ALGORYTMY INSPIROWANE BIOLOGICZNIE
REKURENCYJNA SIEĆ NEURONOWA TRENOWANA ALGORYTMEM GENETYCZNYM
DOKUMENTACJA

Piotr Zborowski, Piotr Klęp

Kraków 2024

Spis treści

Rozdział 1

Gra 2048

1.1 Zasada działania

Gra odbywa się na siatce 4x4, która na początku jest prawie pusta, zawiera jedynie dwie płytki o wartości 2 lub 4. Gracz przesuwa płytki w jednym z czterech kierunków: w górę, w dół, w lewo lub w prawo. Po każdym ruchu nowa płytka o wartości 2 lub 4 pojawia się na planszy w losowo wybranej pustej komórce.

1.2 Zasady gry

1. **Ruch płytek**: Wszystkie płytki na planszy przesuwają się w wybranym kierunku, o ile jest dla nich miejsce. Płytki przesuwają się tak daleko, jak to możliwe, zanim napotkają inną płytkę lub krawędź planszy.

2. **Łączenie płytek**: Jeśli dwie płytki o tej samej wartości zderzą się podczas ruchu, łączą się w jedną płytkę o wartości równej sumie wartości tych dwóch płytek (np. dwie płytki o wartości 2 tworzą jedną o wartości 4).

3. **Połączenia**: Po każdym ruchu, nowe połączenia są możliwe tylko raz na ruch, czyli dwie płytki o tej samej wartości mogą się połączyć tylko raz, po czym tworzą nową płytkę i pozostają w nowej wartości.

4. **Nowe płytki**: Po każdym ruchu, na planszy pojawia się nowa płytka o wartości 2 lub 4 w losowym pustym miejscu.

1.3 Cel gry

Celem gry jest osiągnięcie tytułowej płytki o wartości 2048 w jak najmniejszej liczbie kroków. Trzeba to oczywiście osiągnąć poprzez łączenie ”mniejszych”płytek.

Osiągnięcie płytki 2048 nie oznacza jednak końca gry. Gracz może kontynuować grę.

Jeżeli na planszy zabraknie miejsca na nowe płytki jak i nie będzie możliwości połączenia dwóch takich samych płytek w jedną, to oznacza to koniec gry.

Rysunek 1.1: Plansza gry z płytkami o różnych wartościach

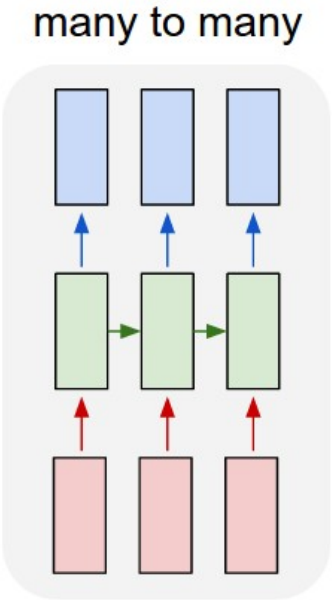
Rozdział 2

Architektura sieci

W naszej sieci neuronowej na wejściu otrzymujemy 16 sygnałów oznaczających stan planszy (zlogarytmizowany), a na wyjściu 4 sygnały, z których największy oznacza określony ruch. Warstwy ukryte posiadają kolejno 32, 64, 64, 16 pola.

Pomiędzy warstwami występują wagi $weights_W$ łączące neurony oraz $weights_B$ służące jako progi.

Oprócz podstawowej sieci neuronowej testujemy także dodanie do niej pamięci - mamy wtedy do czynienia z rekurencyjną siecią neuronową, stosujemy typ many to many. Wagi pamięci oznaczamy jako $weights_V$. Łącznie sieć posiada więc 17364 wag i waży około 136 kB, a w wariancie bez pamięci 7620 wag i 58 kB.

