Sieć wielowarstwowa uczona metodą propagacji wstecznej

Autor: Rafał Behrendt
246643@student.pwr.edu.pl
Prowadząca: dr hab. inż. Urszula Markowska-Kaczmar
29 listopada 2021

Abstract

Celem ćwiczenia jest zapoznanie się technikami poprawiającymi szybkość uczenia sieci. Eksperymenty przeprowadzono na zbiorze MNIST zawierającym 60000 danych treningowych oraz 10000 danych testowych. Każdy test przeprowadzono 10 razy, a wyniki uśredniono.

Kod źródłowy znajduje się w poniższym linku:

https://github.com/aspirew/neuralNetworks/tree/list3

1 Eksperymenty

Za domyślne hiperparametry przyjęto:

- Wielkość sieci: [20]
- Współczynnik uczenia: 0.01
- Inicjalizacja wag za pomocą rozkładu normalnego o parametrach:
  mi: 0
  sigma: 0.1

Każdy test przeszedł przez 50 epok zanim został zakończony.

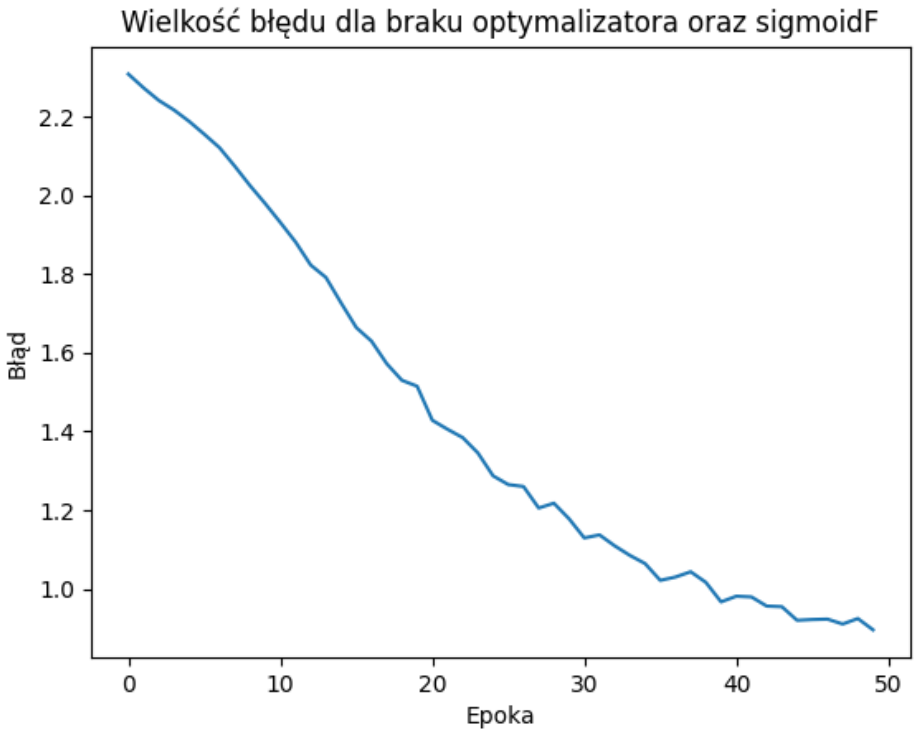
2 Test optymalizatorów współczynnika uczenia

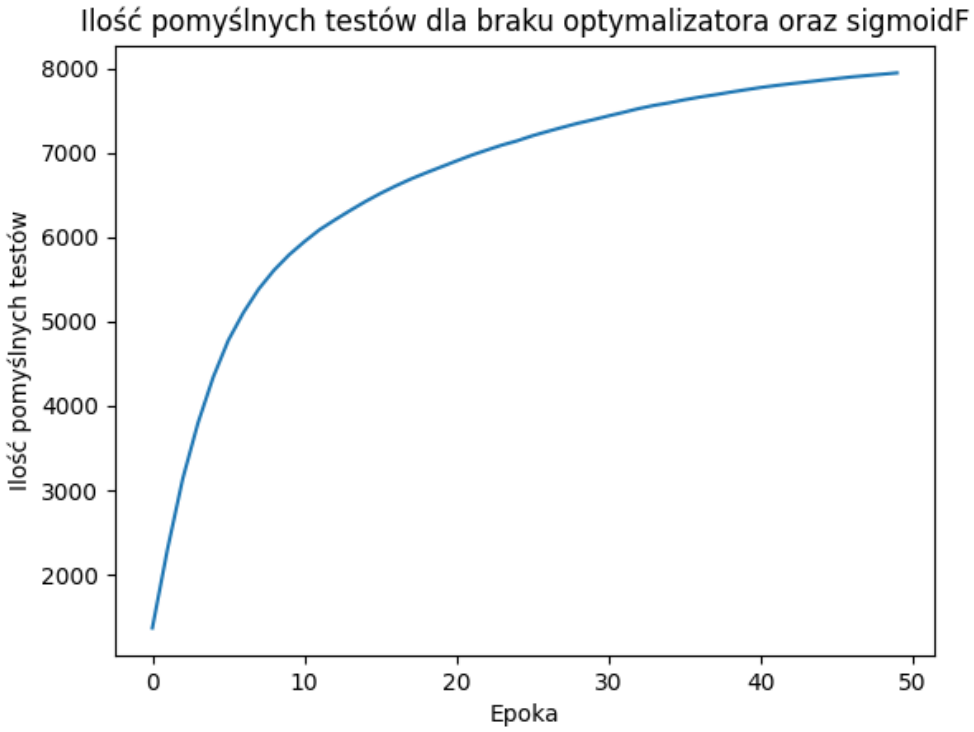
Zbadano jak podane w ćwiczeniu optymaliztory wpływają na skuteczność i szybkość uczenia. Wyniki działania optymalizatorów porównano z wnikami bez ich użycia. Optymaliztory przetestowano na dwóch funkcjach aktywacji: sigmoidalnej i ReLU. W tabelach zawarte są uśrednione błędy oraz oraz uśredniona liczba prawidłowo rozpoznanych liczb ze zbioru testowego. Niżej przedstawione są wyniki na wykresach

2.1 Funkcja sigmoidalna

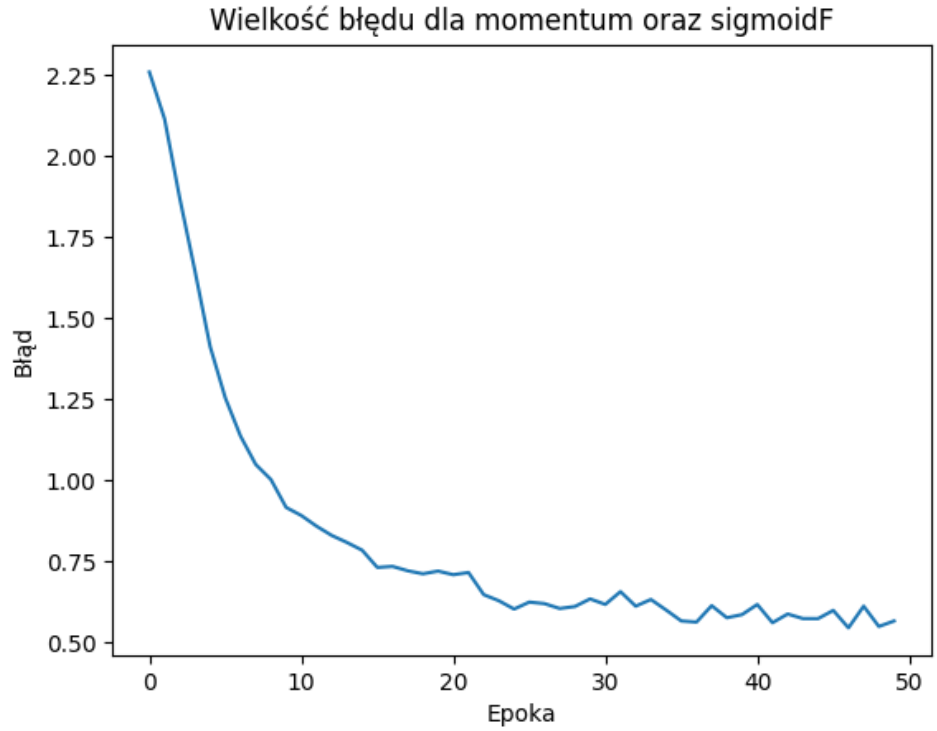
Optymalizator	Błąd	Skuteczność
Brak	1.43	6649.3
Momentum	0.81	7923
Momentum Nesterova	0.81	7956.9
Adagrad	0.91	7583.8
Adadelta	0.83	7764.9
Adam	0.65	7946.4

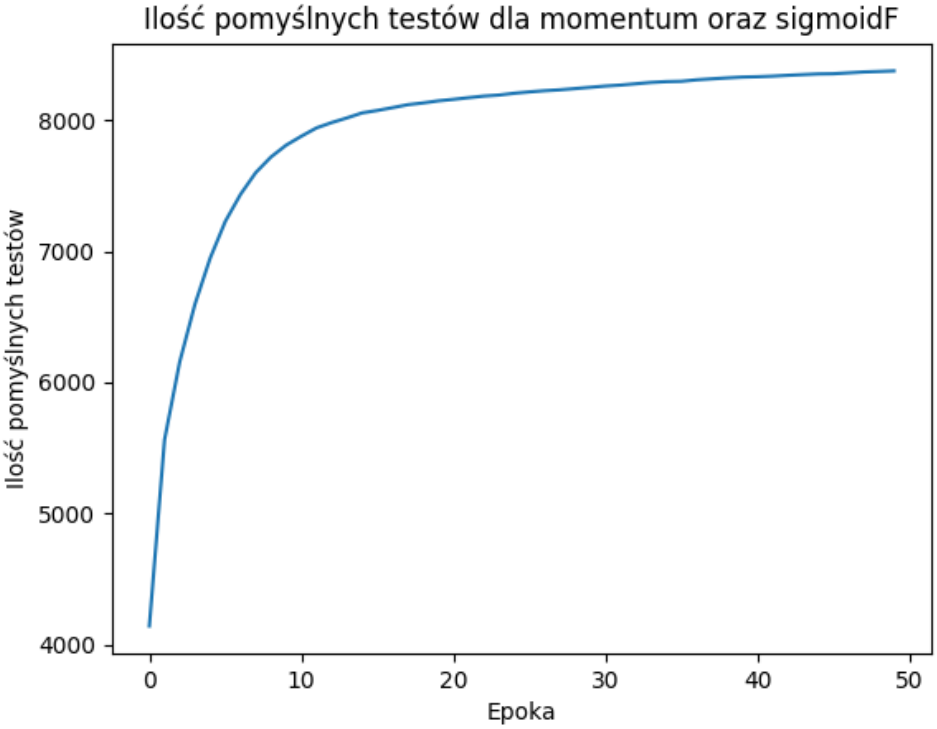
Tablica 1: Optymalizatory - funkcja sigmoidalna



