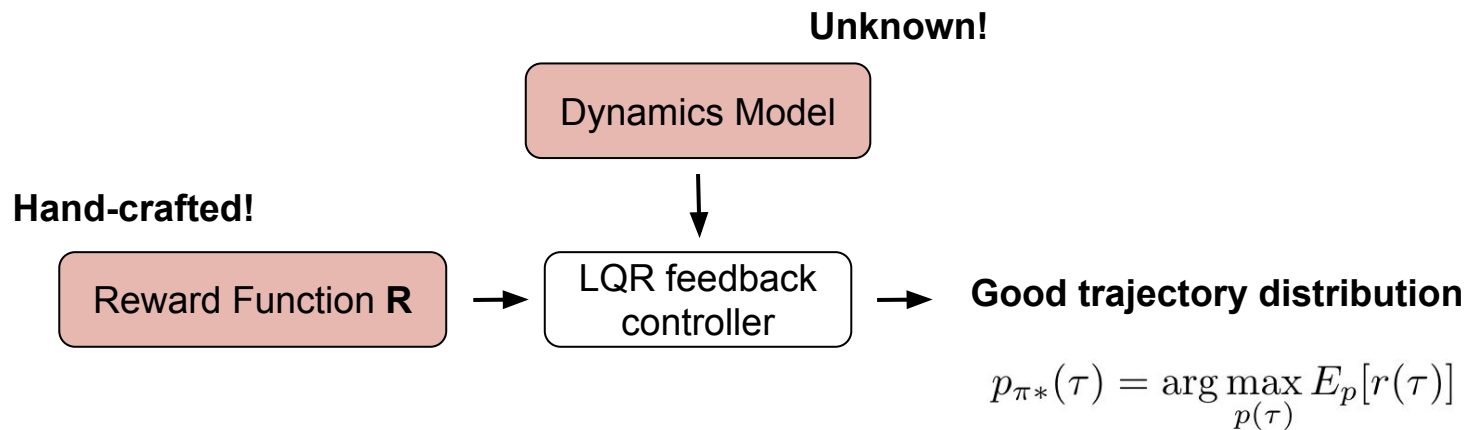
End-to-End Training



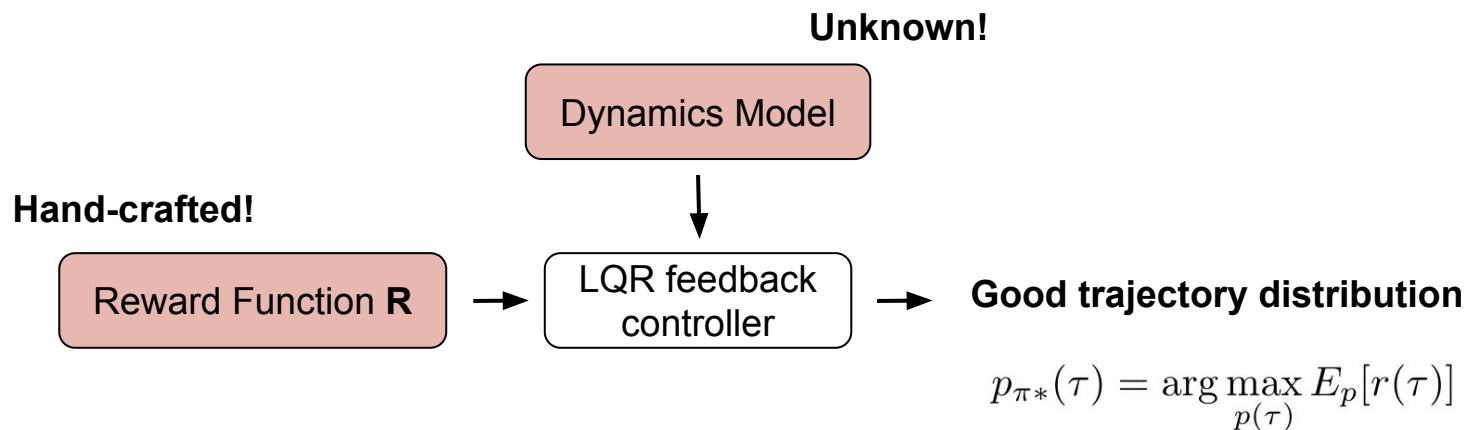
Guided Policy Search (Teacher) - 10k feet view



