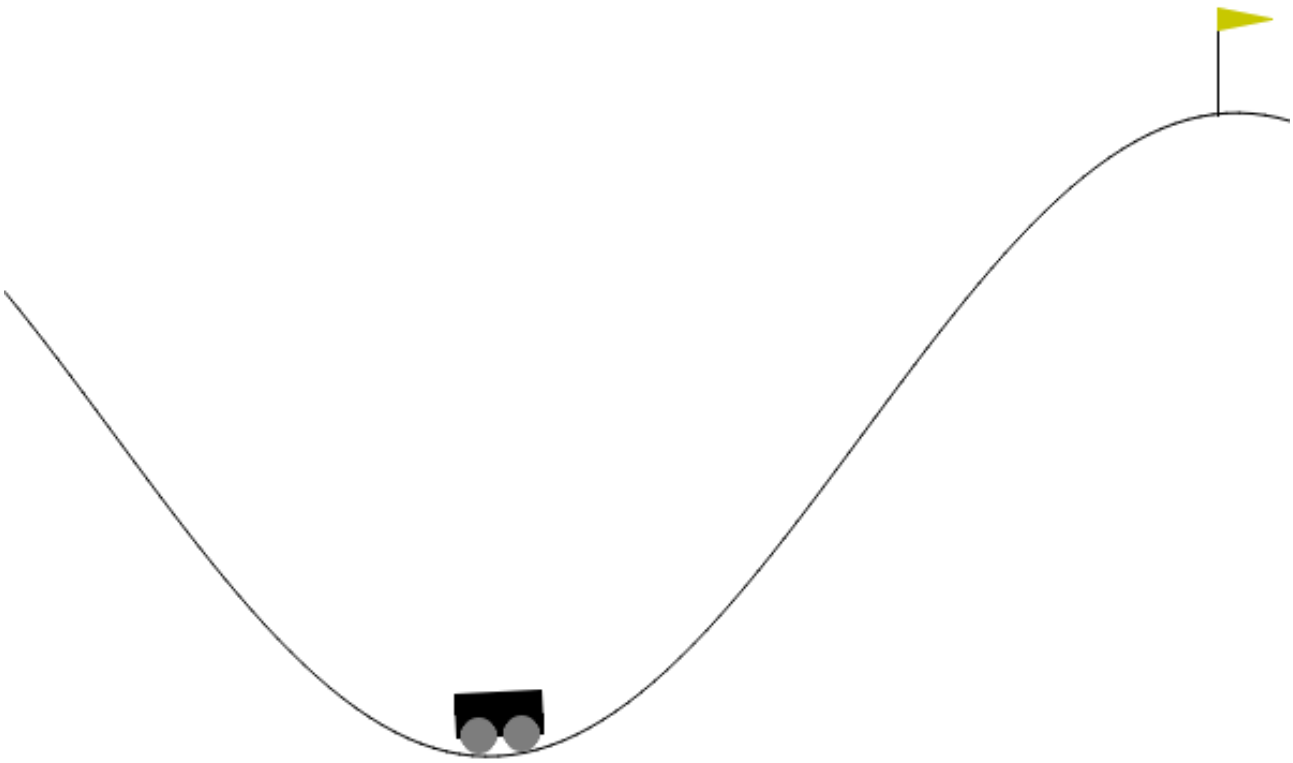
Sztuczna Inteligencja

Uczenie maszynowe na przykładzie problemu górskiego samochodu

Krzysztof Nasuta 193328, Filip Dawidowski 193433, Aleksander Iwicki 193354

Spis treści



1. Wstęp

1.1. Opis problemu

Problem samochodu górskiego to rodzaj gry, w której samochód o niskiej mocy musi wjechać na strome wzniesienie. Ponieważ grawitacja jest silniejsza niż silnik samochodu, pojazd nie może po prostu wjechać pod górę po stromym zboczu. Samochód znajduje się w dolinie i musi nauczyć się wykorzystywać energię potencjalną, wjeżdżając na przeciwległe wzniesienie, zanim będzie mógł dotrzeć do celu na szczycie wzniesienia położonego po prawej stronie.

