Rapport de Projet

Micro-Projet : LunarLander-Gym

Dans le cadre du
MASTER de Sorbonne Université

Spécialité
Informatique

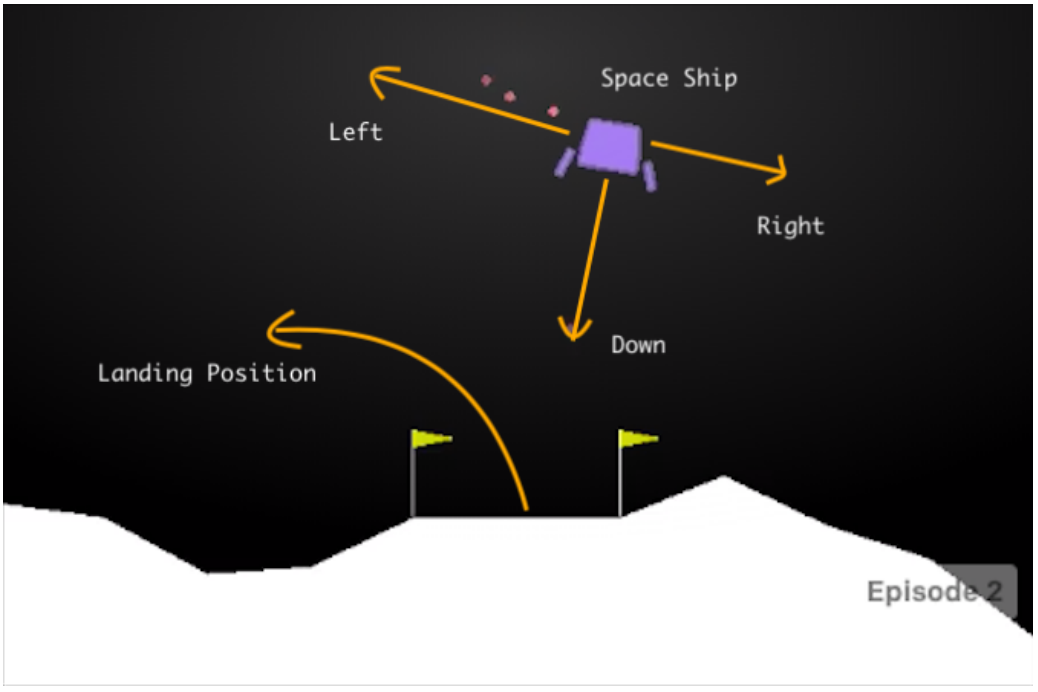
Parcours
ANDROIDE/DAC

ZUO Nicolas — Étudiant
GIANG Cécile — Étudiante
SIGAUD Olivier — Encadrant

Table des matières

1 Introduction

L'aire d'atterrissage est toujours aux coordonnées (0,0). Les coordonnées sont les deux premiers chiffres du vecteur d'état. La récompense pour le déplacement du haut de l'écran vers l'aire d'atterrissage et la vitesse zéro est d'environ 100 à 140 points. Si l'atterrisseur s'éloigne de l'aire d'atterrissage, il perd sa récompense. L'épisode se termine si l'atterrisseur s'écrase ou s'immobilise, recevant des points supplémentaires de -100 ou +100. Le contact au sol de chaque jambe vaut +10. L'allumage du moteur principal rapporte -0,3 points par image. Résolu est de 200 points. L'atterrissage en dehors de l'aire d'atterrissage est possible. Le carburant est infini, de sorte qu'un agent peut apprendre à voler puis à atterrir à sa première tentative. Quatre actions discrètes sont disponibles : ne rien faire, allumer le moteur d'orientation gauche, allumer le moteur principal, allumer le moteur d'orientation droit.



