Baseline Controllers

Spring 2020
Lecture 5

Zackory Erickson

Install PyTorch + RL Library

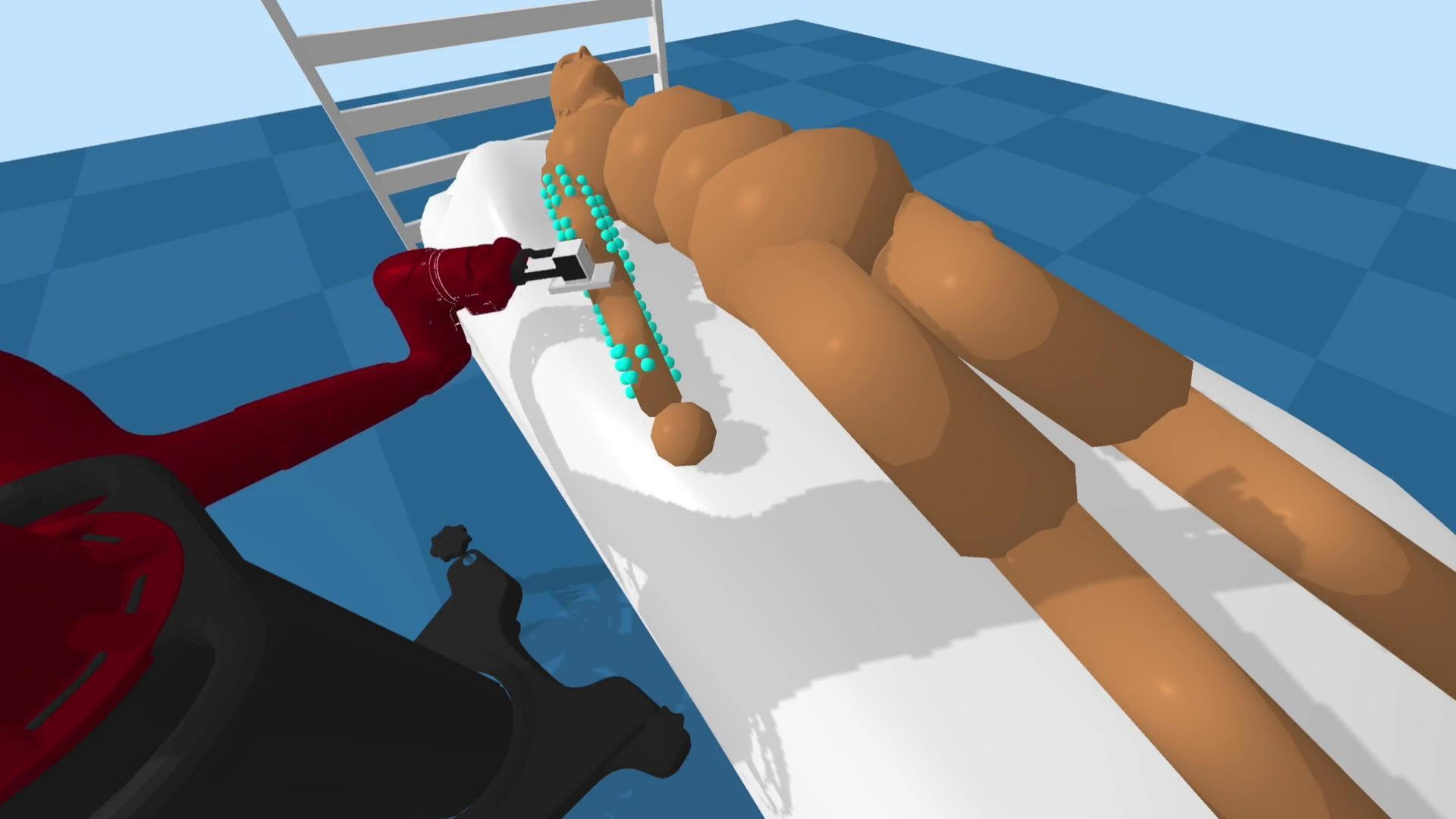
https://github.com/Healthcare-Robotics/assistive-gym/wiki/4.-Running-Pretrained-Policies

Midterm Project Discussion

1. What robots / tasks do you find interesting?
2. Teaming

What are you trying to beat?

