

Jakub Mielczarek 203943 203943@edu.p.lodz.pl
Łukasz Gołębiowski 203882 203882@edu.p.lodz.pl

Zadanie 1.: Samoorganizujące mapy neuronowe z wykorzystaniem algorytmów: Kohonena, gazu neuronowego, k-średnich

1. Cel

Przedmiotem zadania jest implementacja algorytmów służących do przetwarzania zbiorów danych i ich klasyfikacji (grupowania) na mniejsze podzbiory według ściśle określonych zasad.

2. Wprowadzenie

Zadanie zostało podzielone na trzy warianty i w ten sposób zaprezentujemy teorię niezbędną do jego realizacji.

Wariant I: Kwantyzacja przestrzeni za pomocą samoorganizującej się sieci neuronowej przy pomocy algorytmu Kohonena oraz algorytmu gazu neuronowego

Mamy zbiór danych opisujących punkty dla przestrzeni o dowolnej liczbie wymiarów. Pierwszym krokiem algorytmu Kohonena jest wygenerowanie neuronów, względem których będziemy dokonywać klasyfikacji. Neurony generujemy losowo wykorzystując rozkład Gaussa parametryzowany poprzez obliczenie średniej arytmetycznej oraz odchylenia standardowego dla punktów i -tego wymiaru. Neurony są losowane kilka razy wybierając tą strukturę która ma najmniej martwych neuronów. Średnia arytmetyczna:

$$\sqrt{\frac{\sum_{i=1}^n x_i}{n}} \quad (1)$$

Odchylenie standardowe:

$$\sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}} \quad (2)$$

Odległość euklidesowa:

$$\sqrt{\frac{\sum_{i=1}^n (x_{iA} - x_{iB})^2}{n}} \quad (3)$$

Kolejnym etapem jest przyporządkowanie do każdego punktu najbliższego mu neuronu(3). Wynik tej operacji jest zapisywany jako lista indeksów odpowiadających indeksom w liście neuronów. Następnie przechodzimy do najważniejszej części algorytmu. Wybieramy losowy punkt oraz przyporządkowany mu neuron. Neuron ten od tego momentu będziemy określać jako "zwycięzcę" (BMU). Następnie obliczamy wagi (nowe współrzędne) wszystkich neuronów korzystając z następujących wzorów:

$$learningRate = learningRate_0 * \exp\left(\frac{-i}{numberOfIterations}\right) \quad (4)$$

$$timeConst = \frac{numberOfIterations}{(mapRadius_0 * timeConst_0)} \quad (5)$$

$$mapRadius = mapRadius_0 * \exp\left(\frac{-i}{timeConst}\right) \quad (6)$$

$$influence = \exp\left(\frac{-(distFromBMU)^2}{2 * (mapRadius)^2}\right) \quad (7)$$

$$w_{ij}(new) = w_{ij}(old) + learningRate * influence * (x_j - w_{ij}(old)) \quad (8)$$

Podany powyżej wzór (7) wykorzystuje gaussowską funkcję sąsiedztwa. Należy zauważyć, że również neurony-sąsiedzi podlegają modyfikacji, jednakże w słabszym stopniu. Algorytm jest powtarzany do osiągnięcia kryterium zbieżności, którym w tym przypadku jest liczba iteracji. Algorytm gazu neuronowego działa analogicznie jak algorytm Kohonena, przy czym zagadnienie sąsiedztwa rozwiązane jest przez uporządkowanie neuronów w szereg w zależności od odległości ich wektorów wagowych od podanego wektora wejściowego. Współczynnik nauki wyznaczany jest w tym przypadku na podstawie pozycji w szeregu, a nie faktycznej odległości (pozycja w szeregu jest ustalana poprzez odległość od BMU w kolejności rosnącej):

$$influence = \exp\left(\frac{-(i)^2}{2 * (mapRadius)^2}\right) \quad (9)$$

Wariant II: Kwantyzacja przestrzeni za pomocą samoorganizującej się sieci neuronowej przy pomocy algorytmu k-średnich

