

Testrapport

REINFORCEMENT LEARNING DEMONSTRATOR - PACMAN

Finn Alberts, Laurent Dassen en Noud Wijngaards
ZUYD HOGESCHOOL | HBO ICT



Inhoud

| | |
|------------------------|----|
| 1 Inleiding..... | 3 |
| 2 Testplan..... | 3 |
| 3 Testresultaten | 3 |
| Test 1..... | 3 |
| Test 2..... | 6 |
| Test 3..... | 8 |
| Test 4..... | 10 |
| Test 5..... | 11 |
| Test 6..... | 13 |
| Test 7..... | 15 |
| Test 8..... | 17 |
| 4 Verwijzingen..... | 17 |

1 Inleiding

Om te kunnen zien welke rewardfunctie het beste werkt met Pacman zijn een aantal testen gedaan met verschillende rewardfuncties.

2 Testplan

Met eigen observaties en de trial-and-error methode is gezocht naar een goede rewardfunctie of naar een rewardfunctie waar op gebouwd kon worden. Er is begonnen met de trial-and-error methode om zo tot een eerste testgeval te kunnen komen. Daarna is via eigen observaties verandering gebracht aan de rewardfunctie, omdat bijvoorbeeld bepaalde aspecten niet optimaal werkte of omdat bepaalde aspecten nog niet verwerkt waren. Bijvoorbeeld wanneer blijkt dat Pacman vaak te dicht bij ghosts komt, een extra reward te geven voor uit te buurt te blijven van ghosts of door de doodgaan penalty aan te passen om zo optimaler te maken. Hierop is dan verder geborduurd. Er is (tenzij anders vermeldt bij de specifieke test) gebruik gemaakt van het PPO-algoritme.

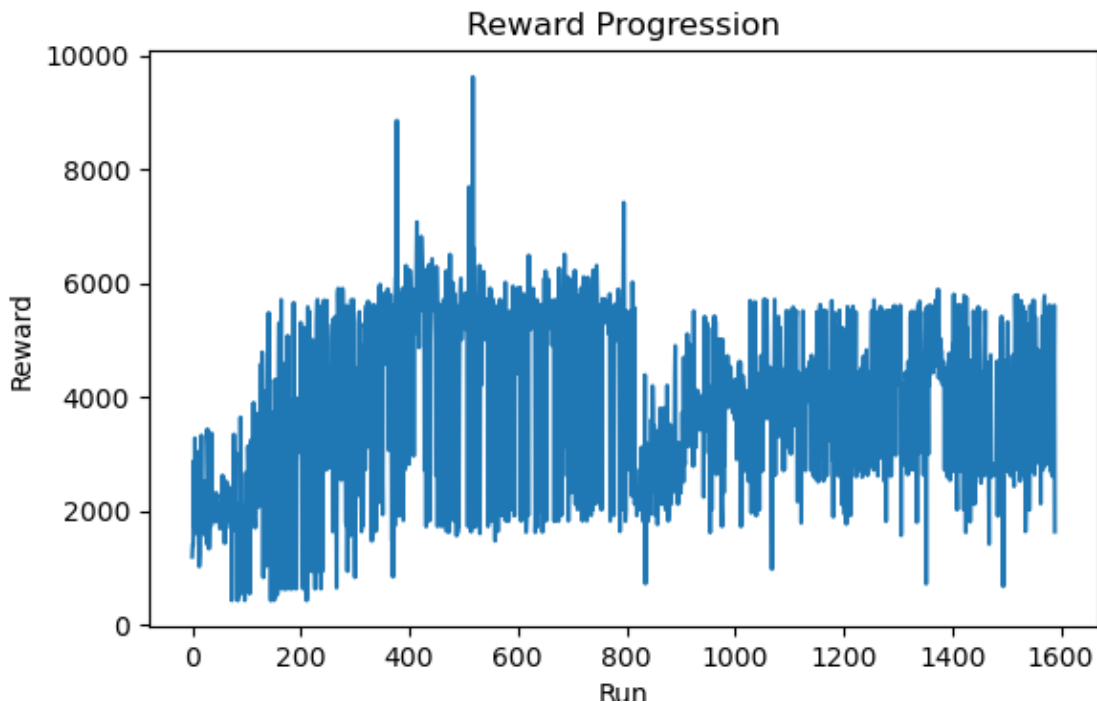
Van iedere test is vastgelegd welke rewardfunctie gebruikt is en is een grafiek gemaakt van de behaalde rewards. Vanaf het moment dat de grafiek minimaal vierhonderd runs stagneert is de test verbroken. Dit is het punt waarop vastgesteld is dat het geen goede rewardfunctie is voor Pacman om goed te kunnen leren met reinforcement learning. Soms is er gekozen om de test langer te laten draaien, omdat de grafiek niet goed aanduidt of de stagnering blijvend is of omdat een langere run (vanwege bijvoorbeeld externe factoren) gewenst was.

