RL Mini-Project 02

Impact of Wrappers on Partially Observable Environments: Training and Analysis of DDPG and TD3 Algorithms

09/10/2024

KADEM Hocine 21309534
BENAHMED Neil 21200977

1 Introduction

Le but de ce projet est d'étudier les algorithmes TD3 (Twin Delayed DDPG) et DDPG (Deep Deterministic Policy Gradient) dans des environnements partiellement observables et d'analyser si les deux wrappers développés, à savoir le *ActionTimeExtensionWrapper* et le *ObsTimeExtensionWrapper*, peuvent aider à améliorer les performances dans ces contextes.

Dans un environnement partiellement observable, certaines informations cruciales sur l'état ne sont pas disponibles, ce qui rend difficile l'apprentissage de politiques optimales. Pour répondre à ce défi, on a conçu un *FeatureFilterWrapper* permettant de masquer certaines caractéristiques de l'observation, rendant ainsi l'environnement partiellement observable. Ensuite, on a utilisé l'*ActionTimeExtensionWrapper* et l'*ObsTimeExtensionWrapper* pour analyser leur impact dans ces environnements spécifiques.

2 Méthodologie

Pour formuler des hypothèses, on a suivi une méthodologie structurée en plusieurs étapes :

1. **Tests dans des environnements partiellement et complètement observables** : On a testé les algorithmes TD3 et DDPG dans un environnement complètement observable comme point de référence. Ensuite, on a réalisé des tests dans des environnements partiellement observables, où certaines informations d'état sont masquées à l'agent. Pour cela, on a retiré d'abord la dérivée de la position ($\dot{x}$), puis la vitesse angulaire ($\dot{\theta}$), et enfin, on a retiré les deux simultanément, créant ainsi des conditions de perception plus difficiles pour l'agent.

2. **Étude des wrappers** : On a étudié les deux wrappers que l'on a implémentés. On a identifié que le nombre d'actions répétées dans l'*ActionTimeExtensionWrapper* et la taille de la mémoire dans l'*ObsTimeExtensionWrapper* sont des paramètres critiques. En effet, dans un environnement partiellement observable, répéter certaines actions pourrait stabiliser les décisions prises par l'agent, tandis que l'extension de la mémoire d'observation permettrait à l'agent de mieux percevoir l'état sous-jacent en conservant les informations passées. On a donc décidé de tester plusieurs configurations, avec différentes tailles de mémoire et valeurs d'actions répétées pour déterminer celles qui sont optimales.

3. **Tests avec les wrappers dans les environnements partiellement observables** : Finalement, on a appliqué les wrappers dans ces environnements partiellement observables. On a testé l'environnement sans $\dot{x}$, puis sans θ, et finalement sans les deux. À chaque étape, on a utilisé l'*ActionTimeExtensionWrapper* pour voir l'impact de la répétition d'actions, puis l'*ObsTimeExtensionWrapper* pour évaluer l'effet de la mémoire temporelle. Enfin, on a combiné les deux wrappers pour voir si leur utilisation conjointe pouvait améliorer encore plus les performances des algorithmes dans ces environnements partiellement observables.

3 Hyperparamètres Choisis

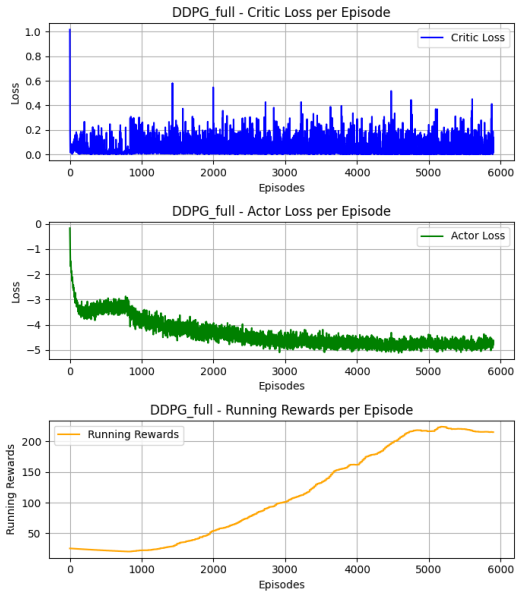
Pour mener à bien notre étude, on a sélectionné ces paramètres.

- **Seed** : La graine aléatoire a été fixée à 1 afin d'assurer la reproductibilité des résultats.
- **Max Grad Norm** : La normalisation des gradients a été fixée à 0.5 pour éviter l'explosion des gradients, ce qui contribue à la stabilité de l'apprentissage.
- **Epsilon** : Un paramètre d'exploration de 0.02.
- **Nombre d'Étapes par Environnement par Époque (n_steps)** : 100 étapes.
- **Taille du Buffer de Replay (buffer_size)** : Un buffer de 1e6 transitions.

