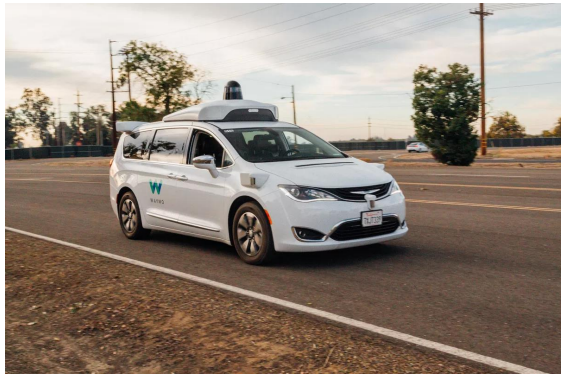



From Inference to Action: AI beyond Pattern Recognition

Pieter Abbeel

@pabbeel

UC Berkeley	AI research and teaching (professor)
Covariant	AI for robotic automation (founder)

Also on advisory board of many AI and robotics companies

Neural Net Learning Image Recognition

- Training / Learning:
  - Cars:
  - Cats:
  - Dogs:

labeled data

Machine Learning

- Test time:

→ Label = Cat? Dog? Car?

Neural Net Image Recognition

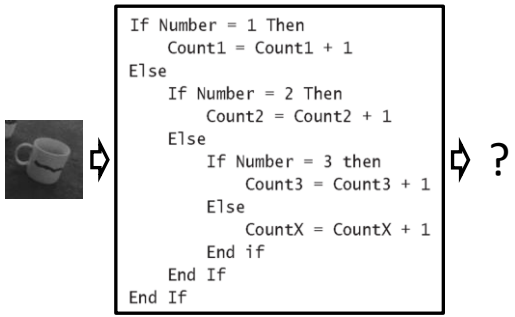
'girl in pink dress is jumping in air.'

'black and white dog jumps over bar.'

'young girl in pink shirt is swinging on swing.'

Change in Programming Paradigm!

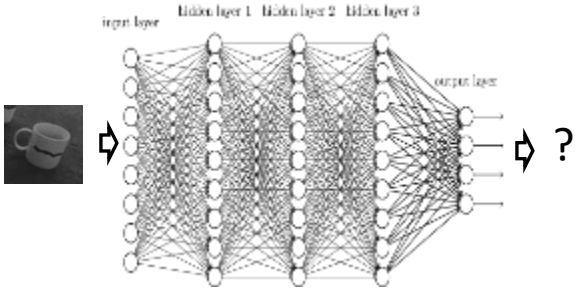
Traditional Programming: program by writing lines of code



```
If Number = 1 Then
    Count1 = Count1 + 1
Else
    If Number = 2 Then
        Count2 = Count2 + 1
    Else
        If Number = 3 then
            Count3 = Count3 + 1
        Else
            CountX = CountX + 1
        End if
    End If
End If
```

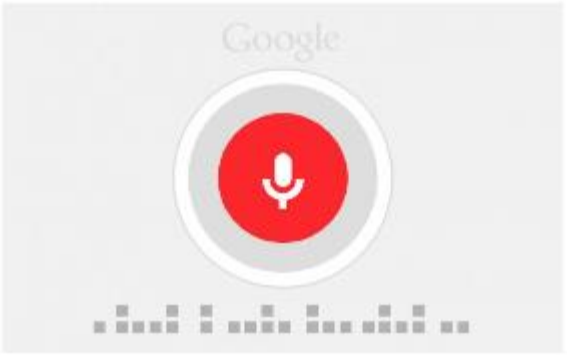
Poor performance

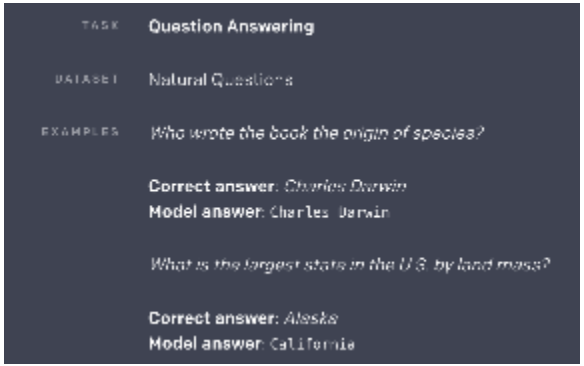
Deep Learning ("Software 2.0") program by providing data

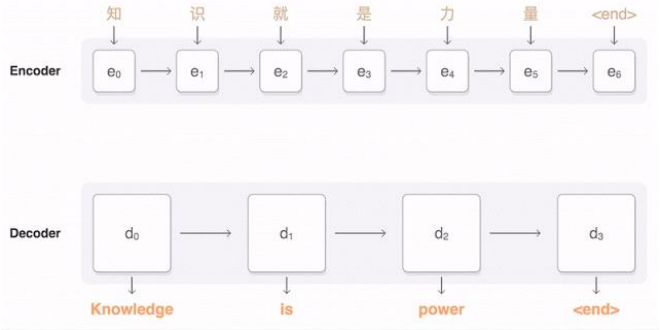


Success!

Deep Supervised Learning

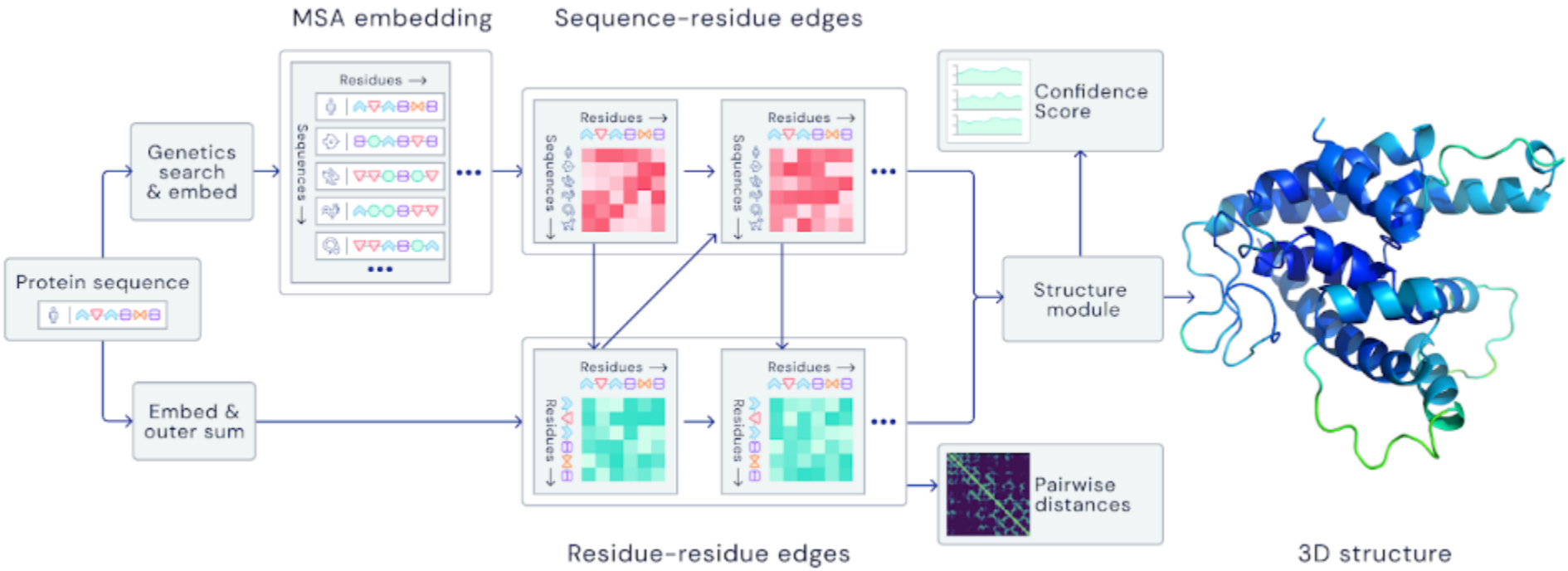






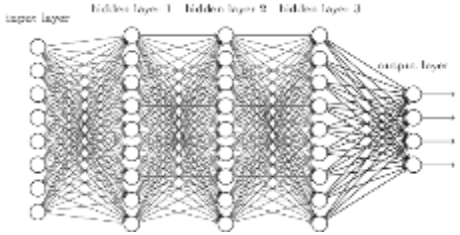
Equation	Solution
$y' = \frac{16x^3 - 42x^2 + 2x}{(-16x^8 + 112x^7 - 204x^6 + 28x^5 - x^4 + 1)^{1/2}}$	$y = \sin^{-1}(4x^4 - 14x^3 + x^2)$
$3xy\cos(x) - \sqrt{9x^2\sin(x)^2 + 1}y' + 3y\sin(x) = 0$	$y = c\exp\left(\sinh^{-1}(3x\sin(x))\right)$
$4x^4yy'' - 8x^4y'^2 - 8x^3yy' - 3x^3y'' - 8x^2y^2 - 6x^2y' - 3x^2y'' - 9xy' - 3y = 0$	$y = \frac{c_1 + 3x + 3\log(x)}{x(c_2 + 4x)}$

Deep Supervised Learning



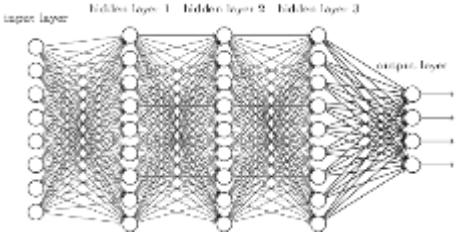
Key Assumption

Request 1



Decision 1

Request 2

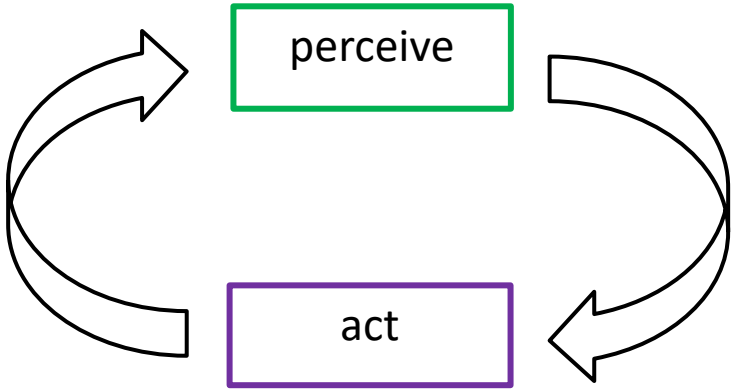


Decision 2

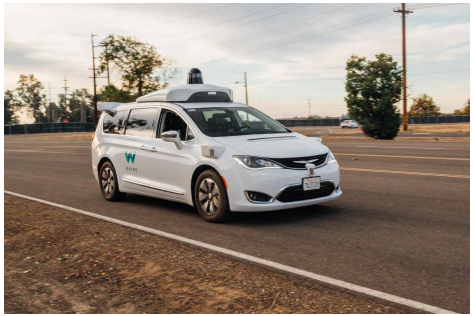
...

Key Assumption: Decision 1 does NOT affect Request 2, …

But: Reality of Perception-Action Loop



E.g.



...

Outline

- Beyond pattern recognition: reinforcement learning
- Addressing sample complexity
- Addressing reward design
- Where are we with real-world applications

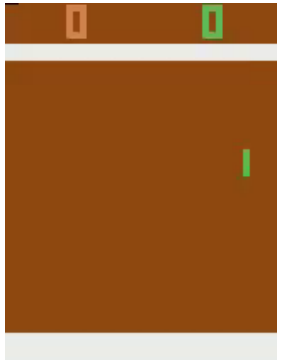
Outline

- Beyond pattern recognition: reinforcement learning
  - Recent successes
  - When does it apply?
  - How does it learn?
  - Some remaining key challenges
- Addressing sample complexity
- Addressing reward design
- Where are we with real-world applications

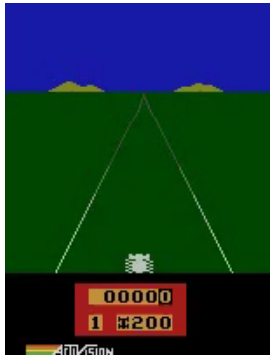
Fast Progress on Deep RL

2013

Atari (DQN)
[Deepmind]