Hieronder zijn enkele testresultaten vastgelegd. In totaal zijn er meerdere testgevallen geweest, maar omdat veel testgevallen op elkaar leken zijn hier alleen de meest opvallende en belangrijke testgevallen gedocumenteerd.

3 Testresultaten

Test 1

Dit is een van de eerste testen. In deze test wordt specifiek gekeken naar elke actie die Pacman doet en kan doen om daar dan een passende reward of penalty voor te geven. Hieronder valt pellets eten, fruit eten, ghosts eten, het level halen, tijd en doodgaan. Deze test laat zien dat er te veel factoren medoen aan de rewardfunctie waardoor de grafiek heel inconsistent gaat uitzien.



```
def _get_reward(self, gamestate: dict, action: int):
    reward = 0

    # For each pellet eaten a reward is given
    if gamestate["pelletsEaten"] > self.pelletsEaten:
        reward += 100
    self.pelletsEaten = gamestate["pelletsEaten"]

    if gamestate["fruitsEaten"] > self.fruitsEaten:
        reward += 500

    if gamestate["ghostsEaten"] > self.ghostsEaten:
        reward += 500
    self.ghostsEaten = gamestate["ghostsEaten"]
    # Game score updated to keep track of score progression
    self.score = gamestate["score"]

    # Reaching Level 2 gives a (big) reward
    if gamestate["level"] > 0:
        reward += 10000

    # Passing of time gives a penalty (quicker runs are better)
    reward -= 0.3

    # Dying gives a penalty. The bigger the score, the lower the penalty
    if not gamestate["is_alive"]:
        reward -= 5000 ** (1 - (gamestate["score"] % 14800))

    # Add reward to total reward
```

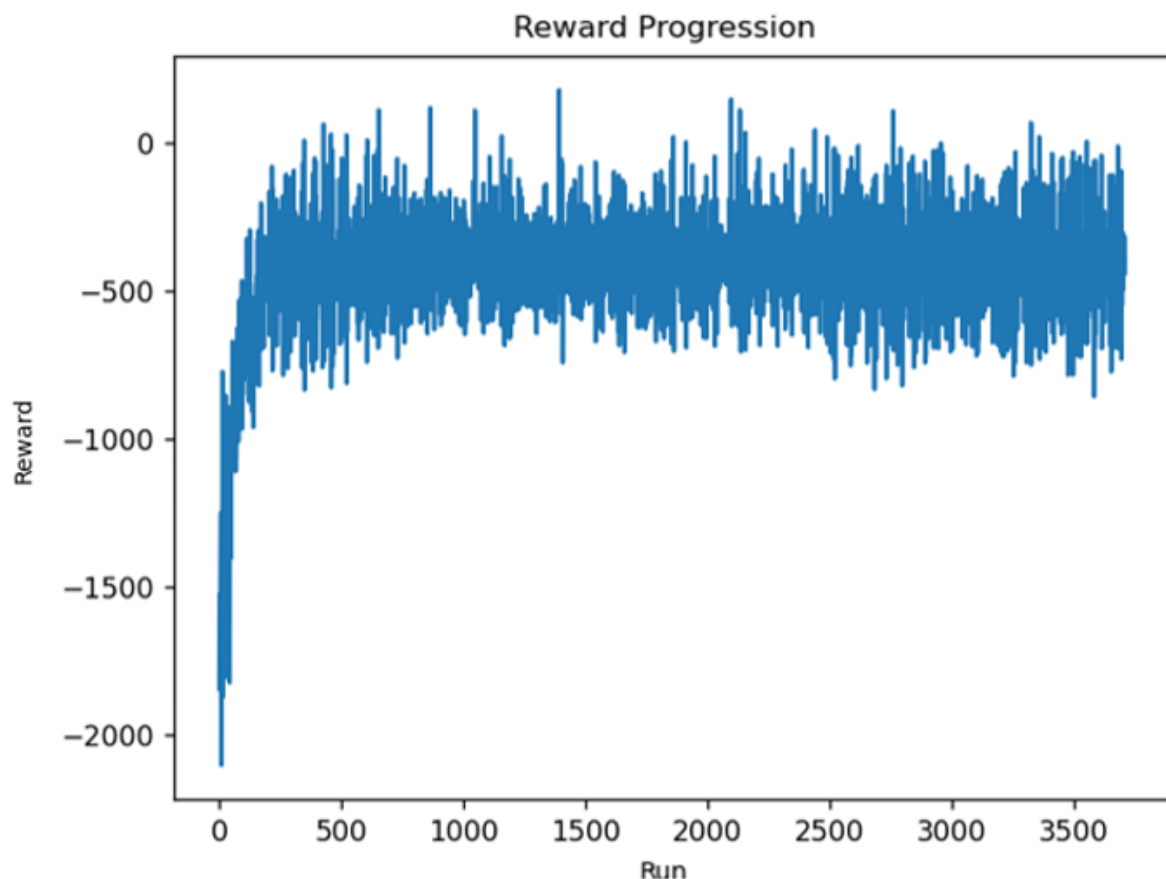


```
self.total_reward["total"] += reward  
  
return reward
```