1.2. Implementacja

Implementację wszystkich metod przeprowadzono w języku Python z wykorzystaniem bibliotek: `numpy`, `matplotlib`, `torch` oraz `gymnasium`. `Open AI Gymnasium` to zestaw narzędzi i środowisk do testowania algorytmów uczenia maszynowego. W naszym przypadku wykorzystaliśmy środowisko `MountainCar-v0`, które jest jednym z dostępnych w bibliotece. Biblioteka `Pytorch` została wykorzystana do implementacji sieci neuronowej dla metody Deep Q-Learning.

2. Środowisko

2.1. Przestrzeń obserwacji

Stan środowiska reprezentują dwie zmienne: `Velocity` reprezentująca prędkość samochodu oraz `Position` reprezentująca jego położenie na osi x. Prędkość przyjmuje wartości z przedziału $[-0.07, 0.07]$, a pozycja z przedziału $[-1.2, 0.6]$.

2.2. Przestrzeń akcji

Samochód ma do dyspozycji trzy akcje: `0` - przyspieszenie w lewo, `1` - brak przyspieszenia, `2` - przyspieszenie w prawo.

2.3. Nagroda

Nagroda wynosi -1 za każdą iterację. Pozwala to na premiowanie samochodu za jak najszybsze dotarcie do mety.