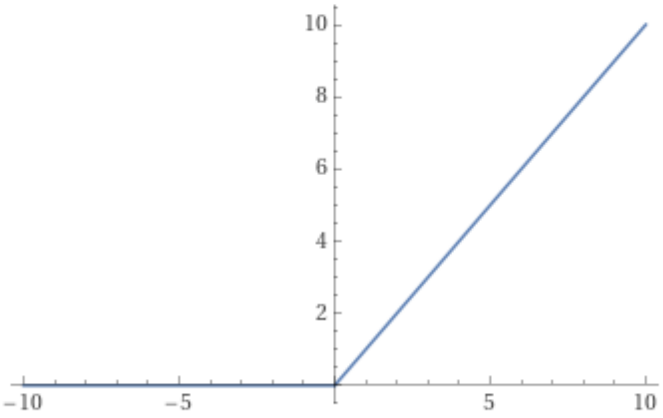


Rysunek 2.1: Architektura RNN many to many. Źródło

2.1 Funkcje aktywacji

2.1.1 ReLU

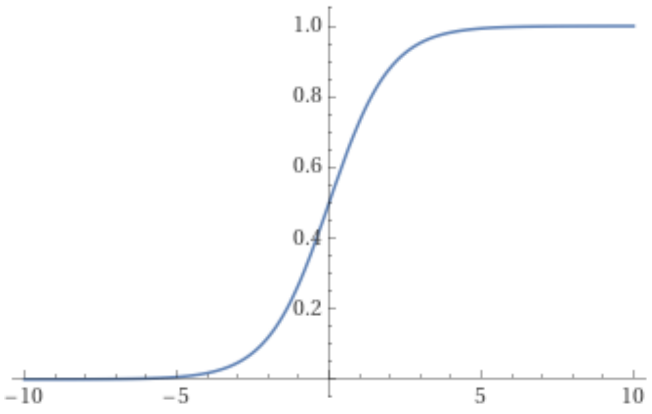
$$ReLU(x) = max(0, x) \tag{2.1}$$



Rysunek 2.2: Wykres funkcji $ReLU$.

2.1.2 sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$



Rysunek 2.3: Wykres funkcji σ.

Rozdział 3

Algorytm genetyczny

3.1 Inicjalizacja sieci

Na początek tworzymy populację 80 sieci z losowymi wagami według rozkładu naturalnego standardowego

$$w = \mathcal{N}(0, 1) \tag{3.1}$$

3.2 Ewaluacja

W ramach ewaluacji każda sieć gra 5-krotnie w grę i średni wynik to przypisana jej wartość. Następnie szeregujemy sieci zgodnie z ich wynikami i zostawiamy 10 najlepszych.

3.3 Crossover

Spośród 10 najlepszych sieci wybieramy za każdym razem losowo dwie, z których generujemy potomka. Potomek każdą wagę przyjmuje z równym prawdopodobieństwem albo od pierwszego rodzica, albo drugiego.

3.4 Mutacje

Następnie potomek poddawany jest mutacji - iterujemy po wszystkich jego wagach i każda może z prawdopodobieństwem 1/1000 być zainicjalizowana na nowo zgodnie z równaniem 3.1.

3.5 Nowa „samorodna” sieć

W każdej iteracji generując nowych potomków z prawdopodobieństwem 1/1000 może on nie dziedziczyć wag po rodzicach, ale dostać je zupełnie losowe, tak jak populacja pierwotna - ułatwia to ewentualne wydostanie się z lokalnego optimum.

3.6 Kolejna iteracja

Po wygenerowaniu nowego pokolenia wracamy do kroku ”ewaluacja”, chyba że osiągnęliśmy już liczbę pokoleń dla której przeprowadzamy eksperyment.

