Optymalizacja uczenia
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 optymalizatorów na przebieg procesu uczenia 4

2.2 Wpływ inicjalizacji wag na przebieg procesu uczenia 12

3 Wnioski **18**

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. Warstwa ukryta składała się z 512 neuronów, a początkowy współczynnik uczenia wynosił 0.01. Z powodów wydajnościowych testowanie modelu przeprowadzano co każde 32 paczki, z których każda składała się z 32 przykładów.
Zgodnie z instrukcją zostały przeprowadzone następujące badania:

- Wpływ optymalizatorów na przebieg procesu uczenia
- Wpływ inicjalizacji wag 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 optymalizatorów na przebieg procesu uczenia

Założenia

Tabela 1: Stałe dla eksperymentu 1

Parametr	Wartość
Inicjalizacja wag	-0.1 – 0.1

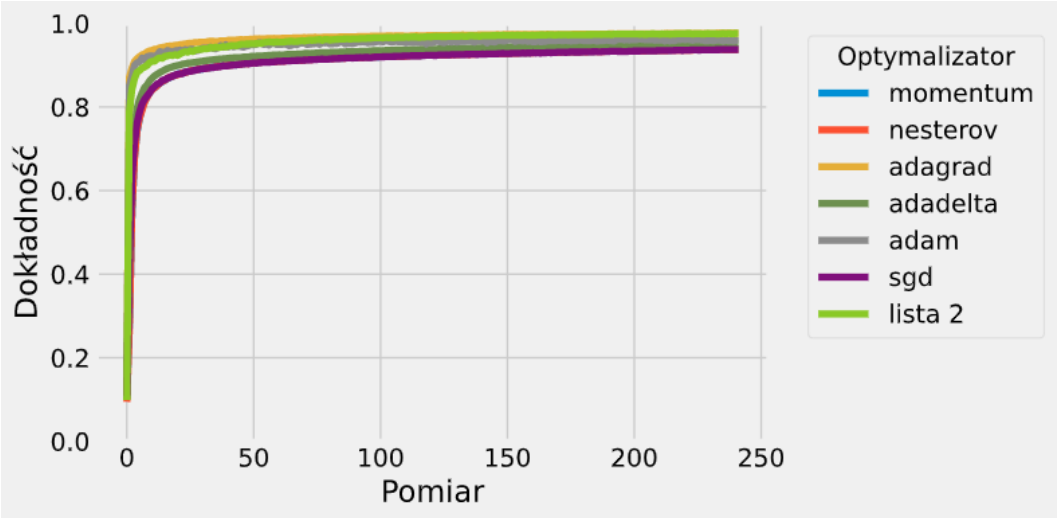
Zmienną w tym eksperymencie był użyty optymalizator uczenia. Użyto metod ze zbioru {SGD, Momentum, Nesterov, AdaGrad, AdaDelta, Adam}

Przebieg

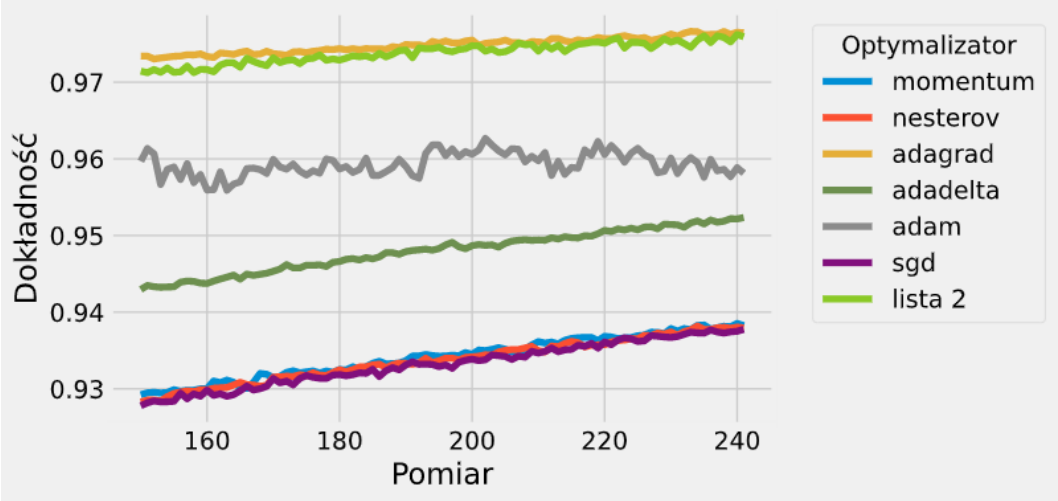
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy. Badania wykonano dla funkcji aktywacji Sigmoid oraz ReLU.

Wyniki

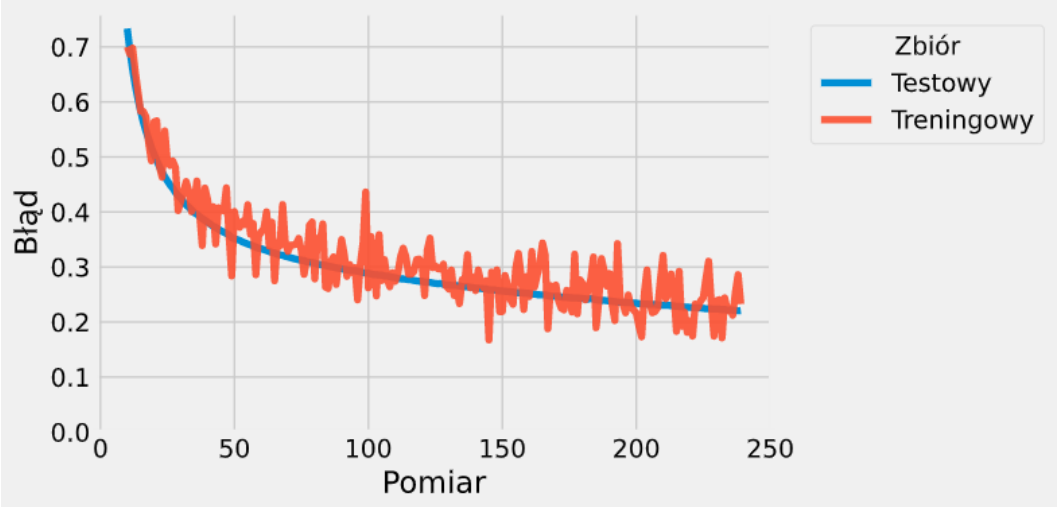
Wykres 1: Dokładność modelu w zależności od użytego optymalizatora dla funkcji ReLU



