Tahar Adam AMAIRI (amat0601), Victor CHAU (chav0401), Omar CHIDA (chim2708), Céline ZHANG (zhac3201)

Gestion d'un portfolio par un BOT de trading entraîné par Reinforcement learning

PROJET étape 2 - Planification en intelligence artificielle - IFT 702 - *Supervised by Froduald Kabanza* - Université de Sherbrooke - 29/03/2023

Introduction. Après l'analyse du modèle TDQN, le modèle semble être moins performant que prévu sur la prédiction, en particulier, il était annoncé que l'agent aurait du mal à prédire sur des cours boursiers trop volatiles. De ce fait, pour explorer une piste d'amélioration et essayer de corriger les problèmes rencontrés à l'étape précédente, nous avons choisi d'essayer une autre méthode d'apprentissage par renforcement, de la famille de *Policy-Gradient*, le *Proximal Policy Optimization* (PPO). L'idée de base derrière PPO est d'optimiser une politique en effectuant de petits ajustements, plutôt que de tout changer à chaque étape. Cela permet de maintenir la stabilité de l'apprentissage tout en améliorant progressivement la politique. PPO est connu pour être efficace dans des environnements de jeux complexes comme les jeux vidéos, où il est primordial de trouver une stratégie optimale pour gagner. Certaines revues appliquent PPO avec un LSTM dans des problèmes de *trading* ou de gestion de portfolio pour améliorer l'entraînement, la stabilité et la capacité de généralisation de l'agent. De ce fait, dans ce qui suit, nous allons discuter de la méthode PPO en général, ses applications en *trading*, des avantages et des inconvénients par rapport à DQN, puis nous présenterons les stratégies d'améliorations ainsi que les analyses sur les résultats obtenus.

L'état de l'art. Le *Proximal Policy Optimization* (PPO) est un algorithme d'apprentissage par renforcement développé par OpenAI en 2017. C'est un type de méthode de *policy gradient* qui vise à trouver la politique optimale pour un agent dans un environnement donné. Pour bien comprendre, nous allons brièvement expliquer les méthodes de policy gradient. Ces méthodes optimisent directement la fonction de politique d'un agent. La fonction de politique associe l'état actuel de l'environnement à une distribution de probabilités sur les actions possibles de l'agent. L'objectif des méthodes de *policy gradient* est d'apprendre une politique qui maximise la récompense cumulative attendue au fil du temps. Il existe plusieurs variantes des méthodes de *policy gradient*, mais elles partagent toutes l'idée de base de l'utilisation de l'ascension du gradient pour mettre à jour les paramètres de la politique. Les gradients sont estimés à l'aide d'échantillons de la distribution de la politique, obtenus en exécutant la politique dans l'environnement et en observant les récompenses et les états qui en résultent. Les méthodes de *policy gradient* souffrent de plusieurs inconvénients majeurs, notamment l'inefficacité des échantillons, l'incohérence des mises à jour de la politique et la variance élevée des récompenses.

Depuis, un certain nombre d'améliorations ont été apportées. L'algorithme TRPO (Trust Region Policy Optimization) a été introduit par Schulman et al. en 2015. L'idée principale derrière TRPO est de contraindre la mise à jour de la politique à une région de confiance autour de la politique actuelle. La région de confiance est définie comme une région dans l'espace des paramètres de la politique dans laquelle la fonction objective est garantie d'avoir une approximation de second ordre qui est suffisamment précise pour assurer l'amélioration. La mise à jour de la politique est alors obtenue en résolvant un problème d'optimisation contraint qui maximise la récompense attendue sous réserve de la contrainte de la région de confiance.

La PPO a été développée en réponse à certaines des limites de la TRPO, et elle introduit une fonction objectif de substitution écrêtée (*clipped*) qui garantit que la mise à jour de la politique ne s'éloigne pas trop de la politique précédente, ce qui améliore la stabilité du processus d'optimisation. Il utilise également une *value function* pour estimer la récompense cumulative attendue, ce qui améliore encore l'efficacité de l'échantillonnage et la stabilité de l'algorithme. Cela permet d'éviter des baisses catastrophiques de performance et conduit à un apprentissage plus stable (*PPO explained* par Wouter van Heeswijk). La PPO introduit également deux hyperparamètres clés qui contrôlent le degré de mise à jour des politiques : le paramètre d'écrêtage et le coefficient d'entropie. Le paramètre de clip limite la taille des mises à jour de la politique, tandis que le coefficient d'entropie encourage l'exploration en pénalisant les politiques trop déterministes *(Proximal Policy Optimization Algorithms* par John Schulman et al.)

