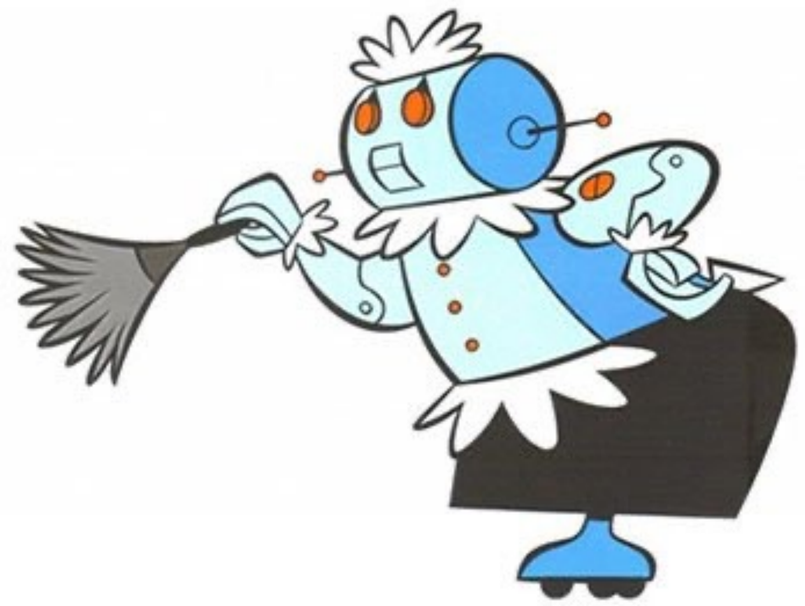
# Autonomous Robotics
# Winter 2026

Abhishek Gupta, Siddhartha Srinivasa
TAs: Carolina Higuera, Entong Su, Rishabh Jain



# Recap

# Idea 1: Imitation Learning via Behavior Cloning

Given: Demonstrations of optimal behavior

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

Goal: Train a policy to mimic the demonstrator

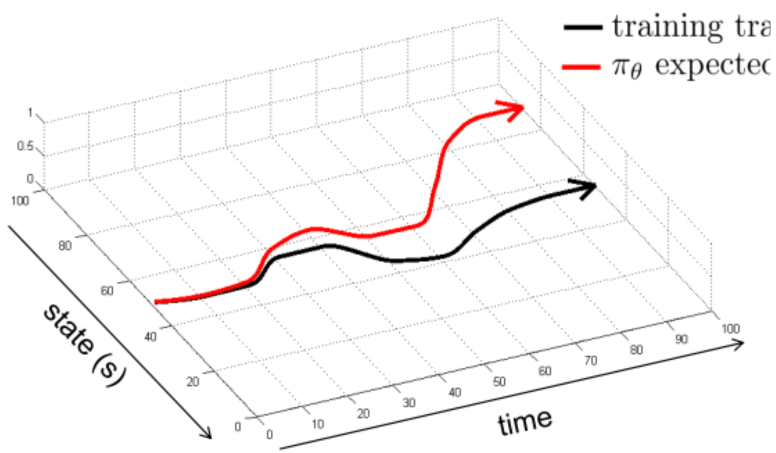
Discrete vs continuous

Maximum likelihood

```
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()
```

# So does behavior cloning really work?

- Imitation Learning ≠ Supervised Learning




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

Not the same!

# 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 $p_{\text{data}}(\mathbf{o}_t)$!

## **D**Agger: **D**ataset **Agg**regation

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

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

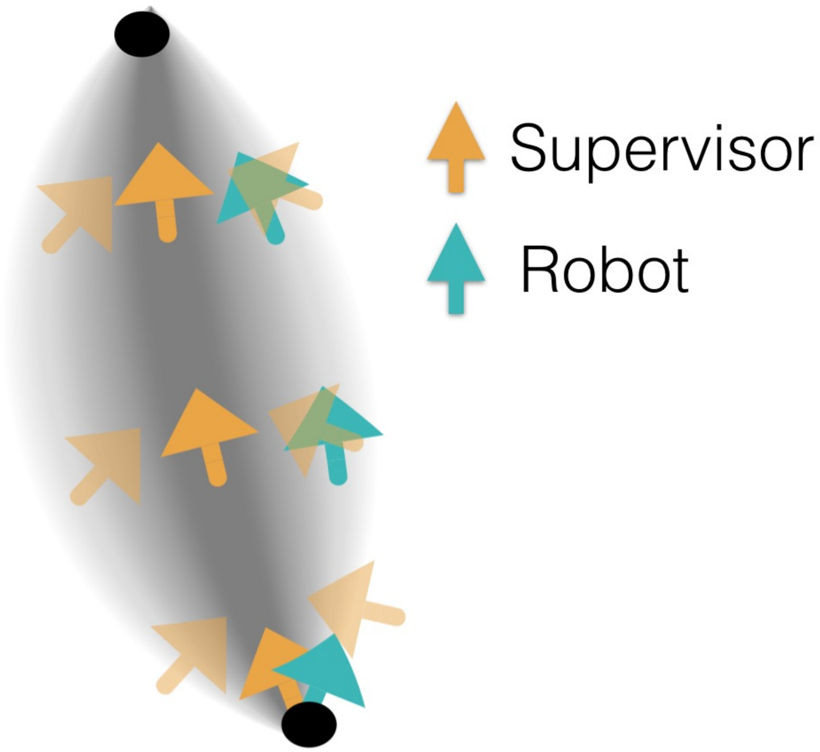
but need labels $\mathbf{a}_t$!

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 $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

Ross et al. '11

# Noising the Data Collection Process

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




Noise Injection

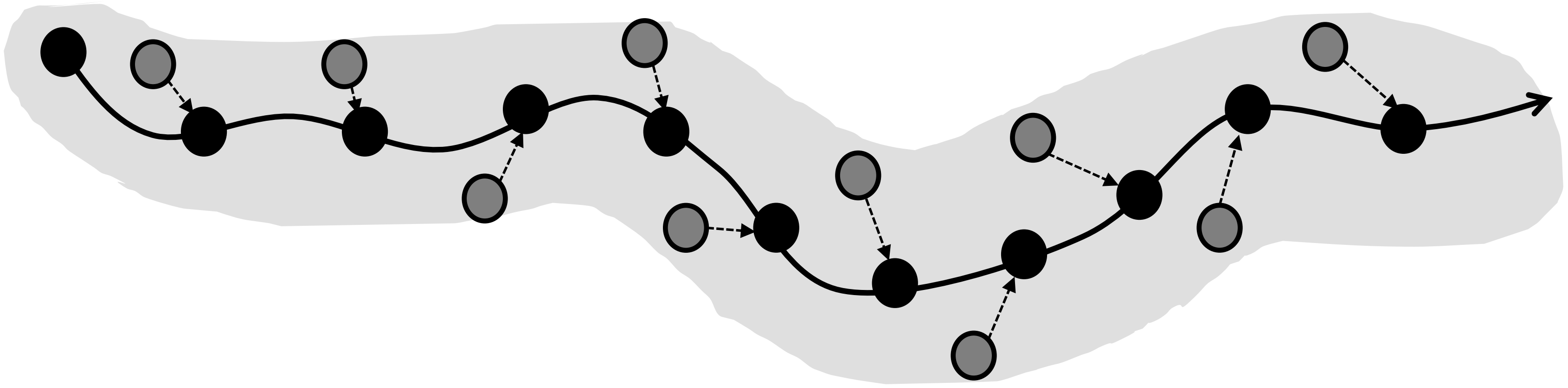
$$\hat{\psi}_{k+1} = \underset{\psi}{\mathbf{argmin}}\, E_{p(\xi|\pi_{\theta^*},\psi_k)} - \sum_{t=0}^{T-1} \log\left[\pi_{\theta^*}\left(\pi_{\hat{\theta}}(\mathbf{x_t})|\mathbf{x_t},\psi\right)\right]$$

Maximize likelihood

Under noise during data collection

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

# Generating Corrective Labels for Imitation Learning with Learned Dynamics



$\hat{f}_\phi$

minimizing MSE on expert data
+ spectral norm

When can we trust learned dynamics $\hat{f}_\phi$ ?

↓

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**

CCIL: Continuity-based data augmentation for corrective imitation learning, Ke et al ICLR '24

# Lecture Outline

Multimodality in Imitation Learning

↓

Recent Frontiers in Imitation Learning

↓

Policy Gradient

↓

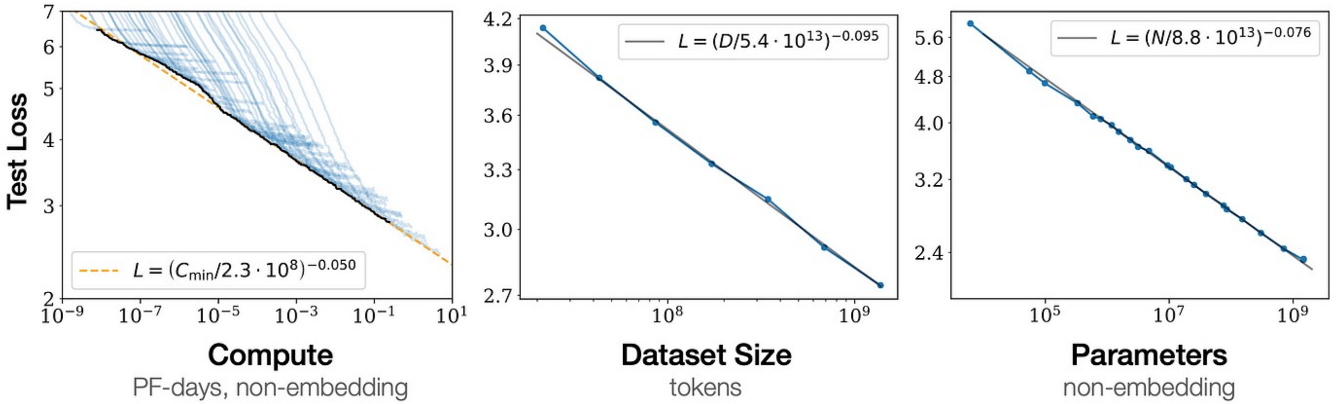
Improving Policy Gradient

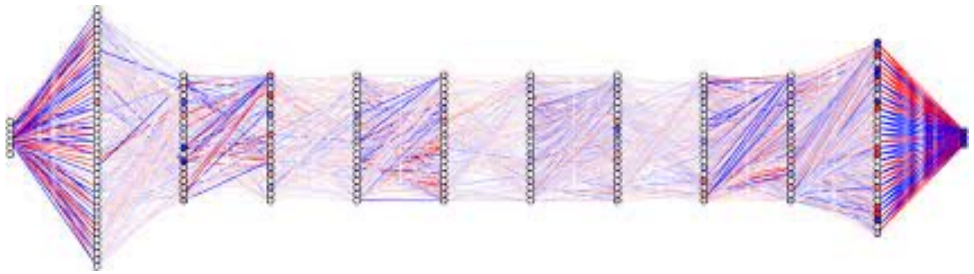
# What if we underfit the data?

