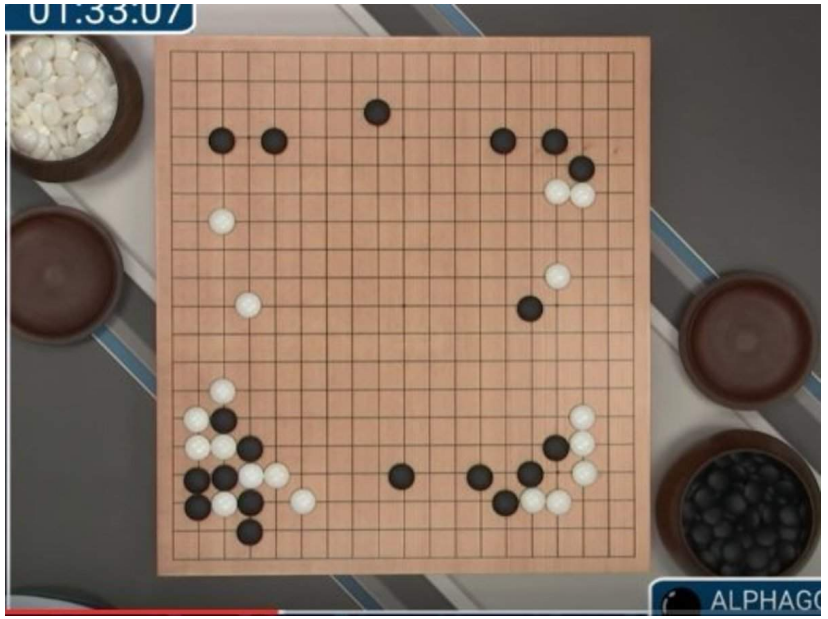
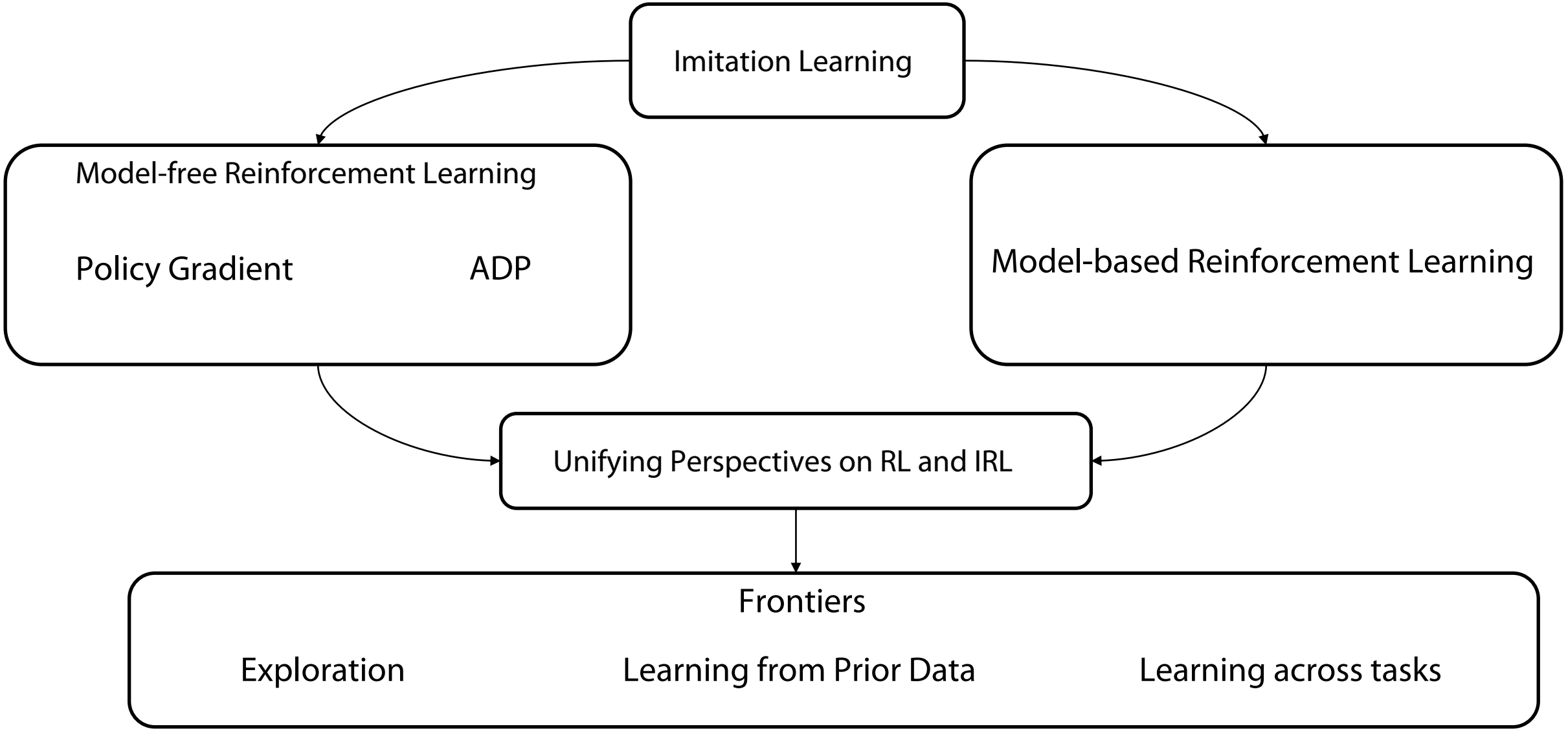
Deep Reinforcement Learning
Spring 2026

Abhishek Gupta

TA: Mateo Guaman Castro



Class Structure



Lecture outline

Recap: Imitation Learning via Behavior Cloning

Challenges with DAgger

Modern Takes on Imitation Learning

Policy Gradient

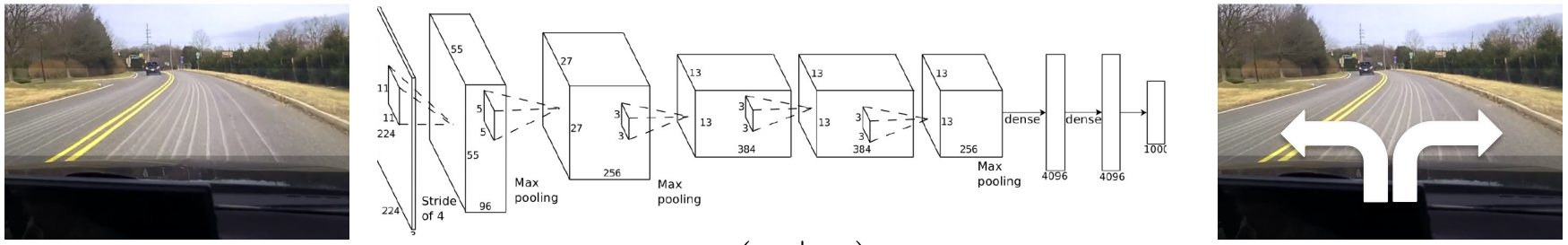
Idea 1: Imitation Learning via Supervised Learning

Given: Demonstrations of optimal behavior

Goal: Train a policy to mimic the demonstrator

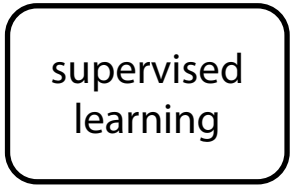
$$\arg\max_{\theta} \mathbb{E}_{(s^*,a^*)\sim\mathcal{D}} \left[\log \pi_\theta(a^*|s^*)\right]$$

Idea: Treat imitation learning as a supervised learning problem! → Behavior Cloning



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

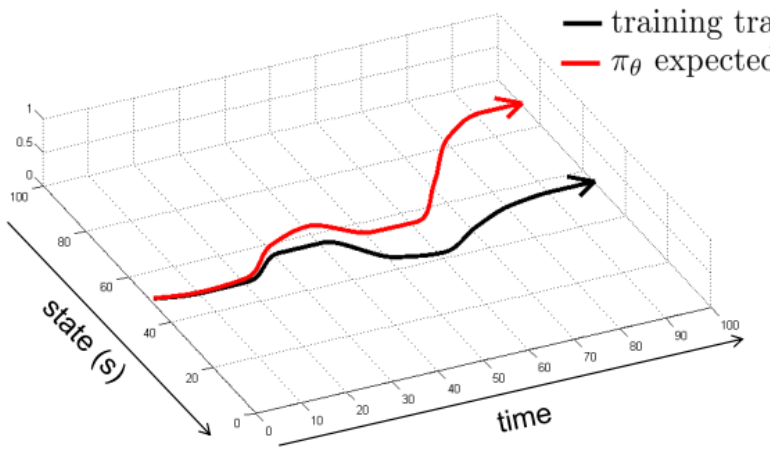
$\mathbf{o}_t$
$\mathbf{a}_t$



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

So does behavior cloning really work?

- Imitation Learning ≠ Supervised Learning



Compounding error!

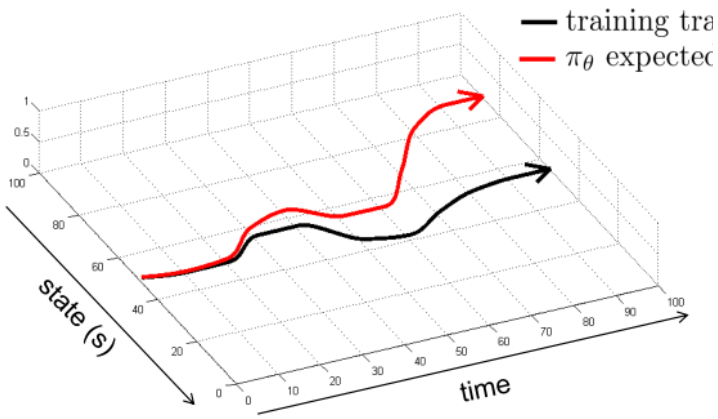
$$\arg\max_\theta \mathbb{E}_{(s^*,a^*)\sim\mathcal{D}}\left[\log \pi_\theta(a^*|s^*)\right] \qquad \mathbb{E}_{(s,a)\sim\rho(\pi)}\left[1(a=a^*)\right]$$

Not the same!

Let's try and understand where the problem lies?

Behavior cloning has challenges in both theory and practice

$$\sum_t \mathbb{E}_{(s_t,a_t)\sim p_{\pi_\theta}(s_t,a_t)}\left[c(s_t,a_t)\right] \leq O(\epsilon H^2)$$



Underfitting

$$\pi_\theta(a \neq \pi^*(s_t)|s_t) \leq \epsilon$$

Compounding error

$$\leq O(\epsilon H^2)$$

Effects of choice of f-divergence on behavior

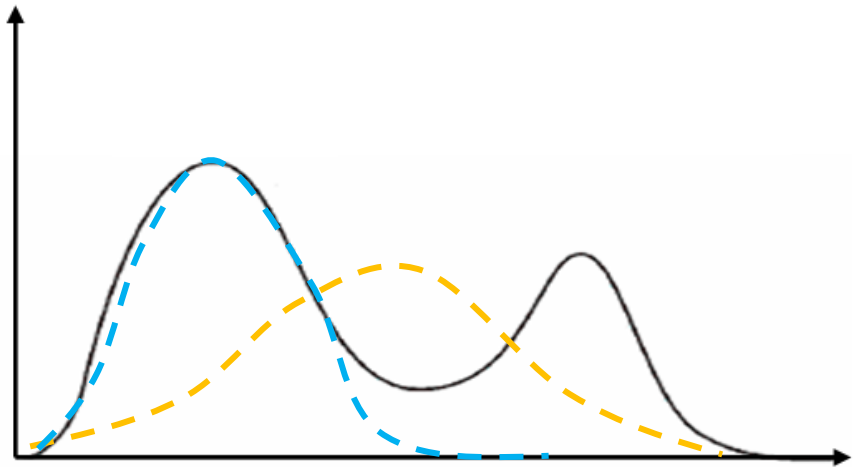
Different divergences lead to different properties

$$\mathbb{E}_{s^* \sim p_{\pi_e}(.)} \left[D_{\mathrm{KL}}(\pi_e(.|s^*) || \pi_\theta(.|s^*)) \right] \longrightarrow \mathbb{E}_{s^* \sim p_{\pi_e}(.)} \left[D_f(\pi_e(.|s^*), \pi_\theta(.|s^*)) \right]$$

Forward KL (behavior cloning)

More general class of divergences

$$D_f(p(x), q(x)) = \mathbb{E}_{q(x)} \left[f\left(\frac{p(x)}{q(x)} \right) \right]$$



Forward KL (mode covering) $f(x) = x \log(x)$

Reverse KL (mode seeking) $f(x) = -\log(x)$

So how do we fix BC?

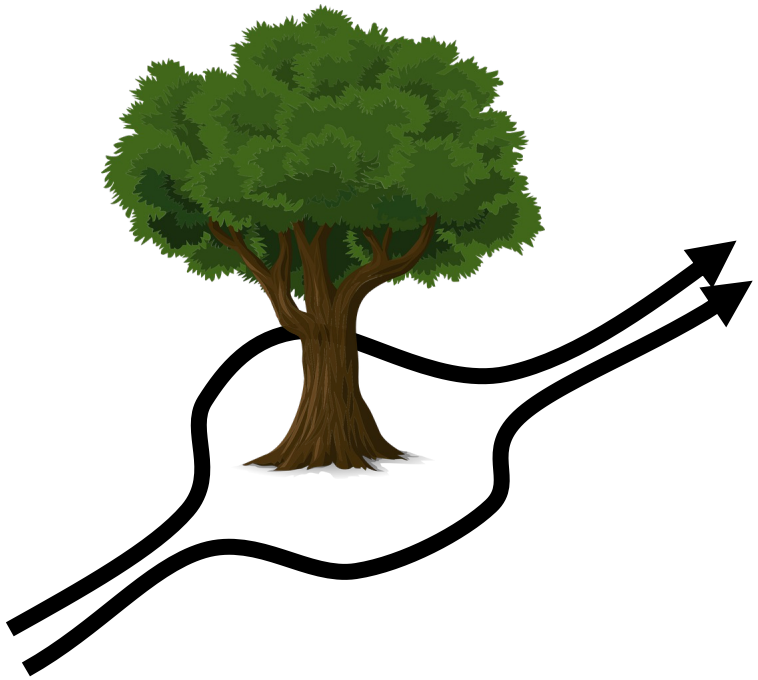
Use a different f-divergence! (Change f)

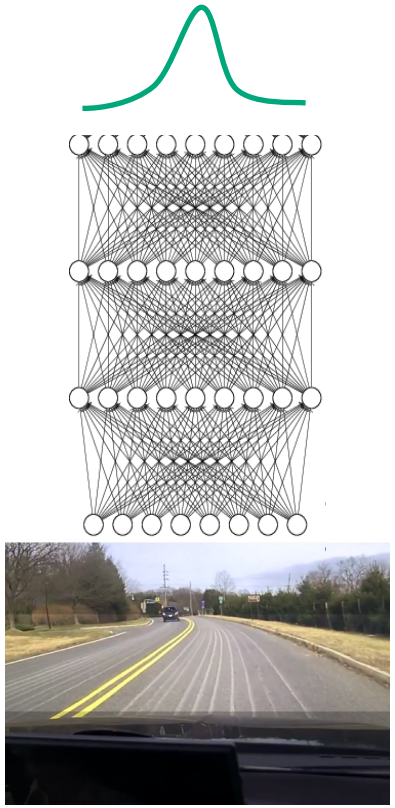
or

Use a richer distribution class! (Change π_θ)

Using Richer Policy Distribution Classes

Multimodal behavior → use more **expressive** probability distributions, no mode averaging issues



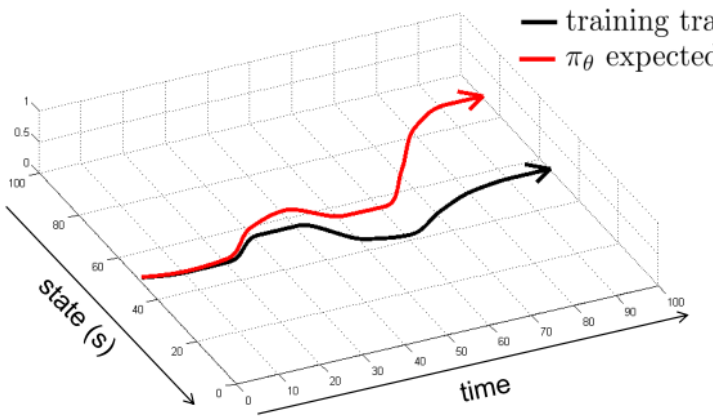


1. Output mixture of Gaussians
2. Latent variable models
3. Autoregressive discretization
4. Diffusion models
5. …

Let's try and understand where the problem lies?

Behavior cloning has challenges in both theory and practice

$$\sum_t \mathbb{E}_{(s_t,a_t)\sim p_{\pi_\theta}(s_t,a_t)}\left[c(s_t,a_t)\right] \leq O(\epsilon H^2)$$



Underfitting

$$\pi_\theta(a \neq \pi^*(s_t)|s_t) \leq \epsilon$$

Compounding error

$$\leq O(\epsilon H^2)$$

Concrete Instantation: DAgger

can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?

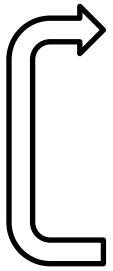
idea: instead of being clever about $p_{\pi_\theta}(\mathbf{o}_t)$, be clever about $p_{\text{data}}(\mathbf{o}_t)$!

