Mini-projet 3

GUERIN Cyril, POT Eline

1 DQN versus Double DQN sur le LunarLander

1.1 DQN

1.1.1 DQN avec BBRL_algos

En utilisant Optuna, on obtient les meilleurs paramètres suivants :

```
batch_size: 512
discount_factor: 0.9486594368703001
epsilon_end: 0.2
epsilon_start: 0.5
learning_starts: 10000
lr:  0.005
max_grad_norm: 4.2497873187262165
max_size: 1000000
optim_n_updates: 9
target_critic_update_interval: 10000
```

En le lançant avec un nombre de steps de 1500000, on a un best reward de 133 à la fin.
La meilleure récompense obtenue ne dépasse donc pas les 200, ce qui est un peu décevant. On peut supposer que cela est lié à l'étape d'exploration, et que l'agent n'a pas eu l'occasion d'apprendre mieux la meilleure politique.
Avec Wandb, on obtient alors les courbes suivantes :

Figure 1: Charts DQN

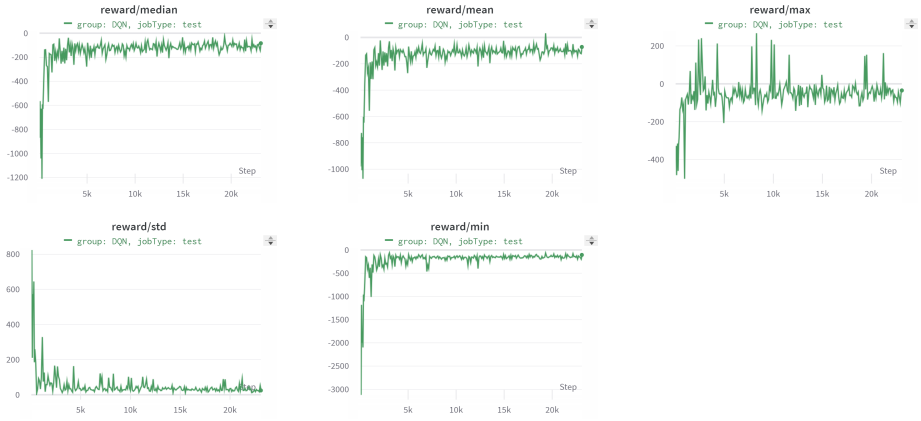

Figure 2: Rewards DQN

1.1.2 DQN en utilisant la classe DQN

En réutilisant le notebook 03-2-dqn-full.student.ipynb vu en TP, et en reprenant les meilleurs hyperparamètres trouvés précedemment avec la libraire bbrl_algos avec Optuna, on obtient une best reward de 241.41 après 768000 steps.
Ces résultats sont plus satisfaisants qu'avec ceux obtenus via la librairie bbrl_algos.

1.2 DDQN

1.2.1 DDQN de BBRL_algos

En utilisant Optuna, on obtient les meilleurs paramètres suivants :

```
discount_factor: 0.9751877145776201
learning_starts: 426
max_grad_norm: 2.0618889621540153
max_size: 50000
optim_n_updates: 2
```

On remarque qu'Optuna préconise un discount factor plus grand qu'avec DQN, et que les autres paramètres learning_starts, max_grad_norm, max_size et optim_n_updates necessitent des valeurs moins importantes.
En le lançant avec un nombre de steps de 1500000, on a un best reward de 6 à la fin.

