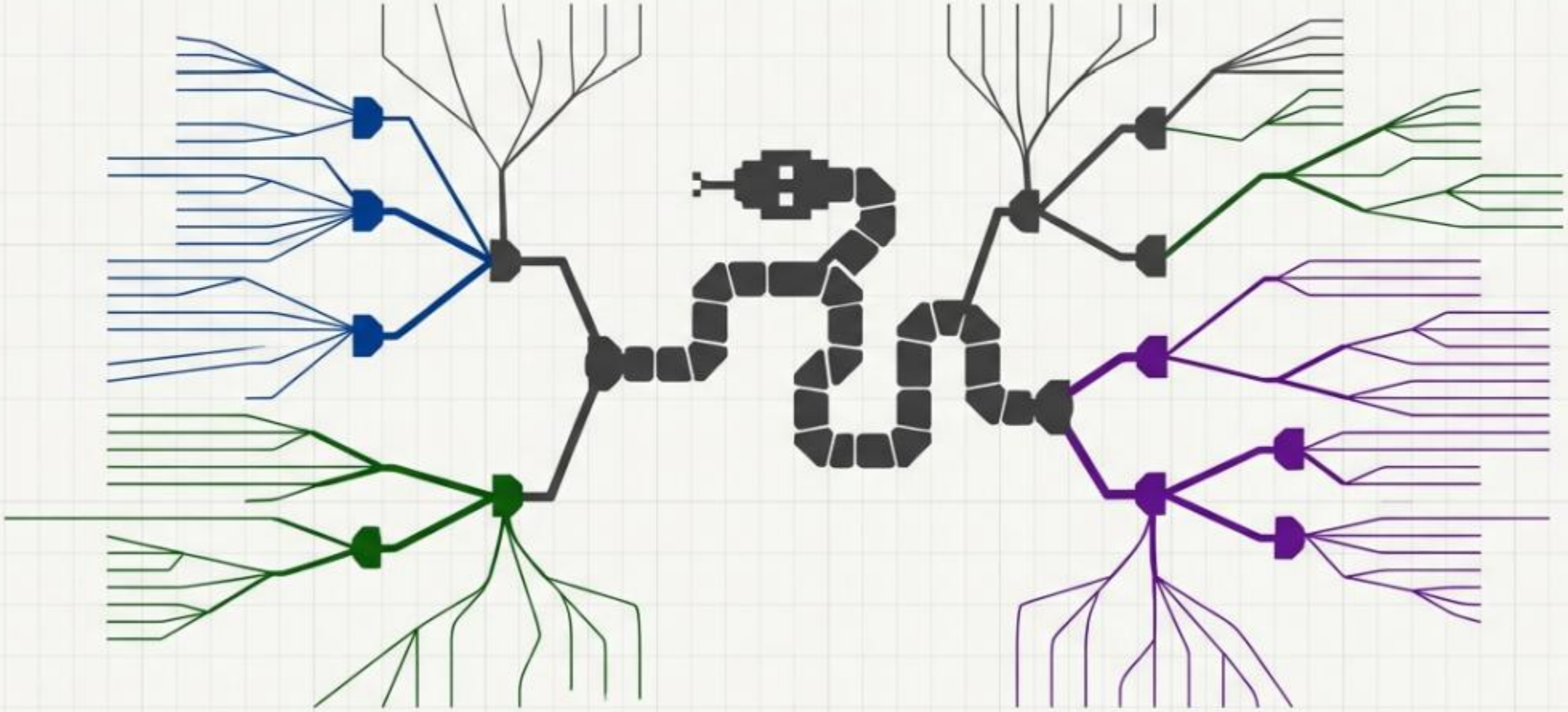
Snake RL : Le Duel des Algorithmes

Une comparaison de PPO, DQN et SAC pour maîtriser le jeu classique.



Notre Mission : Former l'Agent Snake Ultime

1. Objectif Principal : Implémenter et comparer rigoureusement trois algorithmes de Deep Reinforcement Learning (PPO, DQN, SAC) sur une tâche unique et contrôlée.

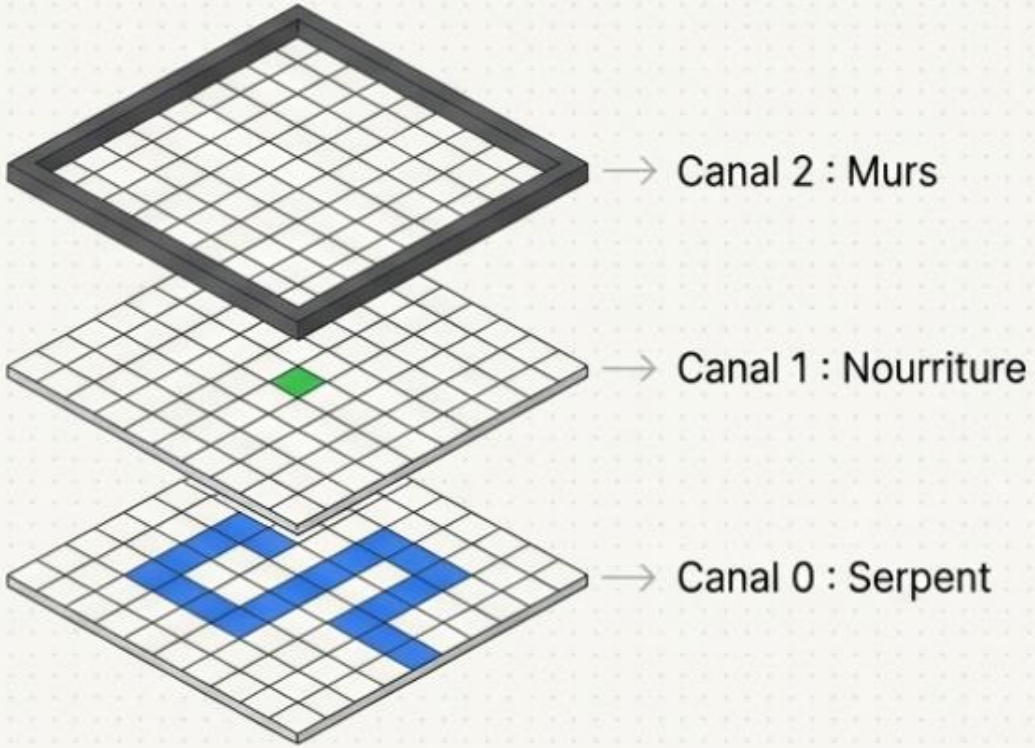
2. Le Défi : Déterminer l'approche la plus performante pour apprendre à une IA à exceller au jeu Snake.

3. La Question : Comprendre les forces et faiblesses réelles de chaque algorithme dans un environnement à actions discrètes bien connu.

L'Arène : L'Environnement de Jeu

L'Espace d'Observation (Ce que l'agent "voit")

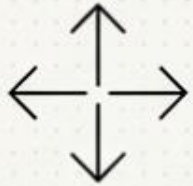
L'agent perçoit le jeu via une grille 3D (6x6x3)



L'Espace d'Action et les Récompenses (Comment l'agent "apprend")

Actions

4 actions discrètes possibles : Haut, Bas, Gauche, Droite.



Système de Récompenses

Récompenses Positives :

- ⭐ +10 + (longueur × 0.5) : Manger la nourriture.
- 📈 +0.3 × (amélioration distance) : Se rapprocher.
- 🏆 +100 : Victoire (grille remplie).
- ✨ +0.05 : Bonus pour espace libre disponible.

Pénalités :

- 💥 -10 : Collision (mur ou auto-collision).
- 📉 -0.4 × (dégradation distance) : S'éloigner.
- ⏳ -0.01 : Pénalité par 'step'.
- 🔄 -0.3 × (nombre de boucles) : Détection de boucles répétitives.
- 🍴 Pénalité de faim croissante (quadratique).