Concernant le trading, il existe quelques différences dans la mise en œuvre et les cas d'utilisation spécifiques de la PPO pour le trading. L'une des principales différences réside dans le type de données utilisées pour former l'agent PPO. Dans le RL traditionnel, l'agent reçoit généralement des observations d'un environnement, comme les pixels d'un écran de jeu. Dans le domaine du trading, l'agent reçoit des données sur les marchés financiers, telles que les prix, les volumes et d'autres indicateurs. L'espace d'action constitue une autre différence. Dans le RL traditionnel, l'espace d'action est souvent discret, comme la sélection d'une action à partir d'un ensemble fixe de choix. Dans le domaine du trading, l'espace d'action est généralement continu, comme la spécification d'un prix d'achat/de vente ou d'une taille d'ordre. L'un des avantages de l'OPP par rapport à d'autres algorithmes de RL est sa capacité à gérer des espaces d'action continus, qui sont courants dans les applications de trading. Des études récentes ont exploré diverses façons de traiter les actions continues dans la PPO. La fonction de récompense est une autre différence essentielle. Dans le RL traditionnel, la fonction de récompense est généralement conçue pour inciter l'agent à maximiser un score ou à atteindre un objectif. Dans le domaine du trading, la fonction de récompense est souvent plus complexe et peut intégrer des facteurs tels que la gestion du risque, la volatilité et d'autres paramètres financiers. Enfin, dans le domaine du trading, l'objectif ultime est de déployer l'agent PPO formé dans un environnement de trading en temps réel. Cela nécessite des considérations supplémentaires, telles que la gestion des risques, l'exécution des ordres et l'optimisation de la latence. Le trading à haute fréquence (HFT) est un domaine difficile qui nécessite une prise de décision rapide et précise. Des études récentes ont exploré l'utilisation du PPO pour le HFT (*Deep RL for Active High Frequency Trading* par Antonio Briola et al.).

Un *papier* écrit par Hongyang Yang et al. propose une nouvelle stratégie d'ensemble qui combine trois algorithmes (PPO, Advantage Actor Critic, and Deep Deterministic Policy Gradient) pour trouver la stratégie de trading optimale dans un marché boursier complexe

et dynamique. La stratégie proposée est plus robuste et plus fiable, elle peut s'adapter à différentes situations de marché et maximiser le rendement sous contrainte de risque.

Avantages du PPO par rapport au TDQN :

- Le PPO est un algorithme "on-policy", ce qui signifie qu'il met à jour la politique directement en fonction du comportement de la politique actuelle. Il est donc plus stable et moins sujet à des baisses de performances catastrophiques que le TDQN, qui est un algorithme “off-politique” et qui peut être instable lorsque la politique d'exploration est trop différente de la politique apprise.
- PPO est conçu pour traiter des espaces d'action continus, alors que TDQN est mieux adapté aux espaces d'action discrets.
- PPO est connu pour converger plus rapidement et atteindre de meilleures performances avec moins de données que le TDQN.
- PPO peut gérer des politiques stochastiques, ce qui peut s'avérer important dans le domaine du trading où le hasard est souvent présent. Le TDQN, quant à lui, nécessite une politique déterministe, ce qui peut limiter son efficacité dans les situations où la stochasticité est présente.

Avantages de TDQN par rapport à PPO :

- TDQN est un algorithme basé sur un modèle qui utilise un réseau neuronal pour estimer les valeurs Q des actions, ce qui peut être plus précis que l'approche basée sur la politique utilisée par PPO.
- TDQN peut gérer des récompenses différées, ce qui est courant dans le trading où la récompense peut ne pas être immédiatement apparente.
- Le TDQN peut gérer des espaces d'état à haute dimension, ce qui est courant dans les applications de trading où il peut y avoir de nombreux indicateurs et caractéristiques à prendre en compte.
- Le TDQN est plus facile à mettre en œuvre et nécessite moins de réglages des hyperparamètres que le PPO.

