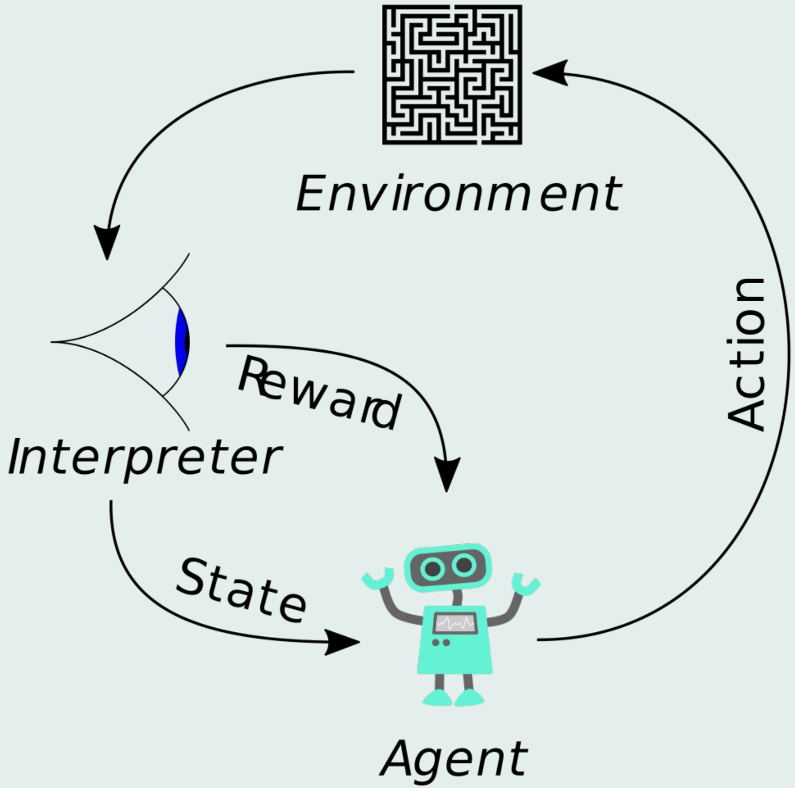
Reinforcement Learning

Introduction

Key Concepts

- Reward
- Interpreter
- Action
- Environment
- Agent



Criteria	Reinforcement Learning	Supervised ML	Unsupervised ML
Definition	Works on interacting with the environment	Learns by using labelled data	Trained using unlabelled data without any guidance.
Type of Data	No – predefined data	Labelled data	Unlabelled data
Type of Problems	Exploitation or Exploration	Regression and classification	Association and Clustering
Training	No supervision	External supervision	No supervision
Approach	Follows the trial-and-error method	Maps the labelled inputs to the known outputs	Understands patterns & discovers the output
Algorithms	Q – Learning, SARSA	Linear Regression, Logistic Regression, SVM, KNN etc.	K – Means, C – Means, Apriori
Aim	Learn a series of action	Calculate outcomes	Discover underlying patterns
Applications	Self Driving Cars, Gaming, Healthcare	Risk Evaluation, Forecast Sales	Recommendation System, Anomaly Detection

State of the art (since 2017)

Distributional Reinforcement Learning with Quantile Regression

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Algorithm 1 Quantile Regression Q-Learning

Require: N, κ
input $x, a, r, x', \gamma \in [0, 1)$
  # Compute distributional Bellman target
  $Q(x', a') := \sum_j q_j \theta_j(x', a')$
  $a^* \leftarrow \arg\max_{a'} Q(x, a')$
  $\mathcal{T}\theta_j \leftarrow r + \gamma\theta_j(x', a^*), \quad \forall j$
  # Compute quantile regression loss (Equation 10)
output $\sum_{i=1}^{N} \mathbb{E}_j \left[\rho^{\kappa}_{\hat{\tau}_i}(\mathcal{T}\theta_j - \theta_i(x, a))\right]$