Test 2

Dit is de beste test die is gedaan. Bij deze test is rekening gehouden met een aantal factoren. Een reward geven aan de hand van meer score behalen, een reward voor het level halen, een penalty voor de tijd, een penalty voor acties uit te voeren en een penalty voor doodgaan. Bij de doodgaan penalty is een ietswat andere functie gebruikt. Aan de hand van de score die is behaald wordt de penalty aangepast, hoe groter de score, hoe kleiner de penalty voor doodgaan.

De grafiek heeft een goed begin, maar stagneert al vrij snel. De vorm van deze grafiek is wel hoe een reinforcement learning grafiek uit hoort te zien, alleen begin het stagneerpunt veel te vroeg waardoor Pacman bijna niks leert.



```
def _get_reward(self, gamestate: dict, action: int):
    reward = 0

    # Increasing score gives a reward
    reward += (gamestate["score"] - self.score)
    self.score = gamestate["score"]

    # Reaching Level 2 gives a (big) reward
    if gamestate["level"] > 0:
        reward += 10000

    # Passing of time gives a penalty (quicker runs are better)
    reward -= 0.5
```



```
# Pressing buttons is not free
if action != 0:
    reward -= 5

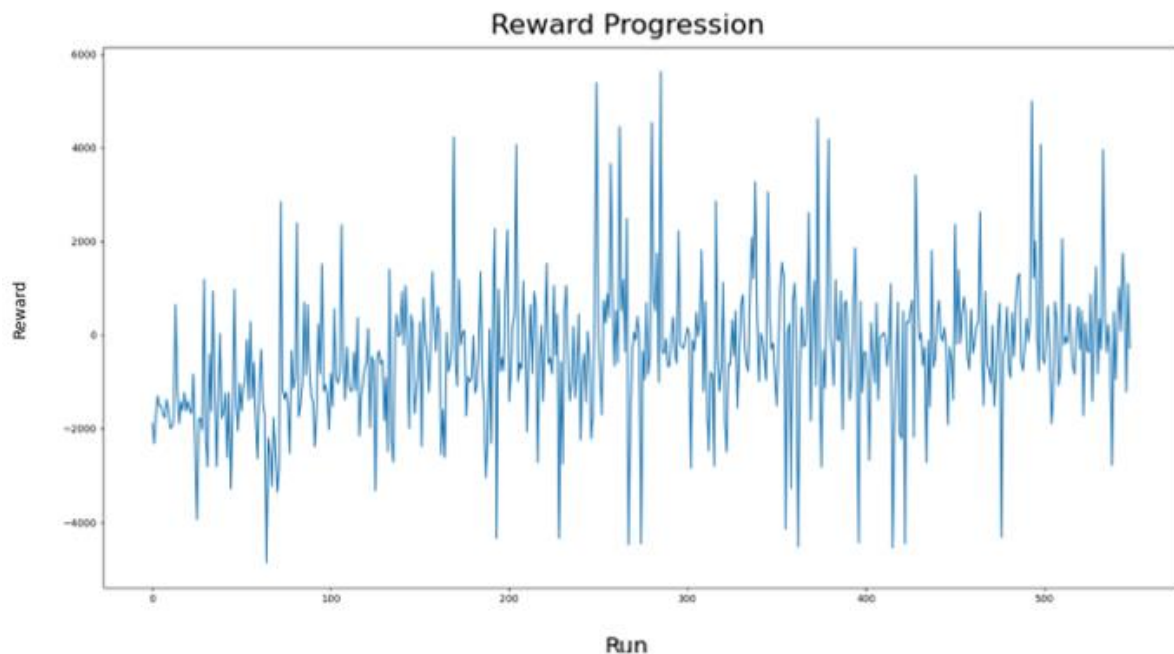
# Dying gives a penalty. The bigger the score, the lower the penalty
if not gamestate["is_alive"]:
    reward -= (500 - (gamestate["score"] * 0.0338))

# Add reward to total reward
self.total_reward["total"] += reward

return reward
```


