Ma première IA jouant aux jeux vidéos

Henri, Axel, Clément, Iliass

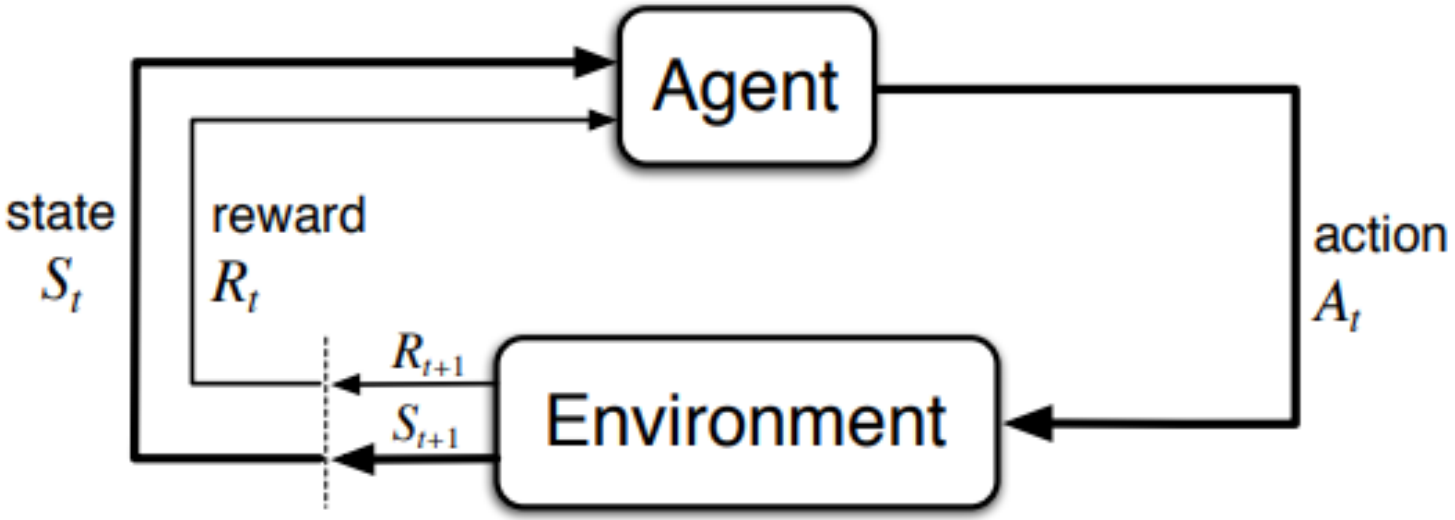
Table des matières

Introduction

- Projet d'IA appliquée aux jeux vidéo
- Entraînement d'IA sur différents jeux (Cartpole, Atari, Mario Bros)
- Utilisation de l'apprentissage par renforcement

Table des matières

Formalisation du problème



Markov Decision Process

Un MDP est un quadruplet (S,A,P,R) caractérisé par $\forall t \in \mathbb{N}$:

1. d'un agent A_t
2. d'un environnement S_t
3. d'un noyau de transition stationnaire $p(s_{t+1})$
4. d'une fonction R de récompense

On définit aussi une politique $\pi(a|s) = P(A_t = a|S_t = s)$

Objectif

$$J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1}]$$

$$V_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{k+t+1} | S_t = s]$$

$$Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{k+t+1} | S_t = s, A_t = a]$$

Équation de Bellman/Équation de Bellman*

$$Q_\pi(s,a) = \sum_{s'} p(s'|s,a)\left(R(s,a,s') + \gamma V_\pi(s')\right)$$

$$Q^\star(s,a) = \sum_{s'} p(s'|s,a).(R(s,a,s') + \gamma \max_{a'} Q^\star(s',a'))$$

Table des matières

Q-learning

1: Initialisation environnement
2: **for** i allant de 1 à M **do**
3: Initialiser l'état s
4: **while** l'agent n'a pas perdu **do**
5: Executer action a qui vérifie $a_t = arg\max_a Q(s_t, a)$
6: Récupération s_{t+1} et r_t
7: Maj environnement
8: Maj de la matrice en appliquant Bellman
9: Maj de s_t avec s_{t+1}
10: **end while**
11: **end for**