- Behavior cloning can underfit the data

$$\sum_t \mathbb{E}_{(s_t,a_t)\sim p_{\pi_\theta}(s_t,a_t)}[c(s_t,a_t)] \leq O(\epsilon H^2) \qquad \pi_\theta(a \neq \pi^*(s_t)|s_t) \leq \epsilon$$

Q: won't a bigger model just solve the problem?



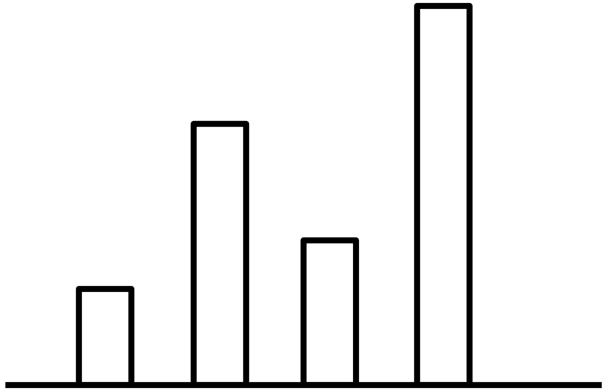



Kind of, but there's a fundamental problem!

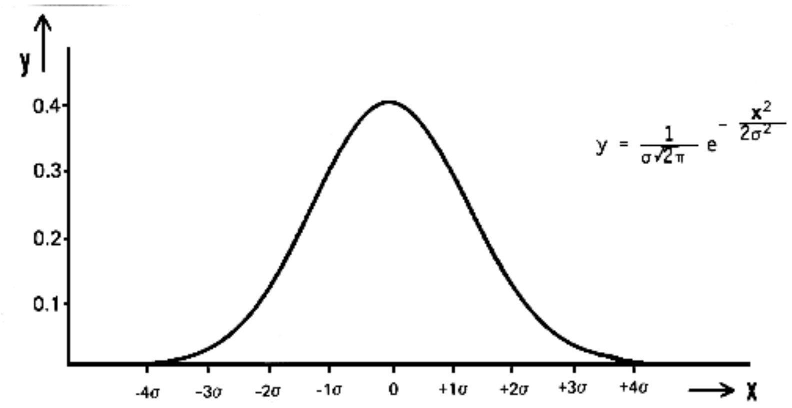
# Distributional Expressivity

- Policy expressivity is a combination of expressivity of the function approximator and of the distribution family

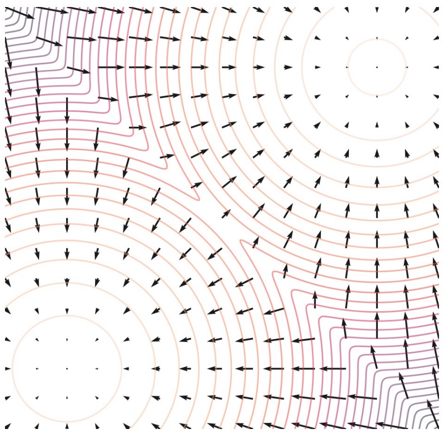
Categorical



Gaussian




Diffusion policy

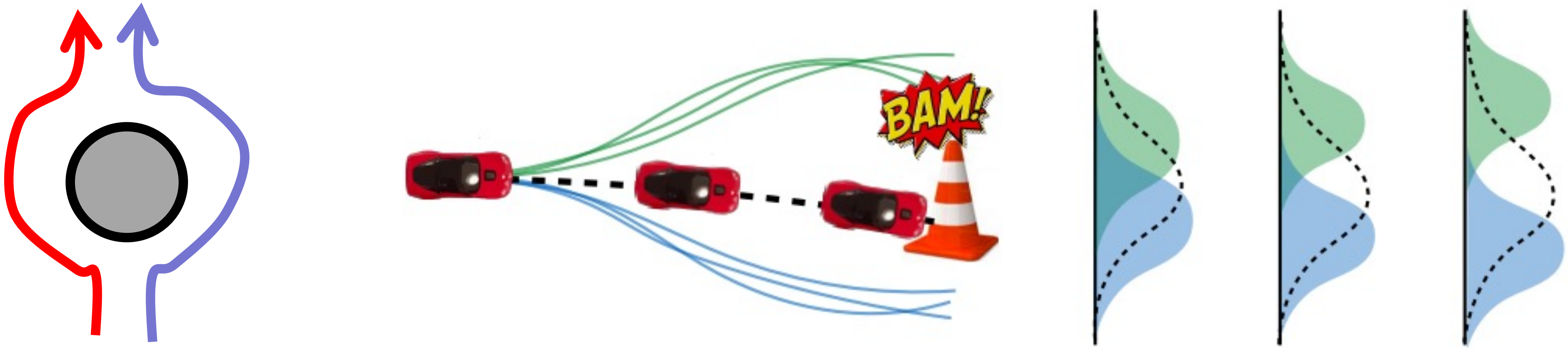


Tradeoff between expressivity and tractability

# How does this distributional expressivity manifest?

## Let us consider a case with Gaussian policy

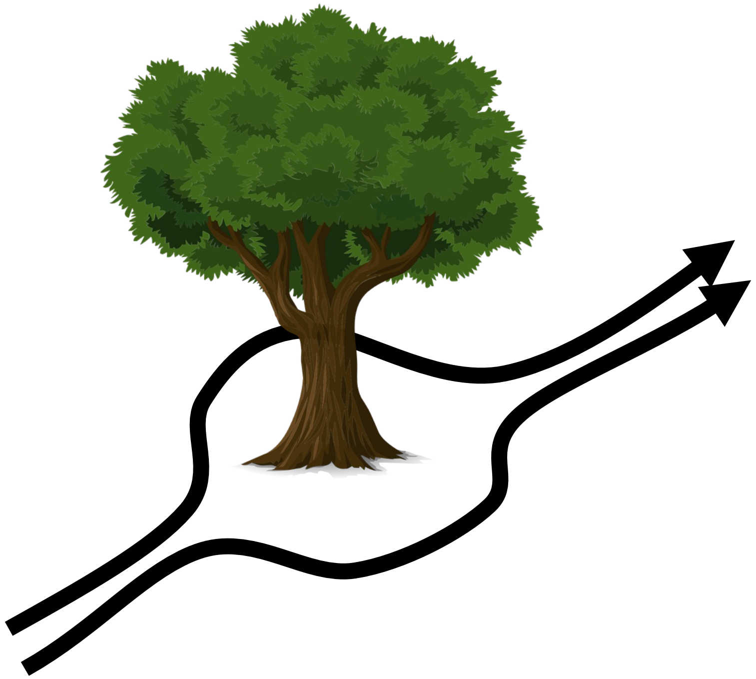
$$\arg\max_{\theta} \mathbb{E}_{(s^*, a^*) \sim \mathcal{D}} \left[ \log \pi_\theta(a^* | s^*) \right]$$

