Imitation Learning

CS 224R

Course reminders

- Start forming **final project groups** (survey due Weds April 16)
- **Homework 1** out today, due Fri April 18
- PyTorch tutorial today at 1:30 pm in **Gates B1**

Partial Recap

state $\mathbf{s}_t$ - the state of the "world" at time t

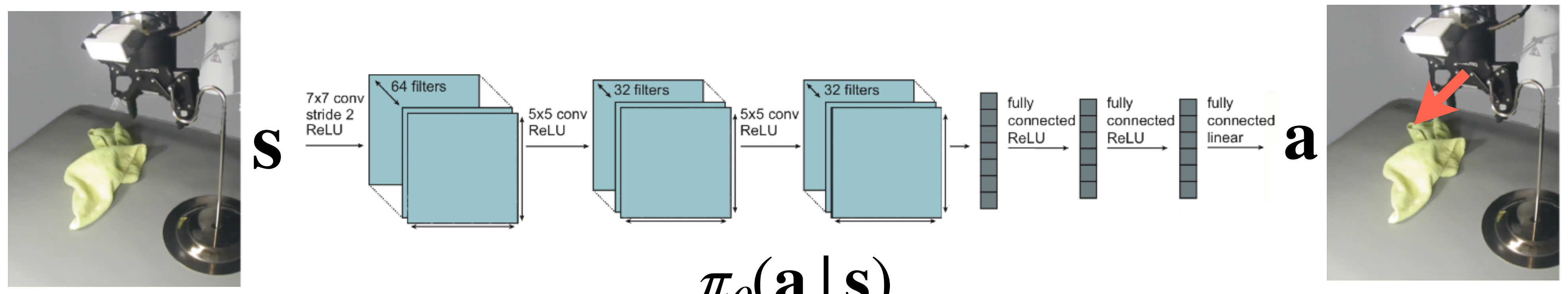
observation $\mathbf{o}_t$ - what the agent observes at time t

action $\mathbf{a}_t$ - the decision taken at time t

trajectory τ - sequence of states/observations and actions

$$(\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \ldots, \mathbf{s}_T, \mathbf{a}_T)$$

reward function $r(\mathbf{s}, \mathbf{a})$ - how good is $\mathbf{s}, \mathbf{a}$?



$\pi_\theta(\mathbf{a} \mid \mathbf{s})$

policy $\pi(\mathbf{a}_t \mid \mathbf{s}_t)$ or $\pi(\mathbf{a}_t \mid \mathbf{o}_{t-m:t})$ - behavior, usually what we are trying to learn

can be represented using a generative model

The plan for today

Imitation Learning

1. Imitation learning basics
2. Learning expressive policy distributions
3. Learning from online interventions
4. Time permitting: how to collect demonstrations

Topic of homework 1!

Key learning goals:

- how to represent distributions with neural networks
- why expressive distributions matter for imitation learning
- what are compounding errors and how to address them

What is the goal of imitation learning?

Data: Given trajectories collected by an expert

"demonstrations" $\mathscr{D} := \{(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T)\}$

(sampled from some unknown policy π_{expert})

Goal: Learn a policy π_θ that performs at the level of the expert policy, by mimicking it.

Example

- Dataset from human drivers
- Sensor readings + steering commands

Imitation learning - version 0

Deterministic policy

0. Given demonstrations collected by an expert $\mathcal{D} := \{(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T)\}$

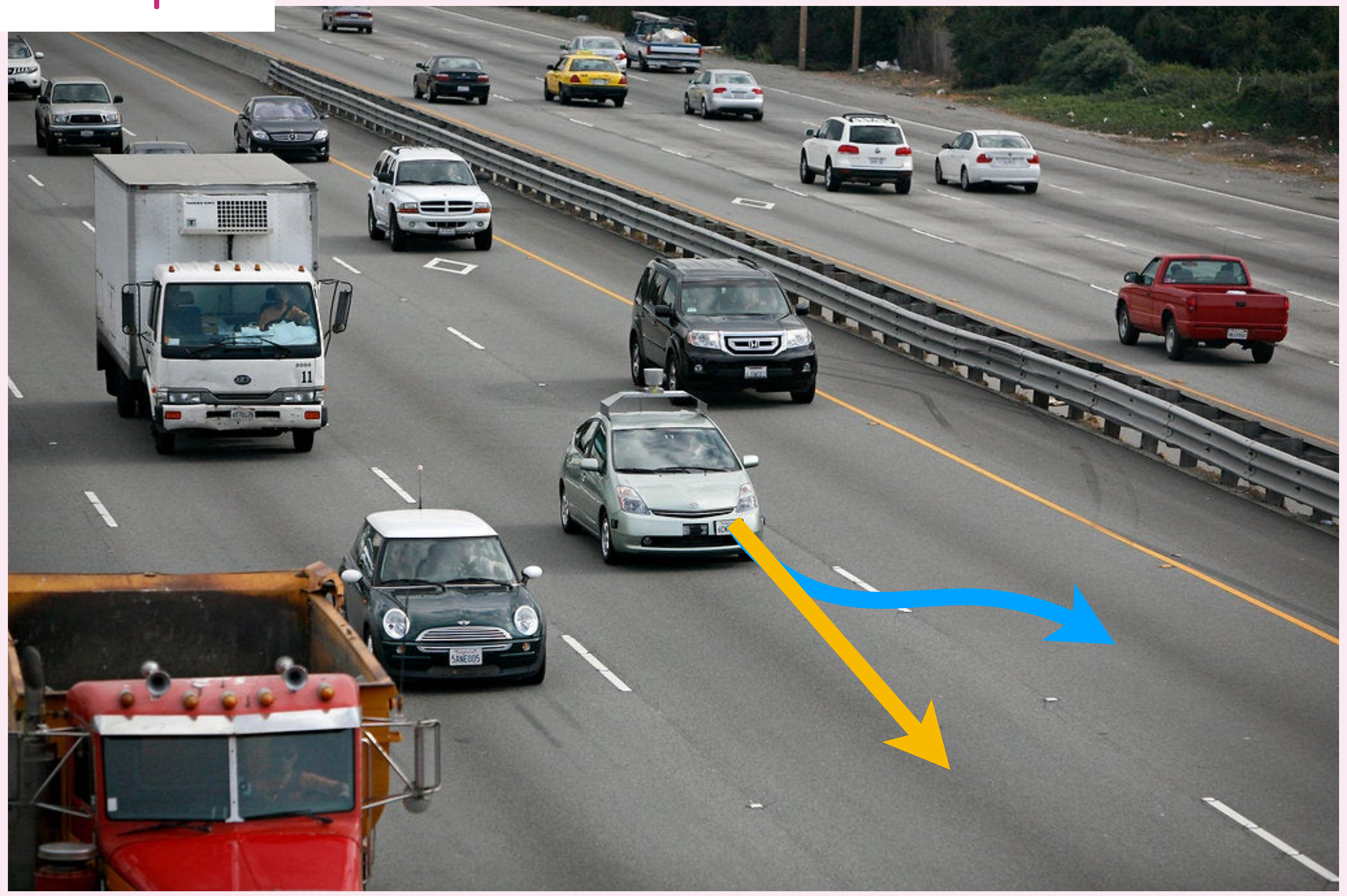
1. For deterministic policy, supervised regression to the expert's actions

$$\min_\theta \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{s},\mathbf{a})\in\mathcal{D}} \|\mathbf{a} - \hat{\mathbf{a}}\|^2 \qquad \text{where } \hat{\mathbf{a}} = \pi_\theta(\mathbf{s})$$

