الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique Et Populaire

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة الشهيد الشيخ العربي التبسي- تبسة

UNIVERSITE LARBI TEBESSI – TEBESSA

Faculté des Sciences et de la Technologie

Département de d'Électronique et Télécommunications

MEMOIRE

Présenté pour l'obtention du **diplôme** de **Master Académique**

Filière : Télécommunications

Spécialité : Réseaux et Télécommunications

Par: **- Maroua ABDELOUAHAD** **- Khaoula AMRANI**

THEME

Apprentissage Par Renforcement Profond Concept et Application

Présenté et évalué, le / / , par le jury composé de :

Nom et prénom		Grade	Qualité
Mme	Malika OUACIFI	MAA	Présidente
M.	Seddik KHEMAISSIA	Pr. Ass	Rapporteur
Mme	Chaima AOUICHE	MCB	Examinatrice


بسم الله الرحمن الرحيم

ملخص

تهدف هذه المذكرة إلى استكشاف تطبيق التعلم العميق المعزز (DRL) على استراتيجيات التداول الآلي. الأهداف الرئيسية هي:

فهم الأساسيات في:

- الفصل الأول: التعلم المعزز (RL)، المفاهيم الأساسية: الوكلاء، البيئات، الإجراءات، الحالات، المكافآت، السياسات، دوال القيمة، دوال قيمة Q، تحدي التوازن بين الاستكشاف والاستغلال، تقارب الخوارزميات.

- الفصل الثاني: التعلم العميق (Deep Learning)، الشبكات العصبية الاصطناعية: الطبقات، دوال التفعيل، الانتشار الأمامي والخلفي. الشبكات العصبية التلافيفية (CNN): التعرف على الصور، معالجة البيانات المكانية. تقنيات التدريب: التحسين عبر النزول التدرجي.

تحليل خوارزميات DRL في الفصل الثالث: التعلم العميق المعزز (DRL):

- DQN: استخدام الشبكات العصبية لتقريب دالة قيمة Q.

- DQN المزدوج: تقليل تقدير القيمة الزائد لدوال Q.

- Dueling DQN: فصل قيمة الحالة عن ميزة الإجراءات.

التنفيذ في الفصل الرابع: تطبيق DRL في سوق الأسهم:

- تطوير استراتيجيات التداول الآلي المناسبة لظروف السوق المتغيرة من خلال المراحل:

  - إعداد البيانات: جمع، تنظيف، معالجة مسبقة للبيانات السوقية التاريخية.

  - نمذجة بيئة التداول: تعريف الحالات، الإجراءات، المكافآت.

  - تنفيذ خوارزميات DRL: تطوير وتدريب نماذج DQN، DQN المزدوج، Dueling DQN.

  - التقييم والتحسين: اختبار الاستراتيجيات، ضبط المعاملات الفائقة، تقييم الأداء.

تهدف هذه المذكرة إلى تحسين ممارسات التداول الآلي من خلال تقليل المخاطر وتعظيم الأرباح بفضل قدرات الذكاء الاصطناعي وDRL. توفر فهماً شاملاً لتطبيق هذه التقنيات في التداول.

Résumé :

Ce mémoire vise à explorer l'application de l'apprentissage par renforcement profond (DRL) aux stratégies de trading algorithmique. Les principaux objectifs sont :

Comprendre les Fondamentaux dans :

- Chapitre 1 : Apprentissage par Renforcement (RL),Concepts clés : agents, environnements, actions, états, récompenses,Politiques, fonctions de valeur, fonctions de valeur Q,Le défis dilemme exploration-exploitation, convergence des algorithmes.
- Chapitre 2 : Apprentissage Profond (Deep Learning) Réseaux de neurones artificiels : couches, fonctions d'activation, propagation avant et arrière.Réseaux de Neurones Convolutionnels (CNN) : reconnaissance d'images, traitement des données spatiales.Techniques d'entraînement : optimisation par descente de gradient.

Analyser les Algorithmes de DRL dans le Chapitre 3 : Apprentissage par Renforcement Profond (DRL)

- DQN : Utilisation des réseaux de neurones pour approximer la fonction de valeur Q.
- Double DQN : Réduction de la surestimation des valeurs Q.
- Dueling DQN : Séparation de la valeur d'un état et de l'avantage des actions.

Implémentation dans Chapitre 4 : Application du DRL au Marché Boursier

Développement de stratégies de trading algorithmique adaptées aux conditions de marché changeantes. par les étapes :

- Préparation des données : collecte, nettoyage, prétraitement des données de marché historiques.
- Modélisation de l'environnement de trading : définition des états, actions, récompenses.
- Implémentation des algorithmes DRL : développement et entraînement des modèles DQN, Double DQN, Dueling DQN.
- Évaluation et optimisation : test des stratégies, ajustement des hyperparamètres, évaluation des performances.

Le mémoire vise à améliorer les pratiques de trading algorithmique en minimisant les risques et maximisant les profits grâce aux capacités de l'IA et du DRL. Il fournit une compréhension détaillée des concepts et techniques, des exemples d'implémentation, et des études de cas pratiques, ouvrant la voie à des systèmes de trading plus intelligents et efficaces

Abstract:

This dissertation aims to explore the application of deep reinforcement learning (DRL) to algorithmic trading strategies. The main objectives are:

Understand the Fundamentals in :

- Chapter 1: Reinforcement Learning (RL),Key concepts: agents, environments, actions, states, rewards,Policies, value functions, Q-value functions,The exploration-exploitation dilemma challenges, algorithm convergence.
- Chapter 2: Deep Learning Artificial neural networks: layers, activation functions, forward and backward propagation.Convolutional neural networks (CNN): image recognition, spatial data processing.Training techniques: gradient descent optimization.

Analyze DRL Algorithms in Chapter 3: Deep Reinforcement Learning (DRL)

- DQN: Use of neural networks to approximate the Q-value function.
- Double DQN: Reduction of Q-value overestimation.
- Dueling DQN: Separation of state value and action benefit.

Implementation in Chapter 4: Applying DRL to the stock market Development of algorithmic trading strategies adapted to changing market conditions. through the steps :

- Data preparation: collection, cleaning and pre-processing of historical market data.
- Modeling of the trading environment: definition of states, actions, rewards.
- Implementation of DRL algorithms: development and training of DQN, Double DQN, Dueling DQN models.
- Evaluation and optimization: strategy testing, hyperparameter adjustment, performance evaluation.

The dissertation aims to improve algorithmic trading practices by minimizing risk and maximizing profit through the capabilities of AI and DRL. It provides a detailed understanding of concepts and techniques, implementation examples, and practical case studies, paving the way for smarter and more efficient trading systems.

Remerciement

*Nous souhaitons exprimer notre plus profonde gratitude à notre encadrant, « **Seddik KHEMAISSIA** » pour son soutien constant, ses conseils éclairés et sa patience tout au long de la réalisation de ce mémoire. Son expertise et son dévouement ont été essentiels pour mener à bien ce mémoire.*

*Nous remercions également chaleureusement le président du jury, « **Malika OUACIFI** » pour avoir accepté de présider ce jury et pour ses précieux commentaires qui ont contribué à enrichir ce travail.*

*Nos remerciements vont également à « **Chaima AOUICHE** » pour avoir pris le temps d'examiner ce mémoire et pour ses remarques constructives qui ont permis d'améliorer la qualité de notre PFE.*

Enfin, nous tenons à exprimer notre gratitude à tous ceux qui nous ont soutenus, directement ou indirectement, dans cette aventure académique.

Merci à tous

Dédicace

Je dédie ce modeste travail de fin d'étude à mes chers parents « Salim » et « Ilhem » ; qui ont sacrifié leur vie pour ma réussite et m'ont éclairé le chemin par leurs conseils judicieux. J'espère qu'un jour, je pourrai leur rendre un peu de ce qu'ils ont fait pour moi, que Dieu leur prête bonheur et longue vie.

Pour mon cher frère « Med Salah » et ma chère sœur « Sara », votre présence a été ma plus grande force. Merci d'être toujours à mes côtés, je vous aime.

Pour ma chère grand-mère « Chama », votre amour a enrichi ma vie de manière inestimable que Allah le tout-puissant la protége.

Pour ma chère tante « Djalila », votre gentillesse, générosité et votre cœur d'or sont une vérité qui restera toujours gravée. Merci pour tout ce que vous avez fait pour moi.

Pour mes chères cousines « Ines, Saoussen, Rofia », merci pour les moments de rires et de complicité qui ont illuminé ma vie. Votre amitié m'a apporté une immense joie.

Pour mon cher oncle « Hamdi », Merci pour votre compréhension et vos encouragements durant toutes mes années d'études.

Pour toutes les personnes que j'aime, votre affection et votre soutien ont été les piliers de ma motivation et de ma force. Merci du fond du cœur.

Maroua Abdelouahad

Dédicace

"À ma chère mère, tu resteras à jamais gravée dans mon cœur. Ton amour et ta sagesse continueront de m'inspirer à chaque étape de ma vie. En ce moment de transition, je te rends hommage avec tendresse et gratitude. Repose en paix, maman." Mes pensées t'accompagnent en cette période difficile.

Merci cher papa, pour tout ton soutien inconditionnel et tes encouragements qui m'ont aidé à atteindre ce moment spécial de ma vie. Ta présence et ton amour ont été les piliers de ma réussite. Je suis reconnaissante pour tout ce que tu as fait pour moi. Merci, papa, pour être mon inspiration constante.

Chères sœurs Ala Chaima Douha Ghoufran Anfel, vous êtes des personnes merveilleuses et je tiens à vous dire à quel point vous êtes importantes pour moi. Merci pour votre soutien et l'amour que vous me donnez toujours.

Chère tante Dalila Samet, ta bienveillance et ton amour inconditionnel illuminent nos vies. Ta présence chaleureuse est un trésor précieux. Merci d'être une source de joie et de réconfort. Tu es vraiment spéciale pour nous.
Ma chère grand-mère Benaarfa Fadila, Nous t'aimons énormément et nous sommes reconnaissants pour tout ce que tu fais pour nous.
Mes chers amis, Avec vous, chaque instant est rempli de rires, de souvenirs précieux et de soutien inconditionnel. Merci d'être là pour moi, et merci d'être les meilleurs amis que l'on puisse rêver d'avoir. Vous êtes les véritables trésors de ma vie.

Khaoula Amrani

Table des Matières

Chapitre I : Apprentissage par renforcement

Chapitre II : Réseaux de Neurones et Apprentissage Profond

Chapitre III : Apprentissage par Renforcement Profond

Chapitre IV : Application De L'apprentissage Par Renforcement profond Dans Le Marche Boursier

Liste Des Figures

Liste Des Tableaux

INTRODUCTION GENERALE

L'intelligence artificielle (IA) est devenue une force motrice majeure de l'innovation technologique au 21ème siècle, transformant divers secteurs de l'économie et de la société. En permettant aux machines d'effectuer des tâches qui nécessitent normalement une intelligence humaine, comme la reconnaissance d'images, la compréhension du langage et la prise de décision, l'IA ouvre des perspectives nouvelles et prometteuses.

Au cœur de l'IA se trouve l'apprentissage automatique (Machine Learning, ML), une discipline qui permet aux machines d'apprendre à partir des données. Les algorithmes de ML peuvent identifier des modèles et faire des prédictions basées sur des ensembles de données historiques. Parmi les branches les plus dynamiques du ML, l'apprentissage par renforcement (Reinforcement Learning, RL) se distingue par sa capacité à résoudre des problèmes complexes nécessitant une prise de décision séquentielle.

L'apprentissage par renforcement est une technique où un agent est capable de prendre des décisions en interagissant avec un environnement. L'agent reçoit des récompenses ou des punitions en fonction de ses actions et vise à maximiser le cumul de ses récompenses sur le long terme. Cette approche est particulièrement efficace pour des tâches telles que la navigation autonome, le contrôle de robots et les jeux vidéo.

L'essor du Deep Learning, qui utilise des réseaux de neurones profonds pour modéliser des relations complexes dans les données, a donné naissance au Deep Reinforcement Learning (DRL). Le DRL combine les principes du RL et du Deep Learning pour créer des agents capables de gérer des environnements à haute dimension et de résoudre des problèmes d'une complexité inégalée.

Un des algorithmes pionniers du DRL est le Deep Q-Network (DQN), développé par DeepMind. Le DQN utilise un réseau de neurones profond pour approximer la fonction de valeur Q, qui évalue la qualité des actions possibles dans un état donné. Ce modèle a démontré des performances remarquables dans des environnements complexes, comme les jeux Atari, ce qui est parfait pour le trading et les marchés financiers.

Pour améliorer les performances et la stabilité du DQN, plusieurs variantes ont été proposées :

- **Double DQN** : Cette variante corrige la surestimation des valeurs Q en utilisant deux réseaux distincts pour sélectionner et évaluer les actions, ce qui conduit à des décisions plus précises.

- **Dueling DQN** : Introduit une architecture où la valeur d'un état et l'avantage des actions sont séparés, permettant au modèle de mieux distinguer les actions optimales et d'améliorer l'efficacité de l'apprentissage.

CHAPITRE 01

APPRENTISSAGE PAR RENFORCEMENT

I.1 Introduction :

L'apprentissage par renforcement (RL) représente l'une des méthodes les plus efficaces en intelligence artificielle pour résoudre les défis complexes de l'interaction entre un agent et son environnement. Inspiré par les mécanismes d'apprentissage observés dans la nature, le RL fournit un cadre conceptuel et des outils algorithmiques permettant à un agent d'apprendre à prendre des décisions successives pour maximiser sa récompense cumulative dans un contexte souvent changeant et incertain. Dans cette section, nous examinerons les bases de l'apprentissage par renforcement, les algorithmes clés qui lui sont associés, ainsi que ses différentes applications et les défis qu'il pose.

Dans ce chapitre, nous explorerons les principes fondamentaux de l'apprentissage par renforcement, les algorithmes clés associés à cette discipline, ainsi que ses applications potentielles et ses défis à relever. L'objectif est de fournir un aperçu complet de cette méthode d'apprentissage dynamique et adaptative qui continue de susciter un intérêt croissant dans la communauté de l'intelligence artificielle [1].

I.2 Composantes de l'apprentissage par renforcement :

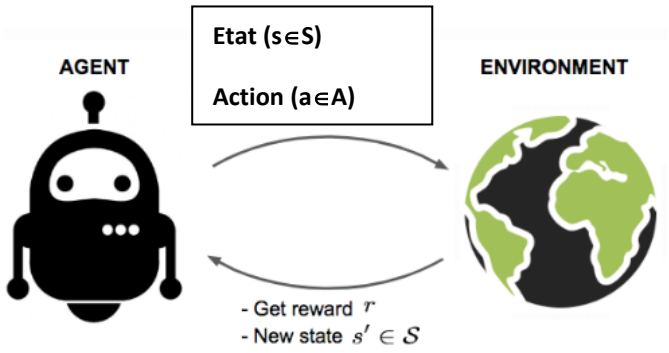

Figure I.1 : L'interaction agent-environnement dans l'apprentissage par renforcement. [https://www.clariba.com/machine-learning-for-busines]

- **Agents (A)** : Les agents sont un concept central dans le domaine du RL. Un agent RL est une entité autonome ou un système informatique qui interagit avec un environnement pour apprendre et prendre des décisions afin d'atteindre des objectifs spécifiques ou de maximiser les récompenses cumulées. Ces agents peuvent être implémentés dans divers domaines, de la robotique aux jeux en passant par les systèmes de recommandation. Par exemple, un agent peut être un joueur du jeu dont le

but est de gagner la partie ou d'obtenir un maximum de récompenses.

- **Environnement (E)** : L'environnement est un système bien défini et structuré avec lequel un agent RL interagit au fil du temps. Il représente le monde ou le domaine dans lequel les actions de l'agent ont des conséquences et fournit un retour d'information à l'agent en fonction de ces actions.

- **États (S)** : L'environnement a un ensemble d'états possibles qui représentent toutes les configurations ou situations que l'environnement peut prendre. Cet ensemble peut être fini ou infini, discret ou continu, selon la nature de l'environnement. Les États fournissent des informations sur le contexte actuel de l'environnement.

- **Actions (A)** : L'environnement définit un ensemble d'actions autorisées que l'agent RL peut entreprendre. Les actions sont les décisions prises par l'agent pour influencer l'état de l'environnement.

- **Récompenses (R)** : Après chaque action, l'environnement fournit à l'agent RL un signal numérique appelé récompense. La récompense indique l'opportunité ou la qualité immédiate de l'action entreprise. L'objectif de l'agent est souvent de maximiser la récompense cumulée au fil du temps.

- **Condition de fin** : certaines tâches RL ont une condition de fin spécifique, qui définit la fin d'un épisode (une séquence d'interactions). Cela pourrait être basé sur l'atteinte d'un état particulier, d'un certain nombre de pas de temps ou d'autres critères.

- **Observations** : L'agent peut avoir accès à des observations de l'environnement, qui sont des descriptions partielles ou complètes de l'état actuel. Les observations ne fournissent pas toujours une vue complète de l'environnement mais sont utilisées par l'agent pour prendre des décisions.

I.3 Fondamentaux de l'apprentissage par renforcement :

Dans le domaine du Reinforcement Learning (RL), Markov Decision Process (MDP), Monte Carlo, Dynamic Programming (DP) et l'équation de Bellman sont des concepts fondamentaux qui interagissent de manière significative [2]:

I.3.1 Le processus décisionnel de Markov (MDP) :

I .3.1.1 Définition : Un Processus de Décision Markovien (MDP) est un modèle mathématique couramment utilisé pour décrire les interactions entre un agent et un environnement dans le cadre de l'apprentissage par renforcement (RL). Il repose sur l'hypothèse de Markov, qui stipule que l'état futur dépend uniquement de l'état présent et de l'action entreprise, indépendamment de l'historique des états et des actions précédentes.

Les MDP définissent les éléments suivants :

- **États (S)** : L'ensemble de tous les états possibles de l'environnement.
- **Actions (A)** : L'ensemble des actions que l'agent peut entreprendre.
- **Probabilités de transition (P)** : Une fonction de transition P qui décrit la probabilité de passer de l'état s à l'état s' en prenant l'action a. Formellement, cela peut être écrit comme :

$$P\left(s_{t+1}, r_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0\right) = P(s_{t+1}, r_{t+1} | s_t, a_t) \quad (1.1)$$

Cela signifie que la probabilité de se retrouver dans l'état s_{t+1} et de recevoir la récompense r_{t+1} dépend uniquement de l'état actuel s_t et de l'action a_t .

- **Récompenses (R)** : Une fonction de récompense $R(s,a,s')$ qui donne la récompense reçue en passant de l'état s à l'état s' via l'action a.

Dans le cadre des MDP, RL consiste à apprendre une politique à partir d'interactions avec l'environnement, en utilisant des méthodes telles que la programmation dynamique, la méthode Monte Carlo et les méthodes de gradient temporel.

I .3.1.2 Les Etapes De Processus :

1. L'agent et l'environnement interagissent à chacune d'une séquence d'étapes de temps discrètes, t = 0, 1, 2, 3,....
2. À chaque étape t, l'agent reçoit une représentation de l'état de l'environnement, St ∈ S, et sur cette base sélectionne une action, At ∈ A(s).

