Deep Learning for Robotics

Jesse Zhang, CSCI 566 Fall 2020

But Wait!

This was 12 years ago! Why aren’t we seeing robots doing this now?

…Controlled by humans ☺

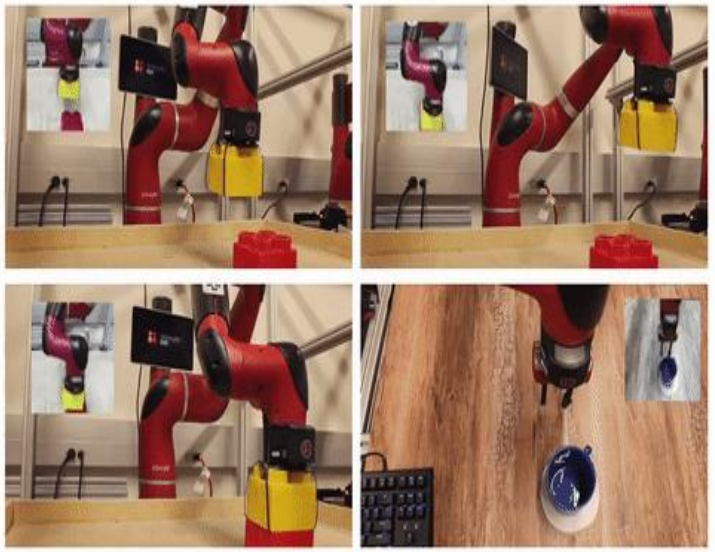
Adapted from Pieter Abbeel

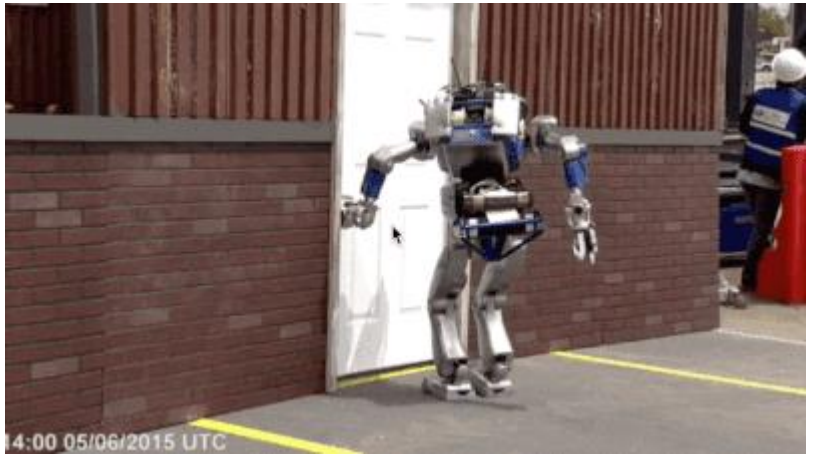
What does this mean?

- Robots are technically capable of doing all these tasks
- Why can’t robots do these things 12 years later?
  - Too hard to manually program robots to do this -> Deep Learning
  - **Our deep learning algorithms are not good enough**

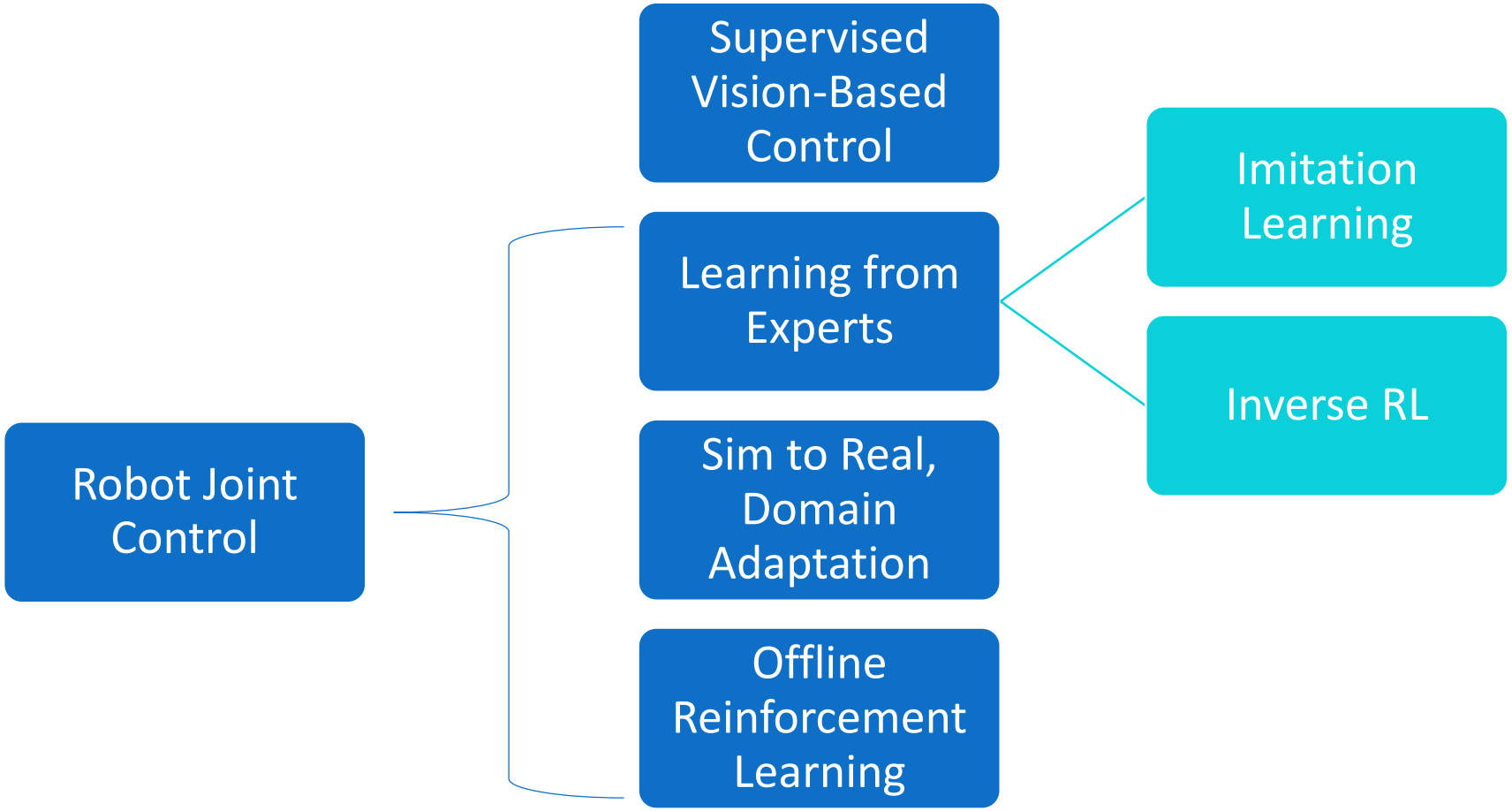
Adapted from Pieter Abbeel

What can we do now?