Résultats

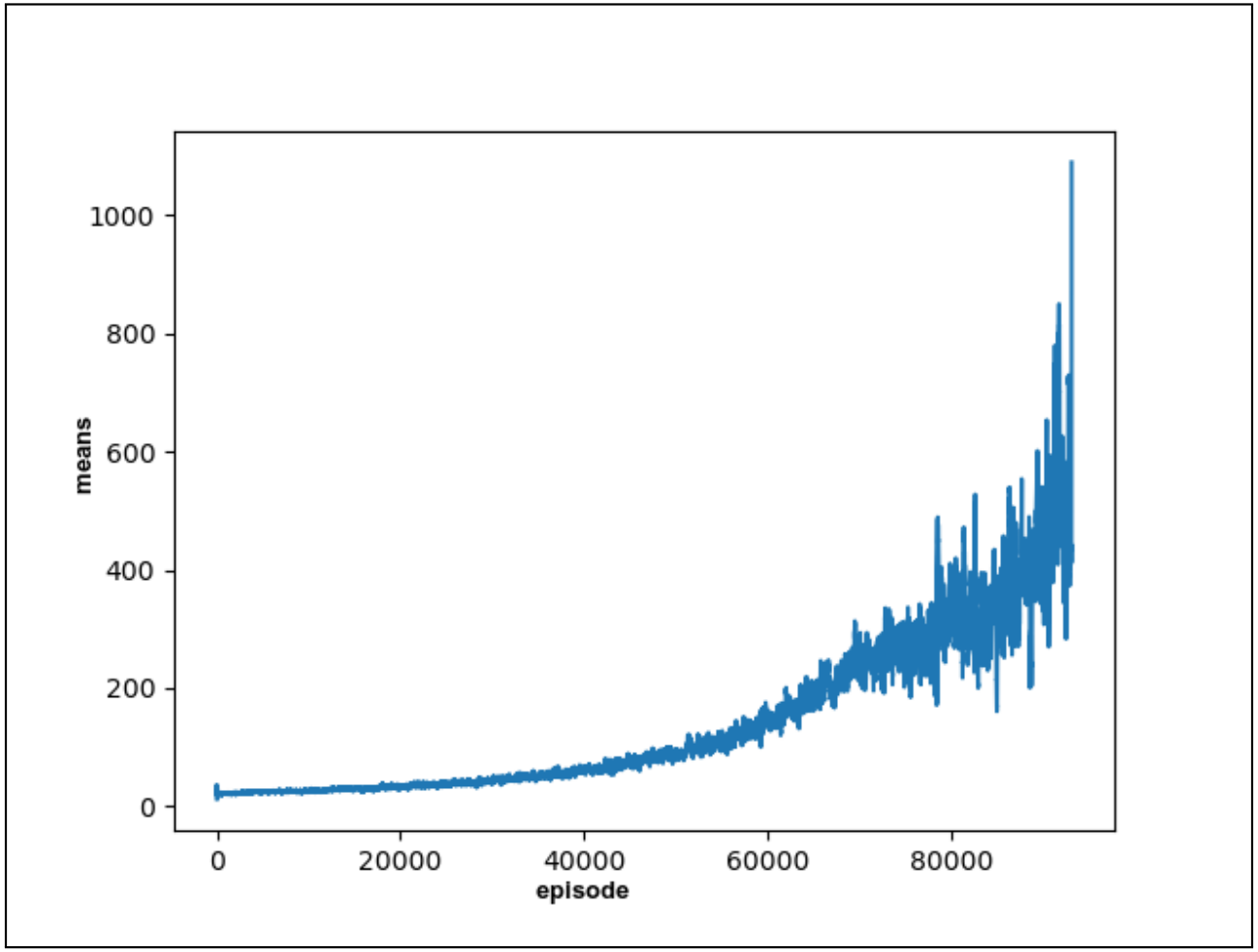


Figure: Rewards en fonction du nombre d'épisodes (Cartpole)

Limites du Q-learning

- Taille de la matrice Q augmente avec la complexité du jeu
- Inefficace pour des environnements complexes
- Incapacité à traiter des environnements RGB

