



Autonomous Robotics

Winter 2026

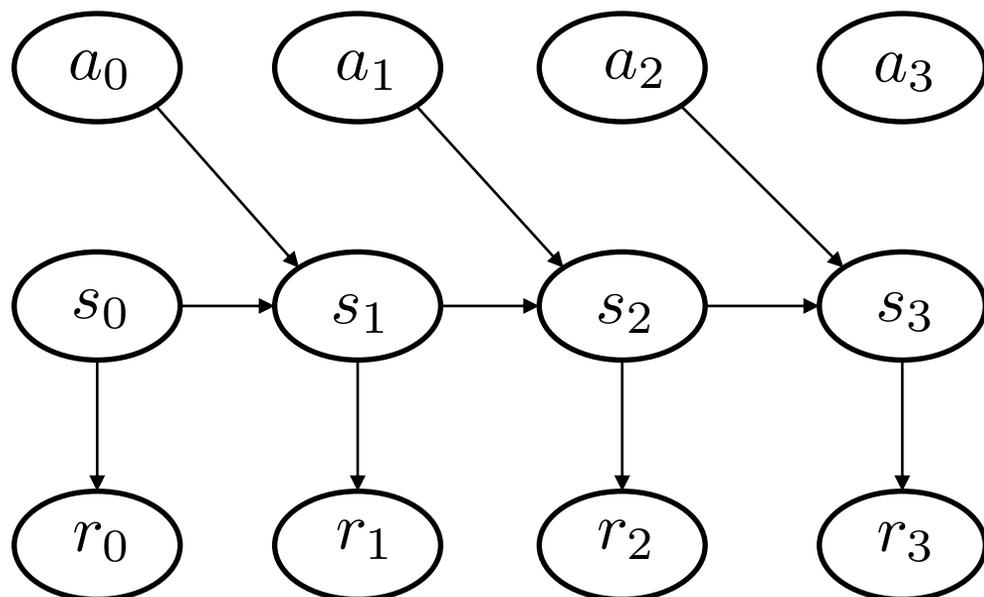
Abhishek Gupta, Siddhartha Srinivasa

TAs: Carolina Higuera, Entong Su, Rishabh Jain



Recap

Framework for Sequential Decision Making - Markov Decision Process



States: \mathcal{S}

Actions: \mathcal{A}

Rewards: \mathcal{R}

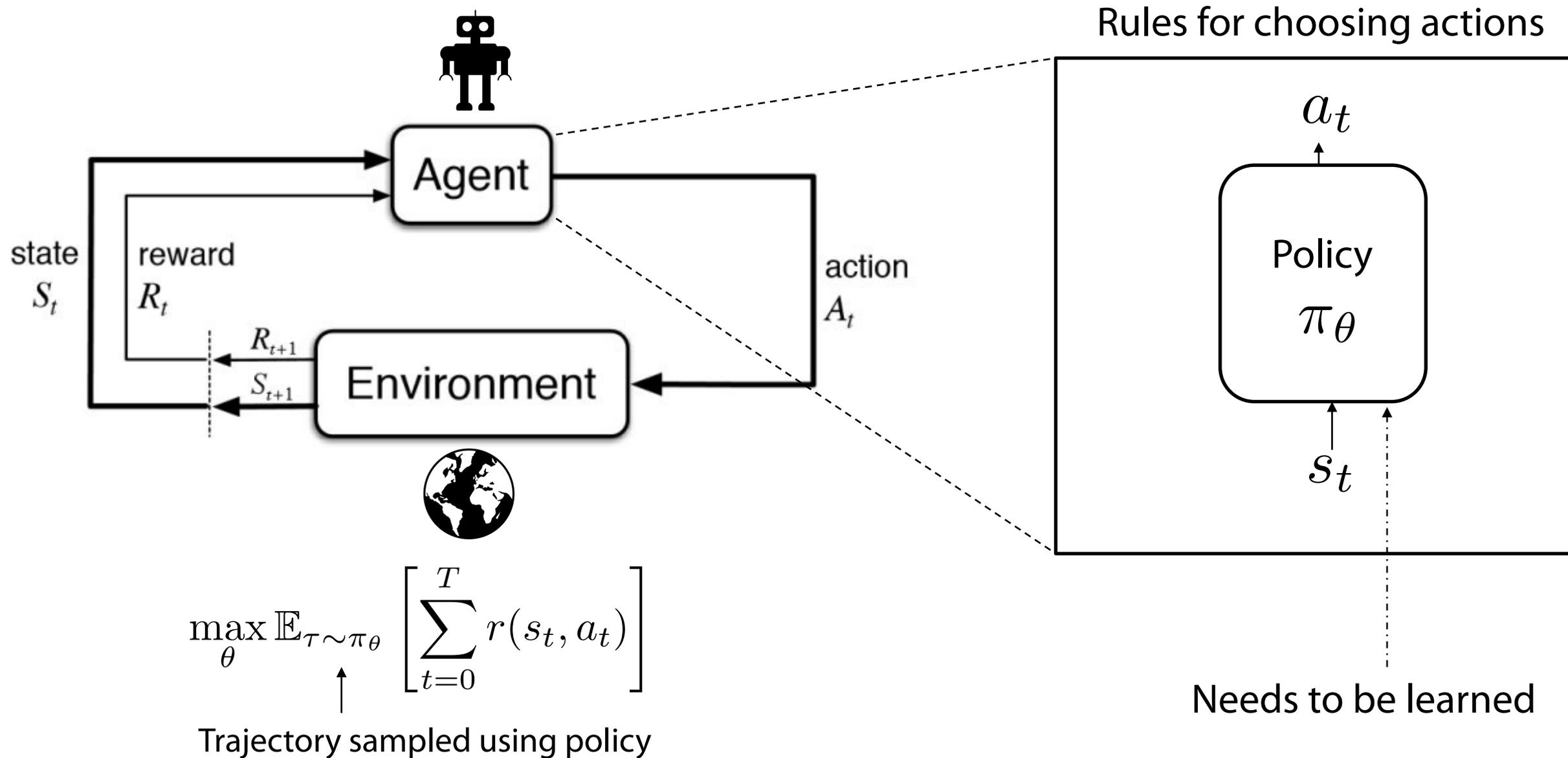
Transition Dynamics - $p(s_{t+1}|s_t, a_t)$

Markov property $p(s_1, s_2, s_3) = p(s_3|s_2)p(s_2|s_1)p(s_1)$

Trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_T, a_T, r_T)$

Key: MDPs obey the Markov property
Past is independent of the future conditioned on the present

Reinforcement Learning Formalism



Lecture Outline

A Formalism for Sequential Decision Making



Imitation Learning: Behavior Cloning



Imitation Learning: Improvements – Compounding Error

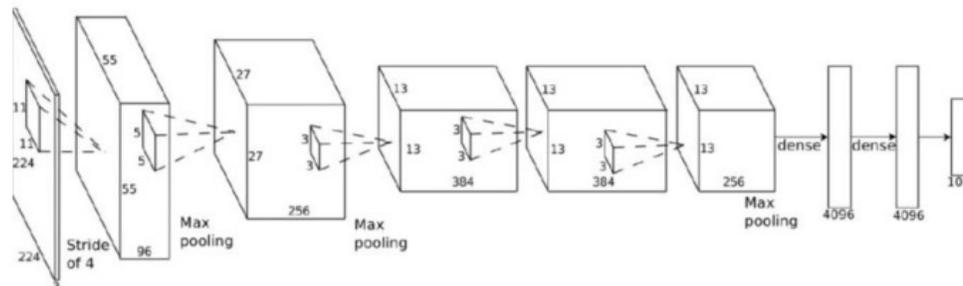


Imitation Learning: Improvements – Multimodality

Imitation Learning: Intuition

Given: Demonstrations of optimal behavior

Goal: Train a policy to mimic the demonstrator



Pros: No rewards, online experience needed (?)

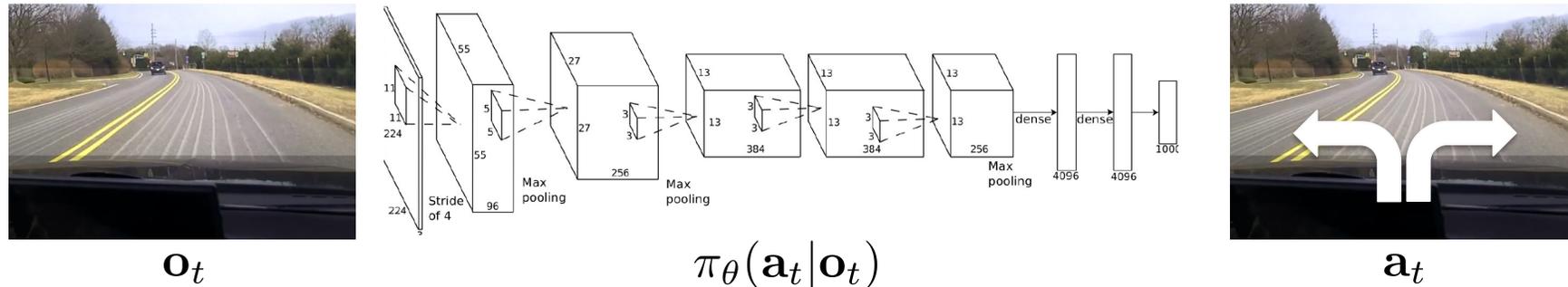
Idea 1: Imitation Learning via Behavior Cloning

Given: Demonstrations of optimal behavior

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

Goal: Train a policy to mimic the demonstrator

Idea: Treat imitation learning as a supervised learning problem!



Idea 1: Imitation Learning via Behavior Cloning

Given: Demonstrations of optimal behavior

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

Goal: Train a policy to mimic the demonstrator

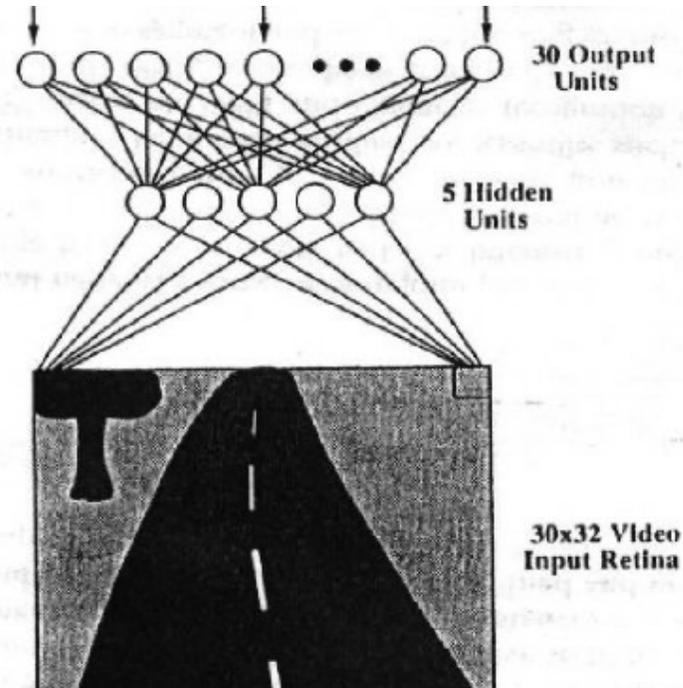
Discrete vs continuous

```
if isinstance(env.action_space, gym.spaces.Box):
    criterion = nn.MSELoss()
else:
    criterion = nn.CrossEntropyLoss()
# Extract initial policy
model = student.policy.to(device)
def train(model, device, train_loader, optimizer):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        if isinstance(env.action_space, gym.spaces.Box):
            if isinstance(student, (A2C, PPO)):
                action, _, _ = model(data)
            else:
                action = model(data)
            action_prediction = action.double()
        else:
            dist = model.get_distribution(data)
            action_prediction = dist.distribution.logits
            target = target.long()
        loss = criterion(action_prediction, target)
        loss.backward()
        optimizer.step()
```

Maximum likelihood

The original deep imitation learning system

ALVINN: Autonomous Land Vehicle In a Neural Network
1989

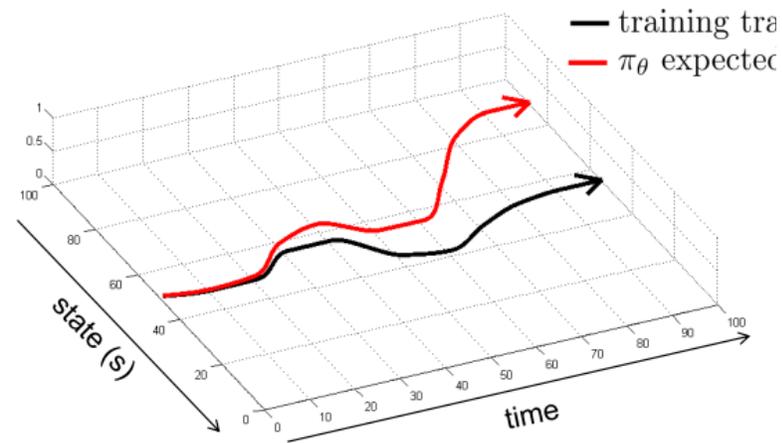


Where we are in 2025?



