UQAC

UNIVERSITÉ DU QUÉBEC
À CHICOUTIMI

Projet Final

8INF974 - Atelier pratique en intelligence artificielle II

Projet Final : Comparaison de méthodes

Maxime SIMARD
msimard104@etu.uqac.ca
SIMM26050001

Tifenn LE GOURRIEREC
tlgourrier@etu.uqac.ca
LEGT08590100

22 avril 2024

Table des matières

I Introduction

L'apprentissage par renforcement (Reinforcement Learning) constitue un domaine crucial de l'intelligence artificielle, se concentrant sur la manière dont les agents prennent des décisions dans un environnement pour maximiser une notion de récompense cumulative. À l'aide des techniques d'apprentissage profond qui s'améliorent de jour en jour, l'apprentissage par renforcement a connu des avancées significatives, permettant des applications allant du jeu automatisé à la robotique avancée et à la gestion autonome des systèmes. Le domaine qui nous intéresse est plus particulièrement son application dans les jeux vidéos et les mondes simulés. Ce rapport se penche donc sur l'évaluation comparative de trois architectures majeures d'apprentissage par renforcement utilisant des réseaux de neurones profonds : le Deep Q-Network (DQN), le Double DQN, et le Double DQN avec Replay Priorisé. Ces techniques représentent une évolution de la méthode classique Q-Learning adaptée aux défis posés par des environnements de grande dimension et des états d'entrée complexes.

I.1 Objectif du Projet

L'objectif principal de ce projet est de benchmarker et de comparer les performances de trois architectures d'apprentissage par renforcement différentes en les testant dans des environnements standardisés et contrôlés. En utilisant des scénarios tels que Cartpole, Acrobot, et Mountain Car, qui sont des benchmarks établis dans la recherche en apprentissage par renforcement, nous évaluons l'efficacité, la rapidité de convergence, et la robustesse des politiques d'apprentissage générées par chaque modèle.

II État de l'Art

L'apprentissage par renforcement a connu des avancées significatives avec l'introduction des réseaux de neurones profonds, permettant de gérer des problèmes complexes de décision séquentielle dans des environnements riches en informations. Cette section explore les développements clés qui ont façonné l'état actuel de l'apprentissage par renforcement profond, en mettant l'accent sur trois architectures innovantes : DQN, Double DQN, et le Prioritized Replay. Chacune de ces méthodes représente une stratégie pour améliorer l'efficacité et la stabilité de l'apprentissage automatique dans des contextes variés, démontrant des améliorations notables par rapport aux approches classiques d'apprentissage par renforcement.

II.1 Deep Q-Networks

Le Deep Q-Network (DQN)a marqué une avancée majeure dans le domaine de l'apprentissage par renforcement à l'aide de réseaux de neurones profonds. Introduit par Mnih et al. en 2015 dans le papier "Playing Atari with Deep Reinforcement Learning", le DQN a été conçu pour traiter des

environnements à grande dimensionnalité d'entrée, ce qui était auparavant un défi majeur dans le domaine. Ce modèle utilise une technique de mémoire de replay pour stocker les expériences passées de l'agent et les réutilise de manière aléatoire, ce qui améliore significativement la stabilité de l'apprentissage. Cette méthode aide également à rompre les corrélations séquentielles dans les observations, offrant un apprentissage plus robuste et évitant les cycles de feedback qui pourraient être nuisibles.

II.2 Double DQN

Le Double DQN est une extension du DQN qui a pour but d'atténuer le biais de surévaluation souvent observé avec le DQN standard. Proposé par Hasselt et al. en 2016 dans le papier "Deep Reinforcement Learning with Double Q-learning", cette architecture utilise deux réseaux de neurones séparés : un réseau pour sélectionner l'action qui maximise la fonction de valeur et un autre pour évaluer cette action. Cette séparation permet de réduire considérablement les erreurs de surévaluation qui peuvent survenir lorsque le même réseau est utilisé à la fois pour la sélection et l'évaluation des actions. Le Double DQN assure que l'estimation des valeurs d'action reste fiable et précise, contribuant à une meilleure performance globale de l'algorithme.