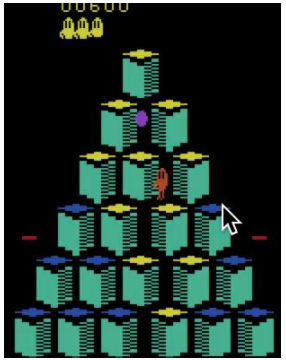


Pong



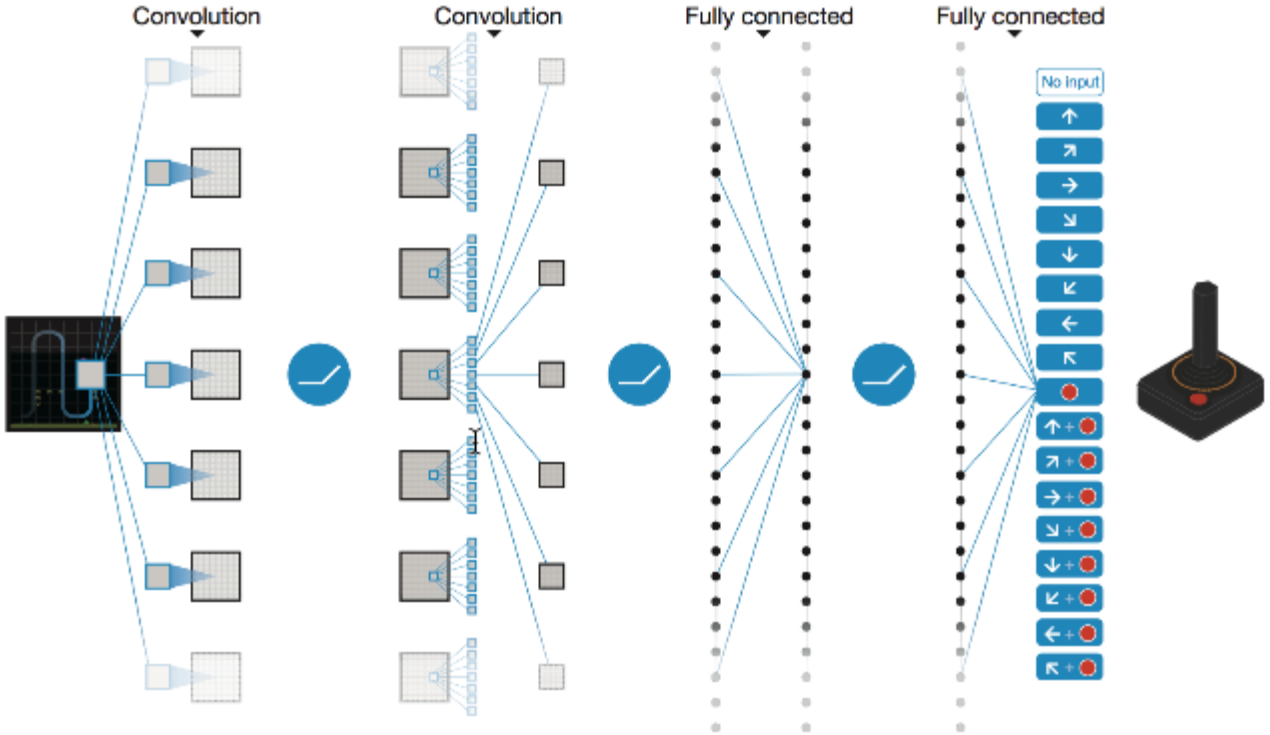
Enduro

Beamrider



Q*bert

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



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

Fast Progress on Deep RL

2013 Atari (DQN) [Deepmind]

2015 AlphaGo [Deepmind]

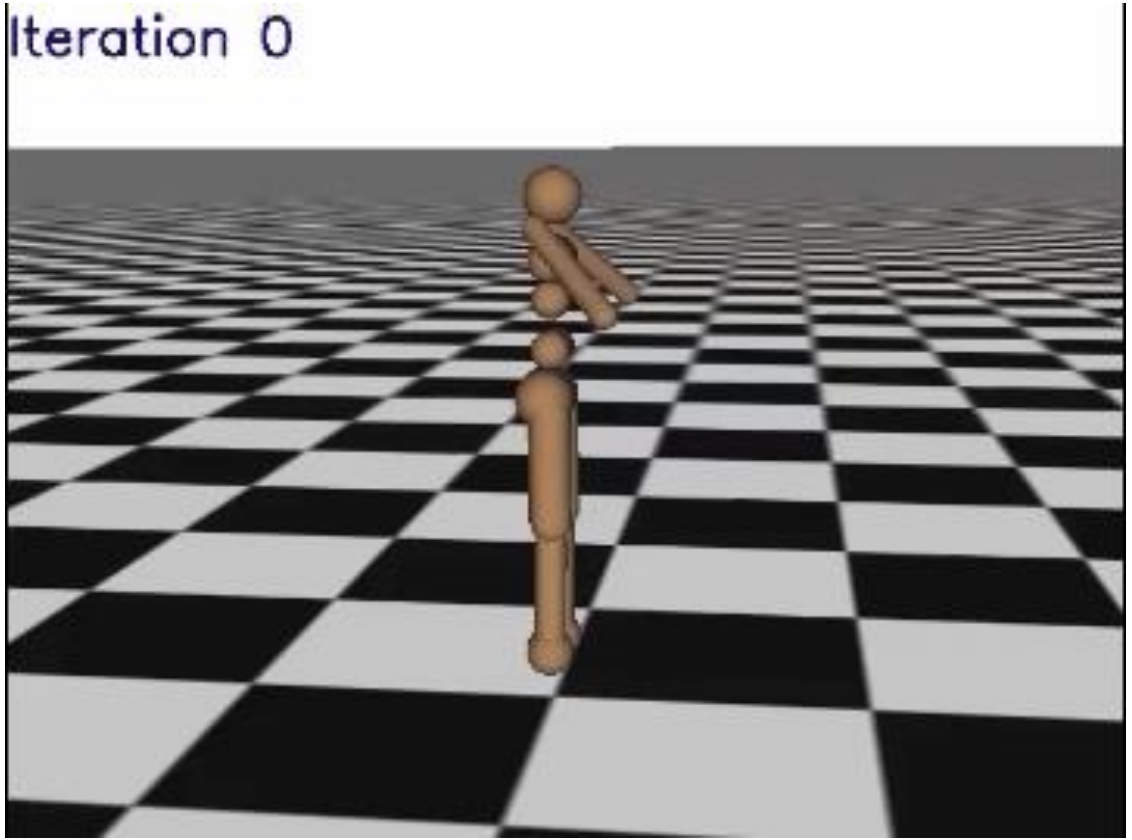
AlphaGo Silver et al, Nature 2015
AlphaGoZero Silver et al, Nature 2017
AlphaZero Silver et al, 2017
Tian et al, 2016; Maddison et al, 2014; Clark et al, 2015

Fast Progress on Deep RL

2013 Atari (DQN) [Deepmind]

2015 AlphaGo [Deepmind]

2016 **3D locomotion (TRPO+GAE) [Berkeley]**



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

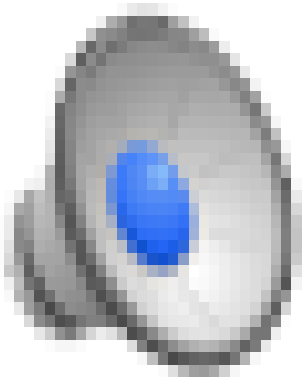
Fast Progress on Deep RL

2013 Atari (DQN) [Deepmind]

2015 AlphaGo [Deepmind]

2016 3D locomotion (TRPO+GAE) [Berkeley]

2016 Real Robot Manipulation (GPS) [Berkeley]



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

Fast Progress on Deep RL

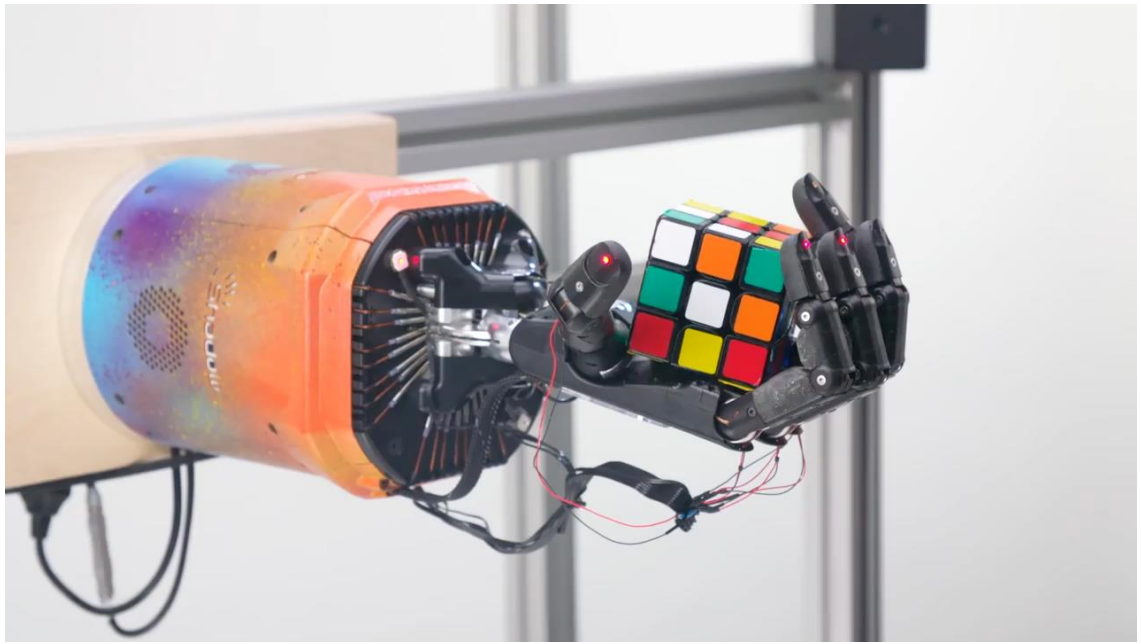
2013 Atari (DQN) [Deepmind]

2015 AlphaGo [Deepmind]

2016 3D locomotion (TRPO+GAE) [Berkeley]

2016 Real Robot Manipulation (GPS) [Berkeley]

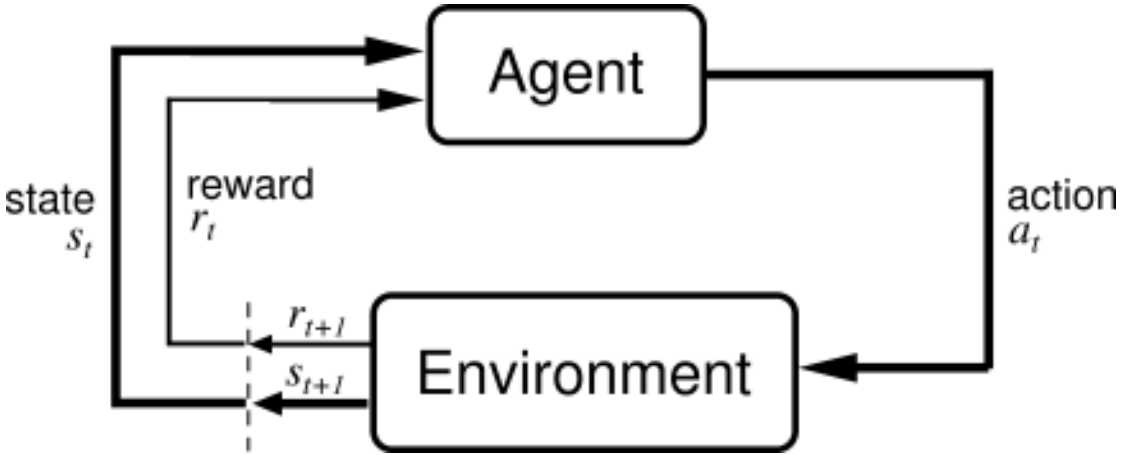
2019 Rubik's Cube (PPO+DR) [OpenAI]



Outline

- Beyond pattern recognition: reinforcement learning
  - Recent successes
  - ***When does it apply?***
  - How does it learn?
  - Some remaining key challenges
- Addressing sample complexity
- Addressing reward design
- Where are we with real-world applications

Formally: Markov Decision Process



Assumption: agent gets to observe the state

[Drawing from Sutton and Barto, Reinforcement Learning: An Introduction, 1998]

Less Formally

- If you can let the neural net ”play” / “control the system” and generate roll-outs of the form:

  s0, a0, r0, s1, a1, r1, …, sH

e.g.

