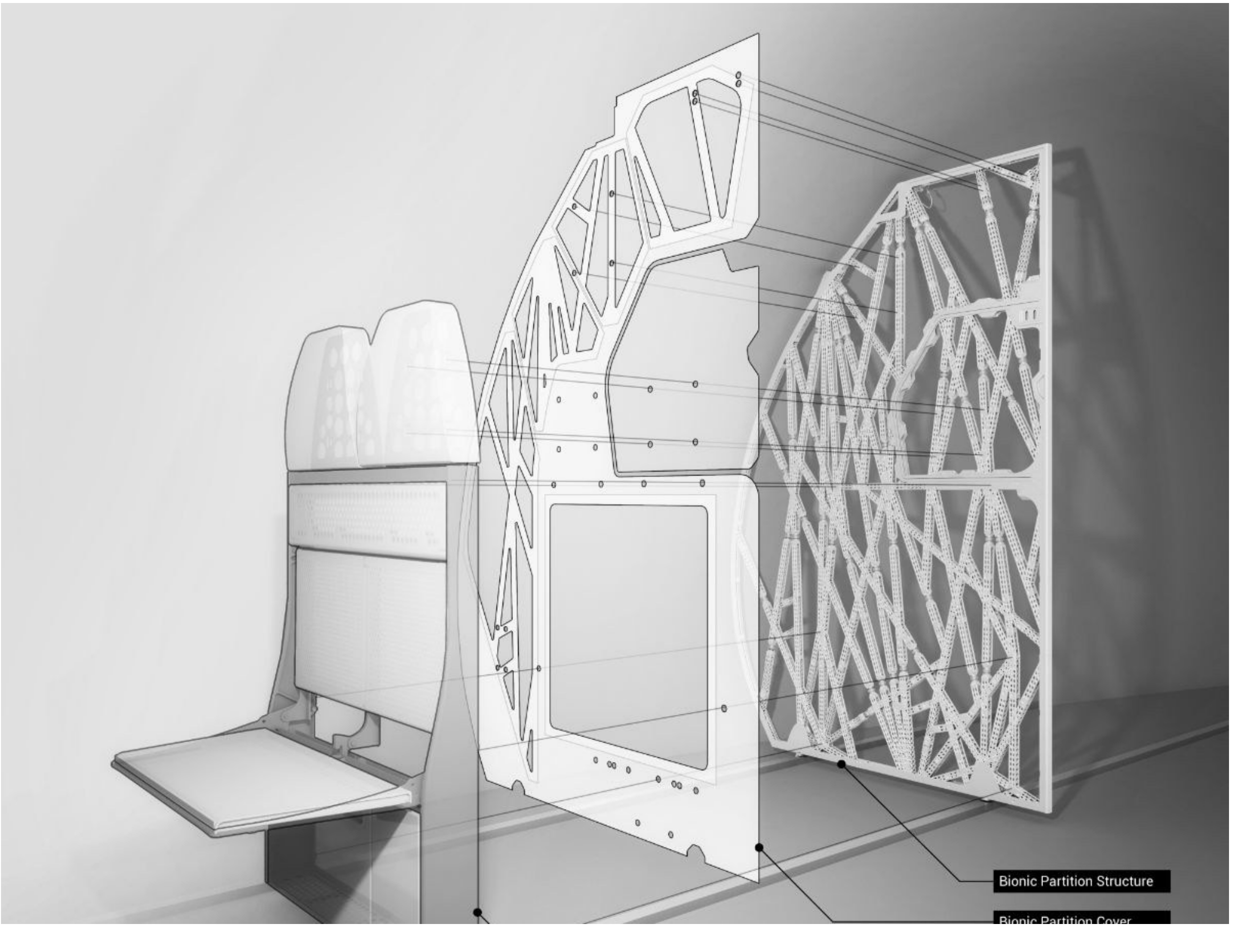