3. Un pas plus tard, en partie à la suite de son action, l'agent reçoit une récompense numérique, Rt+1 ∈ R , et se retrouve dans un nouvel état, St+1.

Ainsi, le processus est modélisé par le quadruplet {S, A, T, R} un ensemble d'états S. ensemble (continu ou discret) des états de l'environnement que perçoit l'agent ; un ensemble d'actions A, discret ou continu ; une fonction de transition T(s,a,s') représentant la probabilité de se retrouver dans l'état s' en effectuant l'action a depuis l'état figure . Illustration d'un MDP [2].

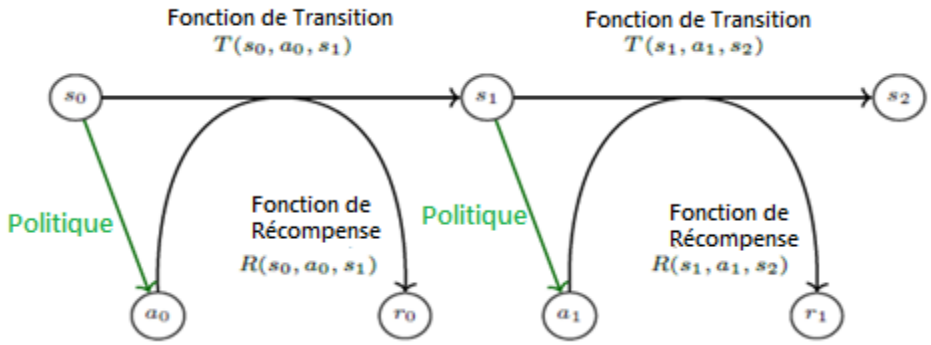

Figure I.2 : Illustration d'un PDM. À chaque étape, l'agent entreprend une action qui modifie son état dans l'environnement et lui procure une récompense [2]

I .3.1.3 Objectif du MDP :

Puisque l'objectif est de maximiser la récompense sur tous les pas de temps - c'est-à-dire le rendement attendu - nous avons également besoin d'un moyen de calculer cette valeur. Naïvement, nous pourrions définir cette récompense en utilisant la somme de la séquence des récompenses :

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ldots + R_T \quad (1.2)$$

Cependant, pour certaines tâches où l'agent entreprend continuellement des actions sans état final, cette approche échouera car la somme de toutes les récompenses tendra vers l'infini lorsque T = ∞. Cela empêcherait l'agent de distinguer les actions qui produisent des récompenses plus importantes plus rapidement. C'est pourquoi nous introduisons un facteur

d'actualisation γ ($0 \leq \gamma \leq 1$) qui est utilisé pour pondérer l'importance des récompenses futures par rapport aux récompenses immédiates.

Le rendement total attendu est alors calculé comme suit :

$$G_t = R_t + \gamma^1 R_{t+1} + \gamma^2 R_{t+2} + \cdots = \sum_{K=0}^{\infty} \gamma^K R_{t+k} \qquad (1.3)$$

Si l'agent connaissait la séquence précise de récompenses qu'il obtiendrait, aucun calcul supplémentaire ne serait nécessaire. Cependant, les valeurs des récompenses à chaque étape dépendent à la fois de l'état de l'environnement et de l'action choisie par l'agent. Pour cette raison, nous définissons une fonction de valeur d'action qui nous aide à approximer les valeurs de paires état-action spécifiques. Cette fonction de valeur d'action est souvent appelée fonction Q, et les valeurs que nous obtenons à partir d'elle sont appelées valeurs Q.

La fonction $Q(s, a)$, représente la valeur attendue de prendre une action a dans un état s, et de suivre ensuite une politique donnée pour les étapes futures. Elle aide l'agent à évaluer l'utilité de chaque action dans chaque état, permettant ainsi de prendre des décisions optimales.

I.3.2 Fonctions de valeur *:* C'est une fonction qui estime la valeur d'un état ou d'une action, c'est-à-dire à quel point il est bénéfique pour l'agent de se trouver dans cet état ou de prendre cette action. Elle est utilisée pour évaluer et comparer les différentes actions ou états, guidant ainsi le processus de prise de décision de l'agent

- **Fonction de valeur de l'état $V^{\pi}(s)$** : Elle estime le retour attendu à partir de l'état s en suivant toujours la politique π. Formellement :

$$V^{\pi}(s) = E_{\tau \sim \pi}\,[G_t \mid s_t = s] = E_{\tau \sim \pi}\,[\sum_{K=0}^{\infty} \gamma^K R_{t+k} \mid s_t = s] \quad (1.4)$$

L'espérance $E_{\tau \sim \pi}$ prend en compte toutes les trajectoires possibles que l'agent peut suivre en commençant à s_t et en suivant les actions dictées par π.

- **Fonction de valeur de l'action $Q^{\pi}(s, a)$** : Elle estime le retour attendu à partir de l'état s, en prenant une action a, puis en suivant toujours la politique π. Formellement :

$$Q^{\pi}(s, a) = E_{\tau \sim \pi}\,[G_t \mid s_t = s, a_t = a] \qquad (1.5)$$

I.3.3 Équations de Bellman : L'équation de Bellman est une relation récursive importante dans le RL qui décompose la valeur d'un état en deux parties : la récompense immédiate obtenue en quittant cet état $r(s, a)$; et la valeur escomptée des états futurs atteints en suivant la politique de décision de l'agent :

- **Pour la fonction de valeur de l'état $V\pi(s_t)$** :

$$V^{\pi}(s_t) = R_t + \gamma(R_{t+1} + \gamma R_{t+2} + \cdots) = V^{\pi}(s) = r(s, a) + V^{\pi}(s_{t+1}) \quad (1.6)$$

où $R_t = r(s,a)$ est la récompense immédiate obtenue en prenant l'action $\boldsymbol{a}$ à l'état $\boldsymbol{s}$, et $\boldsymbol{\gamma}$ est le facteur de discount.

- **Pour la fonction de valeur de l'action $Q\pi$ (s_t, a_t)** :

Où (s, a) est la récompense immédiate obtenue en prenant l'action $\boldsymbol{a}$ à l'état $\boldsymbol{s}$,

et $V*(s_{t+1})$ est la valeur optimale de l'état suivant après avoir pris l'action optimale $\boldsymbol{a_t}$, et $\boldsymbol{\gamma}$ est le facteur de discount.

$$Q^{\pi}(s_t, a_t) = R_t + \gamma(R_{t+1} + \gamma R_{t+2} + \cdots) = r(s, a) + \gamma \cdot V^{*}(s_{t+1}) \quad (1.7)$$

La solution de l'équation d'optimalité de Bellman n'est pas linéaire ; en général, il n'y a pas de solution en forme fermée, les équations d'optimalité de Bellman constituent un outil puissant pour l'apprentissage par renforcement, mais leur résolution devient impraticable pour les grands espaces d'états. Heureusement, la programmation dynamique (DP) offre une solution efficace pour contourner ce défi.

I.4 Approches de l'apprentissage par renforcement :

Dans cette section, nous examinons plusieurs approches de l'apprentissage par renforcement (RL), un domaine puissant de l'intelligence artificielle où un agent apprend à prendre des décisions en interagissant avec son environnement et en recevant des récompenses. Avant d'explorer des algorithmes spécifiques, il est crucial de comprendre les principes fondamentaux et les différentes approches de base du RL. En explorant ces approches, nous pourrons mieux appréhender la diversité des techniques utilisées dans ce domaine. Cela nous préparera à comprendre en profondeur les

algorithmes spécifiques qui seront présentés par la suite, nous permettant ainsi d'apprécier la variété des méthodes disponibles pour résoudre une multitude de problèmes en RL.

En examinant cette taxonomie, on peut mieux comprendre les différences entre les approches et choisir celle qui convient le mieux à un problème spécifique en RL. Cela permet également d'identifier les tendances et les avancées récentes dans le domaine, facilitant ainsi la recherche et le développement de nouvelles méthodes et algorithmes.

La figure suivante présente une taxonomie des algorithmes d'apprentissage par renforcement [3]. Elle peut inclure des categories tells que:

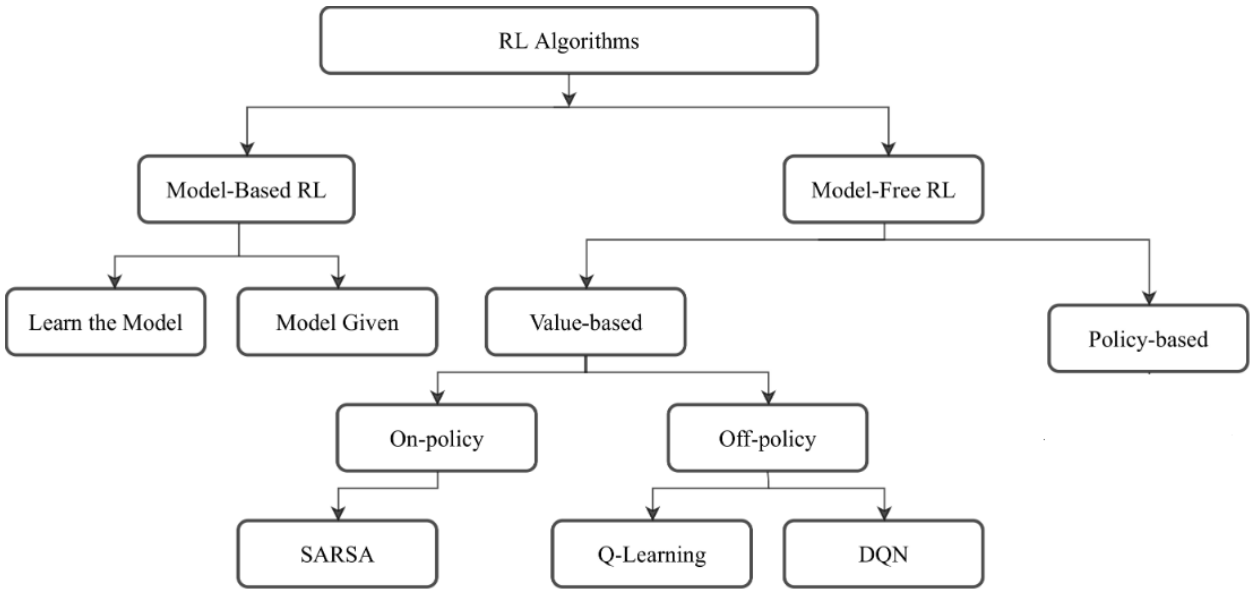

Figure I.3: les approches des algorithmes d'apprentissage par renforcement [22]

I.4.1 L'apprentissage basé sur un modèle (Based-model Learning) :

L'apprentissage basé sur un modèle (Based-model Learning) est une approche de l'apprentissage automatique où un agent utilise des connaissances préalablement acquises pour accomplir des tâches. Cette méthode se distingue principalement par l'utilisation d'un modèle qui décrit les dynamiques de l'environnement, c'est-à-dire comment les actions de l'agent affectent l'état de l'environnement et les récompenses qu'il peut obtenir. Ce type d'apprentissage peut être subdivisé en deux catégories principales : "Learn the Model" et "Given Model" [4].

I.4.1.1 Learn the Model : L'agent doit explorer et estimer les dynamiques de l'environnement à partir de ses interactions. Une fois le modèle construit, l'agent utilise ce modèle pour planifier et prendre des décisions optimales. L'agent estime les transitions d'état et les récompenses, puis utilise ces estimations pour trouver la politique optimale en utilisant DP.

I.4.1.2 Given Model : L'agent dispose d'un modèle a priori des dynamiques de l'environnement, ce qui lui permet de se concentrer directement sur l'optimisation de sa politique en utilisant les méthodes de DP.

I.4.2 L'apprentissage sans modèle (Free-model Learning) :

Dans ce type, l'agent se base uniquement sur l'expérience d'essais et d'erreurs (exploration) pour déterminer les bonnes actions à entreprendre. Cette approche est divisée en deux branches principales :

I.4.2.1 Méthodes basées sur des politiques : l'agent apprend directement la politique qui détermine les actions à prendre dans chaque état sans passer par une estimation explicite des valeurs des états ou des actions. Ils utilisent principalement des techniques de Monte Carlo pour estimer les gradients de la politique à partir de trajectoires d'épisodes complets.

I.4.2.2 Méthodes basées sur la valeur : Dans cette approche, l'agent apprend à évaluer les actions en termes de valeur, généralement représentée par une fonction d'action-valeur $(s,)$ et il est divise en deux sous branches :

- **Les algorithmes on-policy :** sont des algorithmes qui utilisent la politique courante de l'agent pour prendre des décisions et pour mettre à jour sa connaissance de l'environnement en apprenant à partir des actions sélectionnées par celle-ci. Cela signifie que l'agent suit sa politique courante pour explorer l'environnement et pour apprendre à partir de ces expériences. Un exemple courant d'algorithme on-policy est le Sarsa en utilisant l'Apprentissage par Différence Temporelle (TD).
- **Les algorithmes off-policy :** sont des algorithmes qui utilisent à l'inverse une politique différente de celle de l'agent, se qui permet de prendre des décisions plus éclairées et de collecter des données de comportement plus diversifiées afin de mettre à jour sa connaissance de l'environnement. Cela signifie que l'agent suit une politique différente de celle qu'il utilise pour explorer l'environnement. Un exemple courant d'algorithme off-policy est le Q-learning [9]

I.5 Les techniques et les algorithmes de résolution du MDP

Plusieurs algorithmes sont utilisés pour résoudre les MDP, chacun avec des approches différentes pour trouver la politique optimale. Ici, nous aborderons trois méthodes principales : la programmation dynamique, les méthodes de Monte Carlo et l'apprentissage par différence temporelle [5].

I.5.1 Programmation dynamique (DP)

Les méthodes de programmation dynamique nécessitent un modèle complet de l'environnement. Les deux principaux algorithmes DP sont :

- **Itération de valeur** : cet algorithme met à jour de manière itérative la fonction de valeur *(V(s))* en fonction de l'équation de Bellman jusqu'à ce qu'elle converge vers la fonction de valeur optimale.

$$V(s) \leftarrow max_a \sum_{s'} P(s'|s,a) \ [r + \gamma V(s')] \qquad (1.8)$$

À partir de la fonction de valeur optimale, la politique optimale (π^*) peut être dérivée en choisissant des actions qui maximisent les récompenses attendues.

- **Itération de la politique** : cela implique deux étapes : l'évaluation de la politique et l'amélioration de la politique. L'évaluation de la politique calcule la fonction de valeur pour une politique donnée, tandis que l'amélioration de la politique met à jour la politique pour qu'elle soit gourmande par rapport à la fonction de valeur actuelle.

I.5.2 Méthodes de Monte Carlo

Les méthodes de Monte Carlo sont sans modèle et apprennent la fonction de valeur sur la base d'échantillons d'épisodes d'expérience. Ils nécessitent des épisodes complets pour mettre à jour la fonction de valeur.

- **First-Visit Monte Carlo** : estime la valeur d'un État comme la moyenne des rendements suite à la première visite dans l'État.

- **Every-Visit Monte Carlo** : estime la valeur d'un État comme la moyenne des rendements après chaque visite dans l'État.

I.5.3 Apprentissage par différence temporelle (TD)

Les méthodes TD combinent les idées des méthodes DP et Monte Carlo. Ils apprennent directement de l'expérience brute sans avoir recours à un modèle de l'environnement et mettent à jour les estimations en fonction d'autres estimations apprises.

- **TD(0)** : met à jour la fonction de valeur en utilisant la récompense actuelle et la valeur estimée de l'état suivant.

$$V(s) \leftarrow V(s) + \alpha\,(r + \gamma\, V(s') - V(s)) \quad (1.9)$$

- **Q-Learning** : Un algorithme RL sans modèle qui apprend directement la fonction action-valeur ($Q(s,a)$). Les valeurs Q sont mises à jour à l'aide de l'équation de Bellman :

$$Q(s, a) \leftarrow \quad Q(s, a) + \alpha\,(r + \gamma\, max\_a'\, Q(s', a') - Q(s, a)) \quad (1.10)$$

I.6 Conclusion :

L'apprentissage par renforcement est un processus d'essais et d'erreurs où une IA prend des décisions dans un environnement pour maximiser une récompense cumulée. Ce processus est modélisé par des processus de décision markoviens (MDP), qui fournissent une structure pour résoudre des problèmes de prise de décision. La qualité de la solution est mesurée par la récompense cumulée actualisée, et l'objectif est d'apprendre une politique optimale qui mappe les états aux actions. Les algorithmes d'apprentissage par renforcement se divisent en deux catégories : basés sur un modèle et sans modèle. Des méthodes comme la programmation dynamique, les méthodes de Monte Carlo et l'apprentissage par différence temporelle sont utilisées pour trouver des politiques optimales dans des environnements incertains.

CHAPITRE 02

Réseaux de Neurones et Deep Learning

II.1 Introduction :

Les réseaux de neurones (RN) sont des modèles d'apprentissage automatique. Ils offrent une approche novatrice pour résoudre des problèmes complexes en imitant le fonctionnement du cerveau. Grâce à leur capacité à traiter l'information et à leur inspiration des cellules nerveuses, ils sont devenus essentiels dans divers domaines tels que l'industrie, la finance, les télécommunications et l'informatique. L'apprentissage en profondeur, une branche des RN, reproduit la structure et la fonction du cerveau en utilisant des réseaux de neurones artificiels, offrant ainsi des solutions avancées pour des tâches de reconnaissance, de prévision et de modélisation.

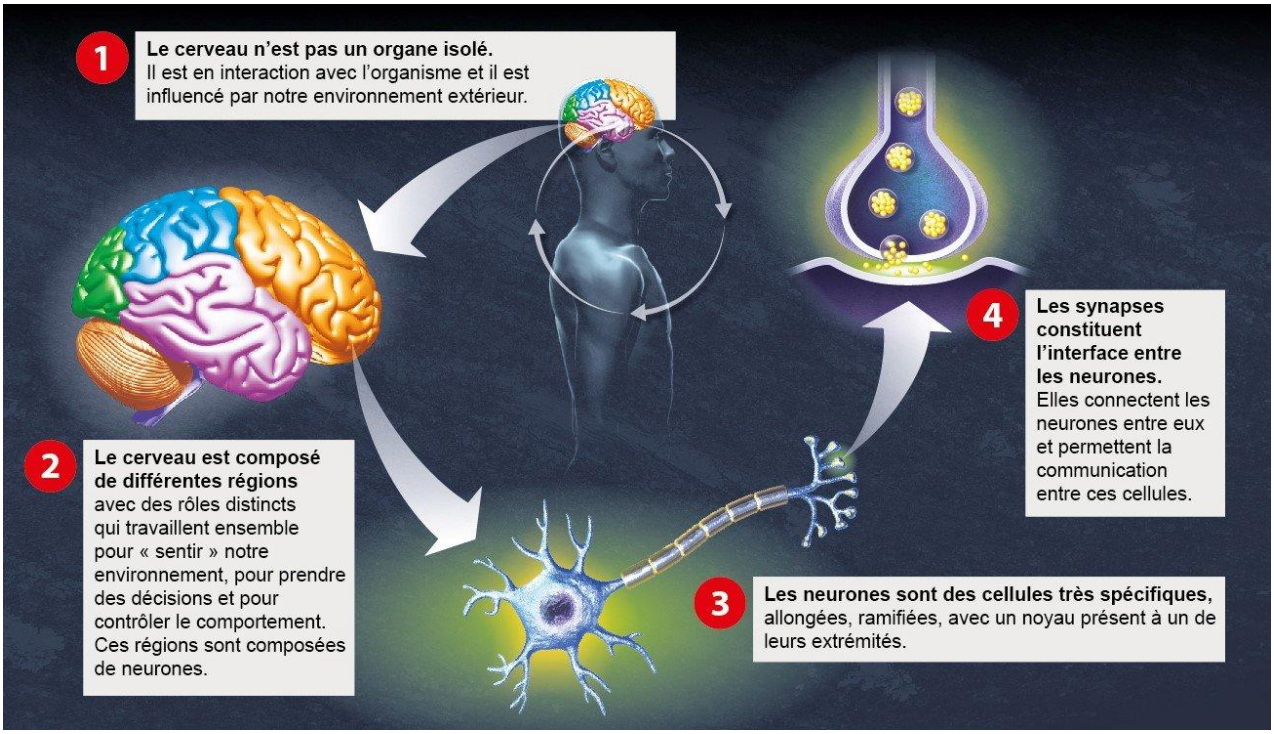

Figure II.1 : réseau de neurones et fonction du cerveau

[https://www.neurosciences.academy/acces-neurosciences-academy-10/]

Les modèles d'apprentissage en profondeur fonctionnent en couches et un modèle typique comporte au moins trois couches. Chaque couche accepte les informations de la précédente et les transmet à la suivante.