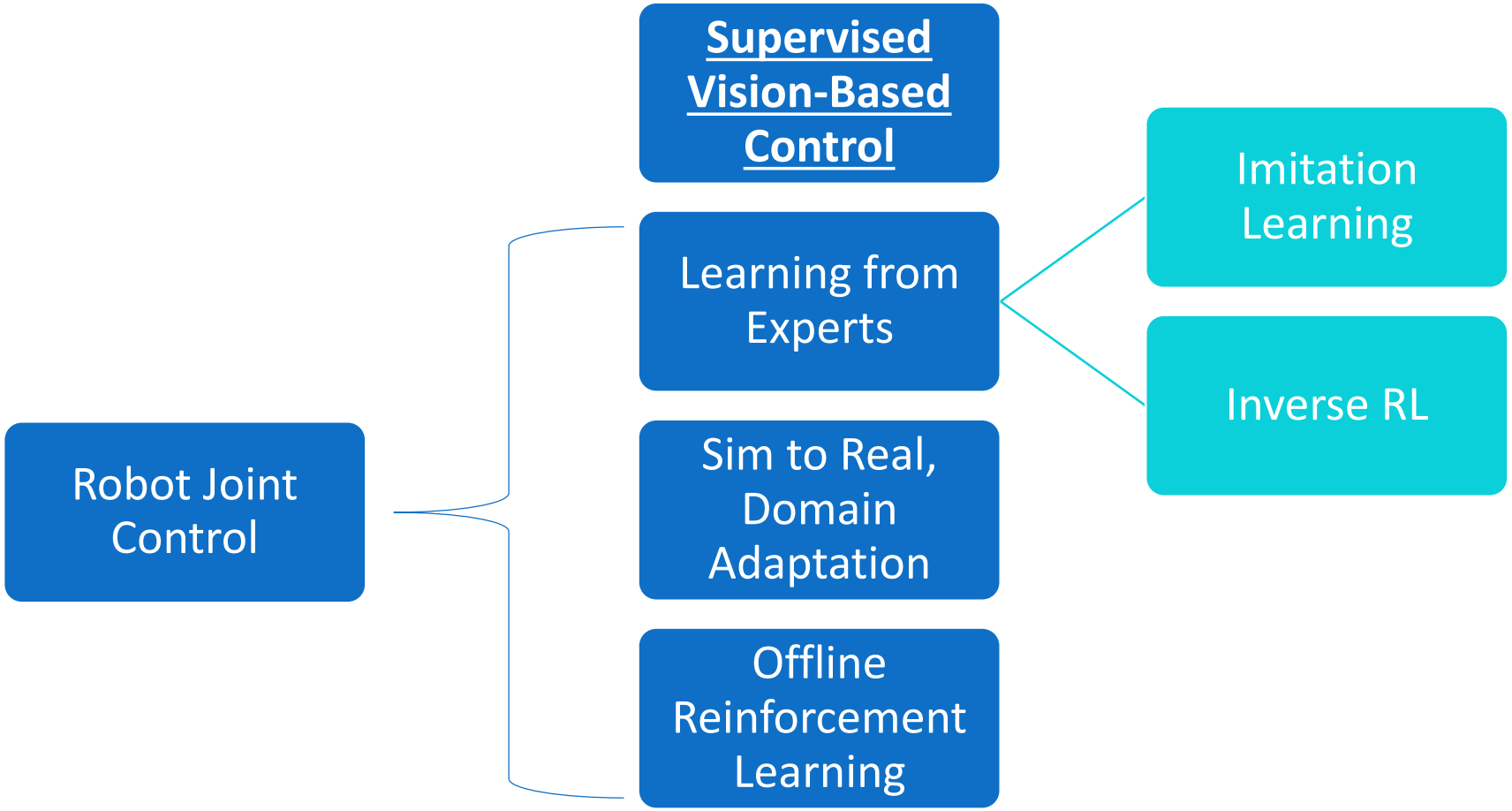




Outline of Today's Lecture

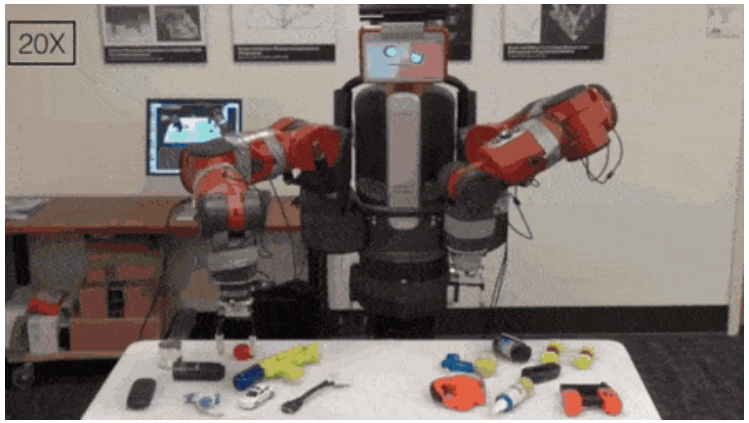


Outline of Today's Lecture



Supervised Vision-Based Learning

Using vision to solve robotic tasks through supervised learning

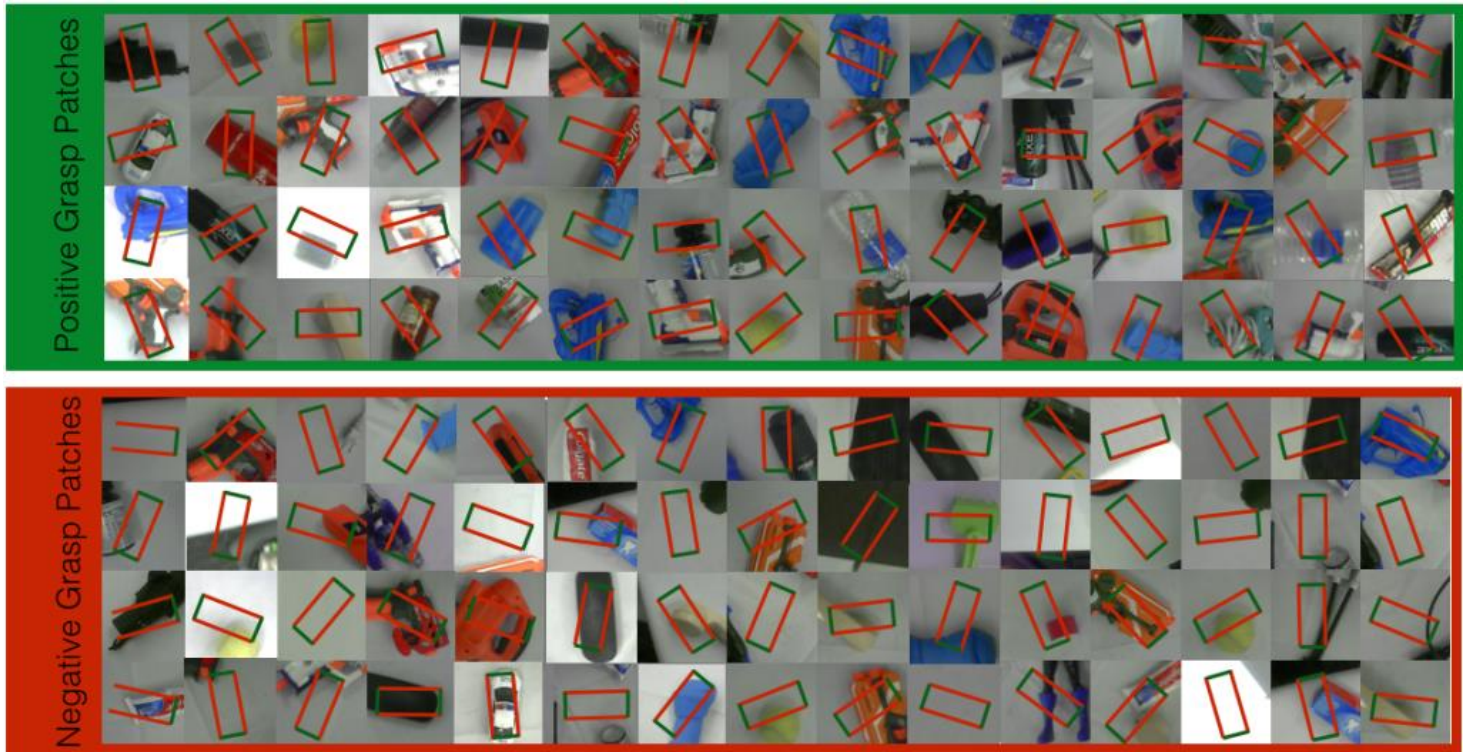


- Mainly focused on object manipulation: grasping
  - We can learn everything from grasping?
- General Idea:
  1. Collect dataset with traditional grasping methods
  2. Train CNN architecture on images to predict grasping success probabilities on regions

Lerrel Pinto, Abhinav Gupta

Supervising Self-Supervision: Learning to Grasp from 50K Attempts and 700 Robot Hours

1. Collect 50k grasping examples: keep track of (x_i, y_i, θ_i)
    1. Sample 380x380 patch with (x_i, y_i) at the center
    2. Randomly rotate that patch



Lerrel Pinto, Abhinav Gupta

Supervising Self-Supervision: Learning to Grasp from 50K Attempts and 700 Robot Hours

1. Collect 50k grasping examples: keep track of (x_i, y_i, θ_i)
   1. Sample 380x380 patch with (x_i, y_i) at the center
   2. Randomly rotate that patch
2. Use AlexNet-based Grasp Network
   1. Loss function: Cross Entropy Loss

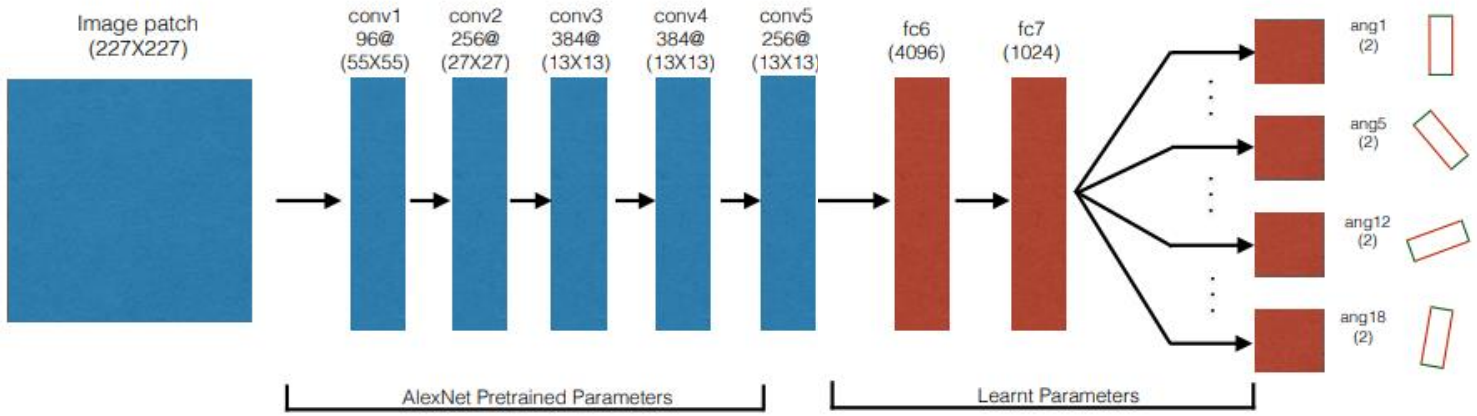


Fig. 5. Our CNN architecture is similar to AlexNet [6]. We initialize our convolutional layers from ImageNet-trained Alexnet.

Lerrel Pinto, Abhinav Gupta

Supervising Self-Supervision: Learning to Grasp from 50K Attempts and 700 Robot Hours

1. Collect 50k grasping examples: keep track of (x_i, y_i, θ_i)
   1. Sample 380x380 patch with (x_i, y_i) at the center
   2. Randomly rotate that patch
2. Use AlexNet-based Grasp Network
   1. Loss function: Cross Entropy Loss
3. Finally, Grasp!
   1. Sample hundreds of random patches, pick patch and grasping angle with highest predicted success probability!

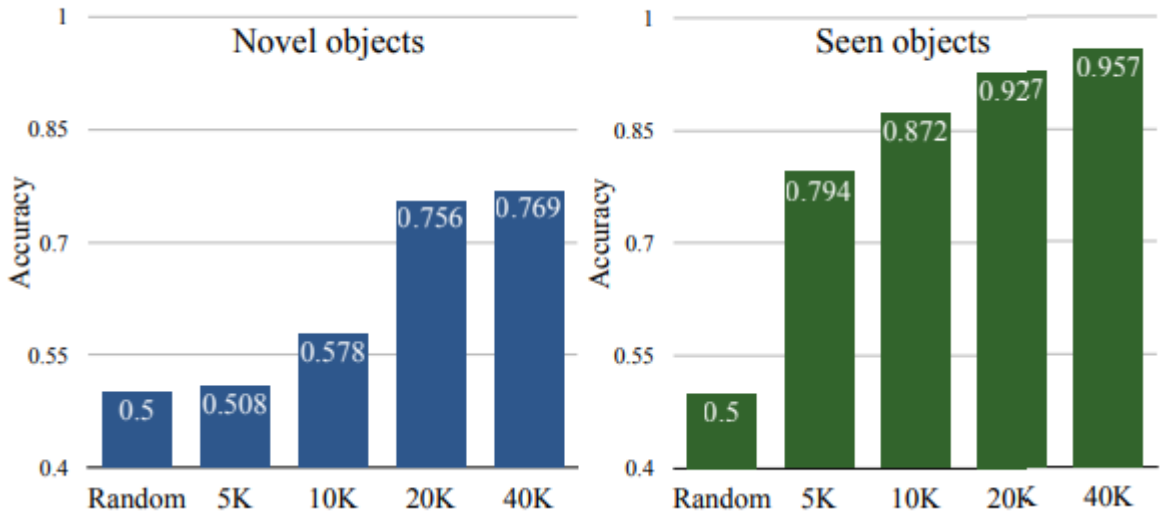


TABLE II

COMPARING OUR METHOD WITH BASELINES

	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**

Lerrel Pinto, Abhinav Gupta

Supervising Self-Supervision: Learning to Grasp from 50K Attempts and 700 Robot Hours

- Q: How is arm control performed?
  - Given (x_i, y_i, θ_i) grasping params, **call upon a robotic kinematic controller to figure out how to grasp**
- Q: What do the best ranked success patches look like?

