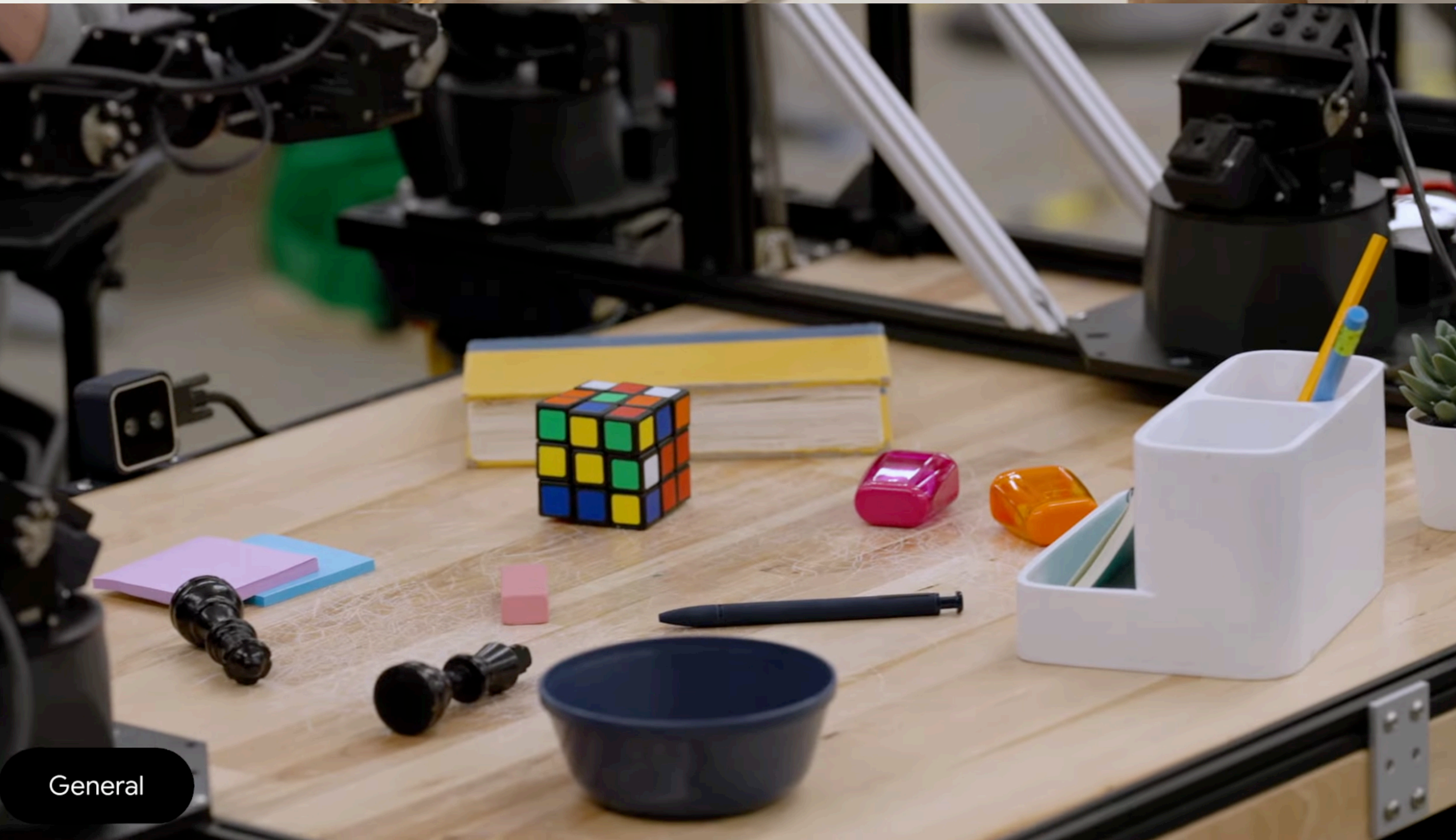
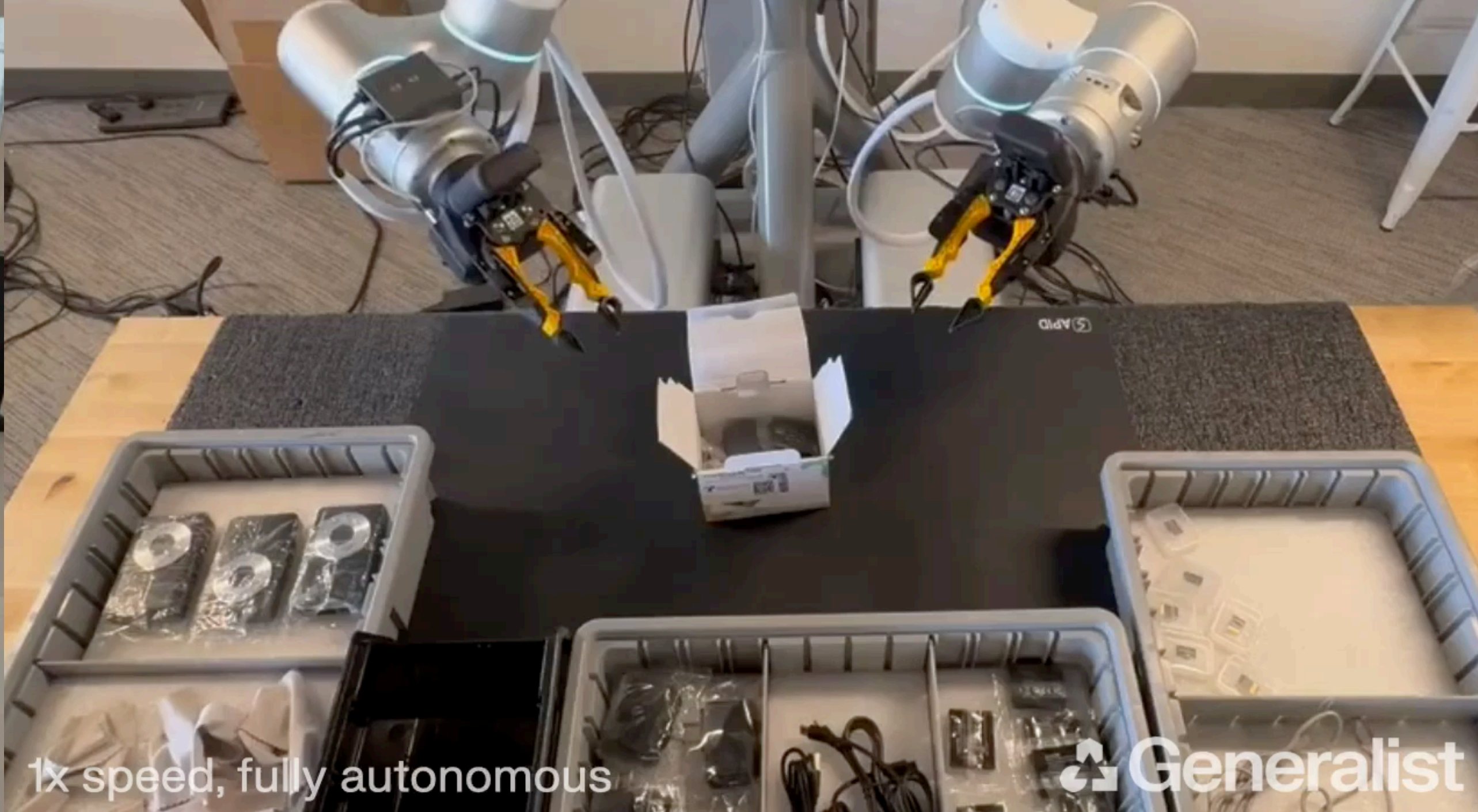


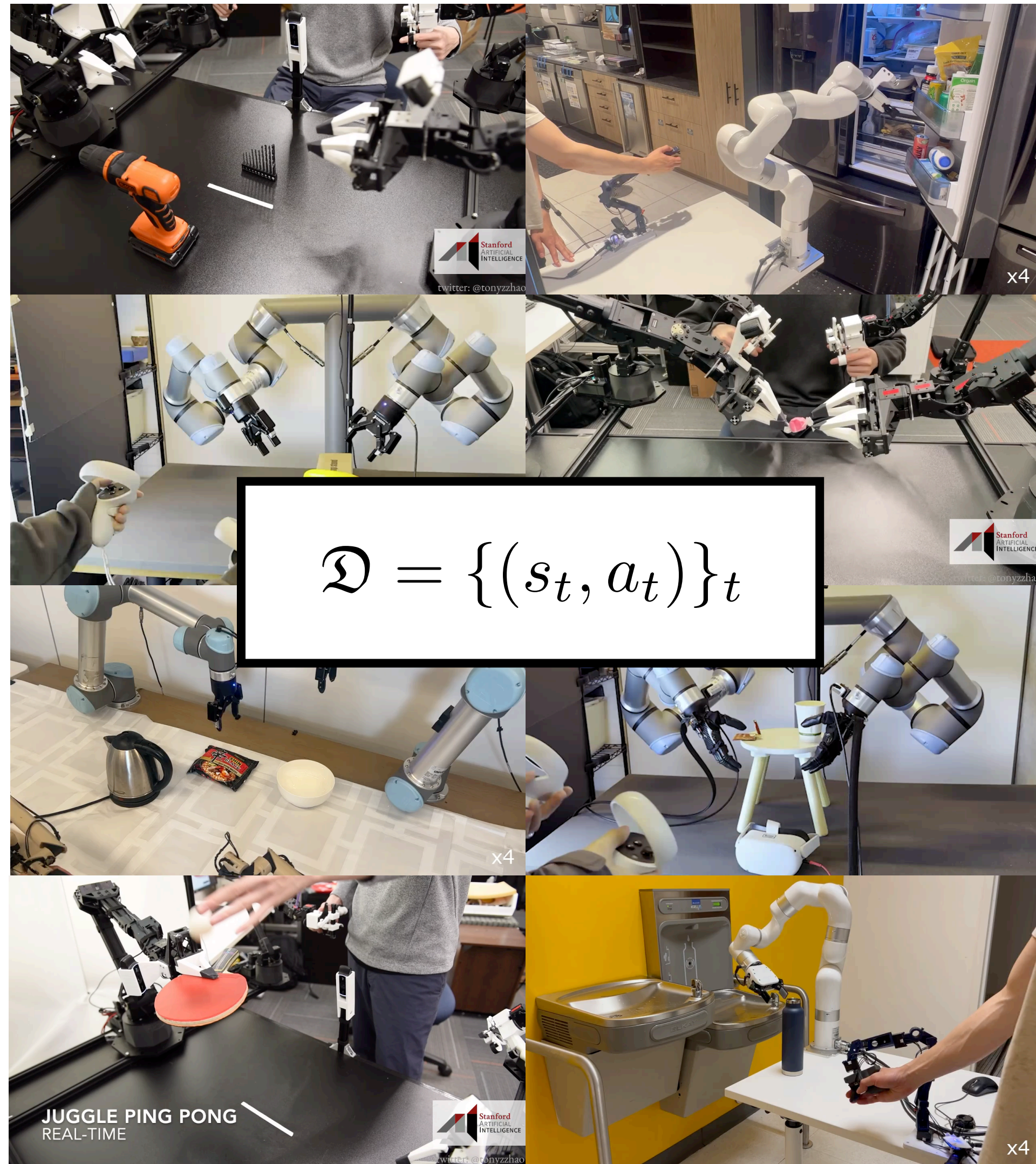
Towards Practical Online Improvement of Pretrained Policies for Robotic Manipulation

Andrew Wagenmaker
UC Berkeley



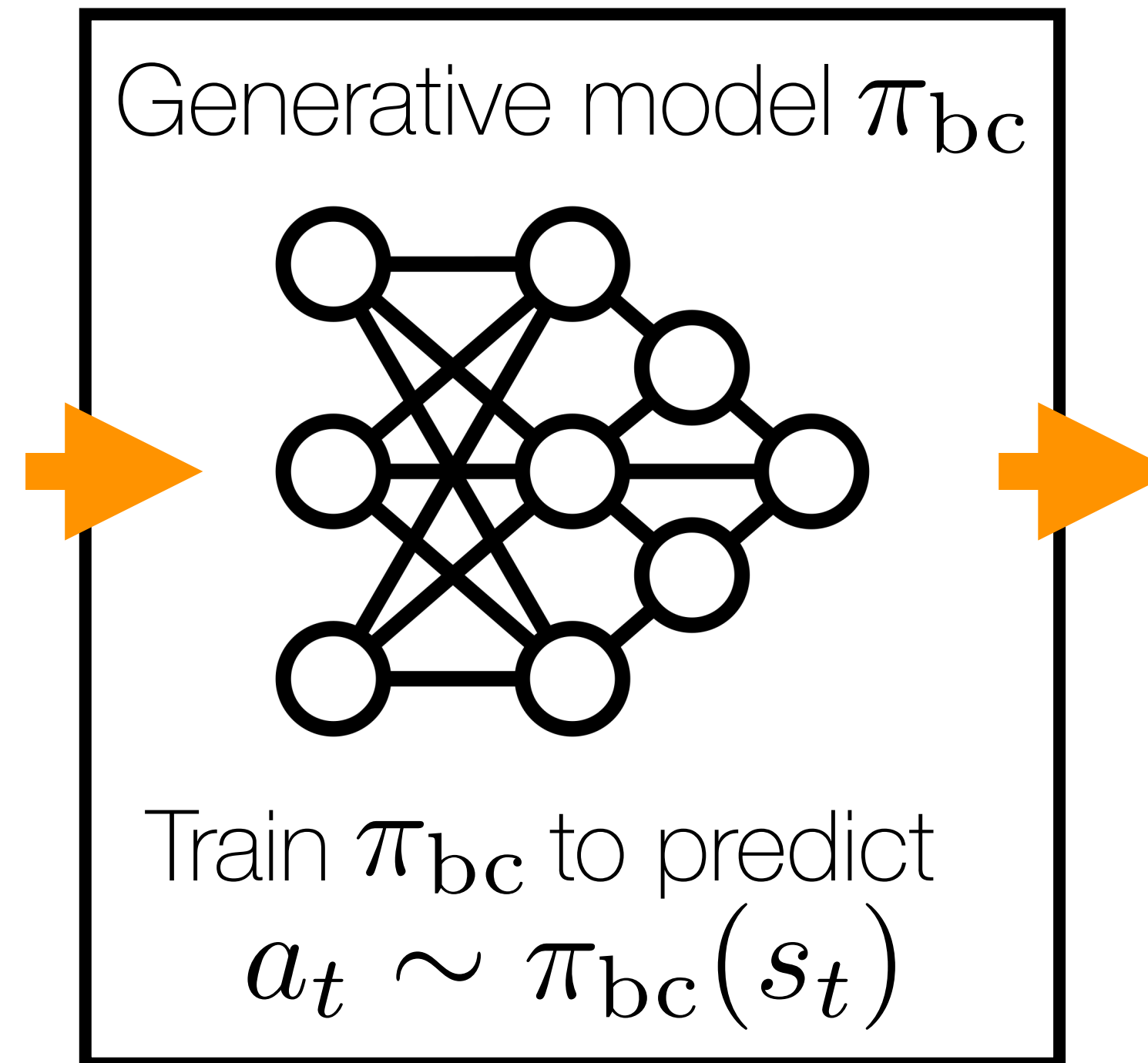


Human demonstrations + behavioral cloning in robotics



Human demonstrations

Reasonable but
not good enough

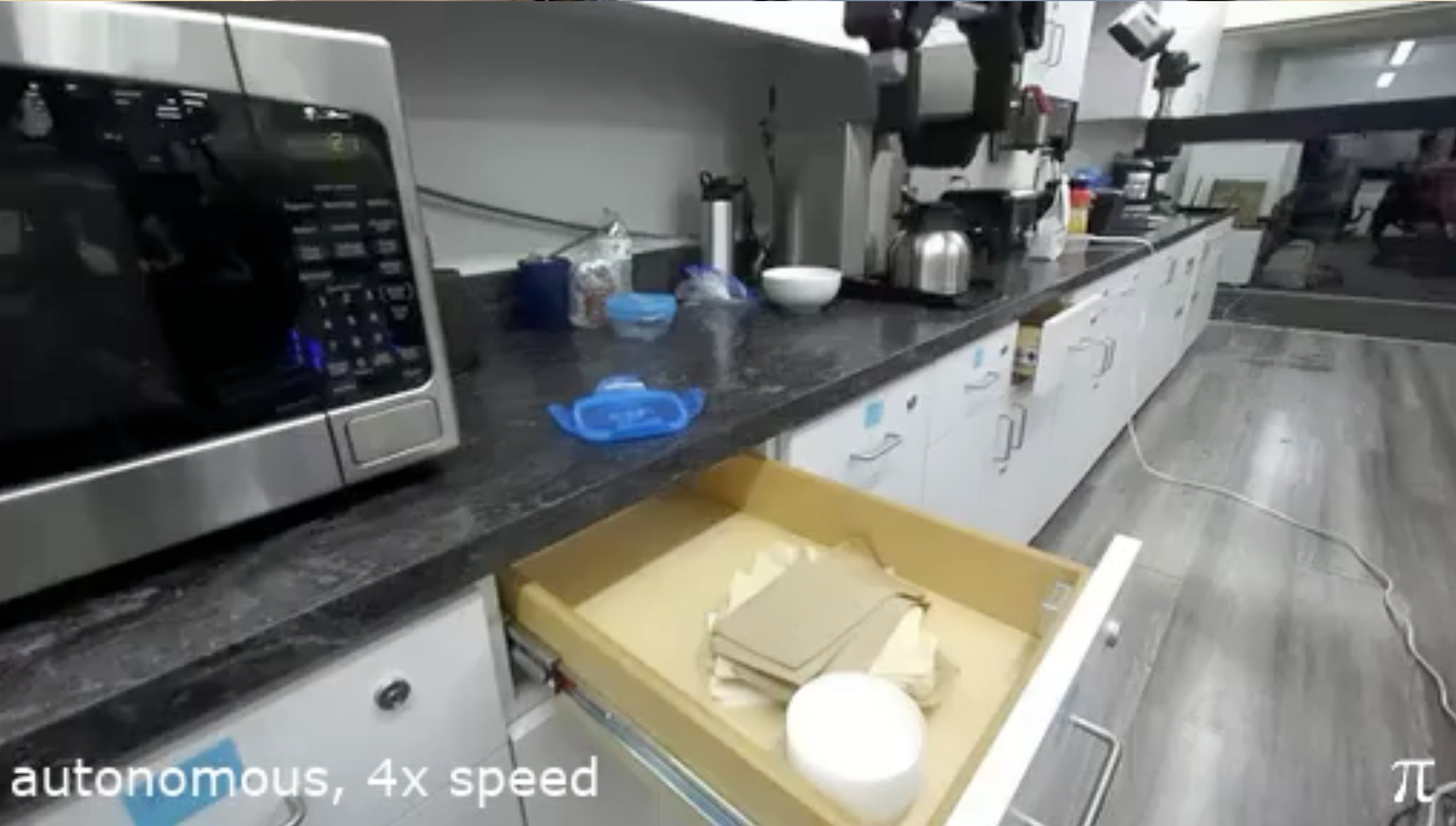
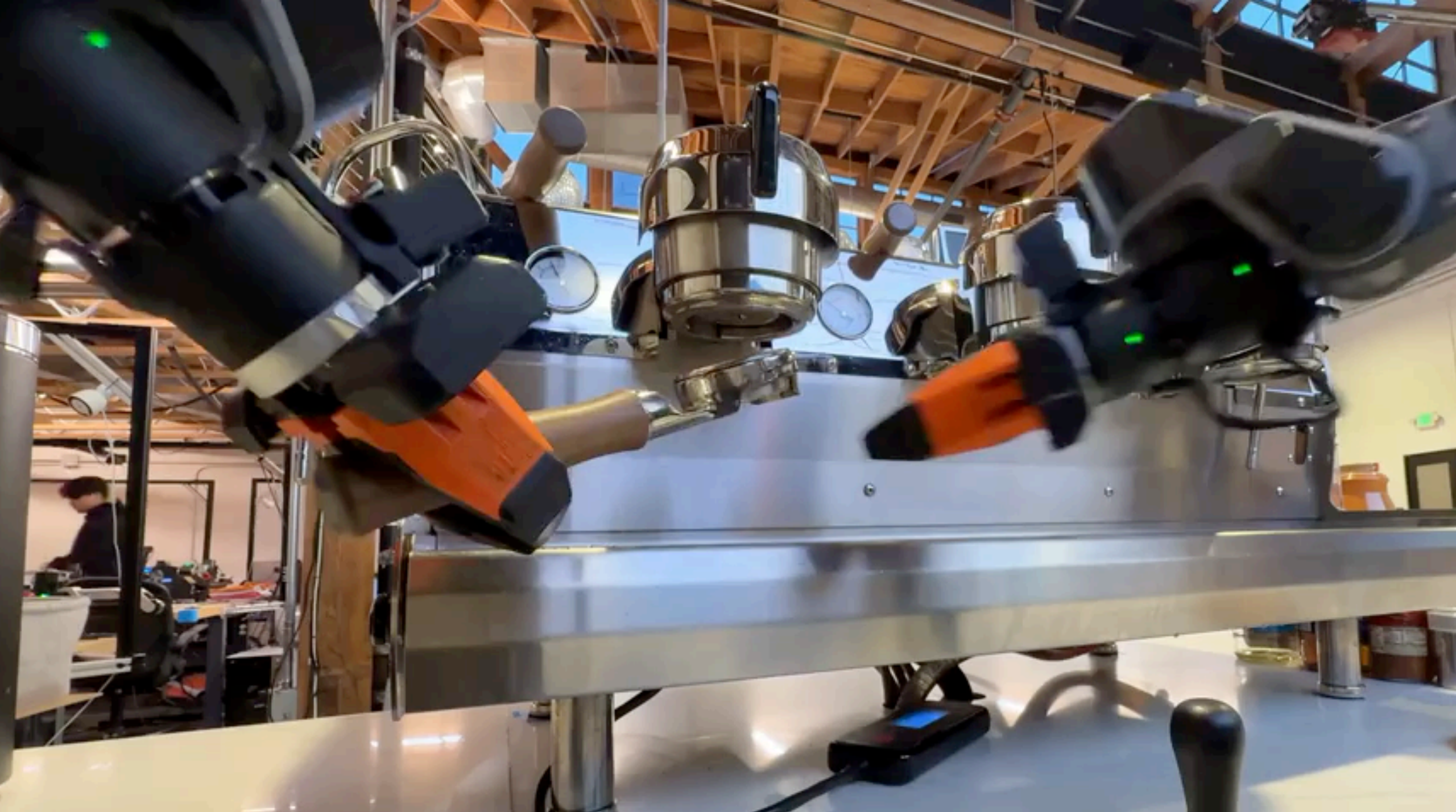