Proximal Policy Optimization

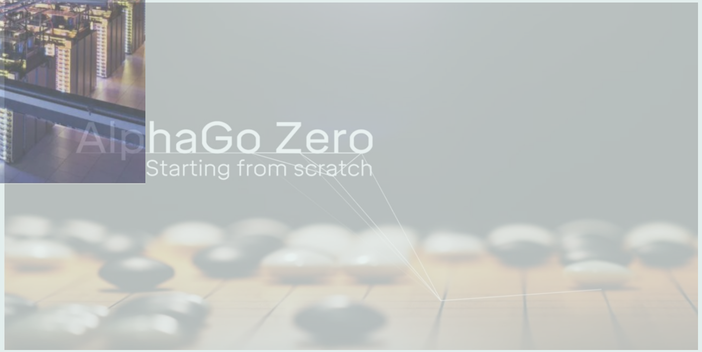
Algorithm 1 PPO, Actor-Critic Style

for iteration$=1, 2, \ldots$ **do**
  for actor$=1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

Reinforcement Learning Applications

Industrial applications

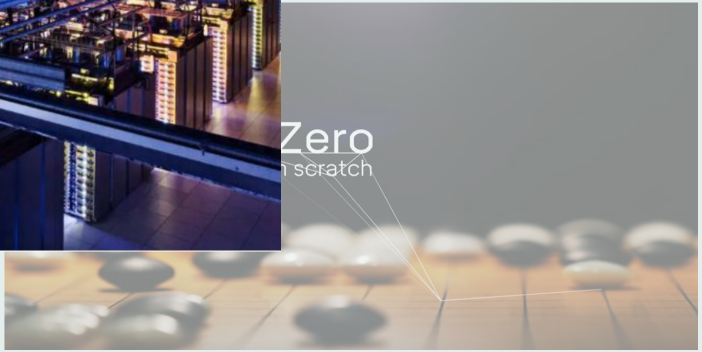
Self-driving car



Source : https://neptune.ai/blog/reinforcement-learning-applications#:~:text=In%20industry%20reinforcement%2C%20learning%2Dbased,to%20cool%20Google%20Data%20Centers

Industrial applications

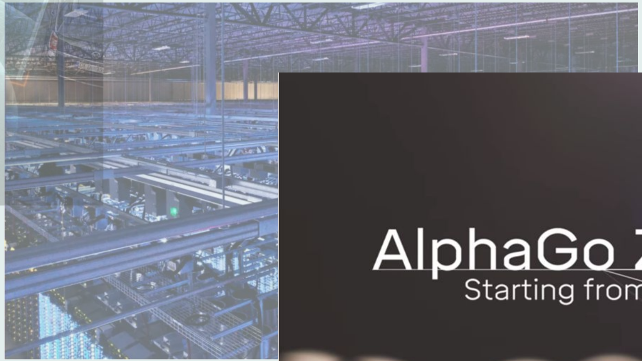
Industry automation



Source : https://neptune.ai/blog/reinforcement-learning-applications#:~:text=In%20industry%20reinforcement%2C%20learning%2Dbased,to%20cool%20Google%20Data%20Centers