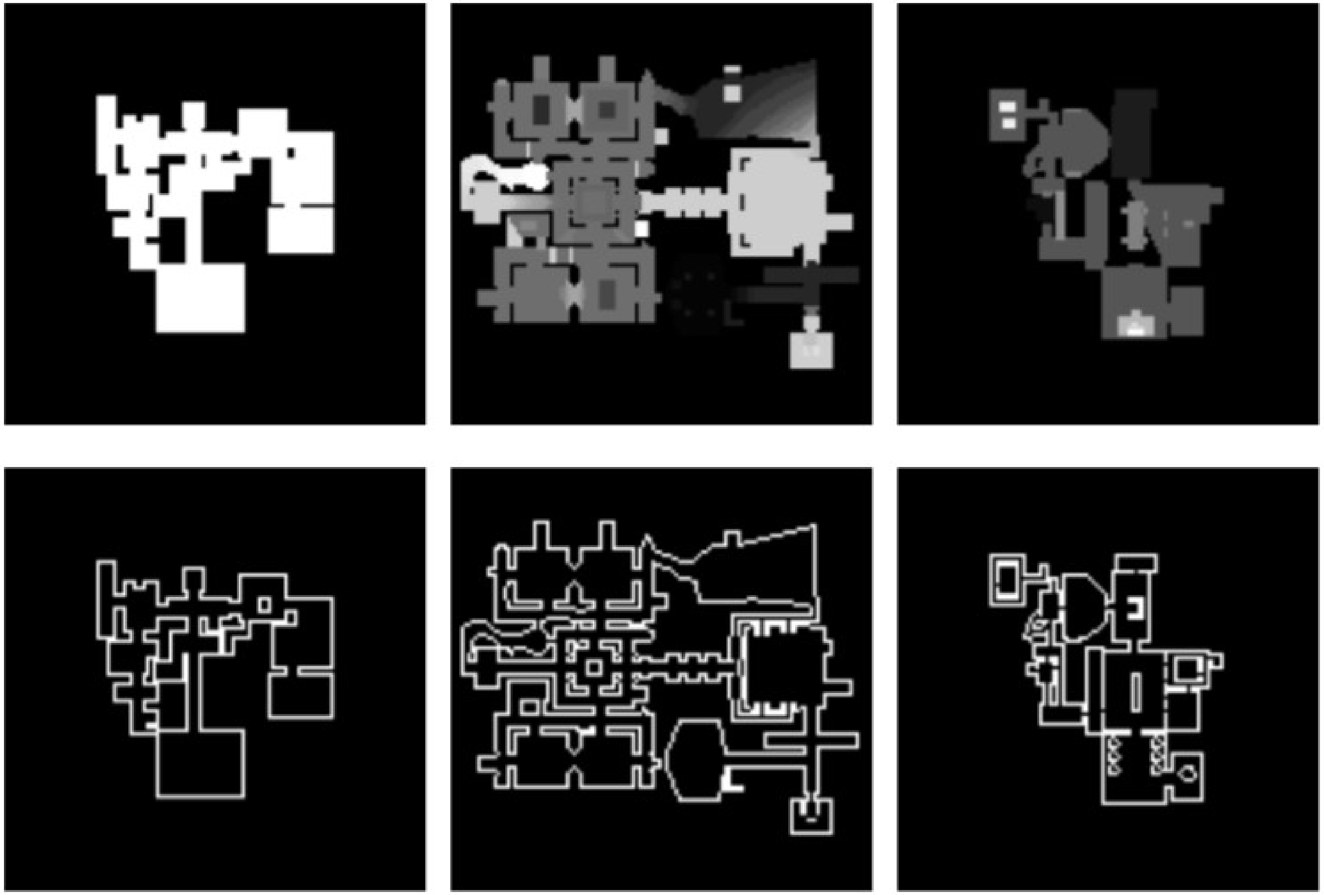
AI in game industry