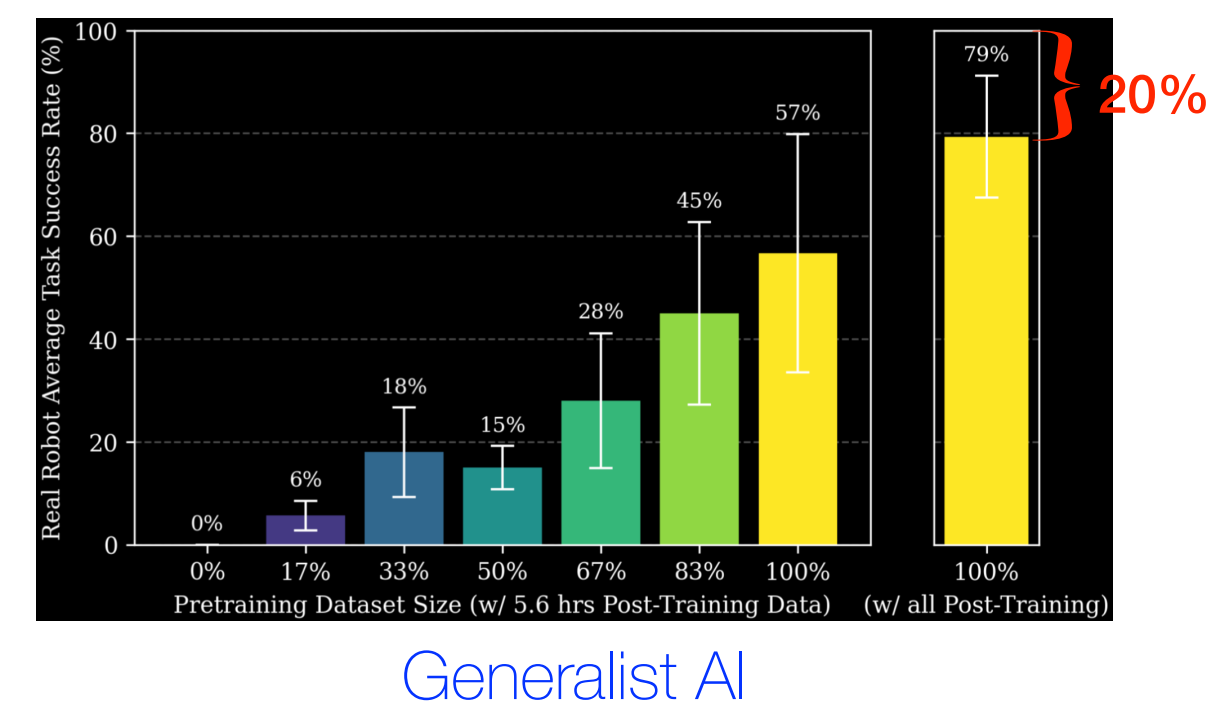
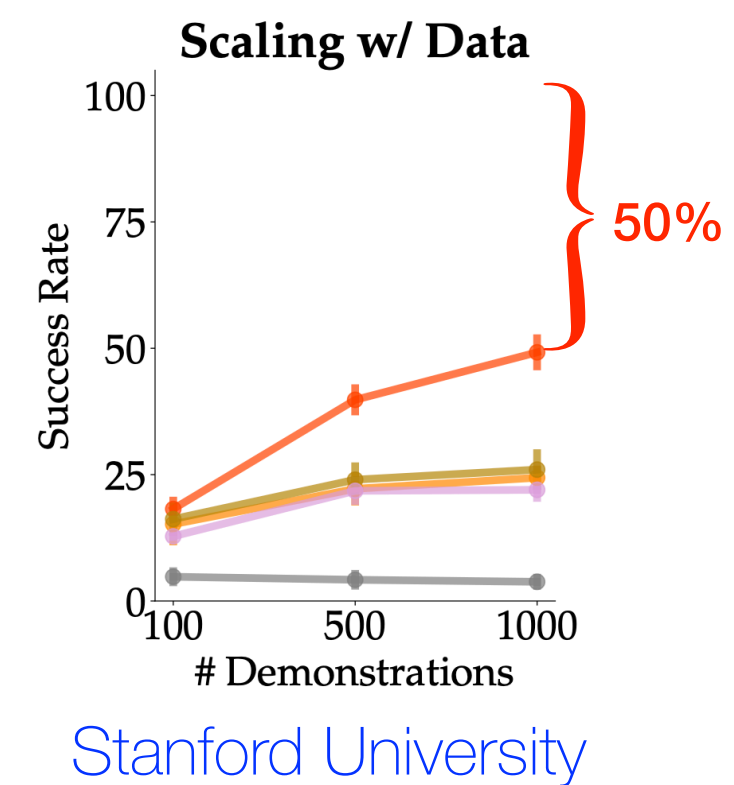
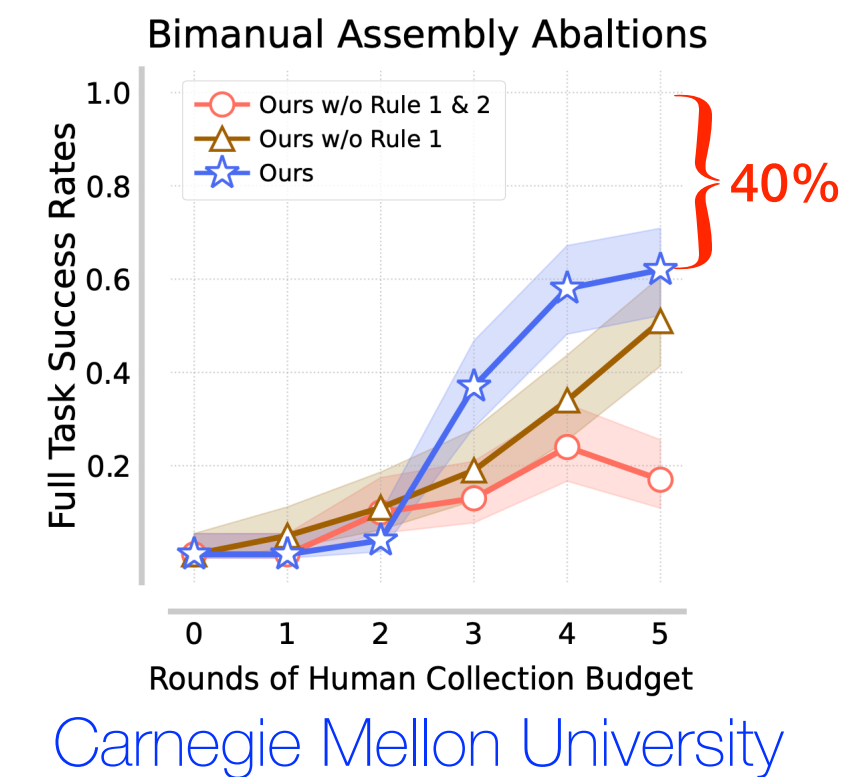
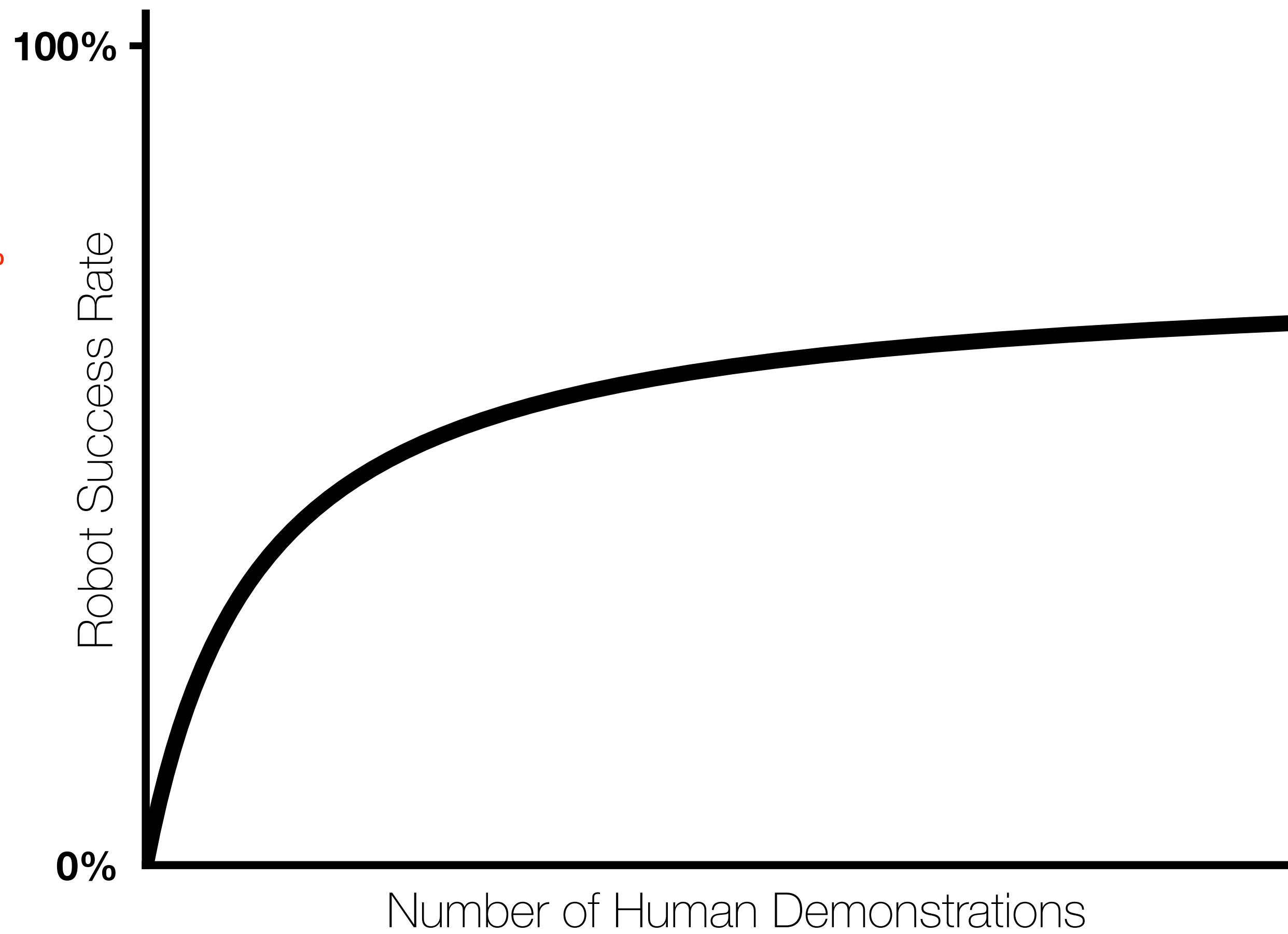
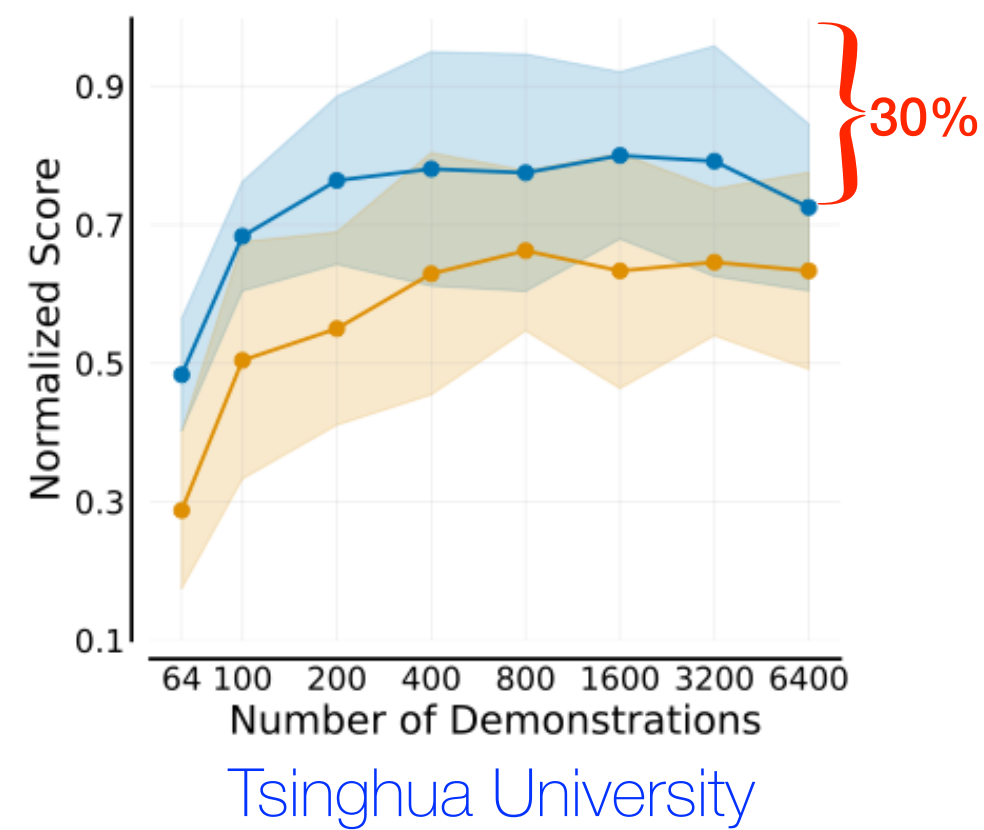
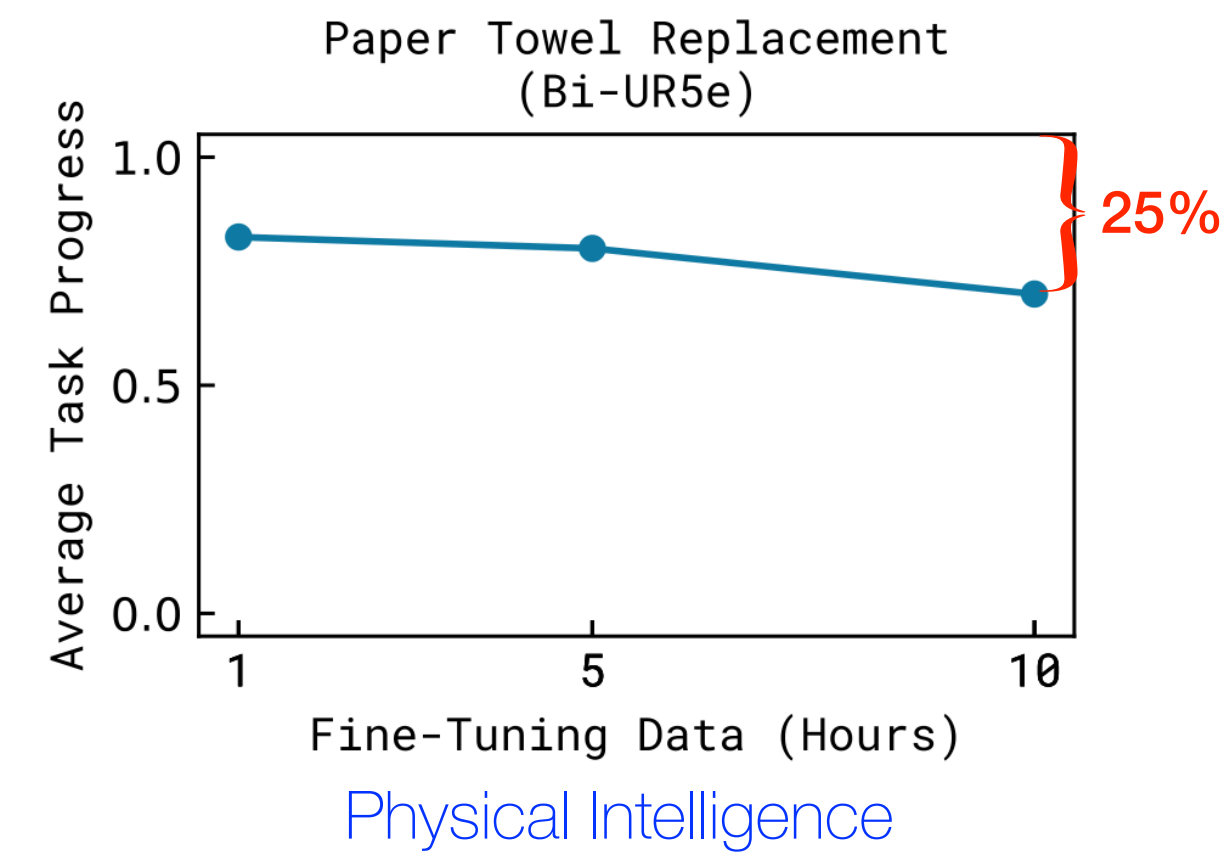
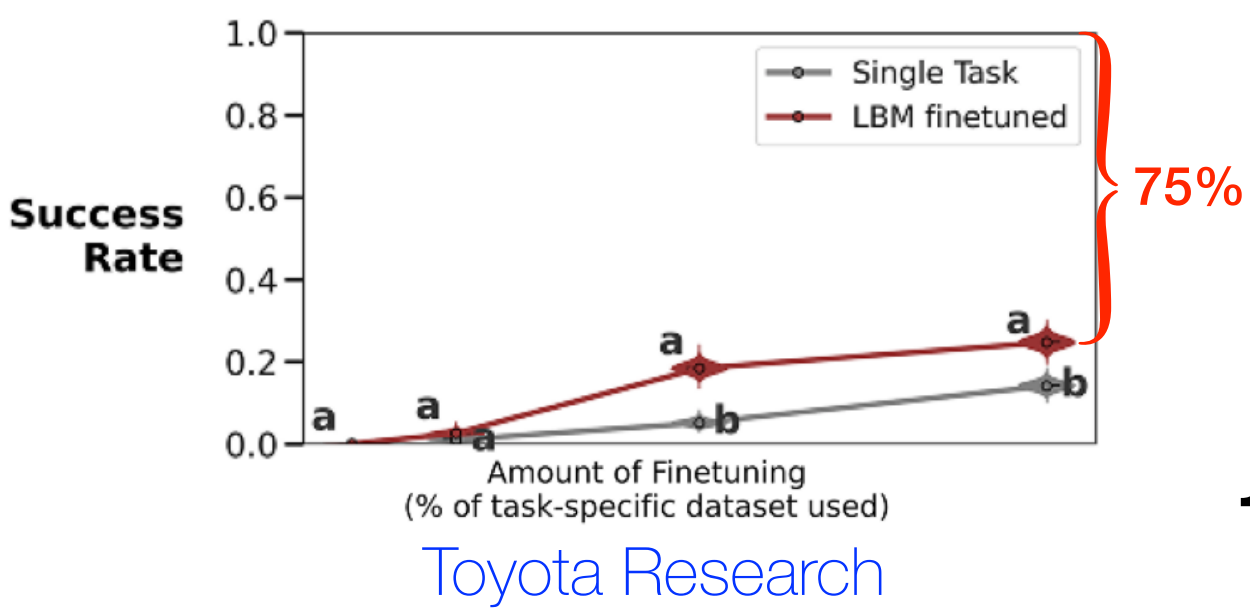
Behavioral Cloning (BC)

[ZKLF RSS '23; WSYLA IROS '23, LZLQYLM ICRA '24]

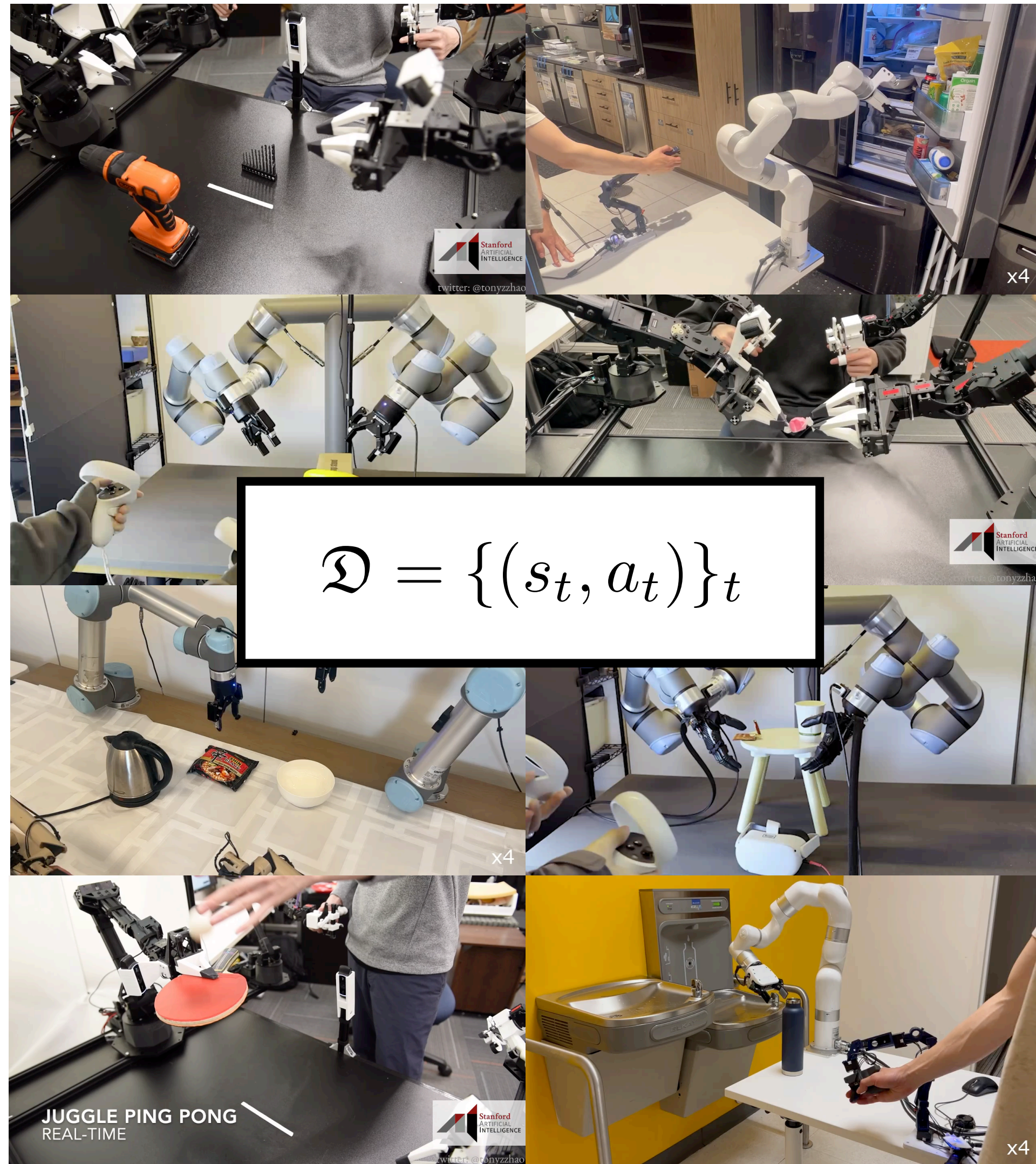


Deployment



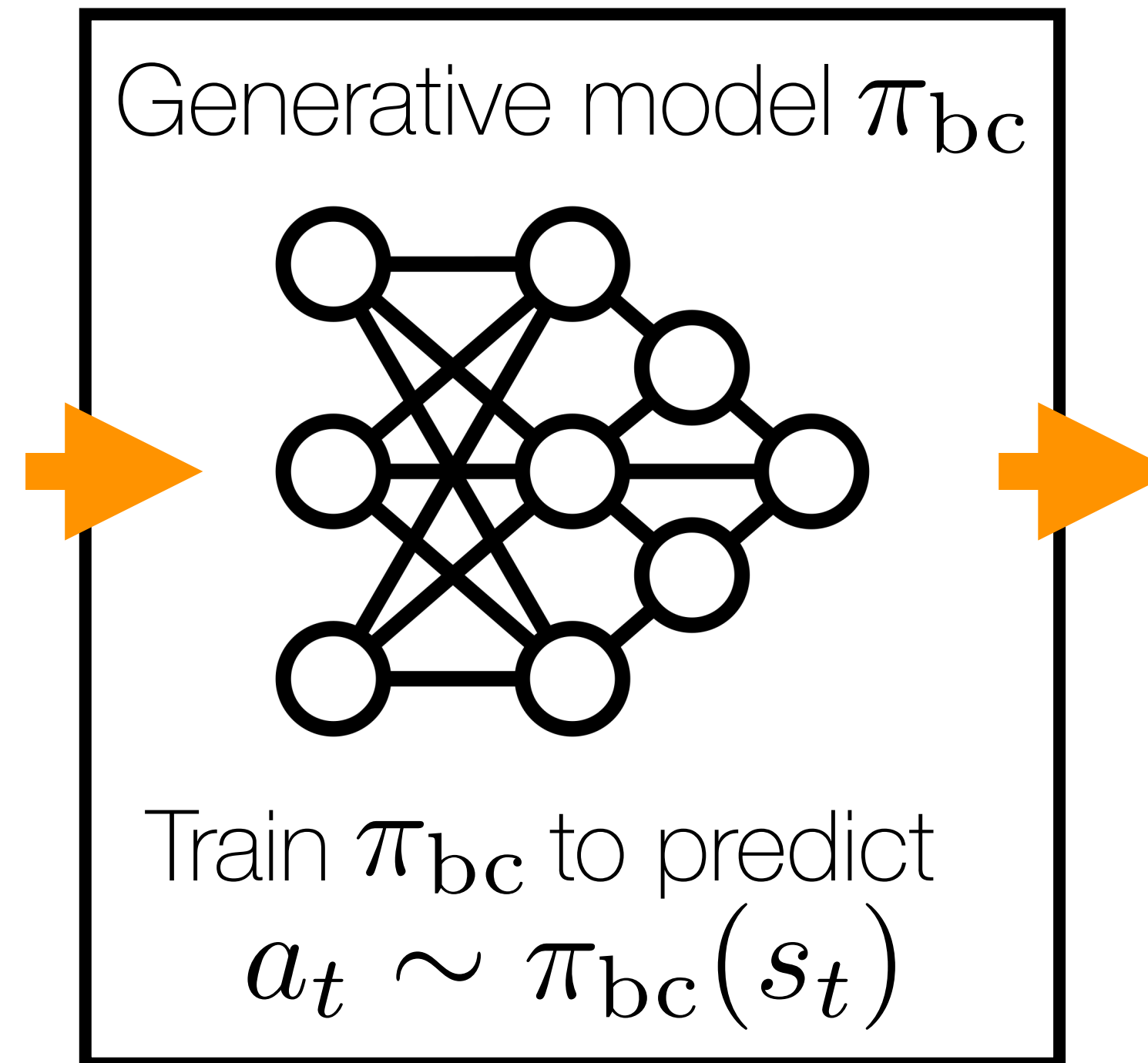


Human demonstrations + behavioral cloning in robotics



Human demonstrations

Reasonable but
not good enough



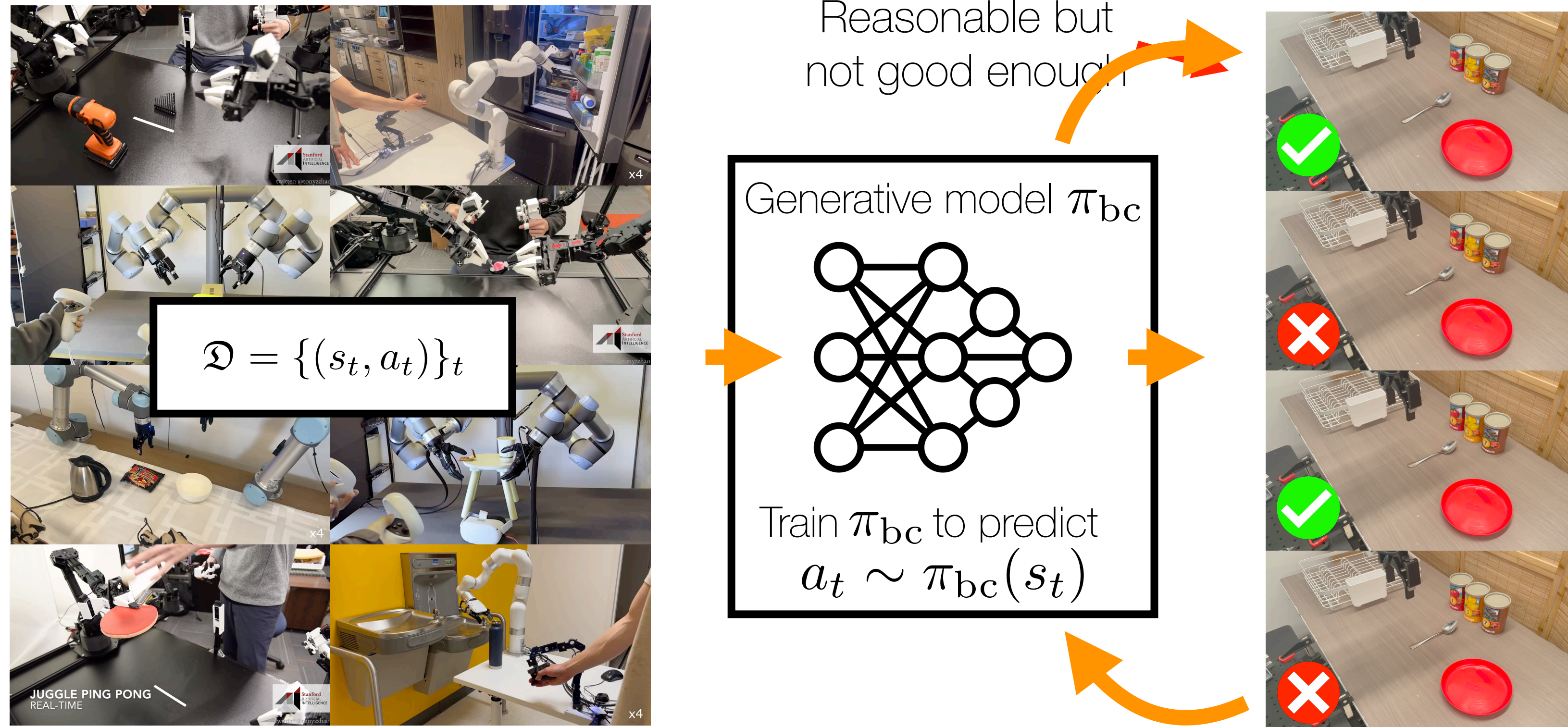
Behavioral Cloning (BC)

[ZKLF RSS '23; WSYLA IROS '23, LZLQYLM ICRA '24]



Deployment

Online improvement of pretrained robot policies



Assume correct behaviors
"in-distribution" for BC policy

Human demonstrations

Behavioral Cloning (BC)

Deployment

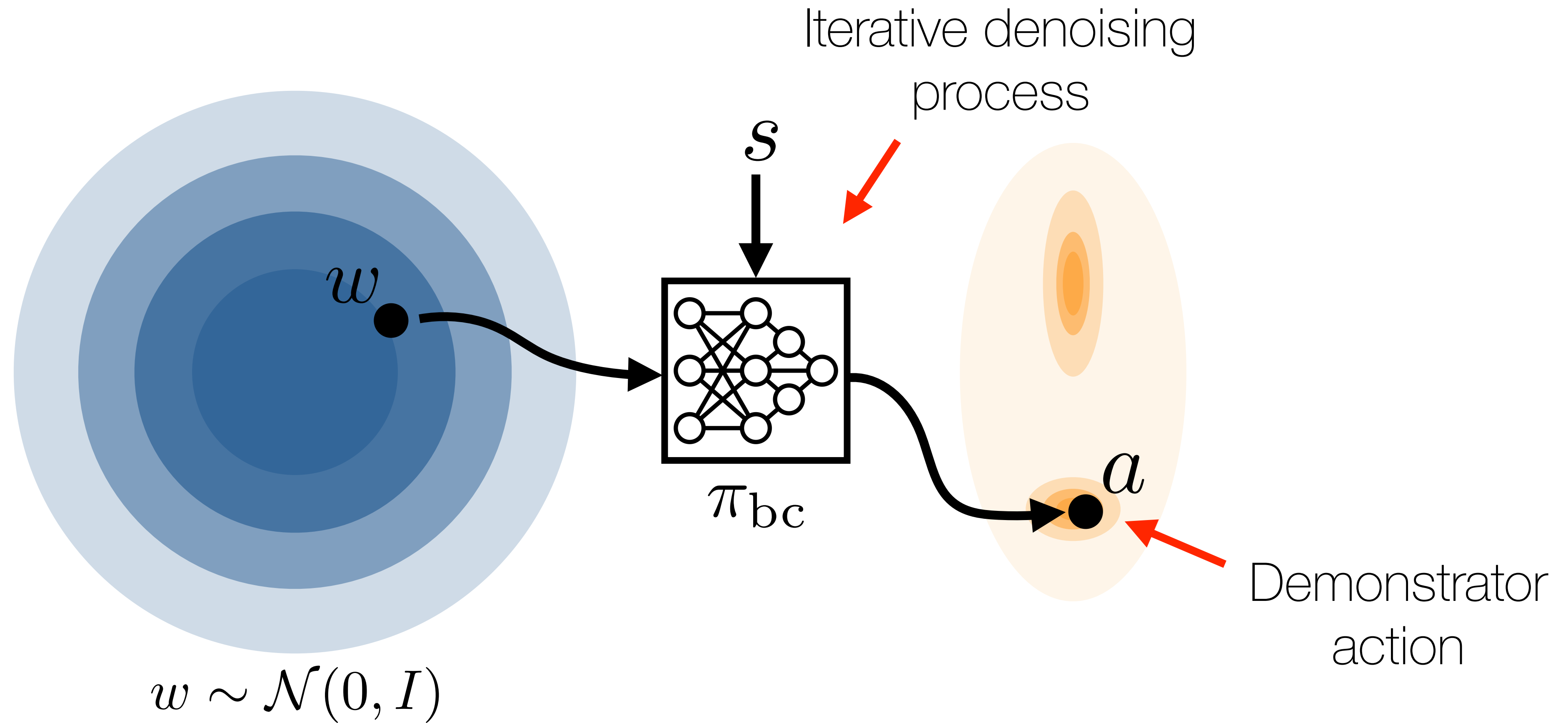
No assumption on BC policy

"Pretraining"