Rozdział 4

Wyniki

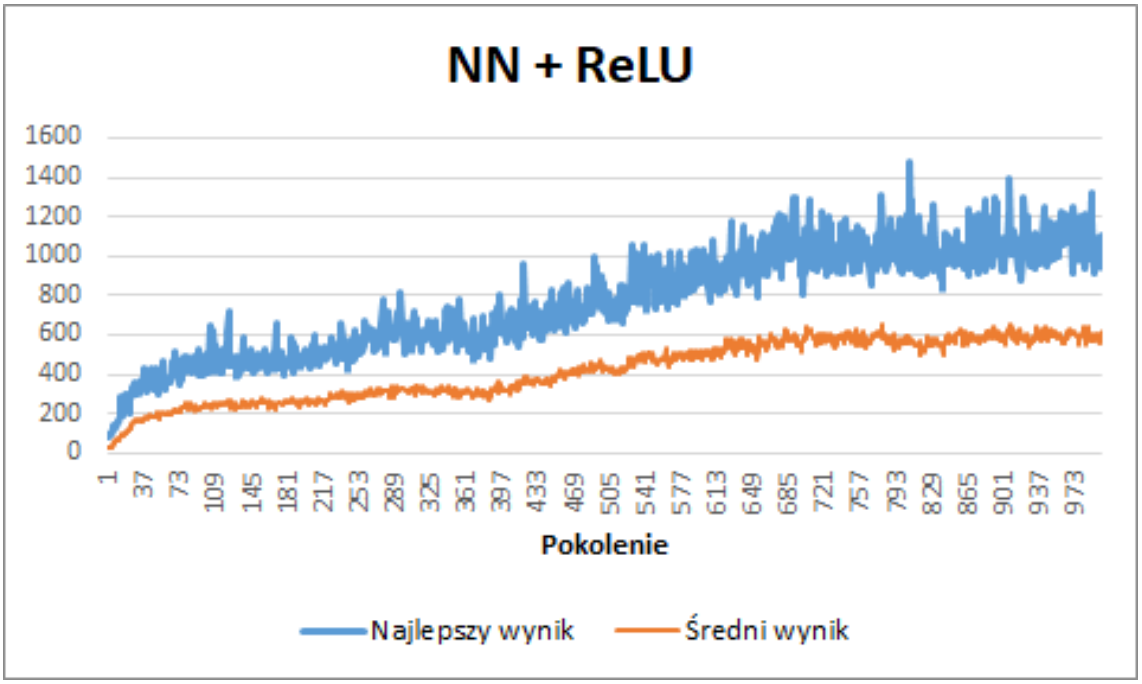
4.1 Informacje sprzętowe i techniczne

- **Język programowania**: Java
- **Biblioteka do mnożenia macierzy**: la4j

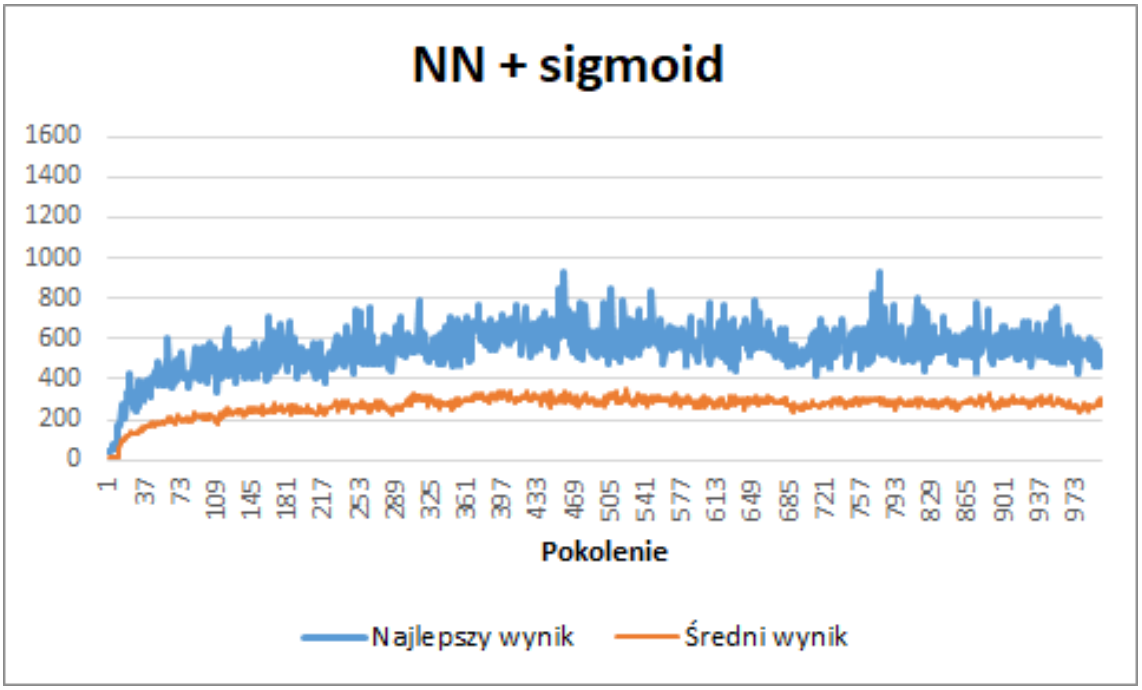
Parametry komputerów na jakich testowana była sieć neuronowa:

1. - **System operacyjny**: Windows 11 Education
   - **CPU**: AMD Ryzen 5 7530U
   - **RAM**: 16GB
2. - **System operacyjny**: Windows 10
   - **CPU**: Intel Core i5-8265U
   - **RAM**: 16GB