II.2 L'apprentissage en profondeur :

Au cours des années 80 et 90, de nombreux progrès significatifs ont été réalisés dans le domaine des réseaux de neurones, mais leur adoption a été freinée en raison du temps et des données nécessaires pour obtenir des résultats satisfaisants. Cependant, avec l'explosion de la puissance de calcul au début des années 2000, de nouvelles techniques de calcul ont vu le jour, donnant ainsi naissance à l'apprentissage en profondeur. Ce dernier a rapidement émergé comme un concurrent sérieux dans le domaine de l'apprentissage automatique, remportant plusieurs concours importants. Depuis 2017 l'intérêt pour l'apprentissage en profondeur a continuellement augmenté, et il est maintenant devenu indispensable à tous les niveaux de l'apprentissage automatique.

Aujourd'hui, l'apprentissage en profondeur est au cœur de nombreuses avancées dans le domaine de la science des données, avec des résultats remarquables dans des domaines tels que la robotique, la reconnaissance d'images et l'intelligence artificielle (IA). Ce chapitre présentera les bases de l'apprentissage en profondeur.

II.3 Neurones artificiels :

Un réseau de neurones est un ensemble d'algorithmes inspirés par le cerveau humain. Le but de cette technologie est de simuler l'activité du cerveau humain, et plus spécifiquement la reconnaissance de motifs et la transmission d'informations entre les différentes couches de connexions neuronales.

Le cerveau est capable d’effectuer des calculs complexes, et c’est de là que vient l’inspiration pour les réseaux de neurones artificiels. Le réseau dans son ensemble est un puissant outil de modélisation.

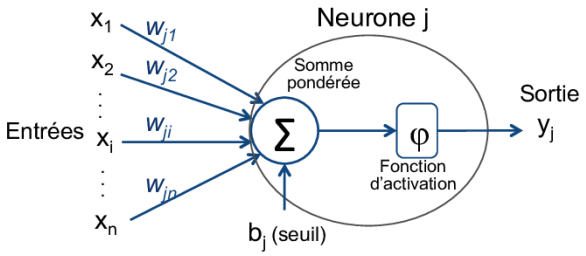

Figure II.2: La structure d'un neurone artificiel [7]

II.3.1 Le Perceptron :

Le perceptron est un type de modèle de neurone artificiel qui a été introduit dans les années 1950 par Frank Rosenblatt. C'est la forme la plus simple de réseau de neurones et est souvent utilisée comme bloc de construction dans des architectures de réseaux de neurones plus complexes.

Le perceptron prend en entrée un vecteur de données, applique des poids à ces entrées, et effectue la somme de ces valeurs pondérées. Ensuite, il applique une fonction d'activation pour produire une sortie. Traditionnellement, la fonction d'activation utilisée est une fonction d'échelon, qui renvoie 1 si la somme pondérée dépasse un certain seuil, sinon elle renvoie 0. Cette sortie peut être considérée comme une classification binaire (0 ou 1).

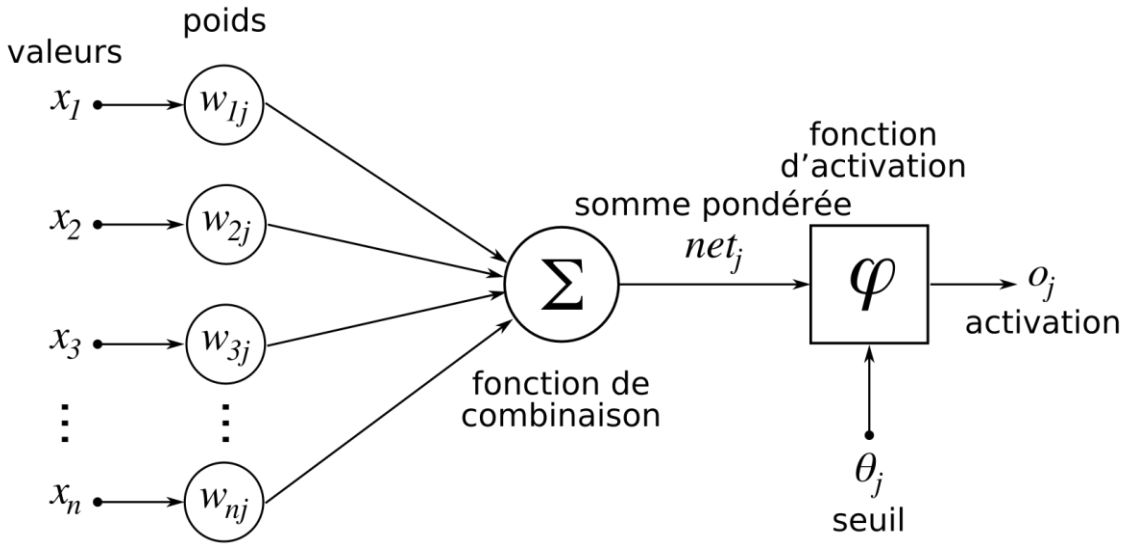

Figure II.3: Les composants d'un Perceptron.

[https://www.researchgate.net/figure/Structure-dun-neurone-artificiel]

Le perceptron est une fonction mathématique qui prend des données d'entrée (x), multiplie ces données par des coefficients de poids (w), et produit une valeur. Si cette valeur est positive, le neurone artificiel s'active. Pour ce faire, la somme pondérée des données d'entrée doit dépasser un seuil. Le résultat prédit est ensuite comparé à un résultat connu, et en cas de différence, l'erreur est rétro propagée pour ajuster les poids.

Selon la règle d'apprentissage du perceptron, l'algorithme apprend automatiquement les poids optimaux.

Il peut également avoir un paramètre supplémentaire appelé biais, qui déplace la fonction d'activation et influence ainsi le succès de l'apprentissage.

II.3.2 Fonction d'activation

Dans le domaine de l'intelligence artificielle, la fonction d'activation joue un rôle crucial dans le fonctionnement des neurones artificiels, et elle peut être considérée comme l'équivalent du "potentiel d'activation" observé dans les neurones biologiques.

La fonction d'activation détermine si un neurone artificiel doit être activé ou non, et dans le cas où il est activé, elle influence également le degré de cette activation. Plusieurs fonctions d'activation existent, chacune offrant des comportements différents.

Par exemple :

- ✓ la fonction sigmoïde produit une sortie comprise entre 0 et 1, ce qui est utile pour modéliser des probabilités ou des non-linéarités douces.
- ✓ La fonction tangente hyperbolique (tanh) produit une sortie entre -1 et 1, permettant une modulation plus prononcée des valeurs d'entrée.
- ✓ La fonction ReLU (Rectified Linear Unit) renvoie simplement la valeur d'entrée si elle est positive, sinon elle renvoie zéro, ce qui introduit une non-linéarité tout en préservant l'efficacité de l'entraînement.

Chaque fonction d'activation offre des avantages et des inconvénients selon le contexte d'application et la nature des données. Le choix de la fonction d'activation peut avoir un impact significatif sur les performances et la convergence des réseaux de neurones artificiels.

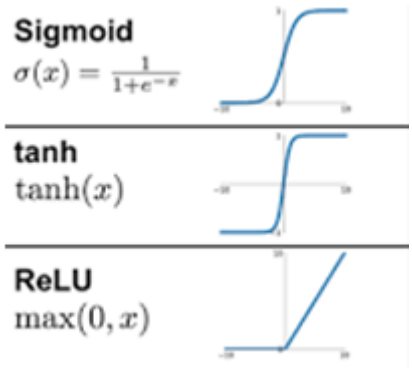

Figure II.4 : Les Fonctions d'activation populaires dans l'apprentissage profond.[6]

II.4 Réseau de neurones artificiels :

Dans un réseau, chaque sous-groupe (couche) fait un traitement indépendant des autres et transmet le résultat de son analyse au sous-groupe suivant. L'information donnée au réseau va donc se propager couche par couche, de la couche d'entrée à la couche de sortie, en passant soit par plusieurs couches intermédiaires (dites couches cachées) ou dans le cas simple sans couches intermédiaires.

Habituellement (sauf pour les couches d'entrée et de sortie), chaque neurone dans une couche est connecté à tous les neurones de la couche précédente et de la couche suivante. Nous distinguons trois types de couches :

- ✓ Couche d'entrée : les neurones de cette couche reçoivent les valeurs d'entrées du réseau et les transmettent aux neurones cachés. Chaque neurone reçoit une valeur, il ne fait pas donc de sommation.
- ✓ Couches cachées : chaque neurone de cette couche reçoit l'information de plusieurs couches précédentes, effectue la sommation pondérée par les poids, puis la transforme selon sa fonction d'activation. Par la suite, il envoie cette réponse aux neurones de la couche suivante.
- ✓ Couche de sortie : Elles produisent les résultats finaux du modèle en transformant les informations provenant des couches cachées en une forme utilisable pour la tâche spécifique

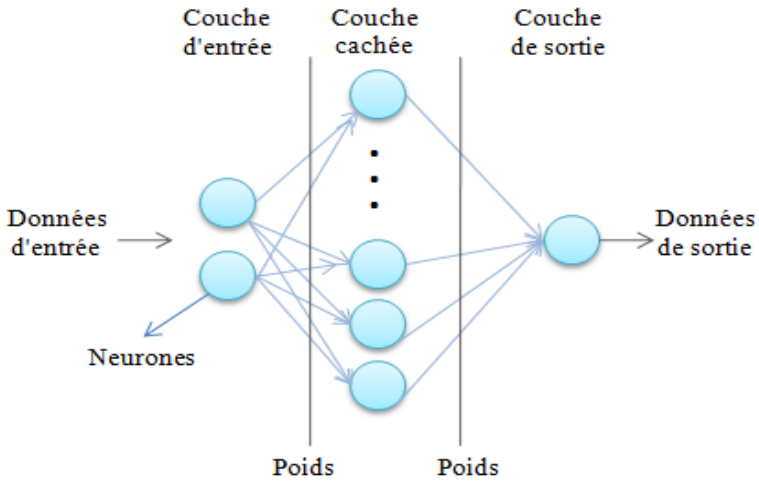

Figure II.5: un réseau de neurone multicouche

[https://europeanwriterstour.com/images-2023/exemples-de-r%C3%A9seaux-de-neurones-artificiels]

II.5 Fonctionnements des réseaux de neurones :

Un réseau de neurones combine plusieurs couches de traitement, utilisant des éléments simples fonctionnant en parallèle et inspirés du système nerveux biologique. Il se compose d'une couche d'entrée, d'une ou de plusieurs couches cachées et d'une couche de sortie. Les couches sont interconnectées par des nœuds, ou neurones, chaque couche utilisant la sortie de la couche précédente en guise d'entrée. Il peut apprendre à partir de données. Il peut ainsi être entraîné sur de nombreux exemples en vue de reconnaître des modèles au niveau de l'image ou de texte par exemple, aussi de classer des données et prédire les cours du marché.

II.5.1 La propagation directe :

La propagation directe dans un réseau de neurones consiste à calculer la sortie du réseau en propageant l'information de la couche d'entrée jusqu'à la couche de sortie. Une fois le réseau de neurones entraîné, ce calcul est effectué en appliquant les fonctions des neurones couche par couche, en passant par toutes les couches intermédiaires.

Pour obtenir une sortie précise, il est essentiel de configurer correctement les paramètres des neurones, notamment les poids. Ces poids sont ajustés de manière à ce que chaque instance des données d'entrée soit associée à une valeur de sortie appropriée.

Lors de la propagation directe, les valeurs de la couche d'entrée sont transmises à travers les fonctions mathématiques de chaque neurone dans les couches suivantes. Ce processus permet au réseau de fournir une valeur de sortie en fonction des données d'entrée.

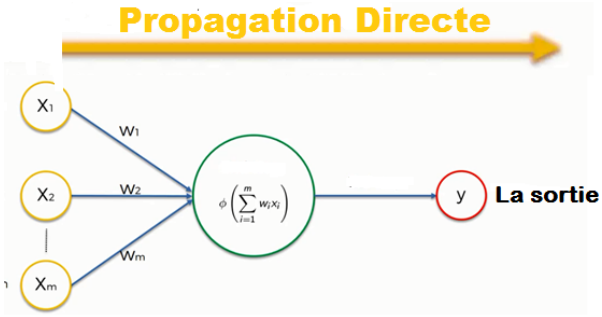

Figure II.6 : Propagation Directe.

II.5.2 La rétro propagation :

La rétropropagation (back propagation) est une méthode standard d'entraînement des réseaux de neurones artificiels. Elle calcule le gradient de la fonction de perte par rapport aux poids du réseau, permettant ainsi de minimiser l'erreur entre la sortie réelle et la sortie cible. Ce processus ajuste les poids des neurones en rétro-propageant l'erreur depuis les couches de sortie vers les couches d'entrée. Pendant l'entraînement, la rétropropagation consiste à:

Calculer l'erreur : Comparer la sortie du réseau avec la sortie cible et calculer l'erreur.

Propager l'erreur : Transmettre cette erreur en arrière à travers le réseau, couche par couche, en utilisant les dérivées des fonctions d'activation pour chaque neurone.

Ajuster les poids : Modifier les poids des neurones en fonction des gradients calculés pour réduire l'erreur. Ce processus est répété jusqu'à ce que l'erreur devienne négligeable.

L'entraînement par rétro-propagation permet d'optimiser les paramètres du réseau pour améliorer ses performances sur les données d'entraînement.

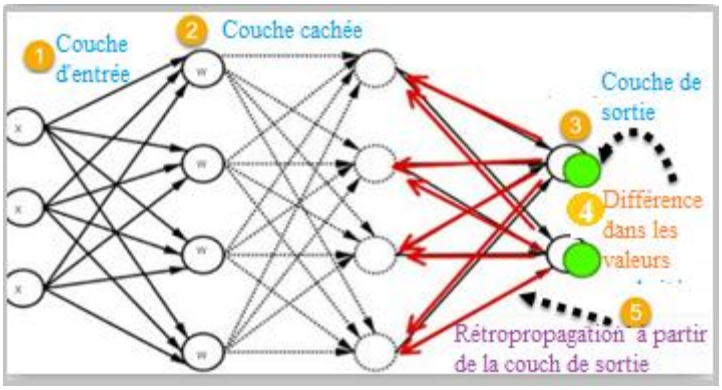

Figure II.7: Rétro-popagation.

.[https://quanteam.fr/reseaux-de-neurones-modeles-et-applications-a-la-finance]

II.5.3 Entraînement :

Dans la phase d'entraînement d'un réseau neuronal, les poids des neurones sont ajustés itérativement pour minimiser l'écart entre la sortie du réseau et les valeurs attendues. Le choix du taux d'apprentissage est crucial, car il détermine la taille des pas effectués lors de la mise à jour des poids. Un taux d'apprentissage élevé peut accélérer l'apprentissage mais risque de provoquer des oscillations ou une divergence, tandis qu'un taux trop bas peut ralentir l'apprentissage.

Les algorithmes d'entraînement, tels que la descente de gradient stochastique (SGD) ou des variantes plus avancées comme Adam ou RMSprop, ajustent les poids du réseau en fonction du gradient de l'erreur par rapport aux poids. Ce gradient indique la pente de la fonction d'erreur par rapport à chaque poids du réseau. En ajustant les poids dans la direction opposée au gradient, l'erreur du réseau est progressivement minimisée.

Le choix de l'algorithme d'entraînement et du taux d'apprentissage dépend du type de données et du problème spécifique à résoudre. Certains algorithmes peuvent mieux fonctionner avec certains types de données ou architectures de réseau que d'autres. Ajuster le taux d'apprentissage est souvent un processus délicat nécessitant de l'expérimentation et des ajustements fins pour obtenir de bons résultats.

Au début de l'entraînement, les poids sont initialisés avec des valeurs aléatoires. Au fur et à mesure que le réseau apprend sur les données d'entrée, il ajuste les poids en fonction des

erreurs de classification résultant des poids précédents. Ce processus itératif se poursuit jusqu'à ce que le réseau atteigne un niveau de performance satisfaisant sur les données d'entraînement.

II.6 Deep Learning (Apprentissage Profond) :

L'apprentissage en profondeur, souvent désigné sous le terme anglais "Deep Learning", est une sous-discipline puissante et en plein essor de l'apprentissage automatique. L'essence Deep Learning réside dans sa capacité à extraire des informations significatives à partir de vastes ensembles de données.

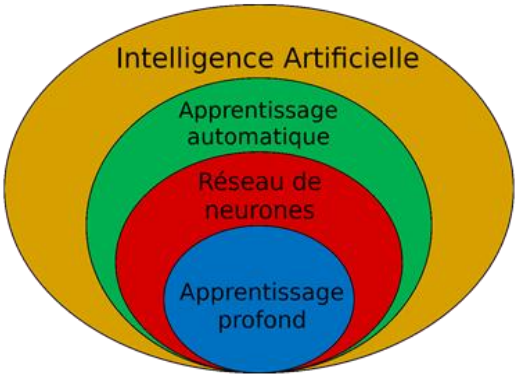

Figure II.8 : Schéma de décomposition du domaine de AI et de ces sous-domaines

[https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/introduction-a-lapprentissage-automatique]

La croissance exponentielle de la quantité de données générées chaque jour - estimée à 2,6 milliards d'octets - constitue la matière première essentielle pour l'apprentissage en profondeur. Cette abondance de données est l'un des moteurs principaux derrière l'essor du Deep Learning ces dernières années. En effet, les algorithmes de Deep Learning s'épanouissent lorsqu'ils sont exposés à des ensembles de données massifs, leur permettant ainsi de découvrir des modèles complexes et des relations subtiles qui échappent souvent aux méthodes traditionnelles d'analyse. Le succès du Deep Learning est également alimenté par l'essor de la puissance de calcul disponible aujourd'hui [8].