1. How can we efficiently improve a pretrained BC policy in deployment?

2. How should we "pretrain" a policy to ensure it can be efficiently improved?

Background: Diffusion policies



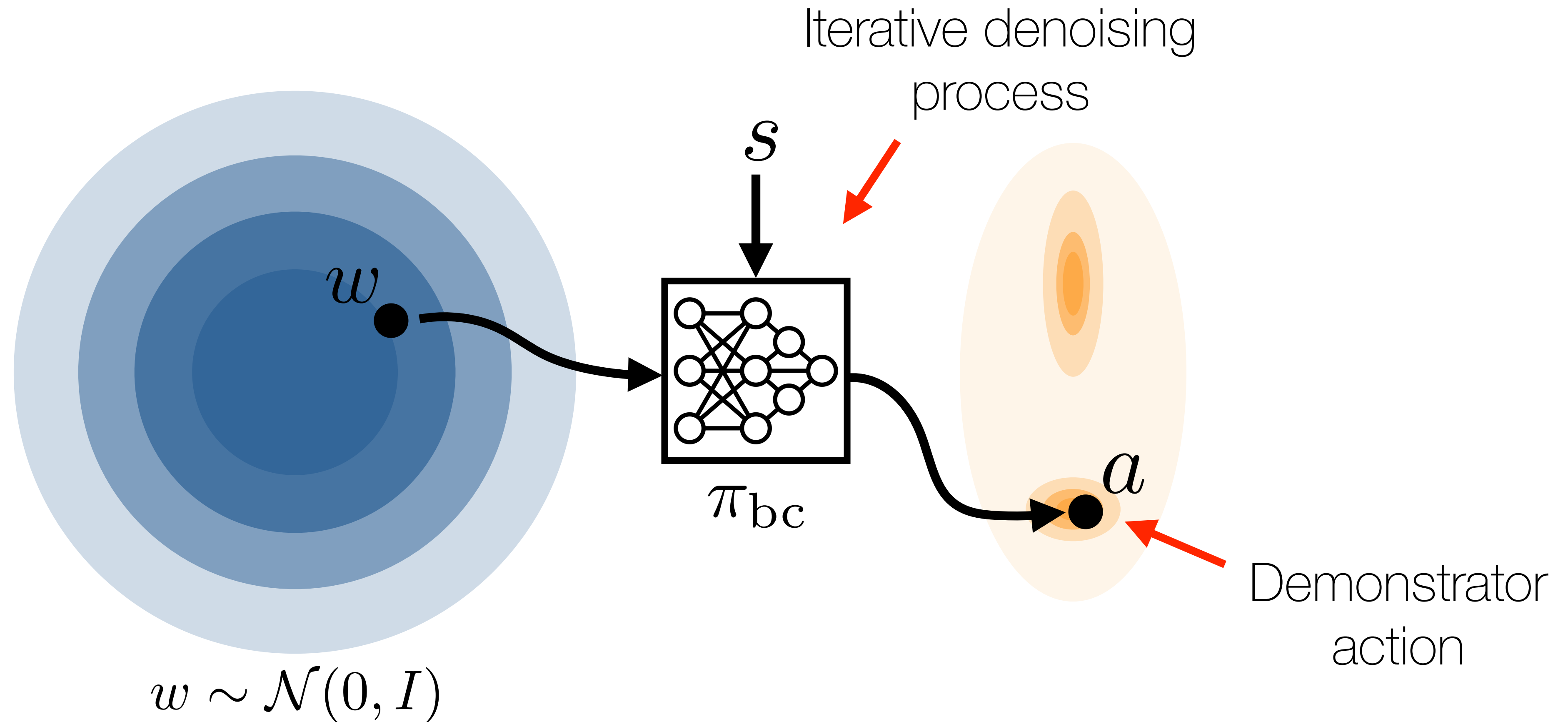
Background: Diffusion policies



Iterative denoising process



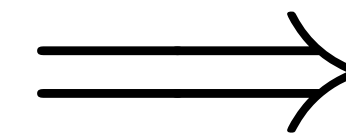
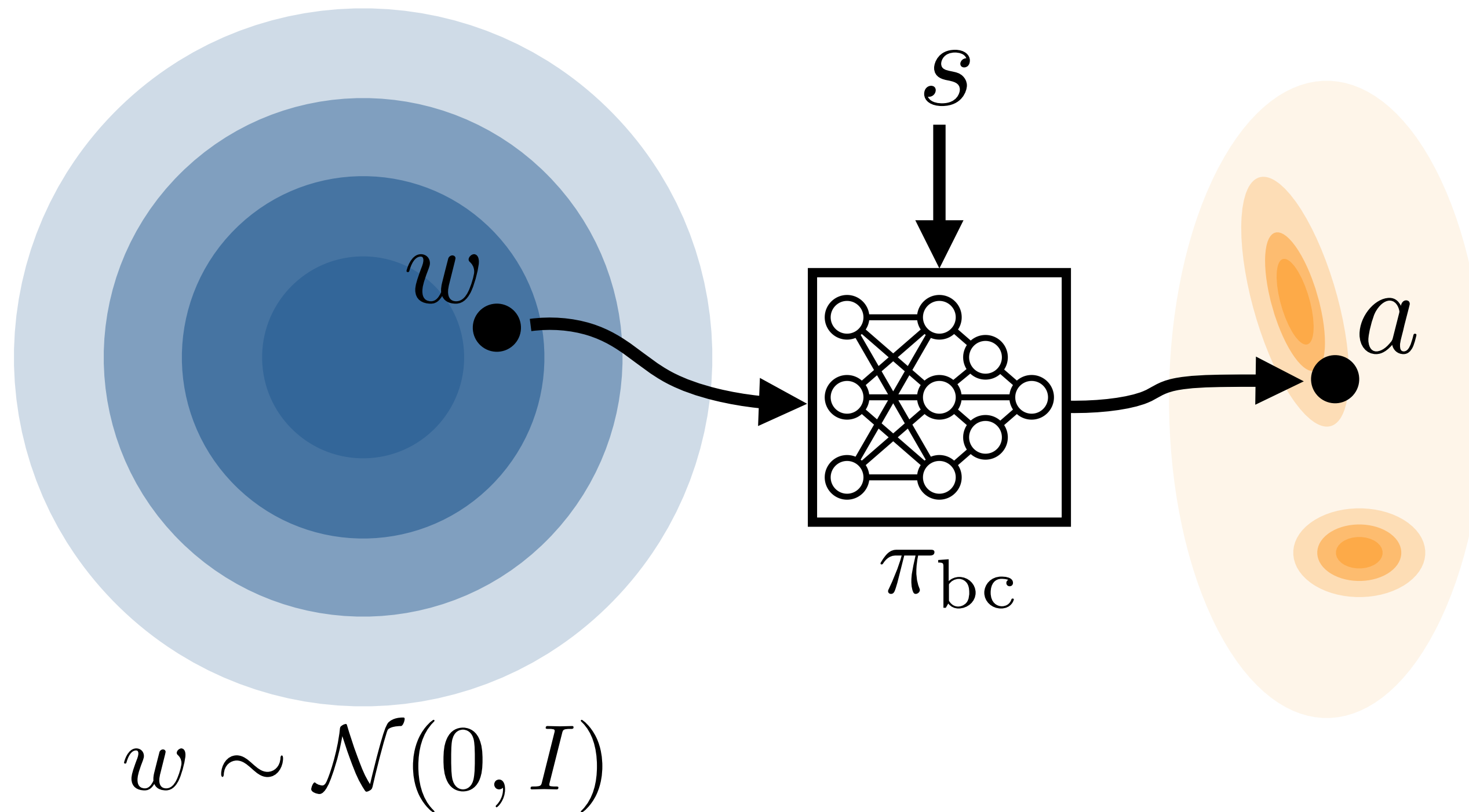
Background: Diffusion policies



A diffusion policy is simply a mapping that transforms noise to demonstrator actions

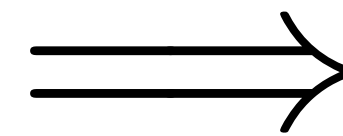
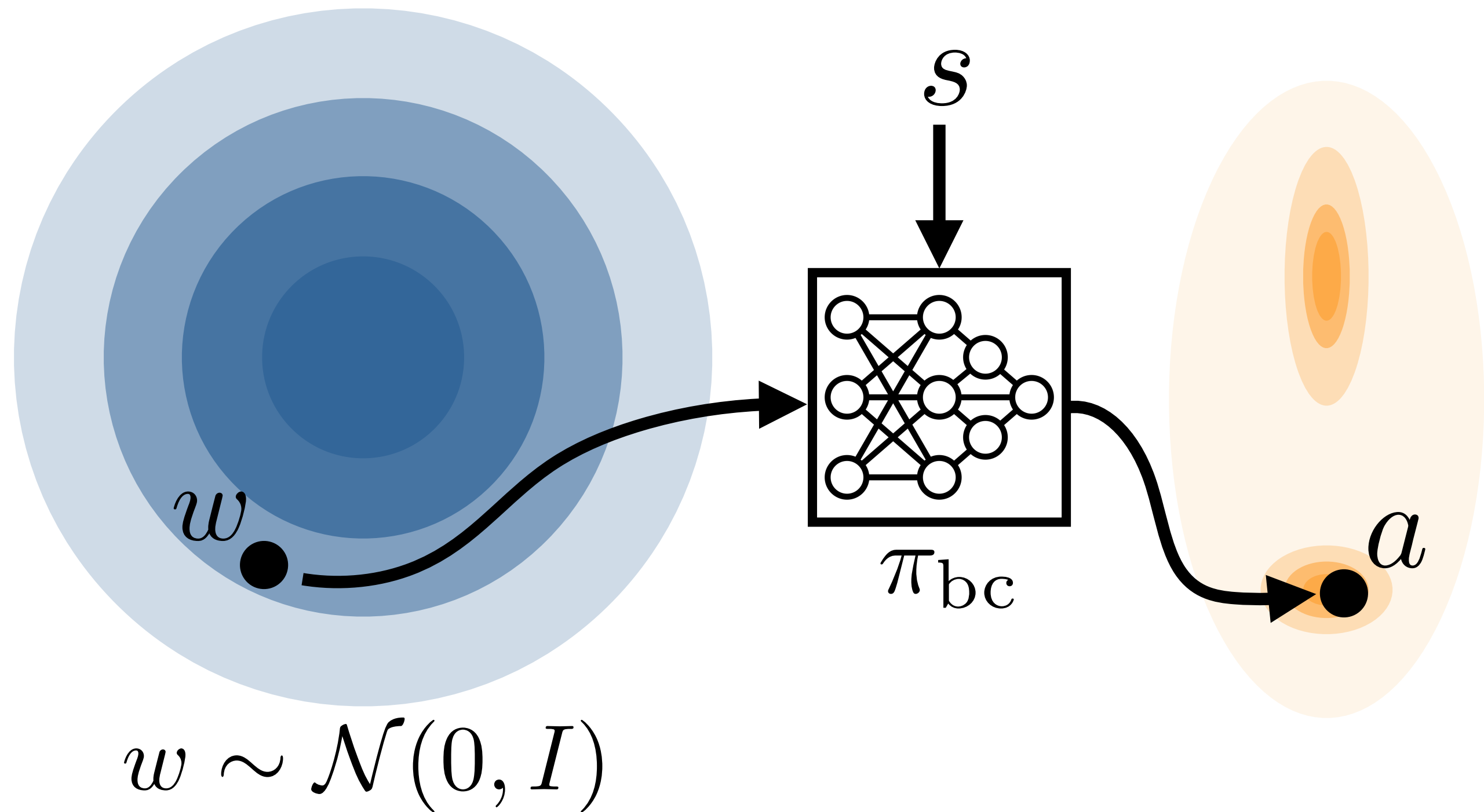
Standard diffusion policy deployment

Sampling process $a \sim \pi_{bc}(s)$

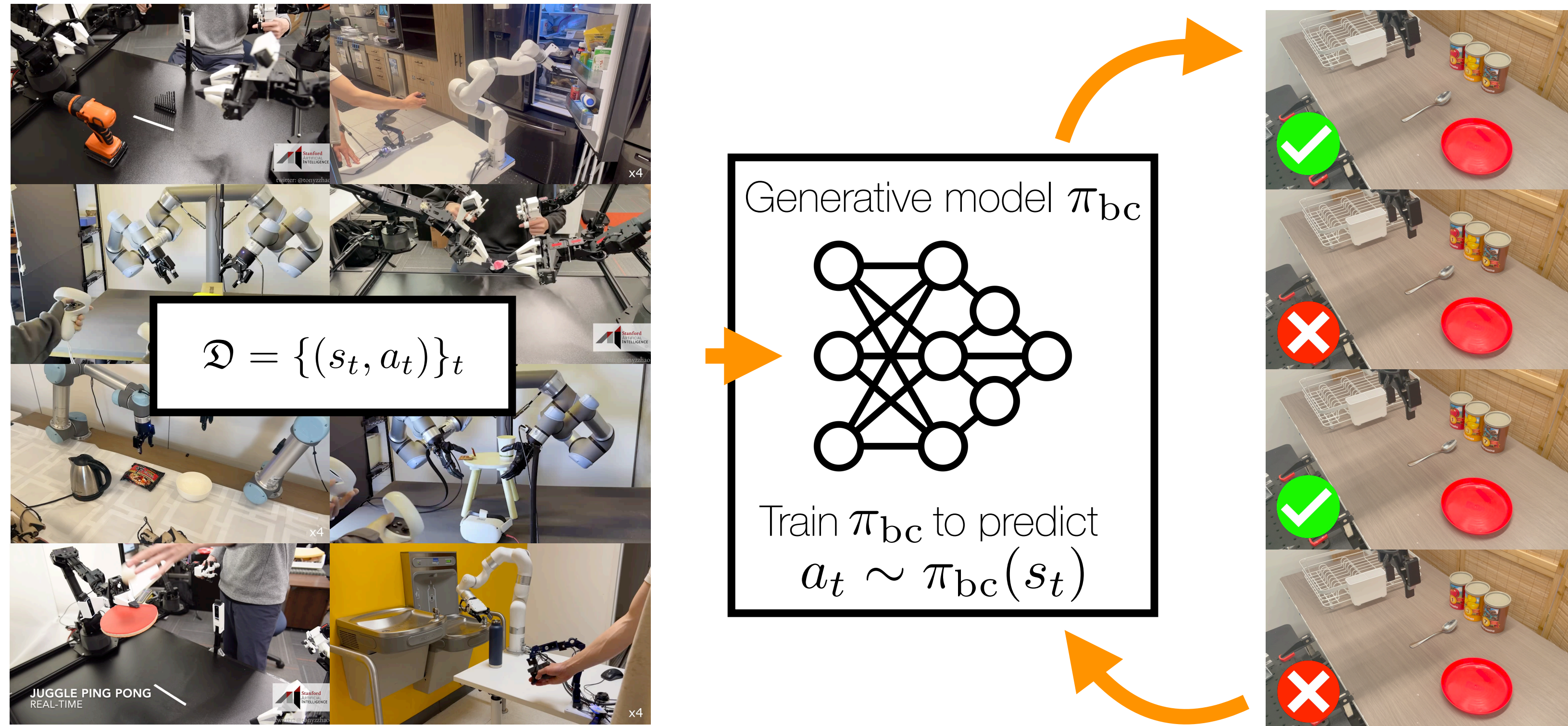


Standard diffusion policy deployment

Sampling process $a \sim \pi_{bc}(s)$



Online improvement of pretrained robot policies



Assume correct behaviors
"in-distribution" for BC policy

Human demonstrations

Behavioral Cloning (BC)

Deployment

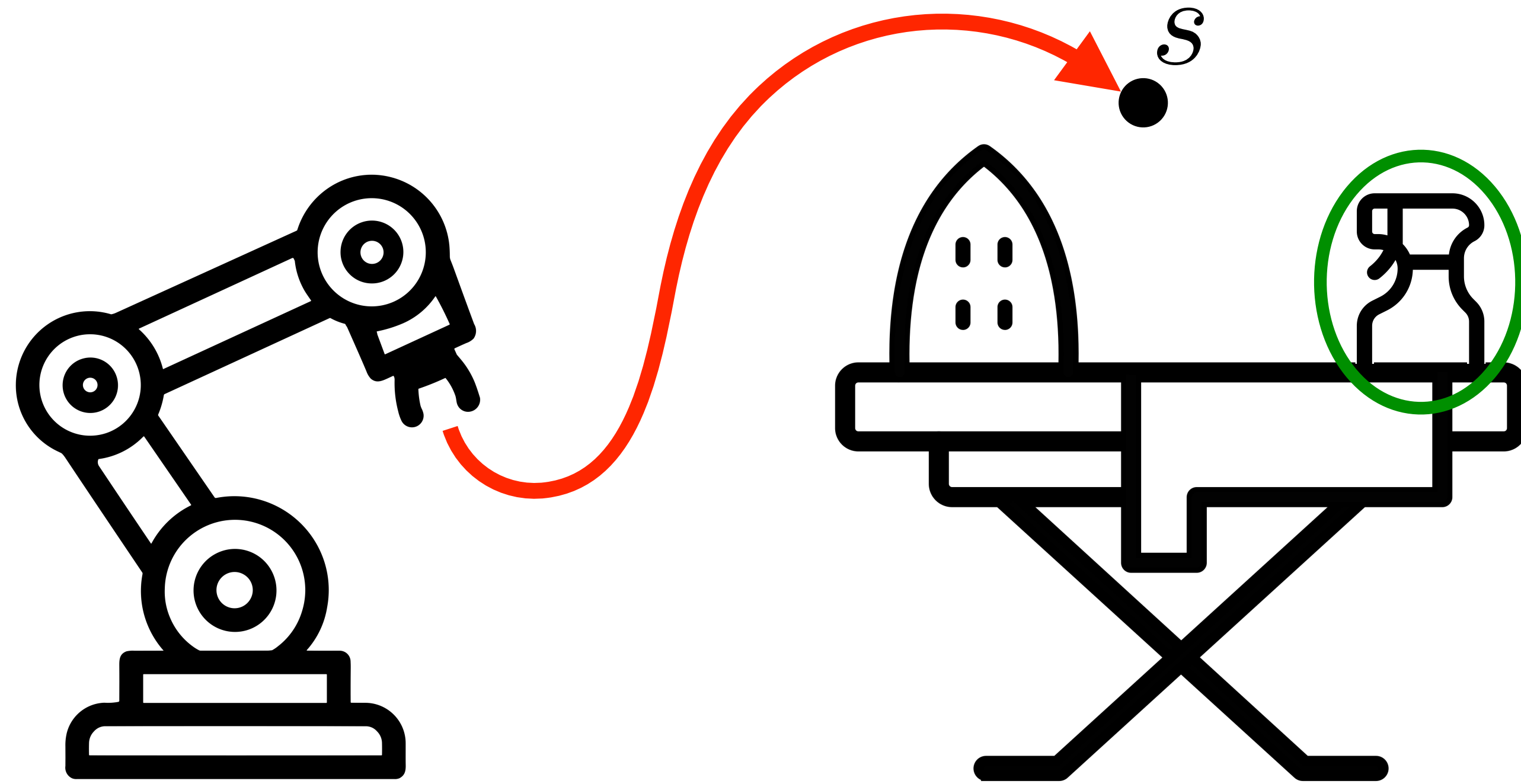
No assumption on BC policy

"Pretraining"

1. How can we efficiently improve a pretrained BC policy in deployment?

2. How should we "pretrain" a policy to ensure it can be efficiently improved?

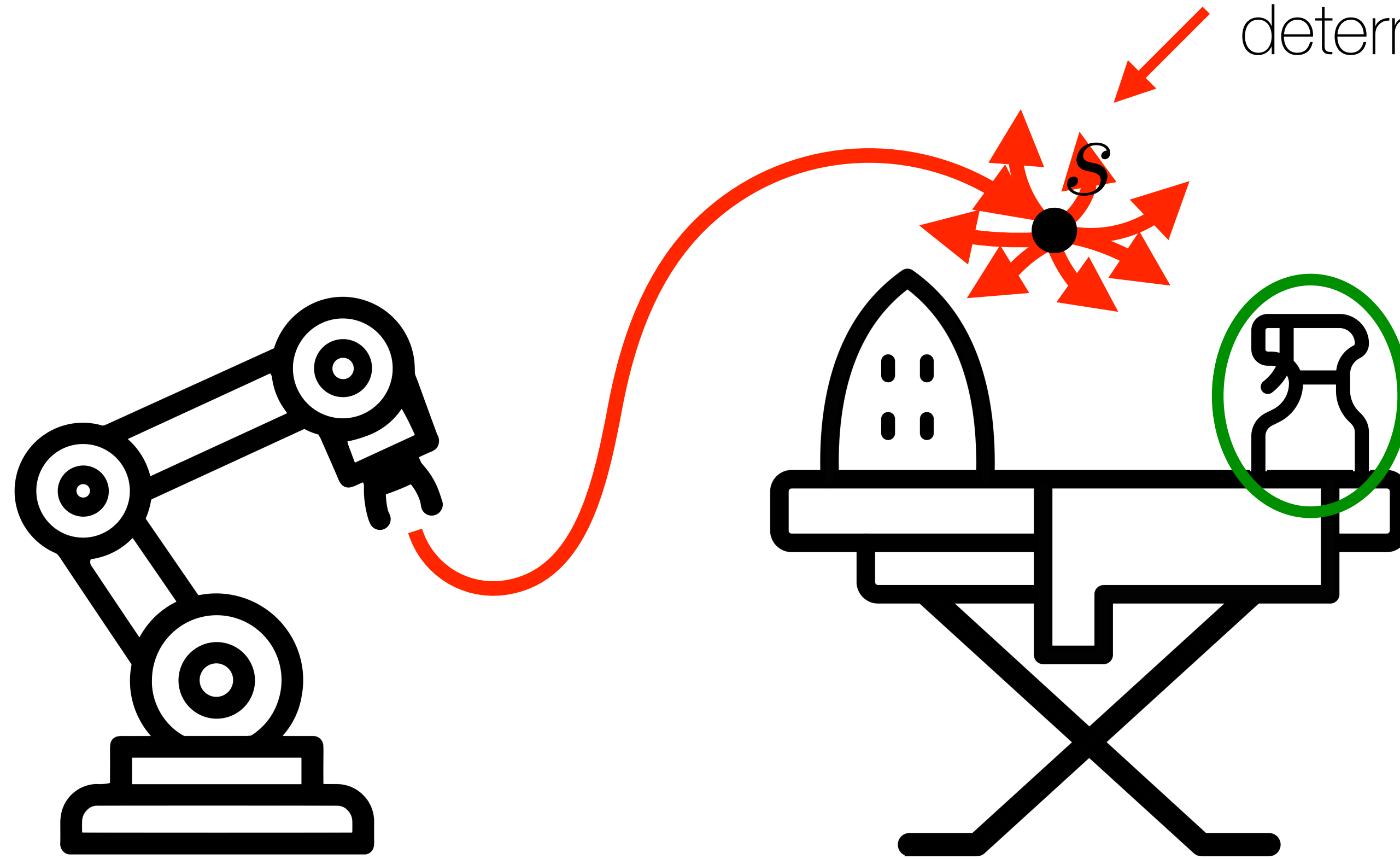
What makes RL hard?



Goal: “pick up the spray bottle”

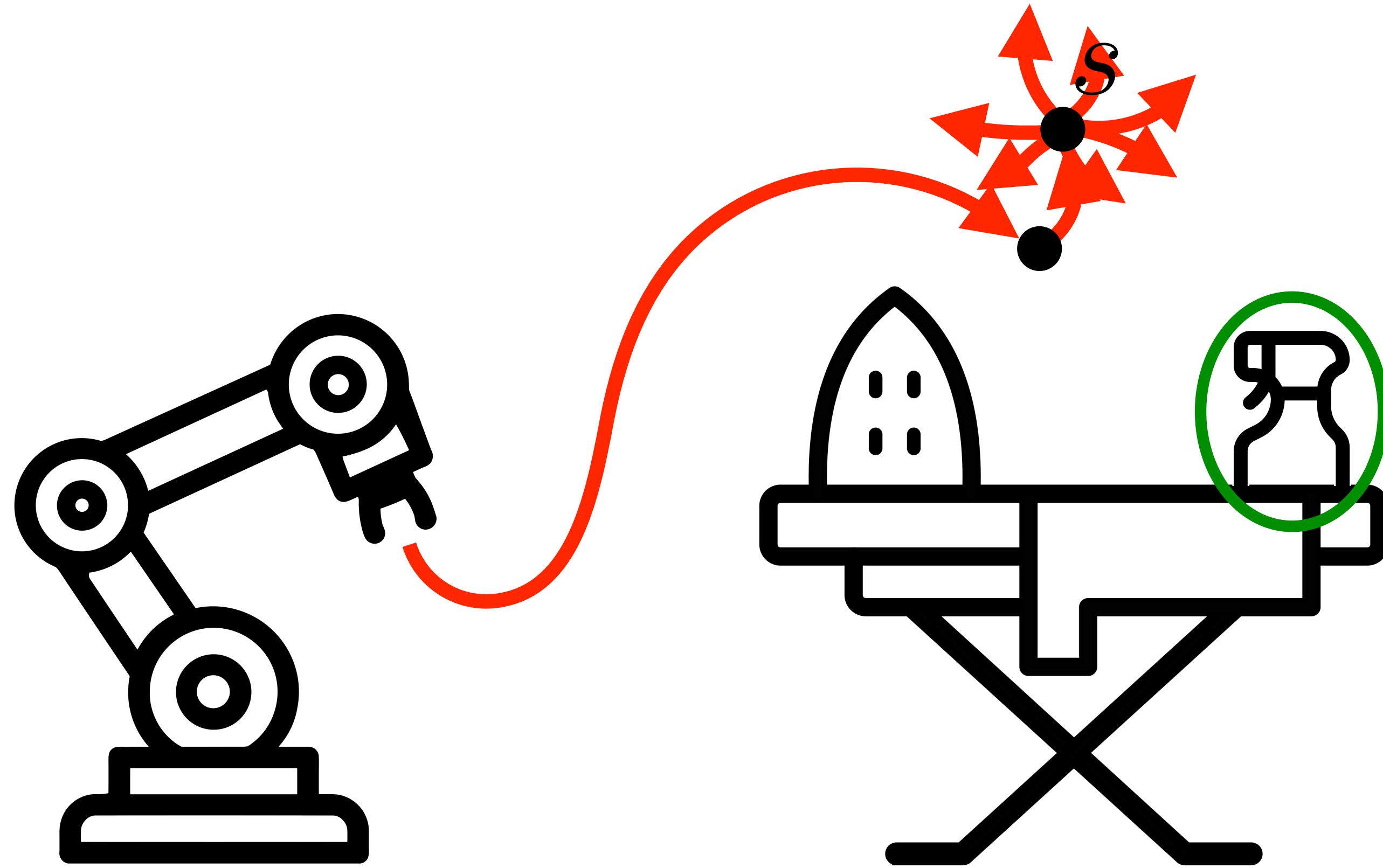
What makes RL hard?