Méthodologie. Pour procéder à l’amélioration de notre système existant, nous avons intégré une nouvelle méthode d’apprentissage, qui est le PPO. A partir du code source comme base, nous avons ajouté une extension au *bot* en ajoutant une classe PPO qui utilise une fonction actor-critic pour mettre à jour ses politiques de décision. La fonction actor-critic est basée sur un réseau de neurones possédant une architecture de couches comme suit :

- Linéarisation : permet d’extraire des caractéristiques des données en les représentant dans un espace plus complexe
- Normalisation : permet de normaliser les sorties de neurones, pour éviter l’étirement des plages de valeurs
- LeakyReLU : permet d’avoir les valeurs négatives non nulles, il peut aider à éviter le phénomène de neurones morts, réduire la corrélation entre les neurones et potentiellement accélérer l'apprentissage.
- Dropout à 20% : permet d’ajouter de la généralisation au modèle en entraînant de manière plus robuste des chemins aléatoirement

La couche est répétée jusqu’à trois fois dans le réseau. L’architecture de l’*actor* et du *critic* est identique à l’exception de la dernière couche qui est linéaire suivi d’un *softmax*. L’optimiseur utilisé est Adam et la fonction de perte est l’erreur quadratique MSE. Nous avons également dû adapter l’environnement du modèle avec GYM. Nous avons ajouté la possibilité de *trade* sur de multiples cours boursiers en appliquant la même stratégie, la gestion parallèle sur plusieurs *stocks* et nous avons introduit une extension vers des cours de crypto-monnaie qui est une amorce pour l’étape suivante.

Comme précédemment, un *Jupyter Notebook* avec des affichages dynamiques est disponible pour simplifier la visualisation et l’analyse des résultats. D’autres fonctionnalités de *rendering* ont été ajoutées pour illustrer comparer nos modèles, tels que la visualisation de l’impact des décisions du temps, les métriques selon les stratégies de *trading*. Une *appréciation du rendu 2D et 3D* de l’application d’une méthode classique (MATF basée sur les tendances) en illustration A est disponible. Une analyse des limites du code et une exploration du comportement de PPO a permis de connaître les faiblesses de l’algorithme actuel, tels que le manque de généralisation, les écarts de résultats entre les phases d’entraînement et de test. La complexité de PPO n’a pas forcément apporté des améliorations sur le problème.

Expérimentations. Le *bot* a été lancé dans les mêmes conditions que l’étape précédente, sur les actions d’Apple et de Tesla en considérant 30 jours pour la taille des états sur 6 ans d’apprentissage et 2 ans de simulation en phase test. En figures 1 et 2, nous avons respectivement les courbes d’apprentissage et de récompenses du *bot* sur les actions d’Apple et de Tesla. Pour cette simulation, les résultats sont assez variables. Nous avons de grosses fluctuations, ceci est légèrement plus exprimé chez Tesla qui est plus volatile. On note également qu’en phase de test, on est d’autant plus instable qu’en entraînement. De plus étrangement, les courbes de récompenses pour les deux cours boursiers semblent suivre la même tendance à quelque exception. Ceci peut être expliqué par le fait que l’agent est trop ancré sur une stratégie générale, ou alors qu’il n’a pas assez appris. Ceci est confirmé par le tableau de performance en figure 3, les profits sont négatifs, donc nous avons essentiellement des pertes, pour les *stocks*, autour de -6059 (Apple) et -9332 (Tesla). Les ratio de Sharpe et Sortino sont tous négatifs avec une valeur plus importante pour Tesla dû à sa volatilité. Sur l’entraînement, PPO a beaucoup moins bien performé que TDQN de l’étape 1 (se référer aux résultats du rapport 1).