DAgger: Dataset Aggregation

goal: collect training data from $p_{\pi_\theta}(\mathbf{o}_t)$ instead of $p_{\text{data}}(\mathbf{o}_t)$

how? just run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

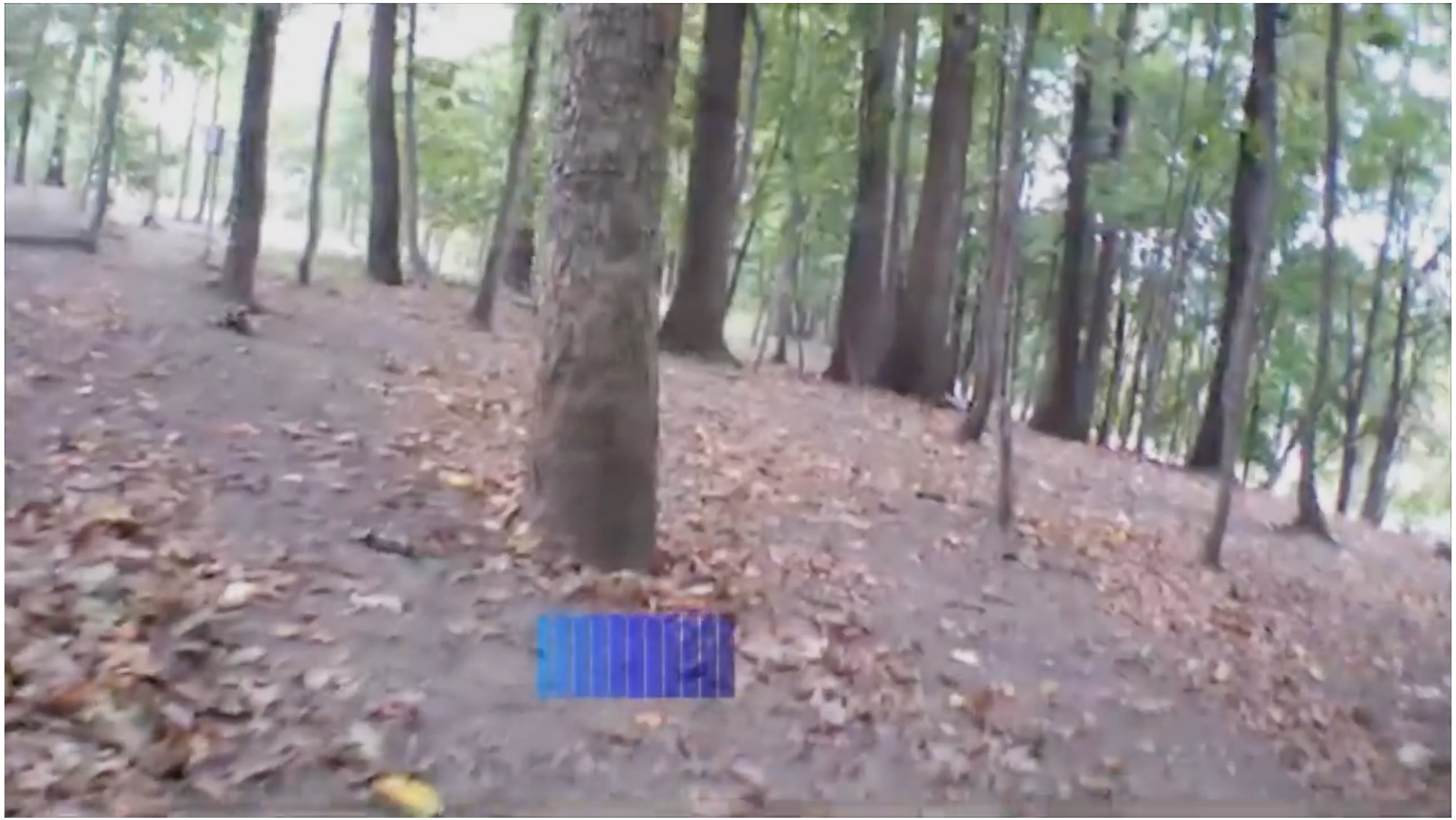
but need labels $\mathbf{a}_t$!



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

Ross et al. '11

DAgger Example



Ross et al. '13

Lecture outline

Recap: Imitation Learning via Behavior Cloning

↓

Challenges with DAgger

↓

Modern Takes on Imitation Learning

↓

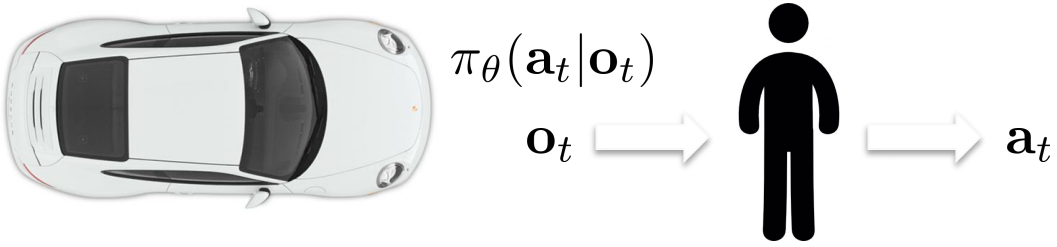
Policy Gradient

What's the problem?

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

Requires human effort

Requires running policies and labeling post-hoc



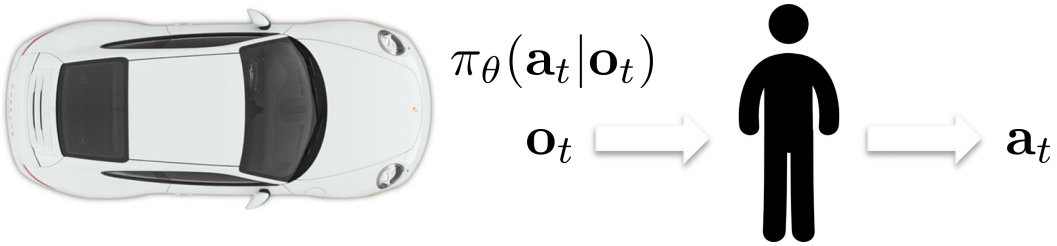
Ross et al. '11

How might we fix this?

Can we reduce human cost?

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

Can we avoid labeling post-hoc?

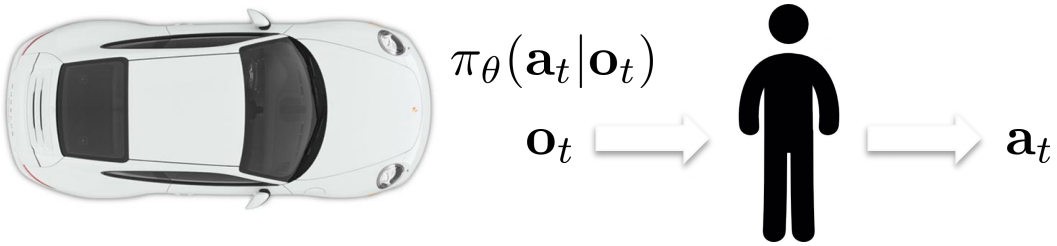


Ross et al. '11

How might we fix this?

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

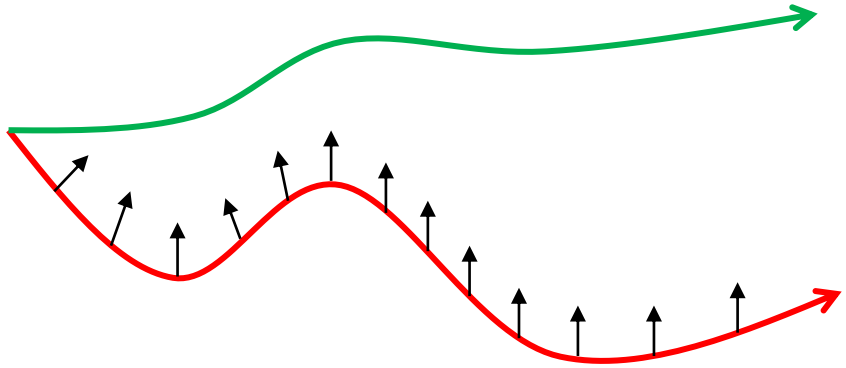
Can we avoid labeling post-hoc?



Ross et al. '11

DAgger is a bit impractical –> HG-DAgger

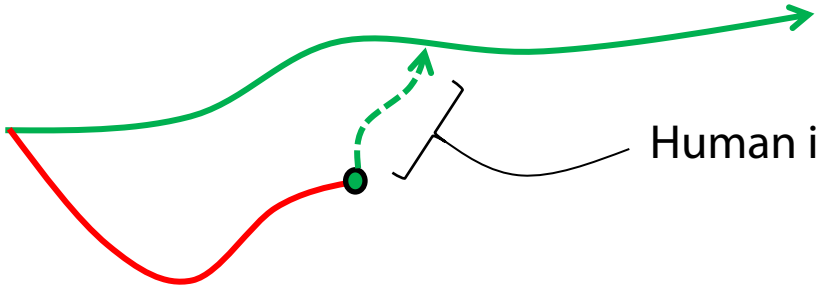
Dagger requires revisiting states and labeling post-hoc



$$s_t \sim d^{\pi_\theta}$$
$$a \sim \pi_h(a|s)$$

Can do "blended" control, but becomes hard to get precision

What if we allowed people to take control for longer?



$s_t \sim d^{\pi_\theta}$ (point of intervention)

$s_{t+1:t+k} \sim d^{\pi_h}|s_t$ (human takeover)

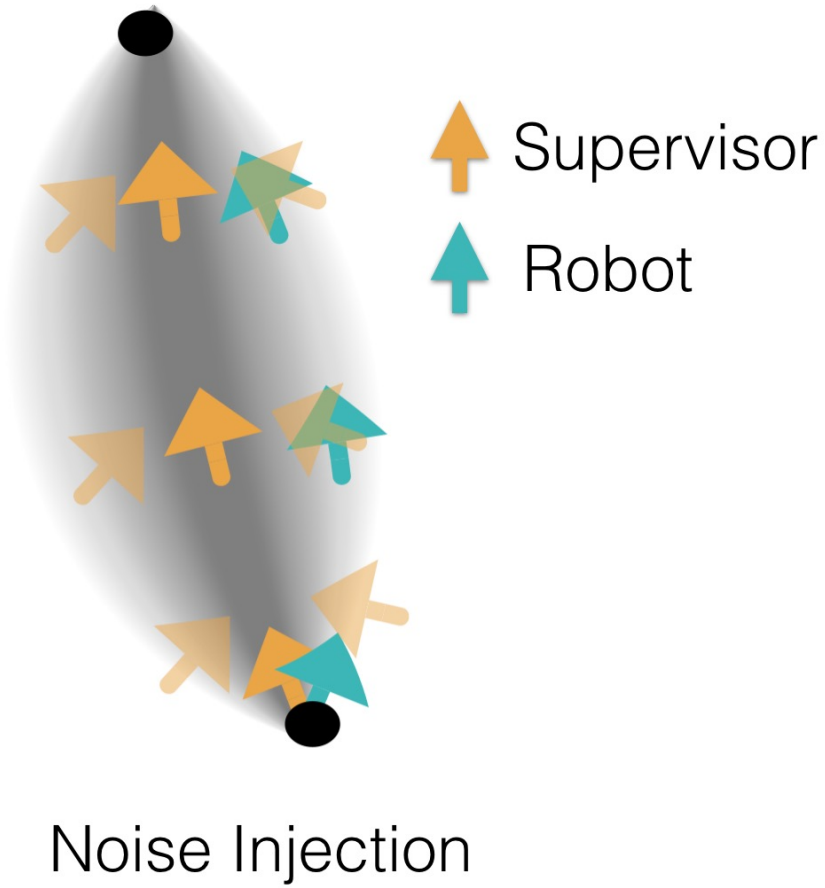
$a \sim \pi_h(a|s)$ (human actions)

Can we do this pre-emptively?

Kelly et al, HG-Dagger, 2018

Noising the Data Collection Process

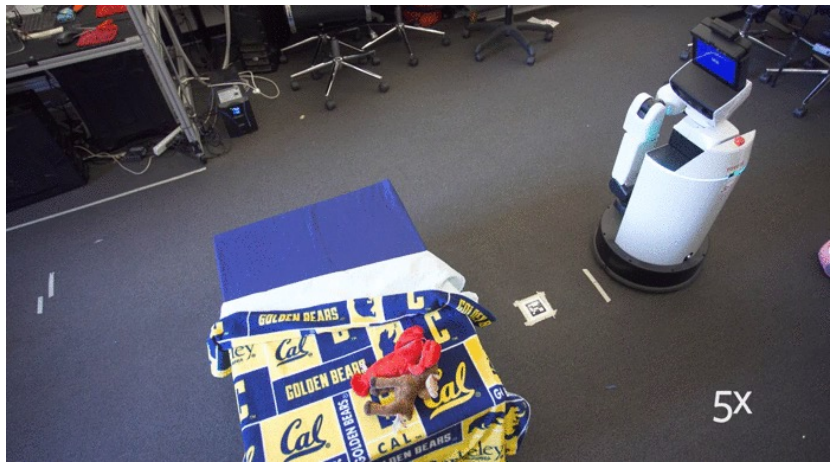
Key idea: force the human to correct for noise **during** training



Under noise during data collection

Maximize likelihood

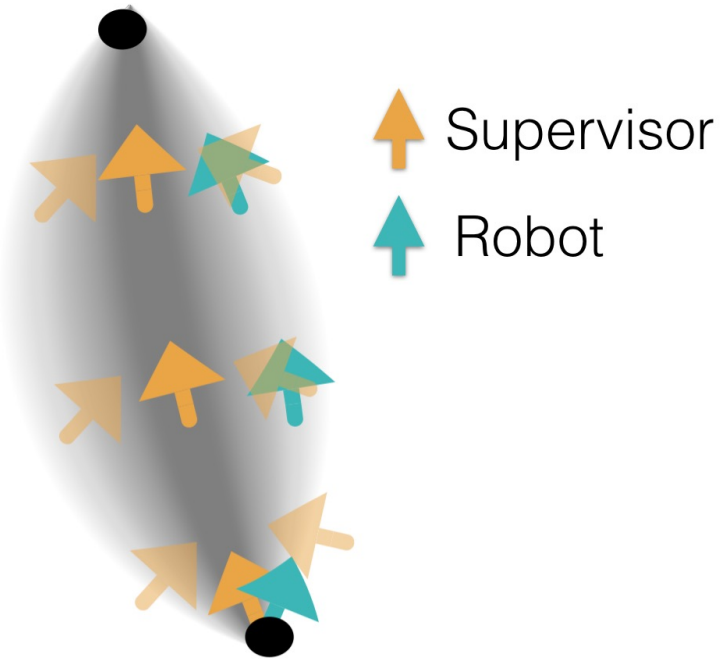
$$\hat{\psi}_{k+1} = \underset{\psi}{\operatorname{argmin}} E_{p(\xi|\pi_{\theta^*},\psi_k)} - \sum_{t=0}^{T-1} \log\left[\pi_{\theta^*}(\pi_{\hat{\theta}}(\mathbf{x_t})|\mathbf{x_t},\psi)\right]$$



DART: Noise Injection for Robust Imitation Learning, Laskey et al CoRL ‘17

Why might this not be enough?

Key idea: force the human to correct for noise **during** training



Noise Injection

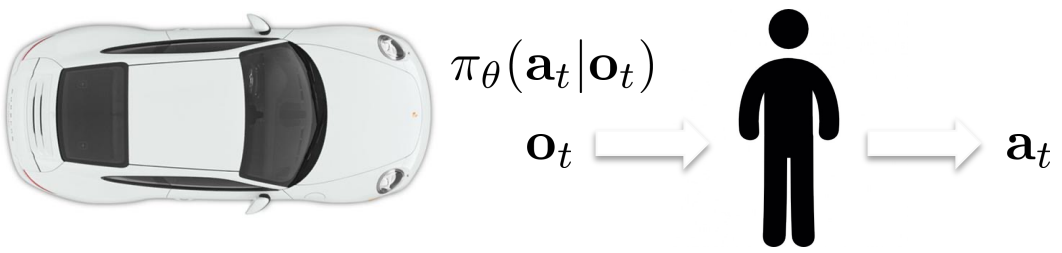
Assumes that the expert **can** actually perform behaviors under noise
→ Not always possible!

DART: Noise Injection for Robust Imitation Learning, Laskey et al CoRL ‘17

How might we fix this?

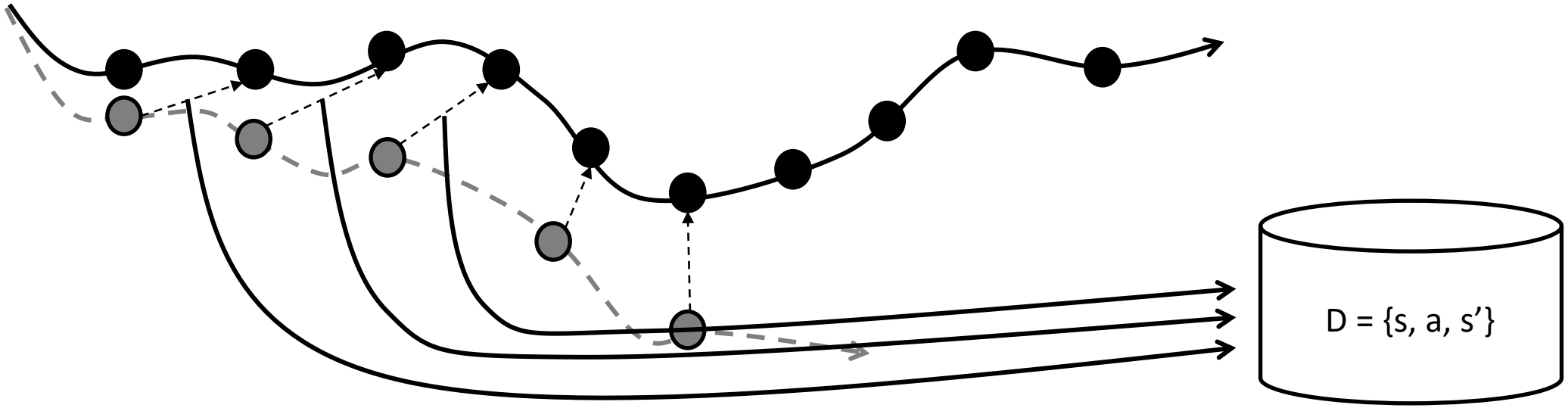
Can we reduce human cost?

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



Ross et al. '11

Can we avoid expensive online data collection/labeling?



Generate corrective labels to dataset for imitation

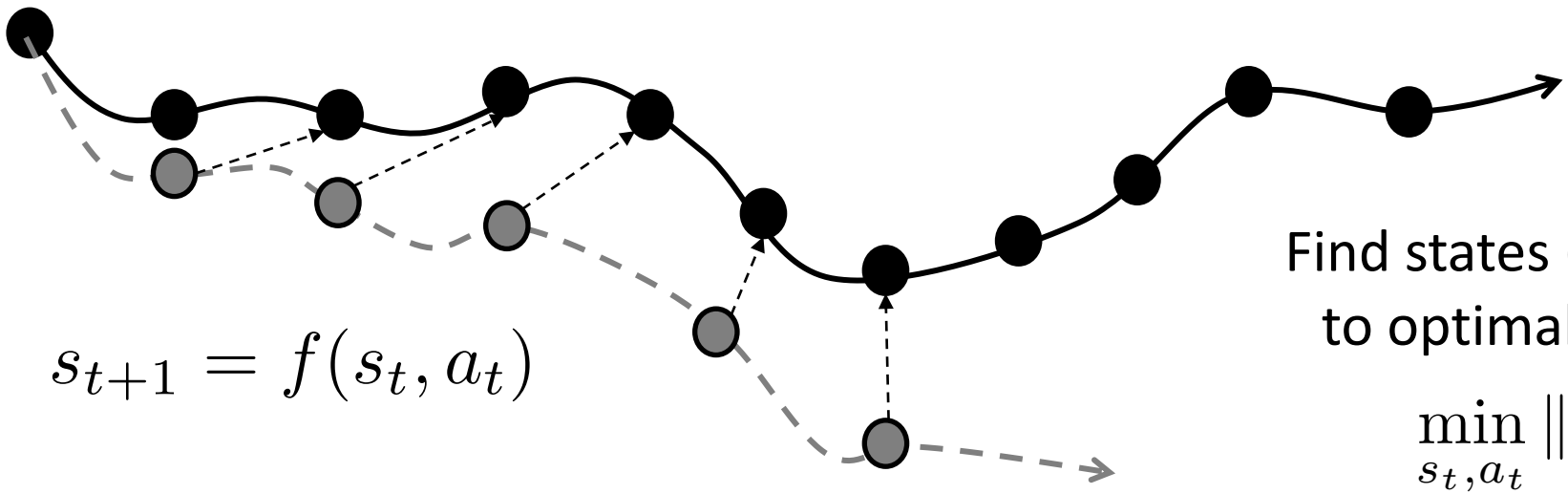
Abhay Deshpande

Yunchu Zhang

Liyiming Ke

How can we find corrective labels without an expensive human in the loop and online data collection?

