Continuous Action Space

Deep Deterministic Policy Gradient

Bushnag

Abstract

Thinking of the future; it is impossible to envision the world without AI or Robots, whether they are being used to solve small or large problems, advance in medicine or make the world a better place, it will continue to make an impact in the present and in the future.

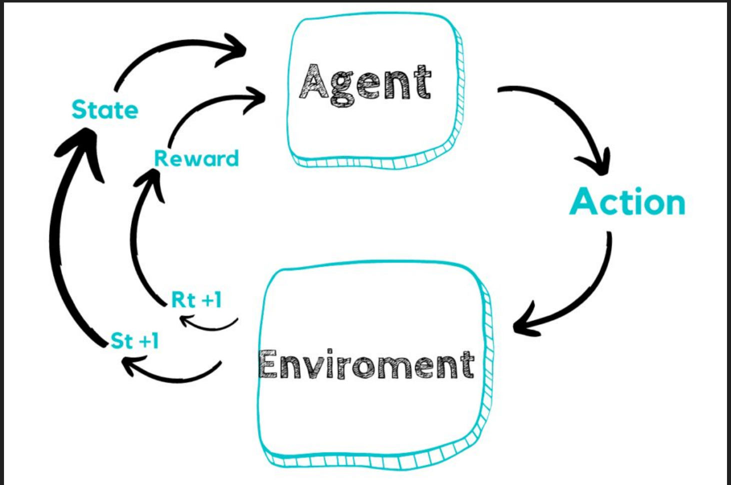
The Problem

- Most of our world consists of a continuous action space. Like the angle of steering on a self-driving car, the force exerted to move the arm of a robot…etc

- To address that issue many methods/techniques have been developed to solve that problem.

Today we will explore: *Deep Deterministic Policy Gradient.*

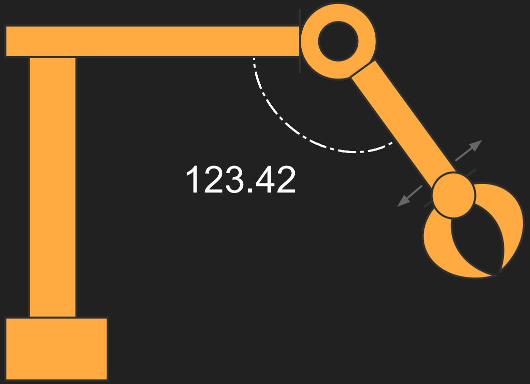
The Setup



Action Spaces

- **Discrete Action Space:**
  1. *Generates logits which passes through softmax.*
  2. *Gives us probabilities between 0 and 1, all adding up to 1.*
  3. *This is then used to pick a random action.*
  4. *The more certain the network become the more we Exploit and the less we Explore.*

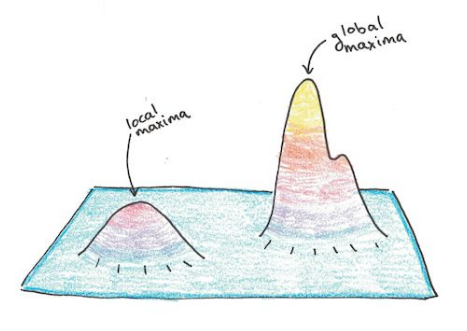
- **Continuous Action Space:**
  1. *Sample from Normal-Probability-Distribution.*
  2. *Standard-Deviation is how far from the mean the sample will be.*
  3. *As the network gets more and more certain about the output the SD gets smaller and smaller.*

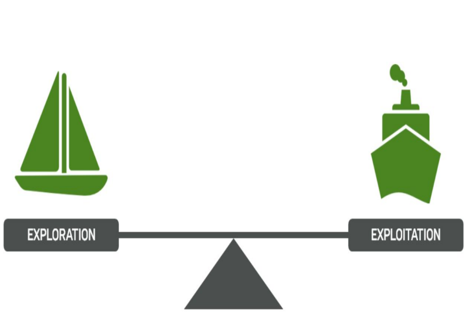


Exploit vs Explore

- Exploit: uses the greedy approach to get the most reward
  - Uses the existing knowledge.