Many possible actions, must determine which is best



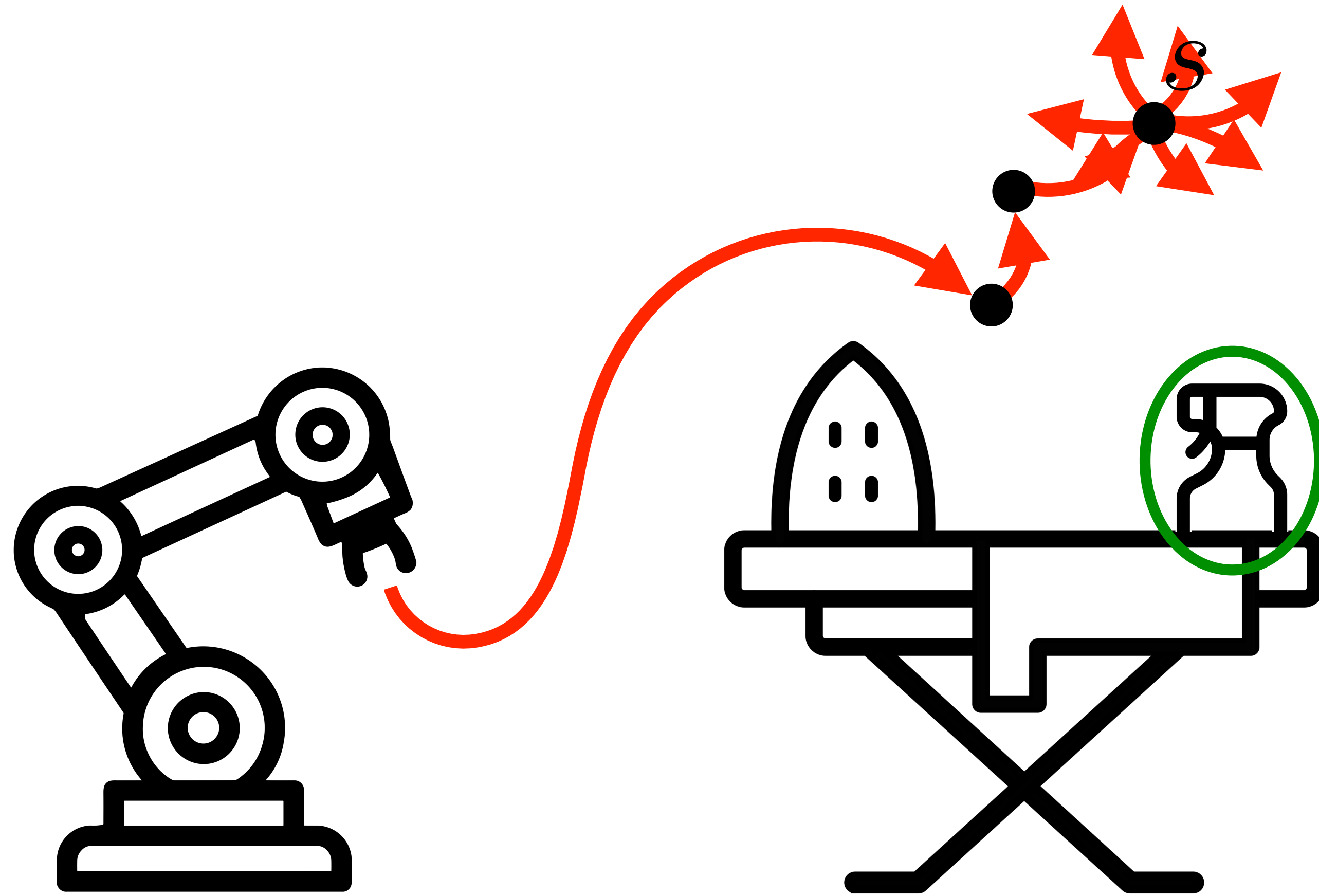
Goal: “pick up the spray bottle”

What makes RL hard?



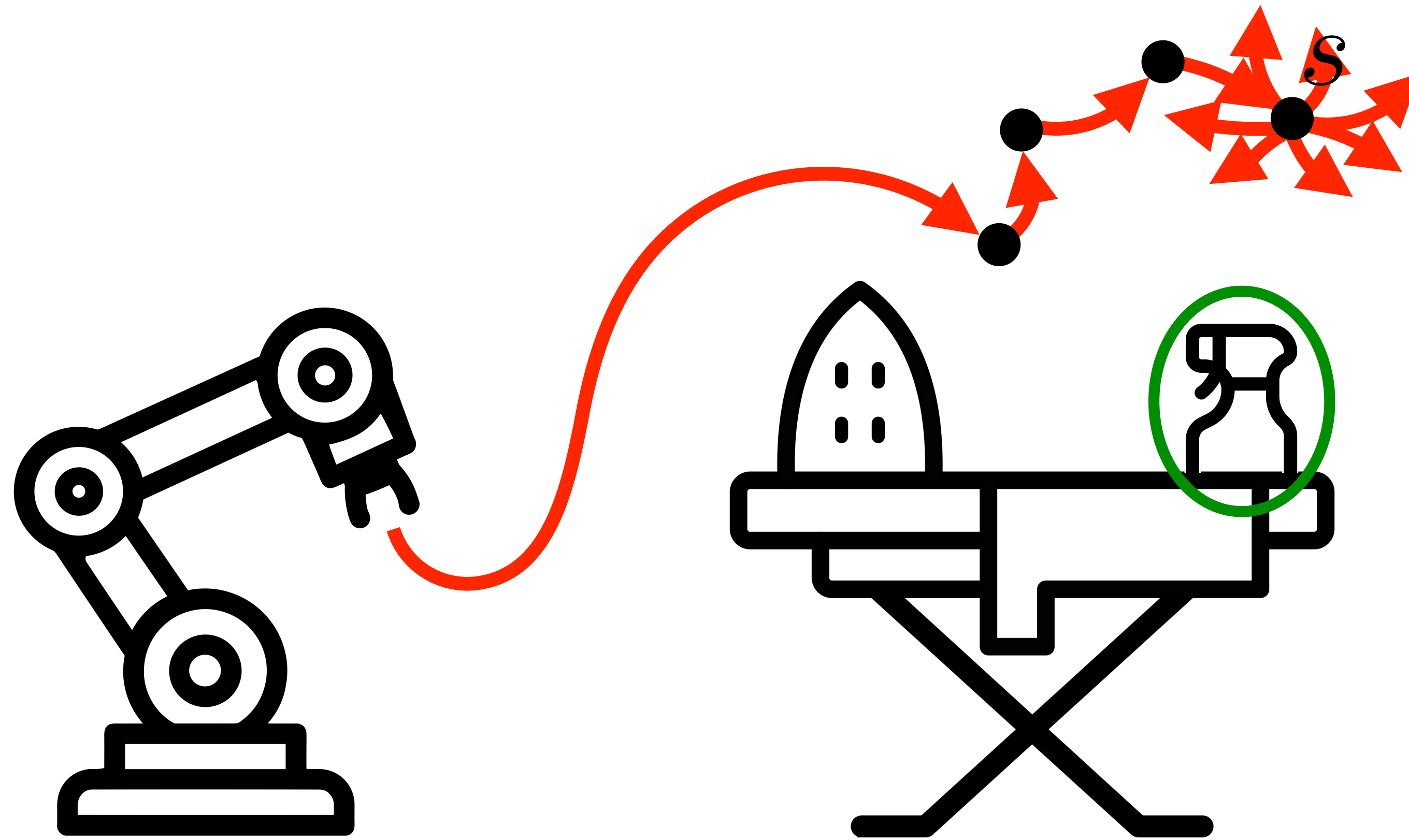
Goal: “pick up the spray bottle”

What makes RL hard?



Goal: “pick up the spray bottle”

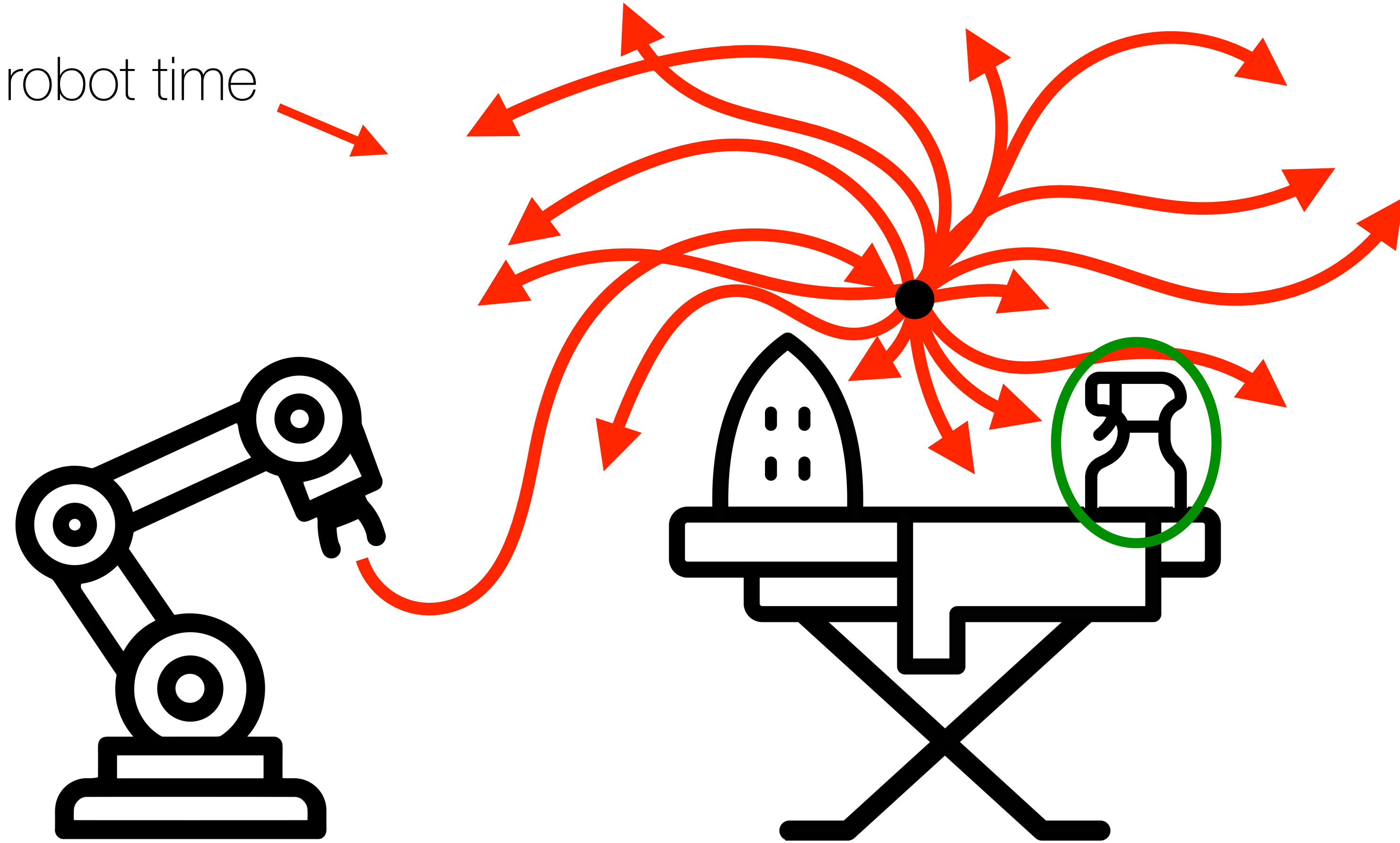
What makes RL hard?



Goal: “pick up the spray bottle”

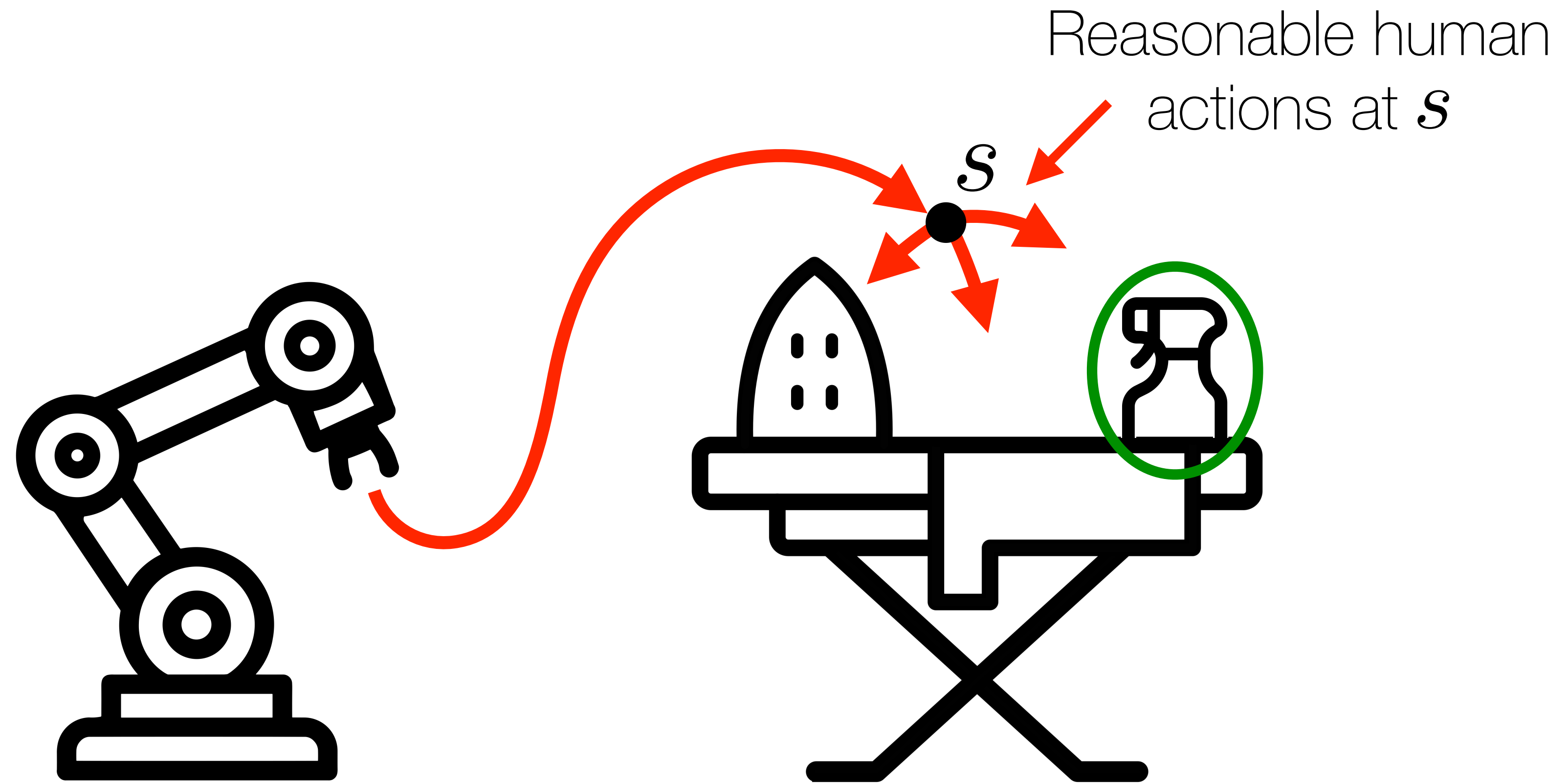
What makes RL hard?

Months of robot time



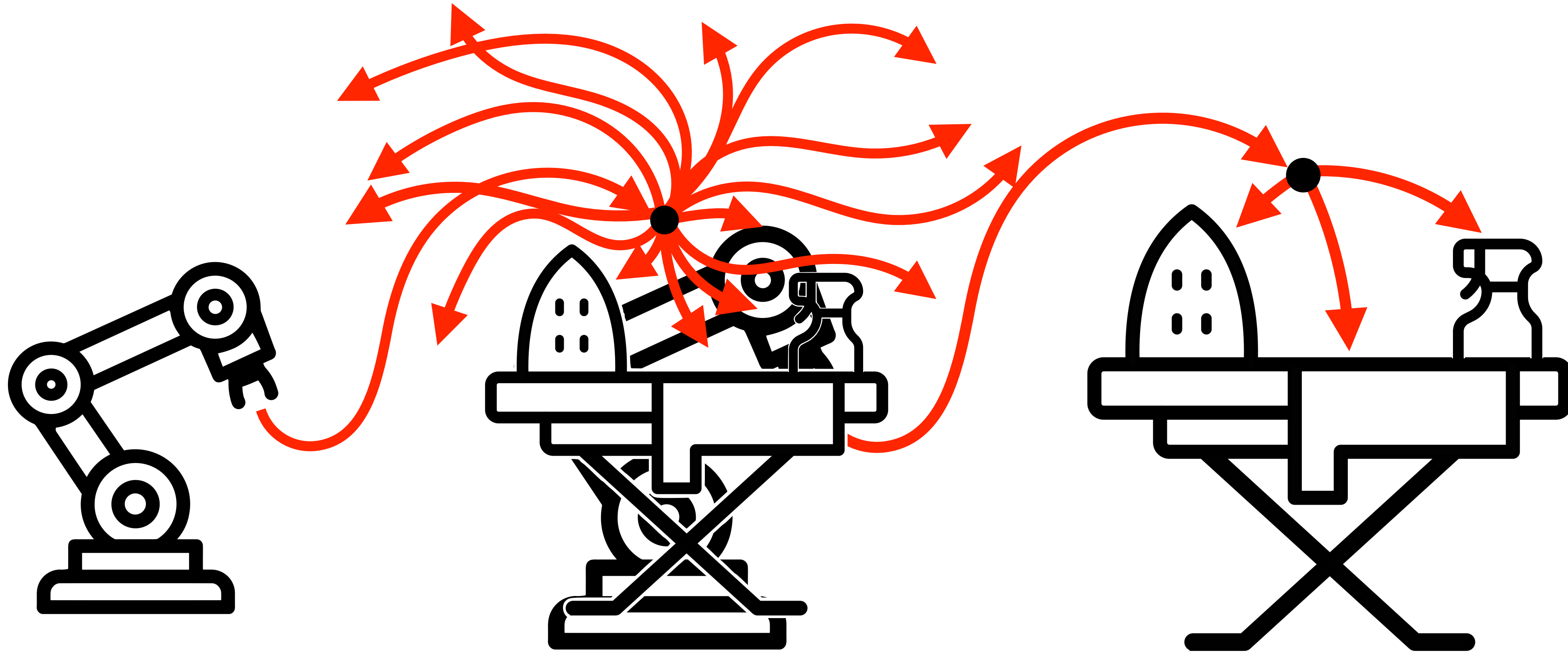
Challenge 1: Naive exploration doesn't scale to the real world

How would a human behave?



Goal: “pick up the spray bottle”

How would a human behave?



What makes RL on diffusion policies hard?

$$\underbrace{\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t \in 0:T} \nabla_{\theta} \ln \pi_{\theta}(A_t | S_t) \sum_{t \in 0:T} (\gamma^t R_t) \right]_{S_0 = s_0}}_{\text{Standard RL policy update}}$$

Running this requires:

1. Evaluating action likelihoods
2. Backpropagating through policy

Challenge 2: For diffusion policies,

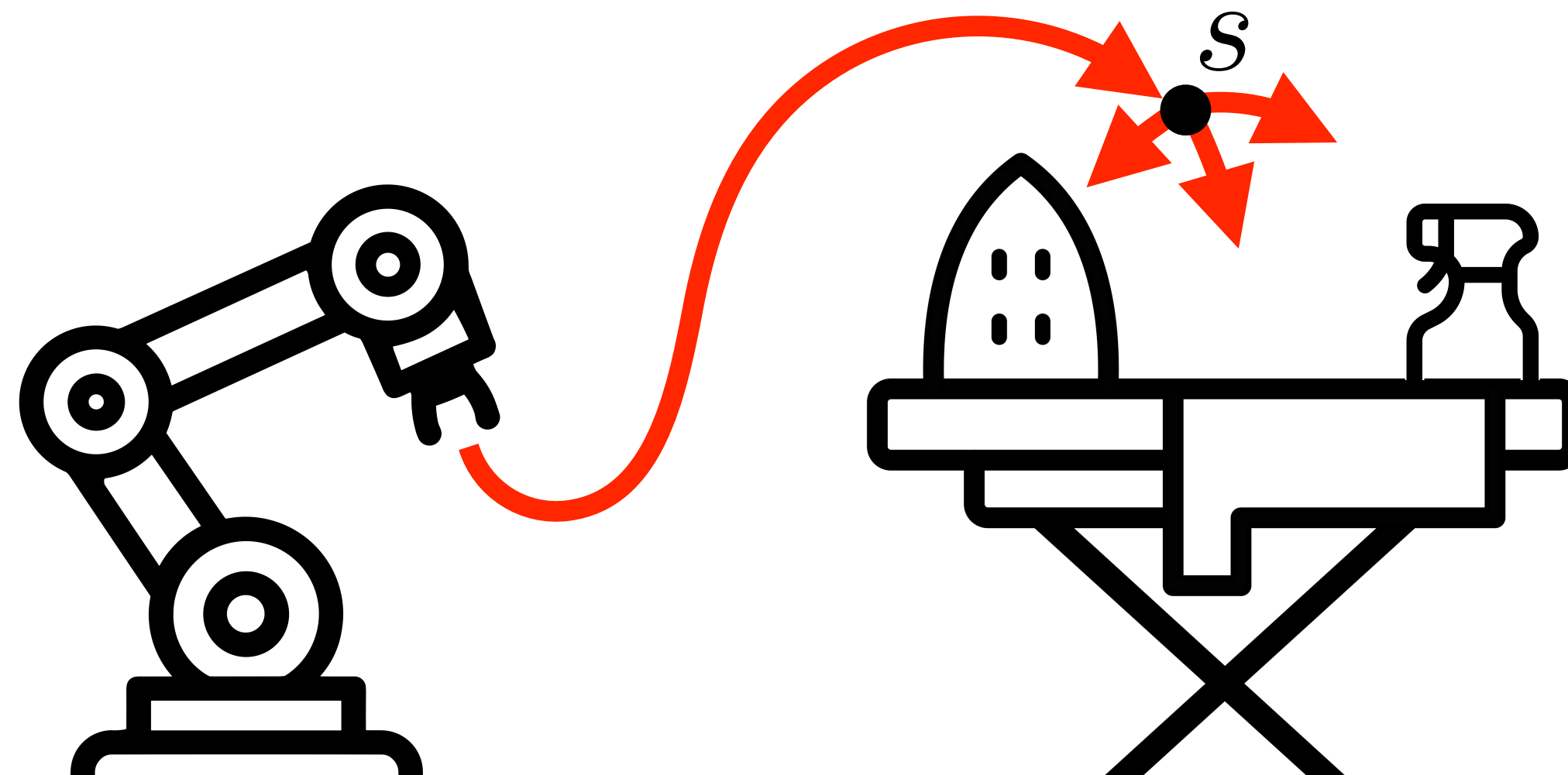
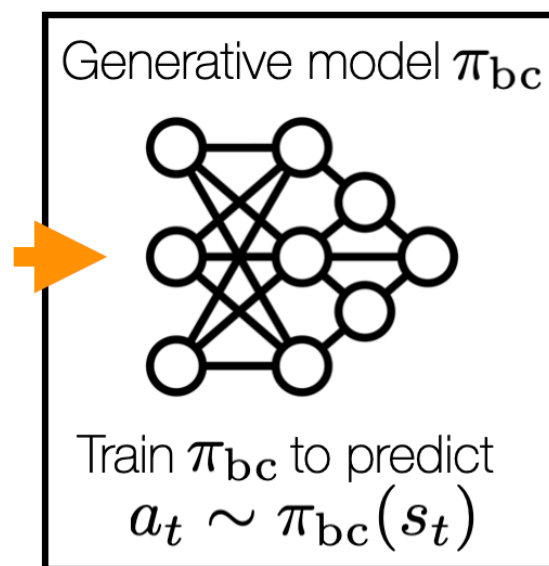
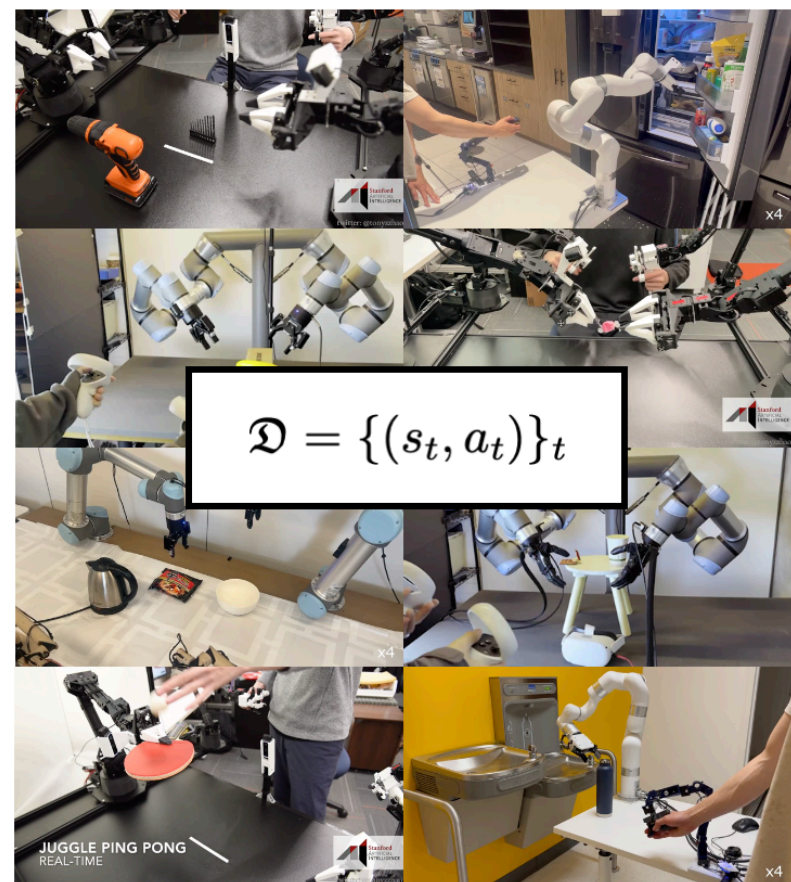
1. Can't evaluate action likelihoods
2. Backpropagation through time unstable



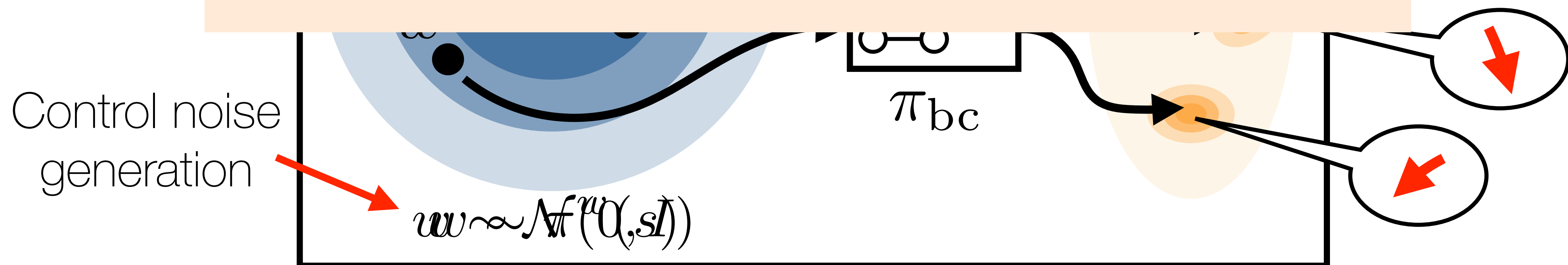
How can we enable efficient online improvement of diffusion policies?