Ces résultats sont donc peu concluants, car on aurait pu s'attendre à une meilleure récompense bien plus élevée après 1.5M de steps
Avec Wandb, on obtient alors les courbes suivantes :

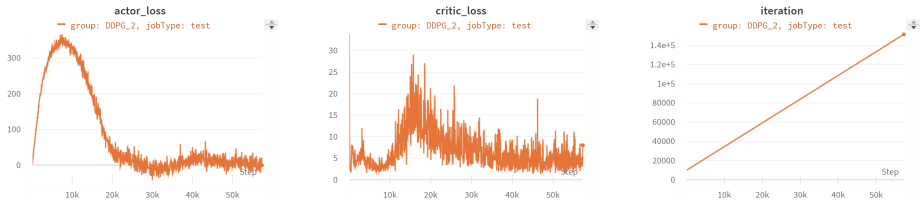

Figure 3: Charts DDQN

1.2.2 DDQN en utilisant la classe DDQN

En réutilisant le notebook 03-2-dqn-full.student.ipynb vu en TP, et en reprenant les meilleurs hyperparamètres trouvés précedemment avec la libraire bbrl_algos avec Optuna, on obtient un best reward de 52.82 après 512000 steps.
Ces résultats sont un peu plus satisfaisants que ceux obtenus précedement.

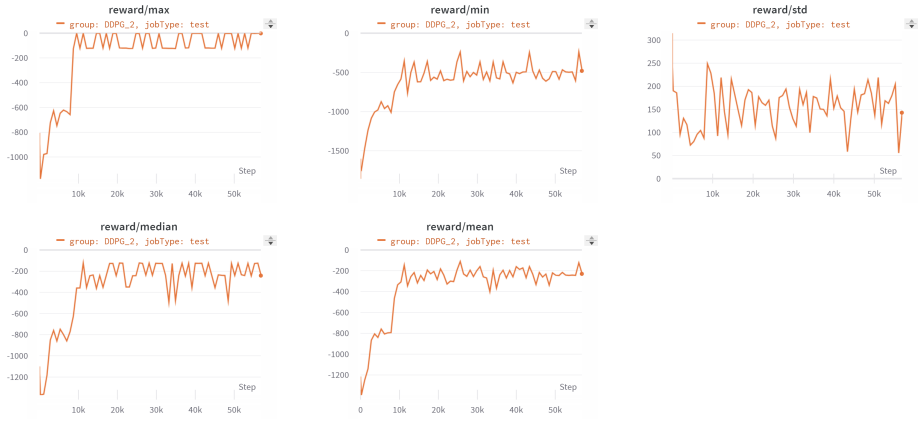

Figure 4: Rewards DDQN

1.3 Welch T Test

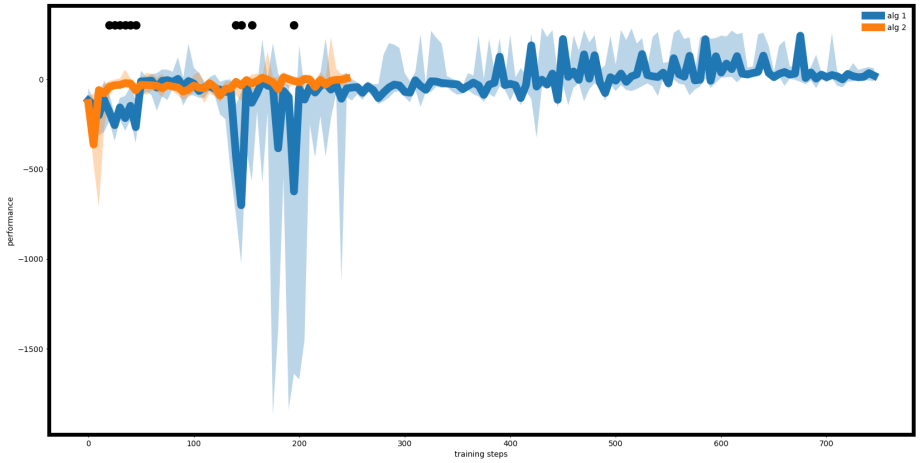

Figure 5: Welch T Test sur DQN et DDQN

Sur la figure 5, on constate que la courbe de DDQN converge plus rapidement vers de meilleures performances que DQN, et contient moins d'oscillations.