The baseline controllers

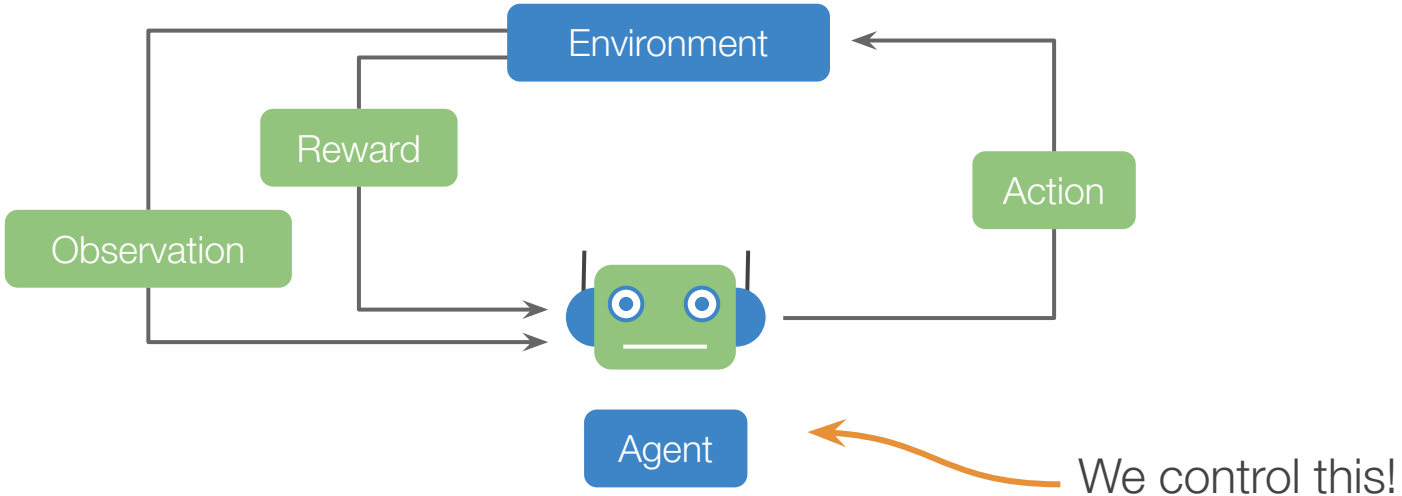


Let’s Run Pretrained Policies

https://github.com/Healthcare-Robotics/assistive-gym/wiki

1. Install the Pytorch RL library and download pretrained policies
2. Static Human: `python3 -m ppo.enjoy --env-name "FeedingPR2-v0"`

Reinforcement Learning



Deep Reinforcement Learning

Reinforcement Learning (Q-Learning)

$$Q : S \times A \rightarrow \mathbb{R}$$

State Action Expected reward

$$Q(s_t, a_t)$$

Common ways to represent Q: Lookup Table or Neural Network

Expected Reward (Discounted)

Expected future reward

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} \left(P\left(s,a,s'\right) \max_{a'} Q\left(s',a'\right) \right)$$

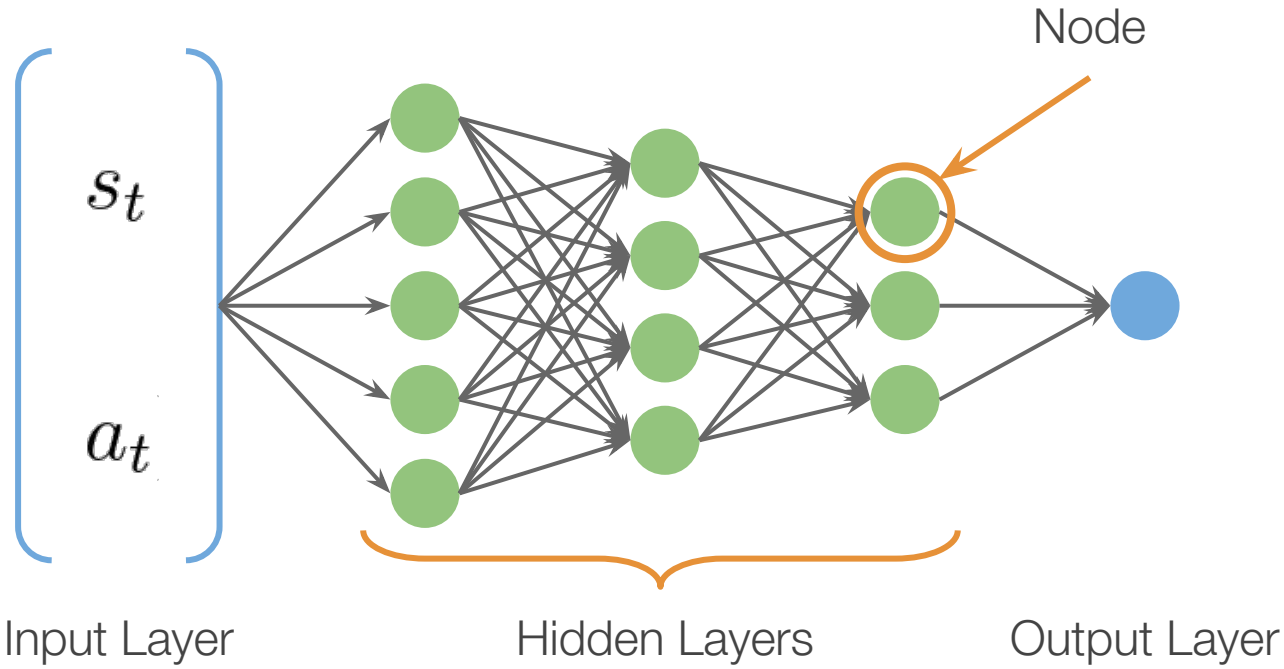
Discount factor, e.g. γ = 0.9

Reinforcement Learning

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

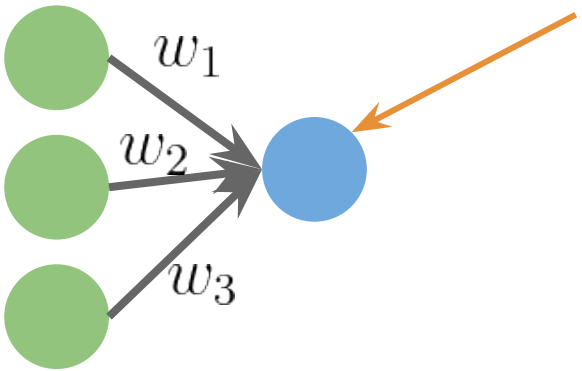
Neural Networks for RL

$$Q(s_t, a_t)$$

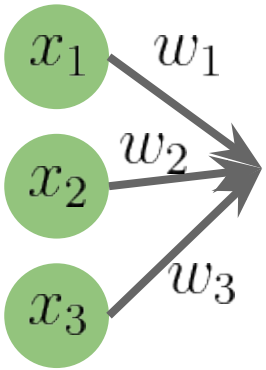


Neural Networks for RL

How do we compute this?



