Projekt – Sztuczna inteligencja

Implementacja Deep Q-Learning: Agent grający w grę typu “Flappy Bird”

Maciej Winiecki 198363, Kacper Grzelakowski 197897, Jędrzej Jasiniecki 197993

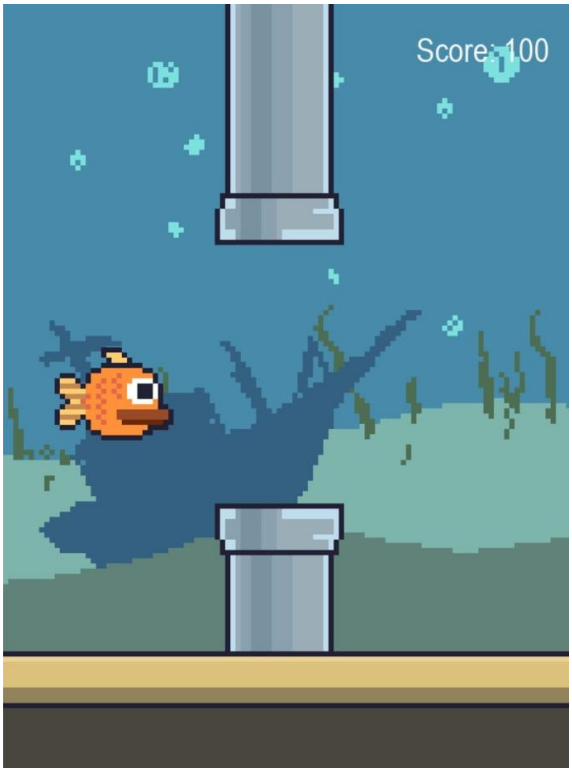
Wstęp

Celem projektu była implementacja i analiza metody uczenia ze wzmocnieniem o nazwie **Deep Q-Learning**. W celu zaprezentowania omawianej metody w praktyce w przystępny sposób, postanowiliśmy stworzyć agenta, który będzie uczył się grania w grę na podstawie swoich wcześniejszych ruchów (pomyłek i sukcesów).

Implementacja gry

Pierwszym krokiem była implementacja gry, w którą będzie grał nasz agent. Wspólnie wybraliśmy znaną grę “**Flappy Bird**”, która bardzo dobrze wpasowała się w koncepcję z uwagi na kilka kluczowych aspektów. Po pierwsze, zasady gry nie są skomplikowane - należy tak pokierować postacią, aby ominęła przeszkody i zdobyła jak największy wynik. Po drugie, gra opiera się na zręczności - gracz musi uważnie obserwować położenie postaci i jej odległość od przeszkód, aby w dobrym momencie wykonać skok i tym samym uchronić się od przegranej. Ostatnim aspektem gry, który utwierdził nas w przekonaniu, że jest ona dobrym wyborem, była stosunkowo prosta implementacja, która pozwoliła nam na przeznaczenie większej ilości czasu na dopracowywanie samego agenta. Do implementacji gry wykorzystaliśmy język **Python** i środowisko **PyCharm**, z uwagi na możliwość dołączenia bibliotek **Pygame** oraz **PyTorch**.

Zasady gry są proste, gracz poprzez wciśnięcie spacji utrzymuje postać w powietrzu (w wodzie) i nie może pozwolić na uderzenie w zbliżające się rury, podłoże lub sufit.



Zrzut ekranu przedstawiający implementację gry typu "Flappy Bird"

Implementacja agenta

Następnym krokiem była implementacja agenta, którego zadaniem było zdobycie jak największej liczby punktów. Zdecydowaliśmy się wykorzystać **uczenie ze wzmocnieniem** (reinforcement learning) w oparciu o algorytm **Deep Q-Learning**. Agent sam uczył się zasad gry i obecnych w niej zależności. W każdej klatce przesyłane były informacje o stanie gry: położenie i prędkość postaci, odległość postaci od najbliższej rury. Ponad to, do agenta przesyłana była specjalna wartość o nazwie **reward**, która określała jak dobre były akcje wykonywane przez agenta w oparciu o zaprojektowany przez nas system kar i nagród. Na podstawie ewaluacji, agent stopniowo poprawiał swoje osiągi.

System nagród i kar obejmuje:

- Kara za zderzenie z otoczeniem - zderzenie z ziemią lub sufitem skutkuje przypisaniem wysokiej ujemnej nagrody (np. -20).
- Kara za zderzenie z rurą - podobnie jak w przypadku zderzenia z otoczeniem, agent jest karany za taką akcję, jednak tutaj wysokość kary zależy od odległości agenta od środka otworu między rurami. Sprawia to, że wszystkie nieudane próby przejścia przez szparę nie są traktowane zero-jedynkowo - niektóre próby

są lepsze od innych. Zderzenie z rurą jest mniej karane niż zderzenie z otoczeniem.

