IAR Mini-Project 01

Tuning TD3 for LunarLanderContinuous-v2

09/10/2024

KADEM Hocine 21309534
NGUYEN Hervé 28625990

Introduction

Ce projet a pour objectif de comparer deux implémentations de l'algorithme TD3 appliqué à l'environnement `LunarLanderContinuous-v2`. Nous avons développé notre propre implémentation de TD3 et l'avons confrontée à celle de la bibliothèque `Stable Baselines3`.

Pour évaluer la performance des deux algorithmes, nous avons défini plusieurs critères d'évaluation : les pertes des acteurs (*actor loss*) et des critiques (*critic loss*), ainsi que les récompenses moyennes obtenues. Ces critères permettent de juger la qualité de l'apprentissage en mesurant la précision des politiques apprises et la stabilité des évaluations.

Notre méthodologie se divise en plusieurs étapes :

- Définir et implémenter l'algorithme TD3 avec des hyperparamètres standards.
- Entraîner et tester l'algorithme sur l'environnement `LunarLanderContinuous-v2`.
- Comparer nos résultats à ceux obtenus avec `Stable Baselines3`.
- Effectuer des tests et analyser les différences de performance.

Le code source de nos travaux se trouve dans notre git :
github.com/TsunomakiWatamelon/M2_IAR_MP1_LunarLanderTuning

1 Rappel de l'algorithme TD3

L'algorithme TD3 (Twin Delayed Deep Deterministic Policy Gradient) est le successeur de DDPG (Deep Deterministic Policy Gradient), il est conçu pour améliorer la stabilité et l'efficacité de l'apprentissage par rapport à DDPG.

L'algorithme TD3 repose sur deux réseaux de critiques indépendants, Q_{ϕ_1} et Q_{ϕ_2}, qui estiment les valeurs d'état-action $Q(s,a)$. L'utilisation de ces deux réseaux permet de réduire le risque de surestimation des valeurs Q, un problème fréquemment rencontré dans DDPG. Lors de la mise à jour des critiques, la valeur cible est calculée selon l'équation suivante :

$$y(r,s',d) = r + \gamma(1-d)\min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s'))$$

où $\phi_{1,\text{targ}}$ et $\phi_{2,\text{targ}}$ sont les paramètres des réseaux critiques cibles, et $a'(s')$ représente l'action suivie d'un bruit.

Les transitions (s,a,r,s') sont stockées dans un *replay buffer* $\mathcal{D}$, afin de casser la corrélation entre les échantillons successifs. Ces transitions sont échantillonnées aléatoirement pour effectuer les mises à jour des réseaux acteur et critiques.

Les critiques sont mises à jour en minimisant les pertes suivantes pour chaque réseau Q_{ϕ_1} et Q_{ϕ_2}, respectivement :

$$L(\phi_1, \mathcal{D}) = E_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_{\phi_1}(s,a) - y(r,s',d)\right)^2\right],$$
$$L(\phi_2, \mathcal{D}) = E_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_{\phi_2}(s,a) - y(r,s',d)\right)^2\right].$$

Un acteur déterministe $\mu_\theta(s)$ est employé pour générer l'action optimale $a = \mu_\theta(s)$.
La mise à jour des paramètres de l'acteur est effectuée tous les d. Et on met à jour l'acteur en minimisant la perte suivante :

$$-E_{s\sim\mathcal{D}}\left[Q_{\phi_1}(s, \mu_\theta(s))\right]$$

Les réseaux cibles sont mis à jour progressivement via une *mise à jour douce*, pour garantir la stabilité de l'apprentissage. La règle de mise à jour pour un paramètre donné ϕ est définie comme suit :

$$\phi_{\text{targ}} \leftarrow \tau\phi + (1-\tau)\phi_{\text{targ}},$$

où $\tau \in [0, 1]$ est un facteur de mise à jour, souvent une petite constante.