So does behavior cloning really work?

- Imitation Learning \neq Supervised Learning



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

$$\mathbb{E}_{(s, a) \sim \rho(\pi)} [\mathbf{1}(a = a^*)]$$



Not the same!

So does behavior cloning really work?

- Fails in practice as well!



So is all hope lost?

Lecture Outline

A Formalism for Sequential Decision Making



Imitation Learning: Behavior Cloning

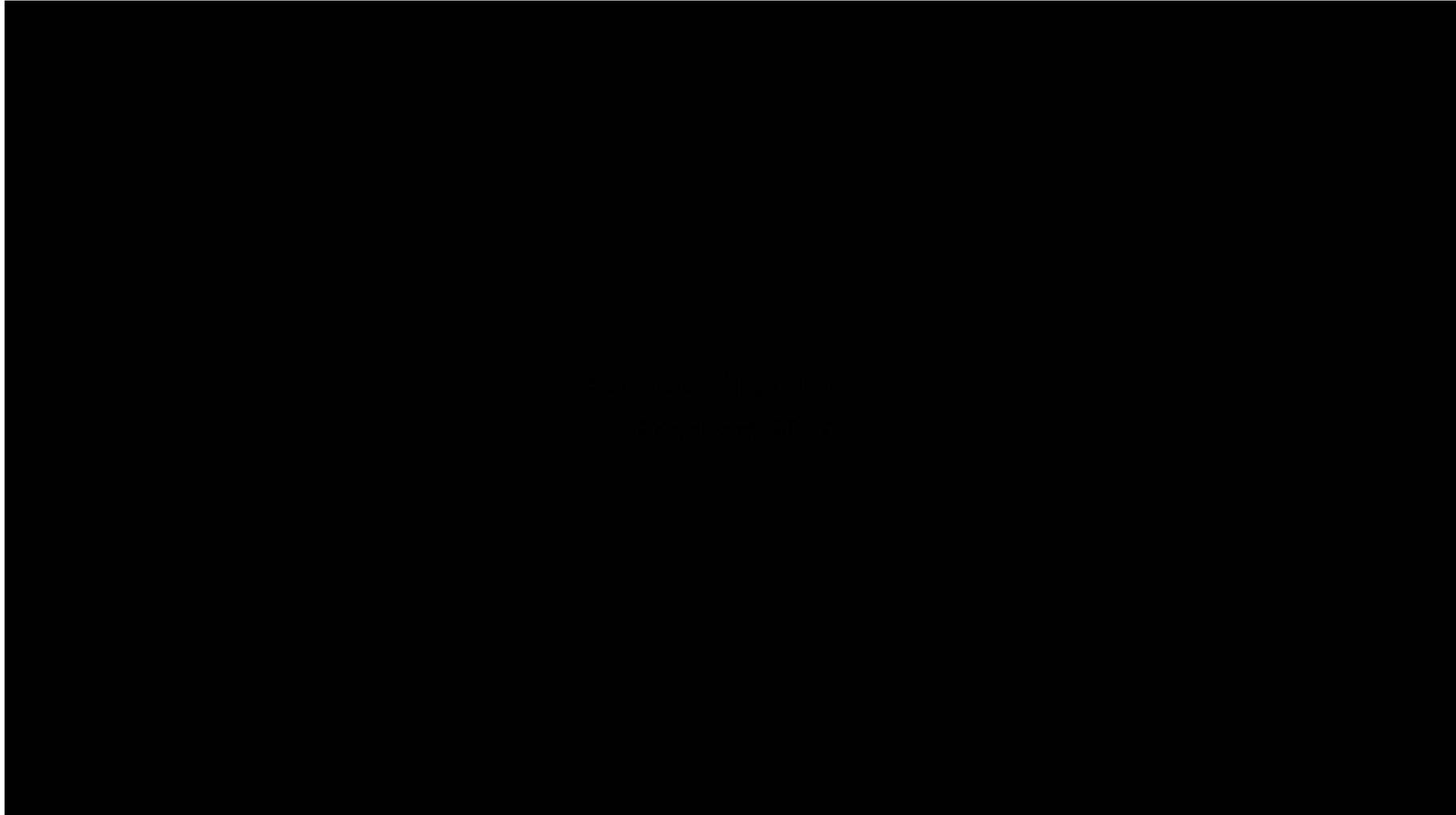


Imitation Learning: Improvements – Compounding Error

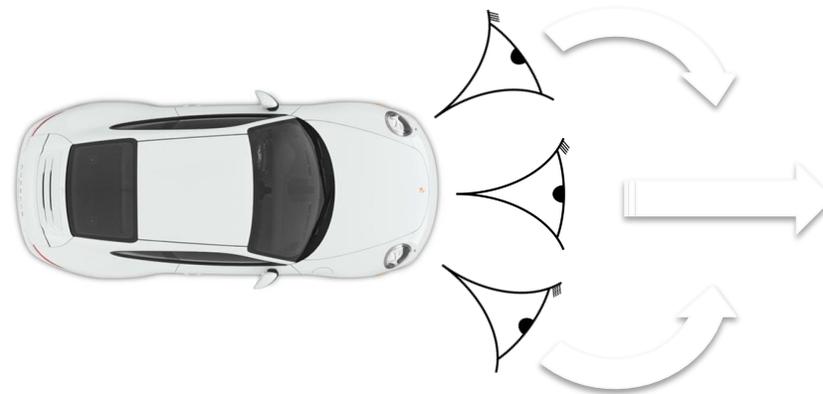
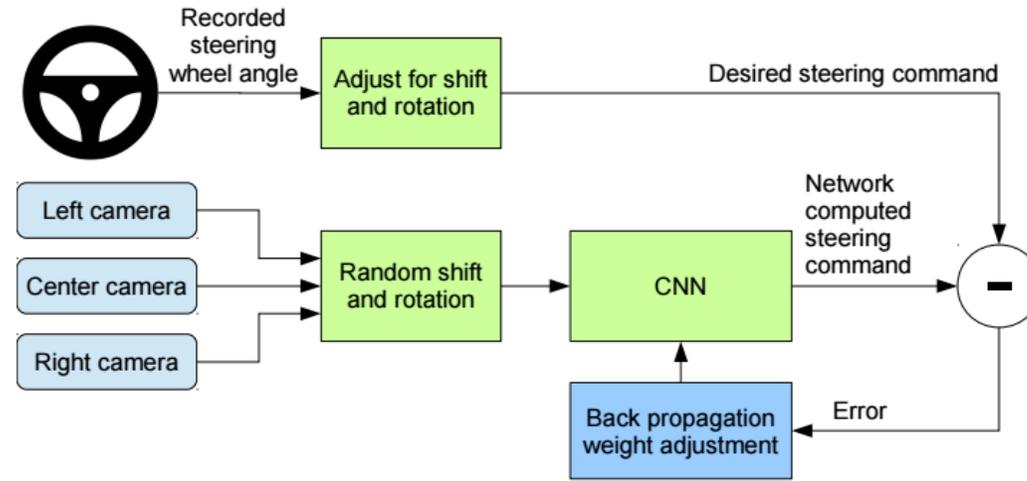


Imitation Learning: Improvements – Multimodality

Can it work in special cases?

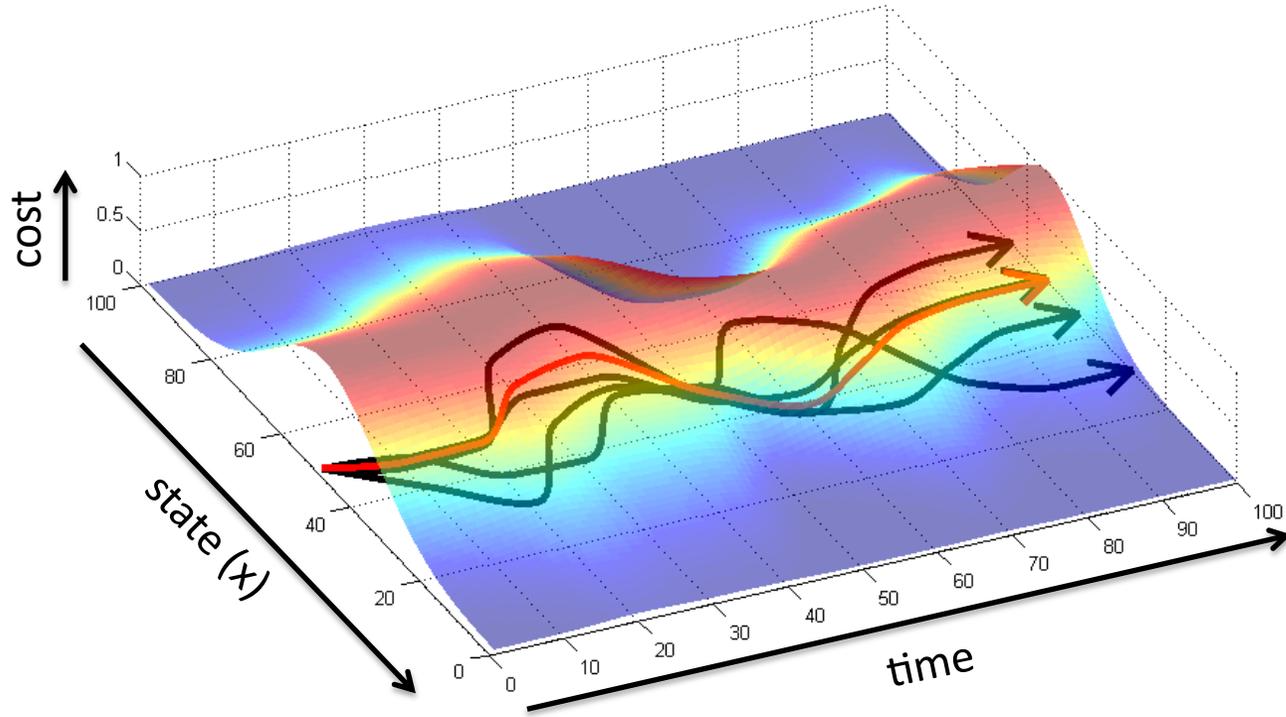


Why did that work?



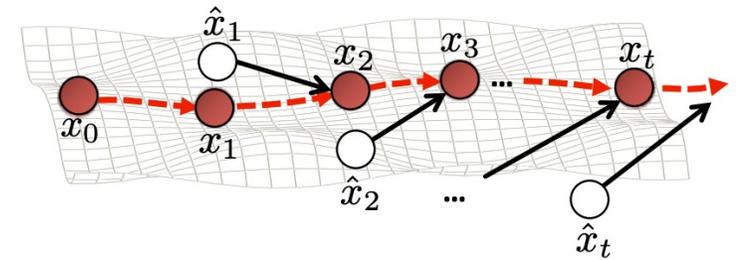
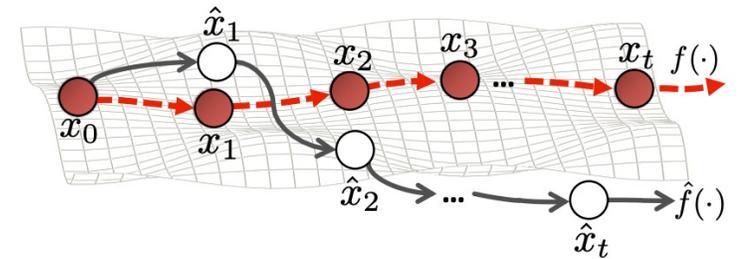
What is the general principle?

