Projekt Sztuczna Inteligencja – Zderzenia cząsteczek przy użyciu sieci neuronowych

Michał Jaroszewicz 188568, Konrad Drozd 188567

1. Wstęp

- Cel projektu:
  - Celem projektu było odwzorowanie zderzeń sprężystych kul w pewnej ograniczonej przestrzeni przy wykorzystaniu sieci neuronowej.
- Skrócony opis działań:
  - Przy pomocy stworzonego wcześniej środowiska generowane są dane służące do uczenia sieci neuronowej. W ich skład wchodzą dane wejściowe, oraz oczekiwane dane wyjściowe. Do sieci neuronowej wprowadzane są wygenerowane dane, a rezultat porównywany z oczekiwanym.

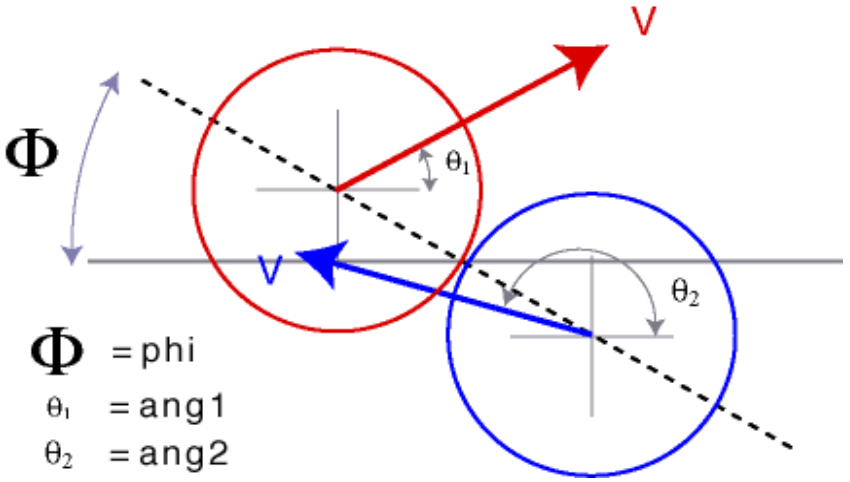
2. Teoria

Zderzenia idealnie sprężyste:

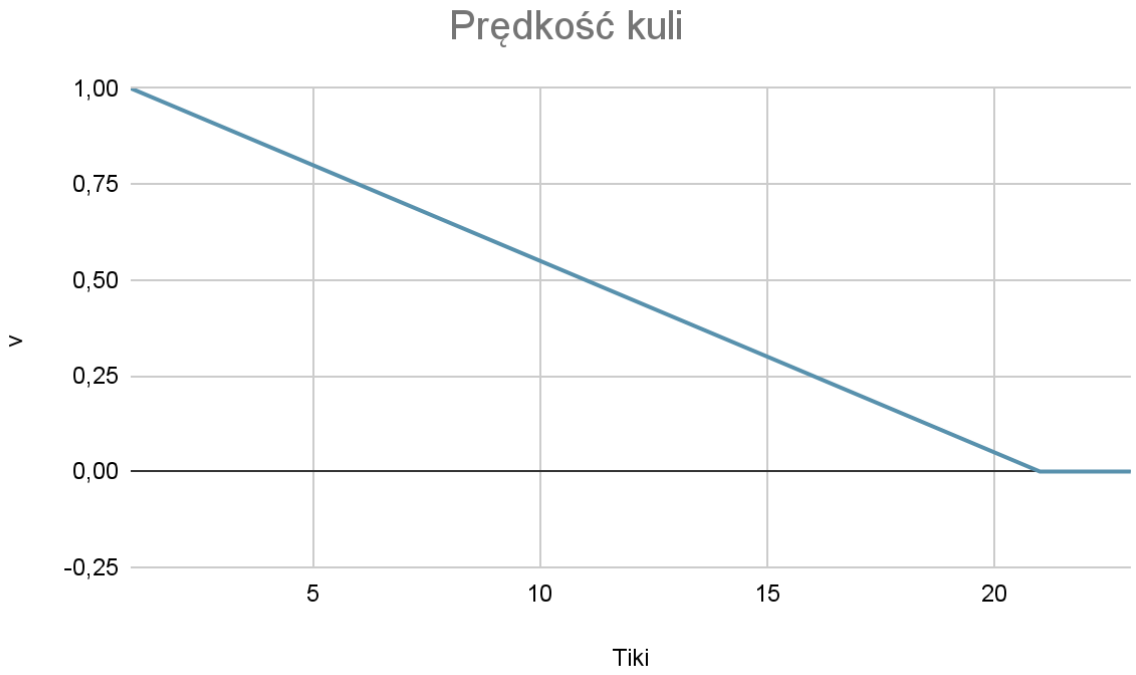
$$v'_{1x} = \frac{v_1 \cos(\theta_1 - \varphi)(m_1 - m_2) + 2m_2 v_2 \cos(\theta_2 - \varphi)}{m_1 + m_2} \cos(\varphi) + v_1 \sin(\theta_1 - \varphi) \sin(\varphi)$$