II.3 Prioritized Replay

Cette variante améliore le modèle en intégrant un mécanisme de replay priorisé. Développée par Schaul et al. en 2016 dans le papier "Prioritized Experience Replay", cette technique modifie la manière dont les expériences sont stockées et récupérées dans la mémoire de replay. Au lieu d'une sélection uniforme, le replay priorisé permet de mettre en avant certaines transitions en fonction de leur erreur de différence temporelle (TD), ce qui permet à l'agent de se concentrer sur les expériences les plus informatives ou les plus inattendues. Cela accélère l'apprentissage en optimisant les ressources de calcul pour se concentrer sur les parties de l'environnement qui bénéficient le plus d'une réévaluation.

III Méthodologie

Cette section détaille les méthodes et les processus utilisés pour évaluer et comparer les performances des architectures d'apprentissage par renforcement étudiées. Nous commençons par décrire les environnements de simulation utilisés pour le benchmarking des modèles, suivi de l'explication de la configuration des architectures et des critères d'évaluation.

III.1 Environnements

Les environnements sélectionnés pour tester les modèles sont des benchmarks classiques dans la recherche en apprentissage par renforcement. Ces environnements offrent des défis diversifiés

qui permettent d'évaluer l'efficacité des algorithmes sous différentes conditions dynamiques et avec différentes complexités de décision.

III.1.a Cartpole

Le problème de Cartpole est un test classique dans l'apprentissage par renforcement. L'objectif est de maintenir un poteau qui se trouve sur un chariot mobile en équilibre vertical en déplaçant le chariot vers la gauche ou la droite. Cet environnement est représentatif des tâches de contrôle dynamique et teste la capacité de l'algorithme à gérer les actions qui ont des conséquences à long terme sur l'état futur du système. La récompense est donnée à chaque pas de temps que le poteau reste en position verticale. Le défi est d'apprendre à prévoir l'impact des mouvements du chariot sur le mouvement du poteau, ce qui requiert une prédiction précise et des actions réactives.

L'environnement Cartpole fournit un espace d'observation à quatre dimensions qui comprend la position du chariot sur la piste, la vitesse du chariot, l'angle du poteau par rapport à la verticale, et la vitesse angulaire du poteau. Ces observations permettent à l'agent de prendre des décisions éclairées sur la manière de déplacer le chariot pour maintenir le poteau en équilibre.

L'ensemble d'actions est discret et comprend deux actions possibles : pousser le chariot vers la gauche (0) ou vers la droite (1). L'action choisie par l'agent a un effet direct sur le mouvement du chariot et, par conséquent, sur la dynamique du poteau.

III.1.b Acrobot

L'Acrobot est un système à deux bras articulés, avec le but d'osciller les bras pour que l'extrémité du bras atteigne une hauteur donnée. Il simule un gymnaste effectuant un mouvement pour se hisser au-dessus d'une barre horizontale. L'environnement Acrobot évalue la capacité des algorithmes à planifier sur plusieurs étapes pour atteindre l'objectif final. Les récompenses sont structurées pour encourager l'agent à atteindre l'objectif le plus rapidement possible, ce qui nécessite une combinaison efficace de mouvements coordonnés.

Les observations dans l'Acrobot sont constituées de six variables : les angles cosinus et sinus des deux joints, ainsi que la vitesse angulaire de ces joints. Cette information fournit une représentation complète de l'état physique actuel de l'Acrobot, nécessaire pour calculer les actions optimales pour balancer les bras et atteindre l'objectif.

L'espace d'actions de l'Acrobot est également discret, incluant trois actions : appliquer un couple positif (0), aucun couple (1), ou un couple négatif (2) aux articulations. Ces actions influencent directement la dynamique des bras et la stratégie requise pour que l'extrémité du bras atteigne la hauteur désirée.