GPS Challenges

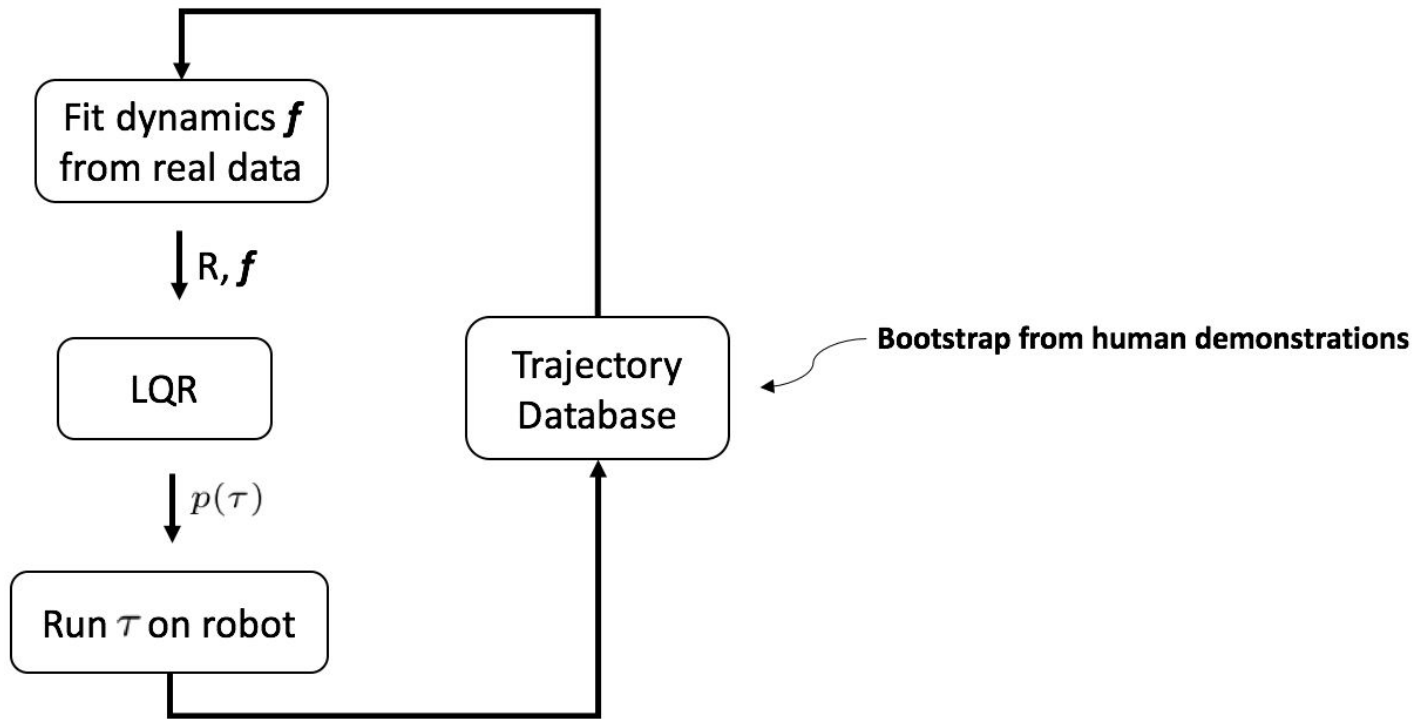


GPS Challenges

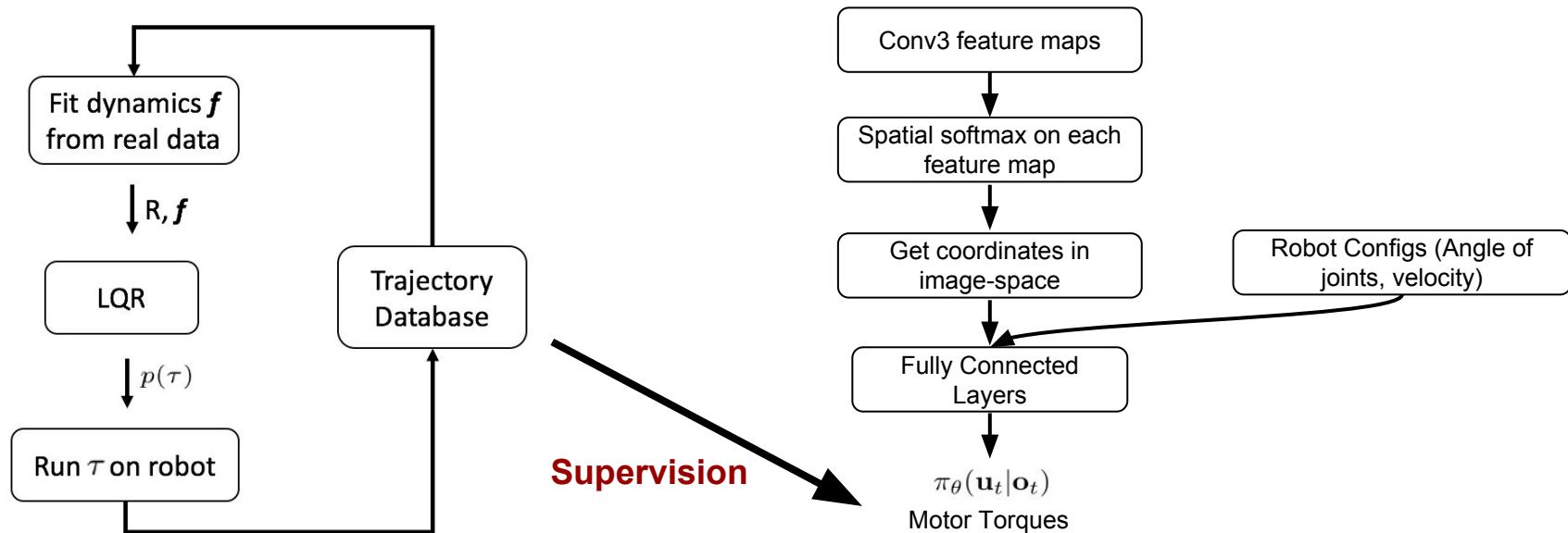


LQR controller - approach similar to value iteration / dynamic programming!

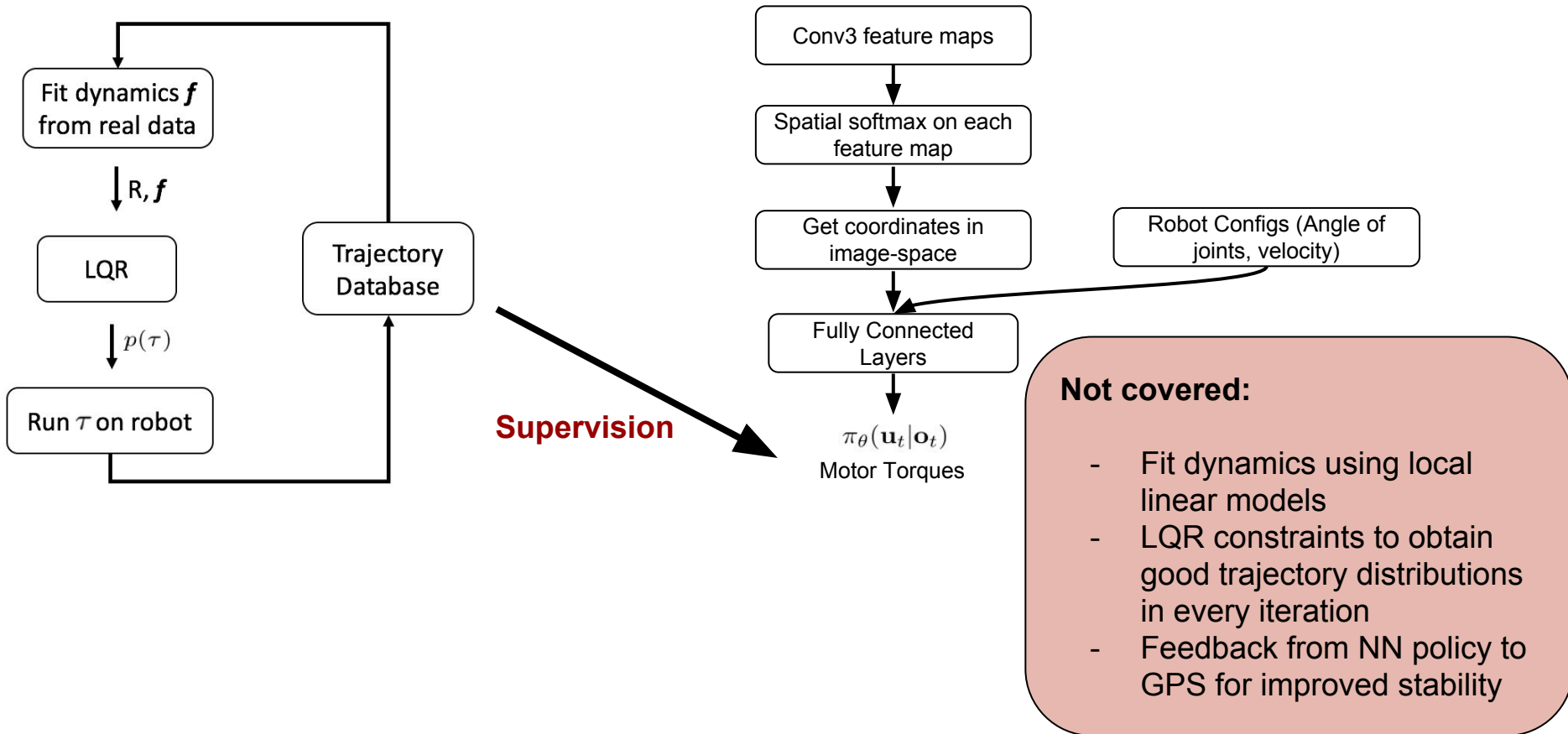
GPS loop



End-to-End Training with GPS



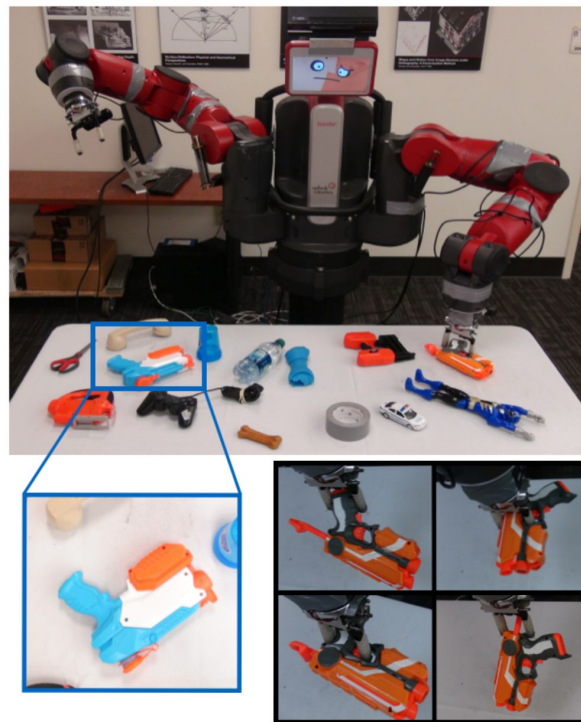
End-to-End Training with GPS



Manipulation and Navigation Overview

Manipulation

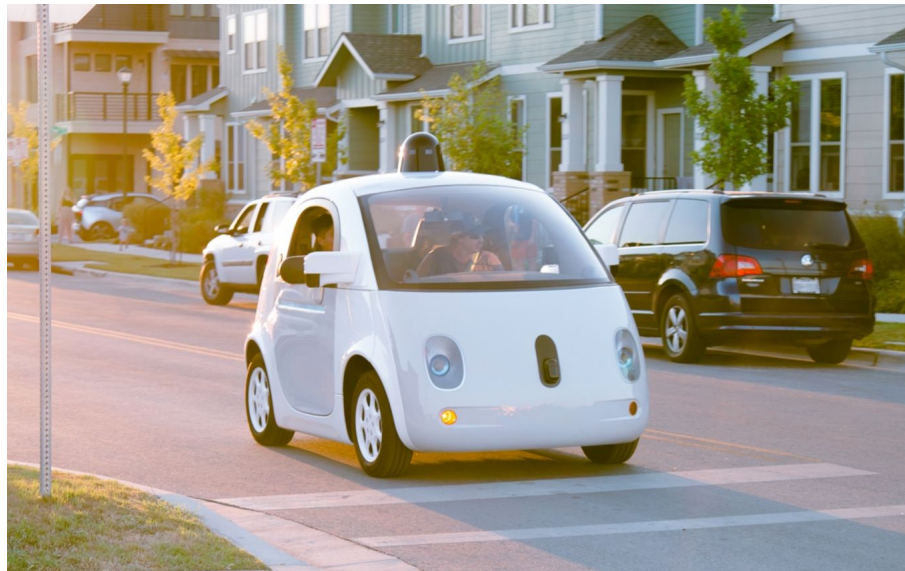
- Tasks
 - Grasping
 - Pushing
 - Poking
 - Tactile sensing
- Pose invariance
- Hand-eye coordination