2. Deploy learned policy π_θ

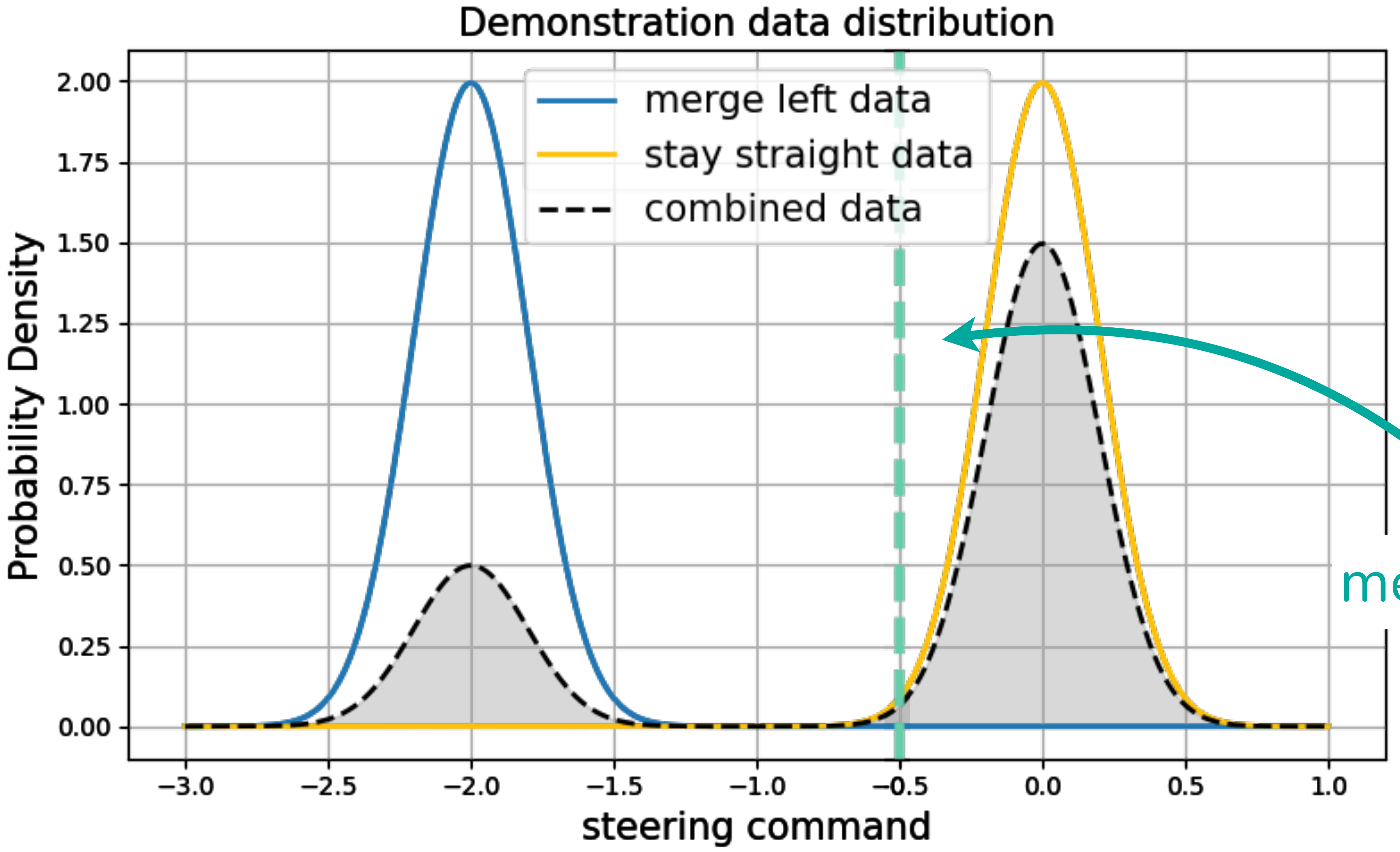
What could go wrong?

Example



- Dataset from human drivers
- Sensor readings + steering commands

Question: what might policy trained with ℓ_2-regression do?



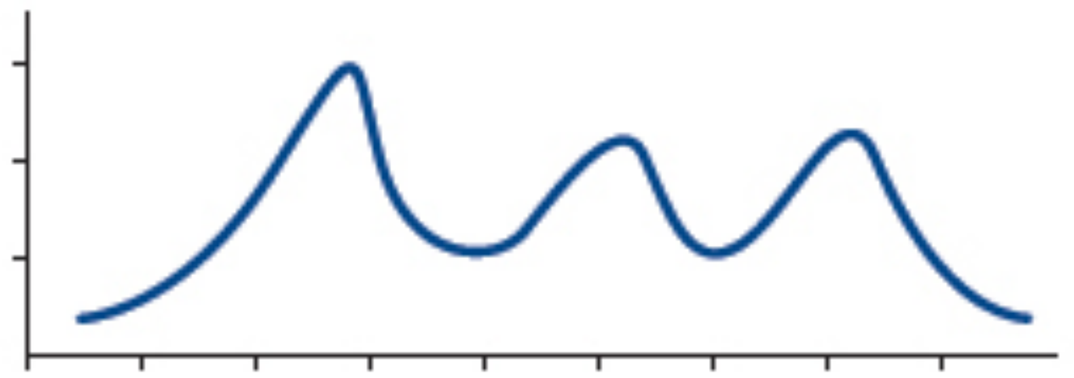
mean of data

How can we represent more than the mean?

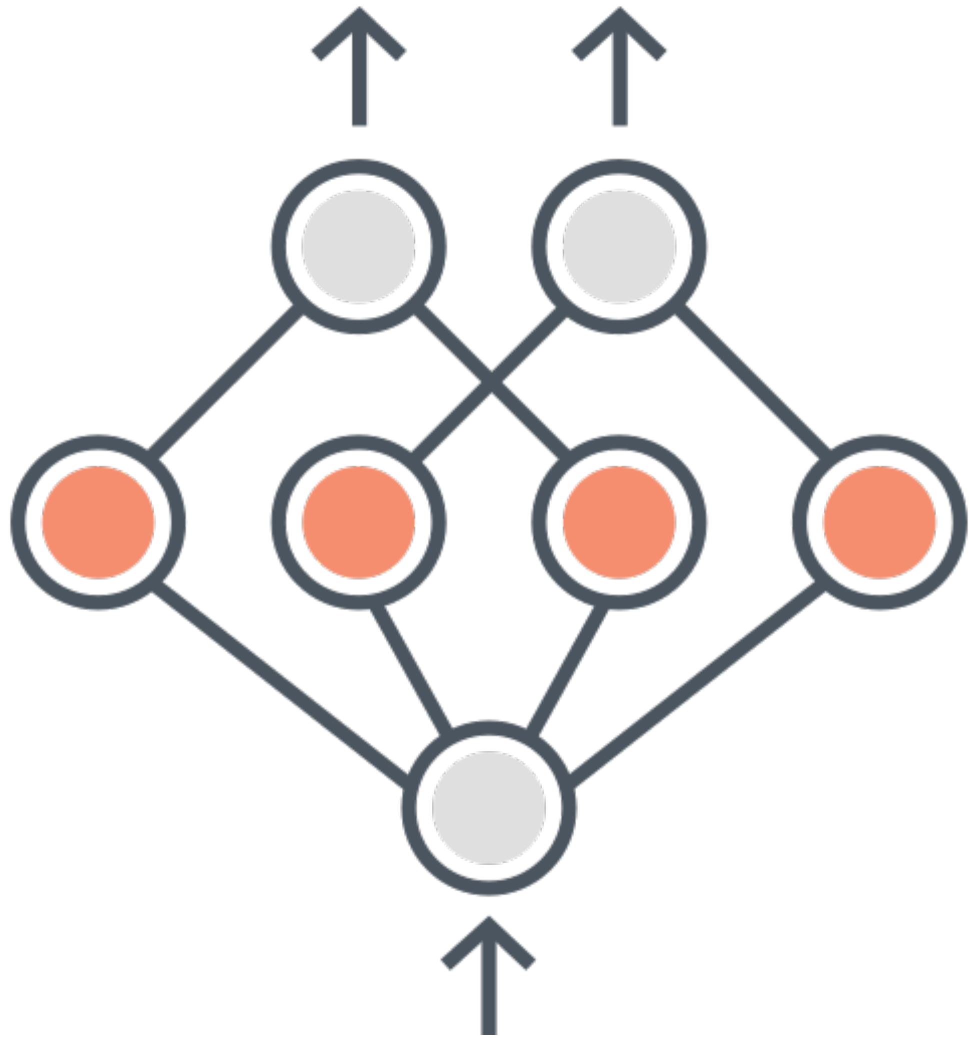
How often does this happen in practice? All the time!

Esp. when data collected by multiple people.

