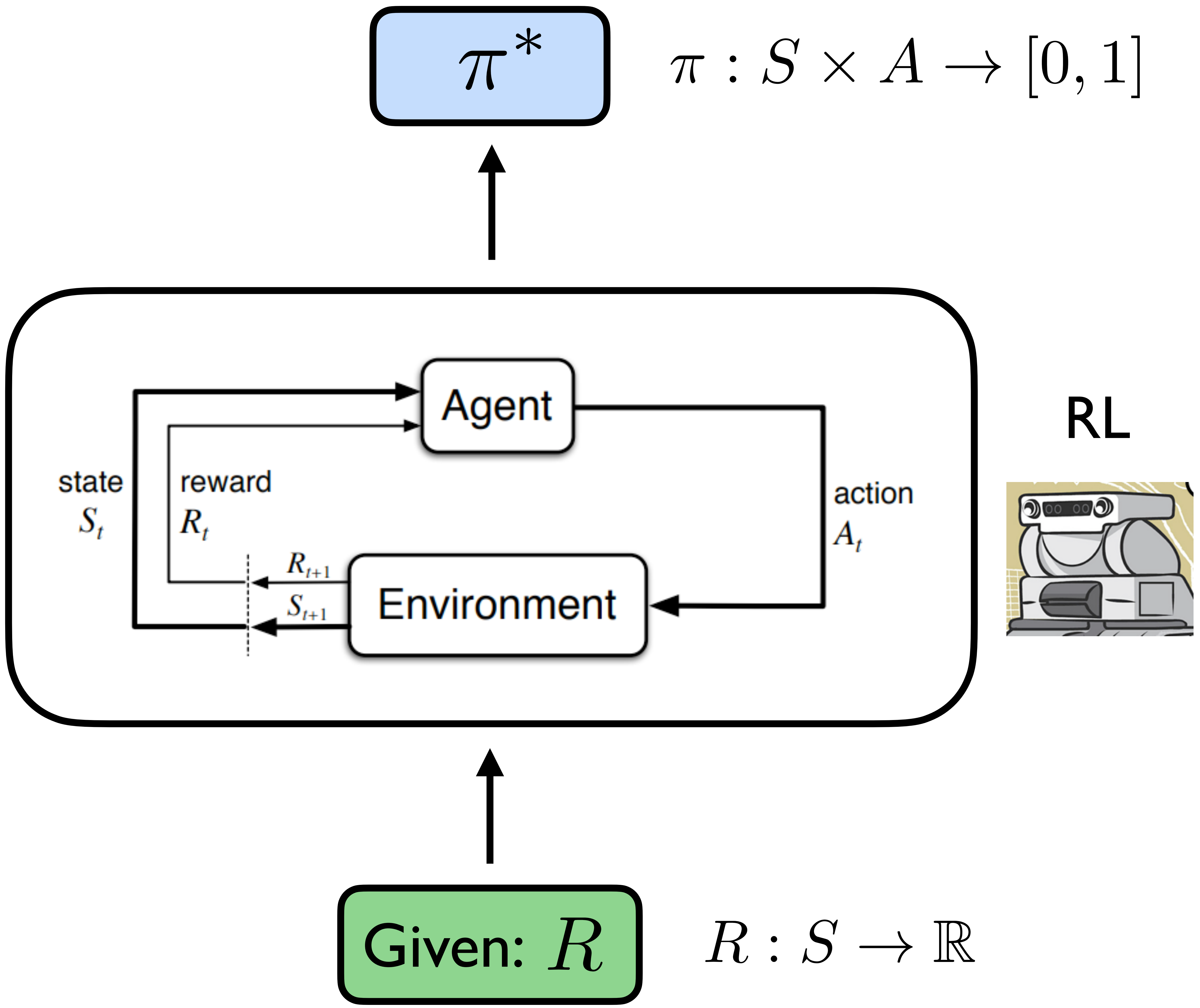
CS 690: Human-Centric Machine Learning

Prof. Scott Niekum

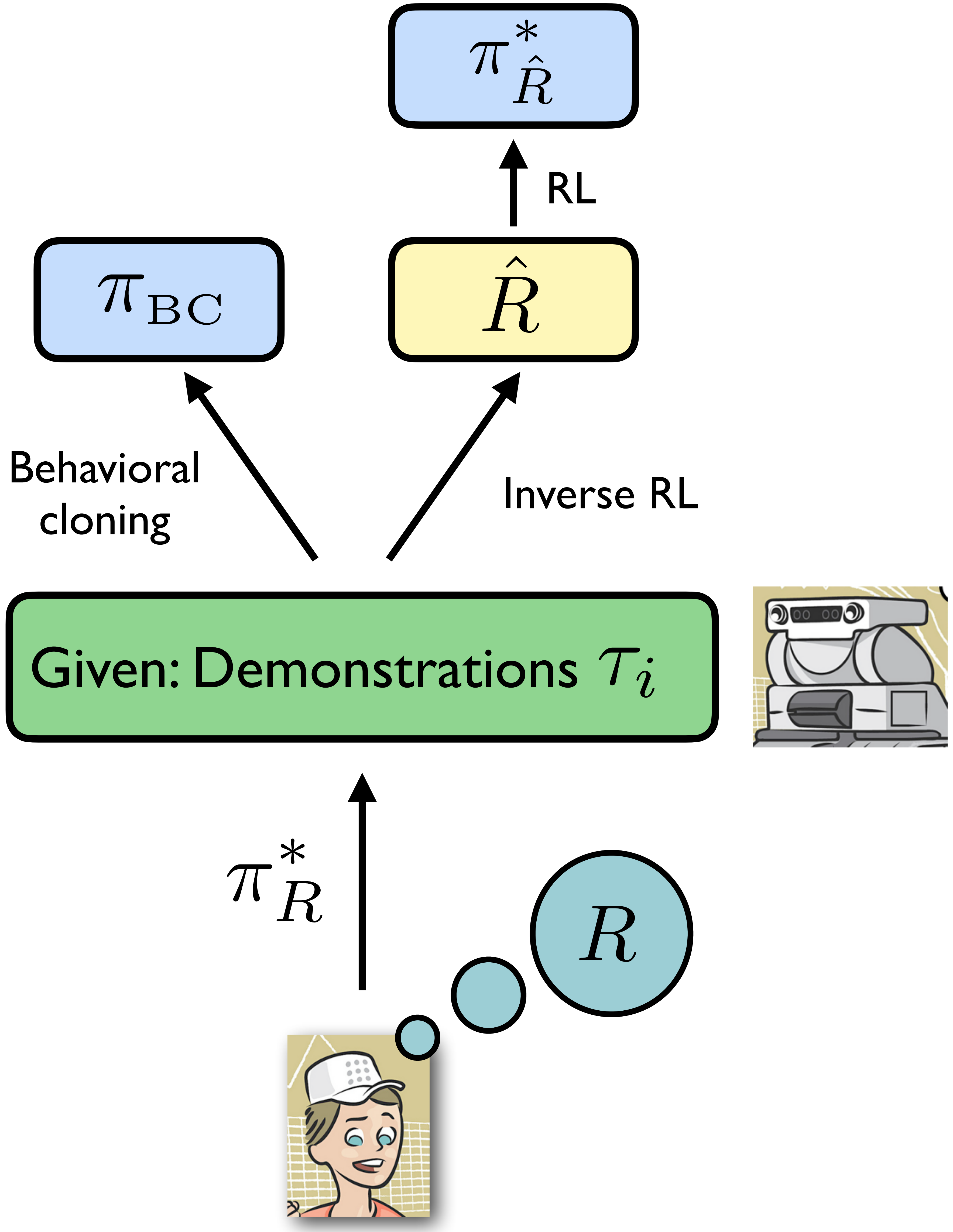
Behavioral Cloning

Reinforcement Learning

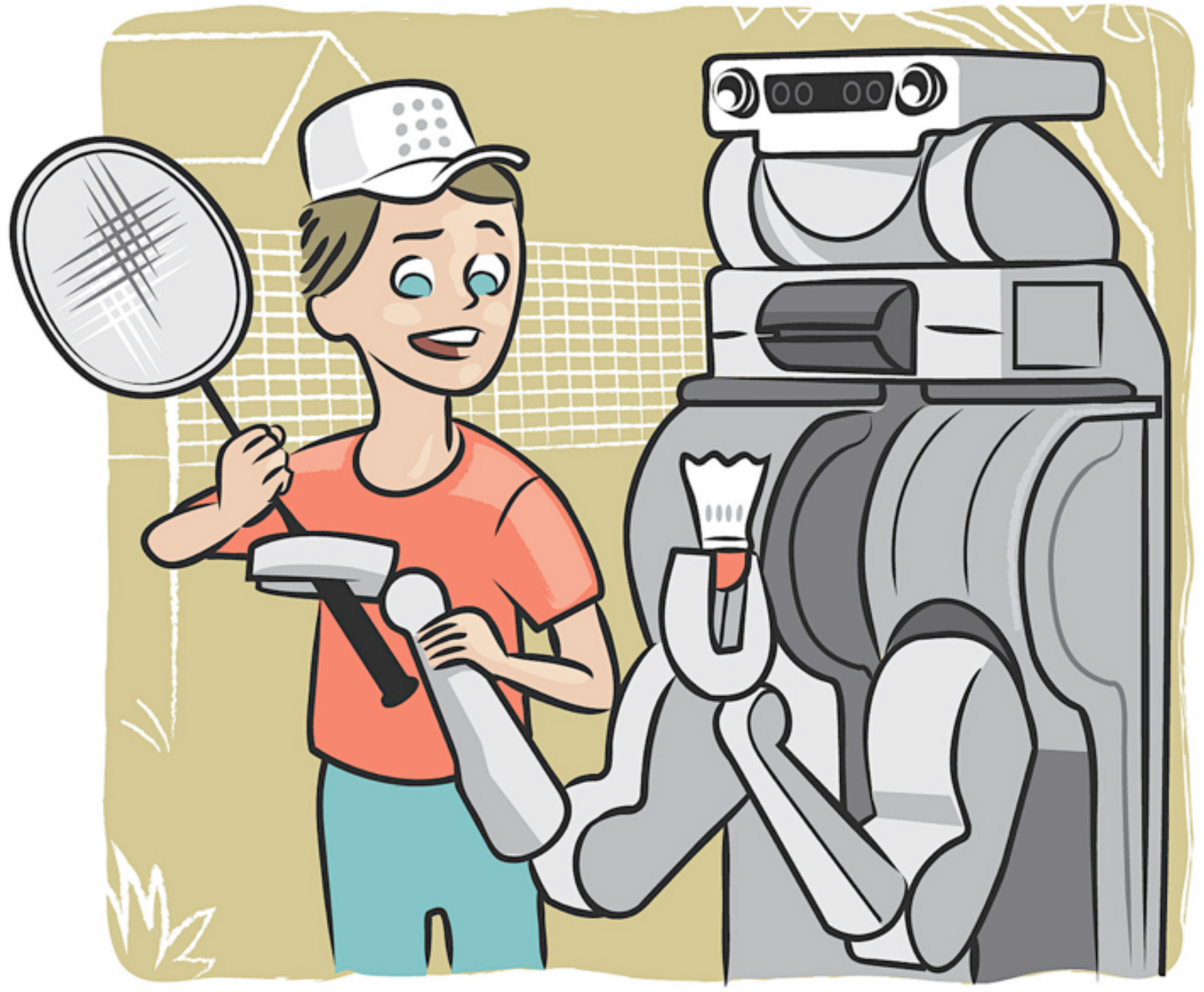
$$V_R^{\pi} = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi\right]$$



Imitation Learning



Why learn from demonstrations?



- Natural and expressive
- No expert knowledge required
- Valuable human intuition