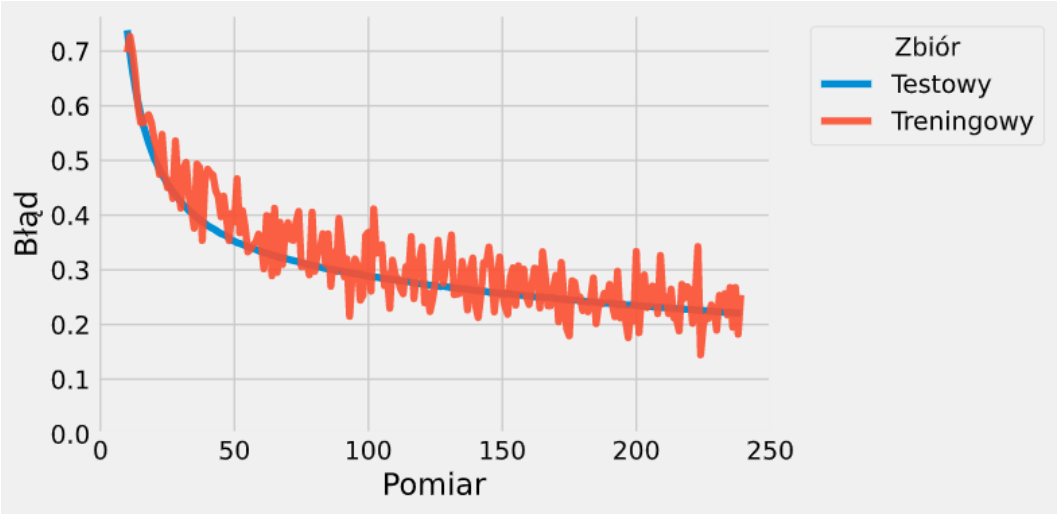
Wykres 2: Dokładność modelu w końcowym etapie uczenia w zależności od użytego optymalizatora dla funkcji ReLU



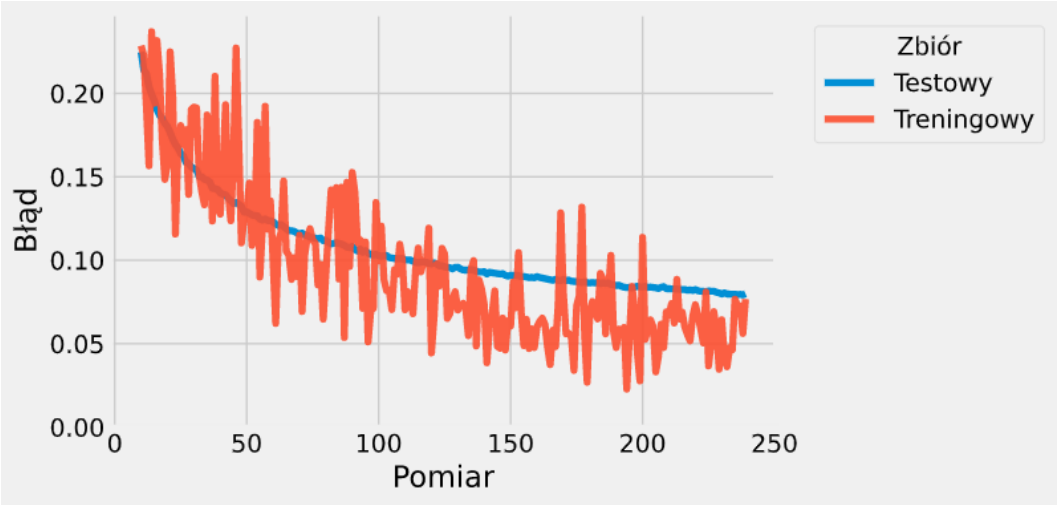
Wykres 3: Zachowanie funkcji błędu dla funkcji ReLU i optymalizatora Momentum



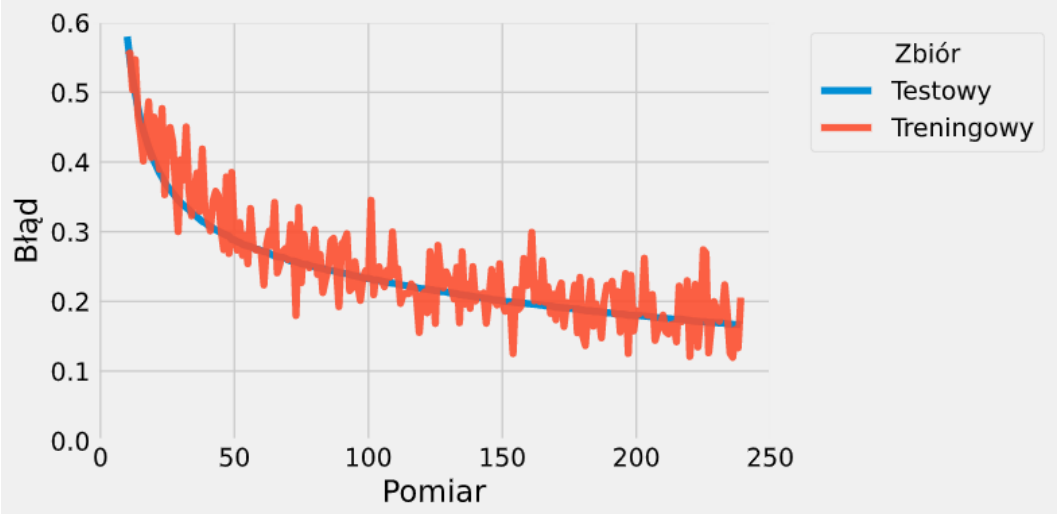
Wykres 4: Zachowanie funkcji błędu dla funkcji ReLU i optymalizatora Nesterov



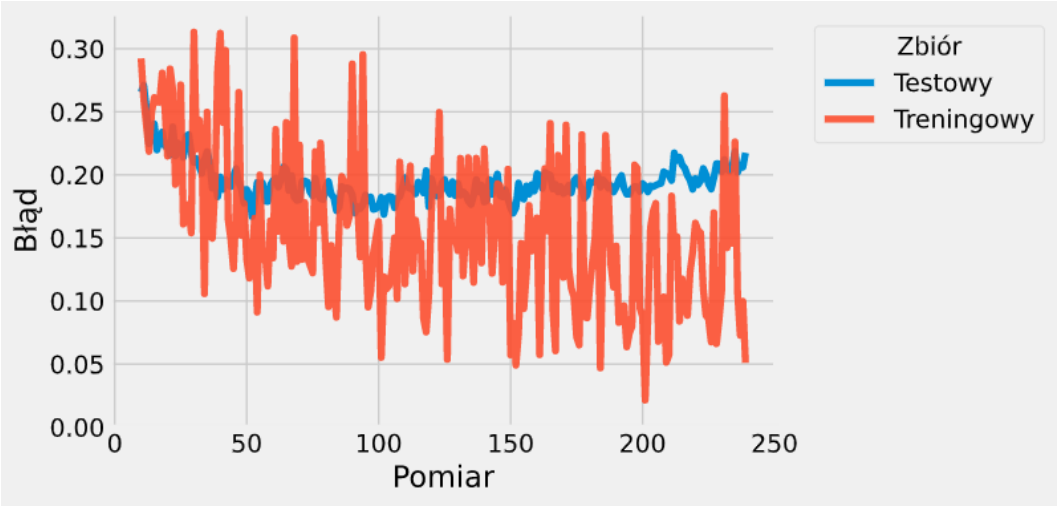
Wykres 5: Zachowanie funkcji błędu dla funkcji ReLU i optymalizatora AdaGrad