Texture creation

Structure design

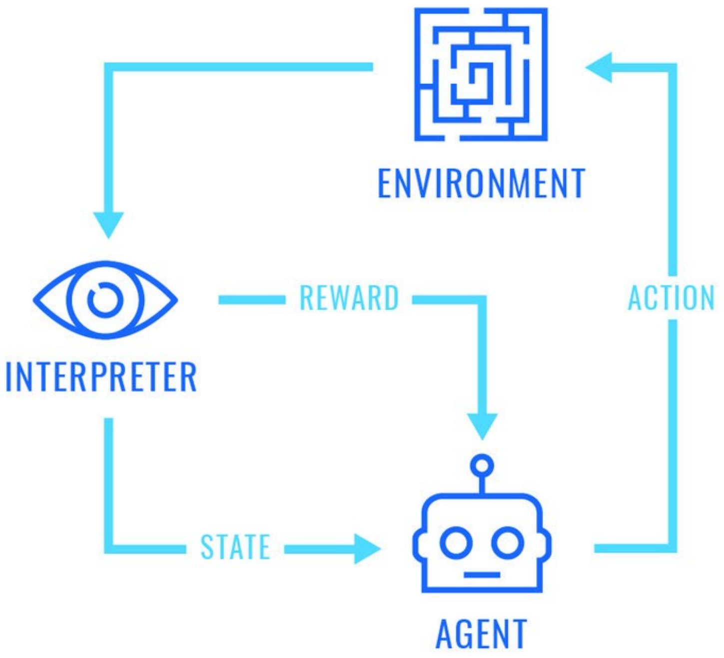


Level generation

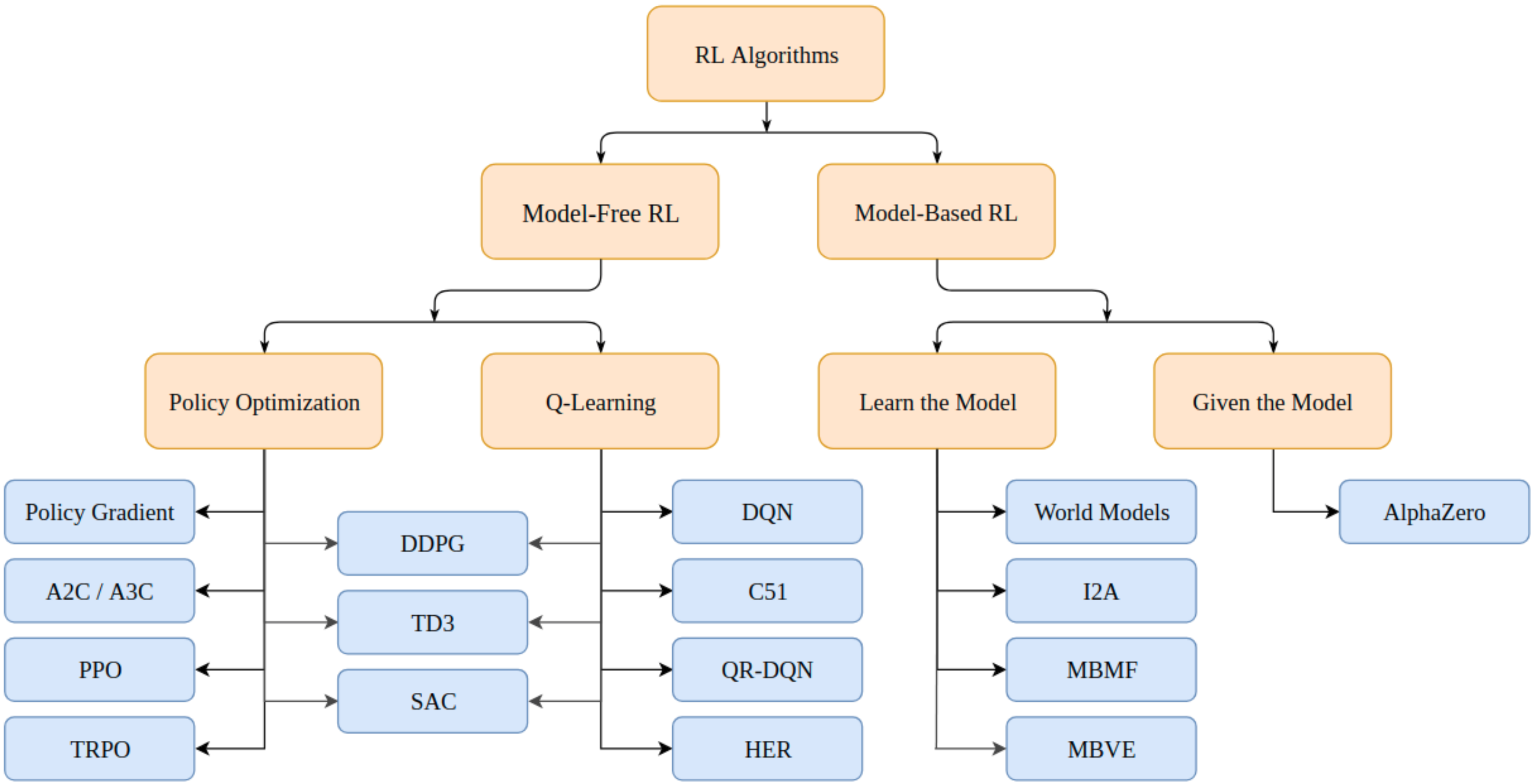


Self driving cars

Reinforcement learning



Algorithms


RL Algorithms
Model-Free RL
Model-Based RL
Policy Optimization
Q-Learning
Learn the Model
Given the Model
Policy Gradient
A2C / A3C
PPO
TRPO
DDPG
TD3
SAC
DQN
C51
QR-DQN
HER
World Models
I2A
MBMF
MBVE
AlphaZero

Value Functions

It's often useful to know the **value** of a state, or state-action pair. By value, we mean the expected return if you start in that state or state-action pair, and then act according to a particular policy forever after. **Value functions** are used, one way or another, in almost every RL algorithm.

