Metody Inteligencji Obliczeniowej

Dokumentacja projektu

Wizualizacja procesu uczenia SSN

Kamil Sudoł
Jakub Strugała
Patryk Śledź

Informatyka Stosowana
Wydział Fizyki i Informatyki Stosowanej
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

1. Opis zadania

Celem projektu było przygotowanie prostej sieci neuronowej, która realizowałaby dowolne zadanie ze zbioru benchmarkowego UCI MLR: (http://archive.ics.uci.edu/ml/datasets.php).
Następnie należało zwizualizować zmianę poszczególnych wag i biasów w trakcie procesu uczenia. Co więcej, projekt powinien działać dla dowolnej sieci typu wielowarstwowego oraz dowolnej metody uczącej.

2. Wykorzystana technologia

W celu rozwiązania zadania skorzystano ze środowiska Matlab w wersji R2020b. Jako przykład zadania testowego wykorzystano zbiór danych dostępny pod adresem: https://archive.ics.uci.edu/ml/datasets/iris. Jest to dataset klasyfikujący Irysy.

Wszelkie operacje związane z obsługą sieci neuronowej zostały wykonane przy pomocy toolbox'a *Deep Learning Toolbox*.

Do wyznaczenia błędu średniokwadratowego pomiędzy danymi oczekiwanymi, a otrzymanymi z wyuczonej sieci, wykorzystano funkcję ***measerr()*** z biblioteki *Wavelet Toolbox.*

W celu przejrzystego wyświetlania generowanych wykresów skorzystano z interaktywnych dokumentów Matlab Live Scripts.

3. Opis implementacji

Kod programu jest podzielony na funkcje, gdzie każda z nich pełni ściśle określoną rolę.
Wyróżniamy następujące funkcje:

- SSNvisualisation(layers, epochs, plottype) - główna funkcja programu, w której następuje ładowanie, przekształcenie zbioru danych oraz uczenie sieci. Ponadto parametr *plottype* pełni rolę flagi decydującej, które z wykresów mają zostać wyświetlone. Istnieją następujące flagi:
    - *all* - wyświetlenie wszystkich danych,
    - *compare* - wyświetlenie porównania wizualizacji z projektu z funkcjami do wizualizacji wbudowanymi w środowisko Matlaba,
    - *animated* - wyświetlenie animowanego wykresu doboru wag neuronu,

- get_uci_mlr_iris_dataset() - funkcja pobierająca zbiór danych Irysów ze strony https://archive.ics.uci.edu/ml/datasets/iris,

- plot_biases(data, title_when_not_last, title_when_last, figure_move_parameter, x) - funkcja pomocnicza, która generuje wykresy zmian biasów dla każdej warstwy,

- plot_first_weights(weights, figure_move_parameter, x, epochs) - funkcja pomocnicza, która generuje wykresy zmian wag neuronów na wejściu dla każdego neuronu,

- plot_layers(data, figure_move_parameter, x) - funkcja pomocnicza, która generuje wykresy zmian wag neuronów w warstwach ukrytych oraz na wyjściach,

- create_gif(filename, frame, counter) - funkcja tworzy plik .gif z klatek wygenerowanej figury. Wykorzystuje funkcje ***frame2im()*** do konwersji klatki filmu na obraz. Następnie korzysta z funkcji ***rgb2ind()*** do konwertowania do indeksowanego obrazu i tzw. colormapy, co w takiej formie zapisuje za pomocą funkcji ***imwrite()*** do pliku graficznego.

- plot_heatmap(weights, biases, layers, size_x, figure_move_parameter) - funkcja tworzy wizualizacje wag neuronów oraz biasów z każdej warstwy sieci w formie heatmap.

- plot_confmat(matrix, figure_move_parameter) - funkcja wizualizuje macierz błędów (confusion matrix), wykorzystuje funkcję ***imagesc***, wyświetlającą obraz podzielony na obszary, które są zdefiniowane przez rozmiar macierzy błędów. Następnie do obszarów wprowadzane są informacje z macierzy,

- plot_one_entire_neuron_weights(weights) - funkcja wyświetla animację zmiany wag połączeń w warstwie pierwszej dla pierwszego neuronu. Ponadto zapisuje każdą z klatek animacji, które w całości składają się na pliki .gif oraz .avi,

- plot_mse_performance(x, epochs, mseOut) - funkcja pokazuje jak zmienia się błąd średniokwadratowy na przestrzeni epok oraz wyświetla w tytule jego ostateczną wartość.

Przykładowa sieć klasyfikująca Irysy

Po wczytaniu zbioru danych Irysów ze strony internetowej https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data podzielono go na klasy, gdzie dla każdej klasy 90% danych przeznaczono do uczenia sieci, a 10% danych do testowania.

