
Sieć wielowarstwowa Sieci Neuronowe 2020

Jakub Ciszek
238035

Spis treści

1	Opis badań	3
1.1	Plan eksperymentów	3
1.2	Charakterystyka zbiorów danych	3
2	Eksperymenty	4
2.1	Wpływ wielkości warstwy ukrytej na przebieg procesu uczenia	4
2.2	Wpływ wielkości paczki na przebieg procesu uczenia	8
2.3	Wpływ zakresu inicjalizacji wag na przebieg procesu uczenia	12
2.4	Wpływ wartości współczynnika alpha na przebieg procesu uczenia	16
2.5	Wpływ użytej funkcji aktywacyjnej na przebieg procesu uczenia	20
3	Wnioski	23

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 α 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

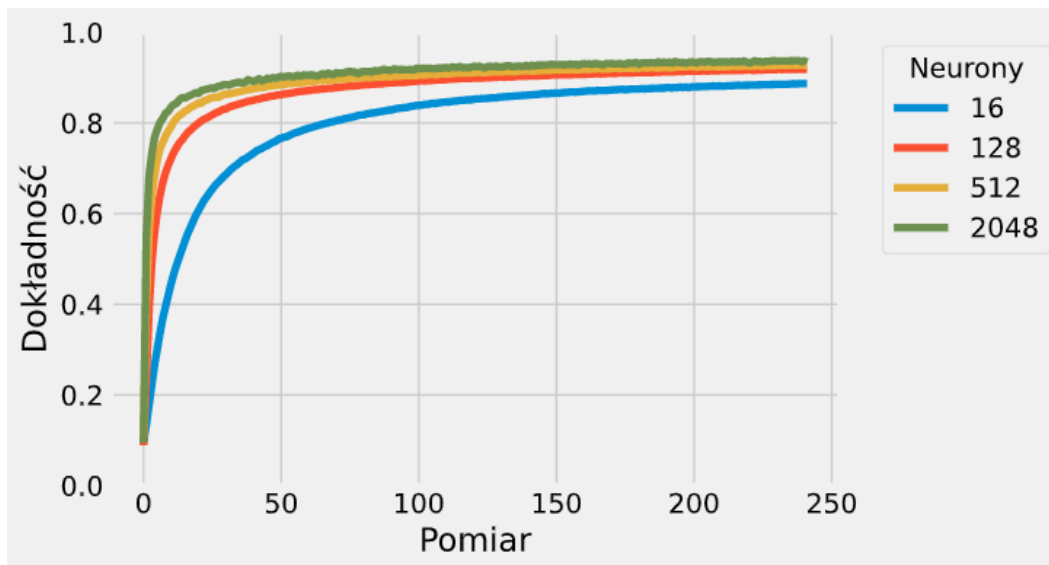
Zmienną w tym eksperymencie była wielkość warstwy ukrytej. Ilość neuronów przyjmowała wartości ze zbioru {16, 128, 512, 2048}

Przebieg

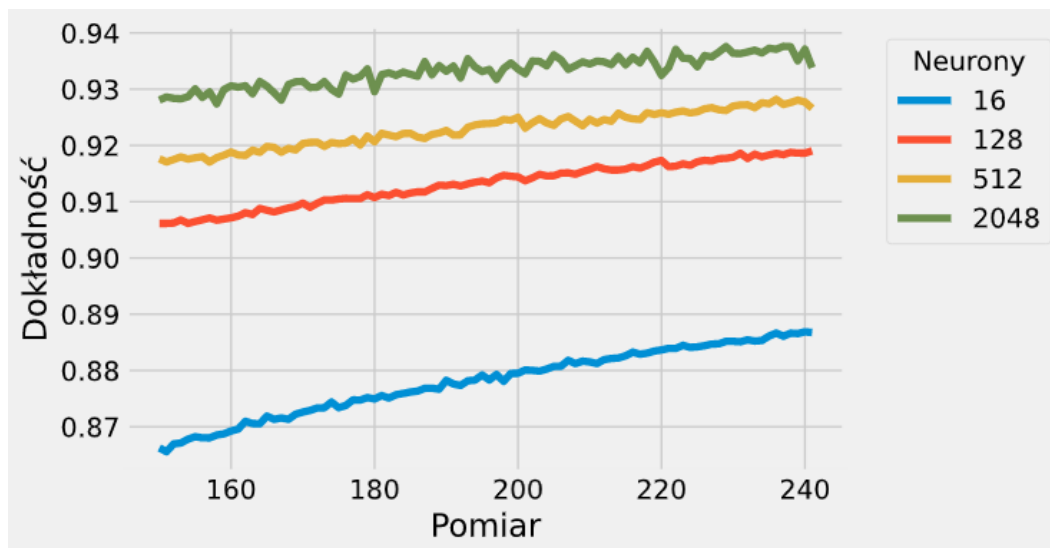
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy.

Wyniki

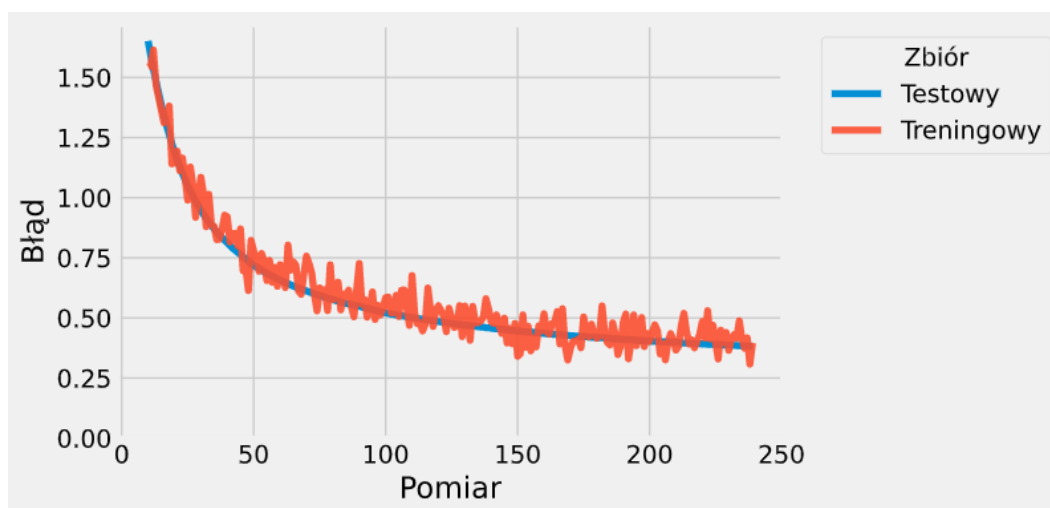
Wykres 1: Dokładność modelu w zależności od wielkości warstwy ukrytej



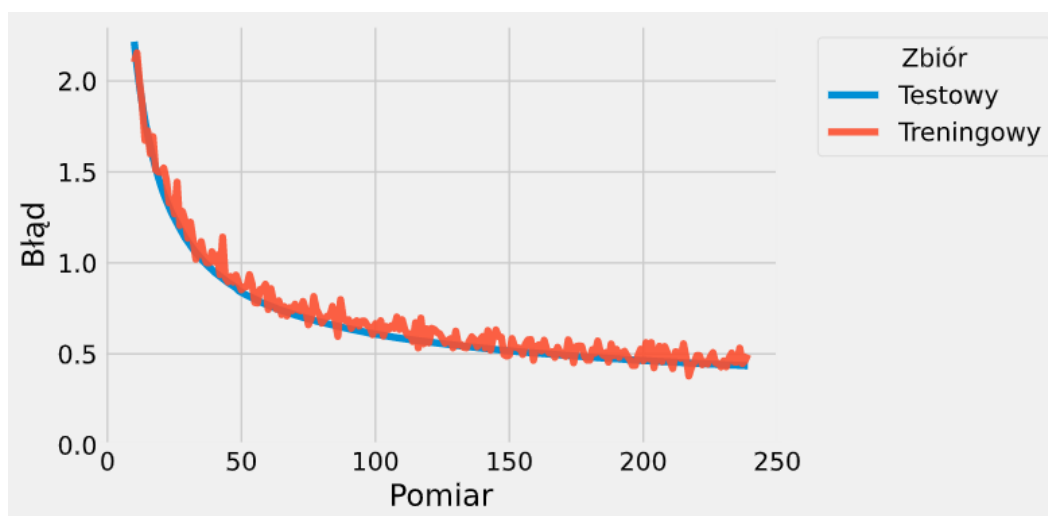
Wykres 2: Dokładność modelu w końcowym etapie uczenia w zależności od wielkości warstwy ukrytej