- training trajectory
- π_θ expected trajectory

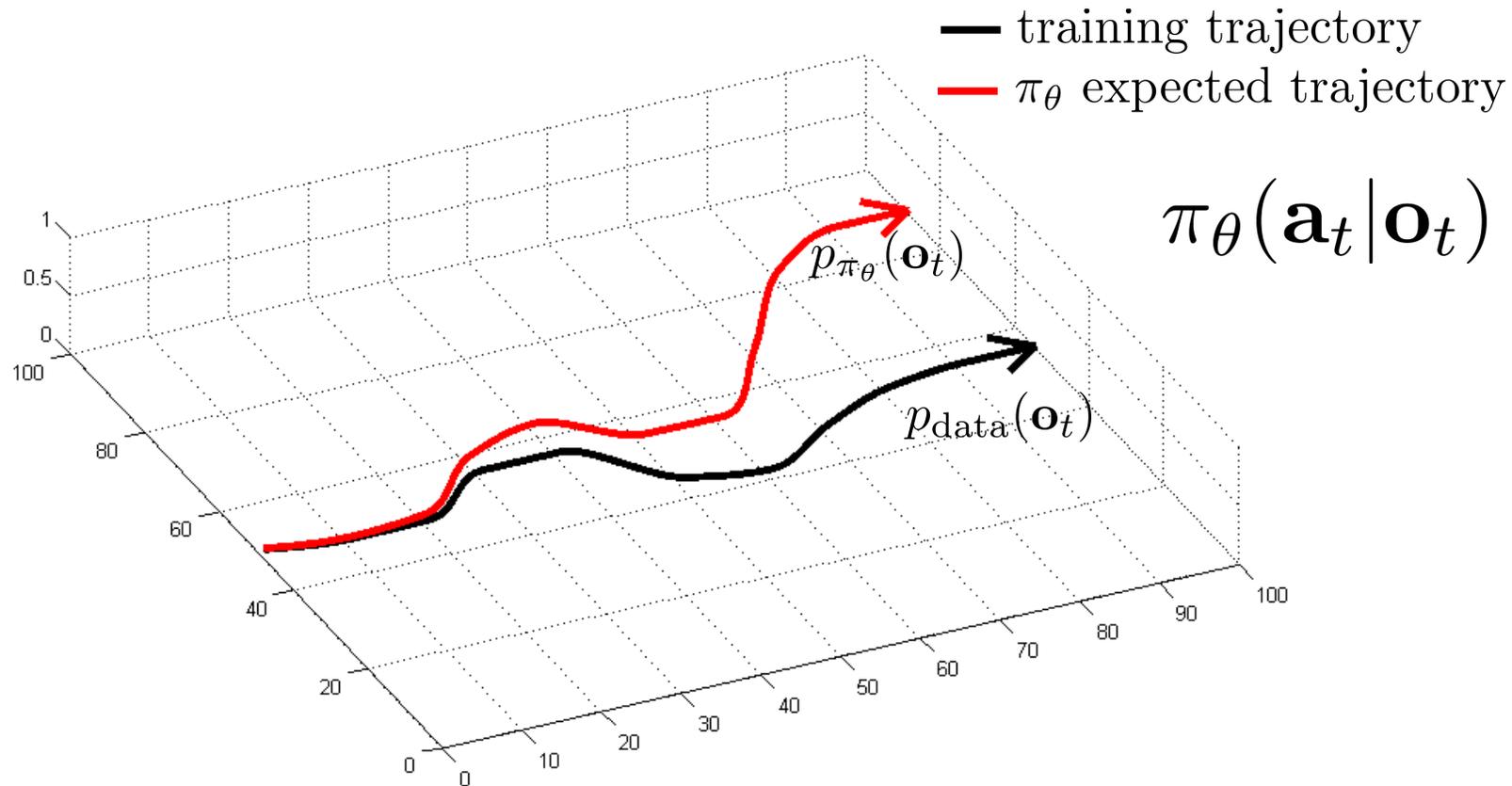


stability

Corrective labels that bring you back to the data



What might this mean mathematically?



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

Concrete Instantiation: DAgger

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

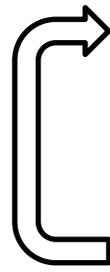
idea: instead of being clever about $p_{\pi_\theta}(\mathbf{o}_t)$, be clever about $p_{\text{data}}(\mathbf{o}_t)$!

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 !

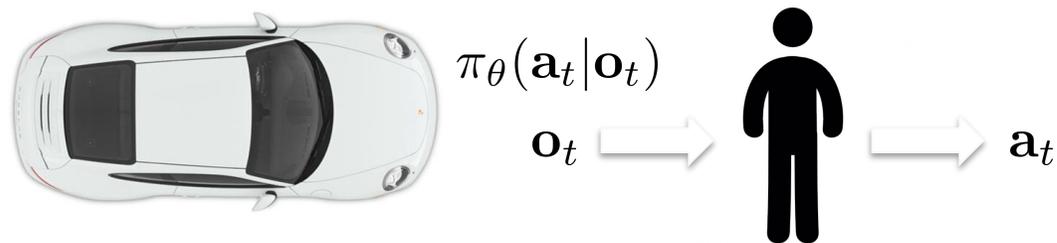
- 
1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \dots, \mathbf{o}_N, \mathbf{a}_N\}$
 2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
 3. Ask human to label \mathcal{D}_π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

Dagger Example



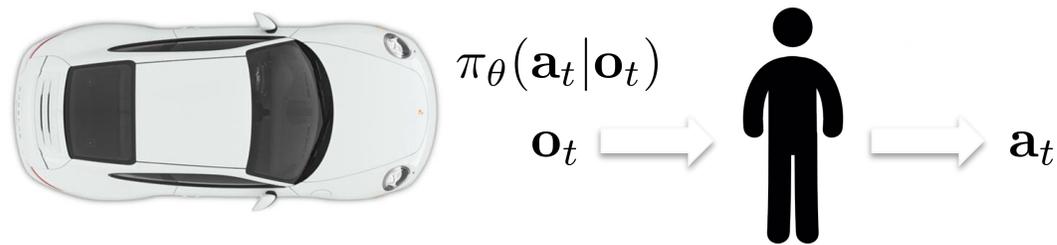
What's the problem?

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



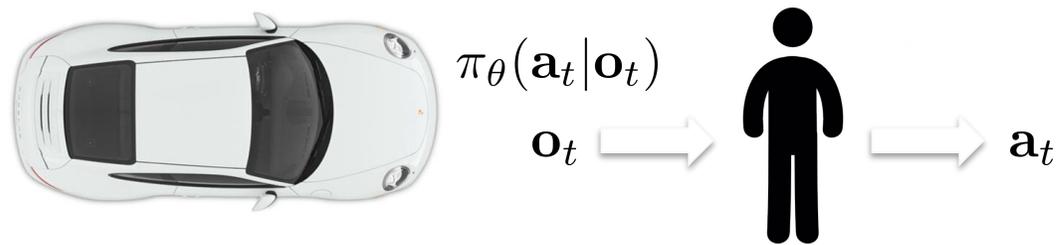
How might we fix this?

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



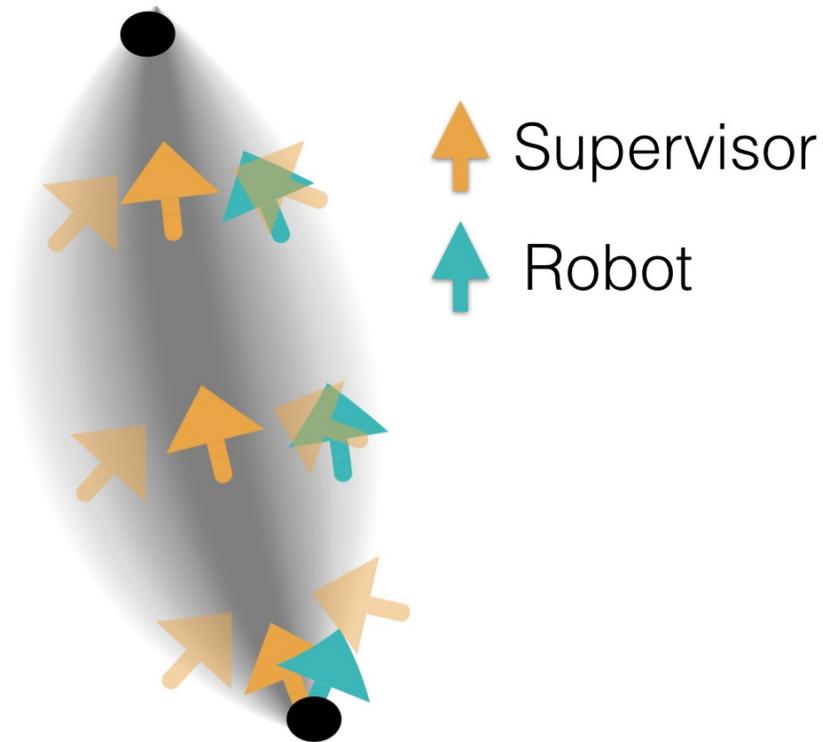
How might we fix this?

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



Noising the Data Collection Process

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



Noise Injection

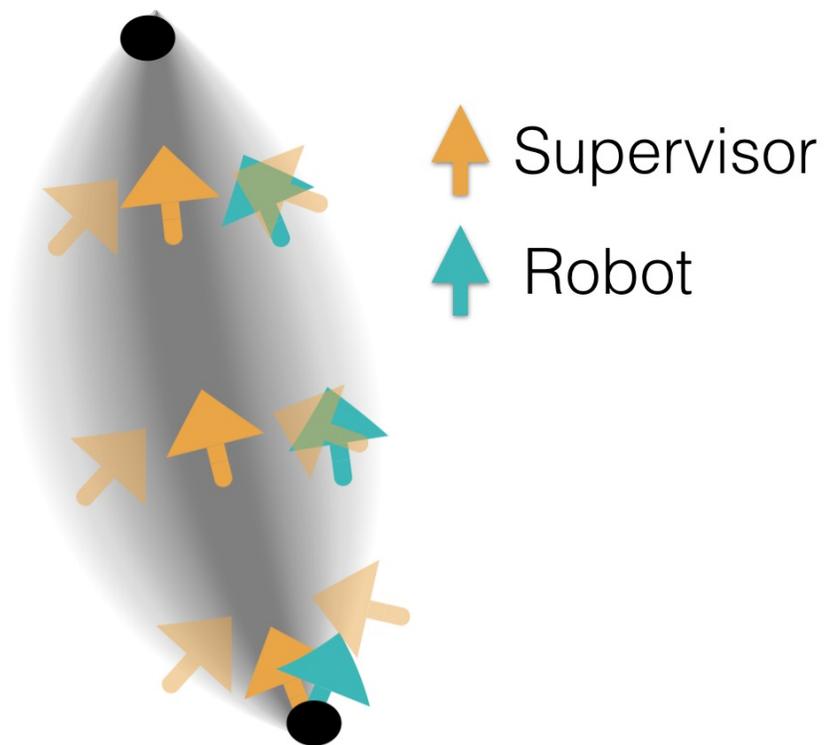
$$\hat{\psi}_{k+1} = \underset{\psi}{\operatorname{argmin}} E_{p(\xi|\pi_{\theta^*}, \psi_k)} - \sum_{t=0}^{T-1} \log [\pi_{\theta^*}(\pi_{\hat{\theta}}(\mathbf{x}_t)|\mathbf{x}_t, \psi)]$$