Neural Networks for RL

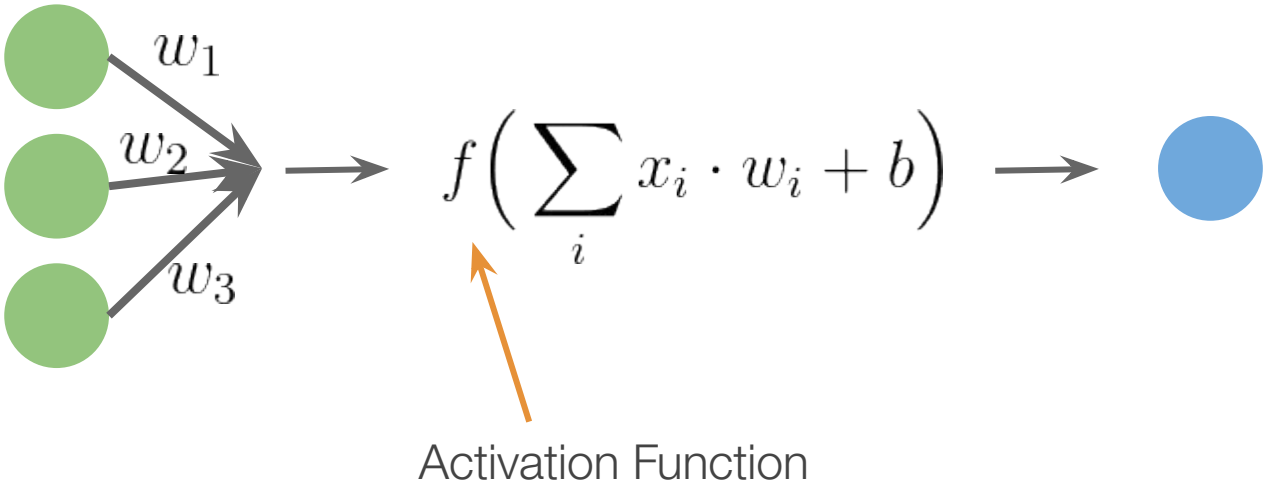


$$x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

$$\sum_i x_i \cdot w_i + b$$

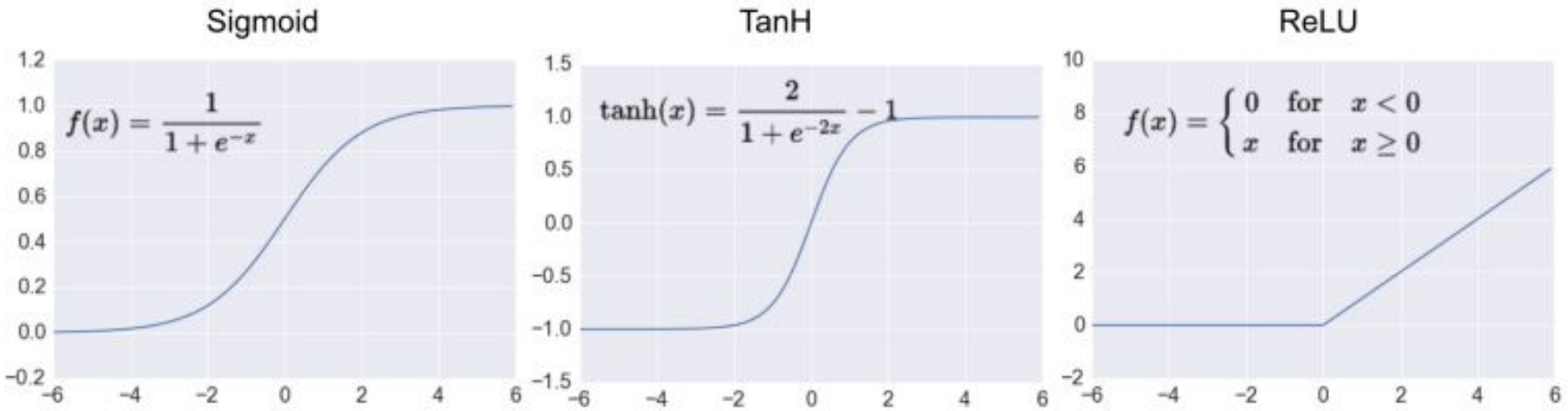
Bias unit

Neural Networks for RL

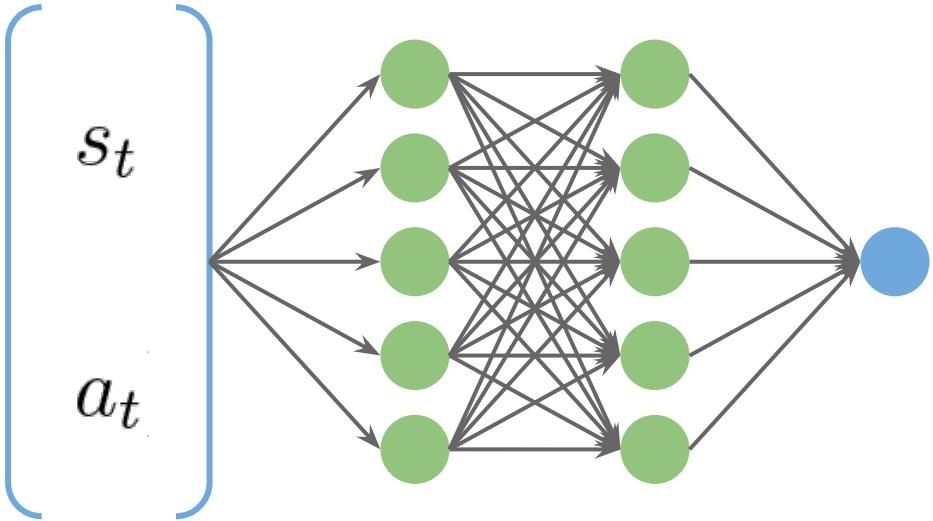


Nonlinear Activation Functions

$$f\left(\sum_i x_i \cdot w_i + b\right)$$