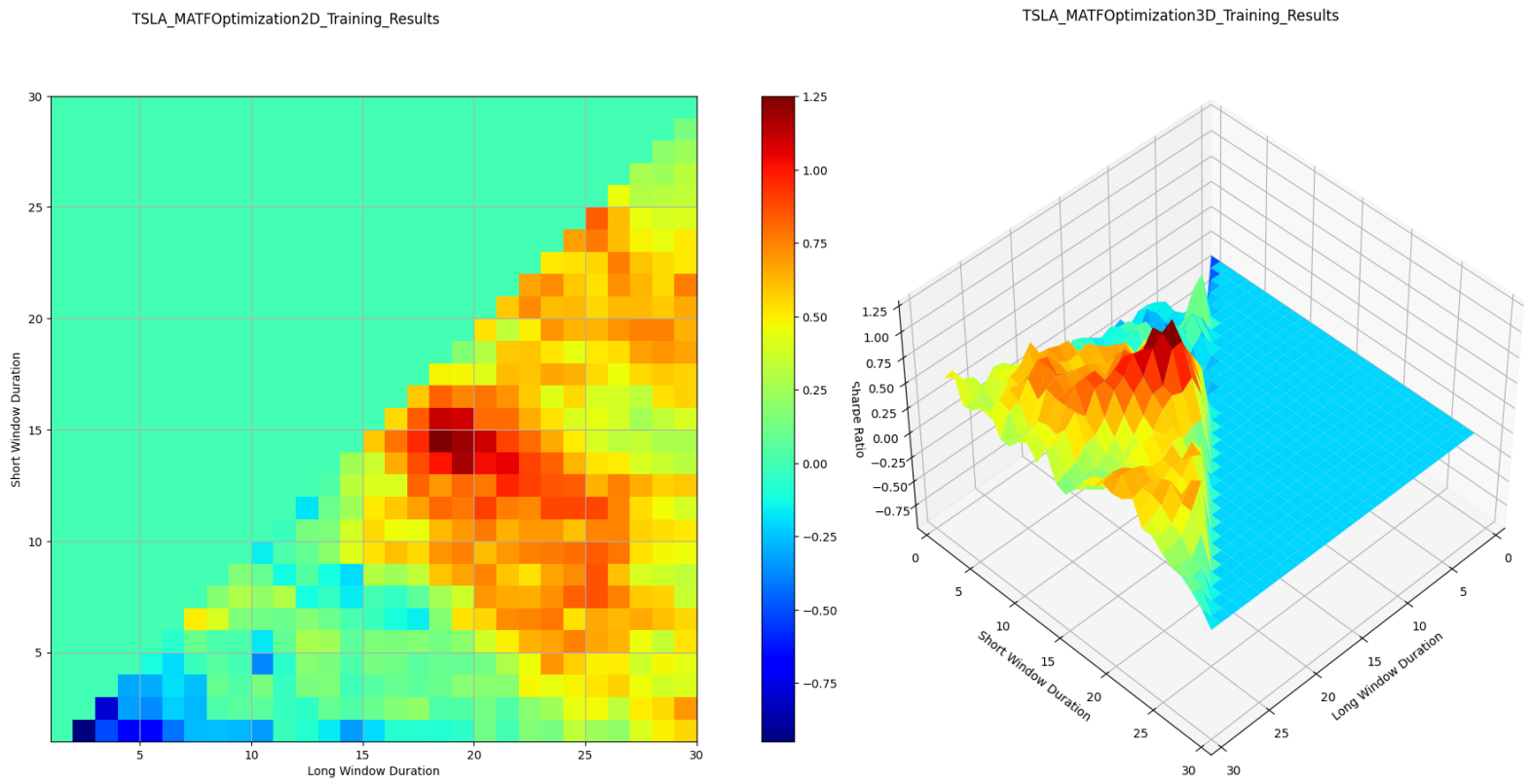

Illustration A - Appréciation 2D et 3D de la méthode MATF de *trading* classique sur le cours boursier de Tesla