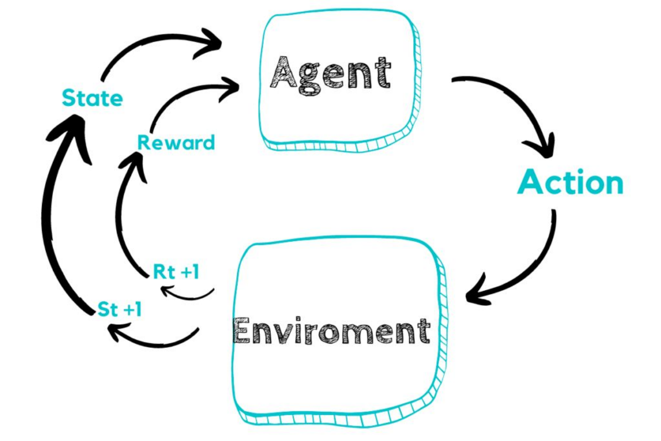
- Explore: uses the greedy approach in the opposite way
  - Instead of picking the best action it uses one of the other options, not necessarily the worst option.





Environment

- The model is off-policy
- There is NO data for these models
- Environment and Agent generate the data



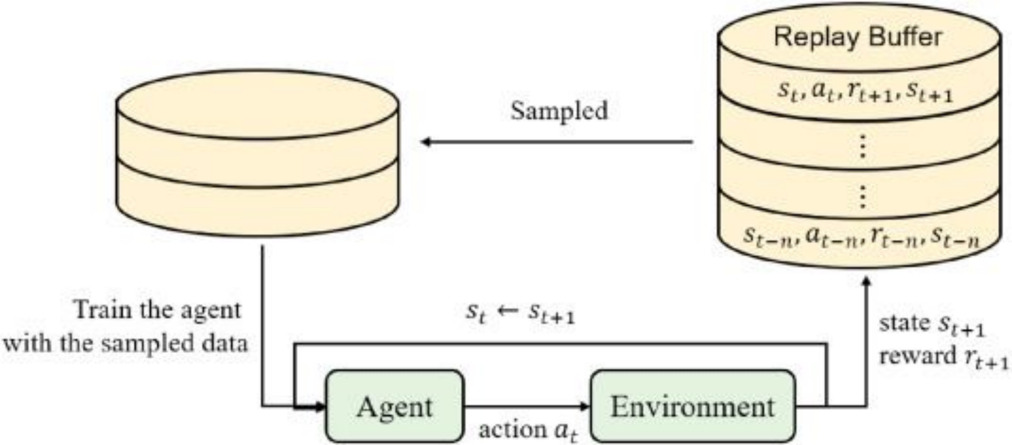
The Memory

1. (State, Action, Reward, Next_State)
2. (State, Action, Reward, Next_State)
3. (State, Action, Reward, Next_State)
4. (State, Action, Reward, Next_State)
5. (State, Action, Reward, Next_State)
6. (State, Action, Reward, Next_State)
7. (State, Action, Reward, Next_State)

   .

   .