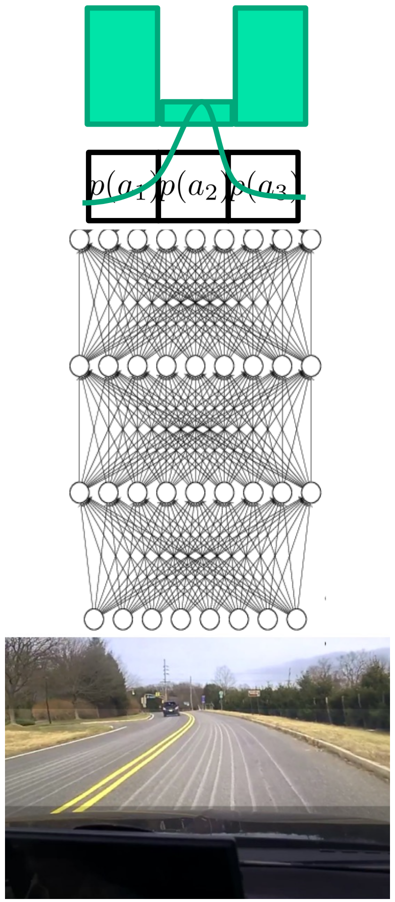



Not a matter of network size! It's about distributional expressivity

# Why might we fail to fit the expert?

Multimodal behavior → use more **expressive** probability distributions



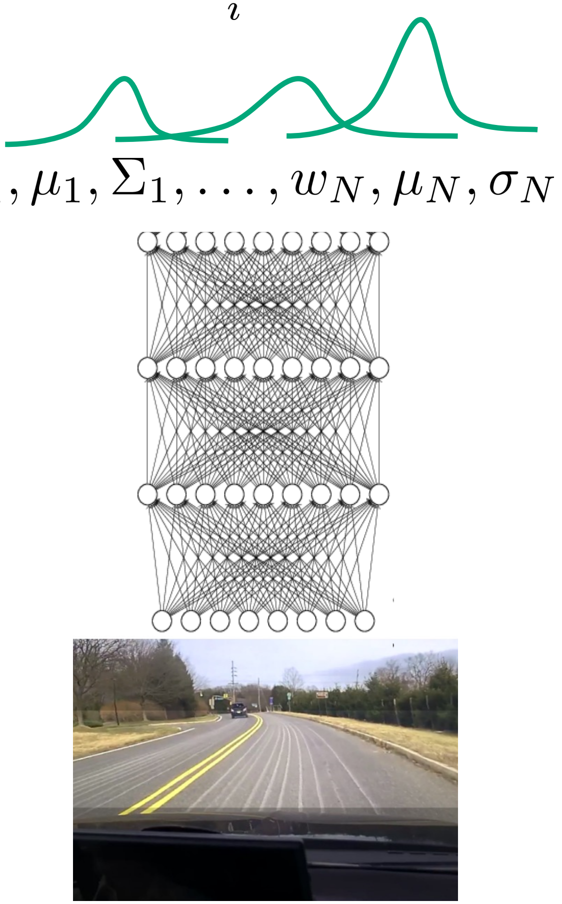


1. Output mixture of Gaussians
2. Latent variable models
3. Autoregressive discretization
4. Diffusion models
5. …

# Why might we fail to fit the expert?

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

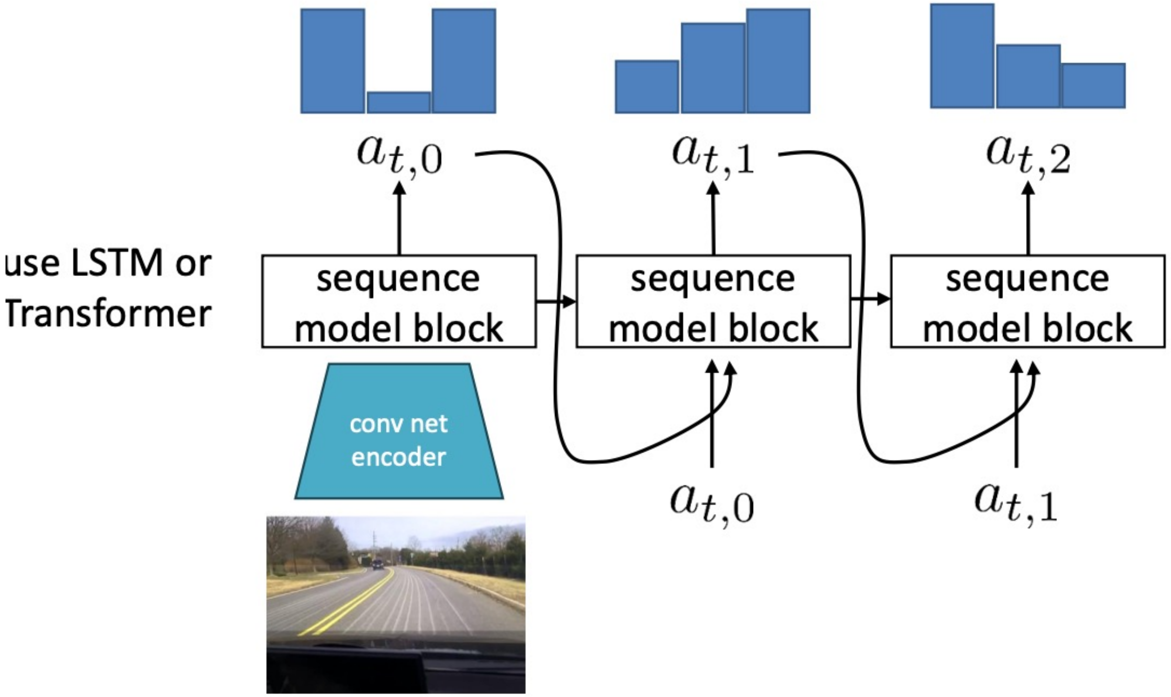
$$\pi(\mathbf{a}|\mathbf{o}) = \sum_i w_i \mathcal{N}(\mu_i, \Sigma_i)$$



# Why might we fail to fit the expert?

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

$$\mathbf{a}_t = \begin{pmatrix} 0.1 \\ 1.2 \\ -0.3 \end{pmatrix} \begin{matrix} a_{t,0} \\ a_{t,1} \\ a_{t,2} \end{matrix}$$



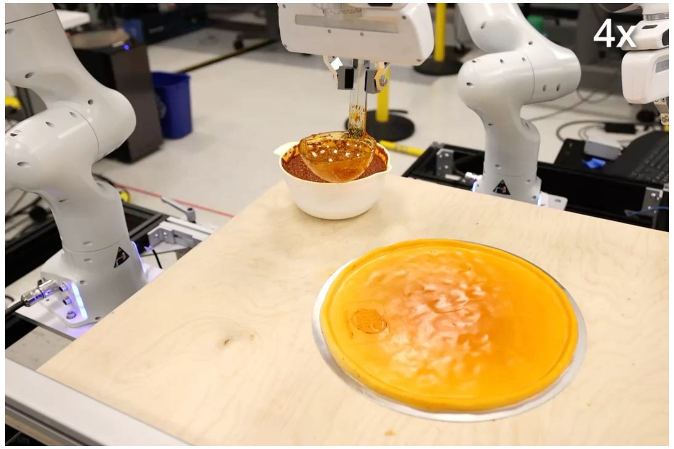

**Why** does this work?

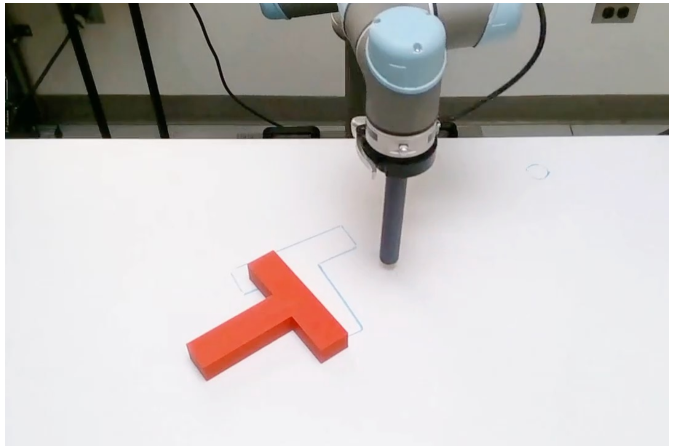
$$\text{first step: } p(a_{t,0}|\mathbf{s}_t)$$
$$\text{second step: } p(a_{t,1}|\mathbf{s}_t, a_{t,0})$$
$$\text{third step: } p(a_{t,2}|\mathbf{s}_t, a_{t,0}, a_{t,1})$$

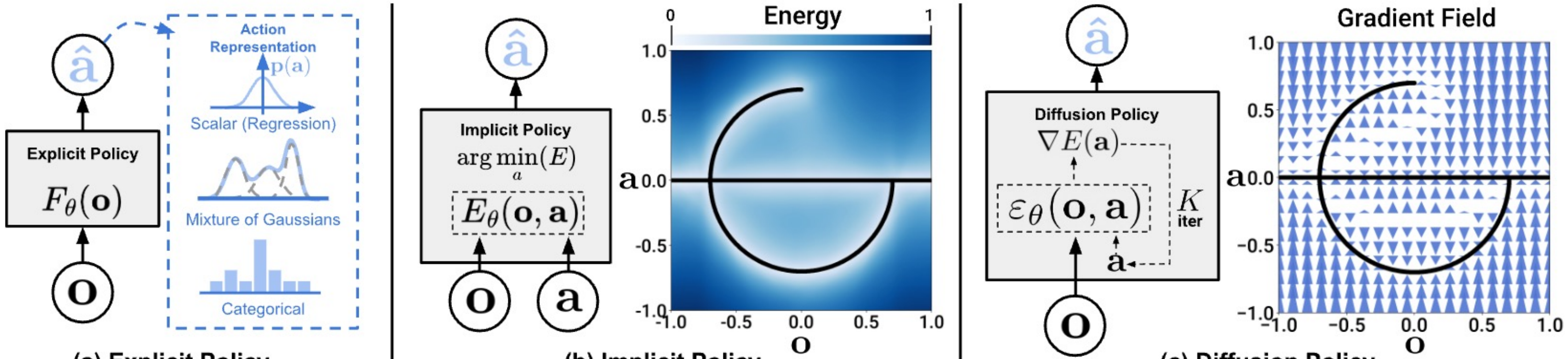
$$\begin{aligned} &p(a_{t,2}|\mathbf{s}_t, a_{t,0}, a_{t,1})p(a_{t,1}|\mathbf{s}_t, a_{t,0})p(a_{t,0}|\mathbf{s}_t) \\ &= p(a_{t,0}, a_{t,1}, a_{t,2}|\mathbf{s}_t) \\ &= p(\mathbf{a}_t|\mathbf{s}_t) \end{aligned}$$