There are four main functions of note here.

1. The **On-Policy Value Function**, $V^{\pi}(s)$, which gives the expected return if you start in state s and always act according to policy π:

$$V^{\pi}(s) = \underset{\tau \sim \pi}{\mathrm{E}} \left[R(\tau) \left| s_0 = s \right. \right]$$

2. The **On-Policy Action-Value Function**, $Q^{\pi}(s,a)$, which gives the expected return if you start in state s, take an arbitrary action a (which may not have come from the policy), and then forever after act according to policy π:

$$Q^{\pi}(s,a) = \underset{\tau \sim \pi}{\mathrm{E}} \left[R(\tau) \left| s_0 = s, a_0 = a \right. \right]$$

3. The **Optimal Value Function**, $V^*(s)$, which gives the expected return if you start in state s and always act according to the *optimal* policy in the environment:

$$V^*(s) = \max_{\pi} \underset{\tau \sim \pi}{\mathrm{E}} \left[R(\tau) \left| s_0 = s \right. \right]$$

4. The **Optimal Action-Value Function**, $Q^*(s,a)$, which gives the expected return if you start in state s, take an arbitrary action a, and then forever after act according to the *optimal* policy in the environment:

$$Q^*(s,a) = \max_{\pi} \underset{\tau \sim \pi}{\mathrm{E}} \left[R(\tau) \left| s_0 = s, a_0 = a \right. \right]$$

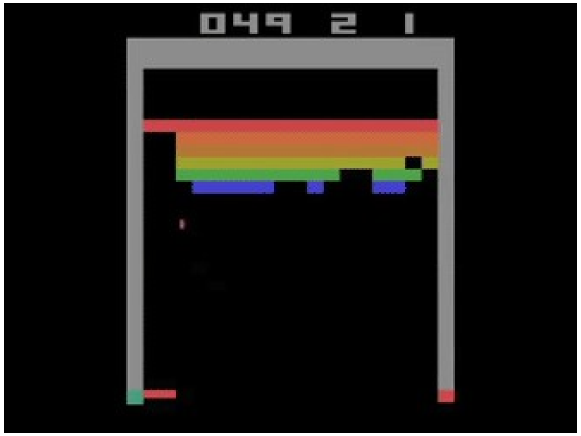
Advantage Functions

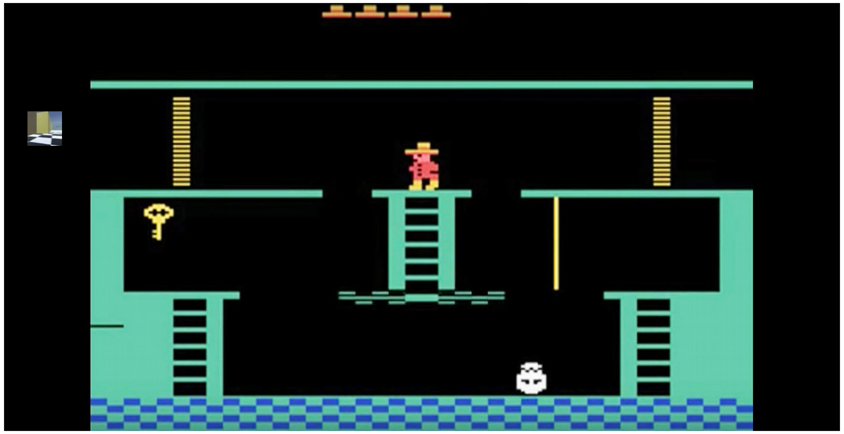
Sometimes in RL, we don't need to describe how good an action is in an absolute sense, but only how much better it is than others on average. That is to say, we want to know the relative **advantage** of that action. We make this concept precise with the **advantage function.**

The advantage function $A^{\pi}(s, a)$ corresponding to a policy π describes how much better it is to take a specific action a in state s, over randomly selecting an action according to $\pi(\cdot|s)$, assuming you act according to π forever after. Mathematically, the advantage function is defined by