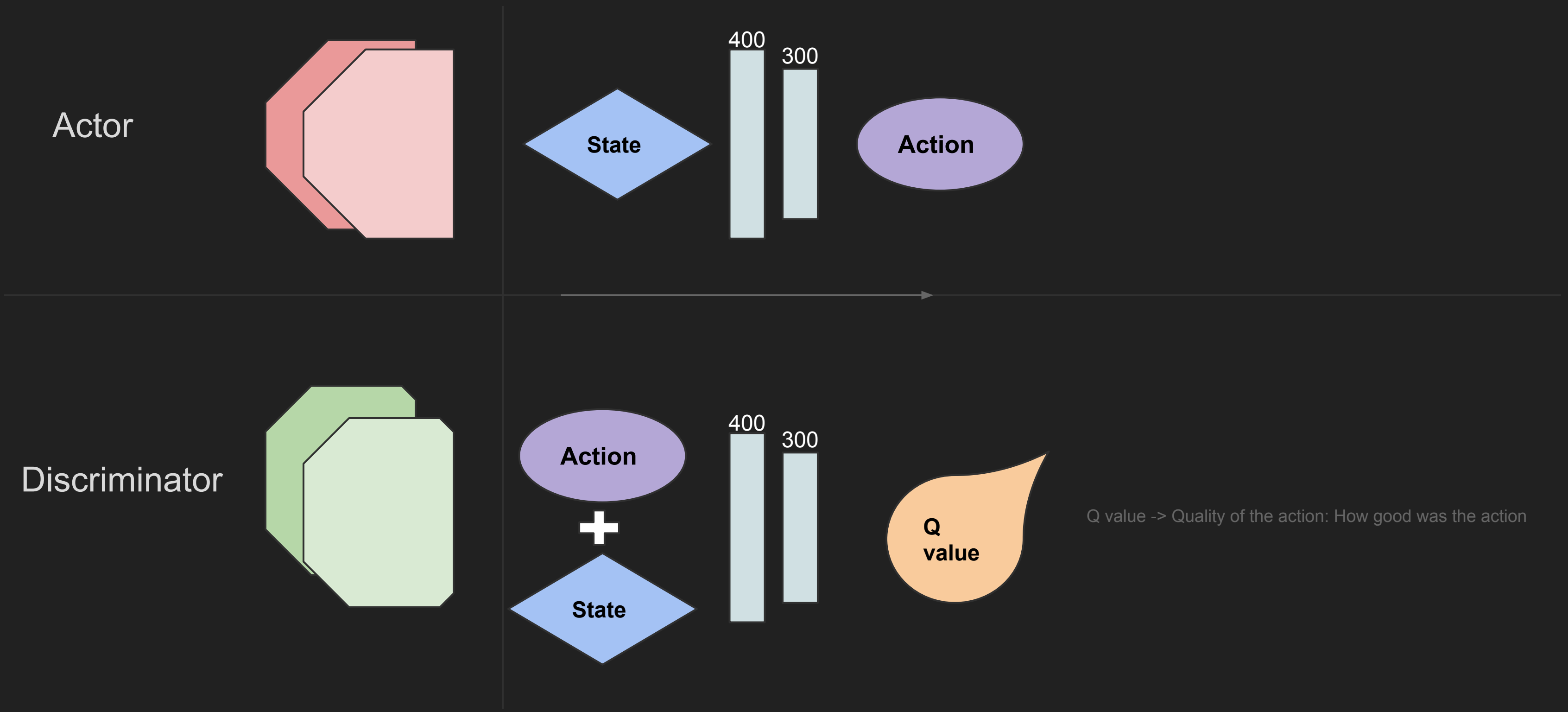
   .