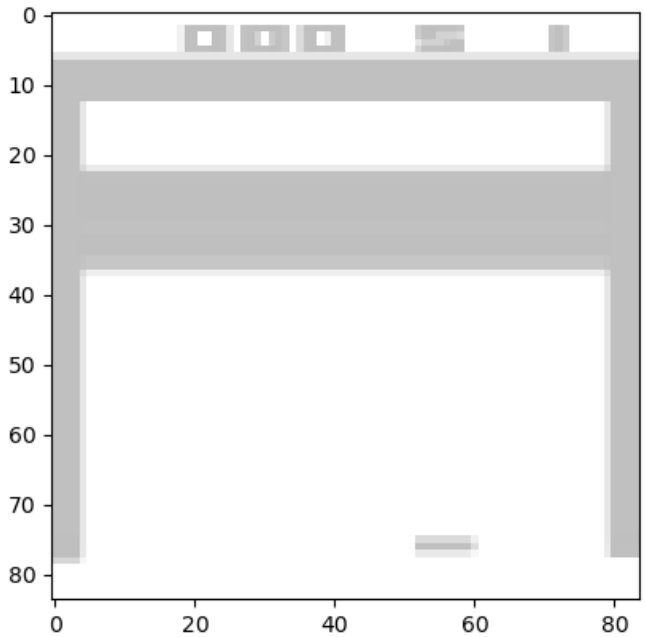
Action Space	`Discrete(2)`
Observation Space	`Box([-4.8000002e+00 -3.4028235e+38 -4.1887903e-01 -3.4028235e+38], [4.8000002e+00 3.4028235e+38 4.1887903e-01 3.4028235e+38], (4,), float32)`
import	`gymnasium.make("CartPole-v1")`

Action Space	Discrete(4)
Observation Space	Box(0, 255, (210, 160, 3), uint8)
Import	`gymnasium.make("ALE/Breakout-v5")`

Table des matières

Deep Q-learning

- Remplacement de la matrice Q par un réseau de neurones
- Analyse des images de l'environnement : besoin de preprocessing

