Exploring Continuous Control with Deep Reinforcement Learning and Unity

Andy Brown, Elliott Skomski, Ian Littke, Katie Hursh, Nick Knowles

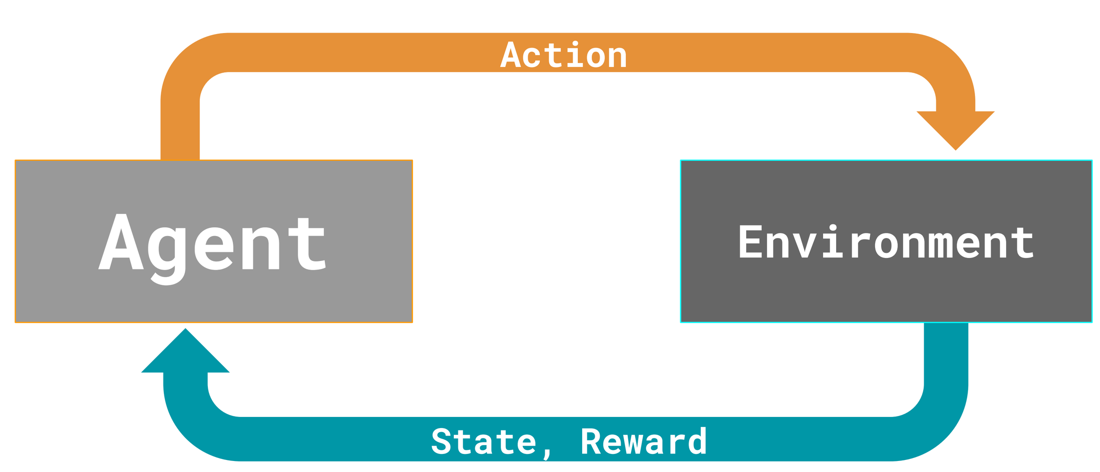
Computer Science Department, Western Washington University

Overview

- **Motivation:** Automating continuous control tasks is an open problem.
- **Goal:** Create a model that can learn complex motor skills and interact with objects.
- **Approach:** Apply reinforcement learning to an environemnt created in Unity.

Reinforcement Learning

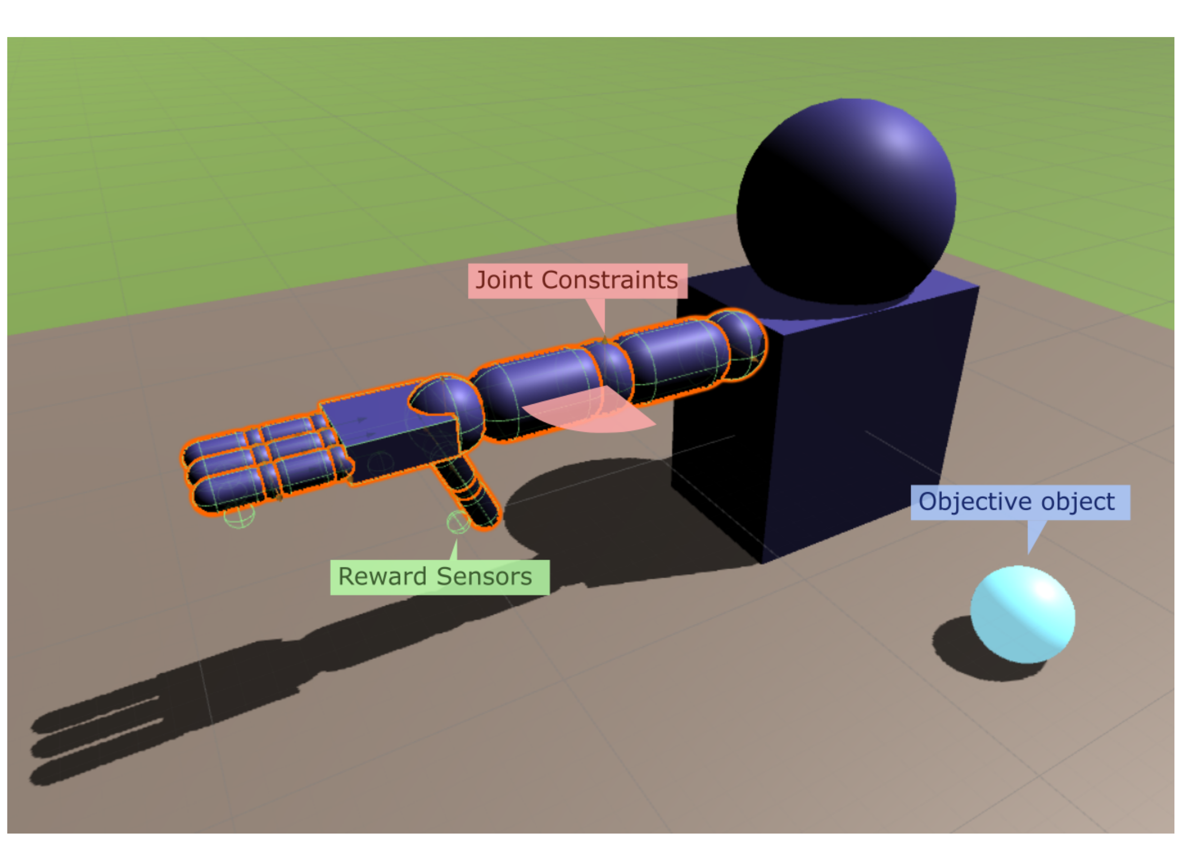
- Reinforcement Learning is a paradigm of machine learning centered around learning through interaction with an environment.