Generating Corrective Labels From True Dynamics



$$s_{t+1} = f(s_t, a_t)$$

Find states (s_t), actions (a_t) that lead back to optimal states under true dynamics

$$\min_{s_t, a_t} \|s^*_{t+1} - f(s_t, a_t)\| \leq \epsilon$$

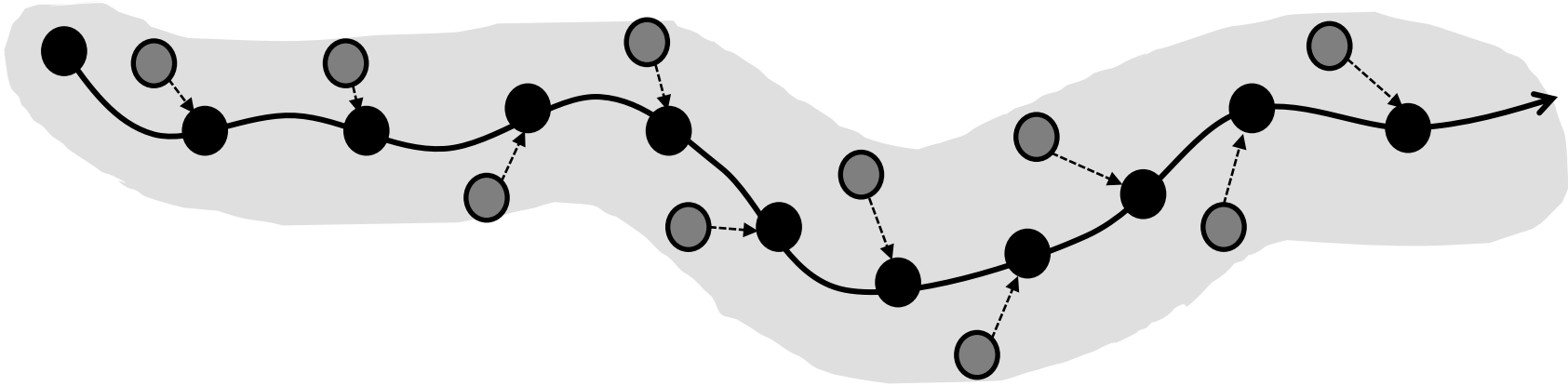
Easy with known dynamics

Intuition: find labels to bring OOD states back in distribution

But models are unknown! ☹

CCIL: Continuity-based Data Augmentation for Corrective Imitation Learning, Ke et al '24
Data Efficient Behavior Cloning for Fine Manipulation via Continuity-based Corrective Labels, Deshpande et al '24

Generating Corrective Labels with **Learned** Dynamics



Ok models are unknown, let's learn them!

$$\min_{\hat{f}} \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}} \left[\|\hat{f}(s_t,a_t) - s_{t+1}\|_2 \right]$$

But learned dynamics $\hat{f}_\phi$ are not globally accurate?

$$\|s^*_{t+1} - \hat{f}_\phi(s_t,a_t)\| \leq \epsilon$$

Under approximately Lipschitz smooth models, trust models around training data

Find states (s_t), actions (a_t) that lead back to optimal states under ~~true~~ learned dynamics, **where learned dynamics can be trusted**

$$\min_{s_t,a_t} \|s^*_{t+1} - \hat{f}_\phi(s_t,a_t)\| \leq \epsilon \longleftarrow \text{Corrective label}$$

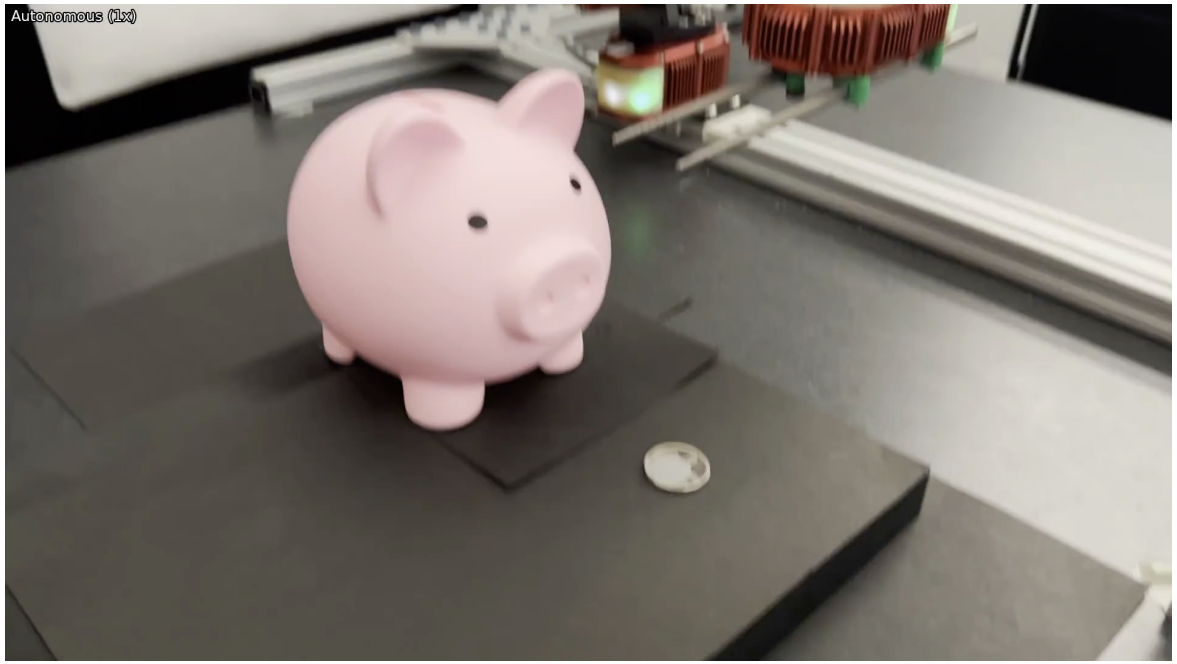
$$\text{s.t} \quad \|s^*_t - s_t\| \leq \epsilon_1, \|a^*_t - a_t\| \leq \epsilon_2 \longleftarrow \text{Close to data}$$

CCIL: Continuity-based Data Augmentation for Corrective Imitation Learning, Ke et al '24
Data Efficient Behavior Cloning for Fine Manipulation via Continuity-based Corrective Labels, Deshpande et al '24

How well does generating corrective labels work?

With corrective labels

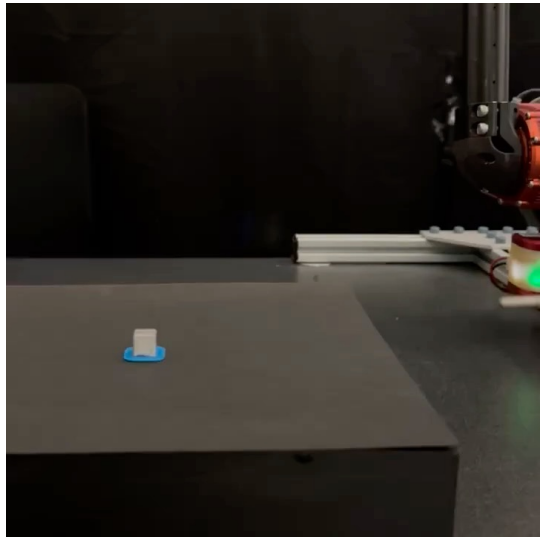
Without corrective labels

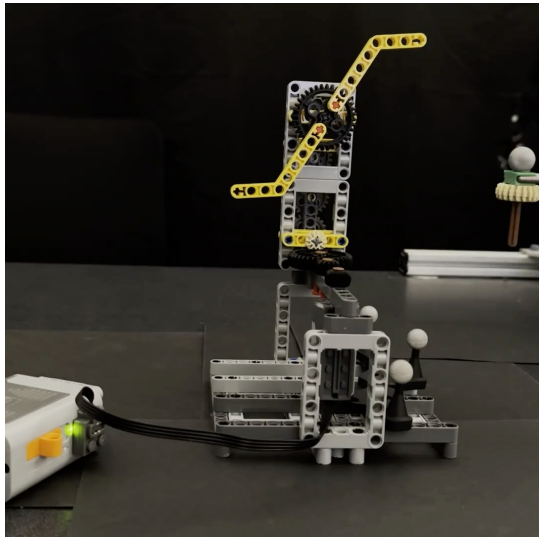


CCIL: Continuity-based Data Augmentation for Corrective Imitation Learning, Ke et al '24
Data Efficient Behavior Cloning for Fine Manipulation via Continuity-based Corrective Labels, Deshpande et al '24

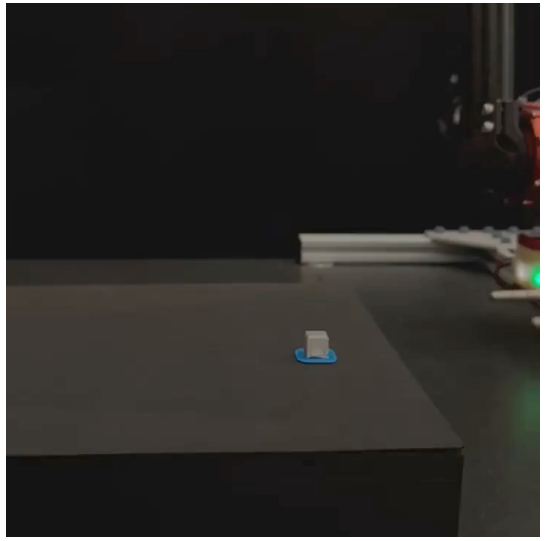
How well does generating corrective labels work?

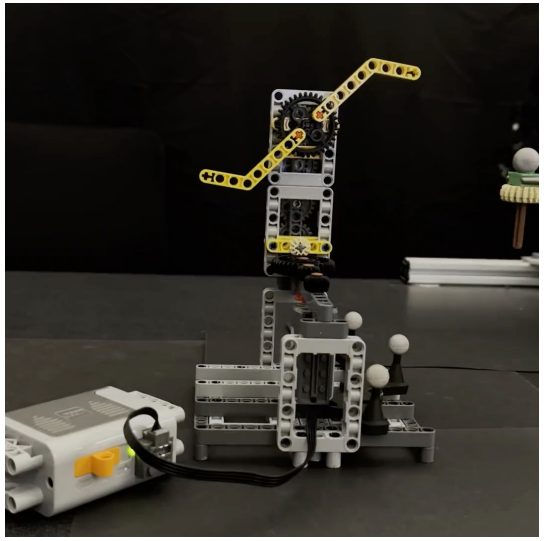
With corrective labels

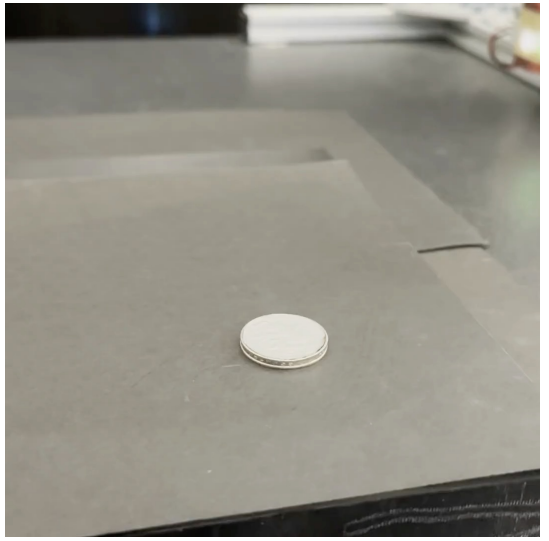




Without corrective labels







Lecture outline

Recap: Imitation Learning via Behavior Cloning

↓

Challenges with DAgger

↓

Modern Takes on Imitation Learning

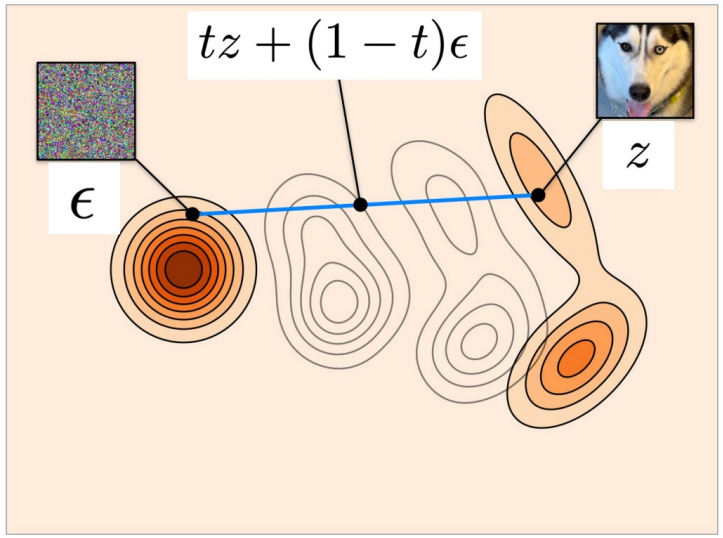
↓

Policy Gradient

But BC works unreasonably well!

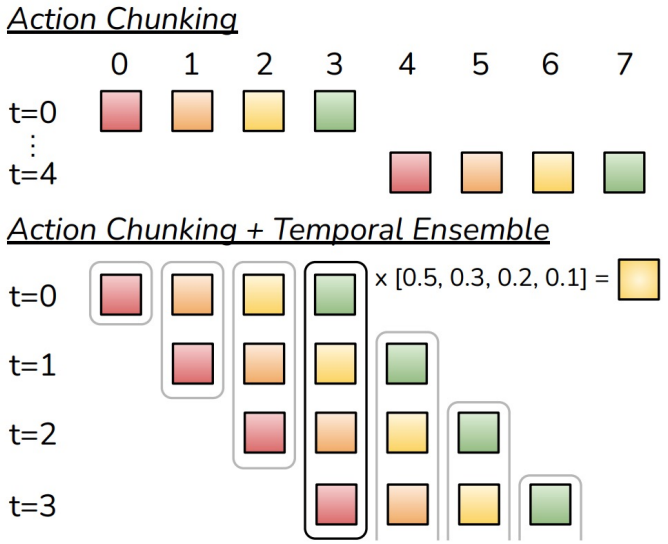
Is this by accident?

Generative Models



Flow matching, diffusion, etc

Action chunking

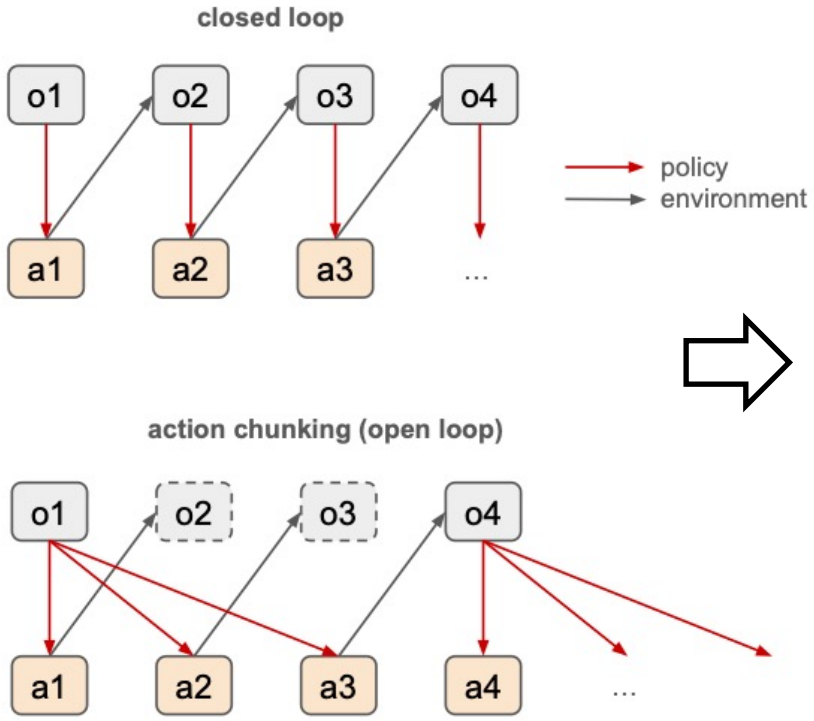


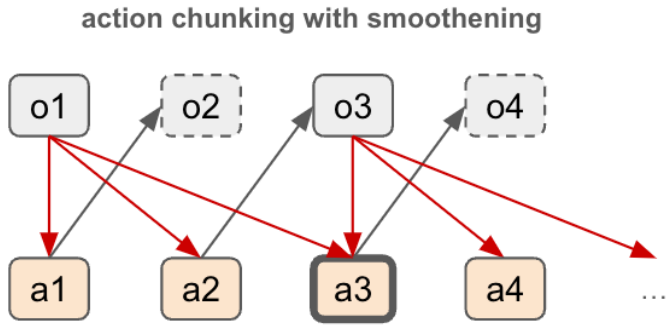
Short-horizon open-loop prediction

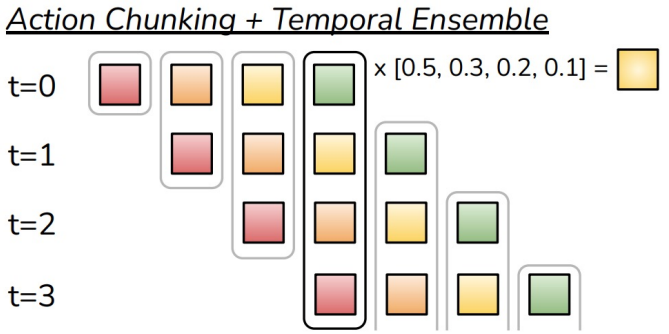
Both are non-trivially important!