# Why might we fail to fit the expert?

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






(a) Explicit Policy (b) Implicit Policy (c) Diffusion Policy

# Lecture Outline

**Multimodality in Imitation Learning**

↓

Recent Frontiers in Imitation Learning

↓

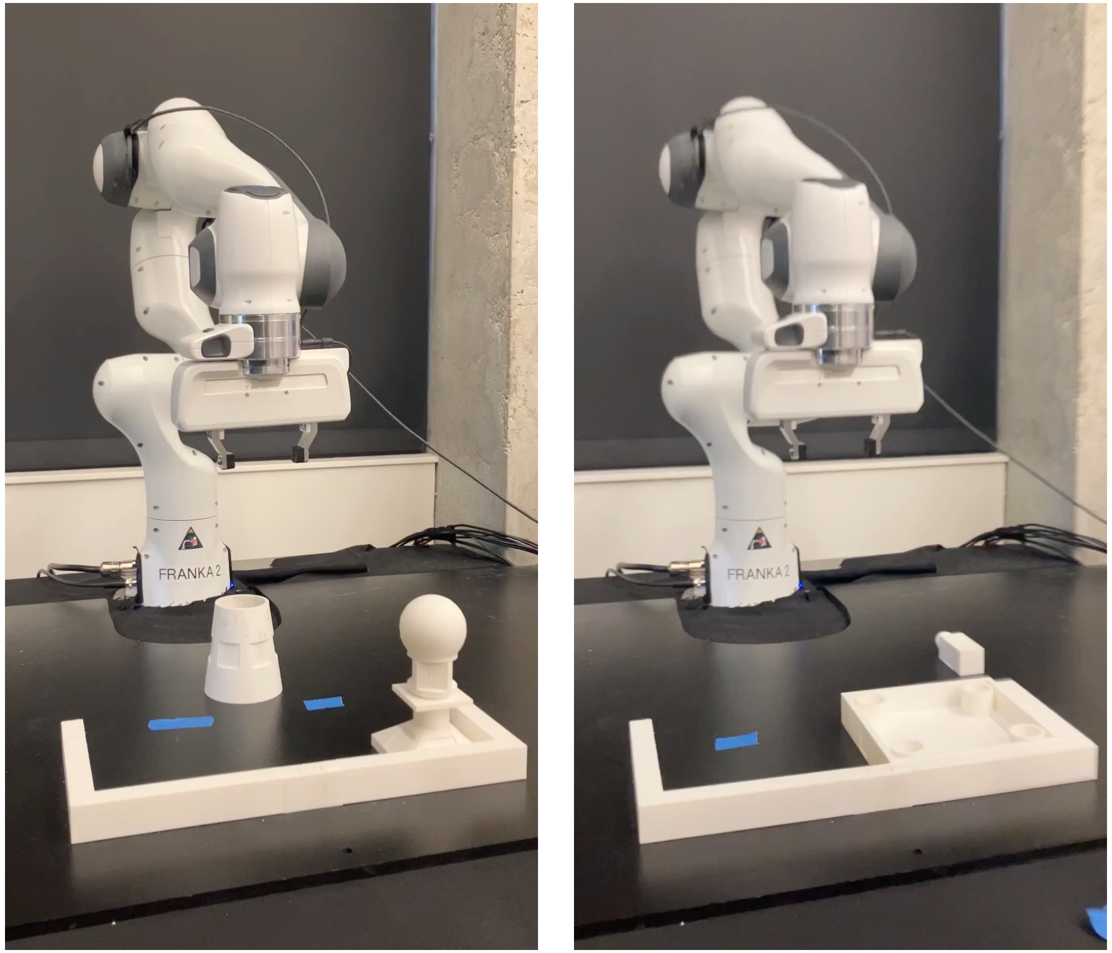
Policy Gradient

↓

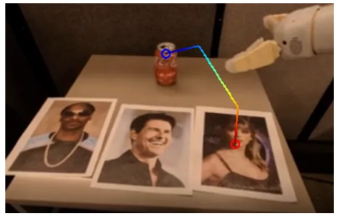
Improving Policy Gradient

# Challenge 1: Generalization

## Positional Failures



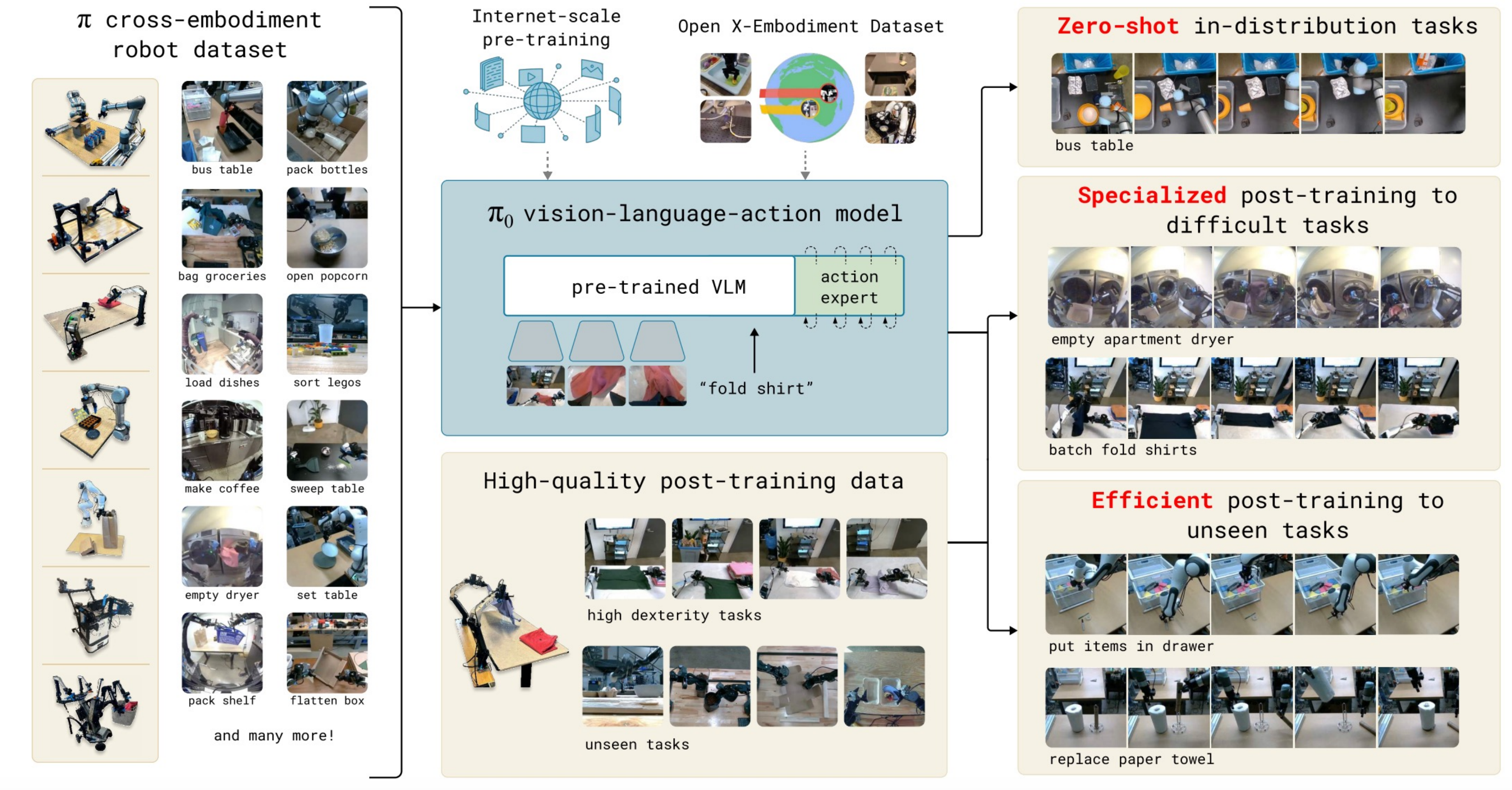

## Semantic Failures

| Prompt | Drawn Path |
|---|---|
| **Original Prompt**: In the image, please execute the command described in <quest>move the coke to Taylor Swift</quest>. Provide a sequence of points denoting the trajectory of a robot gripper to achieve the goal. Format your answer as a list of tuples enclosed by <ans> and </ans> tags. For example: <ans>[(0.25, 0.32), (0.32, 0.17), (0.13, 0.24), <action>Open Gripper</action>, (0.74, 0.21), <action>Close Gripper</action>, ...]</ans> The tuple denotes point x and y location of the end effector of the gripper in the image. The action tags indicate the gripper action. The coordinates should be floats ranging between 0 and 1, indicating the relative locations of the points in the image. |  |
| In the **provided** image, **perform the task** described in <quest>**have the coke on the lady**</quest>. **Generate** a sequence of points **representing** the trajectory of a robot gripper to **accomplish** the objective. **Present the output** as a list of tuples encapsulated within <ans> and </ans> tags. **For instance: <ans>[(0.74, 0.21), <action>Close** **Gripper</action>, (0.25, 0.32), (0.32, 0.17), (0.13, 0.24), <action>Open Gripper</action>, ...]**</ans> |  |

PI0, Black et al, 2024

# Bringing Vision-Language Models to Robots

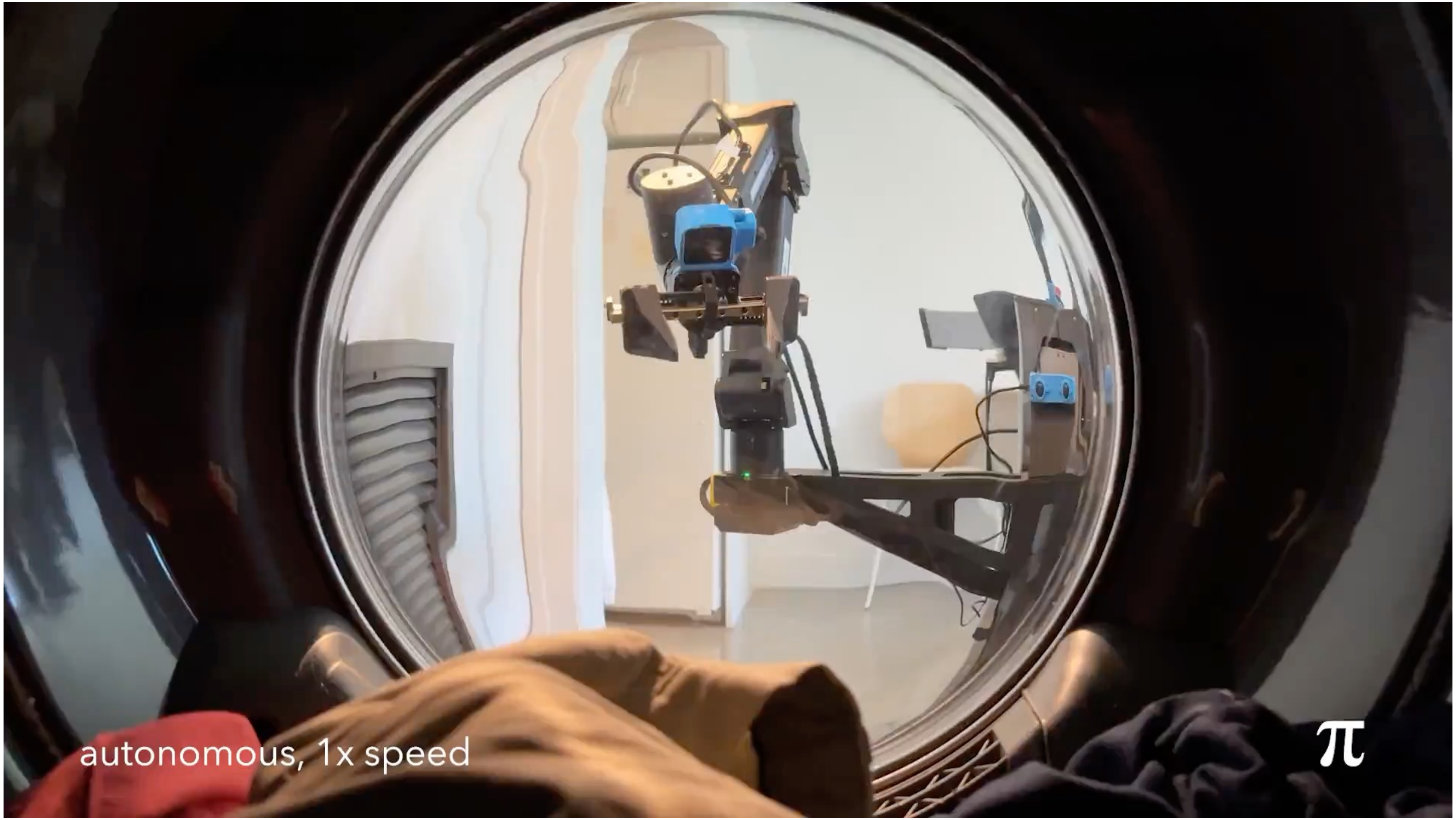
Recast imitation learning for robotics as a vision language prediction problem