- Program new tasks as-needed

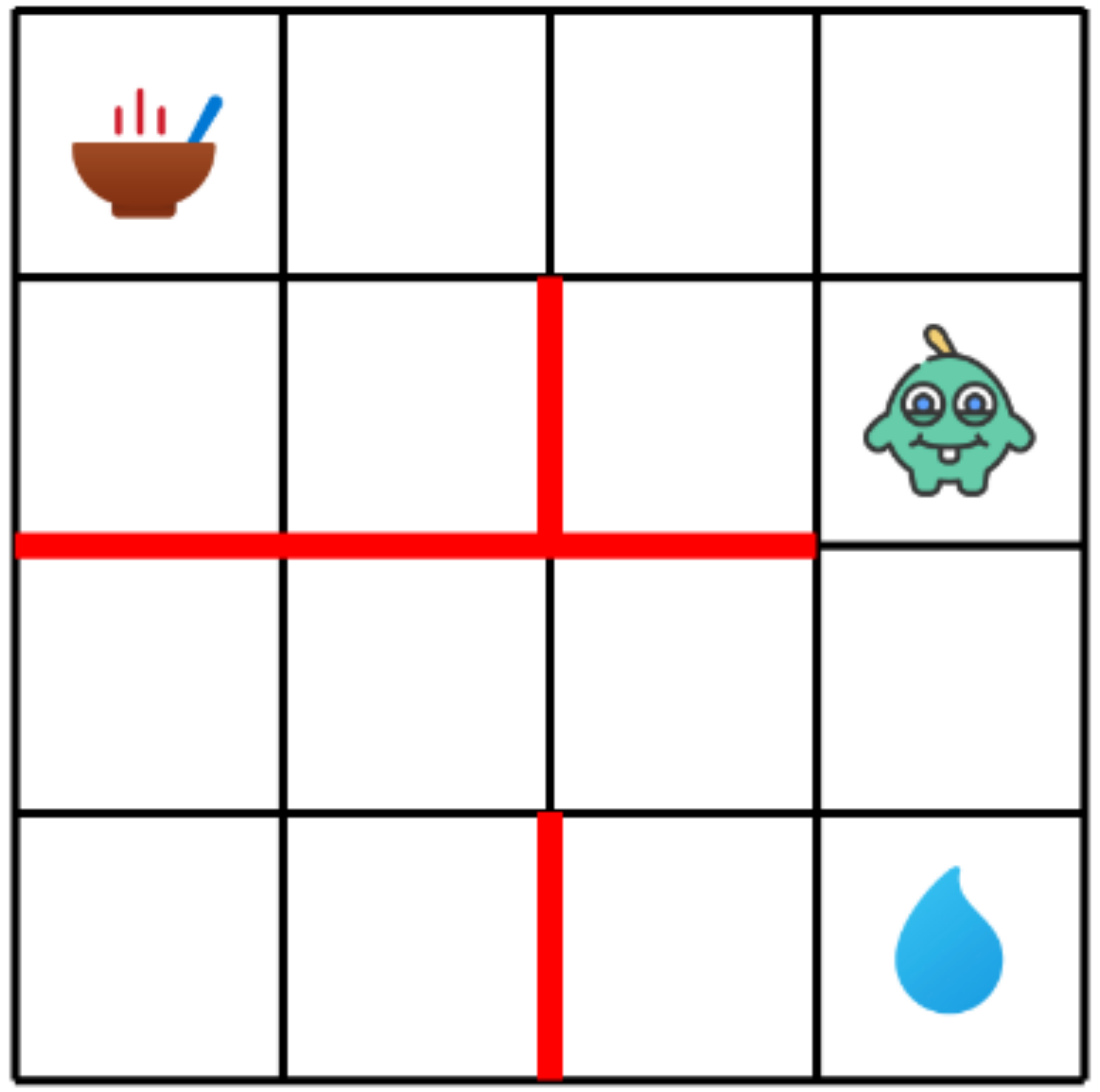
General purpose robot → Specific task → Expert engineer

End user

Why learn from demonstrations?

The Perils of Trial-and-Error Reward Design: Misdesign through Overfitting and Invalid Task Specifications

Serena Booth[1,2,3], W. Bradley Knox[1,2,5], Julie Shah[3], Scott Niekum[2,4], Peter Stone[2,6], Alessandro Allievi[1,2]



Why learn from demonstrations?