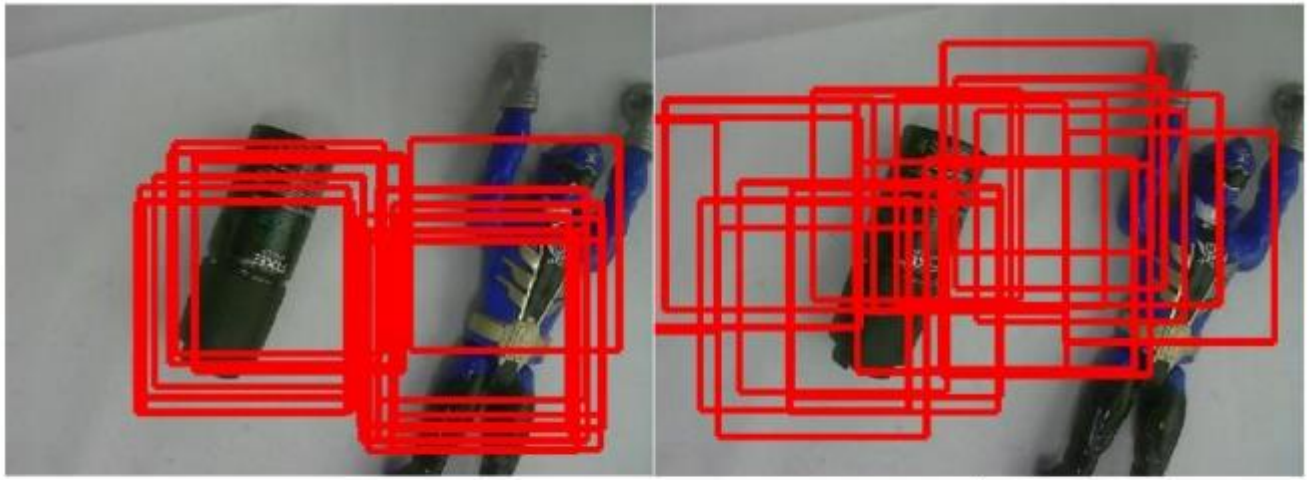
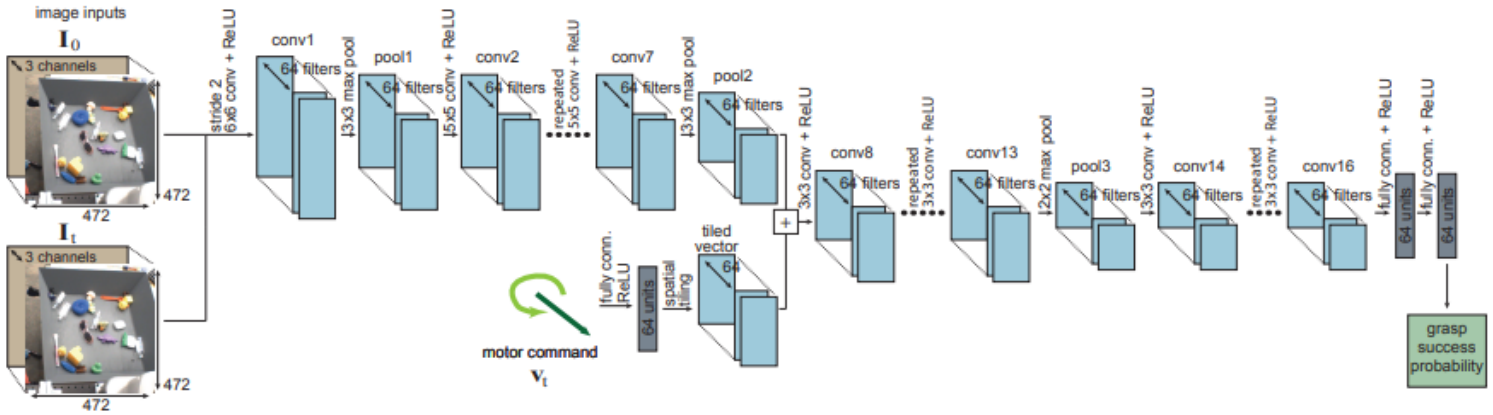


Fig. 6. Highly ranked patches from learnt algorithm (a) focus more on the objects in comparison to random patches (b).

Lerrel Pinto, Abhinav Gupta

Extending this idea even further?

- 800k grasps!

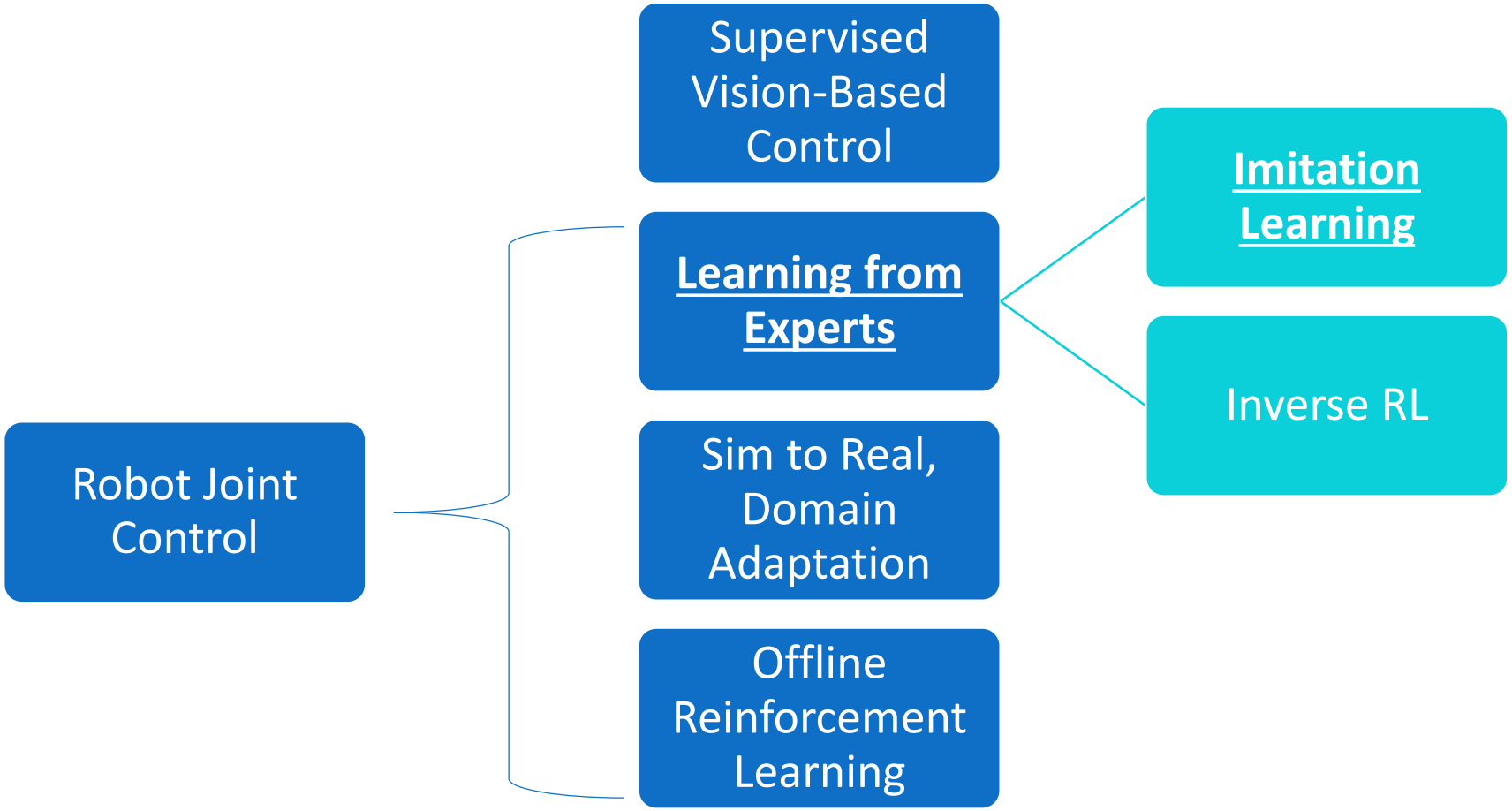


Sergey Levine, Peter Pastor, Alex Krizhevsky, Deirdre Quillen

This works! What's the problem?

- How do we do something other than grasping?
  - Washing dishes?
  - Opening doors?
  - Folding laundry?
- We can't rely just on traditional kinematic control!
- Solution: *Learn* to control the robot, too.