$$v'_{1y} = \frac{v_1 \cos(\theta_1 - \varphi)(m_1 - m_2) + 2m_2 v_2 \cos(\theta_2 - \varphi)}{m_1 + m_2} \sin(\varphi) + v_1 \sin(\theta_1 - \varphi) \cos(\varphi)$$

Powyższe wzory pozwalają obliczyć składowe prędkości ciał po zderzeniach. Θ_1 i Θ_2 oznacza kąty pod jakimi poruszają się kulę, a φ oznacza kąt pod jakim zderzają się ciała.

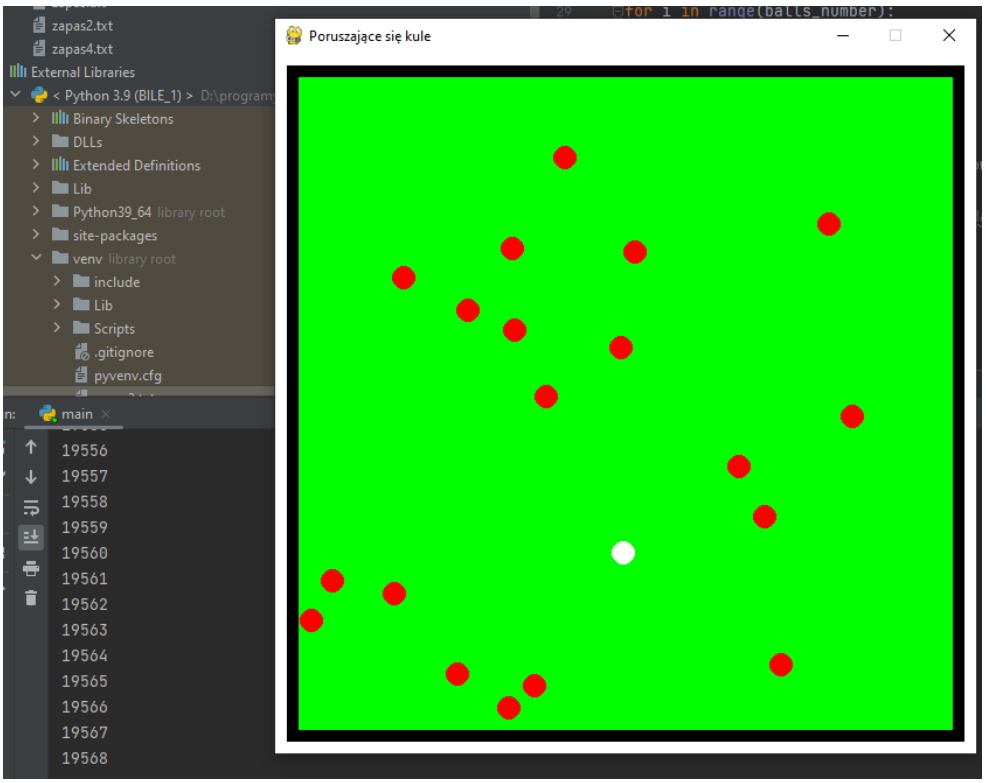


Kule zwalniają oprócz tego cały czas z jednostajnym przyśpieszeniem poruszając się ruchem jednostajnie opóźnionym.



3. Działania podjęte w celu wykonania zadania

W celu wytrenowania sieci neuronowej spełniającej założenia projektu w pierwszej kolejności należało stworzyć przestrzeń z bilami, w której będą one ze sobą oddziaływać i zaimplementowanie do niej wzorów fizycznych. Po zakończeniu implementacji środowiska zostaje do niego następnie wprowadzona pewna ilość bil i cały układ wprawiony jest w ruch. Podczas każdego zderzenia do pliku .csv wpisywane są dane które umożliwią trenowanie sieci neuronowej. Są to: współżędne x i y znormalizowanego wektora łączącego środki obu kulek, składowe prędkości x i y obu kul przed zderzeniem, oraz po zderzeniu (obliczone przy użyciu wzorów fizycznych na zderzenia całkowicie sprężyste z punktu 2.1).