4.2 Sieć bez pamięci

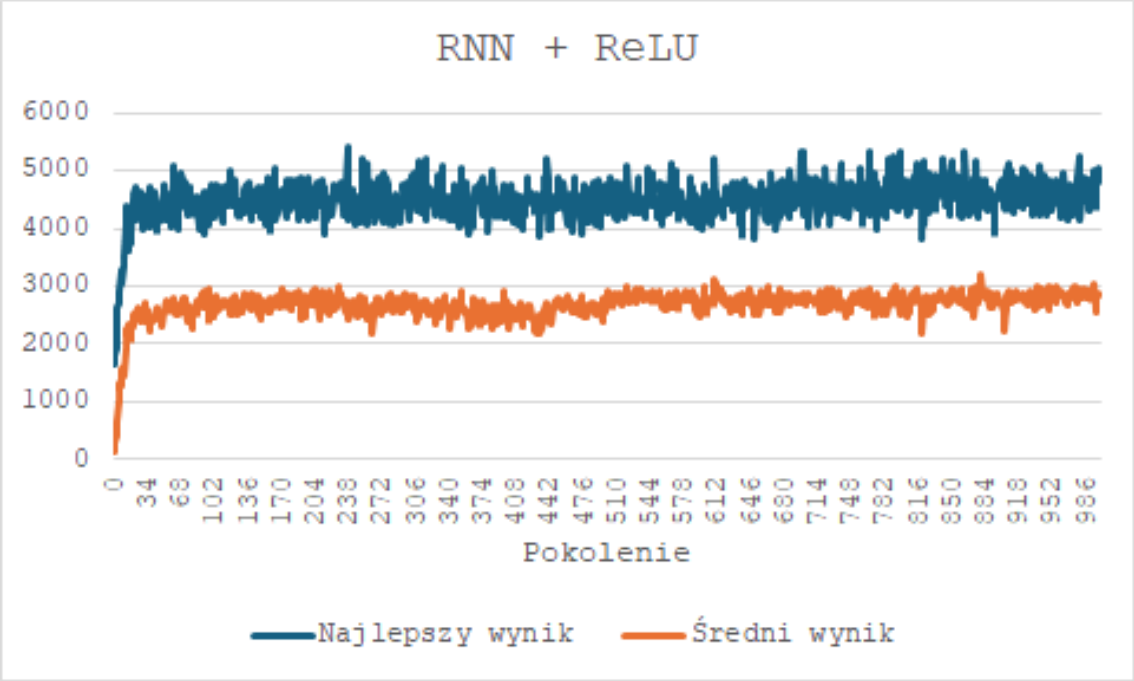


Rysunek 4.1

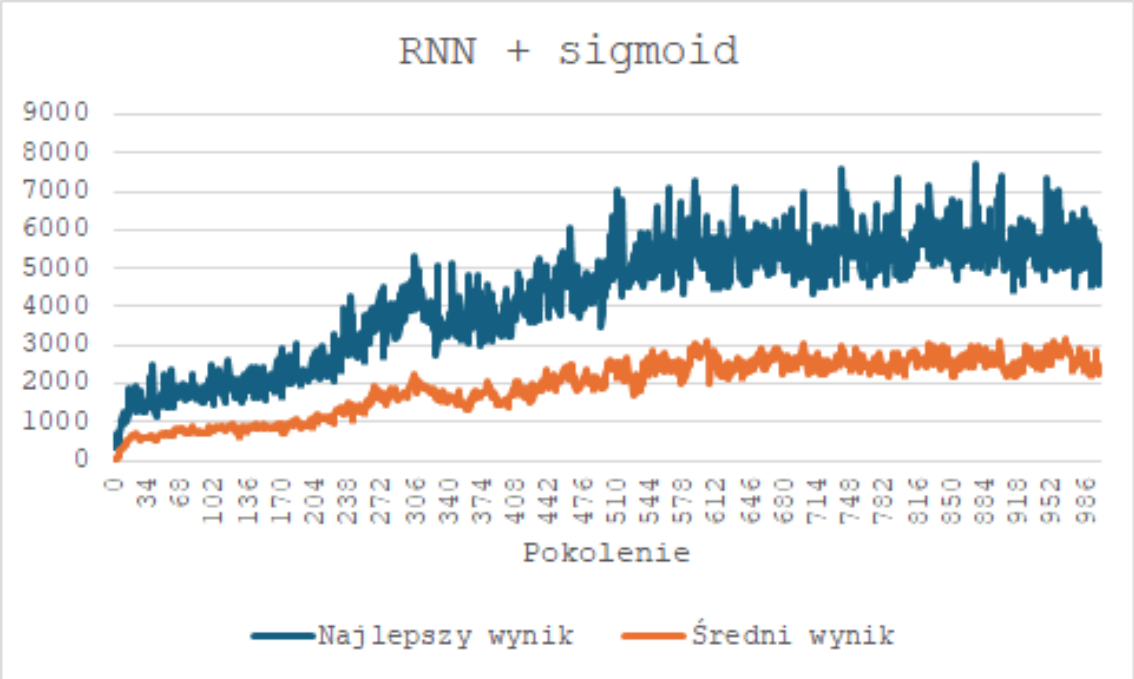


Rysunek 4.2

4.3 Sieć z pamięcią



Rysunek 4.3



Rysunek 4.4

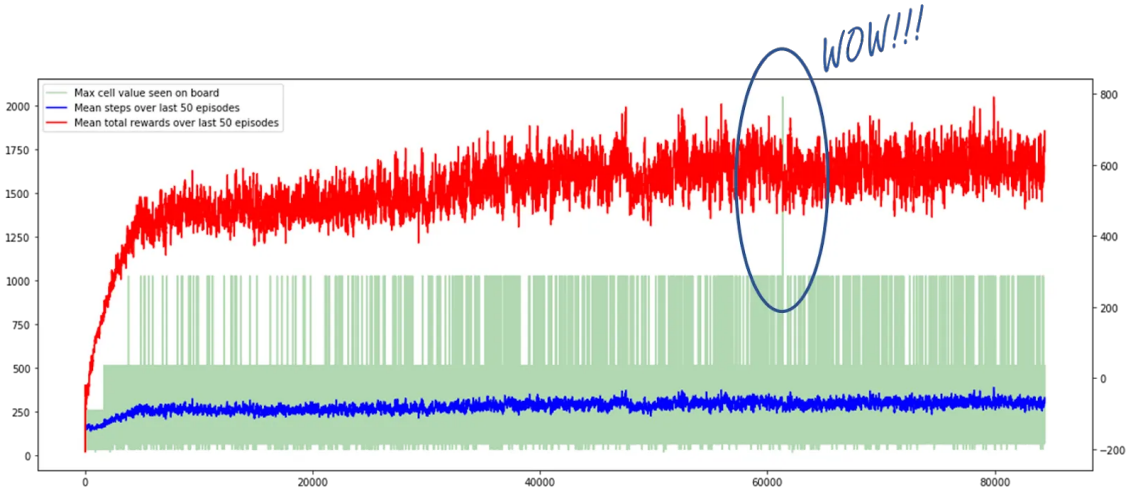
4.4 Najlepszy wynik

Najlepsza pojedyncza rozgrywka najlepszej wytrenowanej sieci posiadała największy kafelek 1024 oraz wynik 16424.

Rozdział 5

Porównanie z podobnym projektem

Autorowi podobnego projektu udało się raz osiągnąć płytkę 2048. Poprzedzone to było jednak bardzo wieloma testami idących w kilkadziesiąt tysięcy. Zastosował on uczenie przez wzmacnianie (Reinforcement learning) będącą jedną z metod stosowanych w uczeniu maszynowym, a jego sieć miała miliony wag.



Rysunek 5.1: Wyniki testów wykonanych przez autora projektu. Poprzez „WOW“ został zaznaczony najlepszy wynik (kafelek 2048), źródło: link

Artykuł opisujący ten projekt: link

Rozdział 6

Wnioski

Widzimy że wyniki okazały się lepsze przy użyciu pamięci, mimo że teoretycznie nie jest ona potrzebna i cała informacja o grze zawiera się w jej aktualnym stanie.