Następnie przystąpiono do konfiguracji sieci. Parametry definiujące ilość ukrytych warstw sieci wraz z ilością neuronów dla każdej warstwy jak również ilość epok są przekazywane jako argument wywołania funkcji **SSNvisualisation**.

Jako rodzaj funkcji aktywacji wybrano funkcję *logsig* wraz z *dividetrain* jako funkcją podziału zbioru uczącego.

Uczenie sieci przebiega iteracyjnie. W każdej iteracji zbierane są informacje na temat aktualnych wartości wag, biasów oraz aktualnego błędu średniokwadratowego. Dane te będą potrzebne do wykonania wizualizacji przebiegu uczenia.

Pobieranie parametrów sieci neuronowej i wyświetlanie wyników

Pobieranie parametrów sieci jest wykonywane w każdej epoce uczenia sieci.

Wykorzystano parametry: net.IW{1}, net.b, net.LW.

Uzyskane parametry są następnie przekazywane do funkcji rysujących wykresy:
- plot_mse_performance,
- plot_biases,
- plot_first_weights,
- plot_heatmap.

Korzystając z danych testowych, przeprowadzono sprawdzenie wytrenowanej sieci neuronowej. Przy pomocy funkcji ***confusion*** zwrócono macierz błędów, którą następnie przekazano do zaimplementowanej funkcji rysującej: ***plot_confmat()***.

Do stworzenia animacji - plików .gif oraz .avi wykorzystano funkcję plot_one_entire_neuron_weights(), do której przekazano argument net_layers() stworzony z macierzy wag zmieniających się z warstwy na warstwę dla poszczególnej epoki. Argument ten przy przekazaniu do funkcji został skonwertowany za pomocą metody ***cell2mat()***.

- Do wygenerowania filmu .avi wykorzystano obiekt ***VideoWriter****,* dla którego liczbę klatek na sekundę ustawiono domyślnie na 10, a klatki do obiektu zapisywano w każdej iteracji za pomocą funkcji writeVIdeo().

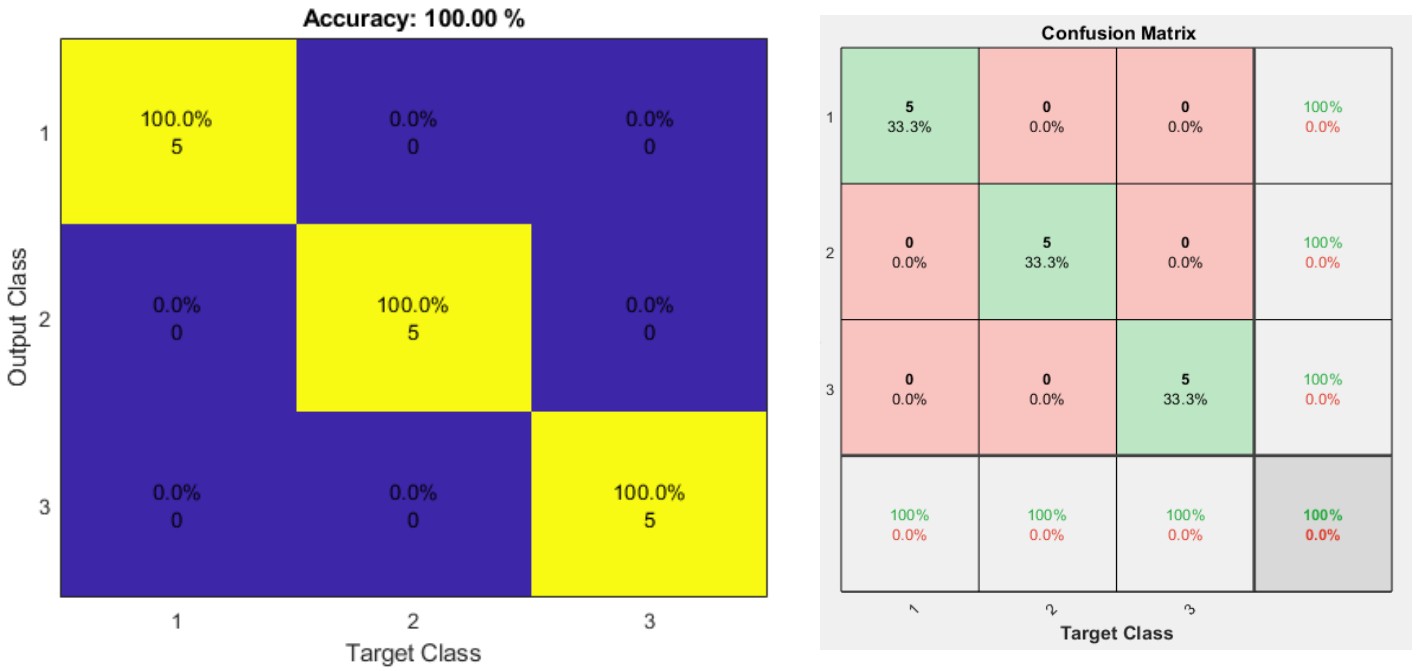
- Do wygenerowania pliku .gif wykorzystano funkcję ***create_gif()***. Jako argumenty przekazano nazwę pliku, klatkę oraz numer klatki, co było potrzebne z racji konieczności innego zapisywania za pomocą funkcji imwrite() dla klatki pierwszej (z parametrem ‘*Loopcount’ i inf*) i innego dla reszty klatek (z parametrem *‘WriteMode’ i ‘append’*).