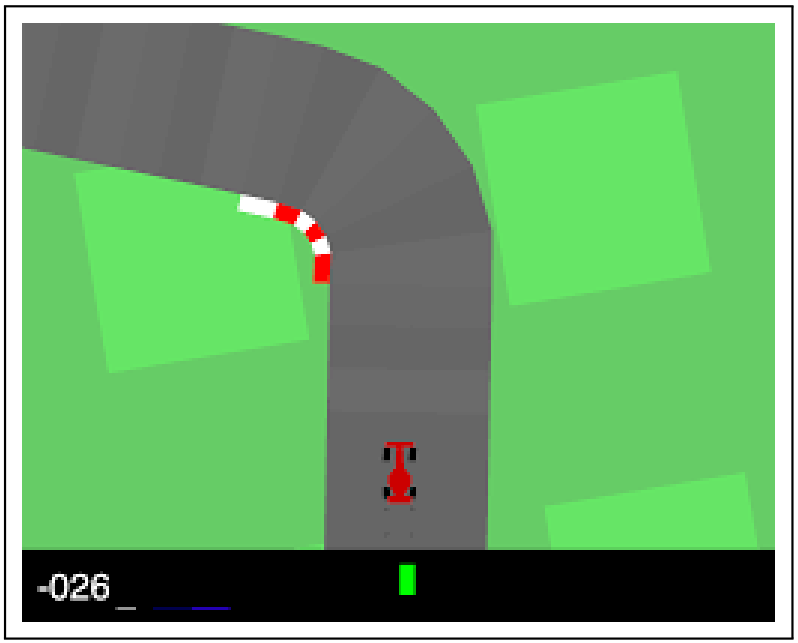


Double Deep Q-learning

Équation de Bellman modifiée

$$Q(s,a) = R(s,a) + \gamma(1 - \mathit{terminated}) \max_{a'} Q'(s',a')$$

Résultats



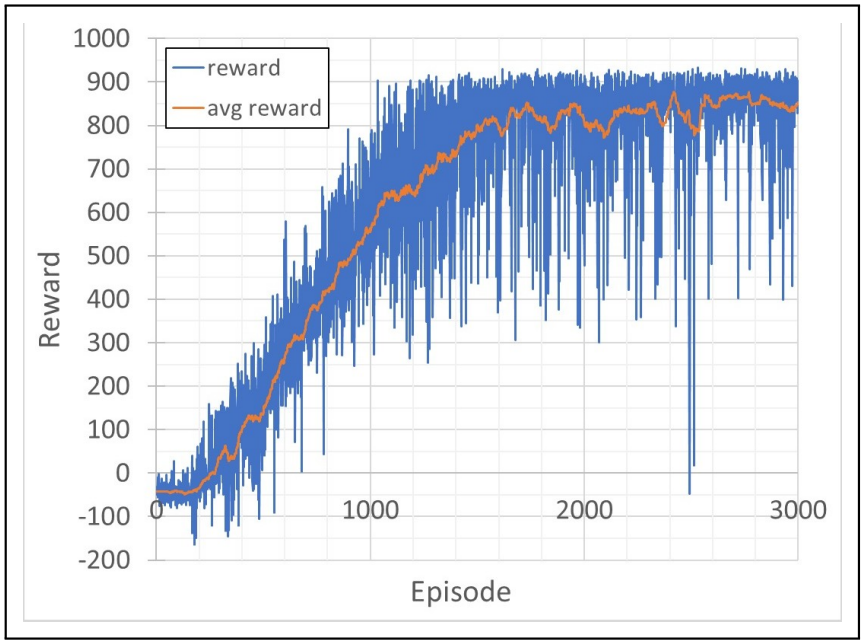


Figure: Rewards de Car-racing

Résultats

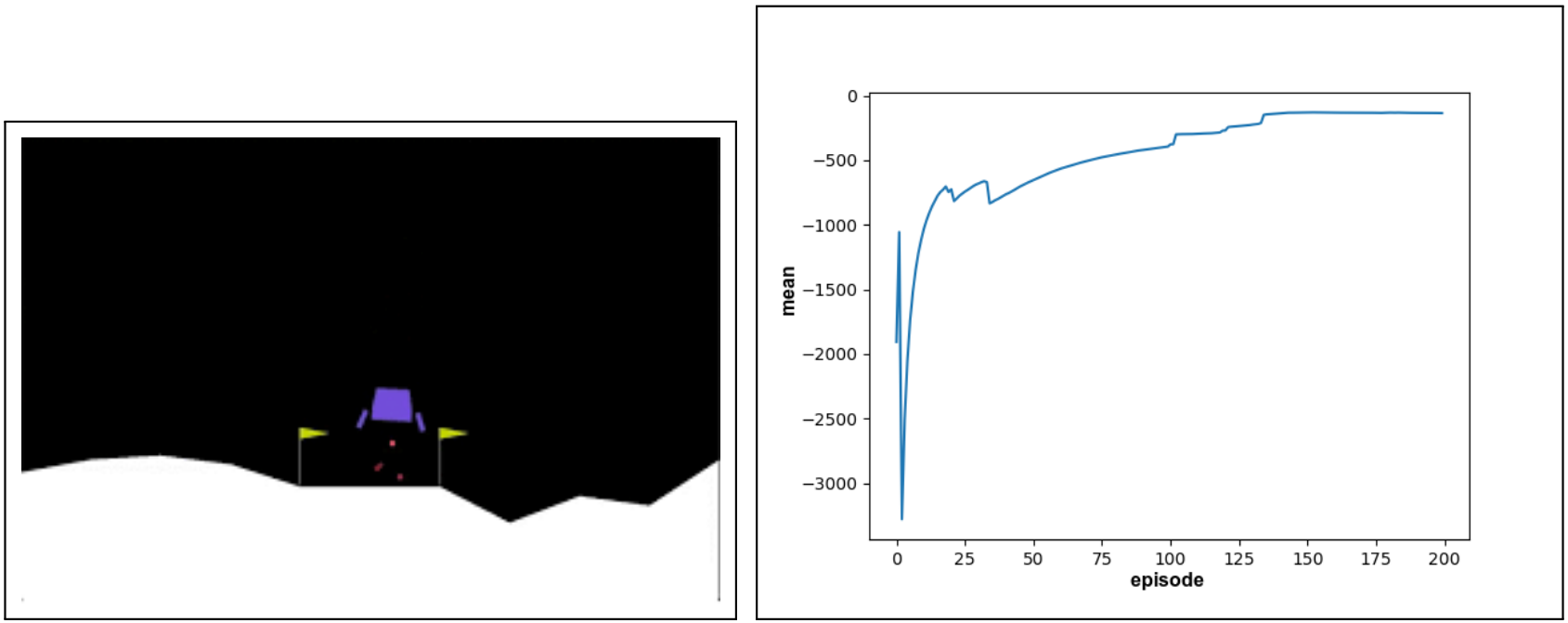


Figure: Rewards de Lunar Lander

Résultats

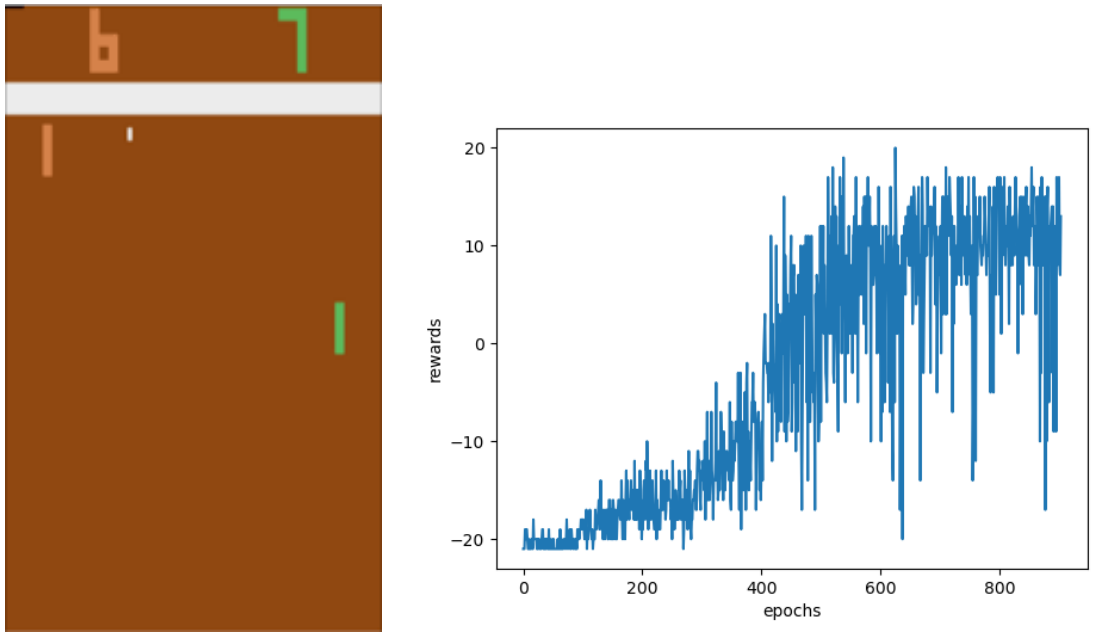


Figure: Rewards de Pong

Limites du Deep Q-learning

- *Value-based*
- Réseau lourd si champ d'action large

Table des matières

REINFORCE

Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right] = \mathbb{E}_{\pi \sim S_t} \left[\sum_a q_\pi(S_t, a) \nabla_\theta \pi_\theta(a|S_t) \right]$$

Policy Gradient Theorem

Corollaire

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi \sim S_t} \left[G_t \nabla_\theta \log \pi_\theta(A_t | S_t) \right]$$

avec $G_t = \sum_{k=0}^{\infty} \gamma^k r_{k+t+1}$

Preuve au tableau (si temps).