The Agent - Architecture

Discriminator 🟩

Actor 🟥



Actor

State

400

300

Action

Discriminator

Action

+

State

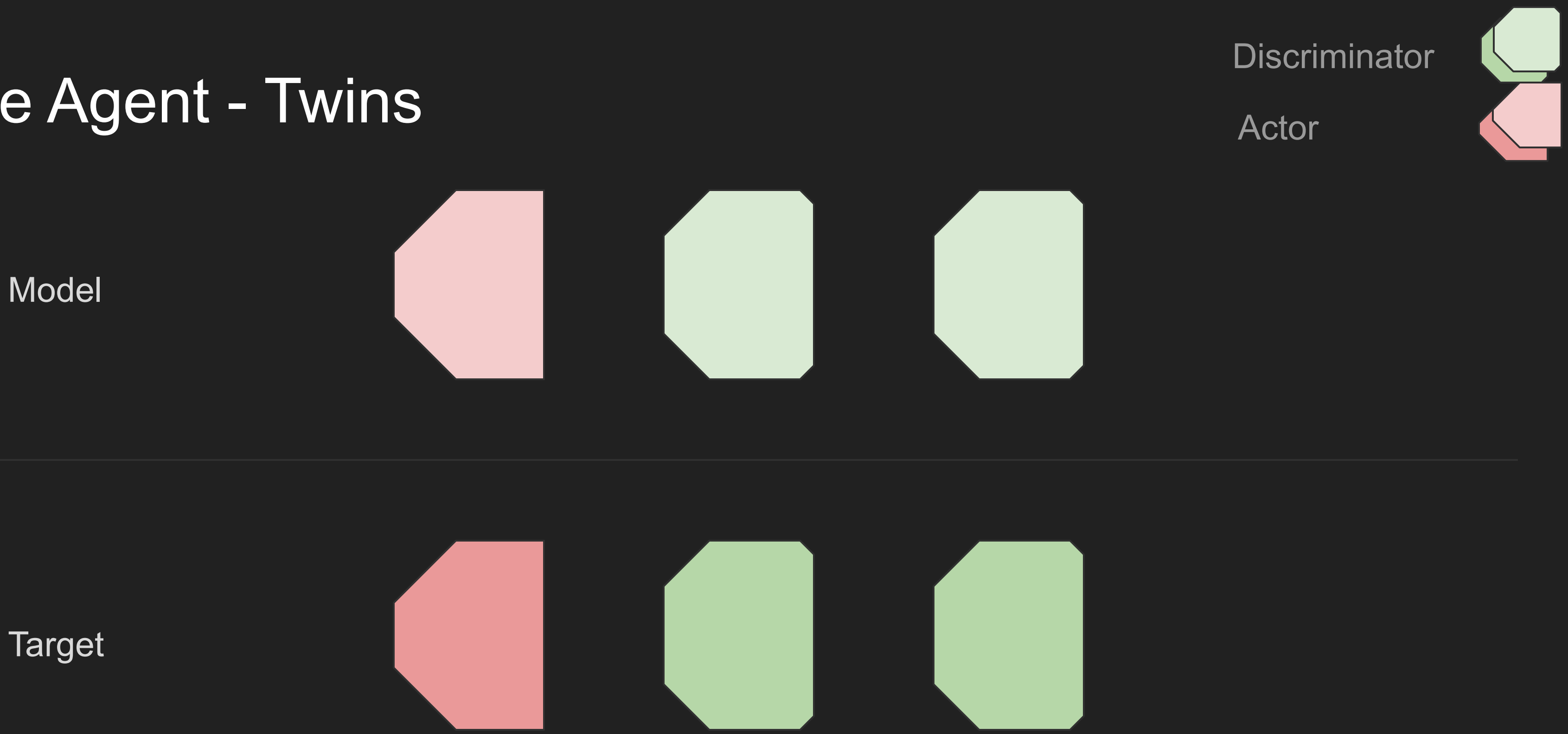
400

300

Q value

Q value -> Quality of the action: How good was the action

The Agent - Twins



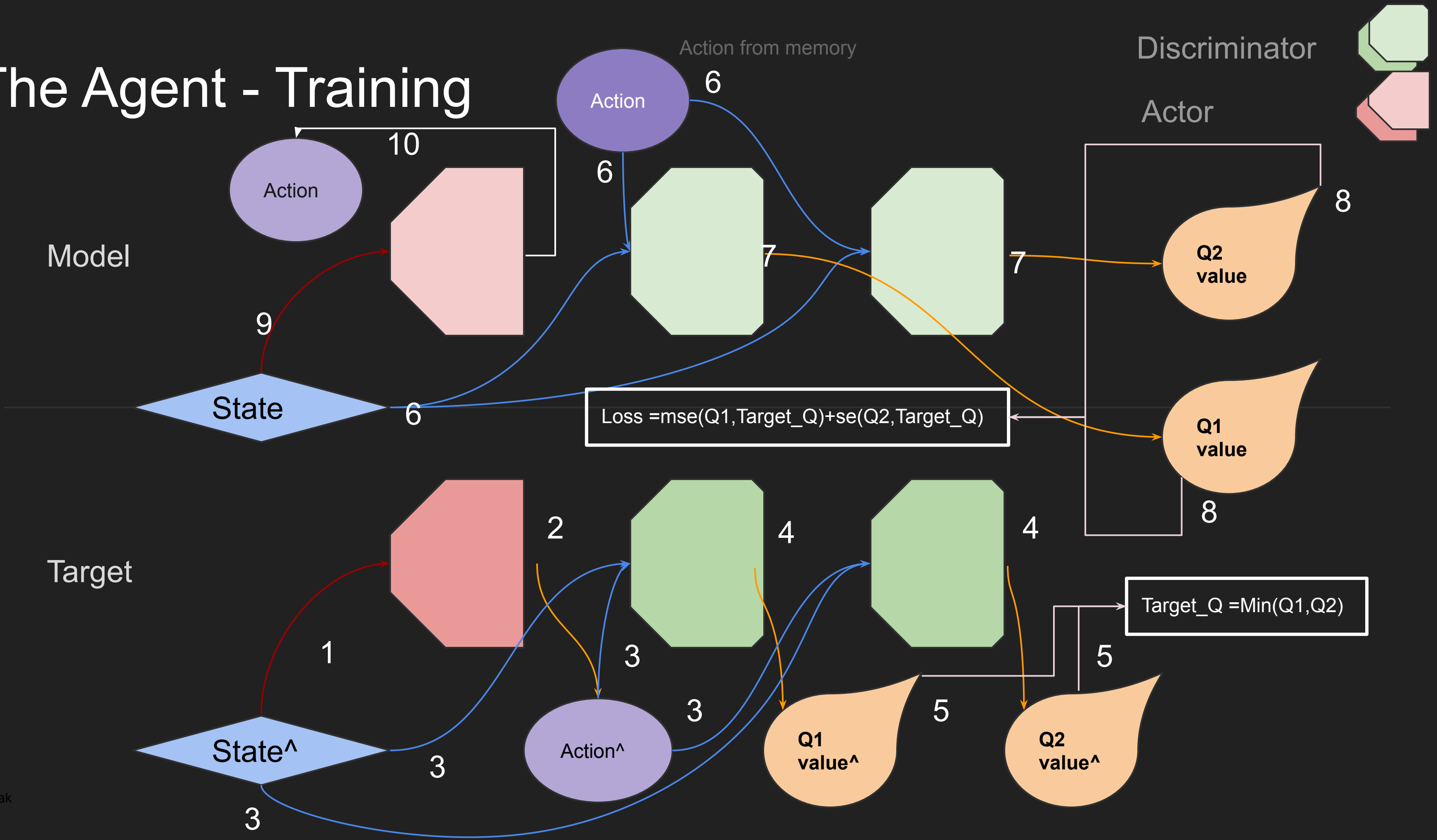
Discriminator

Actor

Model