III.1.c Mountain Car

Dans l'environnement Mountain Car, un véhicule est situé dans une vallée et doit atteindre le sommet d'une montagne. Cependant, la voiture ne possède pas assez de puissance pour escalader directement la montagne en partant d'un arrêt. L'agent doit apprendre à exploiter la gravité en reculant sur la colline opposée avant de pouvoir atteindre l'objectif. Cet environnement est particulièrement utile pour tester les algorithmes sur leur capacité à apprendre des politiques qui nécessitent des séquences d'actions précises et des contre-intuitifs. La récompense est attribuée lorsque la voiture atteint le sommet de la montagne, et l'environnement est conçu pour punir les tentatives qui prennent trop de temps, encourageant ainsi des solutions rapides et créatives.

Pour Mountain Car, l'agent reçoit deux observations continues : la position de la voiture et sa vitesse. Ces deux variables sont essentielles pour que l'agent comprenne non seulement où se trouve la voiture par rapport aux objectifs, mais aussi la dynamique de son mouvement pour pouvoir planifier les actions nécessaires à l'atteinte du sommet de la montagne.

L'ensemble d'actions dans Mountain Car est discret. Il comprend trois options : accélérer vers la gauche (0), ne rien faire (1), ou accélérer vers la droite (2). Choisir la bonne action dans le bon contexte est crucial, car la voiture doit souvent reculer (accélérer vers la gauche) pour prendre suffisamment d'élan pour monter la pente raide.

III.2 Comparaison des architectures

Dans cette section, nous décrivons l'implémentation des modèles utilisés pour chaque algorithme d'apprentissage par renforcement étudié : DQN, Double DQN, et Double DQN avec Replay Priorisé.

III.2.a DQN

Pour l'architecture du DQN, nous avons adopté un modèle conditionné par la nature des observations d'entrée.

Si les observations sont des images (plusieurs dimensions), le réseau utilise des couches convolutionnelles pour traiter les données spatiales.

Le réseau commence avec trois couches convolutionnelles :

— Une première couche avec 32 filtres de taille 8x8 et un pas de 4.

— Une deuxième couche avec 64 filtres de taille 4x4 et un pas de 2.

— Une troisième couche avec 64 filtres de taille 3x3 et un pas de 1.

Chaque couche convolutionnelle est suivie d'une fonction d'activation ReLU pour introduire de la non-linéarité. Après le traitement par les couches convolutionnelles et un aplatissement des données, le réseau se poursuit avec deux couches linéaires :

— Une couche de 512 neurones, suivie d'une activation ReLU.

— Une couche de sortie correspondant au nombre d'actions possibles dans l'environnement, permettant de prédire la valeur Q pour chaque action.

Si les observations sont unidimensionnelles (vecteurs simples), le réseau utilise une série de trois couches linéaires :

— Deux couches cachées de 128 neurones chacune, avec des fonctions d'activation ReLU entre elles.

— Une couche de sortie qui, comme dans le cas des images, correspond au nombre d'actions possibles.

Cette architecture adaptative permet au DQN de gérer efficacement différents types d'entrées, ce qui en fait une solution robuste pour les environnements tels que Cartpole, Acrobot et Mountain Car.

III.2.b Double DQN

L'architecture du Double DQN reprend la base du DQN standard, mais avec une distinction clé dans la gestion des mises à jour de la valeur Q. Deux réseaux de neurones identiques mais indépendants sont utilisés : le réseau d'évaluation et le réseau cible. Le réseau d'évaluation sélectionne l'action maximisant la valeur Q, tandis que le réseau cible estime cette valeur. Cela permet de réduire le biais de surévaluation en désaccouplant la sélection de l'action de son évaluation.

III.2.c Double DQN avec Replay Priorisé

Cette variante améliore le Double DQN par l'intégration d'un mécanisme de replay priorisé. La structure réseau reste similaire à celle du Double DQN, mais le processus de mise à jour est modifié pour incorporer la priorité des expériences dans la mémoire de replay. Les expériences avec des erreurs de prédiction élevées (c'est-à-dire une grande différence entre la valeur Q prédite et la valeur Q obtenue) sont plus susceptibles d'être rééchantillonnées, ce qui conduit à une convergence plus rapide et à une meilleure performance globale.