Les Contenders : Trois Stratégies en Compétition

PPO (Le Stratège Robuste)

DQN (Le Vétéran Efficace)

SAC (Le Moderniste Explorateur)

Contender 1 : DQN (Le Vétéran Efficace)

Type d'algorithme: Off-policy, Value-based.

Principe Général

Utilise un "Replay Buffer" pour apprendre des expériences passées, qu'elles soient bonnes ou mauvaises. Il capitalise sur chaque interaction pour une grande efficacité des données.

Avantages (pour Snake)

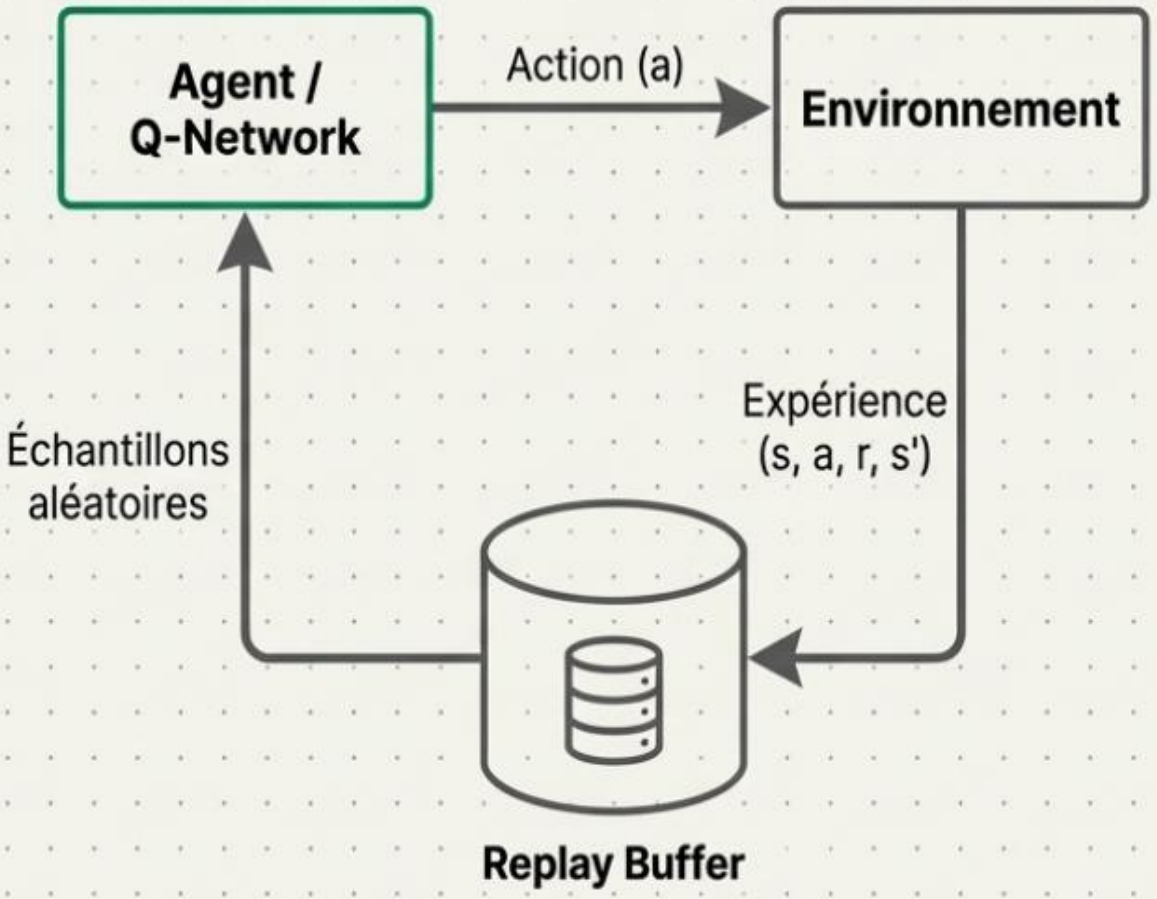
- **Grande efficacité des données (Sample Efficiency) :** Réutilise les expériences, apprenant potentiellement plus vite avec moins d'interactions.

Limites (pour Snake)

- **Instabilité potentielle**: L'apprentissage à partir de politiques "périmées" du buffer peut introduire une variance élevée et des décisions parfois inexplicables pour le serpent.

Formule Clé

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_a Q(s',a') - Q(s,a)]$$



Contender 2 : PPO (La Stratégie Robuste)

Type d'algorithme: On-policy, Actor-Critic.

Principe Général

Apprend directement de l'expérience en cours. Il effectue de petites mises à jour prudentes pour éviter les changements drastiques, garantissant un apprentissage stable.

Avantages (pour Snake)

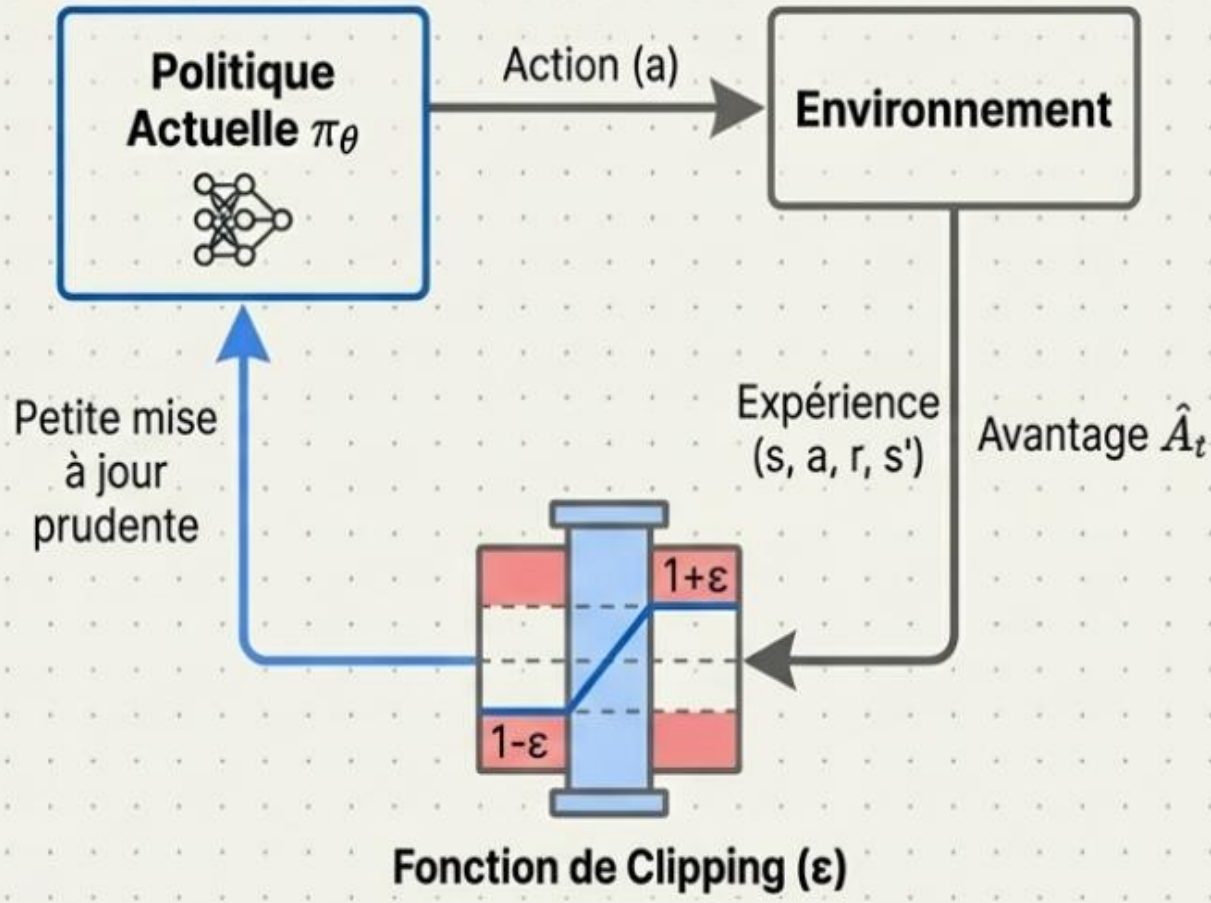
- **Stabilité et performance constante:** Les mises à jour 'clippées' évitent les effondrements de performance, menant à un serpent au comportement fiable et prévisible.

