Sieć wielowarstwowa
Sieci Neuronowe 2020

Jakub Ciszek
238035

Spis treści

Cały kod wykorzystany w zadaniu znajduje się pod adresem: `https://github.com/Greenpp/sieci-neuronowe-pwr-2020`

1 Opis badań

1.1 Plan eksperymentów

Wszystkie eksperymenty zostały przeprowadzone 10 razy. Losowość przy inicjalizacji wag oraz generacji danych nie została narzucona żadnym ziarnem. Podczas badań przyjęto górną granicę 5 epok, po przekroczeniu której, uczenie zostawało przerywane. Ze względu na charakter zadania (klasyfikacja) na ostatniej warstwie użyto funkcji Softmax, a za funkcję straty przyjęto Entropię krzyżową. Z powodów wydajnościowych testowanie modelu przeprowadzano co każde 1024 przykłady, niezależnie od wielkości paczki.
Zgodnie z instrukcją zostały przeprowadzone następujące badania:

- Wpływ wielkości warstwy ukrytej na przebieg procesu uczenia
- Wpływ wielkości paczki na przebieg procesu uczenia
- Wpływ zakresu inicjalizacji wag na przebieg procesu uczenia
- Wpływ wartości współczynnika alpha na przebieg procesu uczenia
- Wpływ użytej funkcji aktywacyjnej na przebieg procesu uczenia

Podczas wizualizacji funkcji straty pominięto pierwsze 10 pomiarów dla lepszej czytelności.

1.2 Charakterystyka zbiorów danych

Danymi użytymi w zadaniu jest zbiór ręcznie pisanych cyfr $0 - 9$ - MNIST. Na zbiór składa się 70,000 obrazów wielkości 28x28 pikseli, co po przekształceniu odpowiadało 784 elementowemu wektorowi wejściowemu i 10 klasom na wyjściu. Użyta w zadaniu wersja została podzielona na 3 zbiory:

- Uczący - 50,000 przykładów.
- Walidujący - 10,000 przykładów.
- Testowy - 10,000 przykładów.

W trakcie eksperymentów wykorzystano jedynie zbiory uczący i testowy.

2 Eksperymenty

2.1 Wpływ wielkości warstwy ukrytej na przebieg procesu uczenia

Założenia

Tabela 1: Stałe dla eksperymentu 1

Parametr	Wartość
Wielkość paczki	32
Zakres wag	-0.5 – 0.5
Współczynnik uczenia	0.01
Funkcja aktywacji	ReLU

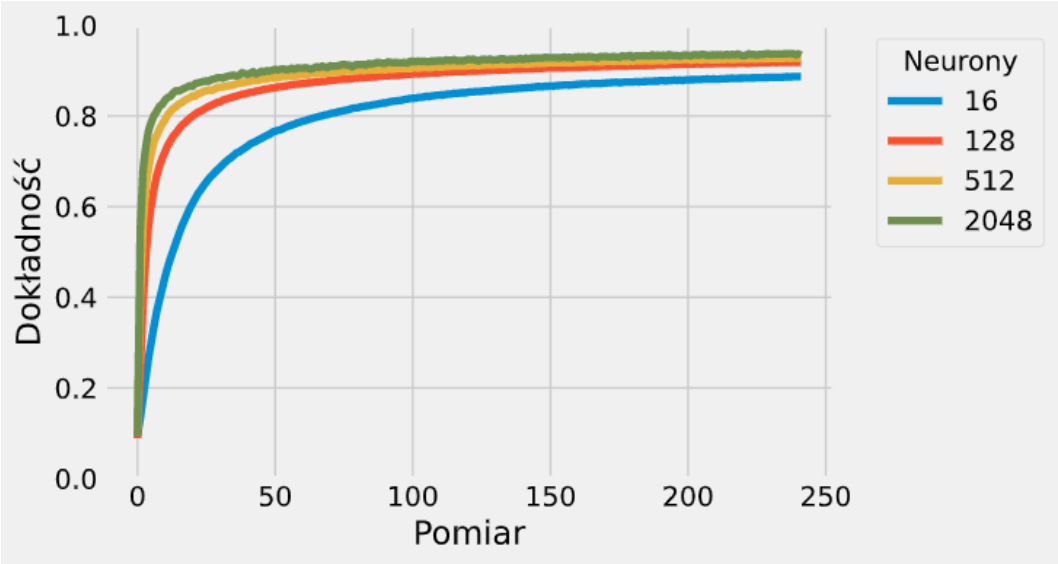
Zmienną w tym eksperymencie była wielkość warstwy ukrytej. Ilość neuronów przyjmowała wartości ze zbioru {16, 128, 512, 2048}

Przebieg

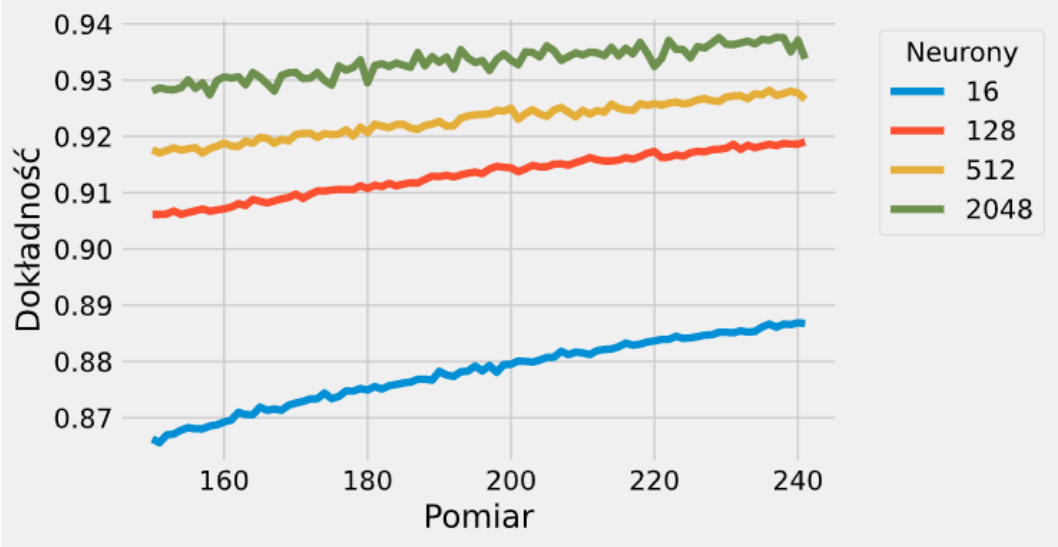
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy.

Wyniki

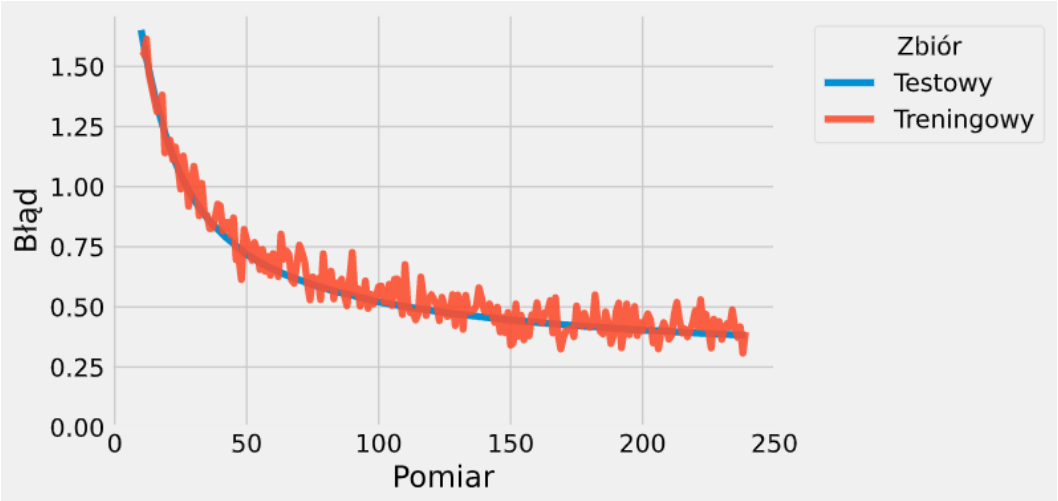
Wykres 1: Dokładność modelu w zależności od wielkości warstwy ukrytej