Les entreprises, grandes et petites, peuvent Maintenant exploiter les capacités de l'IA, y compris les algorithmes de Deep Learning, sans avoir à supporter les coûts initiaux élevés associés à la mise en place d'infrastructures spécialisées. Cette démocratisation de l'IA a

ouvert de nouvelles opportunités et a permis à un plus grand nombre d'acteurs de bénéficier des avantages de l'apprentissage en profondeur.

II.7 Réseaux de Neurones Convolutifs (CNN) :

II.7.1 principe de CNN : Convolutional Neural Networks en anglais se sont une classe de réseaux de neurones profonds, sont révélés très efficaces pour les tâches impliquant des données étroitement liées, principalement dans le domaine de la vision par ordinateur. Un CNN utilise une structure tridimensionnelle, avec trois ensembles de neurones analysant les trois couches d'une image couleur : le rouge, le vert et le bleu. Il analyse une image une zone à la fois pour identifier les caractéristiques importantes.

La structure du réseau neuronal «entièrement connecté», dans laquelle tous les neurones d'une couche communiquent avec tous les neurones de la couche suivante, est inefficace pour analyser de grandes images. Un CNN utilise une structure tridimensionnelle dans laquelle les neurones d'une couche ne se connectent pas à tous les neurones de la couche suivante, mais chaque ensemble de neurones analyse une petite région ou «caractéristique» de l'image. Le résultat final de cette structure est un seul vecteur de scores de probabilité.

Un CNN effectue d'abord une ***convolution***, qui consiste à «numériser» l'image, à en analyser une petite partie à chaque fois, et à créer une carte de caractéristiques avec des probabilités que chaque caractéristique appartienne à la classe requise (dans un exemple de classification simple). La deuxième étape est ***la mise en commun (pooling),*** qui réduit la dimensionnalité de chaque carte de caractéristique tout en conservant ses informations les plus importantes.

Un réseau CNN possède en général plusieurs couches de convolutions et de pooling ainsi que des couches de correction et une couche entièrement connectée les unes à la suite des autres.

Une architecture CNN contient comme le montre la Figure **II.9** des couches de convolution, de pooling, de correction et une couche entièrement connectée.

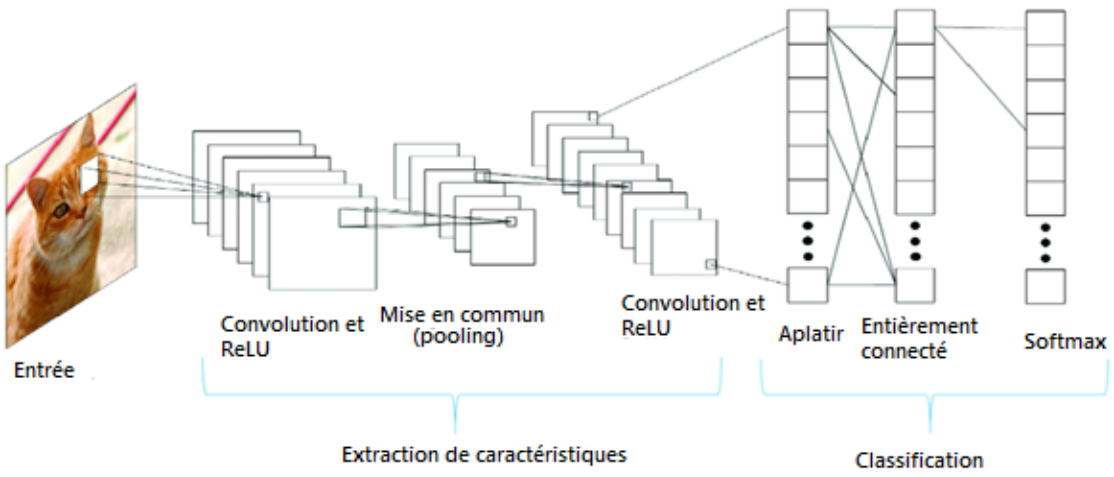

Figure II.9 : Réseau de neurones de convolution (CNN) [15]

Comme illustré ci-dessus, un CNN peut effectuer plusieurs tours de convolution puis de mise en commun. Enfin, lorsque les caractéristiques sont au bon niveau de détail, un réseau de neurones entièrement connecté analyse les probabilités finales et détermine la classe à laquelle l'image appartient. L'étape finale peut également être utilisée pour des tâches plus complexes, telles que la génération d'une légende pour l'image (le processus de créer une phrase ou un texte court qui décrit le contenu ou le contexte de l'image).

CNN est utilisé comme modèle par défaut pour tout ce qui concerne les images. De nos jours, des articles ont mentionné l'utilisation du réseau de neurones récurrents (RNN) pour la reconnaissance d'image (l'analyse de séquences temporelles telles que les séquences vidéo ou les séries temporelles). Les RNN sont traditionnellement utilisés pour la reconnaissance de texte et de parole.

L'utilisation de CNN permet de réduire le nombre de paramètres requis pour les images par rapport au NN normal.

II.7.2 Couche de Convolution

Dans la couche de convolution, au lieu de faire un produit scalaire entre les entrées et les poids de chaque neurone, on applique un produit de convolution. Ce produit de convolution sert à extraire des caractères spécifiques des données en entrée. Une donnée est donc passée à travers une succession de filtres (noyaux de convolution), créant de nouvelles données appelées cartes de convolutions.

Nous considérons une image comme une matrice et le filtre glisse dans la matrice d'image comme indiqué ci-dessous pour obtenir l'image compliquée qui est l'image filtrée de l'image réelle.

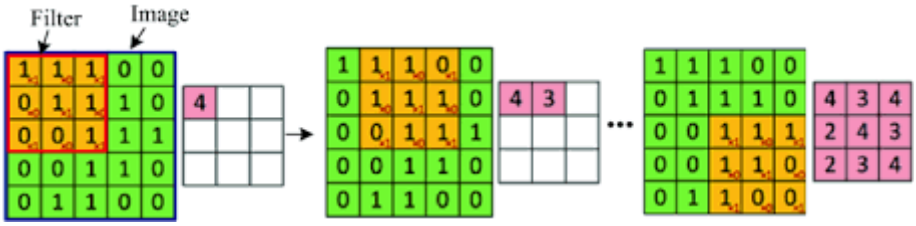

Figure II.10: Extraction des motifs **[21]**

Selon la tâche à effectuer, plusieurs filtres sont disponibles dans le modèle pour prendre en compte la différente fonctionnalité. La valeur de la matrice de filtre est apprise pendant la phase d'apprentissage du modèle.

Trois paramètres permettent de dimensionner le volume de la couche de convolution : la profondeur, le pas et la marge.

- Profondeur de la couche : nombre de noyaux de convolution (ou nombre de neurones associés à un même champ récepteur (surface de traitement)).
- Le pas : contrôle le chevauchement des champs récepteurs. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.
- La marge : permet de contrôler la dimension spatiale du volume de sortie.

II.7.3 Couche de pooling

Cette étape permet de réduire la taille des images d'entrées, de réduire la charge de travail en diminuant les paramètres du réseau et donc la charge de calcul, tout en gardant les principales caractéristiques de l'image. Cela se passe via une fenêtre, qui va glisser pas à pas sur l'ensemble de l'image, et récupérer que certaines valeurs de l'image. Soit en prenant une moyenne des valeurs de pixels de la région analysée par cette fenêtre (meanPool), ou encore le maximum de celles-ci (maxPool), car il existe une multitude de type de pooling.

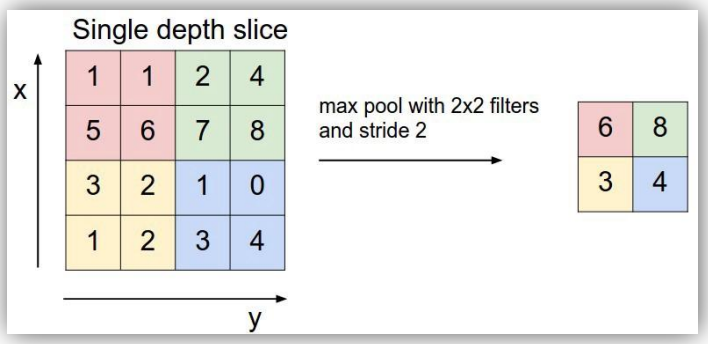

Figure II.11 : Maximum Pooling

[https://patducjacquet.wordpress.com/2017/07/04/cnn-convolution-neural-network-une-introduction/]

II.7.4 Couche de correction (Relu) :

La couche de correction, aussi appelée couche d'activation ReLU (Rectified Linear Unit), en introduisant une non-linéarité (c'est ta dire elle va d'apprendre des relations plus complexes) qui sont essentiels pour de nombreuses tâches d'apprentissage automatique. Elle remplace toutes les valeurs négatives par zéro, tout en laissant les valeurs positives inchangées, et permet d'améliorer l'efficacité du traitement. La couche d'activation ReLU permet de simplifier le traitement.

II.7.5 Couche Flattening :

Cette couche convertit un tableau multidimensionnel en un vecteur unidimensionnel pour préparer les informations à la couche suivante.

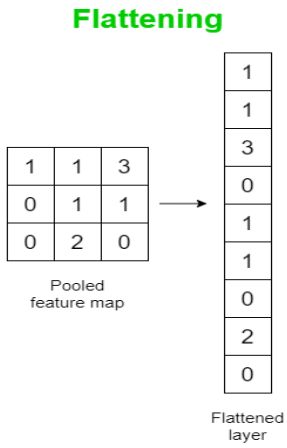

Figure II.12 : Flattening

.[https://towardsdatascience.com/convolutional-neural-network-cnn-architecture-explained-in-plain-english-using-simple-diagrams]

II.7.6 Couche entièrement connectée (Fully Connected Layer) :

Dans une couche entièrement connectée, chaque neurone est connecté à tous les neurones de la couche précédente pour former une représentation globale des données pour réaliser des tâches telles que la classification.

II.7.7 Fonction Softmax:

La fonction Softmax calcule la distribution des probabilités de l'événement sur N différents événements. En règle générale, cette fonction calcule les probabilités de chaque classe cible sur

toutes les classes cibles possibles. Plus tard, les probabilités calculées seront utiles pour déterminer la classe cible pour les entrées données.

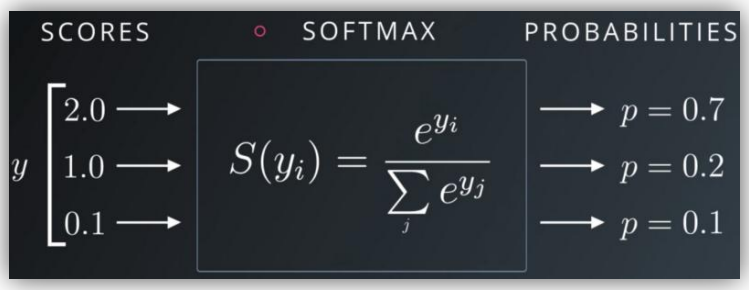

Figure II.13: Fonction Softmax

.[https://dev.to/jai00271/cnn-general-terms-and-their-meaning-42nc]

II.8 Conclusion :

Les réseaux neuronaux (RNA) et les réseaux de neurones convolutifs (CNN) sont des piliers essentiels de l'apprentissage profond, apportant des avancées significatives dans des domaines tels que la vision par ordinateur, le traitement du langage naturel et bien d'autres. Les RNA offrent une polyvalence inégalée pour modéliser des relations complexes entre les données, tandis que les CNN se spécialisent dans le traitement des images en exploitant leur structure spatiale. Ensemble, ils ont révolutionné l'intelligence artificielle et continuent de transformer notre compréhension et notre utilisation des données, ouvrant la voie à de nouvelles applications innovantes et améliorations technologiques.

CHAPITRE 03

Apprentissage par Renforcement Profond

III.1 Introduction :

L'objectif principal de l'apprentissage par renforcement est de trouver la politique optimale, c'est-à-dire la meilleure action à entreprendre dans un état donné pour maximiser la récompense attendue. Pour ce faire, on utilise des fonctions de valeur, telles que la fonction de

valeur d'état (V(s)) et la fonction de valeur d'action (Q(s, a)), qui prennent en compte les états et les actions possibles.

Le Q-learning est une méthode d'apprentissage par essais et erreurs où l'agent explore l'environnement, choisit des actions, évalue les récompenses et met à jour les valeurs Q en conséquence. L'équation de Bellman, qui exprime la valeur Q comme la somme de la récompense immédiate et de la récompense future actualisée, joue un rôle central dans ce processus d'amélioration continue.

Avec l'apprentissage par renforcement profond (Deep Reinforcement Learning, DRL), des réseaux neuronaux sont utilisés pour approximer les fonctions de valeur, permettant de traiter des environnements avec des espaces d'états et d'actions beaucoup plus vastes et complexes. Cette combinaison a conduit à des avancées significatives, ouvrant la voie à des solutions plus intelligentes, efficaces et capables de résoudre des problèmes autrefois considérés comme insolubles [9].

III.2 Q-Learning :

Le Q-learning est un algorithme d'apprentissage par renforcement sans modèle qui optimise une fonction de valeur Q en utilisant l'équation de Bellman.

Au cœur du Q-learning, la Q-table joue un rôle central. Cette matrice comporte des lignes correspondant à diverses actions et des colonnes correspondant à divers états ou caractéristiques environnementales. L'agent utilise les valeurs Q de ce tableau comme des estimations des récompenses cumulées attendues qui sont associées à l'exécution d'une action spécifique dans un état particulier [9].

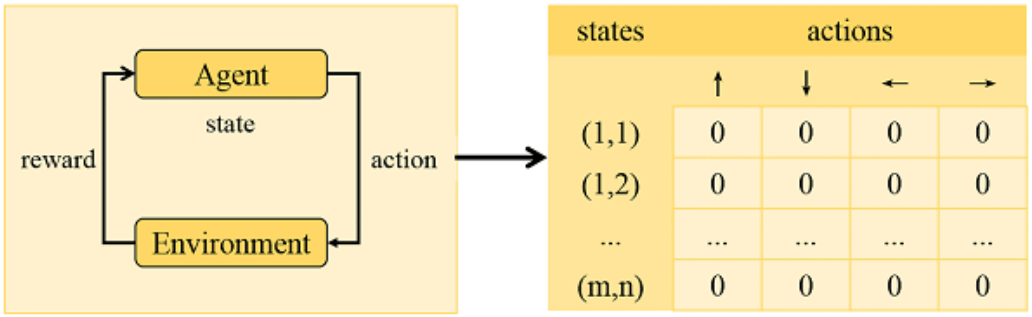

Figure III.1 : principe du Q-Learning.

III.2.1 Fondements théoriques

- **Fonction de valeur Q :** La fonction de valeur Q, notée *Q*(*s*, *a*), représente la valeur d'une paire état-action. Elle est définie comme la valeur attendue cumulée des récompenses futures lorsque l'agent est dans un état *s* et choisit une action *a*.
- **Équation de Bellman :** L'équation de Bellman pour la fonction de valeur Q exprime la relation entre la valeur d'une paire état-action et la valeur des états actions successifs qui peuvent être atteint-en suivant la politique optimale. Elle est définie comme suit :

$$Q(s, a) = \sum_{s'} P(s'|s, a) \left[R(s, a, s') + \gamma \cdot max_{a'} Q^*(s', a')\right] \qquad (3.1)$$

Où:

- $\boldsymbol{S}$ est. l'état actuel.
- $\boldsymbol{a}$ est l'action choisie dans cet état.
- $\boldsymbol{s'}$ est l'état suivant.
- $\boldsymbol{P(s'|s, a)}$ est la probabilité de passer de l'état $\boldsymbol{s}$ à l'état $\boldsymbol{s'}$ en effectuant l'action $\boldsymbol{a}$.
- $\boldsymbol{R(s, a, s')}$ est la récompense immédiate reçue après avoir effectué l'action $\boldsymbol{a}$ et être passé de l'état $\boldsymbol{s}$ à l'état $\boldsymbol{s'}$
- γ est le facteur de réduction temporelle, également appelé taux de remise.
- $\boldsymbol{max_{a'} Q^*(s', a')}$**:** Cela représente la récompense cumulative attendue maximale future à partir de l'état s′ en considérant toutes les actions possibles $\boldsymbol{a'}$**.**

Cette équation exprime la relation entre la valeur d'une action dans un état donné et la valeur attendue de toutes les actions possibles dans l'état suivant, en tenant compte des récompenses immédiates et futures attendues, ainsi que des probabilités de transition entre les états. C'est une formulation clé utilisée dans de nombreux algorithmes d'apprentissage par renforcement

III.2.2 Algorithme Q-learning

Le processus de Q-learning peut être décrit comme suit :

1. **Initialisation de la Table Q :** Une table Q est initialisée avec des valeurs arbitraires pour chaque paire état-action.
2. **Mise à jour de la valeur Q :** la mise à jour est effectuée à l'aide d'une équation mathématique qui prend en compte la valeur Q actuelle, la récompense immédiate

reçue et la valeur Q maximale pour l'état suivant. Cette équation affine de manière itérative les valeurs Q tout au long du processus d'apprentissage, aidant ainsi l'agent à prendre des décisions plus éclairées au fil du temps. Essentiellement, la mise à jour vise à équilibrer les récompenses immédiates avec les récompenses futures attendues, en guidant l'agent dans l'apprentissage de stratégies optimales pour naviguer dans son environnement.

- Dans l'algorithme de Q-learning en utilisant la formule de mise à jour de l'équation de Bellman :

$$Q(s, a) \leftarrow (1-\alpha) \cdot Q(\text{s, a}) + \alpha \cdot (r + \gamma \cdot max_{a'} Q(s', a')) \qquad (3.2)$$

Où

$\boldsymbol{\alpha}$ est le taux d'apprentissage qui contrôle la vitesse à laquelle les nouvelles informations remplacent les anciennes estimations de la valeur d'action Q(s,a).

3. **Politique optimale :** pour chaque état S, la politique optimale choisit l'action a qui maximise la fonction de valeur d'action Q(s, a).

III.2.3 Table Q :

Une table Q ou une matrice est créée lors de l'exécution du Q-learning. Le tableau suit la paire état et action, c'est-à-dire [s, a], et initialise les valeurs à zéro. Après chaque action, la table est mise à jour et les valeurs q sont stockées dans la table. L'agent RL utilise cette table Q comme table de référence pour sélectionner la meilleure action en fonction des valeurs q.

Q Table: γ = 0.95

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
⇧	0.2	0.3	1.0	-0.22	-0.3	0.0
⇩	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
⇨	0.21	0.4	-0.3	0.5	1.0	0.0
⇦	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Figure III.2 : mise à jour de la table Q lors du calcul de la valeur de l'action d'état par l'agent.

[https://towardsdatascience.com/qrash-course-deep-q-networks-from-the-ground-up-1bbda41d3677]

III.2.4 Dilemme Exploration vs Exploitation

Le problème fondamental du dilemme exploitation-exploration en apprentissage par renforcement est de décider si l'agent vaut mieux exploiter les connaissances et les actions déjà connues pour maximiser les récompenses à court terme ou explorer de nouvelles actions pour découvrir des stratégies plus efficaces à long terme.