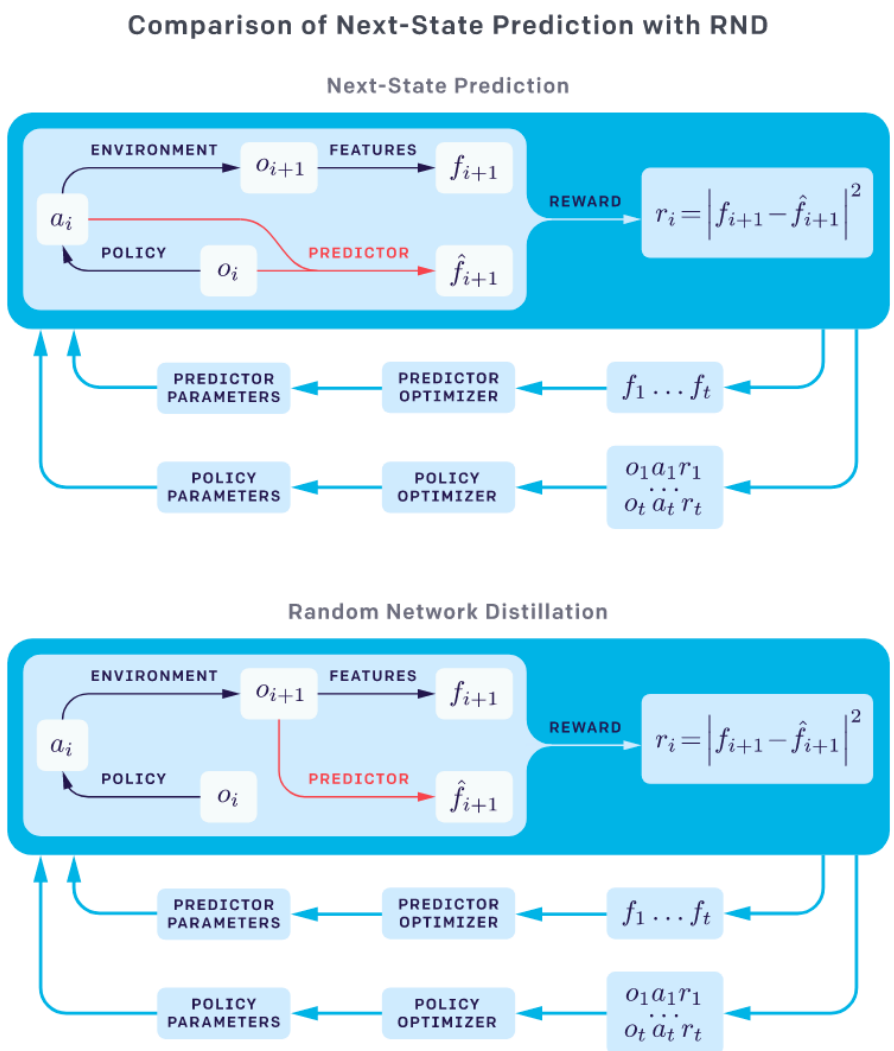
$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s).$$