Limites (pour Snake)

- **Moins efficace en données (Sample Inefficient) :** Jette les expériences après chaque mise à jour, ce qui peut nécessiter plus de temps de jeu pour atteindre des performances de pointe.

Formule Clé

$$\mathcal{L}^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)]$$



Contender 3 : SAC (Le Moderniste Explorateur)

ⓘ **Type d'algorithme:** Off-policy, Actor-Critic.

Principe Général

Cherche à maximiser non seulement la récompense, mais aussi son 'entropie". Cela l'encourage à explorer un maximum d'actions différentes pour ne pas tomber dans des stratégies simplistes.

Avantages (pour Snake)

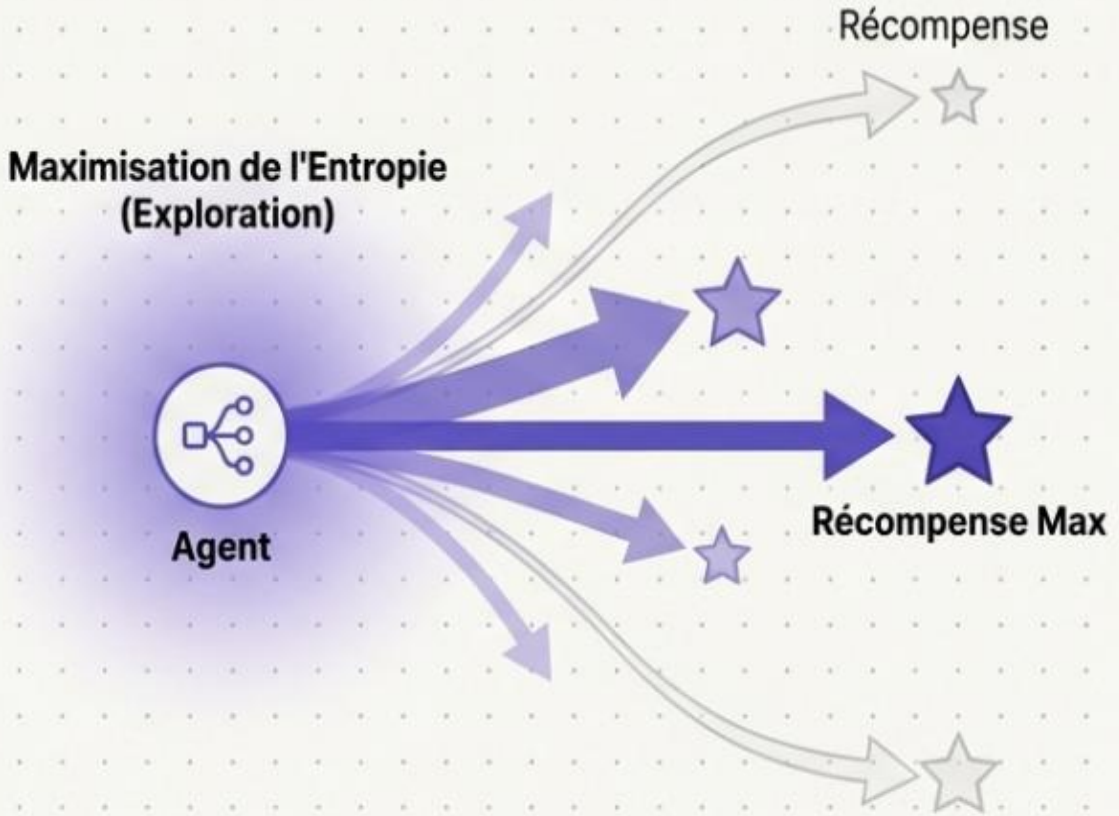
- **Exploration puissante**: Théoriquement très efficace pour éviter les optimums locaux, comme tourner en rond autour de la nourriture.

Limites (pour Snake)

- **Moins adapté aux actions discrètes:** Conçu pour les espaces d'actions continus. Son adaptation pour Snake est complexe, plus lente à entraîner, et tend à produire des comportements sous-optimaux.

Formule Clé

$$J(\pi) = \sum_t E[r(s_t,a_t) + \alpha H(\pi(\cdot|s_t))]$$



L'Arsenal Technologique

RL Core

Framework d'environnement : Gymnasium

Implémentations RL : Stable-Baselines3

Computation & Backend

Backend Deep Learning : PyTorch

Calcul Numérique : NumPy

Visualization & Monitoring

Rendu et Visualisation : Pygame

Monitoring de l'entraînement : TensorBoard

Les Critères de Jugement : Comment Déclarer un Vainqueur ?

Score Moyen

La performance brute. Quelle est la longueur maximale du serpent atteinte en moyenne ?

Stabilité

La variance des performances. L'agent est-il fiable ou ses résultats sont-ils erratiques ?

Efficacité (Sample Efficiency)

La vitesse d'apprentissage. Quel agent atteint une bonne performance avec le moins d'expérience ?