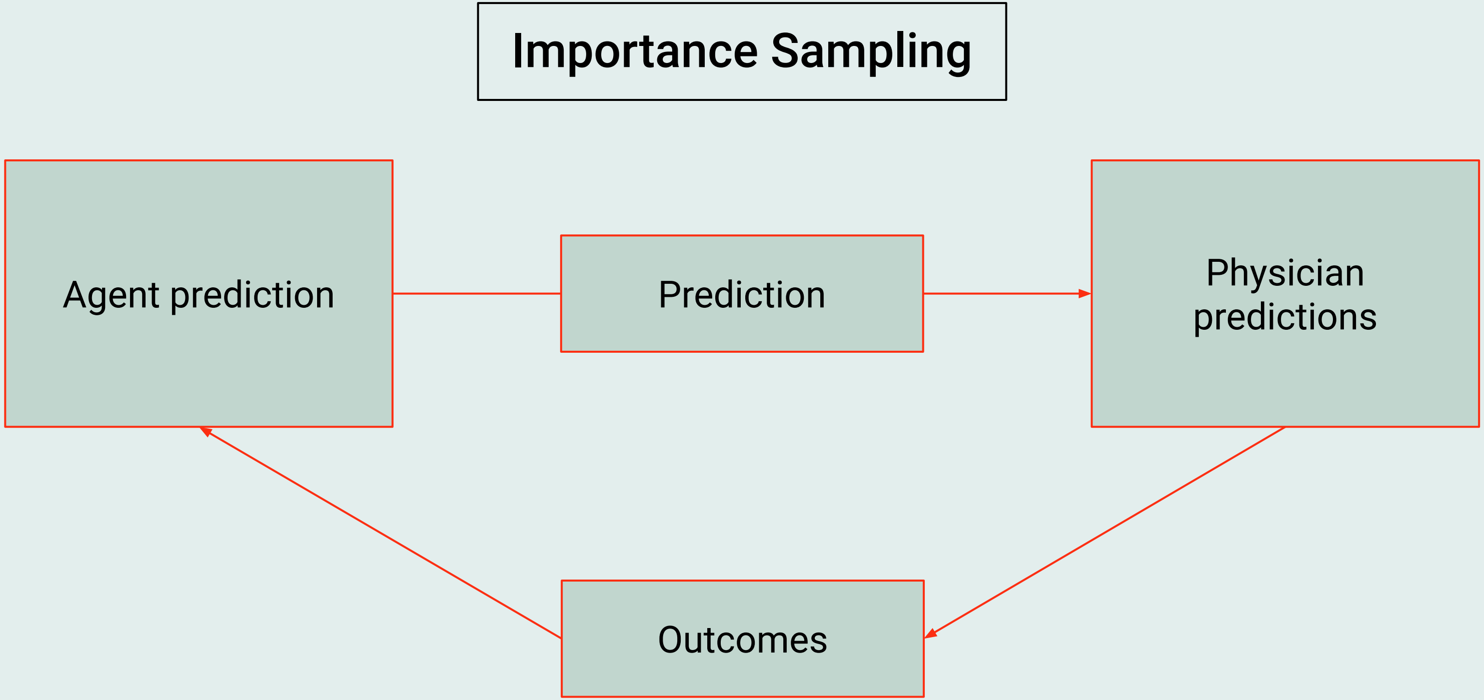
Industrial applications

Gaming



Source : https://neptune.ai/blog/reinforcement-learning-applications#:~:text=In%20industry%20reinforcement%2C%20learning%2Dbased,to%20cool%20Google%20Data%20Centers

Healthcare Use Cases



http://ceur-ws.org/Vol-2872/short02.pdf
https://towardsdatascience.com/a-review-of-recent-reinforcment-learning-applications-to-healthcare-1f8357600407

Long term vs Short term outcomes

Reinforcement Learning algorithms

- **Policy** : The algorithm must find a policy with maximum expected return.

- **Action Space** : Discrete / Continuous.

- **Set Space** : The state space is a set of all the states that the agent can transition to and action

- **Bellman operator** : The Bellman operators are "operators" in that they are mappings from one point to another within the vector space of state values

Reinforcement Learning algorithms

Examples :

1) **Monte Carlo** : require only experience. Meaning, they sample states, actions, and rewards, while interacting with the environment.They are a way to solve RL problems based on averaging sample returns.

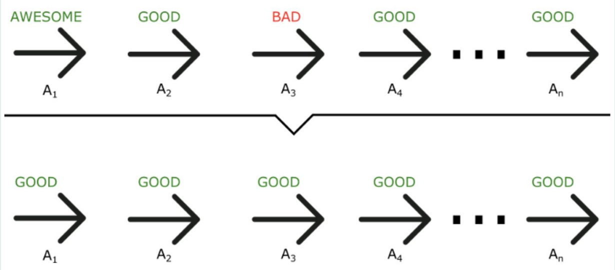
2) **PPO :** Proximal Policy Optimization, or PPO, is a policy gradient method for reinforcement learning. The motivation is to have an algorithm with the data efficiency and reliable performance of TRPO (Trust Region Policy Optimization), while using only first-order optimization.