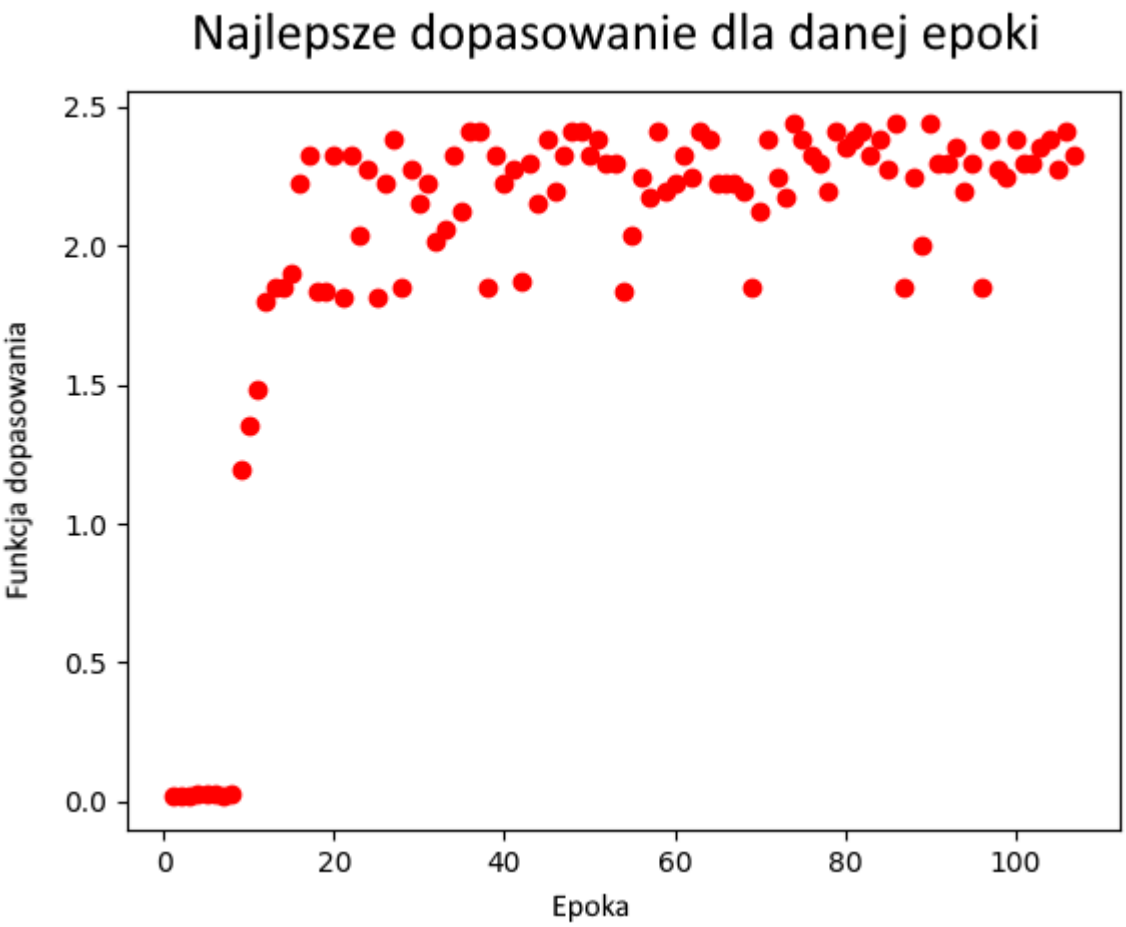
2.4. Warunki końcowe

Epizod kończy się sukcesem, gdy samochód dotrze do mety, czyli osiągnie pozycję 0.5 na osi x, lub porażką, gdy przekroczy maksymalną liczbę kroków, która wynosi 200. W trakcie trenowania modelu zezwolono na większą liczbę kroków, aby model mógł nauczyć się, jak najlepiej poruszać po środowisku.

3. Metody rozwiązania

3.1. Genetyczna metoda aproksymacji funkcji

Metoda polegała na stworzeniu wielomianów dwóch zmiennych pierwszego stopnia $W(x, y) = axy + bx + cy + d$ gdzie x odpowiada położeniu samochodziku, a y jego prędkości. Współczynniki a b c d są wybierane algorytmem genetycznym. Funkcją dopasowania jest maksymalna prędkość samochodziku w danej iteracji, a w przypadku dojazdu do końca, czasu, w jakim dojechał do mety. Model już po 20 epokach dojeżdżał powtarzalnie do mety (przy populacji 200). Ta metoda osiągała najlepsze wyniki i robiła to po bardzo krótkim czasie treningu (najprawdopodobniej też ze względu na wysoką prostotę zrównoleglania). Niestety, metoda aproksymacji funkcji jest najmniej uniwersalna i dużo cięższej jest ją przełożyć na inne problemy.



Epoka	Dopasowanie	Średni czas dojazdu	Najszybszy czas	Procent ukończonych
10	1.351	163.126	149	745/1000 (74.5%)
20	2.326	145.973	85	1000/1000 (100%)
30	2.381	106.284	85	1000/1000 (100%)
40	2.41	115.374	83	1000/1000 (100%)
80	2.439	113.589	83	1000/1000 (100%)

Metoda aproksymacji funkcji osiągnęła sukces już po 20 epokach, a jej skuteczność wynosiła 100%. Wynika to z faktu, że funkcja jest w stanie bardzo dobrze przybliżyć

zależności między pozycją, prędkością a akcją, co pozwala na osiągnięcie optymalnego rozwiązania w krótkim czasie. Niestety, metoda ta jest mało uniwersalna i niekoniecznie sprawdziłaby się w bardziej złożonych problemach.

3.2. Metoda Q-learning

Q-learning to metoda uczenia ze wzmocnieniem, która polega na uczeniu się wartości akcji w określonym stanie. W naszym przypadku stanem jest para (pozycja, prędkość), a akcją przyspieszenie w lewo, brak przyspieszenia lub przyspieszenie w prawo. Wartość akcji w stanie s oznaczamy jako $Q(s, a)$, gdzie a to akcja. Wartość ta jest aktualizowana zgodnie z równaniem Bellmana:

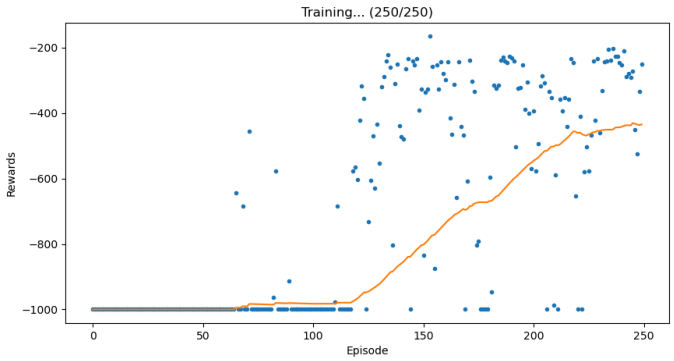
$$Q(s, a) = Q(s, a) + \alpha(r + \gamma * \max(Q(s', a)) - Q(s, a))$$

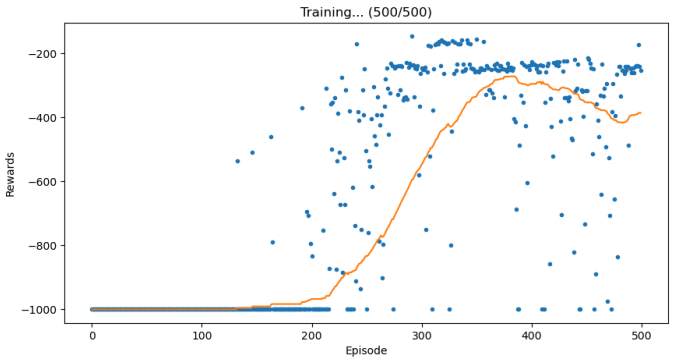
gdzie s-stan, a-akcja, r - nagroda, s' - następny stan, α - współczynnik uczenia, γ - czynnik osłabiający. W naszym przypadku $\alpha = 0.9$, $\gamma = 0.9$.

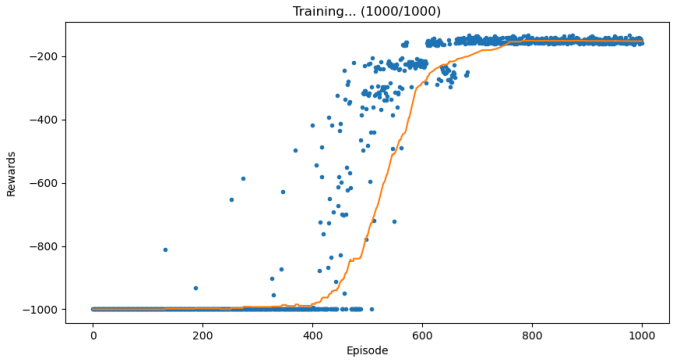
Wartości Q przechowywane są w tabeli, która jest aktualizowana w trakcie uczenia. Są one inicjalizowane są na 0 oraz są aktualizowane w trakcie uczenia, aż do osiągnięcia warunku końcowego. W naszym przypadku warunkiem końcowym jest osiągnięcie pozycji 0.5 na osi x lub przekroczenie maksymalnej liczby kroków, która wynosi w trakcie uczenia 1000. W trakcie uczenia w początkowych epizodach samochód porusza się czasami w sposób losowy, a innym razem na podstawie wartości odczytanych z tabeli Q. Prawdopodobieństwo tego, czy samochód wykona ruch losowy czy zdeterminowany przez tabelę Q zależy od wartości, która na początku ma wartość 1 dla losowego ruchu, a z każdym epizodem zmniejsza się o $\frac{2}{n}$, gdzie n to liczba epizodów. Gdy prawdopodobieństwo osiągnie wartość 0, samochód porusza się tylko na podstawie maksymalnej wartości dla danego stanu odczytanego z tabeli Q.