Action Chunking

Action chunking = predicting small “chunk” of open loop actions



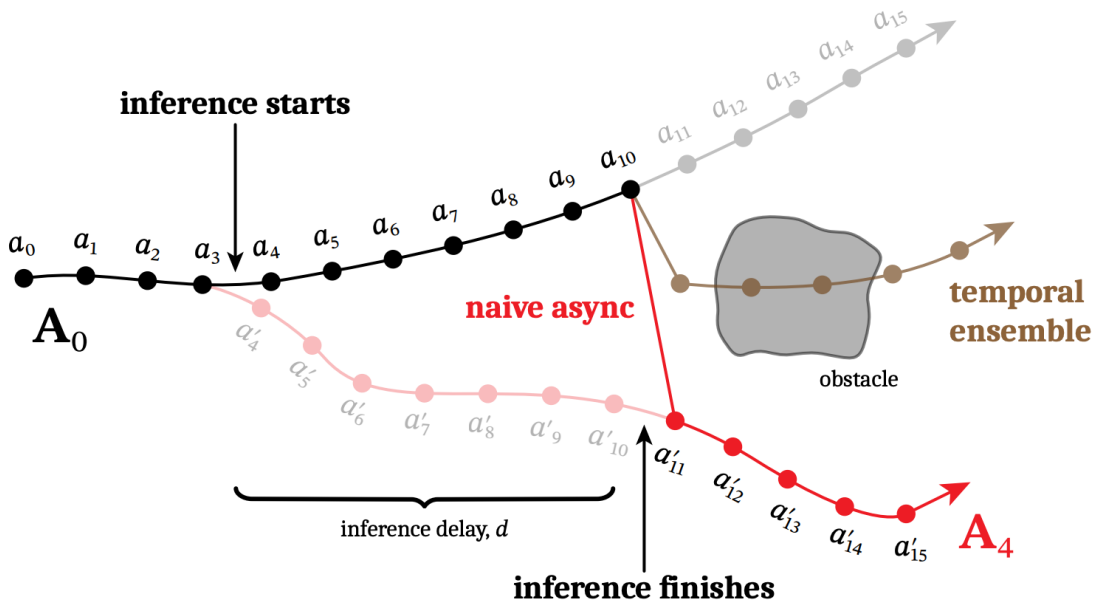




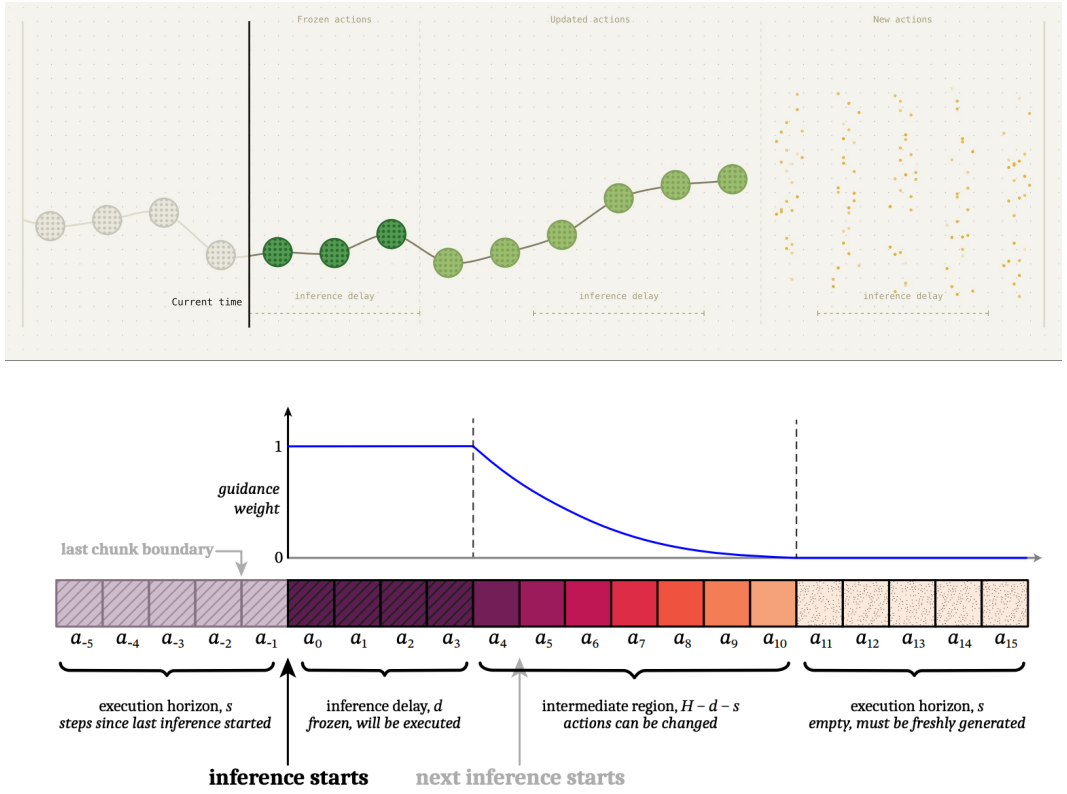
https://www.haonanyu.blog/post/action_chunking/
https://arxiv.org/abs/2304.13705

“Real-Time” Action Chunking

Real challenge when trying to operate robotic policies quickly



Leads to jerky/suboptimal motion



Freeze k steps and inpaint the rest

Real-Time Execution of Action Chunking Flow Policies, Black et al, ‘25

Generative Modeling – why does it help?

Do generative models help because human data is multimodal?

Much Ado About Noising: Dispelling the Myths of Generative Robotic Control

Chaoyi Pan[a,$] Giri Anantharaman[a] Nai-Chieh Huang[a] Claire Jin[a] Daniel Pfrommer[b]
Chenyang Yuan[c] Frank Permenter[c] Guannan Qu[a,†] Nicholas Boffi[a,†] Guanya Shi[a,†]
Max Simchowitz[a,†]

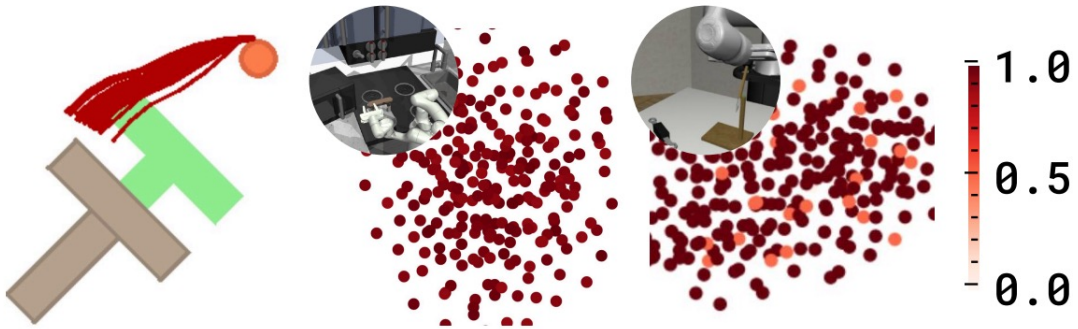
Shows that the benefits of generative models is not only in multimodal settings

Shows that the benefits come from stochasticity injection + iteration

Generative Modeling – why does it help?

Do generative models help because human data is multimodal?

Not much multimodality in existing policies



(a)Push T (b)Kitchen (c)Tool Hang

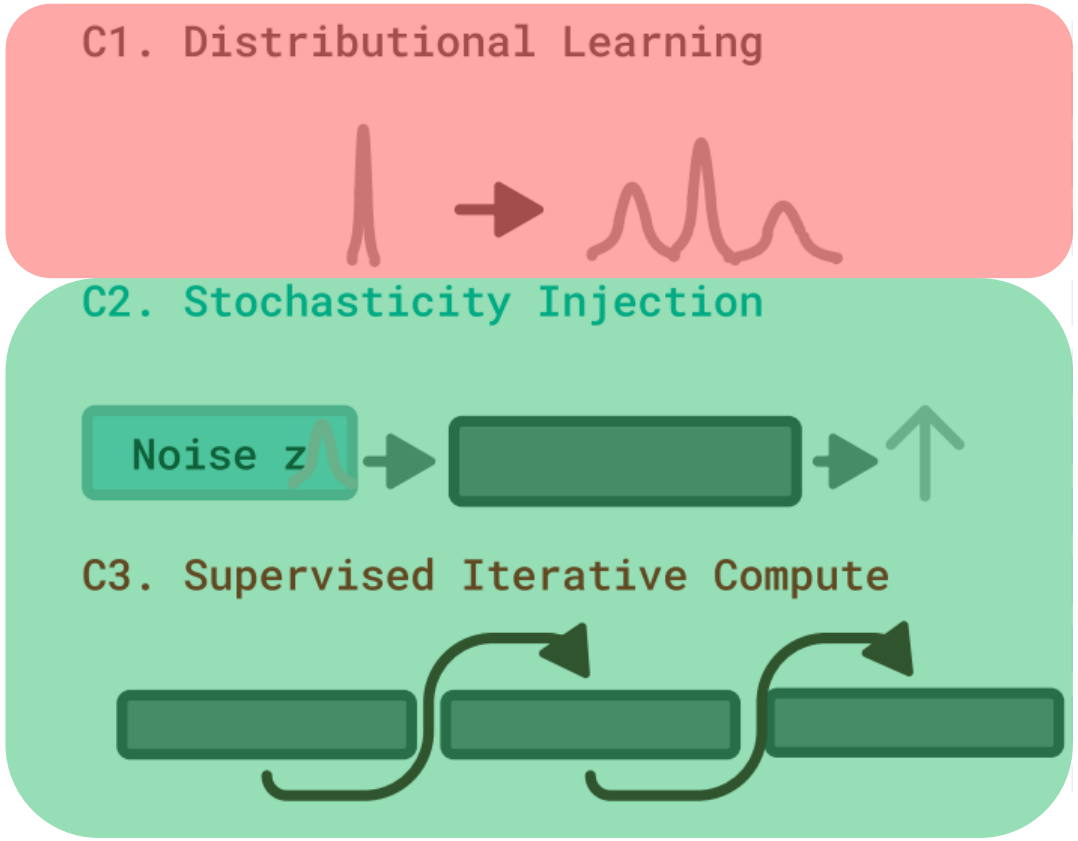
Task	$z \equiv 0$	$N(0, I)$	Mean z
Push-T	0.97	0.97	0.95
Kitchen	0.99	0.99	0.97
Tool-Hang	0.78	0.80	0.76

Better even with deterministic experts

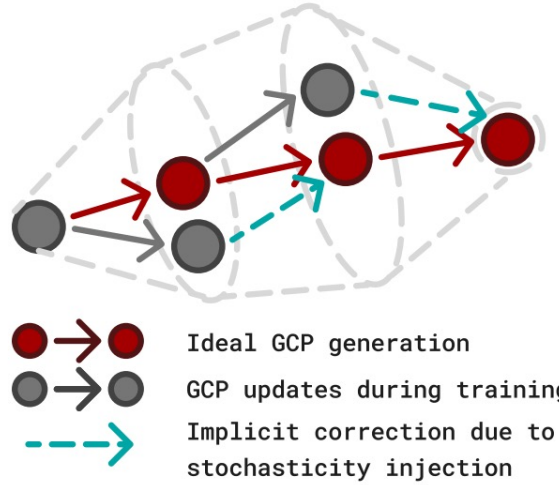
Dataset	Flow	Reg.
Original	0.78	0.58
Deterministic	0.72	0.64

Ok – so why do these flow models help?

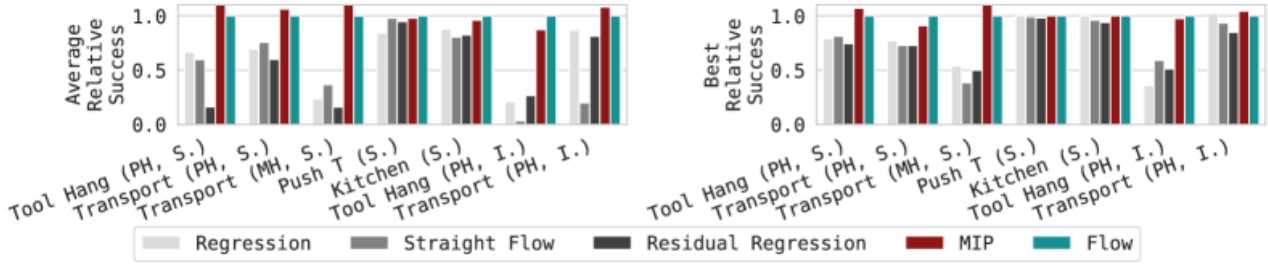
Generative Modeling – why does it help?



Train with stochasticity injected to "correct for errors" in generation

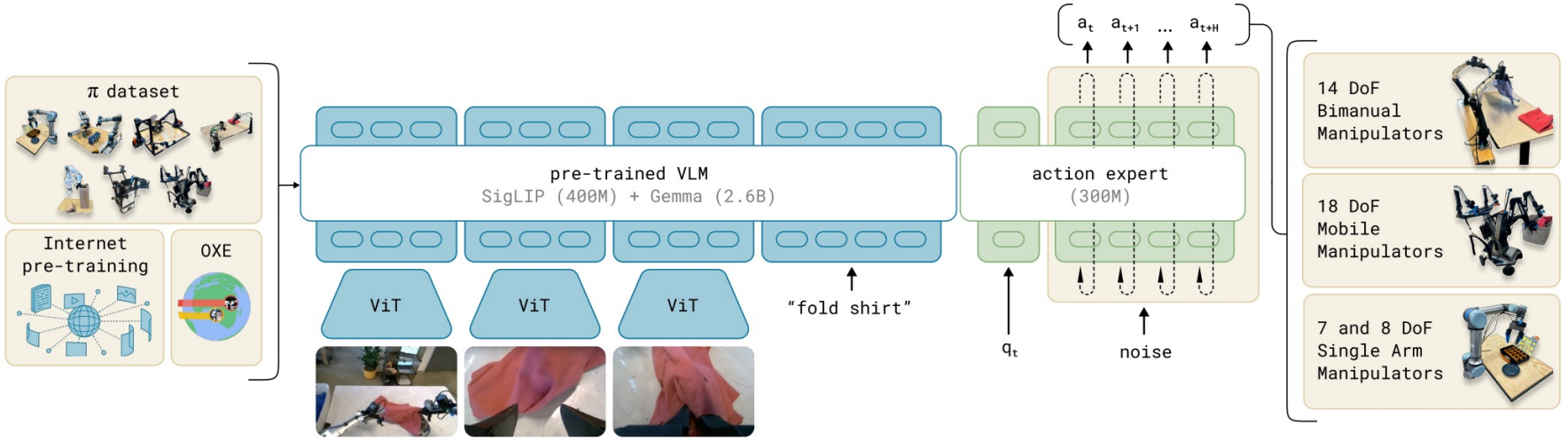


Kind of like DART but inside a single step of action generation

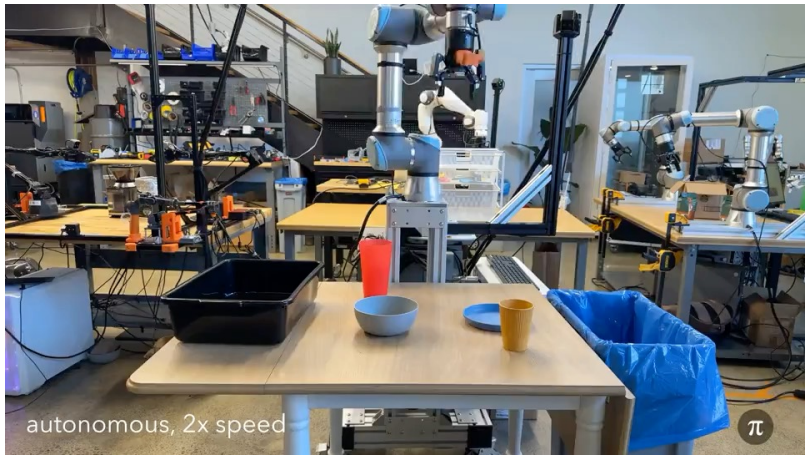


Minimal degradation when flow noise is removed

How is this being actualized today?



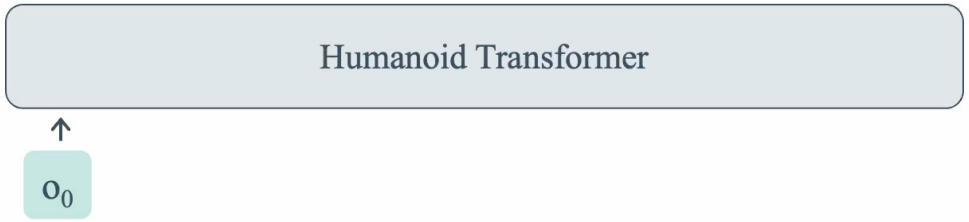
Vision-language-action policies



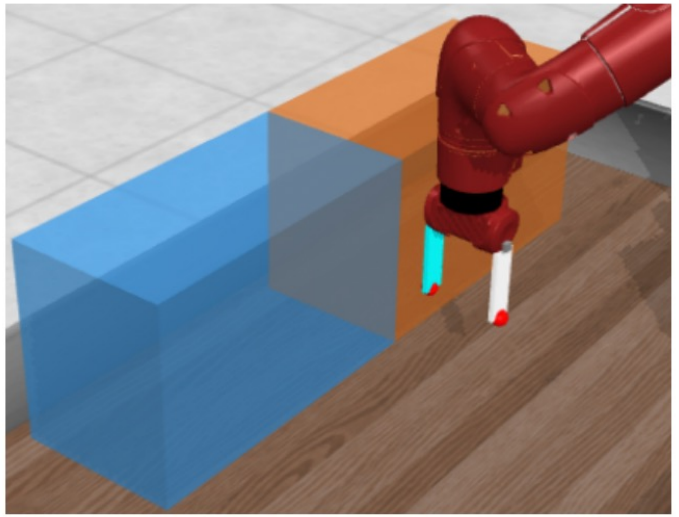
So does this solve all the issues in imitation?