A Simple Neural Network for RL



$Q(s_t, a_t)$

$$\max_{a_t} Q(s_t, a_t)$$

But, this is challenging with continuous actions

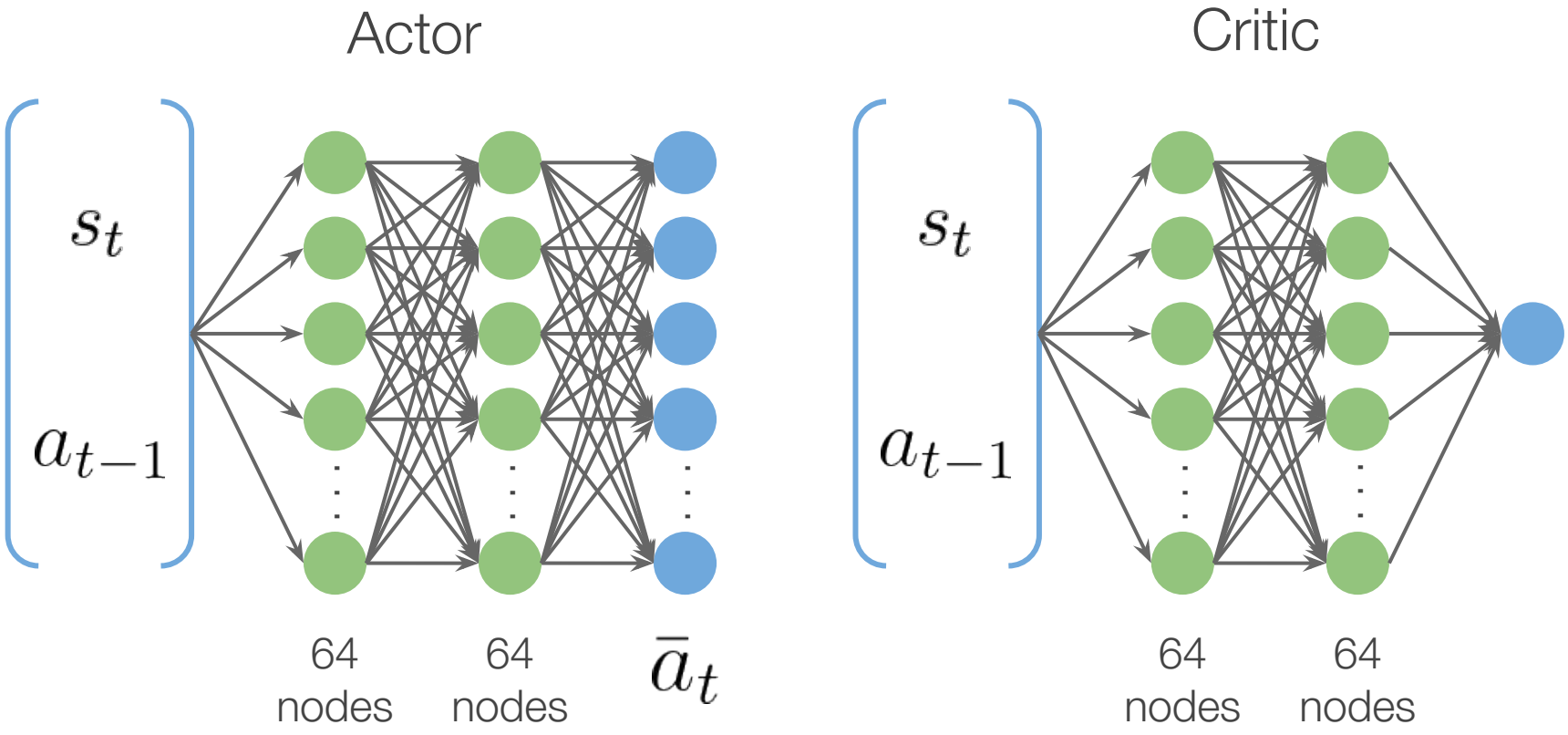
Proximal Policy Optimization (PPO)

An actor-critic architecture

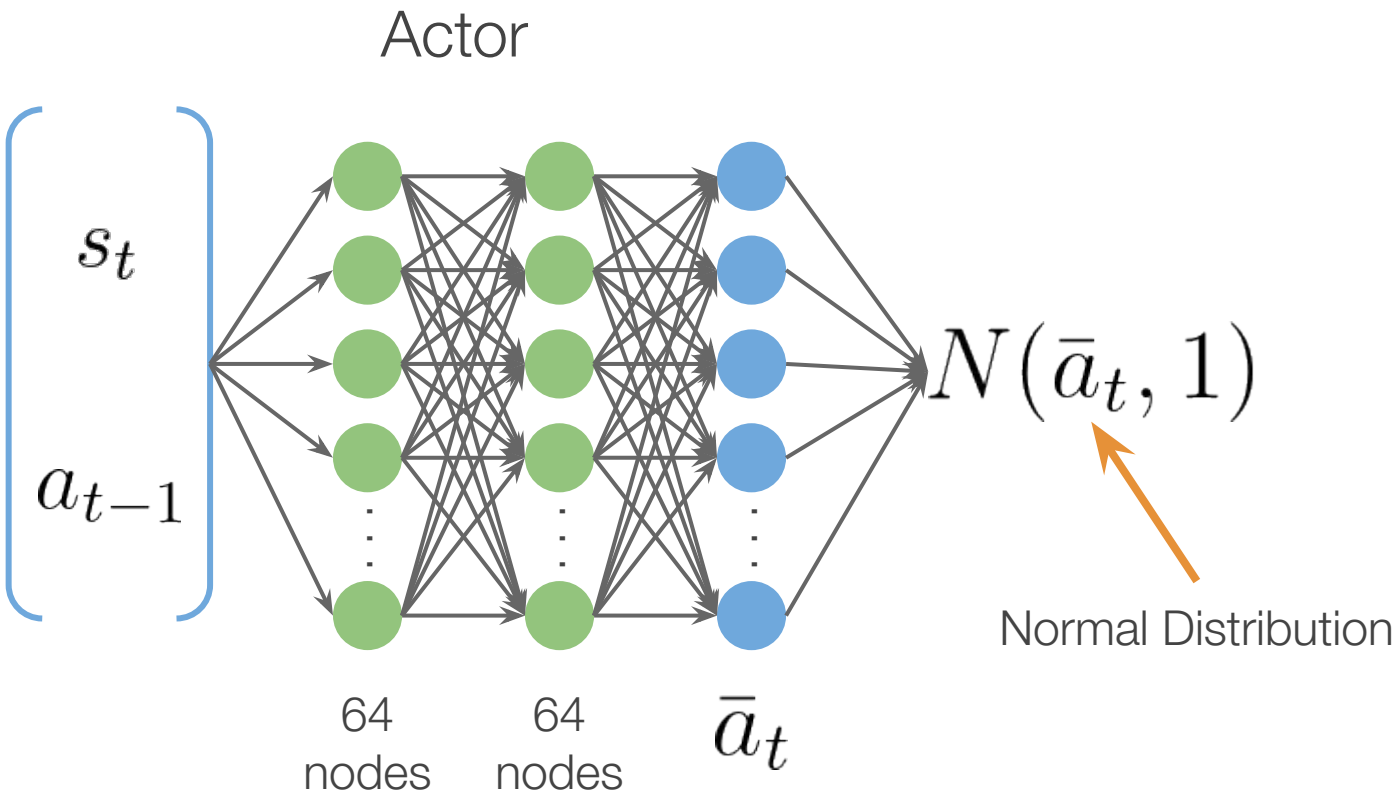
Actor: estimates an action given s_t, a_{t-1}