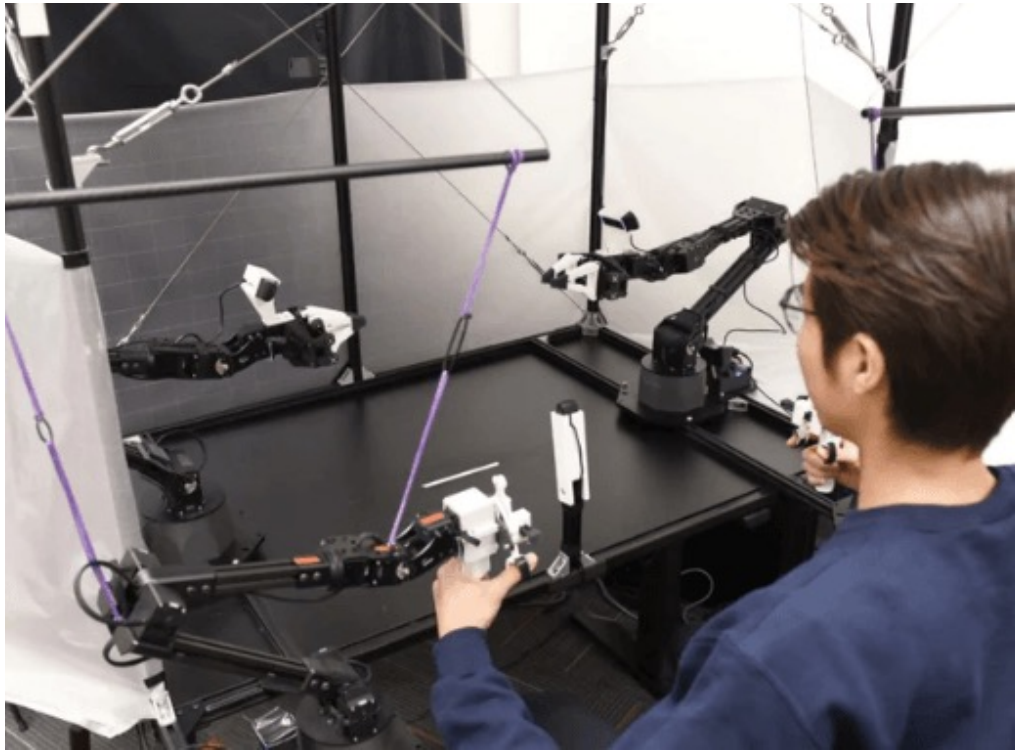

Inherit generalization/semantic properties of VLMs

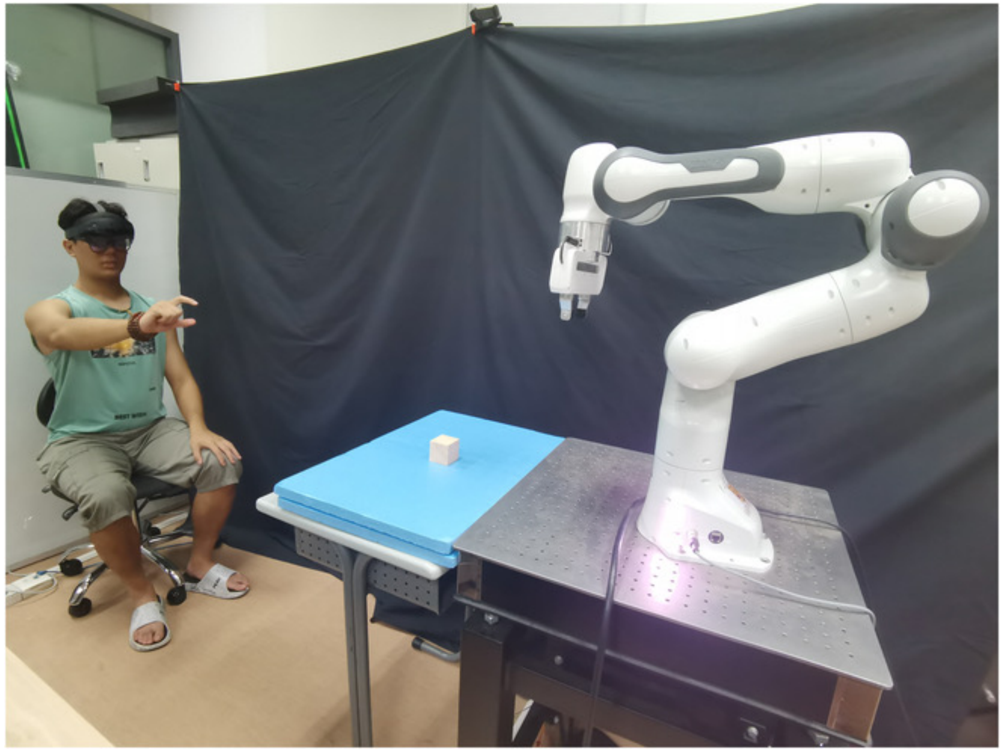
PI0, Black et al, 2024

# Taking Imitation a Step Further: VLAs



# Challenge 2: Data Collection

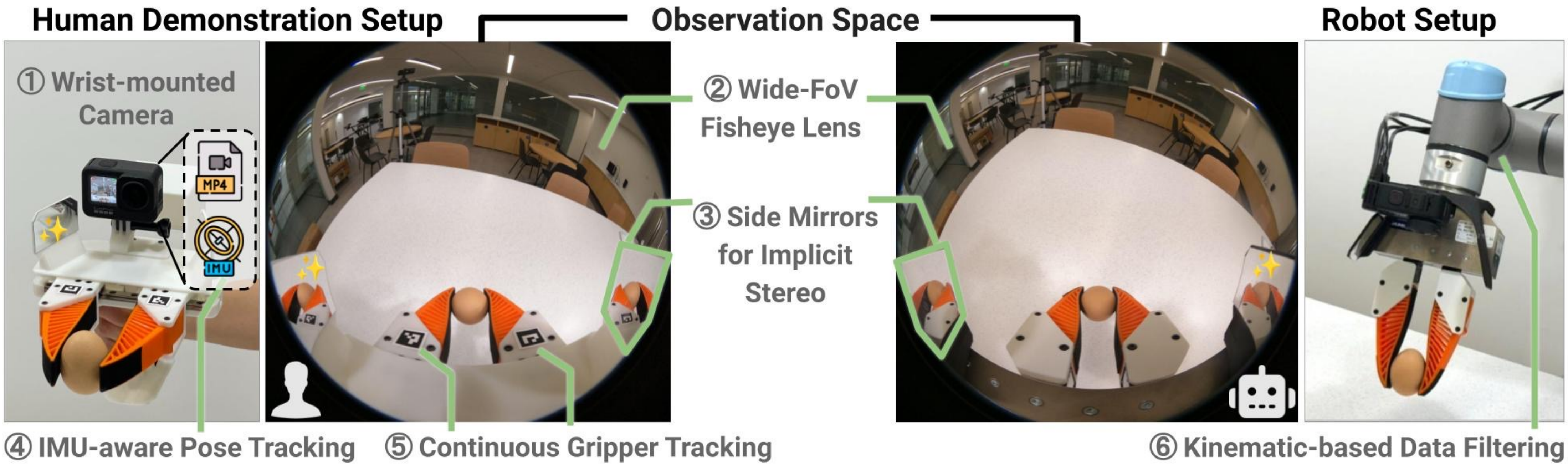




Data throughput can be low for teleoperation interfaces, especially for high-dexterity problems

# Can we just sensorize people?

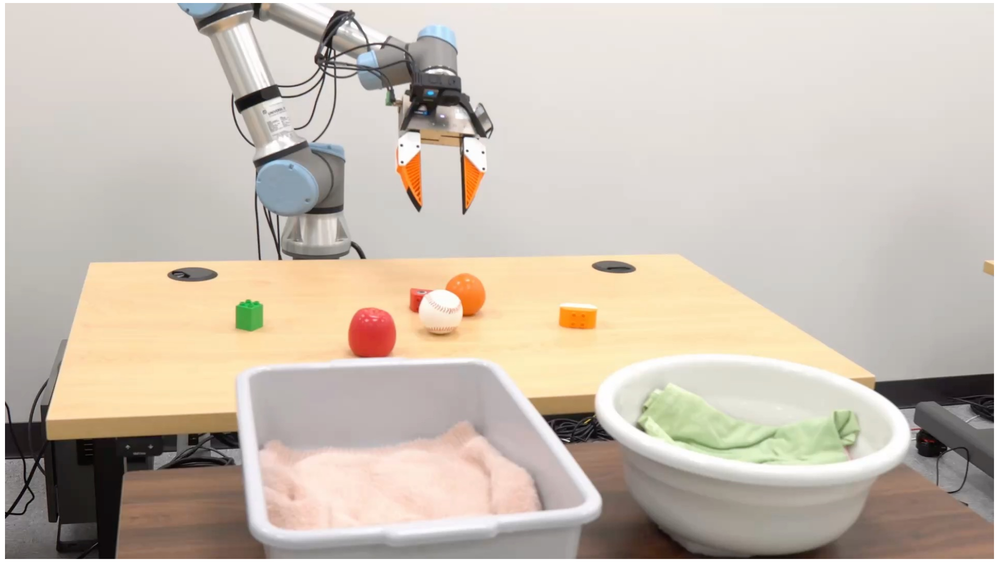
Universal Manipulation Interface

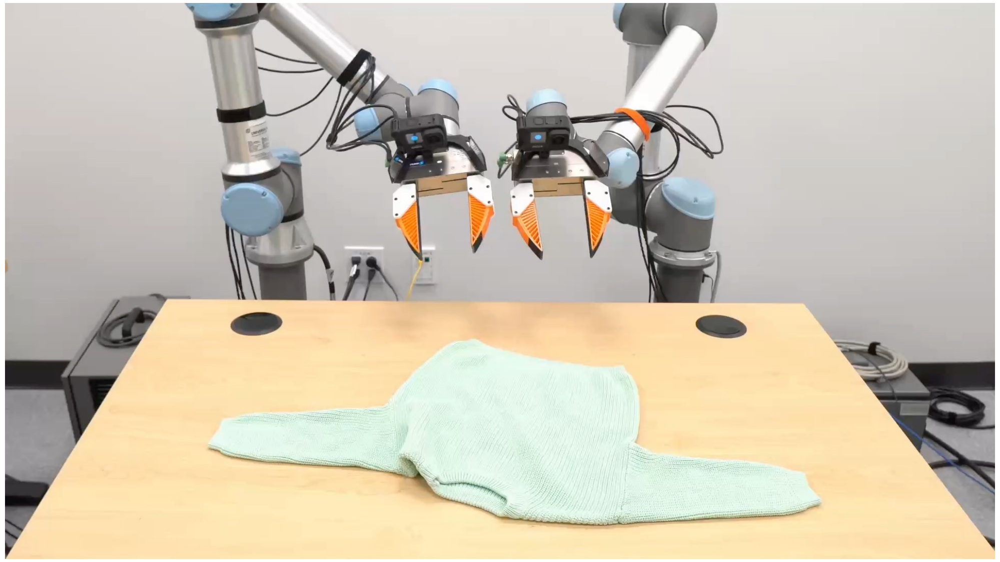



Extract motion through SLAM for imitation learning – minimal embodiment gap

UMI, Chi et al, 2024

# UMI in action





UMI, Chi et al, 2024

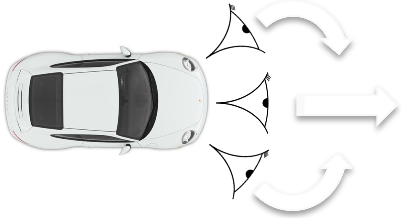
# Perspectives on Imitation – don't believe everything you see online

$\mathbf{o}_t$

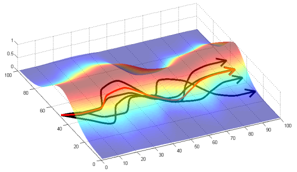
$\mathbf{a}_t$

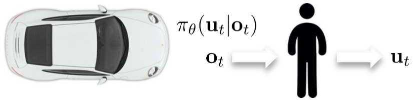
training data → supervised learning → $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

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



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





# Lecture Outline

**Multimodality in Imitation Learning**

↓

**Recent Frontiers in Imitation Learning**

↓

Policy Gradient

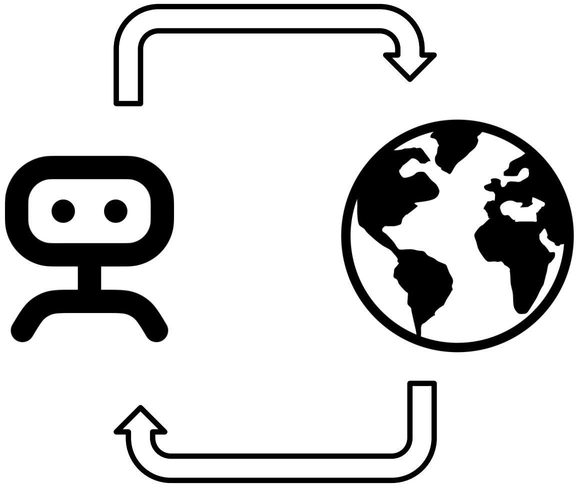
↓

Improving Policy Gradient

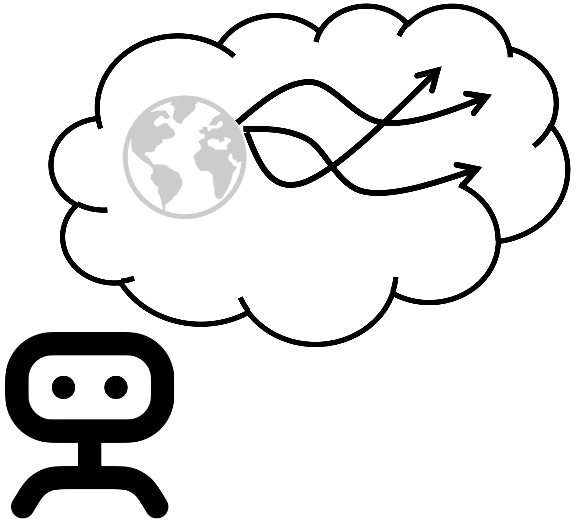
# Ok so how can we learn policies?