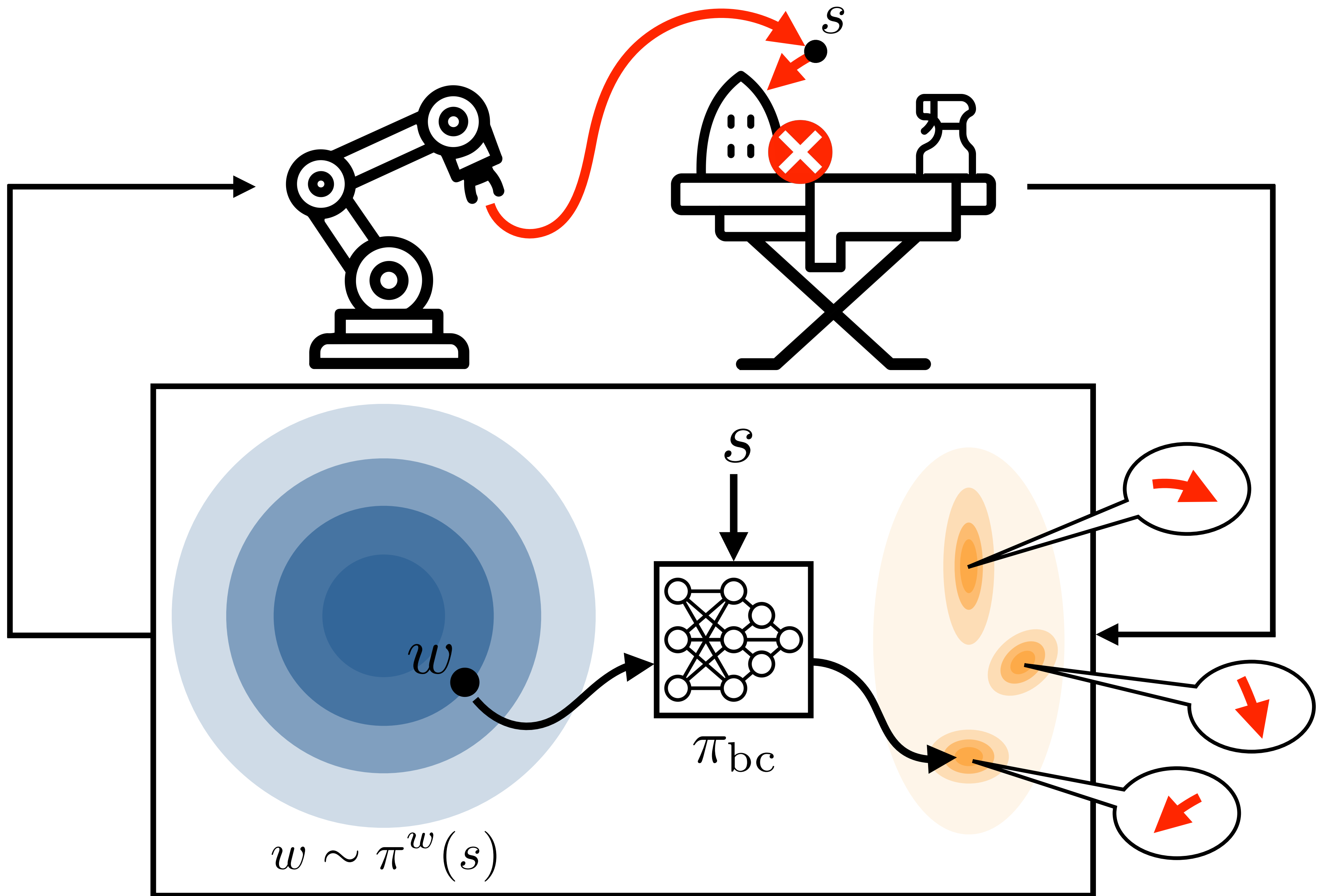
Challenge 1: Naive exploration doesn't scale to the real world

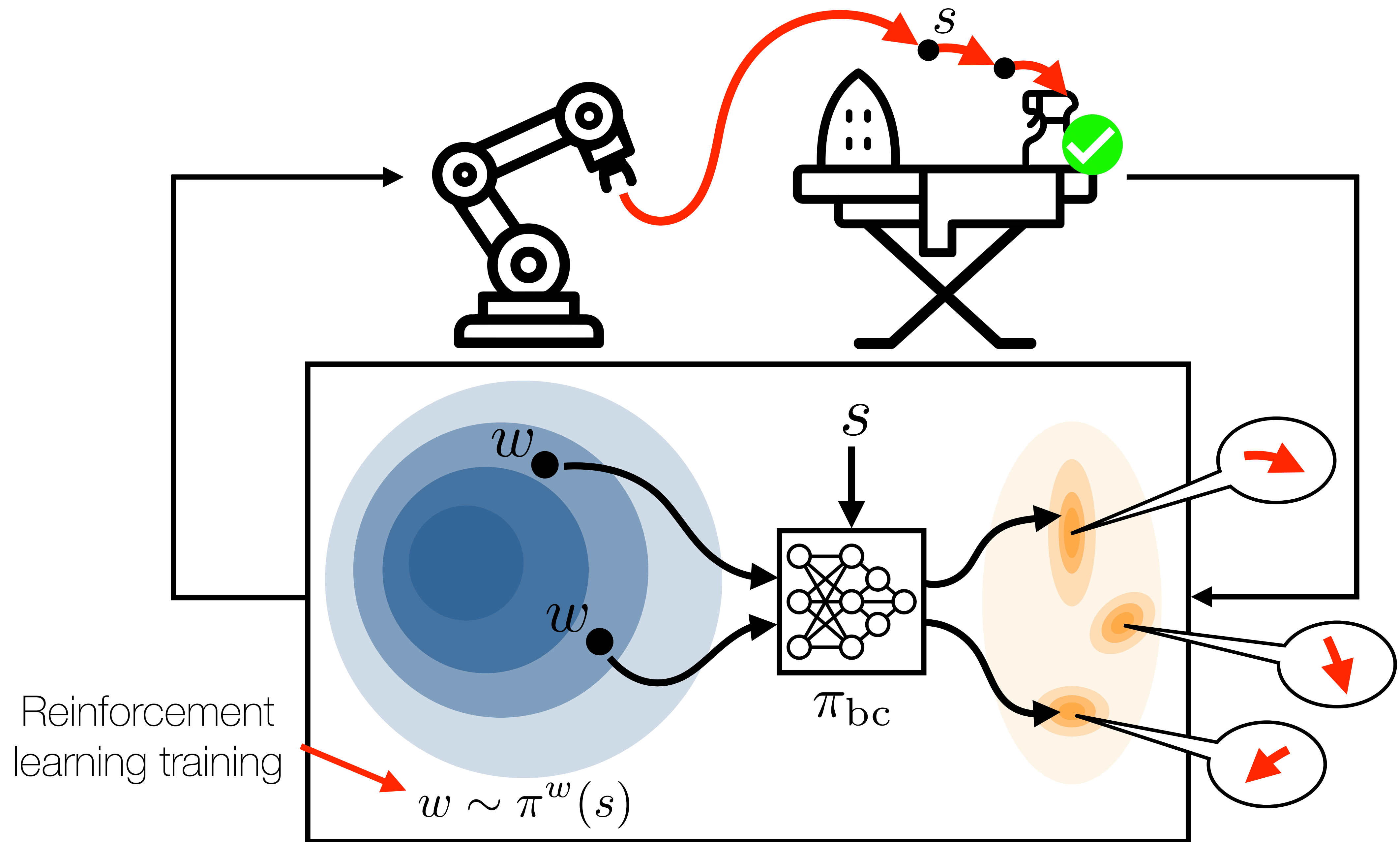
Challenge 2: For diffusion policies,
1. Can't evaluate action likelihoods
2. Backpropagation through time unstable

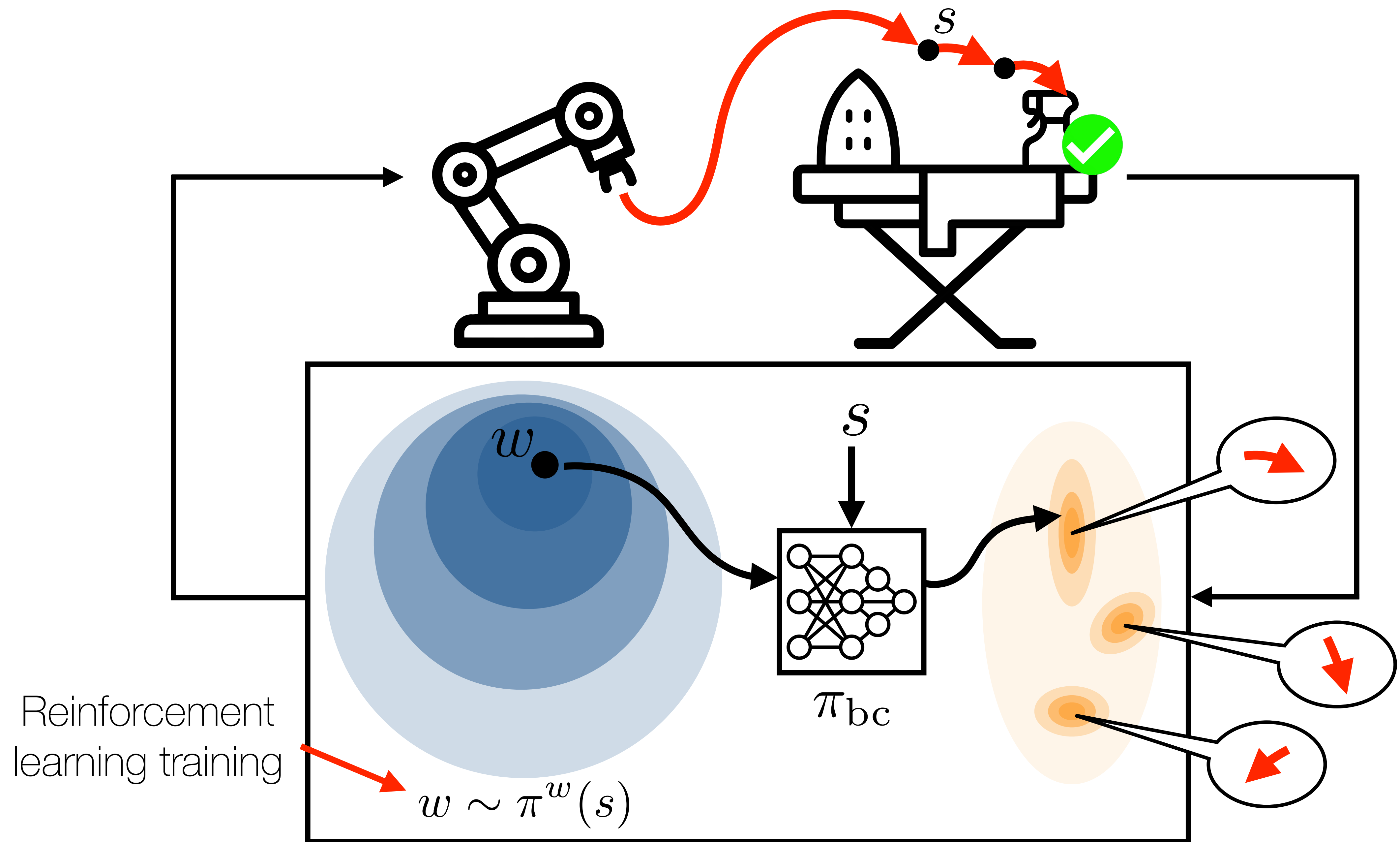


Key Idea: Treat the **noise** as the **action**
 → Rather than optimizing over original action space, optimize over noise space

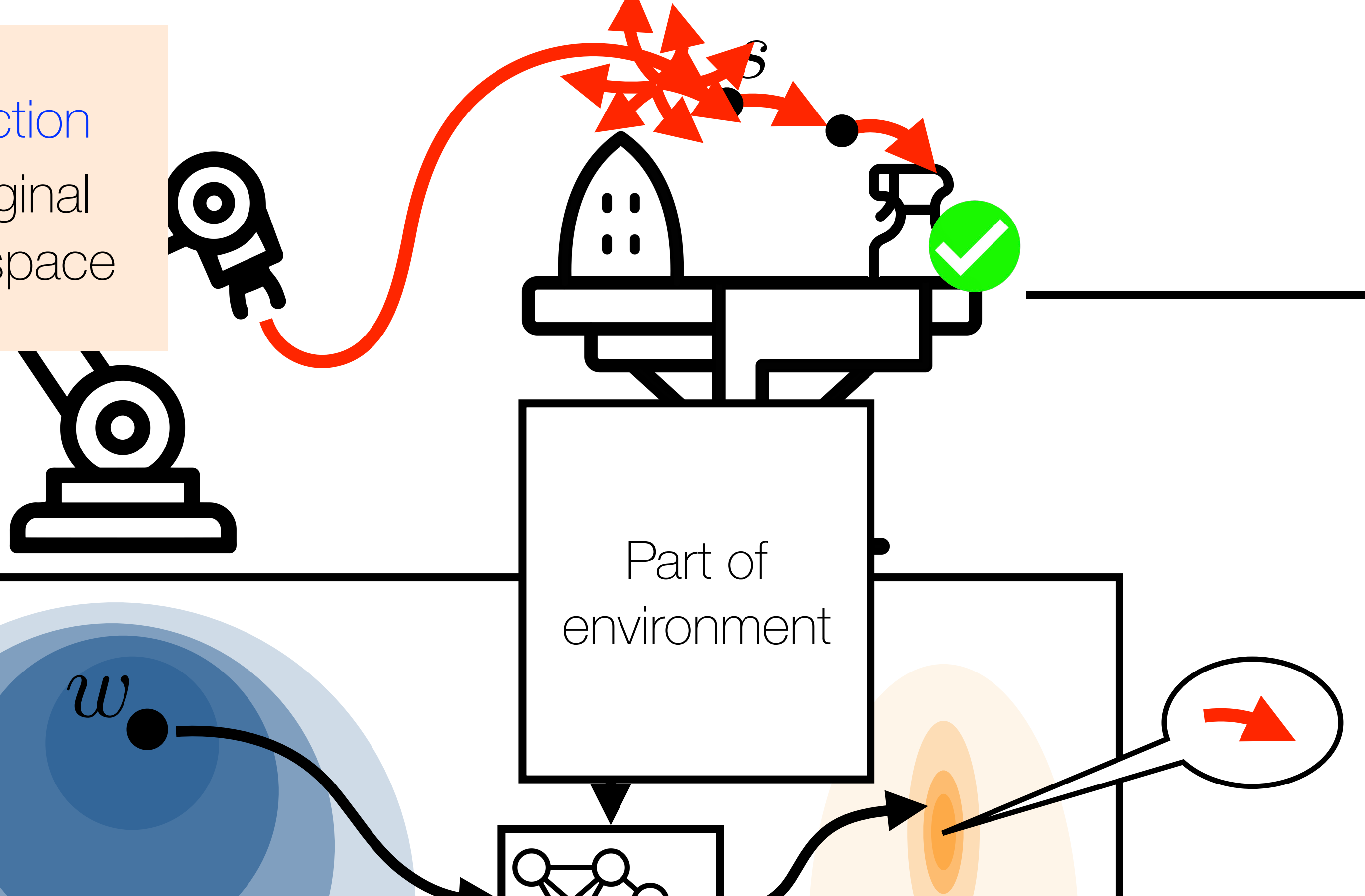








Key Idea: Treat the noise as the action
→ Rather than optimizing over original action space, optimize over noise space



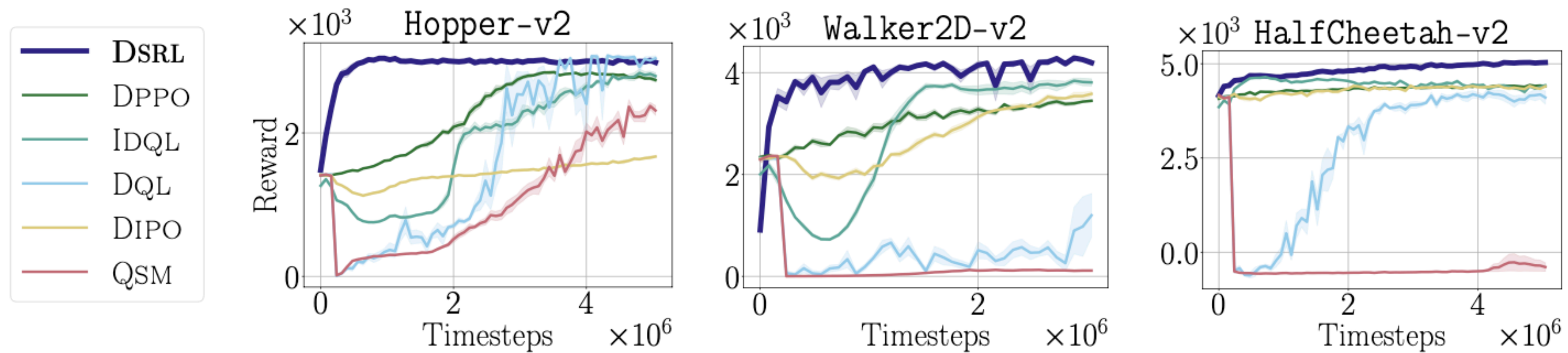
Challenge 1: Naive exploration doesn't scale to the real world

Challenge 2: For diffusion policies,
1. Can't evaluate action likelihoods
2. Backpropagation through time unstable

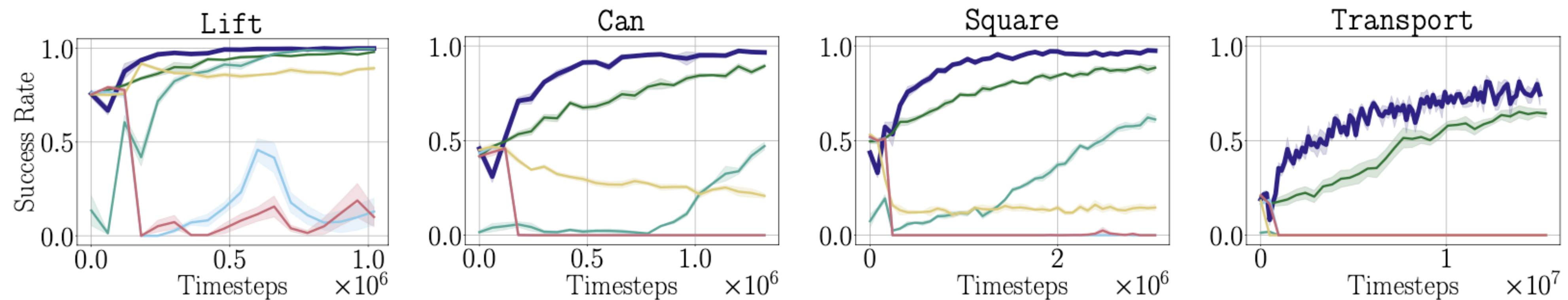
learning training
 $w \sim \pi^w(s)$
Solution: Behaviors learned via BC can be exploited to enable structured exploration

Solution: Treat diffusion policy as black box, optimize over input noise to improve behavior

DSRL: Simulated Results



Online results on OpenAI Gym



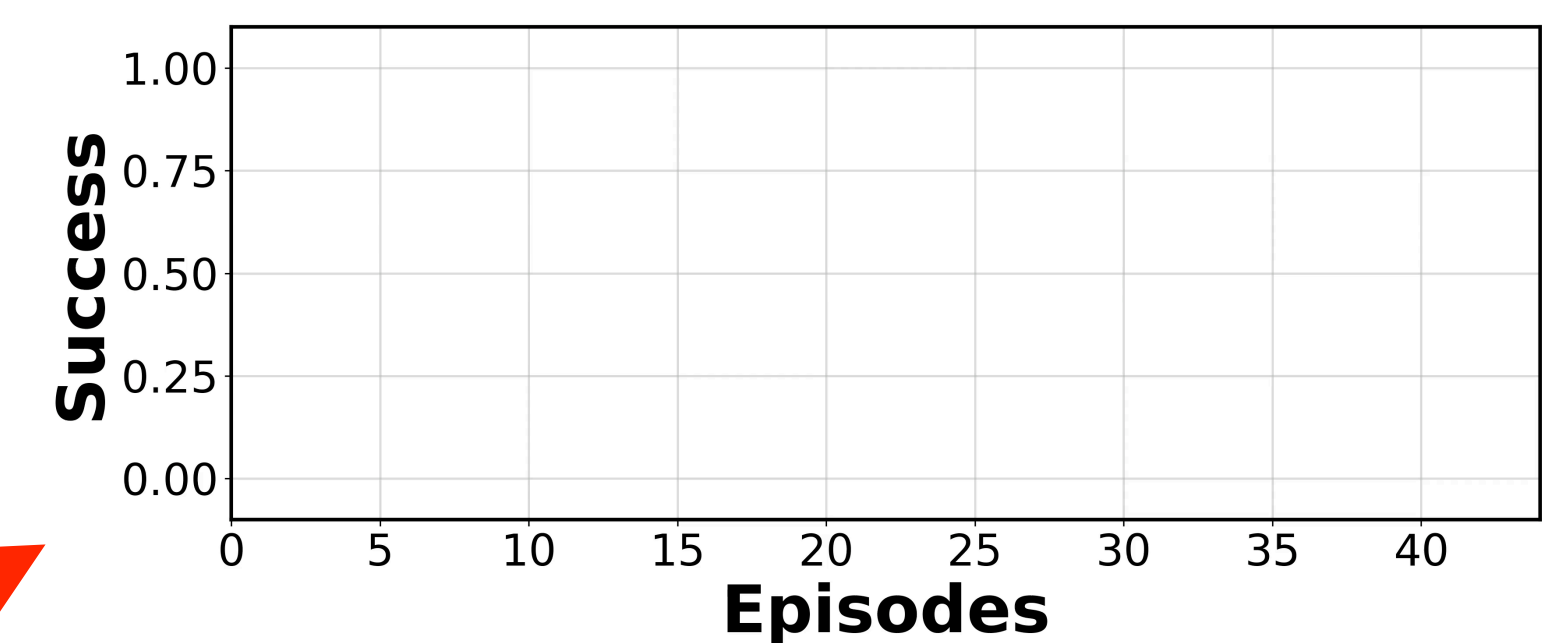
Online results on Robomimic



Goal: "Place the mushroom on the cloth"

Task Success vs.
Amount of Experience

*including approaches that rely on the BC policy



Flow-based VLA
(π_0 from Physical Intelligence)



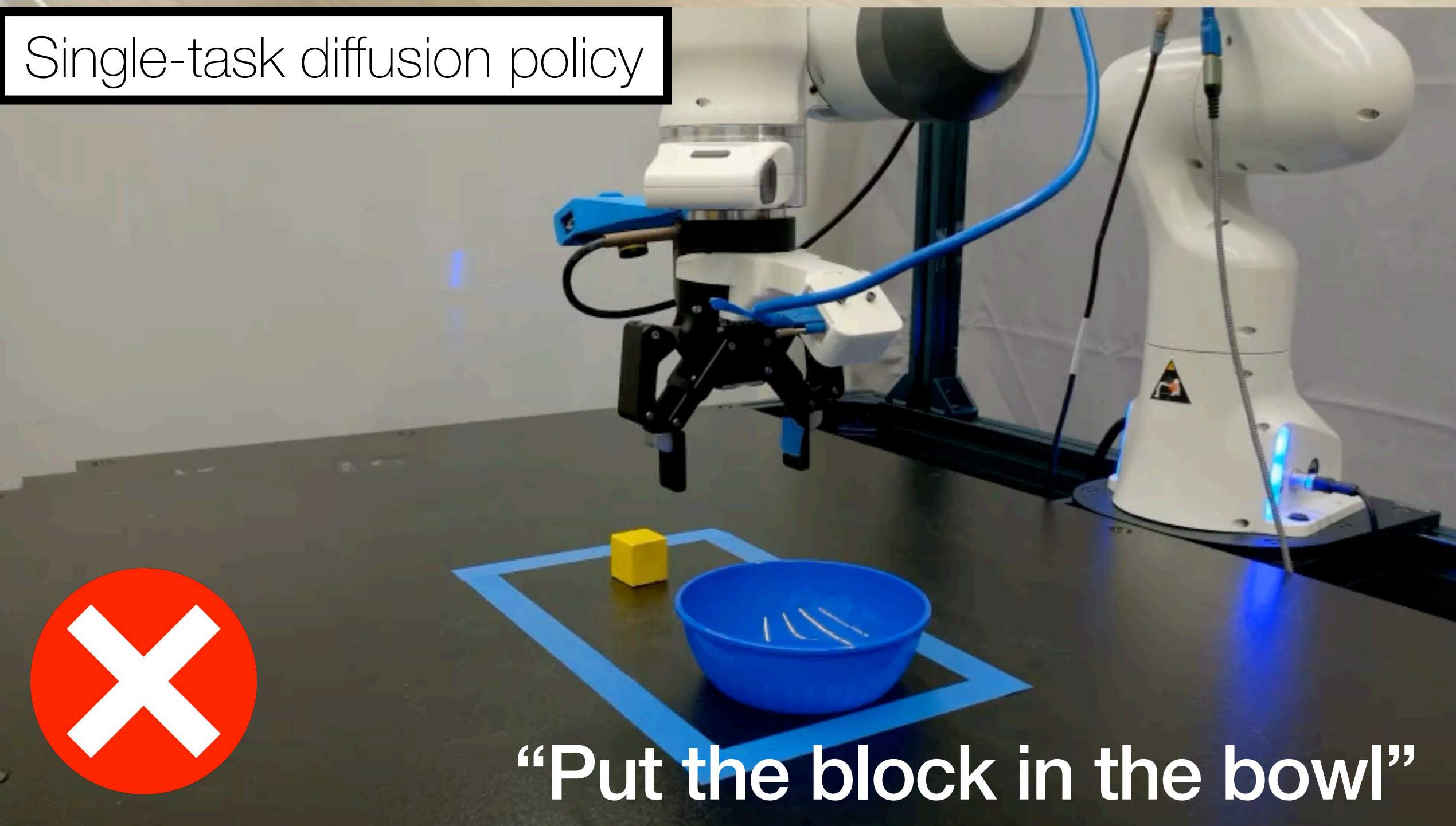
“Turn on the toaster”

Multi-task diffusion policy



“Set the mushroom on the cloth”

Single-task diffusion policy

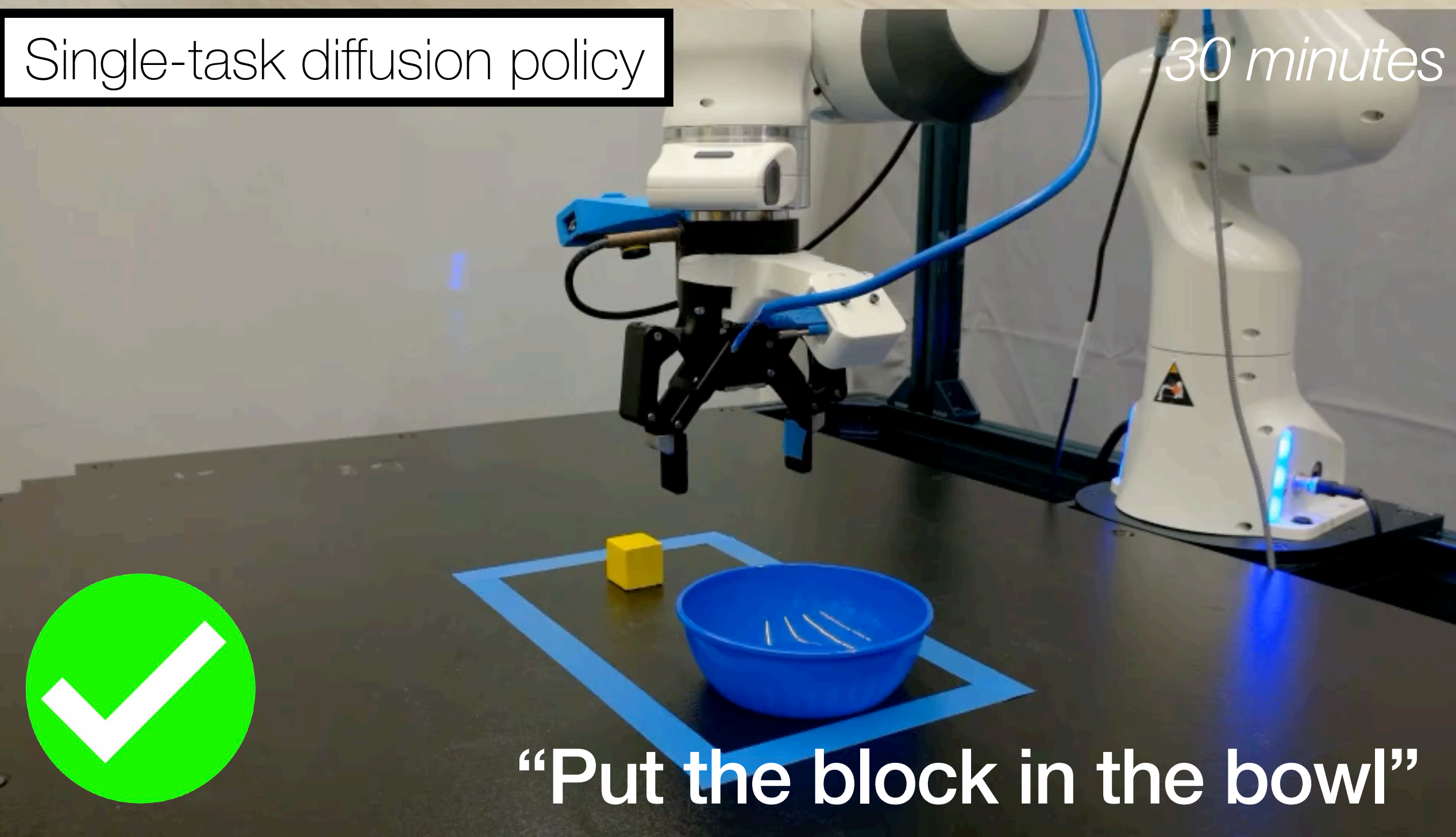


“Put the block in the bowl”

Flow-based VLA
(π_0 from Physical Intelligence)