DEMO

Space invaders

- Input variables : do nothing, go left, go right, fire
- Output variables : reward of the action
- Algorithms : QR-DQN and Maskable PPO
- Number of steps : 10 millions with each algorithm
- The highest score : ~1 800

Breakout



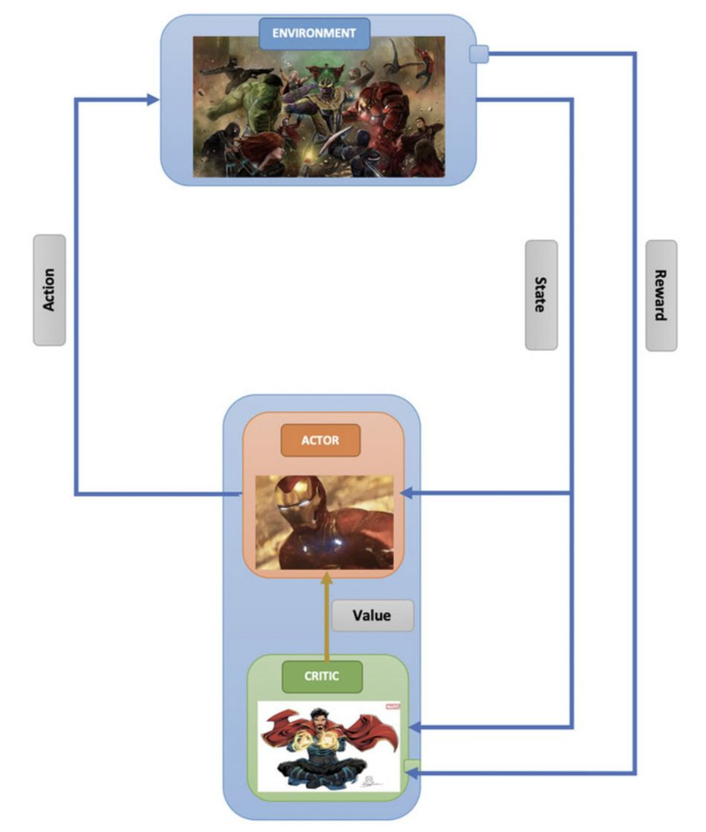


Fig. 11.3 Conceptual design of the actor-critic method

- Input variables : go left or go right
- Output variables : Mean score (and value range)
- Number of steps : up to 4M (different models)
- The Highest score : 29 (mean)
- Algorithm : Advantage Actor Critic (A2C)

Sources :

- https://www.freecodecamp.org/news/an-intro-to-advantage-actor-critic-methods-lets-play-sonic-the-hedgehog-86d6240171d/
- Sewak, Mohit. "Actor-Critic Models and the A3C." *Deep Reinforcement Learning*. Springer, Singapore, 2019. 141-152.

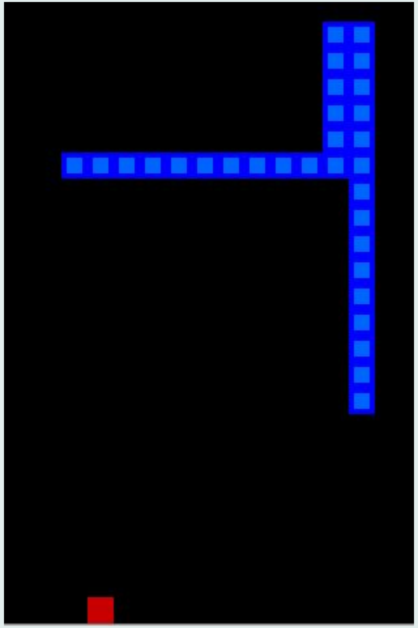
Snake Game - Overview

- **<u>Aim</u>** : create an AI that teaches itself how to play snake.