4. Uzyskane wyniki

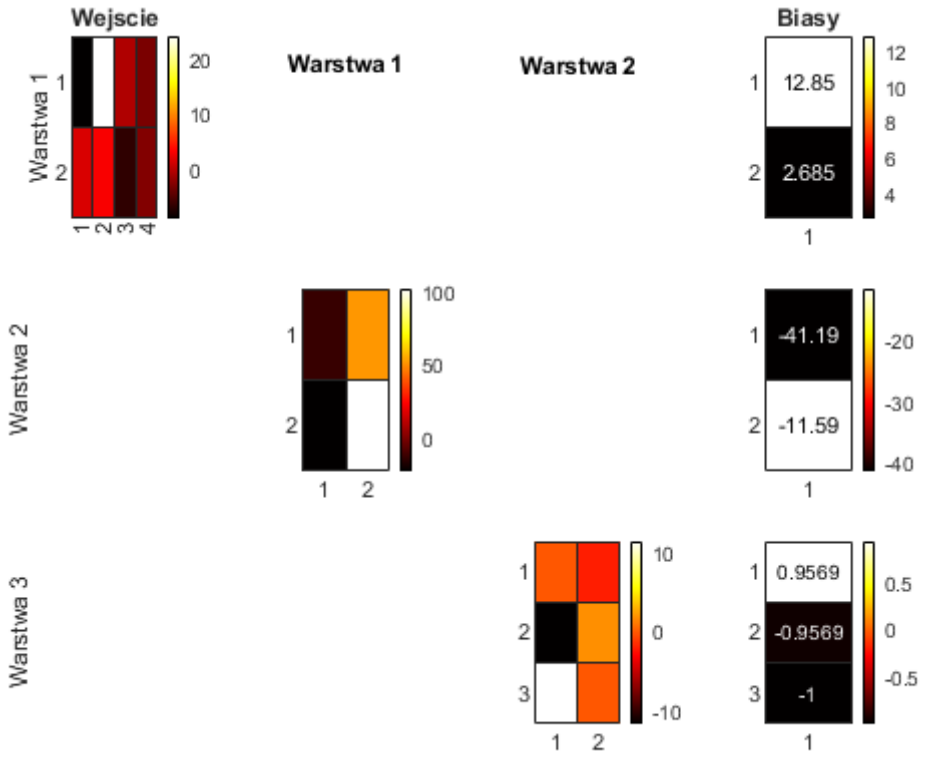
Wykonano testy aplikacji, stosując następujące ustawienia:

- liczba ukrytych warstw sieci: 2,
- liczba neuronów w poszczególnych warstwach ukrytych: [2, 2],
- liczba epok uczenia sieci: 50.

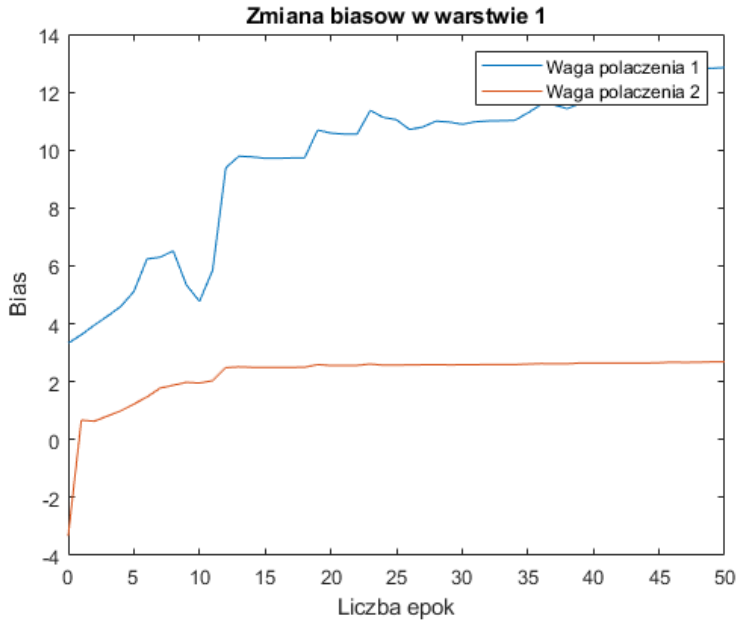
Rys. 1. Przykładowa animacja przedstawiająca zmianę wag połączeń w warstwie 1 dla neuronu 1.