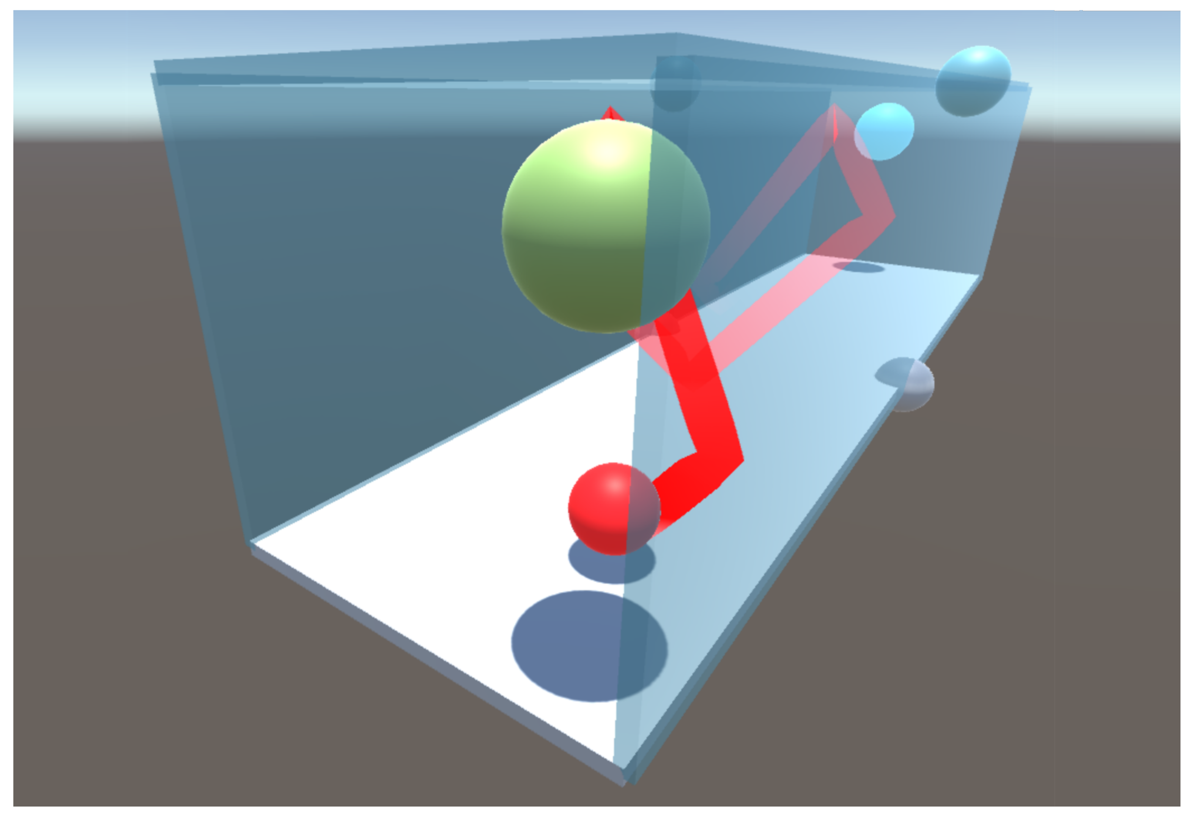
- A *policy function* (denoted $\pi(S)$) is learned, which maps arbitrary states to actions which maximize long-term rewards.