III.3 Métriques d'évaluation

PPour évaluer et comparer efficacement les architectures sélectionnées, nous utiliserons plusieurs critères pour mesurer la performance des modèles dans chaque environnement de test. Chaque architecture sera déployée dans les environnements pour un total de 500 épisodes, avec un modèle distinct entraîné spécifiquement pour chaque environnement. Ce cadre permettra d'assurer que les comparaisons soient basées sur des données uniformément recueillies et pertinentes pour chaque scénario de test spécifique. Les métriques suivantes seront utilisées pour évaluer chaque modèle :

— **Démarrage Rapide :** Cette métrique évalue à quel vitesse le modèle commence à obtenir de bons scores durant les épisodes. Un démarrage rapide à obtenir de bons scores est

indicatif de l'efficacité de l'apprentissage initial de l'algorithme et de sa capacité à s'adapter rapidement aux exigences de l'environnement.

— **Stabilité des Scores :** La stabilité est mesurée par la consistance des scores obtenus au fil des épisodes. Un modèle idéal devrait montrer peu ou pas de pertes de performance aléatoires, indiquant une robustesse et une fiabilité élevées dans ses politiques d'apprentissage. La variance des scores entre les épisodes sera analysée pour évaluer cette stabilité.

— **Score Élevé :** La performance finale du modèle, ou le score le plus élevé atteint, sera comparée à celle des autres architectures pour le même environnement. Cette métrique est cruciale car elle reflète la capacité du modèle à maximiser les récompenses dans l'environnement donné, ce qui est l'objectif central de l'apprentissage par renforcement.

Ces trois points clés formeront la base de notre analyse comparative des modèles. Pour une visualisation claire de ces métriques, des graphiques de l'évolution des scores par épisode seront générés pour chaque modèle. Ces graphiques aideront à illustrer non seulement la progression de l'apprentissage au fil du temps mais aussi à mettre en évidence les différences de performance entre les modèles au sein de la même architecture ainsi qu'entre les différentes architectures.

IV Résultats

Pour fournir une présentation plus claire et concise des performances des modèles DQN au cours de leur entraînement, les scores obtenus ne seront pas affichés pour chaque épisode individuel. Au lieu de cela, nous avons opté pour une approche qui regroupe les scores en moyennes calculées sur des blocs de 10 épisodes. Cette approche réduit la variabilité des performances, permettant une meilleure visualisation des tendances d'apprentissage à long terme. Les résultats seront donc présentés sous forme de graphiques représentant 50 époch, chaque époch représentant la moyenne des scores de 10 épisodes, ce qui offre une vue plus stabilisée et compréhensible de l'évolution des performances au fil du temps.

IV.1 Hyperparamètres

Afin de simplifier le temps d'entraînement, la collecte de résultat, et la comparaison sur un pied égal de trois variations du même algorithmes, nous avons décidé d'utiliser le même ensemble d'hyperparamètres pour chaque modèle. Cette combinaison a été choisi car il a prouvé offrir de bonne performance lorsque combinée avec le DQN traditionnel, lui donnant donc un certain avantage contre ses variations plus avancées.

Nous avons choisi les hyperparamètres suivants :