- The snake knows what it should do and it’s more or less going straight

  for the food and tries not to hit the boundaries.

- With a little math behind the scenes our snake is literally following a strategy



Snake Game - Implementation

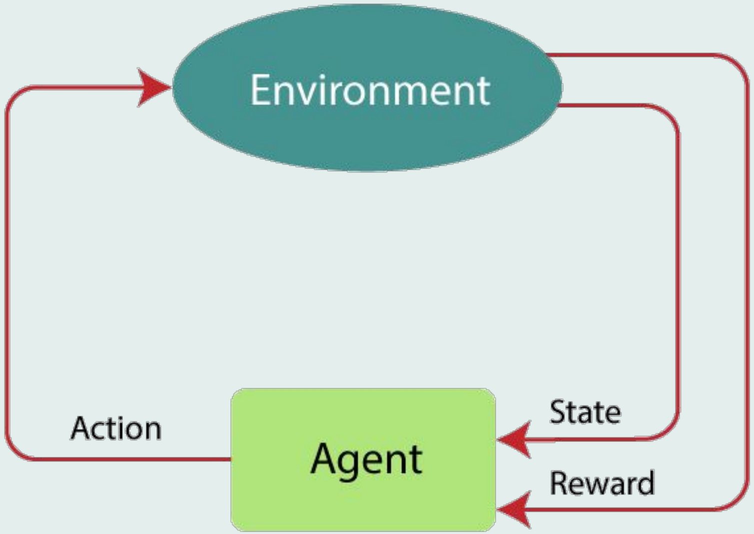
Game ⇨ PyGame

Agent ⇨ PyTorch

The Agent :

- The Game
- The Model

Training : Deep Q learning



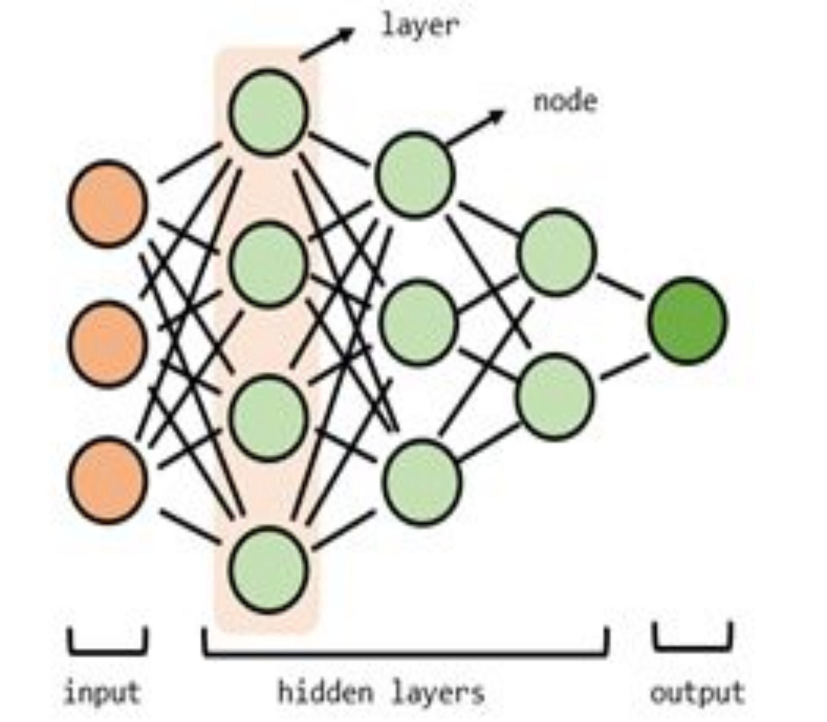
Snake Game - The variables

- **Reward** :
  - +10 : eats food
  - -10 : game over
  - 0 : else

- **Action** :
  - [1, 0, 0] : Current Direction
  - [0, 1, 0] : Right Turn
  - [0, 0, 1] : Left Turn