Learning *distributions* with neural networks

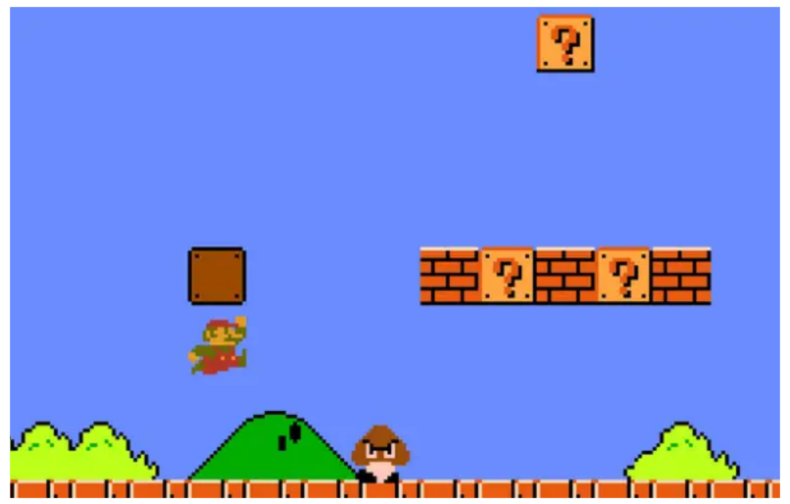


parameters of distribution

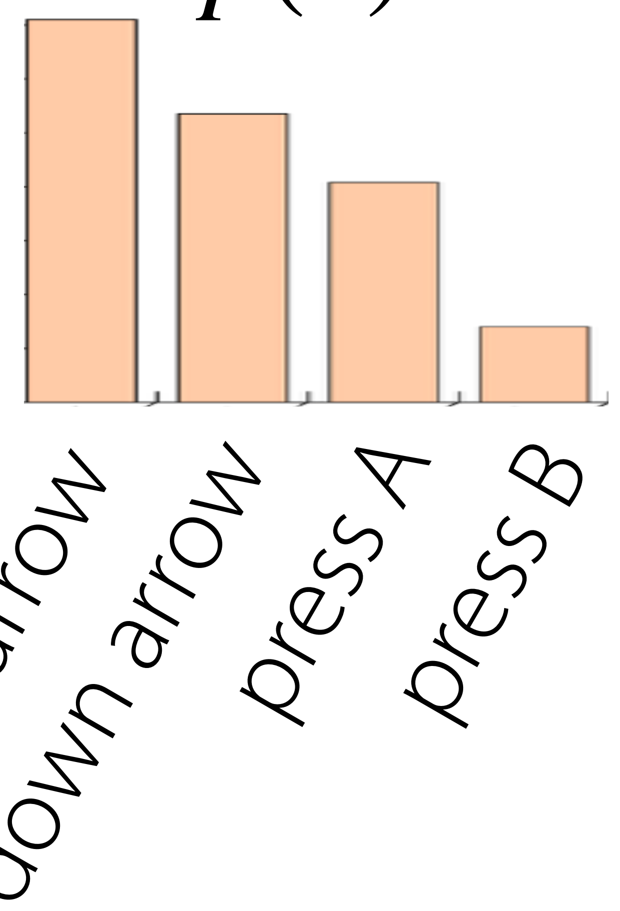


1D discrete actions

$\mathbf{s}$



$p(\mathbf{a})$



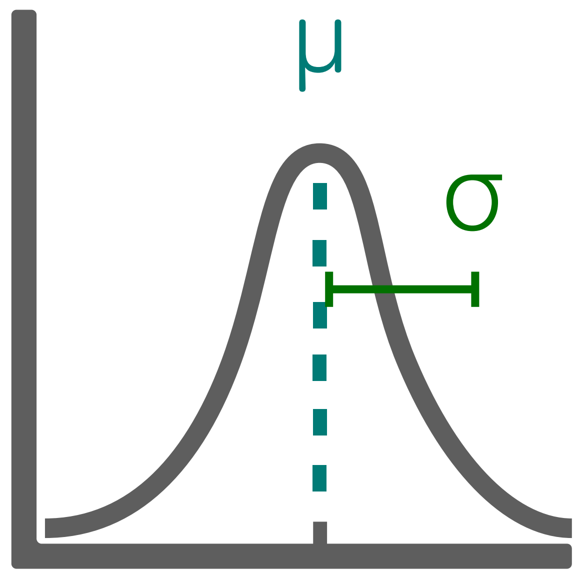
Neural net outputs $p(\text{up}), p(\text{down}), \ldots$ represent categorical distribution.

Maximally expressive

Continuous actions

$\mathbf{s}$

$p(\mathbf{a})$



next steering angle

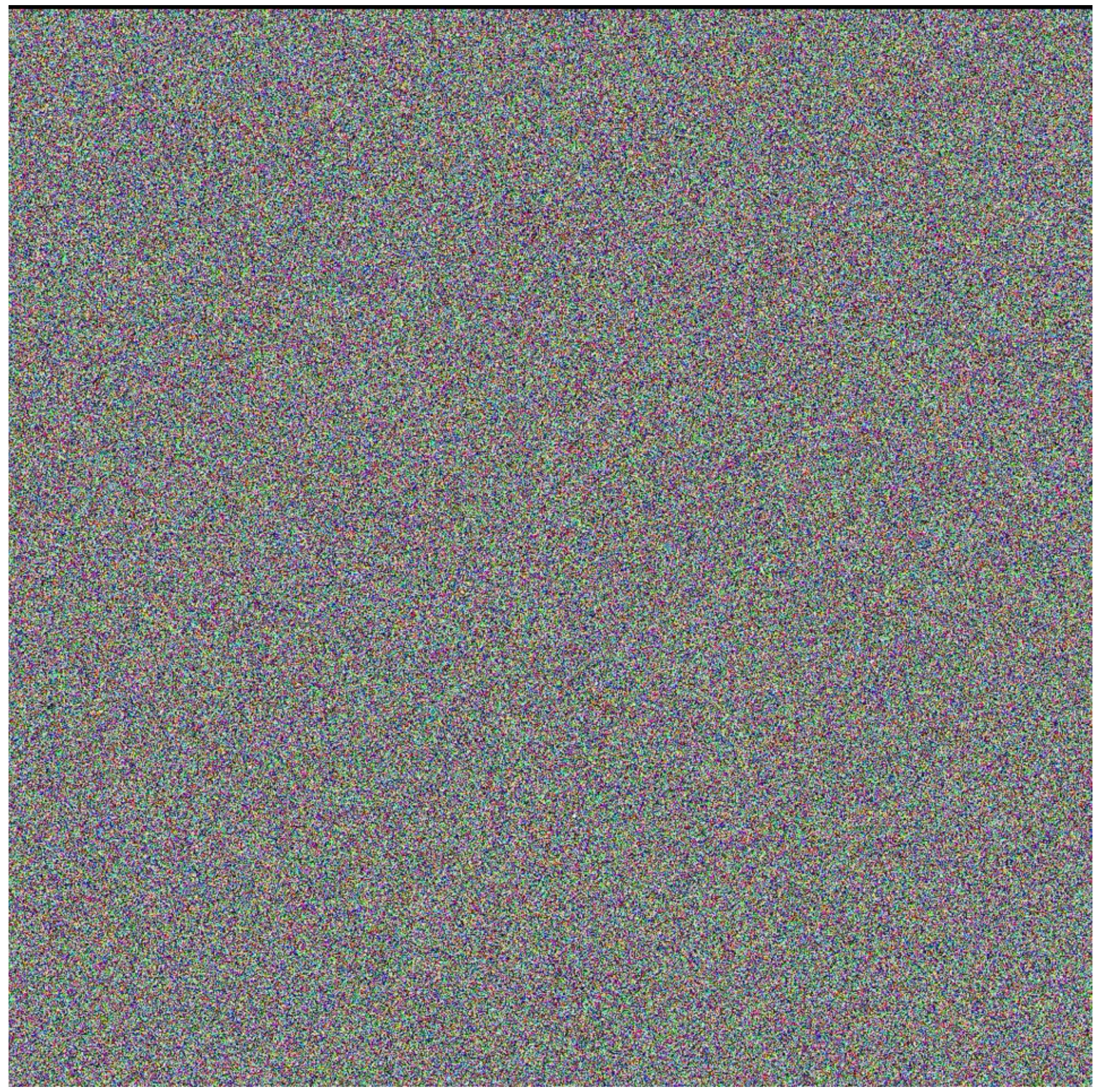
Neural net outputs μ, σ to represent Gaussian distribution.

Not very expressive!

Learning *distributions* with neural networks

💡 Can we use generative modeling?

image diffusion models



learning p(image | text description)

autoregressive models

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

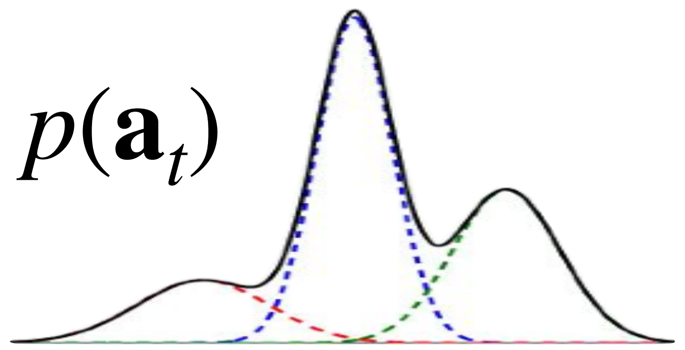
learning p(next word | words so far)

Our goal: learning p(action | observations)

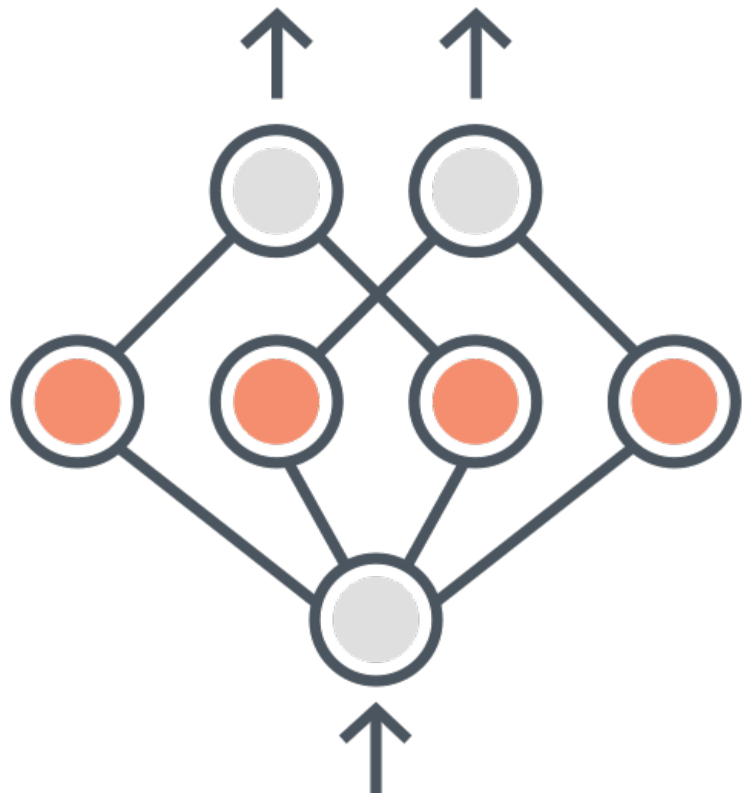
Generative models for policies (approximating $p(\mathbf{a}\,|\,\mathbf{s})$)