— Learning Rate = 1×10^{-4}

— Epsilon Initial = 0.9

— Epsilon Final = 0.05

— Epsilon Decay = 1000

— Memory Capacity = 10000
— Batch Size = 128

IV.2 Cartpole

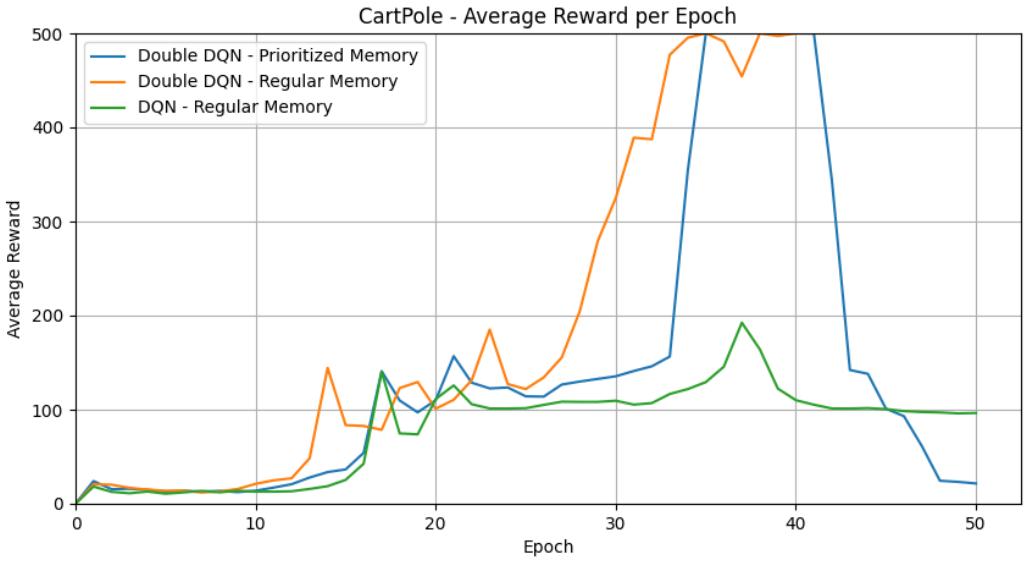

FIGURE 1 – Performance dans l'environnement Cartpole

IV.2.a Démarrage Rapide

DQN : Le modèle DQN a montré un démarrage relativement lent, atteignant son premier pic de performance (récompense de 100) autour de l'époch 18. Ce délai pour atteindre une performance initiale acceptable indique une adaptation plus graduelle à l'environnement.

Double DQN : Le DDQN a démarré plus rapidement, atteignant une récompense de 100 autour de l'époch 14. Cette capacité à s'adapter plus rapidement peut être attribuée à sa méthode de réduction du biais de surévaluation, améliorant l'apprentissage initial.

Double DQN avec Replay Priorisé : Similaire au DQN, le Double DQN avec Replay Priorisé a atteint 100 de récompense pour la première fois autour de l'époch 18, montrant un démarrage initial similaire au DQN malgré l'utilisation d'un replay priorisé.

IV.2.b Stabilité des Scores

DQN : Après son premier pic, le DQN a maintenu un niveau de performance autour de 100 pendant les 30 époch suivantes, démontrant une stabilité dans l'apprentissage malgré l'absence d'augmentation significative des scores.

Double DQN : Le DDQN a non seulement commencé plus tôt à atteindre de bons scores, mais a également montré une amélioration significative entre les épochs 25 et 35, où les récompenses ont grimpé de 150 à 500. Cette performance s'est stabilisée et est restée constante jusqu'à la fin de l'entraînement.

Double DQN avec Replay Priorisé : Bien que ce modèle ait également montré une hausse impressionnante de ses scores entre les épochs 33 et 35, il a connu une chute abrupte à l'époch 42, retombant à 100 et descendant jusqu'à la fin. Cette volatilité suggère que bien que le replay priorisé puisse accélérer l'apprentissage, il peut aussi introduire une instabilité dans les performances.

IV.2.c Score Élevé

DQN : Le score le plus élevé obtenu par le DQN est resté constant à environ 100, indiquant une performance compétente mais sans améliorations majeures au-delà de ce seuil.

Double DQN : Le DDQN a démontré une capacité supérieure à maximiser les récompenses, atteignant un score élevé de 500, qui est significativement plus élevé que celui du DQN.

Double DQN avec Replay Priorisé : Bien que ce modèle ait atteint temporairement des scores élevés similaires à ceux du DDQN, sa performance a chuté dramatiquement. Ce comportement suggère que certaines configurations du replay priorisé peuvent nécessiter des ajustements pour maintenir une performance élevée de manière consistante.