Le Q-learning doit résoudre le dilemme exploration vs exploitation. Pour ce faire, plusieurs approches sont utilisées :

- **Epsilon-greedy :** Introduit un facteur d'exploration contrôlé par epsilon.
- **Softmax :** Utilise les valeurs Q pour calculer la probabilité de choisir chaque action.
- **Decay epsilon-greedy :** (basé sur ε-greedy) : cet algorithme, à chaque étape, diminue l'epsilon de façon exponentielle, comme suit :

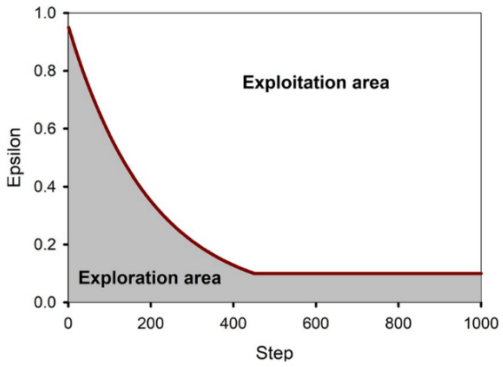

Figure III.3: la sélection d'action epsilon-greedy.
[https://www.researchgate.net/figure/Epsilon-greedy-method-At-each-step-a-random-number-is-generated-by-the-model-If-the_fig2_334741451]

Le Q-learning est un puissant algorithme d'apprentissage par renforcement qui utilise la fonction de valeur Q et l'équation de Bellman pour apprendre efficacement à agir dans un environnement. En gérant le dilemme exploration-exploitation de manière appropriée, il peut converger vers une politique optimale. Le Q-learning est largement utilisé et constitue une base importante pour de nombreuses applications d'apprentissage par renforcement tel que :

III.2.5 Les Limites du Q-Learning traditionnel :

Bien que le Q-learning soit un algorithme puissant d'apprentissage par renforcement, il présente plusieurs limites qui entravent son efficacité dans des environnements plus complexes :

- **Problèmes d'évolutivité :**

Le Q-Learning traditionnel maintient une table Q dans laquelle chaque paire état-action est mappée à une valeur Q. À mesure que l'espace d'états s'agrandit, en particulier dans les environnements continus ou de grande dimension, la table Q devient peu pratique, ce qui entraîne une inefficacité de la mémoire et un processus d'apprentissage lent.

- **Espaces d'états et d'actions discrets :**

Q-Learning fonctionne bien avec des environnements où les états et les actions sont discrets et finis. Cependant, de nombreux problèmes du monde réel impliquent des espaces d'état et d'action continus, que le Q-Learning traditionnel ne peut pas gérer efficacement sans discrétiser ces espaces, ce qui peut entraîner une perte d'informations et des politiques sous-optimales.

Pour relever ces défis, une approche alternative consiste à combiner le Q-learning avec des réseaux de neurones profonds. Cette approche est baptisée Deep Q-Learning (DQL). Les réseaux de neurones dans DQL agissent comme une approximation de la valeur Q pour chaque paire (état, action).

III.3 Deep Q-Network :

En 2013, DeepMind a introduit l'algorithme Deep Q-Network (DQN) et L'article qui le présente :"Playing Atari with Deep Reinforcement Learning", DQN est conçu pour apprendre à jouer aux jeux Atari Il s'agit d'une avancée majeure dans le domaine de l'apprentissage par renforcement et a contribué à ouvrir la voie à de futurs développements dans ce domaine. Deep Q-network est un terme qui se réfère à la combinaison d'un réseau neuronal profond (Deep neural network en anglais) avec l'algorithme Q-learning dans le domaine de l'apprentissage par renforcement [10].

Le réseau neuronal reçoit l'état en entrée et génère les valeurs Q pour toutes les actions possibles. La figure suivante illustre la différence entre le Q-learning et le Q-learning profond dans l'évaluation de la valeur Q :

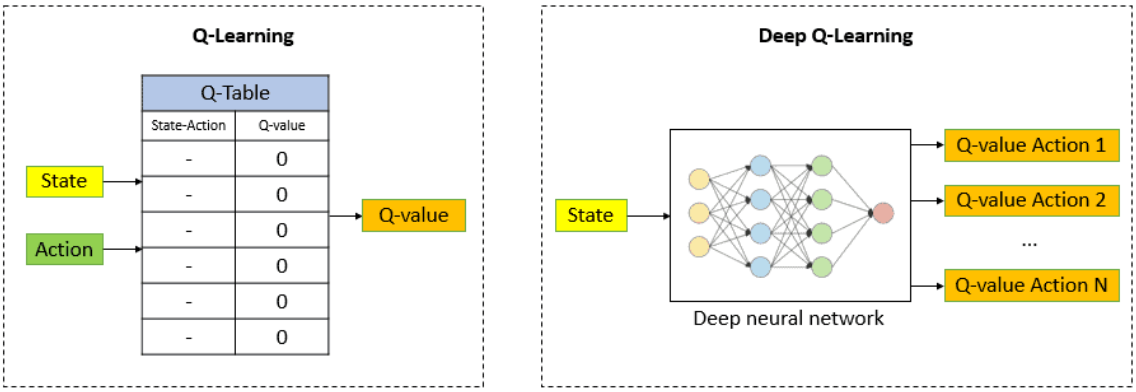

Figure III .4 : Q-learning et Q-learning profond dans l'évaluation de la valeur Q.

[https://www.baeldung.com/cs/q-learning-vs-deep-q-learning-vs-deep-q-network]

Essentiellement, le Q-Learning profond remplace la Q-table habituelle par le réseau neuronal. Plutôt que de mapper une paire (état, action) à une valeur Q, le réseau neuronal mappe les états d'entrée à des paires (action, valeur Q).

En réalité, cet algorithme utilise deux **réseaux de neurones profonds (DNN)** pour stabiliser le processus d'apprentissage

a) Le premier est appelé **réseau neuronal principal**, représenté par le vecteur de poids **θ**, et il est utilisé pour estimer les valeurs Q pour l'état actuel **s** et l'action **a** : **Q(s, a; θ)** en temps réel
b) Le second est le **réseau neuronal cible**, paramétré par le vecteur de poids **θ′,** et il aura exactement la même architecture que le réseau principal, mais il sera utilisé pour estimer les valeurs Q du prochain état s' et de l'action a' .

Tout l'apprentissage a lieu dans le réseau principal. Le réseau cible reste figé (ses paramètres restent inchangés) pendant quelques itérations puis les poids du réseau principal sont copiés dans le réseau cible, transférant ainsi les connaissances apprises de l'un à l'autre. Cela rend les estimations produites par le réseau cible plus précises après la copie.

III.3.2 Equation de Bellman et la fonction de perte pour l'algorithme DQN :

III.3.2.1 l'équation de mise à jour de la valeur Q dans le réseau principal :

$$Q(s, a; \theta) \leftarrow Q(s, a; \theta) + \alpha\,(r + \gamma max_{a'} Q(s', a'; \theta') - Q(s, a; \theta)) \quad (1.10)$$

- **Q(s, a; θ):** Représente la valeur Q estimée pour l'état "s", l'action "a" et les paramètres du réseau neuronal "θ".
- **α:** Représente le taux d'apprentissage.
- **r:** Représente la récompense immédiate.
- **γ:** Représente le facteur d'actualisation.
- **max_a' Q (s', a'; θ'):** Représente la valeur Q maximale attendue pour l'état suivant "s'" et toutes les actions possibles "a'", estimée par le réseau cible avec les paramètres « θ' ».
- **Q(s, a; θ):** Représente la valeur Q actuelle de l'état-action actuel, estimée par le réseau principal avec les paramètres "θ".

L'équation de Bellman a maintenant cette forme, où les fonctions Q sont paramétrées par les poids de réseau θ et θ′.

III.3.2.2 la fonction de perte :

Afin de former un réseau neuronal, nous avons besoin d'une **fonction de perte (ou de coût)**, qui est définie comme la différence au carré entre les deux côtés de l'équation de Bellman, dans le cas de l'algorithme DQN :

$$L(\theta) = E\left[(r + \gamma \max_{a'} Q(s',a';\theta') - Q(s,a;\theta))^2\right] \qquad (3.3)$$

C'est la fonction que nous minimiserons en utilisant la descente de gradient, qui peut être calculée automatiquement à l'aide d'une bibliothèque Deep Learning telle que TensorFlow ou PyTorch.

Alors la fonction de miss à jour des poids :

$$\theta \leftarrow \alpha(r + \gamma \max_{a'} Q(s',a';\theta') - Q(s,a;\theta))\nabla_\theta Q(s,a;\theta) \qquad (3.4)$$

Remarque : les mises à jour des poids du réseau cible ne sont pas effectuées directement à travers cette formule de mise à jour des poids.

III.3.3 Réseau cible :

Le réseau cible est une copie du réseau Q, qui est utilisé pour se rapprocher de la fonction Q.

Le réseau cible maintient un vecteur de poids distinct θ', les poids du réseau cible ne sont pas mis à jour à chaque itération. Au lieu de cela, ils sont copiés périodiquement à partir du réseau d'évaluation, créant un décalage temporel entre les deux réseaux.

Le choix d'un réseau cible distinct rend la divergence peu probable car cela ajoute un délai entre le moment où la valeur Q de réseau principale est mise à jour et le moment où les valeurs Q de cibles sont mises à jour, Une fois les poids copiés, les paramètres du réseau cible restent fixes jusqu'à la prochaine mise à jour. Cela signifie que le réseau cible utilise les mêmes poids pour estimer les valeurs cibles pendant une certaine période, souvent appelée période de gel Cela permet de stabiliser le processus d'apprentissage en évitant les fluctuations brusques dans l'apprentissage.

Le réseau cible joue un rôle crucial en fournissant des valeurs Q cibles stables pour guider l'apprentissage du réseau principal.

III.3.4 Fonctionnement du Q-Learning

Voici les étapes de fonctionnement de DQN [11] :

- **Environnement** : agent DQN interagit avec un environnement avec un état, un espace d'action L'objectif du DQN est d'apprendre la politique optimale qui maximise les récompenses cumulées au fil du temps.
- **Mémoire de relecture** : l'agent DQN utilise une mémoire tampon de relecture pour stocker les expériences passées. Chaque expérience est un tuple (état, action, récompense, état suivant) représentant une transition unique d'un état à un autre. La mémoire de relecture stocke ces expériences pour les échantillonner ultérieurement en fournissant une source de données diversifiée et en permettant à l'agent d'apprendre à partir d'expériences passées de manière répétée et efficace.
- **Réseau neuronal profond** : DQN utilise un réseau neuronal profond pour estimer les valeurs Q pour chaque paire (état, action). Le réseau neuronal prend l'état en entrée et génère la valeur Q pour chaque action. Le réseau est formé pour minimiser la différence entre les valeurs Q prédites et cibles.
- **Exploration Epsilon-Greedy** : DQN utilise une **stratégie d'exploration epsilon-greedy** pour équilibrer l'exploration et l'exploitation. Pendant la formation, l'agent

sélectionne une action aléatoire avec une probabilité epsilon et sélectionne l'action avec la valeur Q la plus élevée avec une probabilité (1 – epsilon)

- **Réseau cible** : DQN utilise un réseau cible distinct pour estimer les valeurs Q cibles. Le réseau cible est une copie du réseau neuronal principal avec des paramètres fixes. Le réseau cible est mis à jour périodiquement pour éviter la surestimation des valeurs Q.
- **Entrainement** : DQN entraîne le réseau neuronal à l'aide de l'équation de Bellman pour estimer les valeurs Q optimales. La fonction de perte est l'erreur quadratique moyenne entre les valeurs prédites et cibles. La valeur Q cible est calculée à l'aide du réseau cible et de l'équation de Bellman. Les poids du réseau neuronal sont mis à jour par rétro propagation et descente de gradient stochastique
- **Tests** : DQN utilise la politique apprise pour prendre des décisions environnementales après la formation. L'agent sélectionne l'action avec la valeur Q la plus élevée pour un état donné

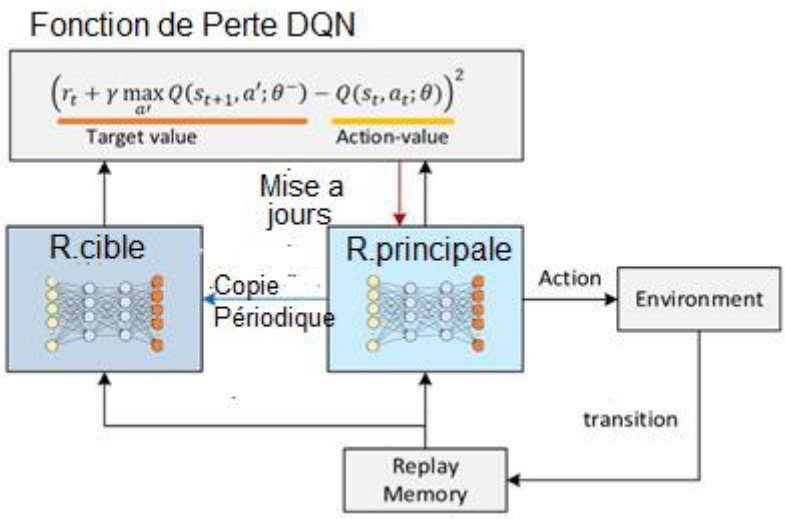

Figure III.5 : structure de réseau cible et réseau Q (DQN) [11]

III.3.5 Implémentation :

1- Pour implémenter l'algorithme DQN, nous commencerons par créer les DNN principal et cible. Le réseau cible sera une copie du réseau principal, mais avec sa propre copie des poids. Nous aurons également besoin de définit une fonction de perte pour mesurer l'écart entre les valeurs Q prédites et les valeurs Q cibles.

2- Ensuite, nous créerons le tampon (REPLAY MEMORY) de relecture d'expérience, pour ajoute (stocké) l'expérience au tampon et l'échantillonner plus tard pour la formation d'apprentissage (pour entraîner le réseau principal en utilisant les données stockées dans le tampon.)
3- Nous écrirons également une fonction contrôle pour exécuter la politique ε-greedy, nous définirons également les hyper-paramètres nécessaires et entraînerons le réseau de neurones.
4- Alors Nous commençons avec un épisode en utilisant la politique ε-greedy, stockerons les données dans le tampon de relecture d'expérience et entraînerons le réseau principal après chaque étape. Nous diminuerons également la valeur d'epsilon « ε » (en commencer avec une exploration élevée) t pour diminuerons l'exploration au fil du temps, et périodiquement, les paramètres du réseau principal sont copiés vers le réseau cible.
5- Nous verrons comment l'algorithme commence à s'entrainer après chaque épisode maintenant l'agent a appris à maximiser la récompense.

III.4 Double apprentissage profond Q

L'un des inconvénients de l'algorithme DQN est qu'il surestimerait les véritables récompenses ; les valeurs Q pensent que l'agent va obtenir un rendement supérieur à celui qu'il obtiendrait dans la réalité. Cela peut conduire à une surestimation des valeurs Q, car le réseau a tendance à favoriser les actions qu'il prédit comme étant les meilleures [12].

Cette surestimation est due à :

a) Utilisation du même réseau pour l'estimation et la sélection d'action
b) Présence de la valeur Q maximale surestimée dans l'équation de mise à jour

Pour éviter cela, nous utiliserons tous les deux réseaux : celui que nous utiliserons pour effectuer la sélection actions et un autre pour évaluation des actions

III.4.1 le principe d'algorithme Double DQN :

Le réseau neuronal principal **décide** quelle est la meilleure action suivante parmi toutes les actions suivantes disponibles, puis le réseau neuronal cible **évalue** cette action pour connaître sa valeur Q. Cette technique résout le problème de la surestimation dans DQN.

III.4.2 Différence entre le DQN et DDQN :

- ✓ **DQN (Deep Q-Network)** :

- La mise à jour de la fonction Q dans le Deep Q-learning network est basée sur l'équation de Bellman pour les valeurs Q :

$$Q(s, a; \theta) \leftarrow Q(s, a; \theta) + \alpha\,(r + \gamma\, max\_a'\, Q(s', a'; \theta') - Q(s, a; \theta)) \quad (3.5)$$

- La mise à jour des poids du réseau dans le DQN se fait en minimisant une fonction de perte qui mesure la différence entre les valeurs Q prédites et les valeurs cibles.
  Les valeurs cibles sont calculées en utilisant l'équation de Bellman :

$$(r + \gamma\ max_{a'}\ Q(s',a' ;\theta^{-})$$

  et la fonction de perte est :

$$L(\theta)= E[(r + \gamma\ max_{a'}\ Q(s',a' ;\theta^{-})- Q(s,a ; \theta))^2 \quad (3.6)$$

- ✓ **Double Q-learning** : La mise à jour des valeurs Q dans Double Q-Learning utilise deux réseaux de Q, notés :

Q1 et θ1 : du réseau principal.

Q2 et θ2 : du réseau cible.

- La mise à jour des valeurs Q est basée sur l'équation suivante :

$$Q1(s,a)\leftarrow Q1(s,a)+\alpha[r+\gamma Q2(s',argmax_a\ \ Q1(s',a ;\theta1);\theta2)-Q1(s,a)] \quad (3.7)$$

- La fonction de perte dans l'algorithme double DQN :

$$L(\theta)= E[(r + \gamma\, Q_2\ (s',\ arg\ max_a\ Q_1(s',a ; \theta_1).\ \theta_2)- Q(s,a ; \theta_1))^2] \quad (3.8)$$

Ou : la valeur Q pour l'action *a* dans l'état s

- la récompense immédiate *r* plus la récompense future escomptée, actualisée par le facteur d'actualisation
- $Q_1(s,a;\theta1)$ est la valeur Q prédite par le réseau principal pour l'action *a* dans l'état *s*.
- $Q_2\ (s',\ arg\ max_a\ Q_{11}(s',a ; \theta_1).\ \theta_2)$ est la valeur Q cible, calculée à partir du deuxième réseau et utilisant l'action maximale selon les prédictions du premier réseau, dans l'état suivant s'.

- E : représente l'espérance mathématique, c'est-à-dire la moyenne sur un ensemble d'échantillons d'expérience.