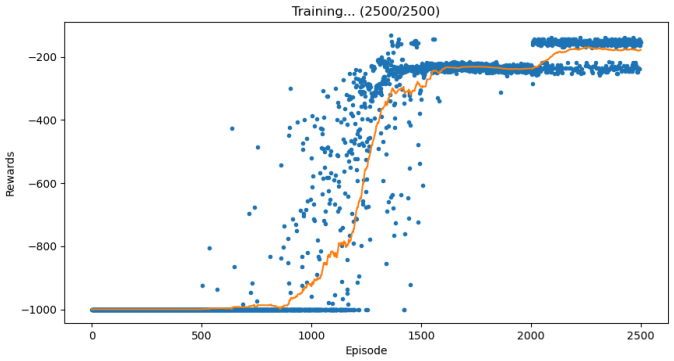
Zaletą metody Q-learning jest prostota implementacji oraz to, że nie wymaga ona wstępnego modelu środowiska, a jest w stanie wytworzyć ten model całkowicie samodzielnie na podstawie rezultatów akcji zwracanych przez środowisko.

Wadą jest to, że występuje konieczność dyskretyzacji stanu, co może prowadzić do utraty informacji, a co za tym idzie, ilość epizodów potrzebnych do wytrenowania modelu może być znacznie większa niż w przypadku innych metod.









Epizod	Średni czas	Minimalny czas	Procent skończonych
250	624.678	252	0/1000 (0.00%)
500	228.961	166	386/1000 (38.6%)
1000	151.065	135	1000/1000 (100%)
2500	182.971	140	655/1000 (65.5%)
5000	199.922	146	434/1000 (43.4%)

Metoda Q-Learningu osiągnęła sukces jedynie dla 1000 epizodów, a jej skuteczność wynosiła 100%. Na wykresie widzimy, że funkcja osiągnęła zbieżność do optymalnego rozwiązania. Dla 2500 i 5000 epizodów metoda nie osiągnęła sukcesu, a jej skuteczność wynosiła odpowiednio 65.5% i 43.4%.

3.3. Metoda Deep Q-Learning

Metoda głębokiego Q-Learningu jest algorytmem bazującym na klasycznym Q-Learningu, który zastępuje Q-table siecią neuronową. W naszym przypadku składa się ona z trzech warstw: wejściowej, ukrytej i wyjściowej. Warstwa wejściowa składa się z dwóch neuronów, które reprezentują pozycję i prędkość samochodu. Rozmiar warstwy ukrytej wynosi 64. Warstwa wynikowa zwraca trzy wartości, reprezentujące wartości akcji: przyspieszenie w lewo, brak przyspieszenia, przyspieszenie w prawo. Wartości te są wykorzystywane do wyboru akcji.