- Nagroda za minięcie rury - agent ponownie otrzymuje nagrodę bazującą na jego położeniu względem środka otworu, tym razem dodatnią, promującą utrzymywanie się blisko optymalnego toru lotu.
- Niewielka nagroda za przeżycie - za każdy krok bez kolizji agent otrzymuje niewielką dodatnią nagrodę (0.1), co wzmacnia strategię unikania ryzyka i długiego przetrwania.

Główne komponenty agenta

- **Policy Network – t**o główna sieć neuronowa wykorzystywana do wybierania akcji podczas eksploracji i treningu. W każdej iteracji uczy się przewidywać wartość nagród (Q-values) dla danego stanu i dostępnych akcji. Jest aktualizowana na podstawie doświadczeń pobranych z bufora.
- **Target Network** – Jest to kopia sieci policy, która służy do stabilizacji procesu uczenia. Jej wagi są synchronizowane z siecią policy co określoną liczbę kroków (np. co 1000 kroków treningowych). Dzięki temu unika się niestabilności wynikającej z jednoczesnej aktualizacji wartości Q i obliczania targetów.
- **Replay Buffor** – To struktura danych przechowująca doświadczenia agenta w postaci krotek: (stan, akcja, nowy stan, nagroda, koniec epizodu (true/false)). Dzięki niej możliwe jest losowe wybieranie doświadczeń do treningu.

Hiperparametry

- Współczynnik uczenia (alpha) = 0.0001
- Współczynnik przyszłej nagrody (gamma) = 0.99
- Wielkość próbki doświadczeń = 32
- Minimalna wartość współczynnika eksploracji (epsilon) = 0.01
- Spadek eksploracji (epsilon decay) = 0.9995

Wybór akcji

Na początku treningu, akcje wykonywane przez agenta są zupełnie losowe. Jest to celowy zabieg zwany **eksploracją** środowiska. Współczynnik eksploracji maleje w trakcie gry, w oparciu o wartość **epsilon decay.** Pozwala to na zmniejszenie liczby losowych ruchów, gdy agent pozna wystarczająco dużo zależności i będzie w stanie podejmować racjonalne decyzje.

Mechanizm ten określany jest jako epsilon-greedy:

- Z prawdopodobieństwem epsilon agent wykonuje losową akcję (eksploracja).
- W przeciwnym razie wybiera akcję o najwyższej przewidywanej wartości Q (eksploatacja).

Uczenie agenta

Trening agenta polega w głównej mierze na aktualizowaniu wag w sieci **policy,** aby lepiej przewidywała które akcje prowadzą do najlepszych nagród. Z bufora Replay Buffor pobierana jest losowa próbka doświadczeń. Każde doświadczenie przechowuje informacje o stanie gry przed wykonaniem akcji, informacje o wykonanej akcji (skok lub brak skoku), informacje o stanie gry po wykonaniu akcji, wartość reward oraz informację o tym, czy gra się zakończyła. Następnie z sieci policy odczytywane są wartości (Q) przewidywanej nagrody po wykonaniu danej akcji. Kolejnym krokiem jest obliczenie tak zwanych **targetów**, czyli docelowych wartości Q. Są one obliczane jako suma otrzymanej nagrody i największej przewidywanej wartości Q dla danego stanu przemnożonej przez współczynnik przyszłej nagrody (discount rate). Porównanie wartości Q i targetów pozwala wyznaczyć **stratę,** którą minimalizujemy poprzez propagację wsteczną. Na sam koniec aktualizujemy wartości sieci policy, aby zmniejszyć różnicę pomiędzy wartościami przewidzianymi, a rzeczywistymi.