Les mises à jour dans le DQN et le Double DQN consistent en l'ajustement des poids du réseau de neurones en minimisant une fonction de perte, mais avec une différence clé dans le calcul des valeurs cibles, ce qui permet au Double DQN de réduire la surévaluation des valeurs

Les mises à jour des poids : $\theta \leftarrow \theta - \alpha \nabla_\theta [Y - Q(s, a; \theta)]$

$\theta \leftarrow \theta - \alpha \nabla_\theta [(r + \gamma Q_2 (s', arg\ max_a Q_1(s', a ; \theta_1) ; \theta_2) - Q_1 (s, a; \theta)]$ (3.9)

Y c'est Q cible : $(r + \gamma Q_2 (s', argmax_a Q_1(s', a ; \theta_1) ; \theta_2)$

III.4.3 Implémentation :

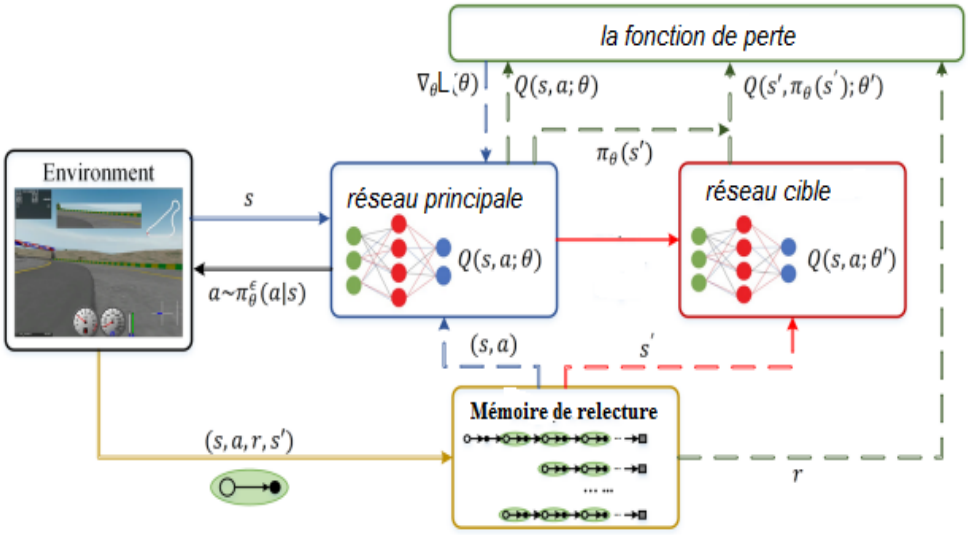

Figure III.6 : Algorithme Double DQN [10]

L'organigramme de l'apprentissage du Double Deep Q Network (DDQN) s'étale sur les étapes suivantes [13]:

1. **Initialisation :**
    - Initialise le réseau de neurones principal (Q-network) avec des poids aléatoires.

- Initialise la mémoire de relecture (replay Memory) pour stocker les transitions expérience.

2. **Collecte des données d'entraînement :**

   - L'agent interagit avec l'environnement.
   - Pour chaque pas de temps, sélectionne une action selon une politique d'exploration (epsilon-greedy) et observe la récompense et l'état résultant.
   - Stocke la transition expérience (état, action, récompense, nouvel état) dans la mémoire de relecture.

3. **Échantillonnage de mini-batch :**

   - À intervalles réguliers, l'agent sélectionne un sous-ensemble aléatoire de transitions, appelé mini-batch, depuis la mémoire de relecture pour effectuer la mise à jour des paramètres du réseau principal.

4. **Calcul de la cible Q :**

Pour chaque transition dans le mini-batch, calcule la cible Q comme suit :

   - Utilise le réseau principal pour estimer la valeur de l'action optimale pour le nouvel état.
   - Utilise le réseau principal pour sélectionner l'action optimale pour le nouvel état.
   - Utilise le réseau cible (une copie gelée du réseau principal) pour estimer la valeur de cette action.

5. **Calculer la fonction de perte**

6. **Mise à jour des poids du réseau :**

   - Utilise un algorithme d'optimisation tel que la descente de gradient stochastique pour mettre à jour les poids du réseau principal.
   - Met à jour périodiquement les poids du réseau cible en copiant les poids du réseau principal.

7. **Répétition :**

- Répète les étapes 2 à 6 pour un nombre fixe d'itérations ou jusqu'à ce que la convergence soit atteinte.

8. **Évaluation :**

    - Périodiquement, évalue les performances de l'agent en testant le réseau appris dans l'environnement sans exploration.

9. **Terminaison :**

    - Arrête l'entraînement lorsque les performances souhaitées sont atteintes ou lorsque le nombre d'itérations prédéfini est terminé.

Ce processus itératif forme la base de l'apprentissage par renforcement profond avec Double DQN, permettant à l'agent d'apprendre efficacement à prendre des décisions dans des environnements complexes.

III.5 Dueling Double Deep Q-Network (DDDQN)

III.5.1 Définition :

En combinant le Double Q-learning avec l'architecture dueling, un Duel DDQN qui utilise deux réseaux de neurones pour estimer séparément la valeur de l'état V et l'avantage de chaque action A.

Les sorties de ces deux réseaux sont ensuite combinées pour obtenir la fonction d'action-valeur totale Q, qui est utilisée pour sélectionner et évaluer les actions dans l'environnement d'apprentissage par renforcement [14].

✓ **Double Q-learning** :

- Dans le Q-learning traditionnel, nous utilisons une seule estimation Q pour évaluer la qualité d'une action dans un état donné. Cependant, cela peut entraîner une surestimation des valeurs d'action.

- Dans le Double Q-learning, nous utilisons deux ensembles de paramètres Q_1 et Q_2 : Pour sélectionner une action dans un état donné s, nous utilisons Q_1 pour choisir l'action optimale $a*$=argmax$aQ_1(s,a)$,

  puis nous évaluons cette action à l'aide de Q_2 : (S$*$)=$Q_2(s,a*)$

  Les mises à jour des valeurs Q_1 et Q_2 se font en alternance pour éviter la corrélation entre les deux estimations.

✓ **Architecture duelling** :

L'architecture Dueling DQN a été développée par Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, et Nando de Freitas. Cette architecture a été présentée dans leur article de recherche intitulé "Dueling Network Architectures for Deep Reinforcement Learning", publié en 2016

Dans l'architecture duelling, la fonction d'action-valeur $Q(s,a)$ est décomposée en deux parties:

- La valeur de l'état $V(s)$
- L'avantage de chaque action $A(s,a)$.

La valeur de l'état $V(s)$ représente à quel point l'état s est précieux, indépendamment de l'action choisie.

L'avantage de chaque action $A(s,a)$ représente à quel point l'action a est meilleure que la moyenne des actions dans l'état s.

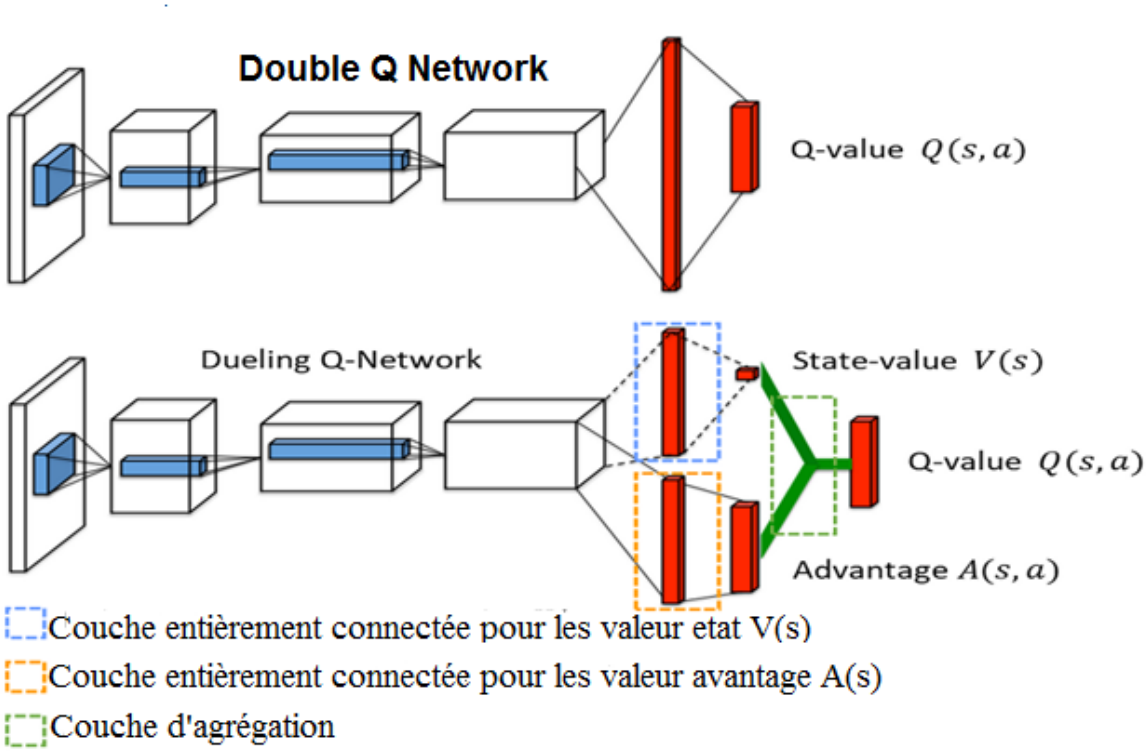

Figure III.7 Image extraite de l'article Dueling DQN

La fonction d'action-valeur totale est alors calculée comme la somme de la valeur de l'état et de l'avantage de chaque action :

$$Q(s,a)=V(s)+A(s,a) \quad (3.10)$$

Pour entraîner le réseau neuronal en additionnant simplement les fonctions de valeur et d'avantage Danc Q=V+A, on ne peut pas déterminer les valeurs de V et A, puisque celles-ci sont « **non identifiables** ».

Pour voir un exemple similaire, si je vous dis que la valeur de Q est 20 et que vous devez savoir quelles deux valeurs totalisent 20 (20 = V + A), il existe une infinité de solutions possibles.

Pour résoudre ce problème, l'article suggère une astuce : **forcer la valeur Q la plus élevée à être égale à la valeur V** , ce qui rend la valeur la plus élevée de la fonction avantage nulle. Cela nous indiquera exactement la valeur de V et nous pourrons en calculer tous les avantages, résolvant ainsi le problème.

$$Q(s, a) = V(s) + A(s, a) - max_{a_i \in |A|} A(s, a_i) \qquad (3.11)$$

Cependant, le document suggère une petite modification à cette procédure. Au lieu de calculer le maximum, il suggère de **le remplacer par la moyenne**, c'est donc ce que nous allons faire et ainsi nous formerons notre réseau :

$$Q(s, a) = V(s) + A(s,a) - \frac{1}{|A|} \sum_{a_i} A(s, a_i) \qquad (3.12)$$

Dueling double deep Q-learning est une variante avancée de l'algorithme Q-learning qui combine les avantages du duel deep Q-learning et du double Q-learning. Il résout le problème du biais de surestimation présent dans les algorithmes Q-learning traditionnels et sépare efficacement les processus d'estimation de la valeur et de sélection d'action.

la fonction Q est décomposée en deux flux distincts : le flux de valeur et le flux Avantage. Le flux de valeur estime la valeur d'être dans un état particulier, tandis que le flux d'avantages estime l'avantage de chaque action par rapport à l'action moyenne. En soustrayant l'avantage moyen de l'avantage spécifique à une action, l'algorithme peut évaluer avec précision l'importance de chaque action.

Pendant le processus de formation, l'agent interagit avec l'environnement, stocke ses expériences dans un tampon de relecture et échantillonne un lot de mémoires pour mettre à jour son réseau Q. L'agent utilise une stratégie de sélection d'actions epsilon-gourmande, qui commence par prendre des actions aléatoires pour explorer l'environnement et évolue progressivement vers la sélection des actions les plus prometteuses en fonction des valeurs Q apprises.

III.5.2 Implémentation :

Processus de formation du Dueling Double Deep Q-Network (DDDQN) :

1. **Initialisation du réseau neuronal** :

   Les deux réseaux de neurones Q_1 et Q_2 sont initialisés avec des poids aléatoires.

2. **Interaction avec l'environnement** :

   - L'agent interagit avec l'environnement en sélectionnant des actions basées sur une politique d'exploration-exploitation, généralement basée sur une stratégie ϵ-greedy.

- À chaque étape, l'agent observe l'état actuel s de l'environnement, sélectionne une action a et reçoit une récompense r de l'environnement.

3. **Stockage des transitions** :

   L'agent stocke la transition actuelle (s,a,r,s') dans une mémoire de relecture (replay memory).

4. **Échantillonnage de minibatch** :

   À intervalles réguliers, un minibatch de transitions est échantillonné aléatoirement à partir de la mémoire de relecture pour la mise à jour des réseaux neuronaux.

5. **Calcul des cibles de mise à jour** :

   Pour chaque transition dans le minibatch, les réseaux $Q1$ et $Q2$ sont utilisés pour calculer les valeurs cibles de mise à jour en utilisant le Double Q-learning. Cela signifie que pour chaque transition (s,a,r,s'), nous utilisons l'un des réseaux pour sélectionner l'action optimale $a*$, puis l'autre réseau pour évaluer cette action dans l'état suivant s'.

   Les valeurs cibles de mise à jour sont calculées comme suit :

   - Pour Q_1, la cible est $r+\gamma Q_2(s',\text{argmax}a'Q_1(s',a'))$.
   - Pour Q_2, la cible est $r+\gamma Q_1(s',\text{argmax}a'Q_2(s',a'))$.

6. **Mise à jour des réseaux neuronaux** :

   - Les réseaux $Q1$ et $Q2$ sont mis à jour en minimisant la différence entre les valeurs Q prédites et les valeurs cibles de mise à jour, utilisant une fonction de perte comme la perte quadratique moyenne.
   - Les poids des réseaux sont ajustés en utilisant une méthode d'optimisation telle que la descente de gradient stochastique (SGD) ou l'optimisation basée sur l'algorithme ADAM.

7. **Répétition des étapes 2 à 6** :

   Ce processus d'interaction avec l'environnement, de stockage des transitions, d'échantillonnage de minibatch, de calcul des cibles de mise à jour et de mise à jour des réseaux neuronaux est répété jusqu'à ce que les performances de l'agent convergent

vers une politique d'action optimale ou jusqu'à ce qu'un critère d'arrêt prédéfini soit atteint (par exemple, un certain nombre d'épisodes).

III.6 Domaines d'application

Le Deep Reinforcement Learning (DRL) a trouvé des applications dans divers domaines réels grâce à sa capacité à apprendre des stratégies complexes et optimisées par l'interaction avec l'environnement. Voici quelques domaines où le DRL est couramment appliqué [15][16]:

- Jeux vidéo et e-Sports : DRL pour jouer à des jeux comme StarCraft II et Dota 2.
- Robotique : Contrôle de robots et robots de service.
- Finance et trading : Trading d'actions et gestion de portefeuilles.
- Transports et logistique : Véhicules autonomes et optimisation des routes.
- Systèmes de recommandation : Personnalisation des recommandations.
- Santé : Diagnostic médical et planification de traitement.
- Énergie : Gestion de l'énergie et maintenance prédictive.
- Publicité en ligne : Achat programmatique et personnalisation des annonces.
- Industrie et fabrication : Automatisation des processus et contrôle de la qualité.
- Environnement et agriculture : Gestion des cultures et surveillance environnementale. Ces exemples illustrent comment le DRL peut résoudre des problèmes complexes et transformer divers secteurs.

Ces exemples montrent comment le DRL peut transformer divers secteurs en offrant des solutions intelligentes et adaptatives à des problèmes complexes, ouvrant ainsi la voie à des innovations et des améliorations continues dans de nombreux domaines.

Dans le prochain chapitre, nous aborderons l'implémentation du Deep Q-Network (DQN) dans le trading financier et mettrons en lumière les avantages clés du DQN par rapport aux méthodes traditionnelles de trading. Notre choix du trading financier comme domaine d'application du Deep Q-Network (DQN) est plus consistant en raison de plusieurs facteurs clés :

Données Abondantes : Le trading financier offre un accès facile à des données historiques abondantes sur les marchés financiers, facilitant ainsi l'entraînement efficace des modèles de DQN.

Critères de Performance Clair : Les performances des stratégies de trading peuvent être facilement mesurées à l'aide de critères clairs tels que le rendement du portefeuille, ce qui facilite l'évaluation de l'efficacité des modèles de DQN.

Marché Dynamique : Les marchés financiers sont dynamiques et compétitifs, offrant de nombreuses opportunités de trading que les modèles de DQN peuvent exploiter pour réaliser des bénéfices.

Potentiel de Rendement Élevé : Le trading financier offre un potentiel de rendement élevé pour ceux qui maîtrisent efficacement les stratégies de trading, ce qui en fait un domaine attrayant pour l'application du DQN.

Recherche Active : Le trading financier est un domaine de recherche actif et en évolution constante, offrant de nombreuses opportunités pour la recherche et l'innovation dans l'application du DQN.

le trading financier présente un ensemble unique de caractéristiques qui en font un choix plus cohérent et prometteur pour l'application du DQN par rapport à d'autres domaines.

III.6 Conclusion :

Dans ce chapitre, nous avons exploré un aperçu de l’apprentissage par renforcement profond, y compris sa définition et son objectif. De plus, nous avons approfondi certains algorithmes d'apprentissage par renforcement importants, à savoir Q-learning et le Deep Q-learning avec ses variants : le DDQN et le DDDQN décrivant leurs concepts et rôles fondamentaux dans le processus de prise de décision.

CHAPITRE 04

APPLICATION DE L'APPRENTISSAGE PAR RENFORCEMENT PROFOND DANS LE MARCHE BOURSIER

IV.1 Introduction :

La bourse représente une opportunité fascinante pour gagner et investir de l'argent, mais elle peut également susciter une cupidité accrue et conduire à des décisions drastiques en raison de la volatilité du marché. Ce contexte instable fait de l'investissement en bourse un pari risqué, où les profits et les pertes sont fréquents. Actuellement, il n'existe pas de modèle de prévision parfait pour les cours boursiers, car les prix sont largement influencés par l'équilibre entre l'offre et la demande.

Dans ce chapitre, nous explorons comment l'apprentissage par renforcement profond (DRL) peut atténuer l'imprévisibilité du marché boursier. Le marché boursier, connu pour sa volatilité et son imprévisibilité, représente un défi majeur pour les investisseurs et les traders. Les méthodes traditionnelles de prédiction et de trading, souvent basées sur des analyses heuristiques et des comportements humains, manquent de précision et sont influencées à des biais irrationnels. L'essor de l'apprentissage par renforcement profond (DRL) offre une opportunité pour surmonter ces limitations et développer des stratégies de trading plus adaptatives et efficaces [17][18][19].

Parmi les approches DRL, le Deep Q-Network (DQN) a démontré un potentiel considérable pour apprendre des stratégies de trading en maximisant les récompenses cumulées. Cependant, le DQN de base présente des limitations, telles que la surestimation des valeurs Q, qui peuvent nuire à la stabilité et à la performance de l'algorithme. Pour pallier ces limitations, des variantes du DQN, notamment le Double DQN et le Duel DQN, ont été développées pour améliorer la robustesse et l'efficacité des algorithmes de trading [20].

Le Double DQN, proposé par Van Hasselt en 2010, dissocie la sélection de l'action du calcul de la valeur Q cible, réduisant ainsi la surestimation des valeurs Q. En utilisant le réseau principal pour sélectionner la meilleure action et le réseau de cibles pour évaluer la valeur Q de cette action, le Double DQN obtient des estimations de valeurs Q plus précises, ce qui améliore la stabilité de l'apprentissage et réduit la variance de la politique.

Le Duel DQN, introduit par Wang et al. en 2016, sépare l'estimation de la valeur de l'état et la fonction d'avantage. Cette architecture permet au réseau d'apprendre de manière plus robuste, car elle met à jour plus fréquemment la valeur de l'état indépendamment des

actions spécifiques. Cela conduit à une meilleure performance globale et à une représentation plus précise des valeurs d'état.