Frontiers in Imitation Learning

Non-Markovian Demonstrators



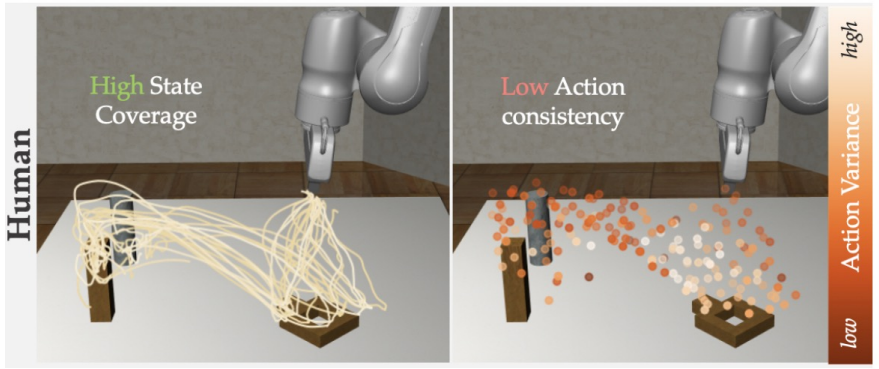
Characterizing generalization



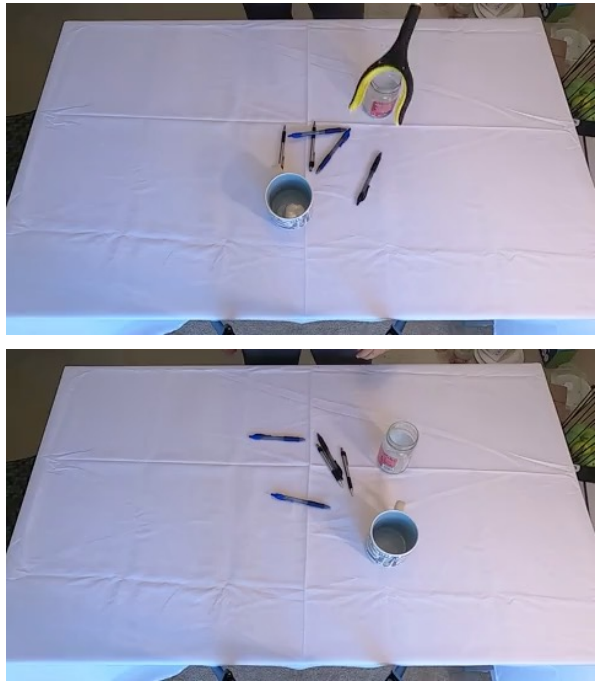
Action-Free Data

Frontiers in Imitation Learning

Data Curation and Quality



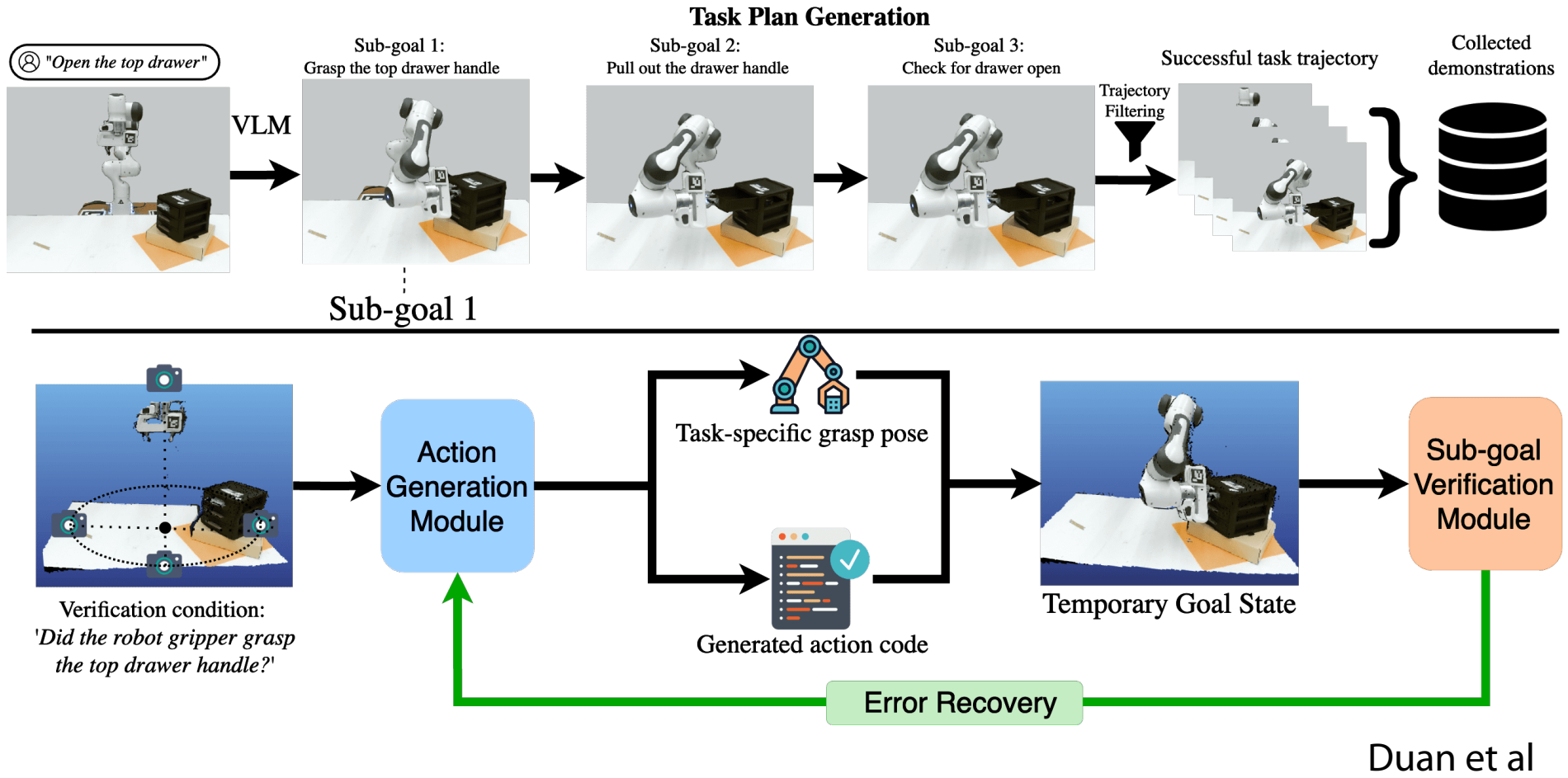
Embodiment Shift



Teleoperation Interfaces

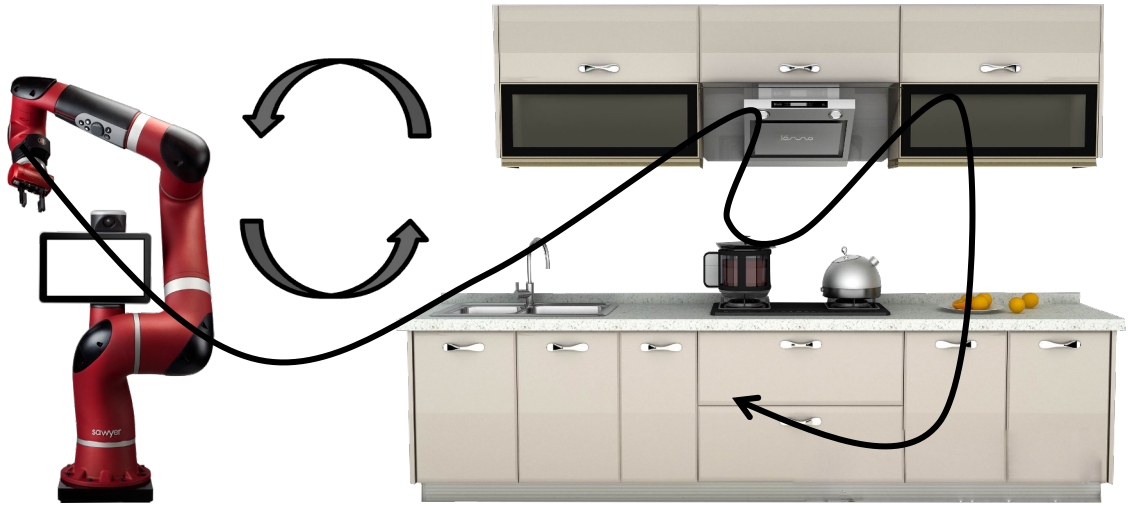
Frontiers in Imitation Learning

Learning how to retry and improve

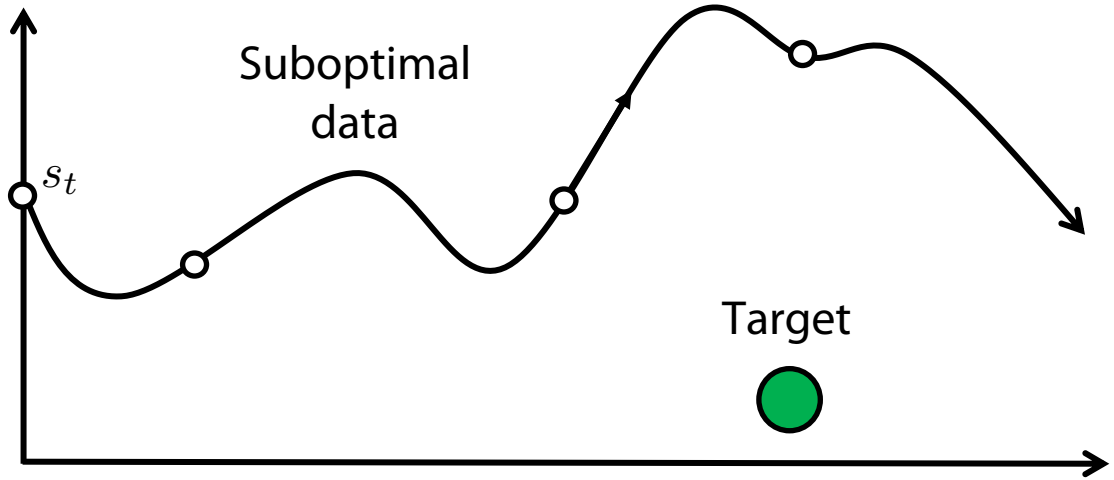


Duan et al

Accounting for Suboptimal Data

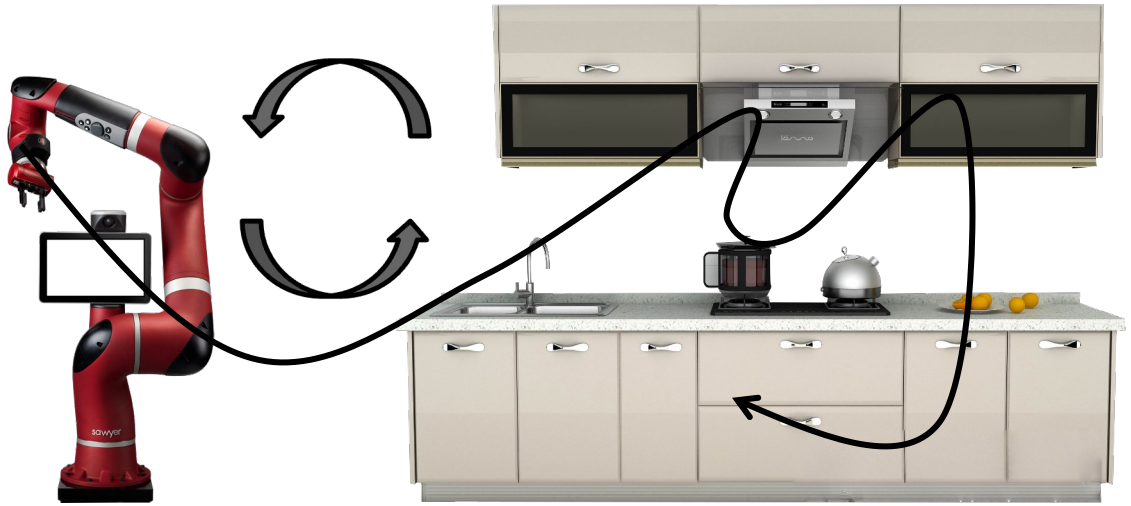


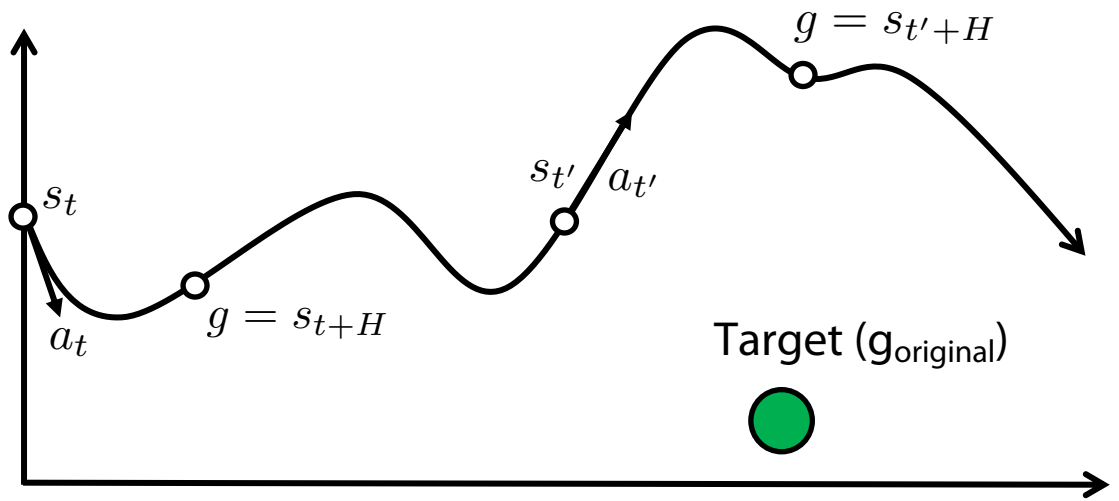
Random Policy π_θ



How can we use this suboptimal data, despite not reaching the target?

Hindsight relabeling for Imitation Learning





Key insight: maybe the data is not bad, it's just been labeled for the wrong problem!

↓

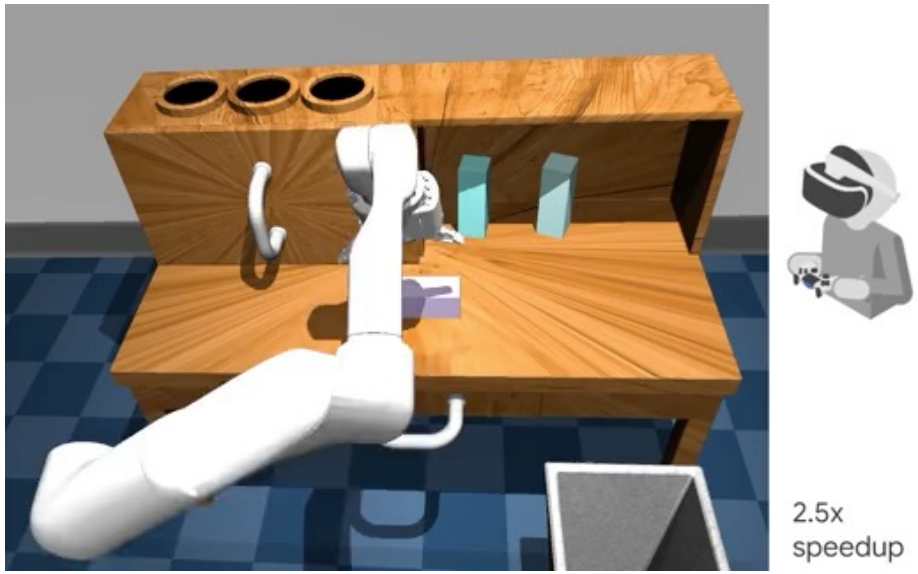
Relabel the right goal in "hindsight"

Learn a multi-goal policy $\pi_\theta(a|s, g)$

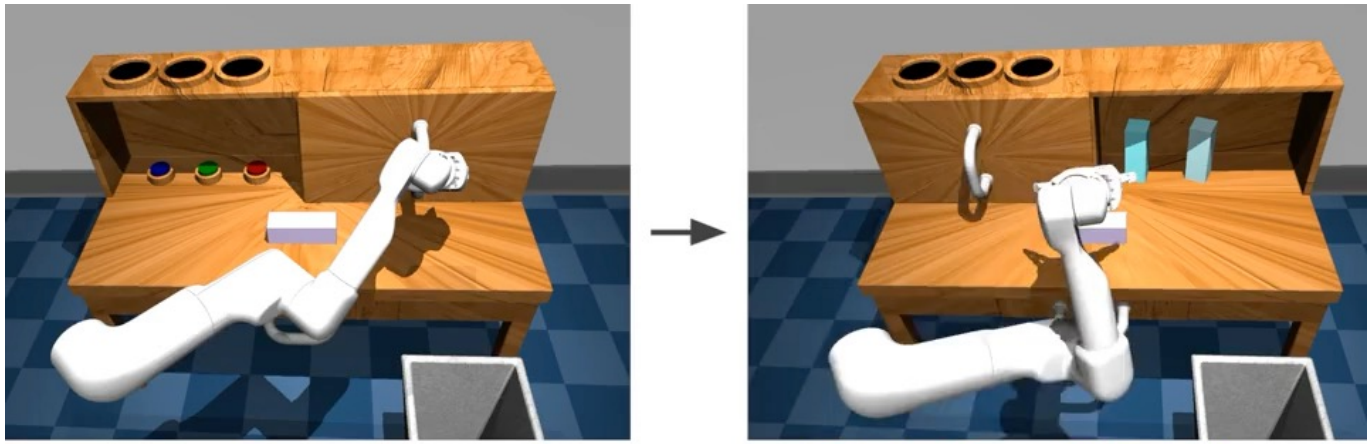
↓

Treat reached states as **optimal** goals

What does this result in?



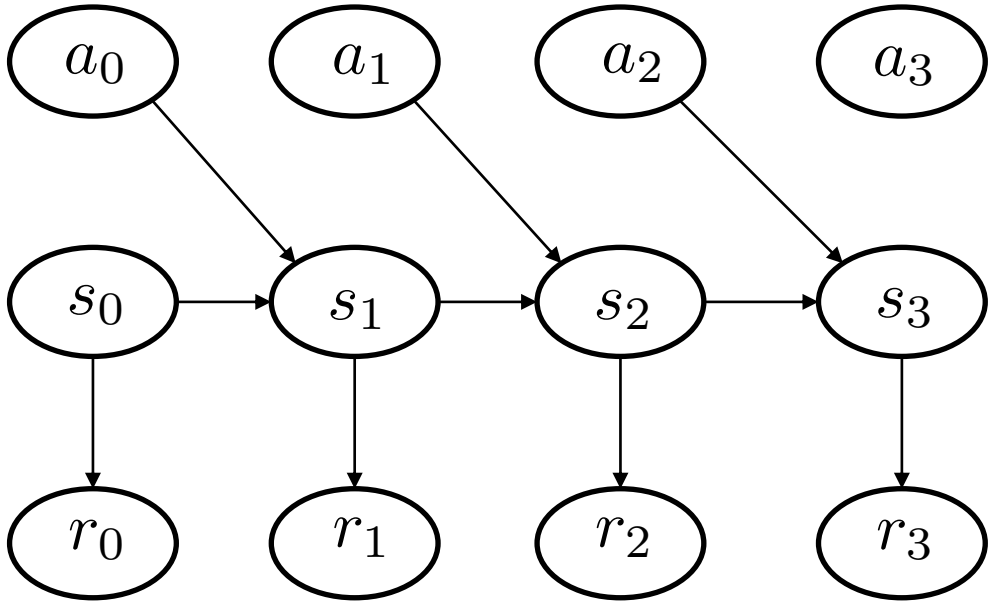
Undirected play data



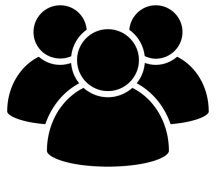
Goal-directed behavior

Dealing with non-Markovian demonstrators

Markov property $p(s_0, s_1, s_2, a_0, a_1, a_2) = p(s_0)p(a_0|s_0)p(s_1|s_0, a_0)p(a_1|s_1)p(s_2|s_1, a_1)p(a_2|s_2)$



Are human demonstrators Markovian?



$\pi_\theta(a|s)$

If we see the same thing twice, we do the same thing twice, regardless of what happened before

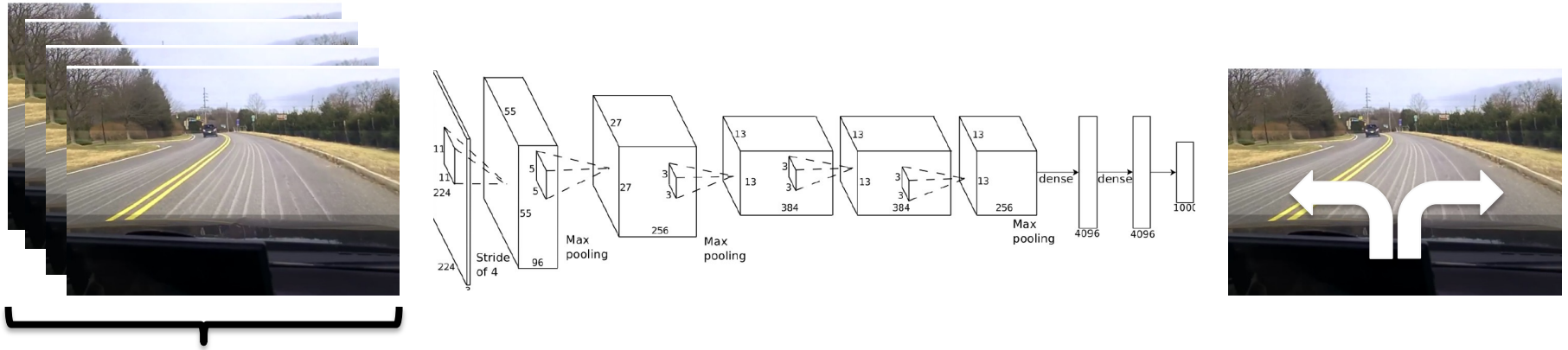
Not necessarily!