Critic: estimates the expected future reward given s_t, a_{t-1}

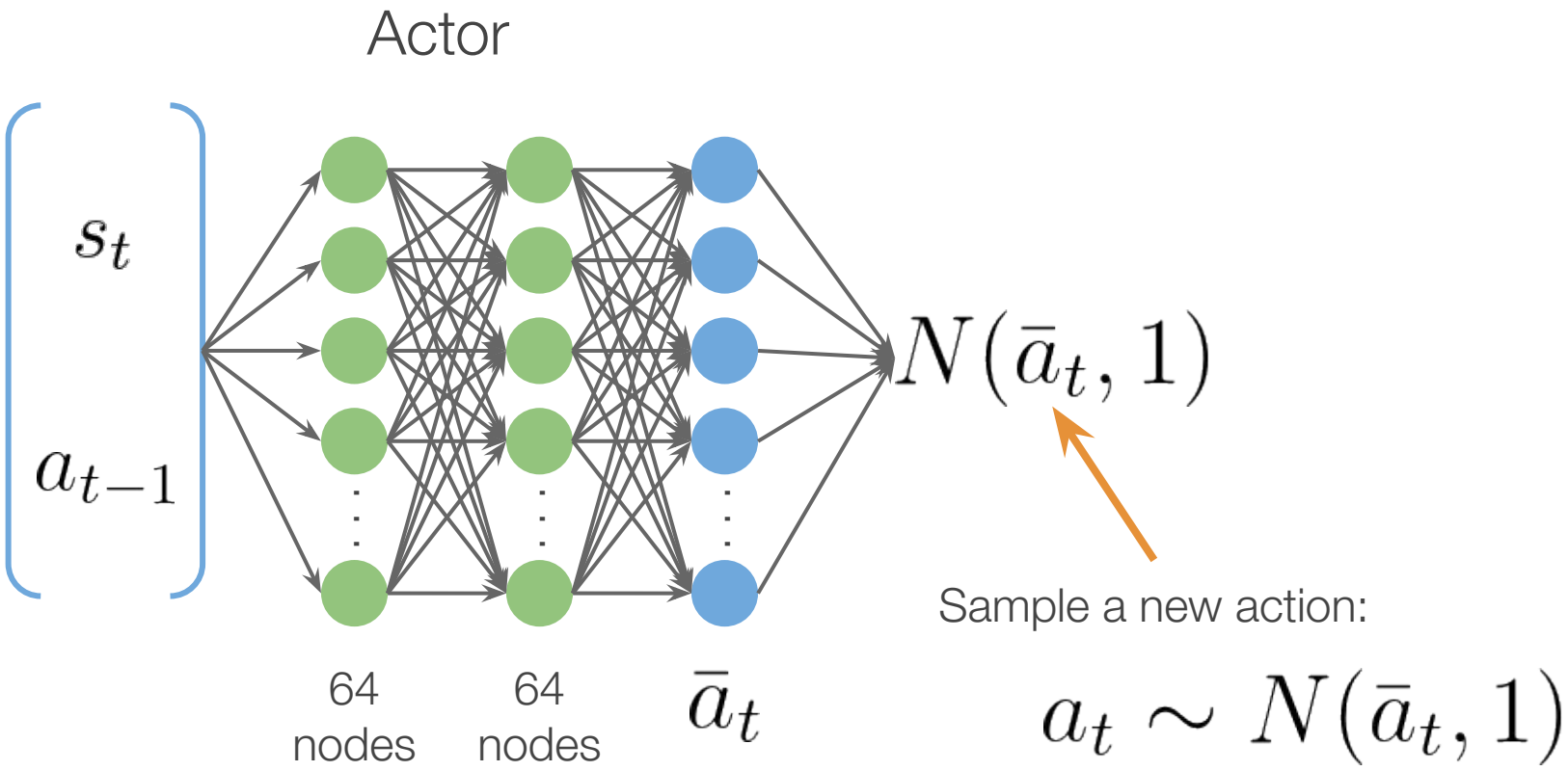
Proximal Policy Optimization (PPO)



Proximal Policy Optimization (PPO)



Proximal Policy Optimization (PPO)