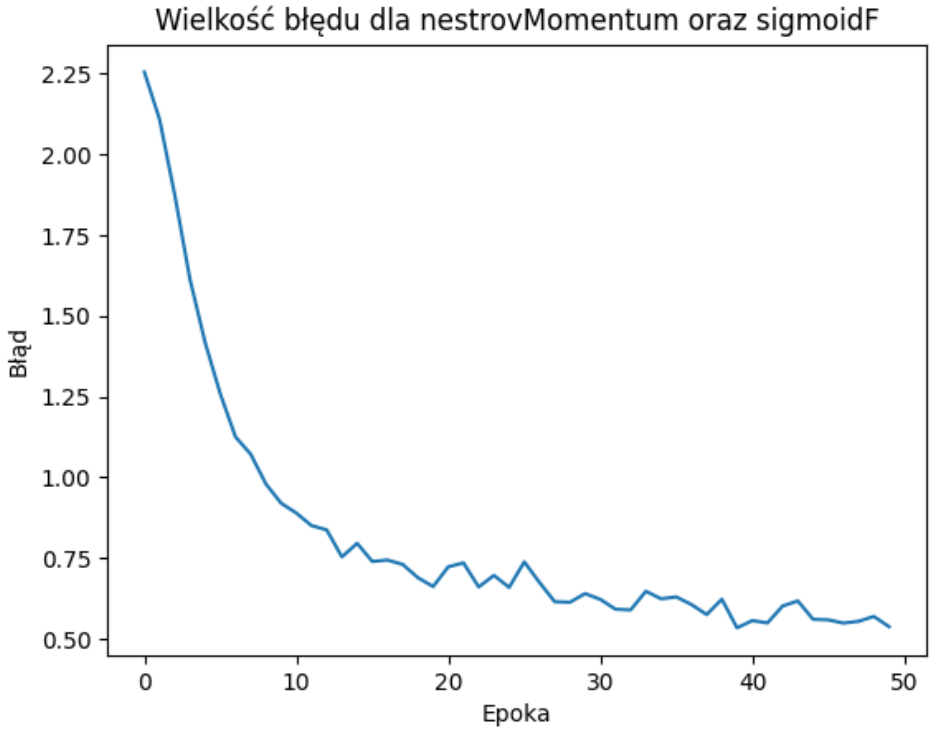


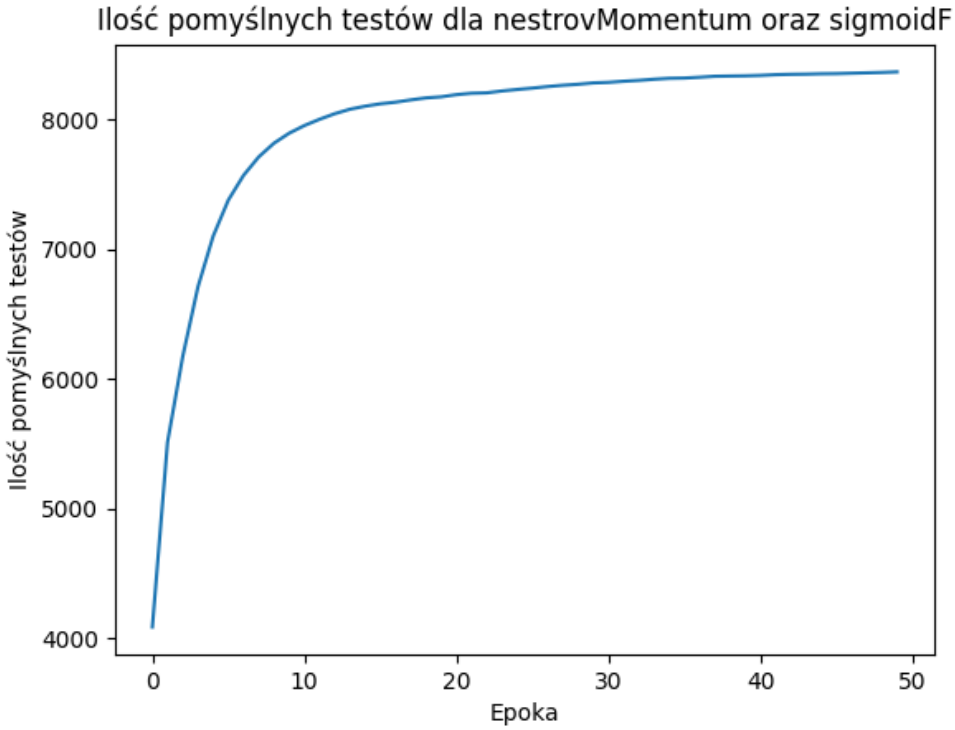
W przypadku braku optymalizatora można zauważyć, że wykres błędu zmniejsza się powoli, a sam średni błąd jest na wysokim poziomie.



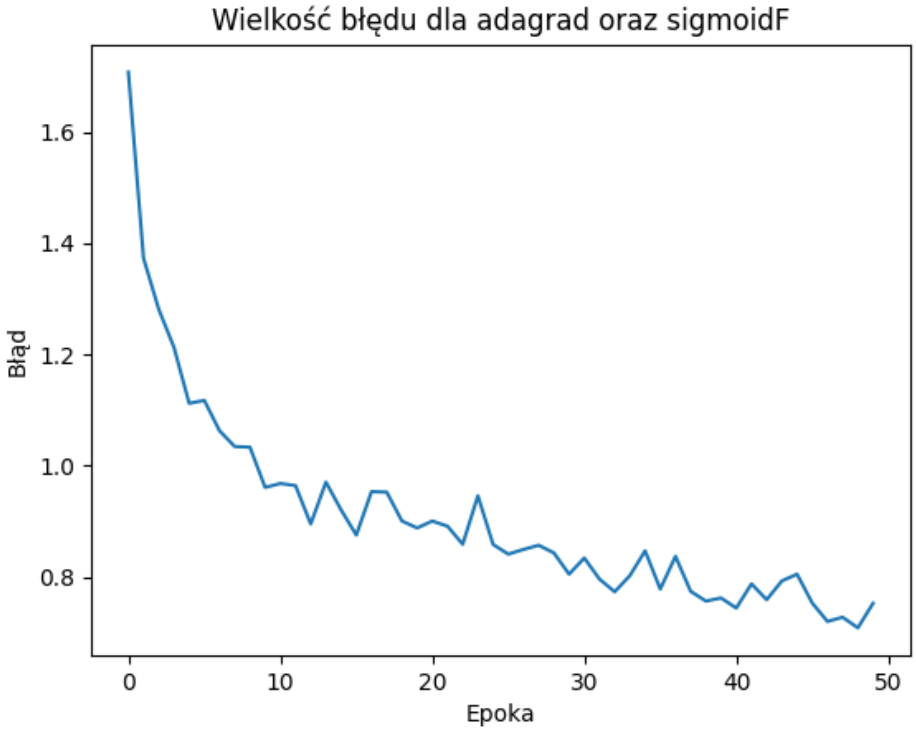


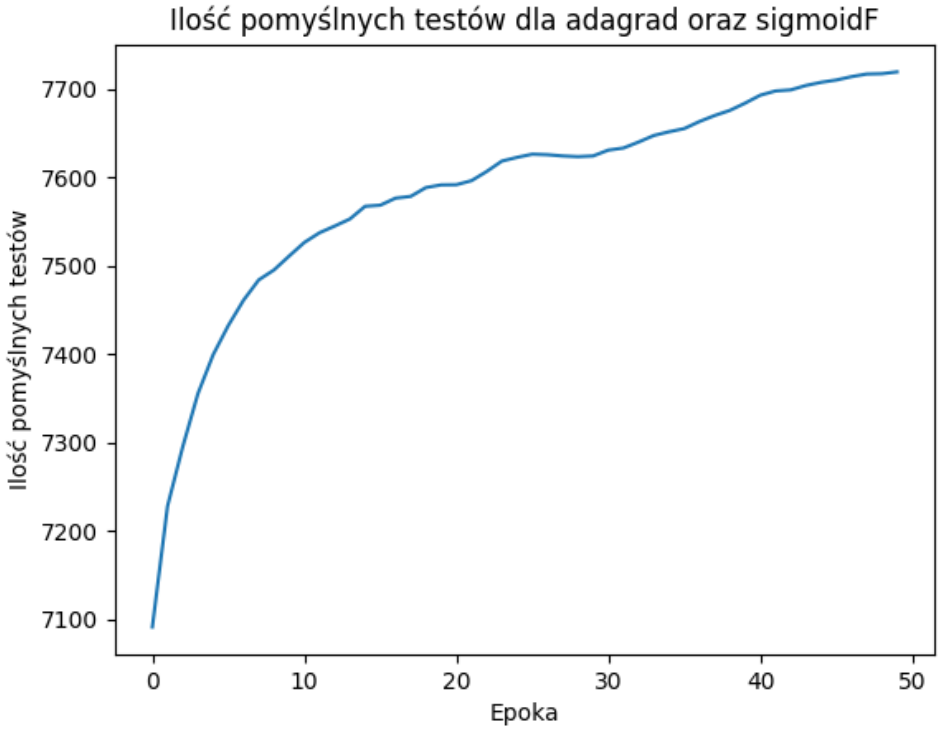
Metoda momentum wyraźnie zwiększa szybkość uczenia sieci - wykres szybciej schodzi do optimum. Ilość pomyślnych testów jest na podobnym poziomie jak przy braku optymalizatora, jednak graniczna wartość osiągnięta jest znacznie szybciej.