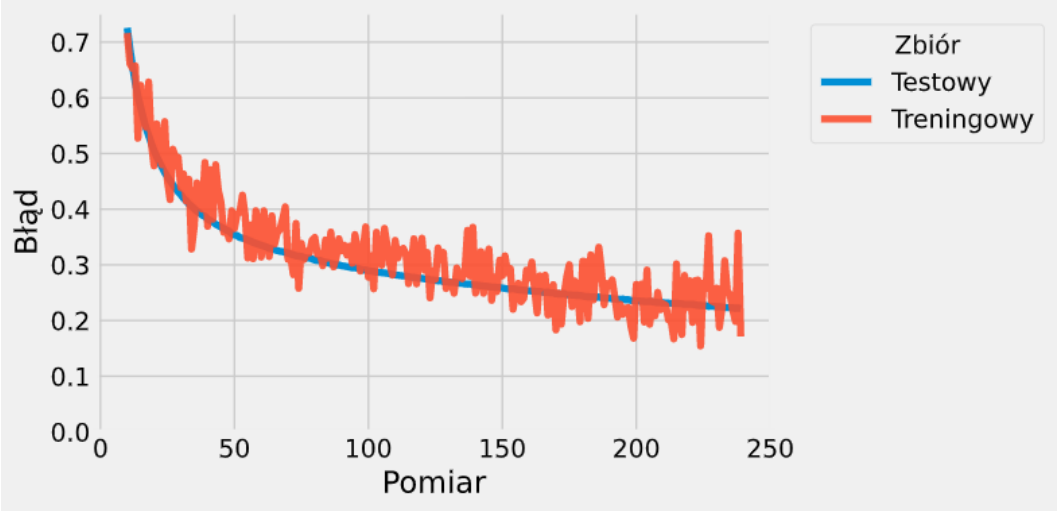
Wykres 6: Zachowanie funkcji błędu dla funkcji ReLU i optymalizatora AdaDelta



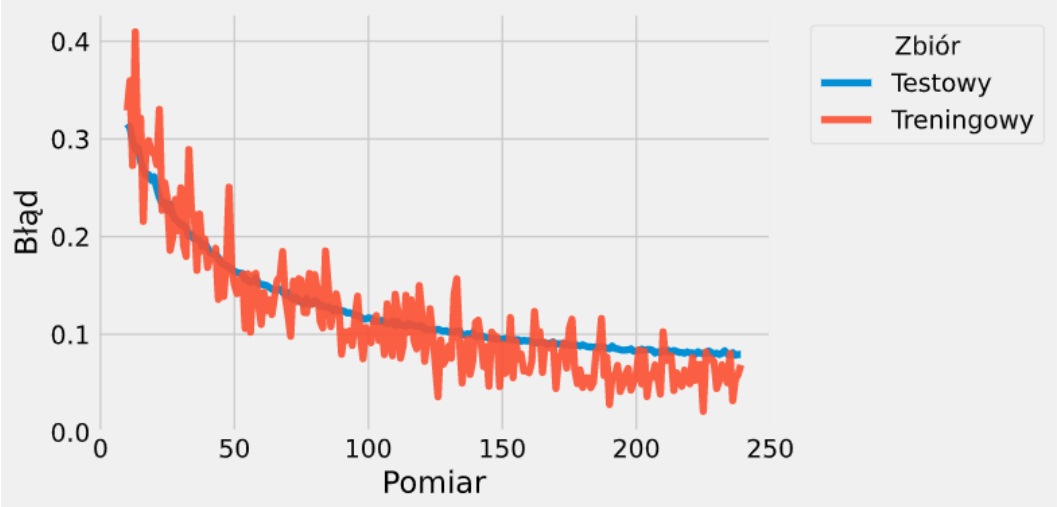
Wykres 7: Zachowanie funkcji błędu dla funkcji ReLU i optymalizatora Adam



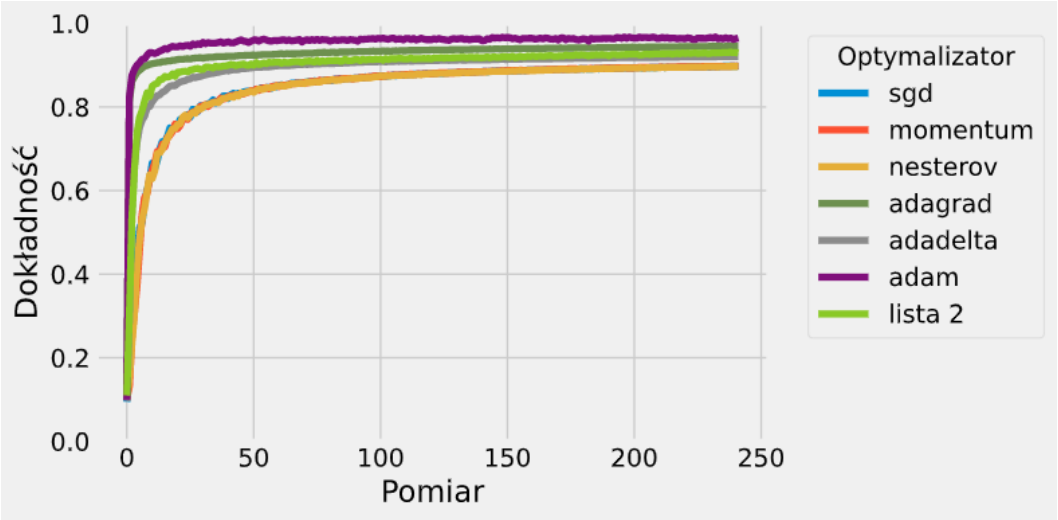
Wykres 8: Zachowanie funkcji błędu dla funkcji ReLU i optymalizatora SGD



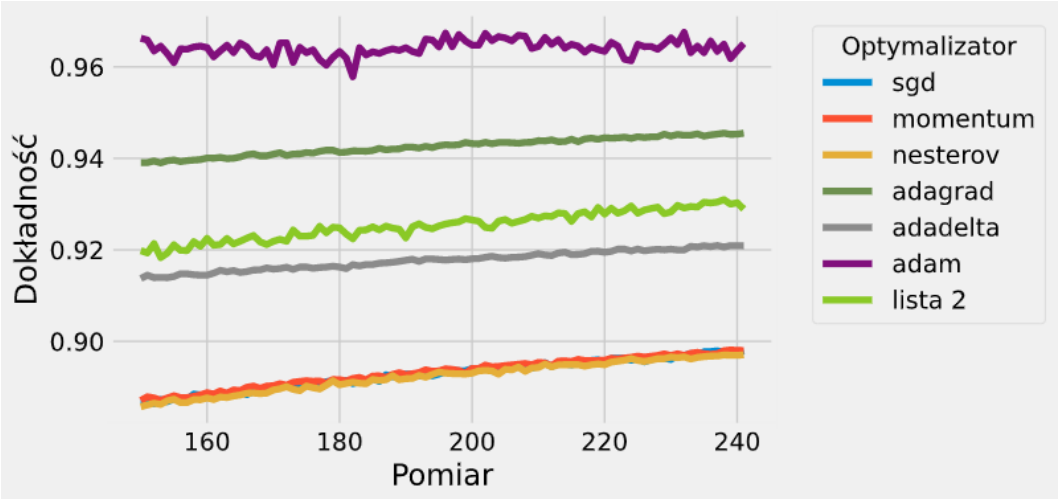
Wykres 9: Zachowanie funkcji błędu dla funkcji ReLU z listy 2