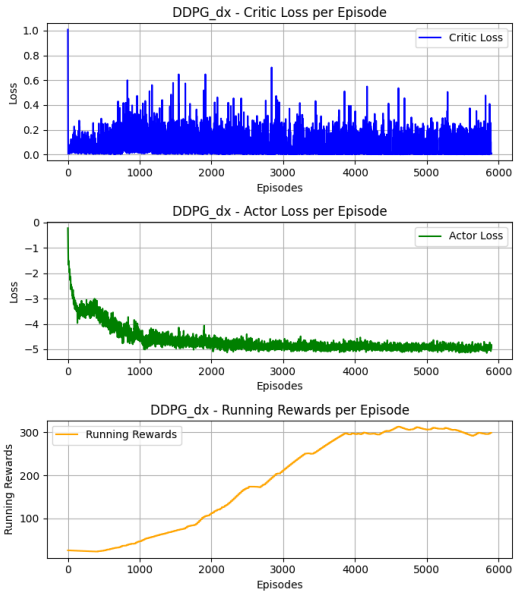
- **Taille du Batch (batch_size)** : On a choisi une taille de batch de 64.
- **Intervalle d'Évaluation (eval_interval)** : Les évaluations de l'agent ont été réalisées toutes les 2000 étapes.
- **Nombre Maximal d'Époques (max_epochs)** : On a fixé le nombre maximal d'époques à 6000 pour permettre un apprentissage approfondi sans entraîner excessivement l'agent.
- **Apprentissage Commence Après (learning_starts)** : L'agent commence à apprendre après avoir collecté 10 000 transitions.
- **Action Noise** : Une intensité de bruit d'action de 0.1.
- **Architecture des Réseaux** : On a utilisé des réseaux d'acteurs et de critiques avec des tailles cachées de [400, 300].
- **Taux d'Apprentissage (lr)** : Un taux d'apprentissage de 1e-3 a été fixé pour l'acteur et le critique.

4 Analyse de l'impact de la réduction d'observabilité sur l'apprentissage

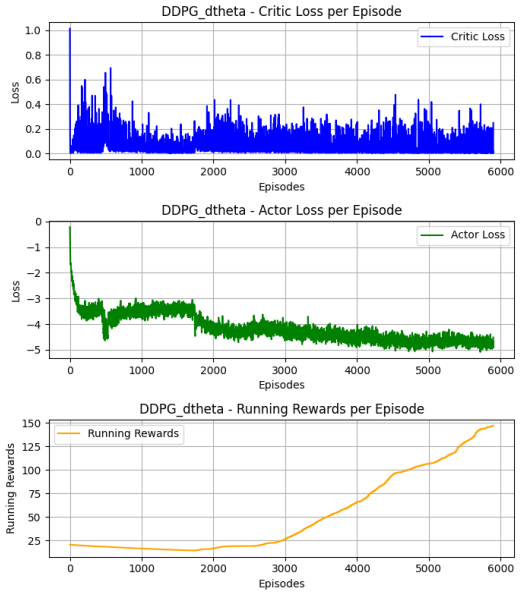
Nous avons testé TD3 et DDPG avec les environnements complètement observables et partiellement observables. Les figures ci-dessous montrent les courbes de pertes (*Critic Loss* et *Actor Loss*) ainsi que les récompenses cumulées (*Running Rewards*) pour chaque scénario. Les *Running Rewards* représentent la moyenne récompenses obtenues sur plusieurs épisodes et permettent de mesurer la performance récente depuis un certain temps de l'agent sur la tâche. Une augmentation des *Running Rewards* est un indicateur d'apprentissage réussi, tandis qu'une diminution ou stagnation montre que l'agent n'a pas trouvé de politique optimale.