[Image taken from Supersizing Self-supervision by Pinto et. al., arXiv 2015](#)

Navigation

- Self-driving cars
- Flying robots, quadcopters
- Tasks
 - Collision avoidance
 - Navigation in mazes
 - Dynamic environments
 - Reinforcement Learning



[Image taken from Waymo](#)

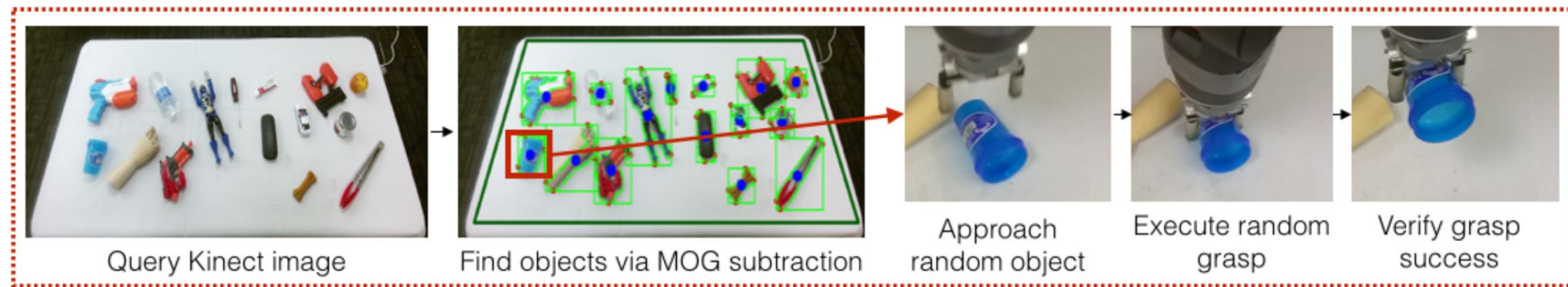
Manipulation

Grasping — Setup

- Robot with arms and camera
- Can control grasp angle and position
- No human interaction besides placing objects
- Given an image want to be able to predict a successful grasping configuration

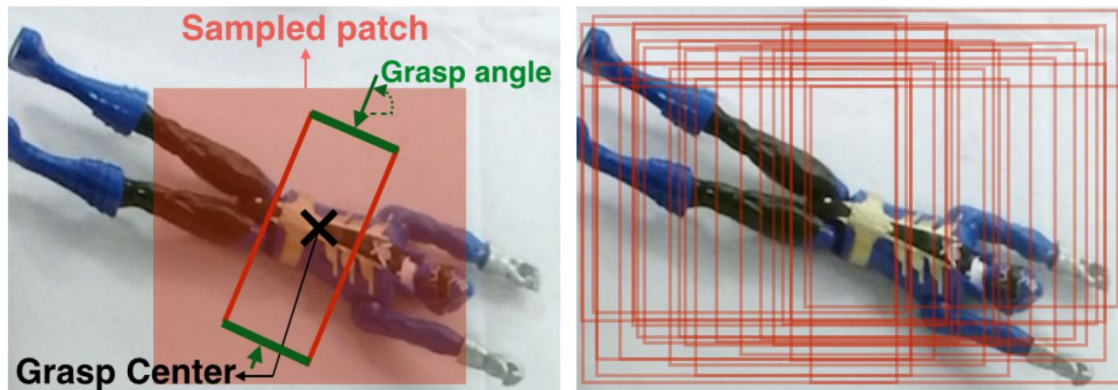


Grasping — Execution



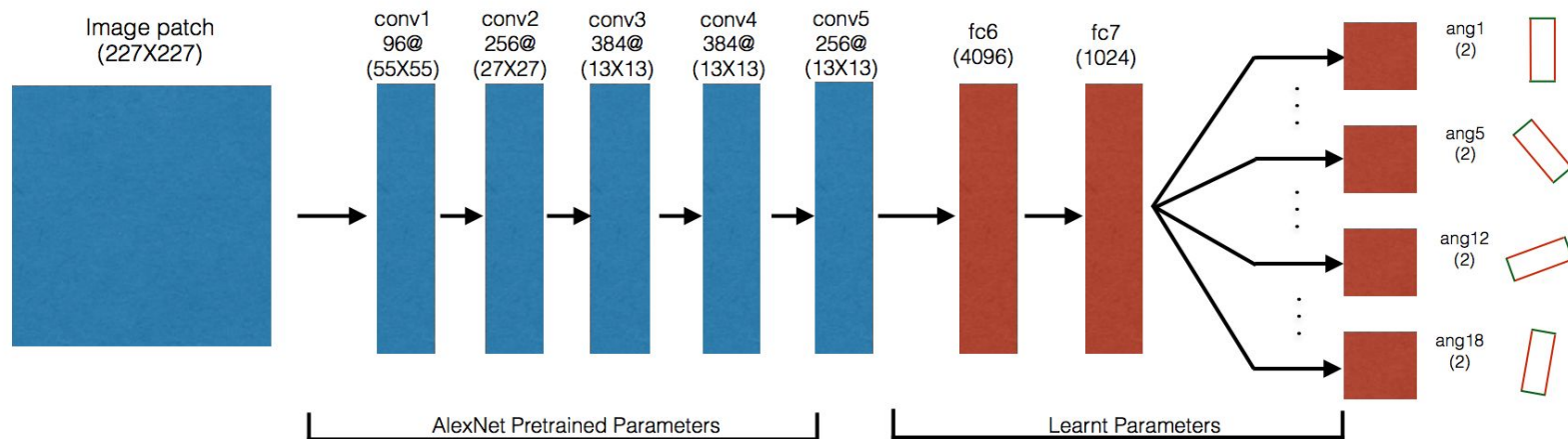
- Use Mixture of Gaussians (MOG) subtraction algorithm to identify objects
- Determine arm placement (next slide) and grasp
- Verify that the grasp was successful using force sensors

Grasping — Execution



- 18-way binary classification problem
- Determine probability of a successful grasp at each angle
- Patches are sampled uniformly from the region of interest

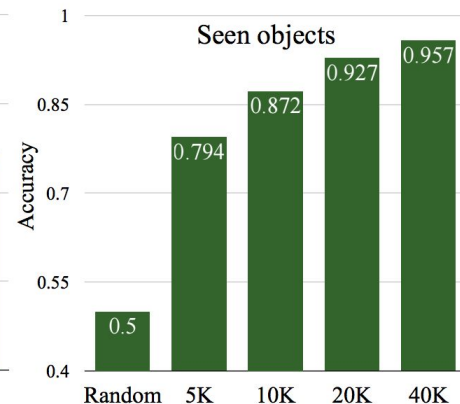
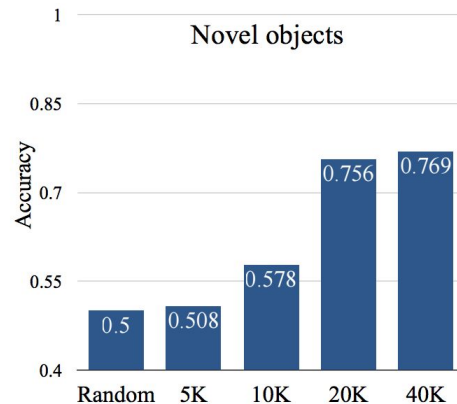
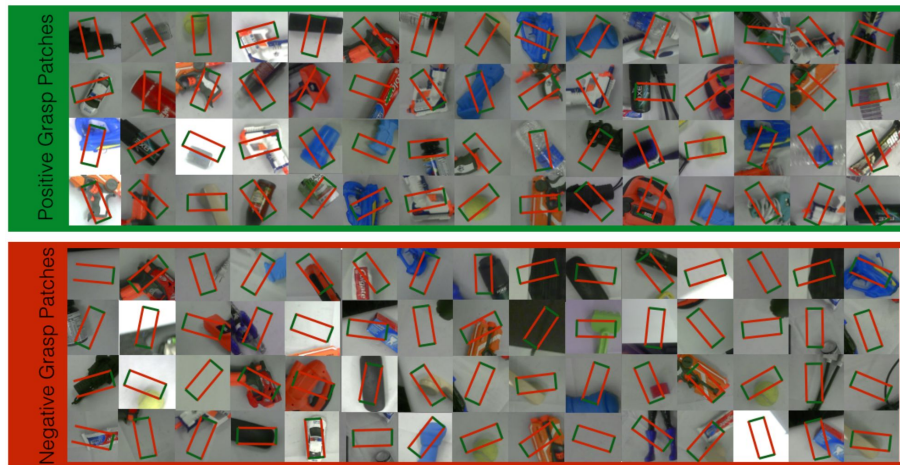
Grasping — Architecture