Target

The Agent - Training



Loss =mse(Q1,Target_Q)+se(Q2,Target_Q)

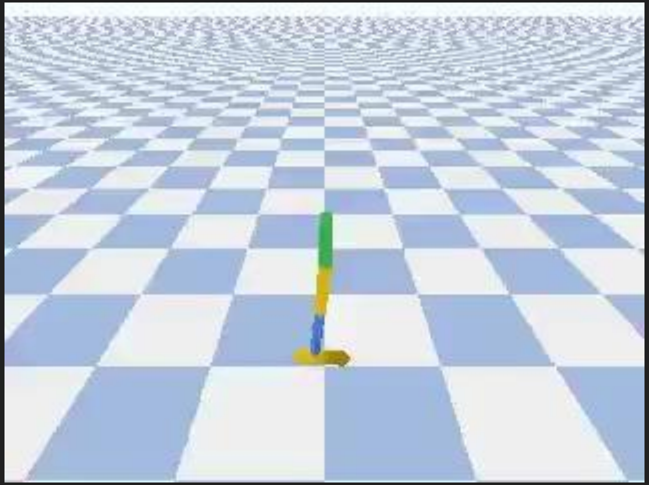
Target_Q =Min(Q1,Q2)

polyak

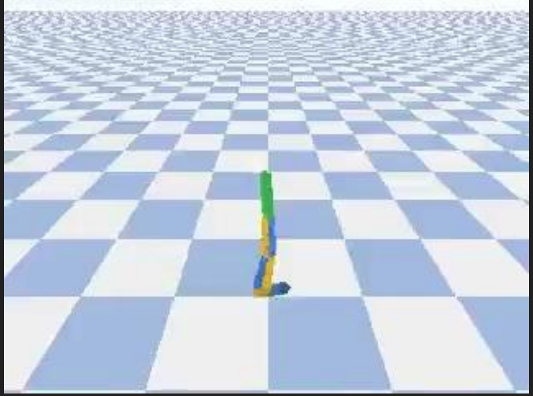
Tools

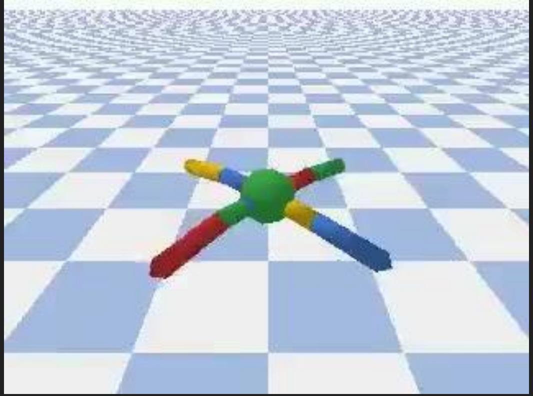
- Google Colab
- Google Drive
- Pytorch: torch
- Numpy: numpy
- OpenAi: gym
- Pybullet: pybullet-gym