1.4 Algorithme DQN personnel

En n'utilisant pas la librairie bbrl_algos ou les codes fournis en TP, et en recodant entièrement l'algorithme DQN, on obtient ce genre de performance sur le Lunar Lander :

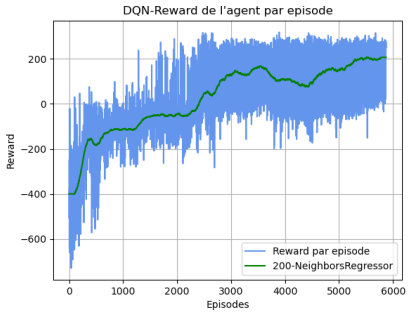

Figure 6: Performances de DQN

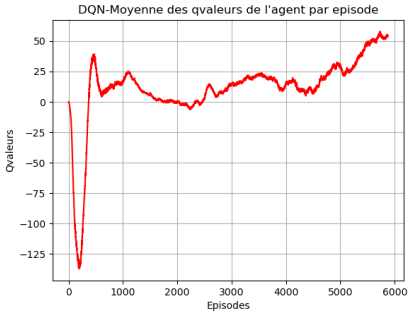

Figure 7: Qvalues de DQN

Sur la figure 6, on trace les rewards obtenus à chaque épisode de l'agent, ainsi que ses k-moyennes. Un épisode représente une simulation sur l'environnement et 10 entrainements sur la mémoire; à titre de comparaison sur une de nos machine personelle un épisode dure en moyenne 0.15 seconde. On fait arrêter l'agent lorsque les 100 derniers scores ont au moins une moyenne de 215.

Les paramètres utilisés ici sont :

```
buffer_size: 100_000
batch_size: 400
gamma: 0.99
learning_rate: 1e-4
epsilon_start: 1
epsilon_end: 0.01
epsilon_decay: 0.999
```

On constate que l'algorithme converge rapidement et que l'agent obtient des rewards qui varient beaucoup mais qui augmentent en moyenne, ce qui est plutot positif.

On trace également sur la figure 7 les Q valeurs en fonction des épisodes afin de voir si un phénomène d'overestimation est identifiable.

1.5 Algorithme DDQN personnel

En n'utilisant pas la librairie bbrl_algos ou les codes fournis en TP, et en recodant entièrement l'algorithme DDQN, on obtient ce genre de performance sur le Lunar Lander :

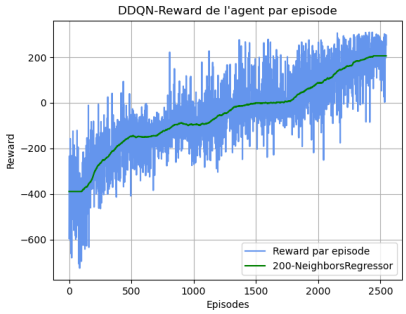

Figure 8: Performances de DDQN

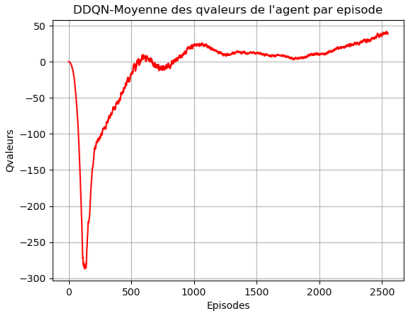

Figure 9: Qvalues de DDQN

De même, sur la figure 8, on trace les rewards obtenus à chaque épisode de l'agent, ainsi que ses k-moyennes.
Les épisodes ainsi que les paramètres utilisés sont les même qu'avec DQN.

On constate que l'algorithme DDQN converge beaucoup plus rapidement que DQN (environ 2300 épisodes contre environ 6000).

On trace également les Q valeurs en fonction des épisodes afin de voir si un phénomène d'overestimation est identifiable.
En comparant la figure 7 et la figure 9, on remarque bien que l'algorithme DQN souffre de sur-estimation.
On observe par ailleurs que l'algorithme DDQN réduit ce biais de sur-estimation, et conduit aussi à de bien meilleures performances sur l'environnement Lunar Lander.