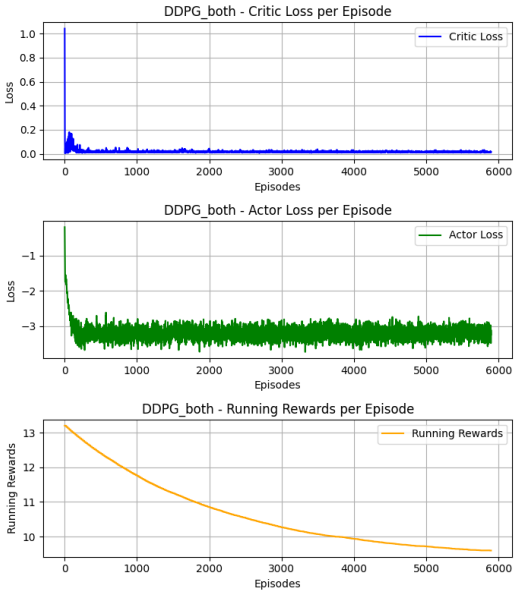
(a) DDPG - Environnement complet



(b) DDPG - Sans $\dot{x}$

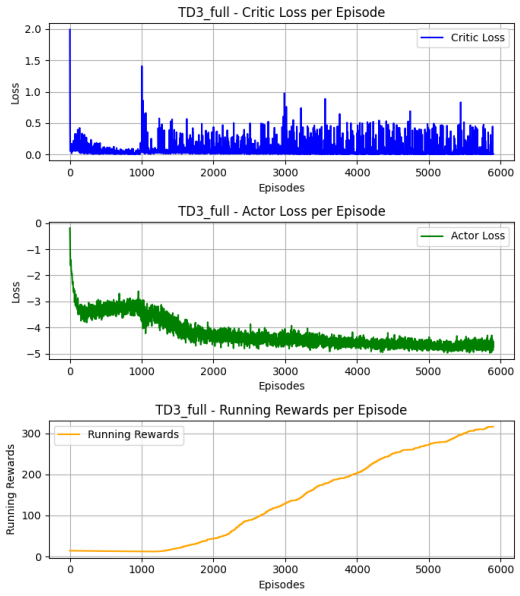


(c) DDPG - Sans θ

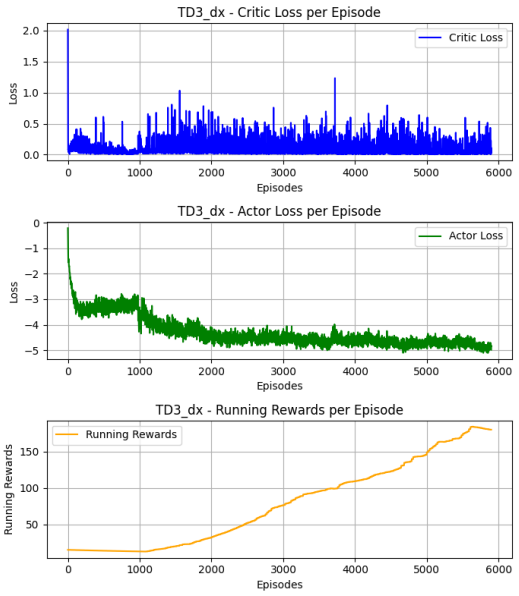


(d) DDPG - Sans $\dot{x}$ et θ

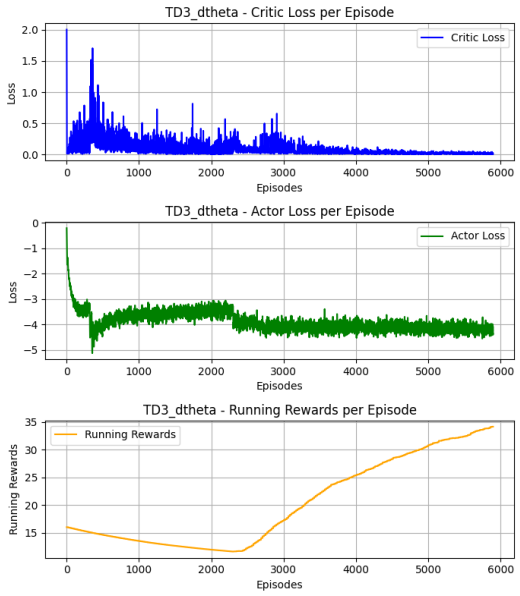
Figure 1: Courbes de performance DDPG : Critic Loss, Actor Loss et Running Rewards pour différents scénarios d'observation partielle.



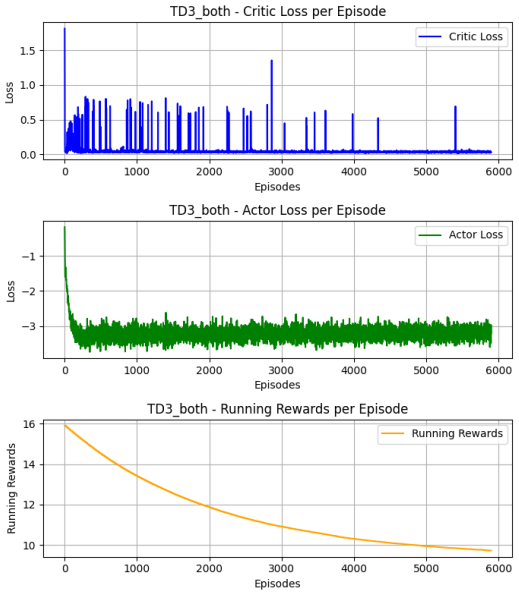
(a) TD3 - Environnement complet



(b) TD3 - Sans $\dot{x}$



(c) TD3 - Sans θ



(d) TD3 - Sans $\dot{x}$ et $\dot{\theta}$

Figure 2: Courbes de performance TD3 : Critic Loss, Actor Loss et Running Rewards pour différents scénarios d'observation partielle.

Analyse des Résultats En analysant les courbes des *Running Rewards*, on peut observer que l'environnement complètement observable (*DDPG_full* et *TD3_full*) donne les meilleures performances avec une progression constante des récompenses cumulées. Cela montre que l'agent est capable d'apprendre efficacement lorsqu'il a accès à toutes les informations.

En revanche, lorsque l'on enlève $\dot{x}$ (*DDPG_dx* et *TD3_dx*), on remarque que les performances de l'agent ne sont pas affectées de manière significative. Cela suggère que $\dot{x}$ n'est pas une caractéristique critique pour la tâche dans cet environnement.