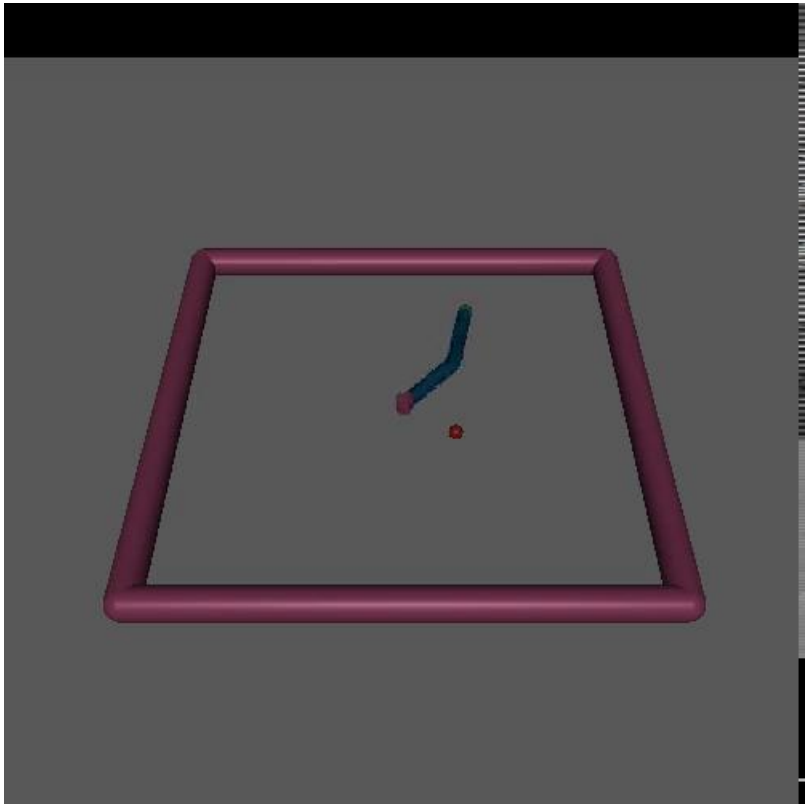


Where Does This Policy Originate From?

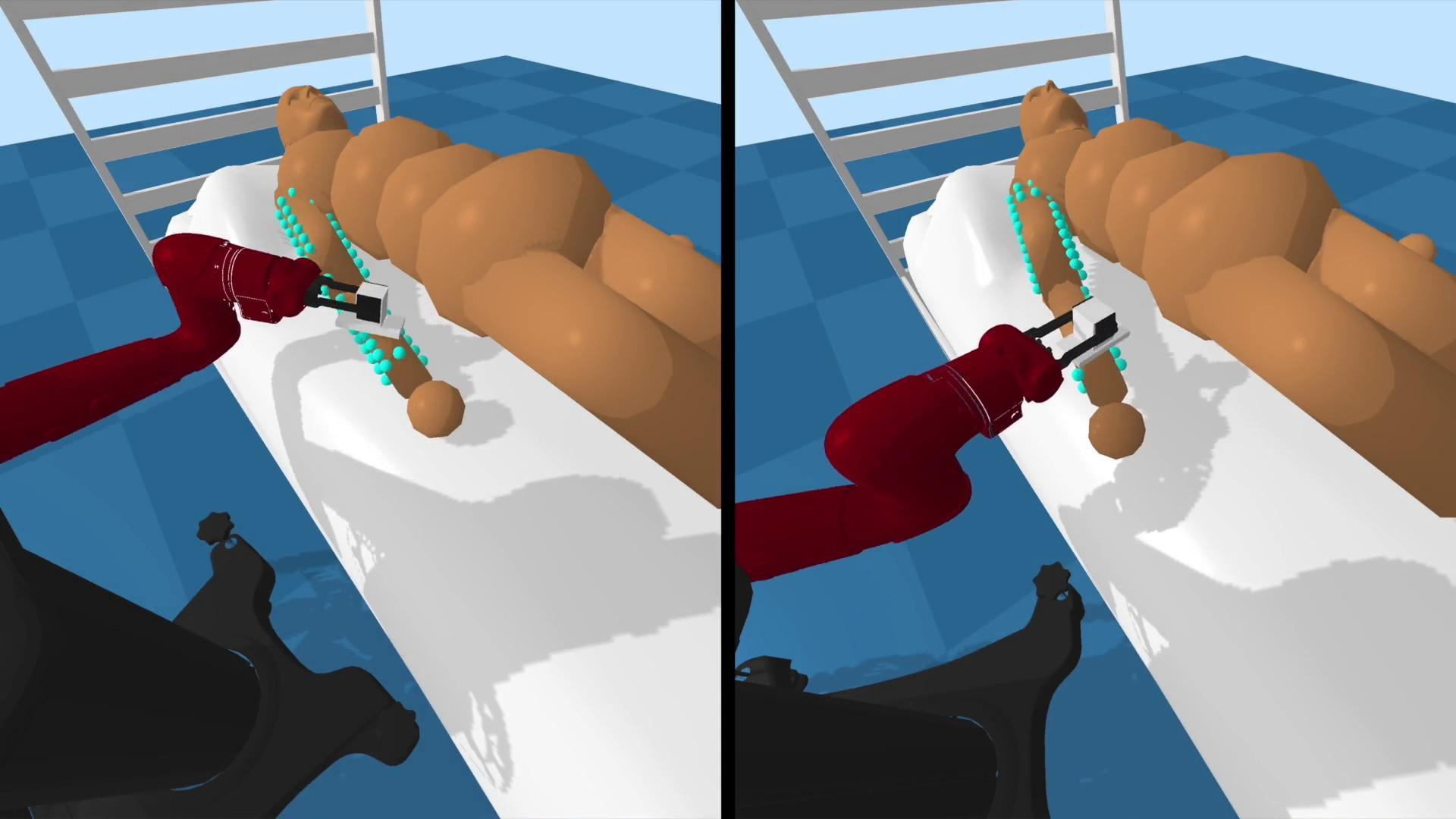
Originally designed for the “Reacher-v2” environment in OpenAI Gym