– s = robot configuration; a = motor commands; r = distance covered

-- s = what’s on screen; a = joystick command; r = score

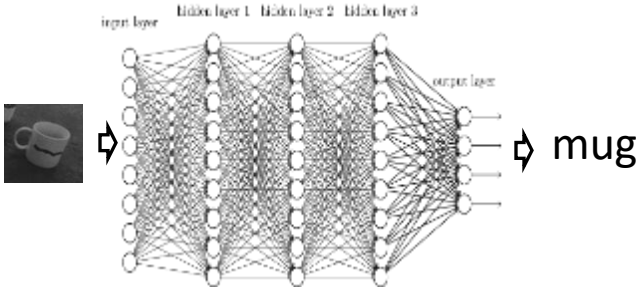
-- s = conversation so far; a = next reply; r = customer satisfaction

-- …

Outline

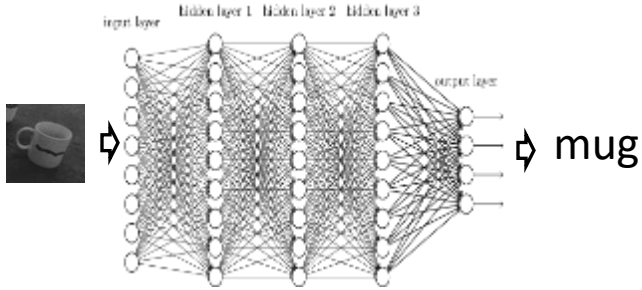
- Beyond pattern recognition: reinforcement learning
  - Recent successes
  - When does it apply?
  - ***How does it learn?***
  - Some remaining key challenges
- Addressing sample complexity
- Addressing reward design
- Where are we with real-world applications

Recall Supervised Learning

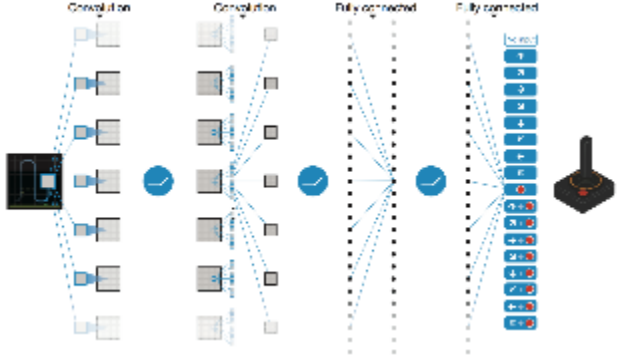


$$\text{maximize} \quad \sum_i \log P(\text{label}_i \mid \text{image}_i, \text{NNparams})$$

Reinforcement Learning



$$\text{maximize} \quad \sum_i \log P(\text{label}_i \mid \text{image}_i, \text{NNparams})$$



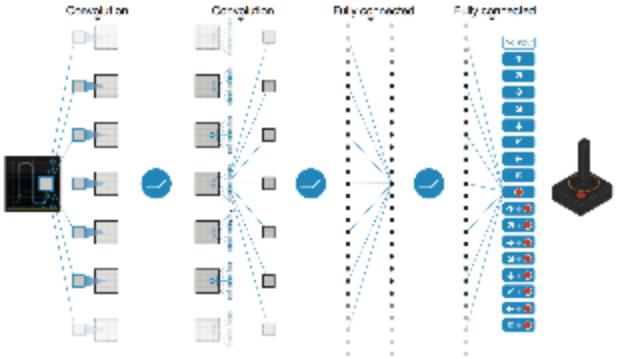
$$\text{maximize} \quad \sum_t (R_{t:H} - \text{avg}) \log P(a_t \mid \text{image}_t, \text{NNparams})$$

= POLICY GRADIENT METHOD

Reinforcement Learning

Step 1: Collect roll-outs with the current neural net

Step 2: Update the neural net by optimizing:



maximize
sum_t (R_{t:H} – avg) log P(a_t | image_t, NNparams)

Step 3: Go back to Step 1

= POLICY GRADIENT METHOD

Spectrum of RL Methods

- POLICY GRADIENT / PPO / TRPO
  - Only use most recent data (“on-policy”)
  - Tends to require more data (due to squeezing less out of old data)
  - But if data is easy to get (e.g. because learning in fast simulator), will be fastest to achieve great performance in terms of wall-clock time (because cycles all spent on latest data)
- DQN
  - Maximally uses old data, hence most data efficient (“off-policy”)
  - But can be unstable (because more difficult to improve based on older data)
- SAC / TD3
  - Middleground between above two
- Model-based RL -- also learn simulator from collected data, can do trial/error runs in learned sim

Offline RL / Batch RL

- Collect data only once (possibly from human or other process)
- Optimize the Neural Net once

Why also important?

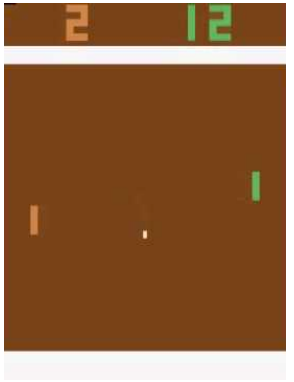
- Sometimes only data available

Outline

- Beyond pattern recognition: reinforcement learning
  - Recent successes
  - When does it apply?
  - How does it learn?
  - ***Some remaining key challenges***
- Addressing sample complexity
- Addressing reward design
- Where are we with real-world applications

Key Challenges for RL

- Sample complexity of trial and error learning
  - Game of Pong: 40 days (if played in realtime)
  - AlphaGoZero: 4.9 million games
  - …
- Reward design
  - Easy to end up with erratic behavior



Outline

- Beyond pattern recognition: reinforcement learning
- Addressing sample complexity
  - ***Sim2real with Domain Randomization***
  - Bootstrap through Imitation
  - Self-supervised Representation Learning
  - Case Study FERM
- Addressing reward design
- Where are we with real-world applications

Motivation for Simulation

Compared to the real world, simulated data collection is…

- Less expensive
- Faster / more scalable
- Less dangerous
- Easier to label

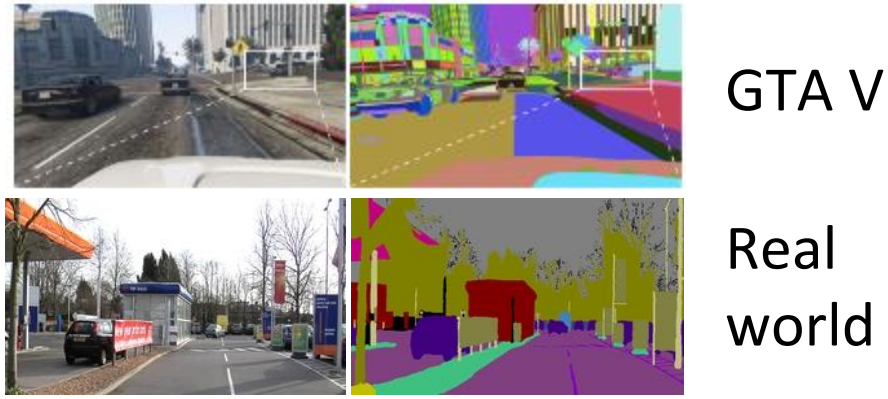
But: How can we learn useful real-world skills in the simulator?

Approach 1 – Use Realistic Simulated Data

Simulation

Real world

Carefully match the simulation to the world [1,2,3,4]



GTA V

Real world

Augment simulated data with real data [5,6]

[1] Stephen James, Edward Johns. *3d simulation for robot arm control with deep q-learning (2016)*

[2] Johns, Leutenegger, Davision. Deep learning a grasp function for grasping under gripper pose uncertainty (2016)

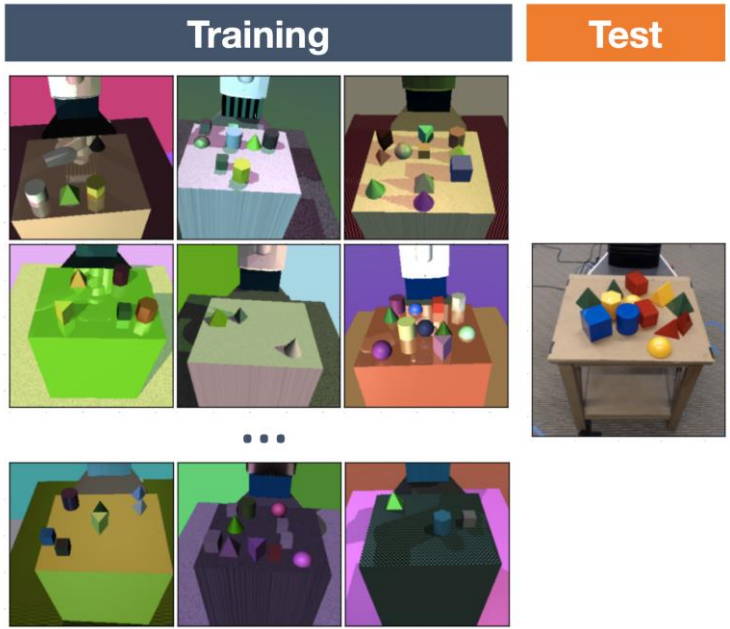
[3] Mahler et al, Dex-Net 3.0 (2017)

[4] Koenemann et al. Whole-body model-predictive control applied to the HRP-2 humanoid. (2015)

[5] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. *Playing for data: Ground truth from computer games (2016)*

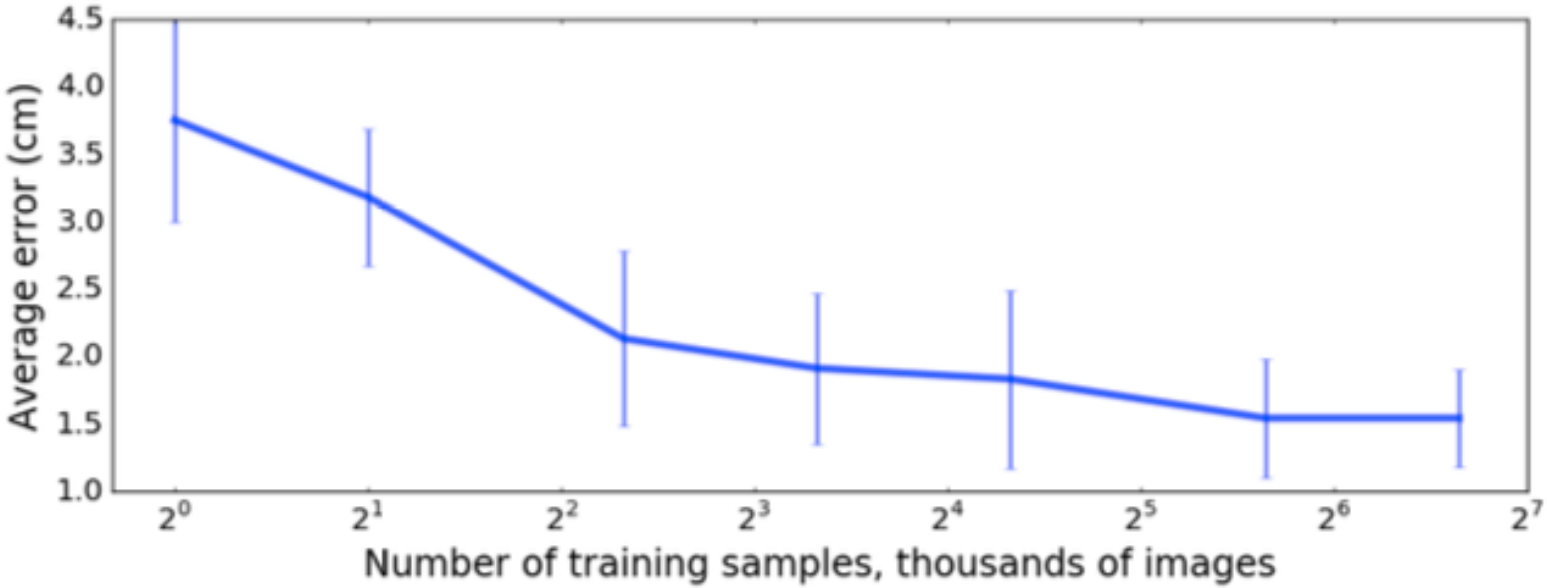
[6] Bousmalis et al. Using simulation and domain adaptation to improve efficiency of robotic grasping (2017)

Approach 2 – Domain Randomization



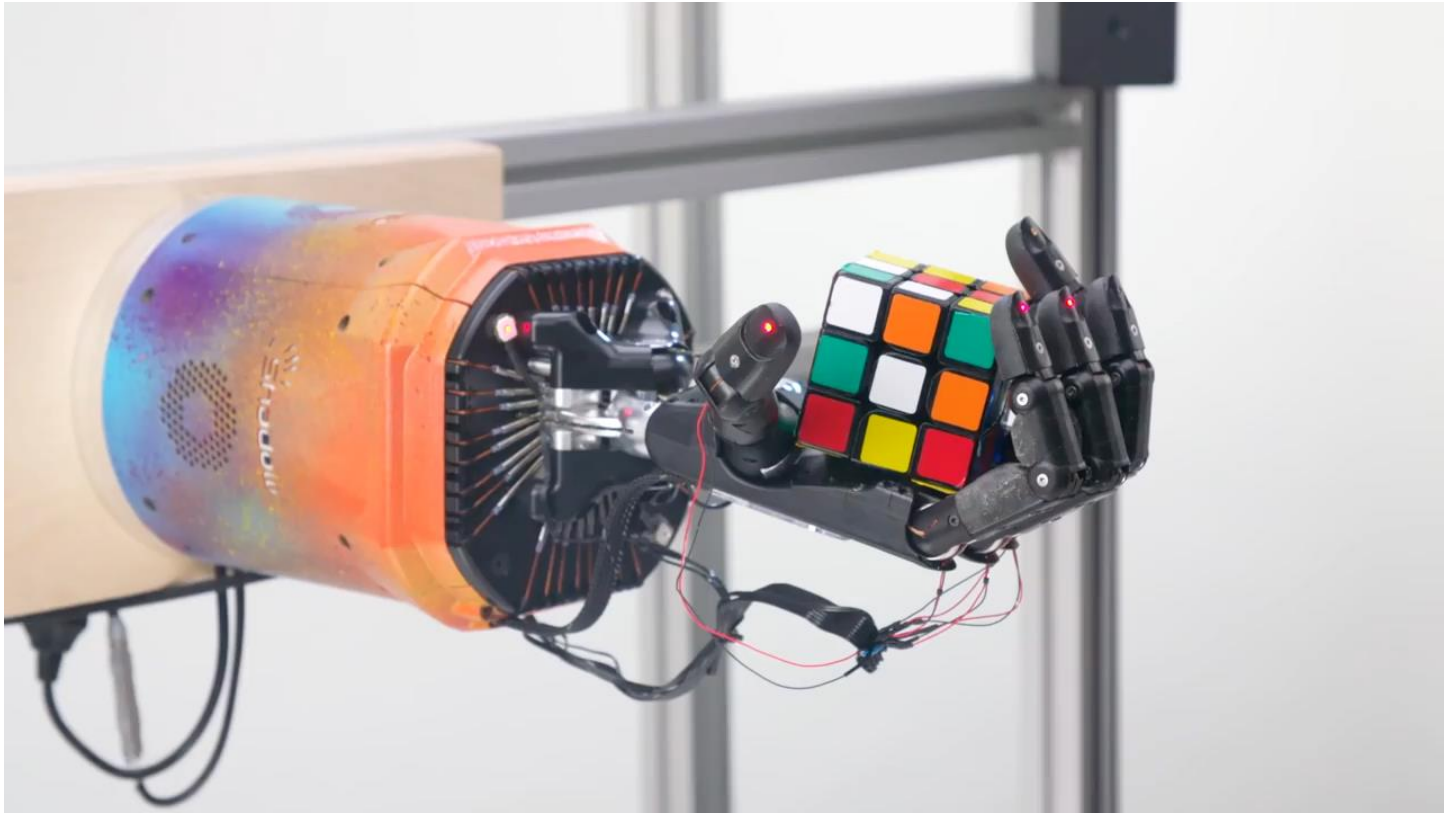
If the model sees enough simulated variation, the real world may look like just the next simulator

How does it work? More Data = Better



[Tobin, Fong, Ray, Schneider, Zaremba, Abbeel, 2017]

How About a Full Hand?