- **State :**
  - [danger straight, danger right, danger left, direction left, direction right, direction down, direction up, food left, food right, food down, food up)

Snake Game - Model



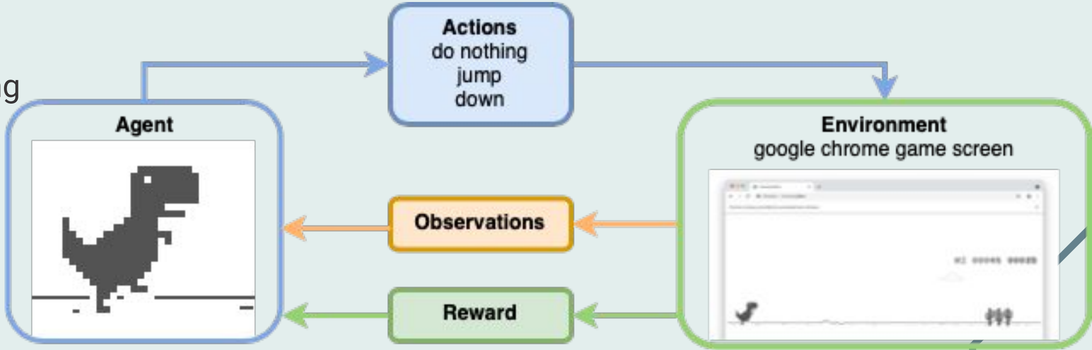
Snake Game - Model Training

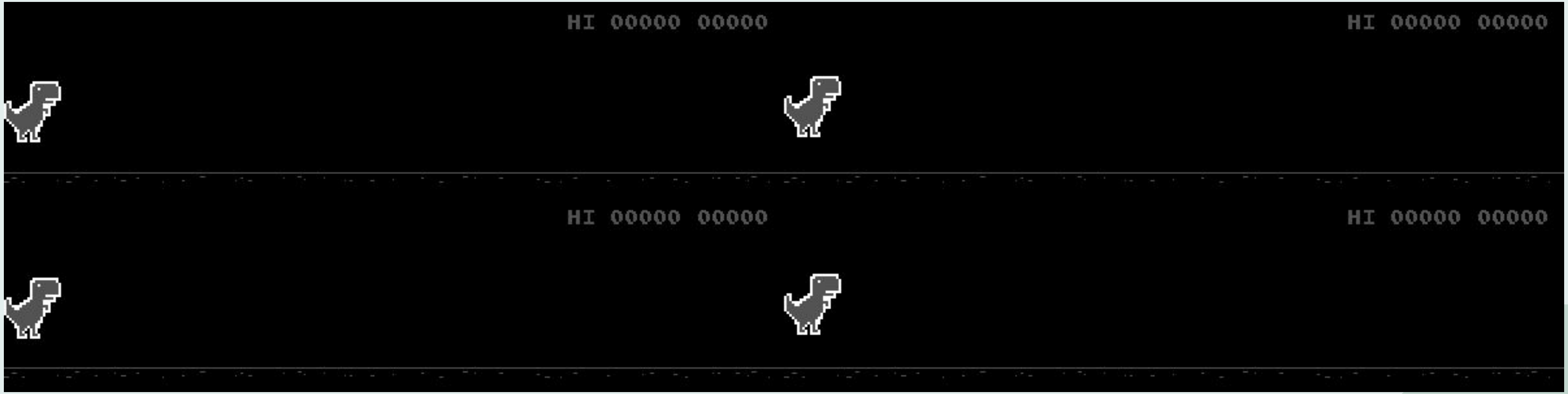
- **Deep Q Learning :**
  Q Value = Quality of Action

1. Init Q Value (= init model)
2. Choose Action (model.predict(state))
3. Perform action
4. Measure reward
5. Update Q Value (+ train model)