- First 5 layers pre-trained on ImageNet

$$L_B = \sum_{i=1}^B \sum_{j=1}^{N=18} \delta(j, \theta_i) \cdot \text{softmax}(A_{ji}, l_i)$$

Grasping — Results



	Heuristic			Learning based			
	Min eigenvalue	Eigenvalue limit	Optimistic param. select	kNN	SVM	Deep Net (ours)	Deep Net + Multi-stage (ours)
Accuracy	0.534	0.599	0.621	0.694	0.733	0.769	0.795

- Generated dataset for future studies (hint: we'll see it again soon)
- Over 50K grasp attempts with random, staged, and training-test splits

Grasping — Demo

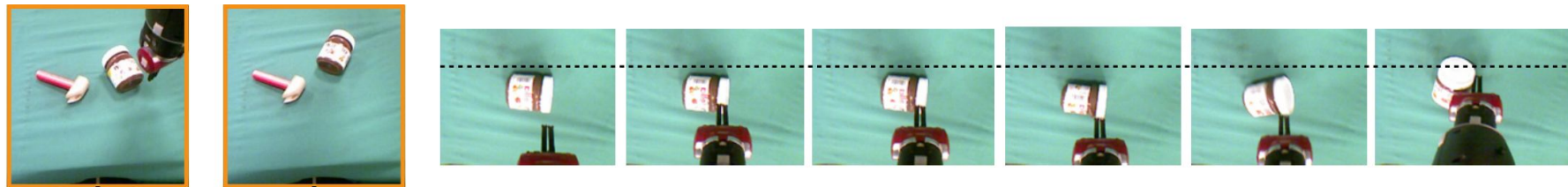


[Supersizing Self-supervision Video on YouTube by Pinto et. al., arXiv 2015](#)

Grasping — Takeaway

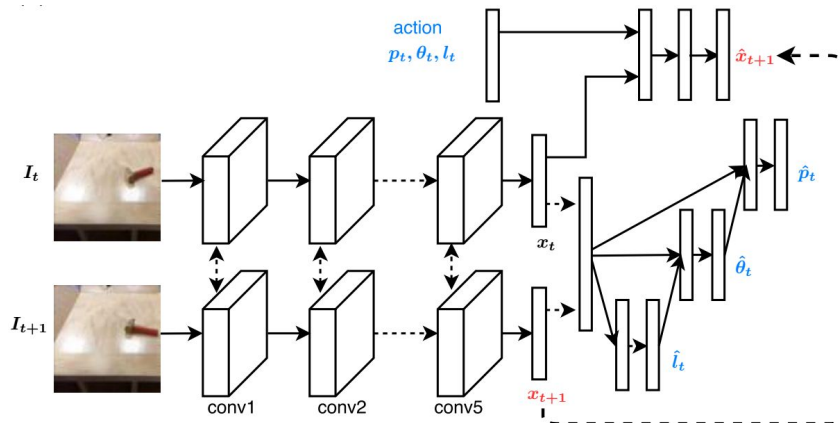
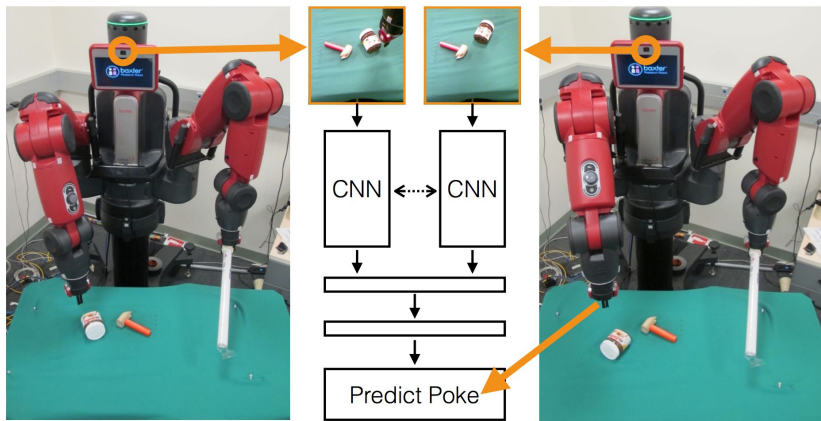
- An example of self-supervised robotic system
- DeepNet performs better than similar methods/heuristics
- Not based on reinforcement learning
- Predicts the way an object is grasped entirely based on one image

Poking — Setup



- Given a robot with arms and camera
- Forward problem: given a state and some actions, determine the outcome
- Inverse problem: given 2 states, determine actions to transition between them
- Joint training of forward and inverse models

Poking — Architecture



- Siamese CNN; first 5 layers are AlexNet
- The output of inverse model (2 images) is used as an input to the forward one

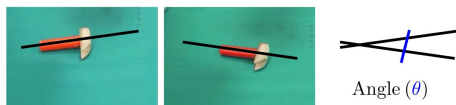
$$L_{joint} = L_{inv}(u_t, \hat{u}_t, W) + \lambda L_{fwd}(x_{t+1}, \hat{x}_{t+1}, W)$$

Poking — Results

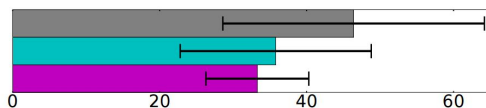
(b) Blob Model



(c) Pose Error Evaluation

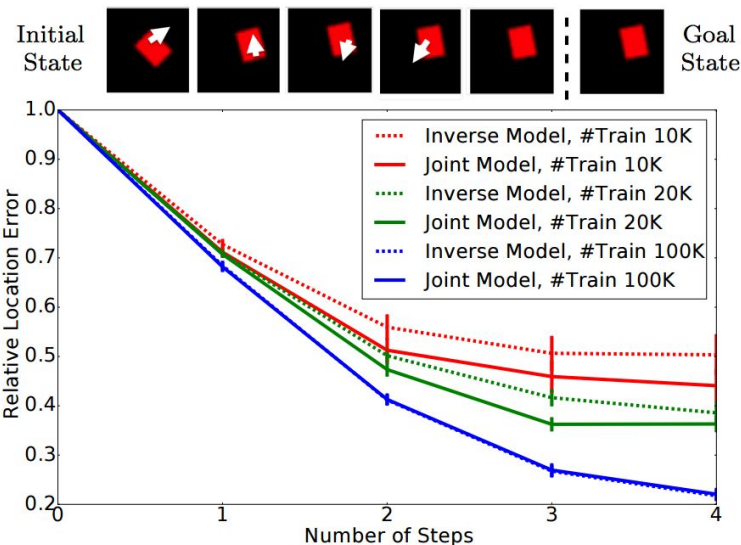
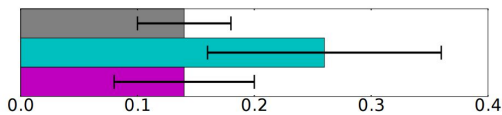


(a) Pose error for nearby goals



Legend: Blob Model (grey), Inverse Model (cyan), Joint Model (magenta)

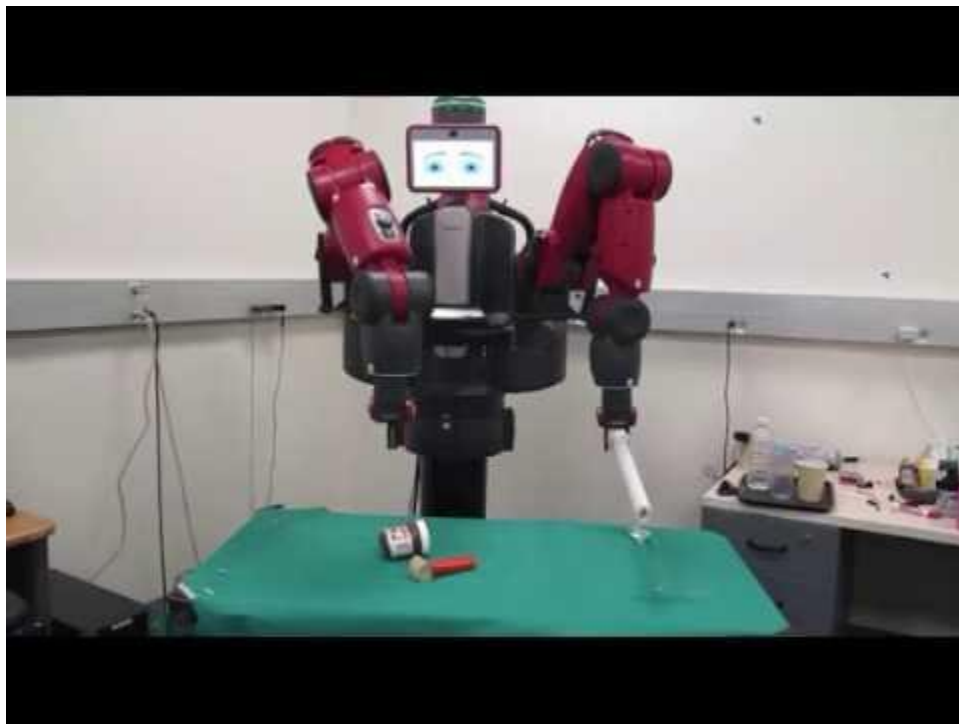
(b) Relative location error for far away goals