En intégrant ces améliorations, Les algorithmes de trading utilisant le DRL (Deep Reinforcement Learning) deviennent plus robustes face aux variations du marché et meilleurs pour maximiser les gains. L'utilisation de ces versions améliorées du DQN (Deep Q-Network) représente une grande avancée dans la création de systèmes de trading intelligents et adaptables, annonçant une nouvelle ère de trading algorithmique sophistiqué et efficace.

IV.2 Définition du Problème

Dans ce cadre d'apprentissage par renforcement pour la stratégie de trading, l'algorithme prend des actions (achat, vente ou conservation) en fonction de l'état actuel du prix de l'action. L'algorithme est entraîné à l'aide du cadre Deep Q-Learning pour nous aider à prédire la meilleure action basée sur les prix actuels des actions.

IV.2.1 Principaux Composants et Leur Application dans le Trading Boursier

- **État :** Les états peuvent inclure diverses informations de marché comme les prix historiques, les volumes, les indicateurs techniques, les nouvelles économiques, etc.
- **Action** : Les actions sont généralement d'acheter, vendre ou maintenir une position sur un stock.
- **Récompense** : La récompense est souvent basée sur le profit ou la perte résultant d'une action, ajustée pour des facteurs tels que les coûts de transaction et les risques.
- **Entraînement** : L'agent est formé en utilisant des données historiques pour apprendre une politique qui maximise les gains cumulés. L'expérience replay et les cibles différées aident à stabiliser l'entraînement et à améliorer la performance de l'agent en temps réel.

Le concept d'apprentissage par renforcement peut être appliqué à la prévision du cours d'une action spécifique, car il utilise les mêmes principes fondamentaux : il nécessite moins de données historiques et fonctionne dans un système basé sur des agents pour prédire des rendements plus élevés en fonction de l'environnement actuel. Nous verrons

un exemple de prévision du cours d'une action donnée en suivant le modèle d'apprentissage par renforcement, en utilisant le concept d'apprentissage Q.

IV.2.2 Étapes pour Concevoir un Modèle d'Apprentissage par Renforcement

- **Importation des Bibliothèques** : Importer les bibliothèques nécessaires pour le développement du modèle.
- **Création de l'Agent** : Créer l'agent qui prendra toutes les décisions de trading.
- **Définition des Fonctions de Base** : Définir les fonctions nécessaires pour le formatage des valeurs, la fonction d'activation, la lecture des fichiers de données.
- **Formation de l'Agent** : Entraîner l'agent avec les données de marché disponibles.
- **Évaluation des Performances de l'Agent** : Évaluer la performance de l'agent en termes de profit et de perte réalisés.

En suivant ces étapes, nous pouvons concevoir un modèle d'apprentissage par renforcement capable de prédire les actions de trading optimales en fonction des données de marché disponibles.

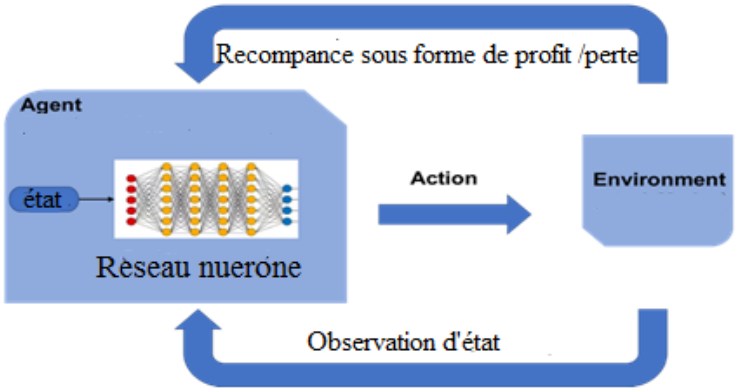

Figure IV.1 : Les composantes du DRL dans le Trading

IV. 3 Importation des données

Il importe des données à partir d'un ensemble de données Kaggle appelé « énorme marché boursier » écrit en Python. Initialement, le code lit les données d'un fichier CSV appelé « goog.us.txt » dans l'ensemble de données. Il lit le fichier CSV et convertit la colonne « Date » en un objet datetime Pandas à l'aide de la bibliothèque Pandas. Les données sont également indexées par la colonne « Date ». Les données sur l'action Apple Inc. (« AAPL

») sont obtenues auprès de Yahoo Finance à l'aide de Pandas DataReader si le fichier spécifié n'est pas trouvé.

Il divise les données en deux parties pour la formation (Train) et les tests. L'index de fractionnement est calculé sur la moitié de la longueur des données et la date correspondante est enregistrée sous le nom « date_split ». Une variable « train » contient des données jusqu'à l'index divisé, tandis qu'une variable « test » contient des données après l'index de division.

La clé d'annotations contient deux dictionnaires, chacun représentant une étiquette de texte au-dessus de la ligne de séparation. Les « données de test » sont ancrées sur le côté gauche de la ligne, tandis que les « données de train » sont ancrées à droite.

Figure IV.2 : Division Des Données En Train et Test.

IV.4 Implémentation

L'algorithme, en termes simples, décide d'acheter, de vendre ou de conserver, lorsqu'il est fourni avec le prix actuel du marché. Cet algorithme est basé sur l'approche "Q-learning" et utilise le réseau neuronal profond Q (Deep Q-Network, DQN) pour élaborer une politique de trading.

ARCHITECTURE : le réseau de neurones est compose de 4 couches : couche d'entrée, 2 couches cachées, couche de sortie avec 3 sorties acheter, vendre et conserver (buy-sel-hold).

FONCTION D'ACTIVATION : dans le premier et le dernier est linéaire, tandis que pour les couches cachées, nous avons utilisé le ReLU.

IV.4.1 Initialisation des Paramètres :

- `BATCH_SIZE` = 50 Taille du lot
- `GAMMA` = 0,77 facteur d'actualisation
- `EPS_START` = 1 epsilon initiale
- `EPS_FINAL = 0, 1` epsilon minimum
- `EPOCH_NUM = 60` nomber d episodes
- `EPSILON_DECREASE = 1e-3` Désintégration d'Epsilon

IV.4.2 Étapes de l'Entraînement

1. **Initialiser l'Agent de Trading** : Initialiser l'agent avec la taille de la fenêtre et du lot.
2. **Lire les Données d'Entraînement** : Lire les données d'entraînement à partir d'un fichier CSV en utilisant une fonction d'assistance

```
2014-03-27 00:00:00 2017-11-10 00:00:00
[3]:
```

Date	Open	High	Low	Close	Volume	OpenInt
2014-03-27	568.00	568.00	552.92	558.46	13052	0
2014-03-28	561.20	566.43	558.67	559.99	41003	0
2014-03-31	566.89	567.00	556.93	556.97	10772	0
2014-04-01	558.71	568.45	558.71	567.16	7932	0
2014-04-02	599.99	604.83	562.19	567.00	146697	0

Figure IV.3 : les données seront affichées avec le nombre de données pour chaque partie (train 446, test 470)

3. **Définir le Nombre d'Épisodes** : Définir le nombre d'épisodes pendant lesquels l'agent va examiner les données. Un épisode représente un passage complet sur les données.
4. **Boucle d'Itération des Épisodes** : Itérer à travers chaque épisode pour entraîner l'agent.
5. **Initialisation de l'État et de l'Inventaire** :
   - Pour chaque épisode, initialiser l'état basé sur les données et la taille de la fenêtre.
   - Initialiser l'inventaire des actions avant de parcourir les données.
6. **Prédiction de l'Action Quotidienne** : À chaque jour de l'épisode, prédire la probabilité de l'action par l'agent.
7. **Exécution des Actions** : Pour chaque jour de trading, l'agent décide d'une action basée sur les données.

   - **Action 1 (Acheter)** : L'agent achète l'action et l'ajoute à l'inventaire.
   - **Action 2 (Vendre)** : L'agent vend l'action, la retire de l'inventaire et calcule le profit ou la perte.

- **Action 0 (Conserver)** : Pas de transaction. L'état est conservé.

8. **Sauvegarde des Détails de l'État** : Sauvegarder les détails de l'état, du prochain état, de l'action, etc., dans la mémoire de l'agent, qui seront utilisés par la fonction **expReplay** pour améliorer l'apprentissage.
9. **Évaluation du Modèle :** Une fois le modèle entraîné, il est crucial de l'évaluer pour interpréter sa performance en conditions réelles. L'évaluation se concentre principalement sur le profit ou la perte généré par le modèle lorsqu'il est testé sur de nouvelles données de marché. Nous évaluons trois variantes de Deep Q-learning : le Deep Q-learning classique, le double DQN et le dueling double DQN .

IV.5 Résultats et Analyse

L'objectif de l'entraînement est de maximiser le profit total en permettant à l'agent de prendre des décisions d'achat, de vente ou de conservation en fonction des états des prix des actions à la fin de chaque journée.

Les couleurs mentionnées dans le graphe des performances font référence aux différentes actions que l'agent peut effectuer dans un environnement de trading.

En français, les couleurs du codage CMYK correspondent aux termes suivants :

1. **Gris (conserver)** : Lorsque l'agent ne prend aucune action (ni achat ni vente).
2. **Bleu clair (Acheter)** : l'agent décide d'acheter des actions.
3. **Rose vif (Vendre)** : l'agent choisit de vendre des actions.

Période de données d'apprentissage (train data): Cette période est utilisée pour entraîner l'agent à reconnaître des patterns et à développer sa stratégie de trading. Les performances durant cette période ne sont pas toujours représentatives de la performance réelle, car l'agent peut encore être en phase d'apprentissage.

Période de données d'évaluation (test data): Cette période sert à tester la stratégie de l'agent en utilisant des données qu'il n'a pas vues pendant l'entraînement. C'est cette période qui montre si l'agent a effectivement appris quelque chose de significatif.

Récompense tronquée (s-reward) : fait référence à la somme des récompenses accumulées par l'agent. Dans ce contexte, les récompenses peuvent être simplifiées à des valeurs de "+1" ou "-1"

Profits : Le **profit** représente le gain final après toutes les transactions effectuées. Cela permet de mesurer le succès global de la stratégie de l'agent en termes de gains ou de pertes financières réels.

IV.5.1 Performance du DQN simple

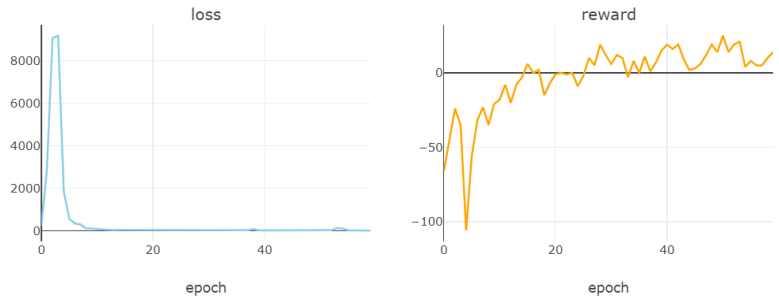

Figure IV.4 : Récompense et perte du DQN

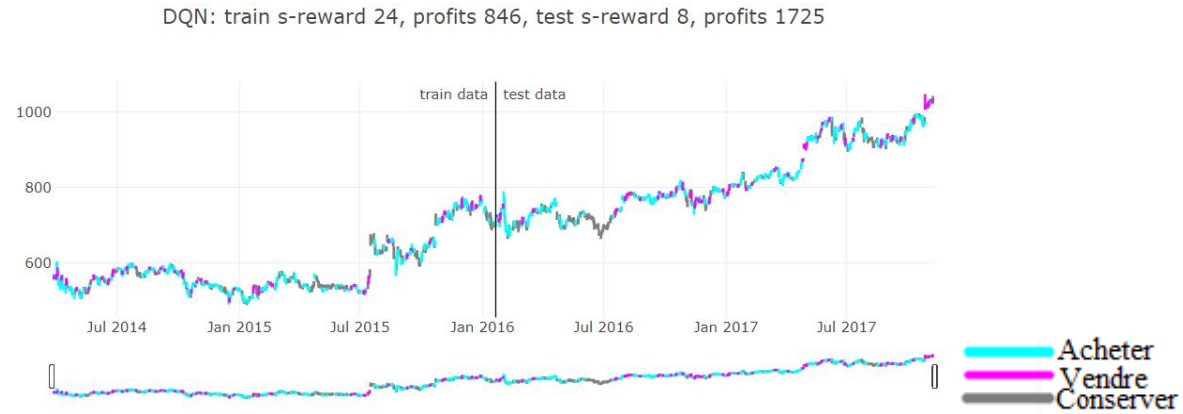

Figure IV.5: Performance du DQN, le profit durant la période de test est de 1725

.

IV. 5.2 Performance du Double DQN

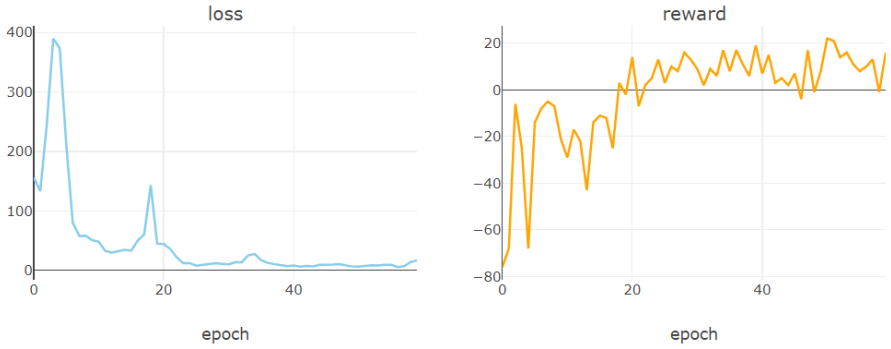

Figure IV.6 : Récompense et perte du Double DQN

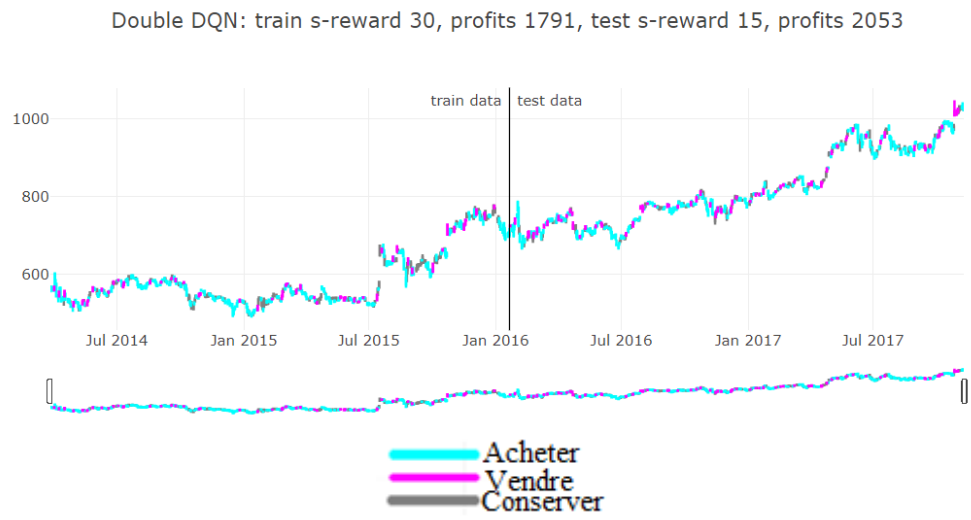

Figure IV.7: Performance du Double DQN, le profit durant la période de test est de 2053

IV. 5.3 Performance du Dueling Double DQN

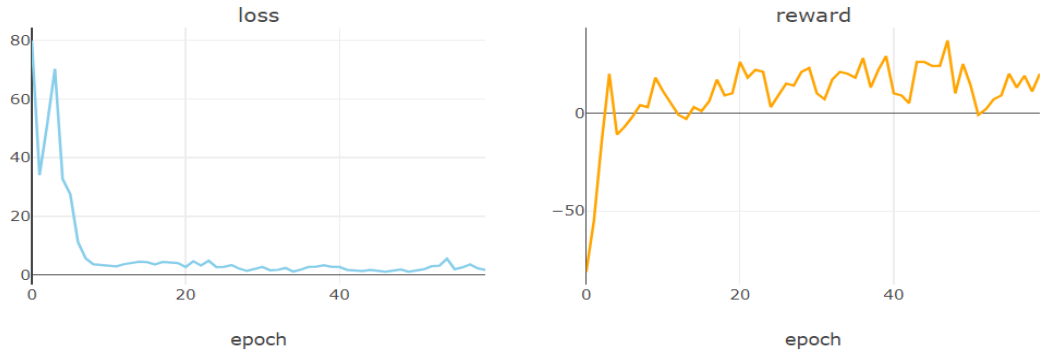

Figure IV.8: Récompense et perte du Dueling Double DQN

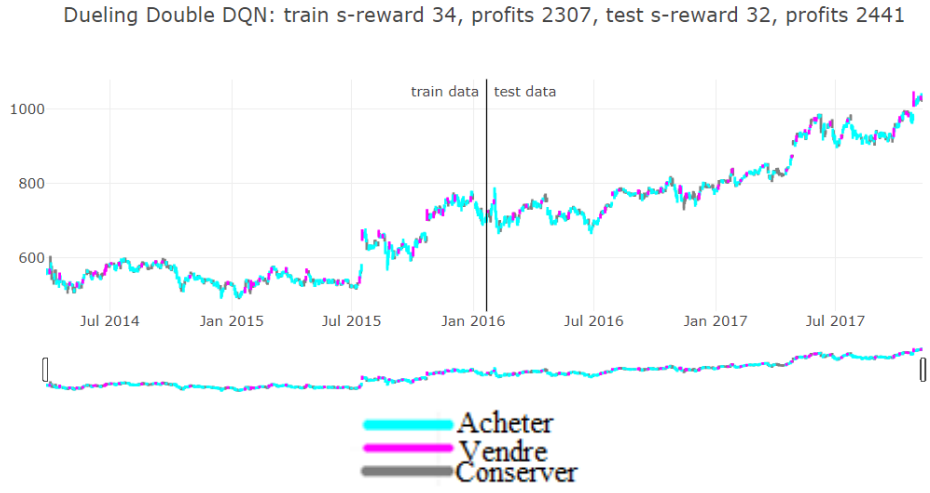

Figure IV.9: Performance du Dueling DDQN, le profit durant la période de test est 2441

IV. 5.4 Analyse des résultats

En tirant des conclusions de notre expérience comparant le DQN, le Double DQN (DDQN) et le Duel Double DQN (Duel DDQN), plusieurs points clés peuvent être soulignés :

1. Performance Globale

- **Duel Double DQN (Duel DDQN)** : Génère le bénéfice moyen le plus élevé (2441), surpassant les autres méthodes. Ce modèle converge plus rapidement et montre une performance supérieure, grâce à sa capacité à décomposer la valeur Q en valeur de l'état et avantage de l'action.

- **Double DQN (DDQN)** : Produit un bénéfice moyen (2053) supérieur à celui du DQN standard, mais inférieur au Duel DDQN. Il corrige la surestimation des valeurs Q, offrant une estimation plus stable et précise des valeurs Q.