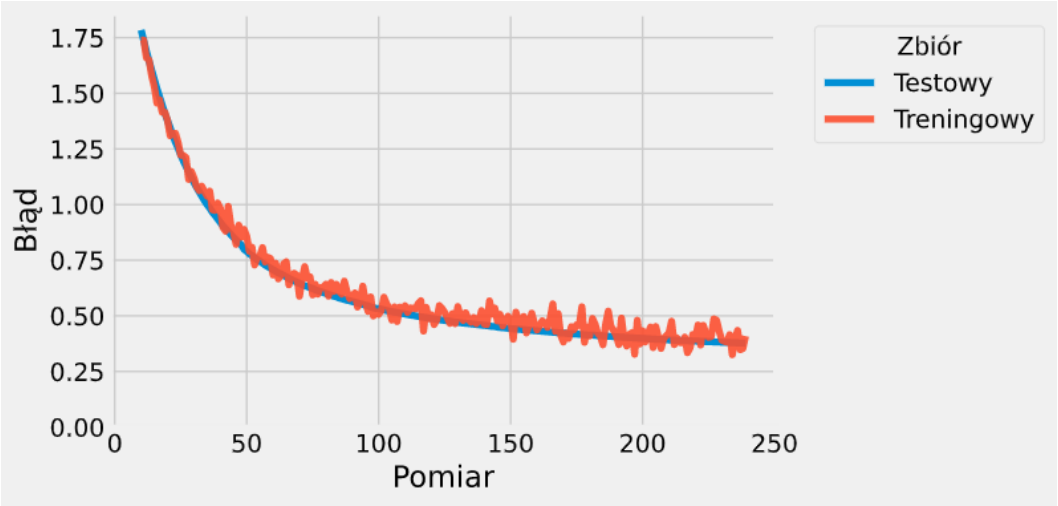
Wykres 10: Dokładność modelu w zależności od użytego optymalizatora dla funkcji Sigmoid



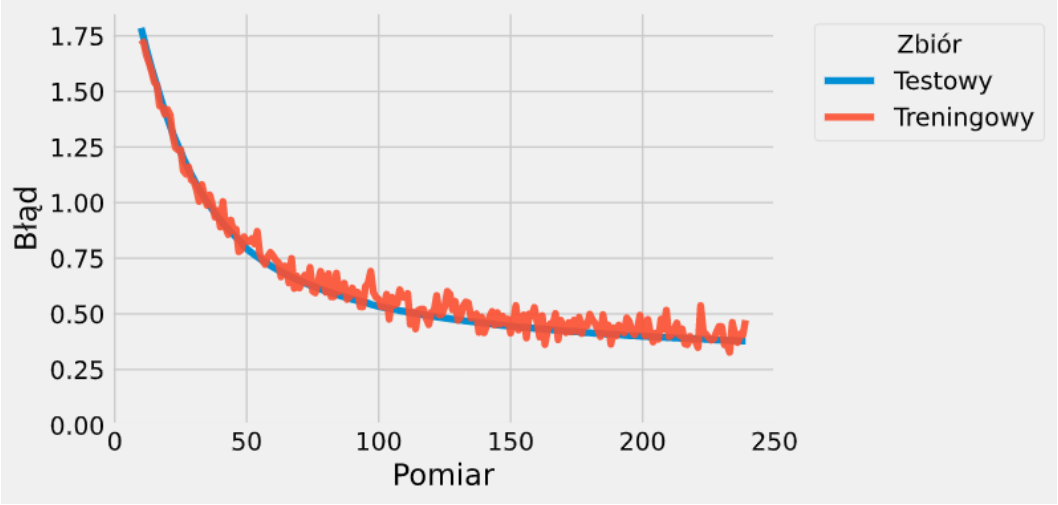
Wykres 11: Dokładność modelu w końcowym etapie uczenia w zależności od użytego optymalizatora dla funkcji Sigmoid



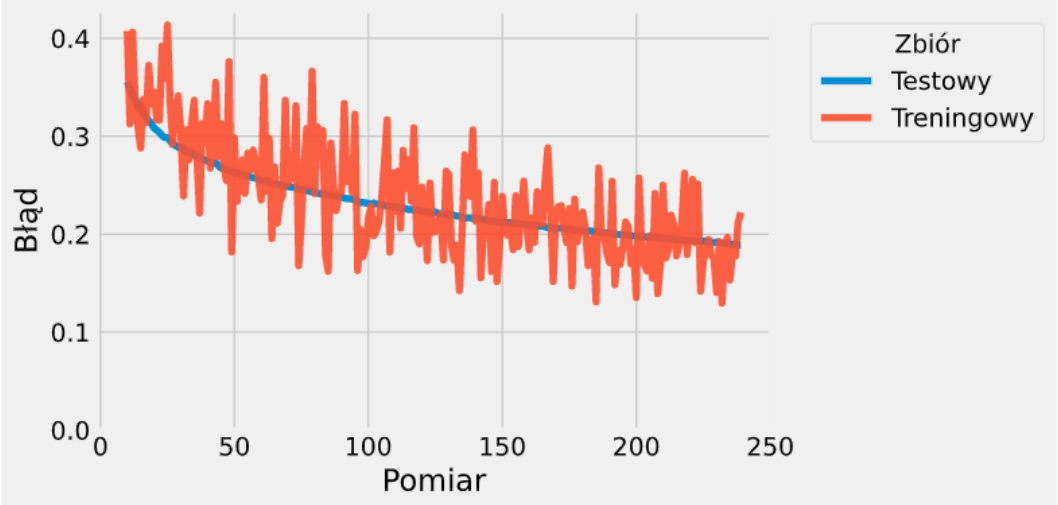
Wykres 12: Zachowanie funkcji błędu dla funkcji Sigmoid i optymalizatora Momentum



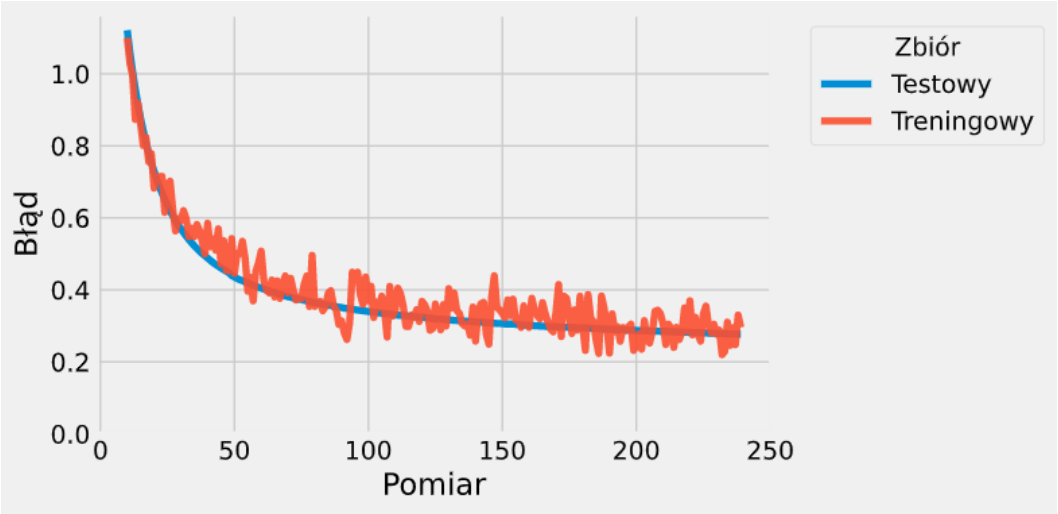
Wykres 13: Zachowanie funkcji błędu dla funkcji Sigmoid i optymalizatora Nesterov