(c) Simulation experiments

- Joint model outperforms both the inverse and the naive one

Poking — Demo



[Learning to Poke by Poking Video on YouTube](#)

Poking — Takeaway

- Practical application of a Siamese network
- Self-supervised robotic system
- Training two complementary models at the same time is beneficial
- Predicts how to move the objects given only 2 images

Learning through Physical Interactions

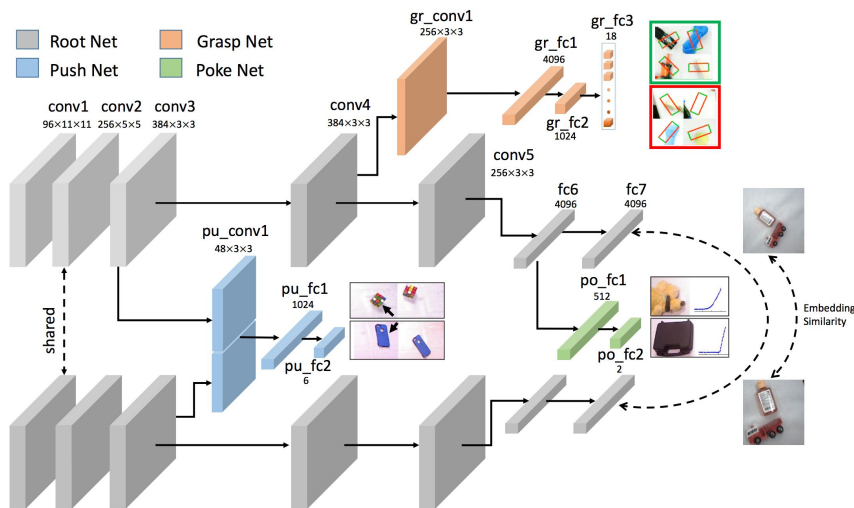
Question

- Is a picture always enough?
- Can we somehow use physical access to the object?

Key Idea

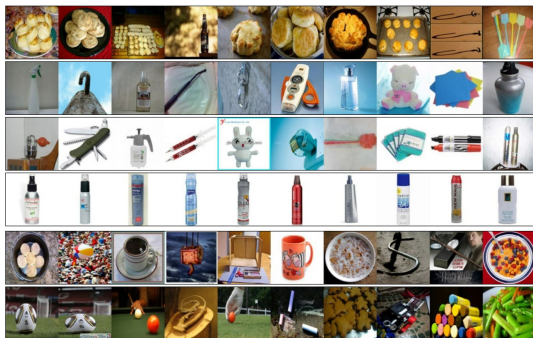
- Given an image we can predict some physical properties
 - How and where to grasp the object
 - How to move it
 - How hard/soft the object is
- Those are high level features
- Let's use them to classify images better!

Architecture — Bringing It All Together



- Root Net is AlexNet
- Poke Net uses a tactile sensor to predict how hard/soft an object is
- Pose invariance

Learning Visual Representations — Results



	Household	UW RGBD	Caltech-256
Root network with random init.	0.250	0.468	0.242
Root network trained on robot tasks (ours)	0.354	0.693	0.317
AlexNet trained on ImageNet	0.625	0.820	0.656
Root network trained on identity data	0.315	0.660	0.252
Auto-encoder trained on all robot data	0.296	0.657	0.280

- Better performance than AlexNet, but not consistently
- Further research into integrating this with vision is needed

Navigation and Auxiliary Supervision

Flight Controller — Challenges

- What are they?



[Image from Wikipedia](#)

Flight Controller — Challenges

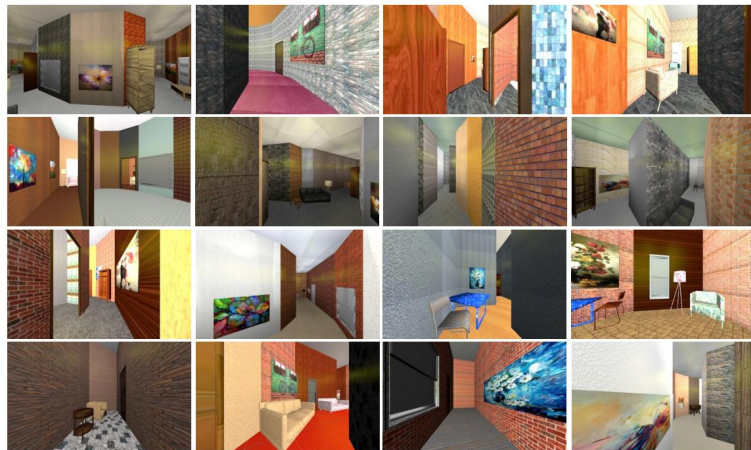
- Real world training is expensive
 - Takes up a lot of time
 - Requires human supervision
 - Limited number of physical robots
 - Collision may be deadly...
- Is there a better way?



Indoor Flight Controller — Approach

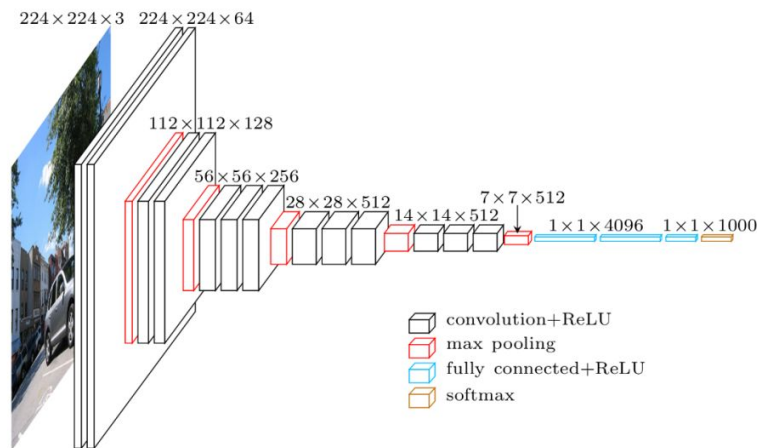
- Train in a simulated environment!
 - Modeled with CAD
- A reinforcement learning problem
- Single image input
- Predict discounted collision probability and pick direction with the lowest one

$$P(C|\mathbf{I}_t, a_t) = \sum_{s=t}^{t+H} \gamma^{s-t} P(c_s|\mathbf{I}_s, a_s)$$

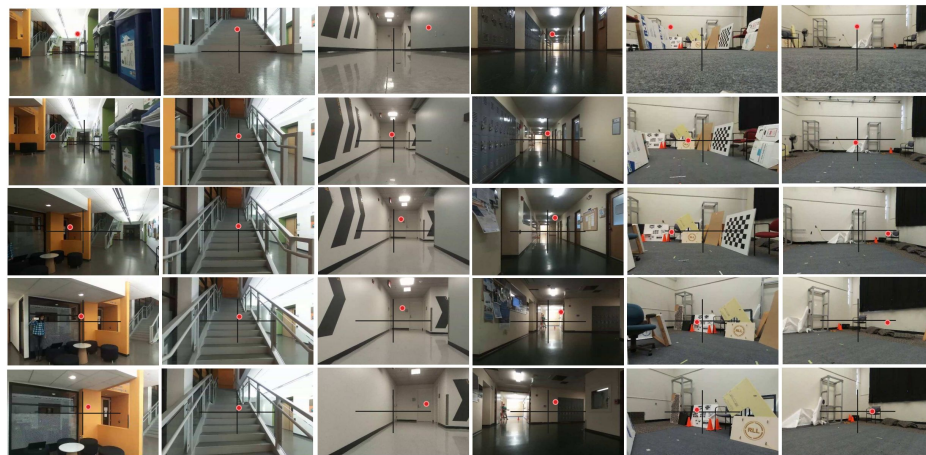
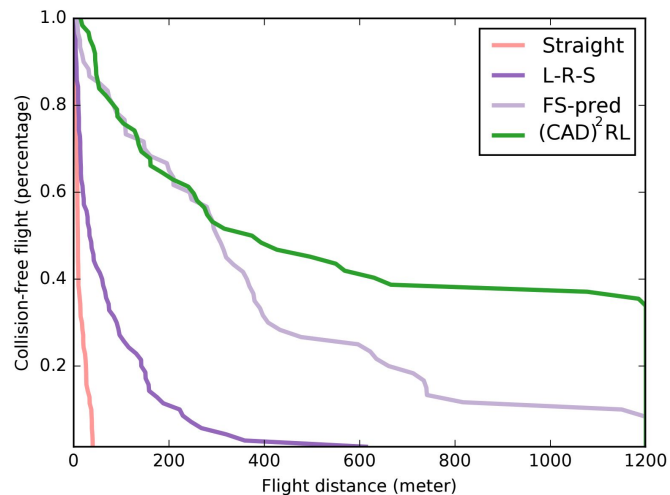