Test 3

Dit is de eerste test dat er rekening wordt gehouden met de afstand van Pacman tot de ghosts. De afstand tot de dichtsbijzijnde ghost wordt berekend en aan de hand van de afstand wordt een reward of penalty gegeven. Zoals de afbeelding hieronder al aangeeft, zorgt dit voor een heel erg inconsistente grafiek.



```
def _get_reward(self, gamestate: dict, action: int):
    reward = 0

    # Increasing score gives a reward
    reward += (gamestate["score"] - self.score)
    self.total_reward["score"] += (gamestate["score"] - self.score)
    self.score = gamestate["score"]

    # Reaching Level 2 gives a (big) reward
    if gamestate["level"] > 0:
        reward += 10000
        self.total_reward["level_complete"] += 10000

    # Passing of time gives a penalty (quicker runs are better)
    reward -= 0.5
    self.total_reward["time_alive"] -= 0.5

    # Pressing buttons is not free
    if action != 0:
        reward -= 5
        self.total_reward["button_presses"] -= 5

    # Dying gives a penalty
    if gamestate["is_alive"] == False:
```



```
        reward -= 5000
        self.total_reward["dying"] -= 5000

    # Distance to ghost gives a penalty/reward
    reward += gamestate["min_ghost_distance"]
    self.total_reward["ghost_distance"] += gamestate["min_ghost_distance"]

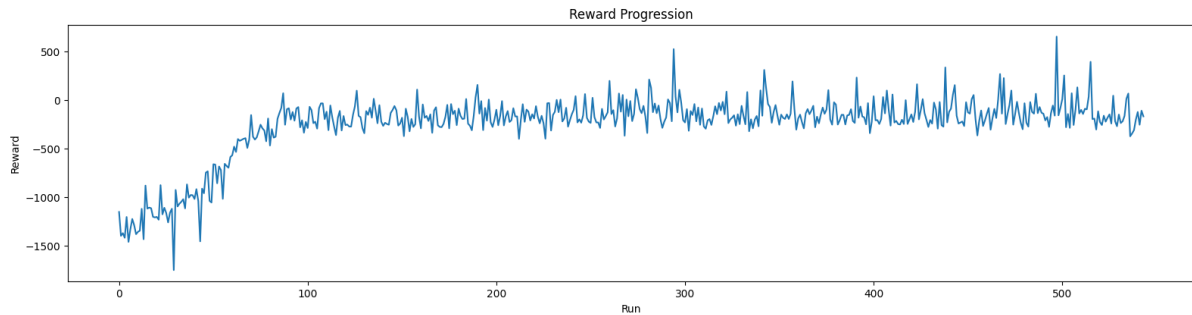
    # Add reward to total reward
    self.total_reward["total"] += reward

    return reward
```


Test 4

Deze test is een run waarbij getest is met maar één ghost in plaats van vier. Hier is voor gekozen om te kijken of Pacman betere scores zou halen dan bij vier ghosts. Ook is hier besloten om Pacman een reward te geven voor de tijd in plaats van een penalty, aangezien het doel was om Pacman zolang mogelijk levend te houden om het level te kunnen halen.

