Projekt sztucznej inteligencji - Zaganianie

Paweł Radosz s180046

Jan Czerech s180060

Paweł Cissewski s176446

9 czerwca 2021

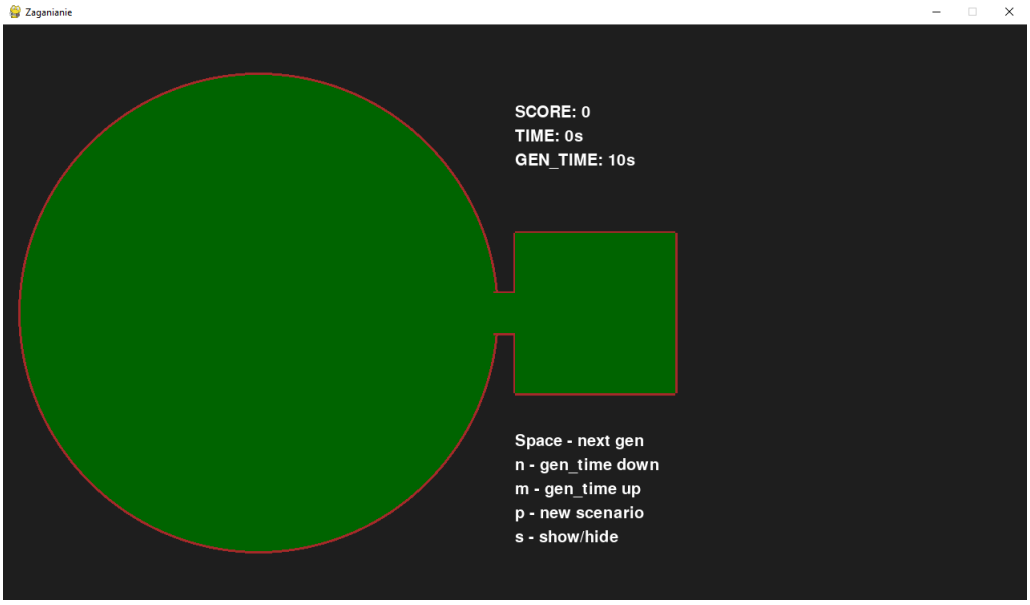
1 Wstęp

Tematem niniejszego projektu jest proces zaganiania. Głównym celem pracy było nauczenie danego modelu sieci neuronowej zagonienia owcy przez psa pasterskiego, dalej zwanego *agentem*. Projekt został stworzony w języku Python przy użyciu biblioteki **Pygame** w celu przeprowadzenia wielokrotnej symualcji oraz dodatkowo umożliwia wizualizacje problemu.

Sztuczna inteligencja została zaimplementowana przy użyciu sieci neuronowej, która wykorzystuje algorytm genetyczny. Szczegóły podanych algorytmów znajdują się w dalszej części dokumentu.

2 Definicja problemu zaganiania

W zaimplementowanej wersji problemu zaganianie jest procesem który polega na wymuszeniu ruchu obiektu (tutaj owcy) w określony kierunek/miejsce przez agenta (wilka), gdzie omawianym miejscem jest zagroda przedstawiona na poniższej ilustracji



Rysunek 1: Plansza symualcji

2.1 Wykaz oznaczeń na planszy

- SCORE - suma punktacji otrzymywana za zagonienie owcy przez wilka
- TIME - czas symualcji
- GEN TIME - czas istnienia pojedyńczej generacji