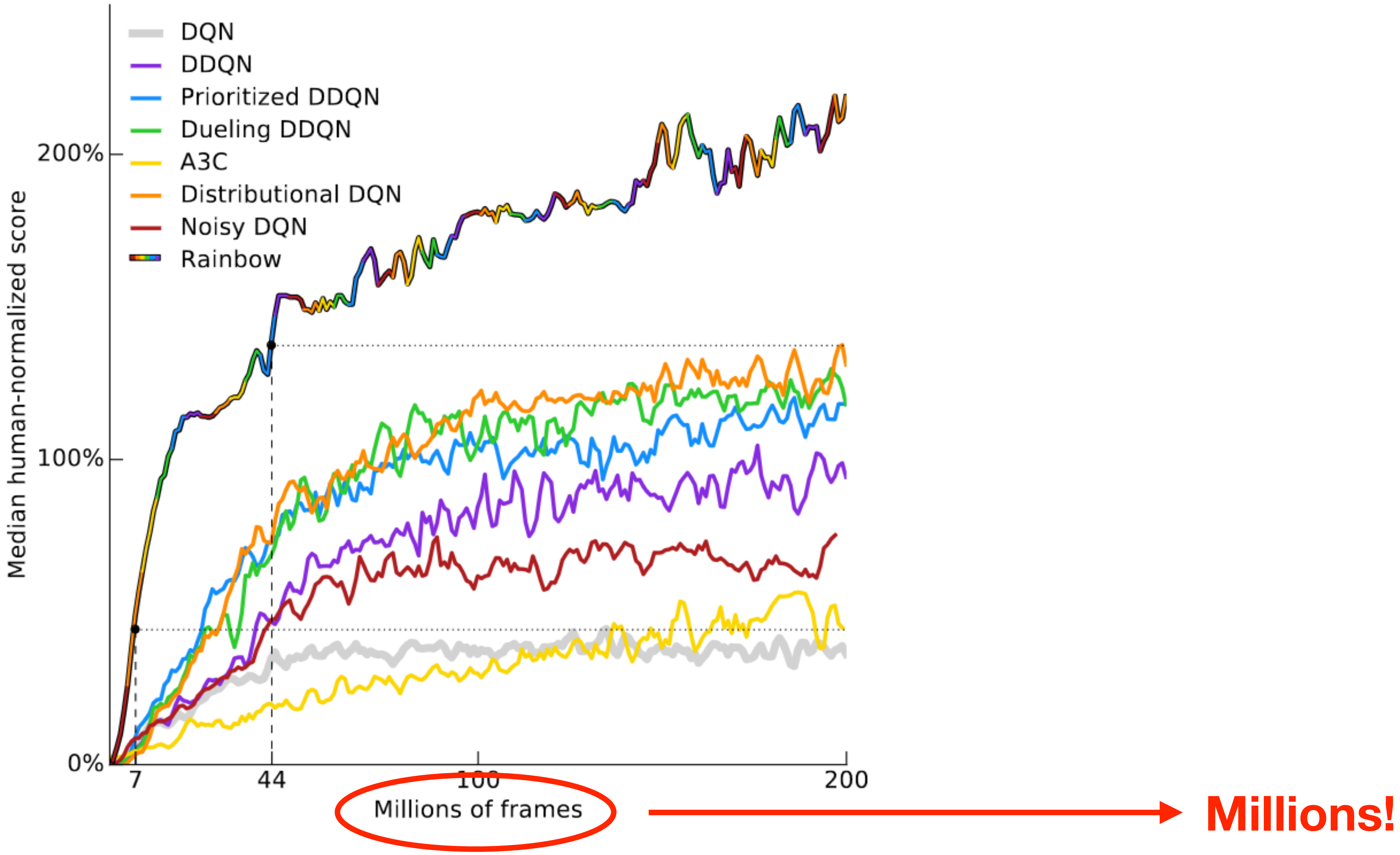
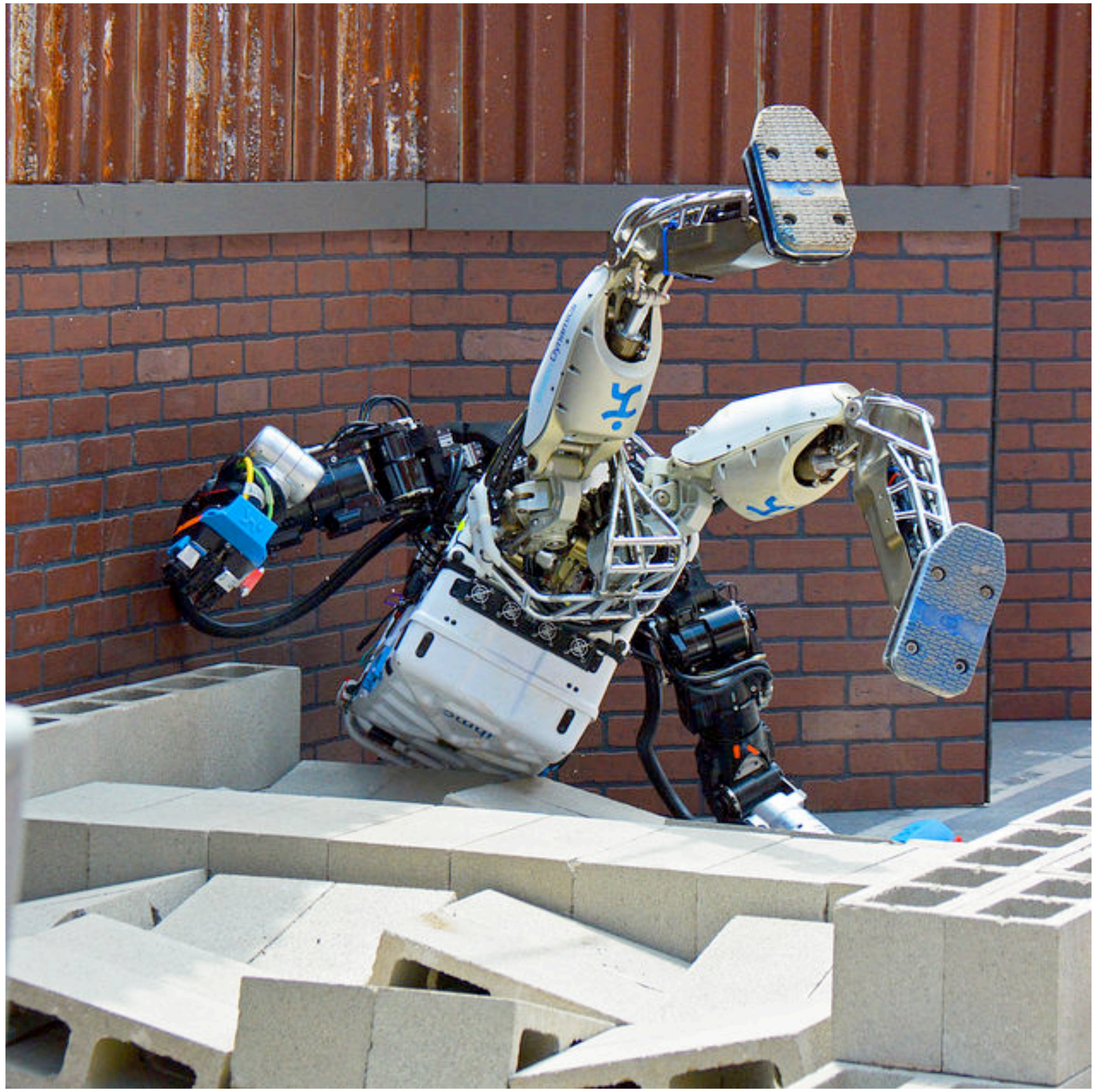


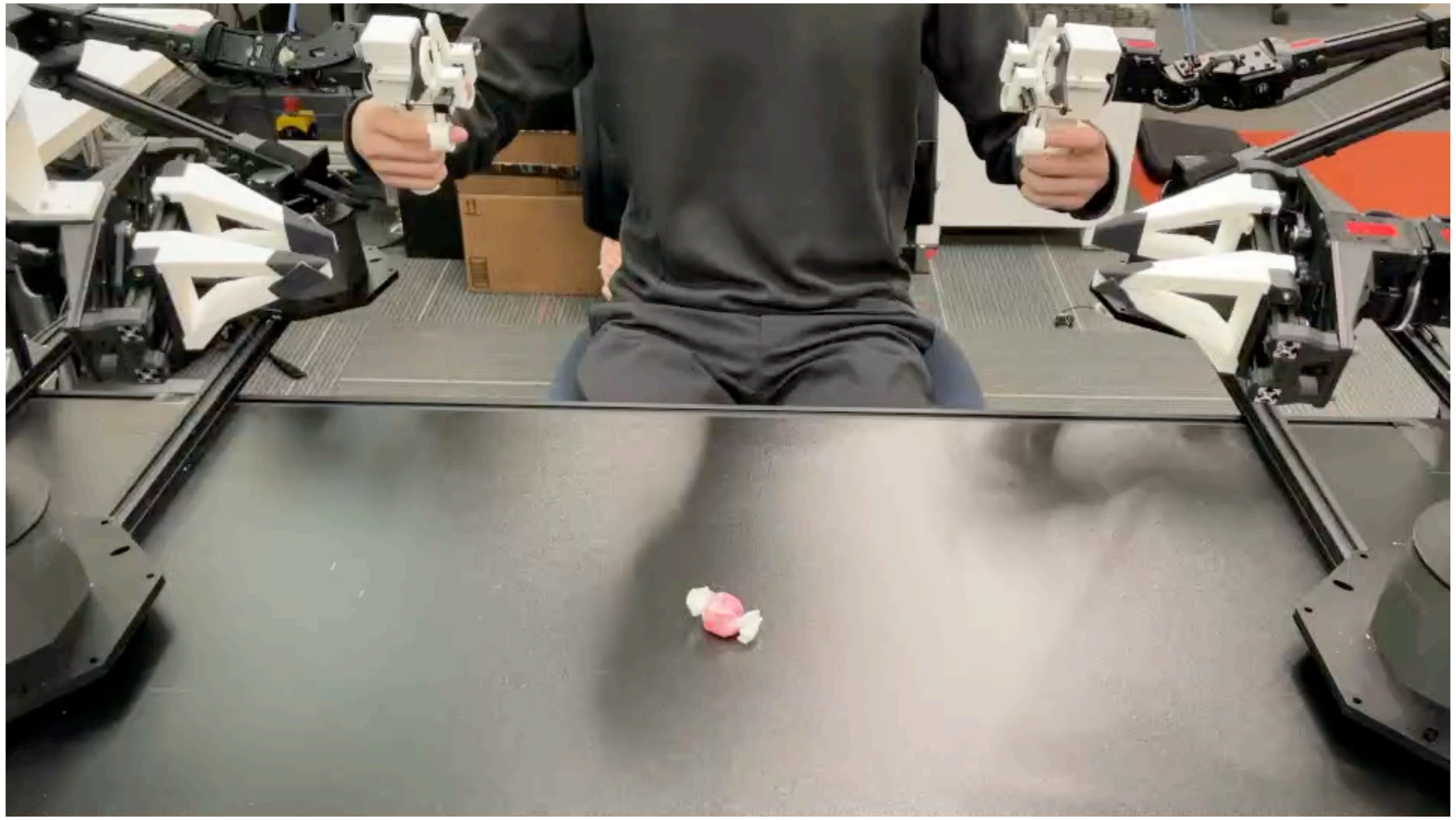
Figure 1: **Median human-normalized performance** across 57 Atari games. We compare our integrated agent (rainbow-