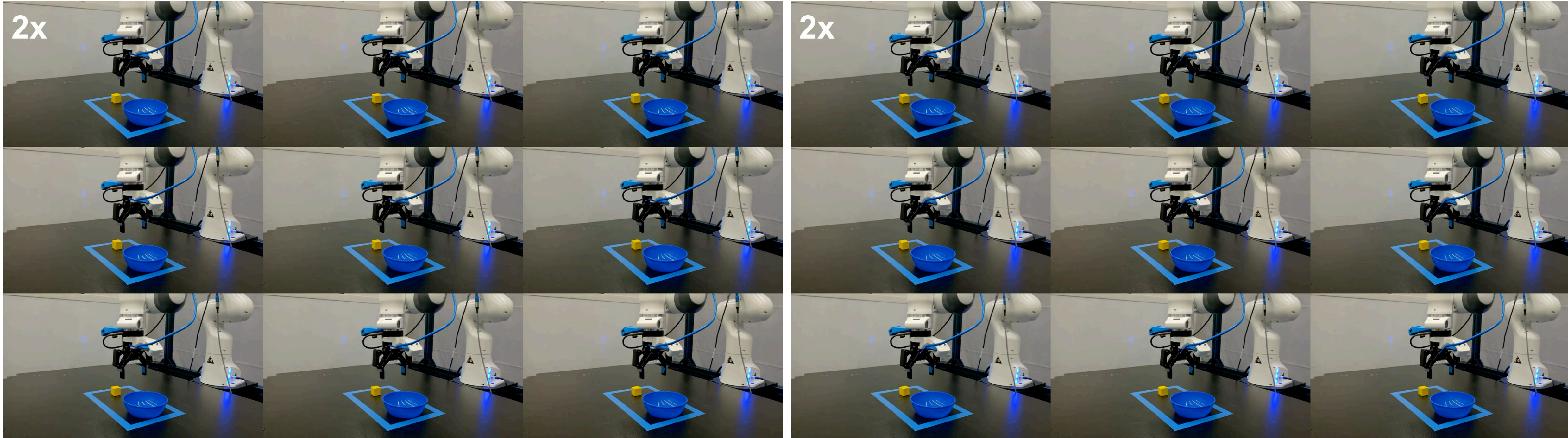


“Set the spoon on the plate”



DSRL Improves Single-Task Diffusion Policies

Diffusion policy trained on 10 human demonstrations



Before DSRL

Goal: "Put the block in the bowl"

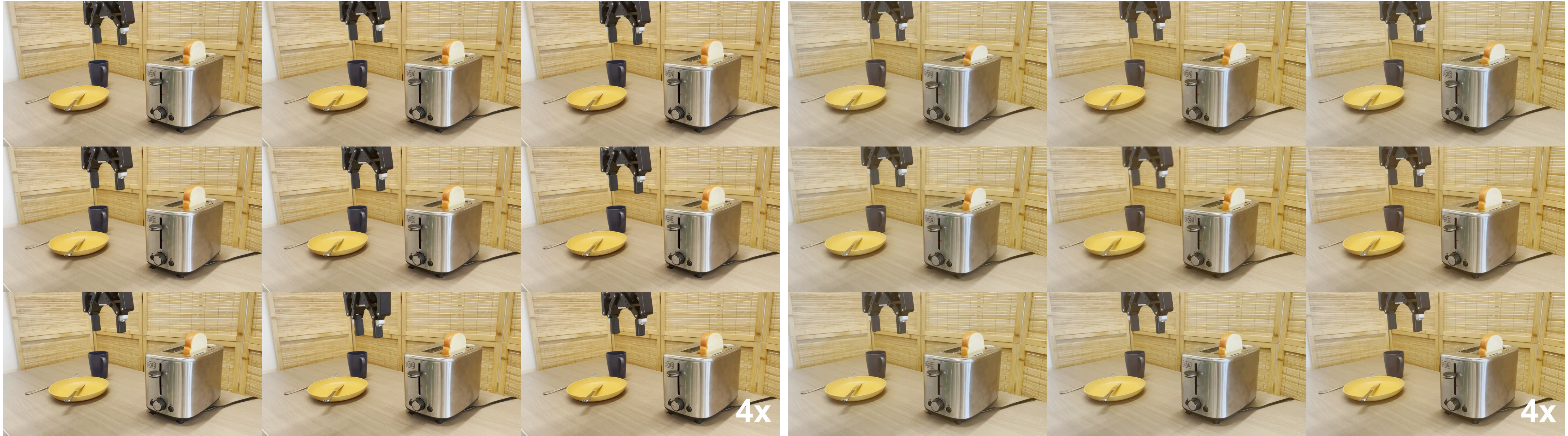
After DSRL

Real-world
training length:
~40 episodes
(~30 minutes)

π_{dp}	RLPD	RLPD + interventions	DSRL
2/10	0/10	0/10	9/10

DSRL Improves VLAs

π_0 from *Physical Intelligence*



Before DSRL

Goal: “Turn on the toaster”

After DSRL

Real-world
training length:
~80 episodes
(~90 minutes)

Task	π_0	DSRL
Turn on toaster	5/20	18/20
Put spoon on plate	15/20	19/20

“the first learning-based policy that can autonomously lace up a shoe”



Right-hand view

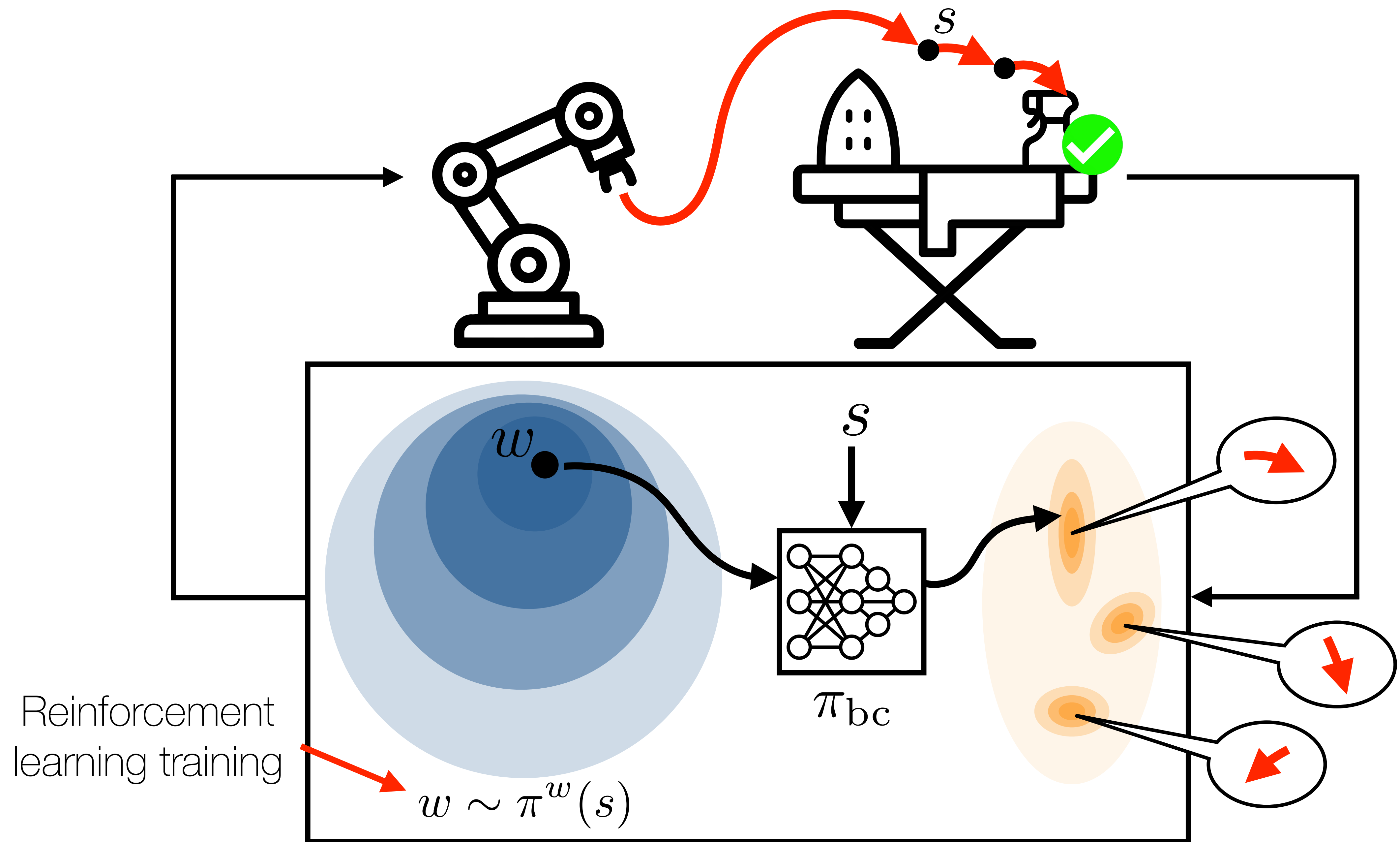


Left-hand view

Offline DSRL

Conceptually, DSRL implements a “support constraint”:

$$\max_{\pi} \mathbb{E}_s[Q(s, \pi(s))] \quad \text{s.t.} \quad \text{support}(\pi(s)) \subseteq \text{support}(\pi_{bc}(s))$$



Offline Diffusion Steering

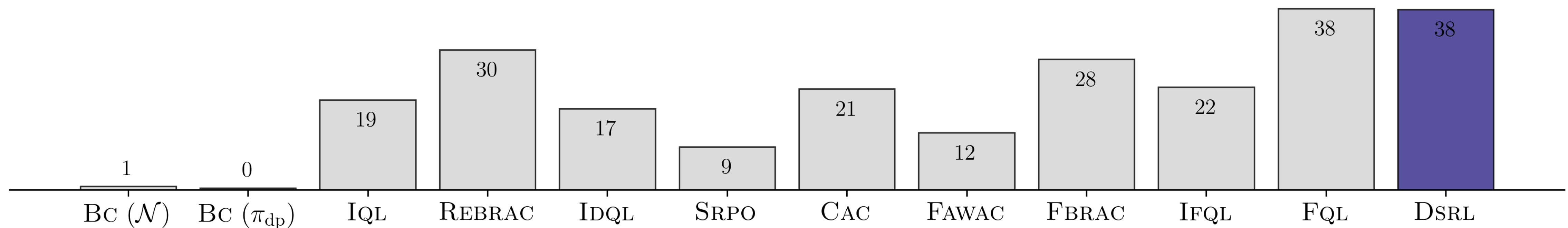
Conceptually, DSRL implements a “support constraint”:

$$\max_{\pi} \mathbb{E}_s[Q(s, \pi(s))] \quad \text{s.t.} \quad \text{support}(\pi(s)) \subseteq \text{support}(\pi_{bc}(s))$$

Given offline dataset $\mathfrak{D} = \{(s_t, a_t, r_t, s_{t+1})\}_t$

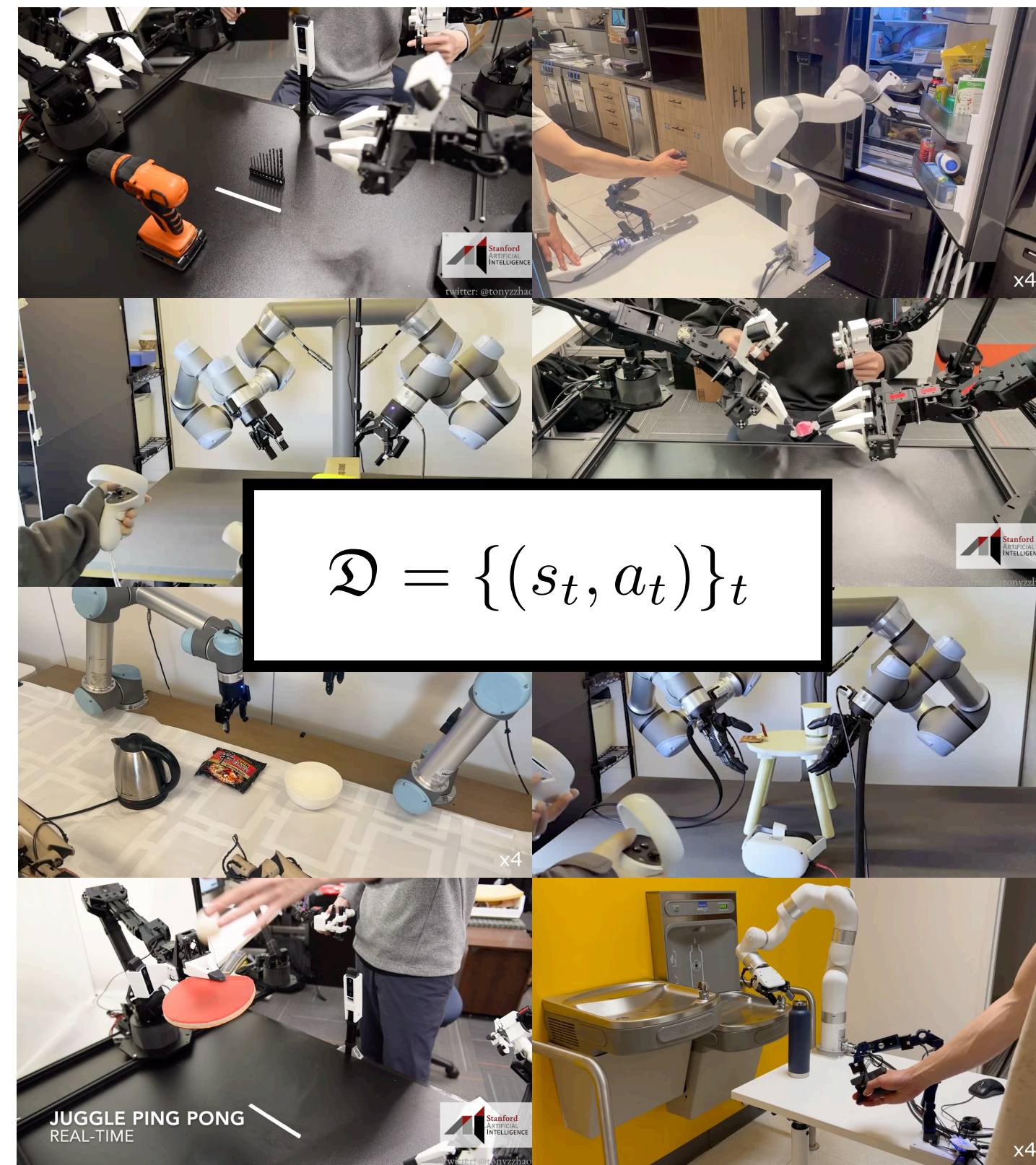
1. Train diffusion policy to predict $a_t \sim \pi_{bc}(s_t)$
2. Use $\mathfrak{D}$ to find optimal “in-distribution” actions

Instantiates conservatism

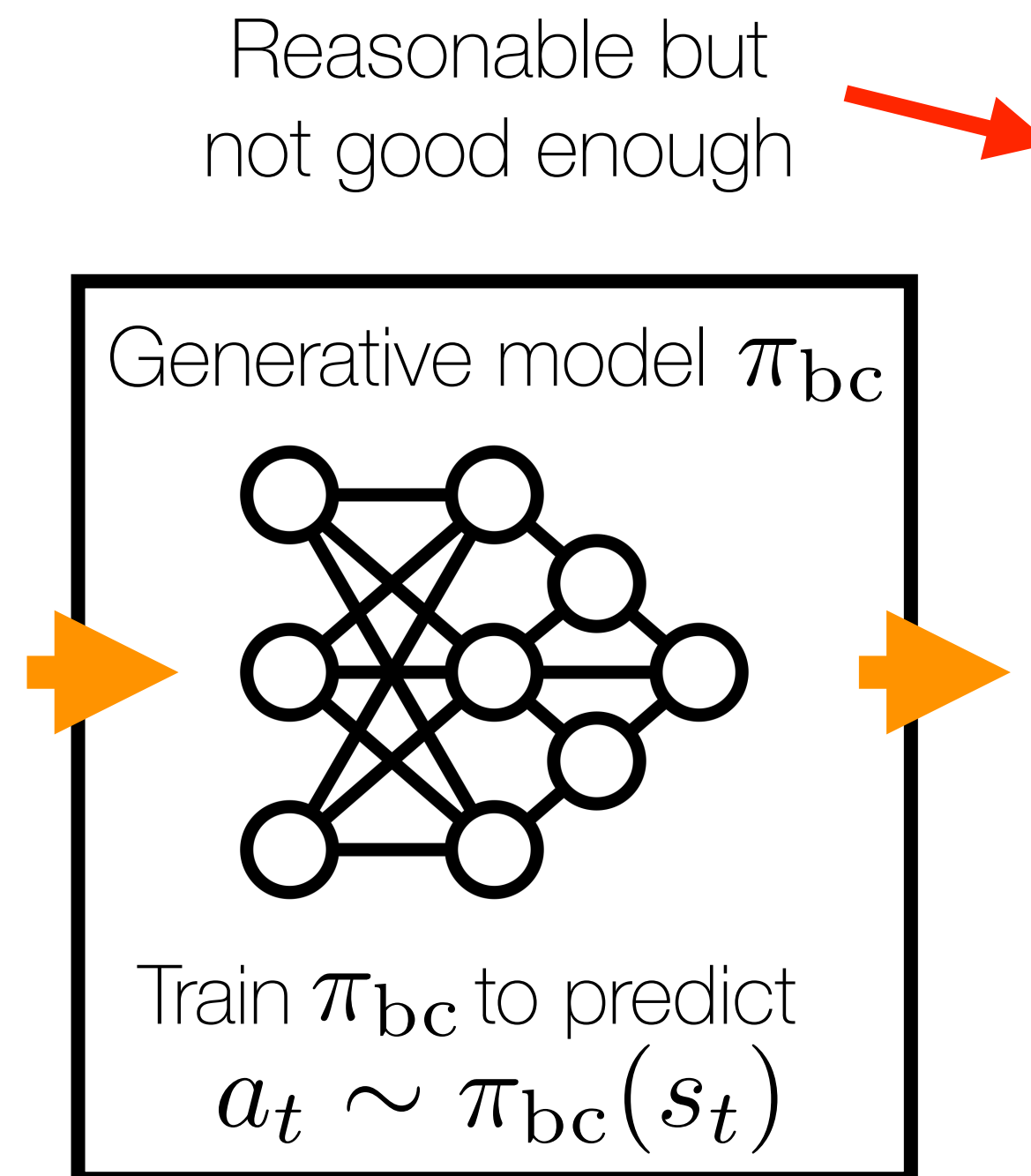


Offline results on OGBench

Learning from experience with human demonstrations



Human demonstrations



Behavioral Cloning (BC)



Deployment

“Pretraining”

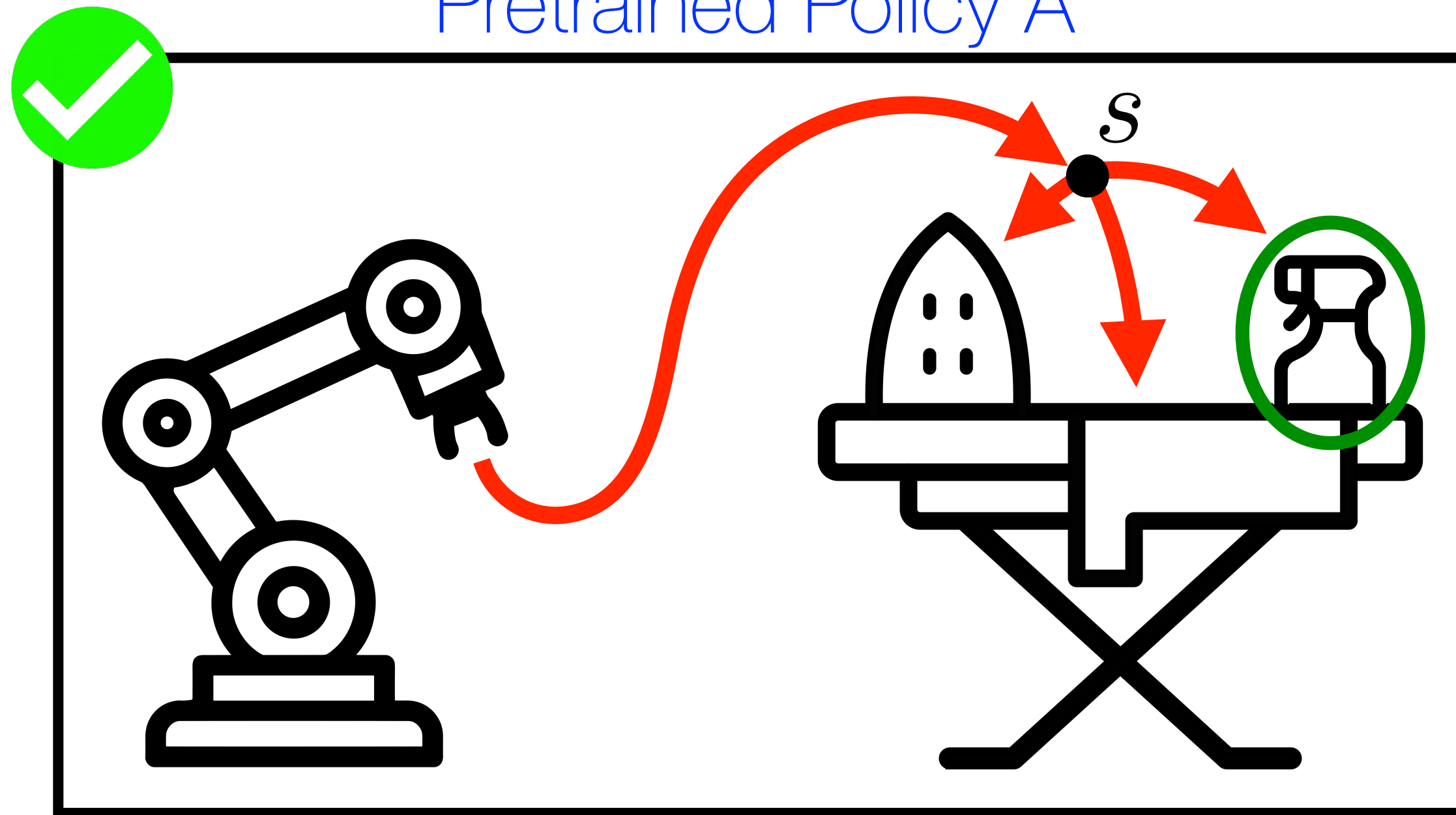
1. How can we efficiently improve a pretrained BC policy in deployment?

2. How should we “pretrain” a policy to ensure it can be efficiently improved?

How should we pretrain to enable online improvement?

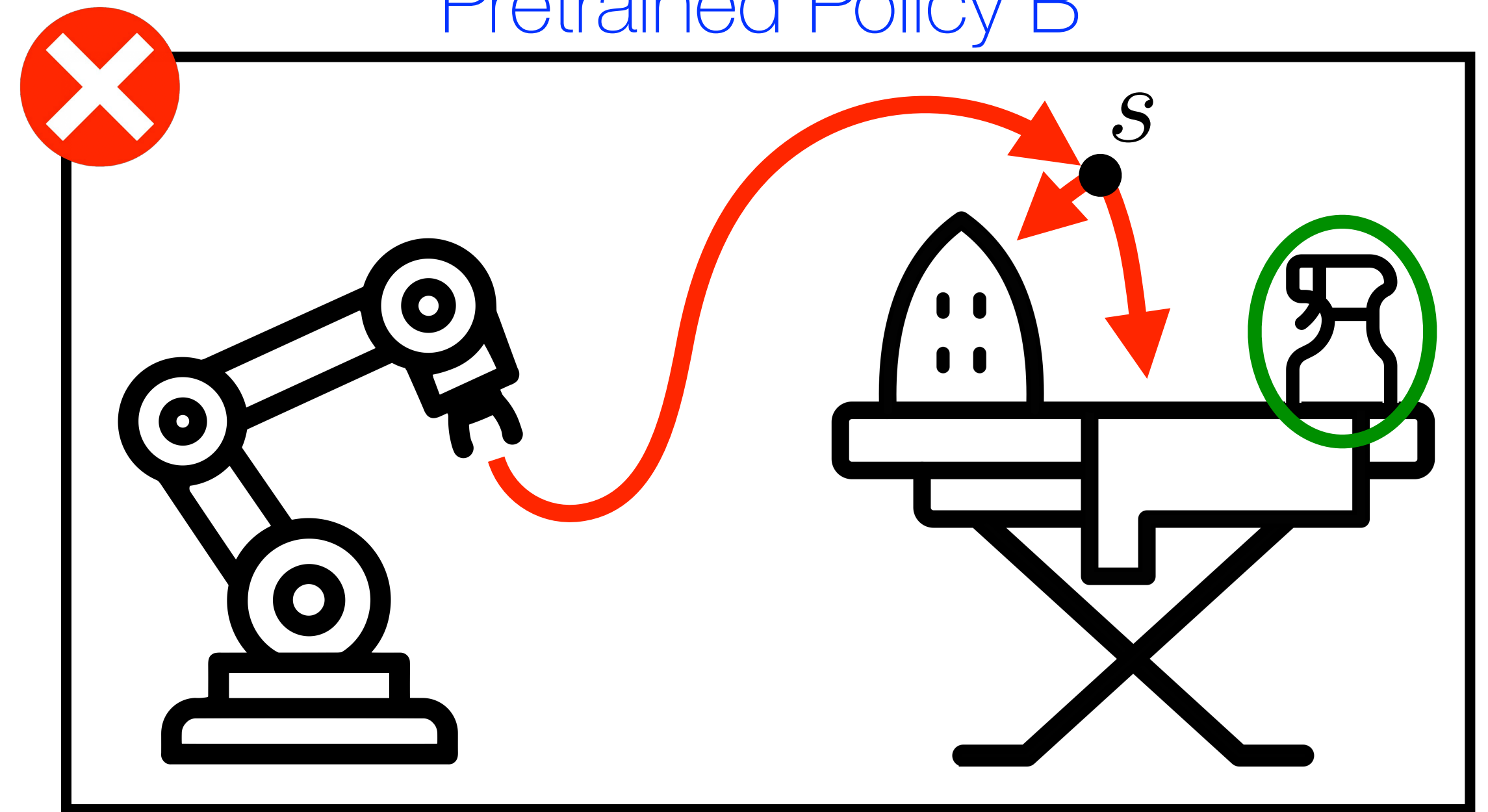
Plays demonstrator behavior when it's sufficient,
collects experience that distinguishes correct behavior when it's not

Pretrained Policy A



Collects experience enabling online learning

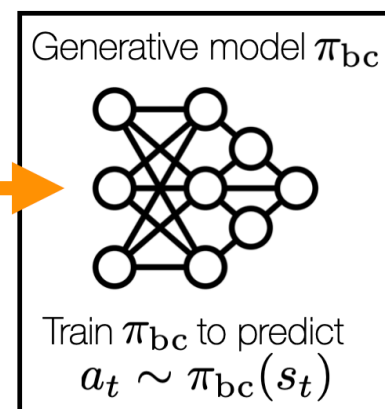
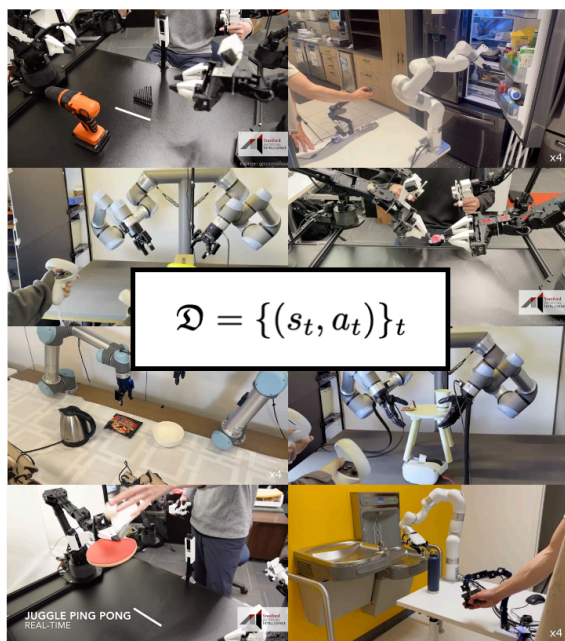
Pretrained Policy B



Fails to collect experience enabling online learning

1. Pretrained policy no worse than BC policy
2. Pretrained policy expresses actions needed for improvement:

$$\forall(s, a) : \pi_{\text{pt}}(a \mid s) \geq \gamma \cdot \pi_{\text{demo}}(a \mid s)$$