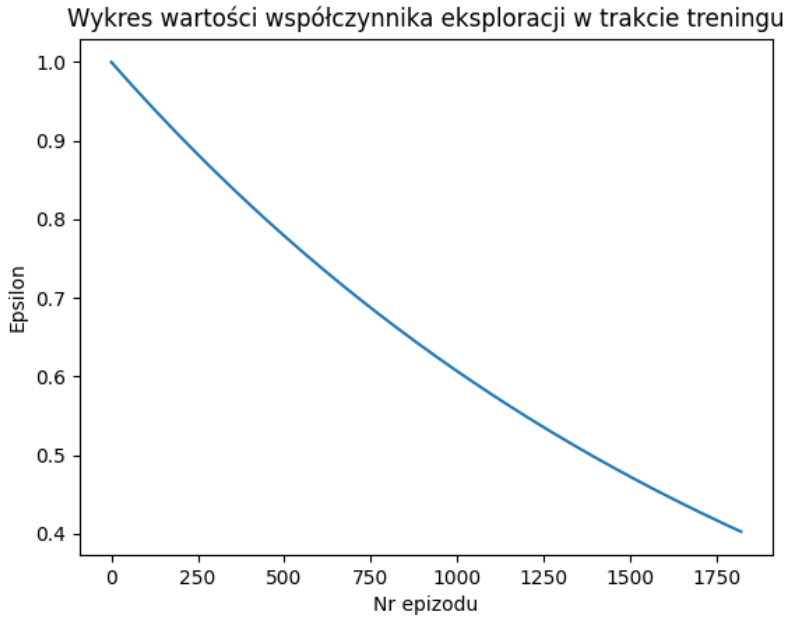
Dodatkowe funkcjonalności

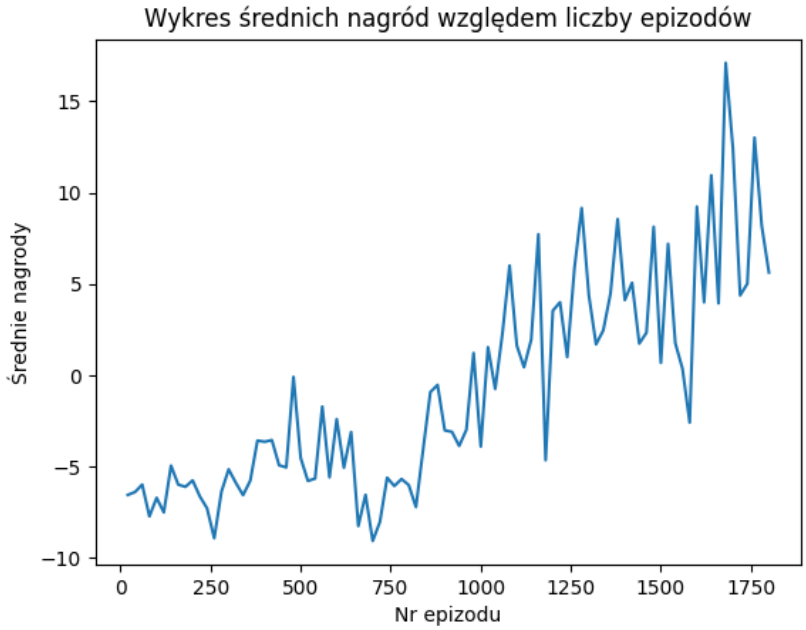
W celu dokładniejszej analizy zaimplementowaliśmy dodatkowe funkcjonalności do naszego programu. Podczas testowania modelu generowane były wykresy przedstawiające wskaźnika eksploracji oraz średnie wartości reward w kolejnych rundach.

Otrzymane rezultaty

Stworzyliśmy dwie instancje agentów, które różniły się czasem treningu. Jeden z nich trenowany był ok. 1 h, a drugi – ok. 2 h 45 min. Otrzymane wyniki znacznie się różniły, co potwierdziło, że czas treningu pełni kluczową rolę w poprawnym nauczeniu agenta.

Agent 1. (1 h treningu):





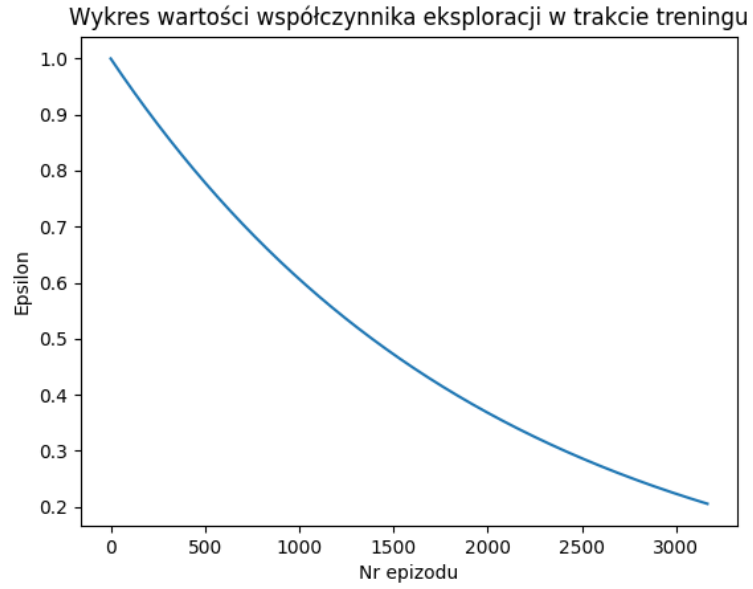
Na podstawie wykresu nagród można zauważyć, że agent uczył się w trakcie treningu – początkowo średnie nagrody na epizod były niższe niż -5, natomiast pod koniec osiągał już wartości w okolicach 10. To wyraźny sygnał, że sieć policy poprawia swoje przewidywania i strategię działania. Aby ocenić rzeczywistą skuteczność agenta, przeprowadzono test: agent rozegrał 100 epizodów, a wyniki przedstawiają się następująco (gdzie 1 punkt oznacza minięcie jednej przeszkody):