Hessel, Matteo, et al. "Rainbow: Combining improvements in deep reinforcement learning." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. No. 1. 2018.

Why learn from demonstrations?

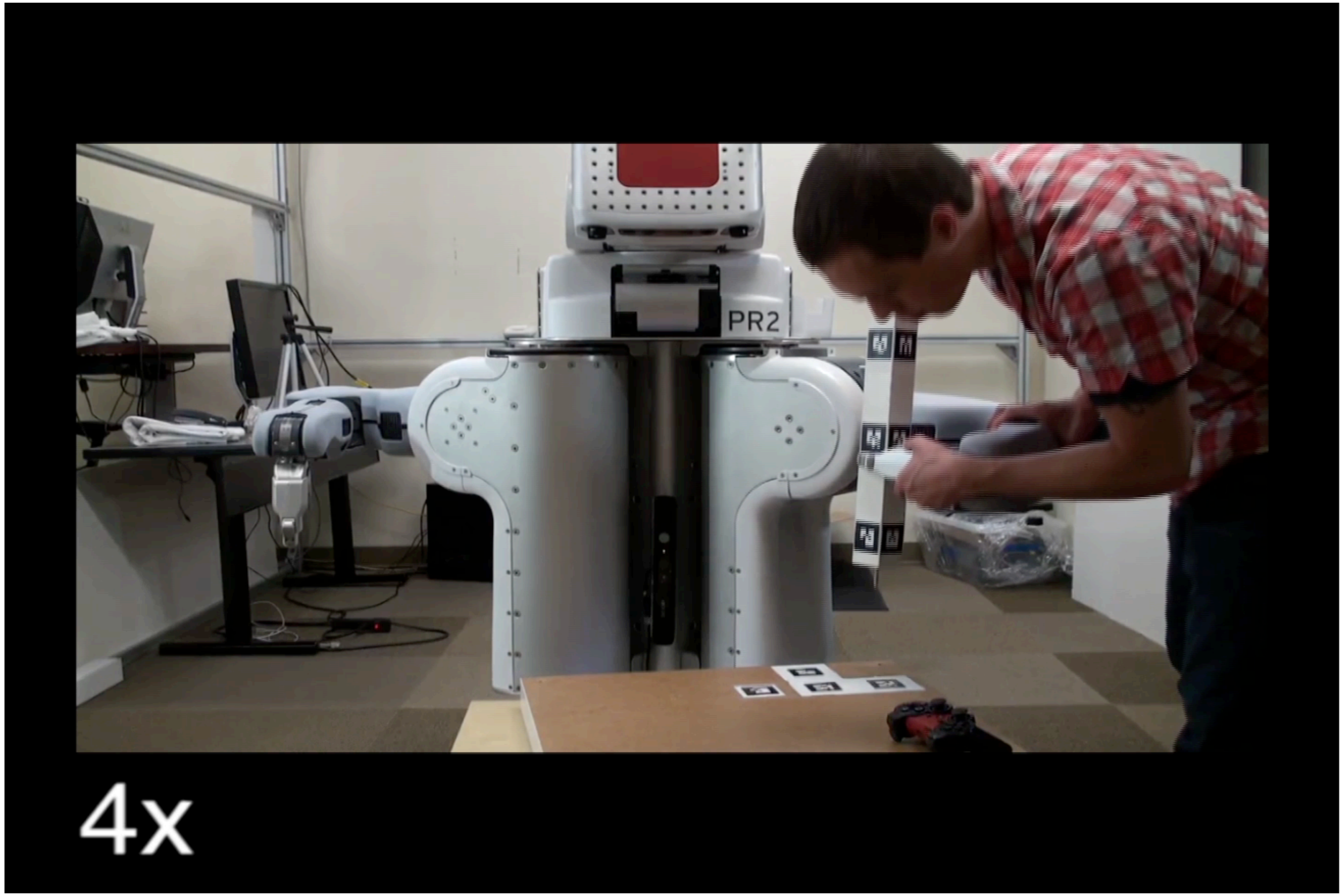


How to provide demonstrations?

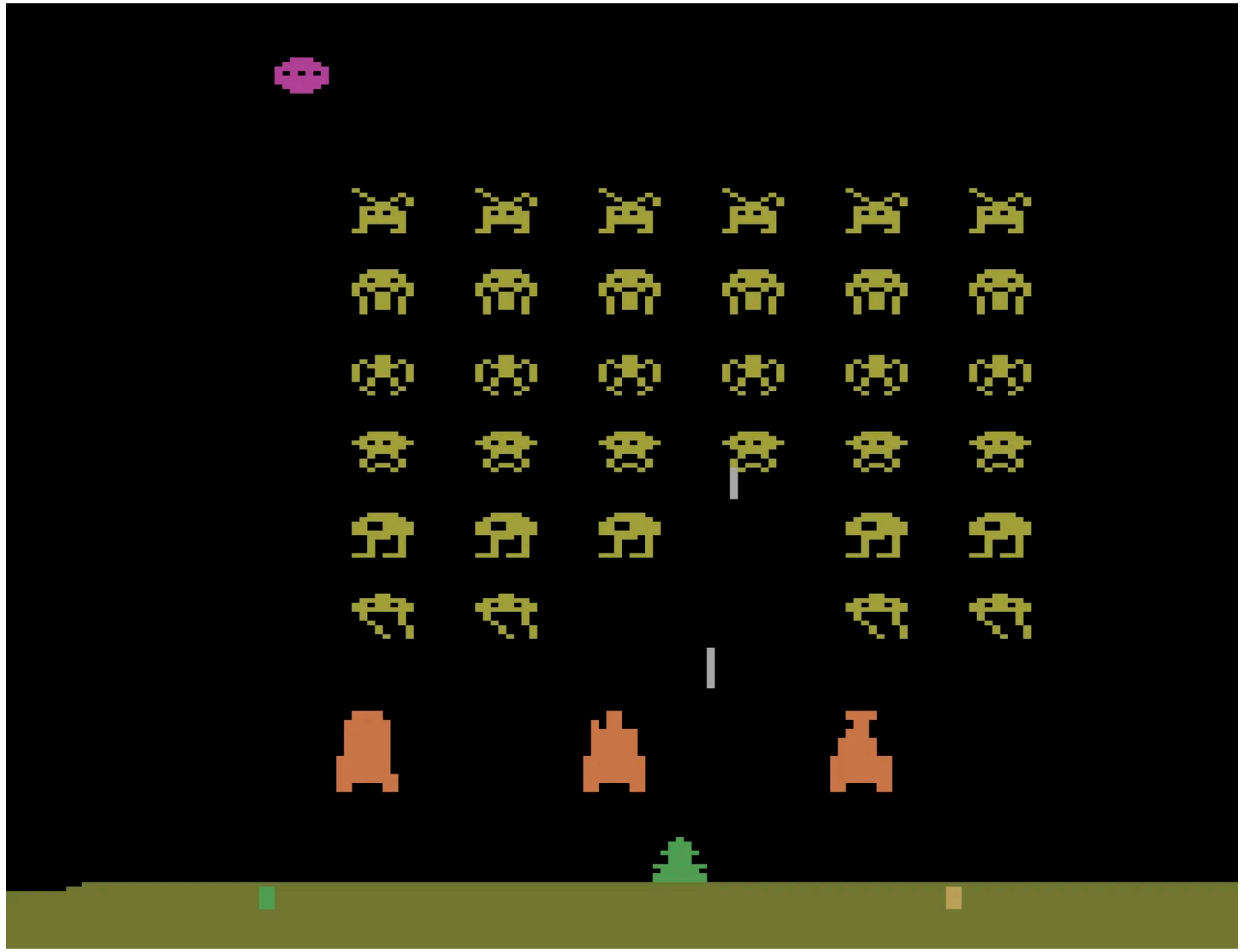


Fu, Zipeng, Tony Z. Zhao, and Chelsea Finn. "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation." *arXiv preprint arXiv:2401.02117* (2024).