Lorsque l'on enlève $\dot{\theta}$ (*DDPG_ dtheta* et *TD3_ dtheta*), on observe une légère dégradation des performances, mais l'agent continue de s'améliorer au fil du temps. Cela indique que θ est une caractéristique plus importante, mais que l'agent est toujours capable d'apprendre sans elle, bien que plus lentement.

Enfin, lorsque l'on enlève à la fois $\dot{x}$ et $\dot{\theta}$ (*DDPG_ both* et *TD3_ both*), on constate une chute significative des *Running Rewards*. Les agents semblent avoir des difficultés à apprendre une politique optimale, et la dégradation des performances est flagrante. Cela montre que la combinaison de ces deux caractéristiques est cruciale pour que l'agent puisse bien naviguer dans l'environnement.

En conclusion, $\dot{x}$ n'a pas un impact majeur sur l'apprentissage, alors que $\dot{\theta}$ joue un rôle plus important. Cependant, c'est surtout la combinaison des deux caractéristiques qui est essentielle pour des performances optimales.

5 Analyse de l'impact des paramètres des wrappers

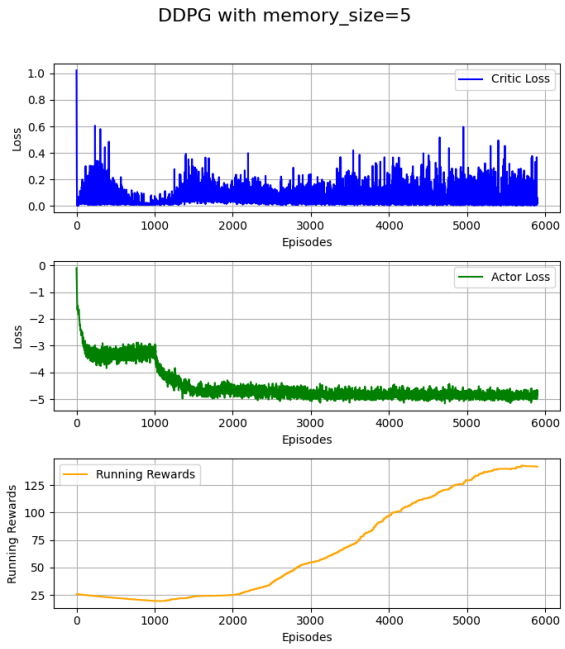
Dans cette section, nous analysons l'impact des paramètres des wrappers utilisés dans les algorithmes **DDPG** et **TD3** avec les parametres de l'environement semblable a ceux ennoncé précedemment. Les deux principaux paramètres étudiés sont :

- *memory size* (taille de l'historique des observations) pour le **ObsTimeExtensionWrapper**.
- *action repeat* pour le **ActionTimeExtensionWrapper**, qui augmente le nombre d'actions effectué.

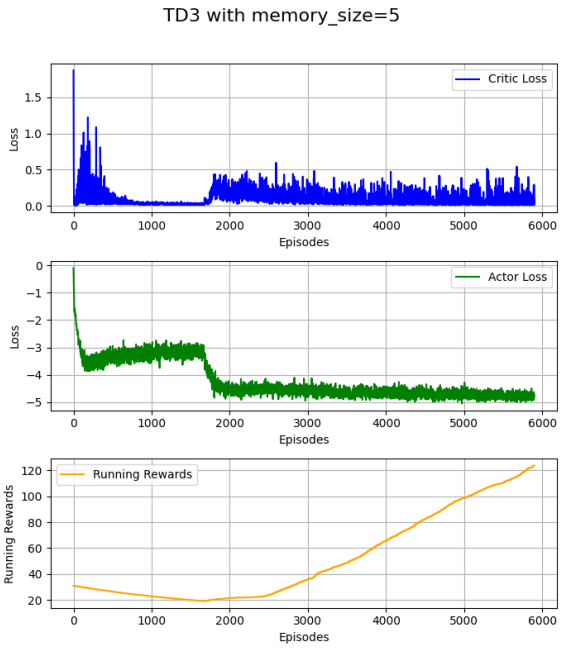
5.1 Impact du *memory size*

Protocole : Nous avons testé différentes valeurs de *memory size* dans un environnement partiellement observable (`CartPoleContinuous-v1`) avec les meme paramétres mentionnés précedement. Les métriques mesurées incluent *critic loss*, *actor loss*, et *running rewards* pour chaque valeur pour les deux algorithmes **DDPG** et **TD3**.
Résultats :

- Pour **DDPG**, les *critic losses* étaient élevées pour les petites valeurs de *memory size*, et les *rewards* se sont améliorés comparé à l'environement partiellement observables sans wrappers autres... Les valeures trouvés ne suivent pas selon notre comprehension un motif clair, néanmoins sur quelques tests nous obtenons une valeur de 5 pour le memory size qui semble assez bien performé contrairement aux autres valeures.
- Pour **TD3**, un comportement similaire a été observé.