2 Premier entraînement

Pour le premier entraînement de notre implémentation TD3, nous avons choisi des hyperparamètres standards afin de garantir un apprentissage stable et initialiser correctement le modèle. Les hyperparamètres principaux incluent :

- **Taille du batch** : 128, permettant de stabiliser les gradients.
- **Taux d'apprentissage de l'acteur** : 1e-3.
- **Taux d'apprentissage des critiques** : 1e-3.
- **Mise à jour douce des cibles** : 0.005, assurant une meilleure convergence des réseaux critiques.
- **Facteur de discount** : 0.99, pour un horizon temporel plus long.
- **Taille du replay buffer** : 1 million, afin d'assurer une diversité suffisante dans l'échantillonnage des transitions.

L'objectif de cet entraînement préliminaire était d'observer les premières tendances d'apprentissage en utilisant un ensemble de paramètres basiques. Nous avons mesuré l'évolution des pertes des acteurs et des critiques, ainsi que les récompenses moyennes sur 100 000 étapes.
Les résultats de cet entraînement sont présentés dans la figure suivante :

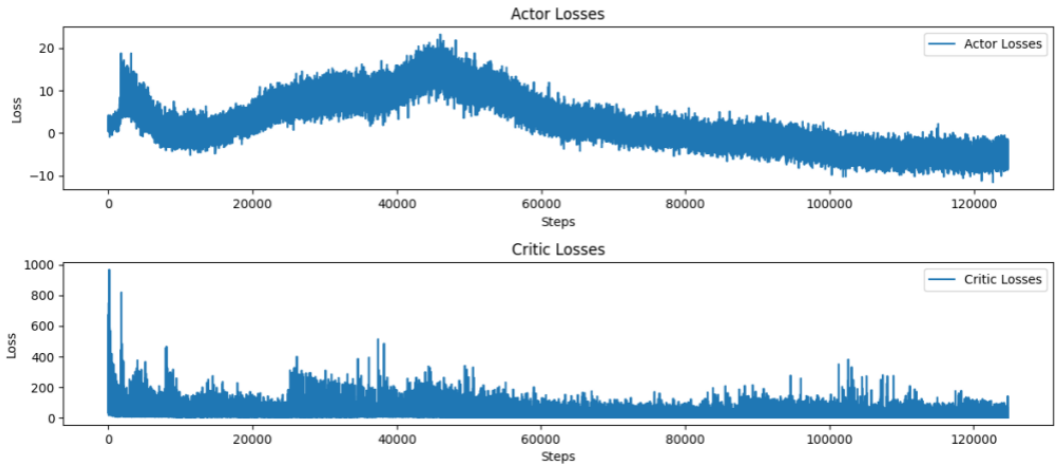


Figure 1: Pertes des acteurs et des critiques durant le premier entraînement de TD3 sur LunarLanderContinuous-v2

L'analyse des courbes montre que l'algorithme TD3 apprend correctement, bien que nous observions une augmentation initiale des pertes de l'acteur et des critiques. Cette augmentation peut être attribuée à la phase d'exploration intense de l'agent au début de l'entraînement, où l'agent tente d'essayer différentes actions pour découvrir les stratégies optimales. Au fur et à mesure que l'apprentissage progresse, les pertes de l'acteur diminuent progressivement, ce qui indique que l'agent affine de mieux en mieux ses actions et commence à exploiter les politiques apprises de manière plus efficace.

Les pertes des critiques, quant à elles, montrent des fluctuations marquées au début de l'entraînement. Cependant, une tendance globale à la baisse se dessine au fil du temps, suggérant que le modèle converge progressivement vers une évaluation plus stable des récompenses et des politiques.

Toutefois, ces premiers résultats doivent être consolidés par des expérimentations supplémentaires. Nous prévoyons d'exécuter l'algorithme sur plusieurs seeds pour vérifier la robustesse de l'apprentissage. De plus, l'analyse des récompenses moyennes sera incluse dans les futures étapes pour fournir des indications supplémentaires sur la performance globale et l'efficacité de l'agent dans l'environnement donné.