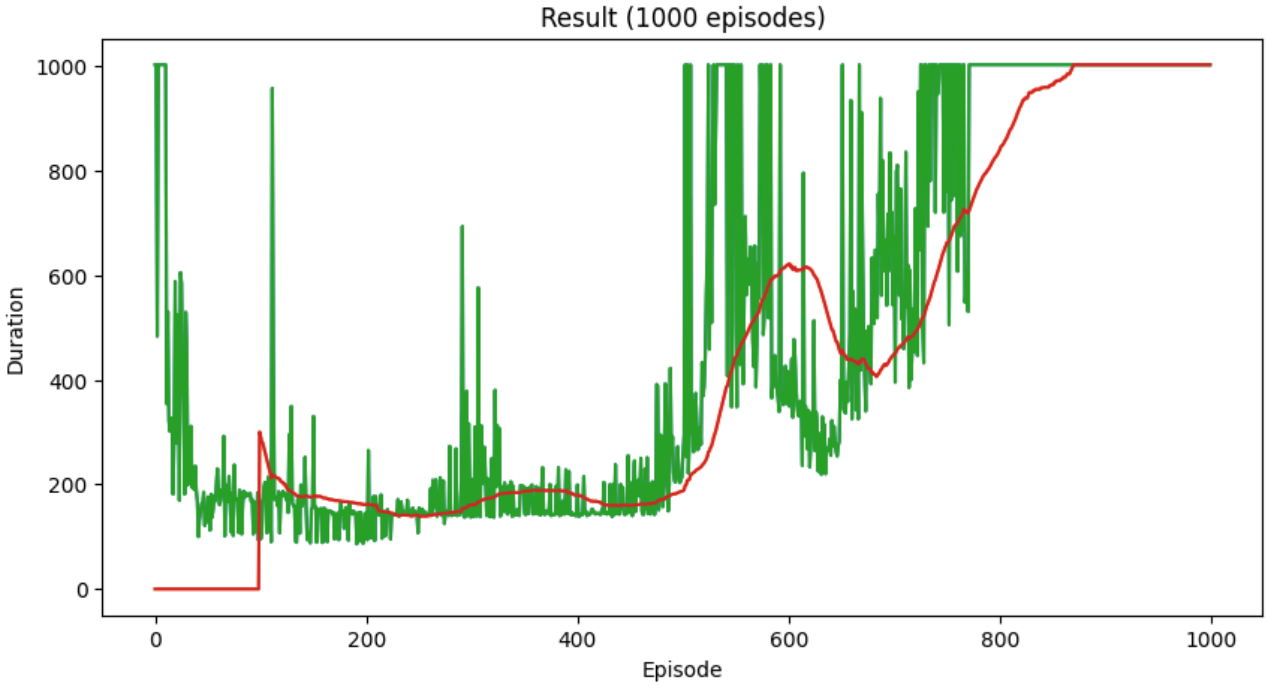
Dodatkowo zaimplementowano `Replay Memory`, czyli strukturę opisującą zmiany stanów w ostatnich n krokach. W trakcie uczenia wybierane są losowe próbki z tej pamięci, które są wykorzystywane do aktualizacji wag sieci. Pozwala to na zwiększenie stabilności uczenia.

W metodzie tej wykorzystano także ‘target network’, czyli sieć, która jest klonem sieci głównej, ale jej wagi są aktualizowane rzadziej. To również pomaga w stabilizacji procesu uczenia.

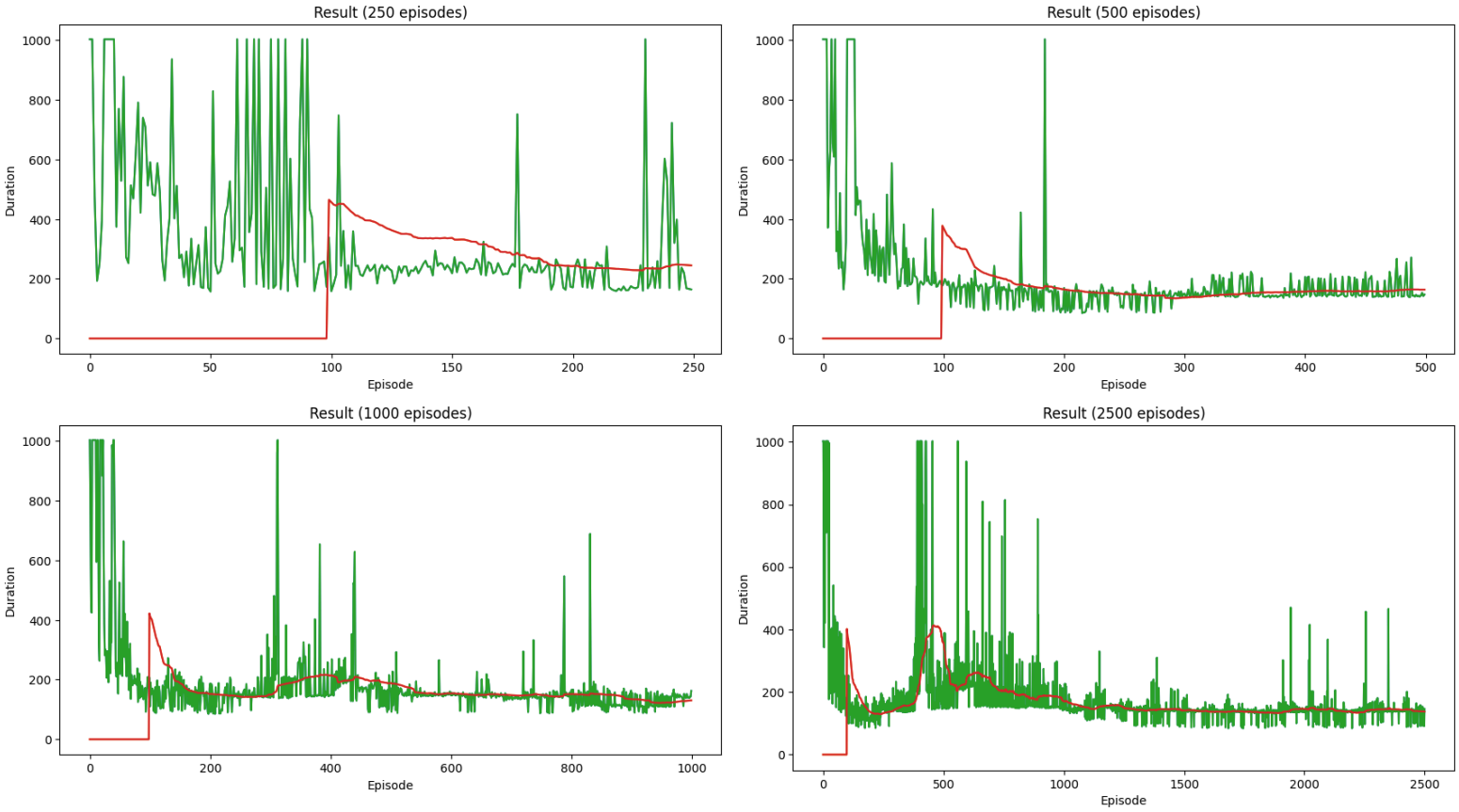
Zaletą względem klasycznej metody Q-Learningu jest brak wymagania dyskretyzacji stanu. Metoda Deep Q-Learning jest bardzo uniwersalna, i można ją zastosować do wielu problemów.