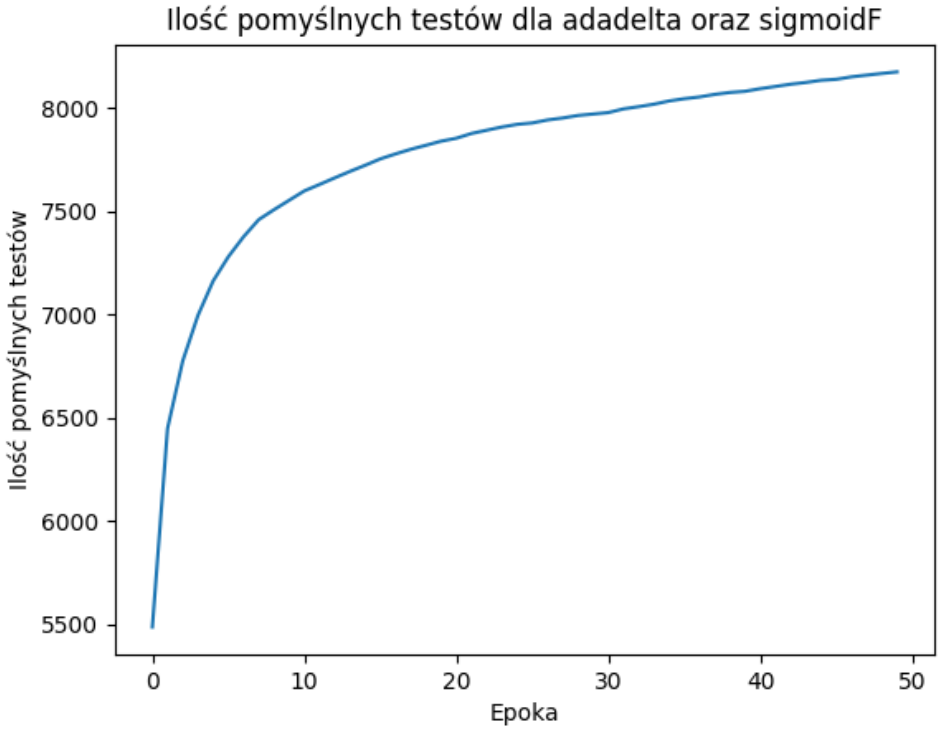


Momentum Nesterova daje zbliżone wyniki jak zwykłe momentum - ponownie widać poprawę względem braku optymalizatora.

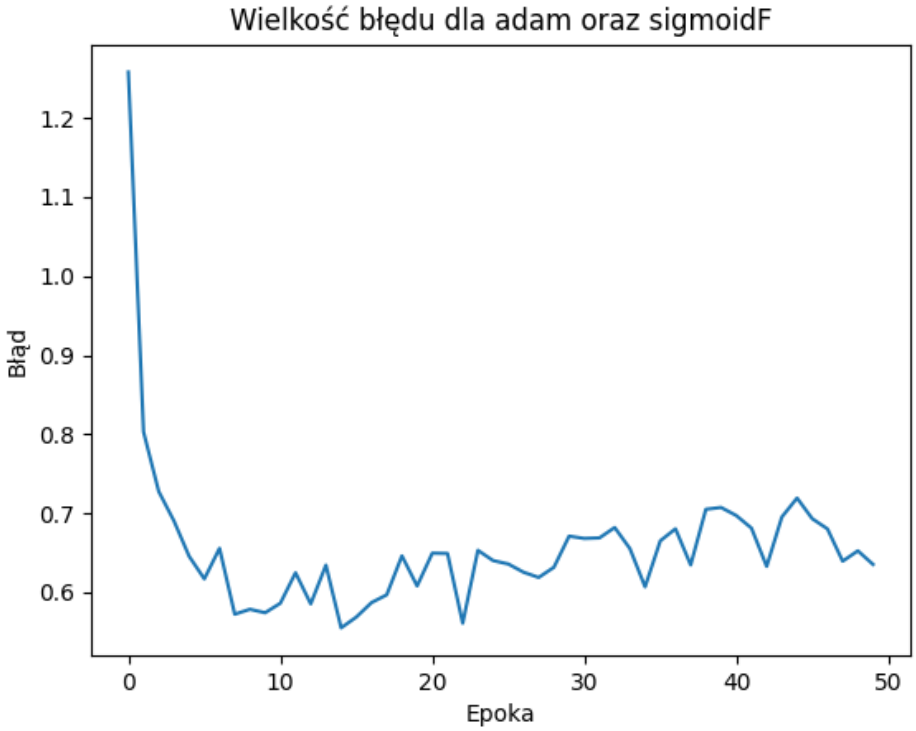


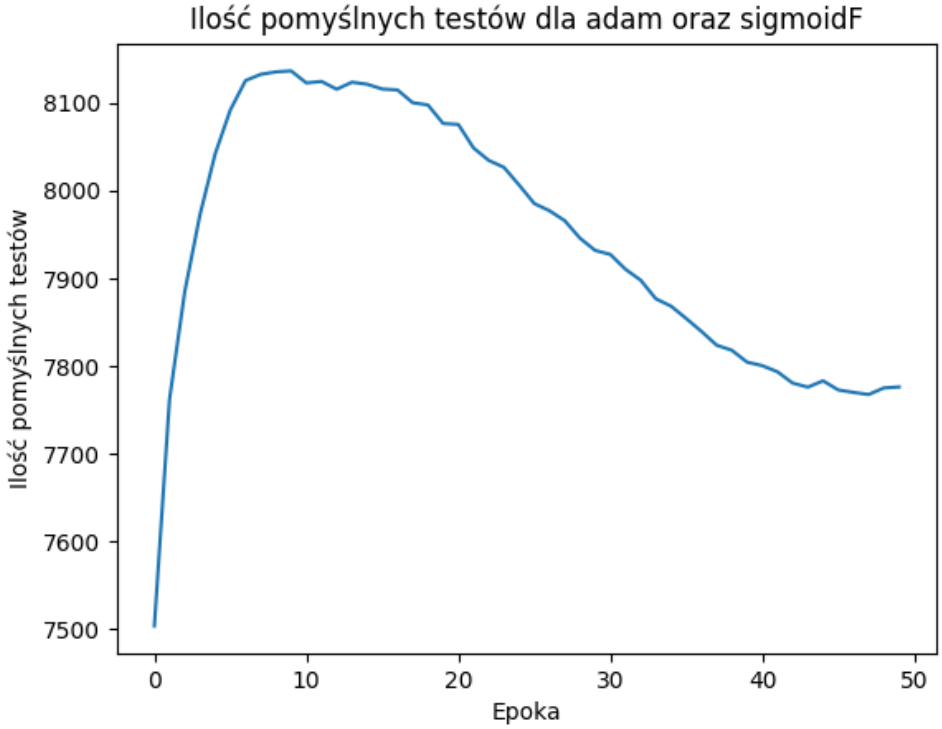


Optymalizator adagrad działa dość niestabilnie. Błąd spada szybciej niż w przypadku braku optymalizatora.



Ustabilizowanie można zauważyć w przypadku optymalizatora adadelta. Widać poprawę względem adagrad.