↑
Maximize likelihood

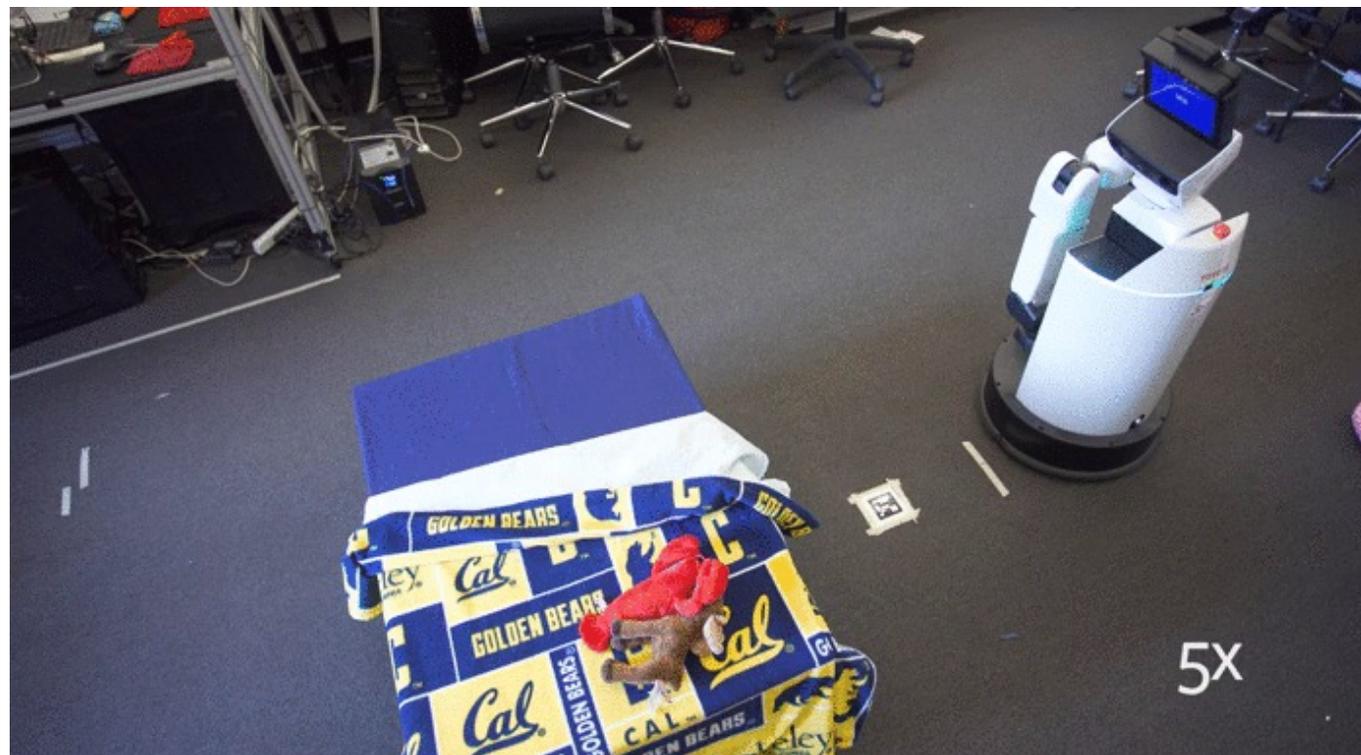
↑
Under noise during data collection

Noising the Data Collection Process

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

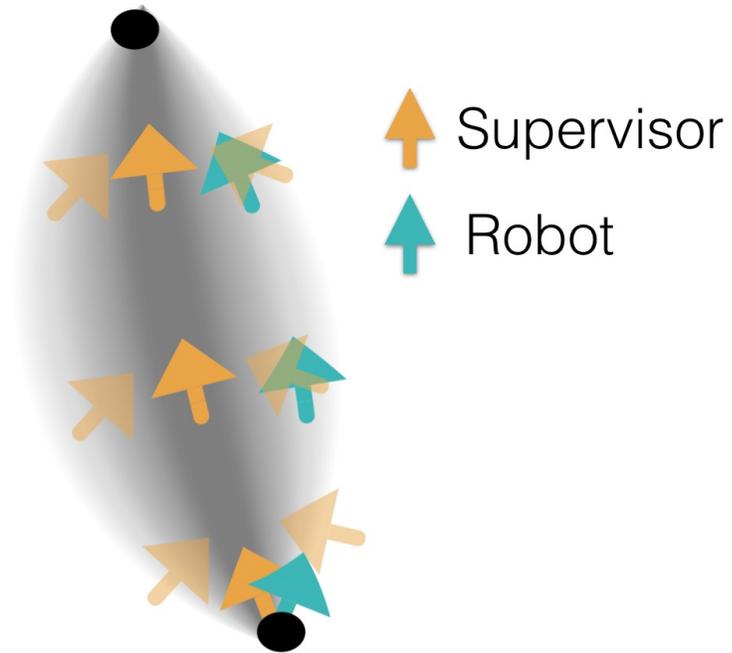


Noise Injection

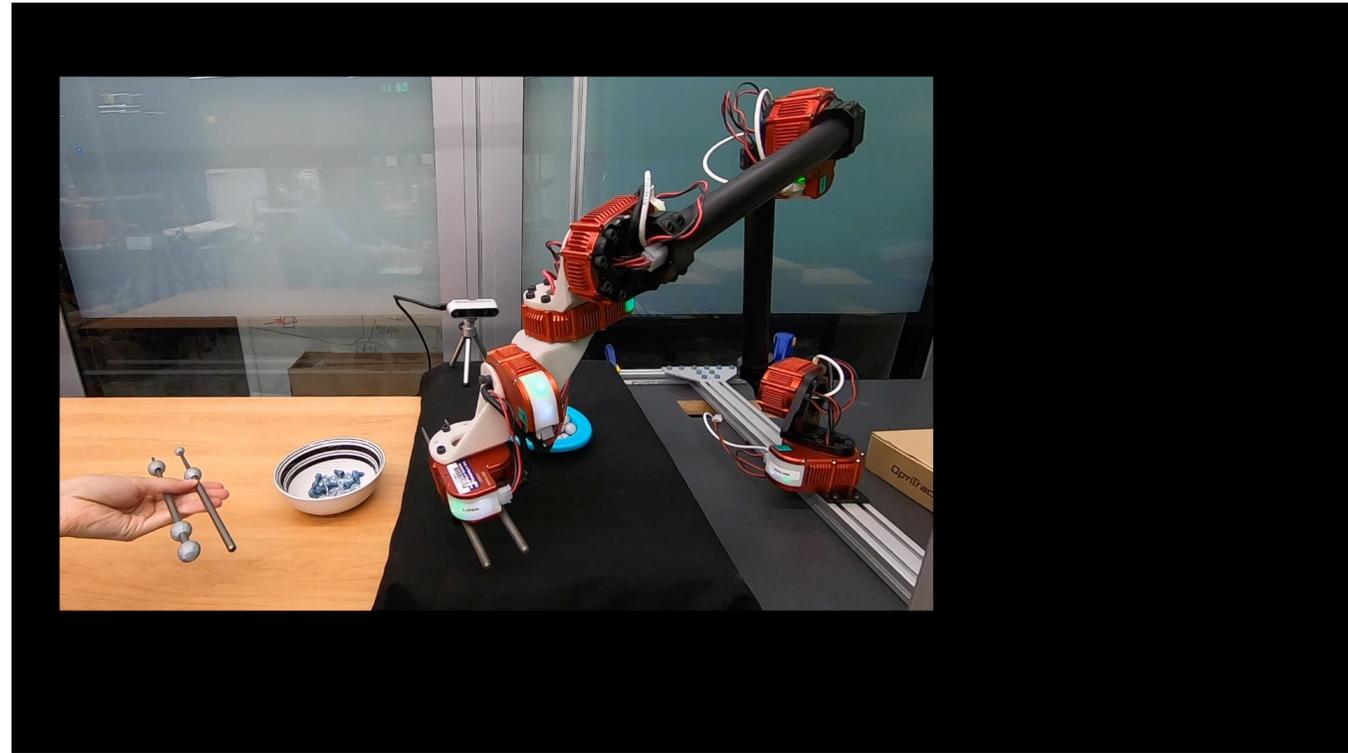


Why might this not be enough?

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



Noise Injection

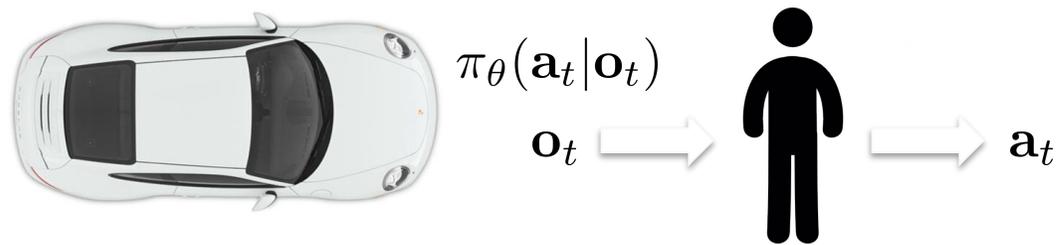


Assumes that the expert can actually perform behaviors under noise
→ Not always possible!

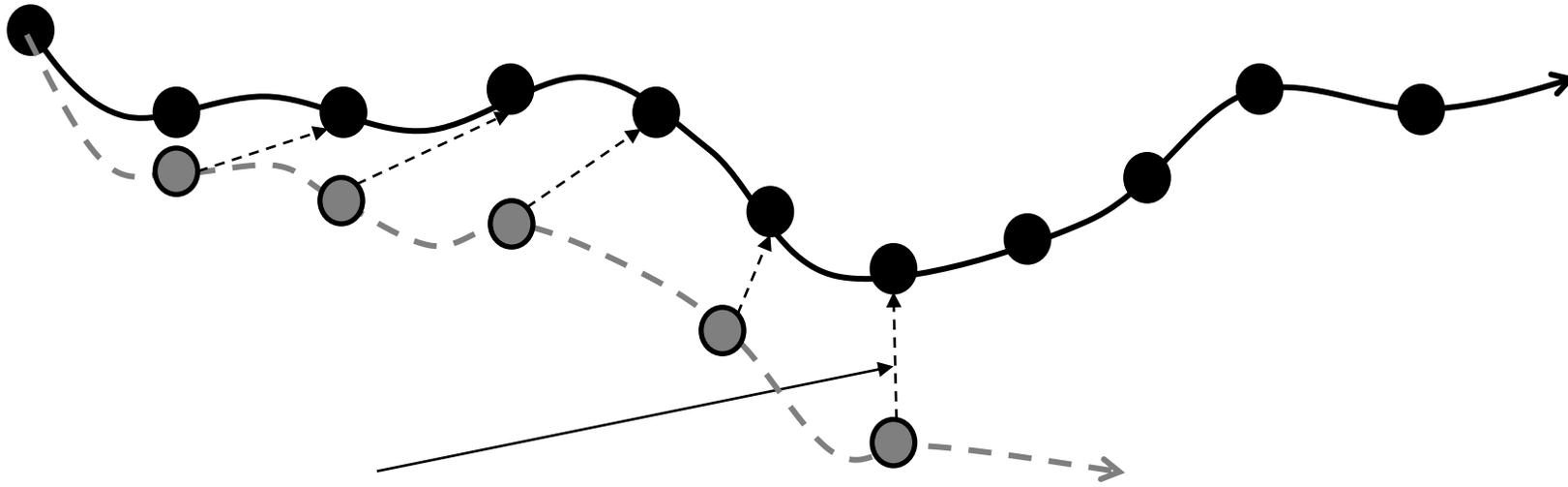
How might we fix this?