Mamy zbiór danych opisujących punkty dla przestrzeni o dowolnej liczbie wymiarów. Pierwszym krokiem algorytmu jest wygenerowanie centroidów (klas), względem których będziemy dokonywać klasyfikacji. Centroidy generujemy losowo wykorzystując rozkład Gaussa parametryzowany poprzez obliczenie średniej arytmetycznej(1) oraz odchylenia standardowego(2) dla

punktów i -tego wymiaru. Neurony są losowane kilka razy wybierając tą strukturę która ma najmniej martwych neuronów. Kolejnym etapem jest obliczenie odległości punktów od centroidów i przypisanie ich do najbliższego k -centroidu(3). Ostatnim krokiem jest obliczenie średniej arytmetycznej(1) punktów należących do danego klastra. Jej wartość to nowe współrzędne k -centroidu. Algorytm jest powtarzany aż do osiągnięcia warunku kończącego. W naszej implementacji jest to określona z góry liczba iteracji lub otrzymanie tych samych współrzędnych dla centroidów w $i+1$ kroku iteracji.

Wariant III: Kompresja obrazu

Aby skompresować obraz za pomocą w/w algorytmów najpierw przygotowujemy dane RGB pikseli z obrazu do pliku tekstowego. Obrazek jest dzielony na kwadratowe ramki o boku równym ilości pikseli zadanej przez użytkownika. Następnie każda ramka przekształcana jest do pliku tekstowego w taki sposób, że reprezentuje jedną daną. Dla przykładu dla kolorowego obrazka dla ramki o boku 3 pikseli jedna ramka będzie reprezentowana jako punkt w przestrzeni 27-wymiarowej - każdy piksel ma jeszcze 3 składowe RGB. Po zakończeniu działania wybranego algorytmu należy jeszcze zastąpić wartości każdej ranki przyporządkowanym jej neuronem. Na sam koniec przeprowadza się operację zapisania przekształconych ramek do obrazka o wybranym formacie.

3. Opis implementacji

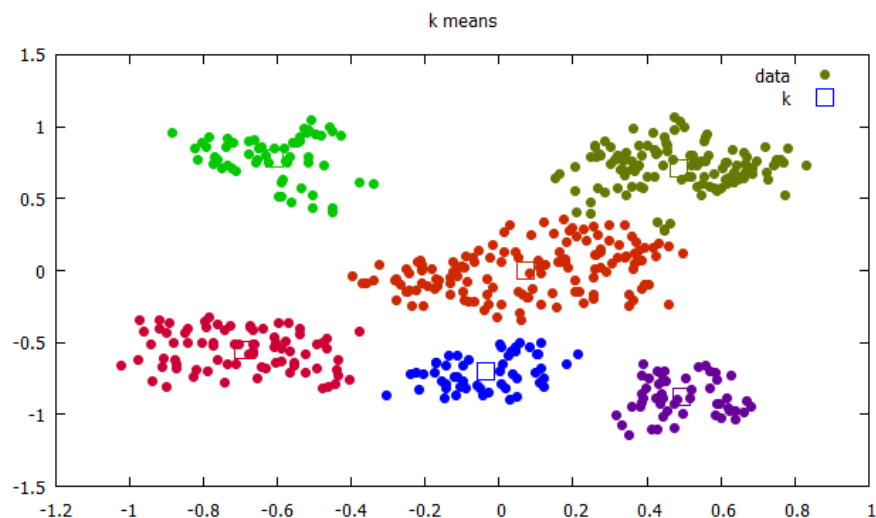
- Podstawowy opis plików klas:
- Neural.java – klasa abstrakcyjna zawierające wspólne pola i metody dla każdego z algorytmów.
 - Kohonen.java – implementacja algorytmu SOM Kohonena.
 - NeuralGas.java – implementacja algorytmu gazu neuronowego.
 - KMeans.java – implementacja algorytmu k -średnich.
 - FileHandler.java – klasa zawierająca statyczne metody do działań związanych z operacjami na plikach np. zapis macierzy danych do pliku w odpowiednim formacie tabulacji i znaków nowej linii. Odczyt danych z obrazu, zamiana pikseli z obrazka na ramki i zapis ich do pliku itp.
 - Utils.java – klasa pomocnicza zawierająca statyczne metody do operacji takich jak normalizacja danych, pobranie wybranej kolumny z podanej struktury danych w postaci macierzy itp.
 - DataMath.java – klasa pomocnicza zawierająca statyczne metody do obliczania średnich, median, odchyleń standardowych itp.
 - Metric.java – klasa pomocnicza zawierająca statyczne metody do obliczania odległości pomiędzy punktami n -wymiarowymi w wybranej metryce.
- Program, który wykorzystaliśmy do rysowania wykresów to Gnuplot. Odpowiednie pliki zawierające skrypty są wywoływane w trakcie działania programu co pozwala na odczyt danych w czasie rzeczywistym. Warto zauważyć, że przygotowane klasy posiadają dość dużą ilość parametrów. Parametry pozwalają na:

- ustalenie wszystkich początkowych wartości parametrów dla każdego algorytmu.
- podanie ścieżki do pliku z danymi
- ustalenie czy chcemy aby program generował wykresy. Jeśli nie, oszczędzamy wówczas na dość kosztownych pod względem czasowym operacji na plikach podczas iteracji algorytmu (program nie musi przygotowywać plików dla gnuplota).

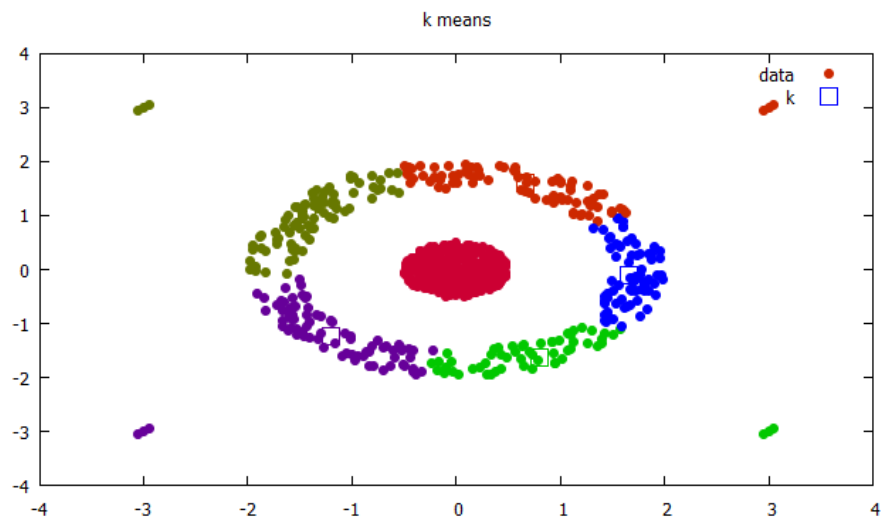
4. Materiały i metody

Dla każdego algorytmu podawana jest ilość iteracji, ilość centroidów(neuronów), plik źródłowy z danymi oraz dodatkowo dla algorytmów kohonena oraz gazu neuronowego wartości odpowiednich współczynników wpływających na przebieg oraz rezultat obliczeń. Dyskusji wyników dokonamy sprawdzając zachowania algorytmów dla określonych danych wejściowych i porównując je ze sobą. Przedstawimy również skrajne przypadki, które wpływają na nieefektywność zastosowanych rozwiązań.