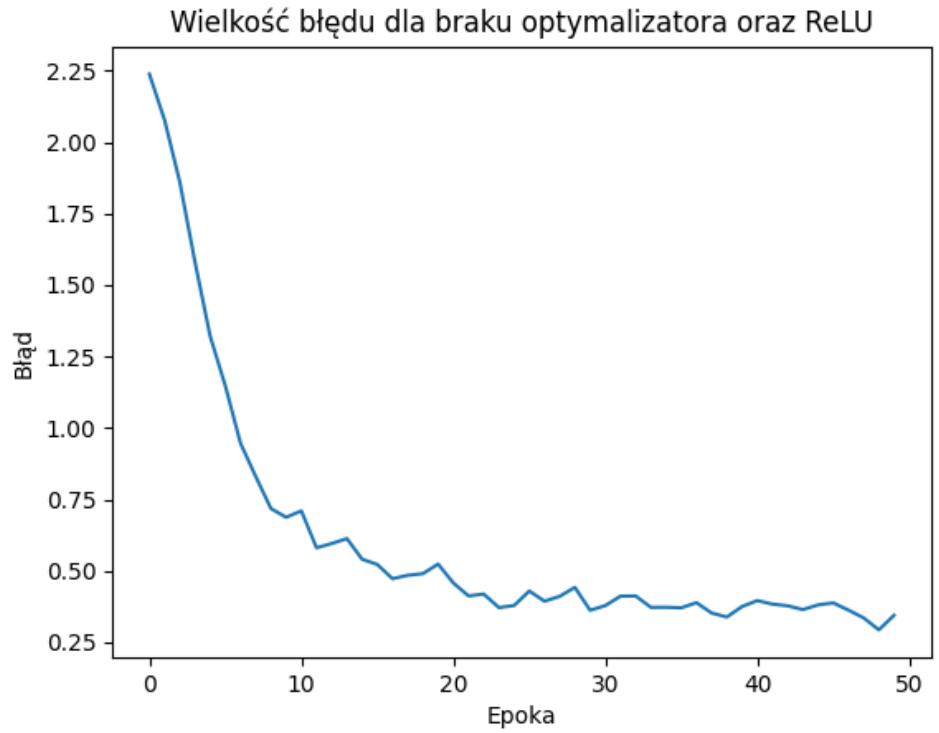


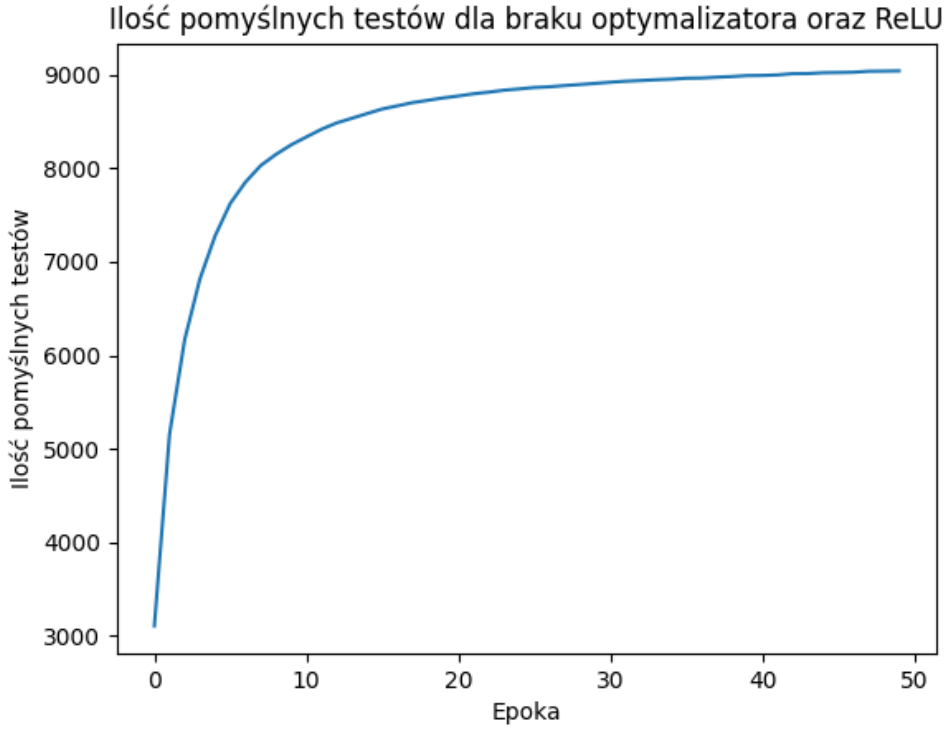
Optymalizator adam nie osiągnął najlepszej średniej wartości błędu. Na wykresie widać jednak, że osiąga on optimum zdecydowanie najszybciej - do tego stopnia że po 20 epoce można zauważyć objawy przeuczenia się sieci, co skutkowało pogorszeniem średnich wyników.

2.2 Funkcja ReLU

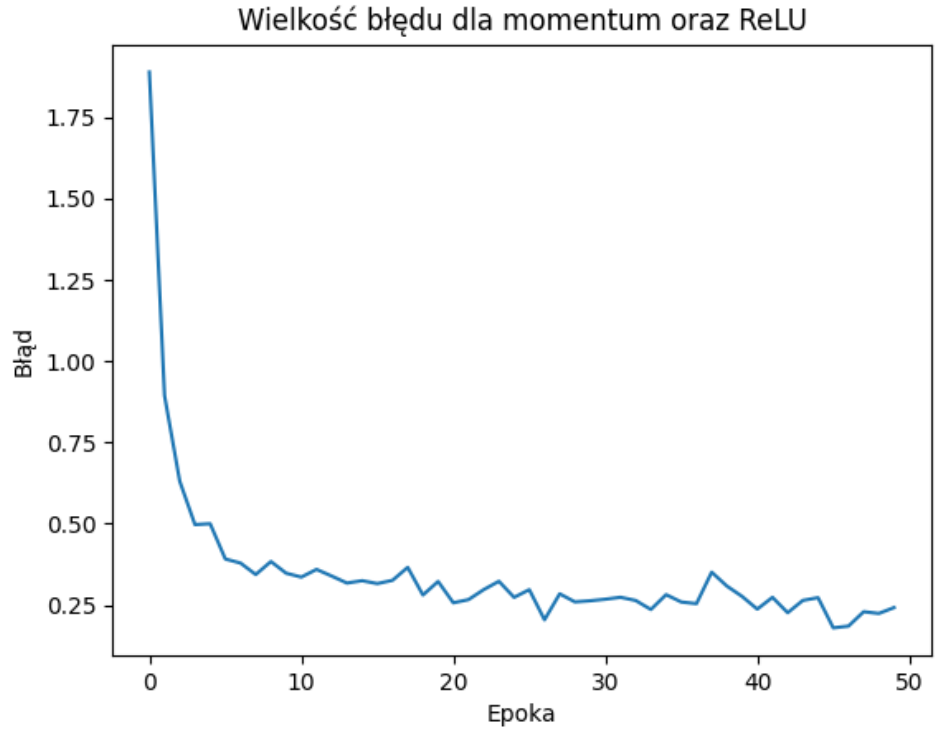
Optymalizator	Błąd	Skuteczność
Brak	0.6	8450.87
Momentum	0.34	9104.6
Momentum Nesterova	0.35	9109.2
Adagrad	0.26	9276.0
Adadelta	0.32	9174.3
Adam	0.13	9526.7

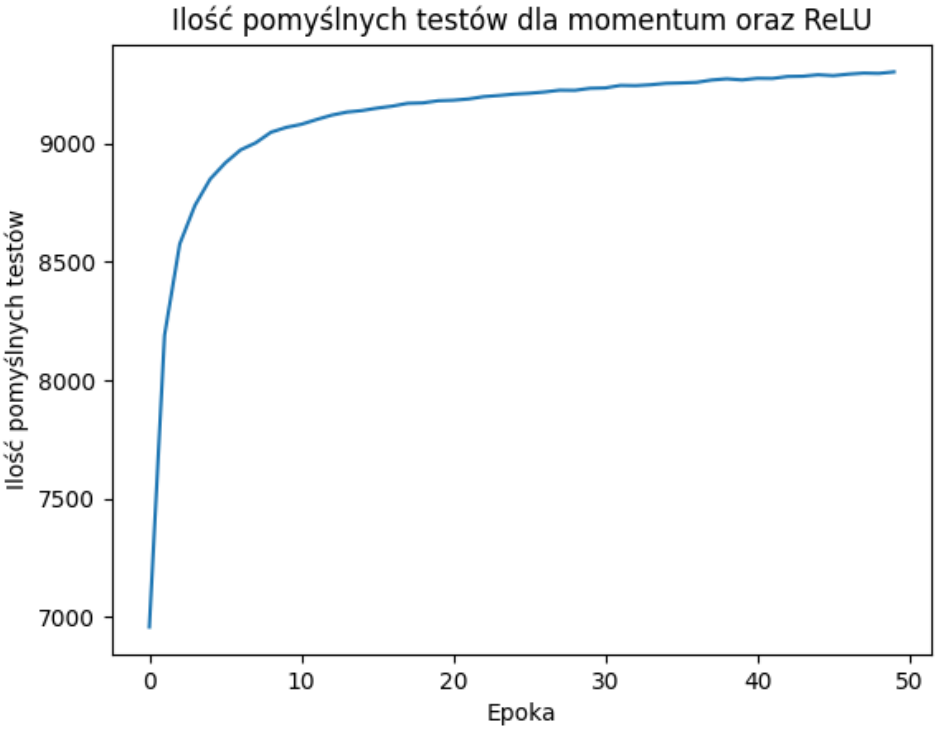
Tablica 2: Optymalizatory - funkcja ReLU