(a) Learning curves de l'acteur, du critique et des running rewards pour l'algorithme DDPG



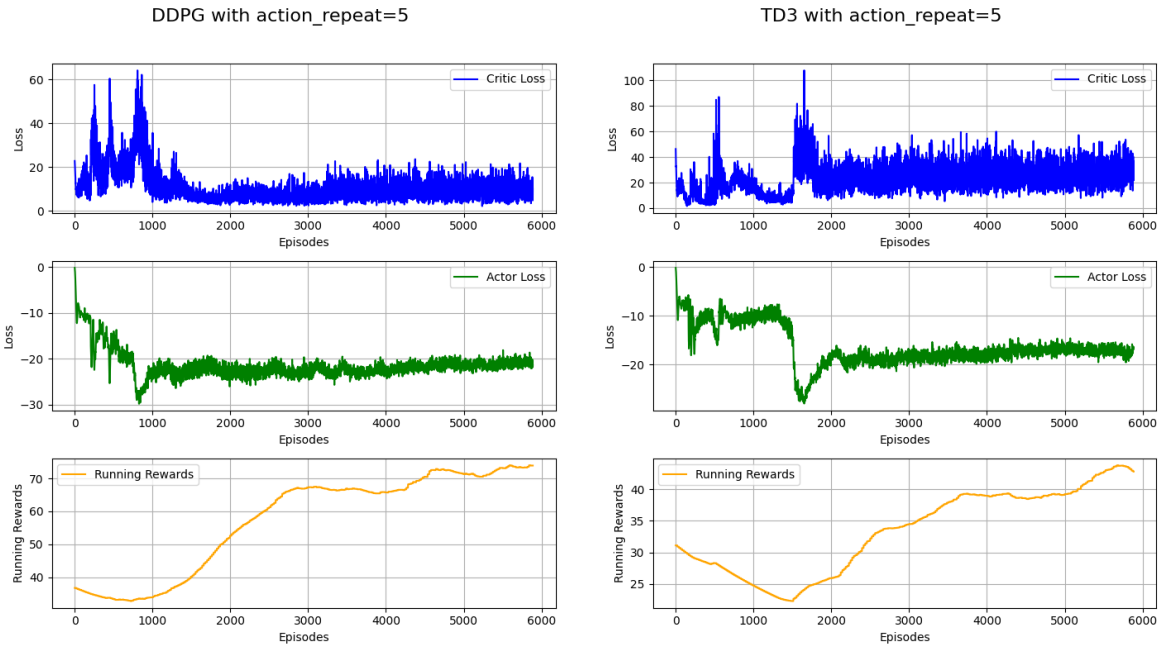
(b) Learning curves de l'acteur, du critique et des running rewards pour l'algorithme TD3

Figure 3: Learning curves pour les deux algorithmes TD3 et DDPG avec la valeure du paramétre memory size à 5

5.2 Impact du *action repeat*

Protocole : Nous avons varié *action repeat* également sur les mêmes algorithmes, en mesurant les mêmes métriques (*critic losses*, *actor losses*, *running rewards*).
Résultats :

- Pour **DDPG**, les valeurs de *action repeat* sont plus efficaces autour de 5. Sur plusieurs tests, nous avons observé que lorsque *action repeat* est trop élevé (vers 30 ou plus), les *running rewards* stagnent et ne s'améliorent pas avec le temps, ce qui suggère un problème de sous-exploration.
- Pour **TD3**, les valeurs de *action repeat* sont également optimales autour de 5. Cependant, des valeurs plus élevées empêchent l'algorithme de converger.*rewards.*



(a) Learning curves de l'acteur, du critique et des running rewards pour l'algorithme DDPG

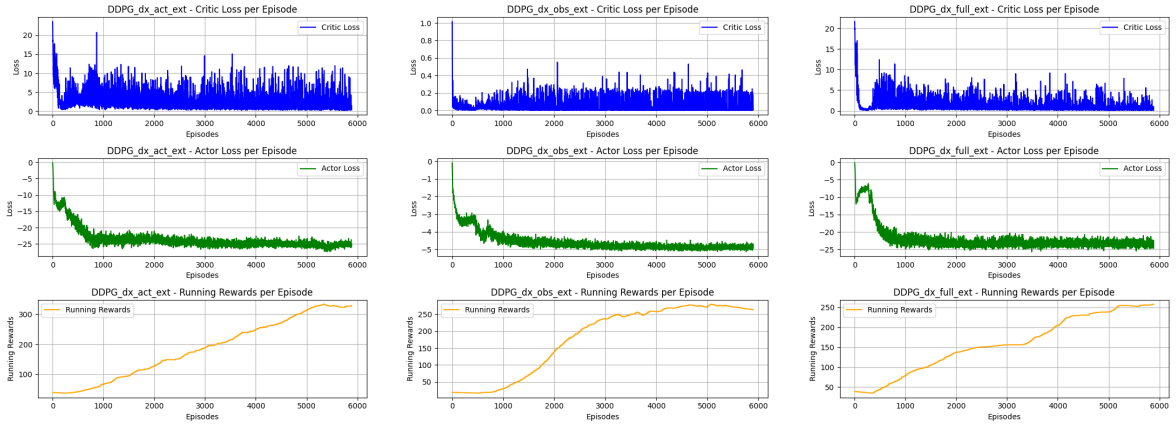
(b) Learning curves de l'acteur, du critique et des running rewards pour l'algorithme TD3