3 Comparaison entre notre implémentation et basicbasline3

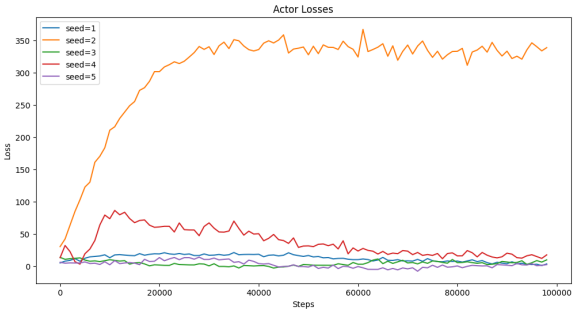
Après avoir effectué un premier entraînement pour vérifier le bon fonctionnement de notre implémentation, nous allons maintenant la comparer avec la bibliothèque `basicbasline3`. Pour minimiser les biais, nous avons utilisé 5 runs d'entraînement avec des seeds différentes [1, 2, 3, 4, 5]. Bien qu'il soit préférable d'utiliser un plus grand nombre de seeds, les contraintes de temps et de matériel nous ont limité à 5 seeds.

Nous avons choisi d'entraîner les deux implémentations avec les mêmes hyperparamètres, en utilisant les paramètres fournis par `basicbasline3` :

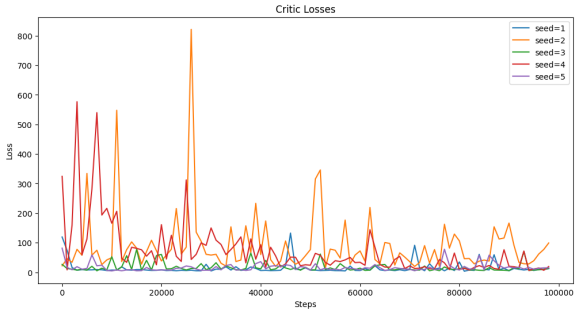
- Taux d'apprentissage (`learning_rate`) : 0.001
- Taille du buffer (`buffer_size`) : 1 000 000
- Nombre d'étapes avant le début de l'apprentissage (`learning_starts`) : 100
- Taille du lot (`batch_size`) : 256
- Facteur de soft-update (`tau`) : 0.005
- Facteur de discount (`gamma`) : 0.99

Nous avons enregistré les gains moyens, ainsi que les pertes de l'acteur et des critiques tous les 1000 pas d'apprentissage. L'acteur et le critique sont mis à jour à cette fréquence pour permettre à l'agent de mieux explorer l'environnement tout en ajustant les actions de manière régulière sans surentraînement excessif. Cela permet de maintenir un équilibre entre exploration et exploitation durant l'apprentissage. L'entraînement a été effectué sur 100 000 pas pour les deux implémentations.

3.1 Résultats de `basicbasline3`



(a) Pertes des acteurs pour sur 100 000 pas d'apprentissage



(b) Pertes des critiques pour sur 100 000 pas d'apprentissage

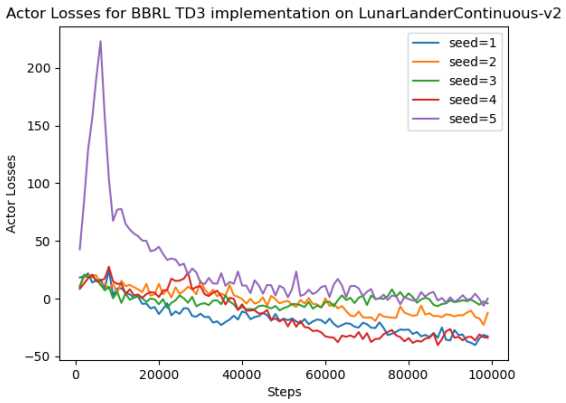
(c) Récompenses moyennes pour sur 100 000 pas d'apprentissage

En observant les pertes des acteurs pour `Stable Baselines3`, nous remarquons que certaines seeds montrent une augmentation anormalement élevée des pertes avant de se stabiliser. Cette fluctuation pourrait indiquer que l'entraînement de l'agent pour ces seeds est moins efficace que pour d'autres, où la perte diminue de manière plus progressive. Cette variabilité peut être due à des différences dans les conditions initiales d'apprentissage ou à une exploration moins bien contrôlée. Dans l'ensemble, malgré ces disparités, les courbes montrent une tendance à la réduction des pertes au fur et à mesure que l'agent affine ses actions.