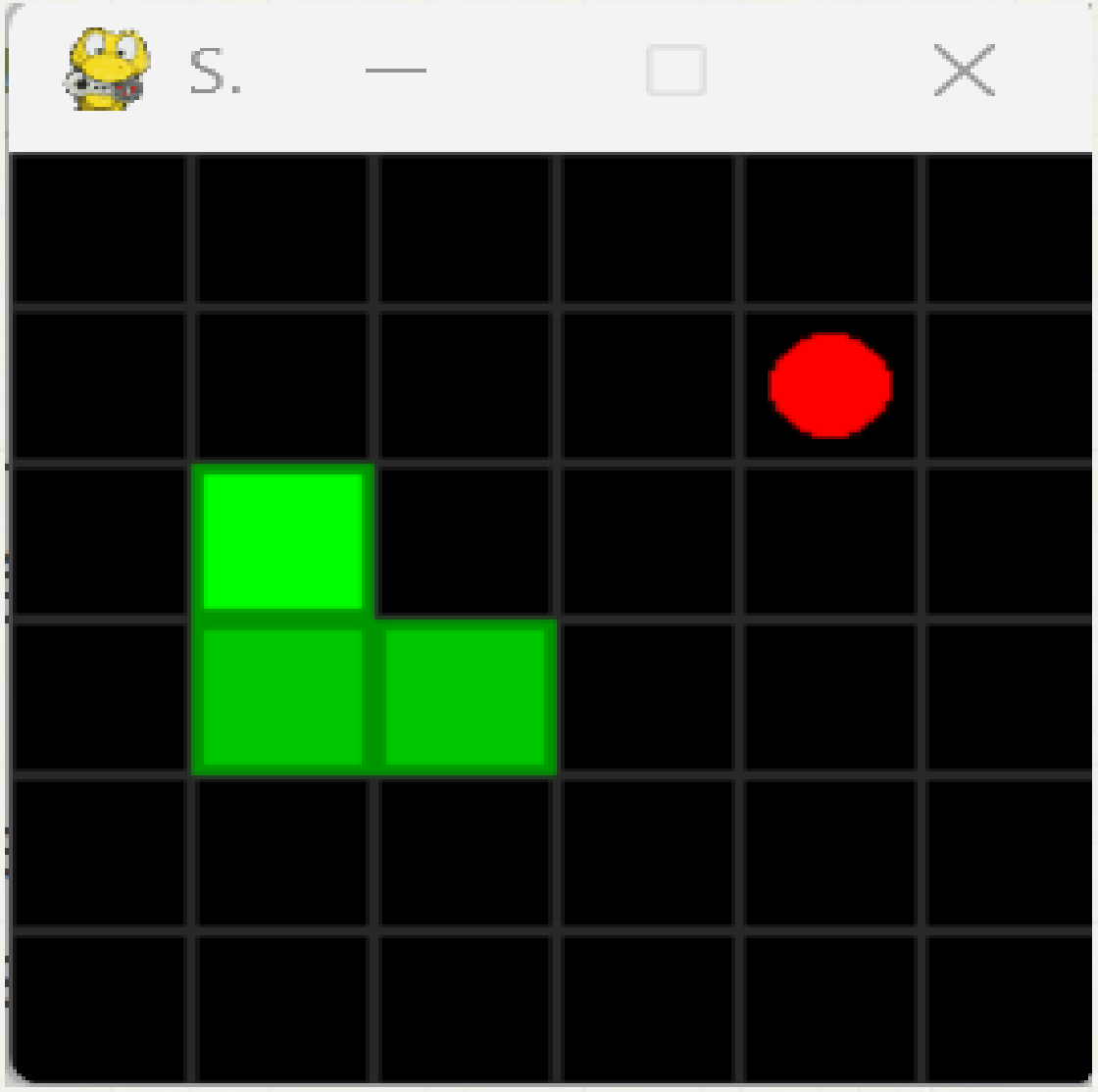
Comportement Qualitatif

L'intelligence de la stratégie. L'agent développe-t-il des stratégies de survie élégantes ou des mouvements chaotiques ?

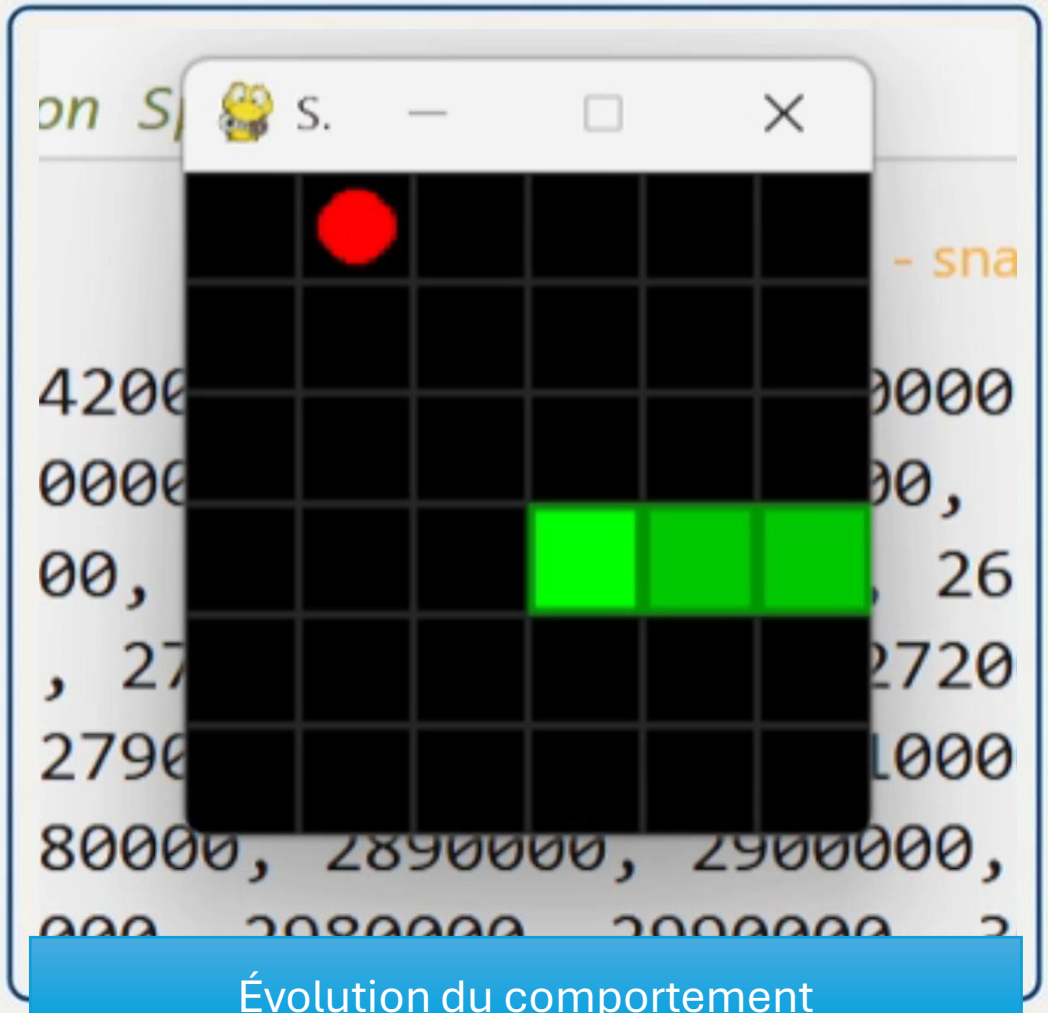
Le Camp d’Entraînement : Processus et Outils

Processus d’Entraînement

- **Durée** : Chaque agent a été entraîné sur un total de **3 millions de "steps"**.
- **Temps estimé** : Environ 1 à 2 heures par algorithme sur un hardware standard.
- **Suivi** : Les courbes d'apprentissage ont été suivies en temps réel via TensorBoard.



Analyse du Champion : La Stratégie Robuste de PPO



Évolution du comportement
(pas d'entraînement espacés de 10 000)

