HOSSAIN Shajjad
BAMBA William

Apprentissage profond par renforcement

Introduction

L'objectif de ce TP était de nous familiariser avec les techniques d'apprentissages profond par renforcement, en particulier le Deep-Q-Learning. Cette technique récente permet à un agent d'apprendre des politiques efficaces même avec des données d'entrées à nombreuses dimensions.
Nous nous sommes grandement inspirés du papier de recherche pointé par l'énoncé du TP, [Mnih et al., 2015], pour l'implémentation de l'algorithme, ces différentes optimisations et les paramètres des modèles. En effet un tel algorithme requiert d'être entraîné longtemps sur un grand nombre de données et, dans certains cas, il n'est pas garanti de converger. Pour adresser ce problème, le TP nous guide vers différentes méthodes inspirées du papier de recherche, à savoir l'Expérience Replay et la Fixed-Q-Target qui améliorent grandement les performances du DQL.

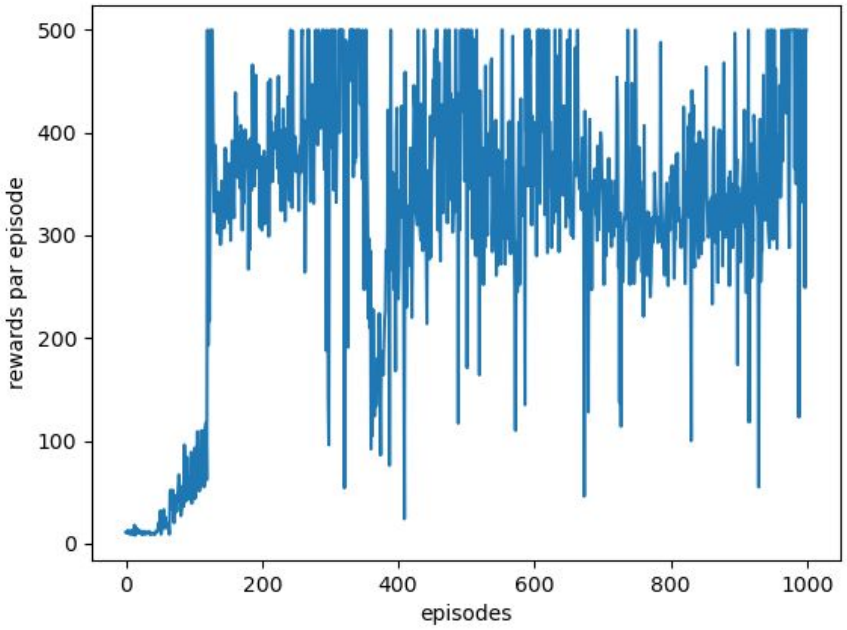
Deep Q-network sur CartPole-v1

Dans cette partie nous avons utilisé un réseau de neurones et l'apprentissage par renforcement pour équilibrer un bâton sur un chariot dans un environnement 2D. Dans cet environnement seulement deux actions sont possibles (0 ou 1) pour déplacer le chariot à droite ou à gauche. L'état de l'environnement est constitué de 4 valeurs représentant la position du chariot et du bâton. Le but est donc de choisir une action en fonction de ces 4 valeurs.
Dans telle situation l'apprentissage par renforcement est la solution idéale. On pourrait donc explorer l'environnement et utiliser l'algorithme de Q-learning classique pour choisir l'action avec la plus grande Q-valeur. Néanmoins même pour cette environnement tout simple l'explorer tous les états est quasiment impossible. D'où l'intérêt d'estimer les Q-valeurs avec un réseau de neurones.
Pour ce simple environnement nous avons donc utilisé un réseau de neurones entièrement connecté en trois couches. De plus, au lieu d'apprendre directement à chaque action nous utilisons un buffer pour stocker les expériences réalisées. A chaque itération nous tirons au hasard les expériences et apprenons l'estimation de Q-valeur. Jusqu'à maintenant nous avons utilisé la même réseau pour prédire et estimer les meilleurs Q-valeurs pour les états suivants. Ce qui nous mène à une l'instabilité des Q-valeurs. Nous utilisons donc deux réseaux, un pour prendre les actions et l'autre pour estimer les Q-valeurs les états suivants. Et on recopie les