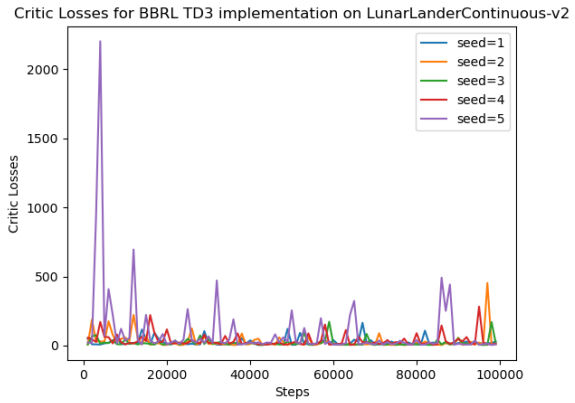
Les pertes des critiques présentent une grande variabilité, notamment avec des pics significatifs pour certaines seeds. Ces fluctuations, particulièrement au début de l'apprentissage, suggèrent que l'agent rencontre des difficultés à évaluer correctement les récompenses. Bien que la tendance globale soit à la baisse, des oscillations importantes subsistent même après 100 000 pas d'apprentissage, indiquant que l'agent n'a pas complètement stabilisé son évaluation des valeurs.

Les récompenses moyennes, quant à elles, montrent peu d'amélioration significative tout au long de l'entraînement. Après environ 50 000 pas, la courbe de récompense semble se stabiliser, mais certaines seeds affichent des chutes importantes, suggérant une exploration prolongée ou des ajustements de politique qui ne conduisent pas nécessairement à une maximisation efficace des récompenses. Cette faible amélioration des récompenses montre que l'agent n'exploite pas toujours les stratégies les plus optimales.

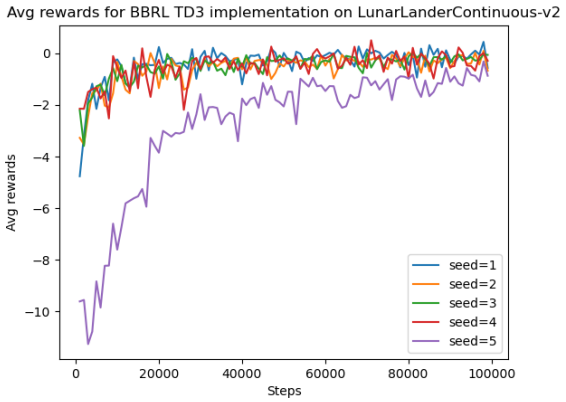
3.2 Résultats de notre implémentation



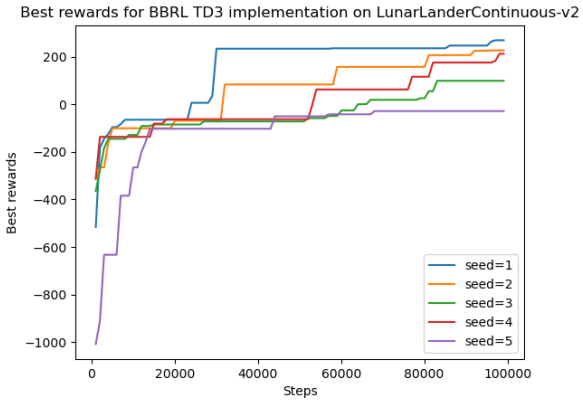
(a) Pertes des acteurs pour notre implémentation sur 100 000 pas d'apprentissage



(b) Pertes des critiques pour notre implémentation sur 100 000 pas



(c) Récompenses moyennes pour notre implémentation sur 100 000 pas



(d) Meilleure récompense pour notre implémentation sur 100 000 pas

Pour notre implémentation, les pertes des acteurs suivent une tendance similaire à celle observée avec `Stable Baselines3`, avec une augmentation initiale des pertes qui diminue progressivement à mesure que l'agent apprend à optimiser ses actions. Cependant, certaines seeds montrent moins de fluctuations extrêmes, indiquant un apprentissage globalement plus régulier.