Można modyfikować niektóre parametry za pomocą następujących klawiszy

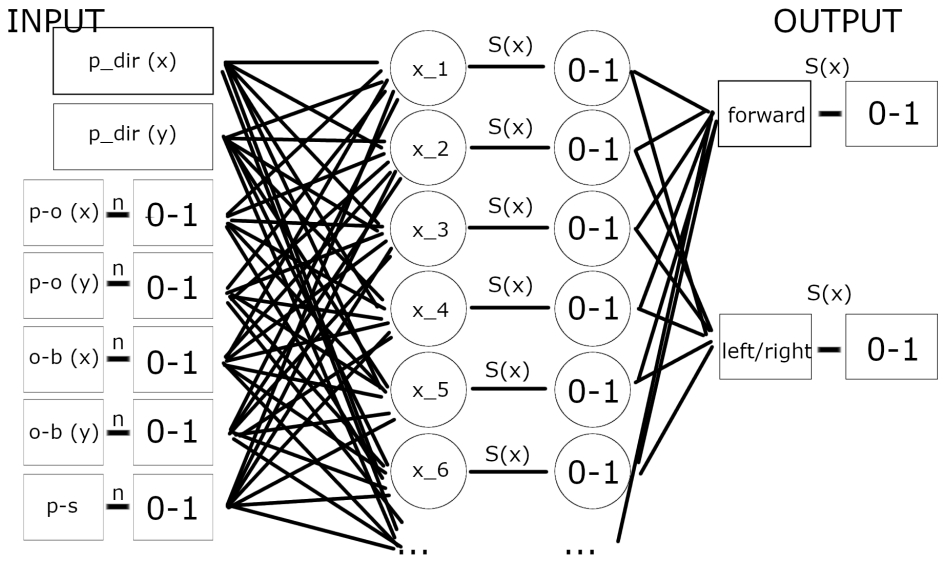
- Spacja - natychmiastowe rozpoczęcie kolejnej generacji
- n - zmniejszenie czasu istnienia generacji
- m - zwiększenie czasu istnienia generacji
- p - nowy scenariusz
- s - wyświetlanie dziesięciu najlepszych osobników z poprzedniej generacji

3 Symulacje

Symulacja rozpoczyna się od umieszczenia wilków oraz przypisanych do nich owiec na pozycjach startowych, dla agenta jest to środek pastwiska po której zaganiana jest owca, owca natomiast jest ustawiana na losowej pozycji. Agent składa się na dwa główne atrybuty, pierwszym z nim jest wektor kierunku w którym się porusza, jest on wykorzystywany na wejście sieci neuronowej opisanej w następnej sekcji dokumentu. Kolejnym atrybutem jest obszar dookoła wilka, służy ona jako odległość w której owca ucieka od wilka, tutaj jest to pole którym wilk odpycha owce w danym kierunku. Owca z kolei jest obiektem nie poruszającym się na planszy, jedyny ruch jaki owca wykonuje, jest wtedy kiedy wilk pcha owce w daną strone. Istnieją również interakcje agenta ze ścianami pastwiska (płotem), w momencie w którym wilk spotyka się z płotem jest wyliczana styczna do punktu zderzenia ze ścianą i na podstawie kąta, wilk przesuwa się wzdłuż płotu przed siebie, który następnie opuszcza w zależności od decyzji jakich podejmie w trakcie nauki.

Owca również posiada swój wektor kierunku w którym zostaje popchnięta przez wilka, zostało to zwizualizowane poprzez dodanie kropki na ciało owcy która oznacza jej głowe, tym samym wskazując stale na kierunek w który jest skierowana

4 Model sieci neuronowej



Rysunek 2: Sieć neuronowa

4.1 Wykaz oznaczeń

- p dir (x) - wektor kierunku psa na osi OX
- p dir (y) - wektor kierunek psa na osi OY
- p-o - odległość psa od owcy
- o-b - odległość owcy od bramy
- p-s - odległość psa od środka pastwiska
- S(x) - funkcja Sigmoid
- n - funkcja normalizująca

4.2 Opis sieci neuronowej

Danych wejściowych na sieć neuronową jest siedem, pierwsze dwa są to wektory kierunku agenta, pozostałe opisują odległość między:

- psem a owcą
- owcą a bramą do zagrody
- psem a środkiem pastwiska

Powyższe dane wejściowe są normalizowane na wartości od 0 do 1 i następnie kierowane do neuronów. Każde połączenie danej wejściowej i neuronu jest określane przez wagę która początkowo jest losową wartością, a następnie w kolejnych epokach algorytmu genetycznego są one aktualizowane na podstawie wyniku najlepszych osobników poprzedniej generacji, krzyżowania danych osobników oraz mutacji gorszych wyników. Testy symulacji wykazały że dwunaście neuronów jest liczbą najoptymalniejszą. Wynik

neuronów jest następnie normalizowany i przekierowywany na wyjście według odpowiednich wag, które działają analogicznie do wag które łączą wejście z neuronami. Wyjście jest również normalizowane na wartości od 0 do 1, które określają czy agent powinien/nie powinien się ruszyć do przodu, skręcić w prawo/lewo.

5 Algorytm genetyczny

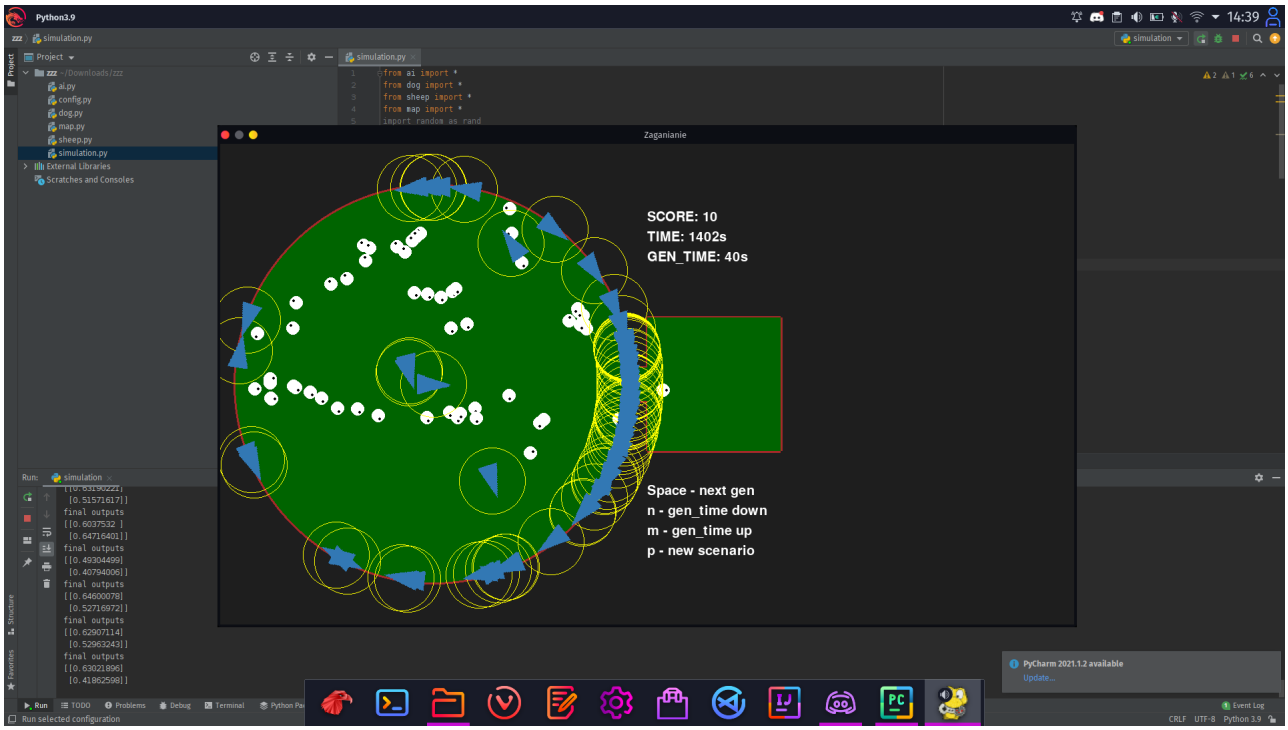
Zaimplementowany algorytm genetyczny składa się na poszczególne zmiennne:

- MUTATION WEIGHT CHANCE - szansa na mutacje wag
- CROSSOVER PROPORTION - proporcja krzyżowania
- CUT OFF - podział populacji na lepszą/gorszą
- BAD TO KEEP - proporcja zachowania gorszej populacji
- GOOD TO MUTATION - proporcja zmutowania lepszej populacji
- CROSSOVER MUTATION CHANCE - szansa na mutacje dziecka z krzyżowania
- GEN SIZE - liczba osobników w populacji

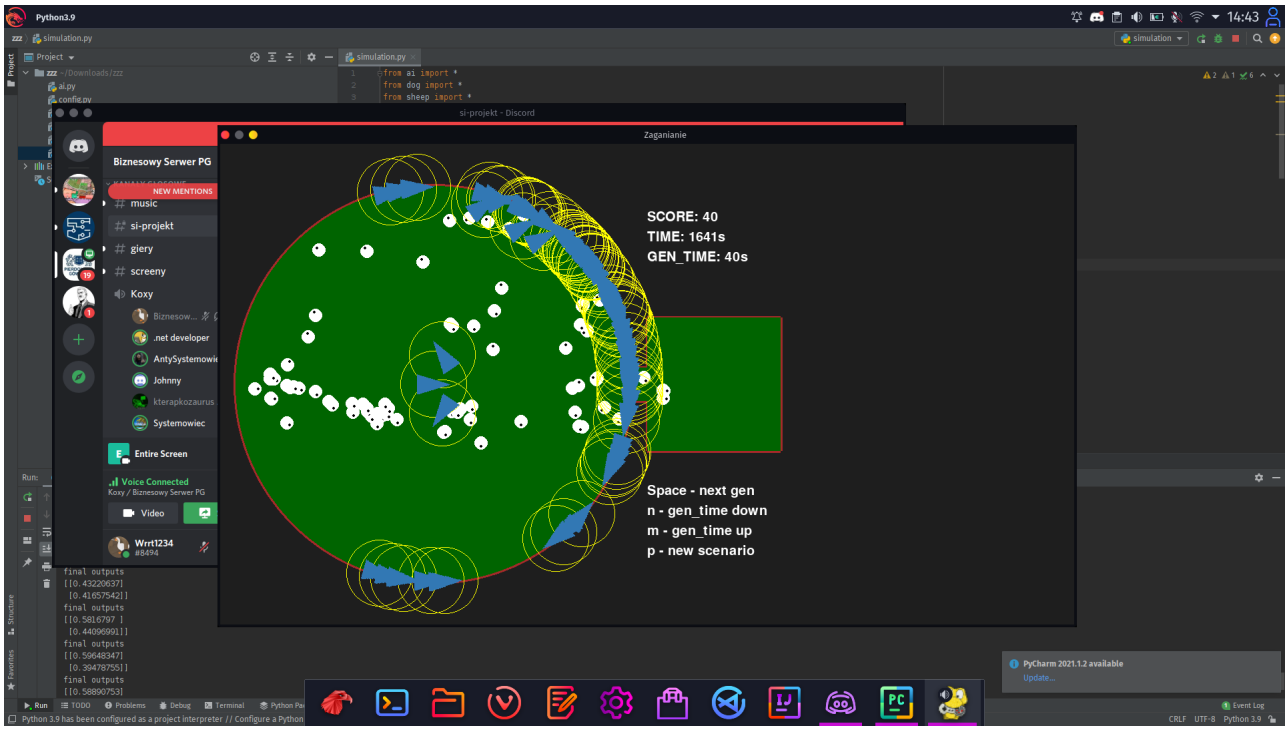
6 Obserwacje i wyniki

W początkowych fazach symulacji (tj. pierwszych epokach) czas istnienia populacji był ustawiany w przedziale od 10 do 20 sekund, ponieważ w tym czasie jesteśmy w stanie znaleźć pionierów dążących do dobrego rozwiązania. W kolejnych epokach można było zwiększyć czas na życie populacji, dając im czas na wykonanie odpowiednich manewrów w celu zagonienia owiec do zagrody. Po średnio 20 minutach od rozpoczęcia symulacji, agent był w stanie zagonić owce, co jest przedstawione na poniższych zrzutach ekranu. Przykładowe działanie zostało również udokumentowane filmem, dostępnym **tutaj**.