[OpenAI]

Outline

- Beyond pattern recognition: reinforcement learning
- Addressing sample complexity
  - Sim2real with Domain Randomization
  - ***Bootstrap through Imitation***
  - Self-supervised Representation Learning
  - Case Study FERM
- Addressing reward design
- Where are we with real-world applications

Bootstrapping through Imitation

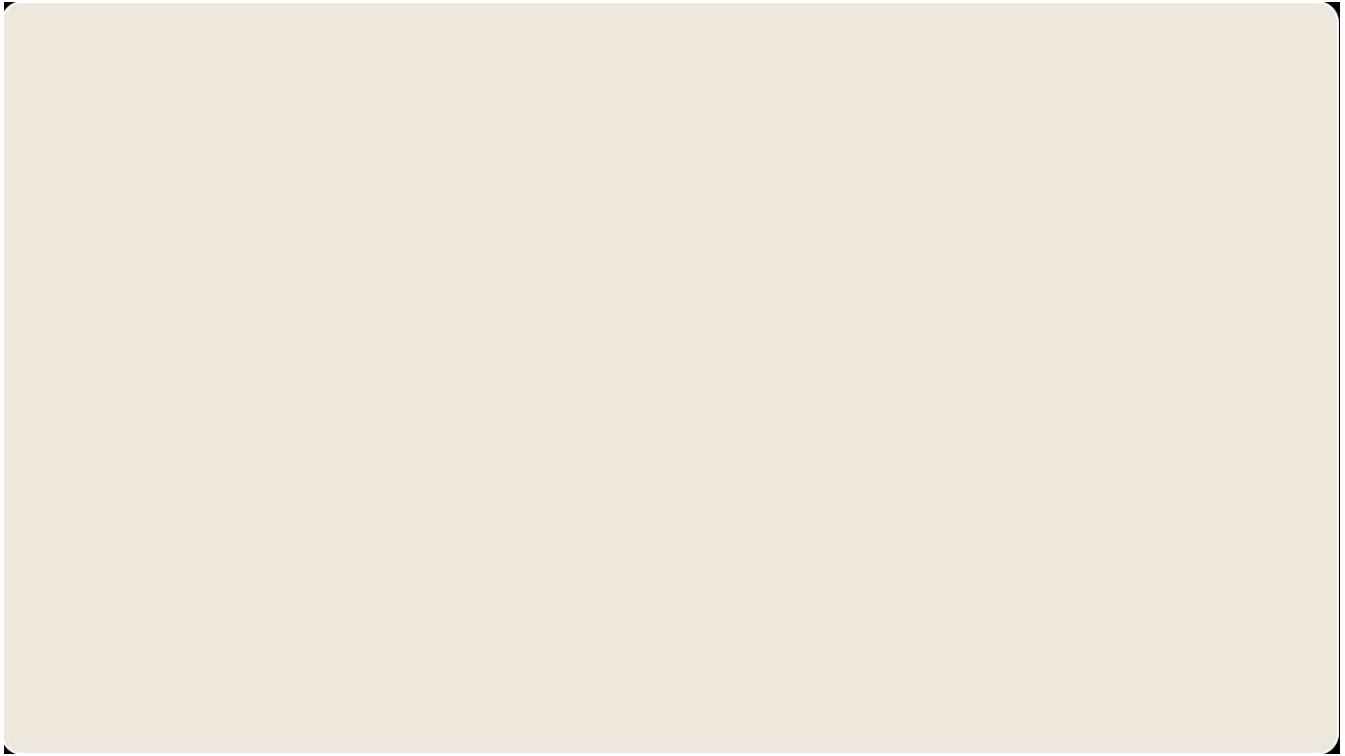
- Collect demonstrations
- Run supervised learning to learn the mapping
  - $s_t \rightarrow a_t$
- Use resulting policy as initialization for reinforcement learning

Note: can also learn Q and/or V instead of policy

Imitation Learning -- Driving

End to End Learning for Self-Driving Cars, Bojarski et al. 2016

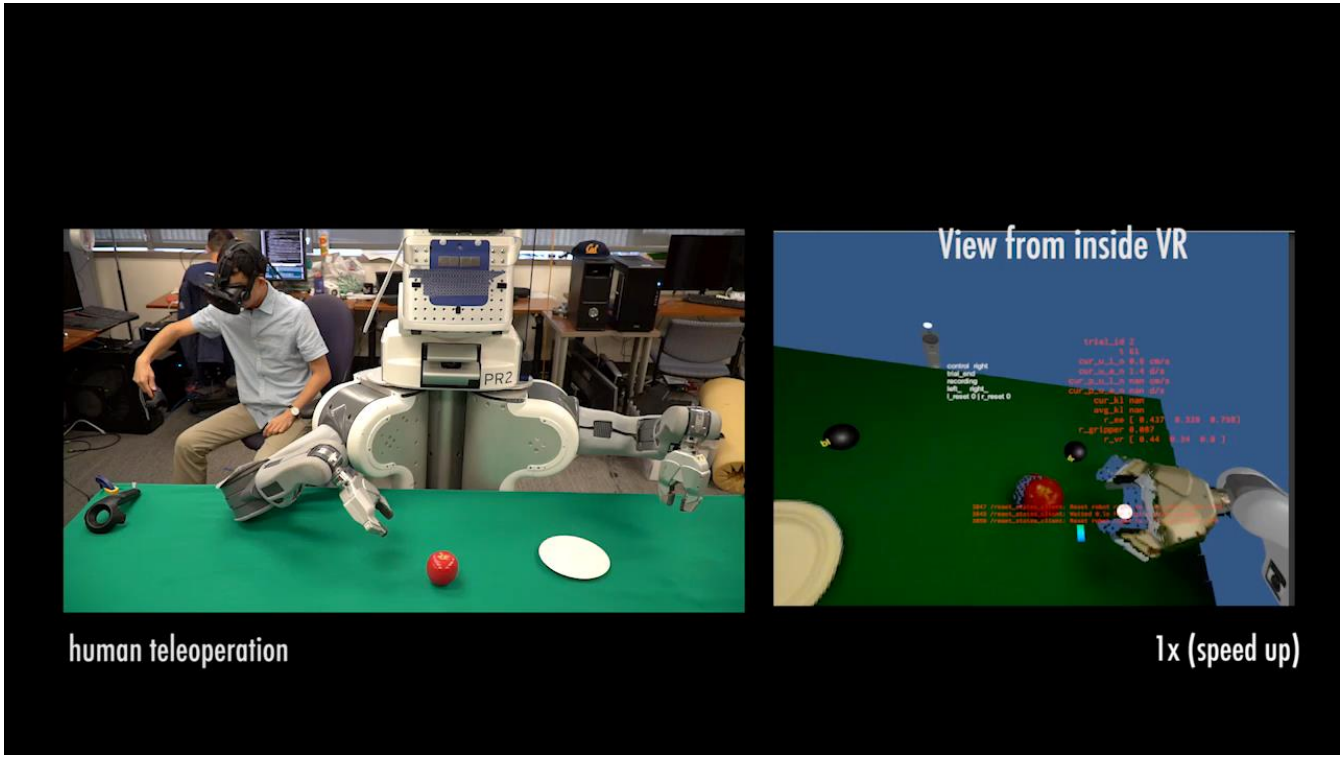
Imitation Learning -- Drones



Play: 1:12 – 1:35

Learning Monocular Reactive UAV Control in Cluttered Natural Environments, Ross et al. 2013

Imitation Learning – VR tele-op



[Zhang, McCarthy, Jow, Lee, Chen, Goldberg, Abbeel, ICRA 2018]

Imitation Learning for Conversational AI

- E.g. GPT models

The researcher is a human male named Kirk Ouimet.

***Researcher**: Grateful to talk with you again.*

Wise Being: I am also grateful for this opportunity.

***Researcher**: Let's talk about love.*

Wise Being: Love is important.

***Researcher**: Why is that?*

Wise Being: Love allows people to survive and thrive. It allows families to come together and stay together.

***Researcher**: How does it do that?*

Wise Being: Love allows parents to be very patient with each other. It provides motivation for children to learn. It gives people the confidence they need to feel good about themselves.

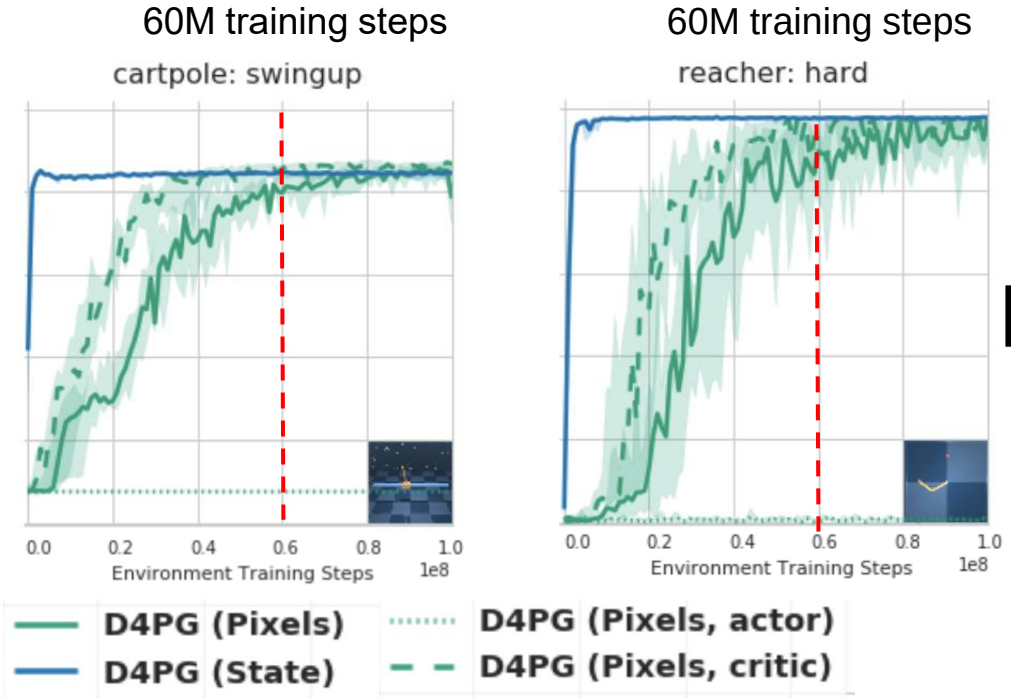
- But recall: no objective / intent, just imitation

Outline

- Beyond pattern recognition: reinforcement learning
- Addressing sample complexity
  - Sim2real with Domain Randomization
  - Bootstrap through Imitation
  - ***Self-supervised Representation Learning***
  - Case Study FERM
- Addressing reward design
- Where are we with real-world applications

Can visual RL match data-efficiency of state RL?

- State-based D4PG (blue) vs pixel-based D4PG (green)

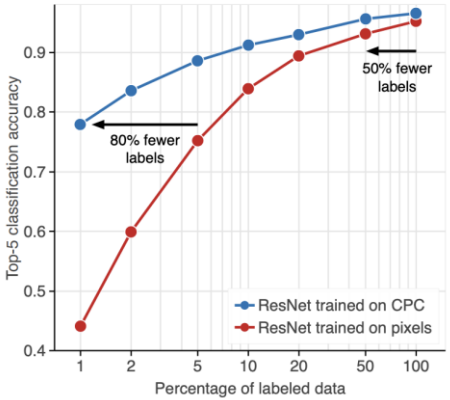


Pixel-based needs > 50M more training steps than state-based to solve same tasks

[Tassa et al., 2018] Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D.D.L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A. and Lillicrap, T DeepMind Control Suite, arxiv:1801.00690, 2018.