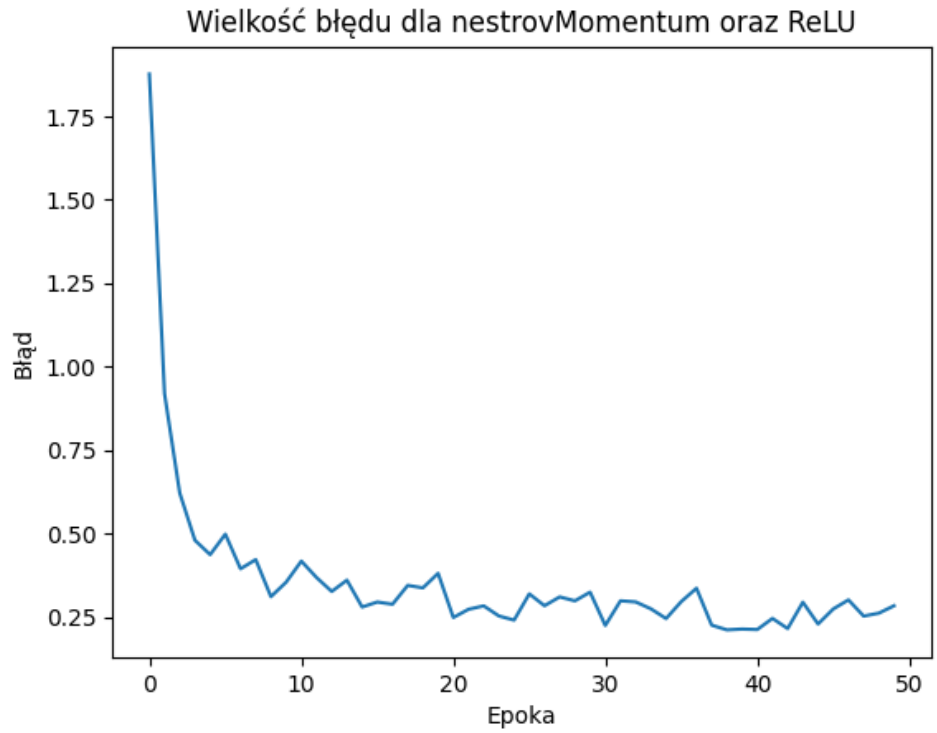


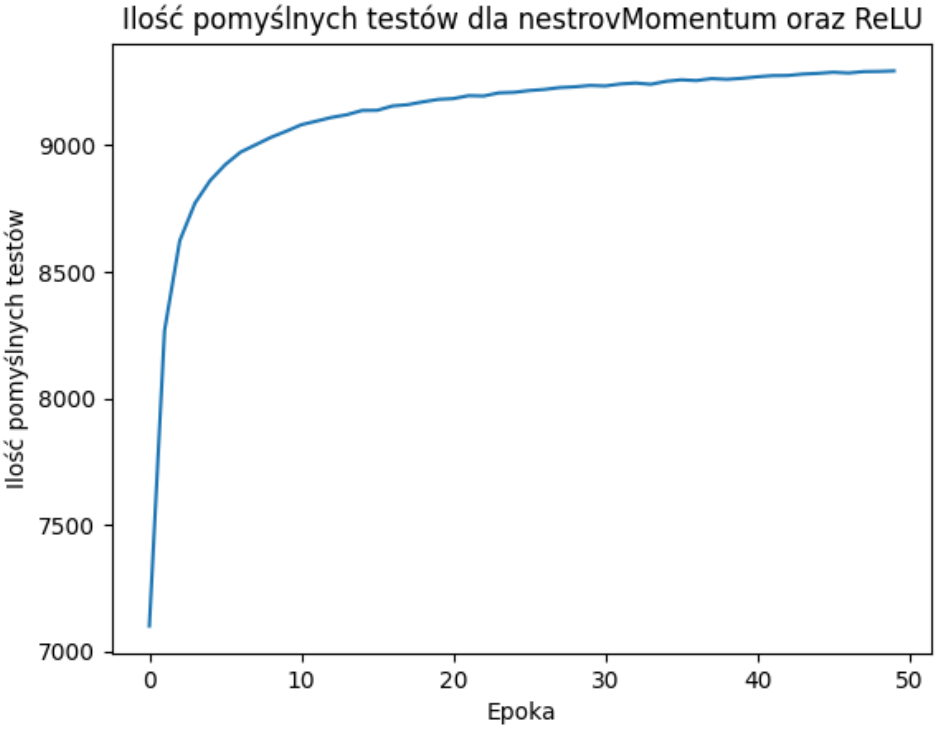
Można zauważyć, że funkcja aktywacji ReLU radzi sobie lepiej niż funkcja sigmoidalna. Wykres szybciej osiąga optimum pomimo tego, że nie zastosowano żadnego optymalizatora.



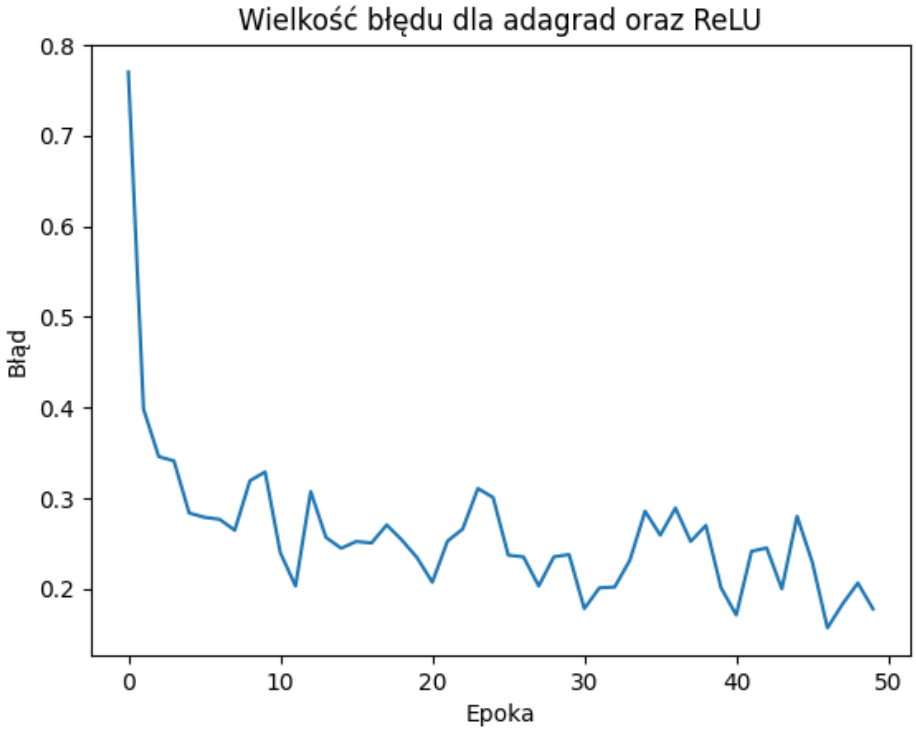


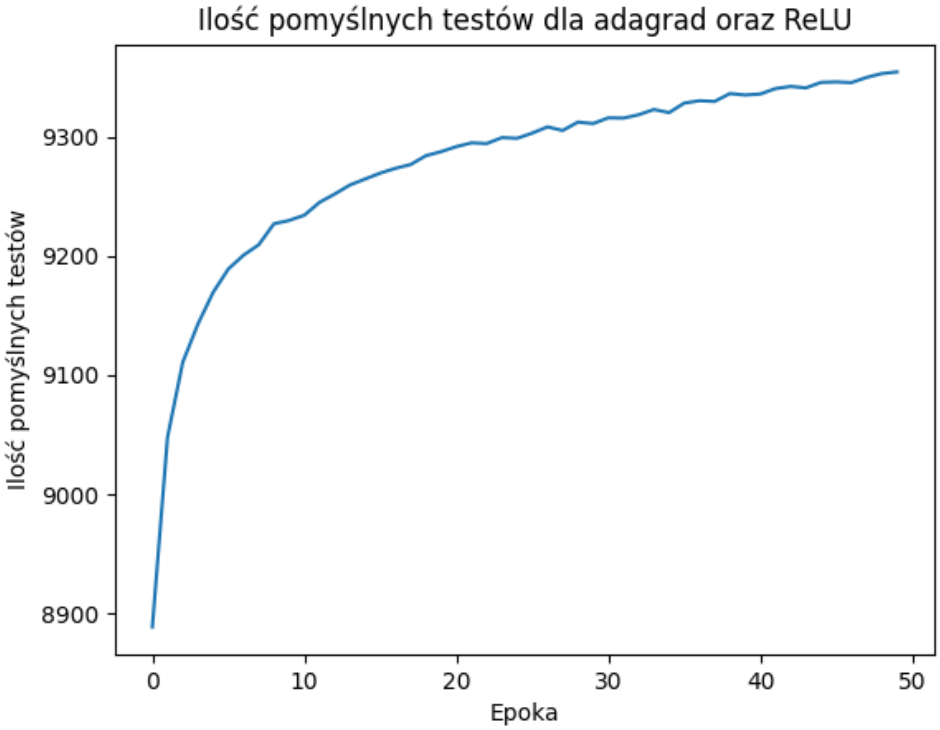
Wydajność jest jeszcze lepsza, dzięki zastosowaniu momentum - wykres szybciej zbiega do optimum.