Mixture of Gaussians

$p(\mathbf{a}_t)$



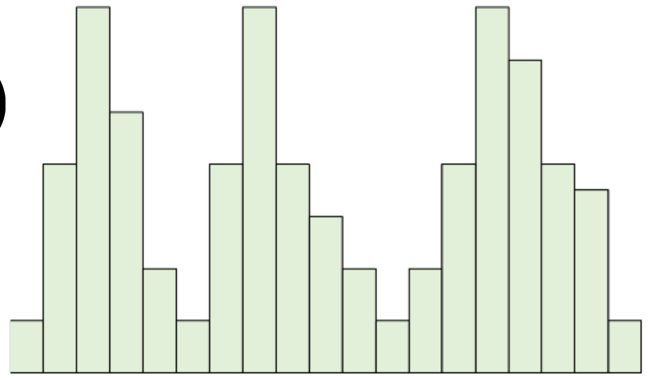
output $\mu_1, \sigma_1, w_1, \mu_2, \sigma_2, w_2, \ldots$



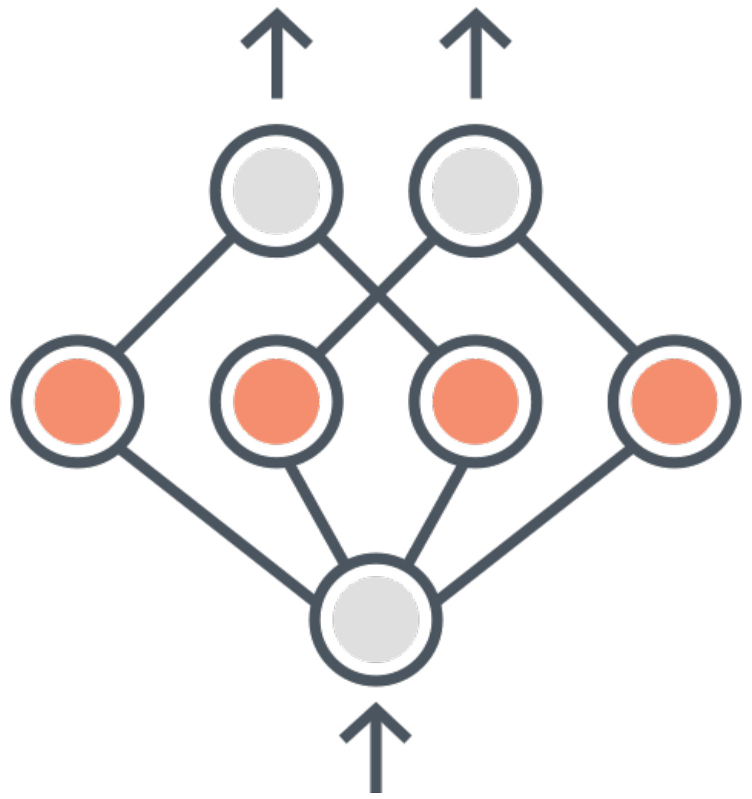
$\mathbf{s}_t$

Discretize + Autoregressive

$p(\mathbf{a}_{t,d})$



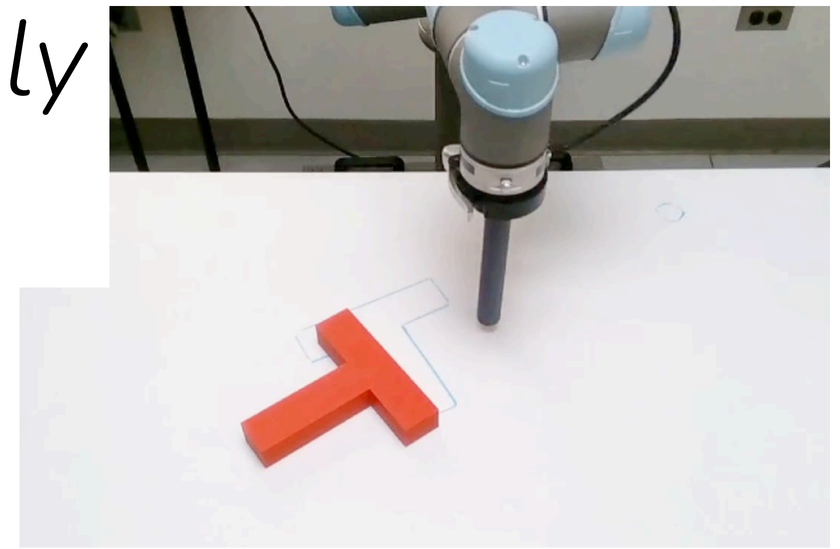
output $p(\mathbf{a}_{t,1}), p(\mathbf{a}_{t,2}\,|\,\hat{\mathbf{a}}_{t,1}), p(\mathbf{a}_{t,3}\,|\,\hat{\mathbf{a}}_{t,1:2}), \ldots$



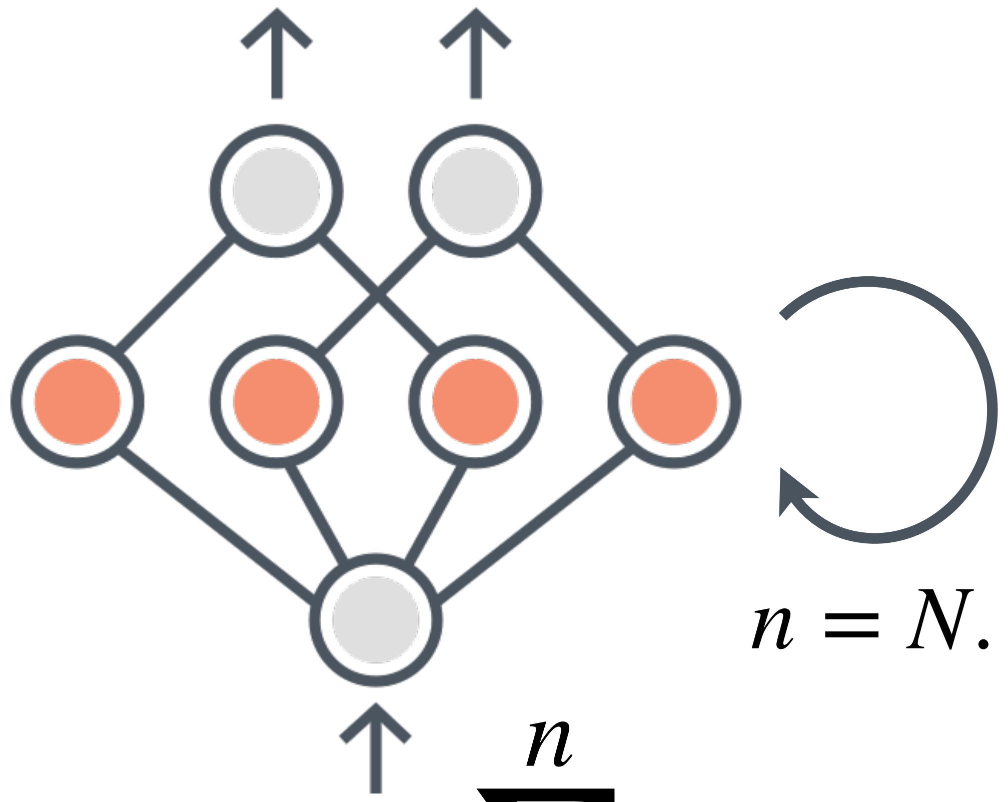
$\mathbf{s}_t$

Diffusion

iteratively denoise



output ϵ_n



$n = N \ldots 1$

$\mathbf{s}_t \quad \mathbf{a}_t + \sum_{i=1}^{n} \epsilon_i$

Question: how are these different from ℓ_2 loss with larger network?

Important Note: Neural network expressivity is often distinct from distribution expressivity!

Imitation learning - version 1

Expressive policies

0. Given demonstrations collected by an expert $\mathscr{D} := \{(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T)\}$