Wadą jest większe zużycie zasobów obliczeniowych, a także wyższy poziom skomplikowania w implementacji.

Ponadto Deep Q-Learning nie gwarantuje zbieżności do optymalnego rozwiązania. W naszym przypadku osiągnął on jednak bardzo dobre wyniki, osiągając sukces w znaczącej ilości testów.



Powyższy wykres przedstawia wyniki metody Deep Q-Learning po 1000 epizodach. Widzimy, że pomimo początkowych sukcesów, model zaczyna popełniać błędy i nie jest w stanie osiągnąć zbieżności do optymalnego rozwiązania.



Epizod	Średni czas trwania	Minimalny czas	Ilość skończonych
250	174.041	149	703/1000 (70.3%)
500	161.635	138	728/1000 (72.8%)
1000	143.31	85	994/1000 (99.4%)
2500	135.903	83	975/1000 (97.5%)

Jak widać, dla 1000 i 2500 epizodów metoda osiągnęła sukces. Jej skuteczność wynosiła odpowiednio 99.4% i 97.5%.

4. Podsumowanie

Wszystkie metody okazały się skuteczne w rozwiązaniu problemu samochodu górskiego. Metoda aproksymacji funkcji osiągnęła najlepsze wyniki najszybciej z wszystkich metod, jednak jest to metoda najmniej uniwersalna, z powodu nadmiernego dopasowania do problemu. Metoda Q-Learningu osiągnęła w większości przypadków niezadowalające wyniki jednocześnie potrzebują wielu epizodów uczenia. Od pewnej wartości epizodów zauważamy też spadek skuteczności tej metody, więc nie jesteśmy w stanie uzyskać za jej pomocą lepszych wyników stale zwiększając ilość epizodów uczenia. Metoda Deep Q-Learningu osiągnęła bardzo dobre wyniki, a także jest najbardziej uniwersalna z wszystkich metod, lecz wymaga znacznie więcej zasobów obliczeniowych oraz jest trudniejsza w implementacji.