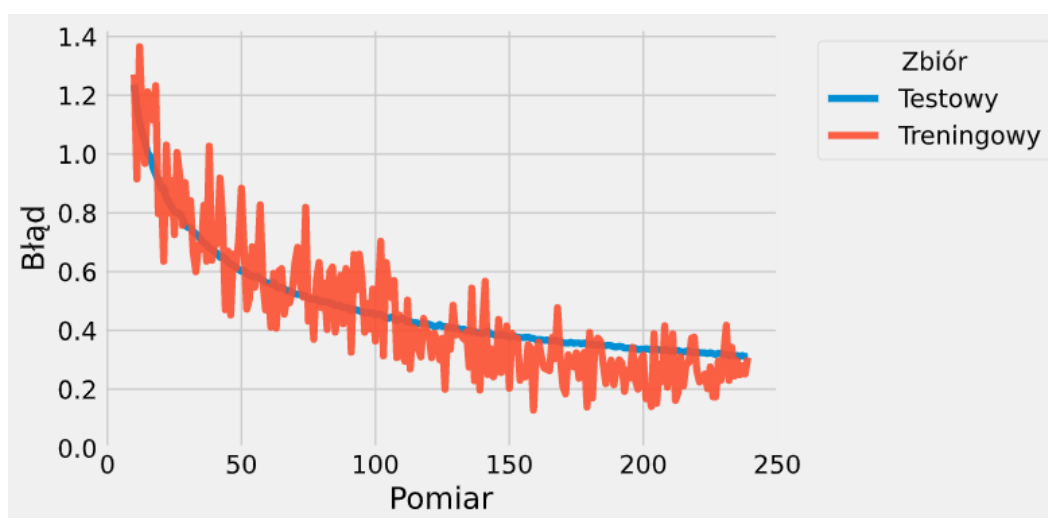
Wykres 3: Zachowanie funkcji błędu dla 16 neuronów



Wykres 4: Zachowanie funkcji błędu dla 128 neuronów



Wykres 5: Zachowanie funkcji błędu dla 512 neuronów



Wykres 6: Zachowanie funkcji błędu dla 2048 neuronów

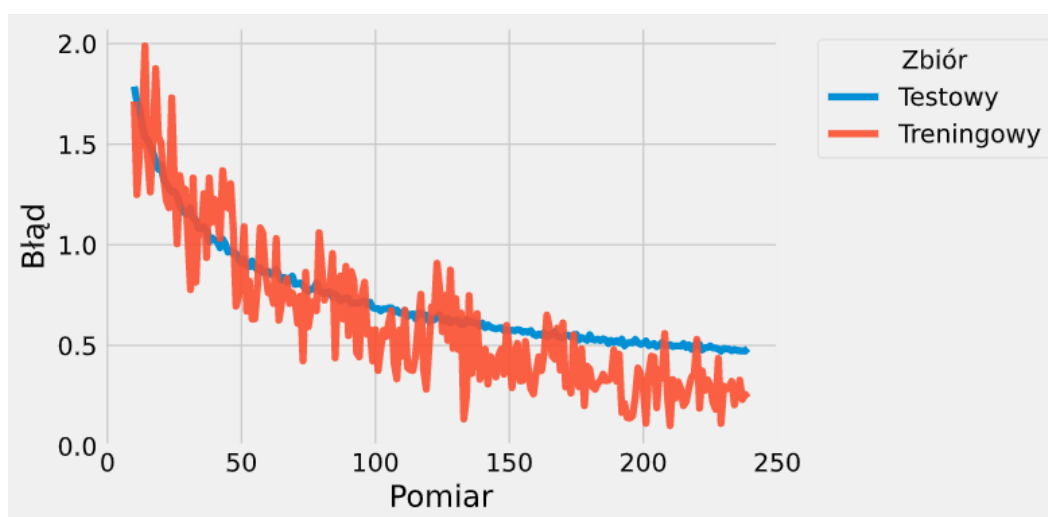


Tabela 2: Średnia maksymalna dokładność w zależności od wielkości warstwy ukrytej

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. Uzyskane wyniki dla wielkości 128, 512 i 2048 są do siebie zbliżone, widać jednak spadek dla mniejszej warstwy (16 neuronów). Najlepszą dokładność uzyskał model z warstwą wielkości 2048, jednak na wykresach 6 i 5 można zauważyć, że błąd treningowy dla tych wielkości spada prawie do 0 co może świadczyć o zbyttnim dopasowaniu do danych treningowych i predyspozycji do przeuczenia. W przeprowadzonych eksperymentach błąd testowy nie wzrastał w końcowych etapach uczenia, co wyklucza przeuczenie przy założonej ilości epok. Na wykresie 3 można natomiast zauważyć, że błędy treningowy i testowy dla małej warstwy ukrytej są na bardzo podobnym poziomie, w przeciwieństwie do pozostałych wielkości, gdzie błąd treningowy jest trochę niższy od testowego. Dodatkową obserwacją jest krótszy czas obliczeń przy uczeniu mniejszych modeli, spowodowany mniejszą ilością parametrów do zaktualizowania oraz przemnożenia podczas ewaluacji. TODO

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

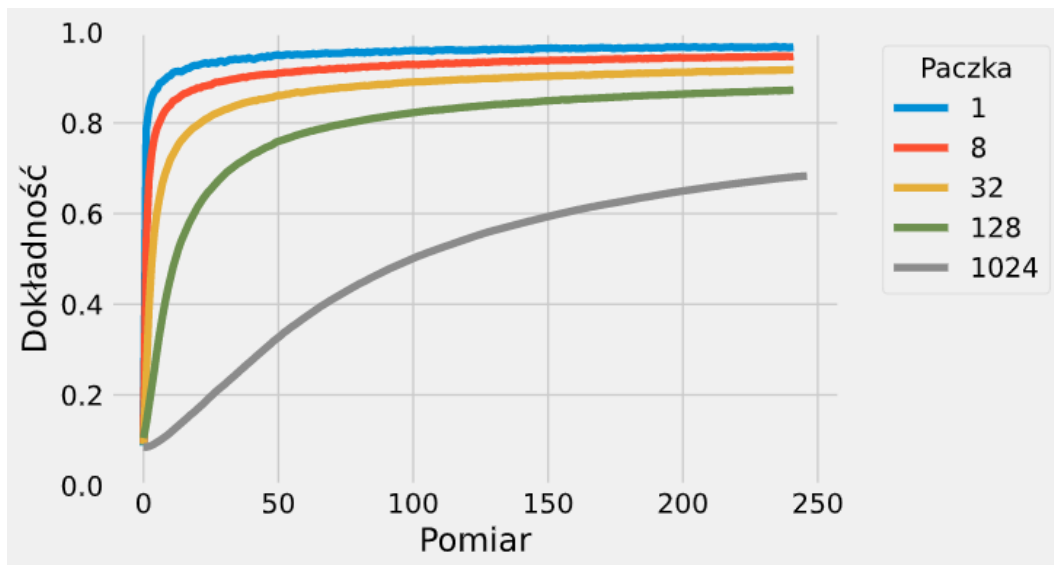
Zmienną w tym eksperymencie była wielkość paczki. Ilość przykładów przyjmowała wartości ze zbioru {1, 8, 32, 128, 1024}