2 Démarche de l'obtention de notre Politique

Les politiques que nous avons décidé de tester dans un premier temps sans essayer d'optimiser les hyperparamètres sont DQN puis PPO et A2C. Afin de trouver les hyperparamètres optimaux pour chacune des trois politiques, nous ne sommes pas passés par la commande proposée par stable baseline zoo mais avons redéfini la fonction `sample_ppo_params` afin de les adapter à nos besoins.

Nous avons par la suite utilisé optuna pour optimiser les hyperparamètres de DQN, mais les résultats obtenus n'étant pas bons, nous avons fait de même pour PPO. Ce dernier nous donnant d'excellents résultats, nous avons par la suite utilisé ces hyperparamètres pour apprendre notre modèle PPO.

— Pour PPO, les hypermaramètres que nous avons choisi d'ajuster et les valeurs optimales obtenues sont `gamma` (0.9959465464371502), `n_steps` (1024) et `n_epochs` (7).

A noter que par la suite nous n'avons pas réutilisé optuna pour rechercher les hyperparamètres optimaux pour DQN et A2C, et nous sommes contentés de prendre les paramètres donnés dans le git de stable-baseline 3, qui propose des hyperparametres rendant des performances très correctes pour contruire nos différents modèles.

3 Courbe d'apprentissage de nos modèles

Légende :

— Bleu clair = DQN
— Bleu foncé = PPO
— Rouge = A2C

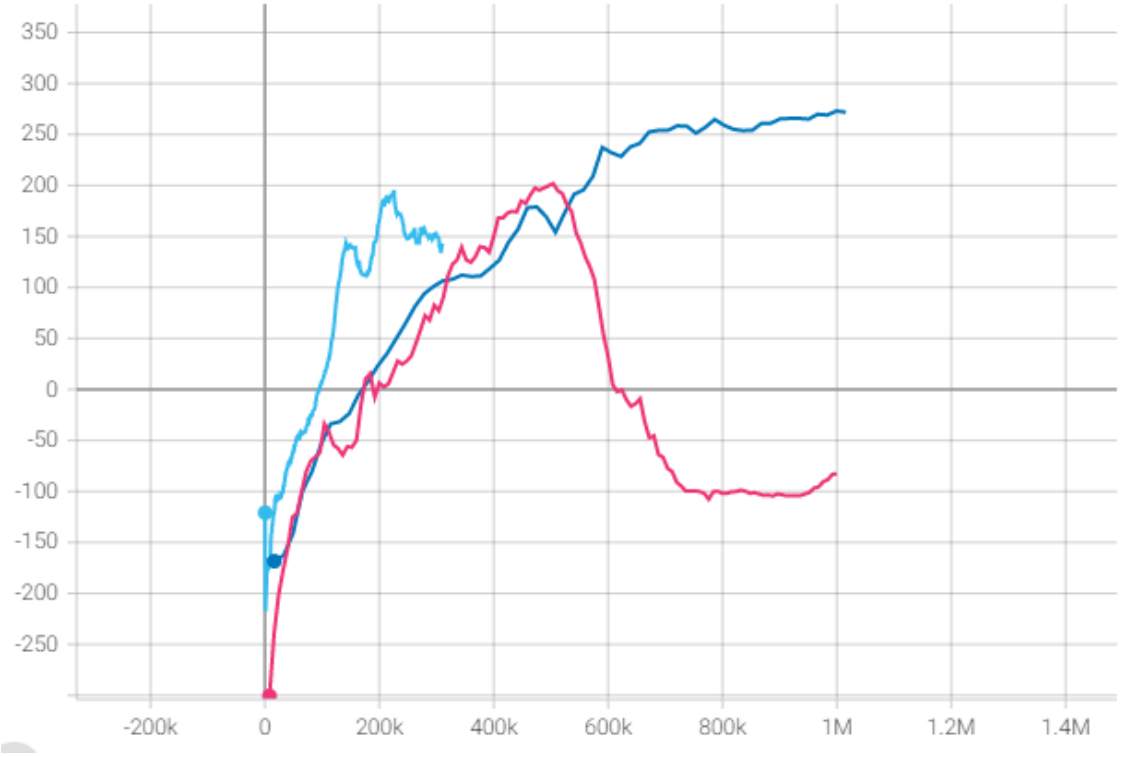


FIGURE 1 – Courbe d'évolution des scores moyens pendant l'apprentissage de nos modèles