1. Train **generative model** of the expert's actions

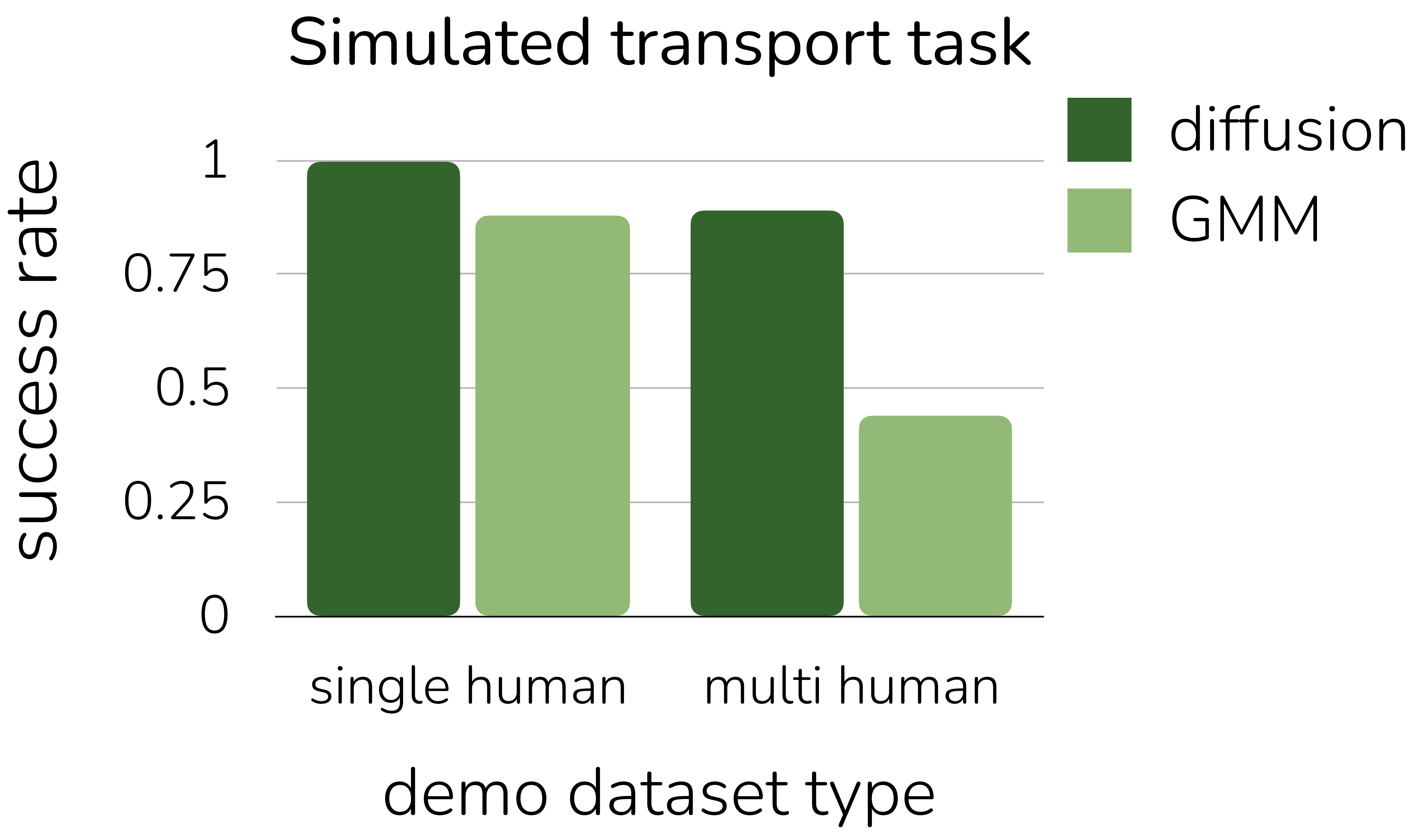
$$\min_{\theta} - \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathscr{D}}[\log \pi_\theta(\mathbf{a} \,|\, \mathbf{s})]$$

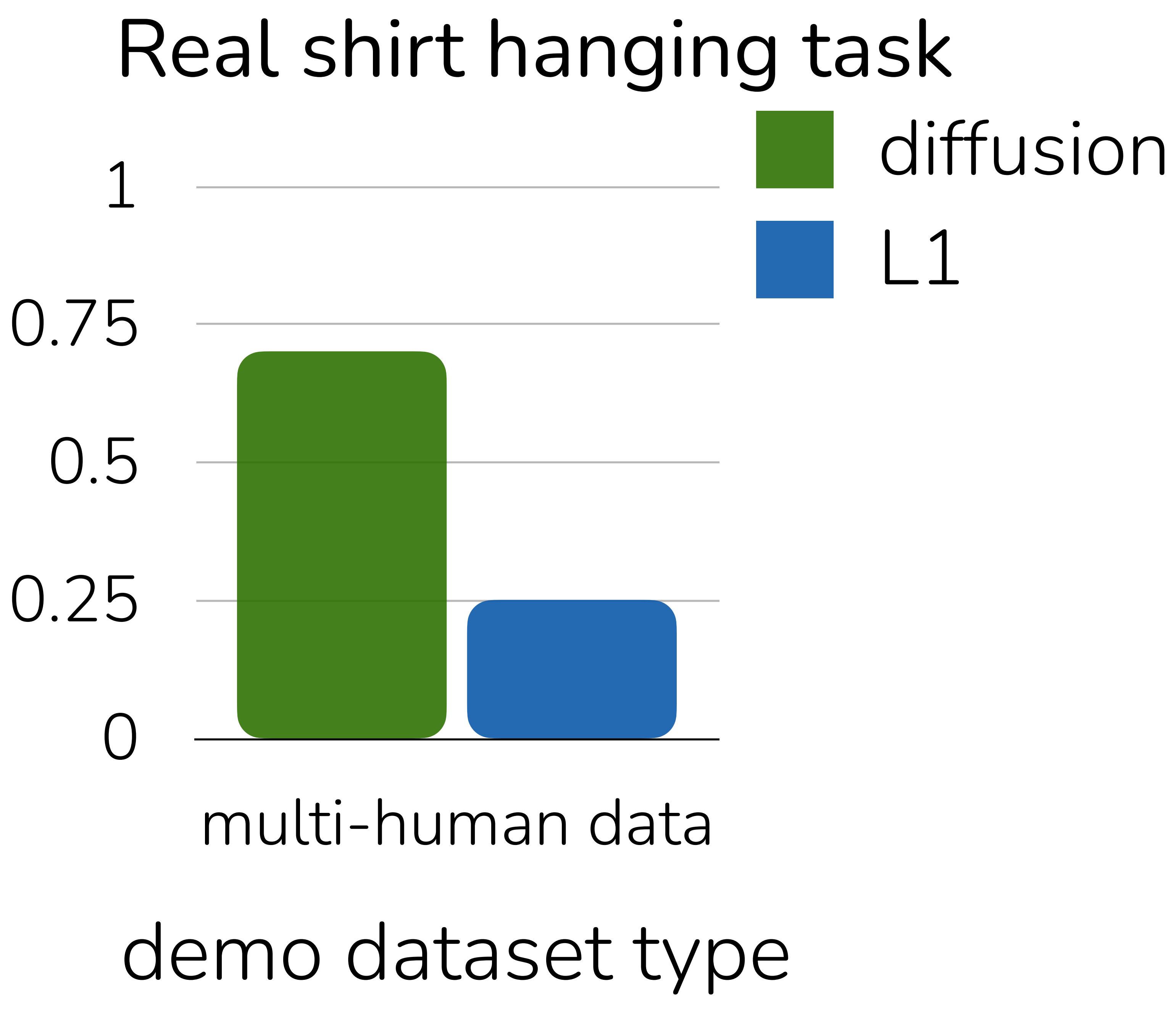
with expressive distribution $\pi(\,\cdot\,|\,\mathbf{s})$

