Justin Klein
1707815

# AS 3.1 Rapportage - Lunar Lander

## Opdracht

Voor AS3.1 moest er een Deep Q-Learning Neural Network geïmplementeerd worden in de Lunar Lander omgeving van de Gymnasium Library. Hierbij moest een algoritme (Deep Q-Learning with Experience Replay) geïmplementeerd worden om de lunar lander succesvol te laten leren en convergeren tot een optimaal beleid (policy).

## Context uitwerking

In deze opdracht heb ik gekozen om een jupyter notebook op te stellen met daarin de main code. Deze code start de omgeving(en), maakt objecten aan en draait de methodes van de gemaakte objecten, inclusief de training. Ik heb de gymnasium omgeving compleet buiten de Agent class gehouden om zo model-free mogelijk te werk te gaan. In verband met tijdgebrek heb ik helaas niet de extra opdracht kunnen implementeren.

## Obstakels

Bij deze opdracht was mijn eerste implementatie volgens de theorie correct, maar mijn gebrek aan kennis van PyTorch maakte het voor mij extreem lastig om de agent correct te trainen. Dit is dan ook de reden dat ik mijn code compleet heb moeten ombouwen om het leerproces in PyTorch correct te implementeren. Ik heb toen ook een unnamed tuple toegevoegd als nieuwe datastructuur voor mijn transitions (ipv reguliere tuples) en ben ik alleen nog maar tensors gaan gebruiken zodat er nog sneller en beter getraind kon worden. Ook heb ik een Deep Q-Learning PyTorch tutorial gevolgd wat veel heeft geholpen in het correct opzetten van mijn model en het leerproces. Helaas kon ik CUDA niet aan de praat krijgen met jupyter-notebook en heb ik dus op mijn CPU het trainingsproces gedraaid.

Justin Klein
1707815

# Resultaten

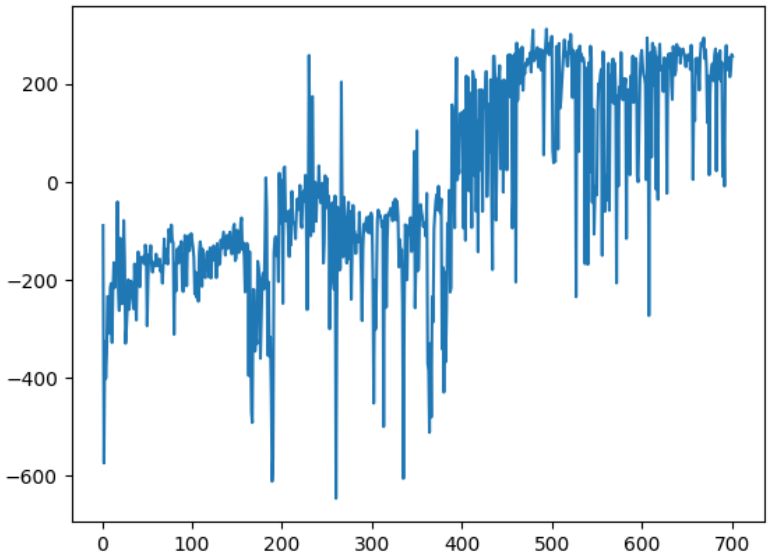
De epsilon in de iteraties wordt gedegradeerd met een decay-functie.
*Zie het notebook en policy.py binnen mijn implementatie voor meer informatie.*

Mijn implementatie heb ik meerdere keren getraind met de volgende resultaten:

| **Iteration** | **Converges on:** | **Average running reward:** |
|---|---|---|
| 1 | ~600 epochs | +207 |
| 2 | ~500 epochs | +209 |
| 3 | ~700 epochs | +203 |
| 4 | ~500 epochs | +214 |

De hoeveelheid epochs verschilt met iedere iteratie. Ik vermoed dat dit komt doordat de omgeving veel ‘randomness’ bevat wat veel effect heeft op de snelheid van de convergentie. Als de Lander aan het begin van de simulatie het geluk heeft dat hij goed landt dan zal de convergentie iets sneller gaan, als hij dit niet doet dan zal de convergentie langer duren. Uiteindelijk zullen wel, met mijn huidige implementatie, alle trainings iteraties convergeren.