Przebieg

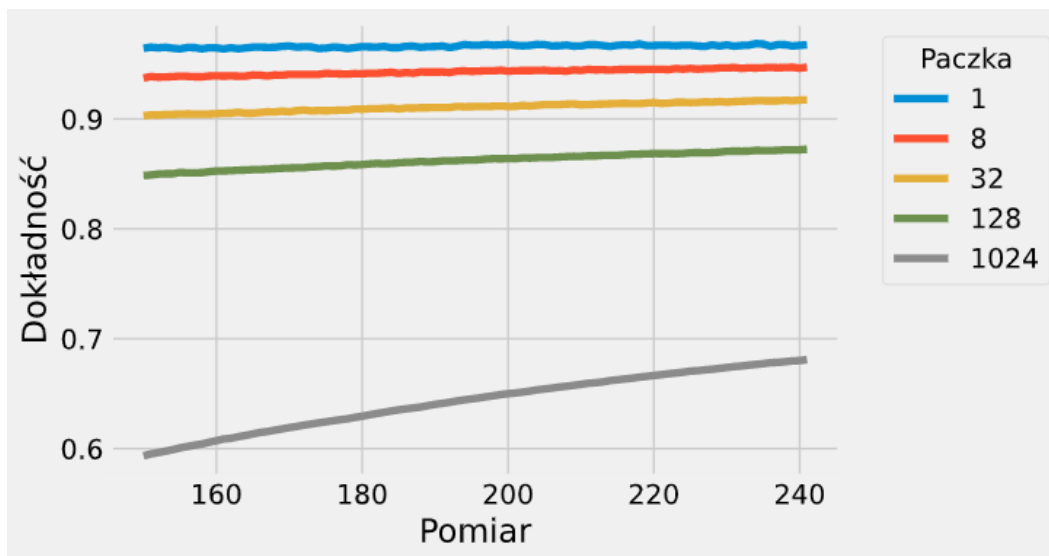
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy.

Wyniki

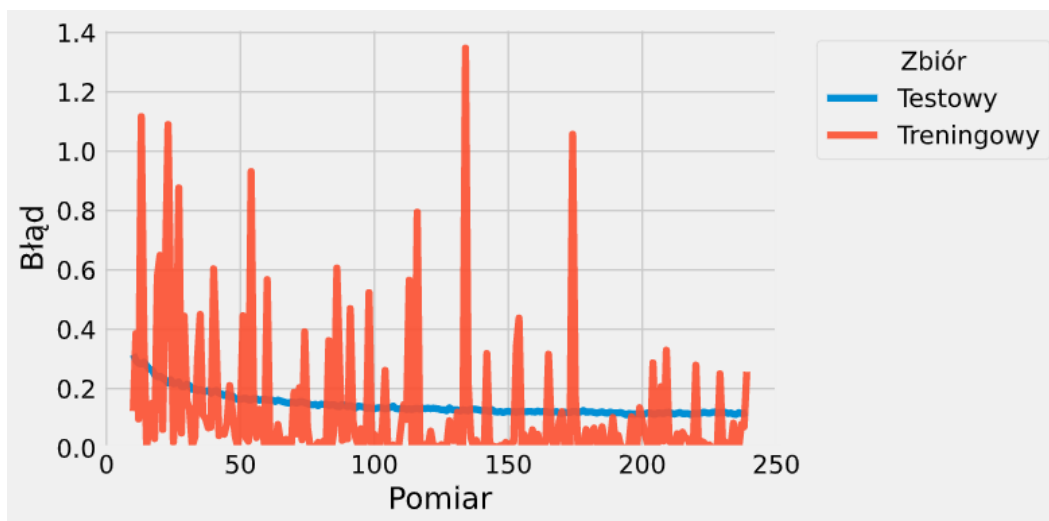
Wykres 7: Dokładność modelu w zależności od wielkości paczki



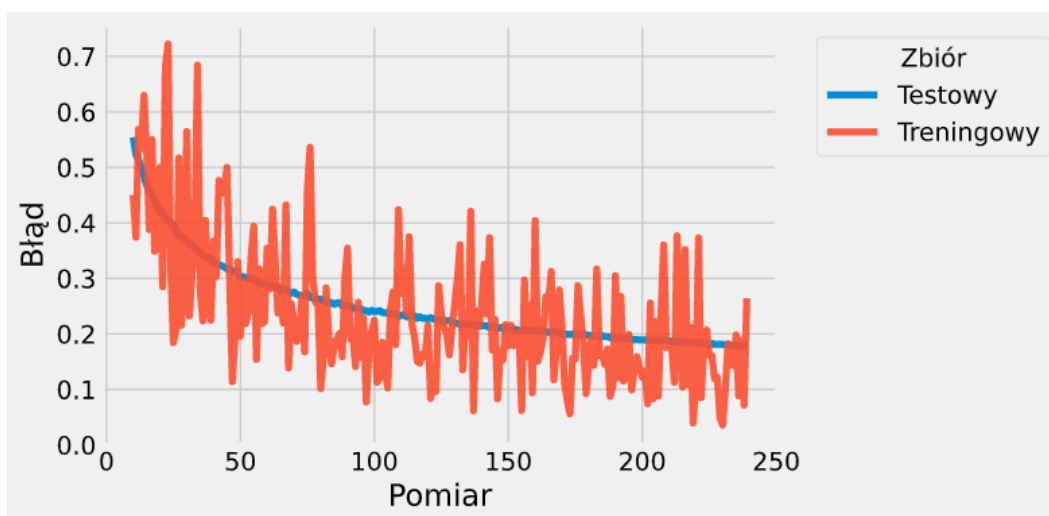
Wykres 8: Dokładność modelu w końcowym etapie uczenia w zależności od wielkości paczki



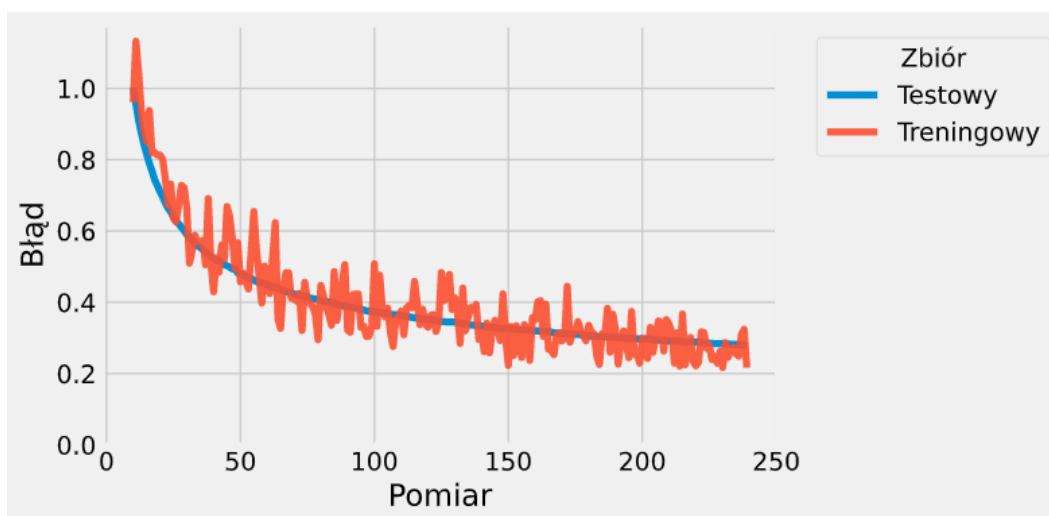
Wykres 9: Zachowanie funkcji błędu dla paczki wielkości 1