Figure 4: Learning curves pour les deux algorithmes TD3 et DDPG avec la valeure du paramétre action repeat à 5

Suggestions sur la cause : Lorsque *action repeat* est trop élevé, l'agent répète une action trop longtemps avant de réagir aux nouvelles observations, ce qui limite sa capacité de generalisation à l'environnement.

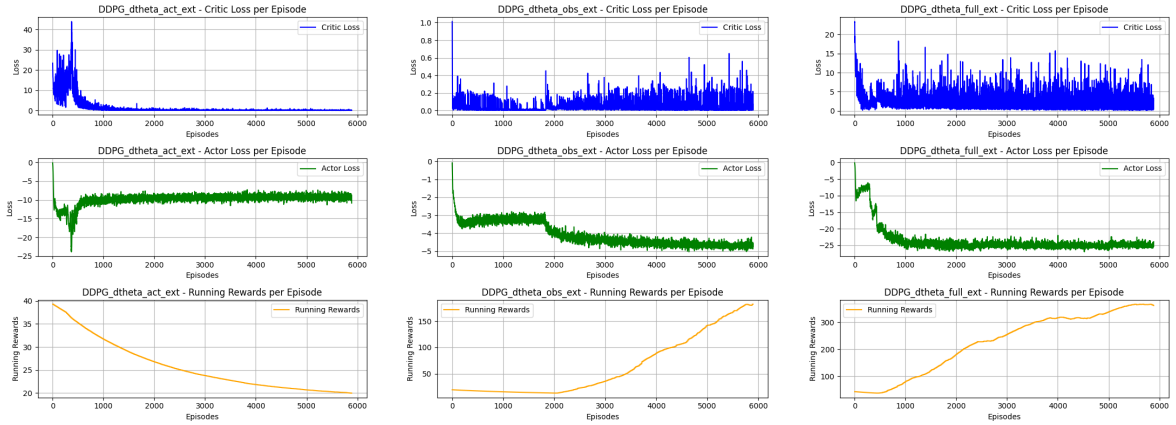
6 Analyse des performances avec les wrappers

D'après l'étude précédente, nous avons fixé la taille de la mémoire à 5 et le nombre de répétitions d'actions à 5. Dans cette section, nous avons testé les algorithmes DDPG et TD3 sur plusieurs environnements partiellement observables, notamment avec la suppression de $\dot{x}$, de $\dot{\theta}$, et des deux, tout en appliquant le *ActionTimeExtensionWrapper*, puis le *ObsTimeExtensionWrapper*, et finalement les deux simultanément.



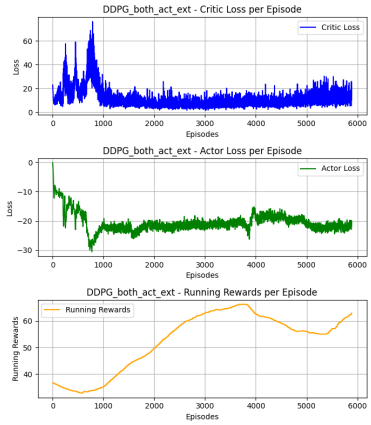
(a) DDPG - Without $\dot{x}$ (Action Wrapper) **(b)** DDPG - Without $\dot{x}$ (Obs Wrapper) **(c)** DDPG - Without $\dot{x}$ (Both Wrappers)

Figure 5: DDPG - Performance Without $\dot{x}$

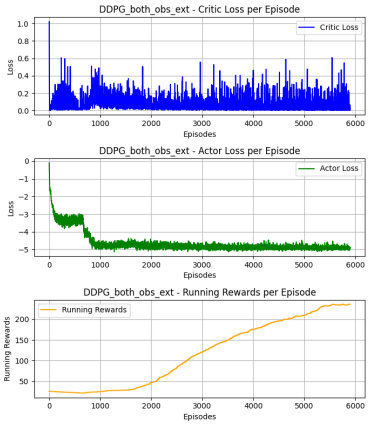


(a) DDPG - Without $\dot{\theta}$ (Action Wrapper) **(b)** DDPG - Without $\dot{\theta}$ (Obs Wrapper) **(c)** DDPG - Without $\dot{\theta}$ (Both Wrappers)

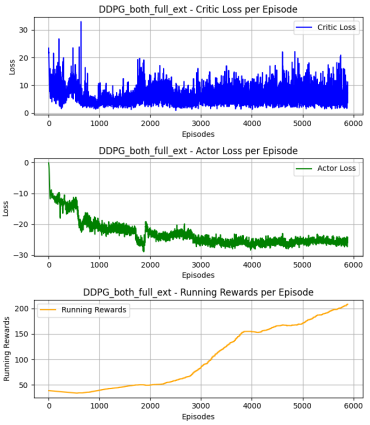
Figure 6: DDPG - Performance Without $\dot{\theta}$