Zoals de grafiek hieronder al aangeeft, heeft dit niet geholpen tot een verbeterde grafiek. De grafiek geeft een zelfde curve aan als bij test 2, alleen met een nog minder stijgende learning curve op het begin.



```
def _get_reward(self, gamestate: dict, action: int):
    reward = 0

    # Increasing score gives a reward
    reward += (gamestate["score"] - self.score)
    self.total_reward["score"] += (gamestate["score"] - self.score)
    self.score = gamestate["score"]

    # Reaching Level 2 gives a (big) reward
    if gamestate["level"] > 0:
        reward += 10000
        self.total_reward["level_complete"] += 10000

    # Passing of time gives a reward (surive Longer)
    reward += 0.5
    self.total_reward["time_alive"] += 0.5

    # Pressing buttons is not free
    if action != 0:
        reward -= 5
        self.total_reward["button_presses"] -= 5

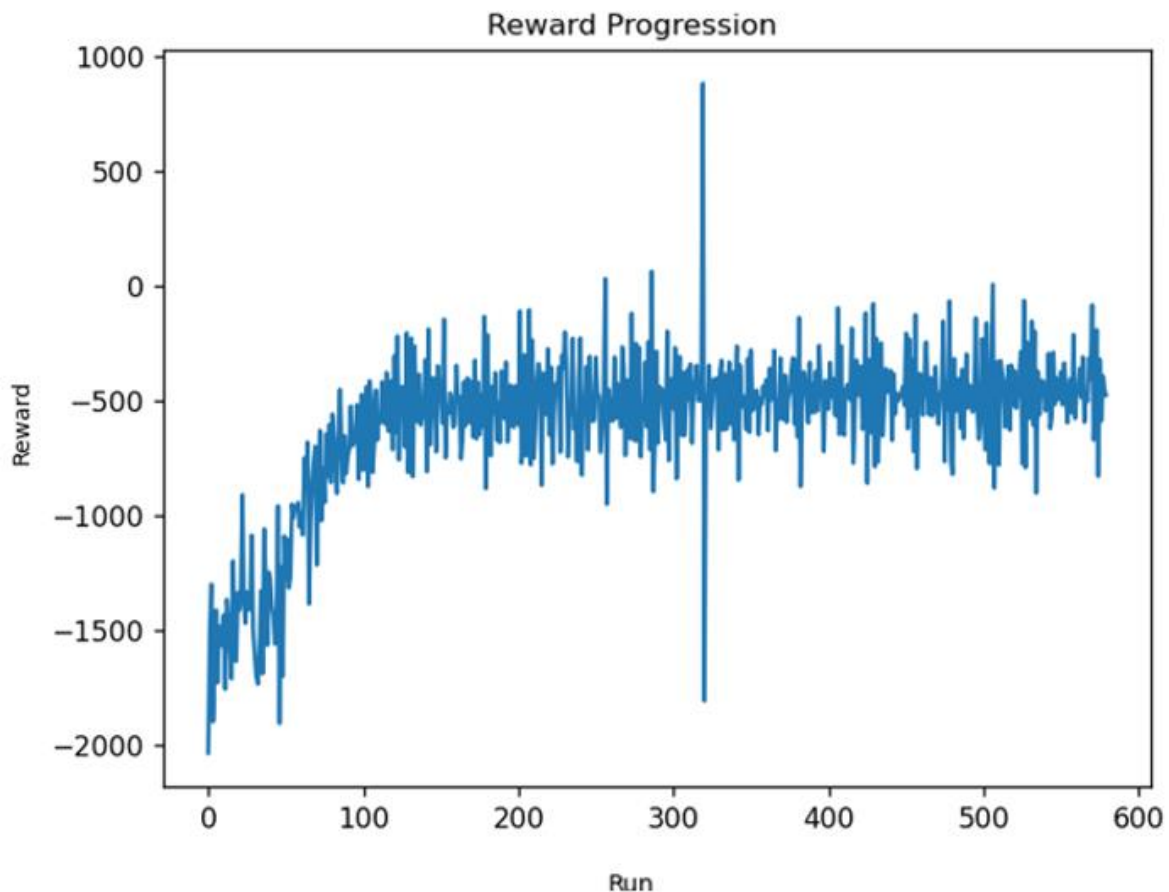
    # Dying gives a penalty
    if gamestate["is_alive"] == False:
        reward -= (500 - gamestate["score"] * 0.0338)
        self.total_reward["dying"] -= (500 - gamestate["score"] * 0.0338)

    # Add reward to total reward
    self.total_reward["total"] += reward
    return reward
```


Test 5

In deze test wordt iedere keer dat de rewardfunctie aangeroepen wordt, en Pacman leeft nog, de reward vermenigvuldigd met 1,1. Dit om ervoor te zorgen dat meer punten behalen en levend blijven ook als een reward gelden. (Naderhand bleek dat deze vermenigvuldiging geen nut had, omdat dit alleen ervoor zorgde dat de reward 110% werd in plaats van de originele 100% en niet hoger werd naarmate Pacman langer bleef leven.) De grafiek hieronder laat zien dat dit een zelfde soort vorm laat zien als test 2, maar absoluut niet beter is.