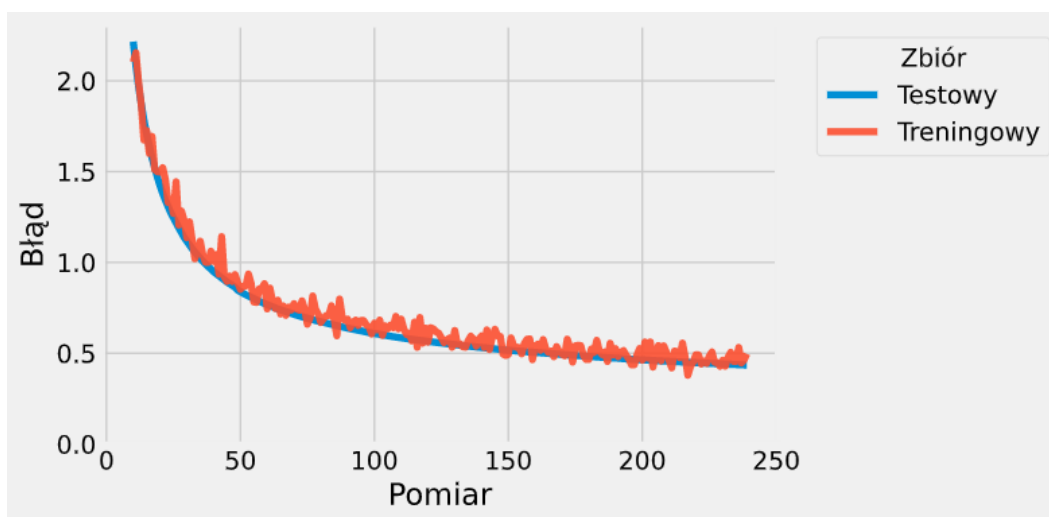
Wykres 10: Zachowanie funkcji błędu dla paczki wielkości 8



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

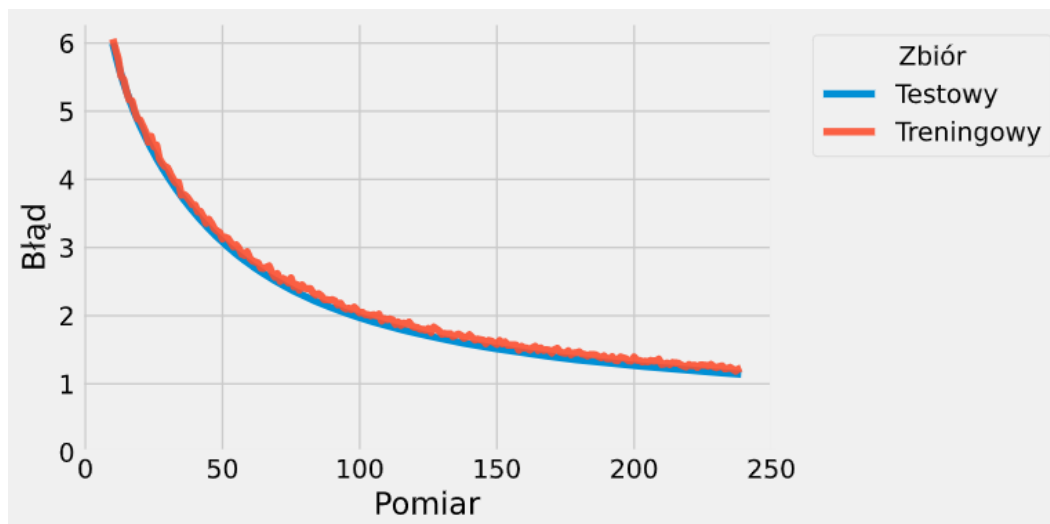


Tabela 4: Średnia maksymalna dokładność w zależności od wielkości paczki

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. Używanie za dużej paczki, jak widać w przypadku wielkości 1024, uniemożliwia wyuczenie. Po przeprowadzeniu dodatkowych badań okazało się jednak, że te wyniki są spowodowane przyjętym współczynnikiem uczenia, dla odpowiednio dobranych wartości α , mniejszych dla większych wielkości paczki, wyniki dla odpowiednio dobranych współczynników uczenia były zbliżone dla wszystkich rozmiarów. Zaletą mniejszych paczek, widoczną na wykresach 9, 10, 11, 12 i 13 jest wprowadzanie szumów do błęd treningowego, co przeciwdziała dążeniu do minimów lokalnych. Dodatkową obserwacją jest krótszy czas obliczeń dla większych paczek spowodowany mniejszą ilością przeprowadzonych propagacji dla takiego samego zbioru uczącego. TODO

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

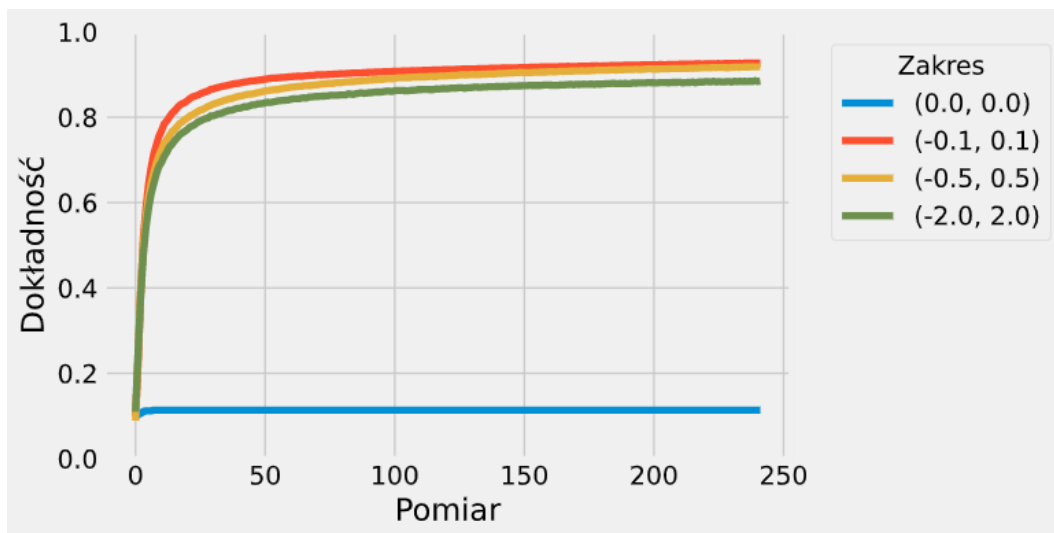
Zmienną w tym eksperymencie był zakres inicjalizacji wag. Przedział inicjalizacji przyjmował wartości ze zbioru $\{0.0, -0.1 - 0.1, -0.5 - 0.5, -2.0 - 2.0\}$

Przebieg

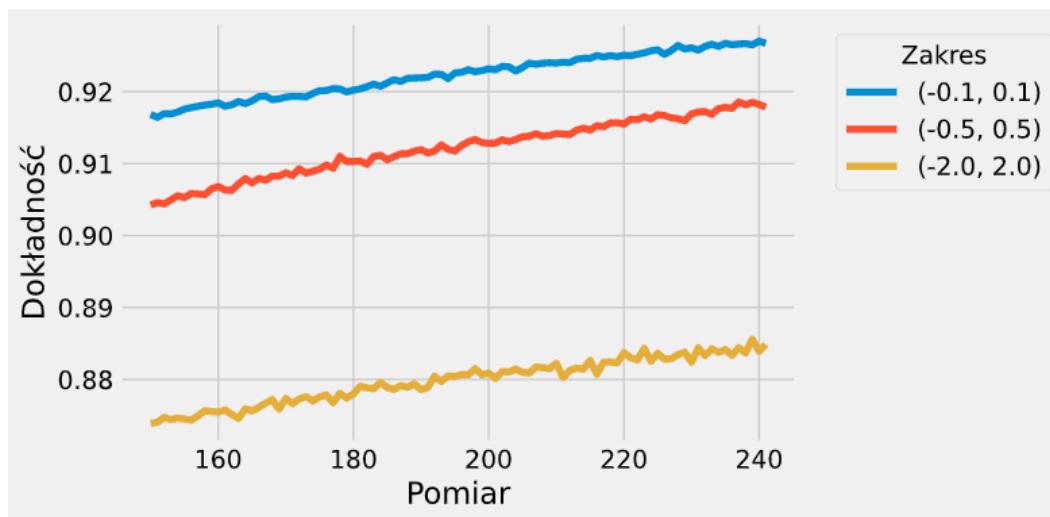
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy.