Algorithme

$$\theta \leftarrow \theta + \alpha G_t \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Résultats

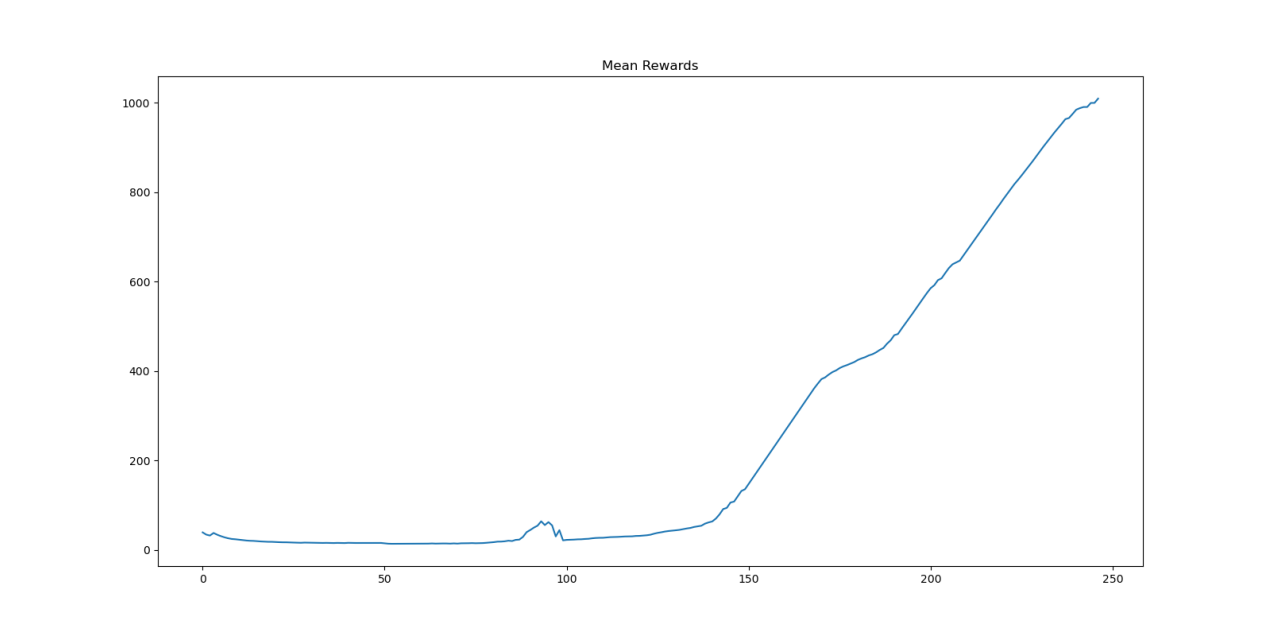


Figure: Rewards moyennes pour Cartpole

Limites de REINFORCE

- **Exploration** : minimum local
- **Variance** : variances des gradients élevée

Solution :

$$H(\pi) = -\sum_{a} \pi(a|s) \log \pi(a|s)$$

REINFORCE with baseline

Advantage function

$$adv(s, a) = q_\pi(s, a) - b(s),$$

Algorithme

$$\theta \leftarrow \theta + \alpha(G_t - b(t))\nabla_\theta \log \pi_\theta(a_t|s_t)$$

Table des matières

Advantage Actor-Critic (A2C)

- Combinaison des méthodes value-based et policy-based
- Réduction de la variance et stabilisation de l'apprentissage

Fonction Advantage

$$adv(s, a) = q_\pi(s, a) - v_\pi(s)$$

Le fait de soustraire v_π ne change pas le gradient ! (preuve tableau si temps)

A2C

Mise à jour de la politique

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t) adv(s_t, a_t)$$

Mise à jour de la critique

$$\phi \leftarrow \phi - \beta \nabla_\phi \left(v_\pi(s_t) - G_t\right)^2$$

Table des matières

Asynchronous Advantage Actor-Critic (A3C)