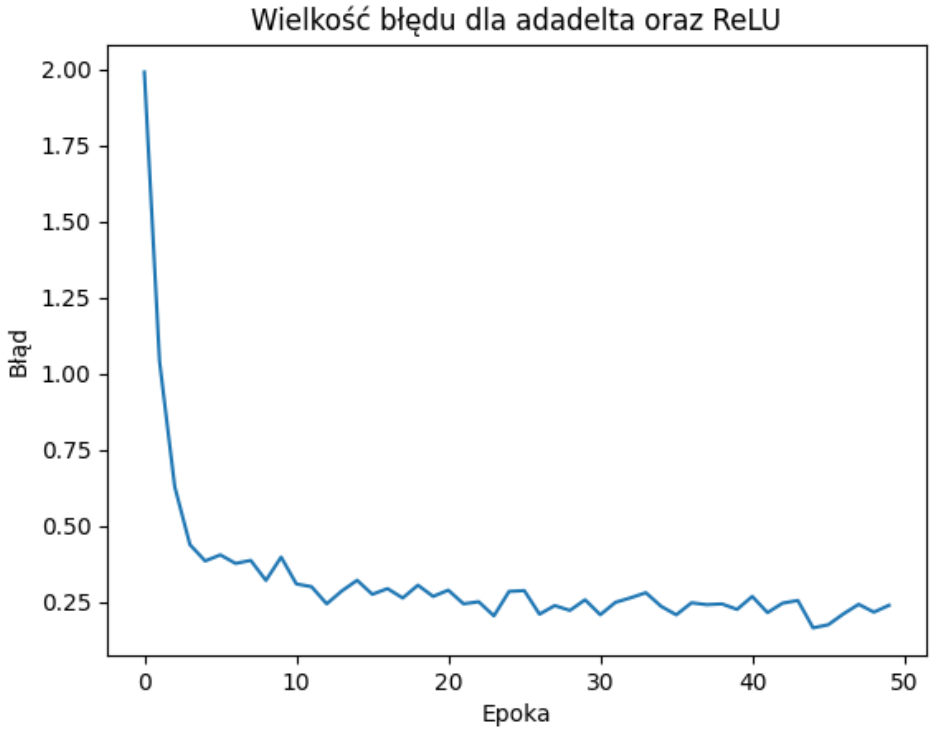


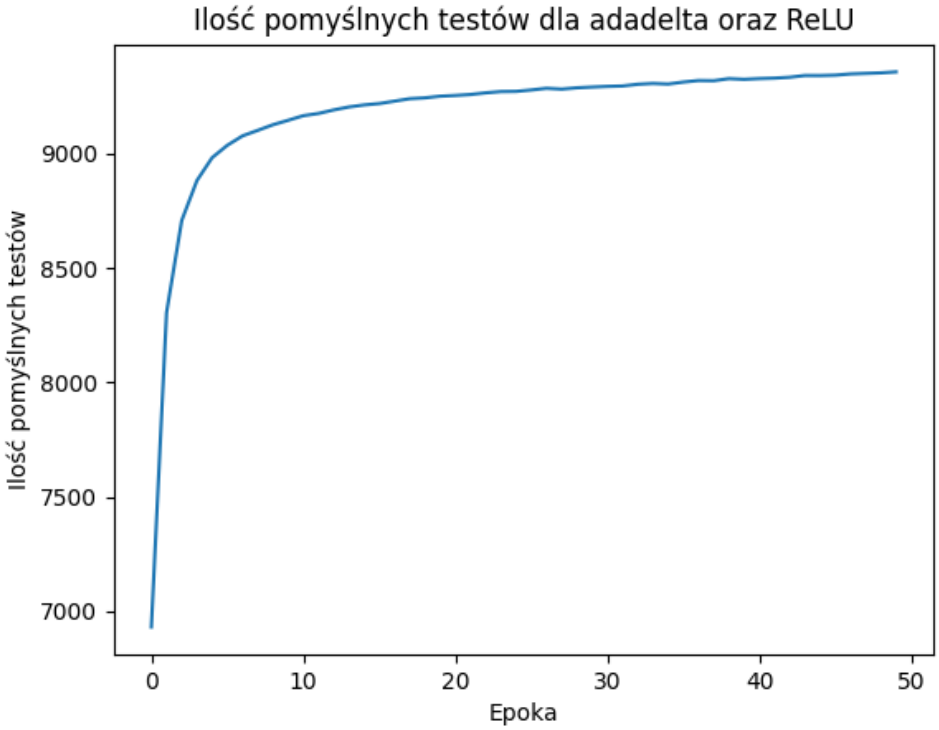
Ponownie momentum Nesterova prezentuje zbliżone wyniki do momentum.



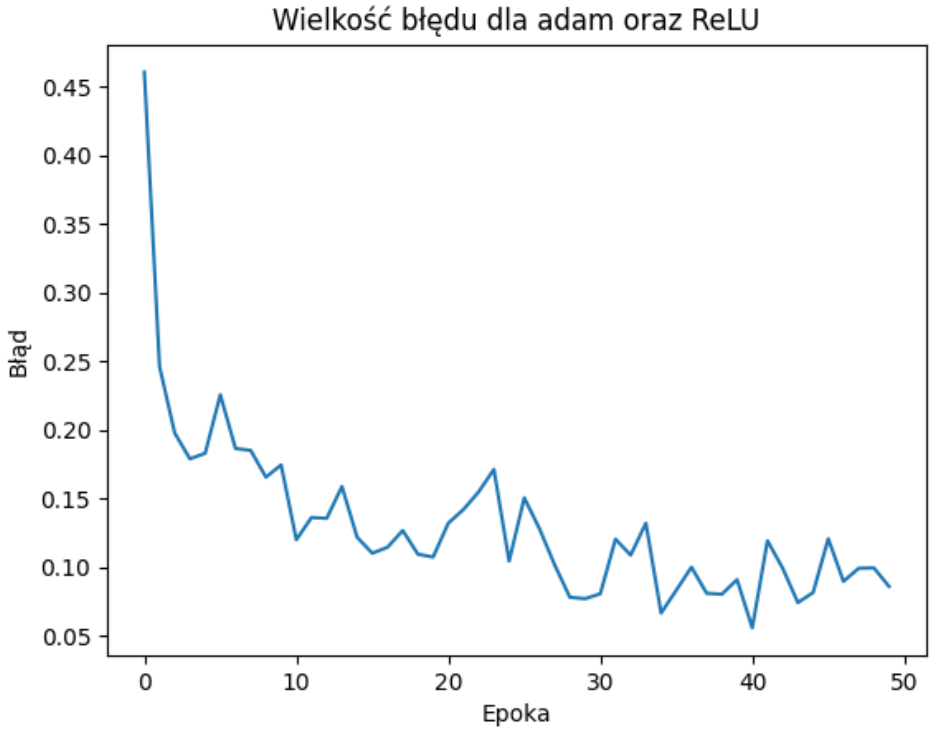


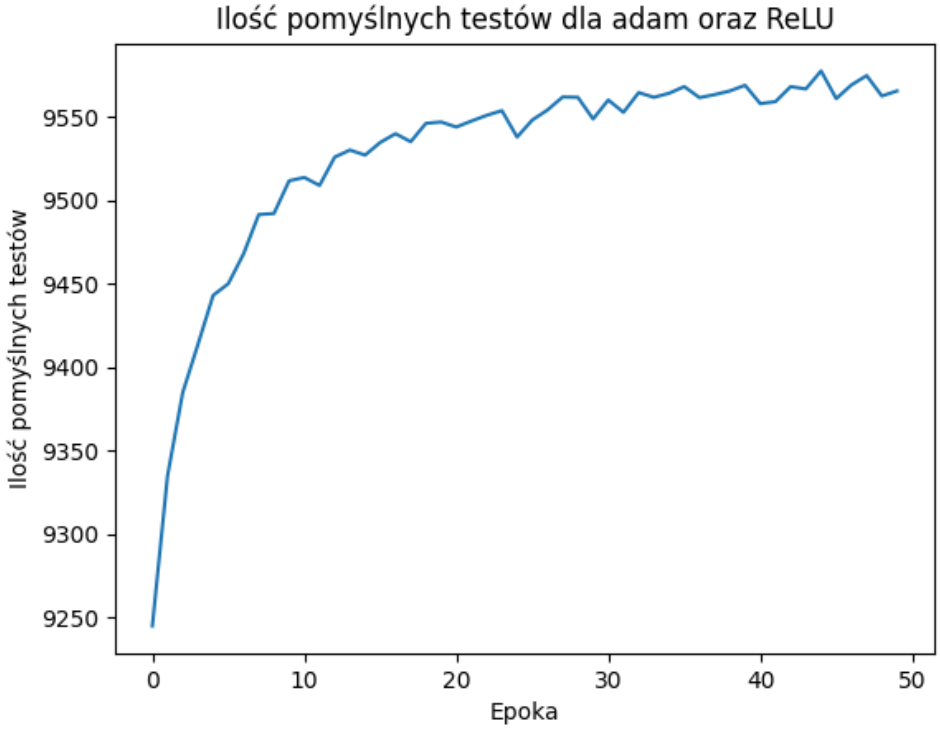
Jak w przypadku funkcji sigmoidalnej, tak i dla funkcji ReLU adagrad wyznacza się niestabilnością.





Adadelta szybko zbiega do optimum, a wykres jest bardziej stabilny niż to było w przypadku adagrad. Średnie wyniki są jednak gorsze - w odróżnieniu do funkcji sigmoidalnej.





Wydawać by się mogło, że adam wykazuje się niestabilnością - wynika to jednak ze zmniejszonej skali, gdyż wyniki są wyjątkowo dobre. Błąd rzędu 0.1 oraz skuteczność ponad 95% pokazują skuteczność tego optymalizatora.