Concernant les pertes des critiques, bien que le comportement soit à peu près similaire à celui observé avec `Stable Baselines3`, nous constatons moins d'oscillations marquées. Cela suggère que notre implémentation parvient à une meilleure stabilité dans l'estimation des valeurs d'état-action, avec une convergence plus rapide, ce qui pourrait indiquer une meilleure réactivité aux ajustements dans les premières étapes de l'apprentissage.

Les récompenses moyennes montrent une tendance similaire à la stabilisation. Cependant, comme pour `Stable Baselines3`, certaines seeds présentent des chutes brusques dans les récompenses, ce qui laisse penser que l'agent continue d'explorer même après avoir trouvé une stratégie efficace. Néanmoins, l'amélioration globale des récompenses semble modérée, suggérant que l'exploration ne conduit pas toujours à une maximisation optimale des récompenses.

3.3 Première comparaison entre notre implémentation et `Stable Baselines3`

En comparant les résultats des deux implémentations, nous observons plusieurs similarités et différences. D'une part, les pertes des acteurs dans les deux cas montrent une réduction progressive après

une phase d'exploration initiale, ce qui est typique dans les algorithmes d'apprentissage par renforcement. Cependant, dans notre implémentation, certaines seeds affichent une augmentation anormale avant de se stabiliser, indiquant que l'apprentissage n'est pas aussi régulier pour toutes les initialisations. Cela contraste avec `Stable Baselines3`, qui présente une stabilisation plus uniforme des pertes des acteurs, bien que certaines seeds montrent des pertes plus élevées que d'autres.

En ce qui concerne les pertes des critiques, notre implémentation montre des résultats globalement similaires à `Stable Baselines3`, mais avec moins d'oscillations. Alors que les deux implémentations affichent des pics de pertes dans les premières étapes de l'apprentissage, notre implémentation converge plus rapidement et de manière plus stable. Cela pourrait indiquer que notre implémentation est plus réactive aux ajustements au cours des premières phases de l'entraînement.

Les récompenses moyennes pour les deux implémentations montrent une tendance à la stabilisation après environ 50 000 pas. Cependant, pour `Stable Baselines3`, certaines seeds montrent des fluctuations plus importantes, avec des chutes de récompenses plus fréquentes avant de se stabiliser à nouveau. Dans notre implémentation, les récompenses ne semblent pas s'améliorer de manière aussi significative, ce qui suggère que l'agent continue d'explorer de nouvelles stratégies, mais sans aboutir systématiquement à une politique plus optimale.

Globalement, notre implémentation semble apprendre plus rapidement dans les premières étapes de l'entraînement, particulièrement en ce qui concerne la stabilisation des critiques. Cependant, des fluctuations plus importantes dans certaines seeds et une amélioration modérée des récompenses laissent supposer que notre implémentation pourrait bénéficier d'ajustements supplémentaires, notamment dans la gestion de l'exploration et de l'exploitation. Des tests supplémentaires avec un plus grand nombre de seeds seraient nécessaires pour confirmer ces observations et déterminer si ces différences sont significatives. s

4 Tests statistiques

Après avoir entraîné notre implémentation de TD3 sur l'environnement `LunarLanderContinuous-v2` en utilisant plusieurs seeds, tant avec notre implémentation qu'avec celle de `Stable Baselines3`, nous avons entrepris une analyse statistique pour approfondir notre étude et valider (ou non) nos hypothèses précédentes.

Afin de comparer de manière rigoureuse les performances des deux implémentations, nous avons réalisé des tests statistiques, notamment le *Welch's t-test*. Ce test permet de déterminer si les différences observées entre les performances des deux implémentations sont significatives, en tenant compte de la variance potentiellement différente entre les groupes de données.