Chrome Dino Game

- Input variables: Trees, Birds
- Output variables: Jump, Duck, Do nothing
- Algorithms: CNN, Deep Learning
- Number of iterations: 3500
- The Highest score:1098





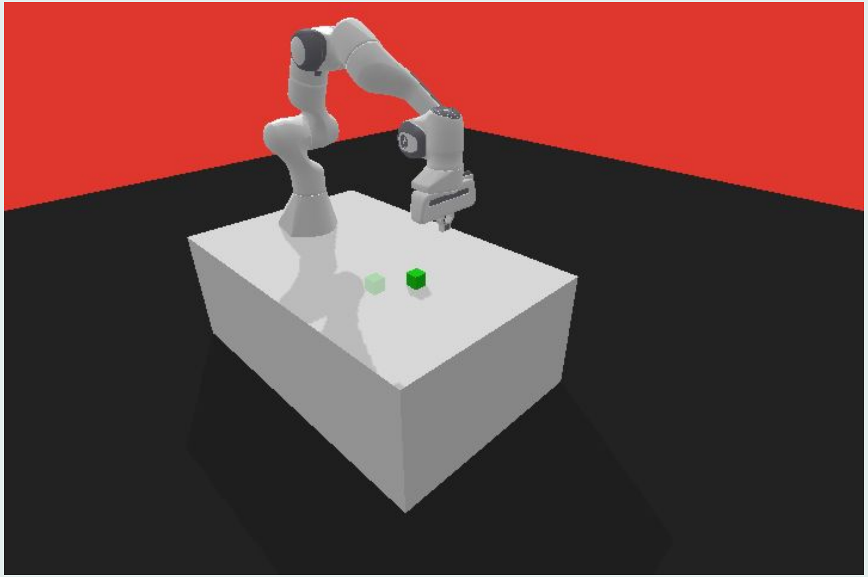
robotic - Panda-gym

Environments :

- Velocity
- Gravity
- friction
- Green bloc position
- Green spot
- …

Action :

- Multiple joint with multiple dimension
- Hook



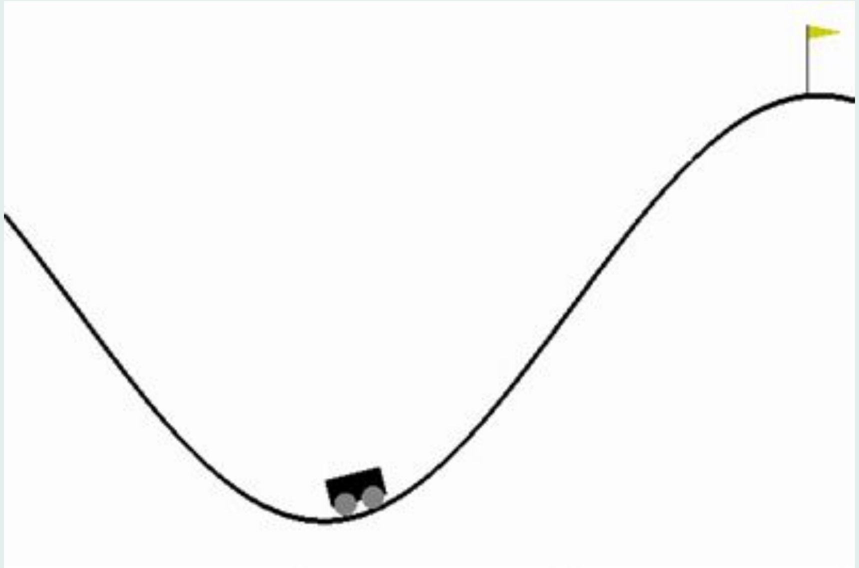
Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research : *https://arxiv.org/pdf/1802.09464.pdf*

Open-source goal-conditioned environments for robotic learning : https://arxiv.org/pdf/2106.13687.pdf