"Generate"
corrective labels
automatically

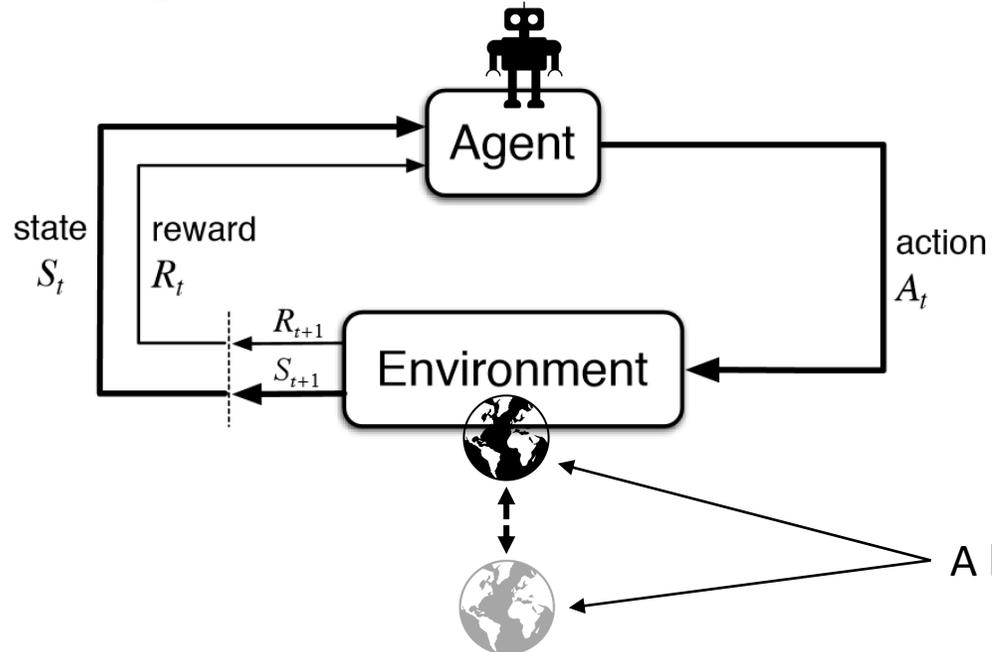
1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \dots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask human to label \mathcal{D}_π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



How can we find corrective labels?



How might we obtain these corrections?



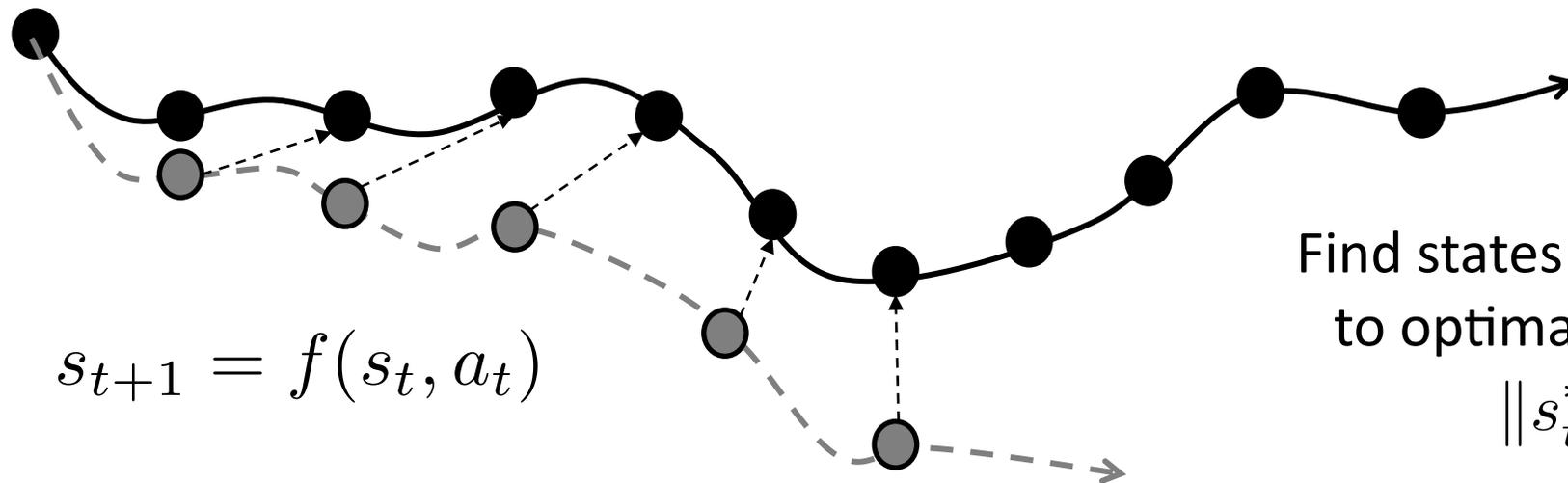
Key insight: Augment D with states (s_t), actions (a_t) that lead back to optimal states under dynamics

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

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

A known/approximate dynamics model can help find corrective labels

Generating Corrective Labels for Imitation Learning



 $f(s_t, a_t)$

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

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

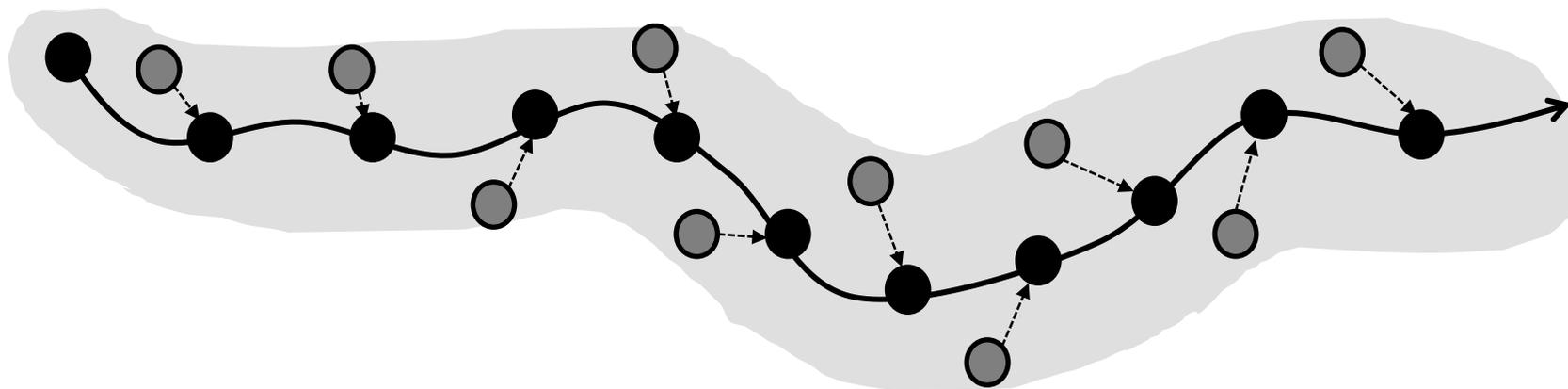
$$\min_{s_t, a_t} \|s_{t+1}^* - f(s_t, a_t)\|$$

Intuition: find labels to bring OOD states back in distribution (where policy can be trusted)

Easy with known dynamics

But dynamics are not known! \longrightarrow More machinery needed with learned dynamics!

Generating Corrective Labels for Imitation Learning with Learned Dynamics



minimizing MSE on expert data
+ spectral norm

When can we trust learned dynamics \hat{f}_ϕ ?

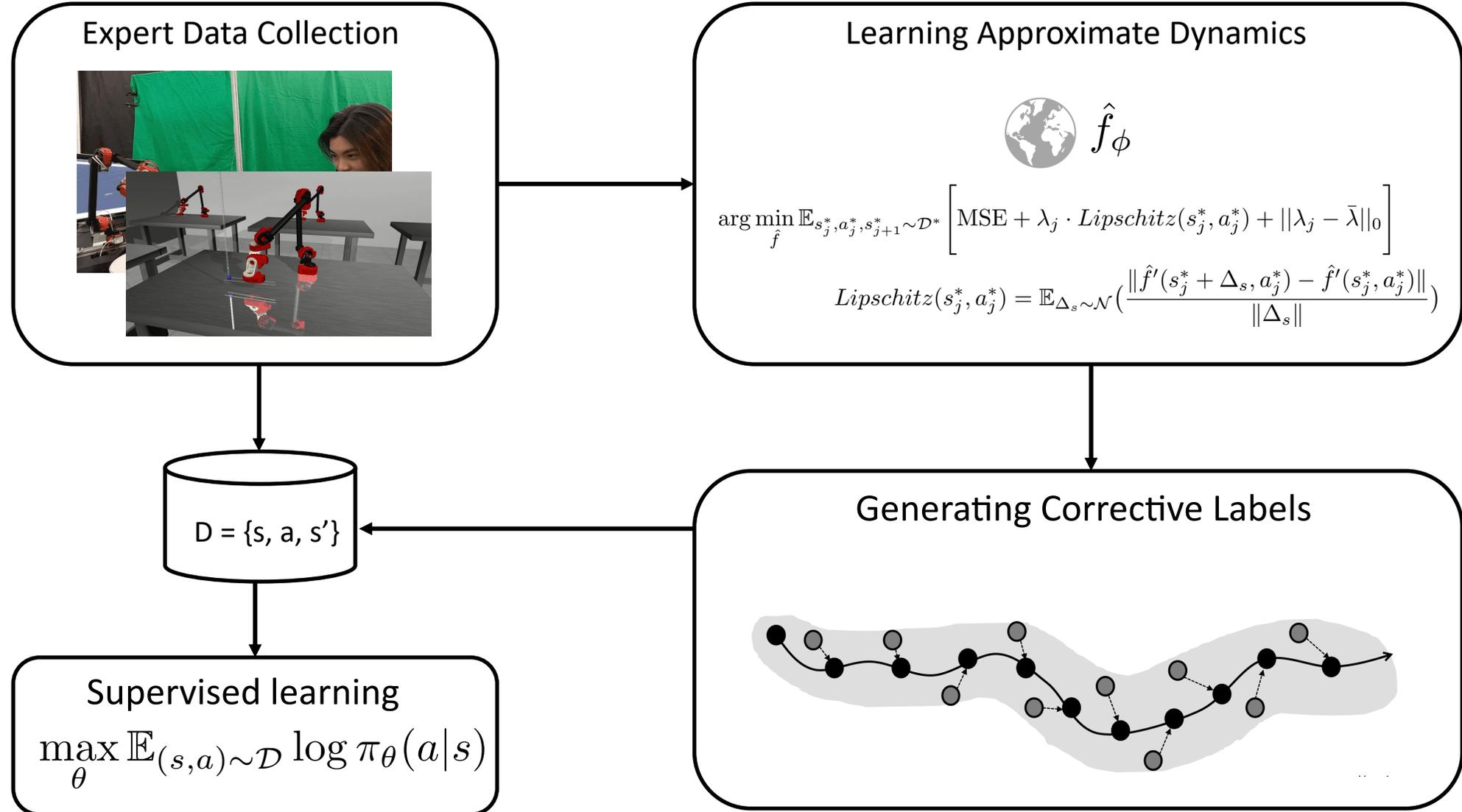


Under approximately Lipschitz smooth models, trust models around training data

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

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

Overall Learning Pipeline with Corrective Labels

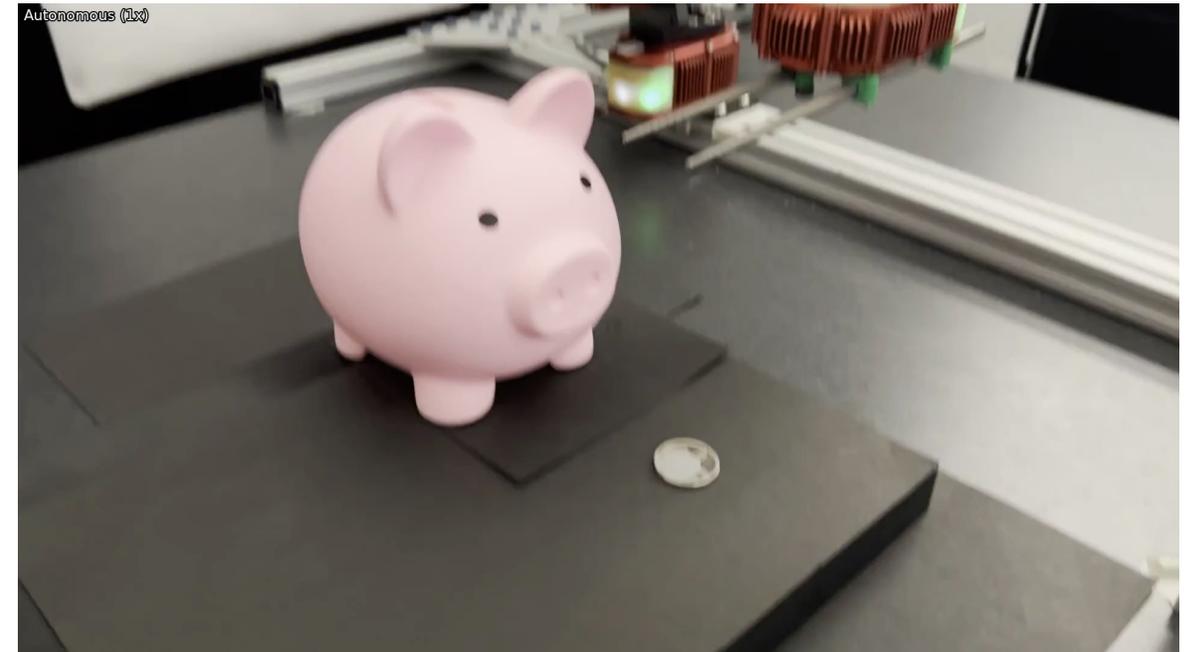


How well does generating corrective labels work?