2 DDPG versus TD3 on Pendulum-v1

2.1 DDPG

2.1.1 DDPG de BBRL_algos

En utilisant Optuna, on obtient les meilleurs paramètres suivants :

```
actor_hidden_size: (256, 256)
batch_size: 128
critic_hidden_size: (400, 300)
discount_factor: 0.99
optim_n_updates: 20
```

En le lançant avec un nombre de steps de 1000000, on a un best reward de -75 à la fin. Ces résultats sont satisfaisants.

Avec Wandb, on obtient alors les courbes suivantes :

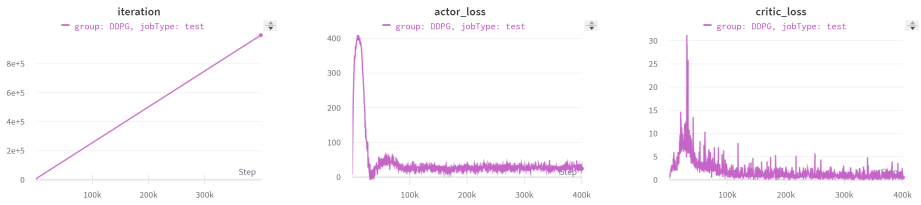

Figure 10: Charts DDPG

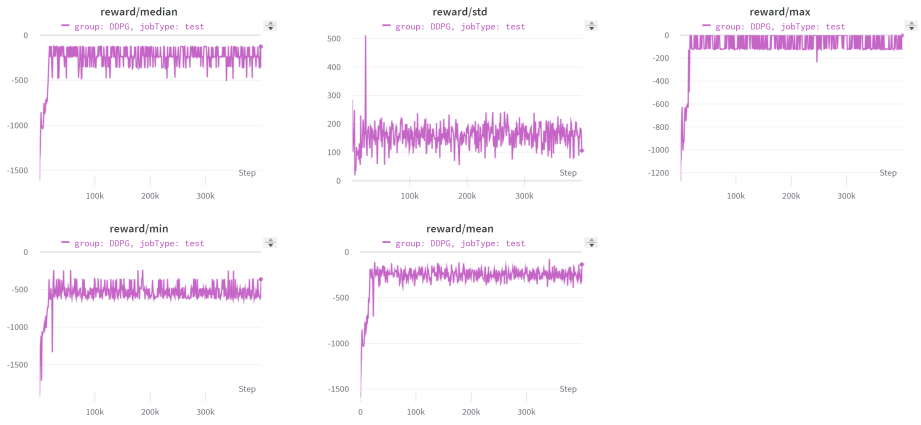

Figure 11: Rewards DDPG

2.1.2 DDPG en utilisant la classe DDPG

En réutilisant le notebook 04-ddpg-td3-new.student.ipynb vu en TP, et en reprenant les meilleurs hyperparamètres trouvés précedemment avec la libraire bbrl_algos avec Optuna, on obtient un best reward de -110.20 après 2000000 steps. Ces resultats sont donc un peu moins bons que ceux obtenus précedement

2.2 TD3

2.2.1 TD3 de BBRL algos

En utilisant Optuna, on obtient les meilleurs paramètres suivants :

```
actor_hidden_size: (128, 128)
batch_size: 64
critic_hidden_size: (64, 64)
discount_factor: 0.999
optim_n_updates: 20
```

On remarque déjà qu'il nous faut des réseaux de neurones pour l'actor et le critic contenant des tailles de couche cachée plus petites que pour DDPG.
Optuna préconise aussi une taille de batch de 64, plus petite que le 128 de DDPG.
Enfin, on obtient un discount factor de 0.999, plus proche de 1 qu'avec DDPG.
En le lançant avec un nombre de steps de 1000000, on a un best reward de -77 à la fin.
Les performances sont donc similaires à celles obtenues avec DDPG.
Avec Wandb, on obtient alors les courbes suivantes :