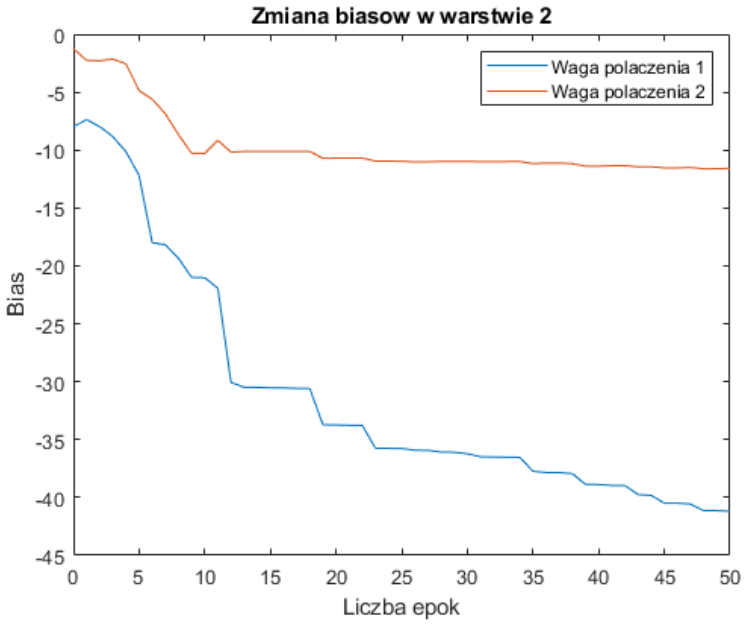


Rys. 2. Wykres przedstawiający wizualizację macierzy błędów wraz z porównaniem z wizualizacją otrzymaną przy pomocy funkcji **plotconfusion**.