Wyniki

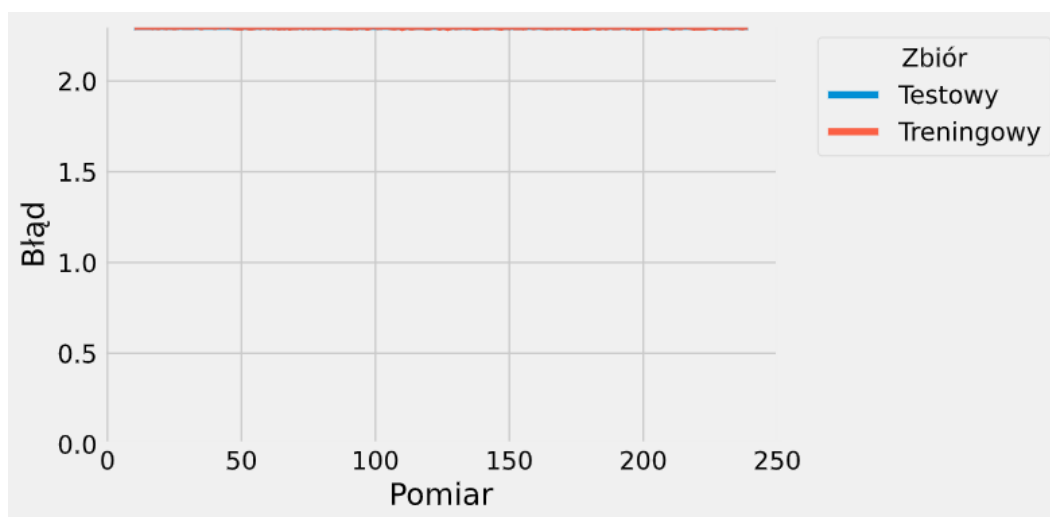
Wykres 14: Dokładność modelu w zależności od zakresu inicjalizacji wag



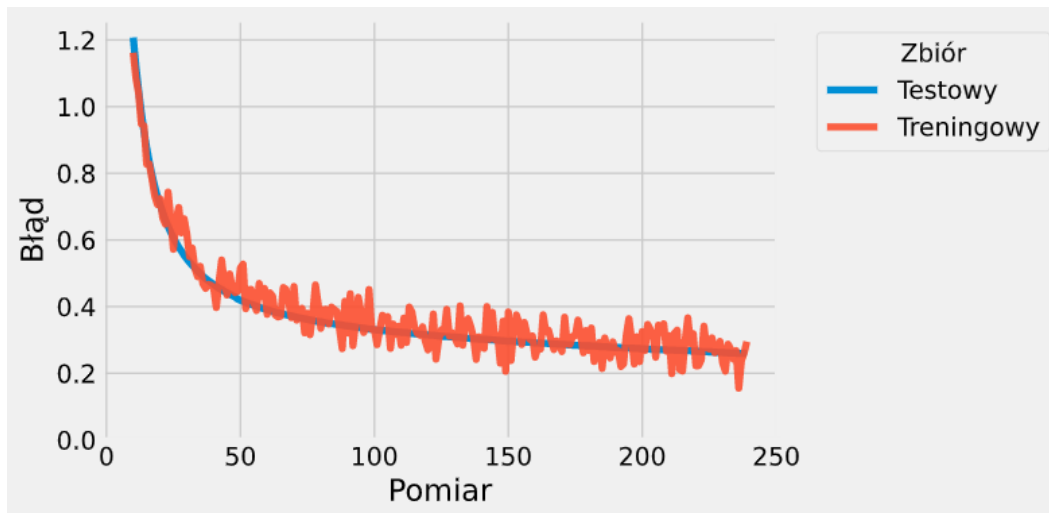
Wykres 15: Dokładność modelu w końcowym etapie uczenia w zależności od zakresu inicjalizacji wag



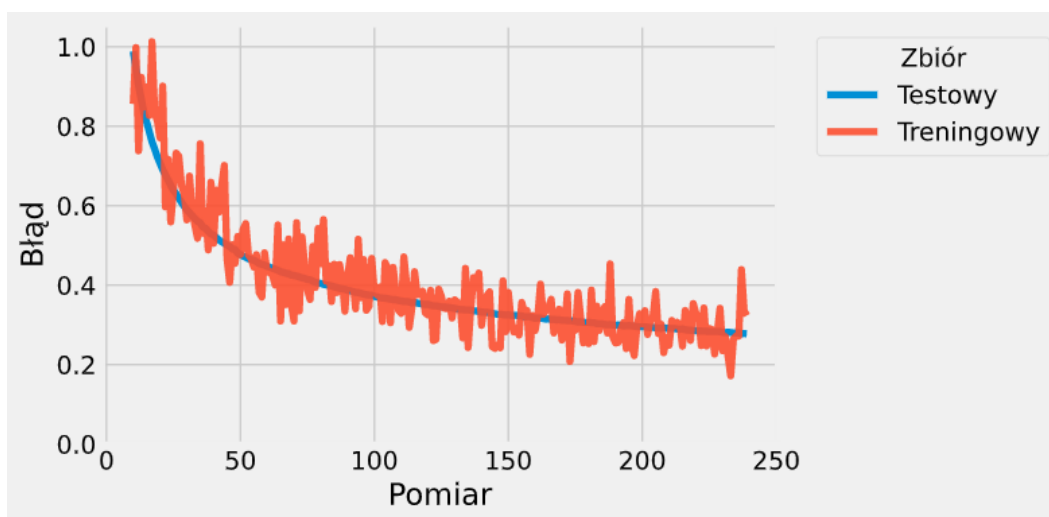
Wykres 16: Zachowanie funkcji błędu dla wag inicjalizowanych z zakresu 0.0 – 0.0



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

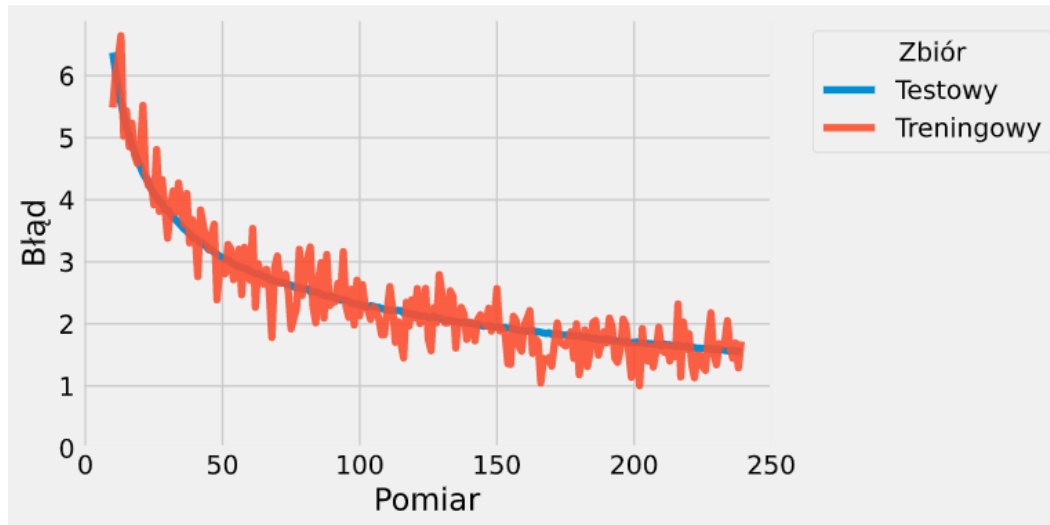


Tabela 6: Średnia maksymalna dokładność w zależności od zakresu inicjalizacji wag

