Rapport d'implementation RL

Introduction

Notre projet s'articule autour de l'implementation d'un modèle d'apprentissage par renforcement profond (Deep Reinforcement Learning), spécifiquement le Deep Q-Network (DQN), pour le jeu Atari Breakout. Nous avons utilisé une architecture de réseau de neurones convolutifs pour traiter les états visuels et estimer la fonction de Q-value pour les actions possible. Cette methode a été choisie pour sa capacité à apprendre des politiques d'action directement à partir des pixels bruts, sans nécessiter de caractéristiques implementer a la main.

1 Implementation

Ma première implémentation a suivi rigoureusement les détails techniques du papier. Leur méthode s'articule principalement autour de deux éléments permettant selon eux d'améliorer les performances du modèle : le Replay Buffer et les deux réseaux de neurones permettant d'approcher la q-value. Mon premier replay buffer se contente de récupérer des actions et des états du jeu afin de pouvoir les réutiliser pendant l'entraînement afin de mettre à jour les poids du modèle à chaque épisode. Cette première implémentation se contente dans un premier temps, dans une phase d'initialisation, de remplir ce dernier d'actions aléatoires sans initialiser des modèles. Le nombre d'expériences présent dans le buffer peut être déterminé via les hyperparamètres. Le buffer pour toutes les expériences a donc été initialisé à 50 000 Sarsd (state, action, reward, next state, done) avec une capacité maximale de 700 000 Sarsd. Cette classe Sarsd permet de stocker efficacement les résultats de env.step(actions) afin de les incorporer dans le buffer ou de les utiliser pour calculer la q-value. Les réseaux quant à eux respectent l'architecture dictée dans le papier, pareil pour l'optimizer utilisé, dans notre cas RMSprop.

1.1 training

La fonction train est principalement composée de deux boucles permettant d'itérer un nombre d'épisodes et la seconde pour effectuer des actions. On a initialisé un nombre de coups maximum par épisode à 5000 mais ce paramètre est plus une sécurité car chaque épisode se terminera après la perte des 5 vies et l'indicateur done qui passera à True et sera intercepté. Comme dit plus haut, chaque action est stockée dans le replay buffer qui efface les plus anciennes si la capacité maximale a été atteinte ; un batch est extrait du buffer et utilisé pour mettre à jour le modèle. Tous les X temps définis par TARGET NETWORK UPDATE FREQUENCY, les poids du modèle principal sont copiés dans le réseau secondaire qui est utilisé pour calculer les q-values. L'utilisation d'un réseau secondaire séparé stabilise l'entraînement de deux manières principales : en fixant les q-values cibles grâce à l'apprentissage par différence temporelle et en minimisant les mises à jour, ce qui empêche les q-values de prédire des valeurs évoluant constamment et réduit les corrélations entre les deux réseaux en maintenant le réseau cible fixe pendant plusieurs épisodes. Cela assure la stabilité des valeurs ciblées et évite des ajustements incessants du réseau principal, limitant ainsi les risques de divergence et d'instabilité. Les modèles sont enregistrés tous les 100 épisodes et les métriques sont affichées tous les 10 épisodes. Précisons que, conformément au papier, la valeur des récompenses a été limitée entre -1 et 1, ce qui n'est pas forcément utile dans cette situation car notre agent n'est entraîné que sur un type de jeu alors que cette manœuvre avait pour but de normaliser les récompenses sur tous les jeux Atari. Cependant, elle permettrait aussi de stabiliser l'entraînement.

1.2 metrics

Les métriques choisies pour l'évaluation du système sont : la moyenne des récompenses sur les 50 derniers épisodes, la récompense maximale à ce jour, la moyenne de la perte de notre réseau ainsi que epsilon, le facteur d'exploration de notre modèle.