https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail

https://gym.openai.com/envs/Reacher-v2/



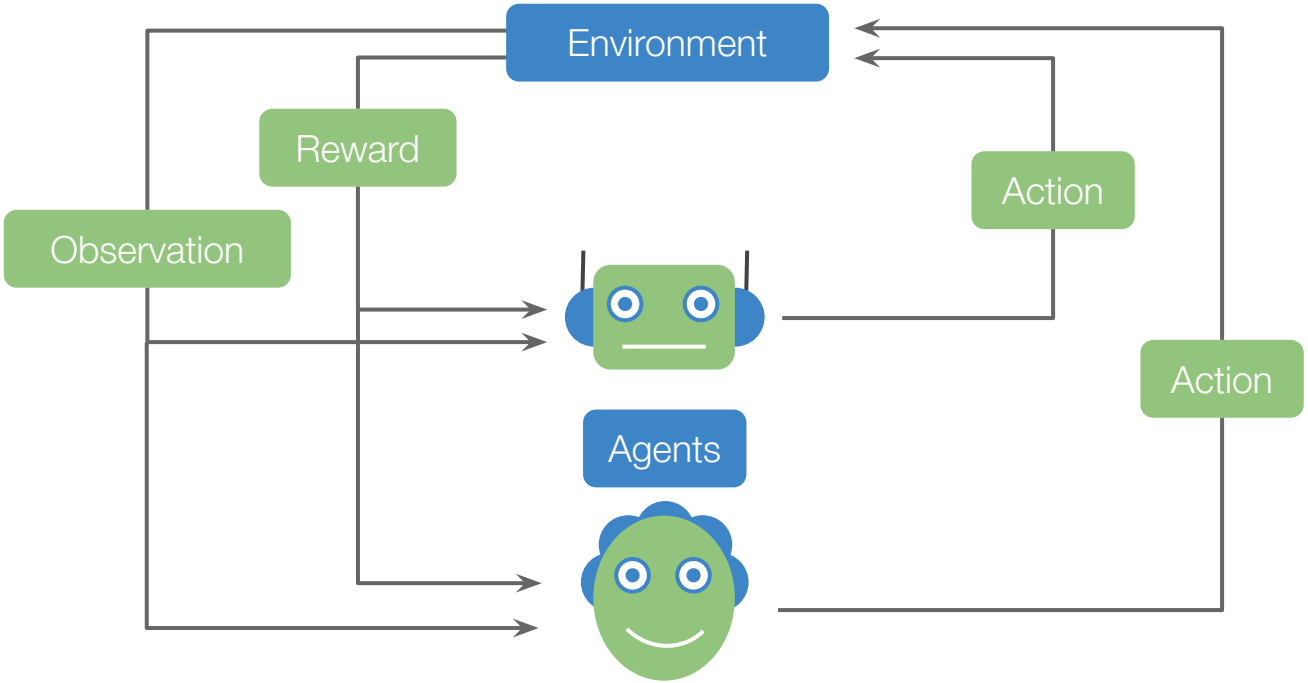
What about active human motion?