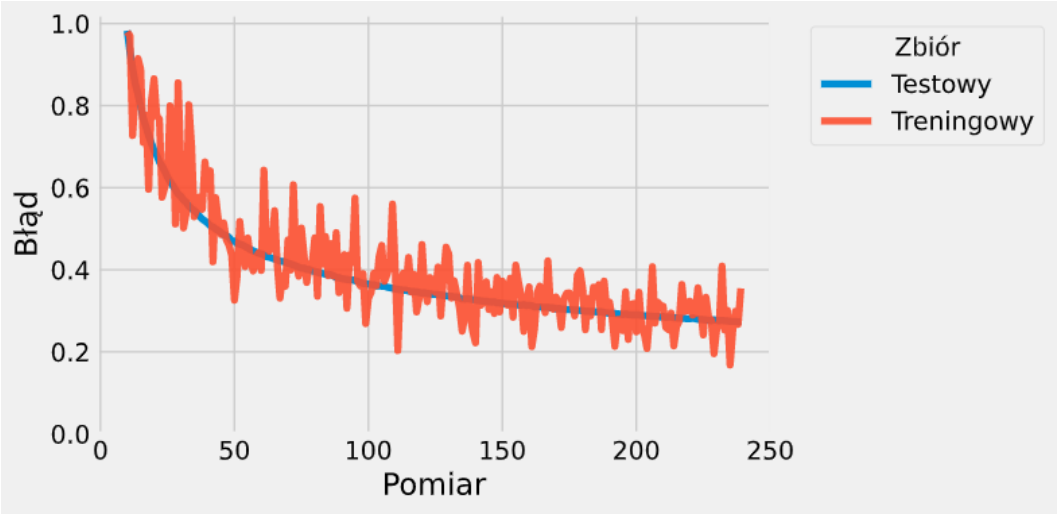
Wykres 2: Dokładność modelu w końcowym etapie uczenia w zależności od wielkości warstwy ukrytej



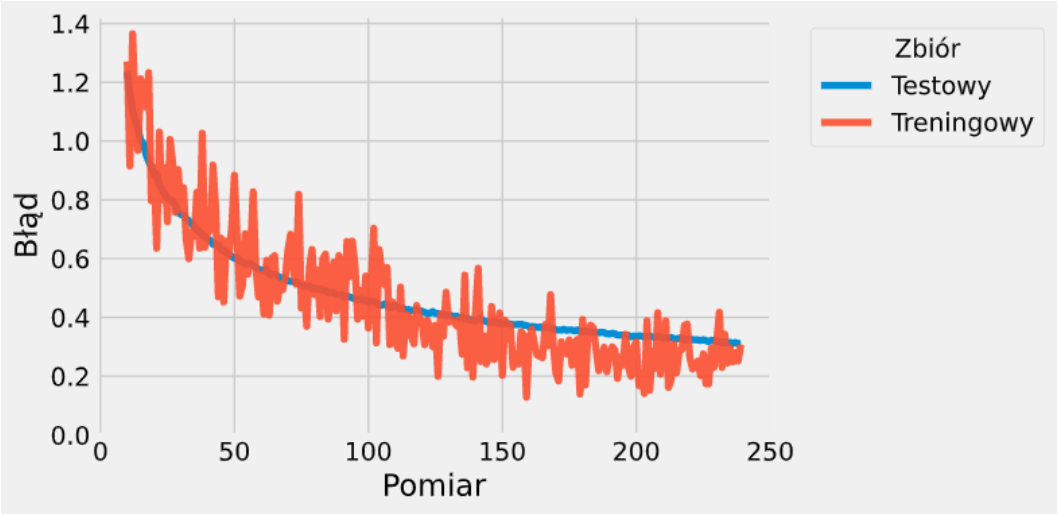
Wykres 3: Zachowanie funkcji błędu dla 16 neuronów



Wykres 4: Zachowanie funkcji błędu dla 128 neuronów



Wykres 5: Zachowanie funkcji błędu dla 512 neuronów



Wykres 6: Zachowanie funkcji błędu dla 2048 neuronów

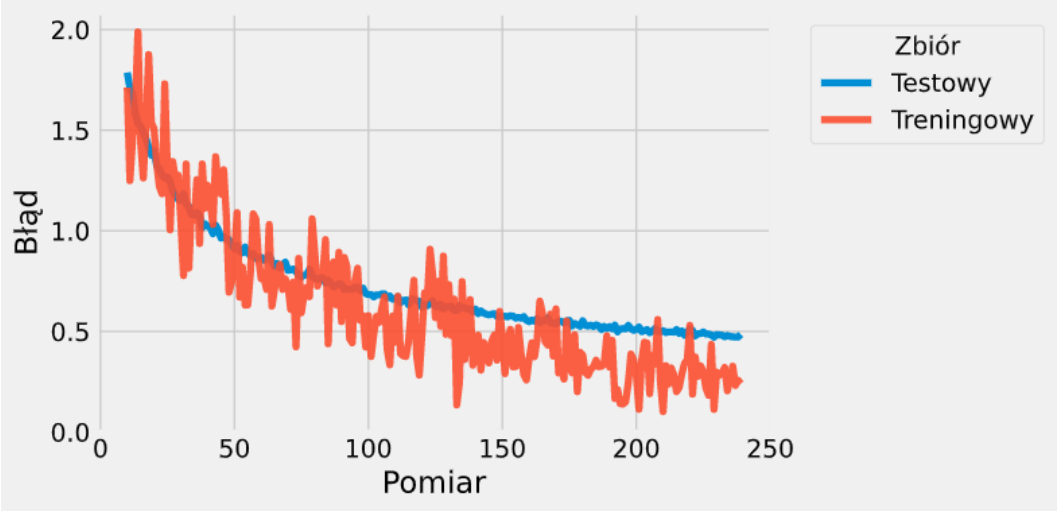


Tabela 2: Średnia maksymalna dokładność w zależności od wielkości warstwy ukrytej

Neurony	Dokładność [%]
16	88.80
128	92.00
512	92.97
2048	**94.04**

Wnioski

Z otrzymanych wyników, widocznych na wykresach 1, 2 oraz tabeli 2, wynika że większa ilość neuronów w warstwie ukrytej poprawia dokładność modelu, większa ilość parametrów pozwala na lepsze dopasowanie do danych w krótszym czasie. Można zauważyć jak wraz ze zmniejszaniem się liczby neuronów spada nie tylko dokładność modelu ale i szybkość uczenia. Taka strategia jednak grozi przeuczeniem przy większej ilości epok. Błąd, widoczny na wykresach 3, 4, 5 i 6, otrzymywany z mniejszych modeli jest stabilniejszy z powodu mniejszej ilości parametrów. Dodatkową obserwacją jest krótszy czas obliczeń przy uczeniu mniejszych modeli, spowodowany mniejszą ilością parametrów do zaktualizowania oraz obliczenia podczas ewaluacji.

2.2 Wpływ wielkości paczki na przebieg procesu uczenia

Założenia

Tabela 3: Stałe dla eksperymentu 2

Parametr	Wartość
Wielkość warstwy ukrytej	128
Zakres wag	-0.5 – 0.5
Współczynnik uczenia	0.01
Funkcja aktywacji	ReLU

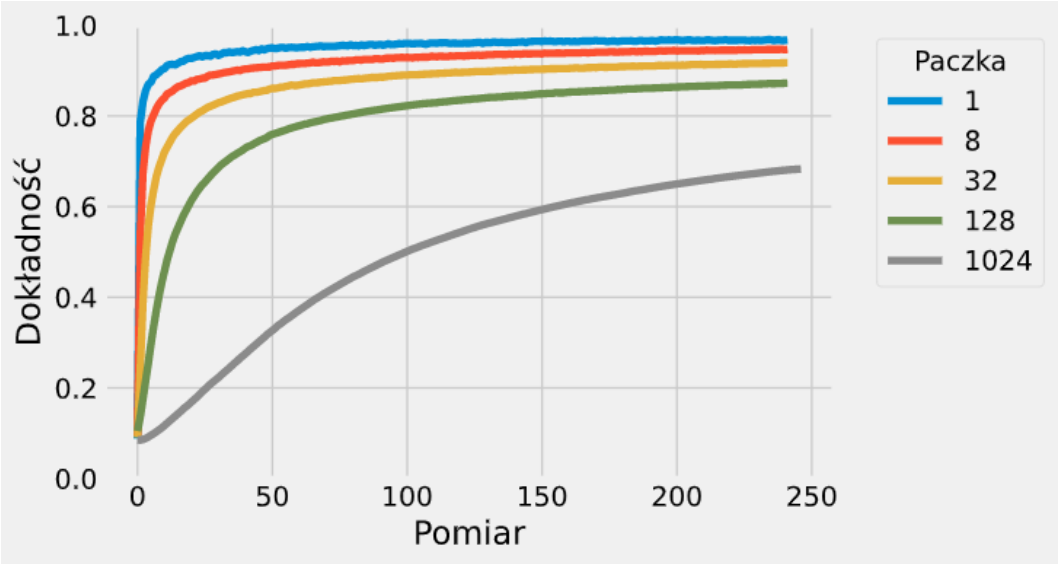
Zmienną w tym eksperymencie była wielkość paczki. Ilość przykładów przyjmowała wartości ze zbioru {1, 8, 32, 128, 1024}