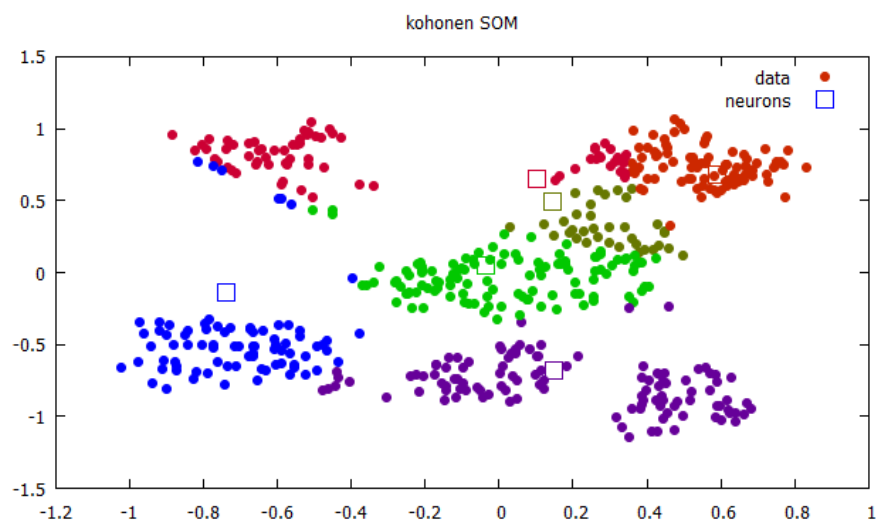
5. Wyniki



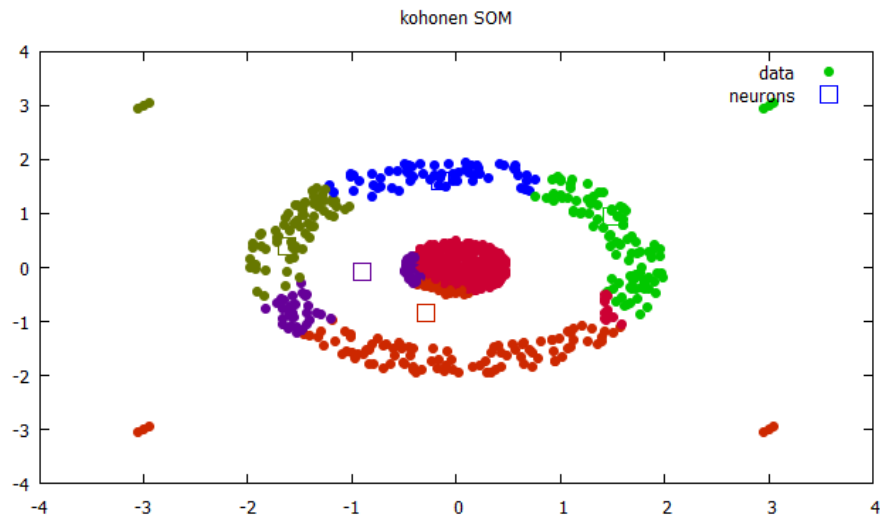
Rysunek 1: Data: sample1 Algorytm: k-średnich $n = 6$ iter = 10



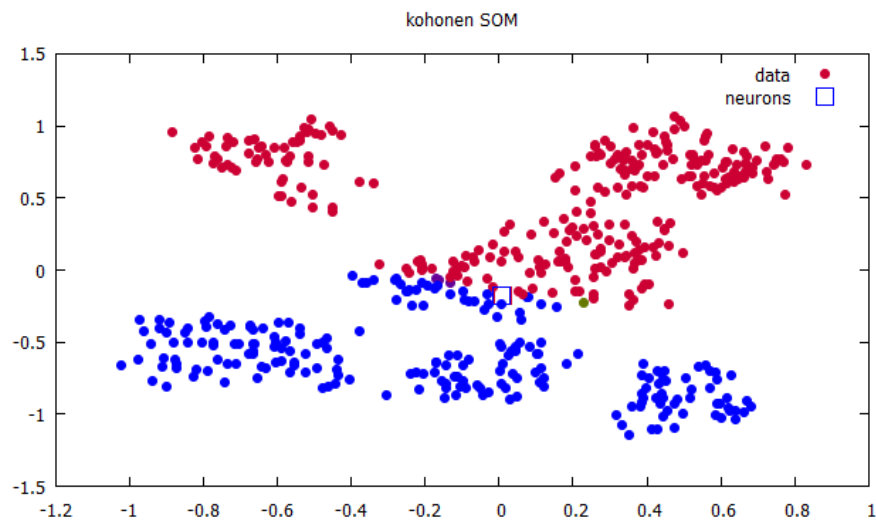
Rysunek 2: Data: circle1 Algorytm: k-średnich $n = 6$ iter = 10