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. Duże wartości powodują wolniejsze zbieganie oraz niższa dokładność końcową. TODO

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

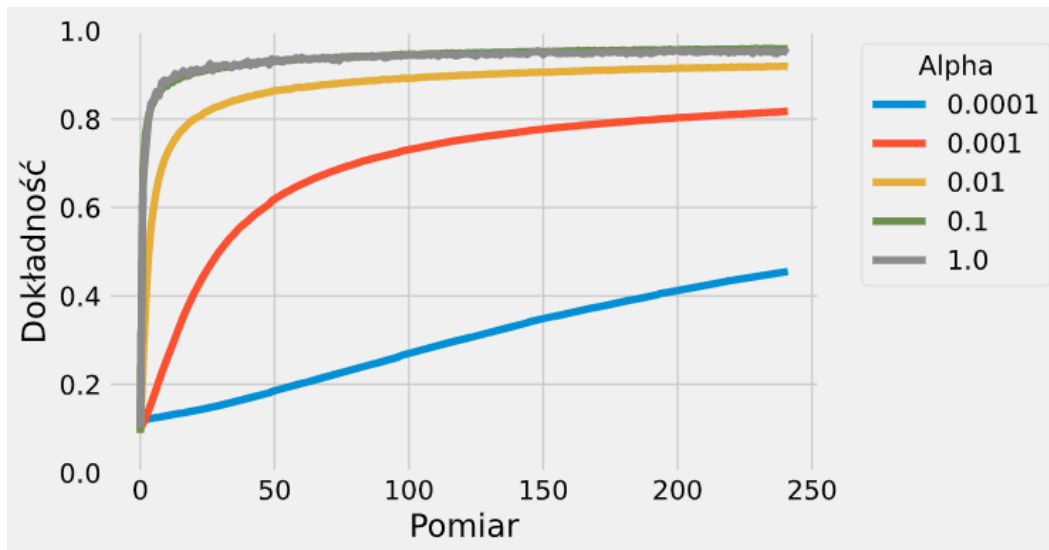
Zmienną w tym eksperymencie był współczynnik uczenia. Przyjmował wartości ze zbioru {0.0001, 0.001, 0.01, 0.1, 1.0,}

Przebieg

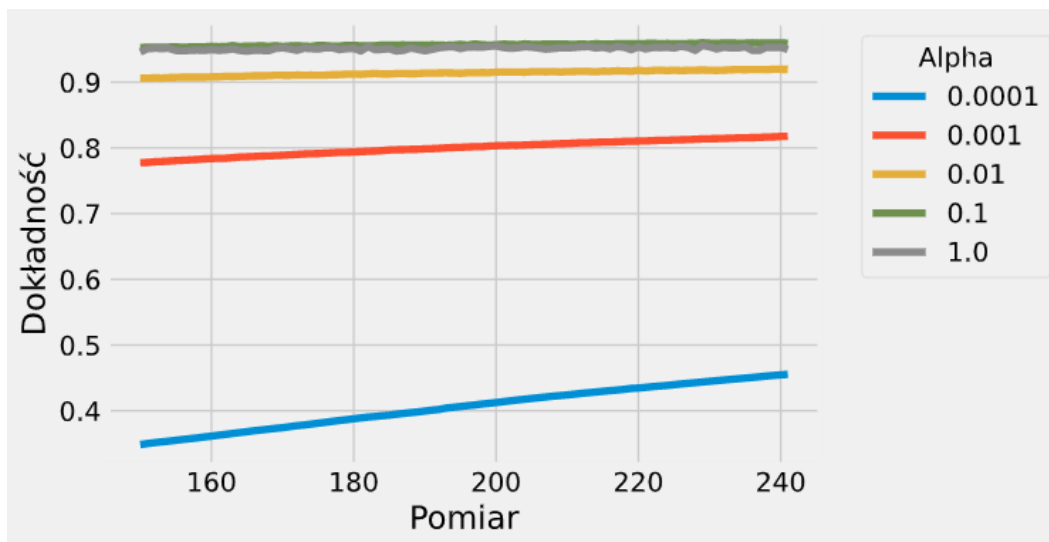
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy.

Wyniki

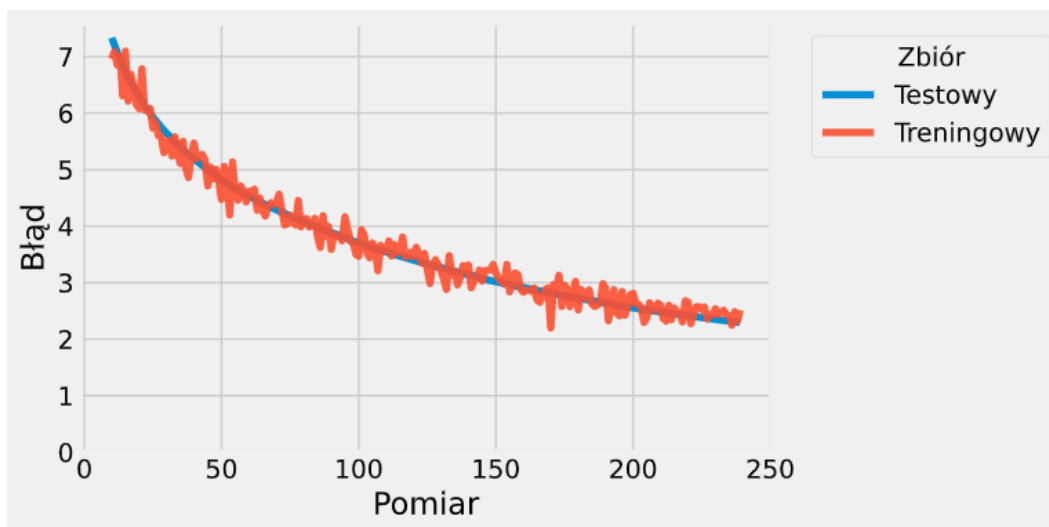
Wykres 20: Dokładność modelu w zależności od współczynnika uczenia



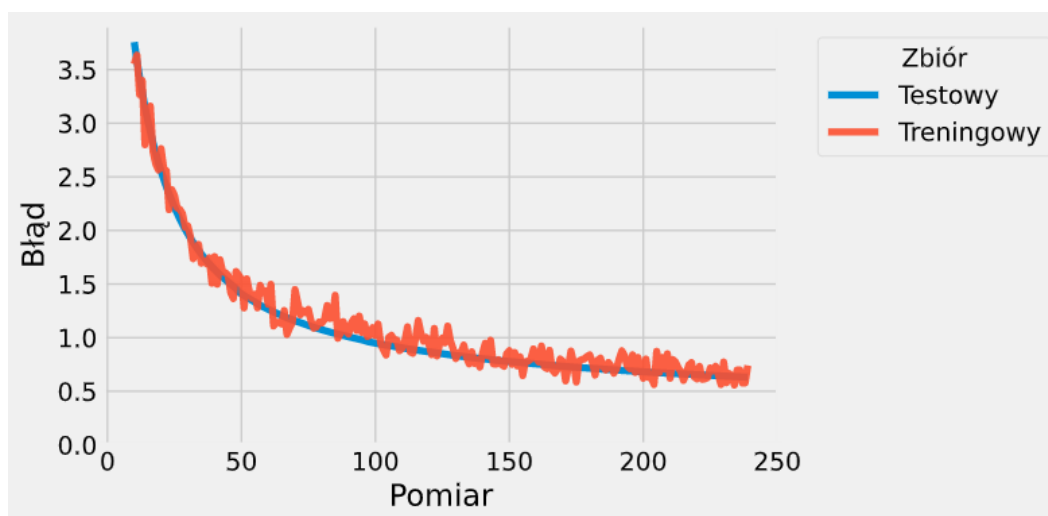
Wykres 21: Dokładność modelu w końcowym etapie uczenia w zależności od współczynnika uczenia