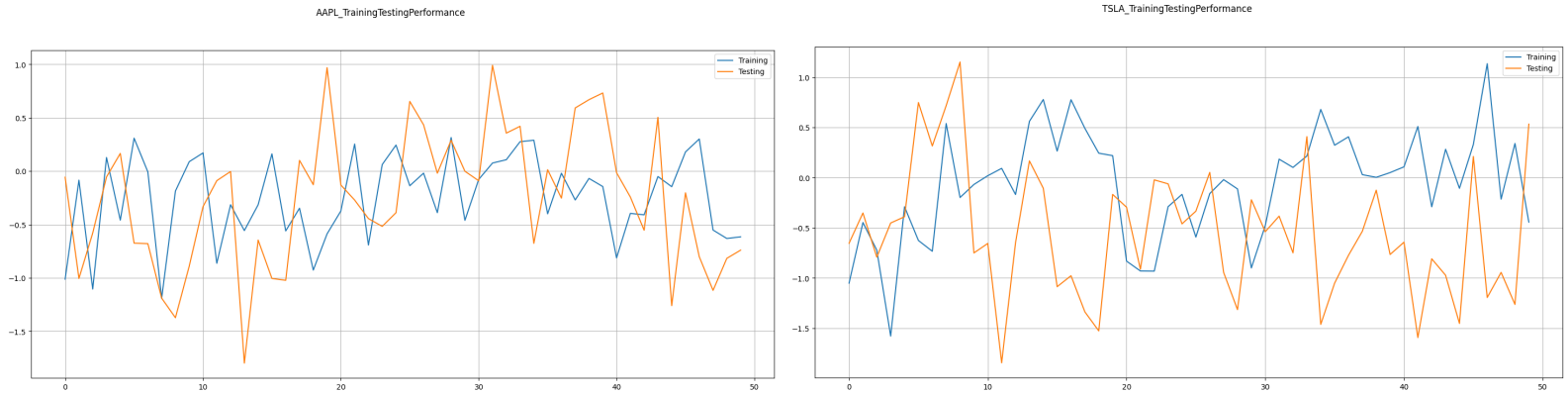

Figure 1 - Courbes d'apprentissage de l'agent sur les *stocks* d'Apple (gauche) et de Tesla (droite), *training* (bleu) et *testing* (jaune)

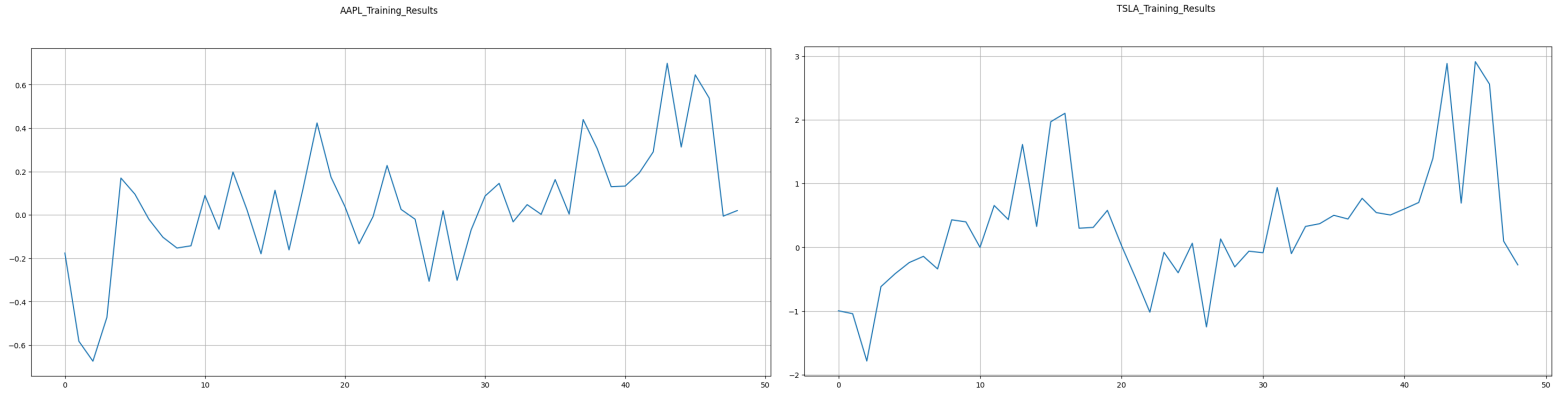

Figure 2 - Courbes de récompenses du système sur les *stocks* d'Apple (gauche) et de Tesla (droite)

Les figures 4 et 5 donnent un aperçu des valeurs du *portfolio* du *bot* selon les actions de celui-ci sur les actions boursières d'Apple et de Tesla. Pareillement que l'étape 1, en comparant la fréquence de *trading*, en phase de test, le *bot* a tendance à réaliser moins d'action effective par rapport à la phase d'entraînement. La figure 6 présente l'évaluation des performances pour la phase de test, en comparaison aux tableaux des métriques de la phase d'entraînement, en figure 3, le *bot* génère des pertes semblables sur les deux *stocks*, avec une légère amélioration pour Apple, qui a cumulé moins de pertes. De plus, les valeurs comme le *Drawdown*, *Ratio Average Profit/Loss*, le *Sortino Ratio* sont plutôt proches, avec une dégradation plus notable pour les cours de Tesla. Sur cette période de test, Apple et Tesla ont un *Drawdown* relativement inférieur à celui de la période d'entraînement et les profits sont plus importants. On peut retenir l'hypothèse faite à l'étape 1, le *Drawdown* impacte négativement les performances du *bot*. De plus, comme confirmé en phase de test, l'agent sous PPO, performe nettement moins bien que sous TDQN. Les pertes sont de l'ordre 10 000 par rapport aux résultats de l'étape 1, dont les gains ont pu atteindre 6 000 et plus.