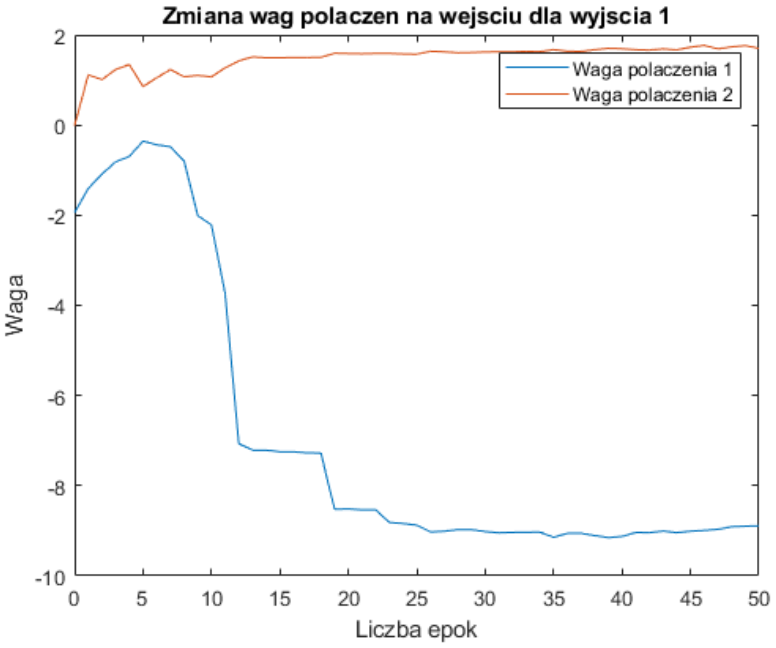


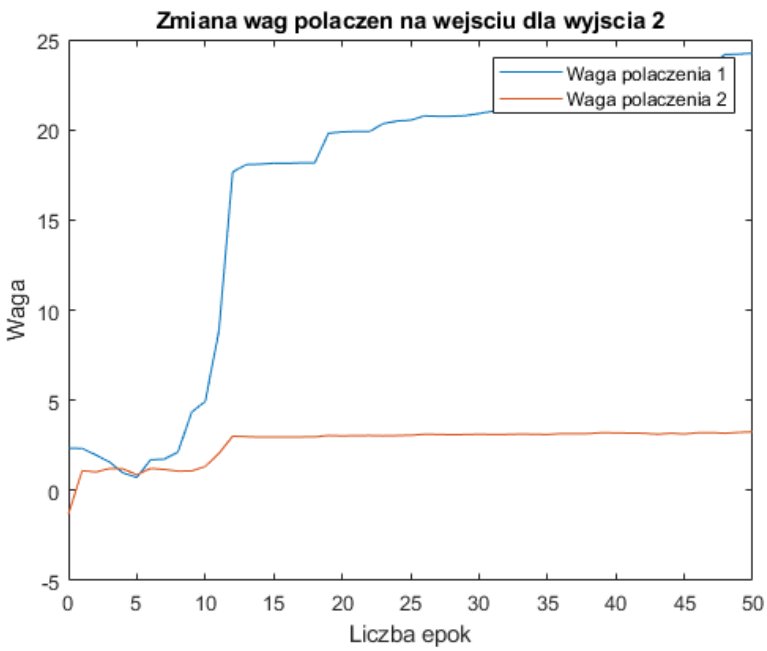
Rys. 3. Wykres wartości wszystkich wag i biasów w formie diagramu.



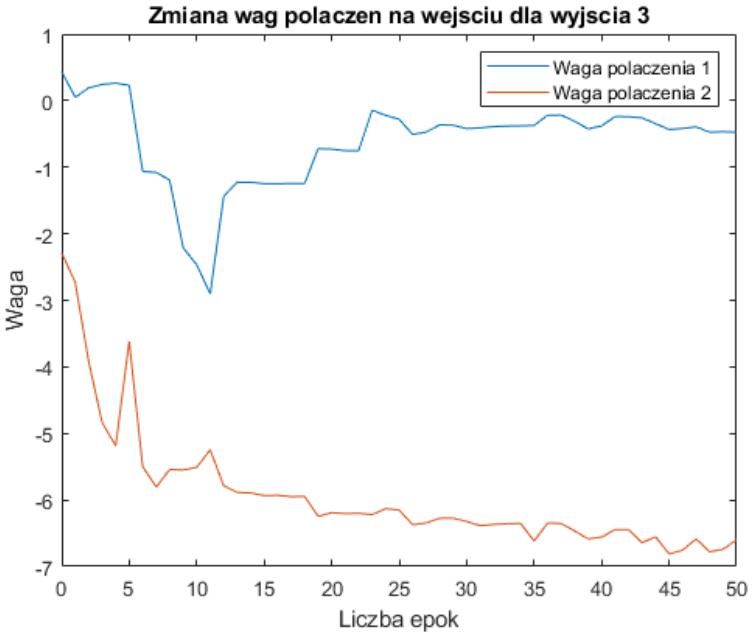


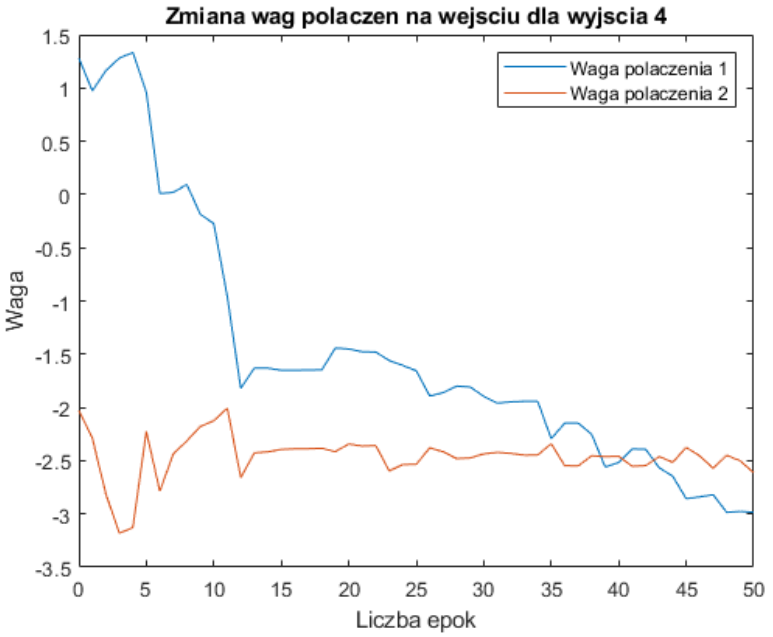
Rys. 4. Wykresy zmiany biasów w warstwie 1 (a) oraz w warstwie 2 (b).