Przebieg

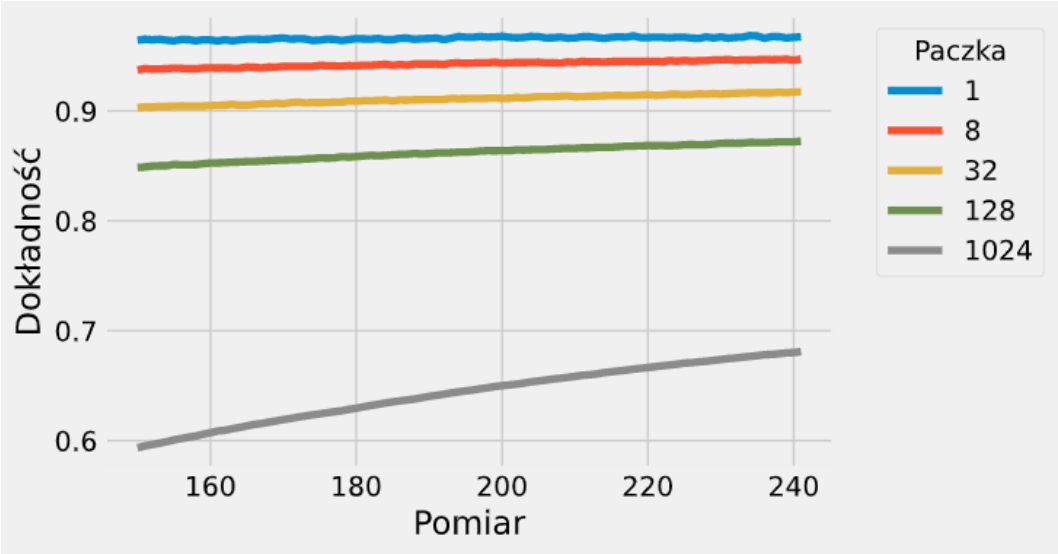
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy.

Wyniki

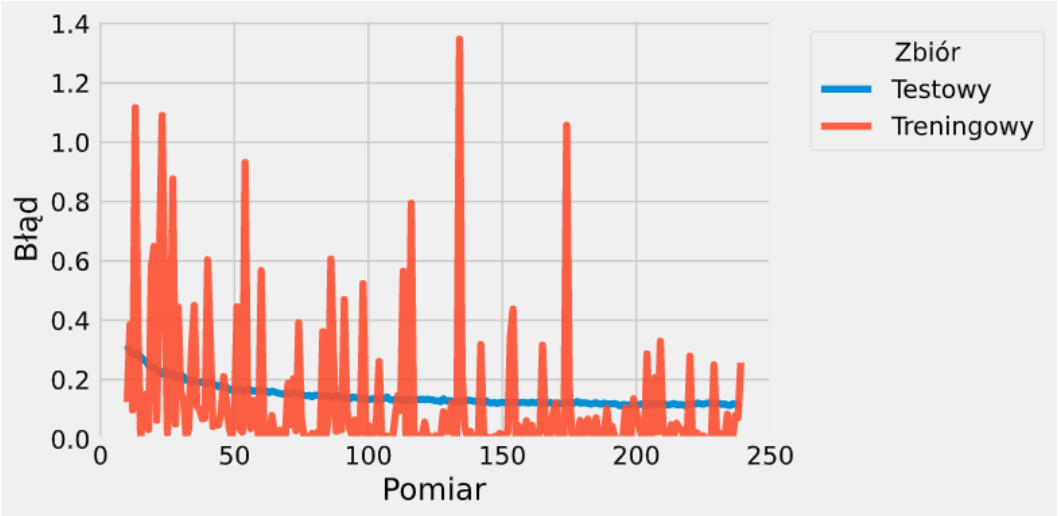
Wykres 7: Dokładność modelu w zależności od wielkości paczki



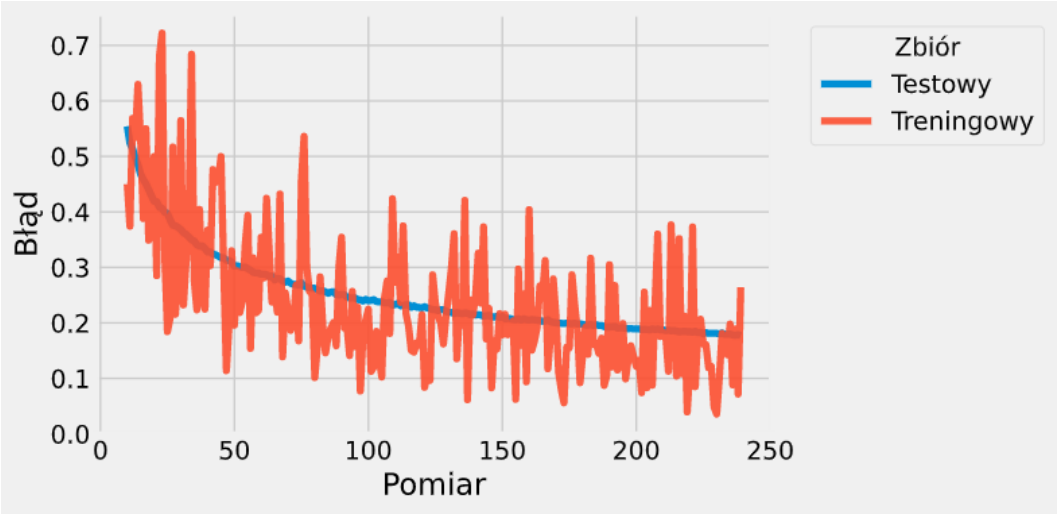
Wykres 8: Dokładność modelu w końcowym etapie uczenia w zależności od wielkości paczki



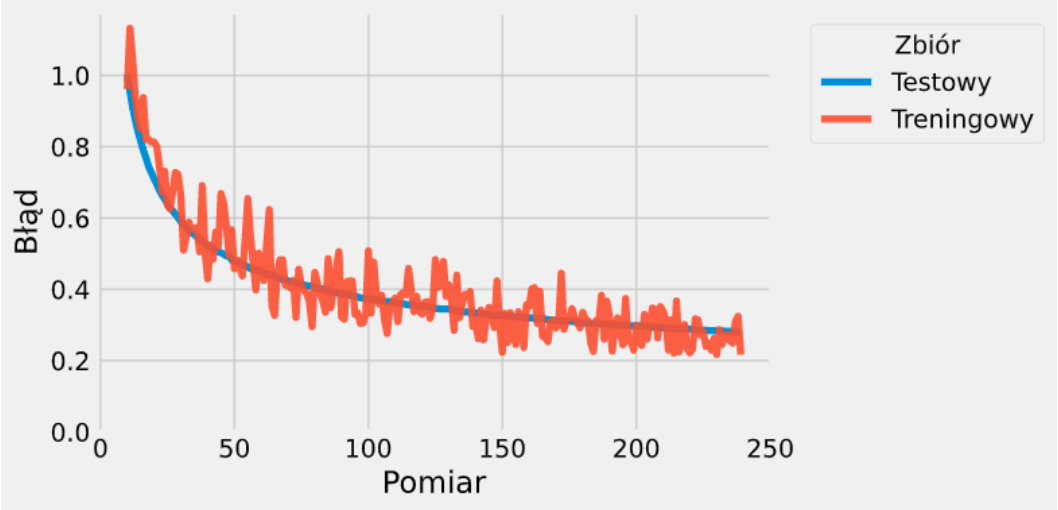
Wykres 9: Zachowanie funkcji błędu dla paczki wielkości 1