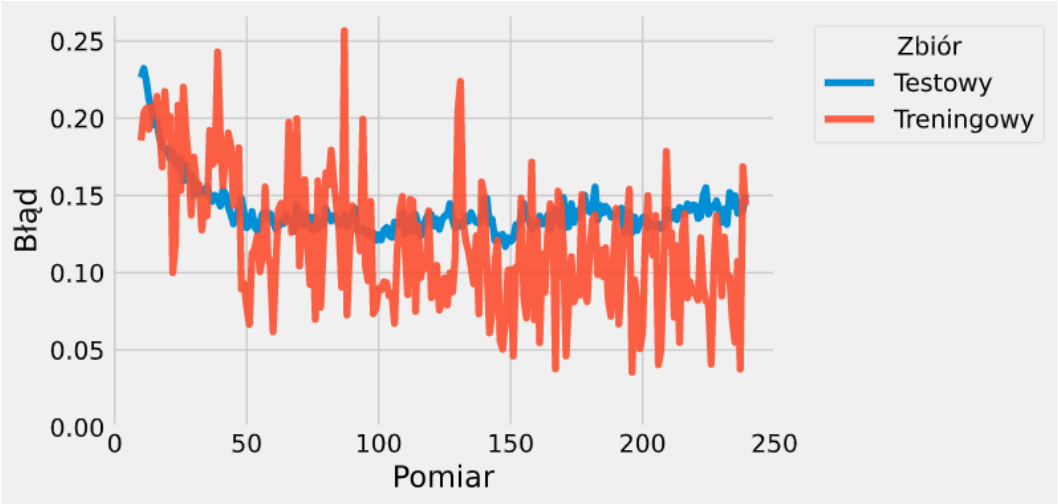
Wykres 14: Zachowanie funkcji błędu dla funkcji Sigmoid i optymalizatora AdaGrad



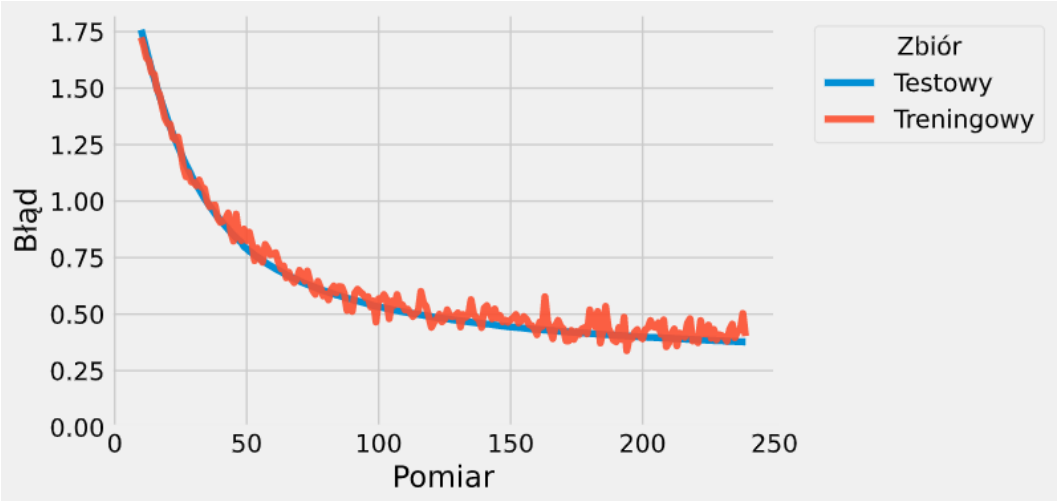
Wykres 15: Zachowanie funkcji błędu dla funkcji Sigmoid i optymalizatora AdaDelta



Wykres 16: Zachowanie funkcji błędu dla funkcji Sigmoid i optymalizatora Adam



Wykres 17: Zachowanie funkcji błędu dla funkcji Sigmoid i optymalizatora SGD



Wykres 18: Zachowanie funkcji błędu dla funkcji Sigmoid z listy 2

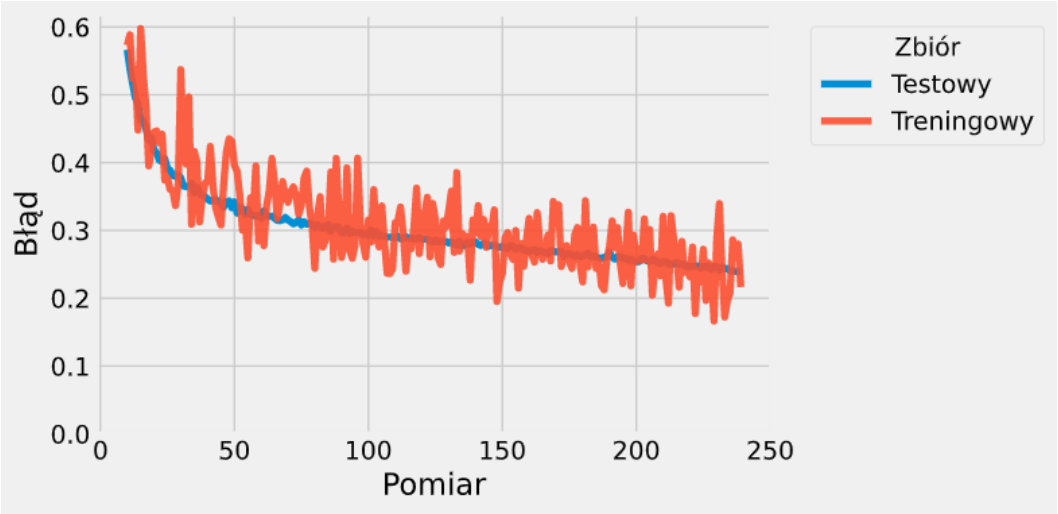


Tabela 2: Średnia maksymalna dokładność w zależności od użytego optymalizatora

Optymalizator	Dokładność [%]	
	ReLU	Sigmoid
Momentum	93.92	89.91
Nesterov	93.91	89.81
AdaGrad	**97.72**	94.62
AdaDelta	95.28	92.21
Adam	93.86	**97.15**
SGD	93.86	89.89
Lista 2	97.70	93.29

Wnioski

Na przedstawionych wykresach 1, 2 oraz tabeli 2 można zauważyć, że najlepszym wyborem przy użyciu funkcji ReLU był optymalizator AdaGrad, kolejne miejsce osiągnął Adam. Patrząc dodatkowo na wykresy 10 i 11, widać, że dla funkcji Sigmoid sytuacja jest odwrotna, najlepszy wynik otrzymują modele uczone metodą Adam, a AdaGrad jest na drugim miejscu. Powodem prowadzenia tych metod może być widoczna na wykresach 3 - 9 i 12 - 18 większa niestabilność błędu treningowego dla tych metod, co może oznaczać lepszą eksplorację minimów lokalnych. Porównując jednak funkcje błędu metod Adam i AdaGrad widać, że w przypadku optymalizatora Adam błąd testowy w pewnym momencie zaczyna rosnąć, co może wskazywać przeuczenie. Nie jest to jednak problemem na zbiorze MNIST. W porównaniu z modelem otrzymanym w poprzednim zadaniu konfiguracja z funkcją ReLU przy użyciu metody AdaGrad daje bardzo zbliżone wyniki, jednak AdaGrad ma przewagę w początkowym etapie uczenia gdzie szybciej osiąga wysoką dokładność, na co wpływa dynamiczny współczynnik dla każdej z wag. W przypadku modelu z funkcją Sigmoidalną, poprzedni model znajduje się dopiero na 3 miejscu, spowodowane jest to najprawdopodobniej dobraniem w poprzednim zadaniu hiperparametrów dla najlepszego wyniku z funkcją ReLU.