Il faut noter ici que le nombre de steps de DQN est plus petit que PPO et A2C, car DQN sous stable baseline 3 ne prend pas en compte un vecteur d'environnement, ce qui fait que DQN a pris le double du temps nécessaire en apprentissage par rapport à PPO et A2C pour s'arrêter à environ 300 000 steps. Mais nous pouvons observer que comparé à A2C et PPO, DQN semble converger très rapidement, mais que arrivé à un moment il commence a diverger et même a perdre en performance. La même observation peut être faite sur A2C qui, arrivé à un certain niveau d'apprentissage, devient contre-productif. Il n'y a que PPO qui converge, lentement mais stablement.

On peut voir dans la figure 2 que le modèle avec A2C semble au cours de l'apprentissage ne plus atterir et reste stabilisé dans les airs (nombre d'épisodes croissants), ce qui explique la baisse de performance des scores moyens. Au contraire, nous remarquons que nombre d'épisodes nécessaire au modèle PPO pour atteindre son objectif décroît, tout en maintenant un score moyen élevé et stable : ce modèle a donc réussi à apprendre une politique lui permettant d'atteindre son objectif plus rapidement, sans compromettre le score.

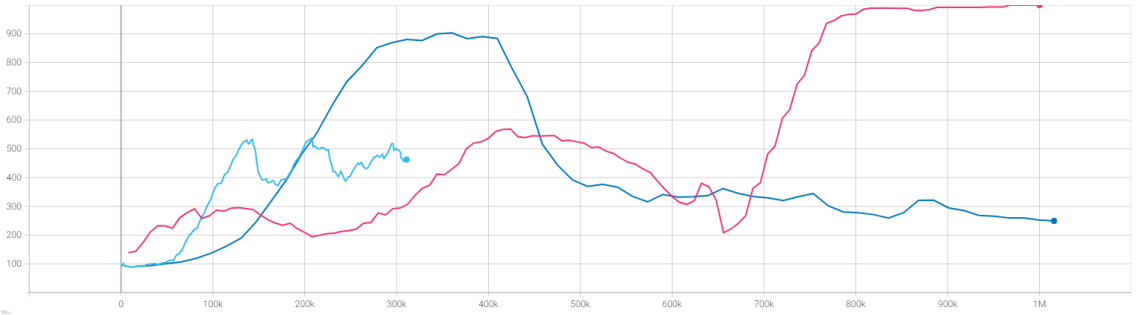


FIGURE 2 – Courbe d'évolution des nombres d'épisode pendant l'apprentissage de nos modèles

4 Choix du modèle et amélioration du modèle

Nous avons donc compris que PPO était parmi les trois modèles étudiés la meilleure option pour nous, car nous avons un budget en temps nécessaire pour bien apprendre notre modèle dans le temps. DQN serait un bon choix si nous avions peu de budget en temps. Nous allons donc profiter de la stabilité de PPO pour continuer à faire converger notre modèle vers une meilleure performance.