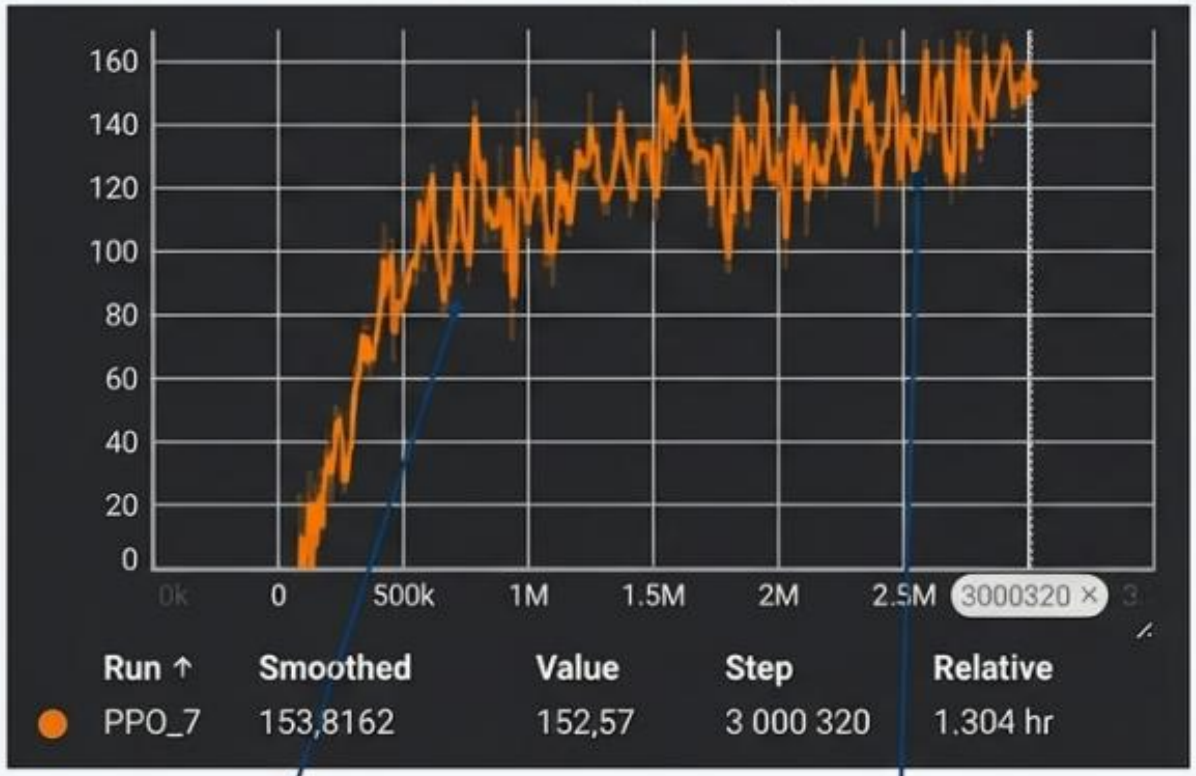
Analyse du Comportement

- Stratégie Dominante : Très direct et efficace pour trouver la nourriture tout en minimisant les risques.
- Comportements Émergents : Développe des stratégies de survie intelligentes, comme longer les murs pour maximiser l'espace disponible et éviter de se coincer.
- Conclusion : L'approche 'on-policy' mène à un apprentissage stable et prévisible, résultant en une politique d'action extrêmement fiable.

L'agent atteint une performance de pointe avec une stabilité remarquable

Les métriques clés sur 3 millions de pas d'entraînement (environ 1.3 heure) démontrent une convergence rapide vers un état expert.

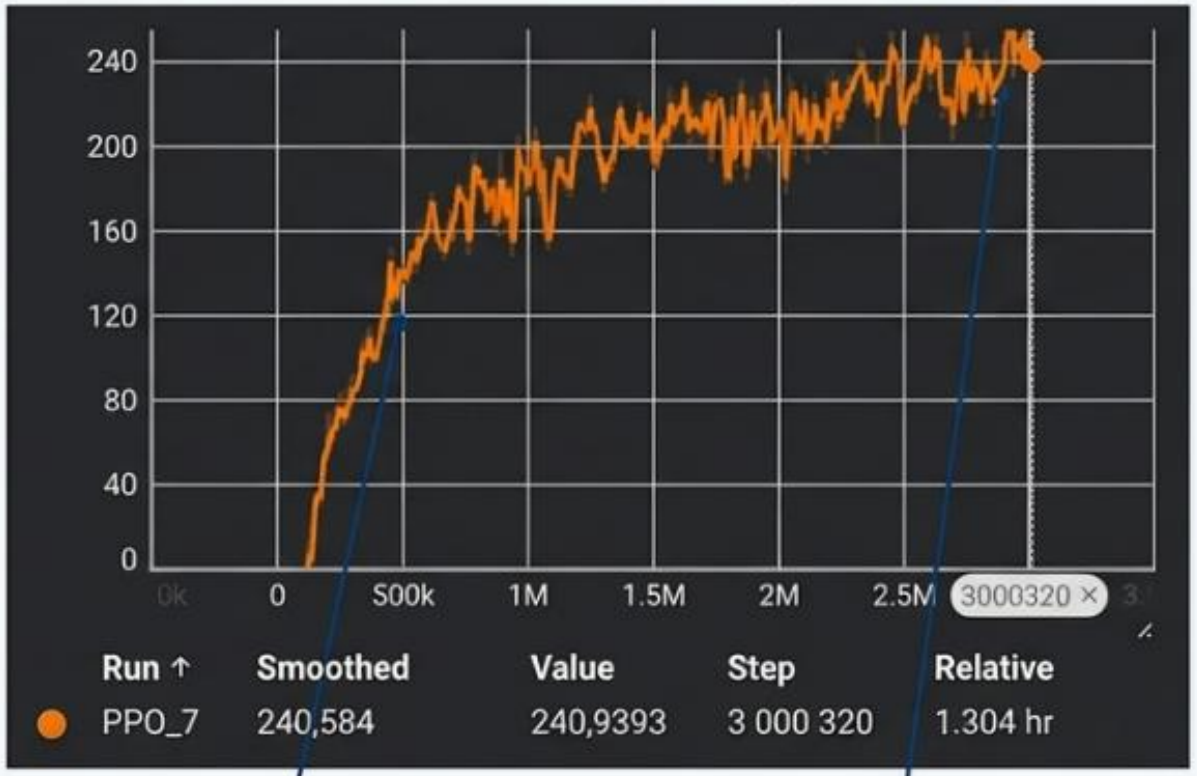
DURÉE DE VIE MOYENNE PAR ÉPISODE (`rollout/ep_len_mean`)



Apprentissage rapide et continu dans les 1.5M premiers pas.

Plateau stable à une durée de vie élevée, indiquant une stratégie de survie maîtrisée. Valeur finale lissée : **152,57**.

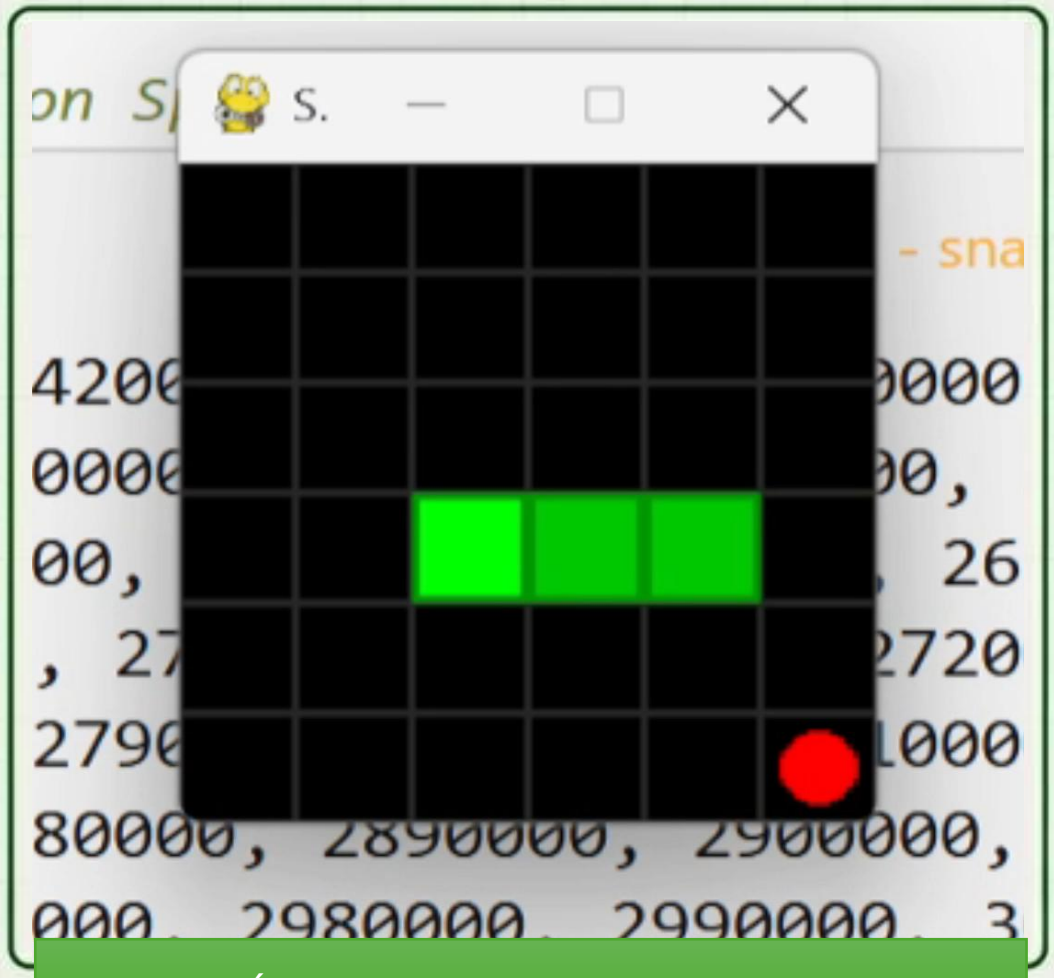
RÉCOMPENSE MOYENNE PAR ÉPISODE (`rollout/ep_rew_mean`)



Corrélation directe entre la durée de vie et l'accumulation de récompenses.