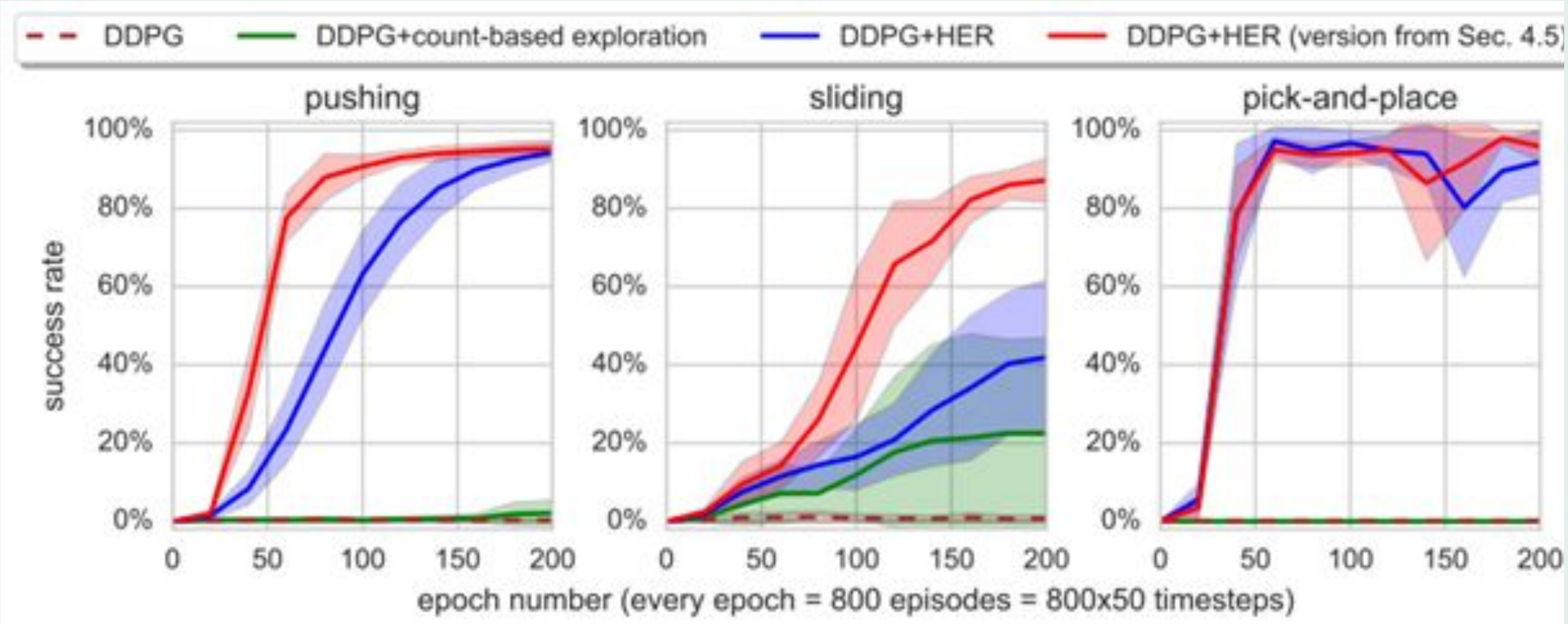
https://github.com/qgallouedec/rl-baselines3-zoo/blob/master/hyperparams/her.yml

https://github.com/qgallouedec/panda-gym from central lyon

robotic - HER



Andrychowicz, Marcin, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. “Hindsight Experience Replay.” *ArXiv.org*, 2017. https://doi.org/10.48550/arXiv.1707.01495.



Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research :
https://arxiv.org/pdf/1802.09464.pdf
Open-source goal-conditioned environments for robotic learning : https://arxiv.org/pdf/2106.13687.pdf
https://github.com/qgallouedec/rl-baselines3-zoo/blob/master/hyperparams/her.yml

Super Mario Bros

- Input variables : do nothing, go right, run to the right,
  jump to the right, run and jump to the right
- Output variables : reward of the action
- Algorithms : PPO and QR-DQN
- Number of steps : 5 millions with each algorithm
- The highest score : ~215