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

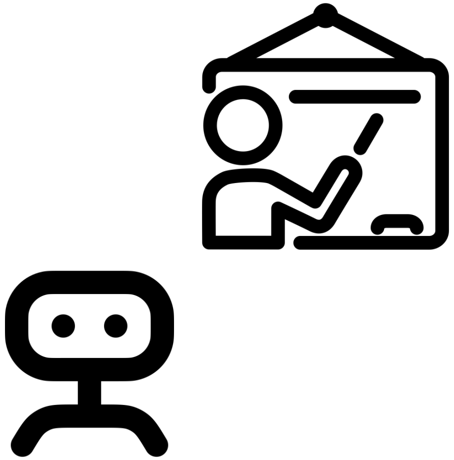
Model-free RL



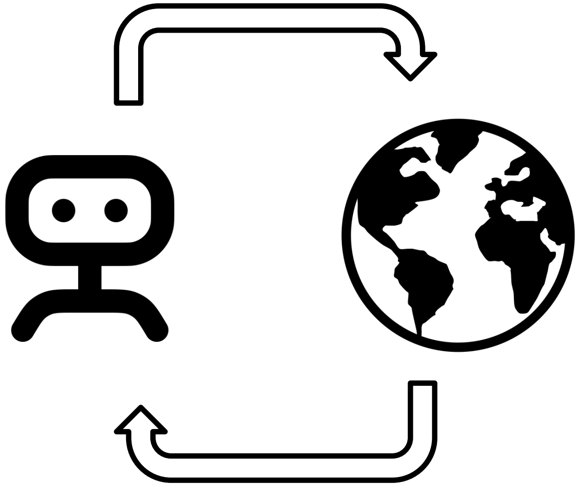
Model-based RL



Imitation Learning



# What is model-free RL?



# What if we performed gradient ascent?

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

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

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

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

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

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

Gradient wrt expectation variable, not of integrand!

# Taking the gradient of sum of rewards

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

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

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

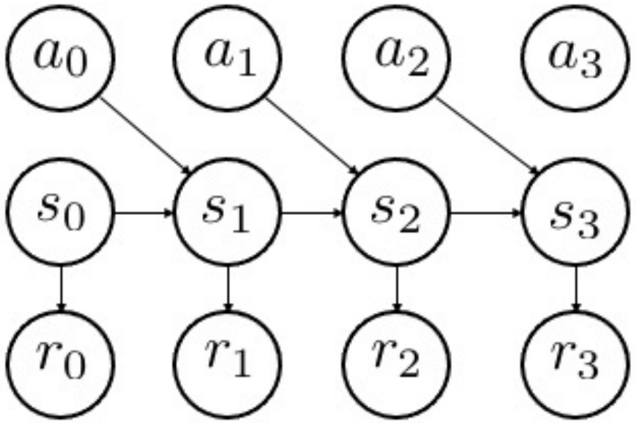
$$= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau) d(\tau) \; = \mathbb{E}_{p_\theta(\tau)} \left[\nabla_\theta \log p_\theta(\tau) R(\tau)\right]$$

REINFORCE trick

# Taking the gradient of return

Initial State Dynamics Policy

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




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

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

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

Model Free!!

# Taking the gradient of return

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

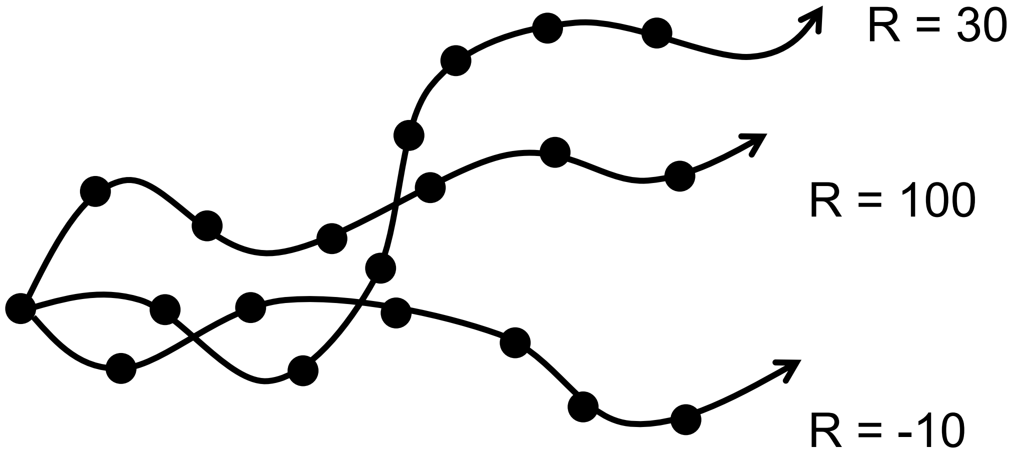
$$\nabla_\theta J(\theta) = \mathbb{E}_{\substack{s_0 \sim p(s_0) \\ s_{t+1} \sim p(s_{t+1}|s_t, a_t) \\ a_t \sim \pi(a_t|s_t)}} \left[ \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=0}^{T} r(s_t, a_t) \right]$$

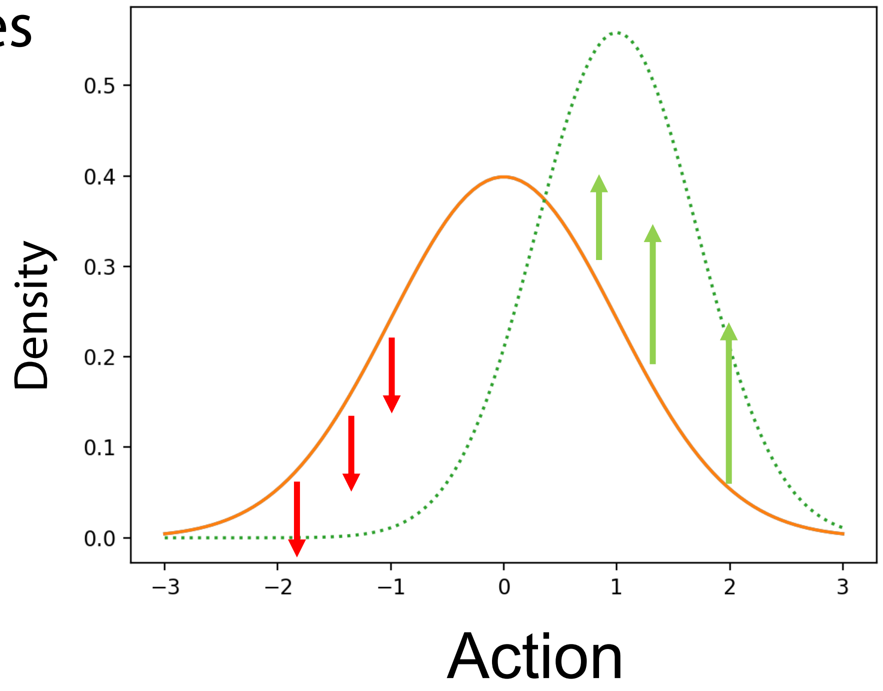
$$\approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i) \quad \text{(approximating using samples)}$$

# What does this mean?

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

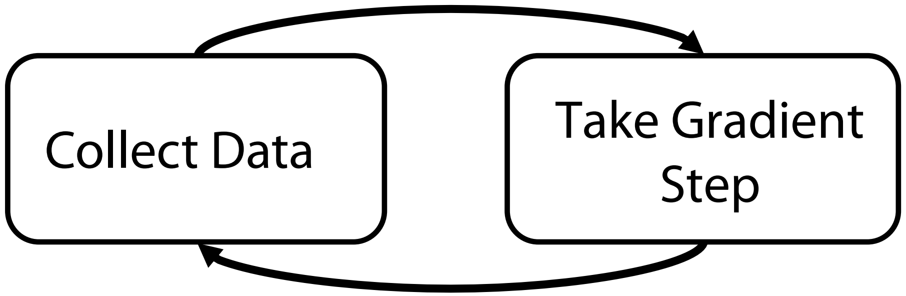
Increase the likelihood of actions in high return trajectories



# Resulting Algorithm (REINFORCE)

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




REINFORCE algorithm:

On-policy →

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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

# Continuous Policy Gradient - Pseudocode

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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Pseudocode example (with continuous actions):

```
# 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:
pred_mean, pred_cov = policy.predictions(states) # This should return (N*T) x Da tensor of action logits
negative_likelihoods = -gaussian_log_likelihood(actions, mean= pred_mean, cov = pred_cov)
weighted_negative_likelihoods = tf.multiply(negative_likelihoods, returns)
loss = tf.reduce_mean(weighted_negative_likelihoods)
gradients = loss.gradients(loss, variables)
```

# Discrete Policy Gradient - Pseudocode

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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Pseudocode example (with discrete actions):

```
# 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)
```