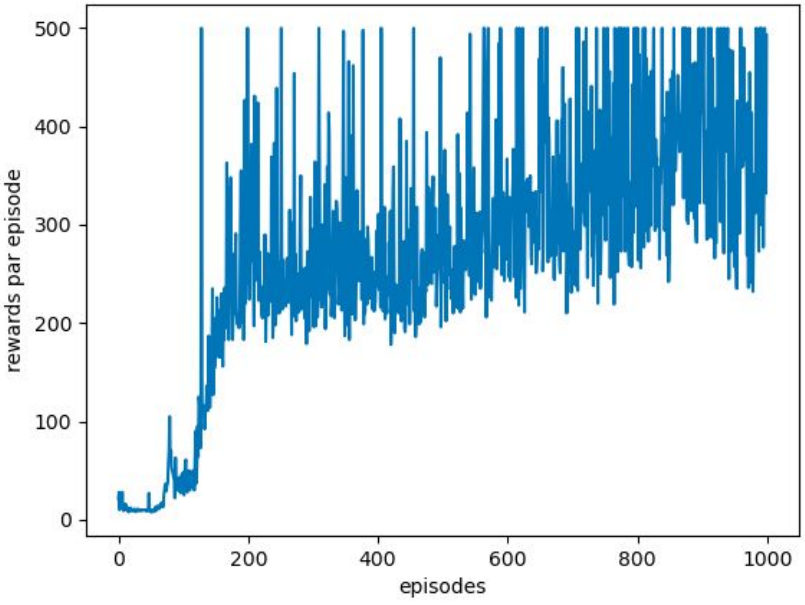
paramètres du réseau de prédiction au réseau d'estimation avec un paramètre alpha (égale à 0,01).

Mais nous avons remarqué certaines fois que les récompenses sont trop basses et c'est dû à la partie aléatoire de greddy exploration. Nous avons donc diminué la probabilité de choisir une action aléatoire au cours des itérations. Et voici la différence entre les deux.

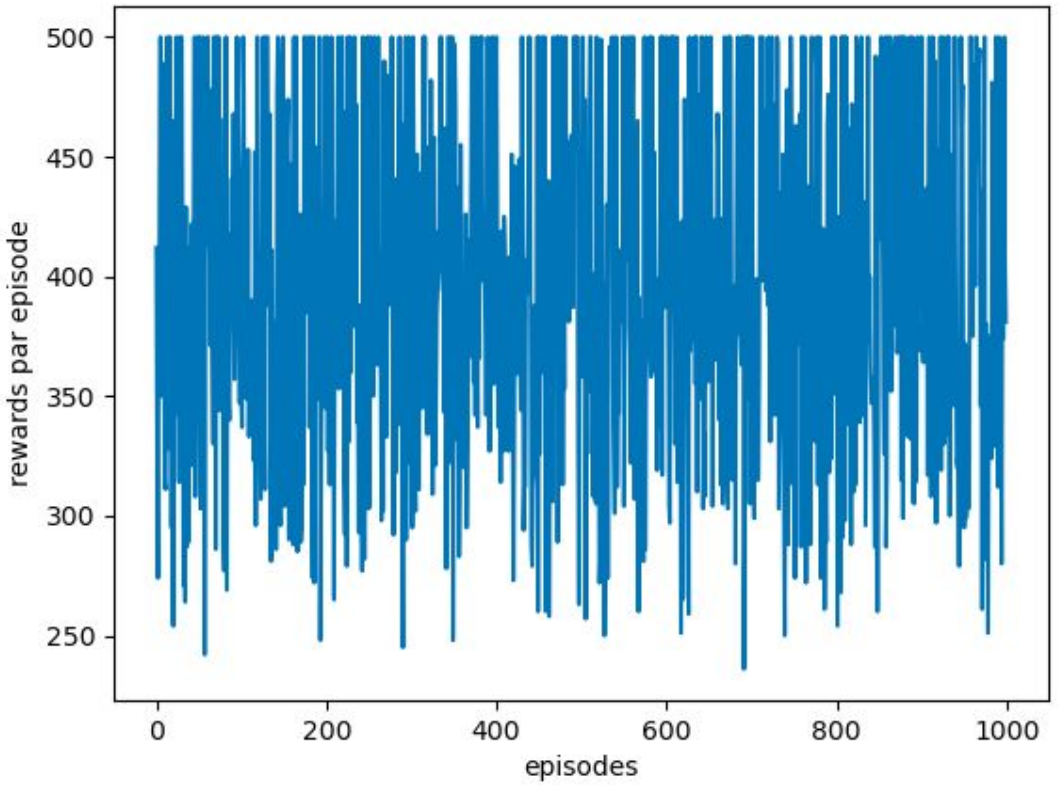
Avant diminution du probabilité:



Après diminution du probabilité:



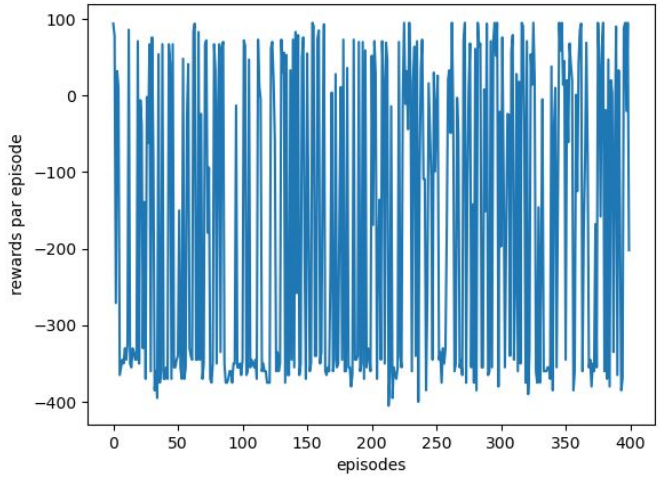
Et voici une image d'évaluation de l’agent cart pole.



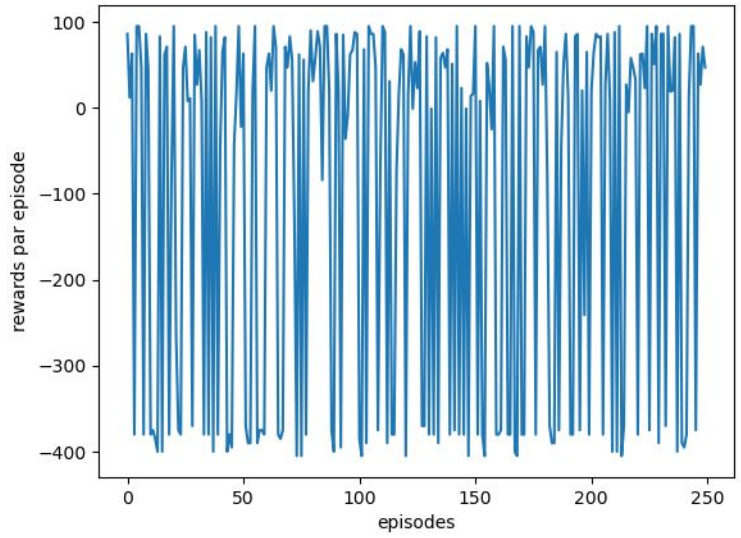
Environnement plus difficile: Vizdoom

Dans cette partie nous allons jouer à doom. L'environnement est donc beaucoup plus complexe, on utilise directement l’image de jeu cette fois pour prédire les actions à prendre. Grâce à l’interface commun de gym seulement la partie prétraitement des états et le réseau de neurone changent.
Pour la partie pré traitement dans le but d’améliorer la performance nous diminuons la taille de l’image et le transformant en noir et blanc. Et pour la partie réseau de neurones nous utilisons un réseau à trois couche avec deux couche convolutionnel et un couche de sortie connecté. L’idée est d’apprendre des patterns sur l’image et prédire les Q-valeurs.

Entrainement de l’agent VizDoom:



Evaluation de l’agent:



Nous sommes conscients que notre modèle n'apprend pas le comportement nécessaire. Il faut probablement l'entraîner plus longtemps avec plus de couche dans la partie la partie Cnn et les taille des sortie un peu plus grande par exemple 64 au lieu de 8. Faute d’avoir un pc assez puissant et le code étant extrêmement long nous n’avons pas pu tester et optimiser les différents paramètres.