With corrective labels

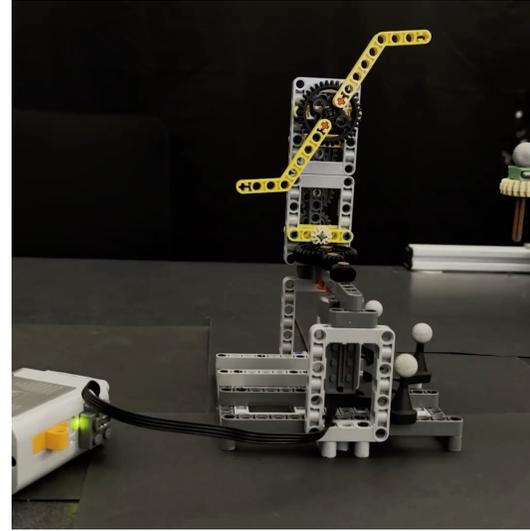
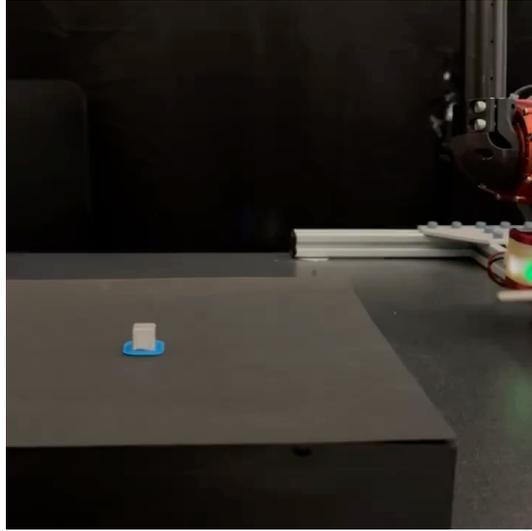


Without corrective labels

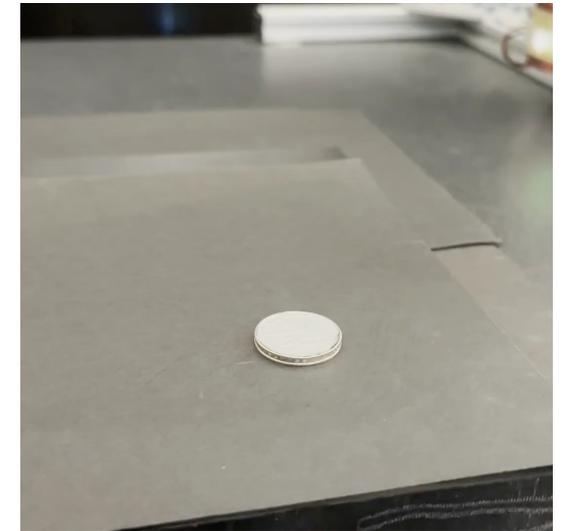
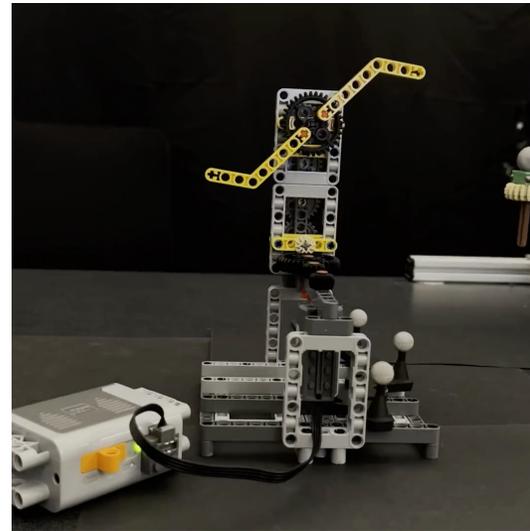
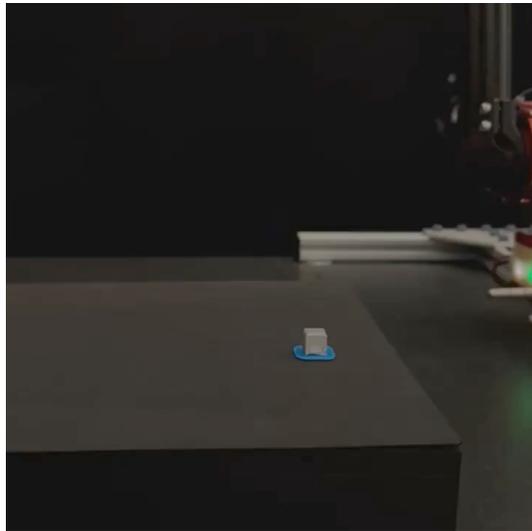


How well does generating corrective labels work?

With corrective labels



Without corrective labels



So does this solve all the issues in imitation?

Lecture Outline

A Formalism for Sequential Decision Making



Imitation Learning: Behavior Cloning



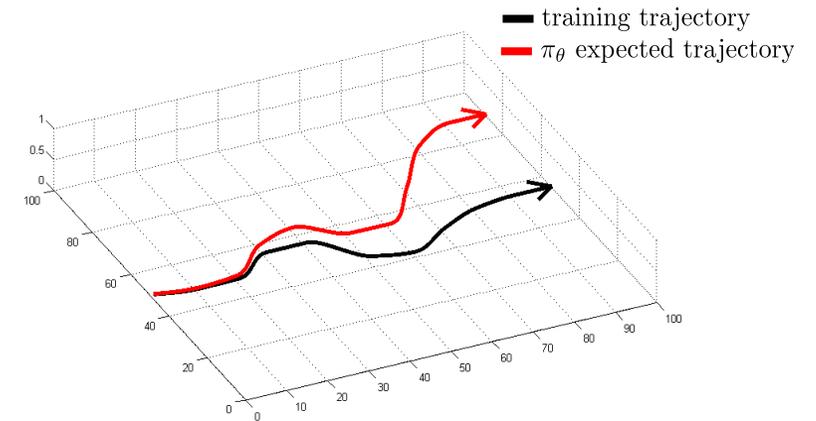
Imitation Learning: Improvements – Compounding Error



Imitation Learning: Improvements – Multimodality

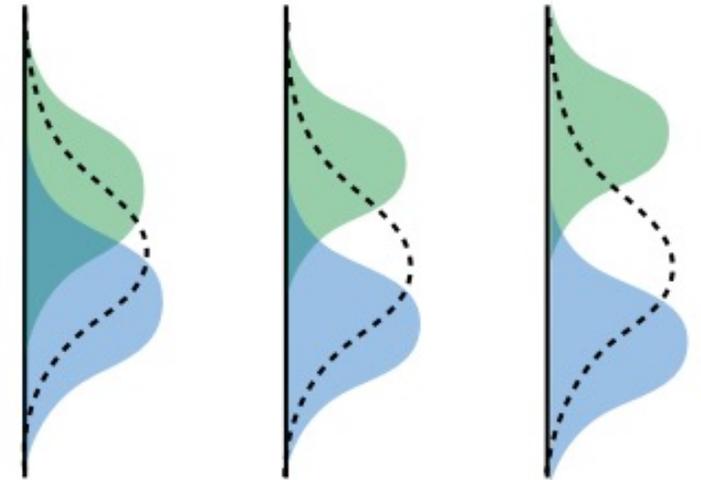
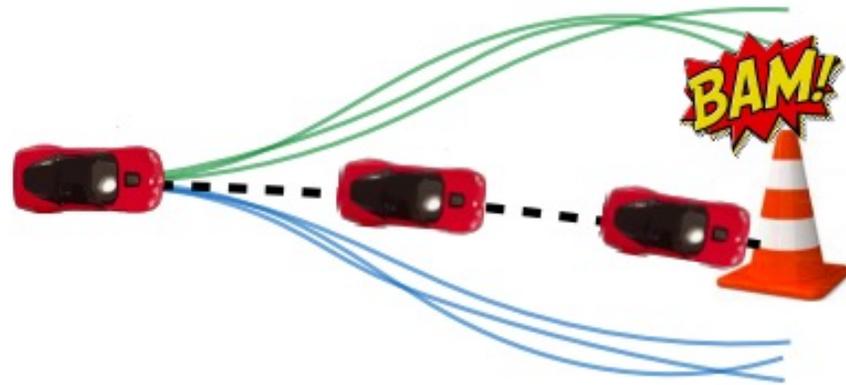
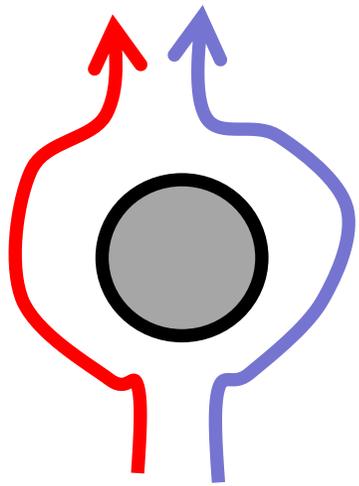
Can we make it work without more data?

- DAgger addresses the problem of distributional “drift”
- What if our model is so good that it doesn’t drift?
- Need to mimic expert behavior very accurately
- But don’t overfit!



Why might we fail to fit the expert?

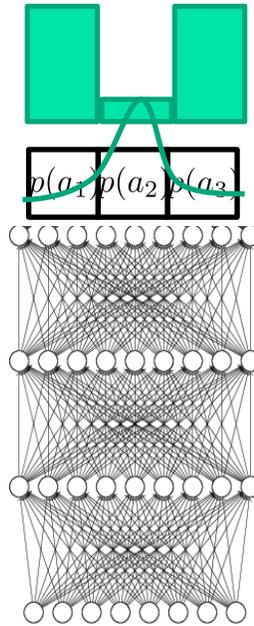
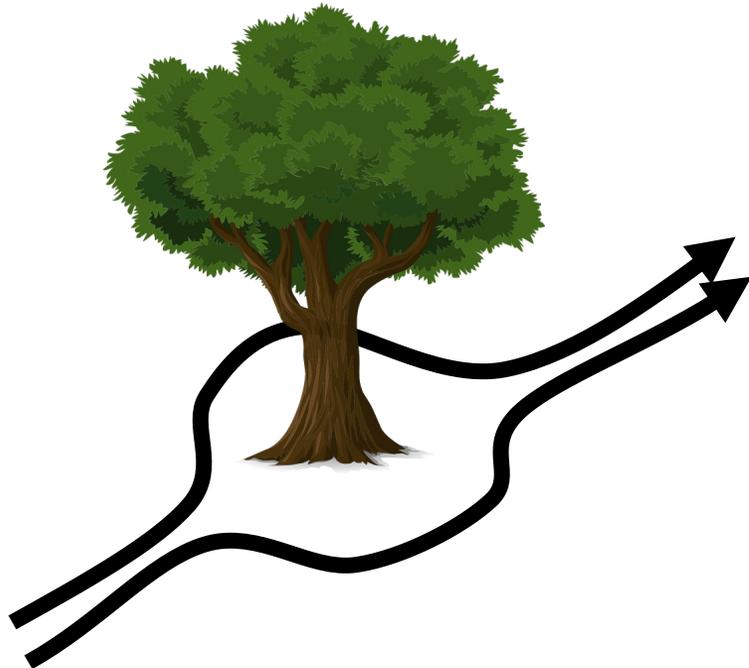
Multimodal behavior.. amongst other reasons



Not a matter of network size! It's about distributional expressivity

Why might we fail to fit the expert?