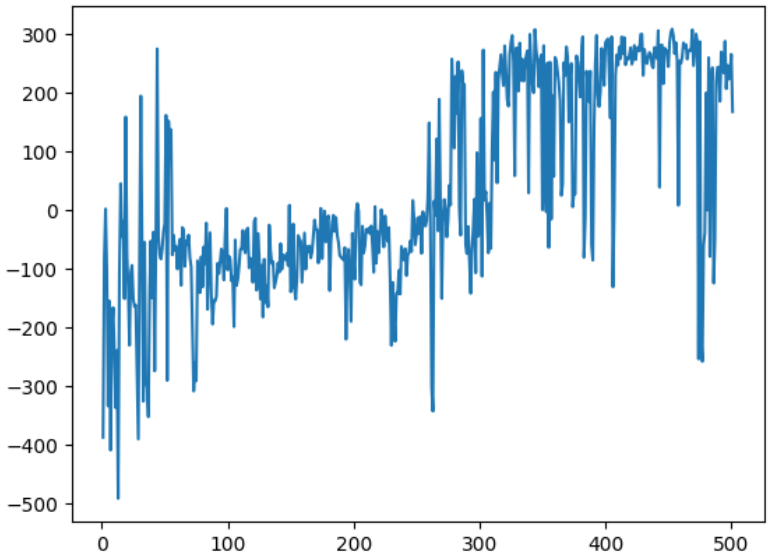
De onderste grafiek is van een van mijn laatste iteraties (~700 epochs) waarbij we de totale som van de rewards van alle stappen binnen iedere epoch plotten.



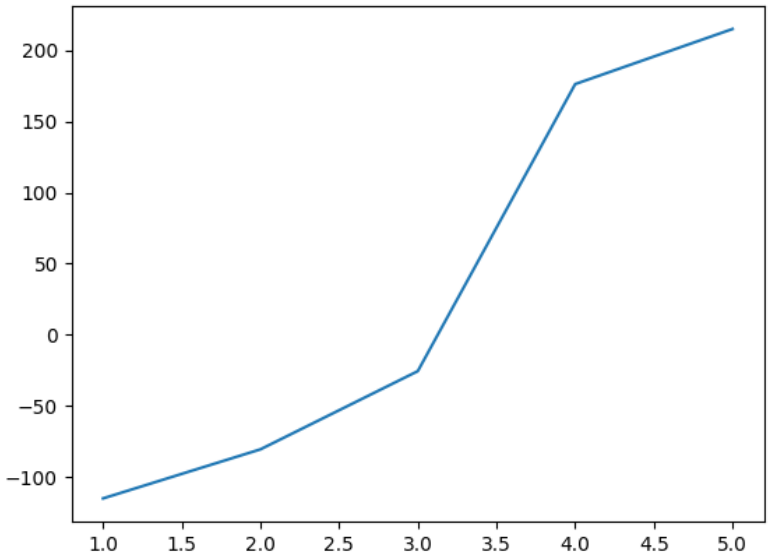

Afbeelding 1.1: X = number of epochs, Y = Sum step rewards per epoch

Bij dit voorbeeld begint het neurale netwerk na circa 200-250 epochs steeds meer positieve rewards te zien, dit is dus vermoedelijk afhankelijk van de beginstaat en de willekeurige acties aan het begin van de simulatie. Na circa 380-390 epochs convergeert het neurale netwerk met nog een aantal uitschieters maar is de running average reward meer dan +200 tussen 600 en 700 epochs. De Lunar Lander land dan ook heel netjes binnen de vlaggetjes of erbuiten, wat in ieder geval het meeste oplevert voor de Lunar Lander binnen de omgeving van Gymnasium.

In dit voorbeeld zien we een eerdere convergentie van 500 epochs:




Afbeelding 1.2: X = number of epochs, Y = Sum step rewards per epoch




Afbeelding 1.3: X = x * 100 is epochs (so 100, 150, 200, etc.), Y = Running average reward

Bij deze iteratie convergeert het neurale netwerk weer veel sneller (~200 epochs sneller) en zien we ook wat mijn vermoedens mogelijk bevestigen. Aan het begin van de iteratie zien we dat de agent veel hoge rewards krijgt wat een sterke invloed heeft op het leerproces van de agent (zie afbeelding 1.2). We zien ook in afbeelding 1.3 dat na 300 epochs er een sterke stijgende lijn voorkomt in de gemiddelde ‘running rewards’.

# Conclusie

Aan de hand van de eerder genoemde resultaten is een vermoeden ontstaan dat de optimale policy van de agent sneller wordt benaderd als de eerste epochs in de iteratie gunstig uitpakken voor de agent. Echter kunnen we aan de hand van deze resultaten dit nog niet 100% bevestigen. Dit zouden we kunnen onderzoeken door nog veel meer iteraties te draaien met dezelfde parameters (behalve de willekeurigheid van de omgeving) om zo de geldigheid van dit vermoeden verder uit te zoeken. Ook kan er nog meer met de hyperparameters gespeeld worden om zo het effect op het leerproces van de agent in kaart te brengen, echter heb ik daar helaas niet de tijd voor gehad en zou dit dus ook als een vervolgstap zien binnen deze opdracht. Kortom, de implementatie werkt correct, de optimale policy wordt bepaald en er zijn veel interessante uitkomsten naar voren gekomen!