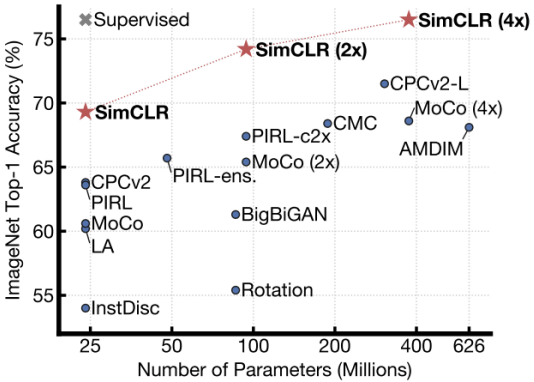
Contrastive learning: SOTA in computer vision

CPCv2 **top-5** ImageNet accuracy as function of labels



[Henaff, Srinivas et al., 2019]

SimCLR **top-1** ImageNet accuracy as function of # of parameters



[Chen et al., 2020]

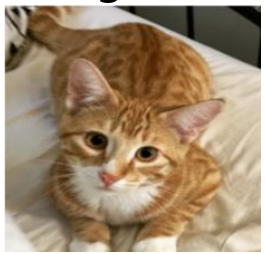
[Henaff et al., 2019] Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, Aaron van den Oord Data-Efficient Image Recognition with Contrastive Coding arxiv:1905.09272, 2019.

[Chen et al., 2020] Chen, T., Kornblith, S., Norouzi, M. and Hinton, G. A Simple Framework for Contrastive Learning of Visual Representations arxiv:2002.05709, 2020.

Contrastive Learning Main Idea

Original 1

Original 2



duplicate
+ semantically
invariant transform

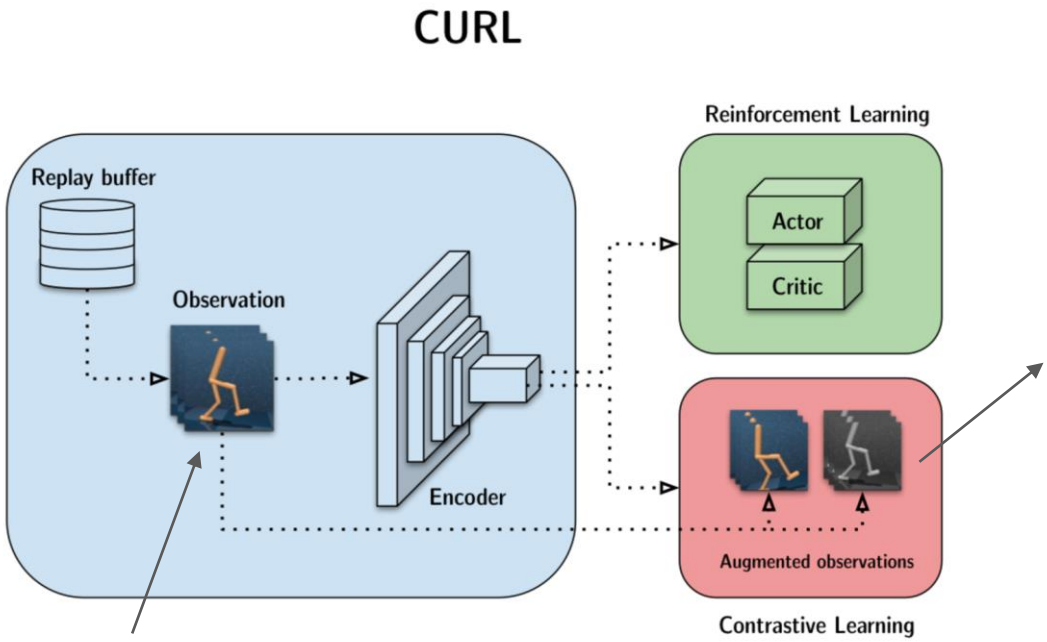
attract

repel

attract

Pieter Abbeel -- UC Berkeley | Covariant

Contrastive + RL



Need to define:

1. query / key pairs
2. similarity measure
3. architecture

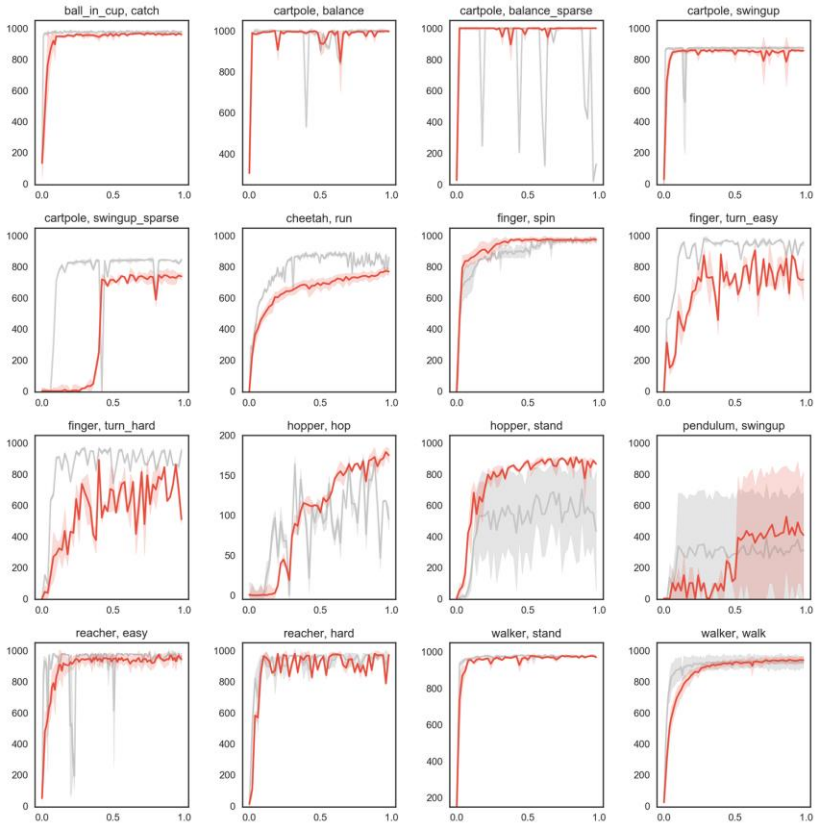
Observations are stacked frames

[CURL: A Srinivas*, M Laskin*, P Abbeel, 2020]

CURL from pixels matches state-based SAC

GRAY: SAC State

RED: CURL



[CURL: A Srinivas*, M Laskin*, P Abbeel, 2020]

CURL Comparison: Atari

Atari performance benchmarked at 100K frames

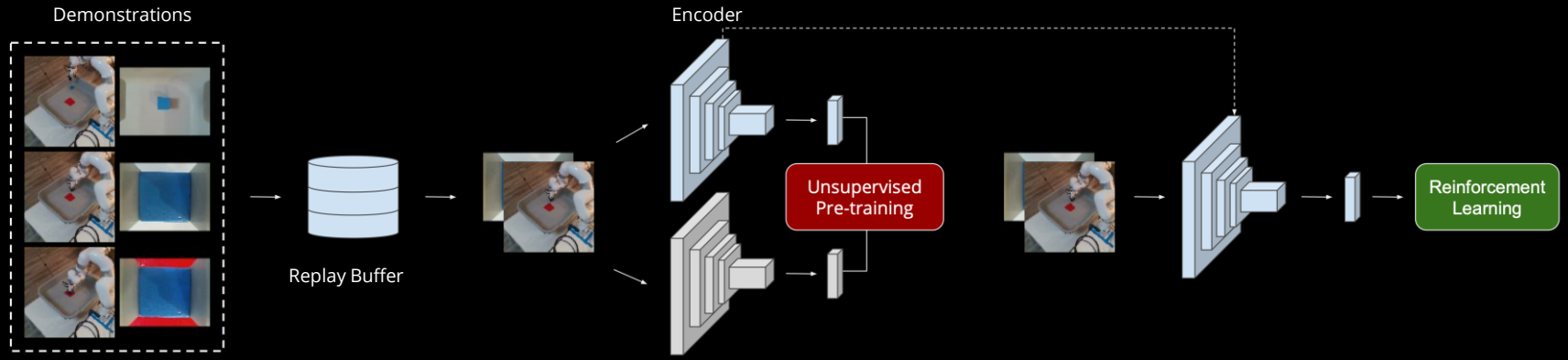
100K step scores	CURL Rainbow	SimPLe	Rainbow	Human	Random
Alien	**1148.2**	616.9	318.7	6875	184.8
Amidar	**232**	74.3	32.5	1676	11.8
Assault	473	**527.2**	231	1496	248.8
BattleZone	**11208**	4031.2	3285.71	37800	2895
Freeway	**27**	16.7	0	29.6	0
Frostbite	**924**	236.9	60.2	4335	74
Jamesbond	**400**	100.5	47.4	406.7	29.2
Qbert	**1352**	1288.8	123.46	13455	166.1
Seaquest	408	**683.3**	131.69	20182	61.1

[CURL: A Srinivas*, M Laskin*, P Abbeel, 2020]

Outline

- Beyond pattern recognition: reinforcement learning
- Addressing sample complexity
  - Sim2real with Domain Randomization
  - Bootstrap from Imitation
  - Self-supervised Representation Learning
  - ***Case Study FERM***
- Addressing reward design
- Where are we with real-world applications

Framework for Efficient Robotic Manipulation



(i) a few demonstrations

(ii) unsupervised representation learning

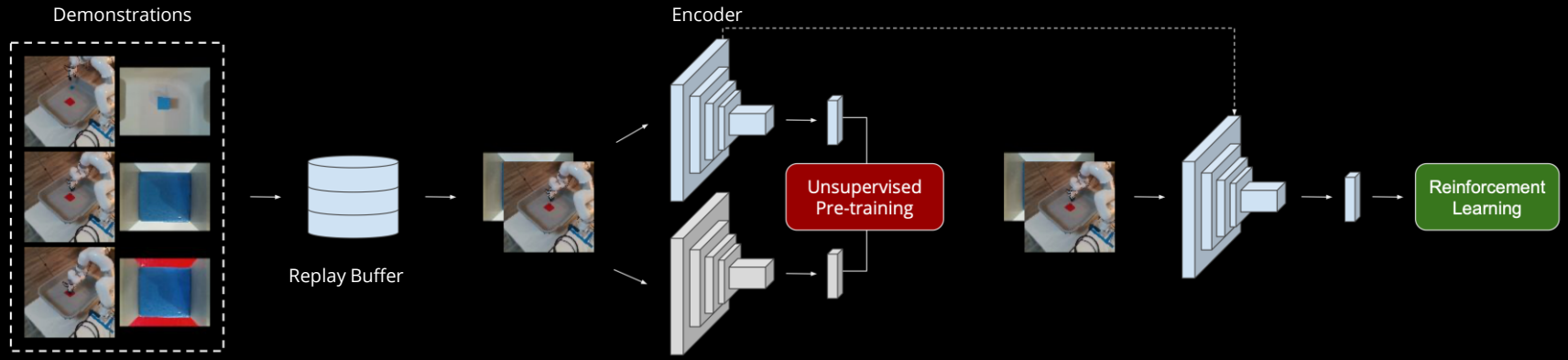
(iii) data augmentations

Takes ~10 mins

Takes ~1 min

Takes ~30 mins

Framework for Efficient Robotic Manipulation



(i) Collect 10 human demonstrations

Takes ~10 mins

(ii) Initialize CNN encoder with contrastive pre-training

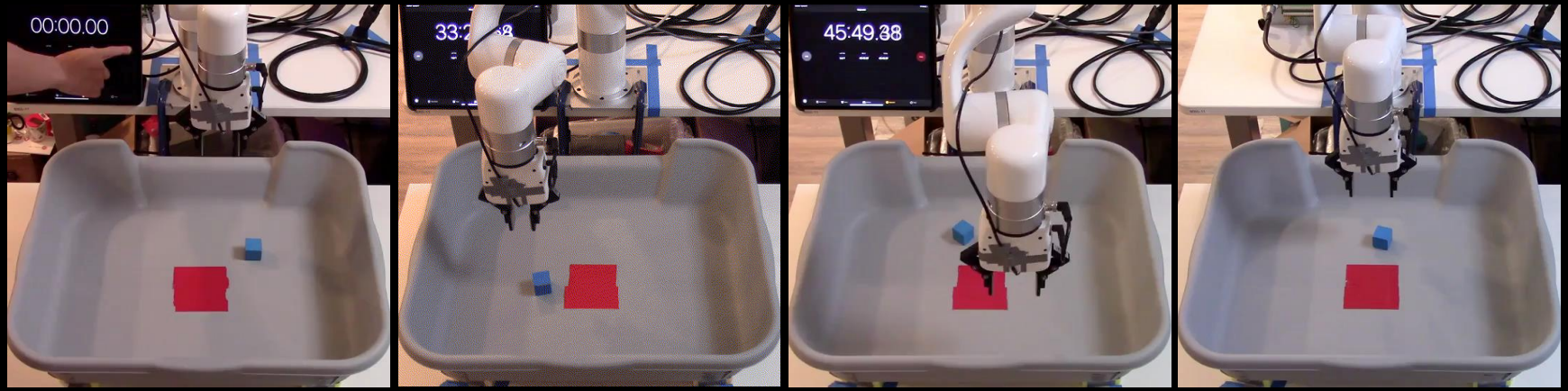
Takes ~1 min

(iii) Continue training with data-augmented RL

Takes ~30 mins

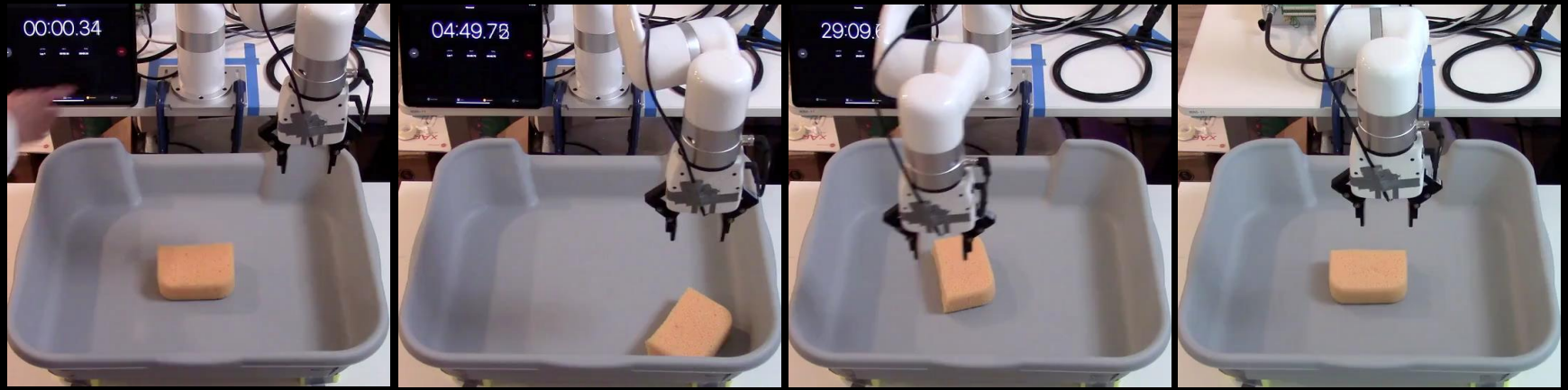
Task: Move

Demonstrations	First Success	Optimal Policy	Evaluation
10 ep ≈ 10:00 min	40 ep ≈ 33:00 min	80 ep ≈ 46:00 min	26/30 Success