Dqn





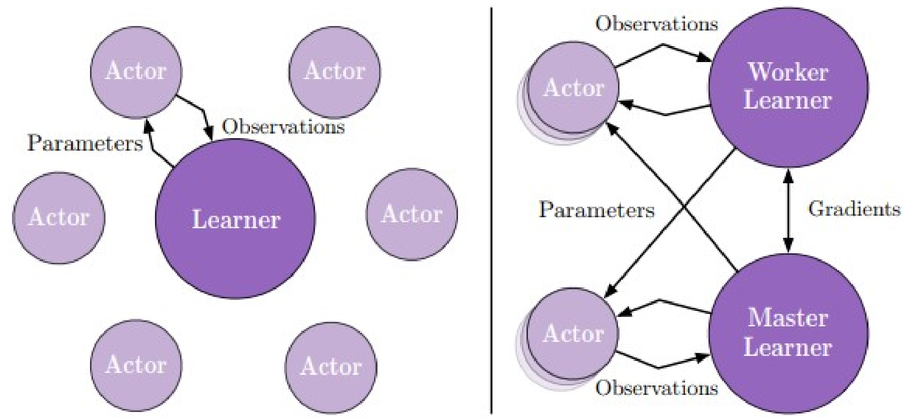
Curiosity

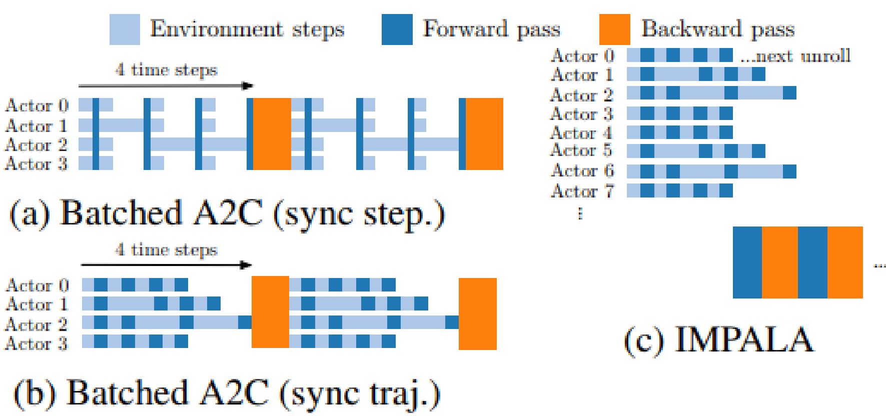

Comparison of Next-State Prediction with RND
Next-State Prediction
ENVIRONMENT
FEATURES
POLICY
PREDICTOR
REWARD
o_{i+1}
f_{i+1}
a_i
o_i
$\hat{f}_{i+1}$
$r_i = \left| f_{i+1} - \hat{f}_{i+1} \right|^2$
PREDICTOR PARAMETERS
PREDICTOR OPTIMIZER
$f_1 \dots f_t$
POLICY PARAMETERS
POLICY OPTIMIZER
$o_1 a_1 r_1 \dots o_t a_t r_t$
Random Network Distillation
ENVIRONMENT
FEATURES
POLICY
PREDICTOR
REWARD
o_{i+1}
f_{i+1}
a_i
o_i
$\hat{f}_{i+1}$
$r_i = \left| f_{i+1} - \hat{f}_{i+1} \right|^2$
PREDICTOR PARAMETERS
PREDICTOR OPTIMIZER
$f_1 \dots f_t$
POLICY PARAMETERS
POLICY OPTIMIZER
$o_1 a_1 r_1 \dots o_t a_t r_t$

A2C/A3C

INTUITIVE RL
INTRO TO ADVANTAGE ACTOR CRITIC (A2C)
WRITTEN BY RUDY GILMAN
ART BY KATHERINE WANG
AGENT: CRANBERRY FOX
IN REINFORCEMENT LEARNING, AN AGENT MOVES THROUGH STATES IN AN ENVIRONMENT BY TAKING ACTIONS, TRYING TO MAXIMIZE REWARDS ALONG THE WAY.
STATE
ACTION
REWARD
A2CS TAKE IN A STATE—SENSORY INPUTS IN CRANBERRY'S CASE—AND GENERATE TWO OUTPUTS:
1) AN ESTIMATE OF HOW MANY REWARDS THEY EXPECT TO GET FROM THAT POINT ONWARDS, THE STATE VALUE.
2) A RECOMMENDATION OF WHAT ACTION TO TAKE, THE POLICY
THE "CRITIC"
WOW, WHAT A WONDERFUL GLEN! THIS WILL BE A FRUITFUL DAY OF FORAGING. I BET I'LL GATHER 20 REWARD POINTS BEFORE SUNSET TODAY.
$V(\hat{S}) = 20$
THE "ACTOR"
THOSE FLOWERS LOOK PRETTY, I'M FEELING DRAWN TOWARDS "A"...
A: 80%
B: 10%
C: 10%
A
B
C