Convergence vers un score maximal et constant, preuve d'une politique optimale. Valeur finale lissée : **240,94**.

Analyse du Challenger : L'Instabilité de DQN

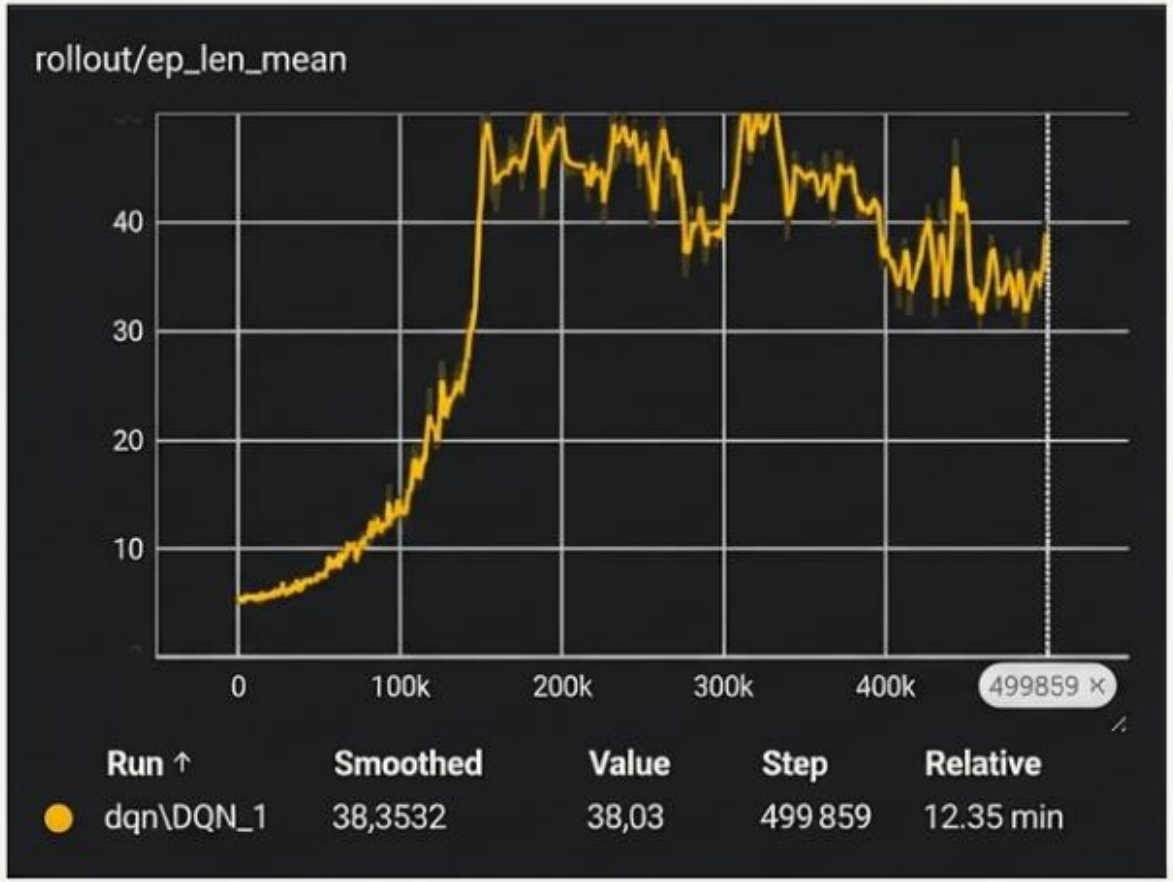


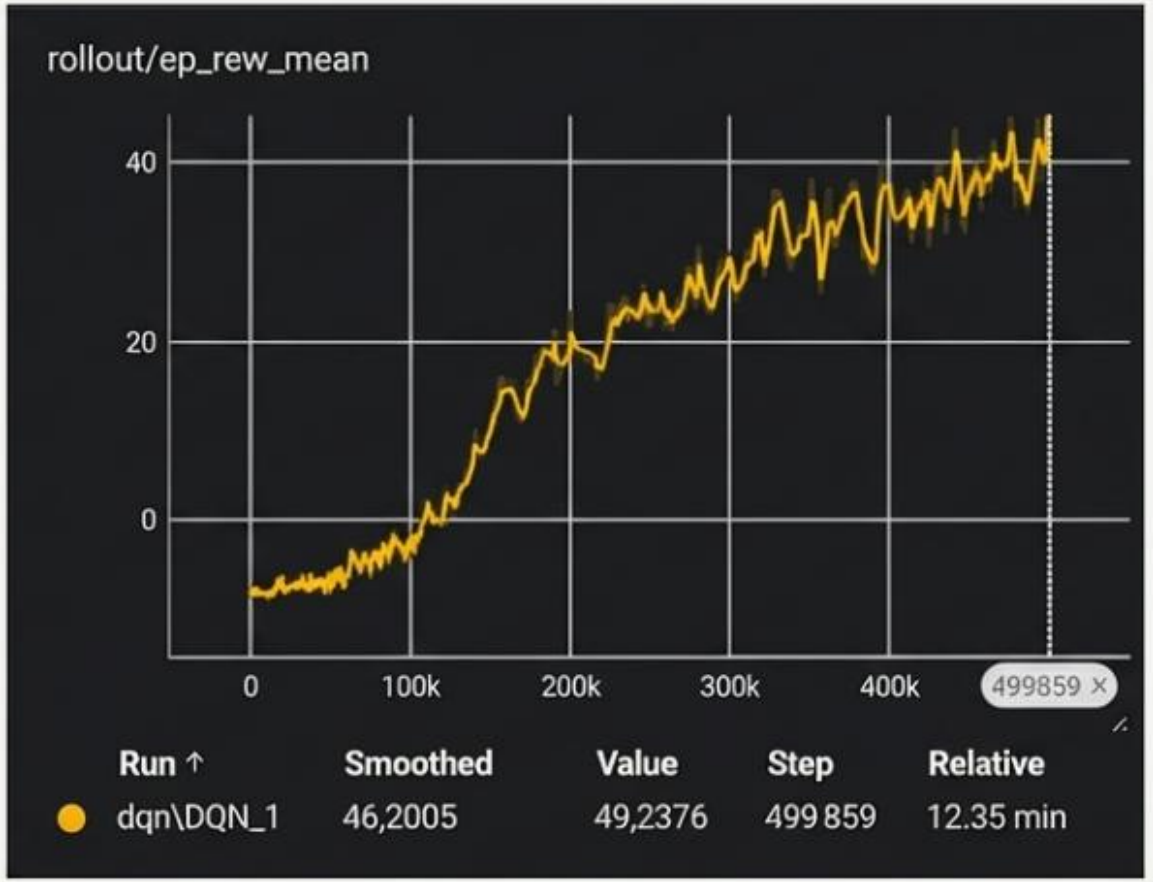
Évolution du comportement
(pas d'entraînement espacés de 10 000)

Analyse du Comportement

- **Performance** : A lutté pour converger vers une stratégie efficace, affichant les scores moyens les plus bas.
- **Point Faible** : Le Replay Buffer, bien que puissant pour l'efficacité des données, peut introduire des politiques basées sur des expériences 'périmées', menant à des instabilités et des décisions parfois inexplicables.
- Conclusion : Un compétiteur puissant mais moins constant, dont la performance peut fluctuer de manière significative.