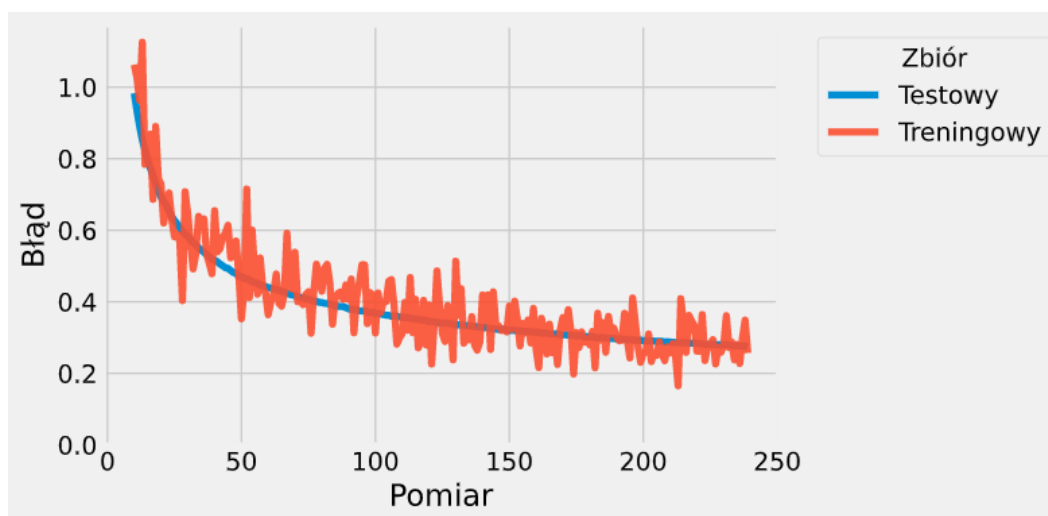
Wykres 22: Zachowanie funkcji błęd dla parametru alpha o wartości 0.0001



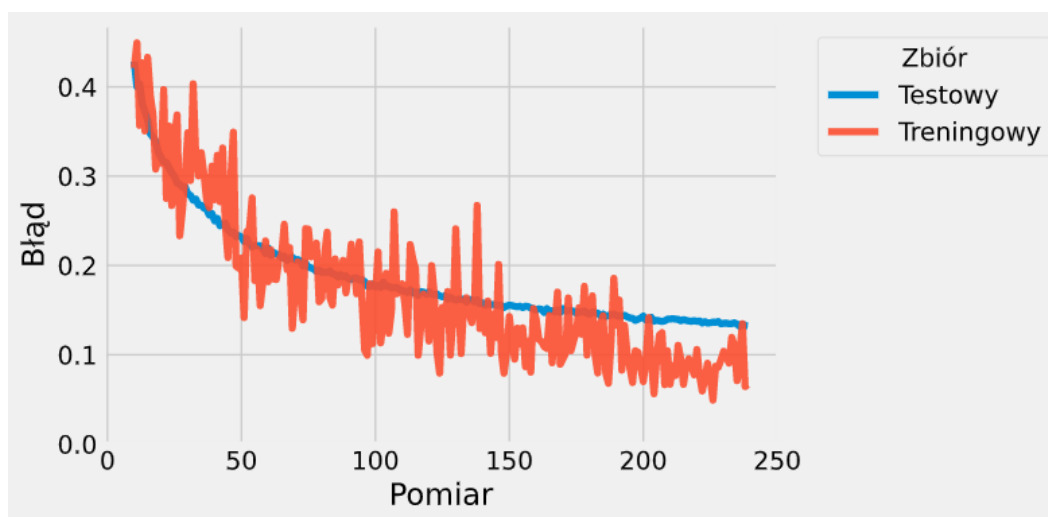
Wykres 23: Zachowanie funkcji błędu dla parametru alpha o wartości 0.001



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

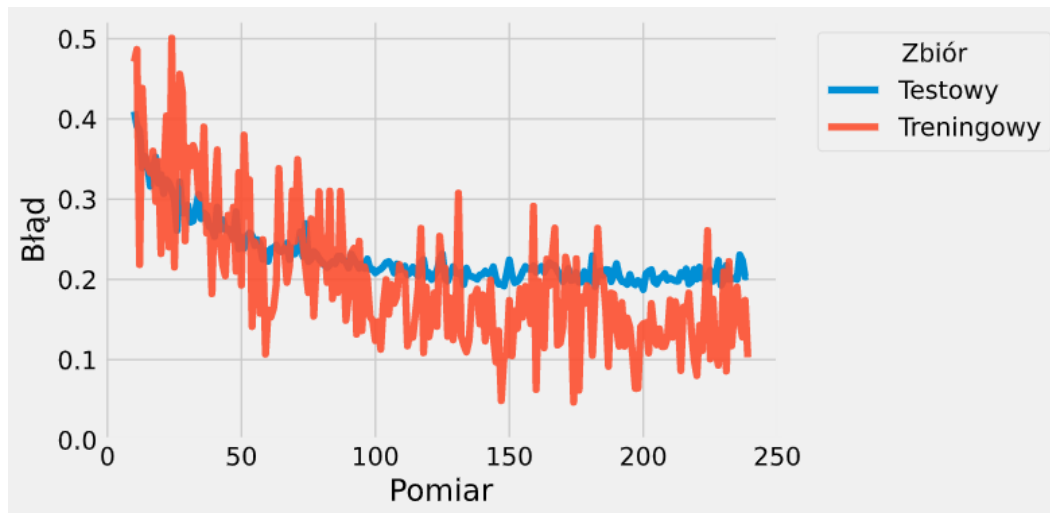


Tabela 8: Średnia maksymalna dokładność w zależności od współczynnika uczenia

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 uniemożliwiają wyuczenie, jest to najprawdopodobniej spowodowane przeskakiwaniem wszelkich minimów przy tak dużych aktualizacjach parametrów. Bardzo małe wartości alpha nałożone od samego początku uczenia skutkują natomiast bardzo powolnymi poprawami wyników i prawdopodobnie utykaniem w minimach lokalnych co przekłada się na niższą końcową dokładność modelu. Na wykresach 22, 23 i 24 widać, że mniejsze wartości alpha wygładzają błąd treningowy. TODO

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

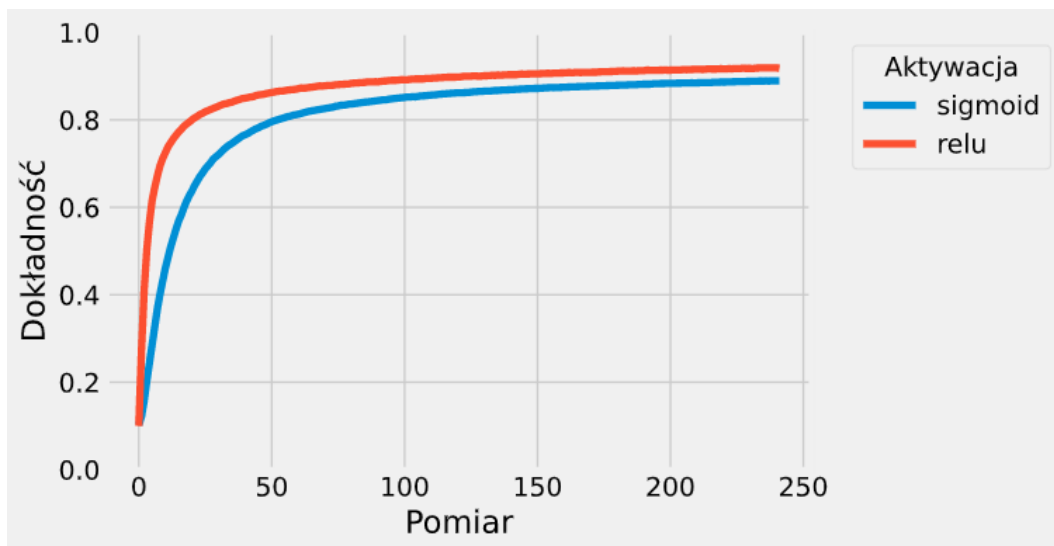
Zmienną w tym eksperymencie była funkcja aktywacji. Przetestowane zostały funkcje Sigmoidalna oraz ReLU.