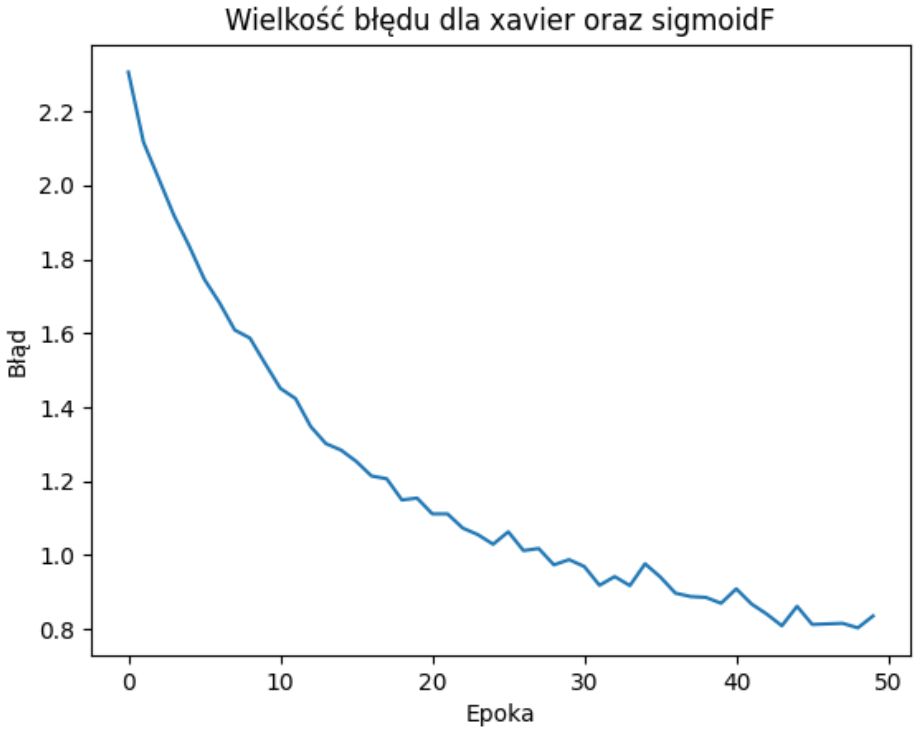
3 Inicjalizacja wag sieci

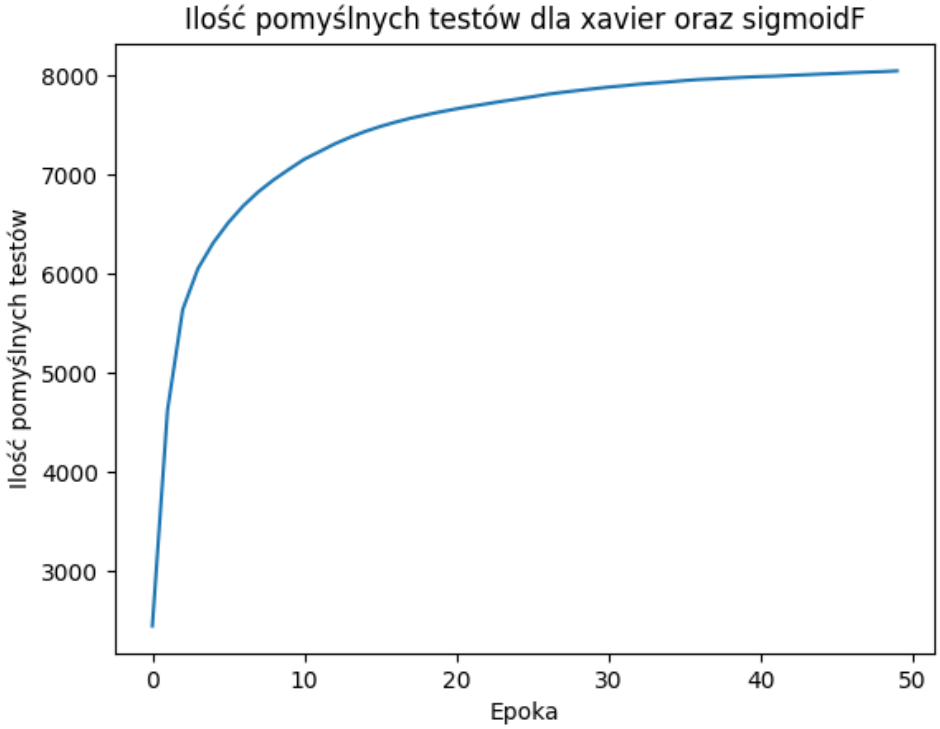
Zbadano jak różne sposoby inicjalizacji wag w sieci wpływają na skuteczność i szybkość uczenia. Do optymalizacji nie użyto żadnego optymalizatora. W tabelach zawarte są uśrednione błędy oraz oraz uśredniona liczba prawidłowo rozpoznanych liczb ze zbioru testowego. Niżej przedstawione są wyniki na wykresach

3.1 Funkcja sigmoidalna

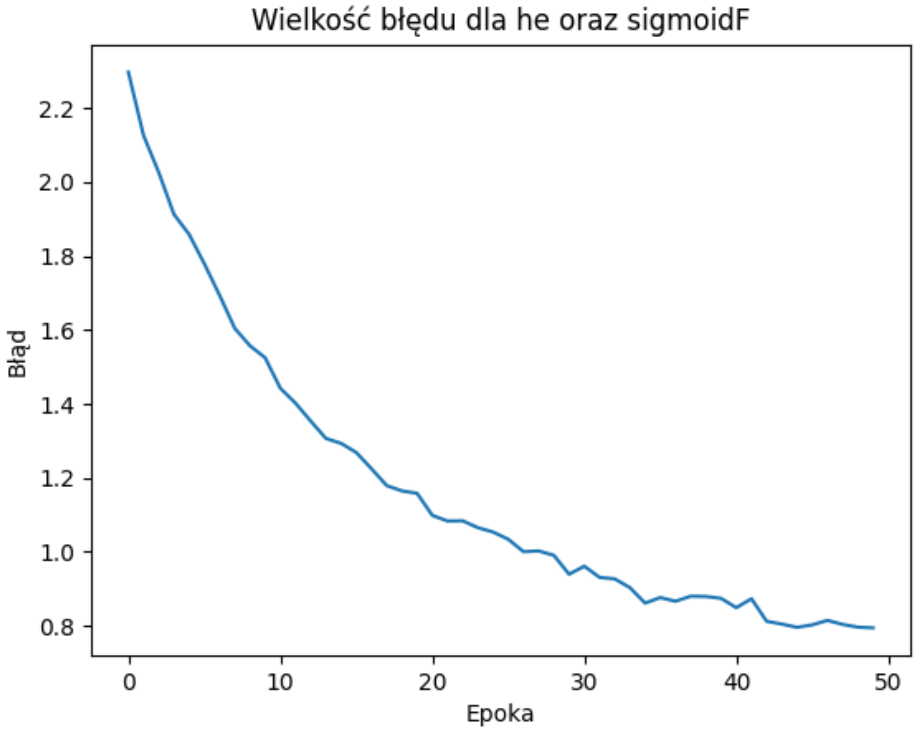
Metoda inicjalizacji	Błąd	Skuteczność
Rozkład normalny $\sigma = 0.1$	1.43	6649.3
Xavier	1.18	7416.43
He	1.17	7543.632

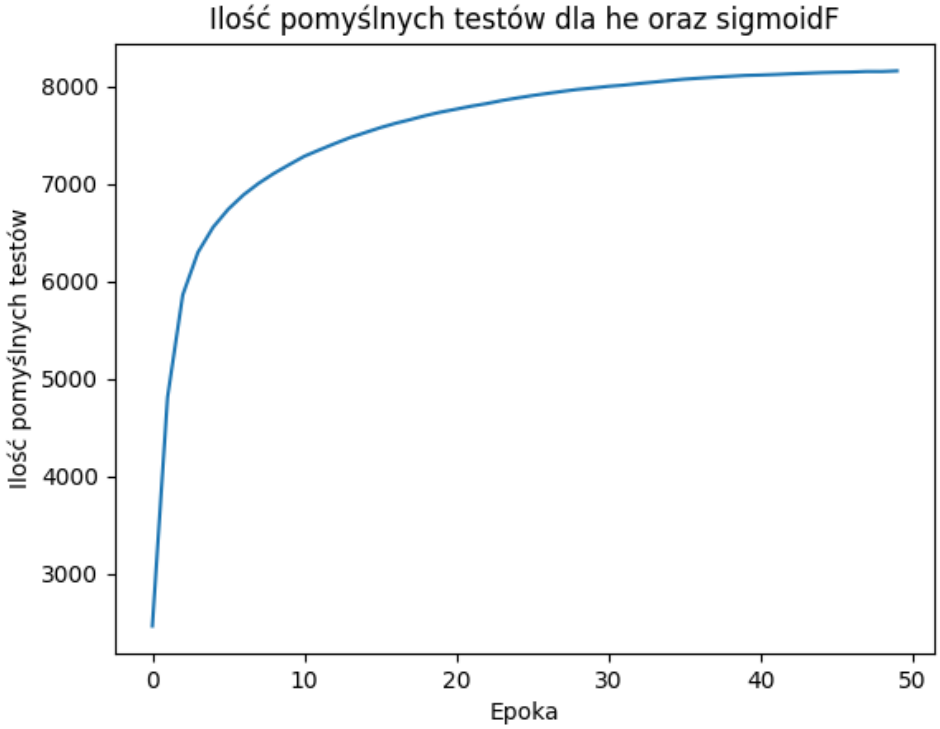
Tablica 3: Inicjalizacja wag - funkcja sigmoidalna





Od razu można zauważyć, że prędkość oraz skuteczność znacząco rośnie w porównaniu do użycia samego rozkładu normalnego.