- Exploitation des résultats de plusieurs agents
- Amélioration du réseau global

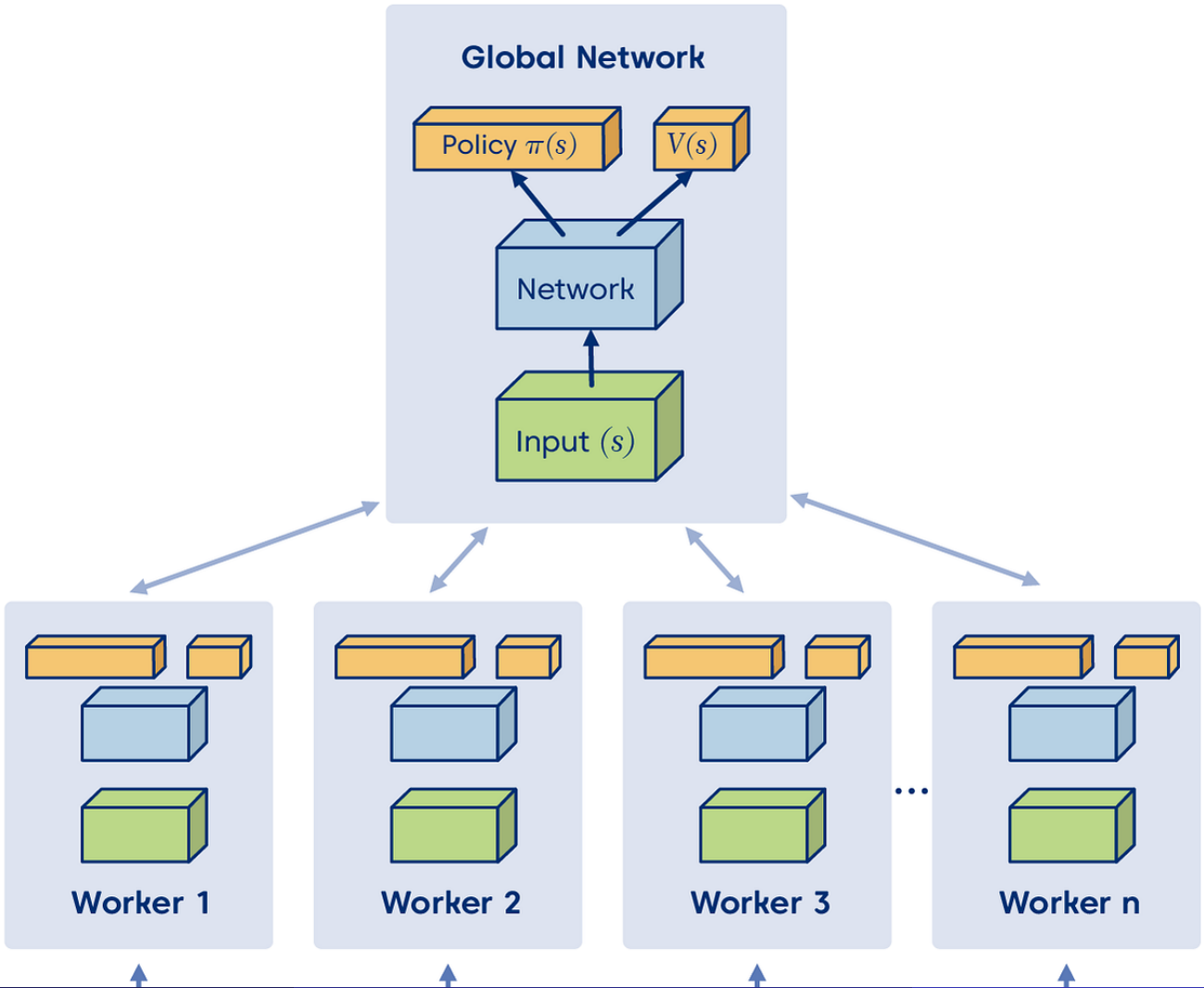


Table des matières

Proximal Policy Optimization (PPO)

Ratio function

$$r_t(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$$

Loss clip

$$L_{CLIP} = min(\mathbb{E}_t(\min(r_t(\theta)A_t, 1-\epsilon, 1+\epsilon)A_t)$$

Loss total

$$L_{TOTAL} = \mathbb{E}_t[L_{CLIP}(\theta) - c_1 L_{VF}(\theta) + c_2 S\pi_\theta(s_t)]$$

Table des matières

Conclusion

- Chaque méthode d'apprentissage par renforcement a ses propres forces et faiblesses.
- Importance de choisir l'algorithme adapté au problème spécifique.
- Innovations rapides ouvrent la voie à des applications de plus en plus complexes.
- Les techniques étudiées posent les bases pour des systèmes intelligents autonomes dans des environnements dynamiques.

Table des matières

Bibliographie

- Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction.
- Mnih, V., et al. (2013). Playing Atari with Deep Reinforcement Learning.
- Lapan, M. (2018). Deep Reinforcement Learning Hands-On.