2.2 Wpływ inicjalizacji wag na przebieg procesu uczenia

Założenia

Tabela 3: Stałe dla eksperymentu 2

Parametr	Wartość
Optymalizator	SGD

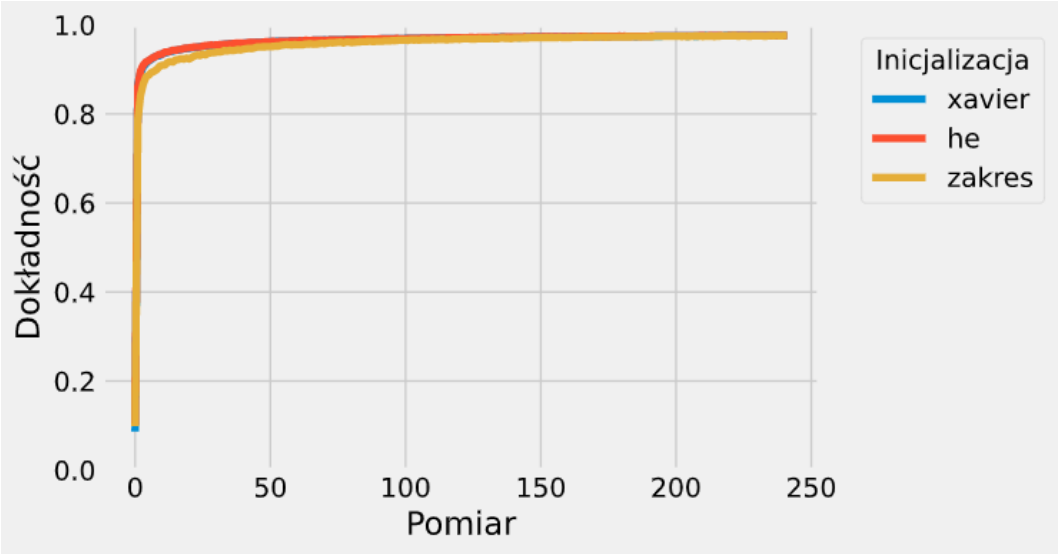
Zmienną w tym eksperymencie był sposób inicjalizacji wag. Użyto metod ze zbioru {Zakres, Xavier, He}. Zakres, z którego losowane były wagi to -0.1 – 0.1.

Przebieg

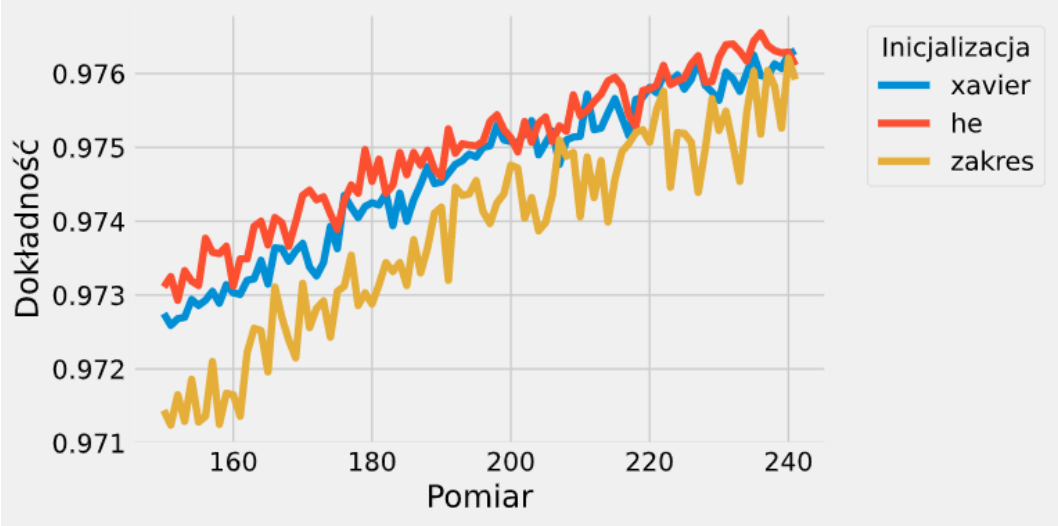
Podczas eksperymentu model został zainicjalizowany 10 razy dla każdej z badanych wartości oraz wyuczony, uzyskane wyniki zostały zapisane w postaci pliku .plk do dalszej analizy. Badania wykonano dla funkcji aktywacji Sigmoid oraz ReLU.

Wyniki

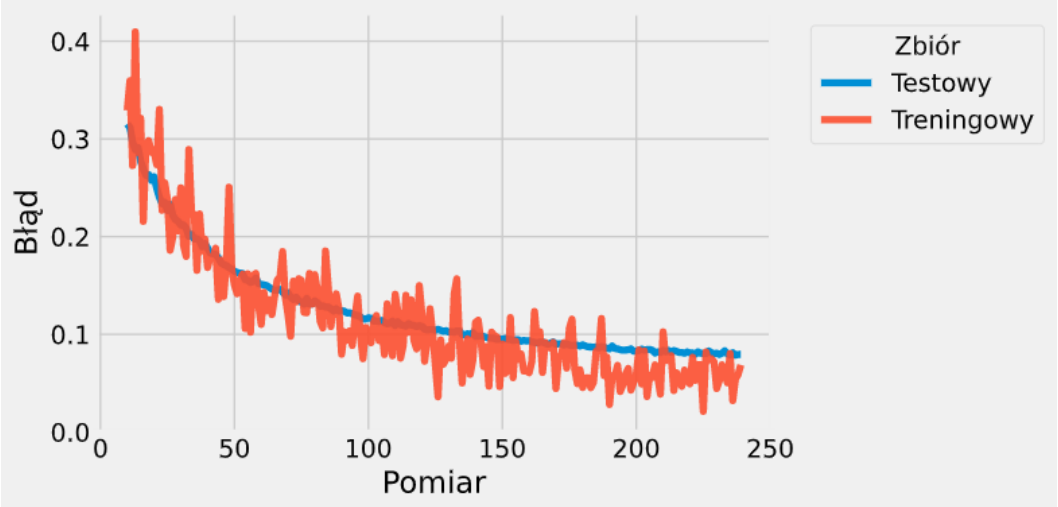
Wykres 19: Dokładność modelu w zależności od sposobu inicjalizacji wag dla funkcji ReLU