Les résultats de notre implémentation de TD3 et ceux obtenus avec `Stable Baselines3` ont été comparés en utilisant les mêmes seeds pour garantir une base de comparaison équitable. Pour chaque seed, nous avons mesuré des indicateurs de performance tels que les rewards moyens, la loss de l'acteur et des critiques, ainsi que la somme des *losses* des critiques. Ces données ont ensuite été soumises à une analyse statistique via le *Welch's t-test* afin d'évaluer si les écarts observés dans les performances des agents sont statistiquement significatifs.

Le choix du *Welch's t-test* s'explique par sa capacité à gérer des variances inégales et des tailles d'échantillons différentes entre les deux implémentations, garantissant ainsi une comparaison robuste des performances.

4.1 Évolution des pertes des acteurs

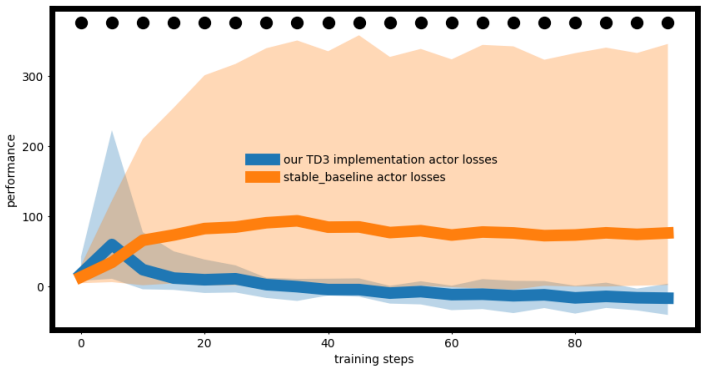


Figure 4: Évolution des pertes des acteurs sur 100 000 pas d'apprentissage pour 5 seeds

En observant l'évolution des pertes des acteurs, nous remarquons que notre implémentation de TD3 semble obtenir des pertes plus faibles que celle de Stable Baselines3 sur la majorité des étapes d'apprentissage. Cela pourrait indiquer que notre modèle parvient à mieux ajuster les actions de l'agent, mais cette observation doit être nuancée par le fait que les deux courbes restent relativement proches. La variance plus élevée observée avec `Stable Baselines3` pourrait également suggérer que la stabilité de notre implémentation est légèrement supérieure.

Cependant, il est important de rester prudent. Une perte plus faible n'implique pas nécessairement une meilleure performance globale, car cela peut parfois signifier une sous-estimation des valeurs critiques. Il est donc nécessaire d'étudier d'autres critères pour confirmer cette tendance.

4.2 Évolution des pertes des critiques

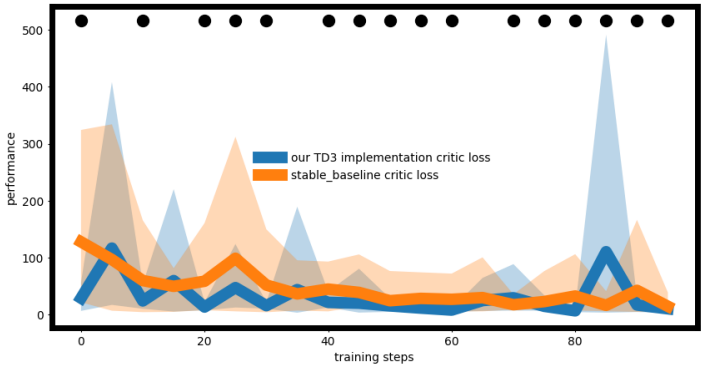


Figure 5: Évolution des pertes des critiques sur 100 000 pas d'apprentissage pour 5 seeds

L'analyse des pertes des critiques révèle que notre implémentation et celle de Stable Baselines3 présentent des comportements similaires, avec une tendance globale à la diminution des pertes au fur et à mesure des étapes d'apprentissage. Néanmoins, on observe quelques pics de perte plus marqués avec notre implémentation, ce qui pourrait traduire une plus grande sensibilité aux fluctuations du gradient.