Performance Indicator	PPO (Training)
Profit & Loss (P&L)	-6059
Annualized Return	-20.59%
Annualized Volatility	24.64%
Sharpe Ratio	-0.507
Sortino Ratio	-0.721
Maximum Drawdown	70.73%
Maximum Drawdown Duration	1023 days
Profitability	44.90%
Ratio Average Profit/Loss	0.979
Skewness	-0.042

Performance Indicator	PPO (Training)
Profit & Loss (P&L)	-9332
Annualized Return	-100.00%
Annualized Volatility	47.33%
Sharpe Ratio	-0.717
Sortino Ratio	-0.970
Maximum Drawdown	94.65%
Maximum Drawdown Duration	1060 days
Profitability	44.42%
Ratio Average Profit/Loss	0.857
Skewness	0.117

Figure 3 - Tableaux des métriques de performance en phase entraînement du *bot* sur les *stocks* d'Apple (gauche) et de Tesla (droite)

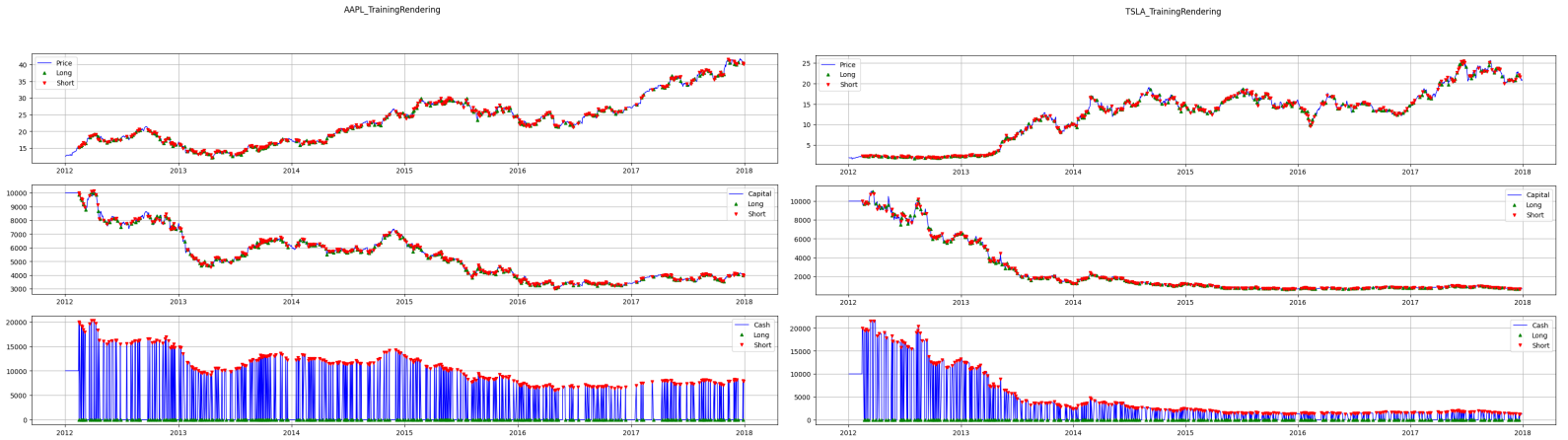

Figure 4 - Apprentissage de l'agent RL PPO entre 2012 et 2017 sur les cours boursiers d'Apple (gauche) et de Tesla (droite)