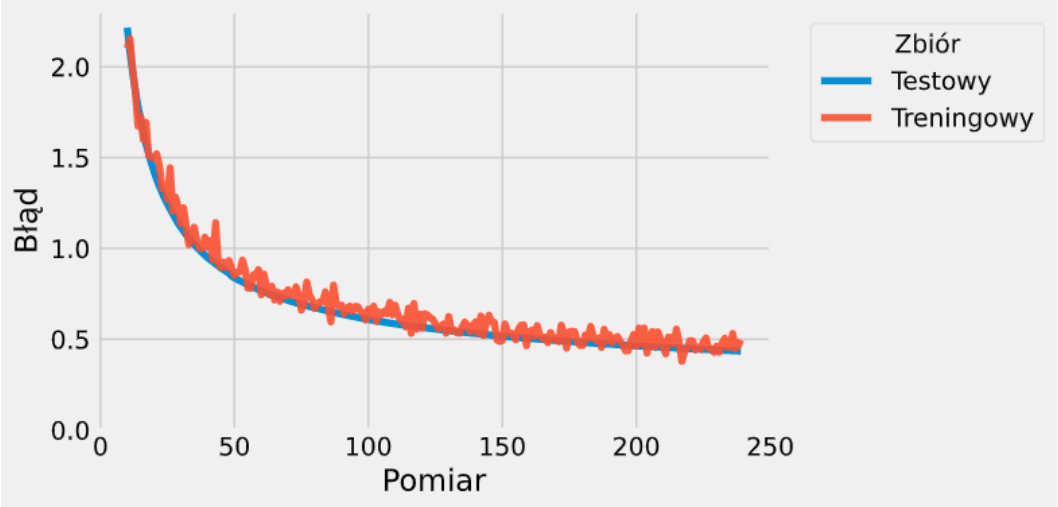
Wykres 10: Zachowanie funkcji błędu dla paczki wielkości 8



Wykres 11: Zachowanie funkcji błędu dla paczki wielkości 32



Wykres 12: Zachowanie funkcji błędu dla paczki wielkości 128



Wykres 13: Zachowanie funkcji błędu dla paczki wielkości 1024

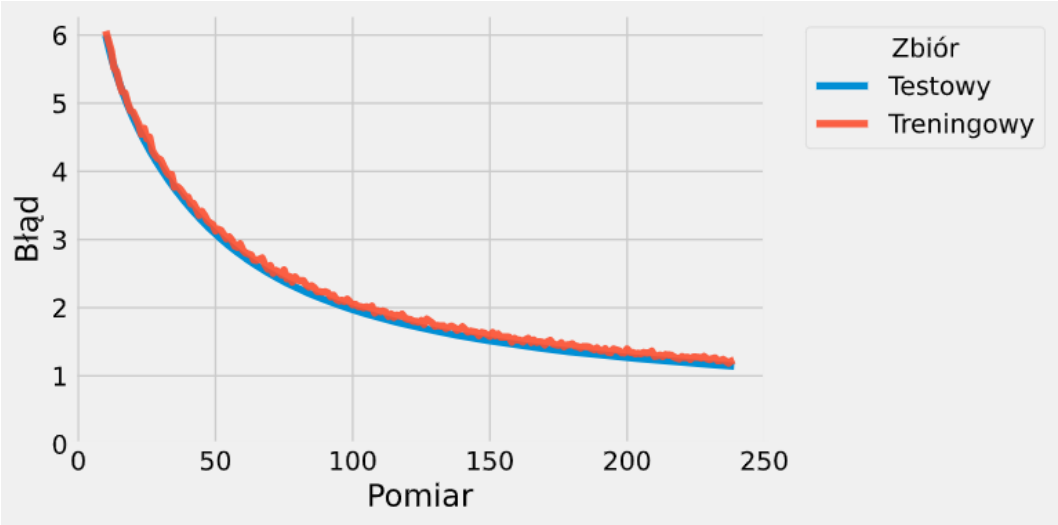


Tabela 4: Średnia maksymalna dokładność w zależności od wielkości paczki

Przykłady	Dokładność [%]
1	**97.07**
8	94.85
32	91.83
128	87.29
1024	68.31

Wnioski

Z otrzymanych wyników, widocznych na wykresach 7, 8 oraz tabeli 4, wynika że małe wielkości paczki skutkują lepszą dokładnością modelu. Spowodowane jest to większą ilością aktualizacji na przestrzeni tej samej ilości przykładów. Zaletą mniejszych paczek, widoczną na wykresach 9, 10, 11, 12 i 13 jest wprowadzanie szumów do błędu treningowego, co przeciwdziała dążeniu do minimów lokalnych i poprawia generalizację. Dodatkową obserwacją jest krótszy czas obliczeń dla większych paczek spowodowany mniejszą ilością przeprowadzonych propagacji dla takiego samego zbioru uczącego.

2.3 Wpływ zakresu inicjalizacji wag na przebieg procesu uczenia

Założenia

Tabela 5: Stałe dla eksperymentu 3

Parametr	Wartość
Wielkość warstwy ukrytej	128
Wielkość paczki	32
Współczynnik uczenia	0.01
Funkcja aktywacji	ReLU

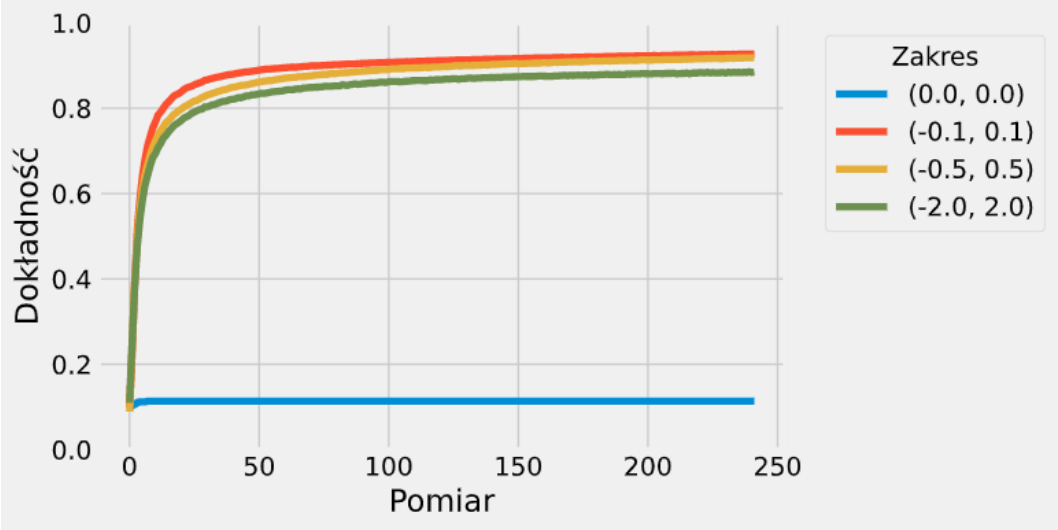
Zmienną w tym eksperymencie był zakres inicjalizacji wag. Przedział inicjalizacji przyjmował wartości ze zbioru {0.0, -0.1 – 0.1, -0.5 – 0.5, -2.0 – 2.0}

Przebieg

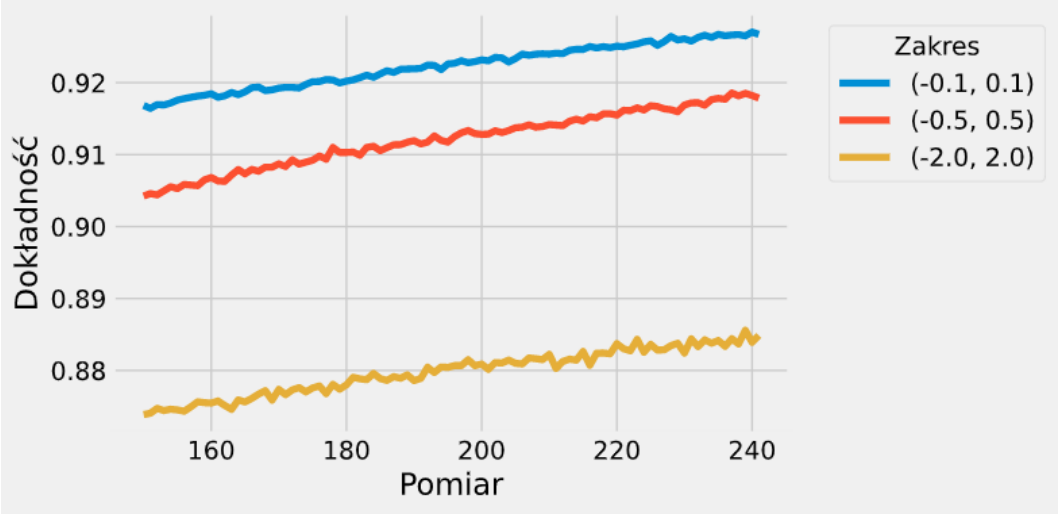
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy.