Humans often rely on history

Mixtures of Markovian humans may not be Markovian

How can we deal with non-Markovian demonstrators?

Learn $\pi_\theta(a_t | s_t, s_{t-1}, \ldots, s_0)$



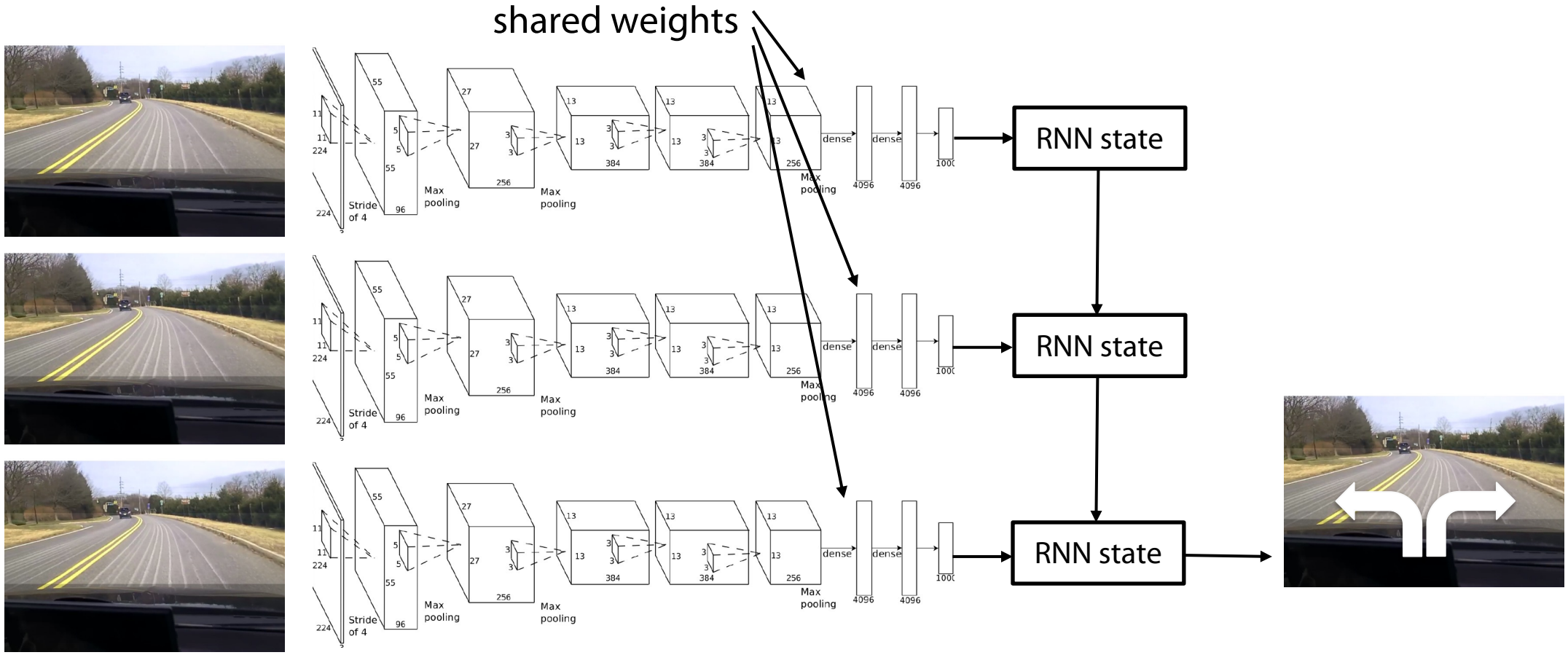
variable number of frames,
too many weights

Option 1: Stack all the past frames into a feedforward NN

Credit: Sergey Levine

How can we deal with non-Markovian demonstrators?

Learn $\pi_\theta(a_t | s_t, s_{t-1}, \ldots, s_0)$



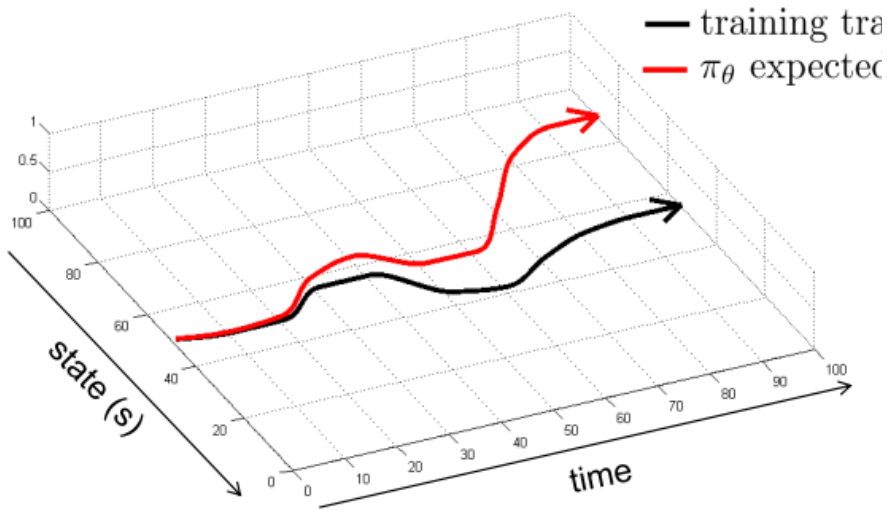
Option 2: Use a recurrent model (LSTM/transformer/RNN)

Credit: Sergey Levine

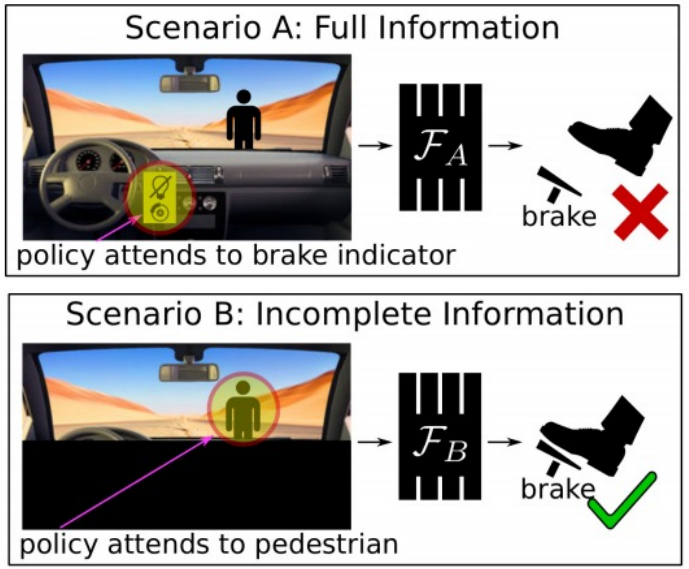
Why might this be challenging?

Learn $\pi_\theta(a_t | s_t, s_{t-1}, \ldots, s_0)$

Easier to go OOD

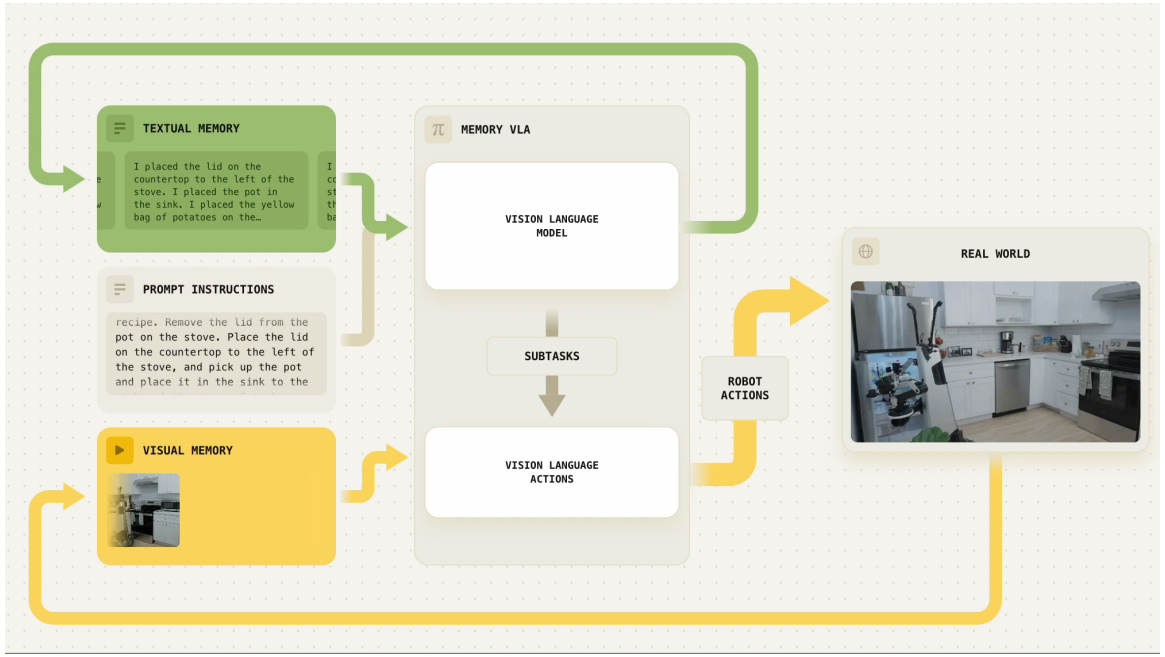


Learns spurious shortcut behaviors



Recent advances in memory for imitation learning

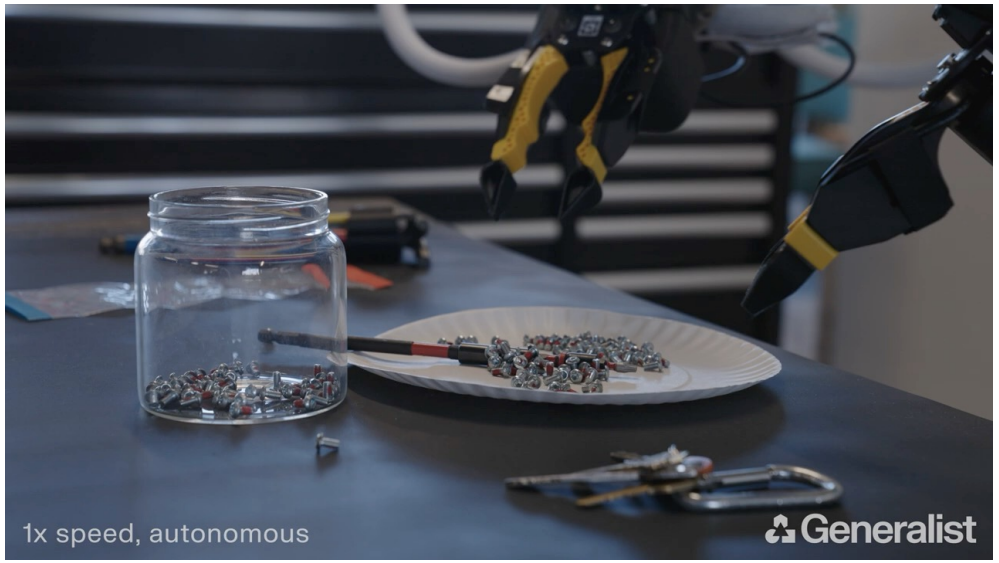
Use text + video encoder for long and short term memory

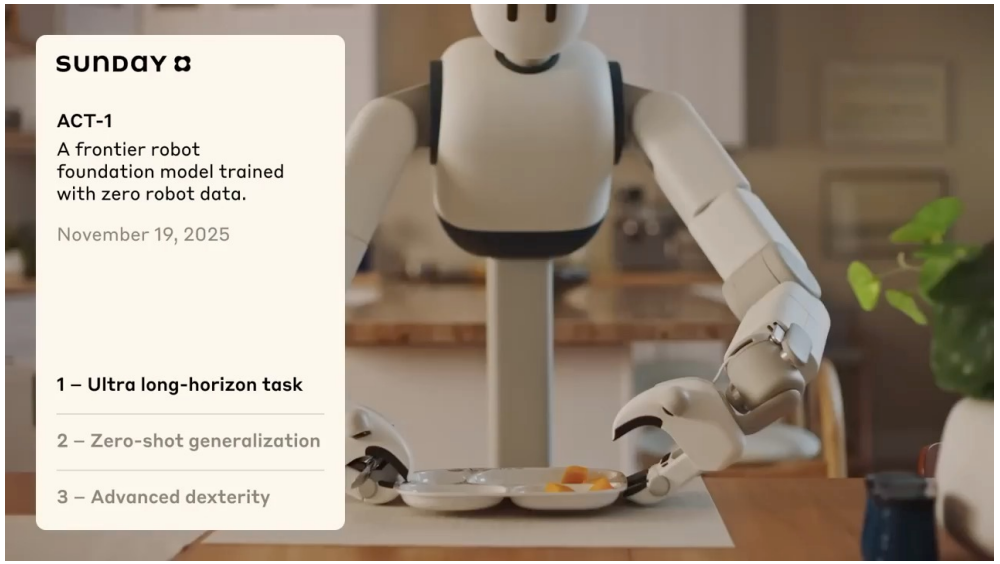


Still many open questions!

Some cool imitation videos

Generalist/Sunday Robots



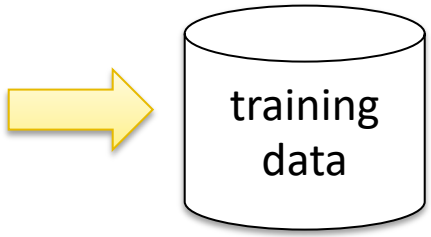


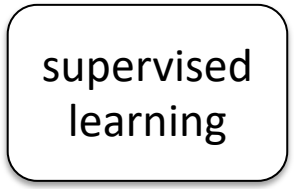
Sensorized people improve data throughput

Inheriting capabilities of vision-language models

Perspectives on Imitation

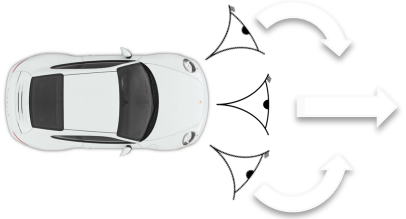
$\mathbf{o}_t$
$\mathbf{a}_t$



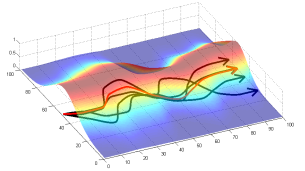


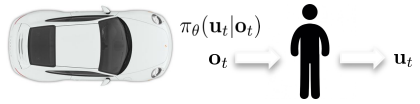
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

- Pros:
  - Easy to use, no additional infra
  - Can sometimes be unreasonably effective



- Cons:
  - Challenges of compounding error, multimodality
  - Doesn't really generalize
  - Very expensive in terms of data collection!





Lecture outline

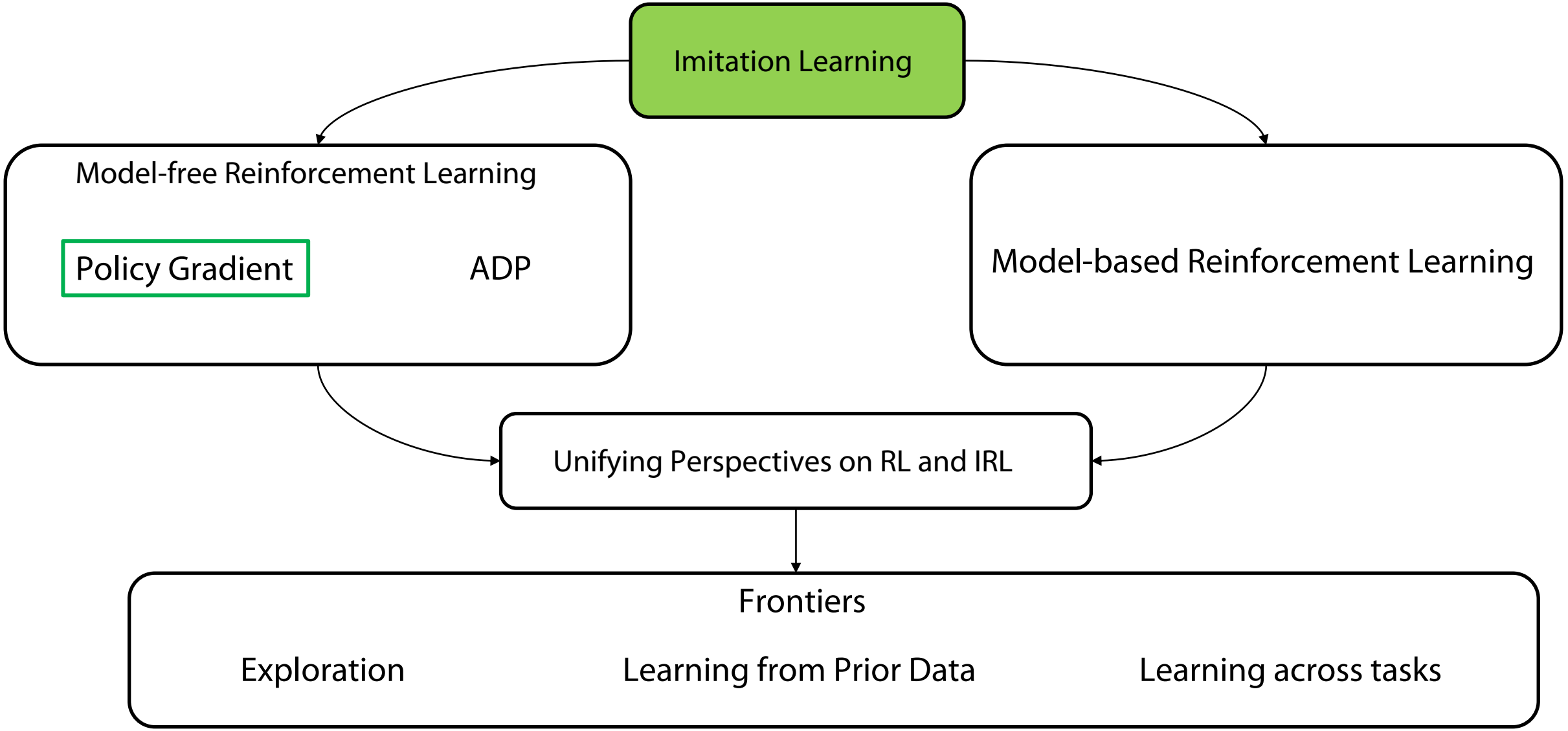
Recap: Imitation Learning via Behavior Cloning