(a) DDPG - Both (Action Wrapper)

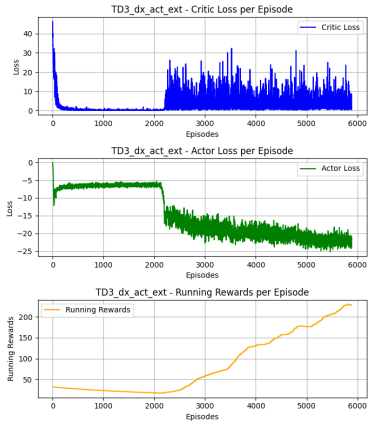


(b) DDPG - Both (Obs Wrapper)

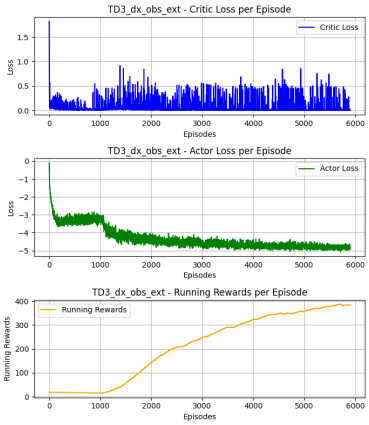


(c) DDPG - Both (Both Wrappers)

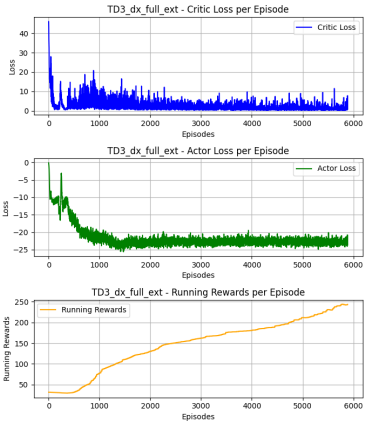
Figure 7: DDPG - Performance with Both Removed (Action, Obs, and Both Wrappers)



(a) TD3 - Without $\dot{x}$ (Action Wrapper)

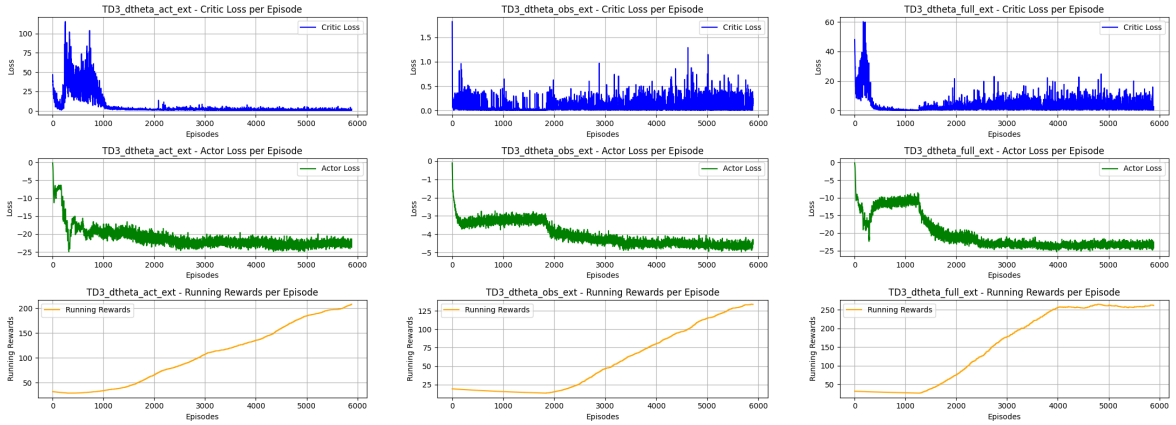


(b) TD3 - Without $\dot{x}$ (Obs Wrapper)



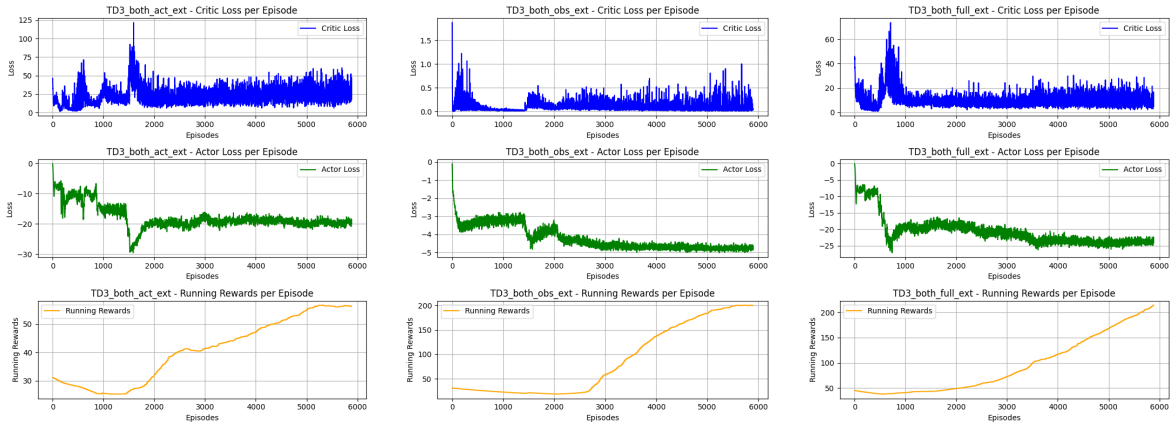
(c) TD3 - Without $\dot{x}$ (Both Wrappers)