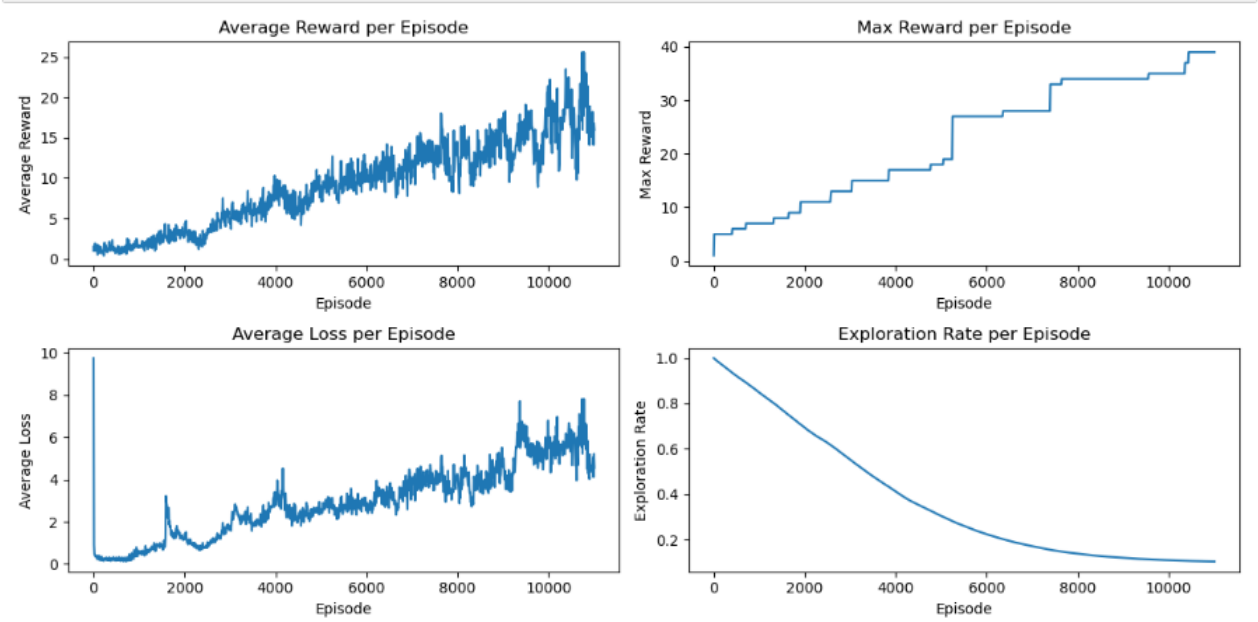

Figura 1: Entrainement 48h, 10000 episodes

1.3 Analyse

L'augmentation progressive de la récompense moyenne et maximale démontre que l'agent a appris de manière significative des stratégies utiles pour améliorer son score. Cependant, l'augmentation de la perte moyenne suggère que l'agent pourrait bénéficier d'un ajustement des hyperparamètres ou d'une fonction de perte différente pour stabiliser et améliorer davantage l'apprentissage. Les échecs étaient souvent dus à une mauvaise anticipation des trajectoires de la balle dans les situations où la vitesse de jeu s'accélérait. Ce qui n'est pas visible dans ces schémas est aussi la proportion d'actions choisies ; en effet, le modèle a une fâcheuse tendance à choisir la droite plutôt qu'une autre action, notamment lorsque la balle accélère.

1.4 Amelioration

J'ai par la suite implémenté une version modifiée du fonctionnement du replay buffer, nommée replay buffer optimisé, qui, plutôt que de remplacer les données les plus anciennes par les nouvelles, leur accorde cette fois-ci un score d'importance et conserve uniquement les plus impactantes. Le critère attribué est un score de priorité basé sur l'erreur de différence temporelle (TD). L'erreur TD est une mesure de la différence entre la q-value prédite et la valeur q-value cible. Dans les buffers standard, les expériences sont prises de manière aléatoire pour briser les corrélations désavantageuses. Le buffer optimisé va encore plus loin en brisant non seulement les corrélations, mais également en sélectionnant activement les expériences les plus bénéfiques pour l'apprentissage.

Par manque de temps, je n'ai pu entraîner les modèles que 10 heures sur environ 3500 épisodes. Nous pouvons cependant voir que les statistiques de score sont relativement identiques dans un premier temps,

Tabela 1: Comparison of Replay Buffers

Classic Replay Buffer				Optimized Replay Buffer			
Epsilon	Avg Rewards	Max Rewards	Loss	Epsilon	Avg Rewards	Max Rewards	Loss
0.9	3.2	5	0.3	0.9	3.0	6	0.2
0.8	5.1	9	0.7	0.8	6.5	14	1.1
0.7	8.2	11	1.2	0.7	12.1	17	1.2
0.6	9.6	17	2.7	0.6			
0.5	11.4	19	3.1	0.5			
0.4	12.2	27	3.0	0.4			
0.3	15.2	27	3.4	0.3			
0.2	14.3	34	4.3	0.2			
0.1	21	39	7.1	0.1			

mais on note une croissance bien supérieure par la suite. La récompense maximale peut varier par chance, mais la récompense moyenne permet d'observer de meilleurs résultats. La perte, quant à elle, continue d'augmenter à la même fréquence indépendamment des deux implémentations.