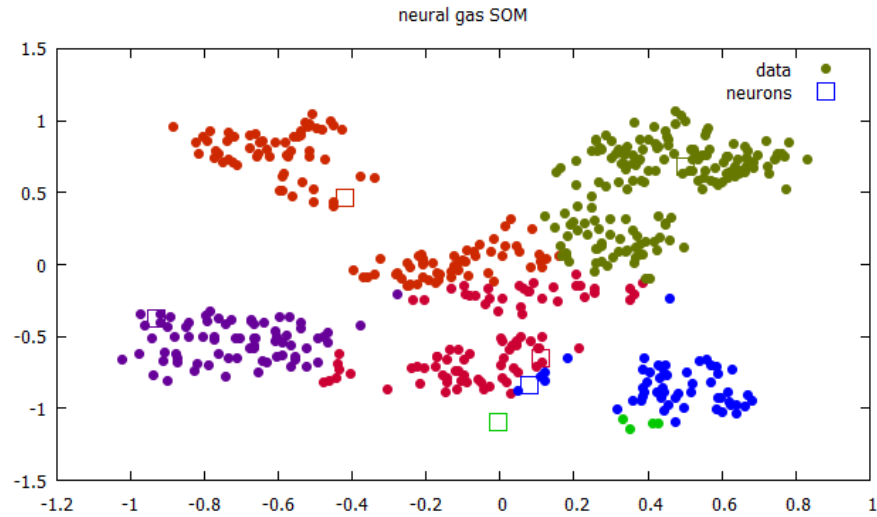
Rysunek 3: Data: sample1 Algorytm: kohonena $n = 6$ iter = 10 mapRadius = 0.1 learningRate = 0.7 timeConst = 1



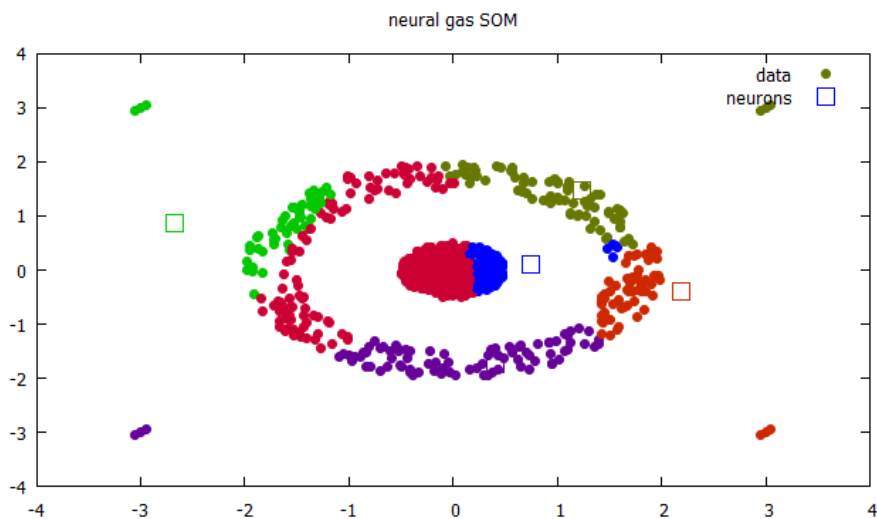
Rysunek 4: Data: circle1 Algorytm: kohonena $n = 6$ iter = 10 mapRadius = 0.1 learningRate = 0.7 timeConst = 1