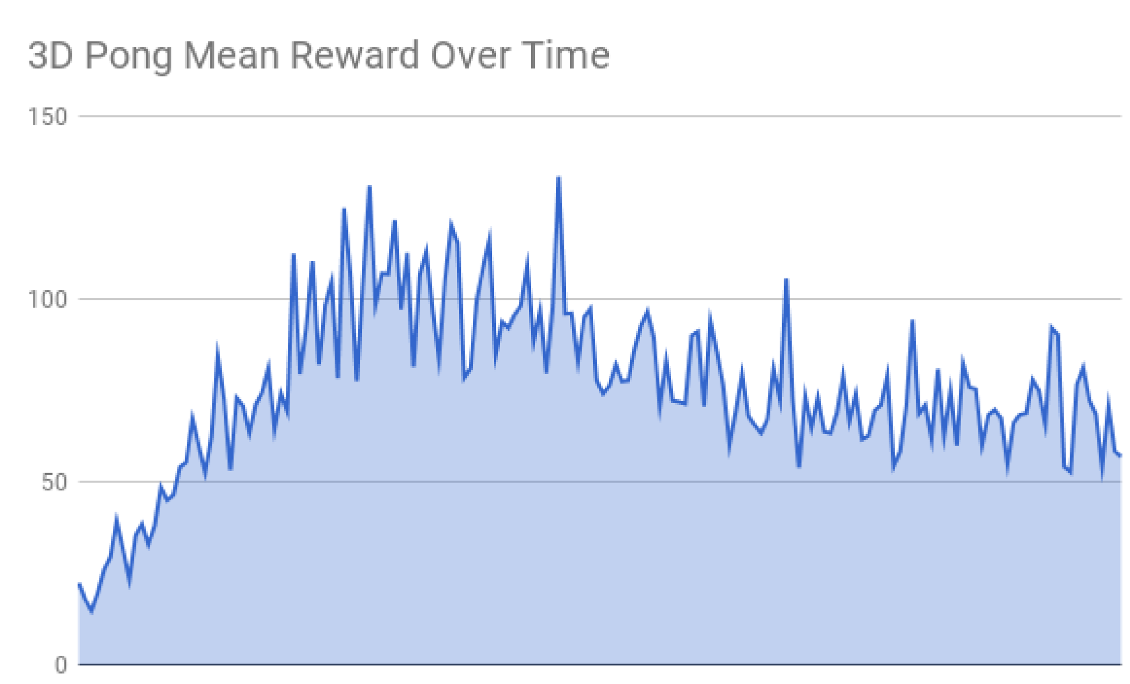
Arm Environment

- Environments are created with the Unity game engine using rigid body physics.
- *Unity ML-Agents* provides an interface for communication between Unity agents and Tensorflow graphs.



- Joints constrained in x, y and z rotations.
- Agent applies torque to each joint individually to move the palm towards the target.
- Reward based on distance to target object.
- Sensors on the fingers provide additional reward for touching the target object.