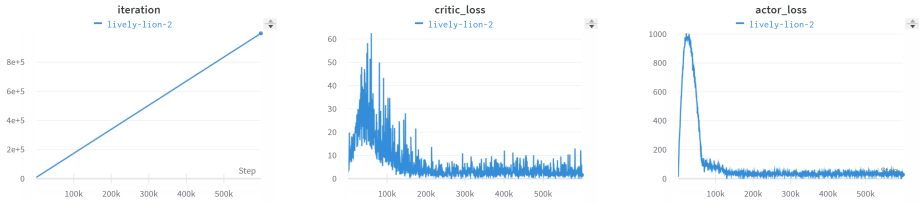

Figure 12: Charts TD3

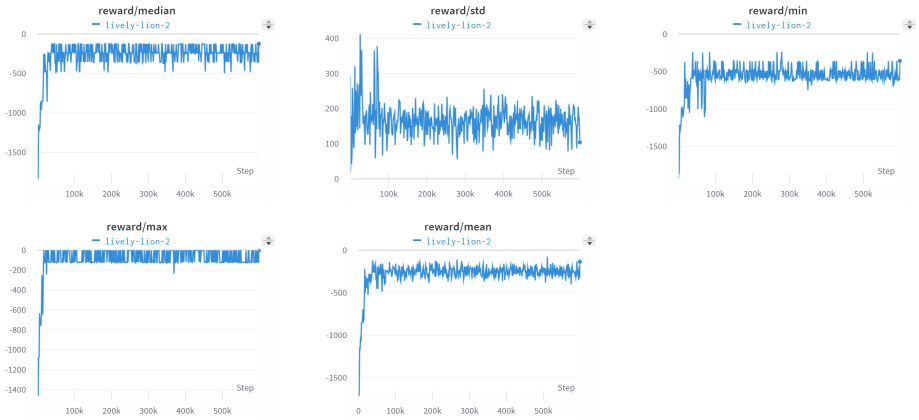

Figure 13: Rewards TD3

2.2.2 TD3 en utilisant la classe TD3

En réutilisant le notebook 04-ddpg-td3-new.student.ipynb vu en TP, et en reprenant les meilleurs hyperparamètres trouvés précedemment avec la libraire bbrl_algos avec Optuna, on obtient un best reward de -156.49 après 2000000 steps. Ces resultats sont moins bons que ceux obtnus précedement

2.3 Welch T Test

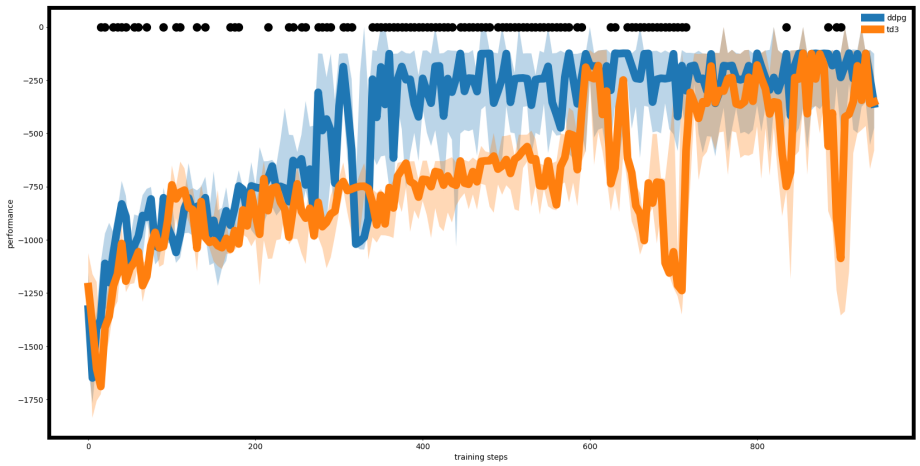

Figure 14: Caption

Sur la figure 14, on constate que la courbe de DDPG atteint de meilleures performances plus rapidement que TD3.