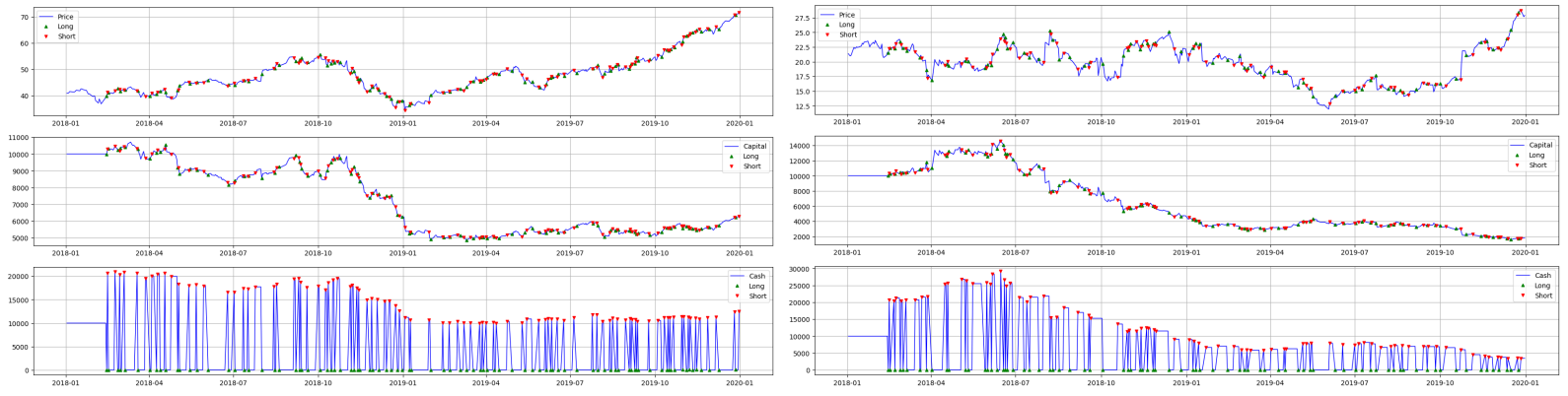

Figure 5 - Application de l'agent RL PPO entre 2018 et 2019 sur les *stocks* d'Apple (gauche) et de Tesla (droite)

Performance Indicator	PPO (Testing)
Profit & Loss (P&L)	-3712
Annualized Return	-22.07%
Annualized Volatility	26.71%
Sharpe Ratio	-0.735
Sortino Ratio	-0.874
Maximum Drawdown	54.75%
Maximum Drawdown Duration	250 days
Profitability	51.10%
Ratio Average Profit/Loss	0.727
Skewness	-0.926

Performance Indicator	PPO (Testing)
Profit & Loss (P&L)	-8221
Annualized Return	-100.00%
Annualized Volatility	53.50%
Sharpe Ratio	-1.340
Sortino Ratio	-1.626
Maximum Drawdown	89.20%
Maximum Drawdown Duration	380 days
Profitability	46.25%
Ratio Average Profit/Loss	0.787
Skewness	-0.872

Figure 6 - Evaluation de performance en phase de test du *bot* PPO pour Apple (à gauche) et Tesla (à droite)

Performance Indicator	B&H (Testing)
Profit & Loss (P&L)	7961
Annualized Return	28.80%
Annualized Volatility	26.55%
Sharpe Ratio	1.239
Sortino Ratio	1.559
Maximum Drawdown	38.43%
Maximum Drawdown Duration	62 days
Profitability	100.00%
Ratio Average Profit/Loss	inf
Skewness	-0.476

Performance Indicator	S&H (Testing)
Profit & Loss (P&L)	-7981
Annualized Return	-100.00%
Annualized Volatility	44.17%
Sharpe Ratio	-1.591
Sortino Ratio	-2.204
Maximum Drawdown	82.28%
Maximum Drawdown Duration	250 days
Profitability	0.00%
Ratio Average Profit/Loss	0.000
Skewness	0.145

Performance Indicator	MATF (Testing)
Profit & Loss (P&L)	6856
Annualized Return	25.92%
Annualized Volatility	24.82%
Sharpe Ratio	1.178
Sortino Ratio	1.801
Maximum Drawdown	14.86%
Maximum Drawdown Duration	20 days
Profitability	42.31%
Ratio Average Profit/Loss	3.181
Skewness	0.404

Performance Indicator	MAMR (Testing)
Profit & Loss (P&L)	-3460
Annualized Return	-19.08%
Annualized Volatility	28.28%
Sharpe Ratio	-0.610
Sortino Ratio	-0.813
Maximum Drawdown	51.09%
Maximum Drawdown Duration	204 days
Profitability	56.67%
Ratio Average Profit/Loss	0.492
Skewness	-0.291

Figure 7 - Evaluation de performance en phase de test des méthodes classiques pour Apple

Performance Indicator	B&H (Testing)
Profit & Loss (P&L)	2960
Annualized Return	24.06%
Annualized Volatility	53.05%
Sharpe Ratio	0.507
Sortino Ratio	0.741
Maximum Drawdown	52.76%
Maximum Drawdown Duration	205 days
Profitability	100.00%
Ratio Average Profit/Loss	inf
Skewness	0.542

Performance Indicator	S&H (Testing)
Profit & Loss (P&L)	-2980
Annualized Return	-7.38%
Annualized Volatility	46.04%
Sharpe Ratio	-0.154
Sortino Ratio	-0.205
Maximum Drawdown	54.04%
Maximum Drawdown Duration	144 days
Profitability	0.00%
Ratio Average Profit/Loss	0.000
Skewness	-0.024