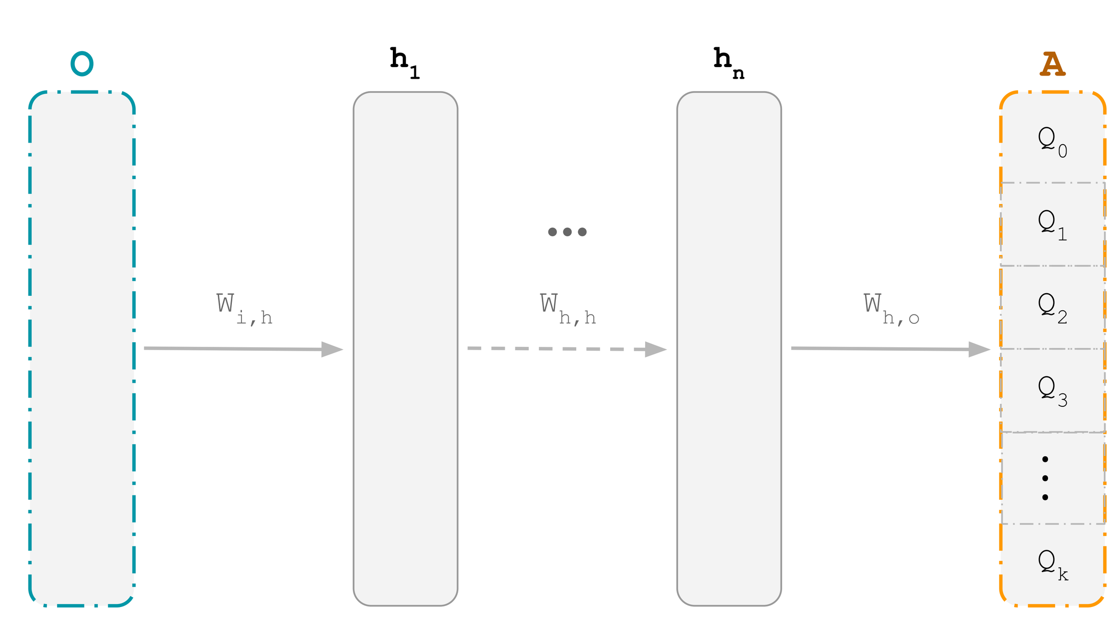
3D Pong Environment

- 3D Pong created as a simple enviroment for testing different model architectures.
- Actions only need to control one object.