Indoor Flight Controller — Architecture

- Q-function represented by VGG16
 - Pre-trained on ImageNet
- Output is 41x41 grid
 - Action-space
- Initialize training on the free-space prediction task
 - Flying 1 meter forward

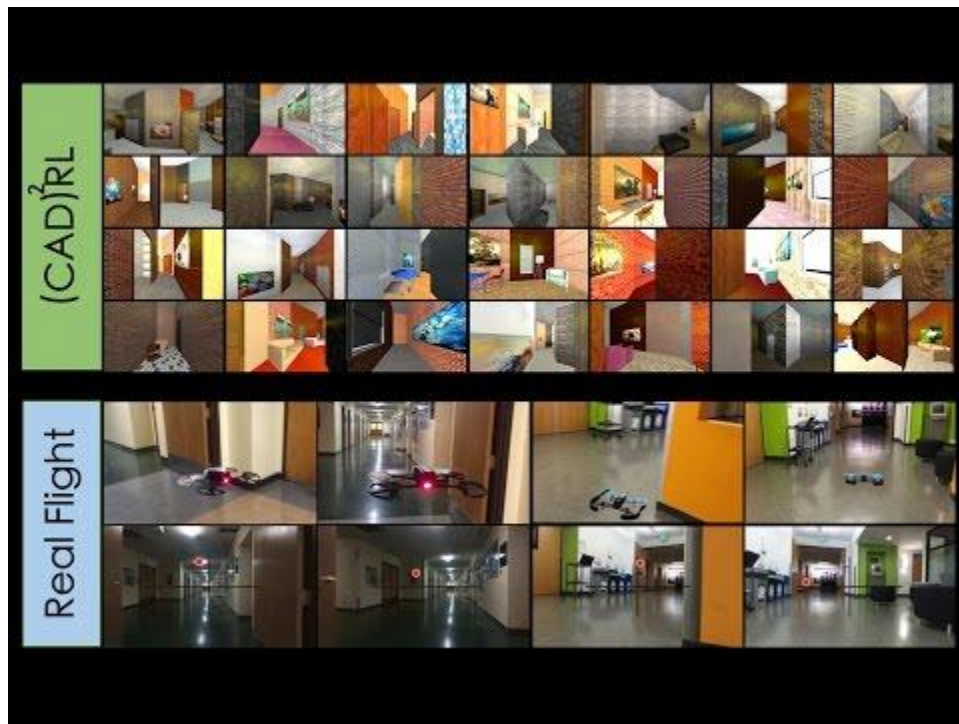


Indoor Flight Controller — Results



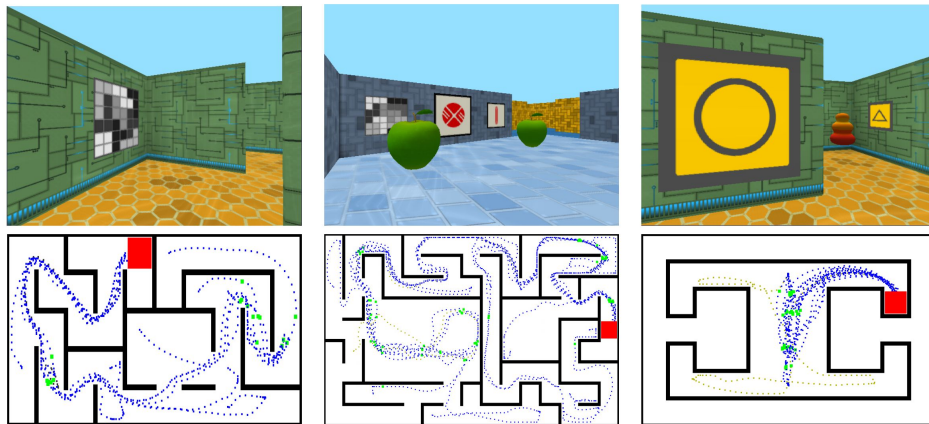
- Demonstrates better performance than heuristic algorithms
- Generalizes to work on a real drone

Indoor Flight Controller — Demo



[\(CAD\)²RL: Real Single-Image Flight without a Single Real Image Video on YouTube](#)

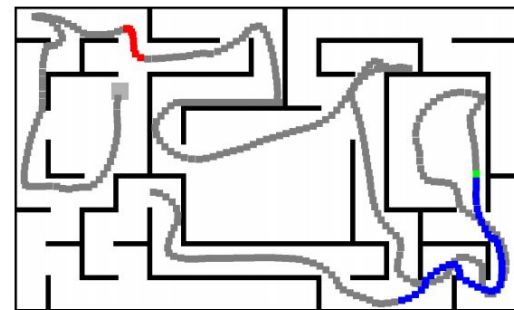
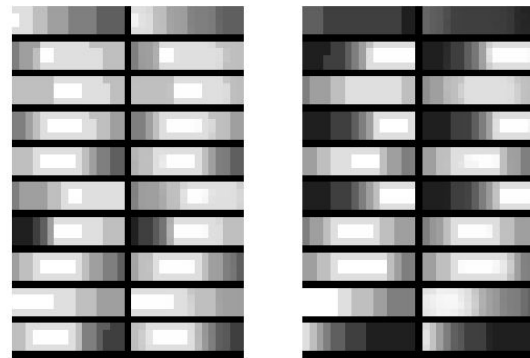
Dynamic Maze Navigation



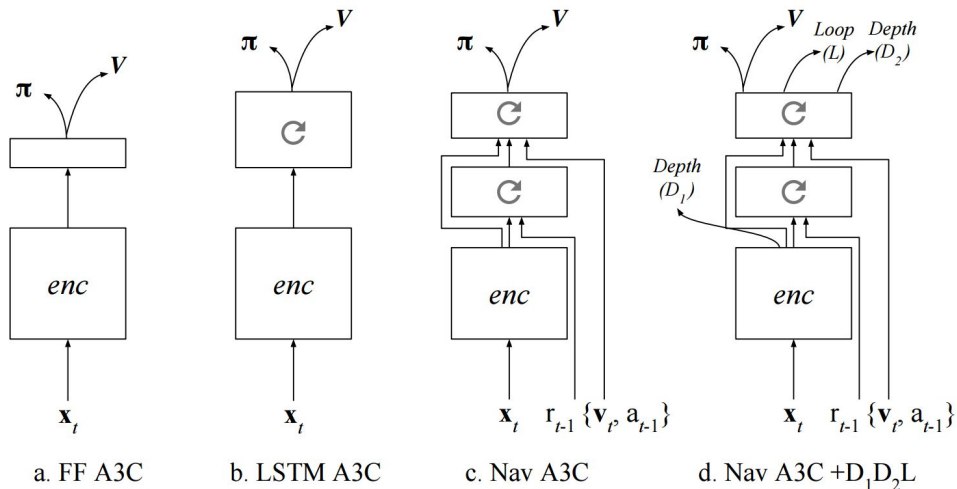
- First-person view
- Related to the problem of indoor navigation
- Landscape may change in the real world
- Rewards are sparse

Dynamic Maze Navigation — Auxiliary Goals

- Bootstrap reinforcement learning with auxiliary tasks
- Depth prediction
- Loop closure prediction
- Use those as tasks as opposed to features for better performance
- Depth as a classification problem with 4x16 map

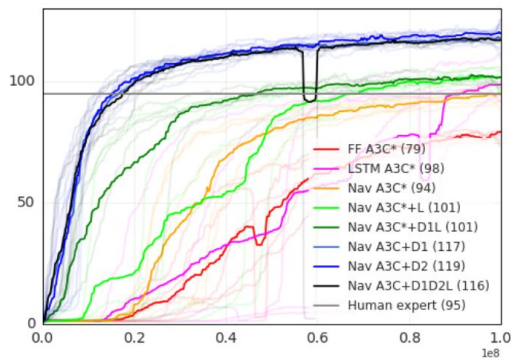


Dynamic Maze Navigation — Architecture

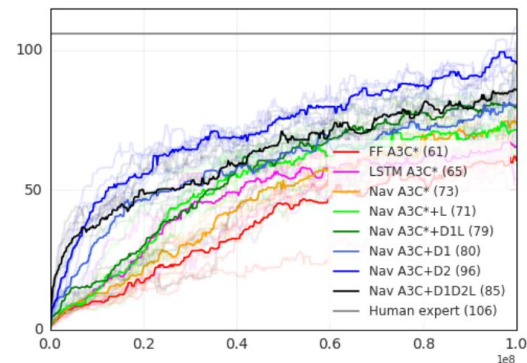


- Stacked LSTM architecture that outputs policy and value function
- Incorporates auxiliary depth and loop closure loss both on the feed-forward stage and LSTM stage and compute loss with both