2. Deploy learned policy π_θ

maximize the log probability of the demo actions under the policy

Imitation learning - version 1 vs version 0

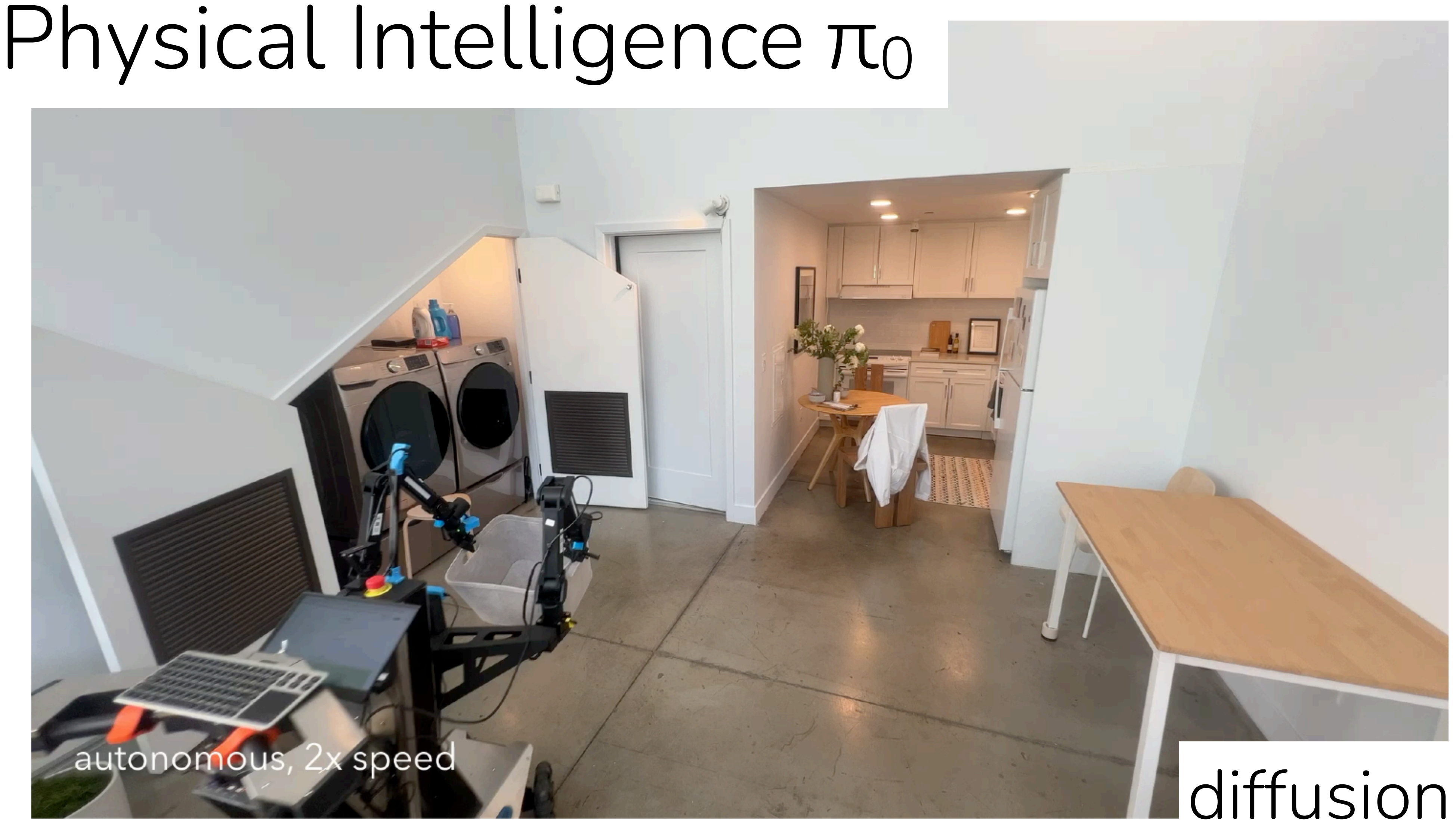




https://diffusion-policy.cs.columbia.edu/
https://aloha-unleashed.github.io/

Robotics: Imitation learning + expressive policies

Physical Intelligence π_0



diffusion

NVIDIA Gr00t N1

diffusion

Figure Helix

diffusion

OpenVLA

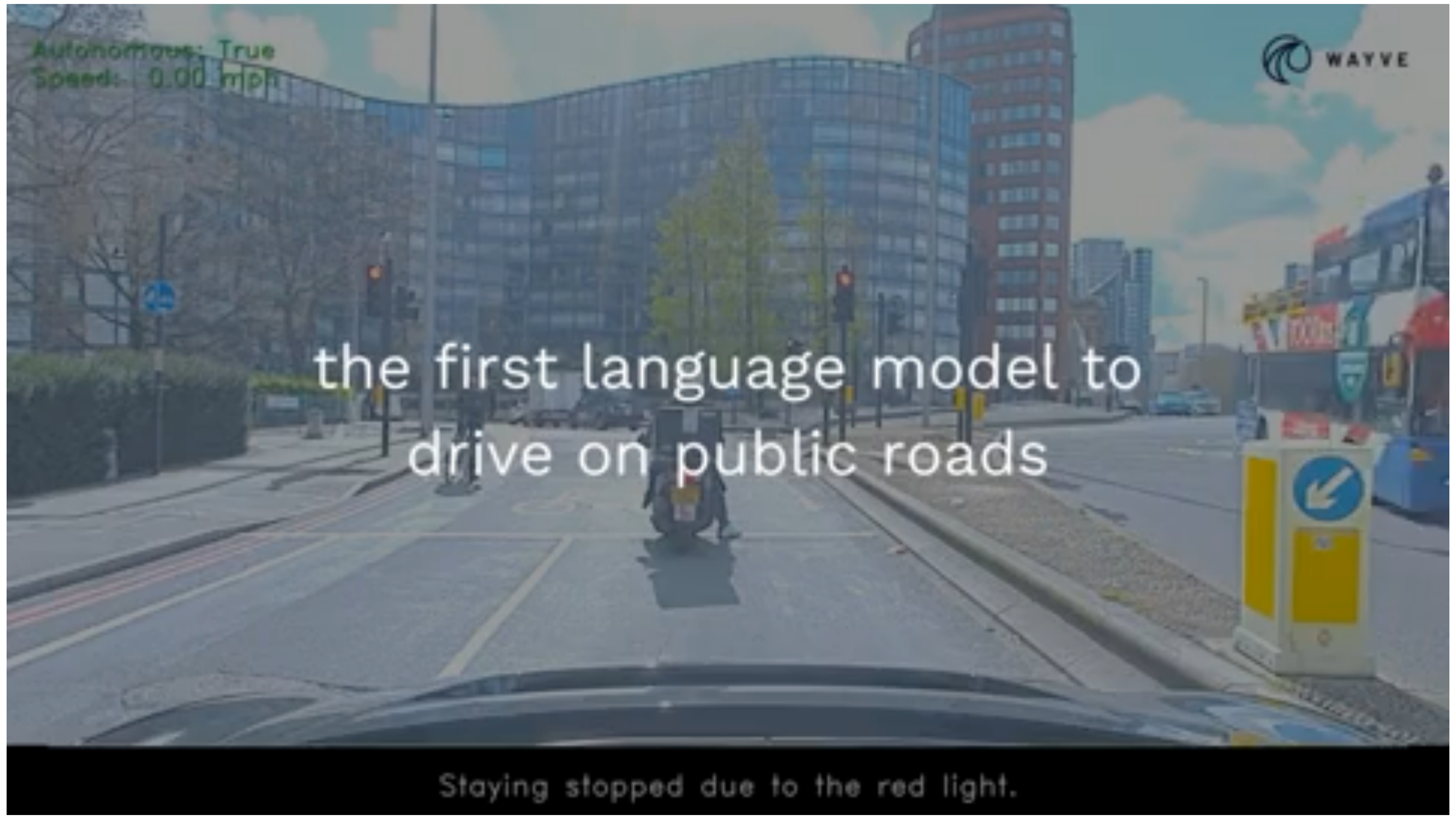
discretize + autoregressive prediction

Autonomous driving: Imitation learning + expressive policies

Waymo EMMA

discretize + autoregressive prediction

Wayve LINGO-2



discretize + autoregressive prediction

Summary so far

Data from one consistent demonstrator

Unimodal policy distribution is enough

Multimodal data, e.g. from multiple demonstrators

Need expressive generative model for policy

Summary so far

- **Key idea**: Train expressive policy class via generative modeling on dataset of demonstrations.
- Algorithm is fully *offline*

Definitions.

offline: using only an existing dataset, no new data from learned policy

online: using new data from learned policy

+ no need for data from policy (online data can be unsafe, expensive to collect)
+ no need to define a reward function
- may need **a lot** of data for reliable performance

The plan for today

Imitation Learning

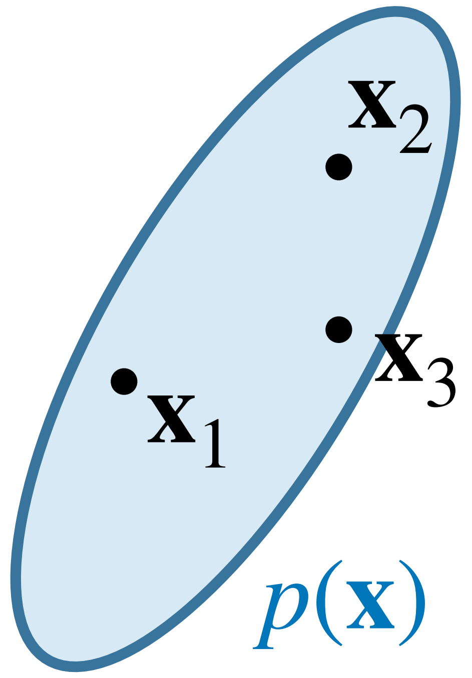
1. Imitation learning basics
2. Learning expressive policy distributions
3. Learning from online interventions
4. Time permitting: how to collect demonstrations

Topic of homework 1!

What can go wrong in imitation learning?

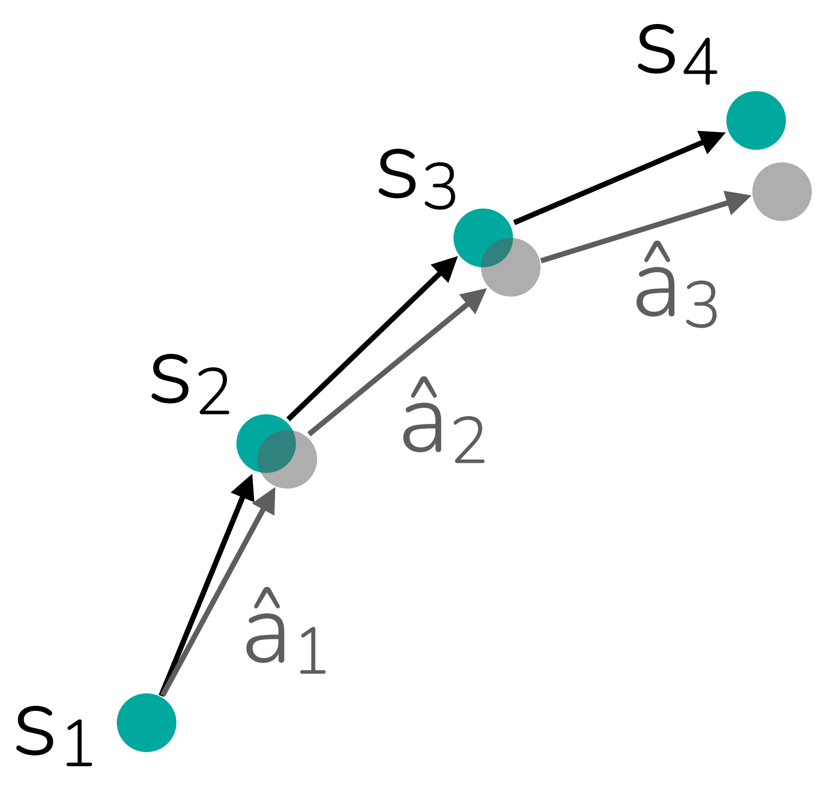
Compounding errors

Supervised learning



Inputs independent of predicted labels $\hat{\mathbf{y}}$

Supervised learning of behavior



Predicted actions affect next state.