Opvallend is wel het hoogte en dieptepunt in het midden van de grafiek. Of dit toeval is, valt niet te zeggen. Wel is het een opmerkelijk fenomeen dat zich heeft voorgedaan.



```
def _get_reward(self, gamestate: dict, action: int):
    reward = 0

    # Higher score yields a higher reward
    if gamestate["score"] > self.score:
        gamestate["score"] *= 1.1

    # Increasing score gives a reward
    reward += (gamestate["score"] - self.score)
    self.score = gamestate["score"]

    # Reaching Level 2 gives a (big) reward
    if gamestate["level"] > 0:
```



```
        reward += 10000

# Passing of time gives a penalty (quicker runs are better)
    reward -= 0.5

# Pressing buttons is not free
    if action != 0:
        reward -= 5

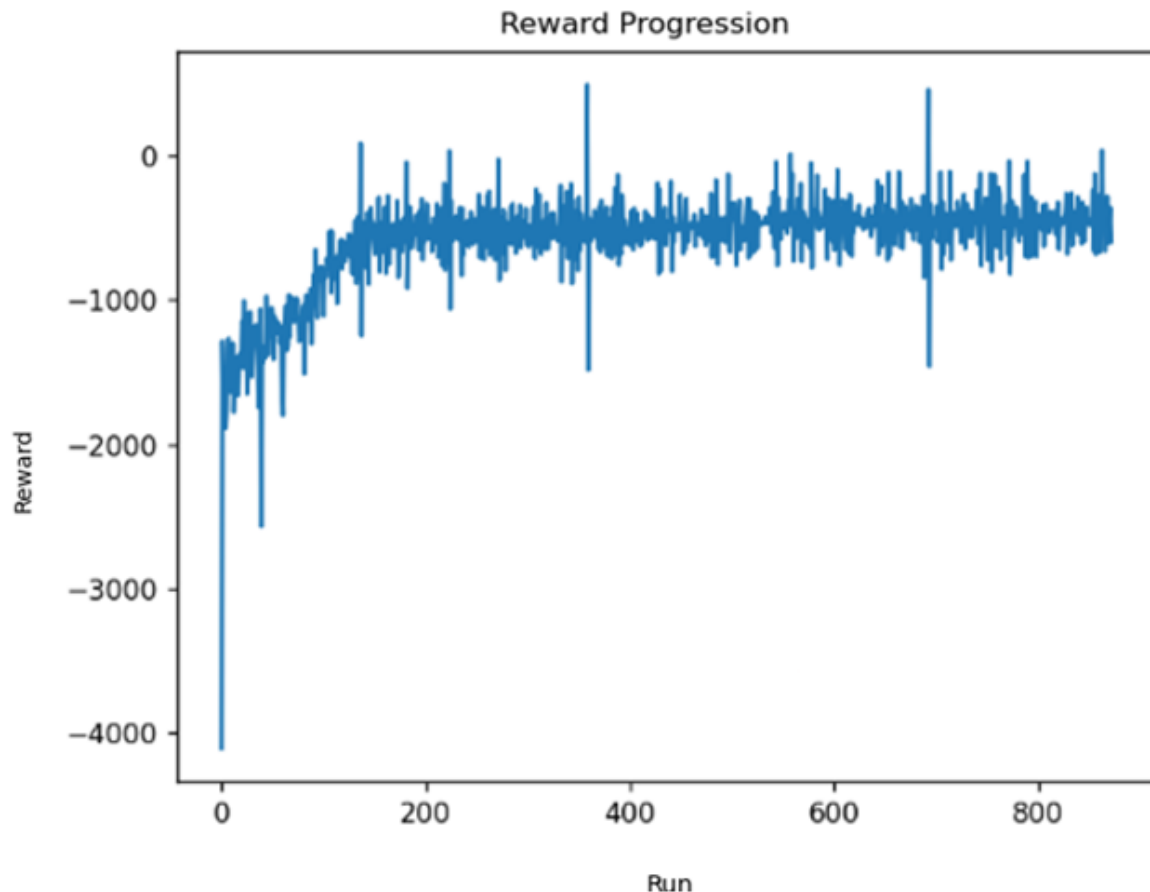
# Dying gives a penalty. The bigger the score, the lower the penalty
    if not gamestate["is_alive"]:
        reward -= (500 - (gamestate["score"] * 0.0338))

# Add reward to total reward
    self.total_reward["total"] += reward

    return reward
```