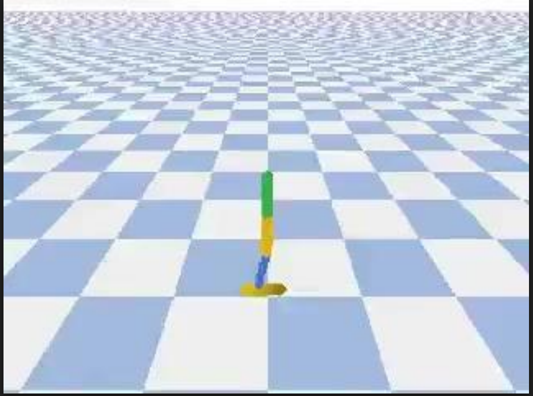
Before Training

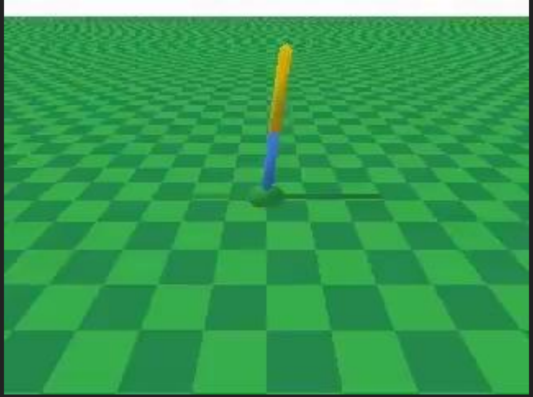


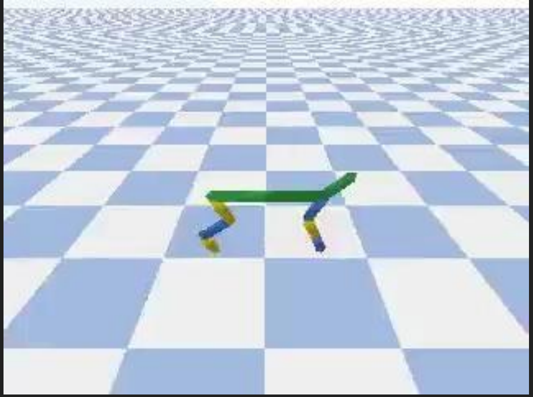
After Training



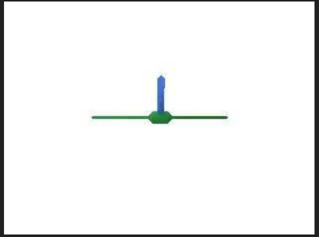


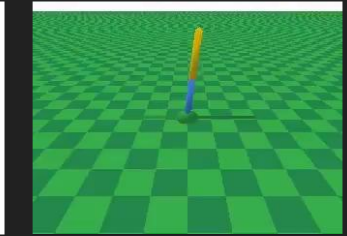


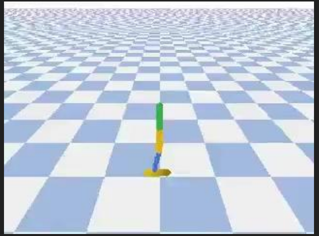


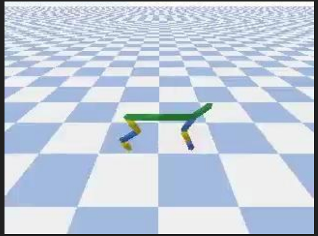


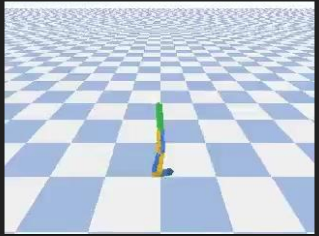
The End

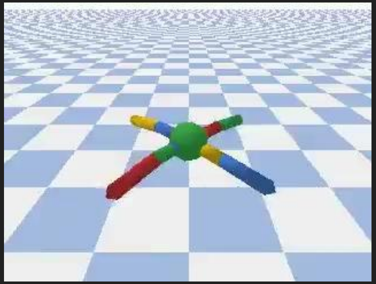












Thank you