Figure 8: TD3 - Performance Without $\dot{x}$



(a) TD3 - Without $\dot{\theta}$ (Action Wrapper) **(b)** TD3 - Without $\dot{\theta}$ (Obs Wrapper) **(c)** TD3 - Without $\dot{\theta}$ (Both Wrappers)

Figure 9: TD3 - Performance Without $\dot{\theta}$



(a) TD3 - Both (Action Wrapper) **(b)** TD3 - Both (Obs Wrapper) **(c)** TD3 - Both (Both Wrappers)

Figure 10: TD3 - Performance with Both Removed (Action, Obs and Both Wrappers)

L'observation des courbes montre que l'ajout du wrapper *ObsTimeExtensionWrapper* améliore significativement les performances, en particulier lorsque $\dot{\theta}$ est supprimé, et encore davantage lorsque les deux variables, $\dot{x}$ et $\dot{\theta}$, sont absentes. Cette amélioration est visible dans les deux algorithmes, DDPG et TD3, où les *running rewards* augmentent régulièrement malgré la perte d'informations critiques dans l'état. En effet, ce wrapper permet de mieux capturer les informations temporelles et de maintenir une continuité dans l'historique des observations, compensant ainsi la diminution de l'observabilité.

En revanche, le wrapper *ActionTimeExtensionWrapper* améliore moins efficacement les performances que le wrapper d'observation. Dans certains cas, comme pour DDPG avec la suppression de $\dot{\theta}$, il n'arrive pas à améliorer les résultats de manière significative. Toutefois, dans d'autres cas, comme l'indiquent les courbes pour TD3, il parvient à apporter une amélioration notable, même si celle-ci reste inférieure à celle du wrapper d'observation.

L'utilisation combinée des deux wrappers (*ActionTimeExtensionWrapper* et *ObsTimeExtensionWrapper*) s'avère être la meilleure stratégie. Elle permet une amélioration plus stable et plus importante des

running rewards, comme le montrent les graphiques des deux algorithmes. Cette combinaison exploite à la fois la répétition d'actions pour stabiliser les prises de décision et l'historique des observations pour compenser la perte d'informations dans les environnements partiellement observables.

En conclusion, bien que le *ObsTimeExtensionWrapper* soit particulièrement efficace dans les cas où $\dot{\theta}$ ou les deux variables ($\dot{x}$ et $\dot{\theta}$) sont supprimées, l'ajout du *ActionTimeExtensionWrapper* seul apporte des résultats mitigés. Cependant, leur combinaison constitue la meilleure solution pour améliorer les performances dans des environnements partiellement observables, comme l'indiquent clairement les résultats des courbes de récompenses cumulées dans les algorithmes DDPG et TD3. La gestion simultanée des actions répétées et de l'historique des observations est cruciale pour optimiser ces algorithmes dans de telles conditions.

7 Conclusion

L'étude réalisée démontre que l'utilisation des wrappers dans des environnements partiellement observables apporte des améliorations notables aux performances des algorithmes DDPG et TD3. En particulier, le *ObsTimeExtensionWrapper* se distingue par son efficacité à compenser la perte d'informations dans les cas où des variables cruciales telles que $\dot{\theta}$ ou à la fois $\dot{x}$ et $\dot{\theta}$ sont absentes. Ce wrapper permet de mieux capturer l'historique des observations, apportant une continuité temporelle essentielle pour stabiliser l'apprentissage. Dans ces environnements, où les informations sont partiellement occultées, les *rewards* cumulées montrent une croissance régulière, soulignant l'efficacité de ce wrapper.

D'un autre côté, le *ActionTimeExtensionWrapper* présente des résultats plus hétérogènes. Dans certains cas, il parvient à améliorer les performances, mais dans d'autres, comme avec DDPG lorsque $\dot{\theta}$ est supprimée, il montre ses limites. Cependant, l'utilisation conjointe des deux wrappers constitue la meilleure solution, offrant des améliorations stables et plus significatives. Les résultats obtenus montrent que cette combinaison exploite à la fois les actions répétées et l'historique des observations, optimisant ainsi l'apprentissage dans des environnements partiellement observables.