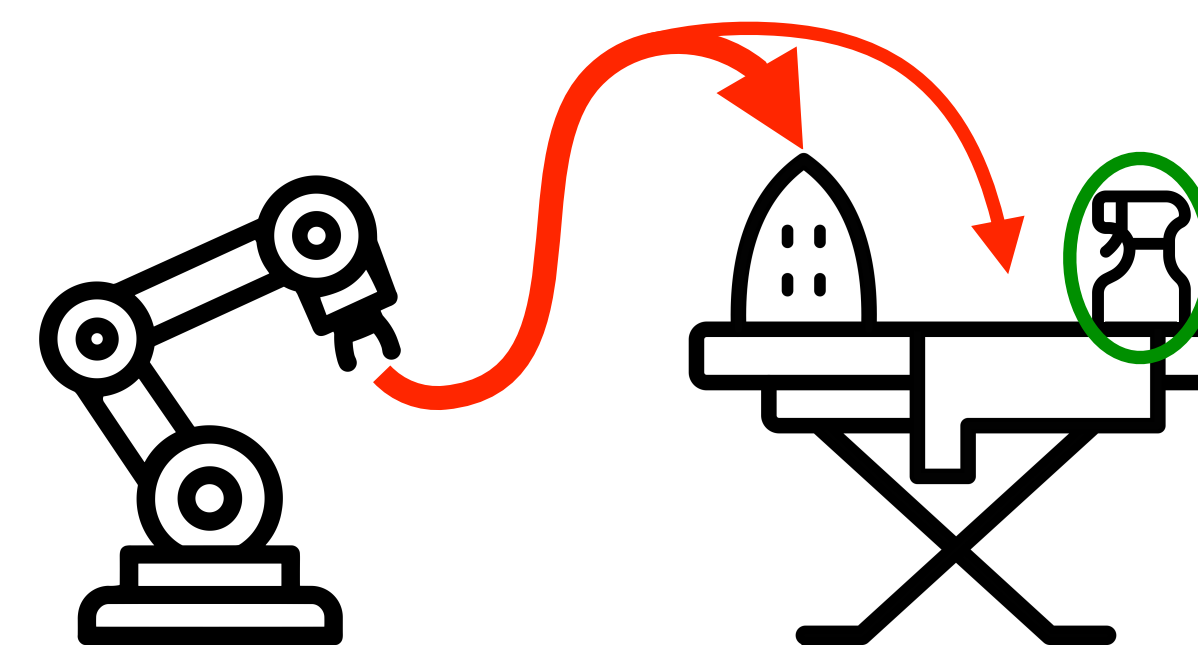
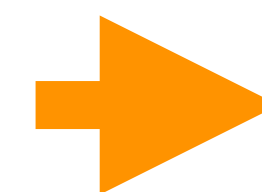
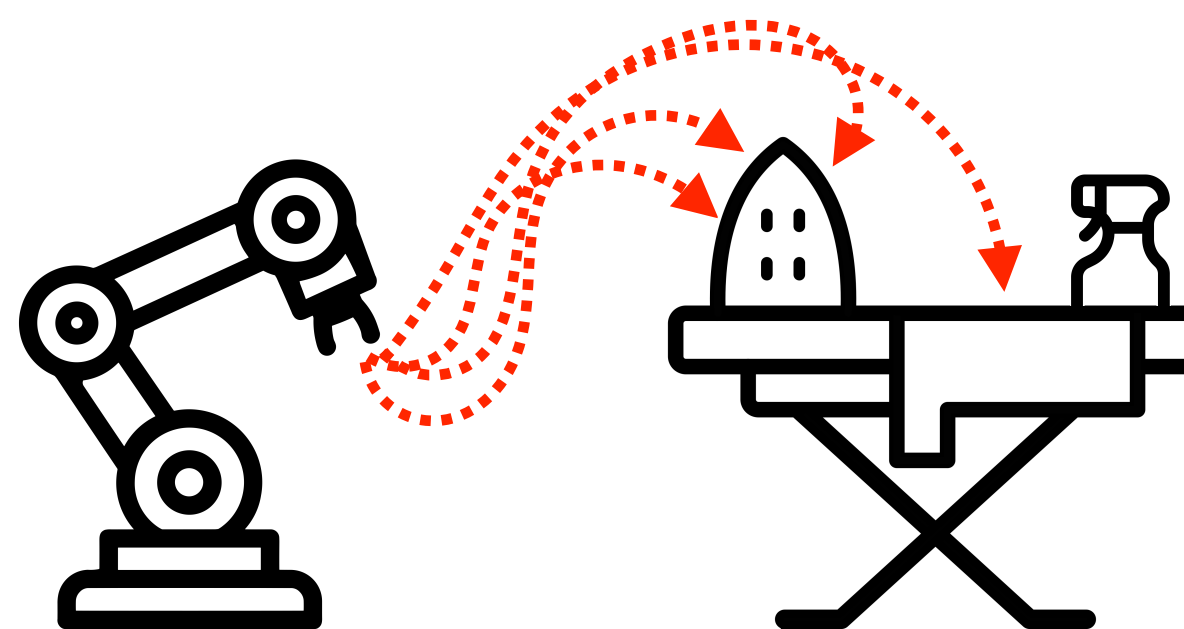
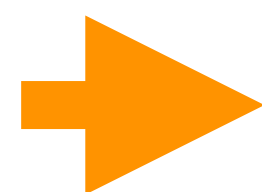
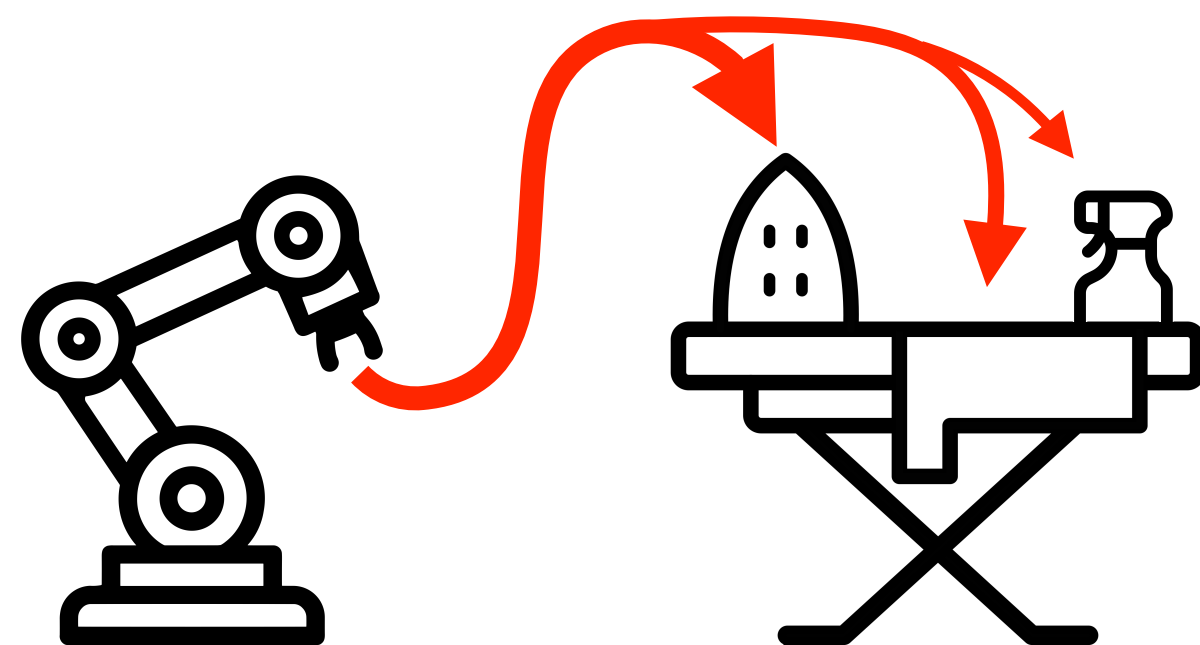
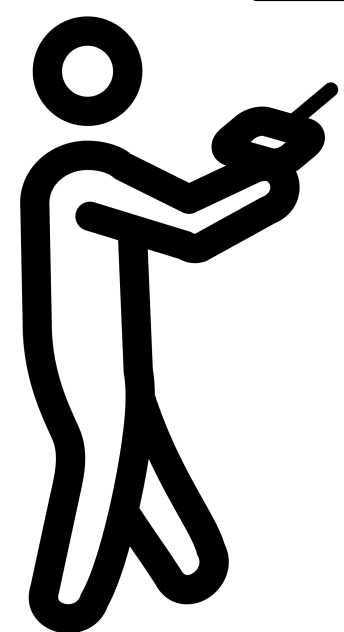
Is BC pretraining sufficient?

Demonstrator

Demonstrations

BC pretraining

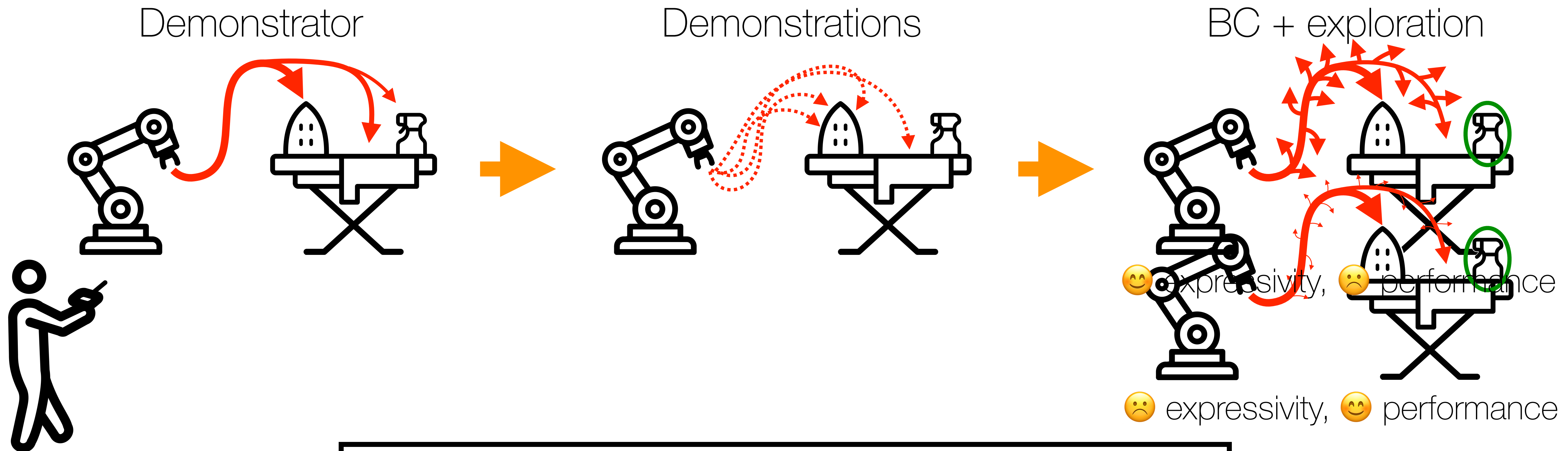
BC policy



Theorem [WDTFL '25]. With constant probability, BC policy will:

1. fail to express all demonstrator actions
2. not collect the experience necessary to improve in deployment

Is BC pretraining + exploration sufficient?



Theorem [WD TFL '25]. BC + exploration either:
(a) expresses actions needed for improvement, or
(b) matches pretrained performance of BC policy

Is BC pretraining + exploration sufficient?

Over-explores

Demonstrations

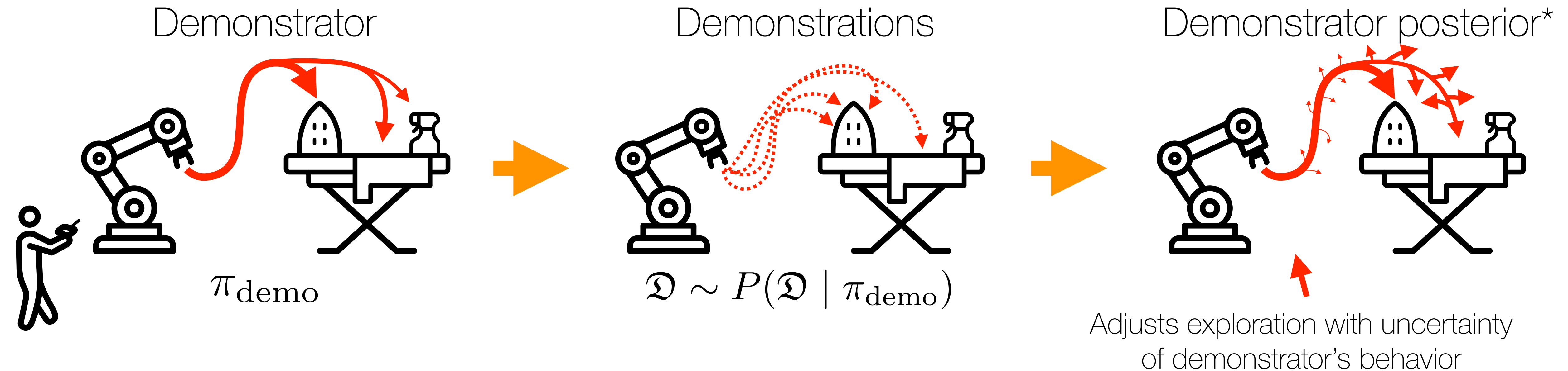
Under-explores

Standard pretraining approaches fail to obtain an effective initialization for online improvement

🙄 expressivity, 😊 performance

Theorem [WDTFL '25]. BC + exploration either:
(a) expresses actions needed for improvement, or
(b) matches pretrained performance of BC policy

A Bayesian perspective on demonstrators



Let's assume uniform prior over demonstrator policy $P_0(\pi)$

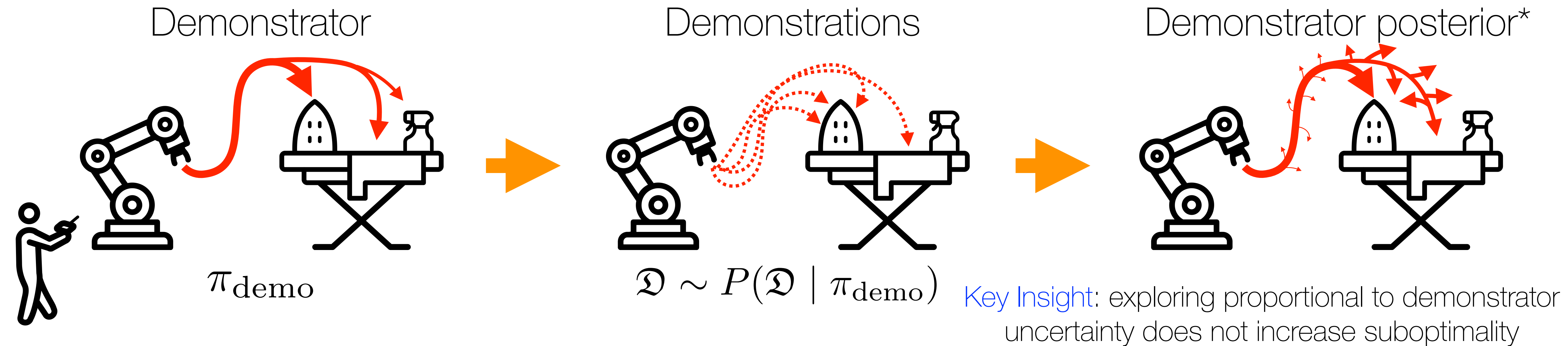
→ Given $\mathcal{D}$ we can form demonstrator posterior $P_{\text{post}}(\pi \mid \mathcal{D})$

→ BC policy is MAP estimate: $\pi_{\text{bc}} = \operatorname{argmax}_{\pi} P_{\text{post}}(\pi \mid \mathcal{D})$

Collapses posterior to a point, doesn't allow for adjusting exploration with demonstrator uncertainty

*Posterior predictive distribution 44

A Bayesian perspective on demonstrators



Theorem [WDTFL '25]. Demonstrator posterior has:

1. Same pretrained performance as BC policy
2. Maximum action expressivity out of all such policies

Rather than pretraining with BC, we should pretrain to fit the demonstrator posterior

In tabular settings: $\pi_{\text{pt}}(a \mid s) \approx \frac{1}{A+H} \cdot \pi_{\text{demo}}(a \mid s)$

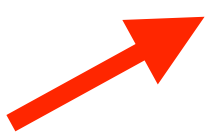
Implementing “Posterior Behavioral Cloning”

Step 1: Estimate posterior variance via ensembling

Algorithm 1 Posterior Variance Approximation via Ensembled Prediction

- 1: **input:** demonstration dataset $\mathcal{D}$, ensemble size K , posterior model class $\mathcal{F}$
 - 2: **for** $\ell = 1, 2, \dots, K$ **do**
 - 3: Set $\mathcal{D}_\ell$ to “noisy” version of $\mathcal{D}$
 - 4: Fit f_ℓ by solving $f_\ell \leftarrow \arg \min_{f \in \mathcal{F}} \sum_{(s, \tilde{a}) \in \mathcal{D}_\ell} \|f_\ell(s) - \tilde{a}\|_2^2$
 - 5: $\text{cov}(\cdot) \leftarrow \sum_{\ell=1}^K (f_\ell(\cdot) - \bar{f}(\cdot))(f_\ell(\cdot) - \bar{f}(\cdot))^\top$ for $\bar{f}(\cdot) \leftarrow \frac{1}{K} \sum_{\ell=1}^K f_\ell(\cdot)$
 - 6: **return** $\text{cov}(\cdot)$
-

Corresponds to true posterior in Gaussian case



Only requires standard supervised learning/BC

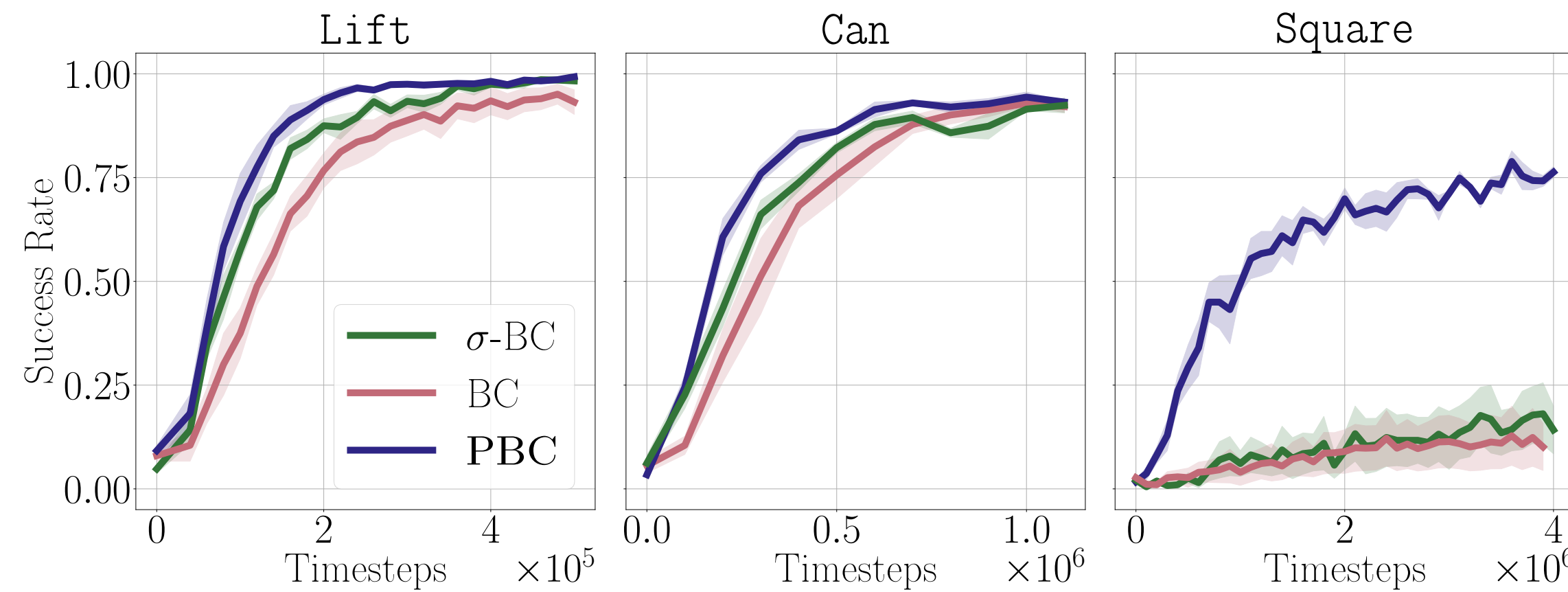


Step 2: Fit posterior demonstrator policy

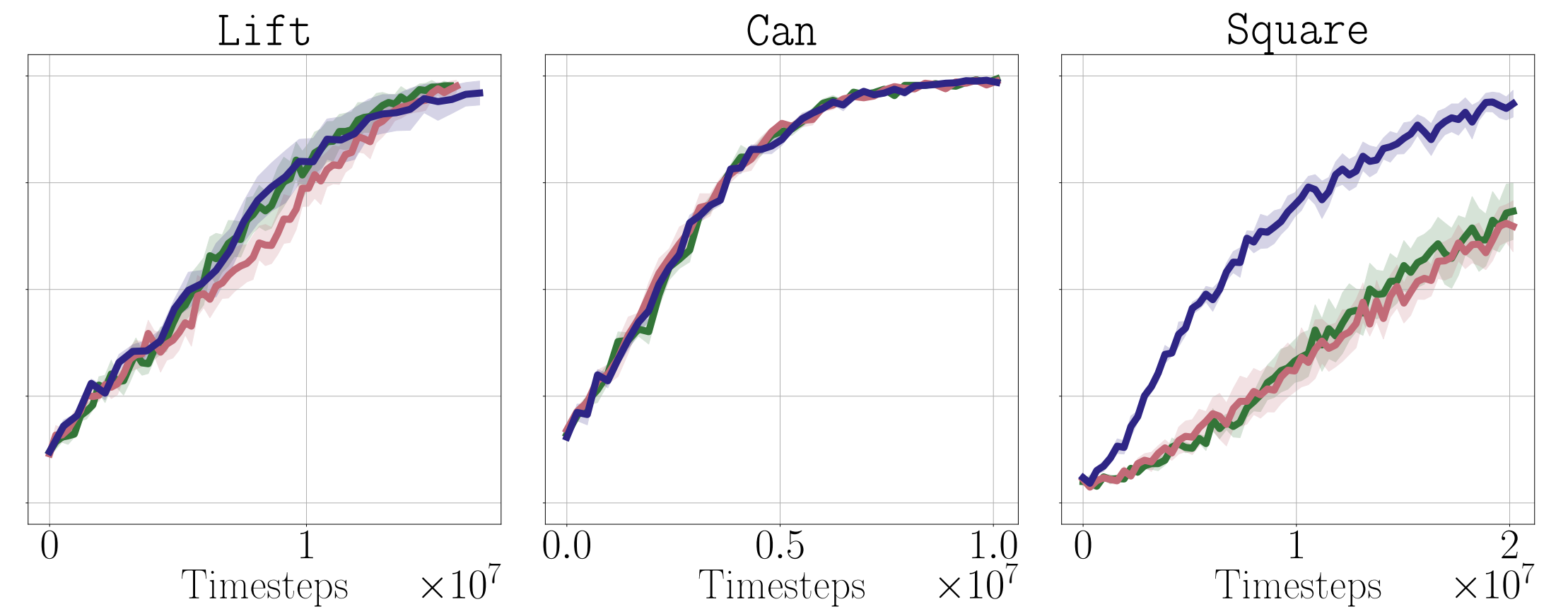
Algorithm 2 Posterior Behavioral Cloning (POSTBC)

- 1: **input:** demonstration dataset $\mathcal{D}$, generative model class $\hat{\pi}^\theta$, posterior covariance $\text{cov}(\cdot)$, posterior weight α
 - 2: **for** $i = 1, 2, 3, \dots$ **do**
 - 3: Sample batch $\mathcal{B}_i \sim \text{unif}(\mathcal{D})$
 - 4: For all $(s, a) \in \mathcal{B}_i$, sample $w_s \sim \mathcal{N}(0, \text{cov}(s))$, and set $\tilde{\mathcal{B}}_i \leftarrow \{(s, a + \alpha \cdot w_s) : s \in \mathcal{B}_i\}$
 - 5: Take gradient step on $\hat{\pi}^\theta$ for loss computed on $\tilde{\mathcal{B}}_i$
 - 6: **return** $\hat{\pi}^\theta$
-

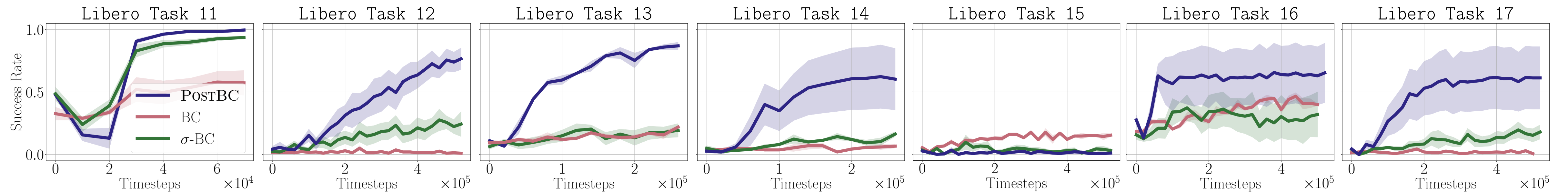
After Improvement: BC policy $\ll$ Demonstrator posterior policy