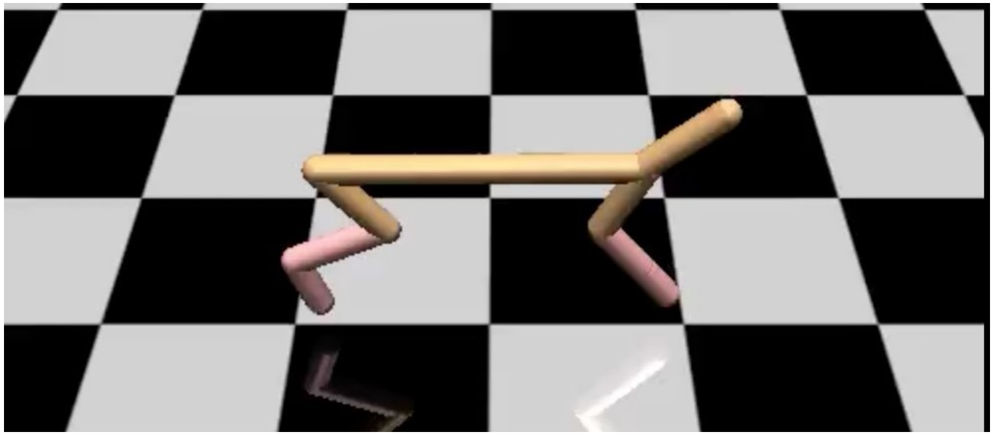
Impala

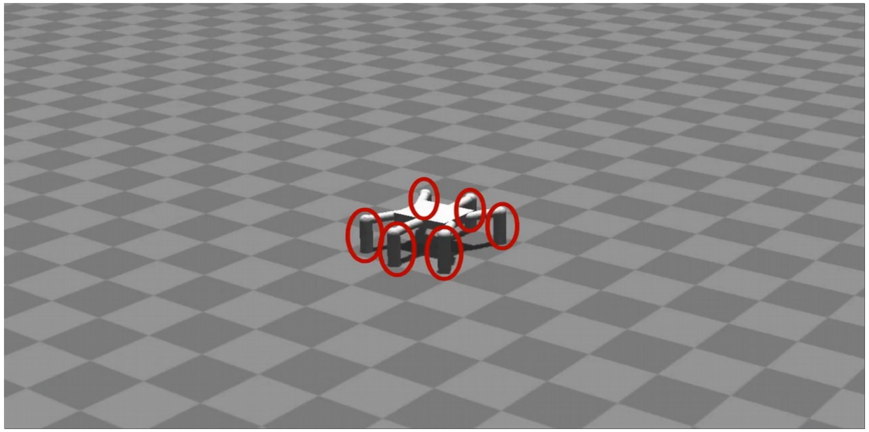




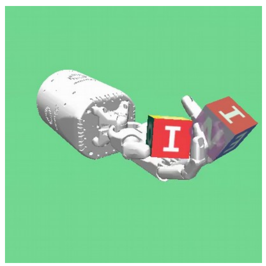
$$
\begin{aligned}
& \sum_{t\geq 0}\gamma^t \mathbb{E}_\mu\left[\left(\prod_{s=0}^{t-2} c_s\right)[\rho_{t-1} - c_{t-1}\rho_t]\right] \\
= & \sum_{t\geq 0}\gamma^t \mathbb{E}_\mu\left[\left(\prod_{s=0}^{t-2} c_s\right)\rho_{t-1}\right] - \sum_{t\geq 0}\gamma^t \mathbb{E}_\mu\left[\left(\prod_{s=0}^{t-1} c_s\right)\rho_t\right] \\
= & \sum_{t\geq 0}\gamma^t \mathbb{E}_\mu\left[\left(\prod_{s=0}^{t-2} c_s\right)\rho_{t-1}\right] - \gamma^{-1}\left(\sum_{t\geq 0}\gamma^t \mathbb{E}_\mu\left[\left(\prod_{s=0}^{t-2} c_s\right)\rho_{t-1}\right] - 1\right) \\
= & \gamma^{-1} - (\gamma^{-1} - 1)\underbrace{\sum_{t\geq 0}\gamma^t \mathbb{E}_\mu\left[\left(\prod_{s=0}^{t-2} c_s\right)\rho_{t-1}\right]}_{\geq 1+\gamma\mathbb{E}_\mu\rho_0} \\
\leq & 1 - (1-\gamma)\mathbb{E}_\mu\rho_0 \\
\leq & 1 - (1-\gamma)\beta \\
< & 1.
\end{aligned}
$$