Wyniki

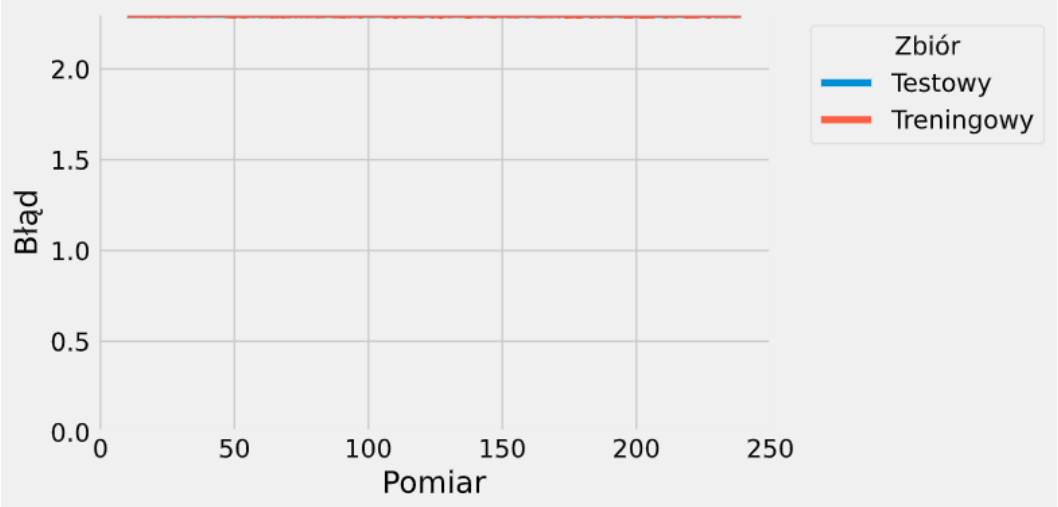
Wykres 14: Dokładność modelu w zależności od zakresu inicjalizacji wag



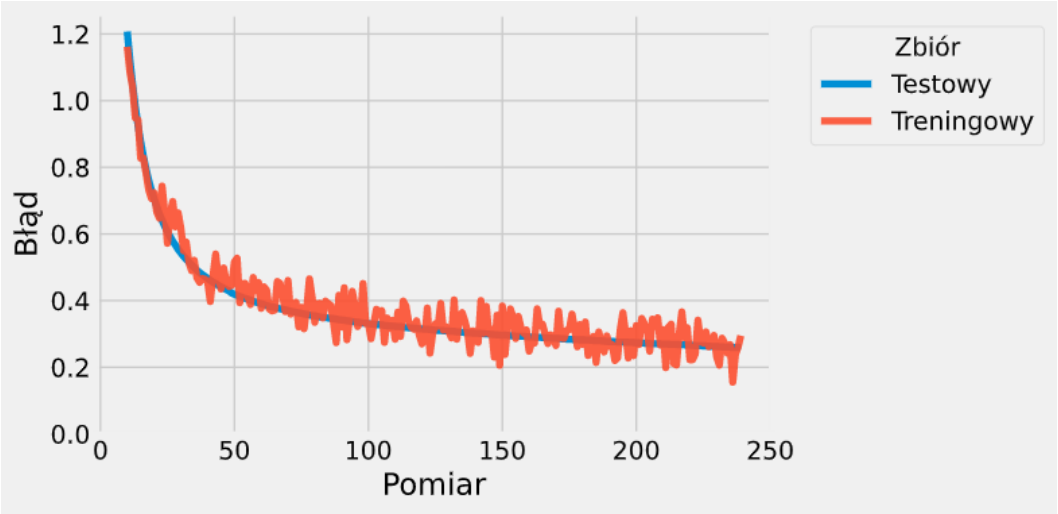
Wykres 15: Dokładność modelu w końcowym etapie uczenia w zależności od zakresu inicjalizacji wag



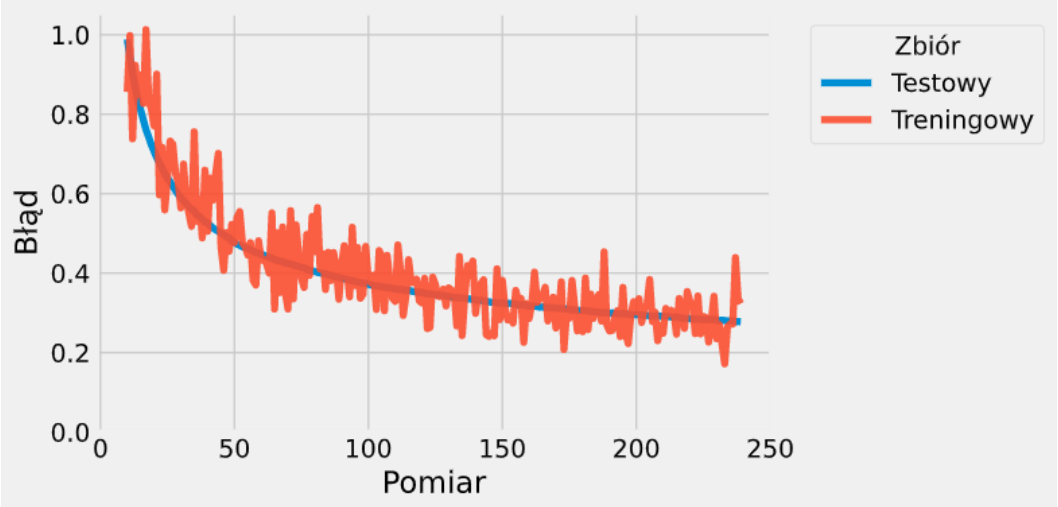
Wykres 16: Zachowanie funkcji błędu dla wag inicjalizowanych z zakresu 0.0 – 0.0



Wykres 17: Zachowanie funkcji błędu dla wag inicjalizowanych z zakresu -0.1 – 0.1



Wykres 18: Zachowanie funkcji błędu dla wag inicjalizowanych z zakresu -0.5 – 0.5



Wykres 19: Zachowanie funkcji błędu dla wag inicjalizowanych z zakresu -2.0 – 2.0

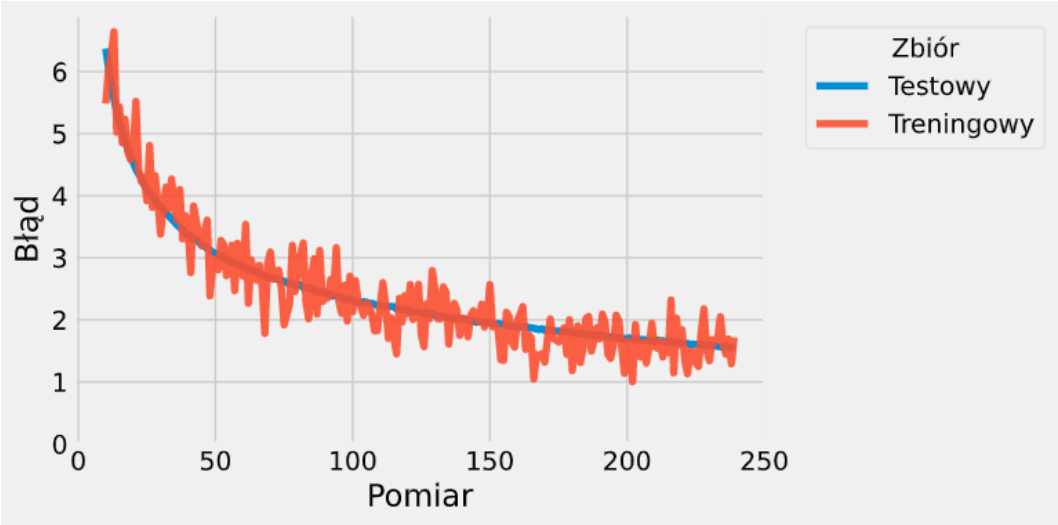


Tabela 6: Średnia maksymalna dokładność w zależności od zakresu inicjalizacji wag

Zakres	Dokładność [%]
0.0	11.35
-0.1 – 0.1	**92.76**
-0.5 – 0.5	91.94
-2.0 – 2.0	88.68

Wnioski

Z otrzymanych wyników, widocznych na wykresach 14, 15 oraz tabeli 6, wynika że mniejsze zakresy inicjalizacji wag dają lepsze wyniki. Wyjątkiem jest inicjalizacja na 0, która skutkuje jednakowymi wartościami parametrów oraz ich aktualizacji co przekłada się na brak możliwości dopasowania do danych.