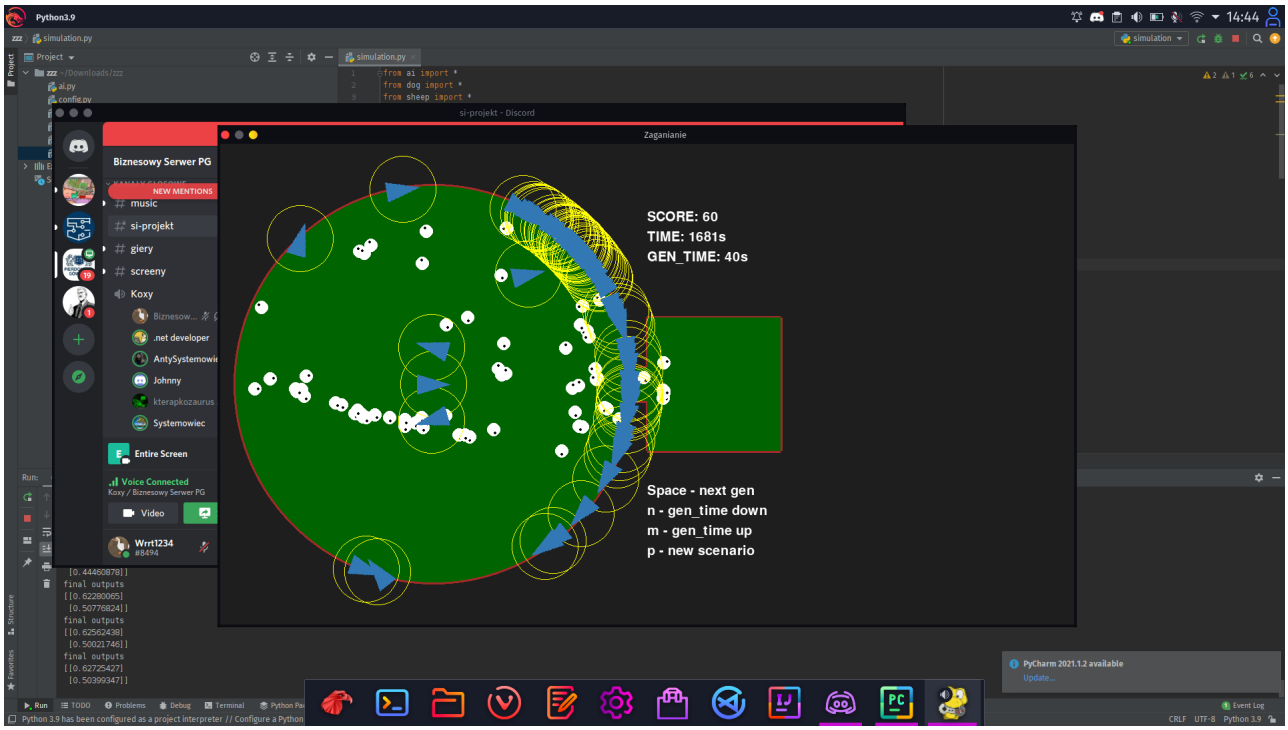
6.1 Wyniki



Rysunek 3: Doprowadzenie jednej owcy



Rysunek 4: Doprowadzenie czterech owiec



Rysunek 5: Doprowadzenie sześciu owiec

7 Wnioski

Finalnie udało się osiągnąć założony cel, tj. aby agent zagonił owce do zagrody, aczkolwiek ze względu na ograniczone możliwości sprzętowe, nie byliśmy w stanie wytrenować większej liczby populacji. Sto osobników z populacji było naszym limitem, ponieważ przy większej liczbie osobników komputer nie radził sobie z obliczeniami.