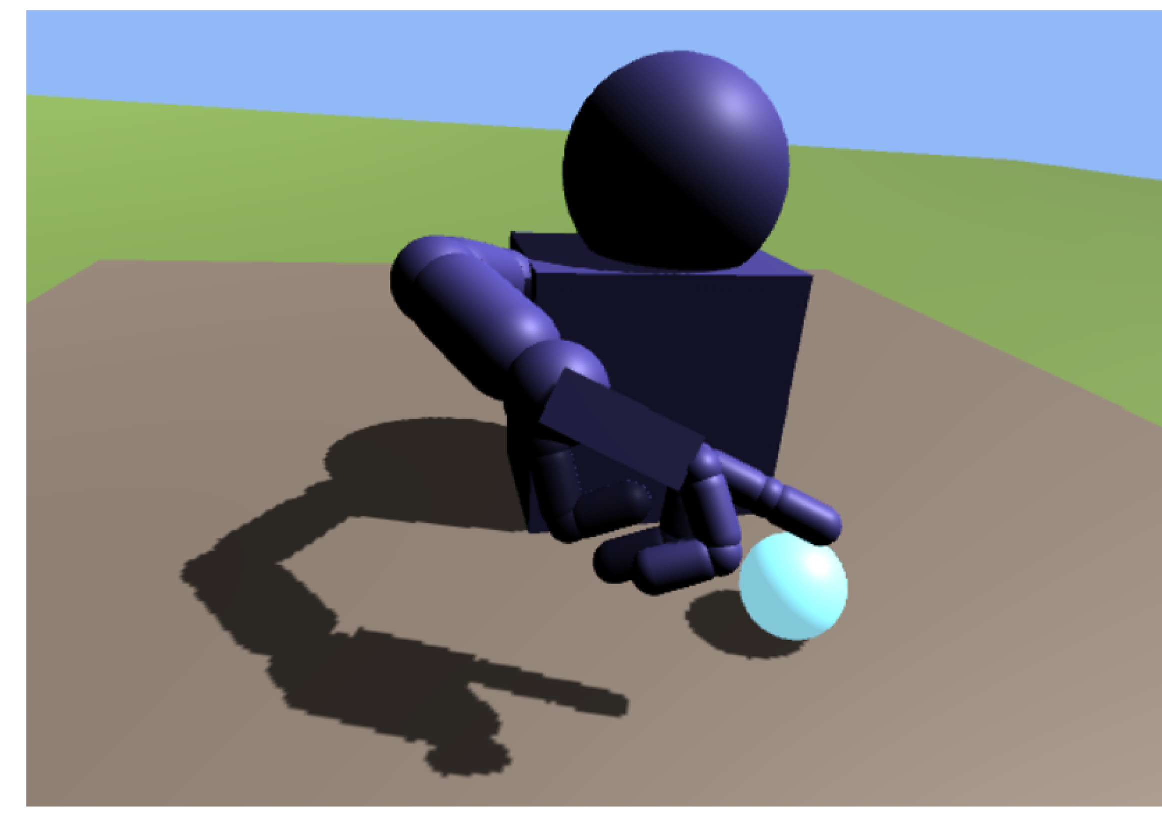
PPO and DQN

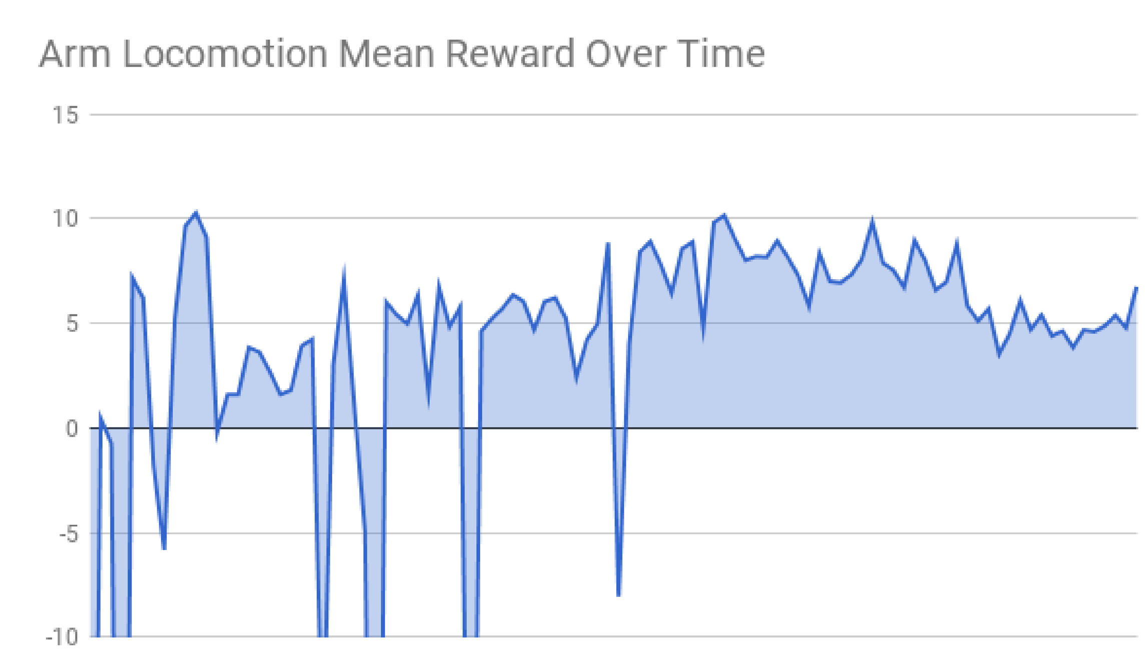
Proximal policy optimization (PPO) and Deep Q-Networks (DQN) are algorithms chosen to optimize our agents' policies:

- **PPO** optimizes policy along with surrogate function that estimates optimal rate of change of policy.
- **DQN** learns an action-value function that is used to determine the best action to take given some observation.



- Simplified DQN model.
  - **O** is an observation of state variables.
  - Vectors $\mathbf{h_i}$ are hidden state vectors.
  - **A** is a vector of long term **Q** values corresponding to a finite set of actions.





Key Equations

Letting θ represent the weights in a neural network, the loss function for PPO is

$$\mathbf{L_t}(\theta) = \mathbb{E}[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta)), 1-\epsilon, 1+\epsilon)\hat{A}_t]$$

A Q-value corresponds to the expected long term value of a state action pair,

$$\mathbf{Q}(s_t, a_t) = \mathbb{E}[\mathbf{r_t} + \gamma\mathbf{r_{t+1}} + \gamma^2\mathbf{r_{t+2}} + \ldots]$$

The loss function for DQN,

$$\mathbf{L_t}(\theta_t) = \left(r + \gamma Q(s_{t+1}, a^*; \theta_t) - y\right)^2$$

Results

- Rigid Body physics proved problematic in training due to inconsistent behavior.
- Arm learned to avoid touching the object when its x and z movement was not locked.
- Locking x and z allowed model to learn a scooping motion and lift the ball.

Future Work

- Scale tasks to multiple agents to examine what strategies develop between agents.
- Implement cameras in our environment that stream images to our model.
- Use other deep architectures in addition to deep neural networks.
- Continue tuning our models' hyperparameters and our agents' reward functions.