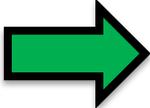
Multimodal behavior → use more expressive probability distributions



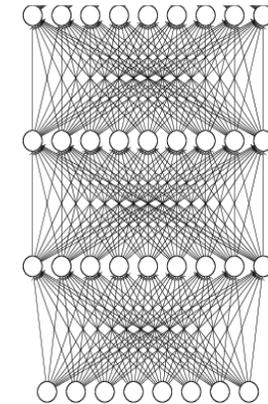
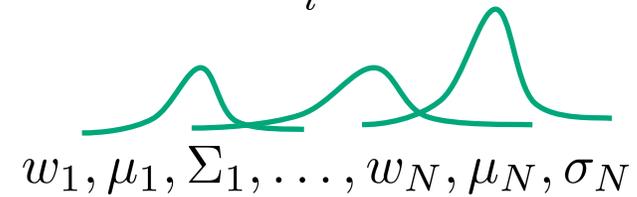
1. Output mixture of Gaussians
2. Latent variable models
3. Autoregressive discretization
4. Diffusion models
5. ...



Why might we fail to fit the expert?

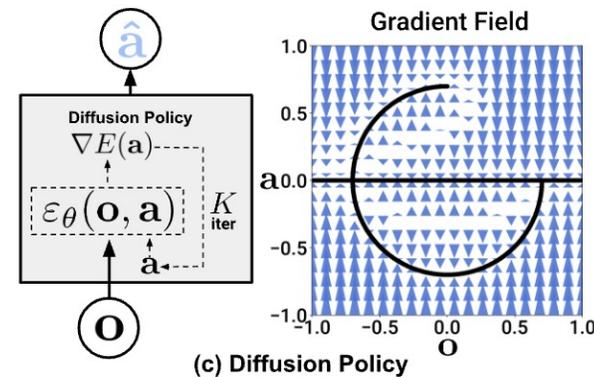
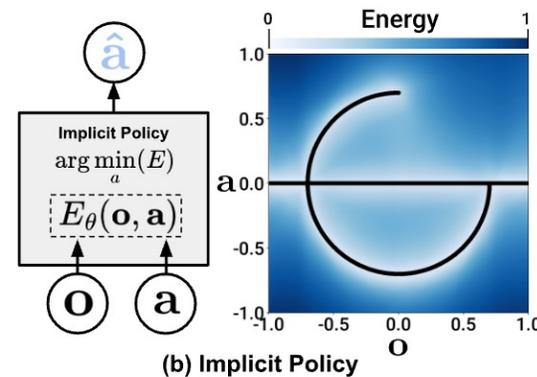
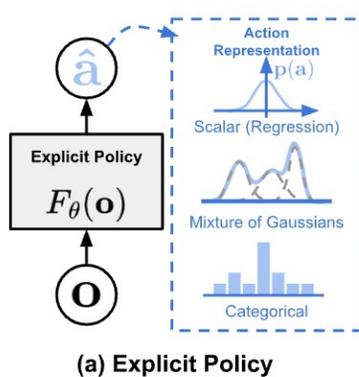
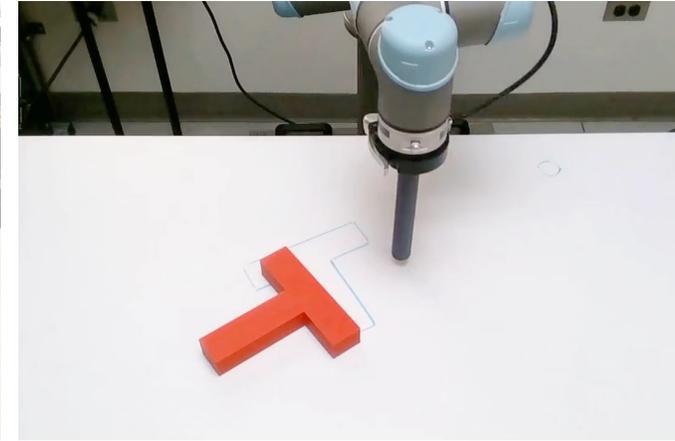
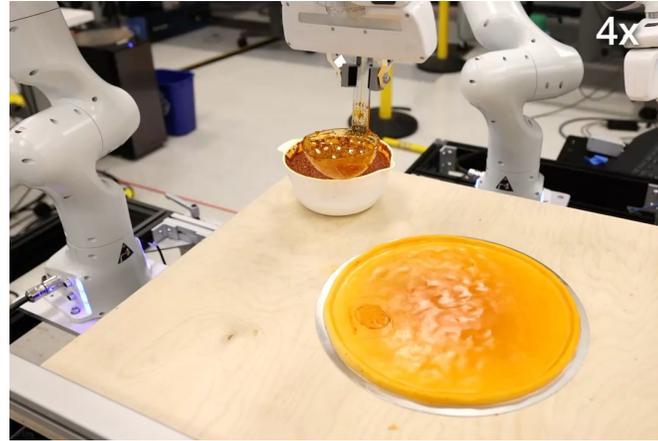
- 
1. Output mixture of Gaussians
 2. Latent variable models
 3. Autoregressive discretization
 4. Diffusion models
 5. ...

$$\pi(\mathbf{a}|\mathbf{o}) = \sum_i w_i \mathcal{N}(\mu_i, \Sigma_i)$$



Why might we fail to fit the expert?

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



Lecture Outline

A Formalism for Sequential Decision Making



Imitation Learning: Behavior Cloning



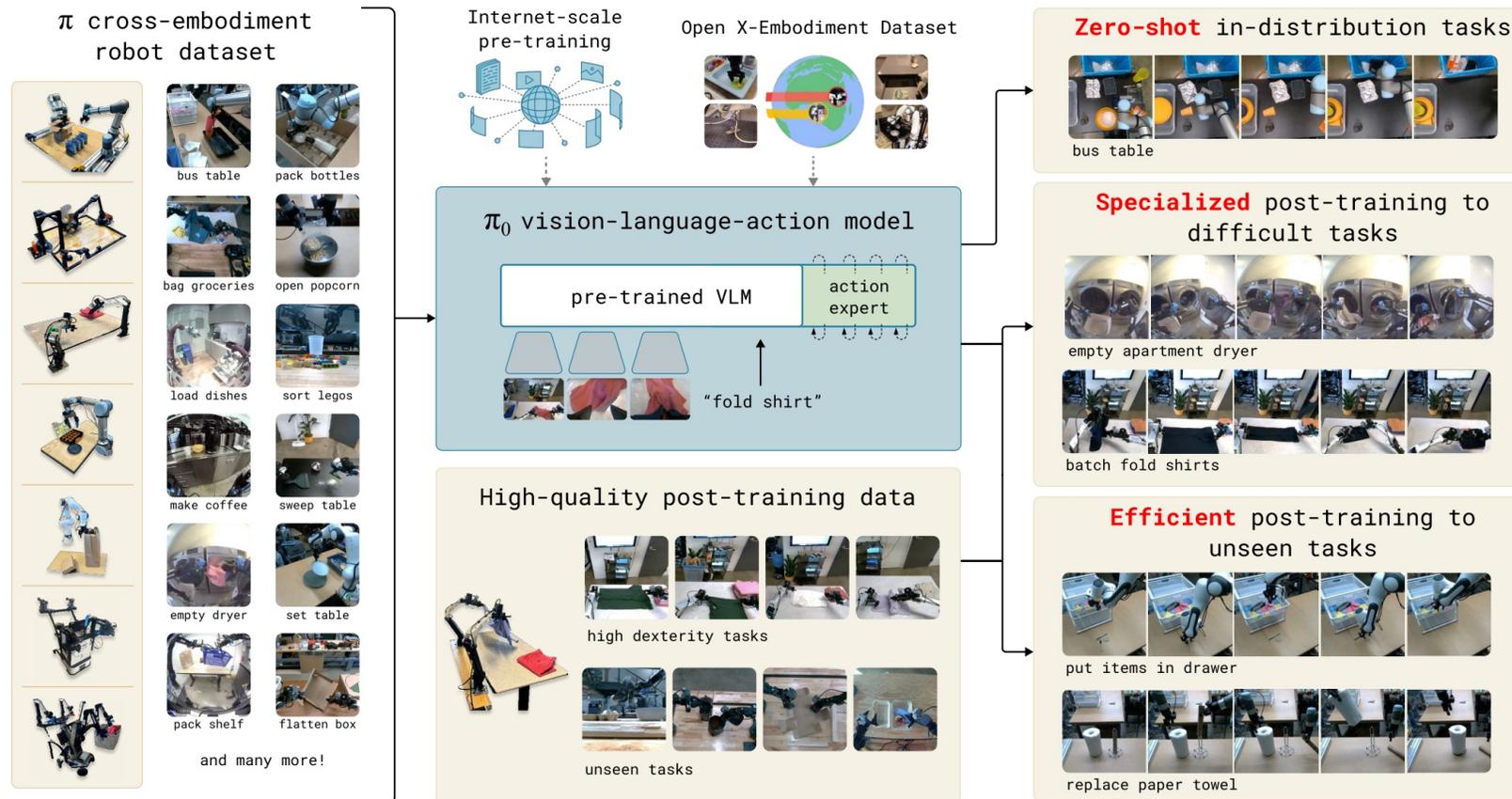
Imitation Learning: Improvements – Compounding Error