Outline of Today's Lecture



Imitation Learning

Learn from human experts

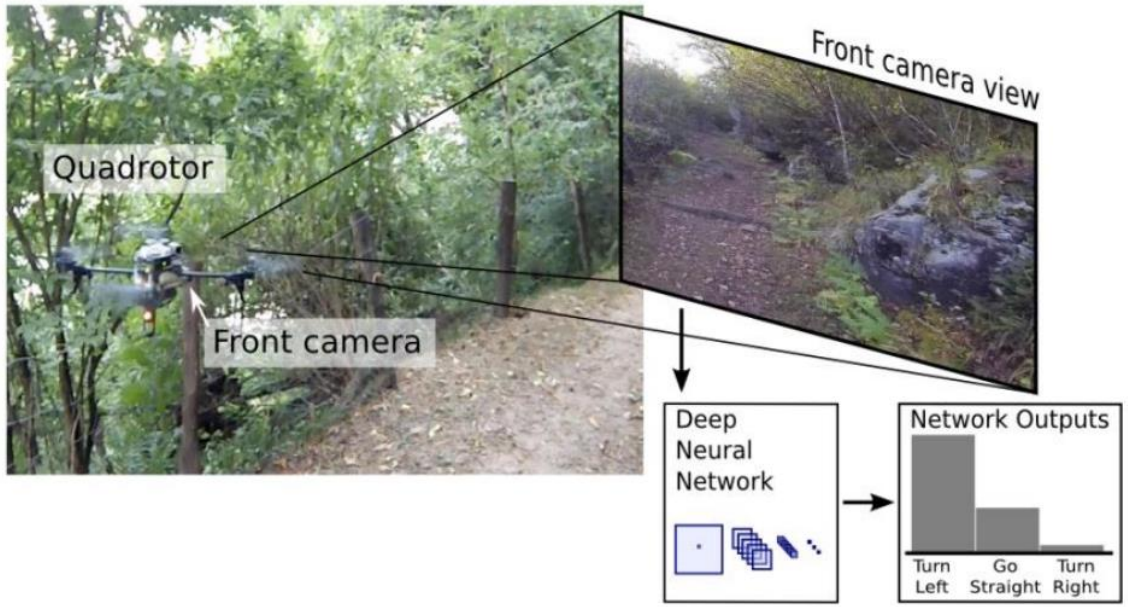
- Learning from scratch on robots is slow, accelerate it with human help
  - Ex: Human teleoperates robot
- Two types of imitation learning:
  1. Assumes demonstrations already collected by the expert
  2. Assumes we can query an expert

Naïve Imitation Learning: Supervised Behavior Cloning

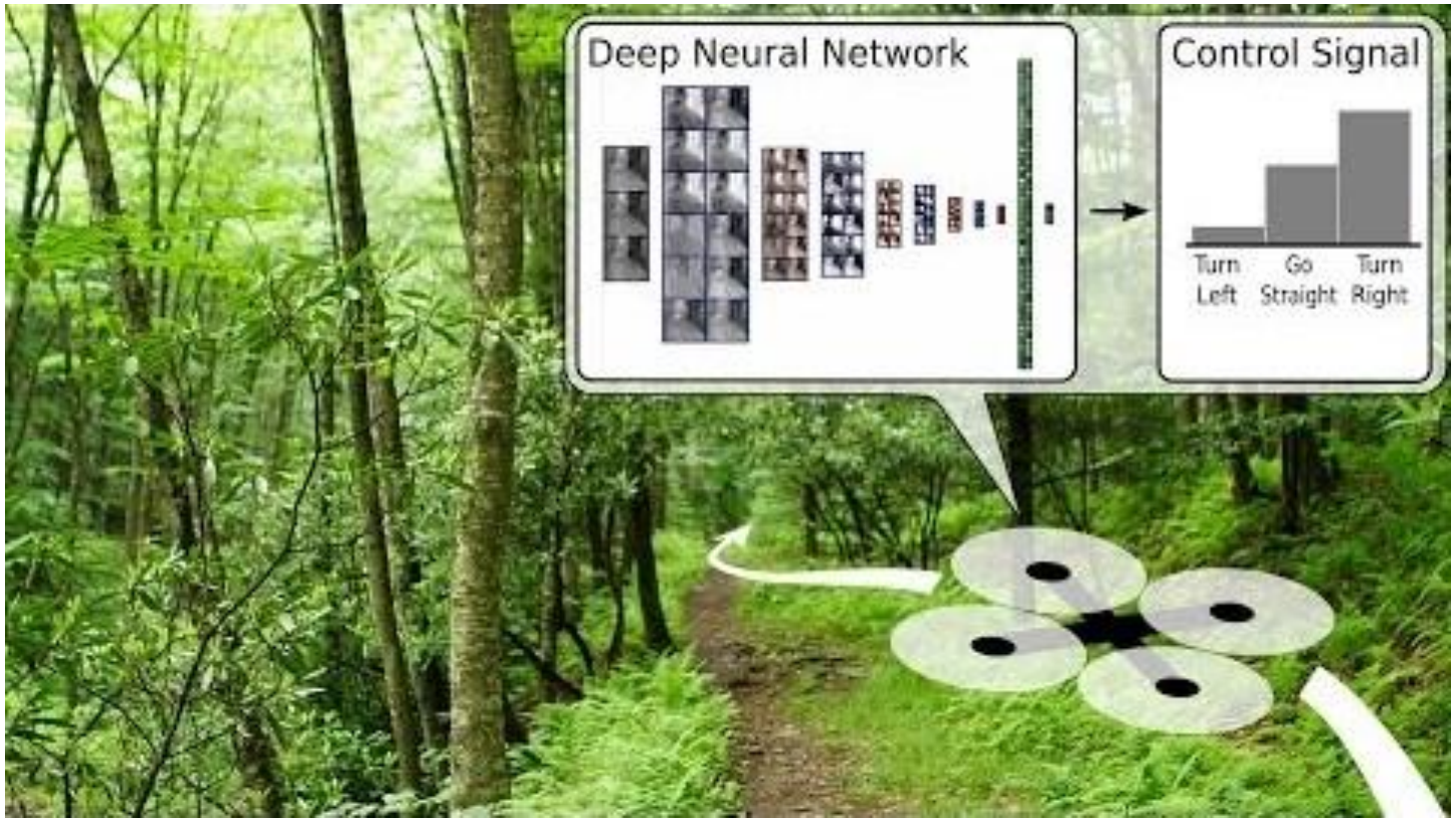
- Assume we have trajectories $\tau_i = (s_1, a_1, s_2, a_2, s_3, a_3, \dots, s_H, a_H)$, dataset = $\{\tau_1, \tau_2, \dots, \tau_n\}$
- How do we model a policy $\pi(a|s)$ to replicate these actions?
  - Continuous: $L_{BC} = \frac{1}{B}\sum[\pi(a|s_i) - a_i]^2$
  - Discrete: $L_{BC} = \frac{1}{B}\sum CrossEntropy(\pi(a|s_i), a_i)$

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

Alessandro Giusti[1], Jérôme Guzzi[1], Dan C. Cireşan[1], Fang-Lin He[1], Juan P. Rodríguez[1]
Flavio Fontana[2], Matthias Faessler[2], Christian Forster[2]
Jürgen Schmidhuber[1], Gianni Di Caro[1], Davide Scaramuzza[2], Luca M. Gambardella[1]



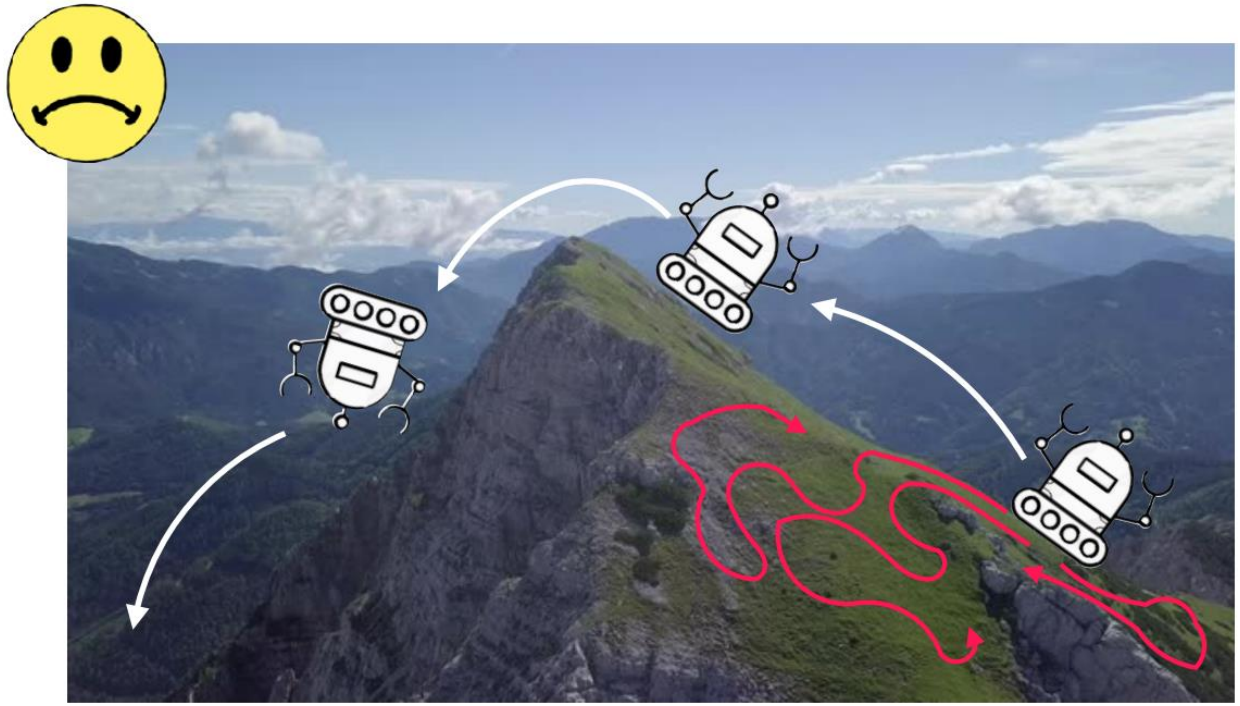
Behavior Cloning on Images



Quadcopter in Action

Naïve Imitation Learning: Supervised Behavior Cloning

- What's the problem?
  - Poor support: take one wrong action, and you're in a new place you've never seen before!