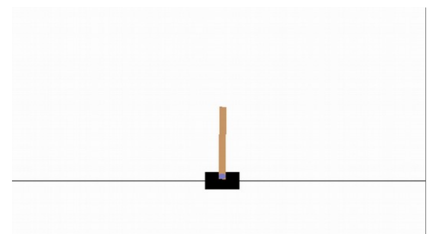
Environments



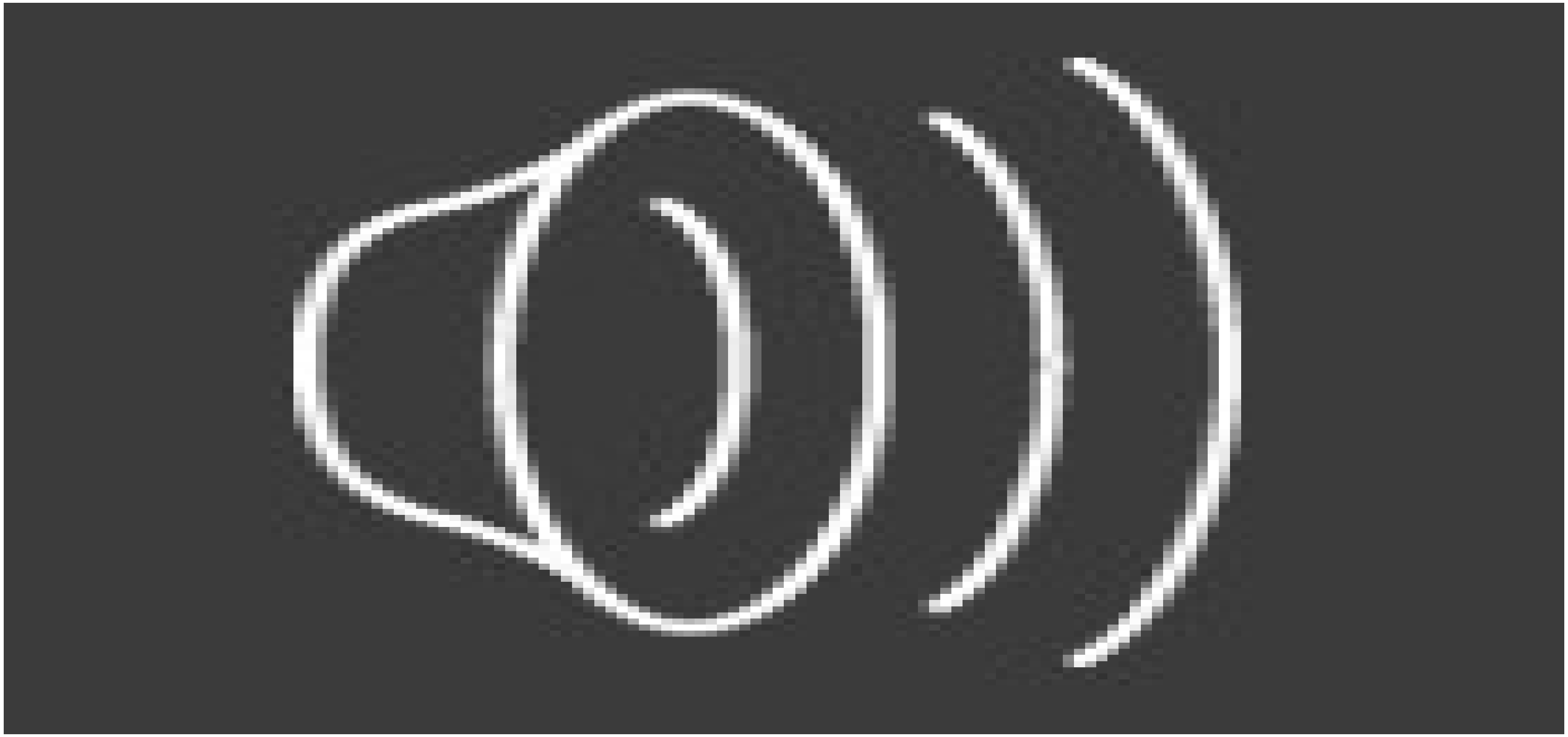


OpenAI gym





DMlab




Go to the blue object

Unity

Links

- https://spinningup.openai.com/en/latest/
- https://ray.readthedocs.io/en/latest/rllib.html
- https://hackernoon.com/intuitive-rl-intro-to-advantage-actor-critic-a2c-4ff545978752
- https://openai.com/blog/
- https://deepmind.com/blog/
- https://distill.pub/

Environments

- https://github.com/mwydmuch/ViZDoom
- https://gym.openai.com/envs/#classic_control
- https://github.com/deepmind/lab
- https://unity3d.com/machine-learning/

Whitepapers

- https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf
- https://arxiv.org/pdf/1707.06347.pdf
- https://arxiv.org/pdf/1802.01561.pdf
- https://arxiv.org/pdf/1602.01783.pdf

Wingman

- https://wingman.ai/careers/