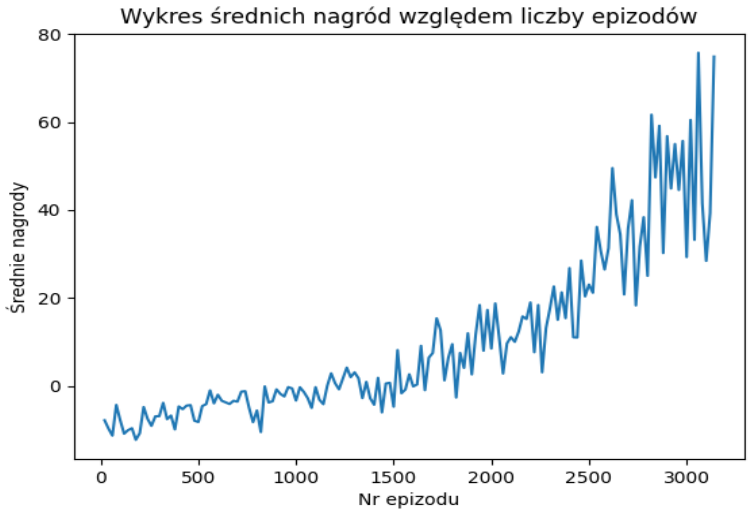
Średnia	Mediana	Najgorszy wynik	Najlepszy wynik
0.69	0.5	0	3

Mimo zauważalnego postępu w nauce, rezultaty testowe są nadal niezadowalające – w połowie przypadków agentowi nie udało się przejść nawet jednej przeszkody. Najwyższy osiągnięty wynik to zaledwie 3, co wskazuje, że strategia podejmowania decyzji wciąż jest bardzo ograniczona.

Z drugiej strony, agent radzi sobie wyraźnie lepiej niż przy losowych działaniach, a trend wzrostu nagród sugeruje, że wydłużenie treningu może przynieść dalsze korzyści.

Agent 2. (2 h 45 min treningu):





W przypadku drugiego agenta, wzrost efektywności jest nie tylko kontynuowany, ale wręcz przyspieszony. Średnie nagrody osiągają poziom 80, co oznacza znacznie większą liczbę przejść przez przeszkody w każdym epizodzie.

Wyniki po 100 testowych epizodach:

Średnia	Mediana	Najgorszy wynik	Najlepszy wynik
21.85	17.0	0	105

Ten agent prezentuje już wyraźnie bardzo dobre wyniki – mediana wyniku na poziomie 17 oznacza, że w połowie gier pokonuje co najmniej 17 przeszkód. Najlepszy wynik – 105 – pokazuje, że potrafi utrzymać się przy życiu przez bardzo długi czas. Najgorszym wynikiem jaki uzyskał pozostaje wciąż 0, agent wciąż nie jest idealny i potrafi w najprostszej sytuacji popełnić błąd.

Obserwując grę agenta na bieżąco, można zauważyć, że często podejmuje on duże ryzyko - szczególnie gdy musi w krótkim czasie z niskiej wysokości polecieć do góry. Bardzo często w takich sytuacjach udaje mu się o włos a w innych uderza głową w rurę, co więcej prawie wszystkie z jego śmierci polegają na tym, że za wolno poleciał w górę. Prawdopodobnie, gdyby był uczony jeszcze dłużej, tego błędu by się również oduczył.

Wnioski:

- **Skuteczność działania agenta rośnie wraz z czasem treningu** – zarówno na podstawie wykresów nagród, jak i rzeczywistych wyników w testowych grach.
- **Agent 1** wykazuje poprawę, ale nadal pozostaje mało skuteczny.
- **Agent 2** osiąga już poziom pozwalający mu przejść wiele przeszkód, a strategia działania jest znacząco lepsza.
- **Obserwowany spadek wartości epsilonu** koreluje z poprawą wyników – agent z czasem podejmuje coraz mniej losowych, a więcej opartych na predykcji działań.

Kontynuowanie treningu może pozwolić agentowi osiągnąć jeszcze większą skuteczność – warto więc eksperymentować z dłuższymi sesjami treningowymi lub ewentualnie modyfikacjami architektury sieci czy hiperparametrów.