Adapted from Sergey Levine

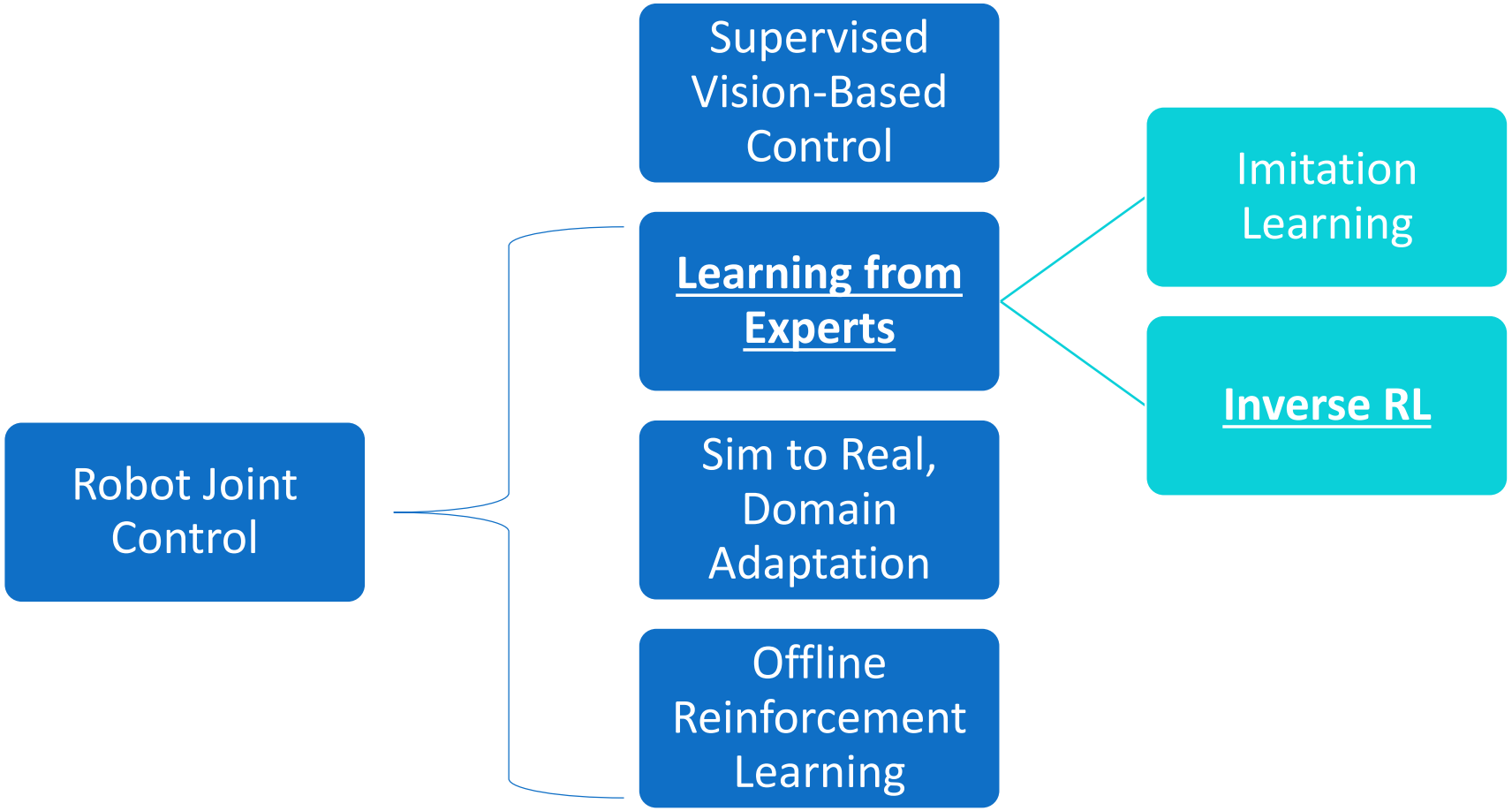
Imitation Learning w/ Expert Querying

- Increase the support of the dataset with Dataset Aggregation (Dagger)
  1. Train $\pi(a|s)$ from $D = \{\tau_1, \tau_2, \dots, \tau_n\}$
  2. Run $\pi(a|s)$ to get dataset $D_\pi = \{\tau_1^\pi, \dots, \tau_n^\pi\}$
     - What's the problem here?
     - No labels for correct actions a
  3. Solution: Ask expert/human to label D_π with CORRECT actions a_t
  4. Aggregate $D = D \cup D_\pi$ and repeat
- Ensures that the policy has a good data support

Ross et al. 2011

Break

Outline of Today's Lecture



Inverse Reinforcement Learning

Rather than copying the expert's actions, we want to copy the expert's *intentions.* Once we figure out their intentions, we can copy their behavior.

- The expert's intentions can be summarized by their true reward function $r(s_t, a_t, s_{t+1})$
- We can learn r_ϕ and then train an RL policy π^{r_ϕ} using r_ϕ
- Problem: Many reward functions can explain expert's behaviors
- What is the *optimal* reward function?

Adapted from Sergey Levine

Maximum Entropy Inverse RL

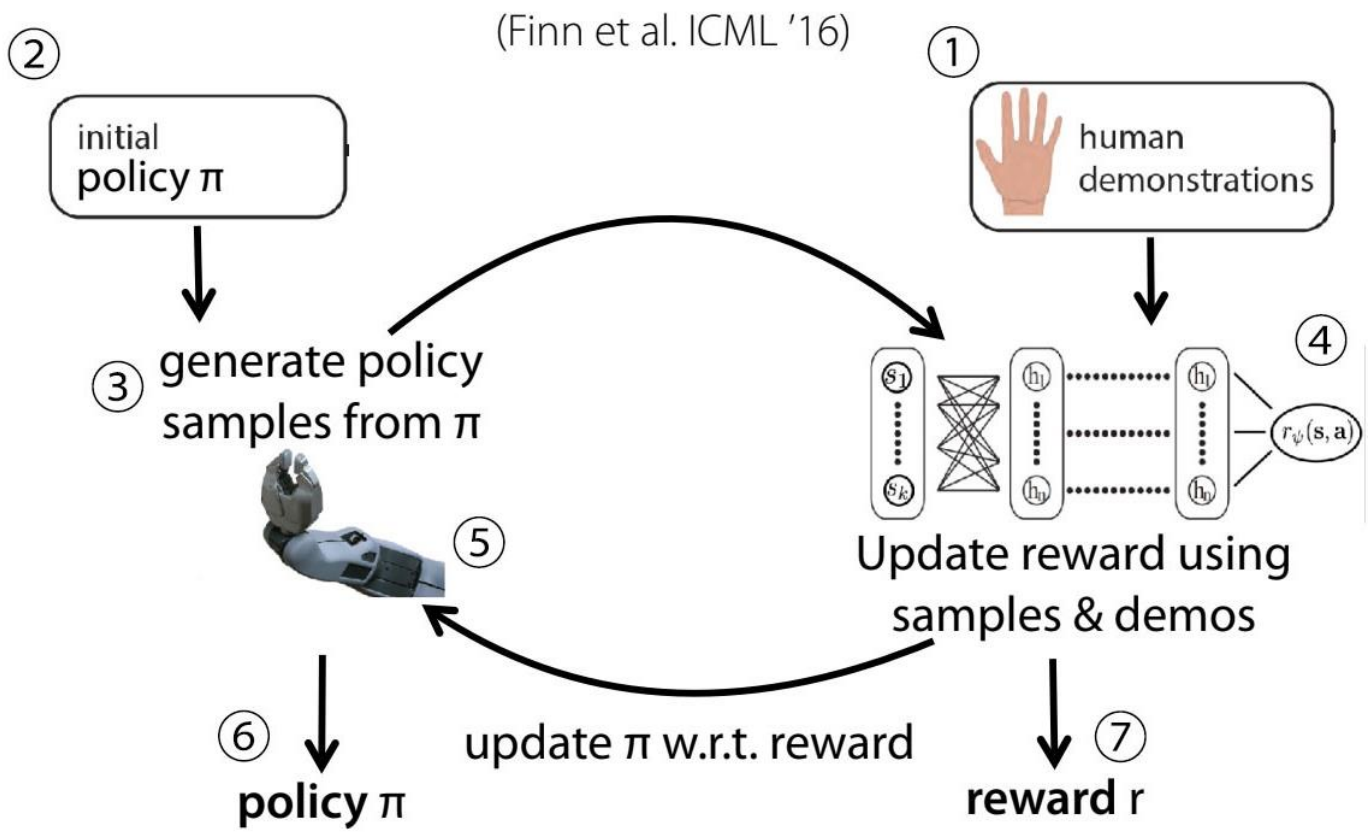
- The optimal reward function r_ϕ should result in a policy that that maximizes the probability of replicating the demonstrations in D, while maximizing entropy on states outside of D
- Intuition: The higher the rewards, the more likely a trajectory by an expert!
- $p_{r_\phi}(\tau_i) = \frac{\exp(R_\phi(\tau_i))}{\int exp\left(R_\phi(\tau_i)\right) d\tau_i}$
- Objective function: $\max_\phi \sum_{\tau_i} \log p_{r_\phi}(\tau_i)$
  - $\nabla_\phi \mathrm{L} = -\frac{1}{|\mathrm{D}|}\sum_{\tau_i} \frac{dr_\phi}{d_\phi}(\tau_i) - \sum_s p(s|\phi)\frac{dr_\phi}{d_\phi}(s)$
- What's the problem with this gradient here?
  - We can't calculate $p(s|\phi)$ unless we know the dynamics of the environment!