Légende :

— Bleu foncé = PPO (même que les précédentes figures)
— Gris = la suite du PPO(bleu foncé) avec un budget de 15 min d'apprentissage en plus
— Orange = la suite du PPO(Gris) avec un budget de 35 min d'apprentissage en plus

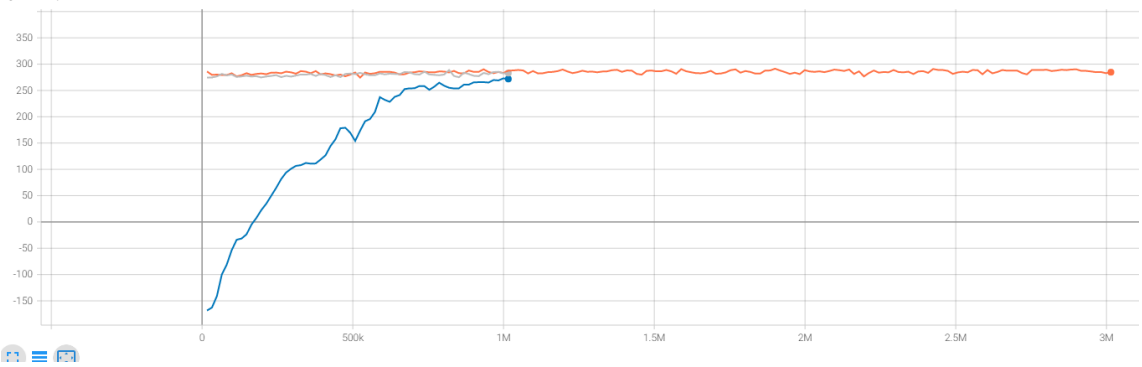


FIGURE 3 – Courbe d'évolution des scores moyens pendant l'apprentissage & l'amélioration du modèles PPO

On observe une amélioration très minime de notre politique du PPO (bleu, avec 15 min d'apprentissage) sur la courbe grise, nous avons donc relancé l'apprentissage sur une durée un peu plus longue, ce qui nous donne la courbe orange.

Mais nous pouvons voir clairement, grâce à la figure 4, que le nombre d'épisodes semble continuer à diminuer de manière assez constante dans le temps, et que notre score n'a pas baissé durant l'apprentissage en contrepartie. On peut donc en conclure que PPO continue tout de même à gagner en performance. Nous allons voir dans la partie suivante les scores obtenus, mais le gain entre la courbe bleu (15 minutes de budget) et la courbe orange (1 heure de budget) est de environ 20 points : on passe

d'un score de 270 à 290 sur l'évaluateur mis à disposition (testé sur plusieurs seeds aléatoires).