2.4 Wpływ wartości współczynnika alpha na przebieg procesu uczenia

Założenia

Tabela 7: Stałe dla eksperymentu 4

Parametr	Wartość
Wielkość warstwy ukrytej	128
Wielkość paczki	32
Zakres wag	-0.5 – 0.5
Funkcja aktywacji	ReLU

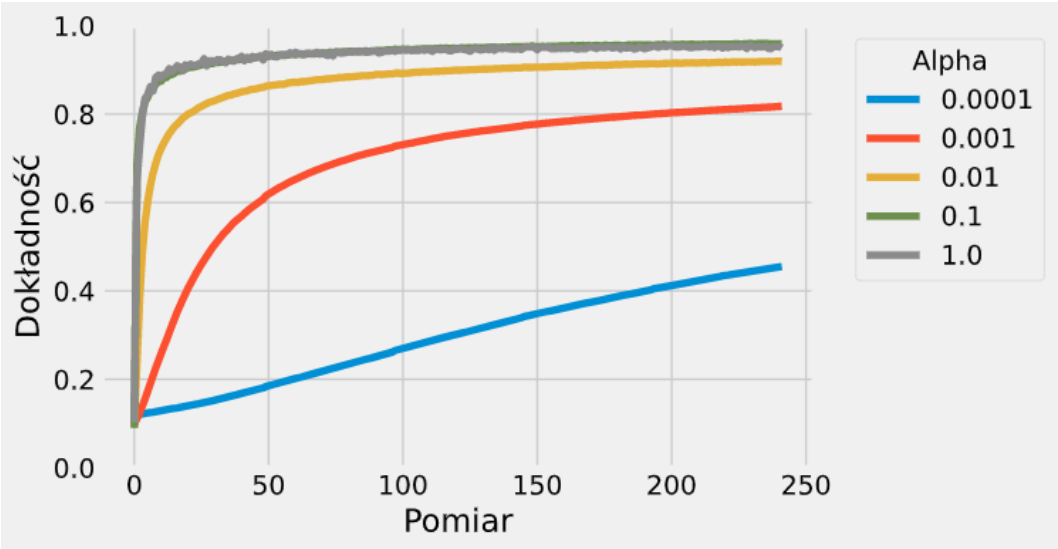
Zmienną w tym eksperymencie był współczynnik uczenia. Przyjmował wartości ze zbioru {0.0001, 0.001, 0.01, 0.1, 1.0,}

Przebieg

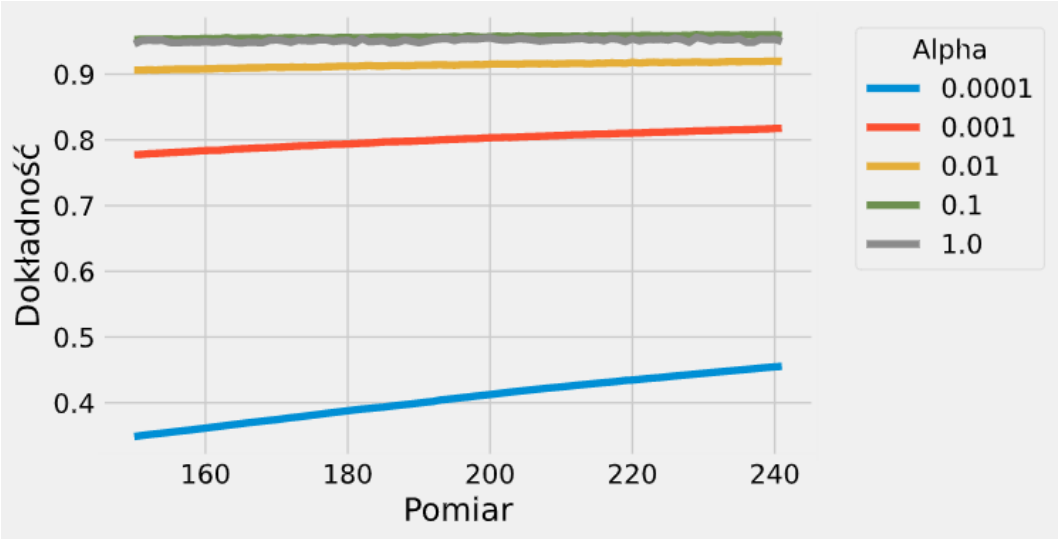
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy.

Wyniki

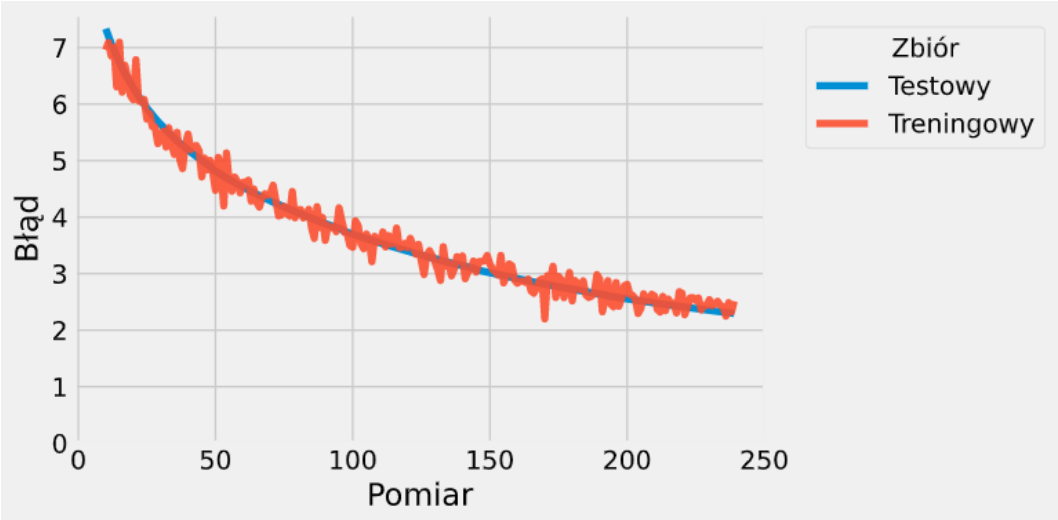
Wykres 20: Dokładność modelu w zależności od współczynnika uczenia



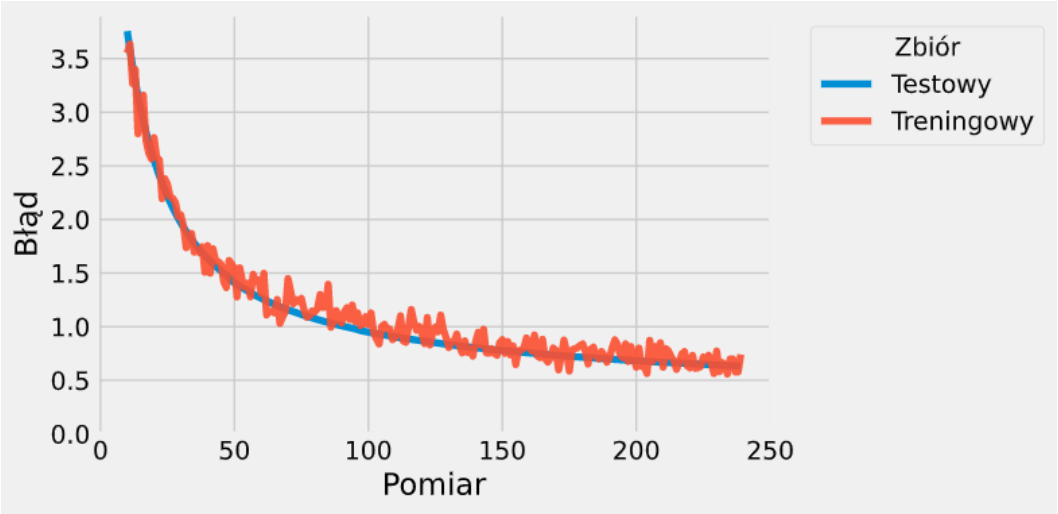
Wykres 21: Dokładność modelu w końcowym etapie uczenia w zależności od współczynnika uczenia