Errors can lead to drift away from the data distribution!

Errors can then compound!

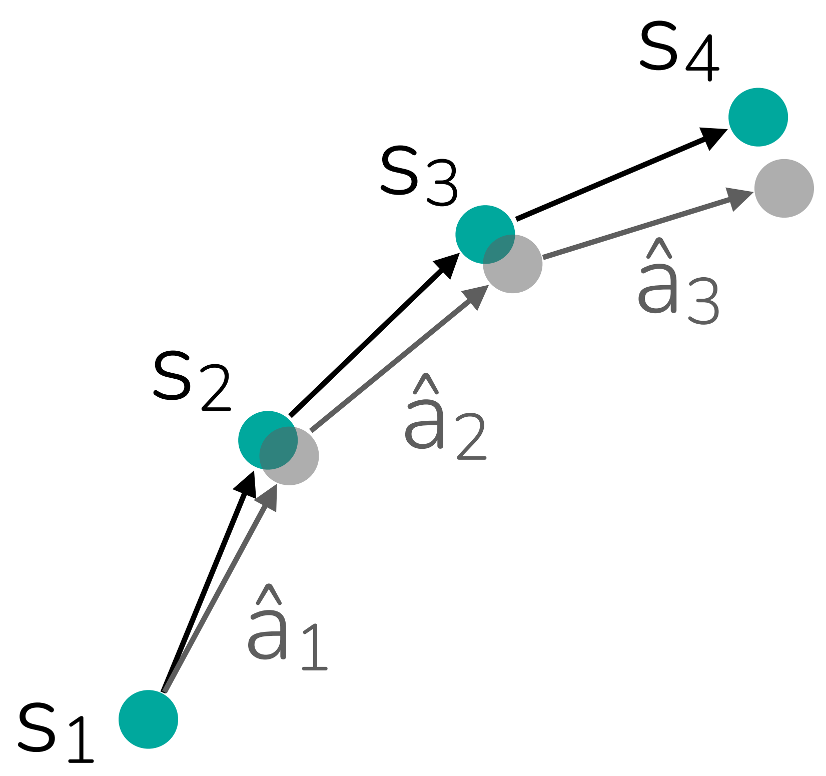
$$\underbrace{p_{expert}(\mathbf{s})}_{\text{states visited by expert}} \neq \underbrace{p_{\pi}(\mathbf{s})}_{\text{states visited by learned policy } \pi}$$

"covariate shift"

What can go wrong in imitation learning?

Compounding errors

Supervised learning of behavior



Predicted actions affect next state.

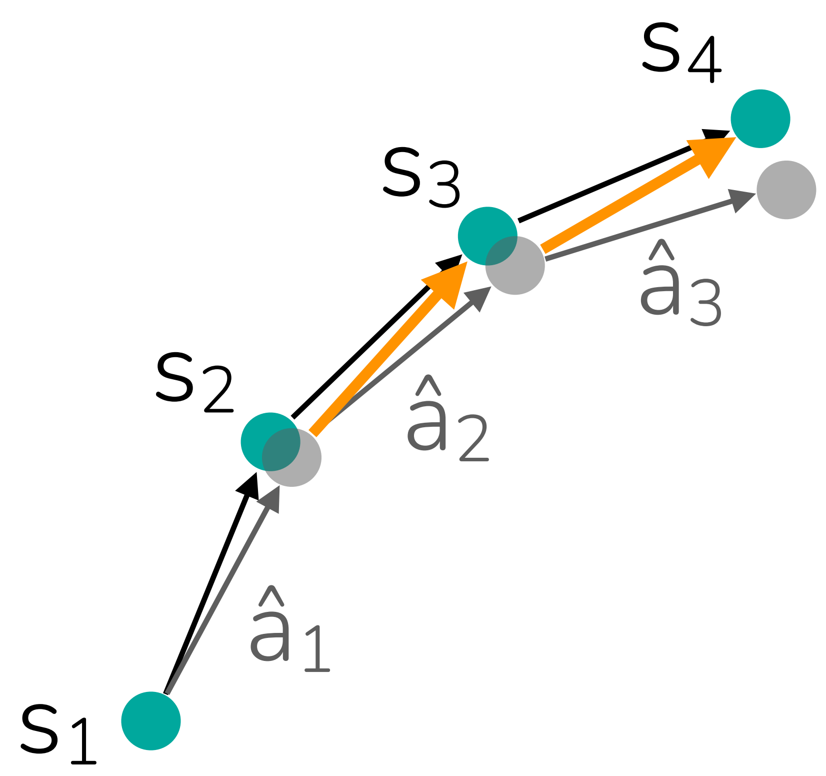
Errors can lead to drift away from the data distribution!

Errors can then compound!

$$p_{expert}(\mathbf{s}) \neq p_{\pi}(\mathbf{s})$$

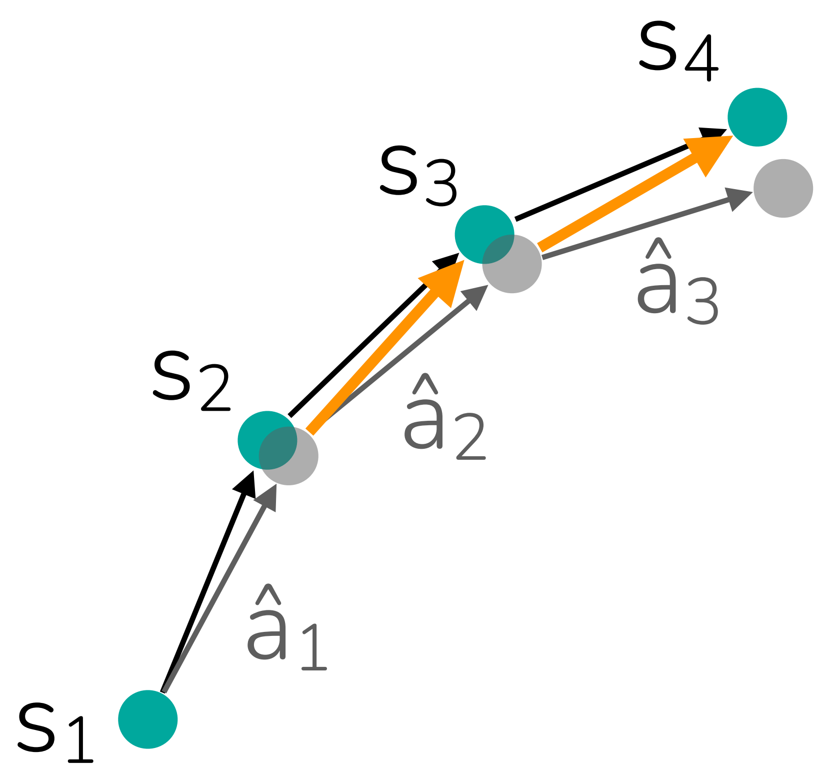
Solutions?

1. Collect A LOT of demo data & hope for the best.
2. Collect corrective behavior data



Addressing Compounding Errors with Interventions

Collect corrective behavior data

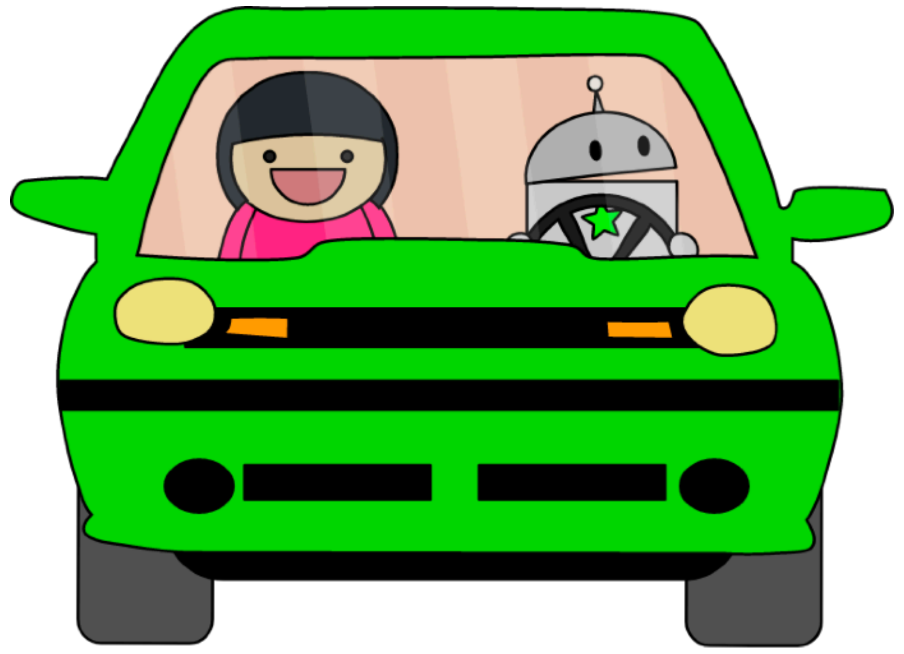


1. Roll out learned policy π_θ: $\mathbf{s}'_1, \hat{\mathbf{a}}_1, \ldots, \mathbf{s}'_T$
2. Query expert action at visited states $\mathbf{a}^* \sim \pi_{expert}(\cdot \mid \mathbf{s}')$
3. Aggregate corrections with existing data $\mathscr{D} \leftarrow \mathscr{D} \cup \{(\mathbf{s}', \mathbf{a}^*)\}$
4. Update policy $\min_\theta \mathscr{L}(\pi_\theta, \mathscr{D})$