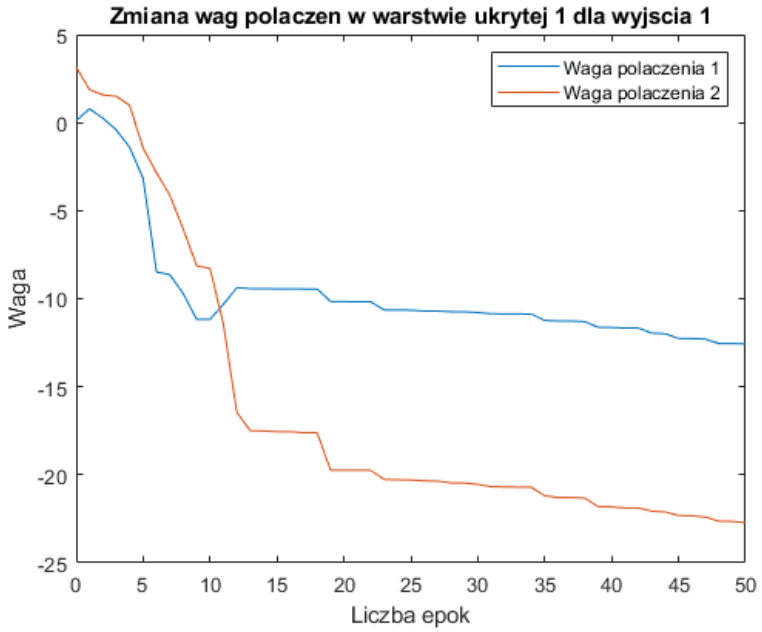


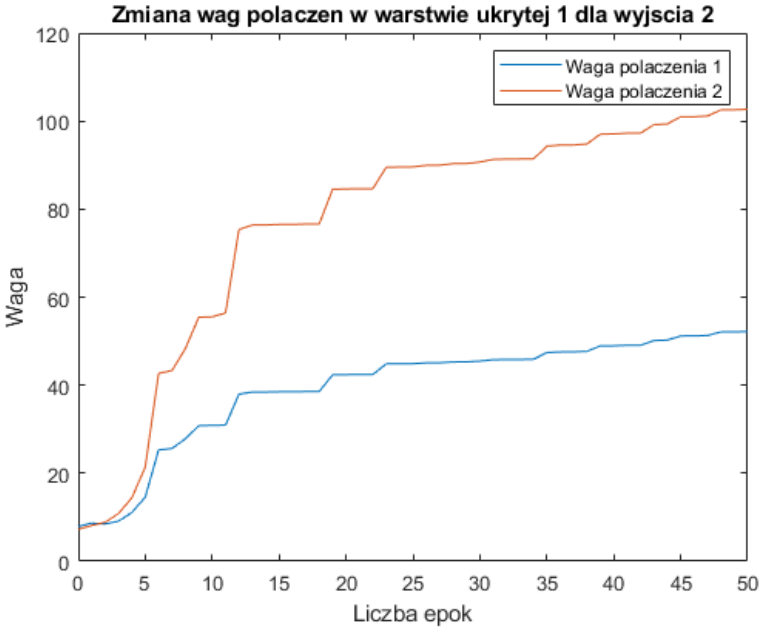
Rys. 5. Wykresy zmiany wag połączeń na wejściu dla neuronu 1 (a) oraz 2 (b).