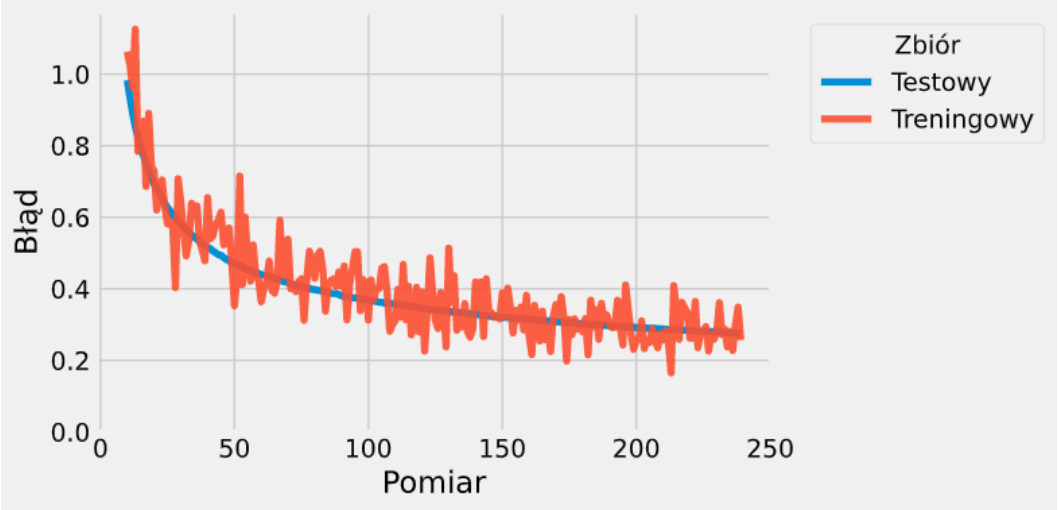
Wykres 22: Zachowanie funkcji błędu dla parametru alpha o wartości 0.0001



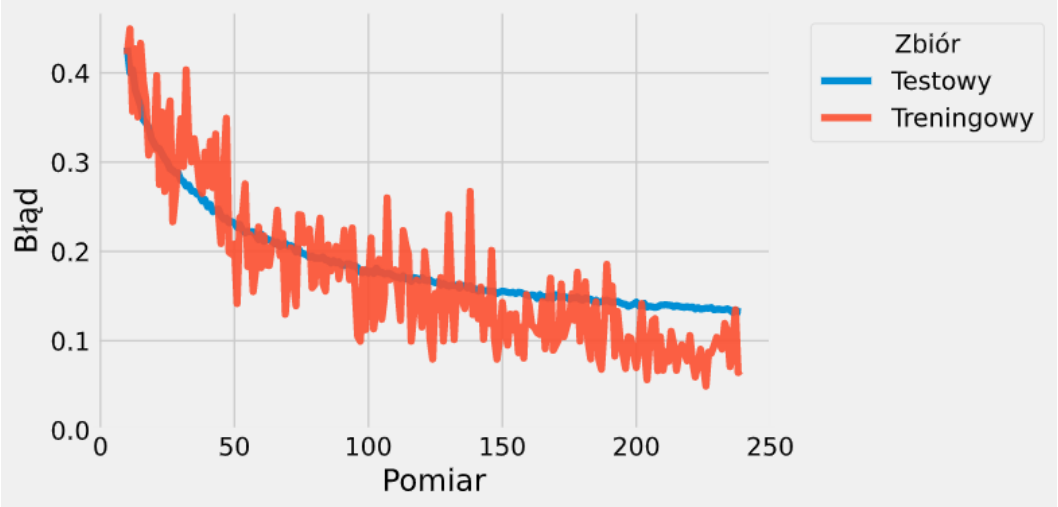
Wykres 23: Zachowanie funkcji błędu dla parametru alpha o wartości 0.001



Wykres 24: Zachowanie funkcji błędu dla parametru alpha o wartości 0.01



Wykres 25: Zachowanie funkcji błędu dla parametru alpha o wartości 0.1



Wykres 26: Zachowanie funkcji błędu dla parametru alpha o wartości 1.0

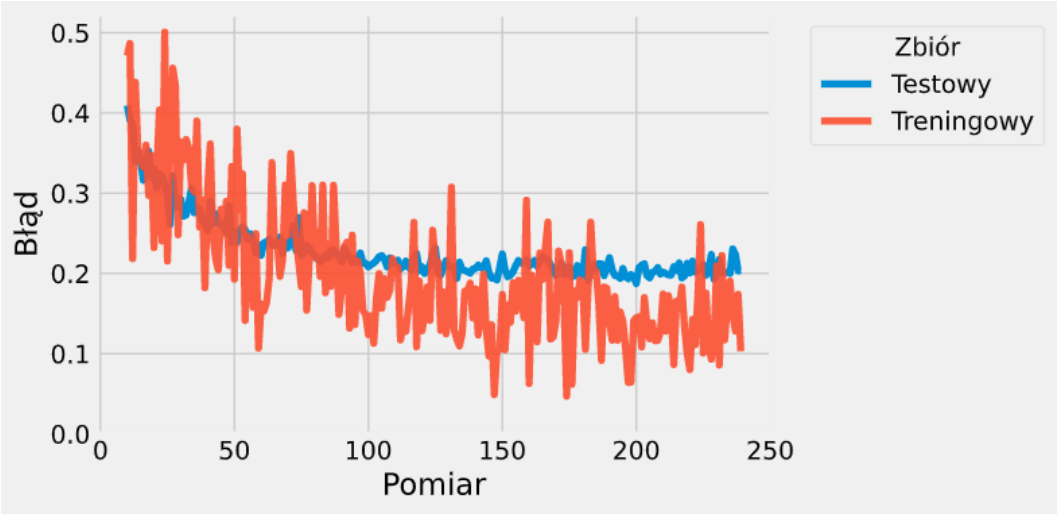


Tabela 8: Średnia maksymalna dokładność w zależności od współczynnika uczenia

Alpha	Dokładność [%]
0.0001	45.54
0.0010	81.77
0.0100	92.06
0.1000	**96.18**
1.0000	95.98

Wnioski

Z otrzymanych wyników, widocznych na wykresach 20, 21 oraz tabeli 8, wynika że duże wartości współczynnika alpha przyspieszają uczenie. Zbyt duża wartość alpha powinna jednak uniemożliwić uzyskanie podobnej dokładności modelu z powodu zbyt dużych kroków i przeskakiwania minimów. Bardzo małe wartości alpha powodują wolne uczenie się modelu oraz możliwe utykanie w minimach lokalnych. Na wykresach 22, 23 i 24 widać, że mniejsze wartości alpha stabilizują błąd treningowy, natomiast przy wartości 1.0 na wykresie 24 można zauważyć wzrastający błąd testowy, co może oznaczać przeuczenie modelu.