Dynamic Maze Navigation — Results



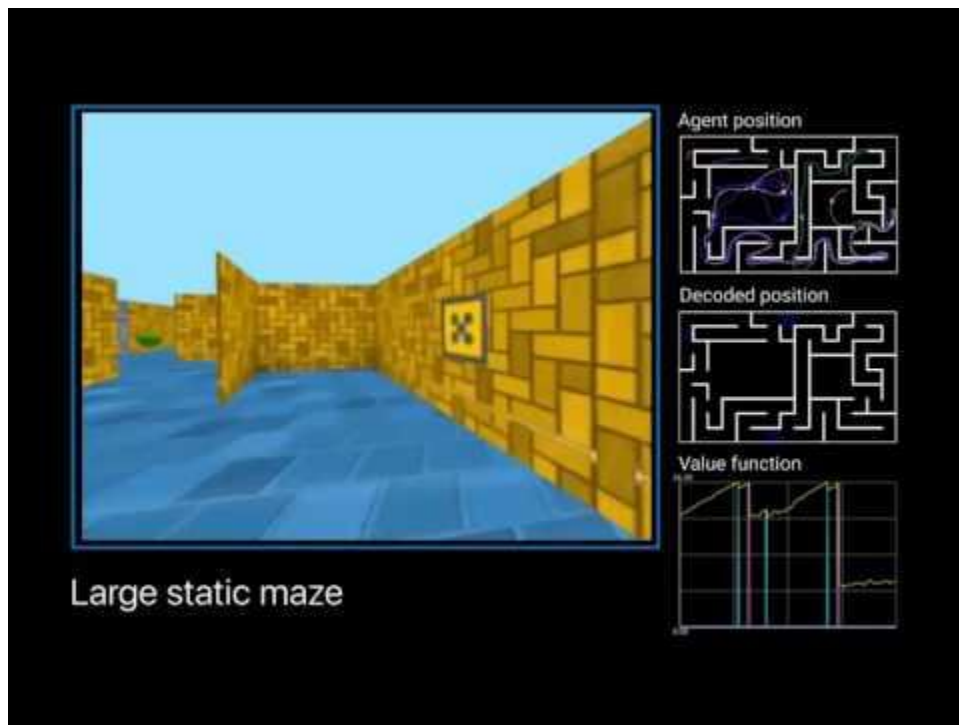
(a) Static maze (small)



(d) Random Goal maze (small)

- Performs better than humans on static mazes
- Around 70-80% of human performance on dynamic ones
- The model with LSTM depth only performs the best (marginally)

Dynamic Maze Navigation — Demo



Questions?

Thank you for your attention!

Reading List

- [End to End Learning for Self-Driving Cars by Bojarski et.al.](#)
- [A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots by Giusti et.al.](#)
- [A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning by Ross et.al.](#)
- [Learning Monocular Reactive UAV Control in Cluttered Natural Environments by Ross et.al.](#)
- [End-to-End Training of Deep Visuomotor Policies by Levine et.al.](#)
- [Supersizing Self-supervision by Pinto et. al.](#)
- [Learning to Poke by Poking: Experiential Learning of Intuitive Physics by Agrawal et.al.](#)
- [The Curious Robot: Learning Visual Representations via Physical Interactions by Pinto et.al.](#)
- [\(CAD\)2RL: Real Single-Image Flight without a Single Real Image by Sadeghi et.al.](#)
- [Learning to Navigate in Complex Environments by Mirowski et.al.](#)

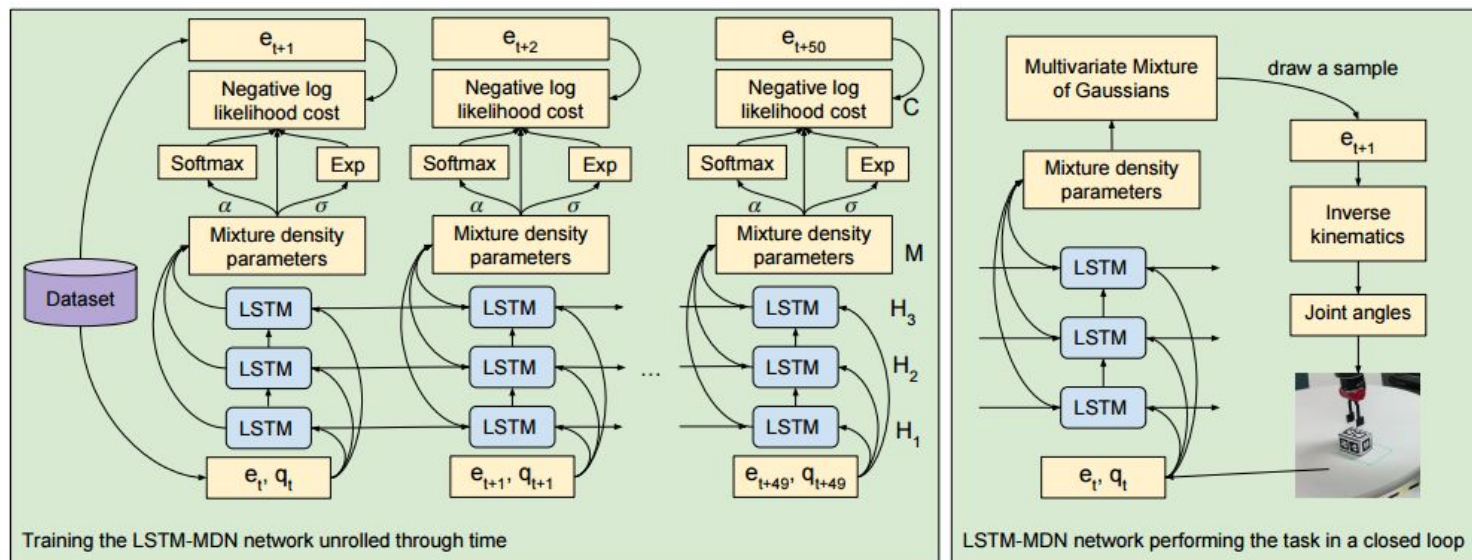
Backup Slides

Imitation with LSTMs



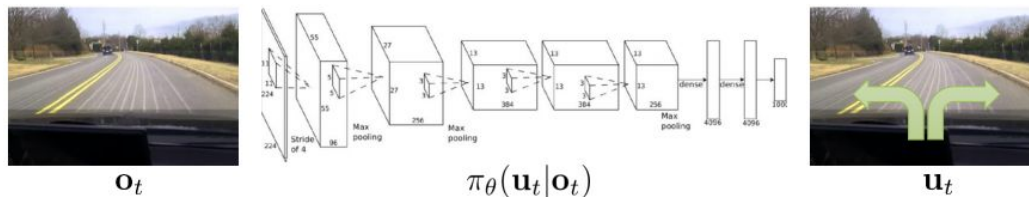
Learning real manipulation tasks from virtual demonstrations using LSTM

Imitation with LSTMs

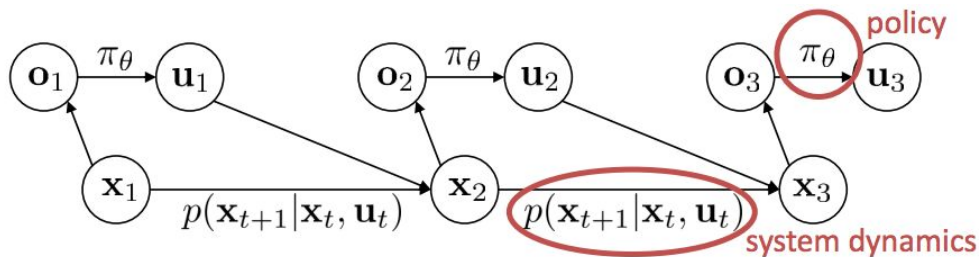


Learning real manipulation tasks from virtual demonstrations using LSTM

Optimal Control



- Obtain “good” trajectories for supervised learning
- **Known** deterministic system dynamics \mathbf{f}