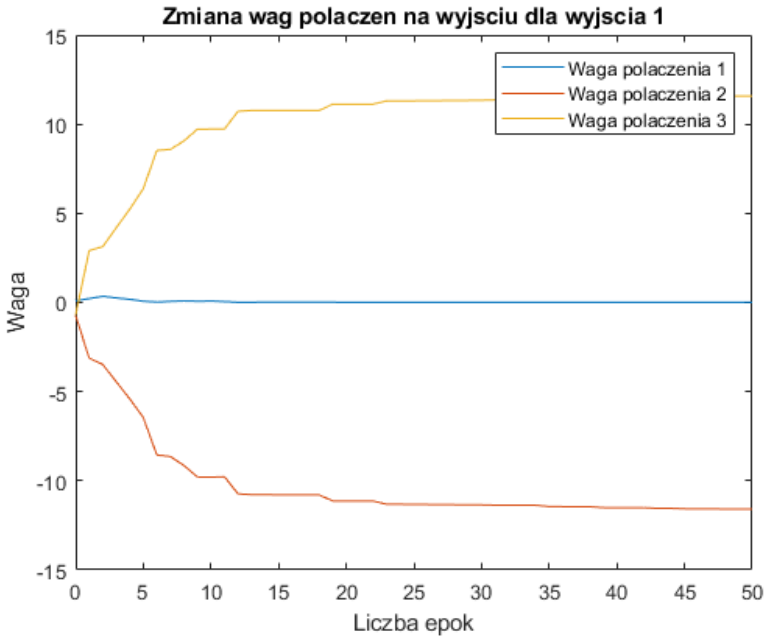


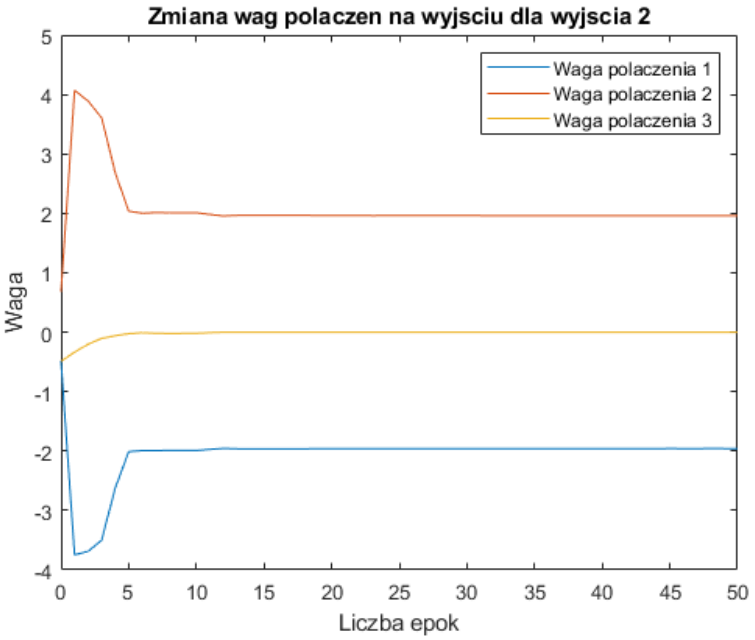
Rys. 6. Wykresy zmiany wag połączeń na wejściu dla neuronu 3 (a) oraz 4 (b).





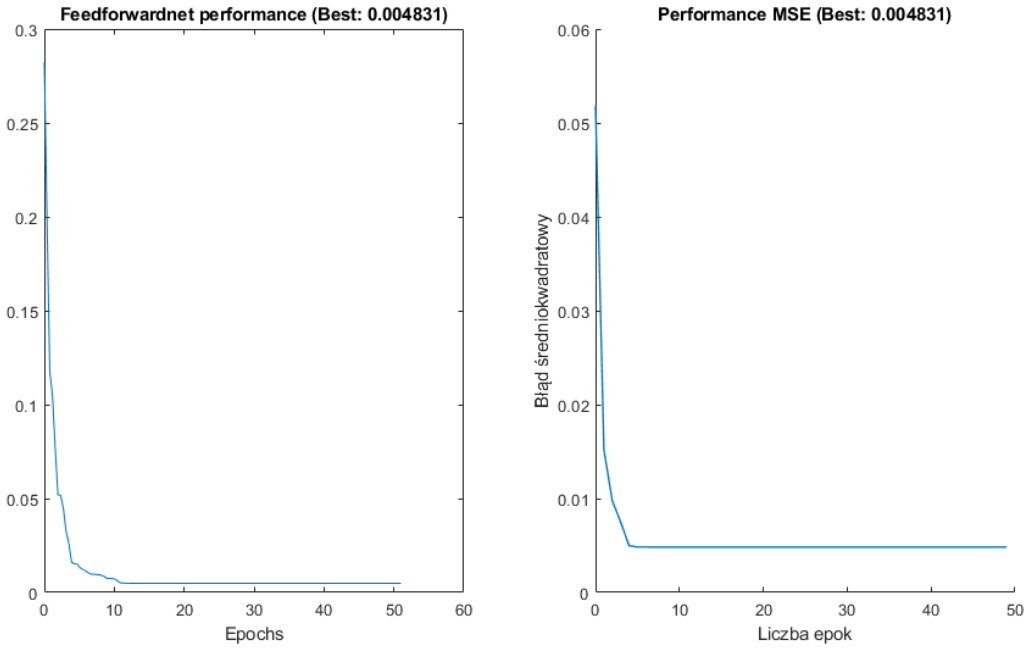
Rys. 7.Wykresy zmiany wag połączeń w warstwie ukrytej 1 dla neuronu 1 (a) oraz dla neuronu 2 (b).