Proces generowania danych

W taki sposób przeprowadzonych zostało 5 sesji generowania danych, dzięki czemu uzyskane zostały informacje dotyczące ok.100 000 zderzeń(później wygenerowano dodatkowe dane sprowadzając tę liczbę do ponad 1 miliona zderzeń).

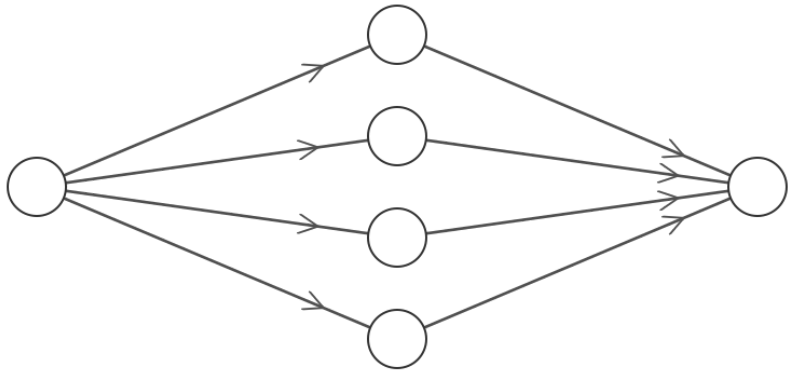
Sieć podczas uczenia otrzymywała jako dane wejściowe wcześniej wygenerowane dane dotyczące wektora między ciałami i ich prędkości, a następnie otrzymany przez siebie wynik porównywała z oczekiwanym, wyliczonym na podstawie wzorów.

4. Architektura Sieci

Obie sieci zostały zaimplementowane przy wykorzystaniu narzędzi biblioteki PyTorch. Pierwsza składa się z pięciu warstw neuronów – wejściowej posiadającej 6 neuronów; trzech warstw ukrytych po 64 neurony i warstwy wyjściowej w postaci 4 neuronów odpowiadających odpowiednio składowym x,y wektorów prędkości obu cząsteczek po zderzeniu. Funkcje aktywacji użyte w sieci to Tanh dla pierwszej warstwie ukrytej i ReLU na pozostałych dwóch. Funkcja straty, użyta do oceny jakości modelu to MSE(mean square error), ponieważ najlepiej nadaje się do oceny

wyników regresji. Z kolei algorytm użyty do korekty wag podczas uczenia to Adam, który jest rozwinięciem algorytmu gradientowego. Sieć była uczona na ok.100 000 linii danych(treningowe + walidacyjne), później zwiększonych do 1 000 000 linii. Wybór funkcji aktywacji jak i liczby neuronów dyktowane były przez rezultaty metody prób i błędów dążąc do zminimalizowania błędu przy jednoczesnym utrzymaniu wydajności symulacji. Ostateczna wersja sieci użytej do obsługi zderzeń jest w stanie stworzyć skuteczną iluzję odwzorowania zderzenia sprężystego przy jednoczesnym zachowaniu płynności całego programu, dzięki swojemu niewielkiemu rozmiarowi. Po ok. 100 epokach sieć nie wykazywała dalszej poprawy w dokładności mimo kontynuowania trenowania na mniejszych wartościach tempa nauczania i większej ilości danych.

Druga sieć jest bardzo prosta posiada 1 ukrytą warstwę z funkcją aktywacji ReLU z 4 neuronami i po jednym neuronie na wejście i wyjście. Do treningu zostały użyte te same funkcje straty i korekty wag co w pierwszej sieci. Nie było potrzeby normalizacji danych dla tak prostego przypadku. Błąd sieci jest tak niewielki, że można go pominąć(10^-12). Diagram sieci poniżej.



Wybór funkcji aktywacji dla pierwszej sieci dyktowany był w przypadku ReLU jej dopasowaniem do potrzeb z kolei Tanh na pierwszej warstwie został wybrany na podstawie prób i błędów podobnie jak liczba neuronów.

Wybór funkcji korekty wag był dyktowany jej wydajnością.

5. Próbowane metody

- Użycie Leaky ReLU - brak zauważalnej poprawy precyzji sieci neuronowej

- Normalizacja danych dla pierwszej sieci - nawarstwiające się błędy aproksymacji spowodowały, że sieć nauczona na znormalizowanych danych miała mniejszą precyzje obliczeń, niż ta nauczona na nie znormalizowanych danych.
- Znaczne zwiększenie sieci - zwiększenie precyzji obliczeń było zbyt małe aby uzasadnić wybór tego rozwiązania, ponieważ było zrównoważone przez znaczny wzrost kosztu obliczeń, co skutkowało spowolnieniem symulacji.
- Użycie samych ReLU jako funkcji aktywacji - brak poprawy dokładności obliczeń