- **DQN** : Affiche le bénéfice moyen le plus bas (1725) et des variations plus grandes en début d'apprentissage. La surestimation des valeurs Q et l'absence de mécanismes stabilisateurs conduisent à des performances et une convergence inférieure.

2. Stabilité et Convergence

- **Duel DDQN** : Converge plus rapidement et montre une plus grande stabilité dès les premiers épisodes d'apprentissage. Cela s'explique par l'amélioration de la décomposition de la valeur Q en deux composants distincts.

- **DDQN** : Converge plus rapidement que le DQN en raison de la réduction de la surestimation des valeurs Q, mais pas aussi rapidement que le Duel DDQN.

- **DQN** : Montre de grandes variations au début de l'apprentissage, ce qui indique une instabilité initiale due à la surestimation des valeurs Q et une exploration intensive.

3. Robustesse d’architecture

- **Duel DDQN** : L'architecture duelle, combinée avec le Double DQN, se révèle être la plus robuste, offrant des bénéfices élevés et une convergence rapide.

- **DDQN** : L'introduction du Double DQN améliore significativement la robustesse par rapport au DQN standard.

- **DQN** : Bien qu'il soit capable d'apprendre et de converger, il est moins robuste et performant que les autres variantes plus avancées.

IV.5.5 Comparaison de l'Impact des hyperparamètres

IV.5.5.1 Impact de GAMMA sur les Performances du DQN, DDQN et DDDQN

Un paramètre crucial dans les algorithmes DQN, DDQN et Duel DDQN est le facteur de discount, ou gamma (γ), qui détermine l'importance des récompenses futures par rapport aux récompenses immédiates. Une **valeur élevée de gamma** incite le modèle à privilégier les **gains à long terme**, tandis qu'une **valeur faible favorise les gains à court terme**. La sélection appropriée de ce paramètre peut donc avoir un impact significatif sur les performances des modèles de trading.

Dans le tableau suivant nous comparons l'impact de gamma sur chacune des méthodes DQN, DDQN et Duel DDQN dans le trading. Nous testerons différentes valeurs de gamma : 0.97, 0.77, 0.57 et 0.3, et analyserons comment ces variations affectent les performances de chaque algorithme.

GAMMA	0.97		0.77		0.57		0.3	
Profits	traning	test	traning	test	traning	test	traning	test
DQN	**2123**	**2019**	846	1725	1643	**2274**	1013	1529
DDQN	349	797	1791	2053	1297	1350	1240	925
DDDQN	1369	1152	**2307**	**2441**	**1663**	2015	**2110**	**2904**

Tableau IV.1 : influence de GAMMA sur les performances des différentes architectures

Dans notre étude comparative des méthodes DQN, Double DQN (DDQN) et Dueling Double DQN (DDDQN) appliquées à la prédiction du trading avec différentes valeurs de gamma (0.97, 0.77, 0.57, et 0.3), nous avons observé que :

DQN : Malgré ses problèmes connus de surestimation des valeurs d'action, le DQN a montré des performances supérieures au DDQN dans certains cas. Cette contre-intuition peut être expliquée par le fait que la surestimation des valeurs d'action peut paradoxalement être bénéfique dans un environnement de marché volatile, favorisant des décisions plus agressives et potentiellement plus profitables.

DDQN : Le DDQN, conçu pour corriger la surestimation des valeurs d'action inhérente au DQN et améliorer la stabilité et la précision des prédictions, a parfois été moins performant que le DQN. Cela est probablement dû à la dynamique rapide des marchés financiers, où la réactivité rapide aux changements est cruciale. La prudence accrue du DDQN peut limiter ses performances dans des environnements aussi volatiles.

Dueling DDQN (DDDQN) : Le DDDQN s'est révélé être le modèle le plus performant et le plus stable. Son architecture innovante, qui sépare la valeur de l'état (V) et l'avantage de l'action (A), réduit la variance et les surestimations. Cela permet au DDDQN de mieux évaluer les situations où certaines actions ont peu d'impact sur la valeur globale, rendant les décisions plus robustes.

Cette séparation permet au DDDQN de s'adapter plus efficacement aux changements rapides du marché, offrant des performances supérieures en termes de profits. Sa robustesse et sa meilleure gestion des hyperparamètres en font un choix idéal pour des environnements volatils comme le trading algorithmique.

L'étude montre que bien que le DQN puisse parfois surpasser le DDQN dans des environnements de trading volatils, le DDDQN reste la meilleure option pour une performance et une stabilité optimales grâce à sa capacité à réduire la variance et à s'adapter rapidement aux dynamiques du marché.

IV.5.5.2 Impact de Epsilon (ϵ) sur les Performances du DQN, DDQN et Dueling DDQN

Dans l'apprentissage par renforcement, le dilemme exploration-exploitation est une problématique centrale. Il s'agit de trouver un équilibre entre :

- **Exploration** : Tester de nouvelles actions pour découvrir leur potentiel (ϵ élevé).
- **Exploitation** : Utiliser les connaissances actuelles pour maximiser les récompenses immédiates (ϵ faible).

Le paramètre ϵ dans les algorithmes ϵ-greedy contrôle cette balance. Un ϵ élevé favorise l'exploration, tandis qu'un ϵ faible favorise l'exploitation.

Expérimentation

Nous avons étudié l'impact de différentes valeurs de **ϵ** sur les performances de trois algorithmes : DQN, Double DQN (DDQN), et Dueling Double DQN (DDDQN). Les résultats des profits obtenus pour chaque valeur de **ϵ** sont résumés dans le tableau suivant :

ϵ	0.80		0.60		0.40		0. 20		0.10	
profits	Traning	test	traning	test	traning	test	traning	test	traning	Test
DQN	1659	1773	334	95	331	584	2167	2241	**3116**	2830
DDQN	938	716	334	579	**1437**	**2336**	891	702	2897	**3077**
DDDQN	**1965**	**2013**	**2831**	**3006**	974	982	**2479**	**2421**	1933	2716

Tableau IV.2: Impact de EPSILON (**ϵ**) sur les performances des différentes architectures

Analyse des résultats :

L'analyse des résultats montre que le choix de **ϵ** a un impact significatif sur les performances des algorithmes de renforcement.

Le **DQN** : Montre une amélioration substantielle de la performance à mesure que **ϵ** diminue, mais n'atteint pas la robustesse et les performances maximales du DDDQN.

Le **DDQN** : Améliore l'estimation des valeurs Q et réduit la surestimation, mais ne parvient pas à atteindre les mêmes niveaux de performance optimaux que le DDDQN, particulièrement à des niveaux de **ϵ** plus élevés.

Le Dueling Double DQN (**DDDQN**) offre une architecture améliorée qui excelle dans la gestion du dilemme exploration-exploitation. Sa capacité à maintenir des performances élevées à différents niveaux de **ϵ** et à atteindre un équilibre optimal entre exploration et exploitation en fait un choix privilégié pour les environnements complexes nécessitant une prise de décision robuste. Les améliorations architecturales du DDDQN permettent une meilleure valorisation des états et des actions, rendant cet algorithme particulièrement efficace pour maximiser les récompenses dans des environnements dynamiques et incertains.

Cas le plus défavorable : $\epsilon = 0.6$ et $\gamma = 0.90$

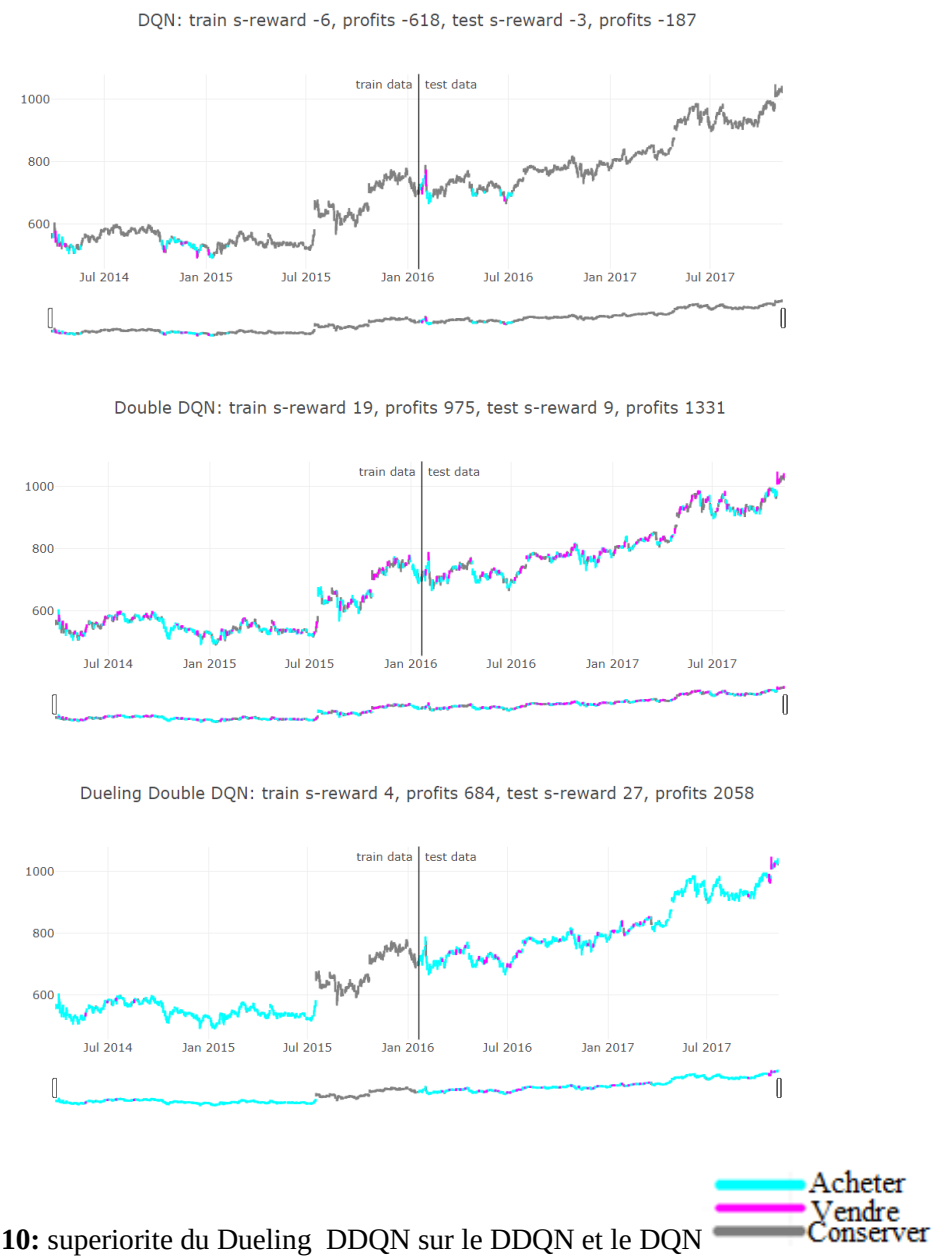

Figure IV.10: superiorite du Dueling DDQN sur le DDQN et le DQN

DQN : durant son apprentissage DQN n a pris aucune décision par consequent il a pris la meme decision durant le test. Le résultat est négatif.

DDQN : malgré les conditions il a pu acheter et vendre. un profit de 1331 durant le test.

DUEL DDQN : durant l apprentissage le DDDQN est passe par 2 etapes. Au debut il n a appris qu'a acheter puis il est reste neutre. Mais durant le test il pu prendre des decisions a vendre et acheter selon la situation du marche et n a jamais reste neutre afin de recuperer a bon profit qui est de 2058. Ce qui confirme que notre analyse notée plus haut.

IV.6 conclusions

Dans ce chapitre, nous avons exploré l'implémentation pratique des algorithmes d'apprentissage par renforcement profond (Deep Reinforcement Learning, DRL) pour développer des stratégies de trading. Nous avons examiné les étapes clés du développement, de l'entraînement et de l'évaluation des agents de trading basés sur trois algorithmes principaux : DQN, Double DQN (DDQN) et Dueling Double DQN (DDDQN). Voici les points clés observés :

Les résultats de notre étude montrent que le Dueling Double DQN (DDDQN) surpasse généralement les autres algorithmes en termes de profit généré et de stabilité.

Cet algorithme a généré le bénéfice moyen le plus élevé et a montré une convergence plus rapide grâce à sa capacité à décomposer la valeur Q en valeur de l'état et avantage de l'action. Cela permet une estimation plus précise et une gestion efficace du dilemme exploration-exploitation.

Le DDDQN s'est montré adaptable à diverses valeurs de gamma, offrant de bonnes performances dans différents scénarios de trading.

Le DDDQN a bien géré le dilemme exploration-exploitation, maintenant des performances élevées à différents niveaux de ϵ.

Le DDDQN a démontré une robustesse supérieure grâce à sa capacité à réduire la variance et les surestimations, rendant les décisions plus robustes et stables.

Les variantes améliorées de DQN, notamment le DDDQN, se sont avérées supérieures en termes de profit moyen, rapidité de convergence et robustesse. Ces résultats confirment la supériorité des méthodes corrigées de surestimation des valeurs Q et de décomposition des valeurs Q pour une estimation plus précise.

L'exploration d'autres architectures DRL, l'intégration de données de marché en temps réel et l'application à d'autres classes d'actifs sont des voies potentielles pour approfondir cette technologie, ouvrant des perspectives pour des stratégies de trading encore plus efficaces et rentables. Le DDDQN se distingue comme une solution robuste et performante pour le trading algorithmique.

CONCLUSION GENERALE

Ce mémoire a exploré de manière exhaustive les concepts fondamentaux et les applications pratiques du deep reinforcement learning (DRL), en mettant particulièrement l'accent sur son utilisation dans le domaine du trading algorithmique. À travers une structuration en quatre chapitres principaux, nous avons construit une compréhension approfondie et intégrée de cette technologie révolutionnaire. La Synthèse des Chapitres est comme suite :

1-Apprentissage par Renforcement (RL) :

- Introduction aux éléments de base du RL : agents, environnements, actions, états et récompenses.
- Explication des concepts de politique, fonction de valeur et fonction de valeur Q.
- Discussion sur les défis du RL, tels que le dilemme exploration-exploitation et la convergence des algorithmes.

2-Apprentissage Profond (Deep Learning) :

- Présentation des réseaux de neurones artificiels, fonctions d'activation et propagation.
- Exploration des architectures complexes comme les Réseaux de Neurones Convolutionnels (CNN).
- Techniques d'entraînement, notamment l'optimisation par descente de gradient.

3-Apprentissage par Renforcement Profond (DRL) :

- Fusion des concepts de RL et Deep Learning pour introduire le DRL.
- Étude des algorithmes principaux : Deep Q-Network (DQN), Double DQN (DDQN) et Dueling DQN (DDDQN).

4-Application du DRL au Marché Boursier :

- Développement de stratégies de trading algorithmique basées sur DRL.
- Préparation des données, modélisation de l'environnement de trading, implémentation et évaluation des algorithmes.
- Les résultats montrent que le DDDQN surpasse les autres algorithmes en termes de profit et de stabilité.

Resultats et Perspectives Futures

Performance du DDDQN : Supérieur en termes de profit moyen, rapidité de convergence et robustesse, grâce à sa capacité à décomposer la valeur Q en valeur de l'état et avantage de l'action.

Adaptabilité : Le DDDQN gère mieux le dilemme exploration-exploitation et s'adapte à différentes valeurs de gamma et d’epsilon.

Perspectives Futures:

- Exploration de nouvelles architectures DRL.
- Intégration de données de marché en temps réel.
- Applications à d'autres classes d'actifs et secteurs.
- Développement de mécanismes avancés de gestion des risques.

En conclusion, ce mémoire démontre la puissance du DRL, en particulier le DDDQN, pour développer des stratégies de trading algorithmique robustes et efficaces. Les concepts théoriques et les applications pratiques montrent le potentiel transformateur du DRL pour résoudre des problèmes complexes et dynamiques, ouvrant des perspectives pour des systèmes de trading plus intelligents et performants.

Référence:

[1] Machine Learning", New York, NY, USA: McGraw-Hill, Inc by TM Mitchell - 1997

[2]An Introduction, vol. 2, 2nd ed. By R. Sutton and A. Barto, Reinforcement Learning Cambridge, MA, USA: MIT Press, 2015.

[3] Taxonomy of reinforcement learning algorithms H Zhang, T Yu Deep reinforcement learning: Fundamentals, research and applications, 2020

[4] Reward-based learning, model-based and model-free QJM Huys, A Cruickshank, P Seriès Encyclopedia of Computational Neuroscience, 2014

[5] "Reinforcement Learning with Deep Q-Networks" A thesis submitted in partial fulfillment of the requirements for the degree Masters of Science, Computer Science. (Western Kentucky University, Bowling Green, Kentucky)By Caleb Cassady.

[6] Mechanistic data science for STEM education and applications By WK Liu, Z Gan, M Fleming 2021

[7] A Practical Non-Profiled Deep-Learning-Based Power Analysis with Hybrid-Supervised Neural Networks by Fancong Kong,Xiaohua Wang,Kangran Pu,Jingqi Zhang and Hua Dang

[8] A Survey of Deep Learning Methods for Cyber Security by Daniel S. Berman Anna L. Buczak Jeffrey S. Chavis and Cherita L. Corbett Johns Hopkins University Applied Physics Laboratory (JHU/APL1), Laurel, MD 20910, USA.

[9] "Q-learning," Machine learning, vol. 8, no.3-4, pp. 279–292,by C. J. Watkins and P. Dayan, 1992.

[10] On-line Q-learning using connectionist systems. by G. A. Rummery and M. Niranjan, University of Cambridge, 1994.

[11] Deep reinforcement learning with experience replay based on sarsa,by D. Zhao, H. Wang, K. Shao, and Y. Zhu, in IEEE Symposium Series on Computational Intelligence, 2016.

[12] Double Q-learning, by H. Van Hasselt, Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information.

[13] Deep reinforcement learning with double Q-learning ,by H Van Hasselt, A Guez, D Silver Proceedings of the AAAI conference on artificial intelligence, 2016.

[14] Dueling network architectures for deep reinforcement learning ,Z Wang, T Schaul, M Hessel… - International …, 2016.

[15] A survey of deep learning methods for cyber security DS Berman, AL Buczak, JS Chavis, CL Corbett Information, 2019.

[16] A gentle introduction to reinforcement learning and its application in different fields ,M Naeem, STH Rizvi, A Coronato IEEE access, 2020.

[17] Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies , E Chong, C Han, FC Park Expert Systems with Applications, 2017.

[18] Stock trading bot using deep reinforcement learning AR Azhikodan, AGK Bhat , MV Jadhav Innovations in Computer Science and Engineering: Proceedings of the Fifth, 2019.

[19] Deep direct reinforcement learning for financial signal representation and trading Y Deng, F Bao, Y Kong, Z Ren, Q Dai IEEE transactions on neural networks and learning systems, 2016.

[20] Literature review: Machine learning techniques applied to financial market prediction

BM Henrique, VA Sobreiro, H Kimura - Expert Systems with Applications, 2019.

[21] Design of Power-Efficient Training Accelerator for Convolution Neural Networks

by JiUn Hong,ˈSaad Arslan,ˈTaeGeon Lee ,HyungWon Kim.

[22] Reinforcement learning approaches in social robotics N Akalin, A Loutfi Sensors, 2021