DSRL on Robomimic

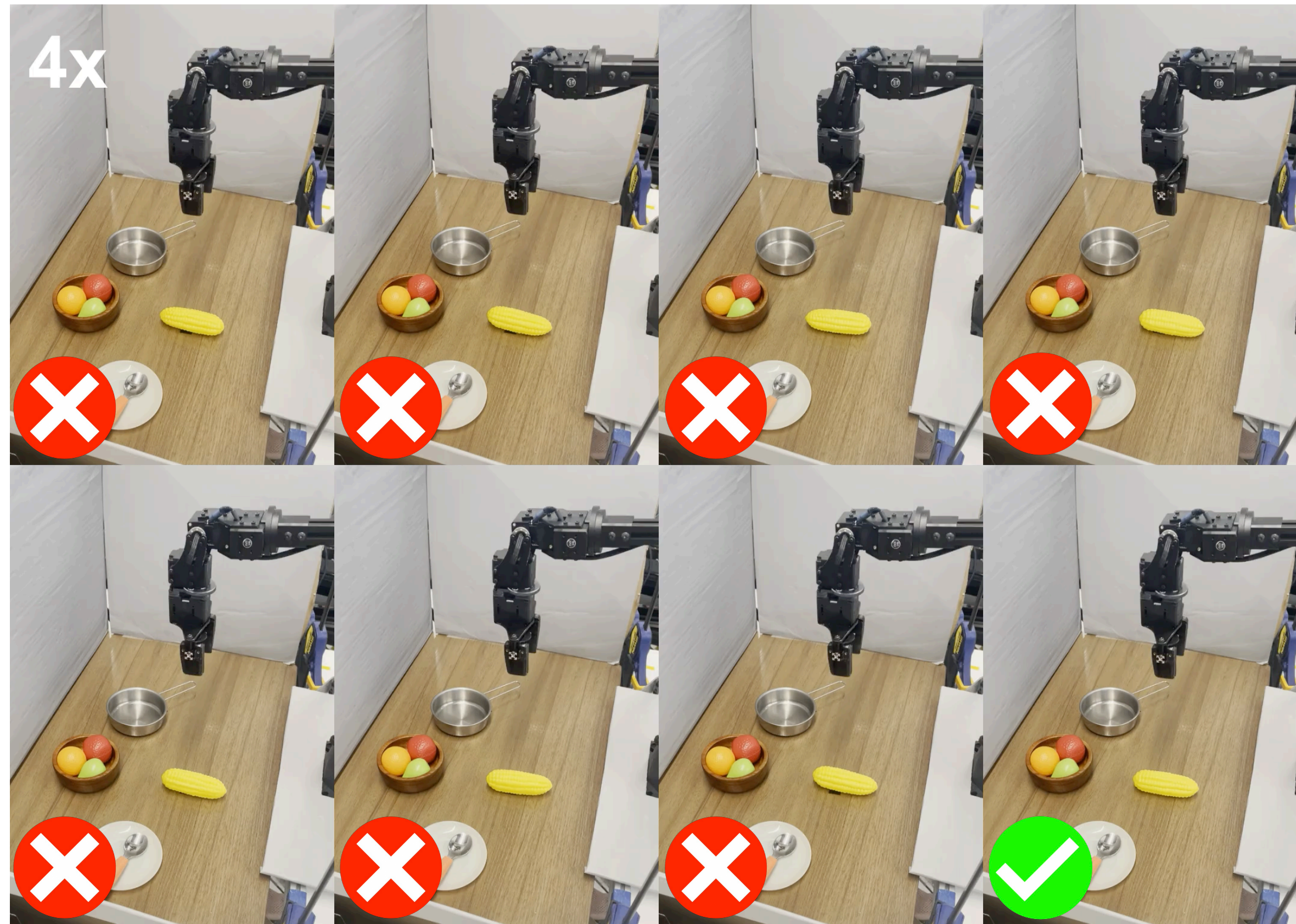


DPPO on Robomimic

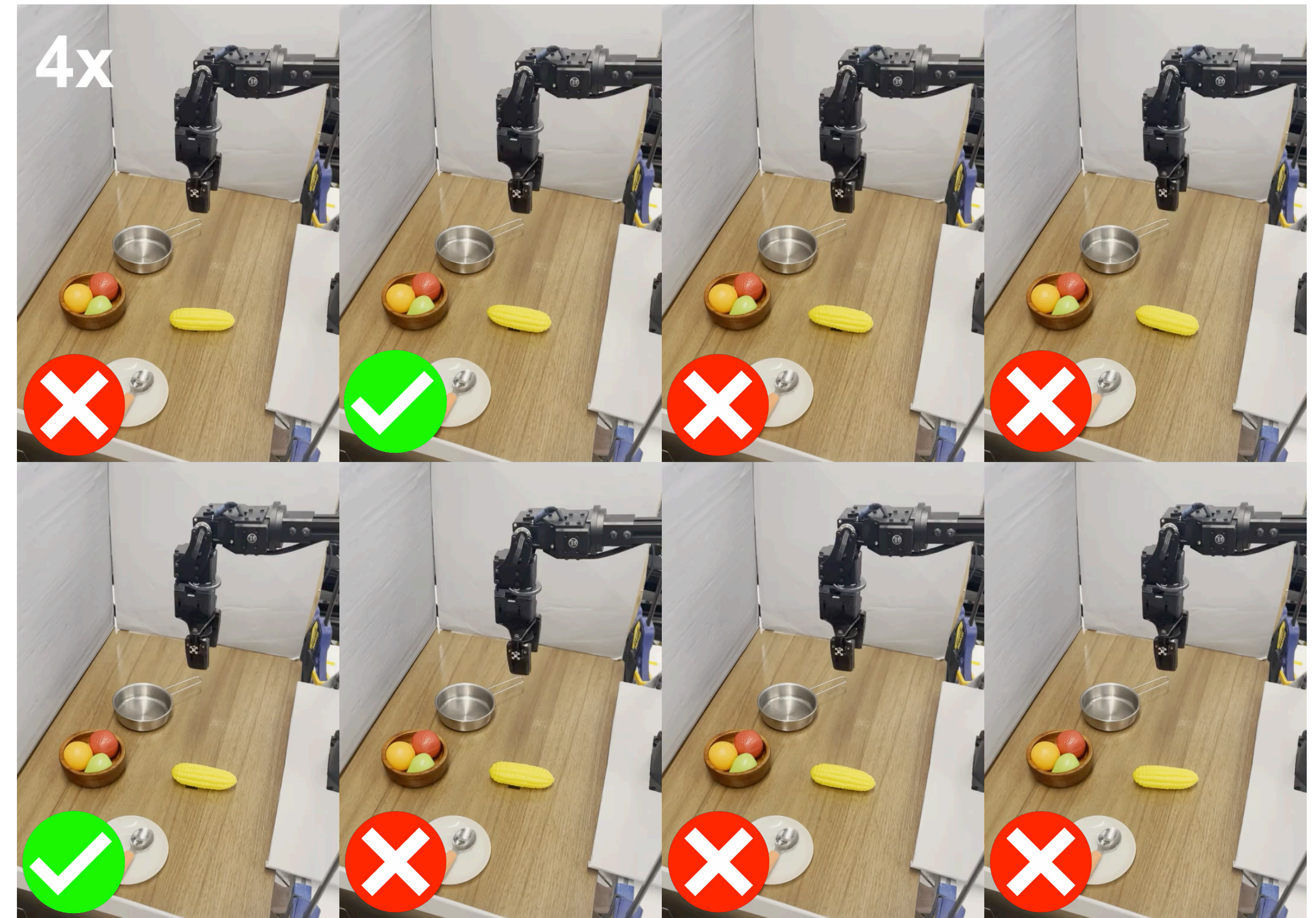


DSRL on Libero

Pretrained: BC policy $\approx$ Demonstrator posterior policy

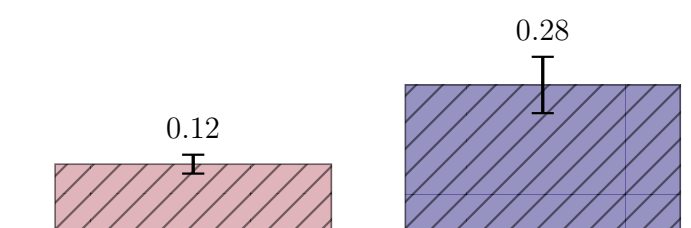


BC (pretrained)

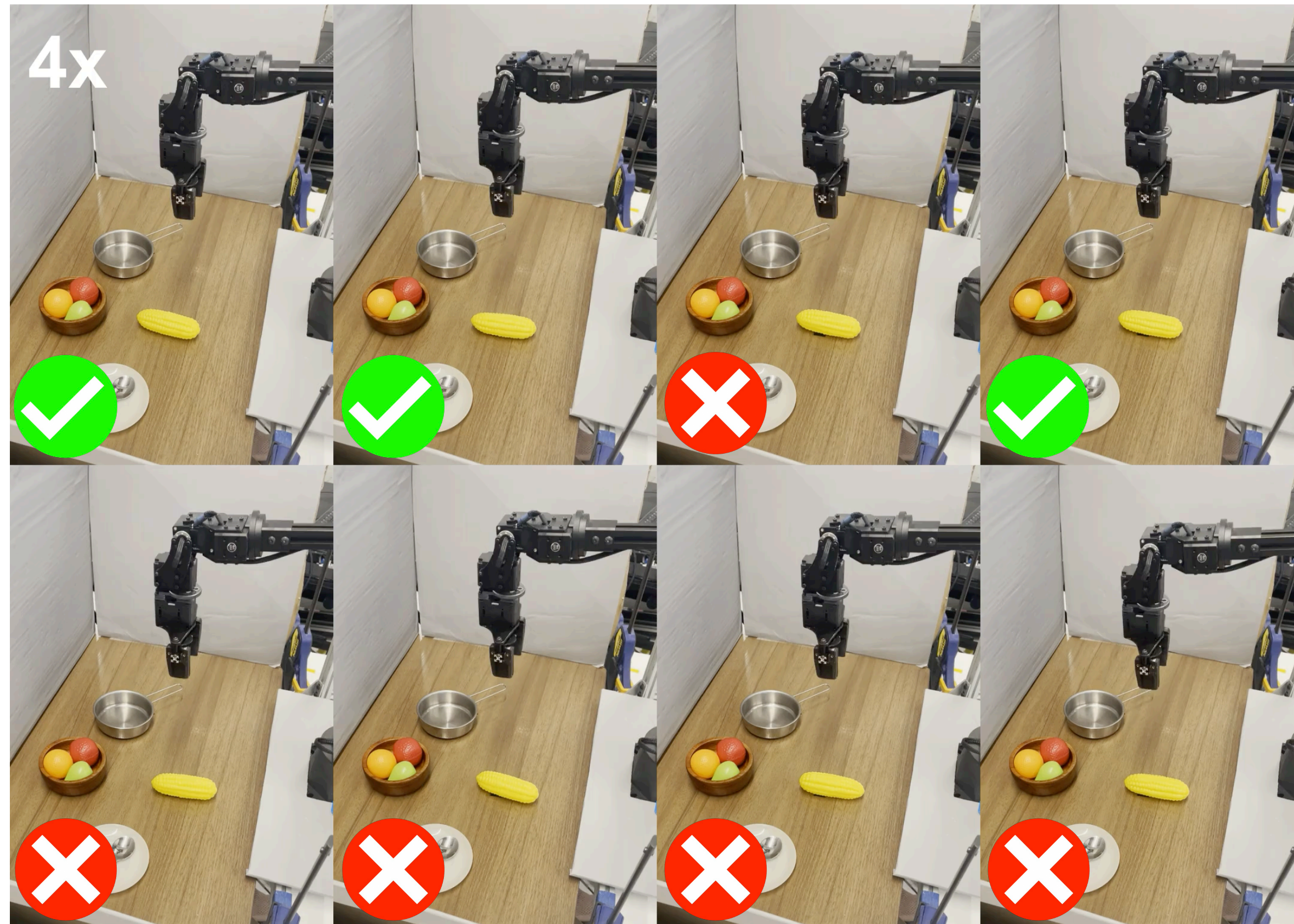


Demonstrator posterior (pretrained)

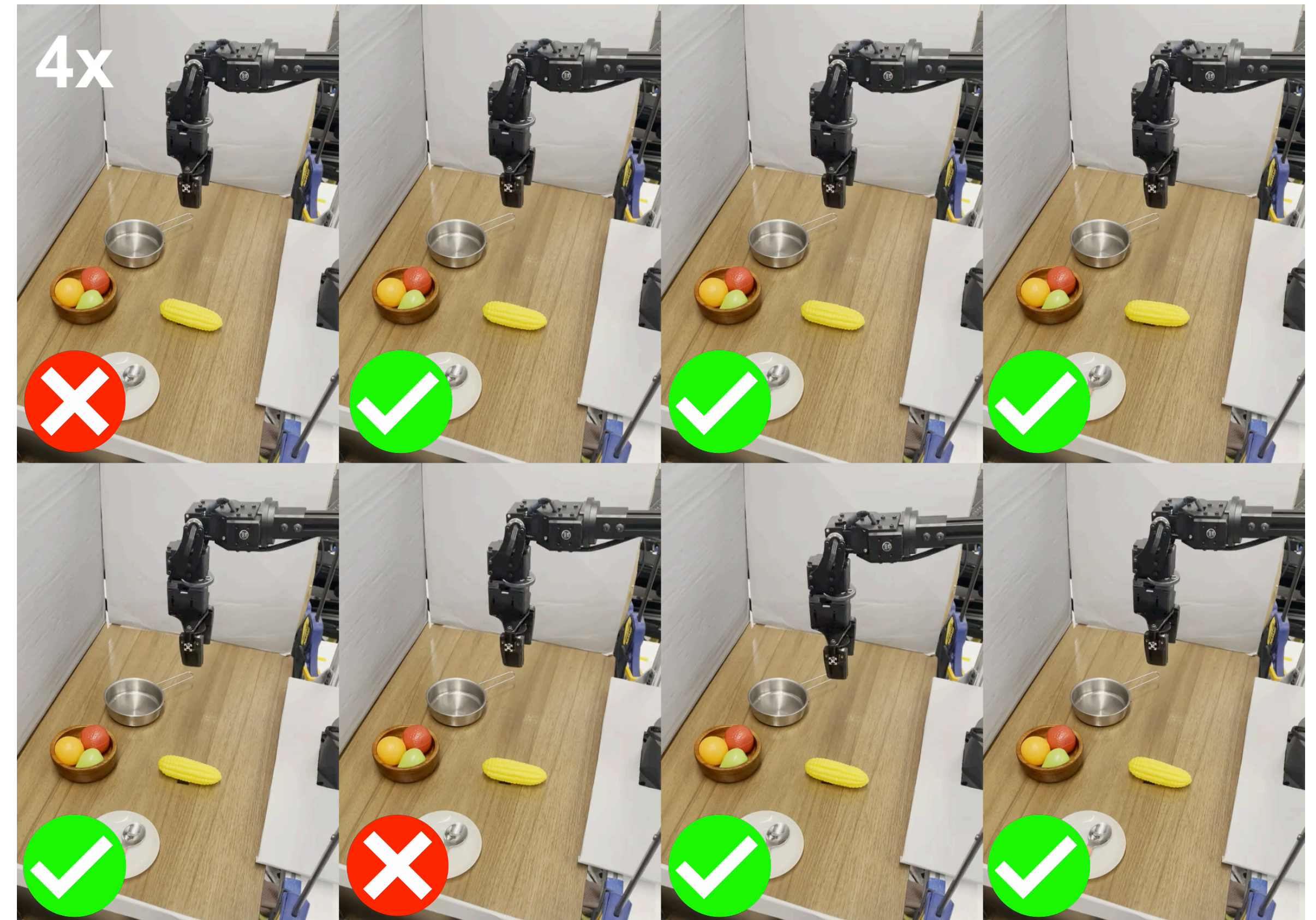
Goal: “put the corn in the pot”



After Improvement: BC policy $\ll$ Demonstrator posterior policy

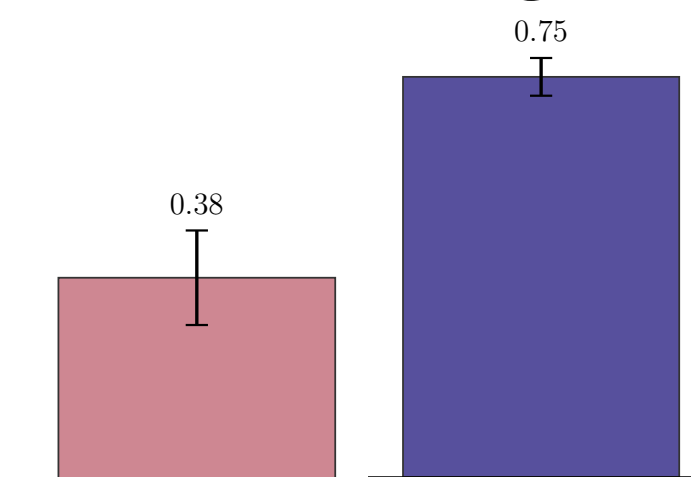


BC (after improvement)

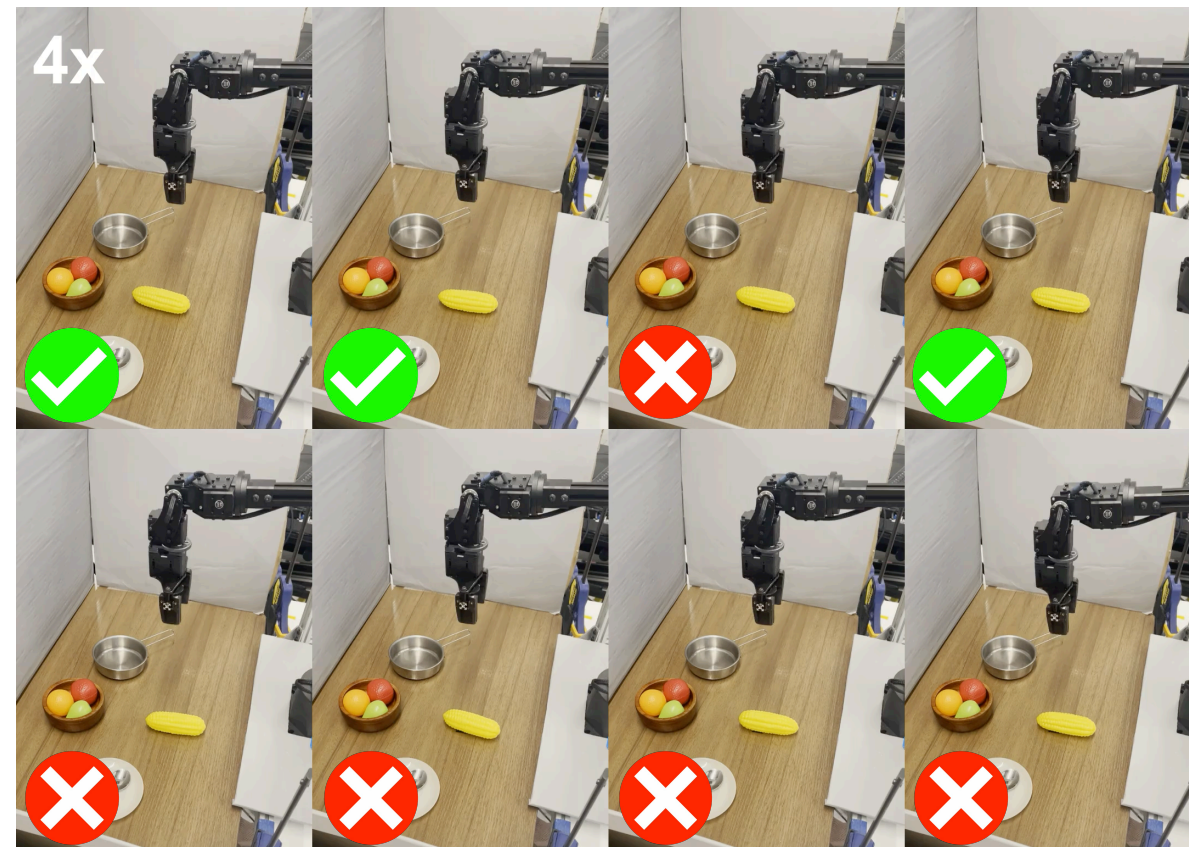


Demonstrator posterior (after improvement)

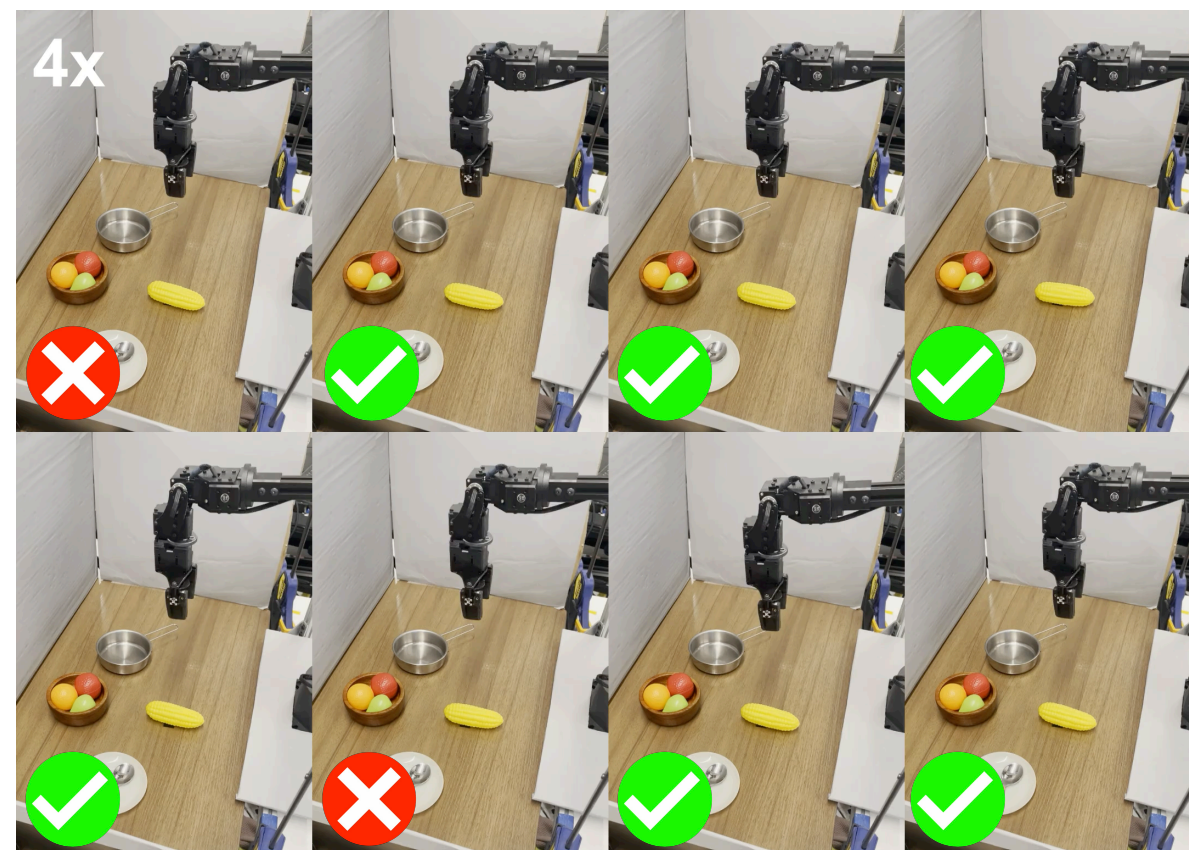
Goal: “put the corn in the pot”



After Improvement: BC policy $\ll$ Demonstrator posterior policy



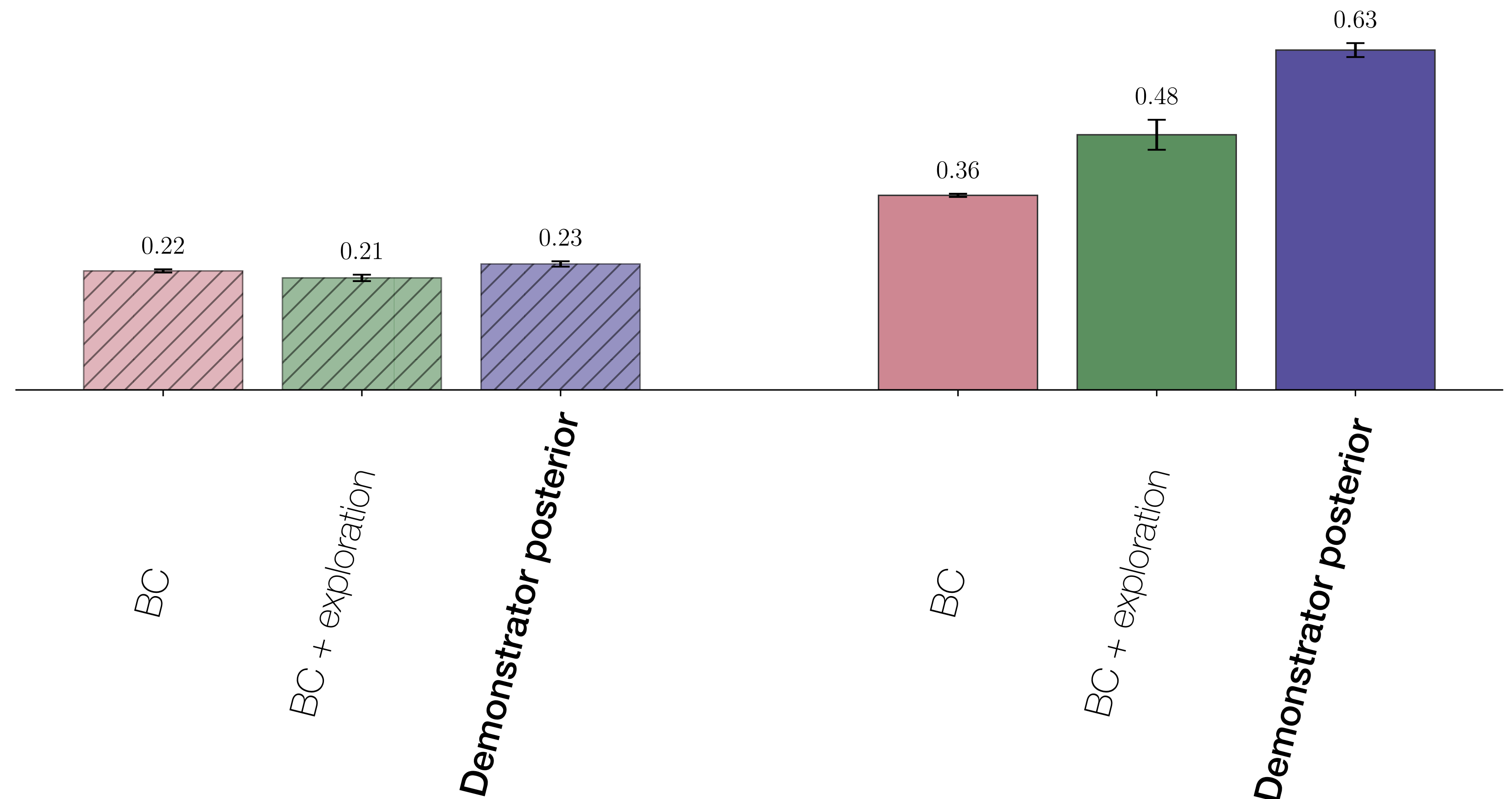
BC (after improvement)



Demonstrator posterior (after improvement)

Pretrained

After Improvement



Takeaway 1: BC pretraining encodes strong behavioral priors that, if properly exploited, can enable highly efficient online policy improvement

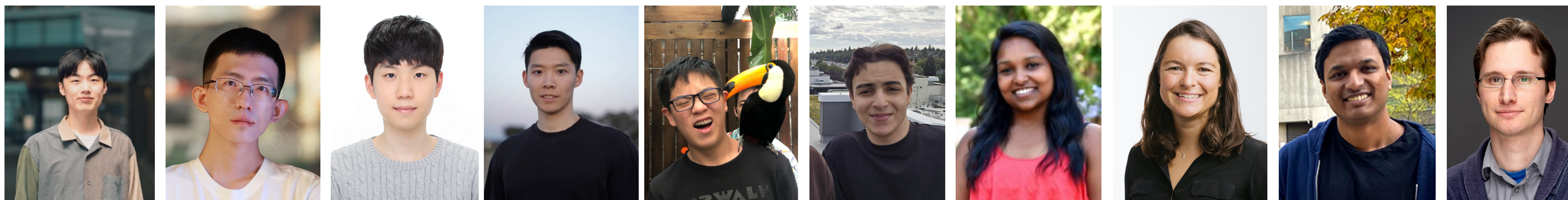


Takeaway 2: We can improve the usefulness of these priors for online improvement by understanding where they are actually reliable

Thank You!

WNZPYNGL. “Steering Your Diffusion Policy with Latent Space Reinforcement Learning”.
CoRL, 2025 (Best Paper Nomination).

WDTFL. “Posterior Behavioral Cloning: Pretraining BC Policies for Efficient RL Finetuning”.
In Submission, 2025.



Can DSRL utilize offline data?

Challenge: Need “noise” actions w to run DSRL. How do we utilize offline datasets where we only have “standard” transitions (s, a, s') ?

Idea: Train two critics

- Q^A trained via standard TD learning over original action space
- Q^W trained by distilling Q^A over noise action space
- Optimize noise policy π^W by maximizing Q^W

Noise-Aliased DSRL

$$\min_{Q^A} \mathbb{E}_{(s,a,r,s') \sim \mathcal{B}, a' \sim \pi_{dp}^W(s', \pi^W(s'))} [(Q^A(s, a) - r - \gamma \bar{Q}^A(s', a'))^2]$$

$$\min_{Q^W} \mathbb{E}_{s \sim \mathcal{B}, w \sim \mathcal{N}(0, I)} [(Q^W(s, w) - Q^A(s, \pi_{dp}^W(s, w)))^2]$$

$$\max_{\pi^W} \mathbb{E}_{s \sim \mathcal{B}} [Q^W(s, \pi^W(s))]$$

Noise-Aliased DSRL enables fully offline learning

Noise-Aliased DSRL

$$\min_{Q^A} \mathbb{E}_{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}') \sim \mathfrak{B}, \mathbf{a}' \sim \pi_{\text{dp}}^{\mathcal{W}}(\mathbf{s}', \pi^{\mathcal{W}}(\mathbf{s}'))} [(Q^A(\mathbf{s}, \mathbf{a}) - r - \gamma \bar{Q}^A(\mathbf{s}', \mathbf{a}'))^2]$$

$$\min_{Q^{\mathcal{W}}} \mathbb{E}_{\mathbf{s} \sim \mathfrak{B}, \mathbf{w} \sim \mathcal{N}(0, I)} [(Q^{\mathcal{W}}(\mathbf{s}, \mathbf{w}) - Q^A(\mathbf{s}, \pi_{\text{dp}}^{\mathcal{W}}(\mathbf{s}, \mathbf{w})))^2]$$

$$\max_{\pi^{\mathcal{W}}} \mathbb{E}_{\mathbf{s} \sim \mathfrak{B}} [Q^{\mathcal{W}}(\mathbf{s}, \pi^{\mathcal{W}}(\mathbf{s}))]$$

How do we usually think about offline RL?

Principle of Conservatism: train your policy to maximize reward over “in-distribution” actions

If we train π_{dp} on some offline dataset $\mathfrak{D}$, and run Noise-Aliased DSRL on this same $\mathfrak{D}$

→ π_{dp} will only play “in-distribution” actions by construction, so Noise-Aliased DSRL only optimizes over “in-distribution” actions, and so implements conservatism!