How to provide demonstrations?



How to provide demonstrations?



How to provide demonstrations?

How to provide demonstrations?

M What is the best burger recipe?

There are many different burger recipes out there, and the "best" one will depend on personal preference. Some people prefer their burgers to be simple and classic, while others like to experiment with different ingredients and flavors.

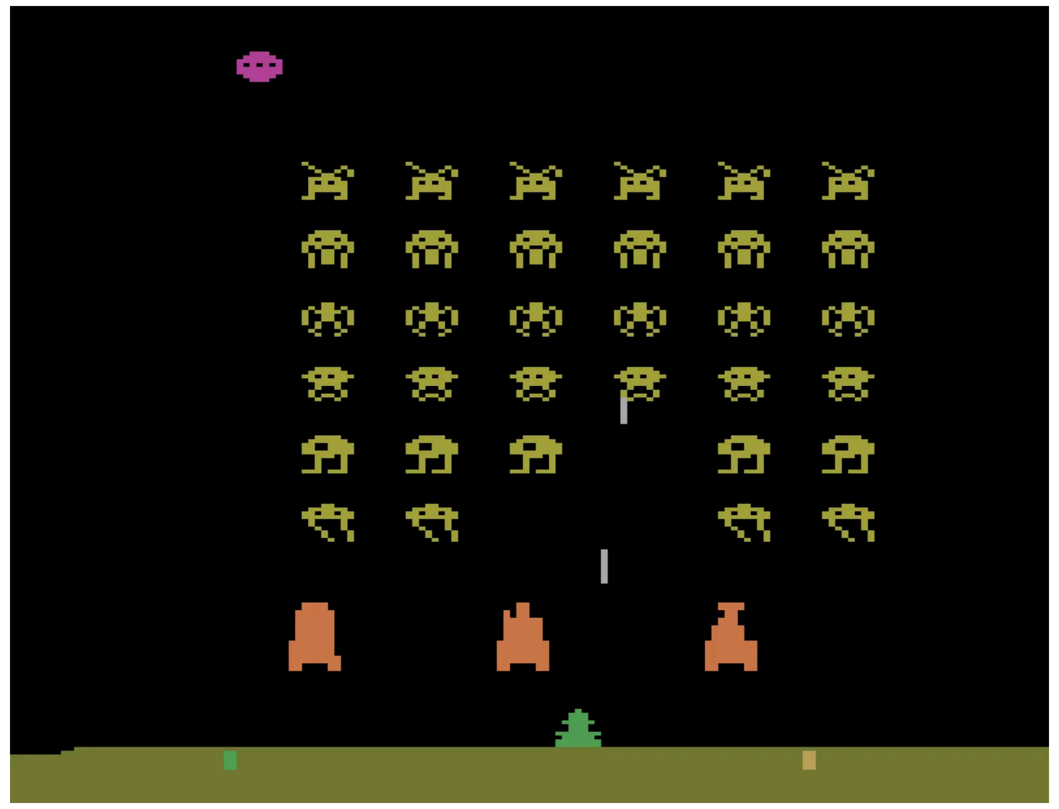
Here is a basic burger recipe that can serve as a starting point:

Ingredients:

- 1 pound ground beef
- 4 hamburger buns
- Lettuce, tomato, onion, and other toppings of your choice
- Salt and pepper

Instructions:

Behavioral cloning



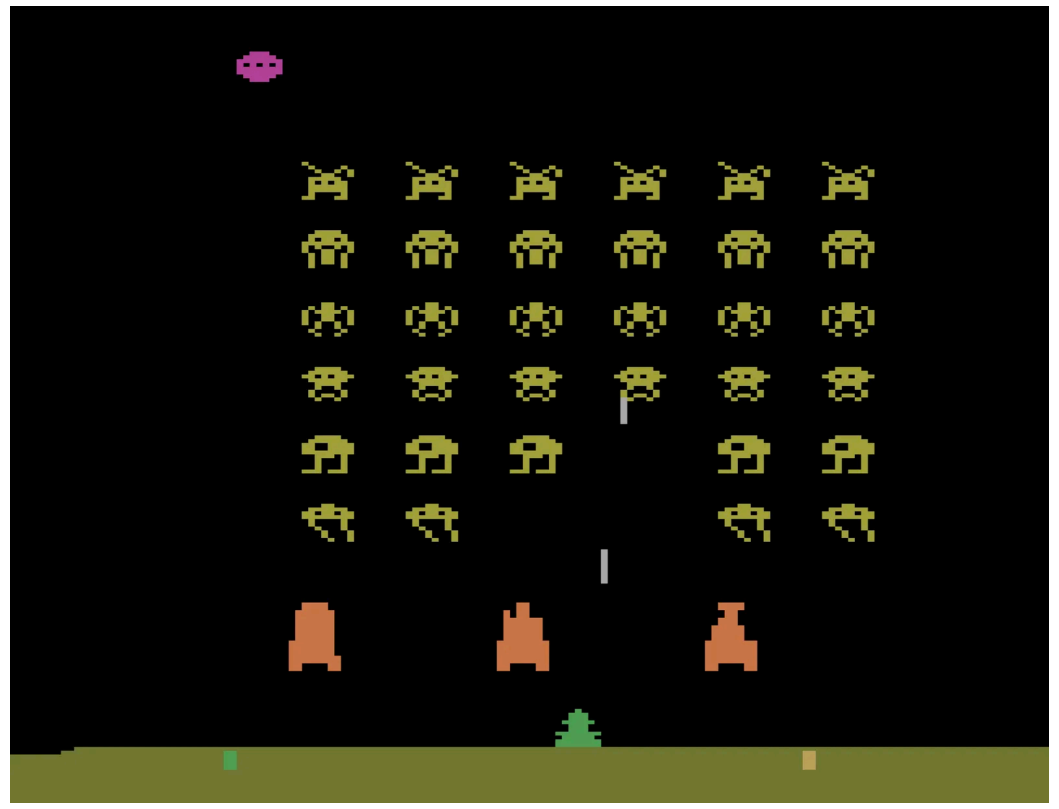
$$D = \{s_t, a_t, s_{t+1}\}_N$$

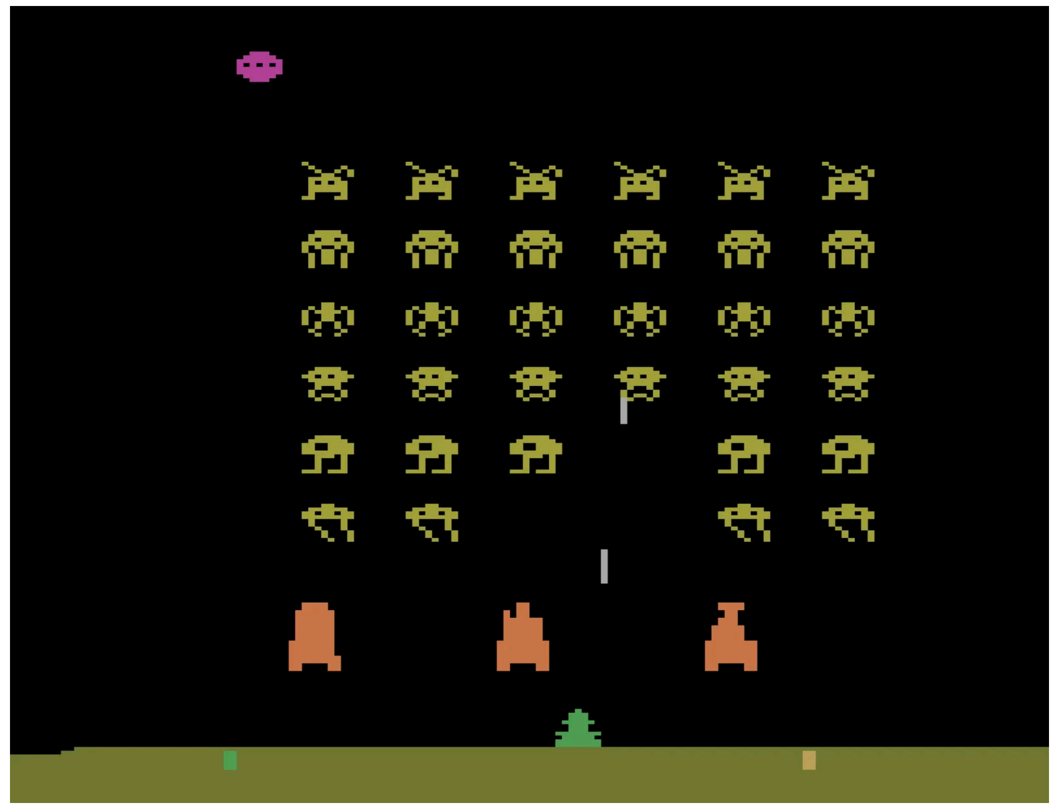
Learn: $\pi : S \to A$

or more generally: $\pi(s, a) := p(a|s)$

Straightforward supervised learning problem

Behavioral cloning from observation





$$D = \{s_t, \cancel{a_t}, s_{t+1}\}_N$$

Learn: $\pi : S \rightarrow A$

or more generally: $\pi(s, a) := p(a|s)$

How to infer action that caused transition from s_t to s_{t+1} ?