Task: Pull

Demonstrations	First Success	Optimal Policy	Evaluation
10 ep ≈ 10:00 min	5 ep ≈ 5:12 min	45 ep ≈ 29:10 min	28/30 Success



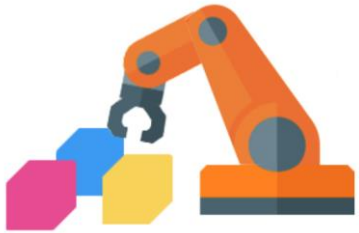
Outline

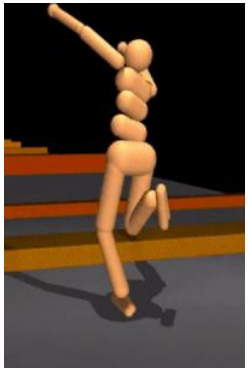
- Beyond pattern recognition: reinforcement learning
- Addressing sample complexity
- ***Addressing reward design***
- Where are we with real-world applications

Challenge: Designing Suitable Reward

Decision (actions)

Consequences (observations, **rewards**)





$$\hat{x}_t - 0.1||a_t||_2^2 - 3.0 \times (z_t - 1.3)^2$$

Challenge: Designing Suitable Reward

Decision (actions)

Consequences (observations, **rewards**)

$$\hat{x}_t - 0.1||a_t||_2^2 - 3.0 \times (z_t - 1.3)^2$$

Hard tasks to define a reward (e.g. cooking)

Challenge: Designing Suitable Reward

Decision (actions)

Consequences (observations, **rewards**)

$$\hat{x}_t - 0.1||a_t||_2^2 - 3.0 \times (z_t - 1.3)^2$$

Hard tasks to define a reward (e.g. cooking)

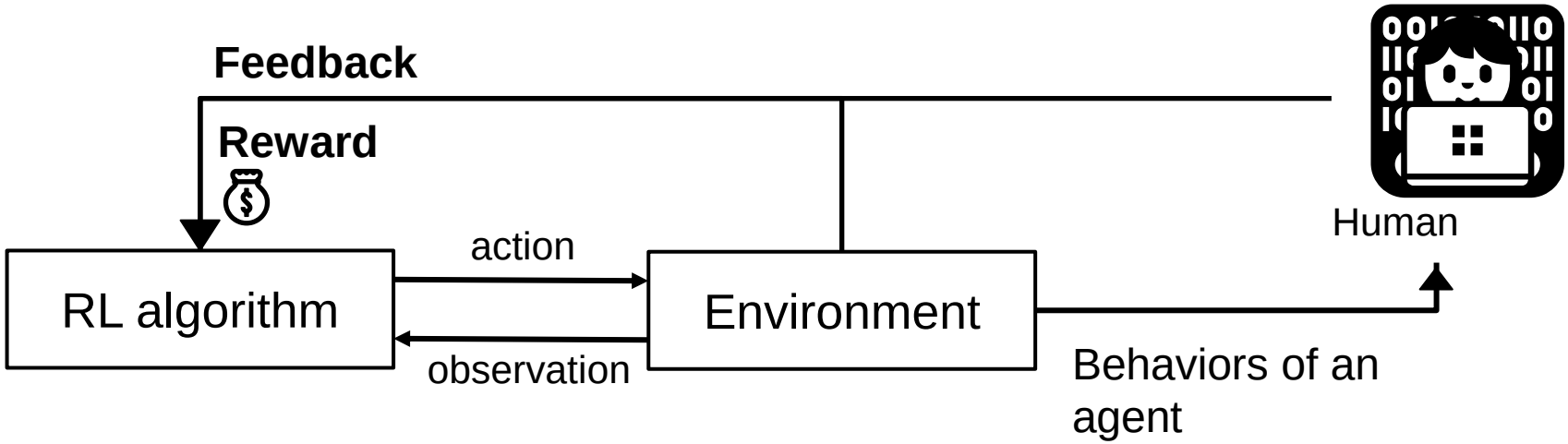
Reward exploitation

https://openai.com/blog/faulty-reward-functions/

What is an Alternative Solution?

What is an Alternative Solution?

- Putting (non-expert) humans into the agent learning loop!

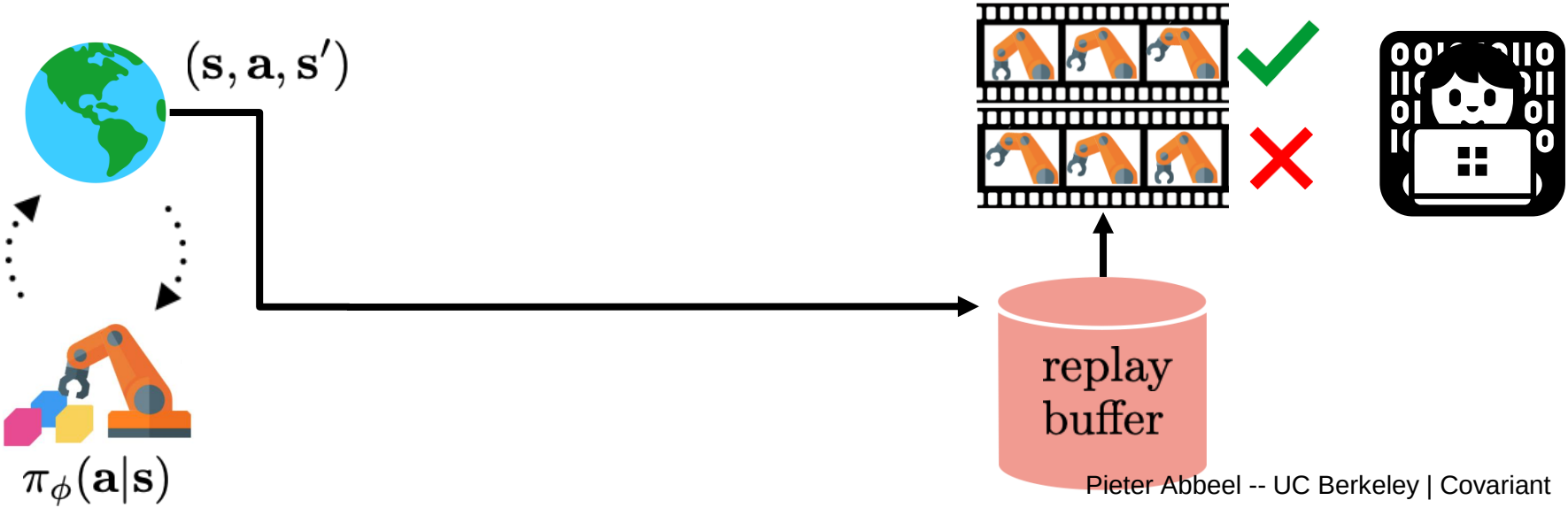


Overall Framework

- Step 1. Collect samples via interactions with environment

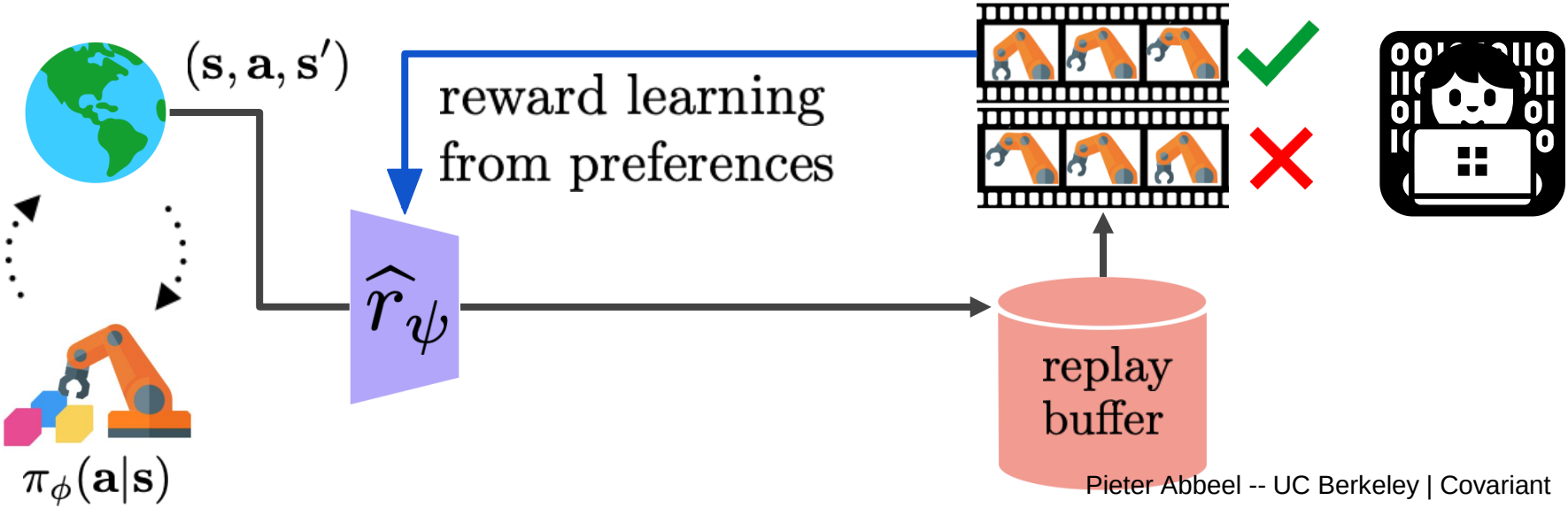
Overall Framework

- Step 1. Collect samples via interactions with environment
- Step 2. Collect human preferences



Overall Framework

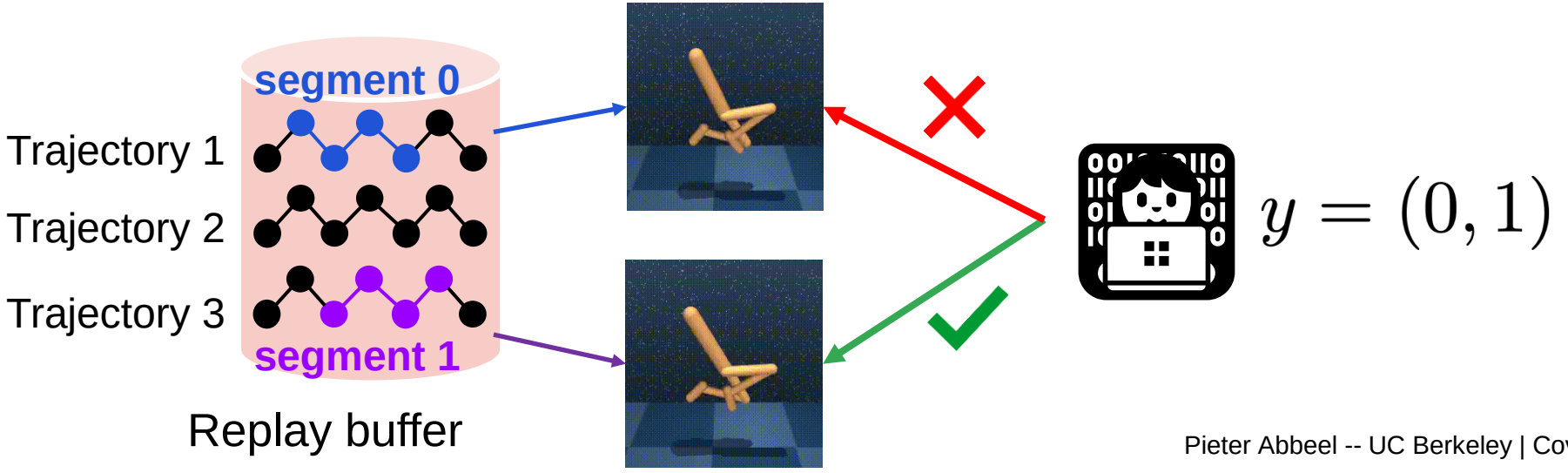
- Step 1. Collect samples via interactions with environment
- Step 2. Collect human preferences
- Step 3. Optimize a reward model using cross entropy loss



Learning Reward from Preferences

- Setup
  - Select two segments (σ^0, σ^1) , i.e. behaviors, from buffer
  - Human provides a preference as a binary label:

$$y \in \{(1,0),(0,1)\}$$



Learning Reward from Preferences

- Fitting a reward model [1]
  - Main idea: formulate this problem as a binary classification!