IV.3 Acrobot

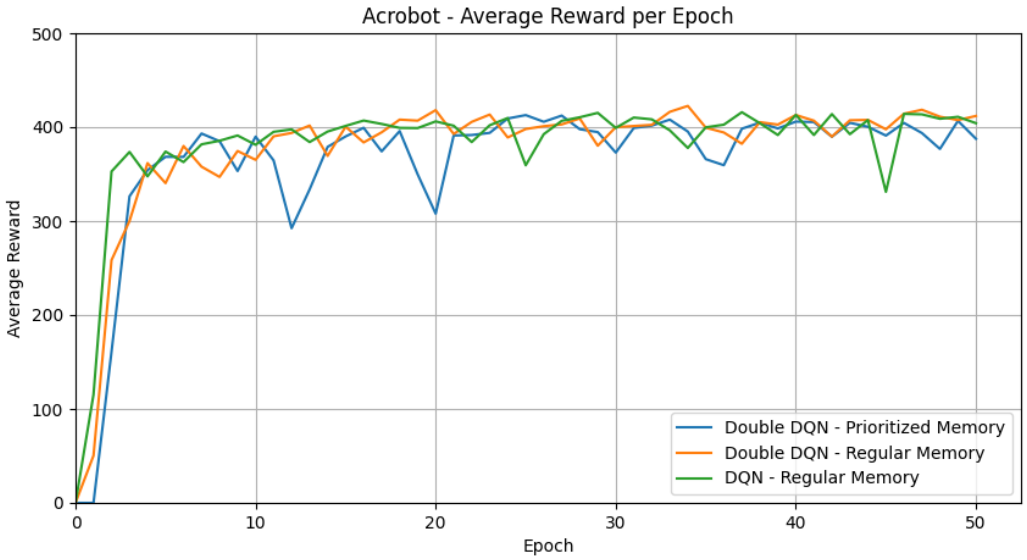

FIGURE 2 – Performance dans l'environnement Acrobot

IV.3.a Démarrage Rapide

DQN : Ce modèle a atteint un score de 350 dès l'époch 2, montrant une capacité rapide à adapter ses stratégies dans l'environnement Acrobot.

Double DQN : Légèrement plus lent que le DQN, le DDQN a atteint le même score à l'époch 3. Bien que ce soit une différence minime, elle illustre une légère variation dans l'efficacité initiale de l'apprentissage entre les deux modèles.

Double DQN avec Replay Priorisé : Le plus lent des trois, ce modèle a atteint 300 à l'époch 4. Le démarrage légèrement retardé peut être attribué à la complexité ajoutée par le mécanisme de replay priorisé.

IV.3.b Stabilité des Scores

DQN : Bien que ce modèle ait atteint rapidement un score élevé, il a montré des fluctuations avec des chutes de 50 points de score de temps en temps, ce qui indique une certaine instabilité dans la réponse aux conditions de l'environnement.

Double DQN : Ce modèle a démontré une grande stabilité après avoir atteint 300, maintenant des performances cohérentes sans chutes notables. La structure du double réseau semble contribuer à une meilleure prévisibilité lors des prises de décision.

Double DQN avec Replay Priorisé : Le plus instable des trois, ce modèle a eu des chutes significatives de 100 points dans le score, illustrant que, malgré les avantages théoriques du replay priorisé en termes d'apprentissage efficace, cela peut aussi mener à des résultats incohérents dans certains scénarios imprévisible comme le balancement d'une double pendule.

IV.3.c Score Élevé

Tous les modèles ont atteint un score de 400 dans cet environnement. Cela indique que malgré leurs différences en termes de stabilité et de vitesse d'adaptation, tous les modèles étaient capables d'atteindre une performance de pointe similaire.