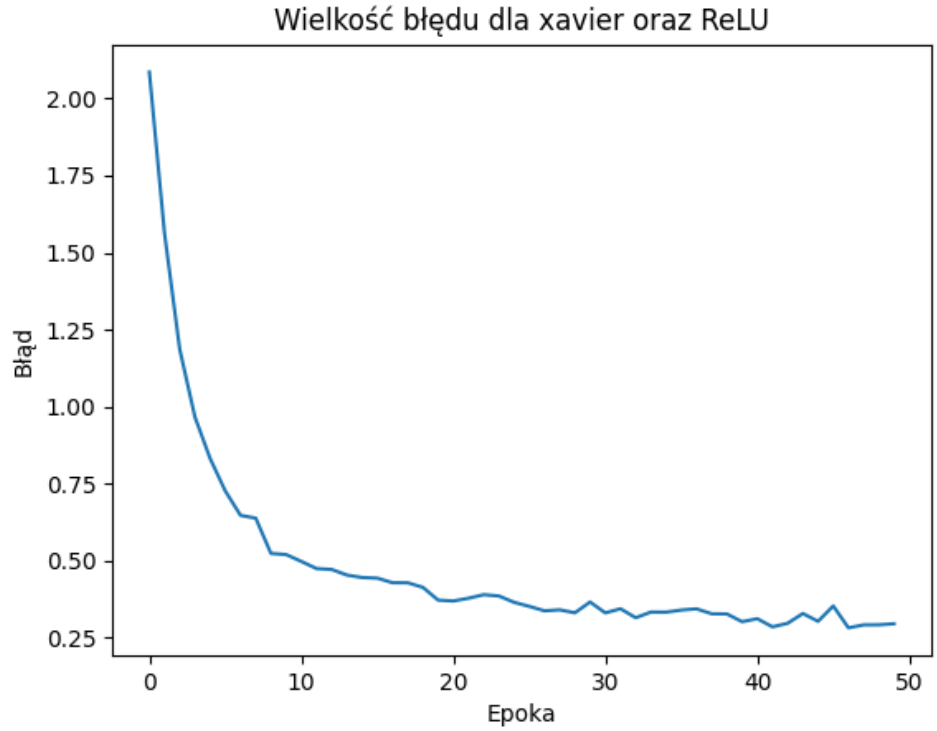


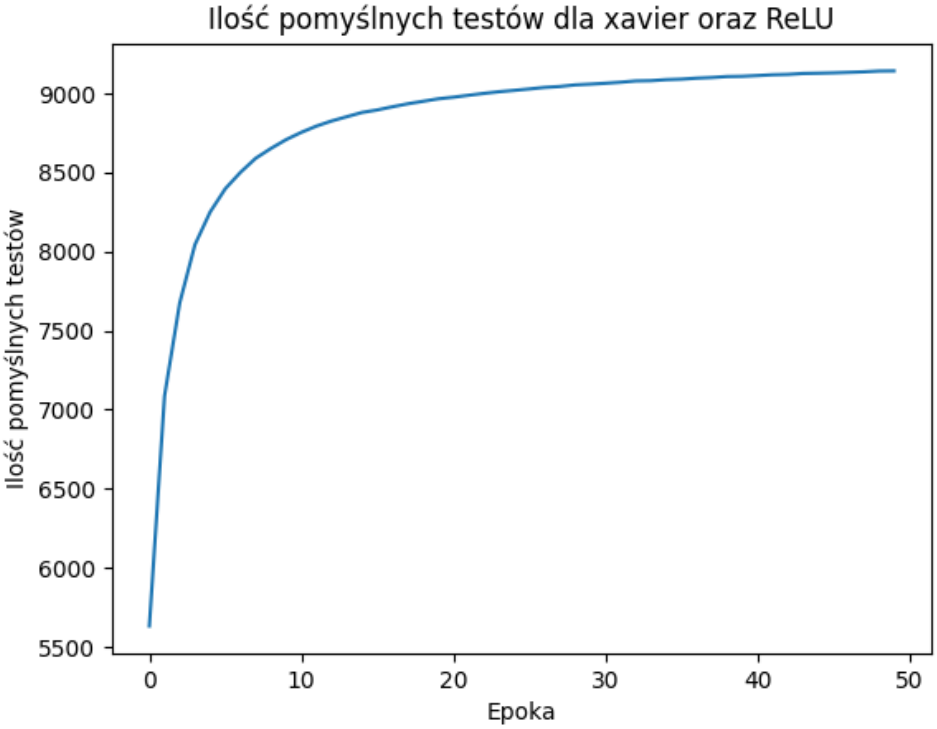
Wyniki są zbliżone do metody xaviera - prędkość uczenia właściwie się nie zmienia, a skuteczność nieznacznie rośnie.

3.2 Funkcja ReLU

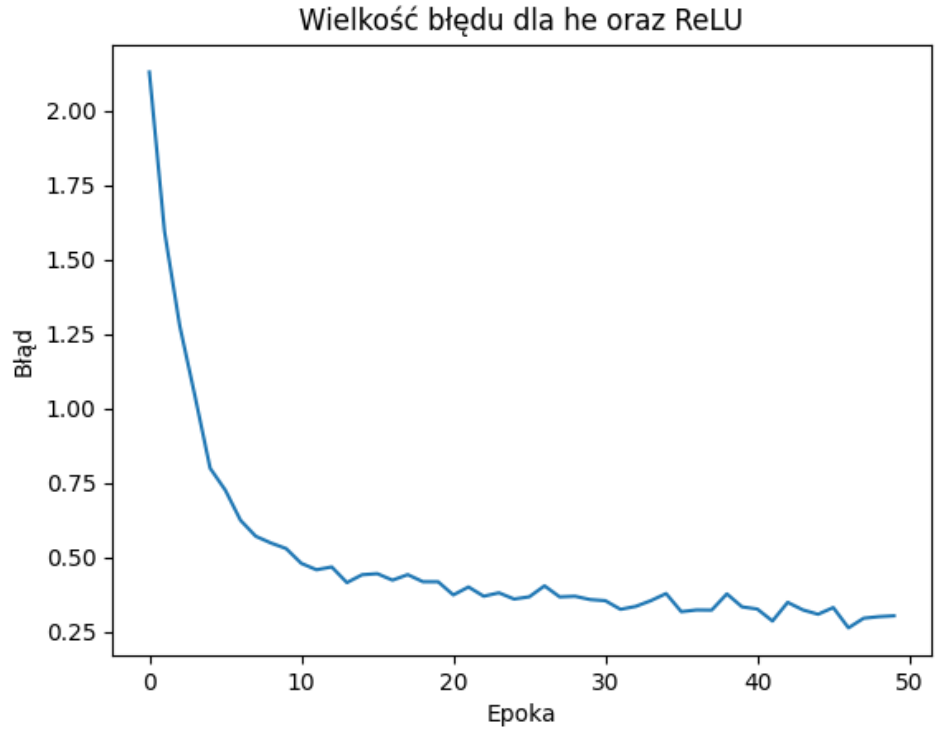
Metoda inicjalizacji	Błąd	Skuteczność
Rozkład normalny $\sigma = 0.1$	0.6	8450.87
Xavier	0.48	8813.91
He	0.48	8799.37

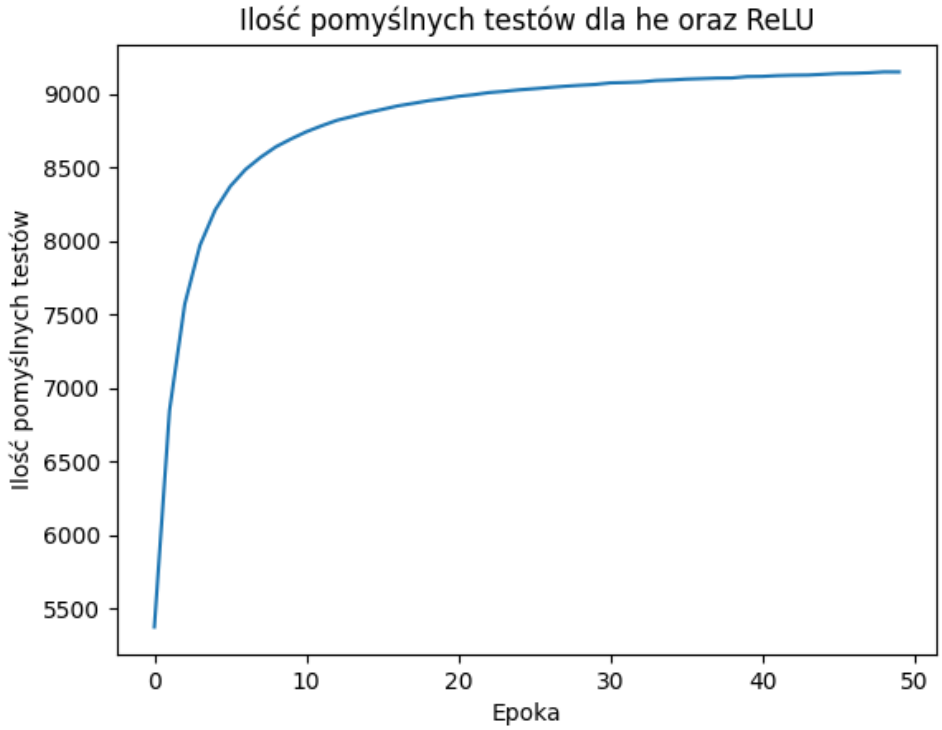
Tablica 4: Inicjalizacja wag - funkcja ReLU





W przypadku funkcji ReLU nie widać aż tak dużej zmiany jak to było w przypadku funkcji sigmoidalnej - jest ale mniej widoczna.





Poprawa nie jest zauważalna - wyniki są na poziomie metody xaviera.

4 Wnioski

4.1 Optymaliztory współczynnika uczenia

Dobranie współczynnika uczenia nie jest prostym zadaniem - jeśli jest zbyt duży proces uczenia może okzać się niezbieżny. Zbyt mały z kolei może skutkować bardzo wolnym uczeniem się sieci. Największy wpływ na dobór współczynnika uczenia ma kształt funkcji kosztu, a ten jest bardzo zmienny. Należy zatem dostosować ten współczynnik w trakcie uczenia.

Momentum pomaga metodzie SGD na wyjście z minimum lokalnego biorąc pod uwagę prędkość w poprzednim kroku. Momentum Nesterova dodatkowo bierze pod uwagę przewidywanie następnej pozycji. Adagrad zmienia współczynnik uczenia w zależności od parametrów - sumuje kwadraty wszystkich gradientów. Adadelta ogranicza sumowanie gradientów do pewnego rozmiaru. Adam używa zarówno poprzednich gradientów, jak i przeszłych kwadratów gradientów.

Eksperymenty wskazują, że wszystkie optymalizatory znacząco ulepszają prędkość jak i skuteczność nauki sieci. Najlepsze wyniki osiąga optymalizator Adam.

4.2 Metody inicjalizacji wag

Jest bardzo istotne aby podczas projektowania sieci neuronowej wziąć pod uwagę sposób inicjalizacji wag. Inicjalizacja wag jest punktem początkowym uczenia i może zaważyć na skuteczności sieci. Złe dobranie wag może pogorszyć skuteczność sieci, a nawet uniemożliwić naukę. Zbyt niskie wagi mogą potencjalnie bardzo spowolnić uczenie sieci - problem zanikającego gradientu. Z kolei zbyt wysoka inicjalizacja może skutkować bardzo dużymi liczbami w trakcie uczenia, które będą powodowały przekroczenie zakresu liczb w komputerze - problem wybuchającego gradientu.

Generalnie wagi powinny być inicjalizowane jako niewielkie wartości w pobliżu 0. Wybór rozkładu normalnego o odchyleniu standardowym = 0.1 jest prawidłowe.

Metoda Xaviera ma na celu zachować stałą wariancję od warstwy do warstwy w obu kierunkach, aby sieć mogła uczyć się optymalnie.

Ze względu na to, że funkcja ReLU zwraca wynik 0 dla wartości ujemnych - połowa wariancji będzie taka sama jak w przypadku metody xaviera, a połowa będzie równa 0. Należało zatem zmodyfikować wzór, aby przyjmował liczbę wejść zmniejszoną o połowę.

Eksperymetny wskazują, że wykorzystanie wymienionych metod okazuje się być lepsze, niż teoretycznie poprawne wykorzytanie samego rozkłądu normalnego o niskim odchyleniu standardowym. Metoda He została stworzona jako lepsza alternatywa dla metody xaviera w przypadku wykorzystania funkcji ReLU. Wyniki eksperymentów są jednak na tym samym poziomie co metoda xaviera - badania autora wykazały lepszą skuteczność na bardzo głębokich sieciach, dlatego tutaj poprawa może się nie uwidaczniać.