[1] Christiano, P., Leike, J., Brown, T.B., Martic, M., Legg, S. and Amodei, D., Deep reinforcement learning from human preferences. NeurIPS, 2017.
[2] Bradley, R.A. and Terry, M.E., Rank analysis of incomplete block designs: I. The method of paired comparisons. Biometrika, 39(3/4), pp.324-345, 1952.

Learning Reward from Preferences

- Fitting a reward model [1]
  - Main idea: formulate this problem as a binary classification!
  - By following the Bradley-Terry model [2], we can model a **preference predictor** as follows:

$$P_\psi\left[\sigma^1 \succ \sigma^0\right] = \frac{\exp \sum_t \widehat{r}(\mathbf{s}_t^1, \mathbf{a}_t^1)}{\sum_{i\in\{0,1\}} \exp \sum_t \widehat{r}(\mathbf{s}_t^i, \mathbf{a}_t^i)}$$

[1] Christiano, P., Leike, J., Brown, T.B., Martic, M., Legg, S. and Amodei, D., Deep reinforcement learning from human preferences. NeurIPS, 2017.
[2] Bradley, R.A. and Terry, M.E., Rank analysis of incomplete block designs: I. The method of paired comparisons. Biometrika, 39(3/4), pp.324-345, 1952.

Learning Reward from Preferences

- Fitting a reward model [1]
  - Main idea: formulate this problem as a binary classification!
  - By following the Bradley-Terry model [2], we can model a **preference predictor** as follows:

$$P_\psi\left[\sigma^1 \succ \sigma^0\right] = \frac{\exp \sum_t \hat{r}(\mathbf{s}_t^1, \mathbf{a}_t^1)}{\sum_{i\in\{0,1\}} \exp \sum_t \hat{r}(\mathbf{s}_t^i, \mathbf{a}_t^i)}$$

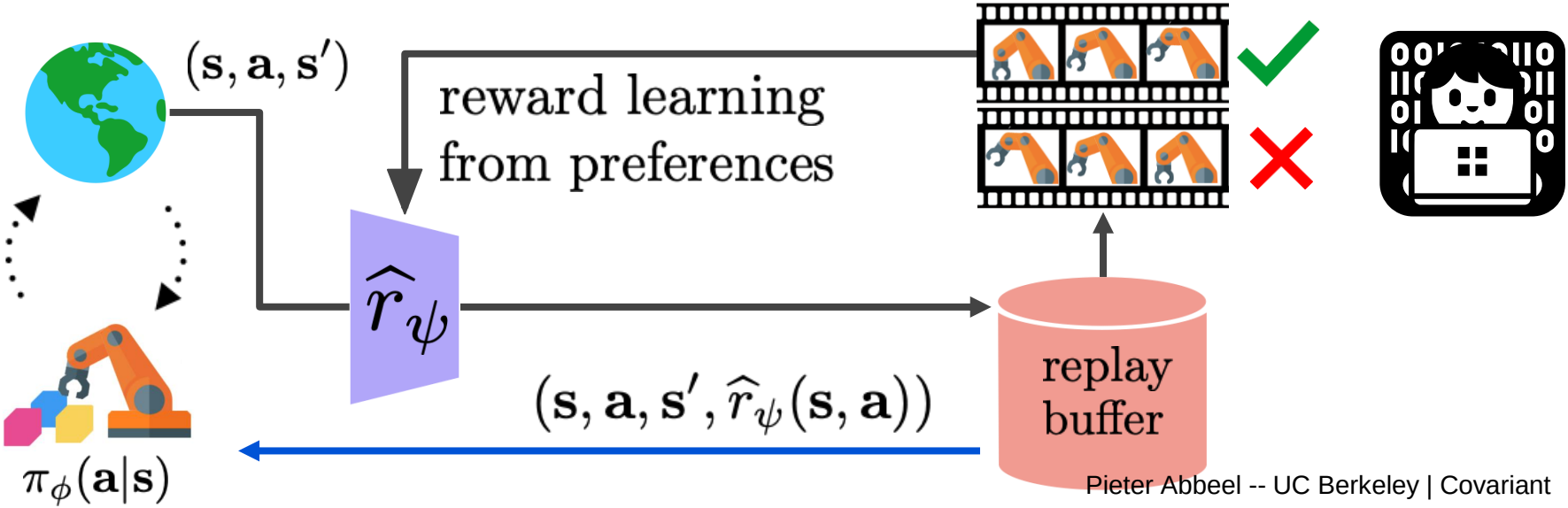
Sum of rewards over segment 1

Event that segment 1 is preferable to segment 0

[1] Christiano, P., Leike, J., Brown, T.B., Martic, M., Legg, S. and Amodei, D., Deep reinforcement learning from human preferences. NeurIPS, 2017.
[2] Bradley, R.A. and Terry, M.E., Rank analysis of incomplete block designs: I. The method of paired comparisons. Biometrika, 39(3/4), pp.324-345, 1952.

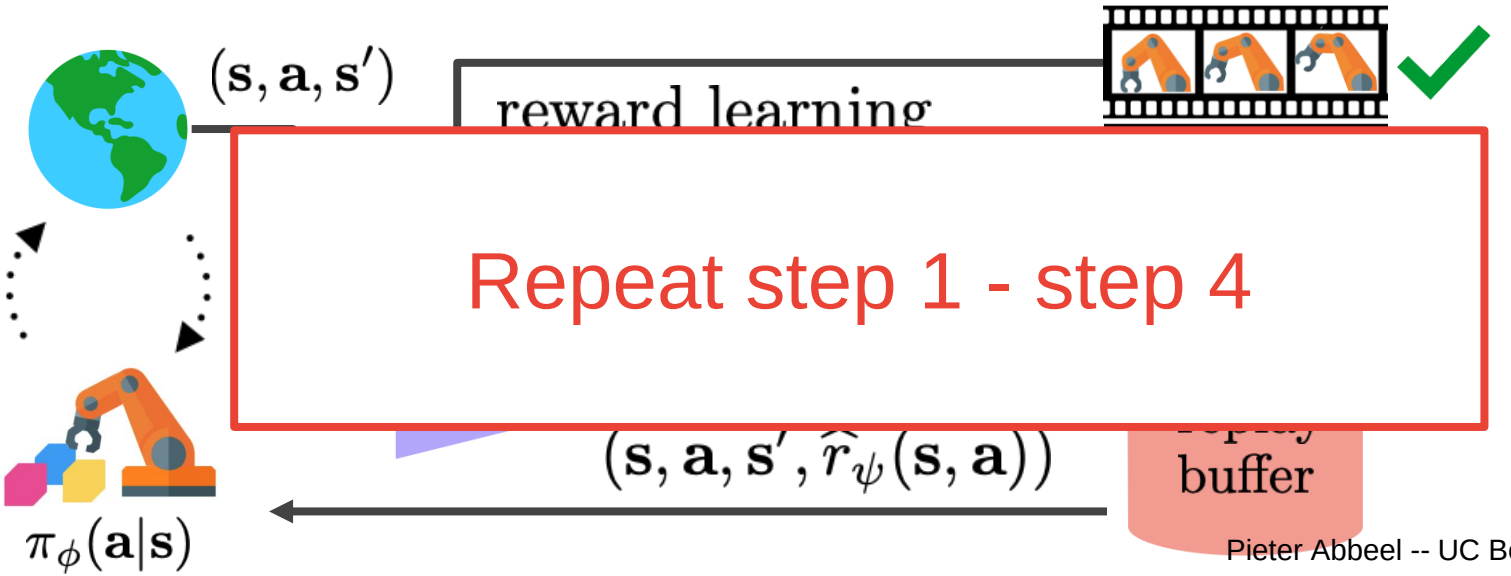
Overall Framework

- Step 1. Collect samples via interactions with environment
- Step 2. Collect human preferences
- Step 3. Optimize a reward model using cross entropy loss
- Step 4. Optimize a policy using off-policy algorithms



Overall Framework

- Step 1. Collect samples via interactions with environment
- Step 2. Collect human preferences
- Step 3. Optimize a reward model using cross entropy loss
- Step 4. Optimize a policy using off-policy algorithms



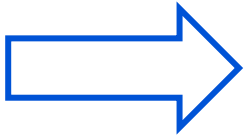
Unsupervised Pre-training: APT

- Obtaining a good initial state space coverage is important!
  - Human can't convey much meaningful information to the agent

Unsupervised Pre-training: APT

- Obtaining a good initial state space coverage is important!
  - Human can't convey much meaningful information to the agent

Behavior from random exploration policy

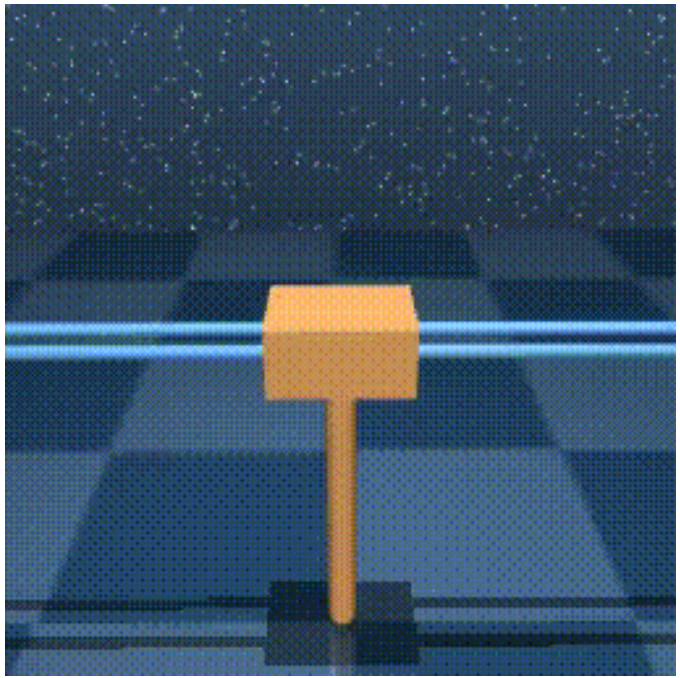


Behavior from pre-trained policy

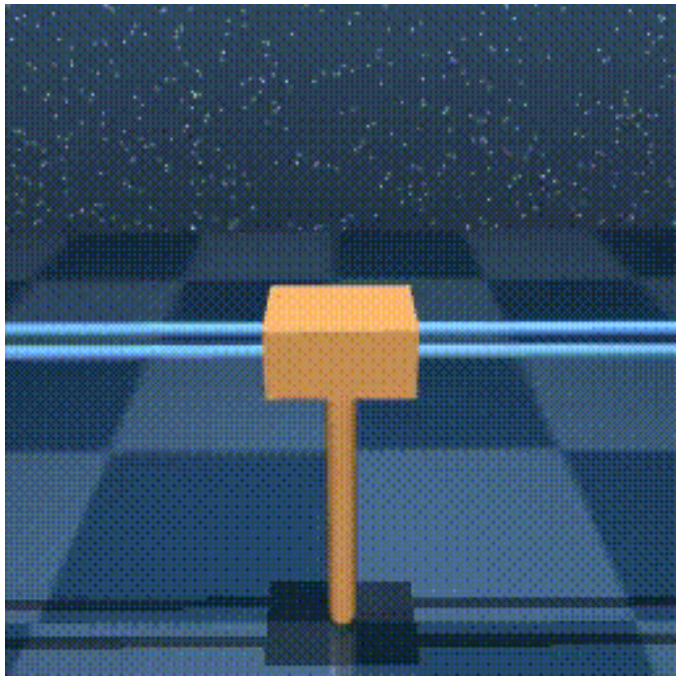
Can We Teach Novel Behaviors?

Can We Teach Novel Behaviors?