"dataset aggregation" (DAgger)

+ data-efficient way to learn from an expert

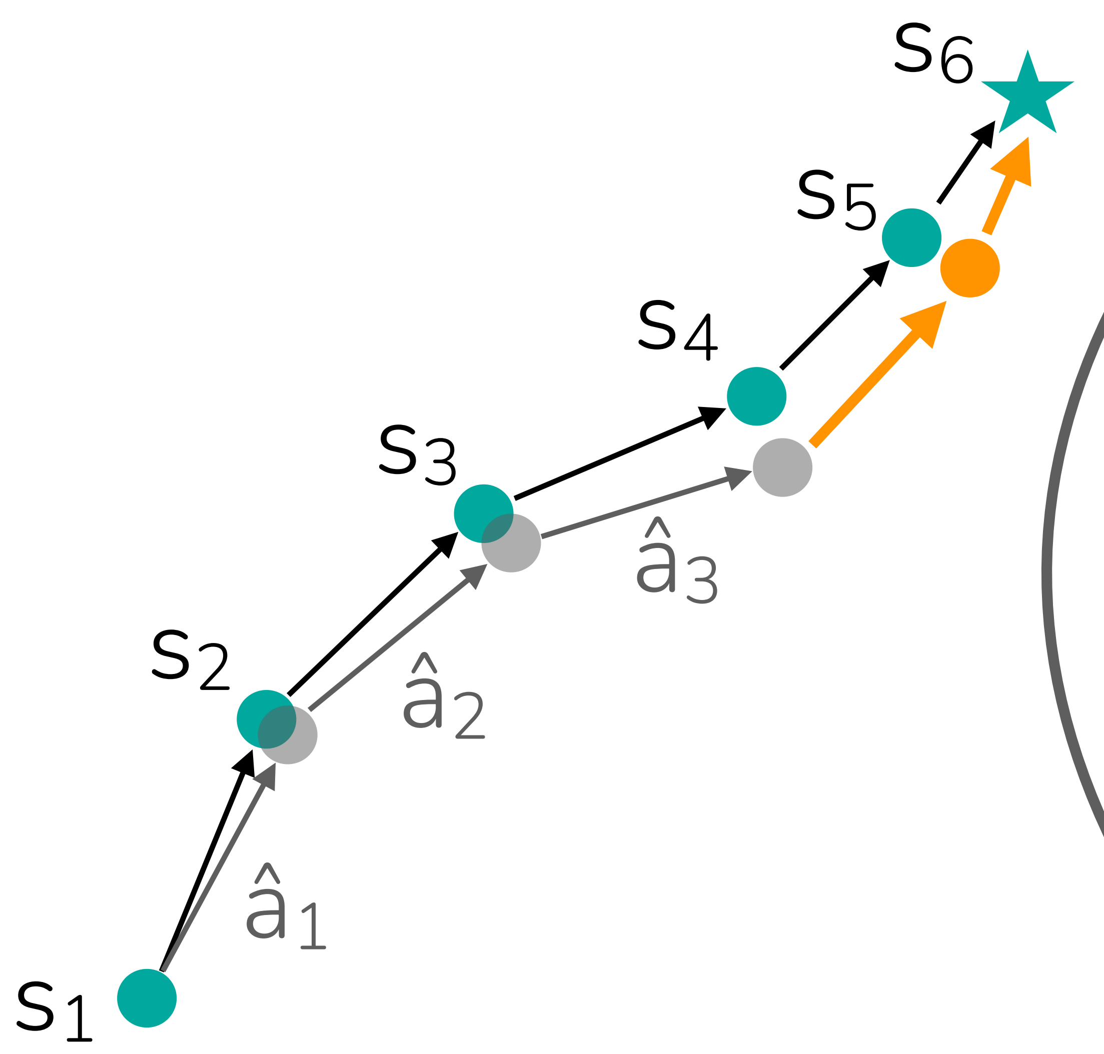
- can be challenging to query expert when agent has control



Is there another way to collect corrective data?

Addressing Compounding Errors with Interventions

Collect corrective behavior data while *taking full control*



1. Start to roll-out learned policy π_θ: $\mathbf{s}'_1, \hat{\mathbf{a}}_1, \ldots, \mathbf{s}'_t$
2. Expert intervenes at time t when policy makes mistake
3. Expert provides (partial) demonstration $\mathbf{s}'_t, \mathbf{a}^*_t, \ldots, \mathbf{s}'_T$
4. Aggregate new demos with existing data $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}'_i, \mathbf{a}^*_i)\}; i \geq t$
5. Update policy $\min_\theta \mathcal{L}(\pi_\theta, \mathcal{D})$

"human gated DAgger"

\+ (much) more practical interface for providing corrections

\- can be hard to catch mistakes quickly in some application domains

Question: could you automatically detect when intervention is needed?

The plan for today

Imitation Learning

1. Imitation learning basics
2. Learning expressive policy distributions
3. Learning from online interventions
4. Time permitting: how to collect demonstrations

Topic of homework 1!

How to collect demonstrations?

In some domains: People already collect demonstrations that can be recorded

e.g. driving cars, writing text messages

What about robotics?

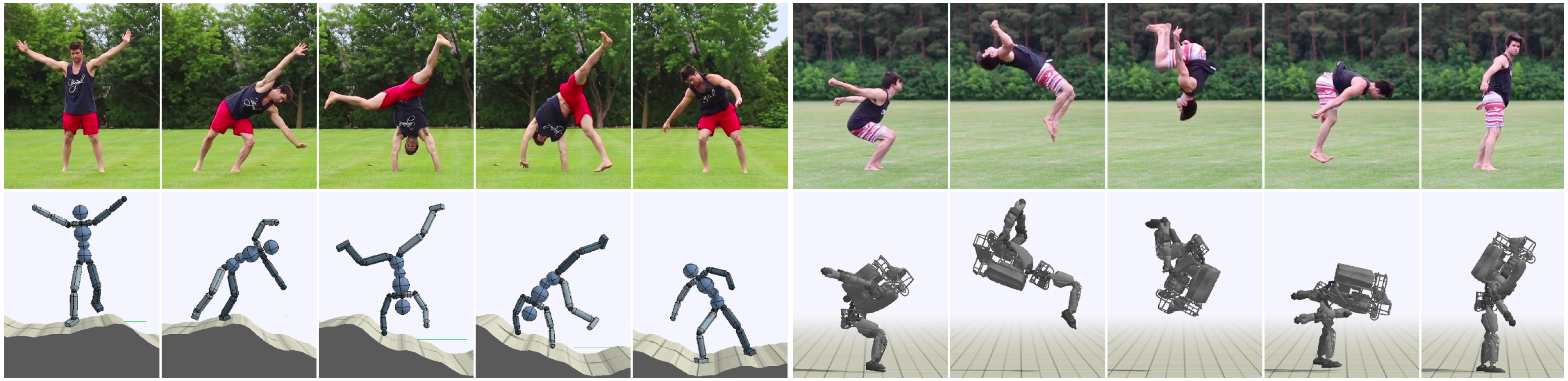
Kinesthetic teaching	Remote controllers	Puppeteering
		
+ easy interface - human visible in scene	~ interface ease varies, can have high latency	+ easy interface - requires double hardware

In other domains: It may not be viable to collect demos! (e.g. quadruped robot)

Can robots directly use videos of people, animals?

Embodiment gap:
- difference in appearance
- difference in physical capabilities, degrees of freedom

Hard to directly imitate human & animal data, but can potentially guide exploration.



Peng, Kanazawa, Malik, Abbeel, Levine. SFV: Reinforcement Learning of Physical Skills from Videos. SIGGRAPH Asia 2018.

Summary

Part 1: Train policy to mimic offline demonstrations

- best if policy is an expressive generative model over actions
- algorithm is fully *offline*

often referred to as **behavior cloning** (**BC**)

Part 2: Improve policy using online interventions

- requires interface for human or expert intervention
- algorithm involves running policy *online*

often referred to as **DAgger**, or **HG-DAgger**

\+ offline BC: simple, no need for data from policy (online data can be unsafe, expensive)

\+ DAgger: possible path to reliable performance, more data-efficient than offline BC

\+ no need to define a reward function

\- may need **impractically large amount** of data for reliable performance

\- doesn't provide a framework for improving on own (from "practicing")

Many successful methods **combine** imitation learning and reinforcement learning!

The plan for today

Imitation Learning

1. Imitation learning basics
2. Learning expressive policy distributions
3. Learning from online interventions
4. Time permitting: how to collect demonstrations

Topic of homework 1!

Key learning goals:

- how to represent distributions with neural networks
- why expressive distributions matter for imitation learning
- what are compounding errors and how to address them

Next time

Start of *reinforcement learning algorithms*

Course reminders

- Start forming **final project groups** (survey due Weds April 16)
- **Homework 1** out today, due Fri April 18
- PyTorch tutorial today at 1:30 pm in **Gates B1**