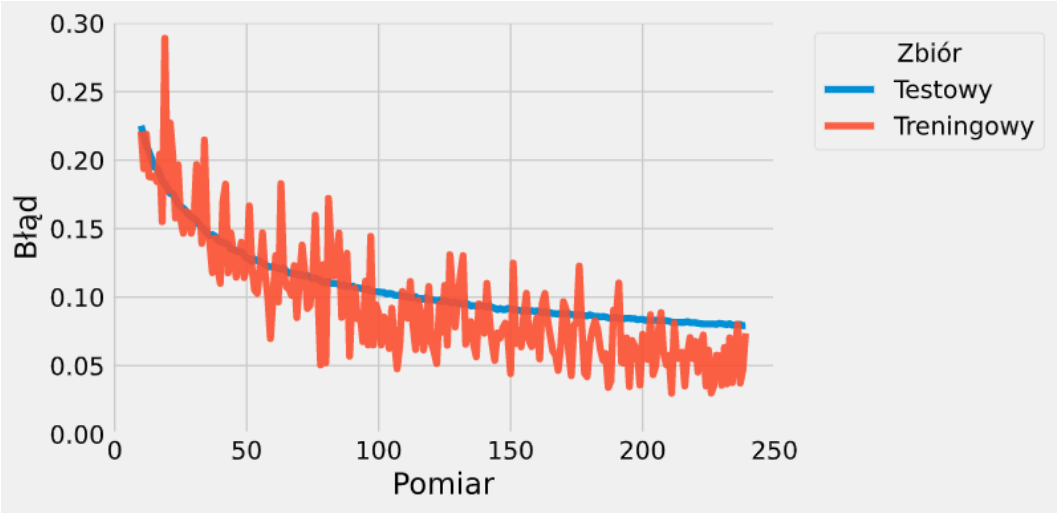
Wykres 20: Dokładność modelu w końcowym etapie uczenia w zależności od sposobu inicjalizacji wag dla funkcji ReLU



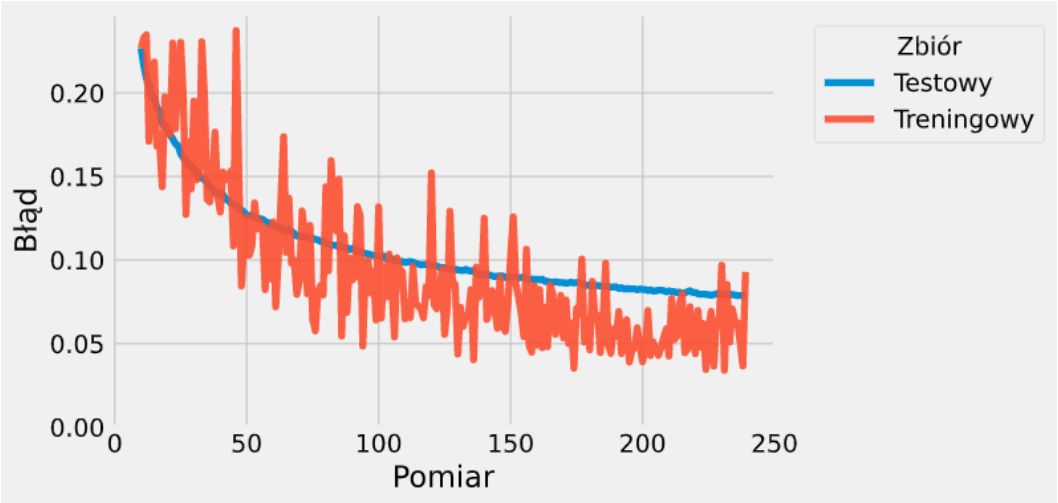
Wykres 21: Zachowanie funkcji błędu dla funkcji ReLU i inicjalizacji z zakresu



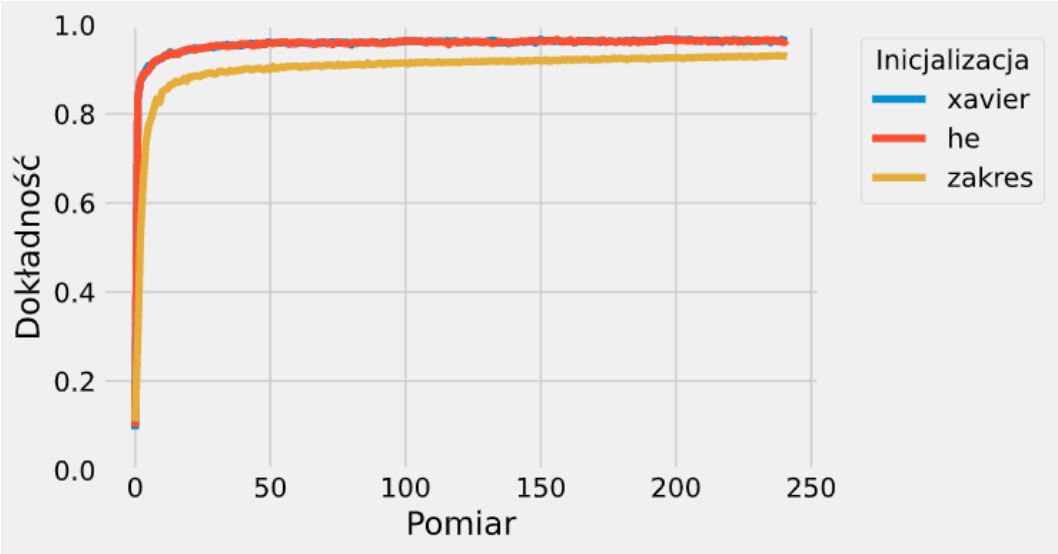
Wykres 22: Zachowanie funkcji błędu dla funkcji ReLU i inicjalizacji Xaviera



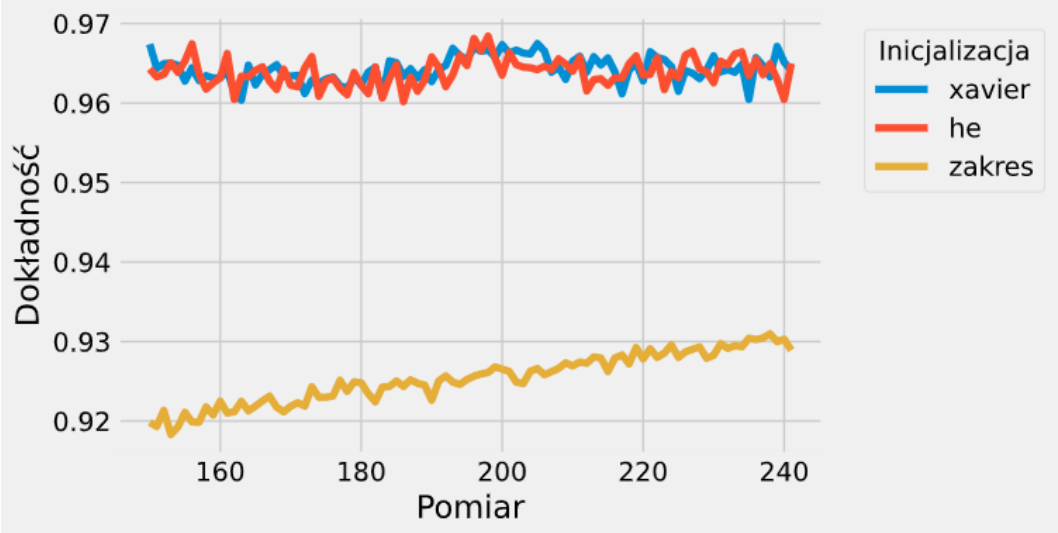
Wykres 23: Zachowanie funkcji błędu dla funkcji ReLU i inicjalizacji He