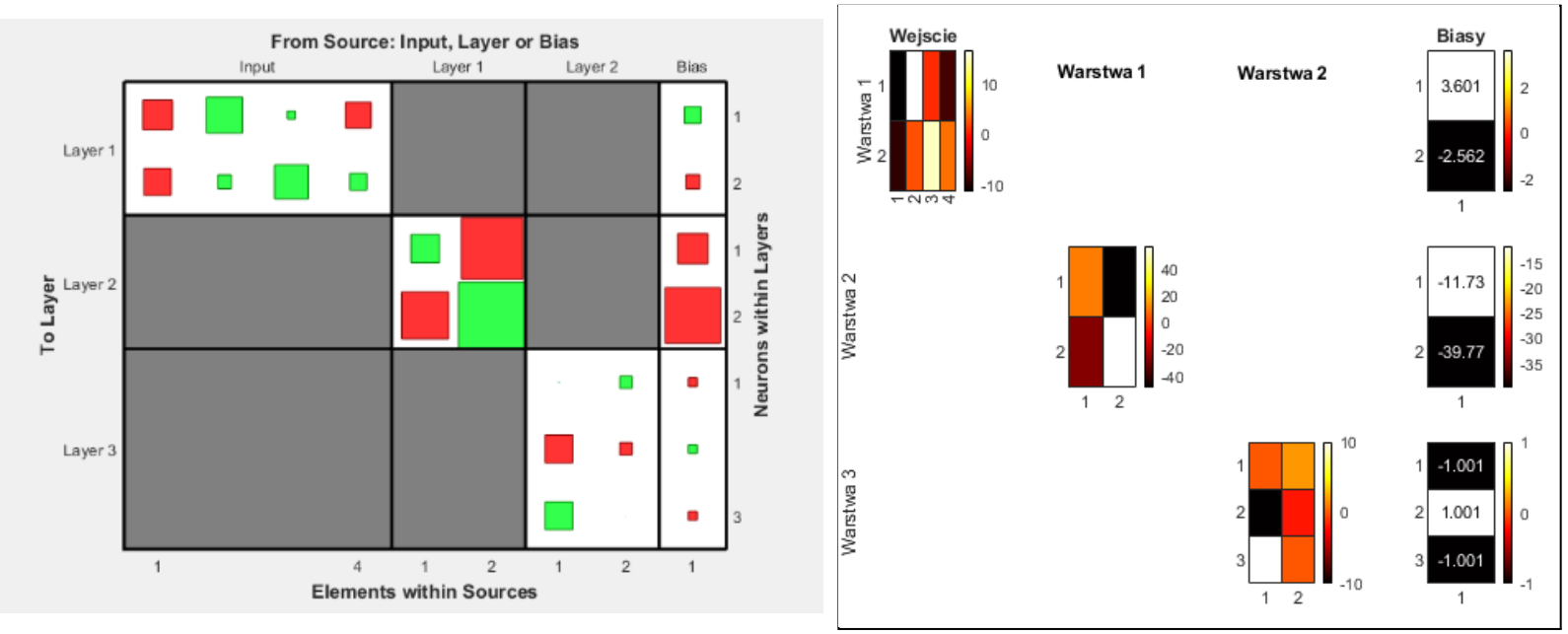


Rys. 8. Wykresy zmian wag połączeń na wyjściu dla neuronu 1 (a) oraz 2 (b).

Rys. 9. Wykres zmiany biasów na wyjściu dla poszczególnych połączeń.



Rys. 10. Porównanie otrzymanego wykresu performance (po prawej) z wykresem zwróconym z biblioteki feedforwardnet (po lewej).



Rys. 11. Porównanie otrzymanego diagramu wag i biasów (po prawej) z diagramem otrzymanym przy pomocy funkcji **plotwb** z *Deep Learning Toolbox* (po lewej).

5. Podsumowanie

Przy realizacji projektu korzystano z systemu kontroli wersji Git.
Projekt został umieszczony w serwisie GitHub i znajduje się pod tym linkiem:

https://github.com/kamilsudol/MIO_PROJEKT

Wyniki zaprezentowane w punkcie 4 są zgodne z oczekiwanymi. Porównano jakość otrzymanych wizualizacji z wbudowanymi funkcjami takimi jak ***plotwb*** (rys. 11), ***plotconfusion*** (rys. 2).

Wykresy wartości biasów i wag połączeń zgodnie z oczekiwaniami stabilizują się wraz ze wzrostem iteracji. Co więcej, stabilizacja w połączeniu z dość niskim błędem średniokwadratowym skutkuje wartością accuracy na poziomie 100%.

Aby uruchomić projekt należy otworzyć plik main.m w oprogramowaniu Matlab. Następnie uruchamiamy Live Editor (klikając prawym przyciskiem myszy na main.m -> Open as Live Script).

Po wprowadzeniu odpowiednich parametrów:

- *layers*, np. [2 2]
- *epochs*, np. 20
- *plottype*, np. ‘animated’, ‘all’ lub ‘compare’

do wywołania funkcji **SSNvisualisation** należy wybrać opcję Run main (F5).