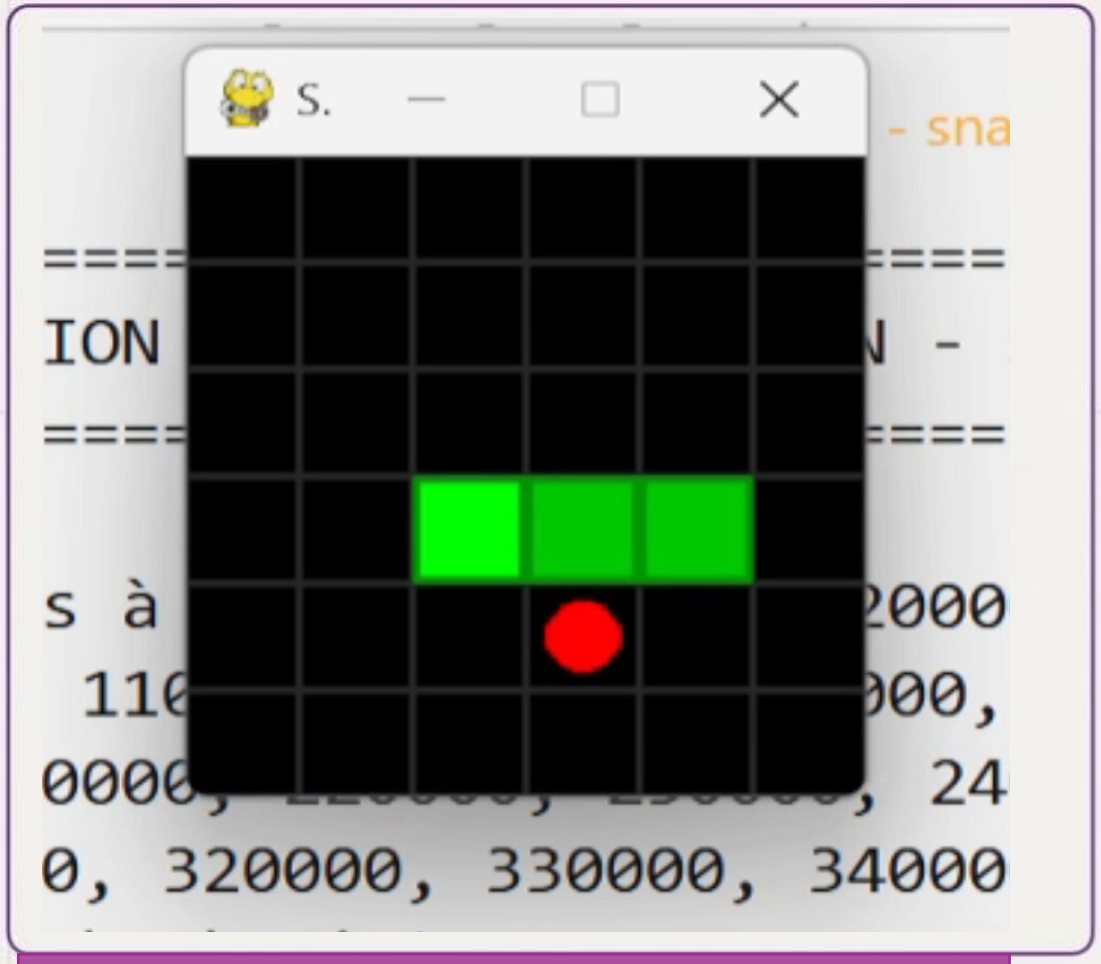
Les Pièces à Conviction : Les Données de Performance Brutes





Données brutes d'une session d'entraînement de DQN sur ~500,000 étapes. Run : `dqn\DQN_1`.