Imitation Learning: Improvements – Multimodality

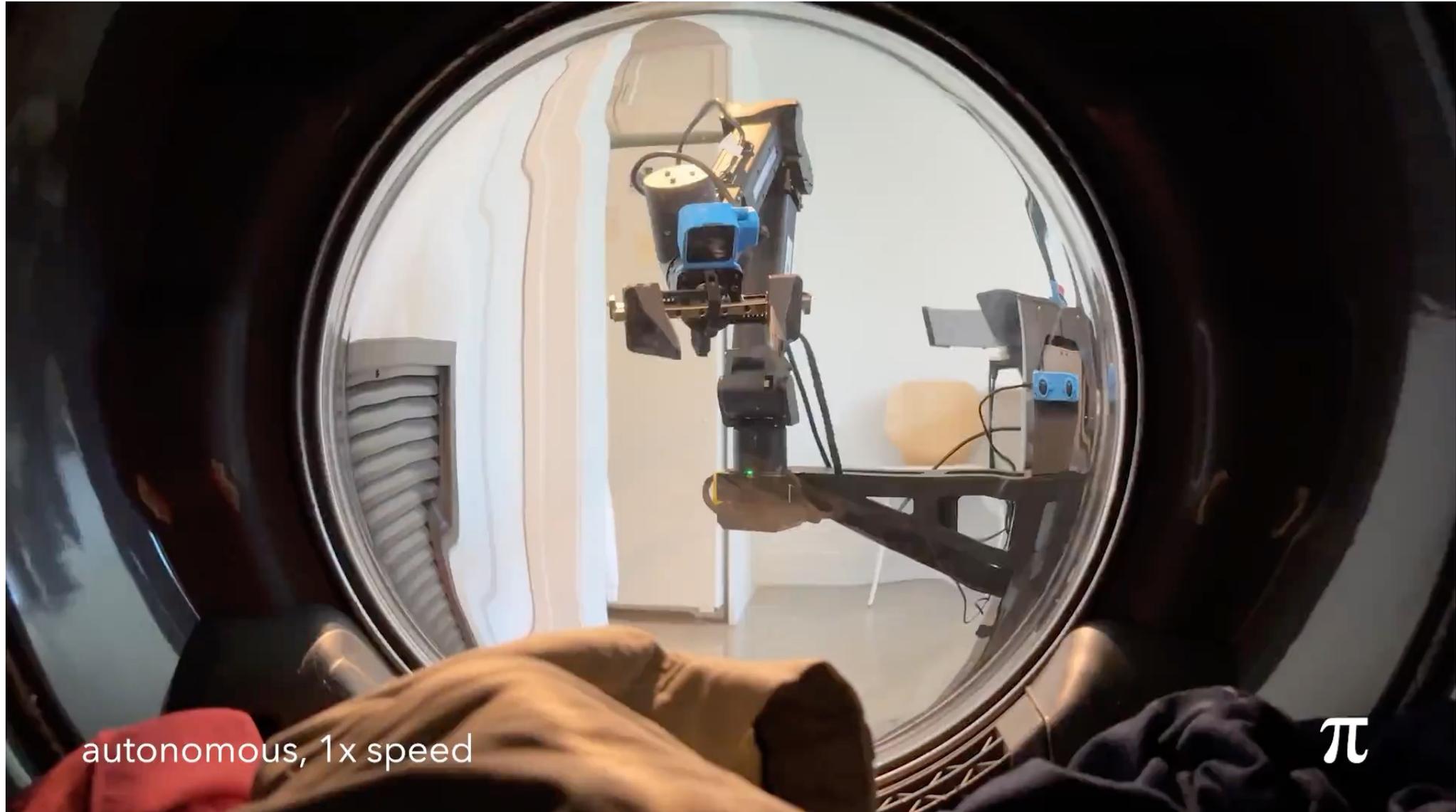
Taking Imitation a Step Further: VLAs

Recast imitation learning for robotics as a vision language prediction problem



Inherit generalization/semantic properties of VLMs

Taking Imitation a Step Further: VLAs

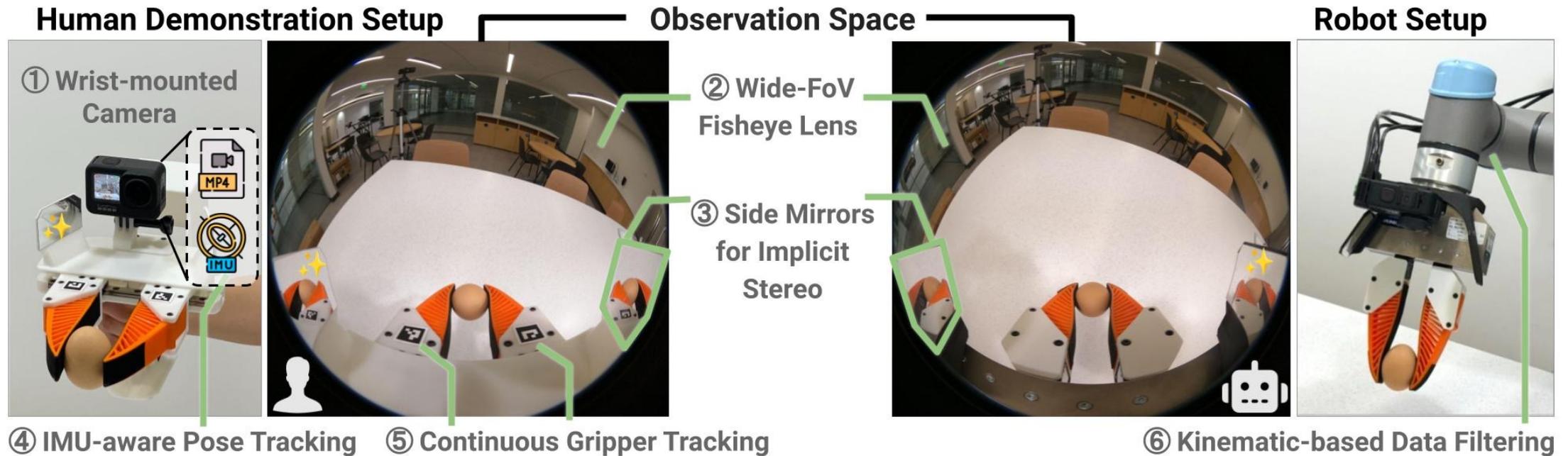


autonomous, 1x speed

π

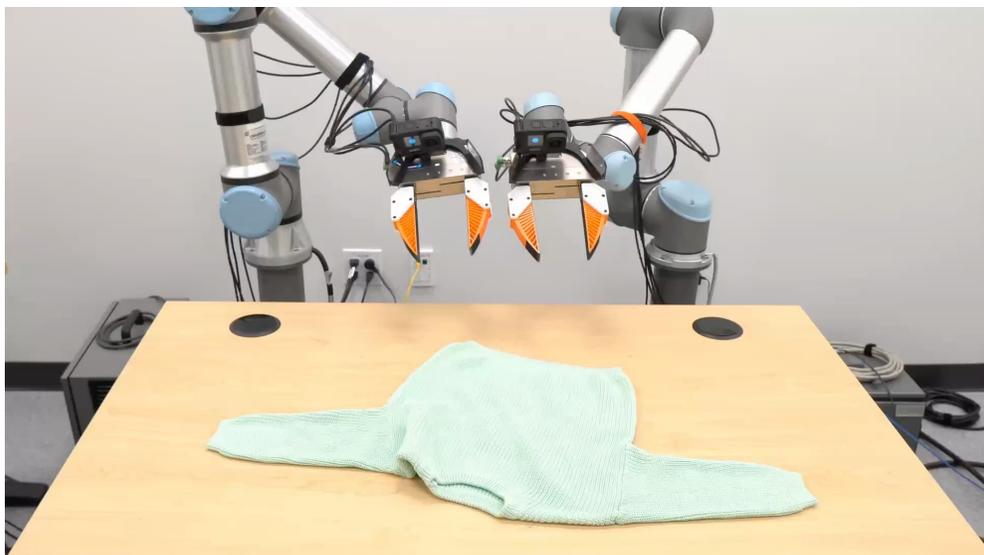
What about the embodiment gap?

Universal Manipulation Interface



Extract motion through SLAM for imitation learning – minimal embodiment gap

What about the embodiment gap?



Some cool imitation videos

1x and tesla humanoid robots



- 1X END-TO-END AUTONOMY
UPDATE, JAN 2024

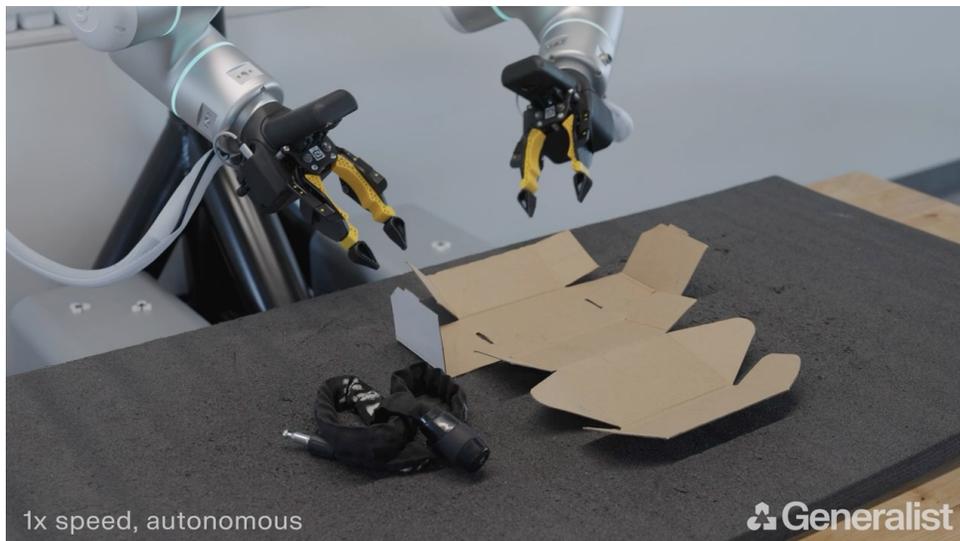
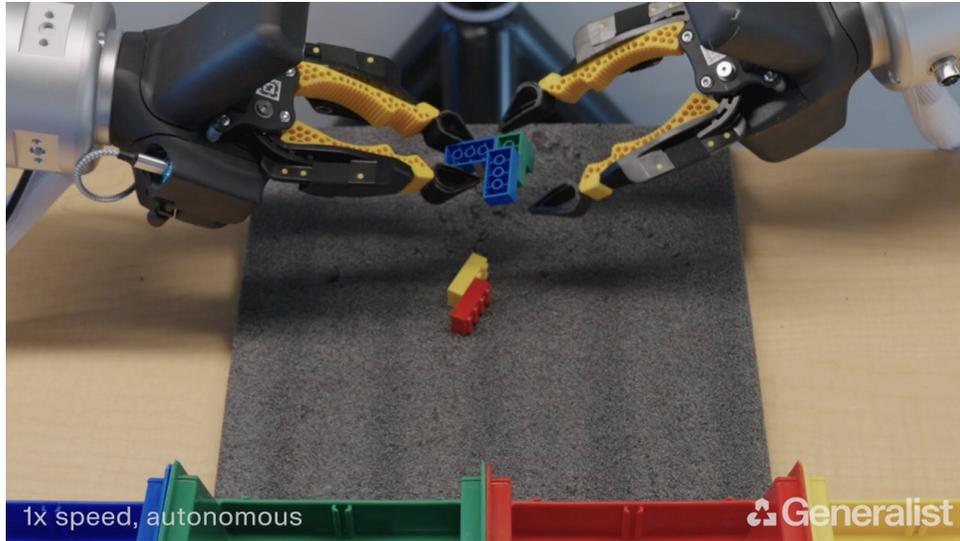
ALOHA Manipulation



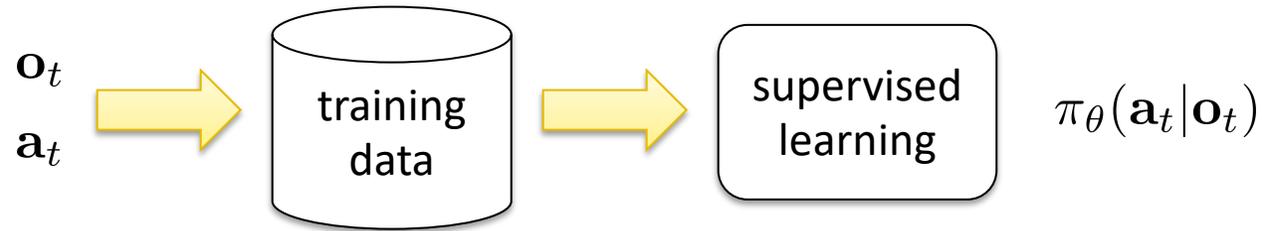
TRI Diffusion Policies



Generalist – Robot Policies with Large Scale Imitation

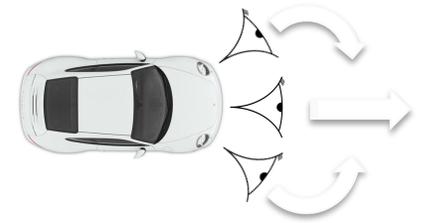


Perspectives on Imitation – don't believe everything you see online



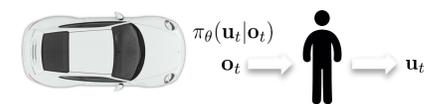
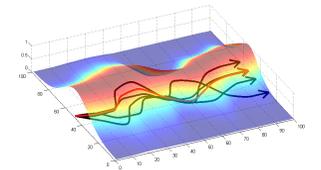
■ Pros:

- 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

A Formalism for Sequential Decision Making



Imitation Learning: Behavior Cloning



Imitation Learning: Improvements – Compounding Error



Imitation Learning: Improvements – Multimodality

Class Outline

State Estimation

Robotic System Design

Filtering

Localization

SLAM

Control

Feedback Control

PID Control

MPC

LQR

Planning

Search

Heuristic Search

Motion Planning

Lazy Search

Learning

Imitation Learning

Policy Gradient

Actor-Critic

Model-Based RL