Challenges with DAgger

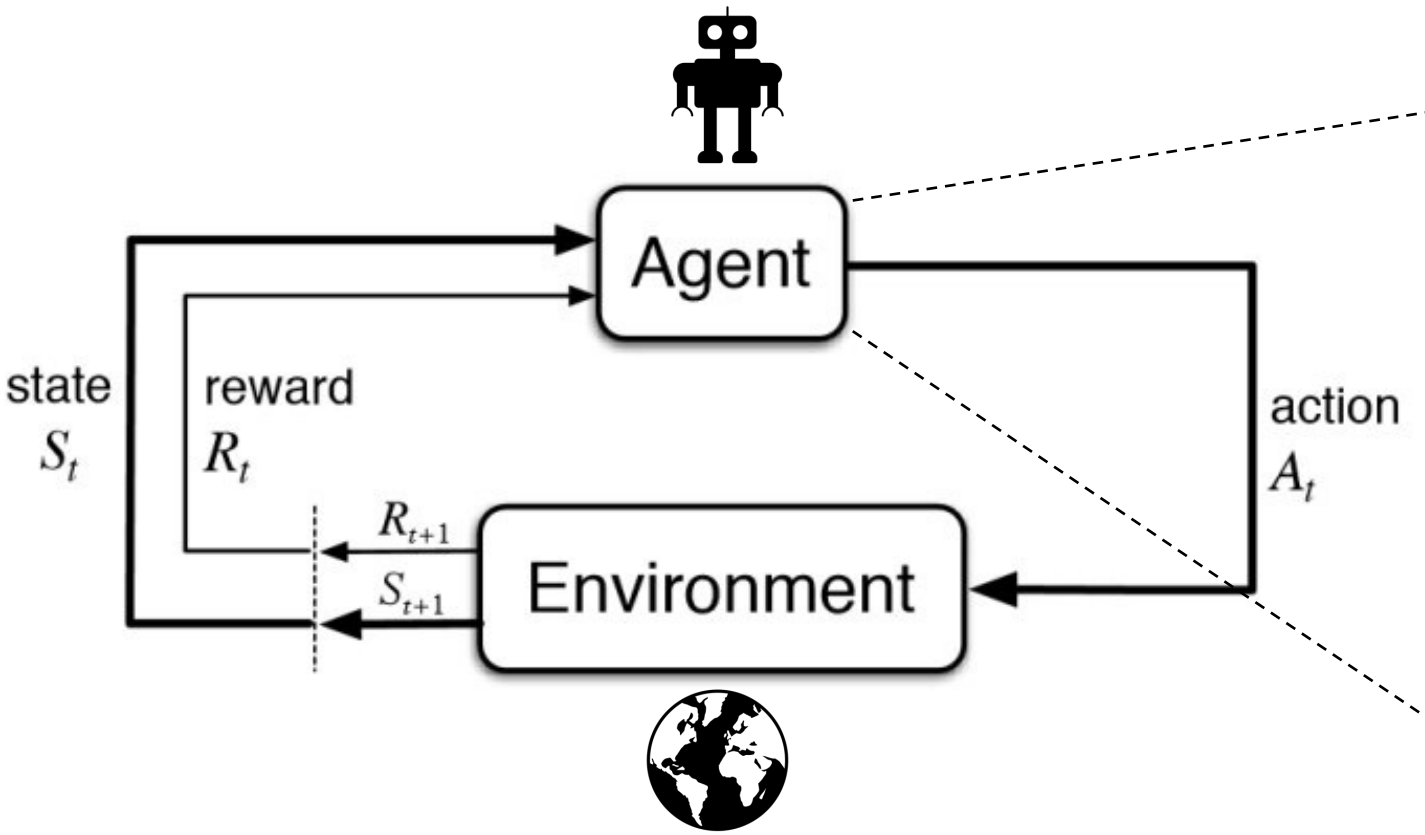
Modern Takes on Imitation Learning

Policy Gradient

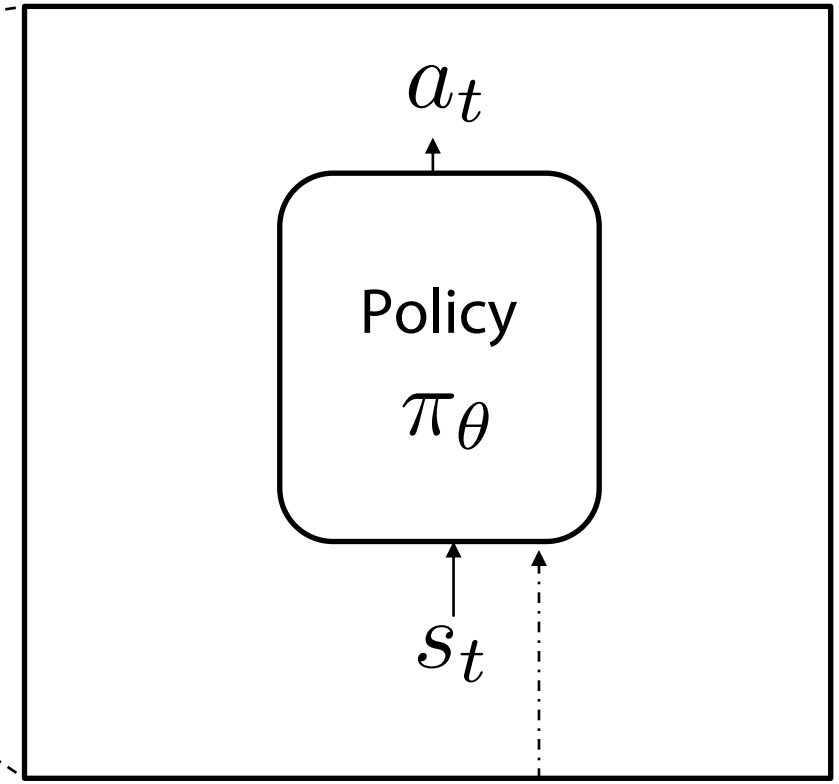
Class Structure



Objective of Reinforcement Learning



Rules for choosing actions



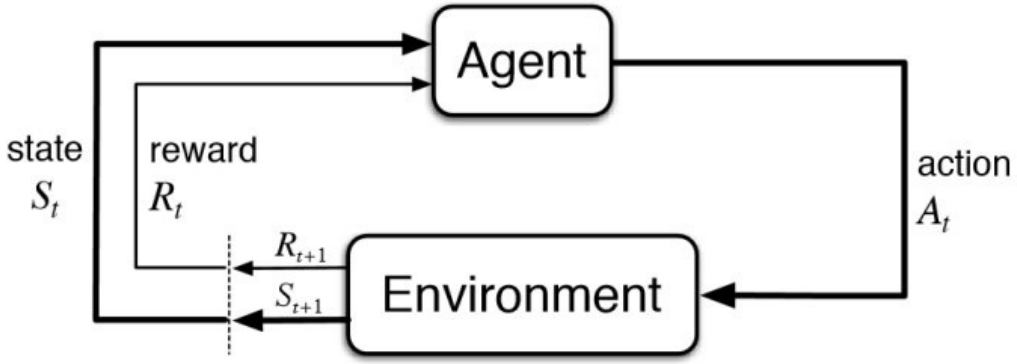
$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Trajectory sampled using policy

Needs to be learned

Finite horizon vs infinite horizon objective

$$\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} r(s_t,a_t)\right]$$



Finite horizon

$$\mathbb{E}_{\pi_\theta^t}\left[\sum_{t=0}^{T} r(s_t,a_t)\right]$$

Time-dependent policy
(not stationary)

$$\mathbb{E}_{\pi_\theta^t}\left[\sum_{t=0}^{T} \gamma^t r(s_t,a_t)\right]$$

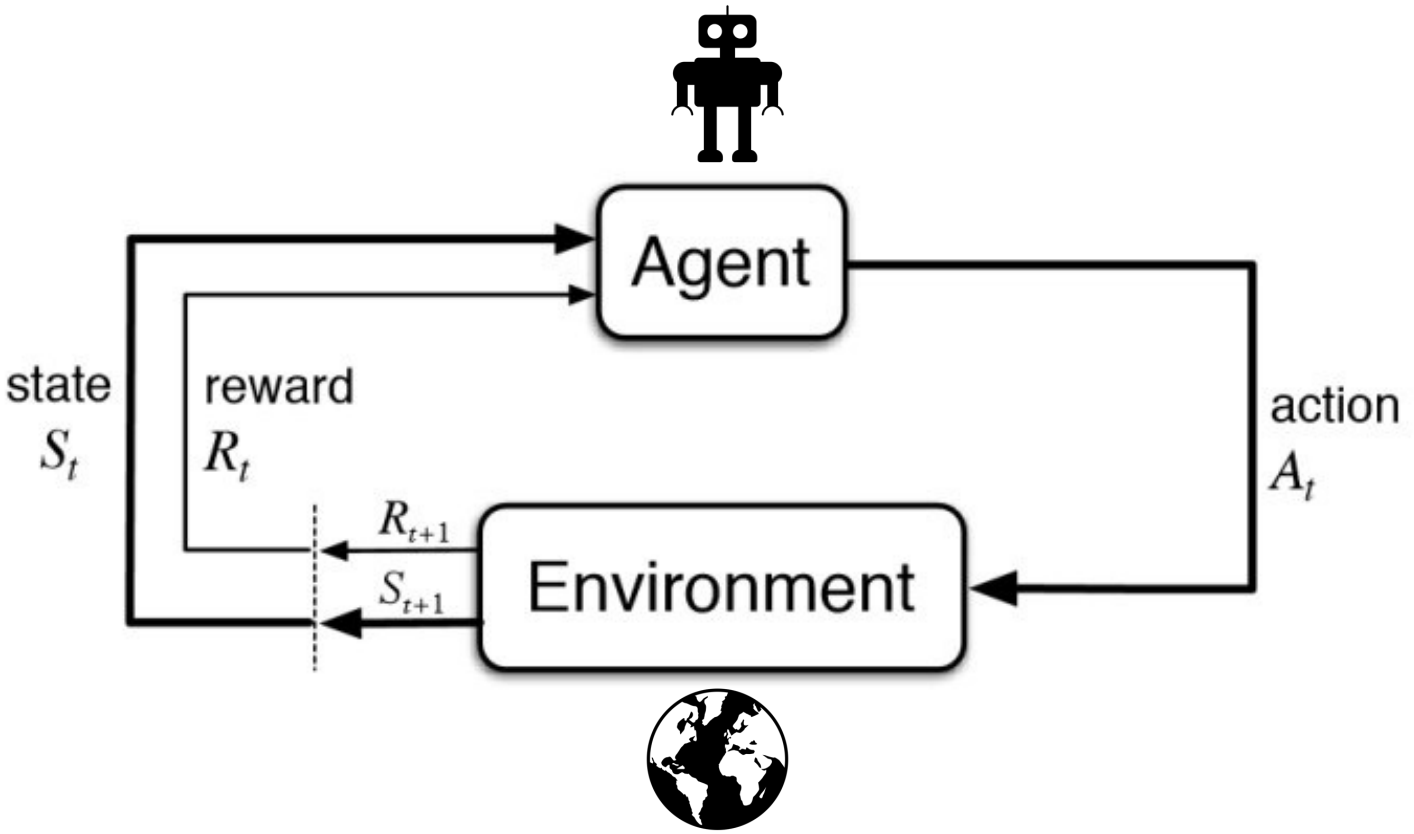
Infinite horizon discounted

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t)\right]$$

Time-independent (stationary) policy
→ Need discount to prevent blow up

Lemma: there always exists a stationary optimal policy

Objective of Reinforcement Learning



$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Trajectory sampled using policy

Assumptions:

1. Rewards are additive
2. Dynamics can be sampled from, but functional form is unknown
3. Rewards are provided as every state is visited, functional form is unknown

Connection to Optimal Control

Closely related: typically problem of finding control given a plant

$$\min_{x,u} \int_0^x L(t, x(t), u(t)).dx$$

$$\text{w.r.t}$$

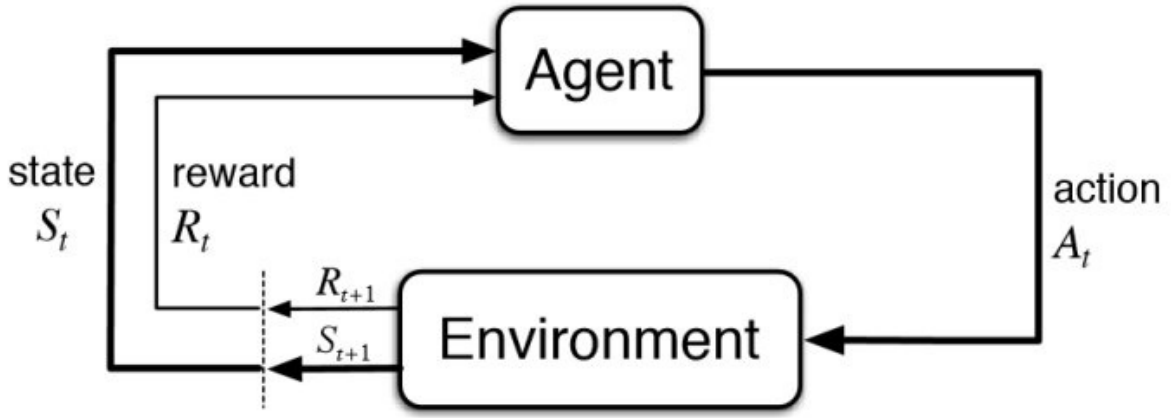
$$x'(t) = f(x(t), u(t))$$

$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Main difference: model known vs unknown

Minor differences: Cost vs reward, discrete vs continuous time

How should we optimize this objective?



$$\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} r(s_t, a_t)\right]$$

Gradient Ascent

Dynamic Programming

Model-Based Optimization

Each method has it's own +/-

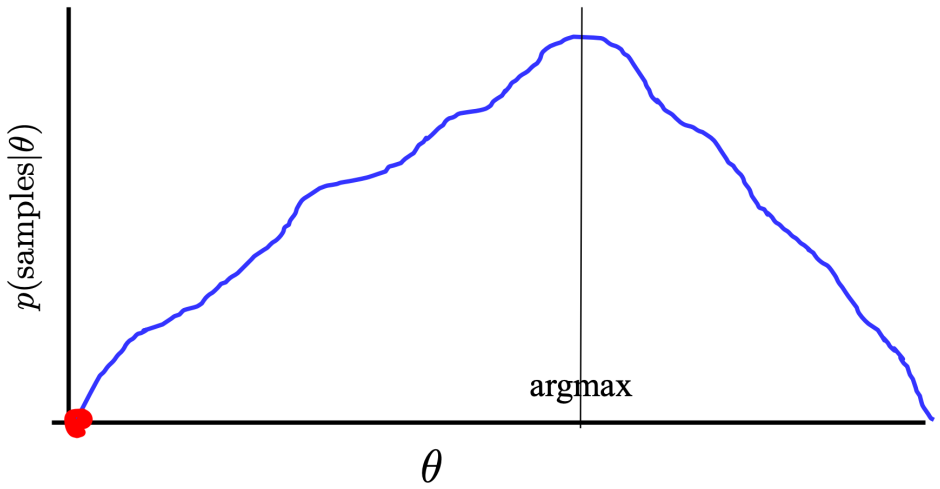
Lecture outline

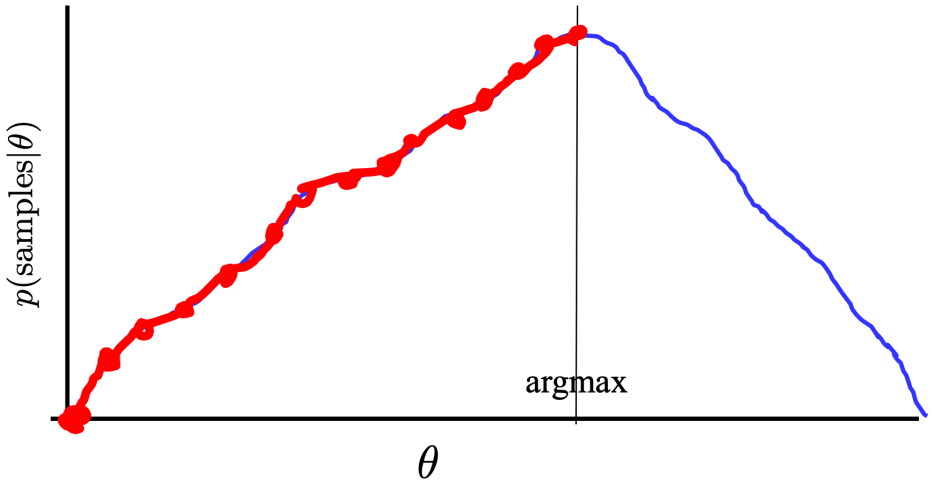
Deriving the Policy Gradient

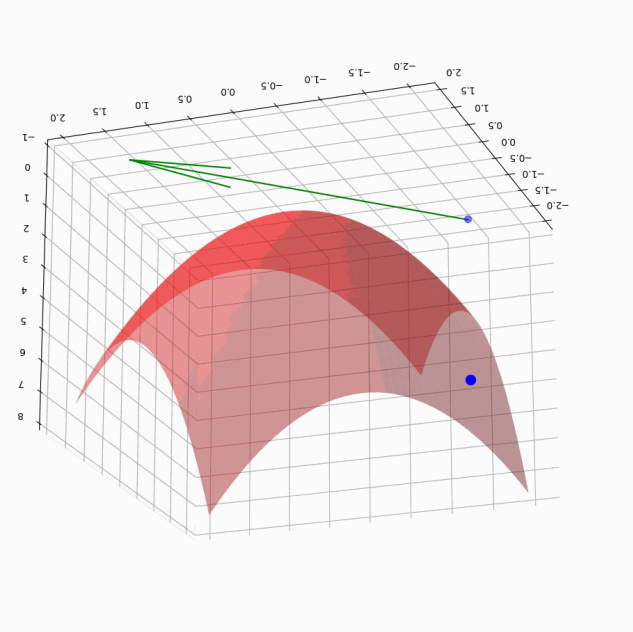
What makes the Policy Gradient Challenging? - Variance

What makes the Policy Gradient Challenging? – Covariant Parameterization

Gradient Ascent







Simple view – move the parameters in the direction of the gradient of the objective

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta J(\theta)|_{\theta=\theta_i}$$

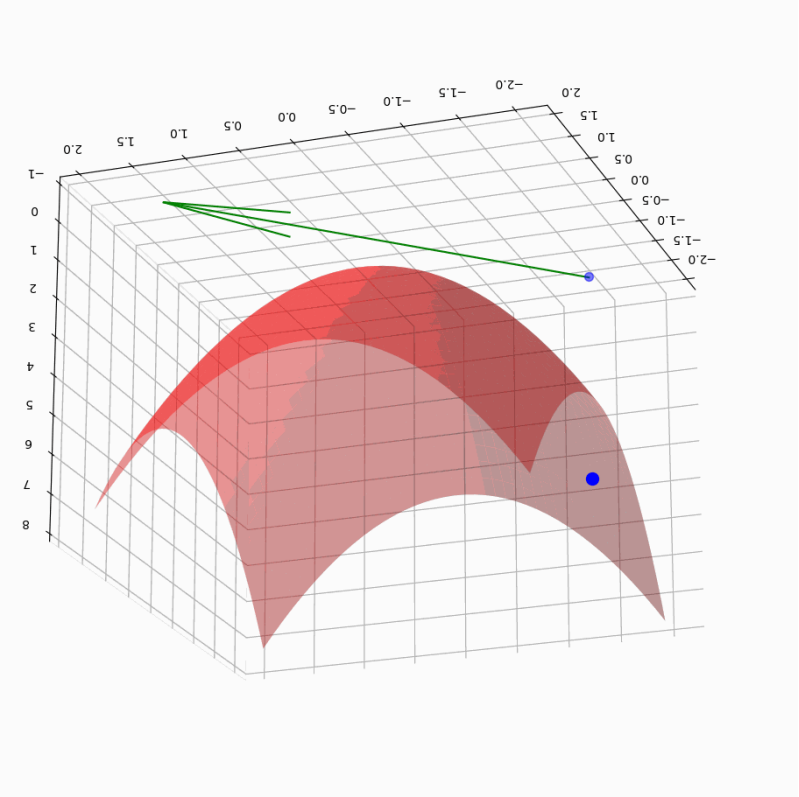
More later: can be derived as steepest ascent in Euclidean norm

Gradient Ascent for Supervised Learning

Recall our imitation learning objective

$$\arg\max_{\theta} \mathbb{E}_{(s^*,a^*)\sim\mathcal{D}} \left[\log \pi_\theta(a^*|s^*)\right]$$

Let's apply gradient ascent



$$\nabla_\theta \mathbb{E}_{(s^*,a^*)\sim\mathcal{D}} \left[\log \pi_\theta(a^*|s^*)\right]$$

$$\nabla_\theta \int p(s^*,a^*) \log \pi_\theta(a^*|s^*) ds^* da^*$$

$$\int p(s^*,a^*) \nabla_\theta \log \pi_\theta(a^*|s^*) ds^* da^*$$

$$\mathbb{E}_{(s^*,a^*)\sim\mathcal{D}} \left[\nabla_\theta \log \pi_\theta(a^*|s^*)\right]$$

Compute gradient and average

Ok let's do gradient ascent for the RL objective

$$\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} r(s_t,a_t)\right]$$

$$= \int p_\theta(\tau)R(\tau)d\tau$$

<u>REINFORCE gradient descent (RL)</u>

$$\nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}\left[f(x)\right]$$

(Cannot simply compute average of expectation)

<u>Standard gradient descent (supervised learning)</u>

Gradient wrt expectation variable, not of integrand!

