Implémentation d’un DQN pour Pong

Table des matières

[Objectif 1](#_Toc182762128)

[Description de l'Implémentation 1](#_Toc182762129)

[1. Prétraitement des Observations 1](#_Toc182762130)

[2. Modèle Dueling DQN 1](#_Toc182762131)

[3. Prioritized Experience Replay 1](#_Toc182762132)

[4. Stratégie d'Exploration 2](#_Toc182762133)

[5. Mécanisme de Mise à Jour 2](#_Toc182762134)

[Analyse des Performances 2](#_Toc182762135)

[1. Récompenses 2](#_Toc182762136)

[2. Convergence 2](#_Toc182762137)

[3. Exploration-Exploitation 2](#_Toc182762138)

[Limites et Recommandations 3](#_Toc182762139)

[1. Temps d'Entraînement 3](#_Toc182762140)

[2. Analyse de l'Apprentissage 3](#_Toc182762141)

[3. Ajustement des Hyperparamètres 3](#_Toc182762142)

[Conclusion 3](#_Toc182762143)

# Objectif

L'objectif de ce projet est de développer un agent capable de jouer au jeu Atari Pong en utilisant des techniques de **Reinforcement Learning** (RL) basées sur l'article "Playing Atari with Deep Reinforcement Learning" de Mnih et al.

# Description de l'Implémentation

## **1. Prétraitement des Observations**

Les images brutes des frames sont converties en niveaux de gris pour réduire la complexité. Les frames sont redimensionnés à une taille de (84, 84) pour standardiser l'entrée. Un **stacking de 4 frames consécutives** est utilisé pour capturer la temporalité des observations. Ces prétraitements permettent de réduire la complexité dimensionnelle tout en conservant les informations temporelles essentielles pour Pong.

## **2. Modèle Dueling DQN**

Le modèle implémente un réseau de neurones convolutifs (CNN) suivi de couches entièrement connectées. En effet, les CNN permettent d'extraire les caractéristiques spatiales des images. Le réseau utilise deux branches distinctes :

* + **Valeur (Value)** : Évalue la qualité globale de l'état actuel.
  + **Avantage (Advantage)** : Calcule l'impact relatif de chaque action.

La sortie combine les deux branches. Cela permet au réseau d’avoir une meilleure convergence grâce à une décomposition explicite des contributions de l'état et des actions et d’améliorer l'efficacité dans les environnements à grande dimension d'action comme Pong.

## **3. Prioritized Experience Replay**

Un Prioritized Experience Replay Buffer est utilisé pour échantillonner des transitions avec une probabilité proportionnelle à leur erreur TD (Temporal Difference). Cela permet à l'agent de s'entraîner davantage sur les transitions importantes, accélérant l'apprentissage. Ce qui donne l’avantage de se concentrer sur les transitions les plus utiles pour l'apprentissage et de corriger les biais pour un apprentissage stable.

## 

## **4. Stratégie d'Exploration**

L'agent suit une politique epsilon-greedy :

* Avec probabilité ϵ, une action aléatoire est choisie.
* Sinon, l'action avec la valeur Q, la plus élevée est choisie.

ϵ décroît progressivement de 1.0 à 0.01, encourageant l'exploration initiale suivie d'une exploitation. Cette stratégie implique une exploration agressive au début pour découvrir différentes stratégies et exploitation progressive des meilleures actions découvertes.

## **5. Mécanisme de Mise à Jour**

Les valeurs Q cibles sont calculées avec un réseau cible (target network). Le réseau cible est synchronisé avec le réseau principal toutes les 1000 étapes. Une fonction de perte MSE pondérée est utilisée, et les gradients sont propagés via l'optimiseur Adam avec un taux d'apprentissage 0.0001. Ce qui nous donne une réduction des oscillations et divergence pendant l'entraînement.

# Analyse des Performances

Après plusieurs épisodes d'entraînement, les résultats peuvent être résumés comme suit :

## **1. Récompenses**

Les récompenses sont initialement très faibles (autour de -21, qui correspond à une défaite totale). Voici les raisons possibles :

* Le modèle explore principalement des actions aléatoires pendant les premières étapes (élevée ϵ).
* Pong est un environnement avec un signal de récompense **sparse** (les points ne sont attribués que lorsqu'une balle est gagnée/perdue).

## **2. Convergence**

L'agent commence à converger vers de meilleures performances (récompenses plus proches de 0) après plusieurs centaines d'épisodes. Cependant, la convergence est lente, en partie à cause :

* De la complexité de Pong.
* De l'insuffisance de la capacité du buffer d'expérience (10,000) pour capturer des états variés.

## **3. Exploration-Exploitation**

La décroissance d'ϵ (0.995) permet une transition progressive de l'exploration à l'exploitation. Cependant, l'agent pourrait bénéficier d'une exploration plus prolongée (ϵdecay=0.999).

# Limites et Recommandations

## **1. Temps d'Entraînement**

Avec seulement 500 épisodes, il est peu probable que l'agent atteigne des performances élevées. Il est recommandé de prolonger l'entraînement à au moins **10,000 épisodes**.

## **2. Analyse de l'Apprentissage**

Ajouter des métriques de suivi :

* **Perte moyenne :** Pour observer la convergence.
* **Valeurs Q moyennes :** Pour évaluer l'évolution des estimations.
* **Graphiques de récompenses :** Suivi des moyennes mobiles des récompenses.

## **3. Ajustement des Hyperparamètres**

1. Augmenter la taille du buffer de mémoire à 100,000.
2. Augmenter la taille des batchs à 64 pour une meilleure estimation des gradients.
3. Tester une stratégie d’exploration plus avancée (ex : **Noisy Nets**)

# **Conclusion**

Le code qu’on a implémenté est une approche robuste pour entraîner un agent DQN à jouer à Pong. Les améliorations telles que le **Dueling DQN** et le **Prioritized Replay** augmentent l'efficacité de l'apprentissage. Cependant, des ajustements hyperparamétriques et un temps d'entraînement prolongé sont nécessaires pour atteindre des performances maximales.