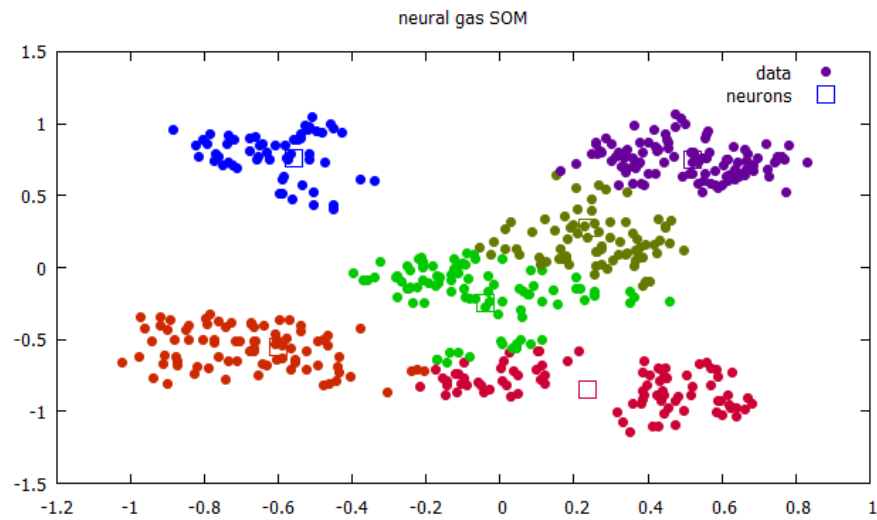
Rysunek 5: Data: sample1 Algorytm: kohonena $n = 6$ iter = 1000 mapRadius = 1 learningRate = 0.1 timeConst = 1



Rysunek 6: Data: sample1 Algorytm: gazu neuronowego $n = 6$ iter = 10 mapRadius = 0.1 learningRate = 0.7 timeConst = 1



Rysunek 7: Data: circle1 Algorytm: gazu neuronowego $n = 6$ iter = 10 mapRadius = 0.1 learningRate = 0.7 timeConst = 1



Rysunek 8: Data: circle1 Algorytm: gazu neuronowego $n = 6$ iter = 1000
mapRadius = 1 learningRate = 0.1 timeConst = 1



Rysunek 9: Data: lena Algorytm: k-średnich frameSz = 4 $n = 10$ iter = 10



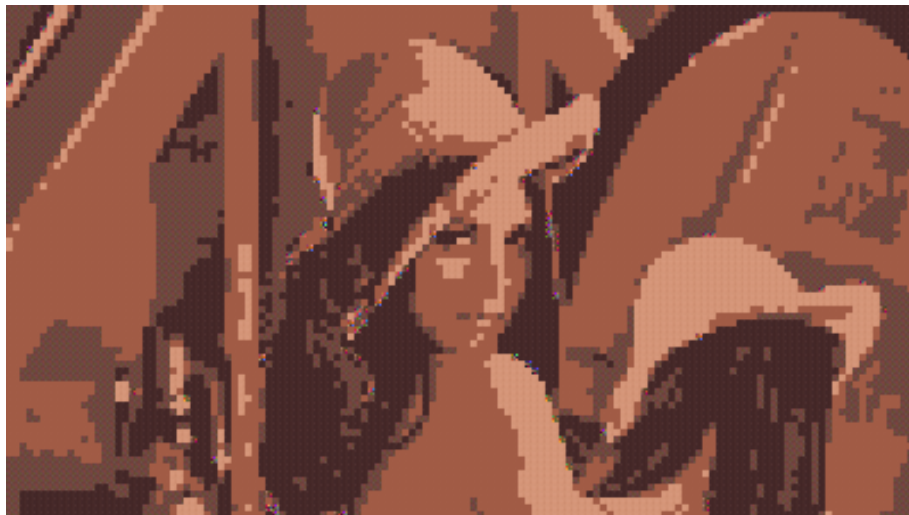
Rysunek 10: Data: lenakolor3 Algorytm: k-średnich $\text{frameSz} = 3$ $n = 20$ iter
= 10



Rysunek 11: Data: lenakolor3 Algorytm: k-średnich $\text{frameSz} = 3$ $n = 2$ iter
= 30



Rysunek 12: Data: lenakolor3 Algorytm: kohonena frameSz = 3 n = 20 iter = 10000 mapRadius = 100 learningRate = 0.1 lrc = 0.1