(Whiteboard)

$$\nabla_\theta \mathbb{E}_{x\sim g(x)}\left[f_\theta(x)\right]$$

(Gradient passes inside the expectation – compute gradient and average)

Taking the gradient of sum of rewards

$$\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} r(s_t, a_t)\right]$$

Let's take the gradient of this objective

$$J(\theta) = \int p_\theta(\tau) R(\tau) d(\tau)$$

Let's think about this from the trajectory view

$$\nabla_\theta J(\theta) = \nabla_\theta \int p_\theta(\tau) R(\tau) d(\tau)$$

We need to express this in a way that we can evaluate with expectations

$$= \int \nabla_\theta p_\theta(\tau) R(\tau) d(\tau) \quad = \int \frac{p_\theta(\tau)}{p_\theta(\tau)} \nabla_\theta p_\theta(\tau) R(\tau) d(\tau)$$

$$= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau) d(\tau) \quad = \mathbb{E}_{p_\theta(\tau)}\left[\nabla_\theta \log p_\theta(\tau) R(\tau)\right]$$

REINFORCE trick

$$\frac{d\log(x)}{d\theta} = \frac{d\log(x)}{dx}\frac{dx}{d\theta} = \frac{1}{x}\frac{dx}{d\theta}$$

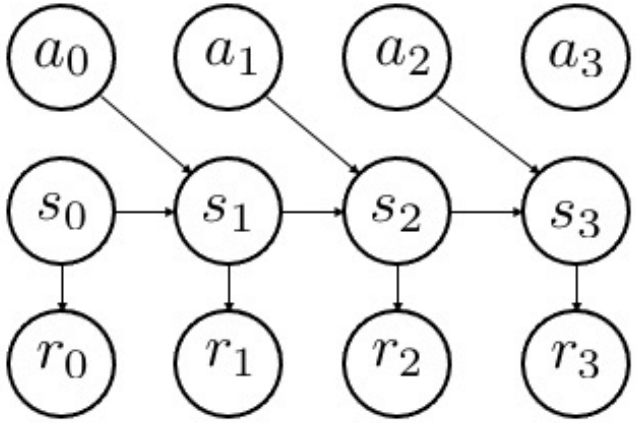
Use chain rule

Taking the gradient of return

Initial State Dynamics Policy

$$p_\theta(\tau) = p(s_0)\Pi_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$$

(Ancestral sampling)



$$\log p_\theta(\tau) = \log p(s_0) + \sum_{t=0}^{T-1} \log p(s_{t+1}|s_t, a_t) + \log \pi(a_t|s_t)$$

$$\nabla_\theta \log p_\theta(\tau) = \nabla_\theta \log p(s_0) + \sum_{t=0}^{T-1} \nabla_\theta \log p(s_{t+1}|s_t, a_t) + \nabla_\theta \log \pi(a_t|s_t)$$

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t)$$

Model Free!!

Taking the gradient of return

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log p_\theta(\tau) \sum_{t=0}^{T} r(s_t, a_t) \right]$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\substack{s_0 \sim p(s_0) \\ s_{t+1} \sim p(s_{t+1}|s_t, a_t) \\ a_t \sim \pi(a_t|s_t)}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=0}^{T} r(s_t, a_t) \right]$$

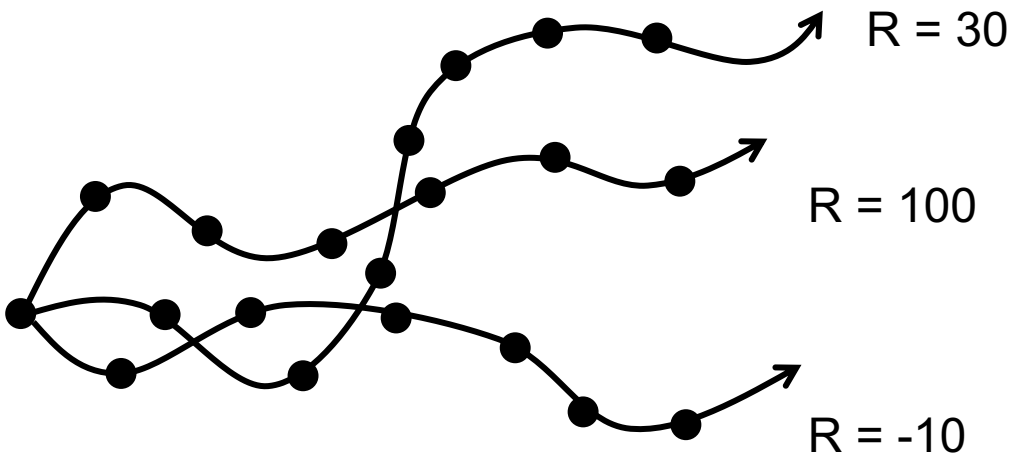
$$\approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$ (approximating using samples)

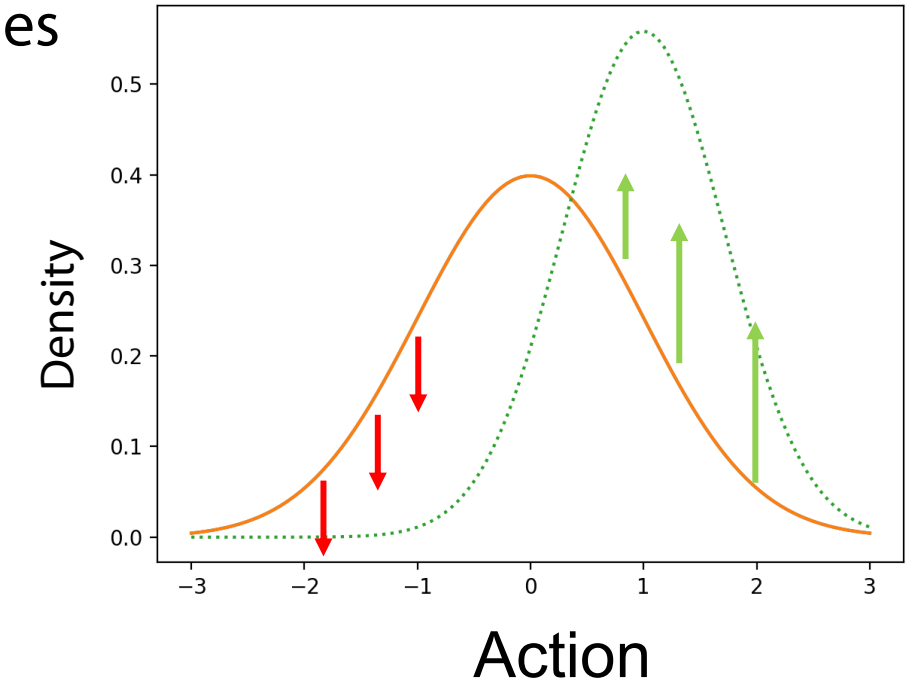
(Monte-Carlo approximation)

What does this mean?

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau \approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

Increase the likelihood of actions in high return trajectories

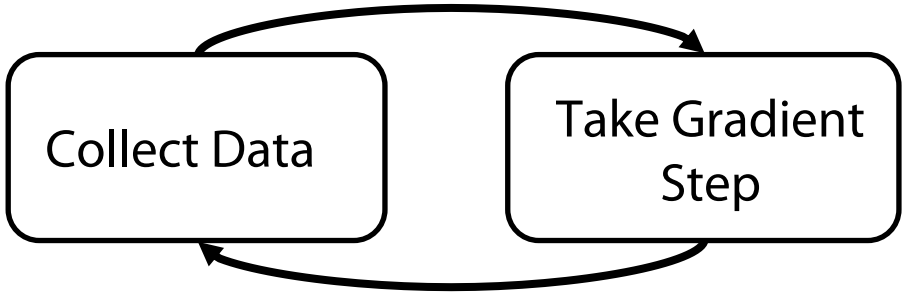




Resulting Algorithm (REINFORCE)

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta J(\theta)|_{\theta=\theta_i}$$



On-policy

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

How is this related to supervised learning?

Reinforcement Learning

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$

$$\approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

Supervised Learning

$$\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\log p_\theta(y|x)\right]$$

$$\approx \frac{1}{N} \sum_i \nabla_\theta \log p_\theta(y^i | x^i)$$

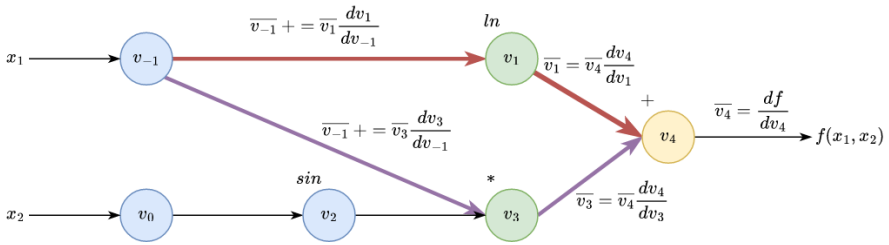
PG = select good data + increase likelihood of selected data

How do we implement this?

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau \approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$



Compute gradients with autodiff

Sum up rewards in a trajectory

PyTorch

How do we implement this?

Maximum likelihood:

```
# Given:
# actions - (N*T) x Da tensor of actions
# states - (N*T) x Ds tensor of states
# Build the graph:
logits = policy.predictions(states) # This should return (N*T) x Da tensor of action logits
negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=actions, logits=logits)
loss = tf.reduce_mean(negative_likelihoods)
gradients = loss.gradients(loss, variables)
```

^Standard maximum likelihood training

How do we implement this?

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Policy gradient:

```
# Given:
# actions - (N*T) x Da tensor of actions
# states - (N*T) x Ds tensor of states
# q_values – (N*T) x 1 tensor of estimated state-action values
# Build the graph:
logits = policy.predictions(states) # This should return (N*T) x Da tensor of action logits
negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=actions, logits=logits)
weighted_negative_likelihoods = tf.multiply(negative_likelihoods, q_values)
loss = tf.reduce_mean(weighted_negative_likelihoods)
gradients = loss.gradients(loss, variables)
```

→ Sum of rewards

Formalizes the notion of trial and error

How do we implement this?

$$\nabla_\theta \log \pi_\theta(a_t|s_t)$$

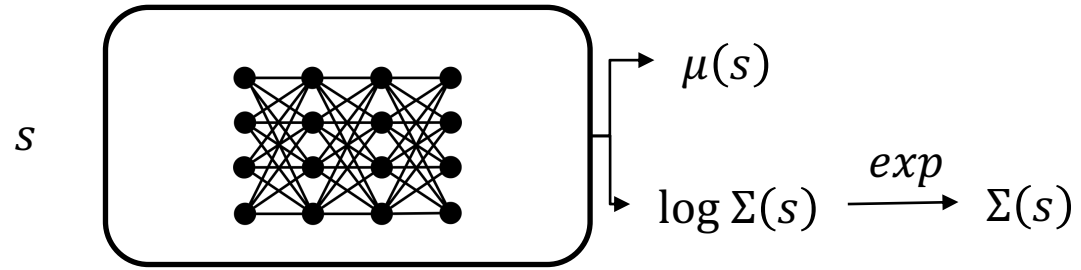
Let's try it for a Gaussian

$$\pi(\mathbf{a} \mid \boldsymbol{s})$$

$$= \pi(\mathbf{a} \mid \boldsymbol{\mu_\theta}(\boldsymbol{s}), \boldsymbol{\Sigma_\theta}(\boldsymbol{s}))$$

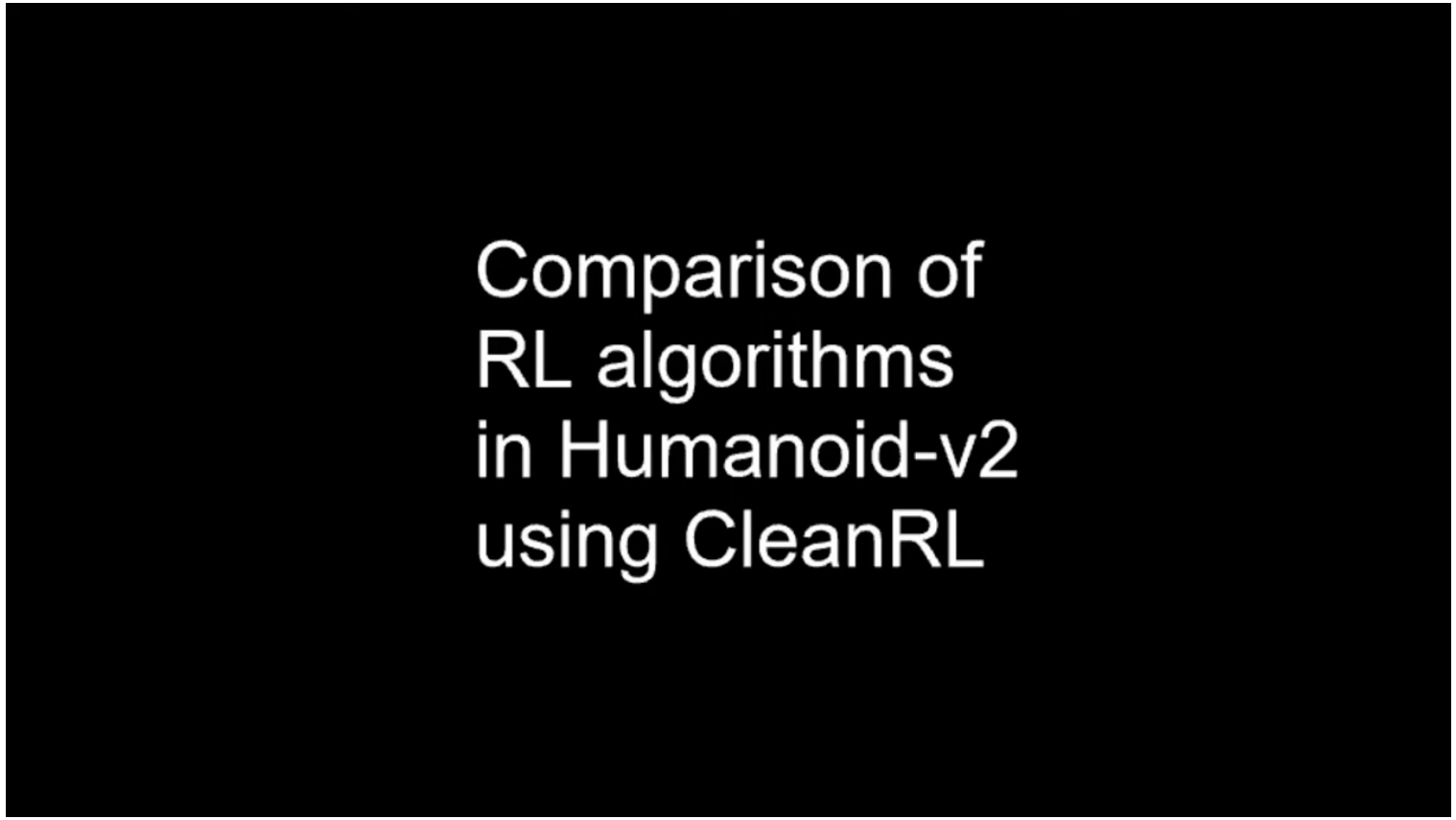
$$= \pi(\mathbf{a} \mid \boldsymbol{\mu_\theta}(\boldsymbol{s}), \boldsymbol{\Sigma_\theta}(\boldsymbol{s})) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma_\theta}(\boldsymbol{s})|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu_\theta}(\boldsymbol{s}))^\top \boldsymbol{\Sigma_\theta}(\boldsymbol{s})^{-1}(\mathbf{x} - \boldsymbol{\mu_\theta}(\boldsymbol{s}))\right)$$

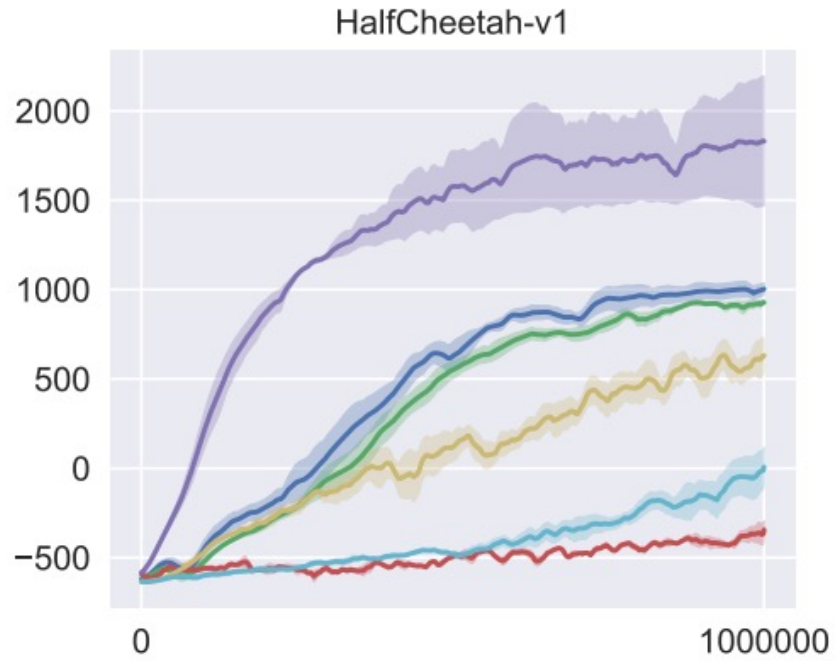
Similar for categorical or other distributions



Easier for distributions where likelihood can be expressed symbolically

Does this work?





Kind of?

Lecture outline

Recap: Imitation Learning via Behavior Cloning

Challenges with DAgger

Modern Takes on Imitation Learning

Policy Gradient

Fin.