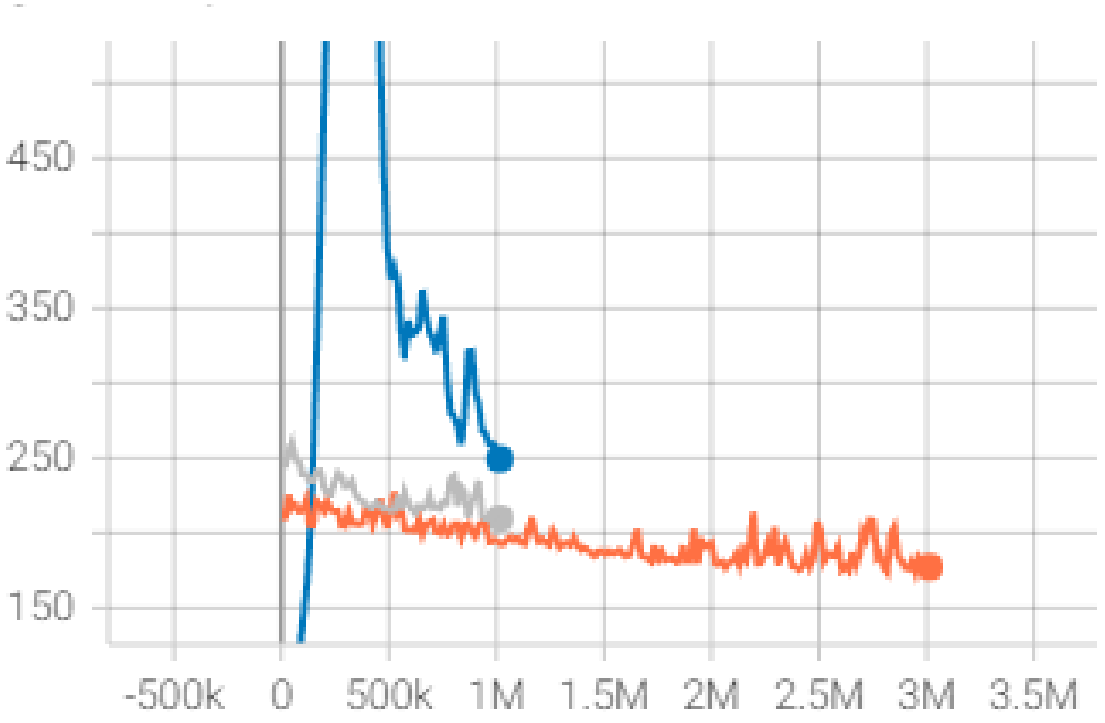


FIGURE 4 – Courbe d'évolution du nombre d'épisodes pendant l'apprentissage & l'amélioration du modèles PPO

5 Scores obtenus

Meilleurs scores obtenus :

— DQN = 195 (30 minutes de budget)
— A2C = 201 (15 minutes de budget)
— PPO = 271 (15 minutes de budget)
— PPO = 288 (30 minutes de budget)
— PPO = 290.5 (60 minutes de budget)

On peut noter que dans la figure 5, on peut apercevoir que quand nous avons un score inférieur à 289, l'hypothèse serait que la force exercée au départ sur l'appareil ne soit pas toujours la même, ce qui nécessite l'utilisation des propulseurs droit et gauche pour se replacer au centre de l'environnement pour atterrir entre les drapeaux.

```
C:\Users\Nico\Desktop\IAR\rl-baselines3-zoo>python sb3_evaluator.py --seed 70
LunarLander-v2#ppo#ZuoGiang.zip
Hall of fame
Environment : LunarLander-v2
team:  ZuoGiang mean:  289.11235998 std:  19.595214740845755
Time : 6s 663ms

C:\Users\Nico\Desktop\IAR\rl-baselines3-zoo>python sb3_evaluator.py --seed 5
LunarLander-v2#ppo#ZuoGiang.zip
Hall of fame
Environment : LunarLander-v2
team:  ZuoGiang mean:  290.88440201000003 std:  19.387920902623613
Time : 6s 928ms

C:\Users\Nico\Desktop\IAR\rl-baselines3-zoo>python sb3_evaluator.py --seed 1111
LunarLander-v2#ppo#ZuoGiang.zip
Hall of fame
Environment : LunarLander-v2
team:  ZuoGiang mean:  286.38817605 std:  33.308934981754796
Time : 6s 553ms

C:\Users\Nico\Desktop\IAR\rl-baselines3-zoo>python sb3_evaluator.py --seed 2811
LunarLander-v2#ppo#ZuoGiang.zip
Hall of fame
Environment : LunarLander-v2
team:  ZuoGiang mean:  291.07106108 std:  19.217798206154637
Time : 6s 495ms

C:\Users\Nico\Desktop\IAR\rl-baselines3-zoo>python sb3_evaluator.py --seed 159
LunarLander-v2#ppo#ZuoGiang.zip
Hall of fame
Environment : LunarLander-v2
team:  ZuoGiang mean:  284.23432378999996 std:  32.56849608215912
Time : 6s 620ms

C:\Users\Nico\Desktop\IAR\rl-baselines3-zoo>python sb3_evaluator.py --seed 19
LunarLander-v2#ppo#ZuoGiang.zip
Hall of fame
Environment : LunarLander-v2
team:  ZuoGiang mean:  290.02138611999993 std:  18.908511378628788
Time : 6s 480ms
```

FIGURE 5 – Evaluation de notre politique (PPO, avec 60 minutes de budget) sur des seeds aléatoirement choisis