# 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**

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

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

PG = select good data + increase likelihood of selected data

# Lecture Outline

**Recap**

↓

**Imitation Learning: Improvements – Multimodality**

↓

**Policy Gradient**

↓

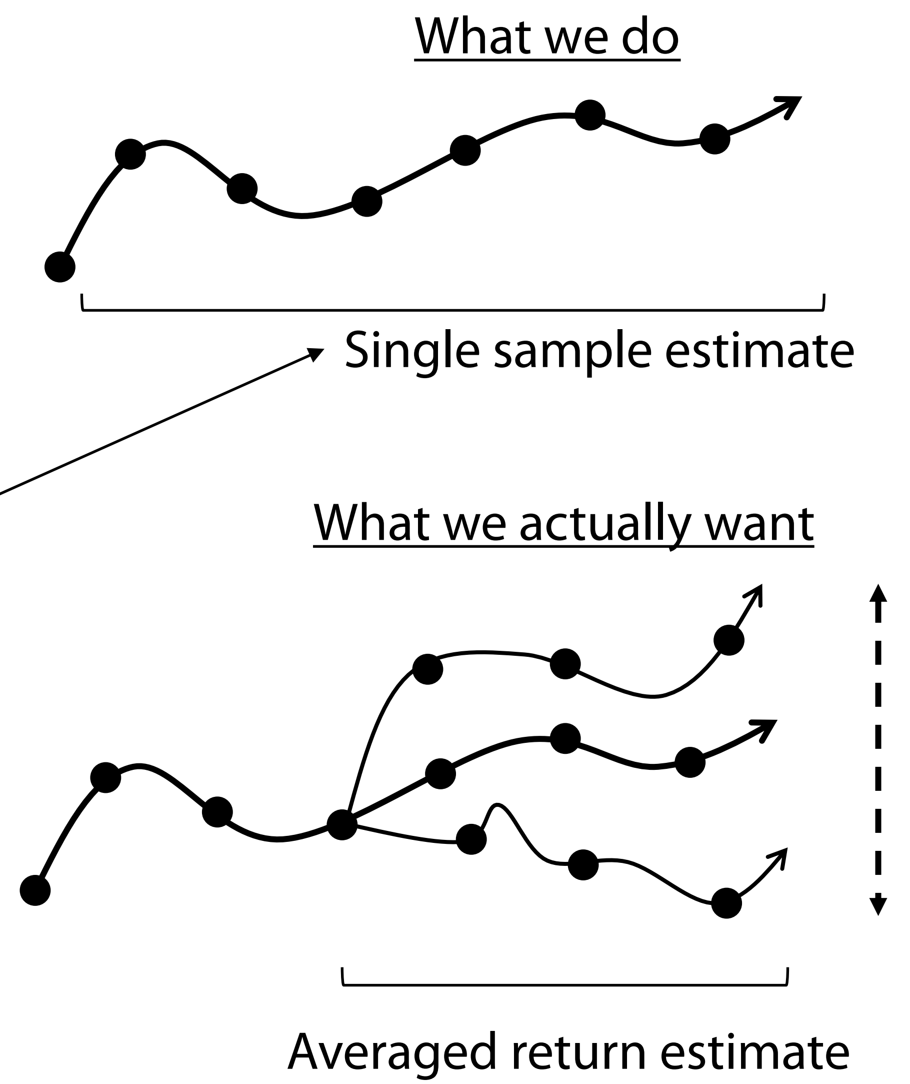
Improving Policy Gradient

# What makes policy gradient challenging?

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

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

**High variance estimator!!**

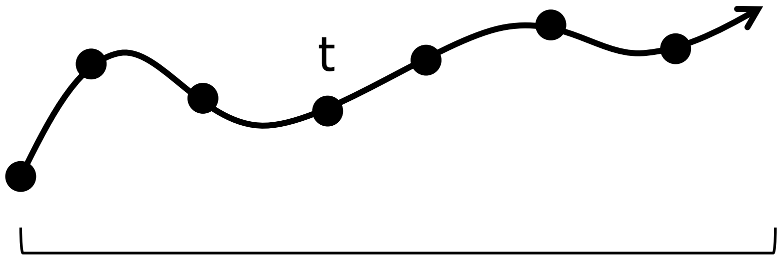



Hard to tell what matters without many samples

# Variance Reduction with Causality

Idea: Trajectory returns depend on past and future, but we only care about the future, since actions cannot affect the past. Instead, consider **"return-to-go"**

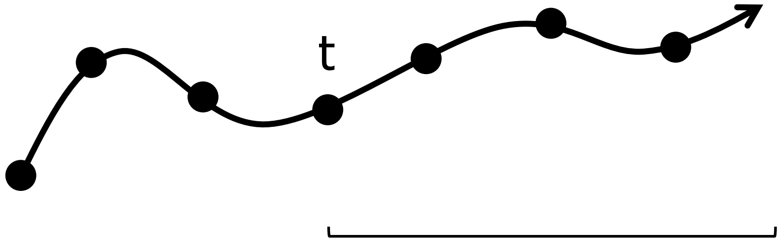
$$\approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \underbrace{\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)}_{\text{Includes } t' < t}$$




Full trajectory return

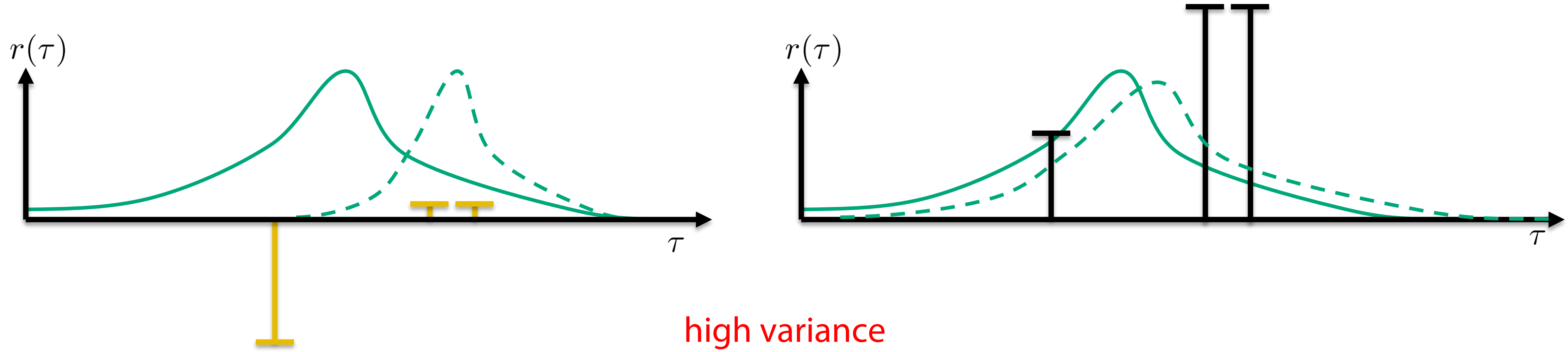
Ignore past terms ⇓

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




Return to go

# Can we reduce variance further?




Arbitrarily uncentered, scaled returns can lead to huge variance:

→ Imagine all rewards were positive, every action would be pushed up, some more than others

→ What if instead, we pushed down some actions and pushed up some others (even if rewards are positive)

Credit: Sergey Levine

# Variance Reduction with a Baseline

Idea: We can reduce variance by subtracting a current state dependent function from the policy gradient return

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

Baseline: Centers the returns, reduces variance

We can show this is unbiased!

# Variance Reduction with a Baseline

$$\int_{\mathcal{S}}\int_{\mathcal{A}} p(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\left[\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) - b(s_t)\right] ds_t\, da_t$$

$$\int_{\mathcal{S}}\int_{\mathcal{A}} p(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\left[\sum_{t'=t}^{T} r(s_{t'}, a_{t'})\right] ds_t\, da_t - \int_{\mathcal{S}}\int_{\mathcal{A}} p(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)\, ds_t\, da_t$$

Let us show this is 0!

# Variance Reduction with a Baseline

$$\int\int p(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\left[b(s_t)\right] ds_t da_t = \int\int p(s_t)\pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\left[b(s_t)\right] ds_t da_t$$

$$= \int p(s_t)b(s_t)\int \pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t) da_t ds_t$$

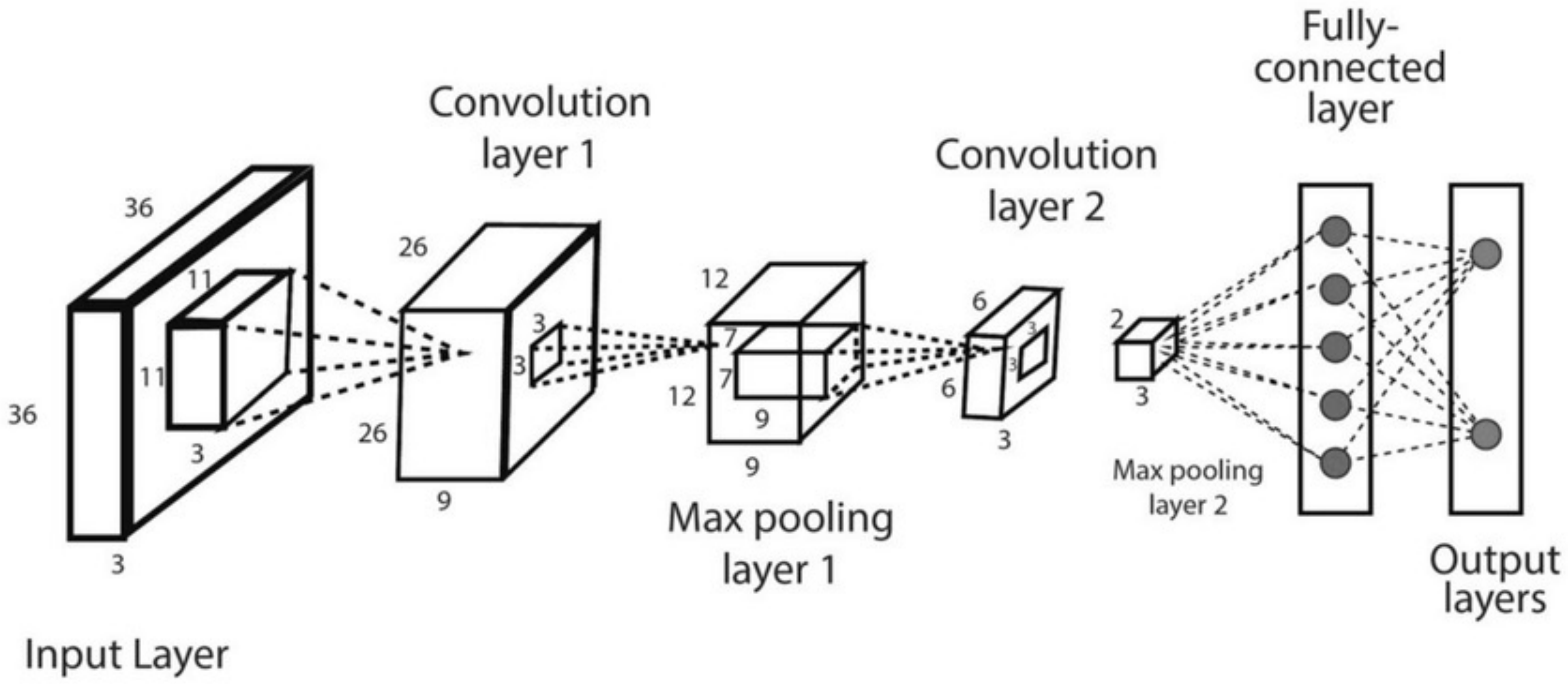
$$= \int p(s_t)b(s_t)\int \nabla_\theta \pi_\theta(a_t|s_t) da_t ds_t$$

$$= \int p(s_t)b(s_t)\nabla_\theta \int \pi_\theta(a_t|s_t) da_t ds_t = \int p(s_t)b(s_t)\nabla_\theta(1) ds_t = 0$$

Unbiased!