Performance Indicator	MATF (Testing)
Profit & Loss (P&L)	-7327
Annualized Return	-100.00%
Annualized Volatility	52.61%
Sharpe Ratio	-0.989
Sortino Ratio	-1.231
Maximum Drawdown	79.85%
Maximum Drawdown Duration	229 days
Profitability	34.38%
Ratio Average Profit/Loss	0.533
Skewness	-0.309

Performance Indicator	MAMR (Testing)
Profit & Loss (P&L)	854
Annualized Return	18.99%
Annualized Volatility	58.03%
Sharpe Ratio	0.358
Sortino Ratio	0.538
Maximum Drawdown	65.30%
Maximum Drawdown Duration	159 days
Profitability	67.65%
Ratio Average Profit/Loss	0.496
Skewness	0.550

Figure 8 - Evaluation de performance en phase de test des méthodes classiques pour Tesla

Les figures 7 et 8 montrent les résultats pour les méthodes de *trading* classique (hors IA). Nous voyons qu'il y a beaucoup de changement selon les techniques adoptées. En effet, pour une haute volatilité comme Tesla, la technique de MATF, qui suit une tendance fonctionne moins bien que sur Apple (moins volatile), de même pour la technique de *Buy & Hold*. Les moyennes mobiles et les *Sell & Hold* ne donnent pas de bons résultats pour Apple, contrairement à TDQN dans le rapport précédent, nous avons vu que l'agent basé de TDQN pouvait mieux performer. Concernant Tesla, *Buy & Hold* et les moyennes mobiles fonctionnent mieux comparé aux autres, les résultats pourraient être confrontés à ceux de TDQN. Dans tous les cas PPO n'excelle pas TDQN, ni les techniques de *trading* classiques.

La solution d'une amélioration par PPO pour augmenter la stabilité du modèle et la vitesse d'apprentissage n'a pas donné de résultats satisfaisants. Les gains du portfolio sont fortement impactés par cette stratégie, l'agent ne fait que perdre les biens qu'il possède. Les métriques tels que le ratio de Sharpe, de Sortino, le profit moyen et total indiquent clairement une dégradation de la qualité des décisions de l'agent. Nous avons vu que la volatilité pouvait être un avantage pour l'agent s'il est entraîné à profiter de ces tendances, cependant ici l'agent n'est pas capable d'utiliser cette opportunité, la volatilité dans ce modèle impacte les performances de l'agent, et empire les résultats, en essayant de changer la taille du réseau, les conclusions sont similaires.

Conclusion. L'objectif était d'appliquer la méthode *Proximal Policy Optimization* et de la comparer avec le *Trading Deep Q-Network* proposé par *T. Théate et D. Ernst* sur le problème du *trading* algorithmique. L'application de la méthode PPO augmente la complexité de l'implémentation, mais ceci montre qu'elle peut être appliquée au marché boursier et que l'agent peut apprendre à agir sur le marché indépendamment des résultats obtenus. Les résultats sont moins bons que ceux obtenus avec le TDQN et ceux obtenus par méthode classique de *trading*, en particulier, les réglages effectués par les auteurs du TDQN de l'article source sont bien meilleurs que ceux apportés au PPO à l'étape actuelle. Par faute de temps, l'investigation sur les réglages des paramètres n'a pas été poussée en profondeur, car l'agent PPO apprend à agir même si la performance moyenne conduit l'agent à perdre de l'argent par rapport au capital initial. Il est envisageable qu'un ajustement d'un certain critère ou paramètre pourrait inverser l'effet. Une technique d'entraînement différente pourrait améliorer les performances. Par les analyses précédentes, nous pouvons conclure que le *bot* basé sur un PPO a un faible potentiel en *trading*, du moins pour le modèle conçu. Les pistes d'amélioration possible sont les suivantes, un changement complet du réseau de neurones en intégrant un réseau LSTM ou bien TDRQN. L'avantage est que ces réseaux sont récurrents, cela va garder en mémoire les résultats précédents de la série temporelle (*stocks*), l'avantage de cela aux réseaux à convolution classique est qu'ils possèdent un effet mémoire à long terme. Des liens, des corrélations entre des sous-séquences éloignées pourront être caractérisés et être appris par le réseau. De plus, d'autres métriques d'analyse pourraient être introduites comme *Moving Average Convergence Divergence* (MACD) et *Relative Strength Index* (RSI) dans la limite du temps disponible.