Ces pics peuvent suggérer que notre implémentation est plus réactive aux changements dans les valeurs des états-action, mais cela pourrait également introduire une certaine instabilité. L'analyse de ces fluctuations pourrait indiquer la nécessité d'ajuster certains hyperparamètres pour améliorer la stabilité.

4.3 Récompenses moyennes

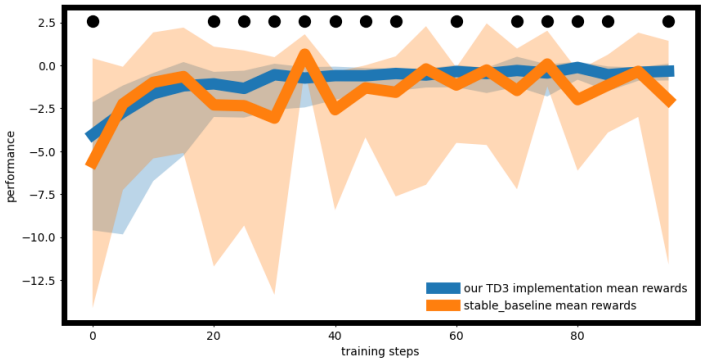


Figure 6: Récompenses moyennes sur 100 000 pas d'apprentissage pour 5 seeds

Les récompenses moyennes obtenues avec notre implémentation et avec `Stable Baselines3` suivent des tendances assez similaires. Toutefois, on remarque que `Stable Baselines3` commence avec un léger avantage en termes de récompenses, mais que notre implémentation rattrape rapidement cette différence et parvient à maintenir une performance constante.

Cette observation suggère que notre modèle est tout aussi capable de maximiser les récompenses que celui de `Stable Baselines3`, bien qu'il prenne un peu plus de temps à atteindre une performance optimale. La variance importante des deux courbes montre également que l'entraînement reste sujet à des fluctuations significatives, ce qui peut affecter la robustesse des résultats.

4.4 Conclusion des tests statistiques

D'après les analyses précedente, nous pouvons conclure que notre implémentation de TD3 présente des performances comparables à celles de `Stable Baselines3` sur les critères de perte des acteurs, des critiques, et de récompenses moyennes. Bien que notre implémentation semble offrir une stabilité légèrement supérieure au niveau des pertes des acteurs, elle montre aussi une plus grande variabilité dans les pertes des critiques.

Globalement, les différences entre les deux implémentations ne sont pas suffisamment marquées pour affirmer une supériorité claire de l'une ou l'autre. Il serait intéressant de poursuivre cette étude en explorant d'autres paramètres, comme la convergence des politiques ou l'impact de la variance des critiques, afin de mieux comprendre les comportements observés.

Conclusion

Ce projet avait pour objectif de comparer deux implémentations de l'algorithme TD3, appliquées à l'environnement `LunarLanderContinuous-v2`, afin de mieux comprendre leurs performances respectives. Nous avons implémenté notre propre version de TD3 et l'avons confrontée à celle de la bibliothèque `Stable Baselines3`. Plusieurs critères d'évaluation, tels que les pertes des acteurs, les pertes des critiques, et les récompenses moyennes, ont été utilisés pour juger la qualité de l'apprentissage de chaque implémentation.

Globalement, nos résultats montrent que les deux implémentations affichent des performances comparables. Les deux implémentations montrent des performances similaires en termes de pertes des acteurs et des critiques, ainsi que des récompenses moyennes. Cependant, notre implémentation a montré une meilleure stabilité dans les pertes des critiques, tandis que `Stable Baselines3` a offert une gestion plus uniforme des pertes des acteurs.

Les tests statistiques réalisés, notamment le Welch's t-test, confirment que les écarts observés entre les deux implémentations ne sont pas significativement distincts. Néanmoins, des fluctuations observées dans certaines seeds et des différences mineures dans la convergence méritent des explorations plus approfondies.

Il serait pertinent de poursuivre cette étude en examinant d'autres aspects, comme la convergence des politiques ou l'impact de la variance des critiques, afin d'obtenir une meilleure compréhension des dynamiques d'apprentissage et de l'efficacité des algorithmes dans divers scénarios.