IV.4 Mountain Car

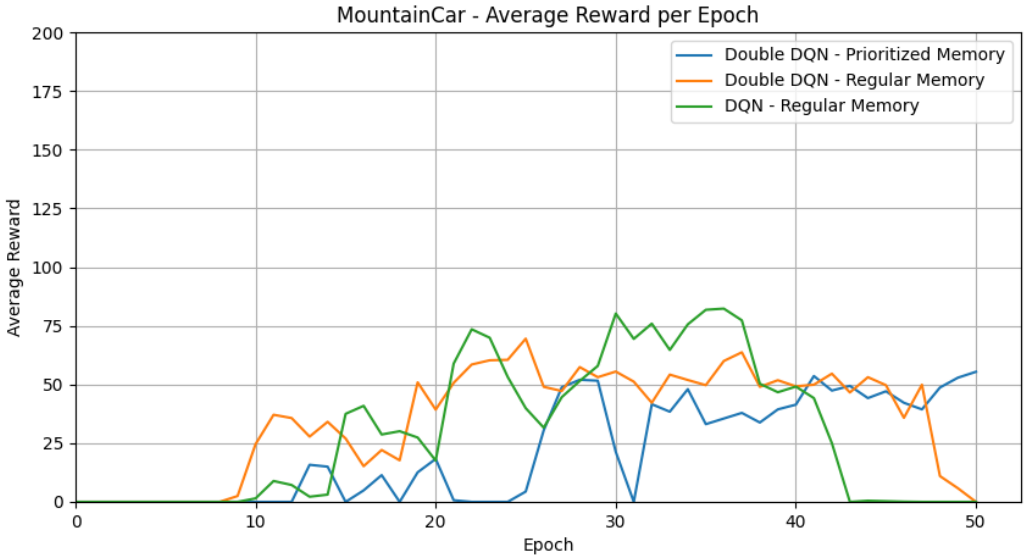

FIGURE 3 – Performance dans l'environnement Mountain Car

IV.4.a Démarrage Rapide

DQN : Ce modèle a atteint son premier score notable à l'époch 15, montrant une adaptation progressive aux exigences de l'environnement.

Double DQN : Le DDQN a démontré un démarrage plus rapide, réalisant un score initial dès l'époch 10. Cette rapidité d'adaptation illustre une efficacité supérieure dans l'apprentissage initial.

Double DQN avec Replay Priorisé : Le plus lent à démarrer, le Double DQN avec Replay Priorisé a montré son premier score notable seulement à l'époch 25, ce qui peut refléter la complexité accrue due au mécanisme de replay priorisé.

IV.4.b Stabilité des Scores

DQN : Le DQN a montré une grande volatilité dans ses scores, avec des oscillations entre 25 et 75 tout au long de l'entraînement. Cette instabilité pourrait être un désavantage dans des contextes nécessitant une performance fiable et prévisible.

Double DQN : Ce modèle a maintenu une stabilité impressionnante, avec des scores régulièrement autour de 50, indiquant une performance plus constante et fiable.

Double DQN avec Replay Priorisé : Bien qu'il a démaré plus lentement, ce modèle a progressé de manière plus régulière et avec qu'une grande chute de score, ce qui pourrait être bénéfique dans des applications où une accumulation graduelle de compétences est préférable.

IV.4.c Score Élevé

DQN : Ce modèle a atteint un score de 75, le plus élevé des trois, montrant que malgré son instabilité, il peut réaliser des performances élevées sous certaines conditions.

Double DQN : Avec un score légèrement inférieur de 70, le DDQN combine une performance élevée avec une stabilité remarquable, ce qui peut le rendre plus approprié pour des applications pratiques où la fiabilité est cruciale.

Double DQN avec Replay Priorisé : Avec un score de 55, ce modèle a la performance globale la plus modeste, montrant une capacité limitée à maximiser les récompenses dans cet environnement spécifique.

IV.5 Comparaison des Performances

Démarrage Rapide : Dans l'ensemble, le Double DQN a souvent montré une capacité à atteindre rapidement de bons résultats, illustrant l'efficacité de la réduction du biais de surévaluation dans divers environnements. Par contre, le Double DQN avec Replay Priorisé a tendance à démarrer plus lentement, ce qui peut être attribué à la complexité accrue de la gestion des priorités de replay, nécessitant plus de temps pour que l'efficacité de l'apprentissage se manifeste. Le DQN standard a généralement affiché des performances initiales plus lentes mais reste compétitif une fois qu'il atteint son rythme.