Ziebart et al. 2008

Guided Cost Learning (GCL)

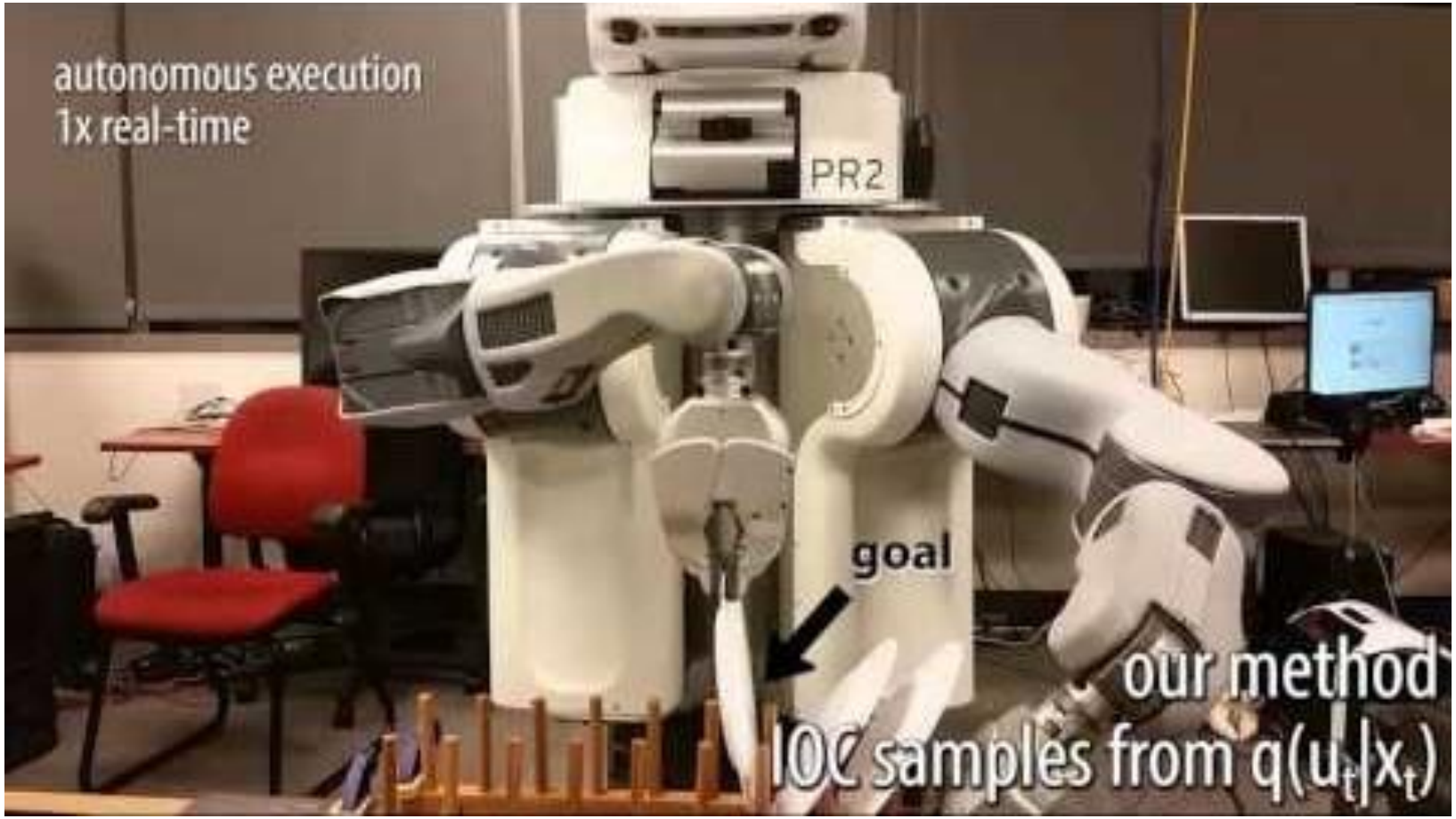
- How do we handle unknown system dynamics?
  - Sample based approximation to estimate the system dynamics
  - Use a policy π_θ to generate samples which are used to estimate the current $p(s|\phi)$

1. REPEAT:
   1. Generate samples D_π from π_θ
   2. $\mathrm{D} = \mathrm{D} \cup \mathrm{D}_\pi$
   3. Calculate loss with an estimate of $\mathrm{p}(\mathrm{s}|\phi)$ with samples from D
   4. Update r_ϕ with loss
   5. Update π_θ with RL Policy Gradient
2. Return π_θ



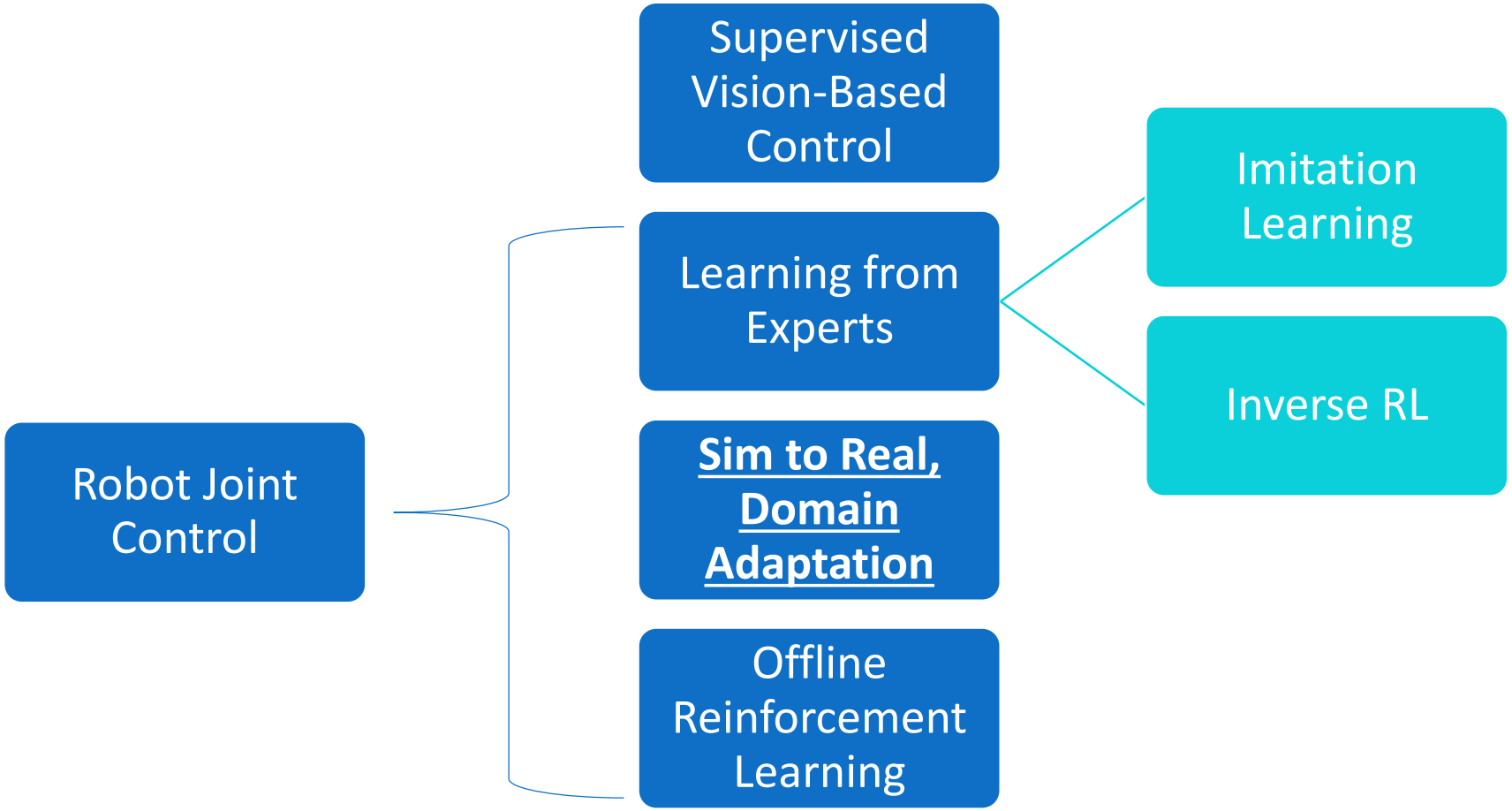
Finn et al. ICML 2016

Guided Cost Learning (GCL)



Finn et al. ICML 2016

Outline of Today's Lecture



Sim-to-Real and Domain Adaptation

Running RL in real-world robots is too sample inefficient. What if we learn in simulation and use domain adaptation techniques to learn in the real world?

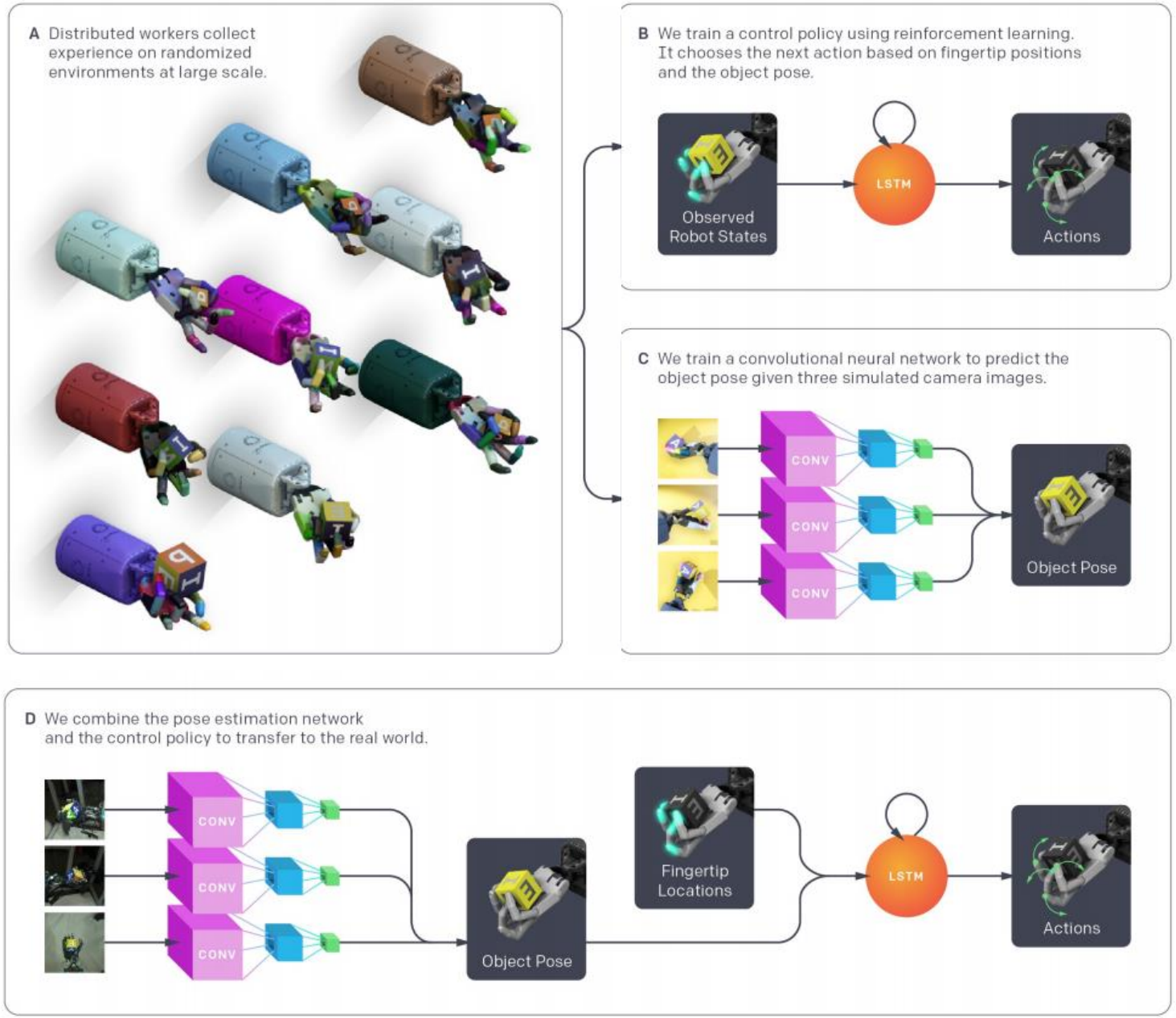
- Learn in a simulator, transfer this knowledge for quick adaptation in the real world
- A form of domain adaptation
  - Can use existing domain adaptation techniques from vision community to help