2.5 Wpływ użytej funkcji aktywacyjnej na przebieg procesu uczenia

Założenia

Tabela 9: Stałe dla eksperymentu 5

Parametr	Wartość
Wielkość warstwy ukrytej	128
Wielkość paczki	32
Zakres wag	-0.5 – 0.5
Współczynnik uczenia	0.01

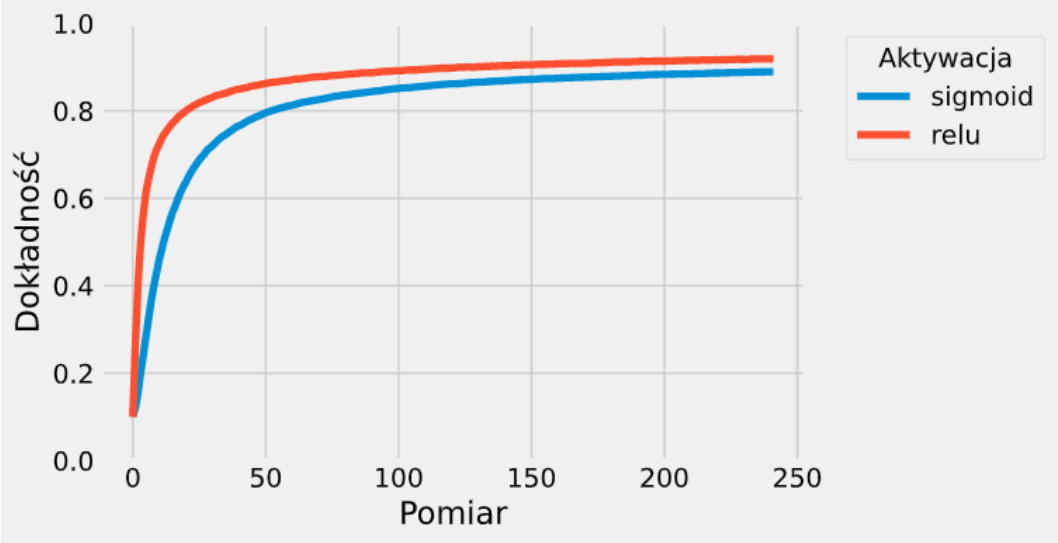
Zmienną w tym eksperymencie była funkcja aktywacji. Przetestowane zostały funkcje Sigmoidalna oraz ReLU.

Przebieg

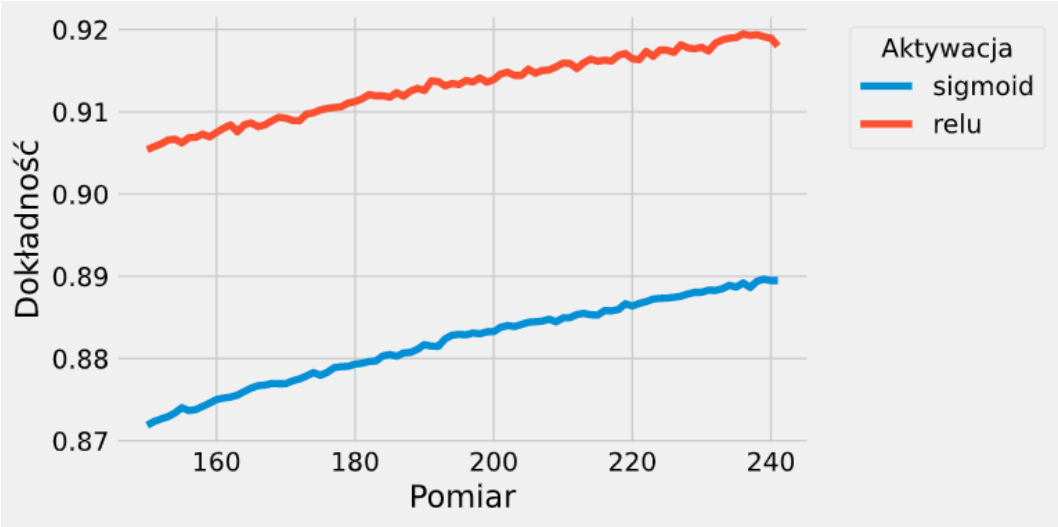
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy.

Wyniki

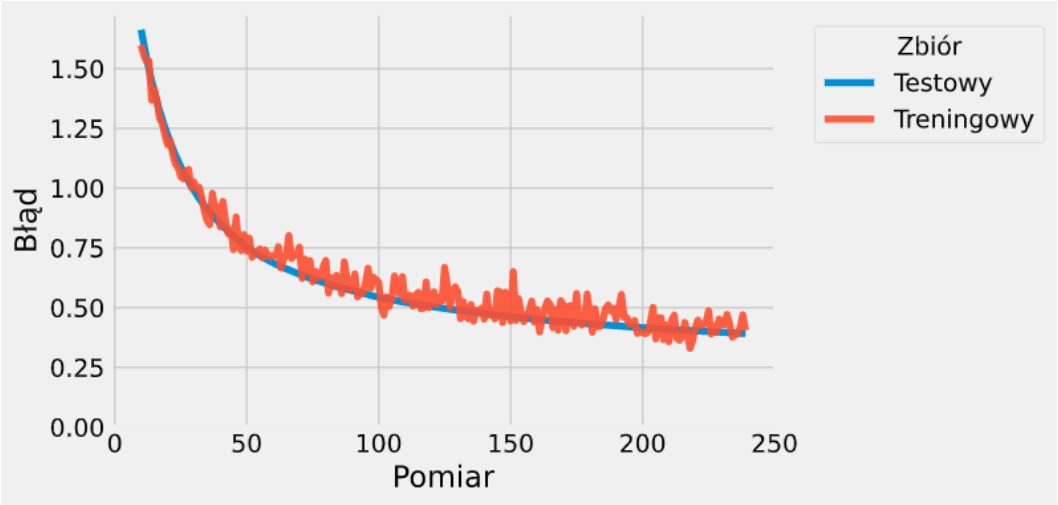
Wykres 27: Dokładność modelu w zależności od funkcji aktywacji



Wykres 28: Dokładność modelu w końcowym etapie uczenia w zależności od funkcji aktywacji



Wykres 29: Zachowanie funkcji błędu dla funkcji aktywacji Sigmoidalnej



Wykres 30: Zachowanie funkcji błędu dla funkcji aktywacji ReLU

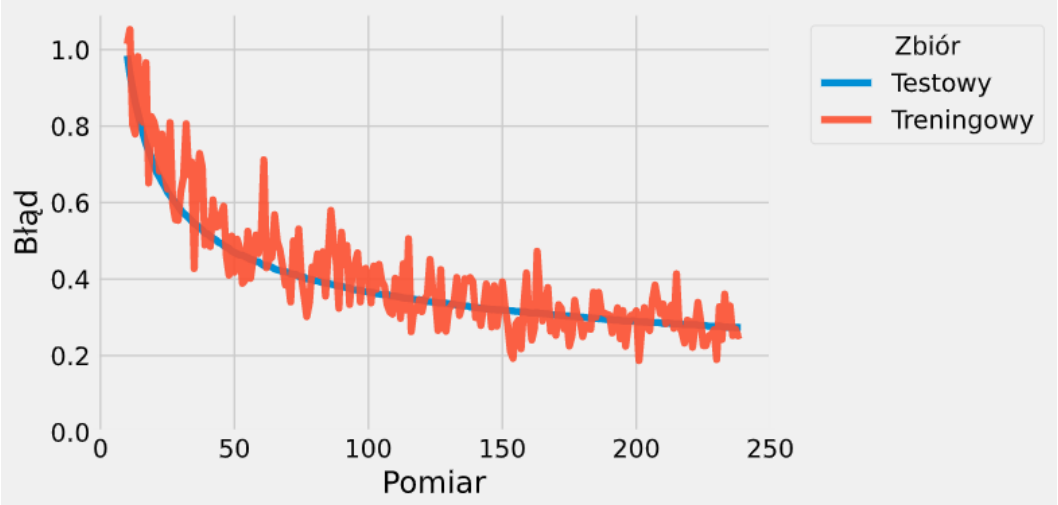


Tabela 10: Średnia maksymalna dokładność w zależności od funkcji aktywacji

Funkcja	Dokładność [%]
Sigmoid	89.03
ReLU	**92.04**

Wnioski

Z otrzymanych wyników, widocznych na wykresach 27, 28 oraz tabeli 10, wynika że funkcja ReLU daje znacząco lepsze wyniki niż Sigmoid. Używanie funkcji ReLU jest mniej kosztowne obliczeniowo, co dodatkowo przyspiesza trening. Na wykresach 29 i 30 można zauważyć, że funkcja ReLU powoduje bardziej niestabilny błąd, co może prowadzić do polepszenia efektów uczenia.

3 Wnioski

- Wielowarstwowe sieci neuronowe pozwalają na modelowanie nieliniowych funkcji.
- Powtarzającym się motywem jest korelacja niestabilnej funkcji błędu z lepszymi efektami uczenia. Może to być spowodowane faktem, że takie błędy wprowadzają większą różnorodność podczas uczenia i wpływają pozytywnie na generalizację.
- Przy połączeniu wszystkich najlepszych hiperparametrów otrzymanych z eksperymentów, model uzyskuje dokładność >98%.
- Odpowiednie ustawienie hiperparametrów może znacząco przyspieszyć proces uczenia.
- Architektura sieci ma duży wpływ na jej możliwości i wydajność.