Stabilité : La stabilité est une métrique où le Double DQN excelle, offrant des performances plus prévisibles, ce qui est crucial pour des applications pratiques nécessitant fiabilité et constance. Le DQN standard et le Double DQN avec Replay Priorisé montrent plus de fluctuations dans leurs performances, avec des épisodes de baisse de scores qui peuvent être problématiques dans des scénarios d'application réels.

Scores Élevés : En termes de score maximal atteint, le Double DQN se distingue souvent par sa capacité à maximiser les récompenses, particulièrement dans des environnements comme Cartpole et Mountain Car. Cela indique que malgré sa complexité, il parvient à exploiter efficacement son architecture pour obtenir de meilleurs résultats globaux. Le Replay Priorisé, bien que prometteur en théorie pour améliorer l'apprentissage, ne se traduit pas toujours par une meilleure performance en termes de score élevé, peut-être en raison de son instabilité.

IV.6 Causes des Variations de Performance

Choix des Hyperparamètres : Les hyperparamètres choisis, notamment un taux d'apprentissage faible et une grande capacité de mémoire, sont généralement bien adaptés pour une convergence stable dans les environnements d'apprentissage par renforcement. Cependant, la valeur d'epsilon final relativement élevée (0.05) pourrait prolonger la phase d'exploration au-delà de l'optimal, affectant particulièrement les performances dans des environnements plus complexes ou

moins prévisibles comme le Mountain Car. Cette exploration continue peut empêcher les modèles de se stabiliser pleinement et de tirer pleinement parti de leurs politiques apprises.

Instabilité Induite par les Mécanismes Avancés : Le Double DQN avec Replay Priorisé montre que l'ajout de fonctionnalités sophistiquées pour accélérer l'apprentissage peut parfois introduire de l'instabilité. Cela pourrait être dû à des erreurs de prédiction initialement élevées qui dominent le processus d'apprentissage, conduisant à un sur-apprentissage des états particulièrement difficiles ou inhabituels au lieu de développer une politique plus généralement efficace.

V Conclusion

Ce projet a comparé les architectures DQN, Double DQN, et Double DQN avec Replay Priorisé dans divers environnements de simulation. Le Double DQN s'est distingué par sa capacité à démarrer rapidement et à maintenir une performance stable, ce qui le rend idéal pour des applications nécessitant fiabilité et prédictibilité. En comparaison, le DQN standard, malgré un démarrage plus lent, a démontré sa robustesse une fois adapté, affirmant sa viabilité dans des environnements moins exigeants en rapidité d'adaptation.

Bien que le Double DQN avec Replay Priorisé soit prometteur pour accélérer l'acquisition de connaissances, son instabilité notable dans les performances pourrait limiter son utilité dans des applications nécessitant une stabilité d'apprentissage. L'évaluation de ces architectures souligne l'importance de sélectionner des solutions adaptées aux spécificités des tâches et des environnements, tout en pointant vers des améliorations potentielles pour augmenter leur stabilité et efficacité.

Ces résultats enrichissent notre compréhension de l'efficacité des différentes architectures et démontrent que le choix doit être soigneusement adapté aux spécificités de l'environnement et aux objectifs de performance. Le projet souligne aussi l'importance de prendre en compte non seulement le score maximal mais également la stabilité des performances et la vitesse d'adaptation des modèles lors de leur sélection.

VI Bibliographie

1. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, "Playing Atari with Deep Reinforcement Learning", arXiv preprint arXiv :1312.5602, 2013.
2. H. van Hasselt, A. Guez, D. Silver, "Deep Reinforcement Learning with Double Q-learning", arXiv preprint arXiv :1509.06461v3, 2015.
3. T. Schaul, J. Quan, I. Antonoglou, D. Silver, "Prioritized Experience Replay", arXiv preprint arXiv :1511.05952v4, 2016.
4. Documentation Gymnasium. Disponible sur : `https://gymnasium.farama.org/environments/classic_control/`

5. Deep Q Networks (DQN), "Deep Q Networks (DQN)". Disponible sur : https://nn.labml.ai/rl/dqn/index.html

6. Tutoriel sur l'apprentissage par renforcement (DQN) — Documentation des tutoriels PyTorch 2.3.0+cu121. Disponible sur : https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html