W dłuższej perspektywie funkcja aktywacji sigmoid okazała się lepsza niż ReLU, która bardzo szybko dochodzi do wysokich wartości, ale prawdopodobnie utyka w lokalnym optimum.

Dobrym dla wyników okazało się także zlogarytmizowanie wejścia (czyli np. 2 na polu reprezentujemy jako 1, 8 jako 3, 1024 jako 10 itd.), jest to naturalniejszy stan dla sieci neuronowych.

Rozdział 7

Co można usprawnić?

Jeszcze naturalniejszym niż logarytmiczne wartości na wejściu byłoby dla każdego pola na wejściu podawać vektor $\boldsymbol{v}$, taki że wszystkie jego wartości wynoszą 0, oprócz jednej, która wynosi 1, a jej indeks oznacza potęgę dwójki która znajduje się na polu, na przykład 4 byłoby reprezentowane jako $[0, 1, 0, 0, \dots]$. Zwiększyłoby to także rozmiar pierwszej macierzy przez zwiększenie pierwszej warstwy.

Nie jest natomiast dobrym pomysłem zastąpienie algorytmu genetycznego ewolucją strategiczną, mimo że ta pozornie wydaje się bardziej sensowna do niedyskretnych wag - w fazie testów prowadziła ona do malejących wyników i mimo że teoretycznie wyszłaby z lokalnego optimum, w praktyce jest to zbyt niewydajne.