Test 6

Deze test heeft een combinatie gebruikt van vorige tests. Hier wordt in plaats van een penalty een reward gegeven voor de tijd die is verstreken, met het idee dat Pacman daardoor langer levend zou blijven. Ook wordt de doodgaan penalty verminderd aan de hand van de score die is behaald. Dit geeft een zelfde grafiek als test 2, maar ook deze is niet beter dan test 2.



```
def _get_reward(self, gamestate: dict, action: int):
    reward = 0

    # Increasing score gives a reward
    reward += (gamestate["score"] - self.score)
    self.score = gamestate["score"]

    # Reaching Level 2 gives a (big) reward
    if gamestate["level"] > 0:
        reward += 10000

    # Passing of time gives a reward (survive longer)
    reward += 0.5

    # Pressing buttons is not free
    if action != 0:
        reward -= 5
```



```
# Dying gives a penalty. The bigger the score, the lower the penalty
if not gamestate["is_alive"]:
    reward -= (500 - (gamestate["score"] * 0.0338))

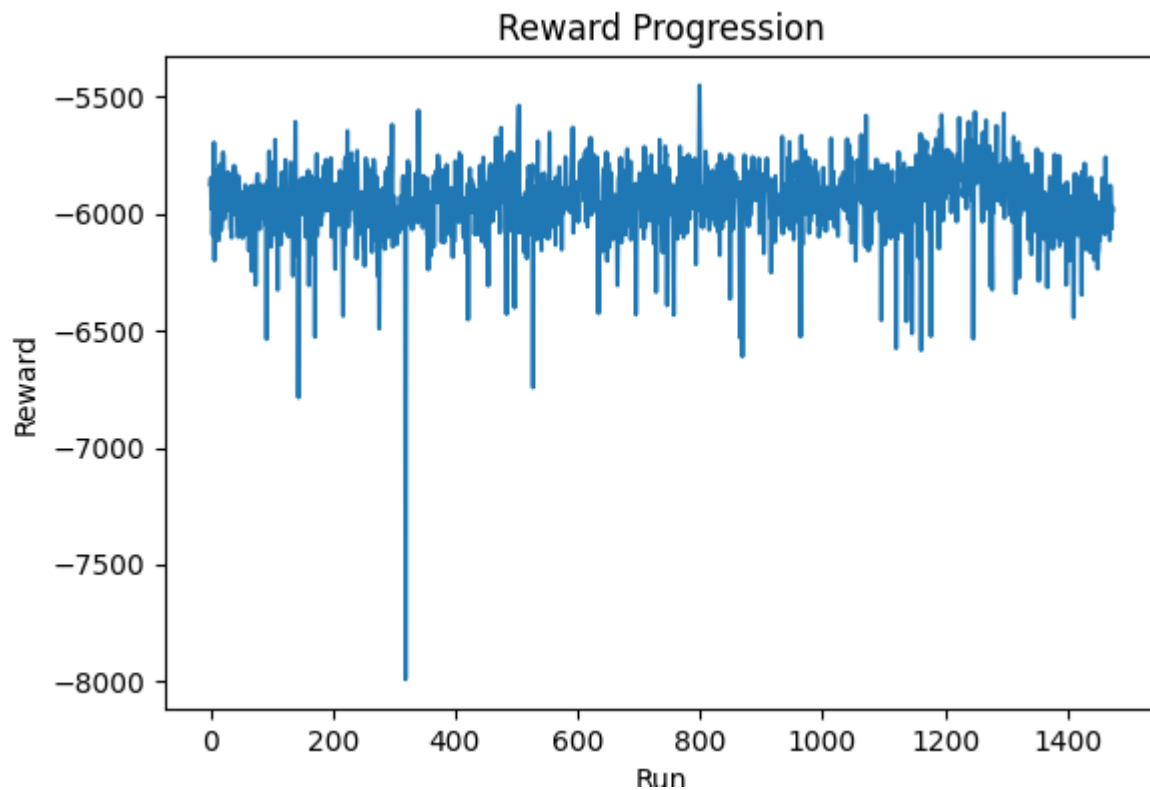
# Add reward to total reward
self.total_reward["total"] += reward

return reward
```