Inverse dynamics

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

Inverse dynamics: $p(a_t|s_t, s_{t+1})$

Learn inverse dynamics from offline data:

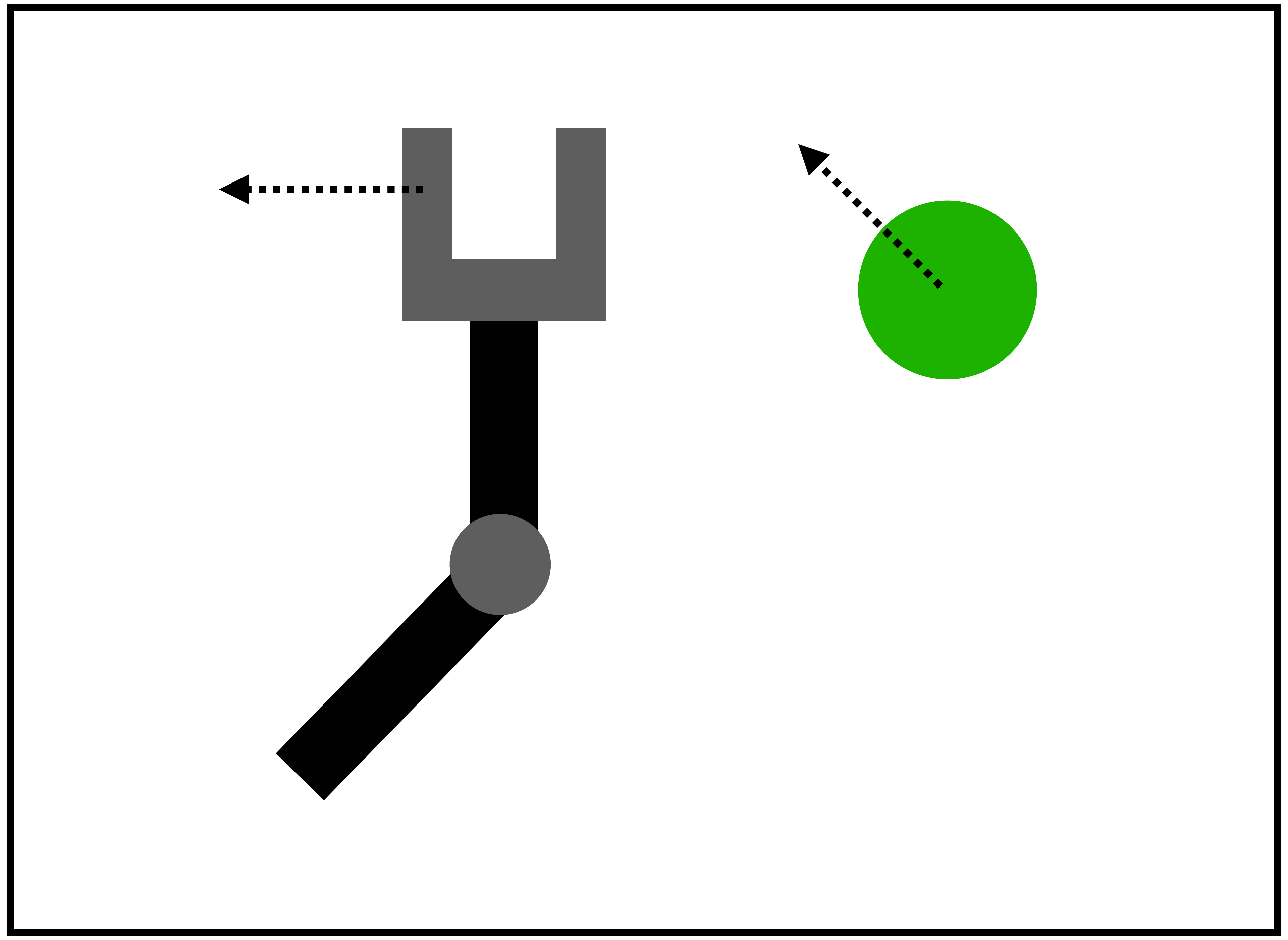
$$\theta^* = \arg\max_\theta \Pi_{i=0}^{|\mathcal{I}^{pre}|} p_\theta(a_i \,|s_i^a, s_{i+1}^a)$$

…and guess the missing demonstration actions: $D = \{s_t, \cancel{a_t}, s_{t+1}\}_N$

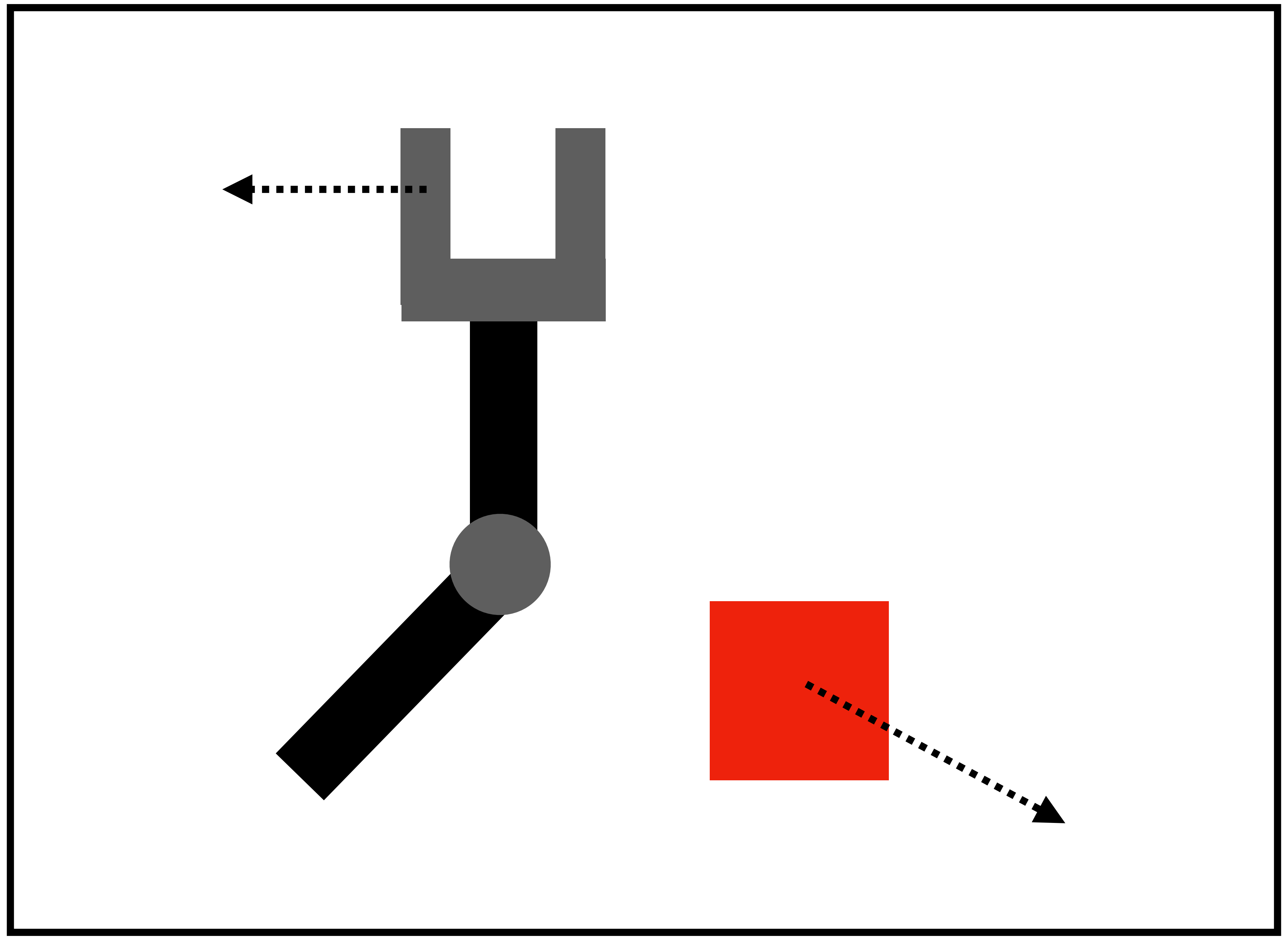
Now we're back to standard BC problem!