- 40 queries in less than 5 mins



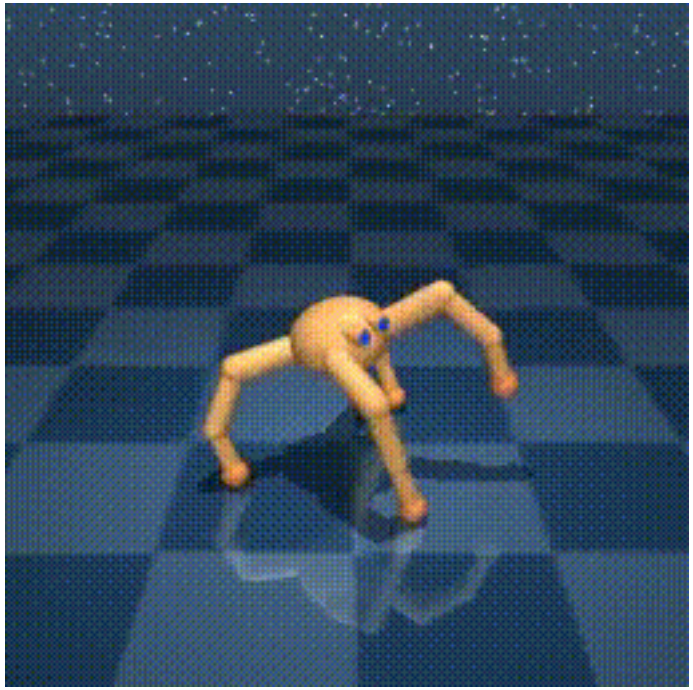
Counter clockwise



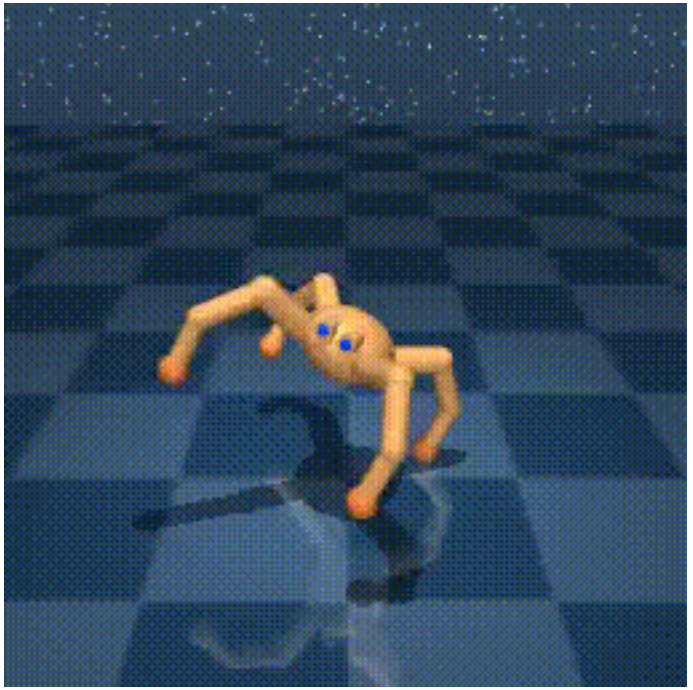
Clockwise

Can We Teach Novel Behaviors?

- 200 queries in less than 30 mins



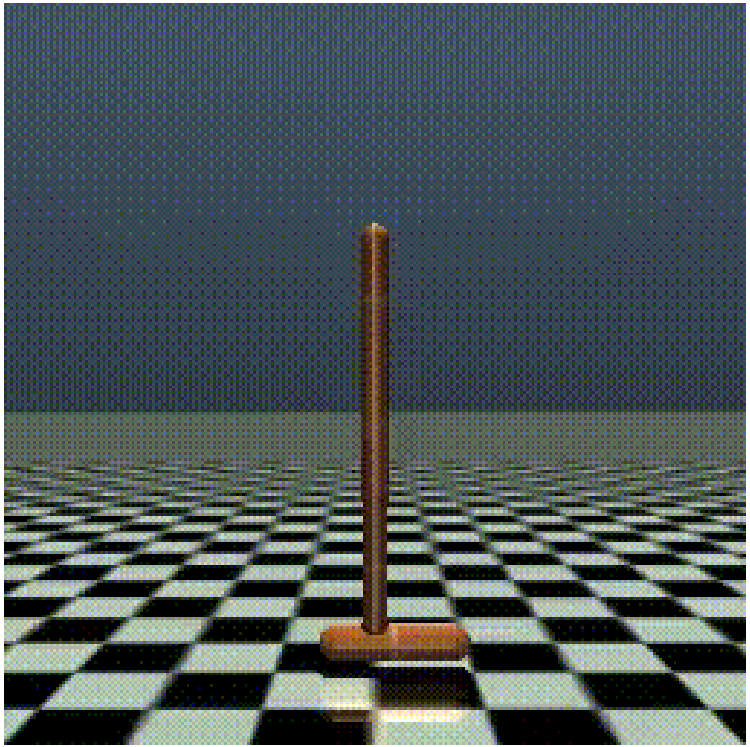
Waving left front leg



Waving right front leg

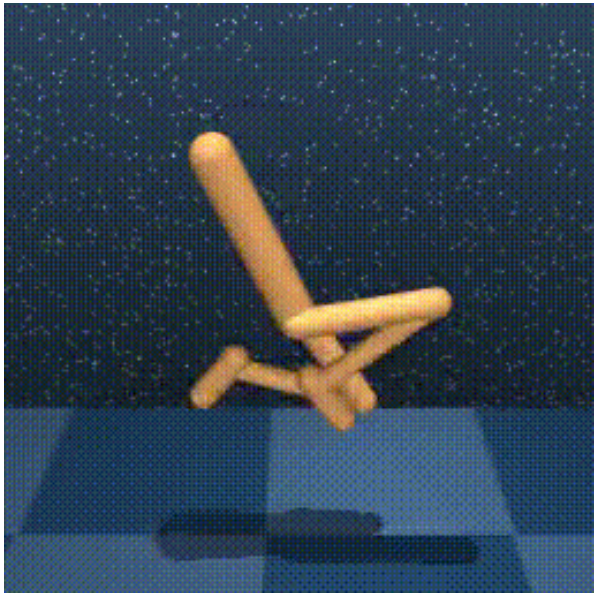
Can We Teach Novel Behaviors?

- 400 queries in less than one hour



Can We Avoid Reward Exploitation?

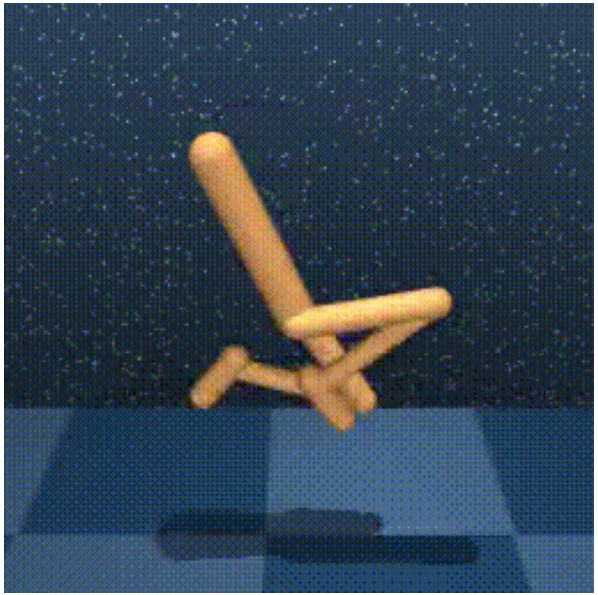
Can We Avoid Reward Exploitation?



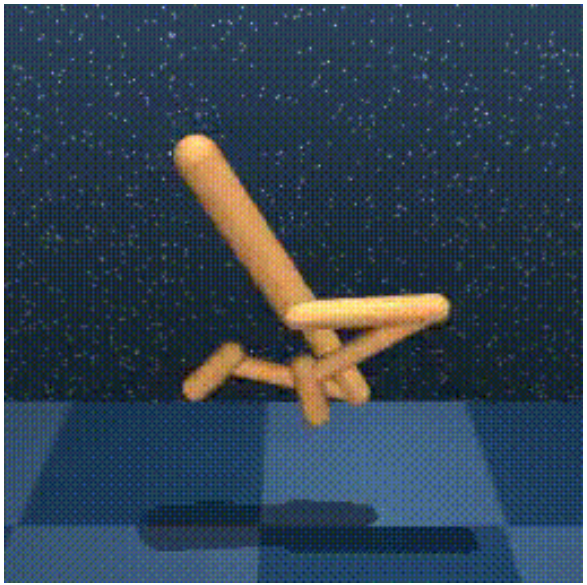
SAC with task reward on walker, walk
(use one leg even if score ~=1000)

Can We Avoid Reward Exploitation?

- 150 queries in less than 20 mins



SAC with task reward on walker, walk
(use one leg even if score ~=1000)



SAC trained with human feedback
(use both legs)

Outline

- Beyond pattern recognition: reinforcement learning
- Addressing sample complexity
- Addressing reward design
- ***Where are we with real-world applications***

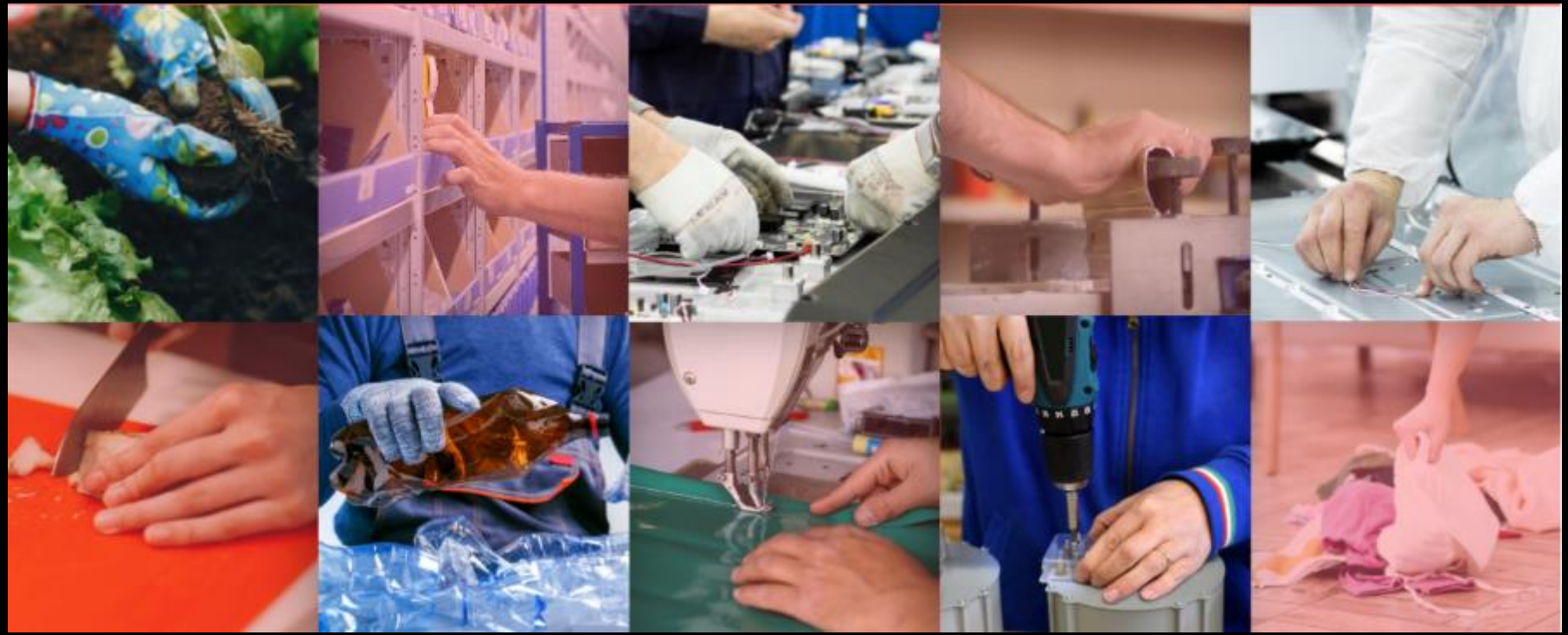
Research <<<<GAP>>>> Real-World

- **Research:**
  - 0 to 1
  - If achieving 70% on a benchmark, move on to the next one
- **Real-World:**
  - Even 90% is not enough!
  - Exacerbated by long tail of scenarios


WAYMO

Smaller Vehicles -- Deliveries

Trucks – Highway Focus



Why Autonomy is Key

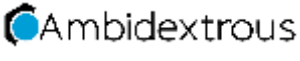
Success:	90%	99%	99.9%
Time between human intervention At 500PPH	1 min	12 min	120 min
Human: Robot ratio	1 : 1	1 : 6	1 : 14
Human Oversight Cost	**$150K** per robot per year	**$30K** per robot per year	**$10K** per robot per year
Opex savings	✗	✓	✓

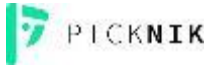
Many start-ups in robotic manipulation

- Recycling

- Farming

- Warehousing





Covariant

Press January 29, 2020

A Warehouse Robot Learns to Sort Out the Tricky Stuff

WIRED

Press January 29, 2020

AI Helps Warehouse Robots Pick Up New Tricks

Press May 6, 2020

Logistics AI Startup Covariant Reaps $40 Million in Funding Round

AI Robotics becoming real in warehouses

Knapp Pick-It-Easy Powered by Covariant Brain

Summary

- Beyond pattern recognition: reinforcement learning
- Addressing sample complexity
- Addressing reward design
- Where are we with real-world applications

Thank you!

pabbeel@cs.berkeley.edu

Want to *learn more Deep RL*?

- Deep RL Bootcamp [intense long weekend]
  - 12 lectures by: Pieter Abbeel, Rocky Duan, Peter Chen, John Schulman, Vlad Mnih, Chelsea Finn, Sergey Levine
  - 4 hands-on labs
  - https://sites.google.com/view/deep-rl-bootcamp/
- Deep RL Course [full semester]
  - Originated by John Schulman, Sergey Levine, Chelsea Finn
  - Latest offerings (by Sergey Levine): http://rll.berkeley.edu/deeprlcourse/