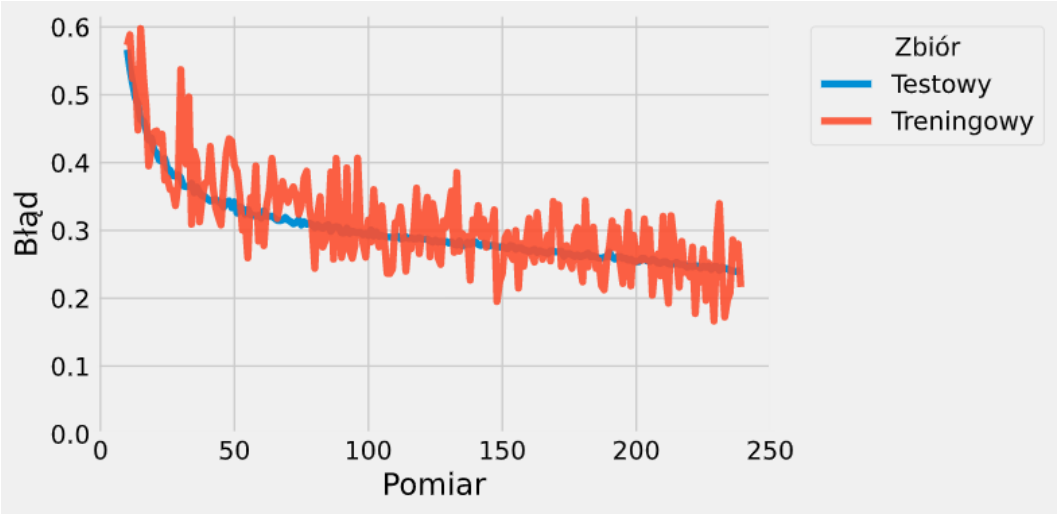
Wykres 24: Dokładność modelu w zależności od sposóbu inicjalizacji wag dla funkcji Sigmoid



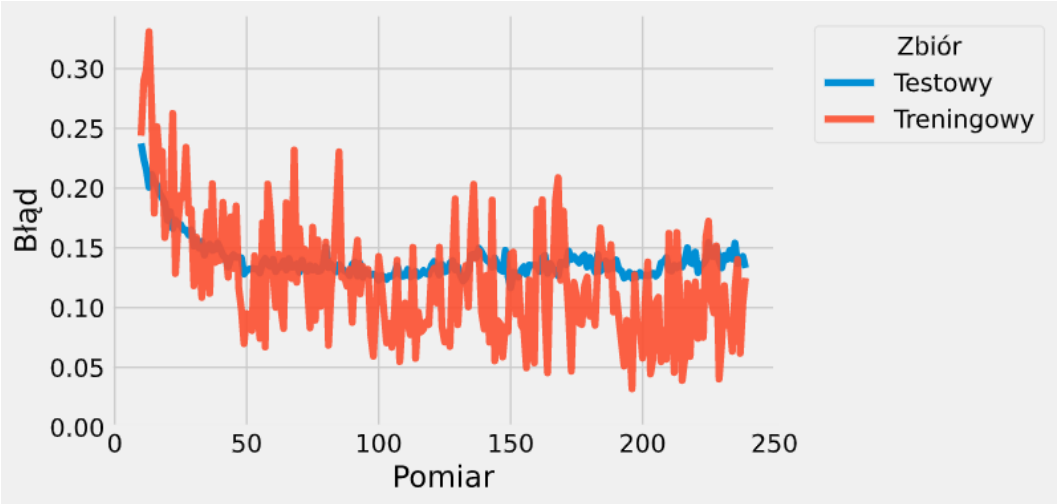
Wykres 25: Dokładność modelu w końcowym etapie uczenia w zależności od sposóbu inicjalizacji wag dla funkcji Sigmoid



Wykres 26: Zachowanie funkcji błędu dla funkcji Sigmoid i inicjalizacji z zakresu



Wykres 27: Zachowanie funkcji błędu dla funkcji Sigmoid i inicjalizacji Xaviera



Wykres 28: Zachowanie funkcji błędu dla funkcji Sigmoid i inicjalizacji He

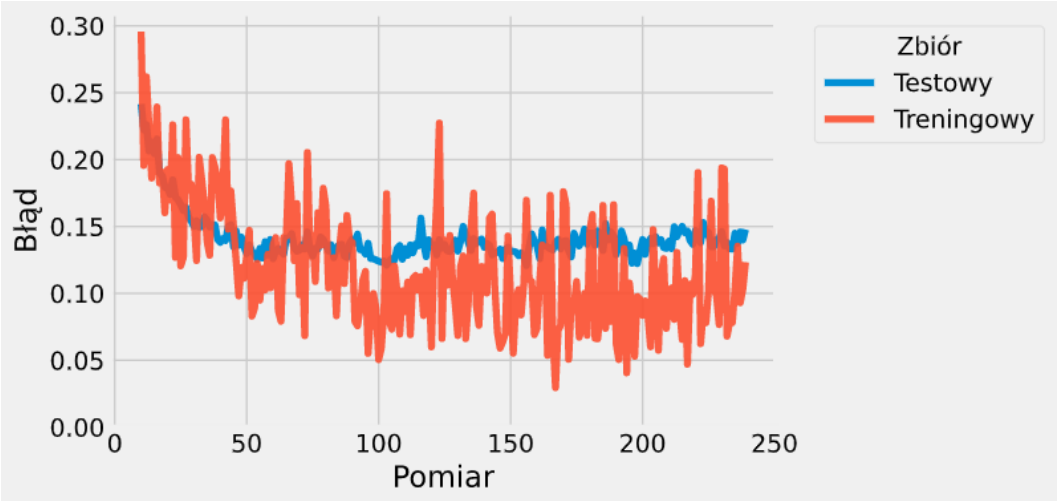


Tabela 4: Średnia maksymalna dokładność w zależności od sposóbu inicjalizacji wag

Inicjalizacja	Dokładność [%]	
	ReLU	Sigmoid
Zakres	97.70	93.29
Xavier	97.71	**97.18**
He	**97.73**	97.16

Wnioski

Na przedstawionych wykresach 19, 20 oraz tabeli 4 można zauważyć, że dla funkcji ReLU rodzaj inicjalizacji nie miał dużego wpływu. Pomimo różnicy 2 rzędów wielkości pomiędzy inicjalizacjami He (0.0025) i Xaviera (0.0015) a z zakresu (0.1) funkcja ReLU wydaje się być obojętna na wariację wag po przekroczeniu pewnej granicy. Dodając jednak wykresy 24 i 25 widać, że dla funkcji Sigmoidalnej sytuacja wygląda inaczej. W tym przypadku inicjalizacja He oraz Xaviera dają dużo lepsze wyniki od zakresu. Może to być spowodowane tym, że Sigmoid może operować bez nasycenia tylko w małym przedziale, w przeciwieństwie do ReLU która nie jest ograniczona dla wartości dodatnich. Ponadto na wykresach 21- 23 i 26- 28 zachowanie funkcji błędu testowego jest znacznie stabilniejsze dla funkcji ReLU, gdzie dla Sigmoida wykazuje symptomy przeuczenia. Jest to jednak najprawdopodobniej wina użytej funkcji, a nie inicjalizacji wag.

3 Wnioski

- Użycie odpowiedniej metody uczenia może przyspieszyć cały proces, a nawet dać lepsze wyniki.
- Różne metody dają różne wyniki dla takiego samego początkowego współczynnika uczenia, większość otrzymanych wyników można poprawić dostrajając hiperparametry dla danej metody.
- Inicjalizacja wag powinna być przeprowadzana pod użytą funkcję aktywacji.
- Dobranie wag w taki sposób aby funkcja aktywacji mogła pracować poza swoimi zakresami nasycenia daje dużo lepsze wyniki.
- Różnica pomiędzy inicjalizacjami Xaviera i He nie jest widoczna na małych sieciach.