L'Épreuve de l'Outsider : Le Défi de SAC

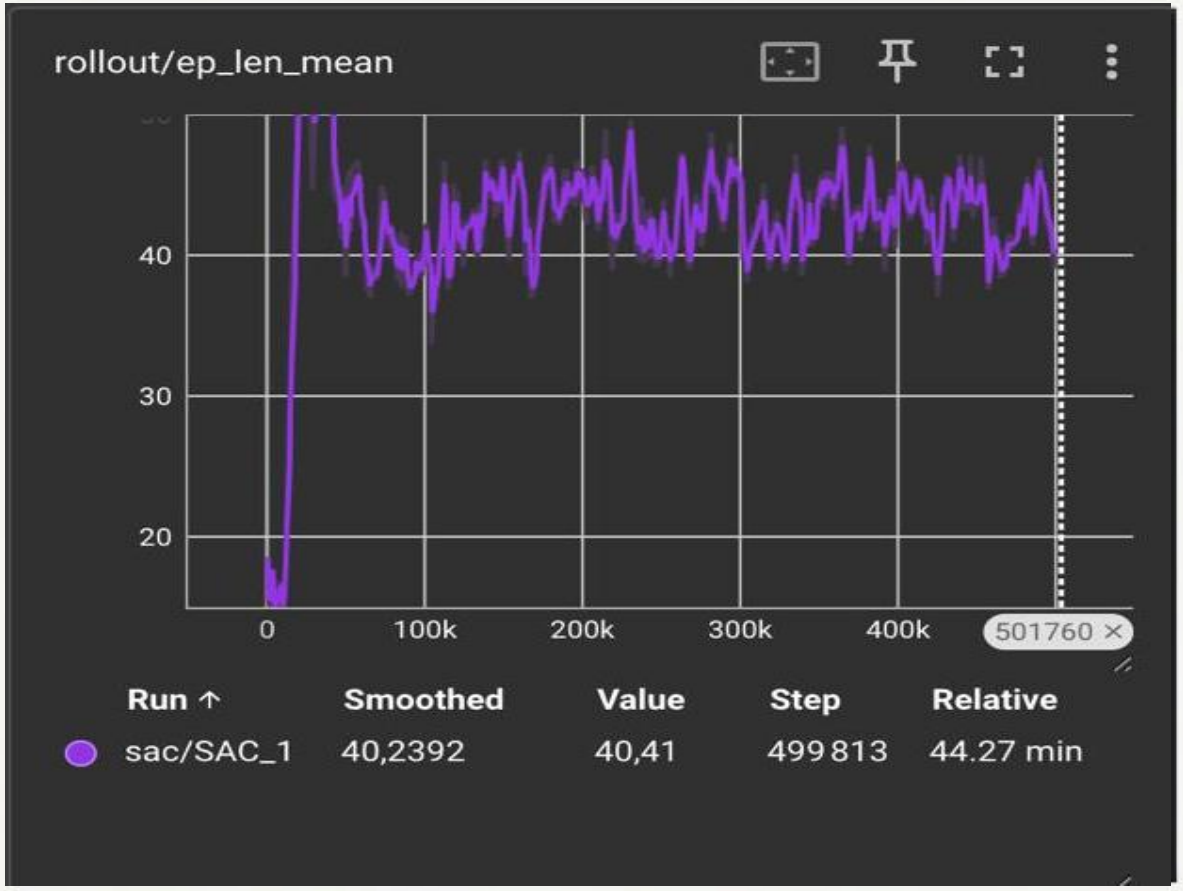


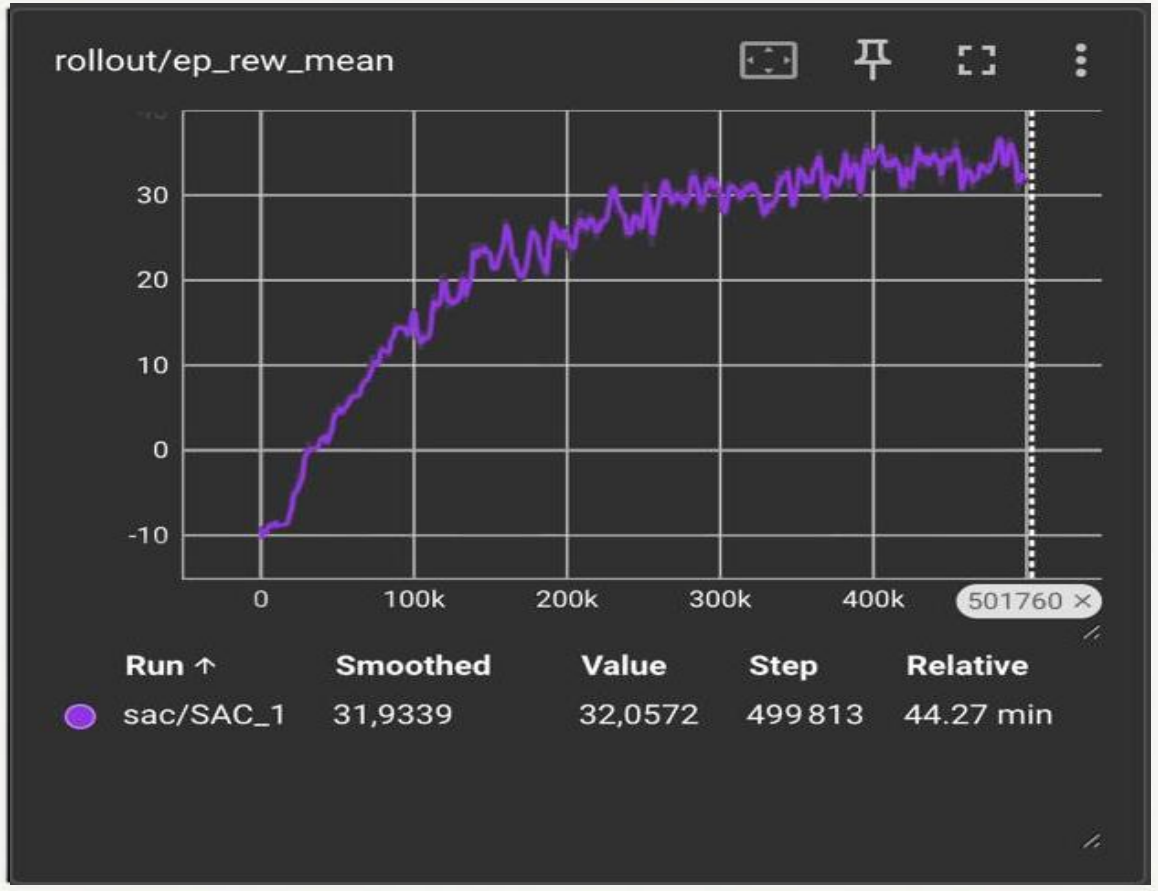
Évolution du comportement
(pas d'entraînement espacés de 10 000)

Analyse du Comportement

- **Performance** : Capable d'atteindre des scores élevés, mais avec un temps d'entrainement nettement plus long que PPO.
- **Comportement Observé** : Tendance marquée à tomber dans des 'boucles locales', où l'agent répète une séquence d'actions sans progresser.
- **L'Explication Technique** : SAC est un algorithme optimisé pour les espaces d'actions **continus**. Son adaptation aux actions discrètes et son architecture intrinsèquement plus complexe (6 réseaux de neurones) le rendent plus lent et moins adapté à ce problème spécifique.

Les Pièces à Conviction : Les Données de Performance Brutes





Données brutes d'une session d'entraînement du SAC sur ~500,000 étapes.

Le Verdict Numérique : PPO Domine le Duel, SAC Devient Le Nouveau Challenger

Performance en Évaluation Finale (PPO vs. DQN vs. SAC)			
Métrique	**PPO (Le Champion)**	**DQN (Le Challenger)**	**SAC (Le Potentiel)**
Score Moyen	**18.00 ± 1.67**	9.80 ± 4.66	10.20 ± 2.54
Score Max	**21**	15	18
Score Min	**16**	1	5
Récompense Moyenne	**269.93 ± 35.05**	129.89 ± 69.45	203.64 ± 29.82
Steps moyen	182.8	73.2	114.7

Le Verdict : PPO S'impose par sa Stabilité

Critère	PPO	SAC	DQN
Score Moyen	★★★	★★	★
Stabilité	★★★	★	★
Temps d'Entraînement	Rapide	Lent	Rapide
Comportement Appris	Robuste & Efficace	Suboptimal	Variable

Conclusion Clé : PPO démontre le meilleur équilibre global entre performance de score, stabilité et comportement fiable, le désignant comme le vainqueur pour cette tâche spécifique.

Dans les Coulisses : Défis Rencontrés et Solutions

Défi 1 : Les Boucles Locales

Problème

Initialement, les agents (surtout SAC) tournaient souvent sur eux-mêmes ou autour de la nourriture sans la prendre.

Analyse

Un symptôme classique d'un mauvais équilibre entre exploration et exploitation, ou d'une récompense par 'step' mal calibrée.

Solution

Affinage du système de récompenses, notamment la pénalité par 'step', pour décourager l'inaction.

Défi 2 : La Lenteur Excessive de SAC

Problème

L'architecture complexe de SAC (6 réseaux de neurones) rendait l'entraînement prohibitif.

Analyse

Le cycle d'apprentissage par défaut n'était pas optimisé pour un environnement simple et rapide comme Snake.

Solution

Optimisation ciblée des hyperparamètres, notamment `train_freq=4` et `batch_size=128`, pour accélérer significativement le cycle d'apprentissage sans sacrifier la stabilité.

Les Leçons du Duel : Ce que nous avons appris

1

La Stabilité Prime sur la Complexité

Pour les problèmes à actions discrètes comme Snake, la convergence stable d'un algorithme on-policy comme PPO a surpassé la flexibilité des approches off-policy plus complexes.

2

L'Environnement est Roi

La conception minutieuse du système de récompenses est absolument critique. De légers ajustements ont un impact profond sur le comportement final de l'agent.

3

Il n'y a pas de 'Meilleur' Algorithme Universel

Le choix de l'algorithme doit être guidé par la nature du problème (discret vs. continu, complexité, etc.). Ce projet est une démonstration pratique de ce principe fondamental.

Le Duel Continue : Prochaines Étapes

- [ ] Nouveaux Contenders : Ajouter et comparer l'algorithme A2C et des modèles récurrents (PPO avec LSTM) pour la gestion de la mémoire.
- [ ] Arènes Plus Complexes : Tester les agents sur des grilles plus grandes (15x15, 20x20) et introduire des obstacles fixes ou mobiles.
- [ ] Apprentissage Progressif : Implémenter le 'Curriculum Learning', en commençant par un environnement simple et en augmentant progressivement la difficulté.
- [ ] Le Défi Ultime : Explorer le Multi-Agent RL avec plusieurs serpents en compétition dans la même arène.