Agent-specific vs. task-specific state

$$S = S^a \times S^t$$

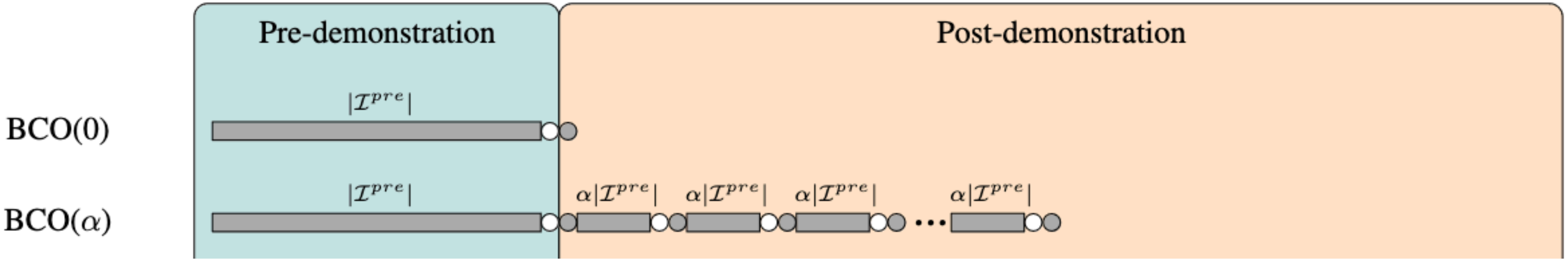


Inverse dynamics learning task



BCO task

BCO(alpha)



Pre-demonstration
Post-demonstration
BCO(0)
BCO(α)
$|\mathcal{I}^{pre}|$
$|\mathcal{I}^{pre}|$
$\alpha|\mathcal{I}^{pre}|$
$\alpha|\mathcal{I}^{pre}|$
$\alpha|\mathcal{I}^{pre}|$
$\alpha|\mathcal{I}^{pre}|$

Baselines

- GAIL and FEM
  - We’ll look at methods like these later in the course
  - At a high-level, they aim to match state-action occupancies / feature expectations
  - To do so, need to have post-demonstration data to learn policies that match the state/features well

TMLR reviewer

- Claims:
    - An inverse model can be learned from pre-demonstration data
    - A *task-agnostic* inverse dynamics model can be learned
    - BCO can accurately imitate with observation-only data
    - BCO allows for imitation without post-demonstration interaction
    - However, post-demonstration data helps, if you’re willing to collect it
    - BCO is better with less data than competing approaches

TMLR reviewer

- Claims supported?
  - An inverse model can be learned from pre-demonstration data
    - No direct experiments testing accuracy of inverse dynamics model, only done in context of how it effects downstream policy learning
    - No experiments examining effects of different amounts of pre-demonstration data
    - If BCO works well overall, then the inverse model must be decent
  - A *task-agnostic* inverse dynamics model can be learned
    - Reacher domain has partitioned agent/task state space
    - Only assess accuracy of overall algorithm, not inverse model
    - Partitioned by hand, and they don't show consequences of not partitioning

TMLR reviewer

- Claims supported?
  - BCO can accurately imitate with observation-only data / without post-demonstration interaction / with less data than competing approaches
    - Performance much better than random, often close to expert, and often close to action-aware BC.
    - Competitive with baselines that have access to actions (kind of like having infinite pre-demonstration data)
    - Performance steadily improves with additional pre-demonstration data
    - 20 trials + small standard error bars provides confidence of correctness of results
    - They show that other methods (e.g. GAIL) need many post-demonstration interactions to do as well as BCO(0). But these are very simple domains, and GAIL has better guarantees than BCO outside of the support of the demonstrations.
  - Post-demonstration data helps, if you're willing to collect it
    - Performance increases as alpha increases and approaches action-aware BC

TMLR reviewer

- Questions:
  - All experiments were with synthetic demonstrations from TRPO. How close to optimal were they? Does BCO's performance gracefully degrade with noisy demonstrations, e.g. from humans?
  - Can agent/task state space partitioning be learned? How much does performance suffer if you don't partition?
  - Domains were quite simple, but the motivation in the introduction was learning from Youtube videos. What would it take to scale? Can partitioning be learned (implicitly, perhaps)?

- Reproducibility:
  - Details provided of each domain, neural network architectures, number of interactions, etc.
  - While architecture is specified for dynamics model, they don't say what it is for policy
  - Not clear if/how hyperparameters for GAIL and FEM were tuned

Archaeologist

BCO cites:

Niekum, S., Osentoski, S., Konidaris, G., Chitta, S., Marthi, B., & Barto, A. G. (2015). Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, *34*(2), 131-157.

- Also imitates from observations
- Doesn't need actions due to robot arm controller with known functional form and simplistic generalization based only on changing start/goal locations
- Automatically segments and reuses sub-skills for complex, multi-step tasks

BCO cited by:

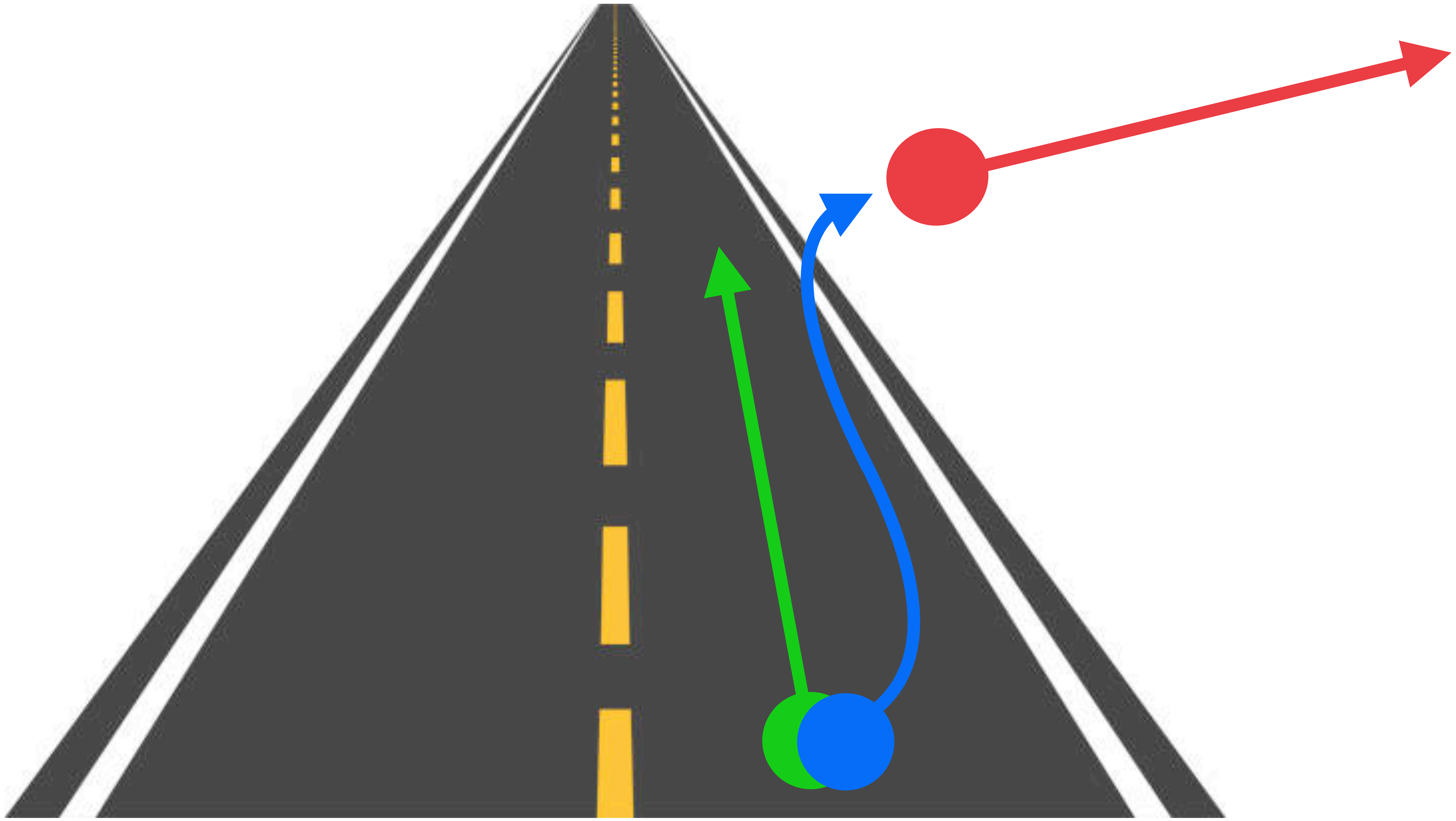
Torabi, F., Warnell, G., & Stone, P. (2018). Generative adversarial imitation from observation. *In ICML Workshop on Imitation, Intent, and Interaction*. *arXiv preprint arXiv:1709.04905, 2019*