$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$$

Optimal Control in Discrete Space

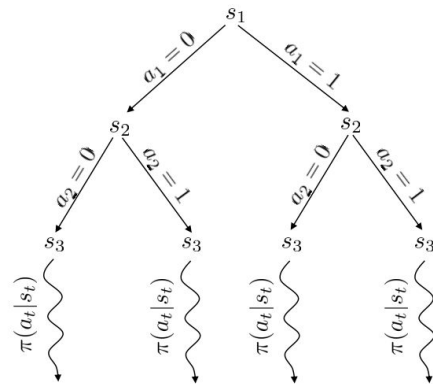
- Obtain “good” trajectories under known discrete dynamics \mathbf{f}

$$\max_{p(\tau)} E_p[r(\tau)]$$

- Monte Carlo tree search (MCTS) for Atari
 - Obtain “good” action for a given state under known dynamics

function *get_best_action_from_root()*:

1. Find a suitable leaf node from root
2. Evaluate the leaf using random policy; get reward
3. Update score of all nodes between leaf and root
4. Take action from root leading to highest score




Atari Games by MCTS

- Directly using MCTS to play game

MCTS	DQN
Model-based; need access to simulator for tree rollouts	Model-free; no information about game dynamics; can generalize better under POMDP
Choosing action is slow (tree expansion)	Choosing action is fast
High scores	Lower scores than MCTS

Atari Games by **Imitating** MCTS

- Imitating optimal control (MCTS) with Dagger

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

- Replace the human with MCTS
- Train NN policy on pixels using supervision from MCTS at training time
- No MCTS at test time, NN is fast!
- Reduce distribution mismatch by getting **on-policy** data

$$D_{KL}(p_{\pi^*}(\tau) || p_{\pi_\theta}(\tau)) < \epsilon$$

Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning

Xiaoxiao Guo
Computer Science and Eng.
University of Michigan
guoxiao@umich.edu

Satinder Singh
Computer Science and Eng.
University of Michigan
baveja@umich.edu

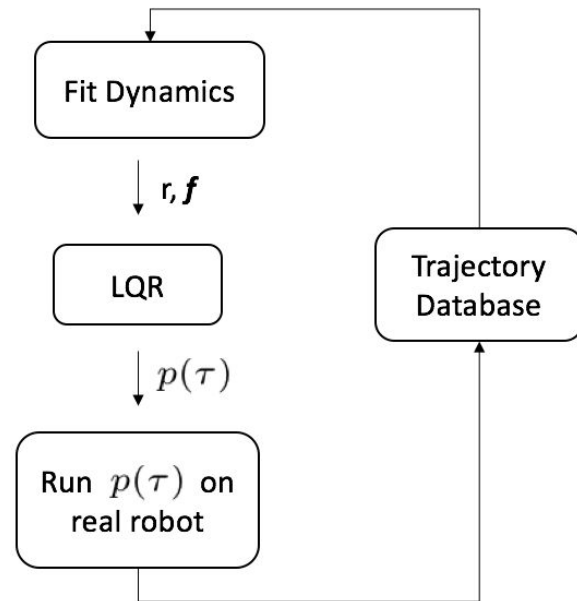
Honglak Lee
Computer Science and Eng.
University of Michigan
honglak@umich.edu

Richard Lewis
Department of Psychology
University of Michigan
rickl@umich.edu

Xiaoshi Wang
Computer Science and Eng.
University of Michigan
xiaoshiw@umich.edu

GPS loop

- A policy search method that scales to high dimensional policies
 - Contrast with gradient methods like REINFORCE
- GPS details
 - Fit local linear models
 - $p(\tau)$ is stochastic so that we get a trajectory distribution
 - Bound D_{KL} between trajectory distributions at consecutive iterations

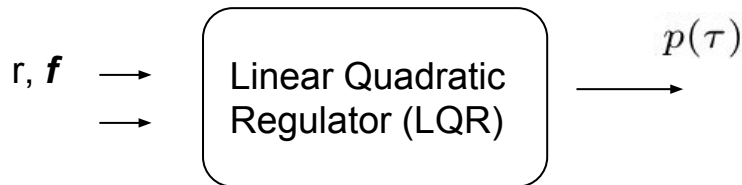


Optimal Control in Continuous Space

- Obtain “good” trajectories under **known** continuous dynamics \mathbf{f}

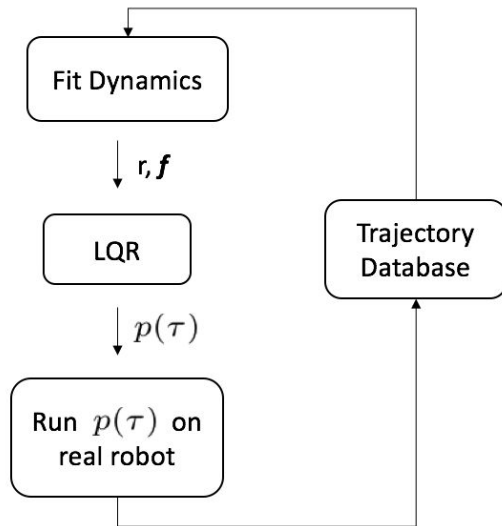
$$\max_{p(\tau)} E_p[r(\tau)]$$

$$\max_{u_1, u_2, \dots, u_T} \sum_{t=1}^T r(x_t, u_t) \quad \text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$



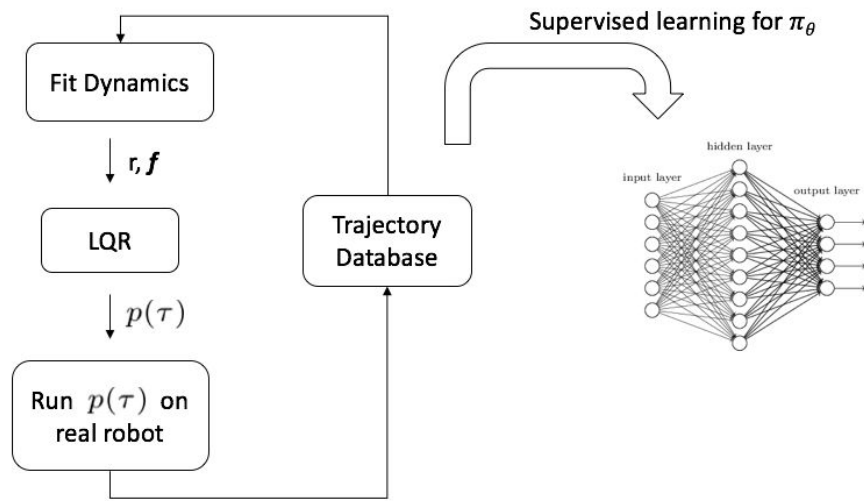
Unknown Dynamics

- Obtain “good” trajectories under **unknown** continuous dynamics
- Learning the dynamics model using regression $\sum_i ||f(x_i, u_i) - x'_i||^2$
 - Deep nets, GP etc.



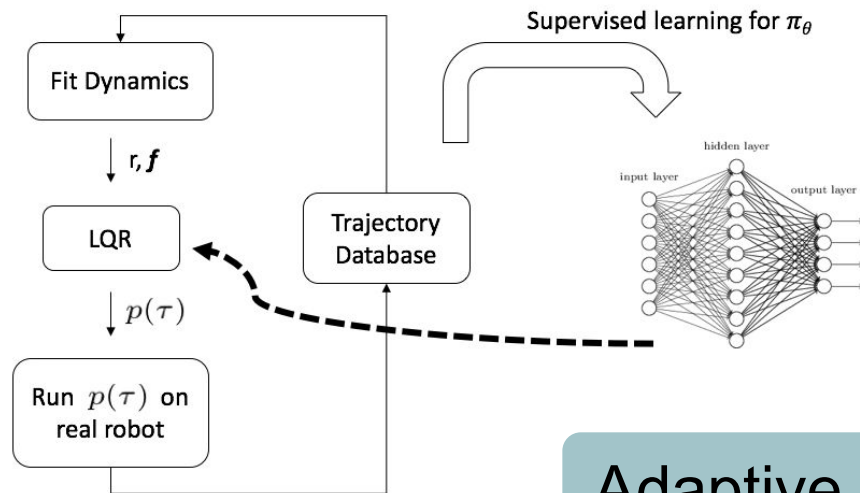
Imitating Optimal Control

- GPS obtains “good” trajectories under **unknown** continuous dynamics



Imitating Optimal Control

- GPS obtains “good” trajectories under **unknown** continuous dynamics



Adaptive Teacher

GPS in action

End-to-End Training of Deep Visuomotor Policies

Sergey Levine[†]
Chelsea Finn[†]
Trevor Darrell
Pieter Abbeel

SVLEVINE@EECS.BERKELEY.EDU
CBFINN@EECS.BERKELEY.EDU
TREVOR@EECS.BERKELEY.EDU
PABBEEL@EECS.BERKELEY.EDU