Rysunek 13: Data: lenakolor3 Algorytm: kohonena frameSz = 3 n = 20 iter = 1000 mapRadius = 0.01 learningRate = 0.1 lrc = 1



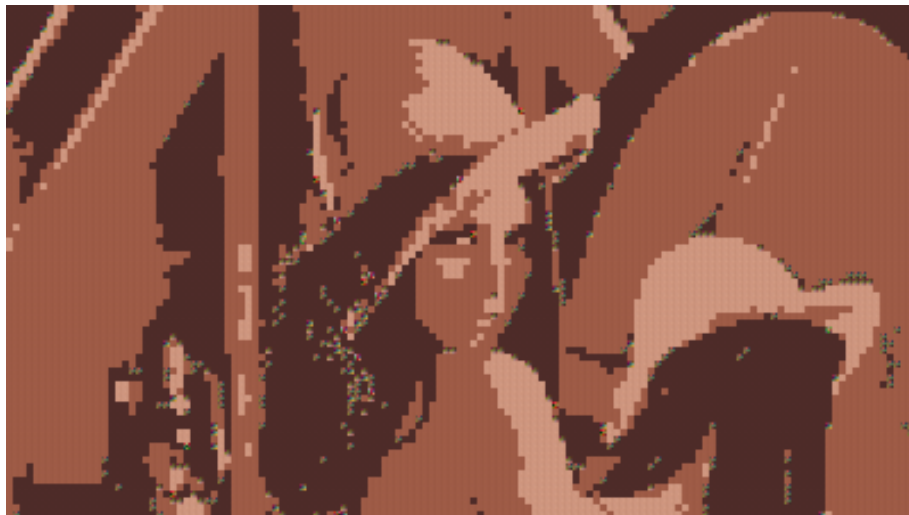
Rysunek 14: Data: lenakolor3 Algorytm: kohonena frameSz = 3 n = 20 iter = 10000 mapRadius = 300 learningRate = 0.1 lrc = 0.01



Rysunek 15: Data: lenakolor3 Algorytm: kohonena frameSz = 3 n = 20 iter = 1000 mapRadius = 100 learningRate = 0.1 lrc = 1



Rysunek 16: Data: lenakolor3 Algorytm: kohonena frameSz = 3 n = 20 iter = 1000 mapRadius = 300 learningRate = 0.1 lrc = 0.01



Rysunek 17: Data: lenakolor3 Algorytm: kohonena frameSz = 3 n = 20 iter = 1000 mapRadius = 0.01 learningRate = 0.1 lrc = 1



Rysunek 18: Data: lenakolor3 Algorytm: kohonena frameSz = 3 n = 20 iter = 1000 mapRadius = 100 learningRate = 1.5 lrc = 1

6. Dyskusja

Sekcja ta powinna zawierać dokładną interpretację uzyskanych wyników eksperymentów wraz ze szczegółowymi wnioskami z nich płynącymi. Najcenniejsze są, rzecz jasna, wnioski o charakterze uniwersalnym, które mogą być istotne przy innych, podobnych zadaniach. Należy również omówić i wyjaśnić wszystkie napotkane problemy (jeśli takie były). Każdy wniosek powinien mieć poparcie we wcześniej przeprowadzonych eksperymentach (odwołania do konkretnych wyników). Jest to jedna z najważniejszych sekcji tego sprawozdania, gdyż prezentuje poziom zrozumienia badanego problemu.

7. Wnioski

W tej, przedostatniej, sekcji należy zamieścić podsumowanie najważniejszych wniosków z sekcji poprzedniej. Najlepiej jest je po prostu wypunktować. Znow, tak jak poprzednio, najistotniejsze są wnioski o charakterze uniwersalnym.

Literatura

- [1] T. Oetiker, H. Partl, I. Hyna, E. Schlegl. *Nie za krótkie wprowadzenie do systemu L^AT_EX2e*, 2007, dostępny online.

Na końcu należy obowiązkowo podać cytowaną w sprawozdaniu literaturę, z której grupa korzystała w trakcie prac nad zadaniem.