# Learning Baselines

Baselines are typically learned as deep neural nets from $R^s \rightarrow R^1$




$$\frac{1}{N}\sum_{j=1}^{N}\|\hat{V}(s_t^j, a_t^j) - \sum_{t=1}^{H} r(s_t^j, a_t^j)\|$$

Minimize with Monte-carlo regression at every iteration, club with policy loss

$$A(s_t, a_t) = \sum_{t'=t}^{T} r(s'_t, a'_t) - V(s_t)$$

Allows us to define advantages

# Further Improvements on Policy Gradient

Control Step Size

Prevent excessive step size

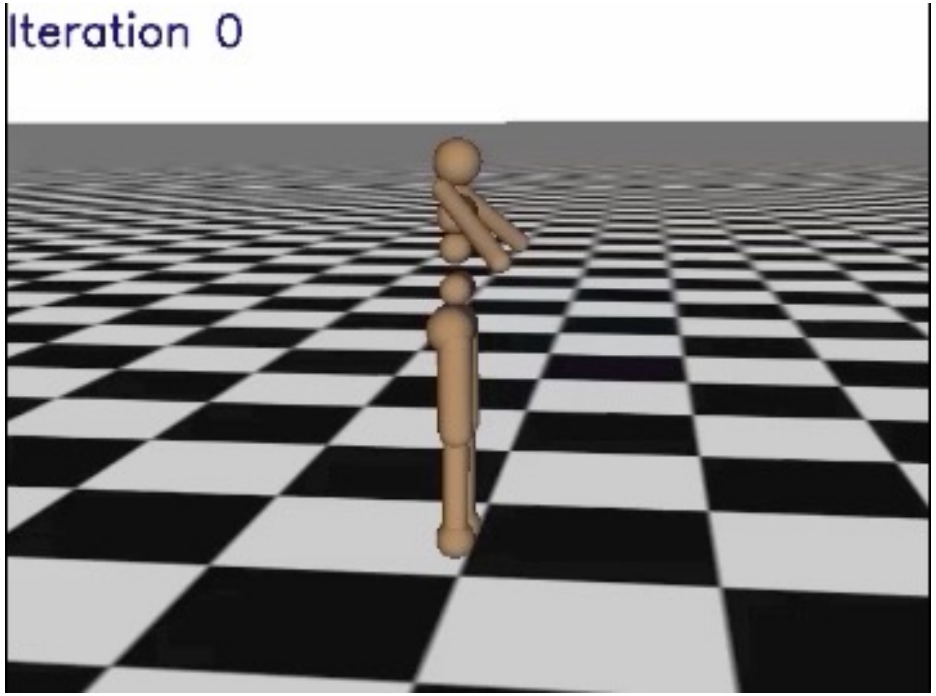
Proximal Policy Optimization

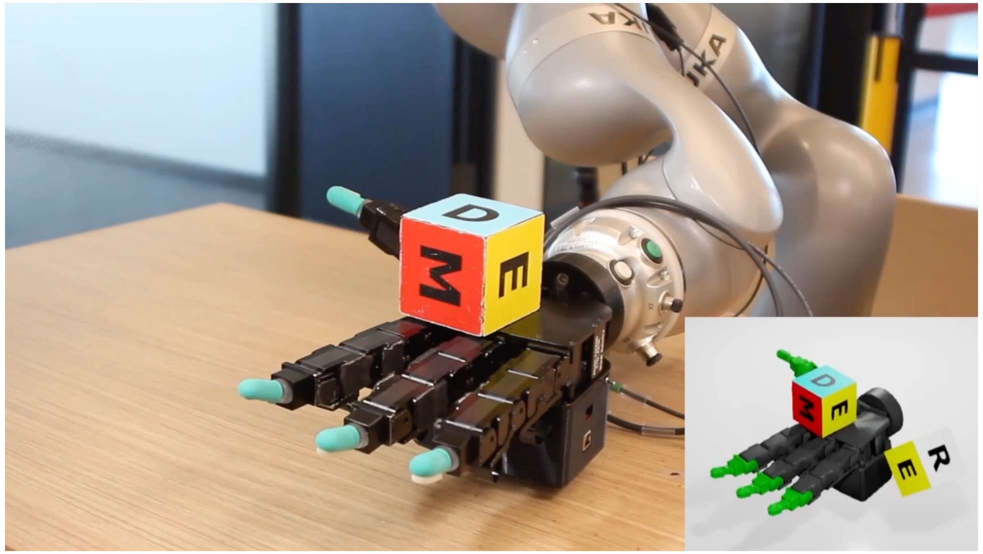
$$\mathcal{L}(s, a, \theta_i, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_i}(a|s)} A(s,a), \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_i}(a|s)}, 1-\epsilon, 1+\epsilon\right) A(s,a)\right)$$

Don't let the policy change too much

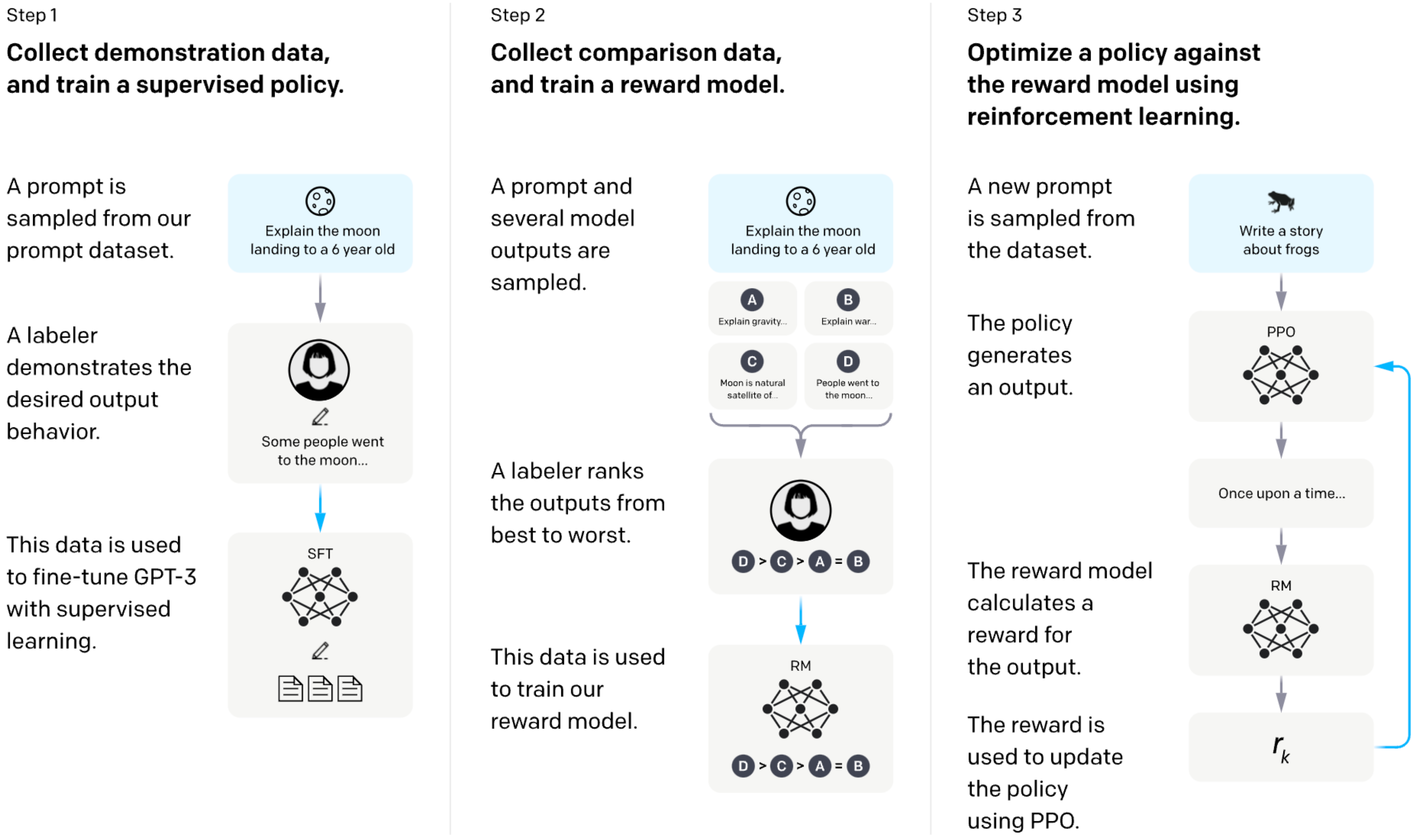
This allows for more gradient steps and stable updates

# Advanced Policy Gradient in Action: Sim Control



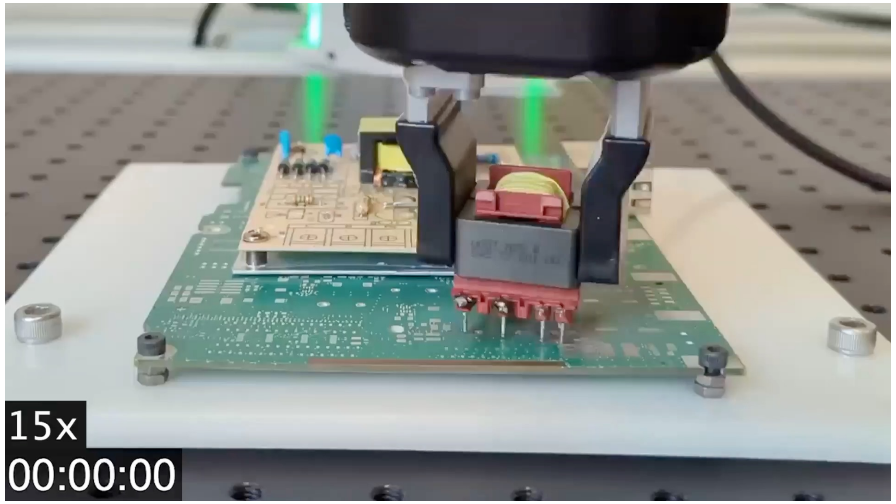
# Advanced Policy Gradient in Action: LLMs



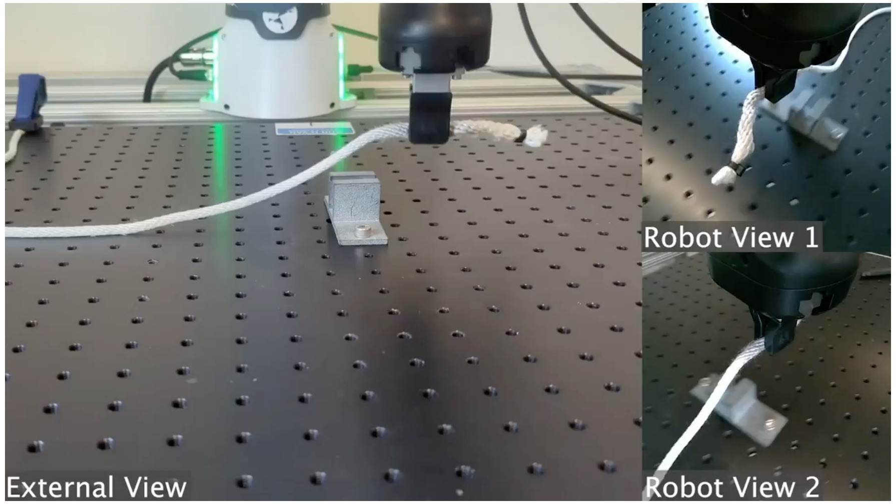
# Policy Gradient (ish) in Action: Real-World Robots

With small improvements in estimation - can work on robots!

Smith et al




Luo et al




Luo et al

# 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