Test 7

In deze test is gebruik gemaakt van een ander algoritme, namelijk Deep Q-Learning. Dit algoritme zou beter werken voor kleinere structuren zoals Pacman levels (Gnanasekaran, n.d.).

Deze grafiek laat echter zien dat dit algoritme ook geen goede optie is omdat, net zoals een aantal vorige grafieken, snel stagneert.



```
def _get_reward(self, gamestate: dict, action: int):
    reward = 0

    # Increasing score gives a reward
    reward += (gamestate["score"] - self.score)
    self.total_reward["score"] += (gamestate["score"] - self.score)
    self.score = gamestate["score"]

    # Reaching Level 2 gives a (big) reward
    if gamestate["level"] > 0:
        reward += 10000
        self.total_reward["level_complete"] += 10000

    # Passing of time gives a reward (survive longer)
    reward += 0.5
    self.total_reward["time_alive"] += 0.5

    # Pressing buttons is not free
    if action != 0:
        reward -= 5
```



```
        self.total_reward["button_presses"] -= 5

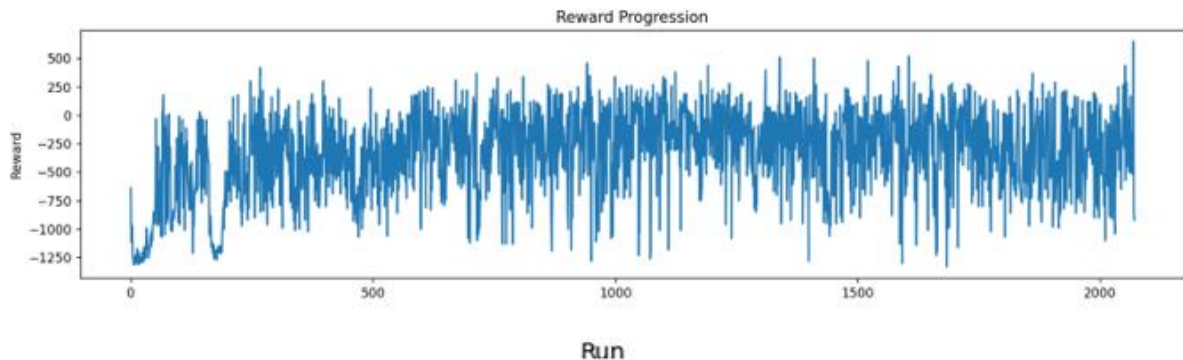
    # Dying gives a penalty
    if gamestate["is_alive"] == False:
        reward -= 5000
        self.total_reward["dying"] -= 5000

    # Add reward to total reward
    self.total_reward["total"] += reward

    return reward
```


Test 8

Tijdens deze run is specifiek gekeken naar het gedrag van Pacman en daarom zijn alle ghosts uit het spel gehaald. Het doel van deze test was om te kijken hoe goed reinforcement learning zou werken als Pacman in een statische omgeving zou zitten. Wat bleek is dat, ondanks er geen ghosts zijn, Pacman niet goed leert met reinforcement learning.



```
def _get_reward(self, gamestate: dict, action: int):
    reward = 0

    # Increasing score gives a reward
    reward += (gamestate["score"] - self.score)
    self.total_reward["score"] += (gamestate["score"] - self.score)
    self.score = gamestate["score"]

    # Reaching Level 2 gives a (big) reward
    if gamestate["level"] > 0:
        reward += 10000
        self.total_reward["level_complete"] += 10000

    # Passing of time gives a penalty (quicker runs are better)
    reward -= 0.5
    self.total_reward["time_alive"] -= 0.5

    # Add reward to total reward
    self.total_reward["total"] += reward

    return reward
```

4 Verwijzingen

Gnanasekaran, A., Faba, J. F., & An, J. (sd). *Final Reports*. Opgehaald van Stanford:
<http://cs229.stanford.edu/proj2017/final-reports/5241109.pdf>