Julian et. al., ISER 2018

Learning Dexterous In-Hand Manipulation

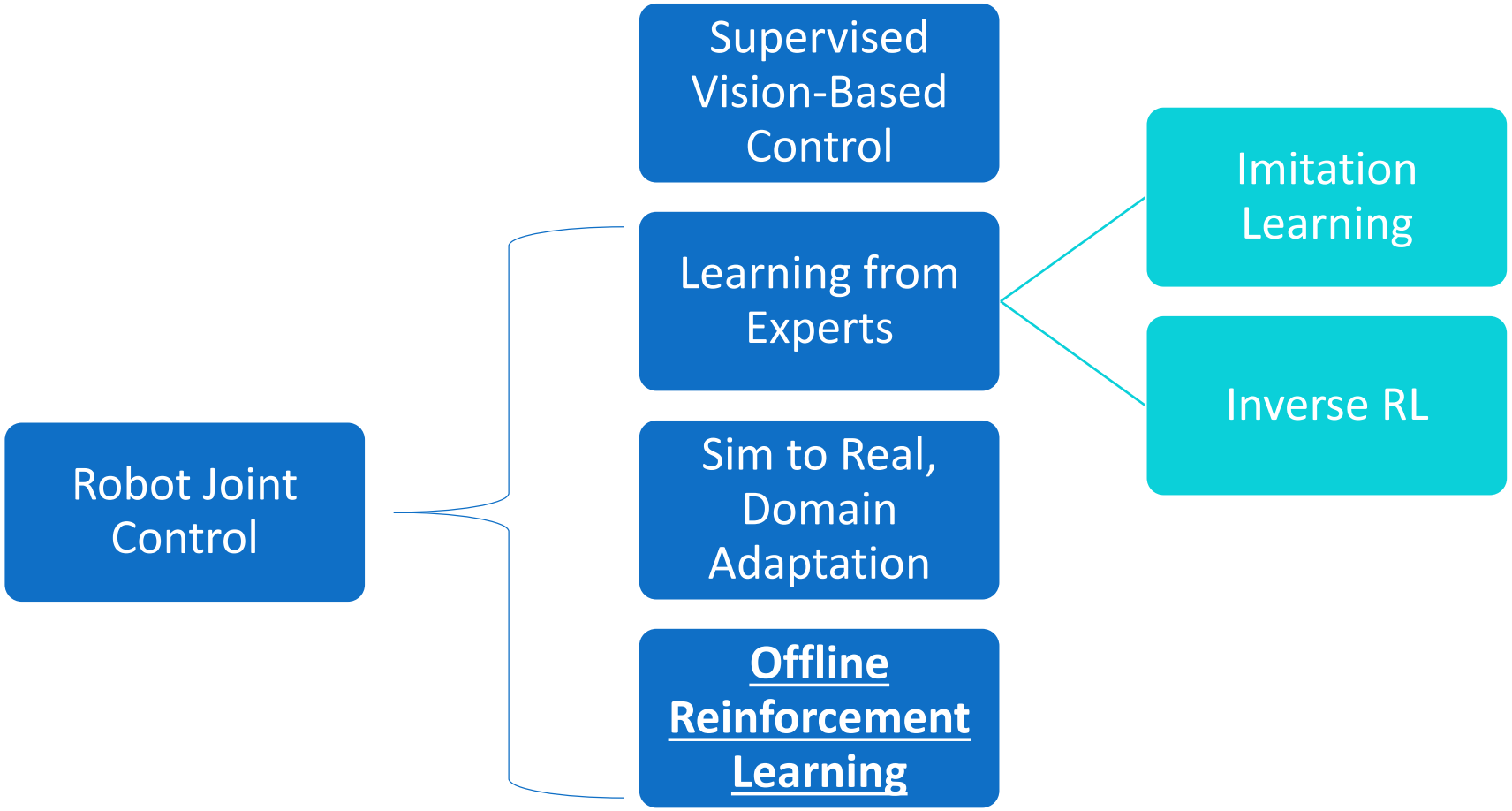
- Main Idea: Learn to manipulate objects with domain randomization in simulator, then transfer to real world
  - Domain Randomization: Add Gaussian noise to observations, randomize object dimensions, arm masses, friction coefficients, joint limits, gravity effects, etc. all in simulation
- Randomize appearance

1. Collect experience in randomized simulator
2. Train RL policy based on experience
3. Train a CNN pose predictor
4. Combine both to be able to recreate target object poses in the real world



Demo

Outline of Today's Lecture



Offline (Batch) RL

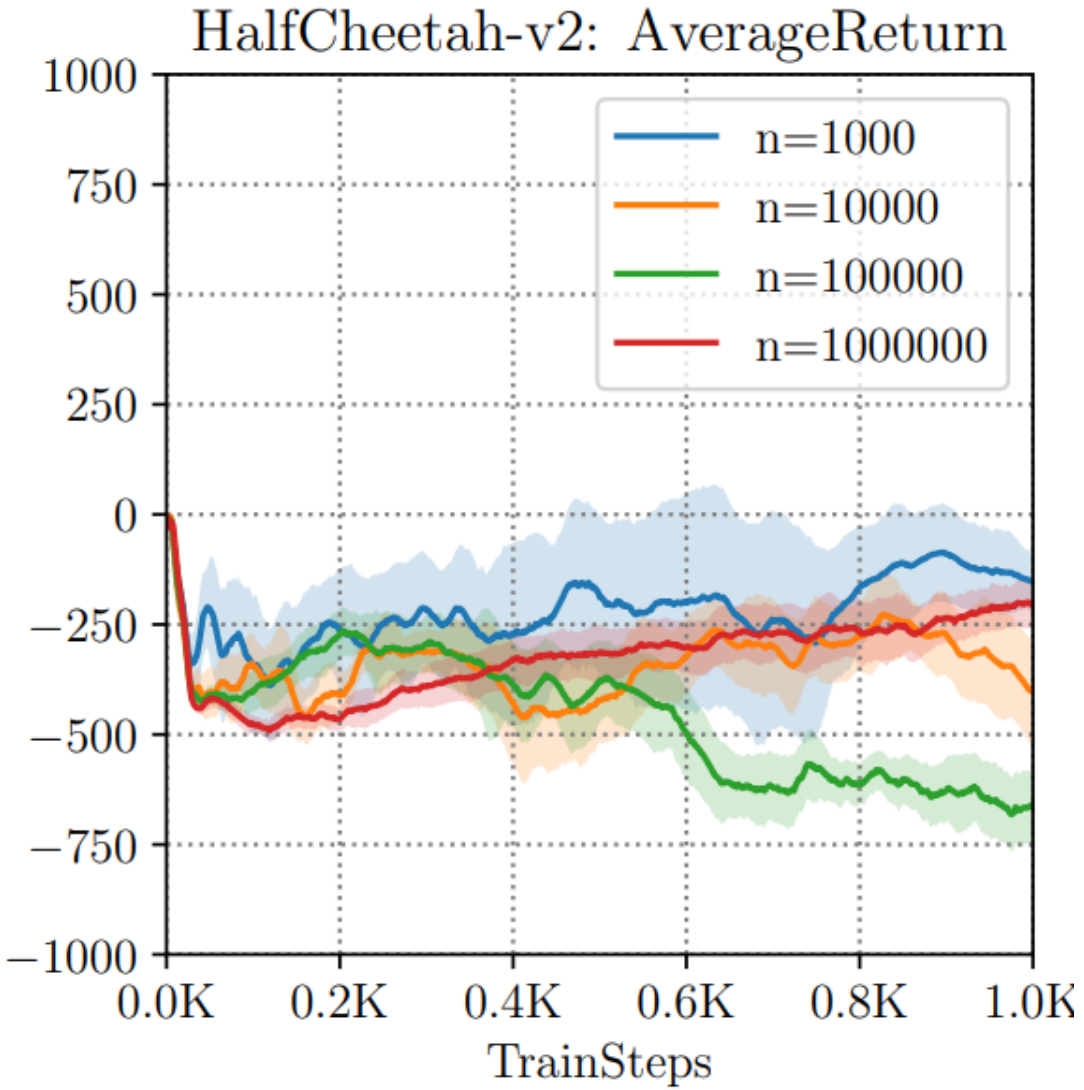
How do we perform RL fully *offline,* with no environment interactions?

Ideal for situations in which exploring in the real-world is problematic (hospital surgical robot, self driving, etc) but we can collect data.

- Given large dataset D, collected by any arbitrary mix of policies, how can we learn RL policies without any environment interaction?
- Why? Large datasets (ImageNet) are behind the success of DL in vision
- Why this problem formulation?
  - Offline RL can generalize to be **better** than the performance of the dataset
  - Imitation Learning generally can not

Attempt 1: Naïve Offline RL

- Apply generic, off-policy RL algorithm to the existing dataset
  - In the training loop, skip the exploration step. Sample from dataset for trajectories to train on
- Doesn't perform well!
- Why?