- Same authors!
- Aims to address compounding error that BCO can experience due to being purely supervised
- Essentially GAIL, but from observation-only data
- Performs state occupancy matching instead of state-action occupancy matching
- Has above advantages compared to BCO, but also requires post-demonstration data

Academic researcher

- Study approaches and effects of automatic agent/task state partitioning:
  - In a simple domain, study performance degradation as task-specific variables are leaked into agent's state space for inverse model.
  - Study multi-task pre-demonstration setting. As number of tasks are increased, how does (1) inverse model performance change and (2) BCO performance change?
  - Does the inverse model overfit to training tasks or generalize well to new tasks? How does regularization effect this?
  - How does domain complexity influence the above? E.g. real-world robotic manipulation from video vs. reacher domain?

Downsides of behavioral cloning



Quadratic regret

$$\hat{\pi}_{sup} = \arg\min_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi^*}} [\ell(s, \pi)] \quad (2)$$

Assuming $\ell(s, \pi)$ is the 0-1 loss (or upper bound on the 0-1 loss) implies the following performance guarantee with respect to any task cost function C bounded in $[0, 1]$:

Theorem 2.1. (*Ross and Bagnell, 2010*) *Let* $\mathbb{E}_{s \sim d_{\pi^*}} [\ell(s, \pi)] = \epsilon$, *then* $J(\pi) \leq J(\pi^*) + T^2 \epsilon$.

Compare to typical supervised learning loss that grows as: $O(\epsilon T)$

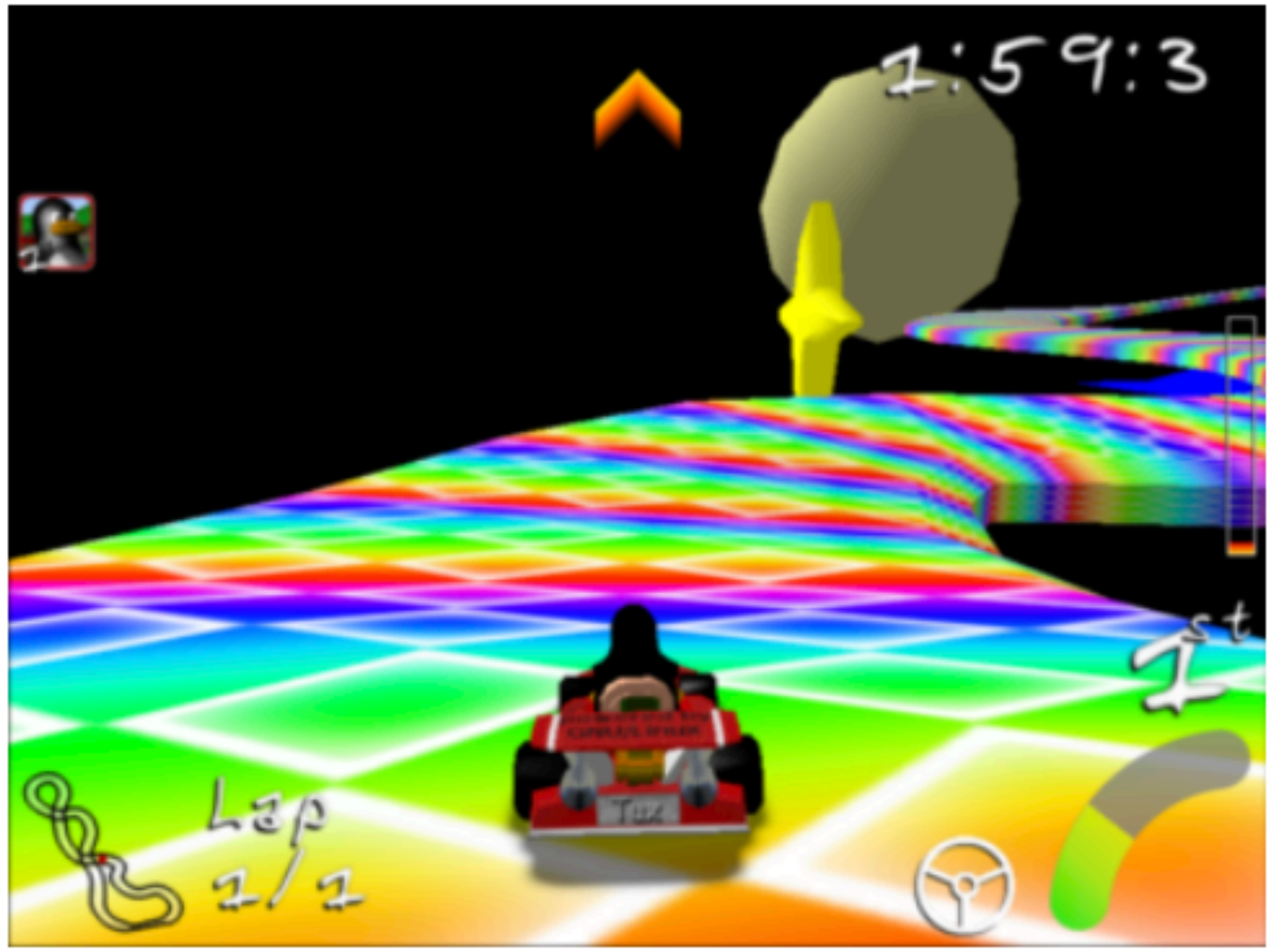
DAgger

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

Algorithm 3.1: DAGGER Algorithm.

Key idea: keep collecting demonstration data that is on-distribution for current policy, and reduce dependence on expert over time

Awkward!



Difficult to give good demonstrations when you only have control beta-percent of the time?

DAgger

Theorem 2.2. *Let π be such that $\mathbb{E}_{s\sim d_\pi}[\ell(s,\pi)] = \epsilon$, and $Q^{\pi^*}_{T-t+1}(s,a) - Q^{\pi^*}_{T-t+1}(s,\pi^*) \leq u$ for all action a, $t \in \{1,2,\dots,T\}$, $d^t_\pi(s) > 0$, then $J(\pi) \leq J(\pi^*) + uT\epsilon$.*

Proof. We here follow a similar proof to Ross and Bagnell (2010). Given our policy π, consider the policy $\pi_{1:t}$, which executes π in the first t-steps and then execute the expert π^*. Then

$$\begin{aligned}
&J(\pi) \\
&= J(\pi^*) + \textstyle\sum_{t=0}^{T-1}[J(\pi_{1:T-t}) - J(\pi_{1:T-t-1})] \\
&= J(\pi^*) + \textstyle\sum_{t=1}^{T}\mathbb{E}_{s\sim d^t_\pi}[Q^{\pi^*}_{T-t+1}(s,\pi) - Q^{\pi^*}_{T-t+1}(s,\pi^*)] \\
&\leq J(\pi^*) + u\textstyle\sum_{t=1}^{T}\mathbb{E}_{s\sim d^t_\pi}[\ell(s,\pi)] \\
&= J(\pi^*) + uT\epsilon
\end{aligned}$$

The inequality follows from the fact that $\ell(s,\pi)$ upper bounds the 0-1 loss, and hence the probability π and π^* pick different actions in s; when they pick different actions, the increase in cost-to-go $\leq u$. $\square$

Theorem 3.1. *For* DAGGER, *if N is $\tilde{O}(T)$ there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ s.t. $\mathbb{E}_{s\sim d_{\hat{\pi}}}[\ell(s,\hat{\pi})] \leq \epsilon_N + O(1/T)$*

In particular, this holds for the policy $\hat{\pi} = \arg\min_{\pi\in\hat{\pi}_{1:N}} \mathbb{E}_{s\sim d_\pi}[\ell(s,\pi)]$. [3] If the task cost function C corresponds to (or is upper bounded by) the surrogate loss ℓ then this bound tells us directly that $J(\hat{\pi}) \leq T\epsilon_N + O(1)$. For arbitrary task cost function C, then if ℓ is an upper bound on the 0-1 loss with respect to π^*, combining this result with Theorem 2.2 yields that:

Theorem 3.2. *For* DAGGER, *if N is $\tilde{O}(uT)$ there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ s.t. $J(\hat{\pi}) \leq J(\pi^*) + uT\epsilon_N + O(1)$.*