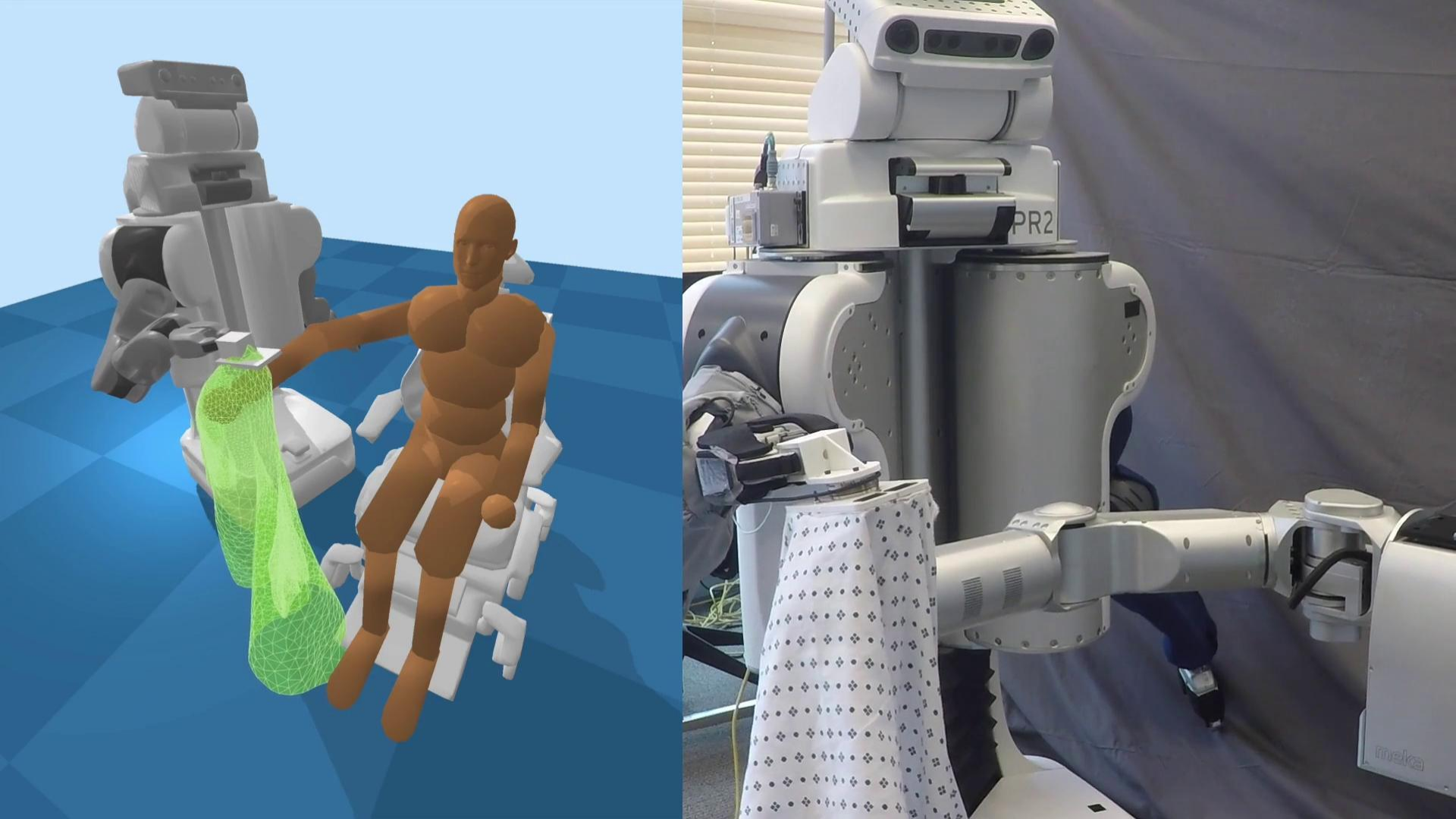
Let’s Run Pretrained Policies

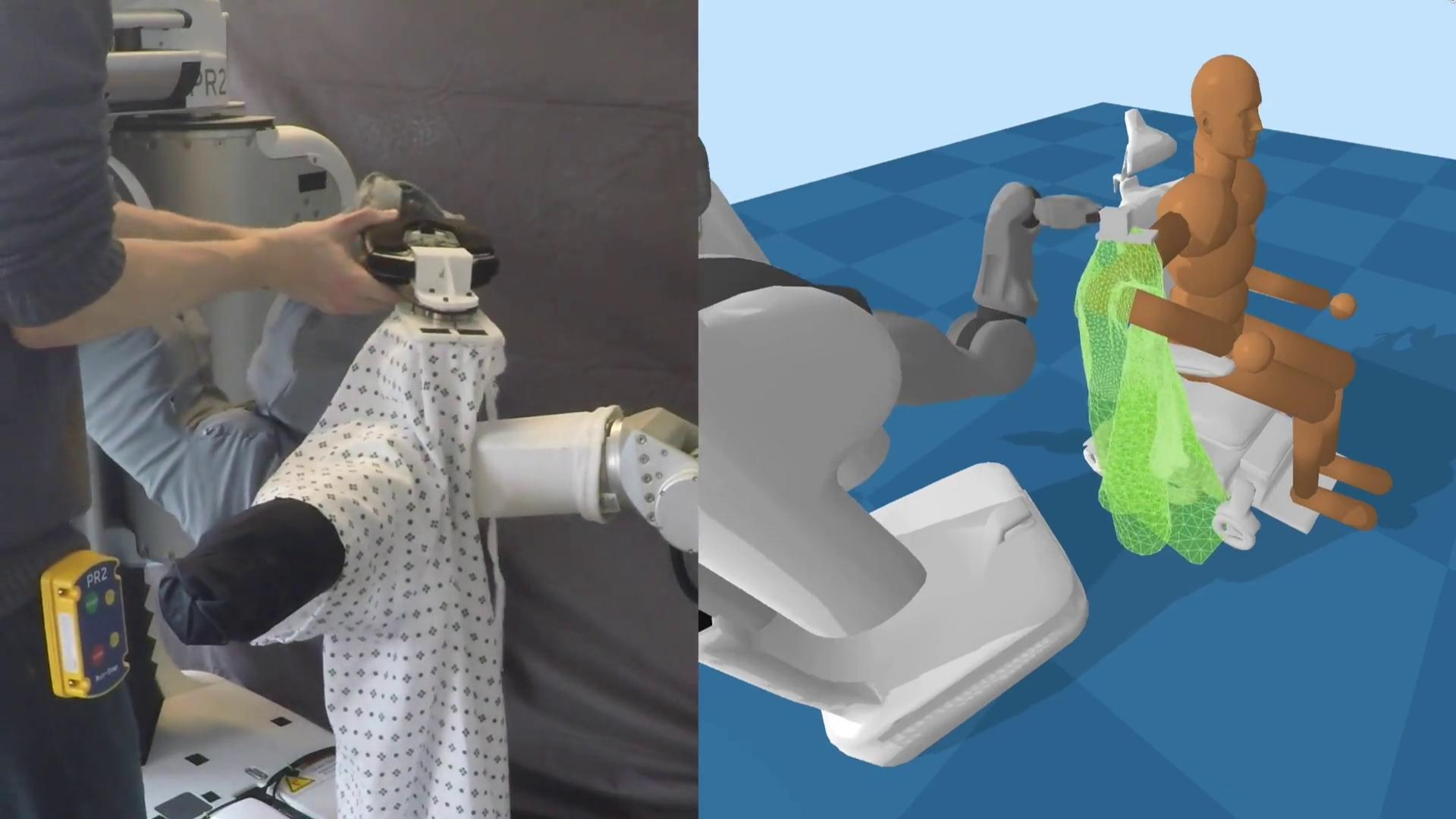
Active Human: `python3 -m ppo.enjoy_coop --env-name "DrinkingSawyerHuman-v0"`

Co-optimization



Can Assistive Gym help train real robots?


PR2


PR2
PR2

Let’s Train a New Control Policy

Training Command:

```
python3 -m ppo.train --env-name "ScratchItchJaco-v0" --num-env-steps 100000 --save-dir ./trained_models_new/
```

Now, evaluate the trained policy:

```
python3 -m ppo.enjoy --env-name "ScratchItchJaco-v0" --load-dir trained_models_new/ppo
```

We can use the `--load-policy` argument to continue training an existing policy.

Basic Code Structure

```
class NewTaskEnv(AssistiveEnv):
    def __init__(self, robot_type='pr2', human_control=False):
        ...

    def step(self, action):
        ...

    def _get_obs(self, forces, forces_human):
        ...

    def reset(self):
        ...
```