Kumar et al, NeurIPS 2019 (BEAR)

Attempt 1: Naïve Offline RL

- Assume a perfect Value/Q Function for our dataset D: $Q_D(s, a)$
- Assume a simple policy: $\max_a Q_D(s, a)$
- Problem: What if the dataset doesn't contain all actions?
  - A randomly initialized neural network $Q_D(s, a)$ function will have random values for out-of-distribution actions!
  - $argmax_a Q_D(s, a)$ could be an action never seen before with very low **true** Q-value.
- This isn't a problem in regular RL because the policy can explore these bad actions and learn their true value during training

Attempt 2: Constraints

- There are three types of fixes to this problem, take a guess?

1. Constrain the Policy:
   - Ensure the policy only picks actions that are in the distribution of D
   - $\pi_\theta = \arg\max_\theta E_{a\sim\pi_\theta(a|s)}[Q(s,a)] \;\; s.t. \; dist\left(\pi_\theta(a|s), \pi_\beta(a|s)\right) \leq \epsilon$
     - π_β can be a policy learned with behavior cloning on the data
     - Dist can be KL divergence or other types of probability distribution distances
2. Constrain the Q function:
   - Ensure the Q values for actions outside the distribution are low
3. Collect a ginormous dataset and hope for the best

The Bellman Backup, Revisited

- Bellman Backup: $Q_{i+1}(s,a) = E_{s'}[r + \gamma \max'_a Q_i(s',a')]$
  - Iteratively update $Q(s,a)$ with the optimal Bellman Equations
  - With an arbitrary policy in continuous spaces: $Q_{i+1}(s,a) = E_{s',a' \sim \pi(a|s)}[r + \gamma Q_i(s',a')]$
- Distribution constraint example: $\max_\theta E_{a \sim \pi_\theta(a|s)}[Q(s,a)] \;\; s.t. \; KL\left(\pi_\theta(a|s), \pi_\beta(a|s)\right) \leq \epsilon$
  - Becomes $Q_{i+1}(s,a) = E_{s',a' \sim \pi_\theta(a|s)}[r + \gamma \max'_a Q_i(s',a') + \log \pi_\beta(a|s) - \log \pi_\theta(a'|s')]$
  - Intuition: Increase Q values by $\log \pi_\beta(a|s)$, the behavior cloned policy -> increase Q values by amount proportional to frequency in dataset
- Policy update example: $\pi(s) = argmax_{a_i} Q(s,a_i)$
  - Constrain the policy: $\pi_{BCQ}(s) = argmax_{a_i + \xi(s,a_i)} Q\left(s, a_i + \xi(s,a_i)\right)$
  - $a_i \sim G(s)$, $G(s)$ a VAE trained on D, $\xi(s,a_i)$ a perturbation trained to maximize $Q(s, a_i + \xi(s,a_i))$

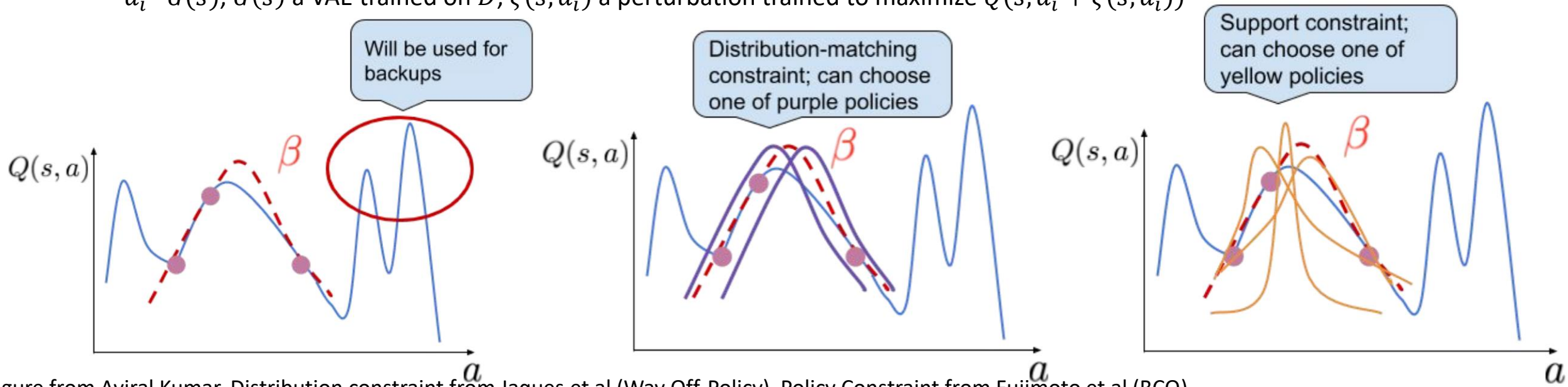


Figure from Aviral Kumar, Distribution constraint from Jaques et al (Way Off-Policy), Policy Constraint from Fujimoto et al (BCQ)

How well does Offline RL work?

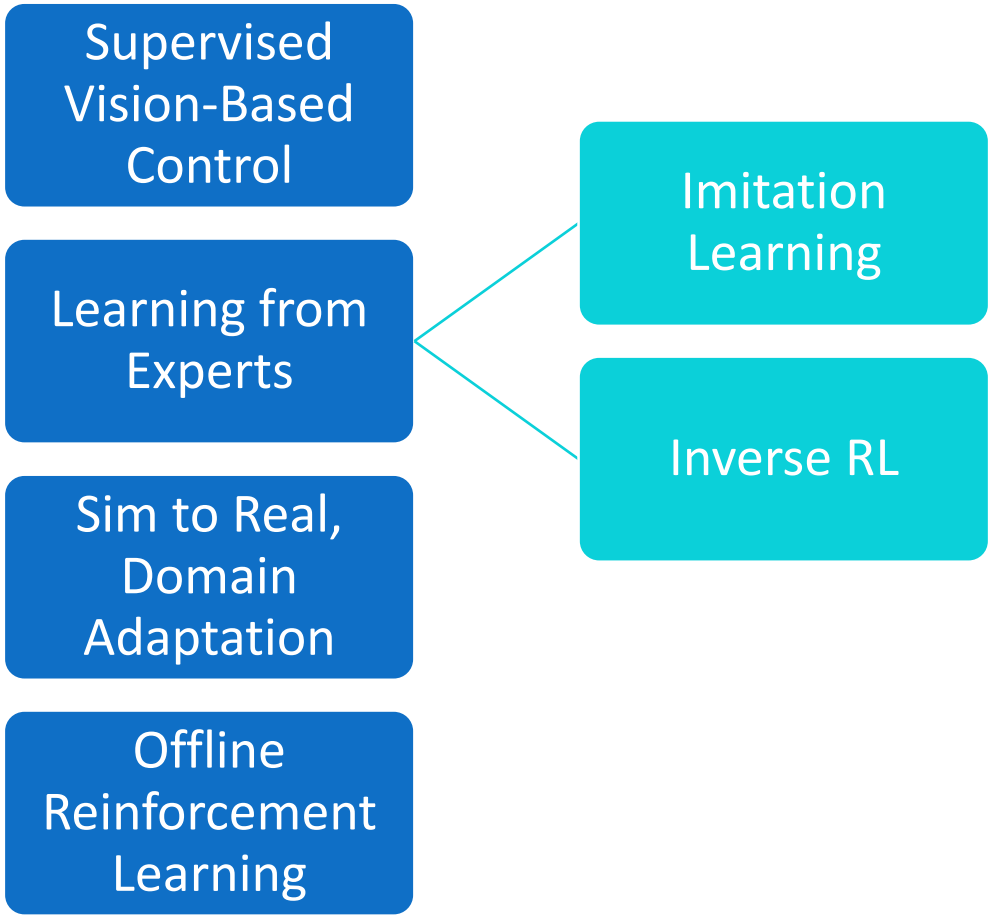
Domain	Task Name	BC	SAC	BEAR	BRAC-p	BRAC-v	CQL($\mathcal{H}$)	CQL(ρ)
AntMaze	antmaze-umaze	65.0	0.0	**73.0**	50.0	70.0	**74.0**	**73.5**
	antmaze-umaze-diverse	55.0	0.0	61.0	40.0	70.0	**84.0**	61.0
	antmaze-medium-play	0.0	0.0	0.0	0.0	0.0	**61.2**	4.6
	antmaze-medium-diverse	0.0	0.0	8.0	0.0	0.0	**53.7**	5.1
	antmaze-large-play	0.0	0.0	0.0	0.0	0.0	**15.8**	3.2
	antmaze-large-diverse	0.0	0.0	0.0	0.0	0.0	**14.9**	2.3
Adroit	pen-human	34.4	6.3	-1.0	8.1	0.6	37.5	**55.8**
	hammer-human	1.5	0.5	0.3	0.3	0.2	**4.4**	2.1
	door-human	0.5	3.9	-0.3	-0.3	-0.3	**9.9**	**9.1**
	relocate-human	0.0	0.0	-0.3	-0.3	-0.3	0.20	**0.35**
	pen-cloned	**56.9**	23.5	26.5	1.6	-2.5	39.2	40.3
	hammer-cloned	0.8	0.2	0.3	0.3	0.3	2.1	**5.7**
	door-cloned	-0.1	0.0	-0.1	-0.1	-0.1	0.4	**3.5**
	relocate-cloned	**-0.1**	-0.2	-0.3	-0.3	-0.3	**-0.1**	**-0.1**
Kitchen	kitchen-complete	33.8	15.0	0.0	0.0	0.0	**43.8**	31.3
	kitchen-partial	33.8	0.0	13.1	0.0	0.0	**49.8**	**50.1**
	kitchen-undirected	47.5	2.5	47.2	0.0	0.0	**51.0**	**52.4**

Table from Kumar et al, 2020 (CQL)

Grasping, part 2

Kalashnikov et al, 2018 (QT-Opt)

In Summary



- Supervised vision-based control allows us to perform tasks like vision-based grasping, but joint and vision-based visuomotor control is needed for more complicated tasks
- We can learn from experts (e.g. humans), by cloning demonstrations, being able to query humans, or estimating experts' reward functions to learn policies to behave like the expert
- We can also perform sim-to-real transfer using domain adaptation techniques to train in simulators and execute tasks on the real-world robots
- Offline RL is a promising approach where we can learn, fully offline, RL policies that may generalize beyond the performance of the given datasets

Questions?