Przebieg

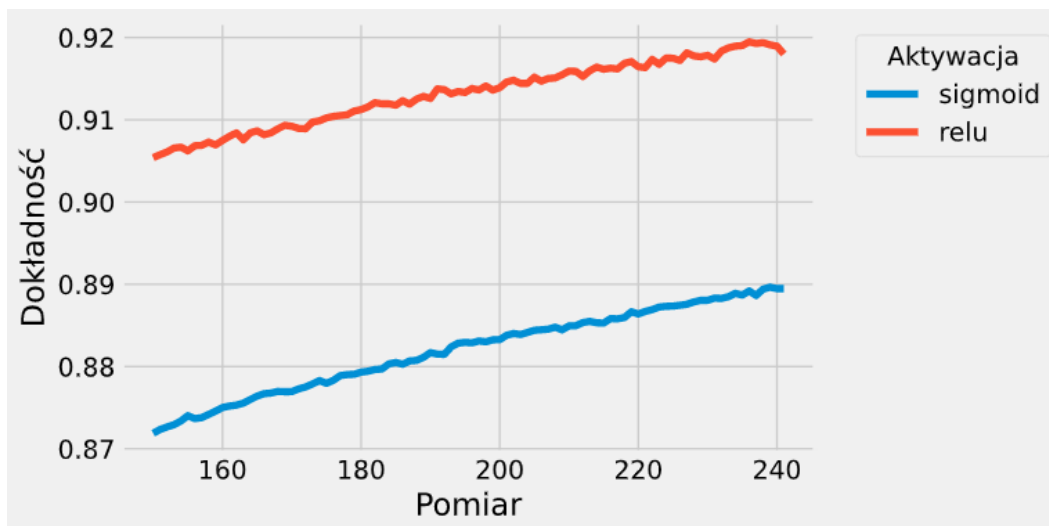
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy.

Wyniki

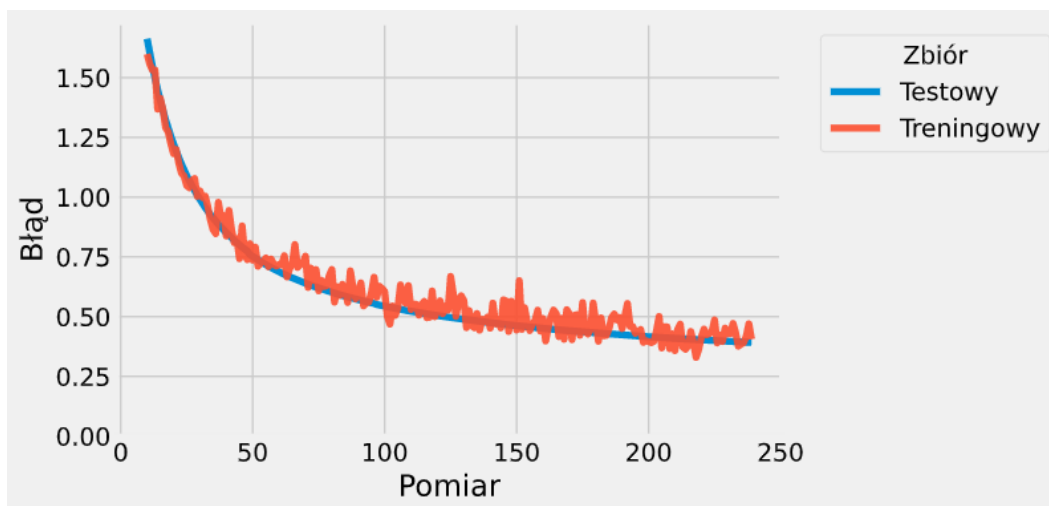
Wykres 27: Dokładność modelu w zależności od funkcji aktywacji



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

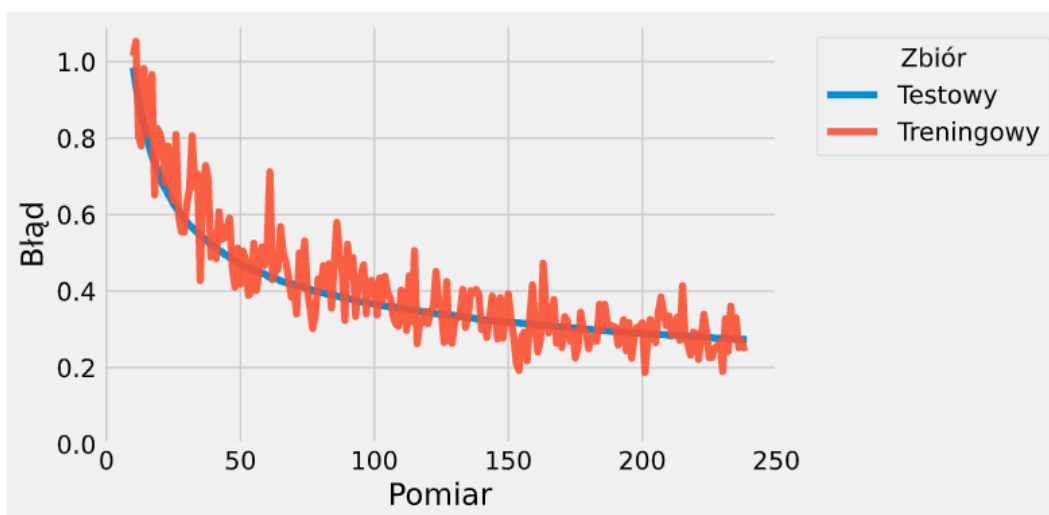


Tabela 10: Średnia maksymalna dokładność w zależności od funkcji aktywacji

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 lepsze wyniki. Używanie funkcji ReLU jest mniej kosztowne obliczeniowo, co dodatkowo przyspiesza trening. TODO

3 Wnioski

- Wielowarstwowe sieci neuronowe pozwalają na modelowanie nieliniowych funkcji.
- Powtarzającym się motywem była korelacja niższego błędu treningowego niż testowy dla lepszych konfiguracji hiperparametrów. Może to oznaczać po prostu lepsze dopasowanie do danych oraz to, że MNIST jest na tyle prostym zbiorem danych, że prawie zerowy błąd na zbiorze uczącym nie implikuje przeuczenia, lecz dobra generalizację.
- Przy połączeniu wszystkich najlepszych hiperparametrów otrzymanych z eksperymentów, model uzyskuje dokładność >98%.
- Hiperparametry wpływają na siebie nawzajem, podczas badań najbardziej wyraźny był wpływ współczynnika uczenia na możliwość używania większych wielkości paczki.
- Odpowiednie ustawienie hiperparametrów może znacząco przyspieszyć proces uczenia.
- Architektura sieci ma duży wpływ na jej możliwości i wydajność.
- TODO