Projekt Zespołowy: Zaawansowane Metody Uczenia Maszynowego

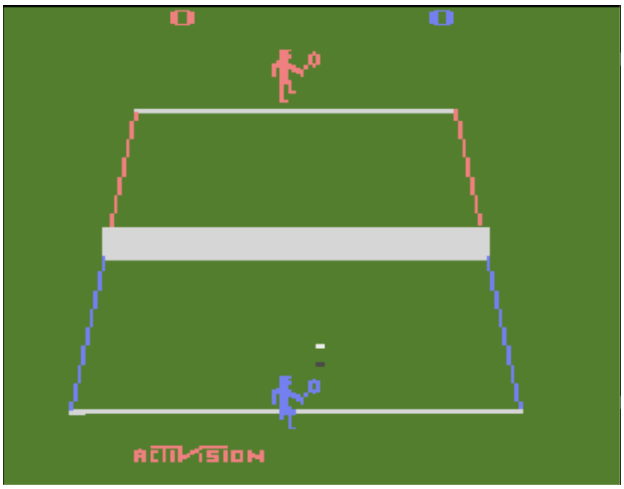
Alicja Osam-Gyaabin Mikołaj Zawada Karol Kociołek

***Streszczenie*—Celem projektu jest opracowanie dwóch modeli uczenia przez wzmacnianie w wybranym środowisku. W ramach projektu przeprowadzono analizę środowiska Tennis, opisano dwa różne algorytmy RL oraz wytrenowano modele, które zostały porównane.**

I. Wybór i Opis Środowiska

A. Wybór Środowiska

Do realizacji projektu zdecydowano się na wykorzystanie środowiska *Tennis*, będącego częścią rodziny środowisk Atari dostępnych w bibliotece Gymnasium. Szczegółowa dokumentacja dostępna jest pod adresem: https://www.gymlibrary.dev/environments/atari/tennis/.



Rysunek 1. Przykładowy obraz ze środowiska *Tennis*

B. Opis Środowiska

Środowisko Tennis symuluje mecz tenisa, w którym użytkownik kontroluje pomarańczowego zawodnika rywalizującego z komputerowo sterowanym niebieskim przeciwnikiem. Rozgrywka odbywa się zgodnie z zasadami tenisa, gdzie zwycięzcą zostaje ten zawodnik, który pierwszy zdobędzie co najmniej sześć gemów z przewagą minimum dwóch gemów. W przypadku osiągnięcia remisu 6-6, mecz kontynuowany jest aż jeden z zawodników zdobędzie przewagę dwóch gemów.

1) Przestrzeń Akcji: Przestrzeń akcji w środowisku *Tennis* jest dyskretna i obejmuje 18 możliwych działań, które agent może podjąć w trakcie gry. Akcje te pozwalają na realizację podstawowych ruchów, takich jak poruszanie się w górę, w dół, w lewo i w prawo, a także uderzenia piłki w różnych kierunkach i serwy.

2) Przestrzeń Obserwacji: Przestrzeń obserwacji w środowisku *Tennis* jest reprezentowana przez obraz w formacie RGB o wymiarach 210 pikseli wysokości, 160 pikseli szerokości i 3 kanałach kolorów (odpowiadających czerwieni, zieleni i niebieskiemu). Obraz dostarcza agentowi szczegółowych informacji o stanie środowiska, takich jak pozycje zawodników, trajektoria piłki, linie kortu czy siatka.

3) Importowanie Środowiska: Aby zaimportować środowisko *Tennis* w języku Python, można skorzystać z poniższego fragmentu kodu:

```
env = gym.make("ALE/Tennis-v5")
```

II. Wybór i Opis Algorytmów

Wybrano dwa algorytmy RL:

A. Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) to algorytm uczenia ze wzmocnieniem, który optymalizuje politykę tak, aby maksymalizować oczekiwany zwrot (*reward*), ograniczając jednocześnie zbyt duże zmiany w polityce.

Poniżej przedstawiono kluczowe elementy algorytmu PPO:

- *Polityka:* Polityka $\pi_\theta(a|s)$ określa prawdopodobieństwo wykonania akcji a w stanie s. Parametry θ są trenowane przy użyciu sieci neuronowej.
- *Funkcja korzyści:* Funkcja korzyści $A(s,a)$ mierzy, jak dobra była dana akcja a w stanie s w porównaniu do oczekiwań.
- *Ograniczenie zmian w polityce:* PPO wprowadza funkcję celu z ograniczeniem (*clipped objective*), która ogranicza zbyt duże zmiany w polityce, aby zapewnić stabilność uczenia.

1) Funkcja celu PPO: Funkcja celu PPO opiera się na współczynniku prawdopodobieństwa:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)},$$

gdzie π_θ to nowa polityka, a $\pi_{\theta_{\text{old}}}$ to polityka sprzed aktualizacji.

Celem PPO jest maksymalizacja następującej funkcji:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right],$$

gdzie:

- $r_t(\theta)$ to stosunek nowych do starych prawdopodobieństw,

- A_t to funkcja korzyści, która uwzględnia różnicę między oczekiwaniami a rzeczywistym wynikiem,
- ϵ to hiperparametr, który określa dopuszczalny zakres zmian (zazwyczaj $\epsilon = 0.1$ lub $\epsilon = 0.2$),
- clip$(r_t(\theta), 1-\epsilon, 1+\epsilon)$ ogranicza wartość współczynnika, aby zmiany w polityce były umiarkowane.

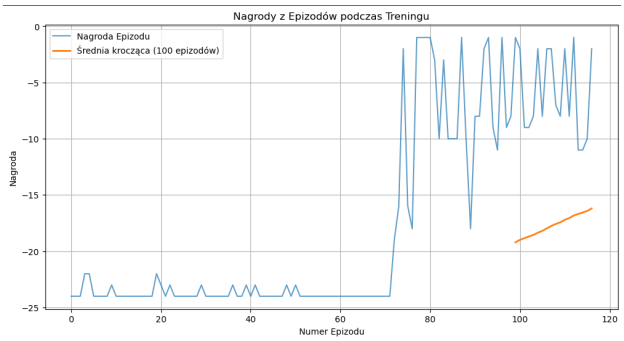
2) Główne kroki algorytmu PPO: Algorytm PPO składa się z następujących etapów:

1) Generowanie trajektorii: Symulator zbiera dane (stany, akcje, nagrody) w oparciu o bieżącą politykę π_θ.
2) Obliczanie funkcji korzyści: Na podstawie zebranych danych obliczana jest funkcja korzyści A_t, np. przy użyciu metody Generalized Advantage Estimation (GAE).
3) Aktualizacja polityki: Maksymalizowana jest funkcja celu $L^{\text{CLIP}}(\theta)$ przy użyciu algorytmu gradientowego.
4) Aktualizacja funkcji wartości: Model uczy się funkcji wartości $V(s)$, która przewiduje oczekiwaną nagrodę w danym stanie.
5) Powtarzanie procesu: Po aktualizacji modelu zbierane są nowe dane i proces jest kontynuowany.

PPO jest skutecznym i stosunkowo prostym algorytmem, który znajduje zastosowanie w różnych dziedzinach, takich jak gry komputerowe.

III. Trenowanie Modeli RL

A. Model 1: Proximal Policy Optimization (PPO)



Rysunek 2. Trening PPO *Tennis*

IV. Wnioski