# Analiza i Ewaluacja Modeli Perceptronu Wielowarstwowego
# Opis zadań NN1 - NN6

Mikołaj Mróz

11 kwietnia 2024

## 1 Wprowadzenie

Seria eksperymentów z perceptronem wielowarstwowym (MLP) miała na celu zbadanie wpływu różnych konfiguracji i hiperparametrów na proces uczenia. Celem było nie tylko osiągnięcie wysokiej skuteczności w zadaniach predykcyjnych, ale także dogłębna analiza dynamiki adaptacyjnych właściwości sieci neuronowych w kontekście modyfikacji ich hiperparametrów.
*Wszystkie wartości MSE oraz wizualizacje zbieżności MSE są liczone na zbiorze testowym

# 2 NN1: Implementacja Podstawowa

## 2.1 Cel Eksperymentu

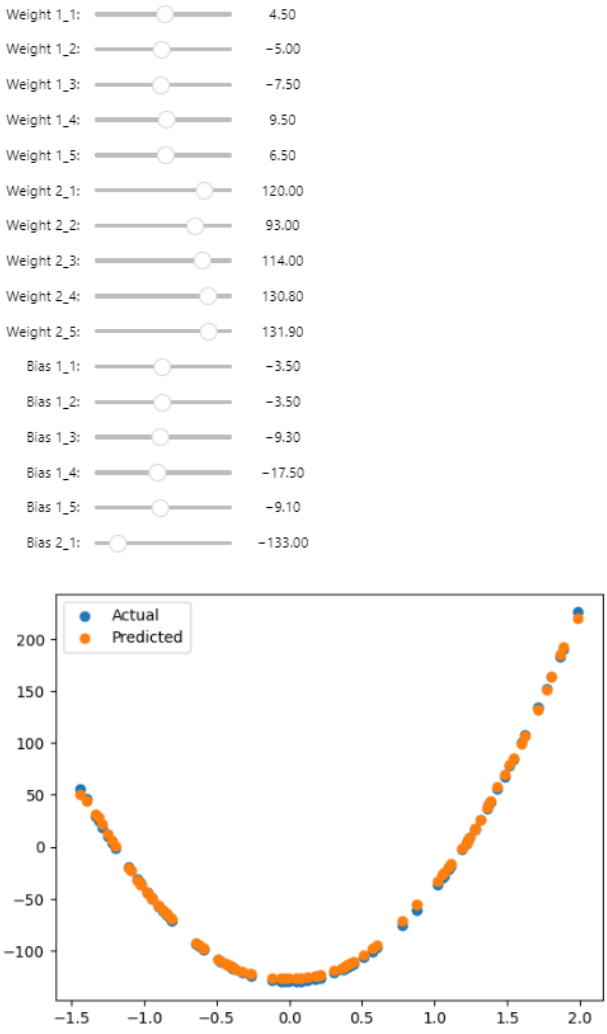
Początkowy etap badań poświęcono na konstrukcję fundamentalnej struktury MLP, co umożliwiło dogłębne zrozumienie mechanizmów sieci feedforward oraz eksplorację wpływu liczby warstw, neuronów i rodzajów funkcji aktywacji na efektywność modelu poprzez serię eksperymentalnych iteracji.

## 2.2 Eksperymenty

Pierwsze zadanie skoncentrowano na analizie danych (square-simple, steps-large), gdzie jeden zbiór reprezentował funkcję kwadratową, a drugi - funkcję schodkową.

### 2.2.1 Analiza Zbioru square-simple

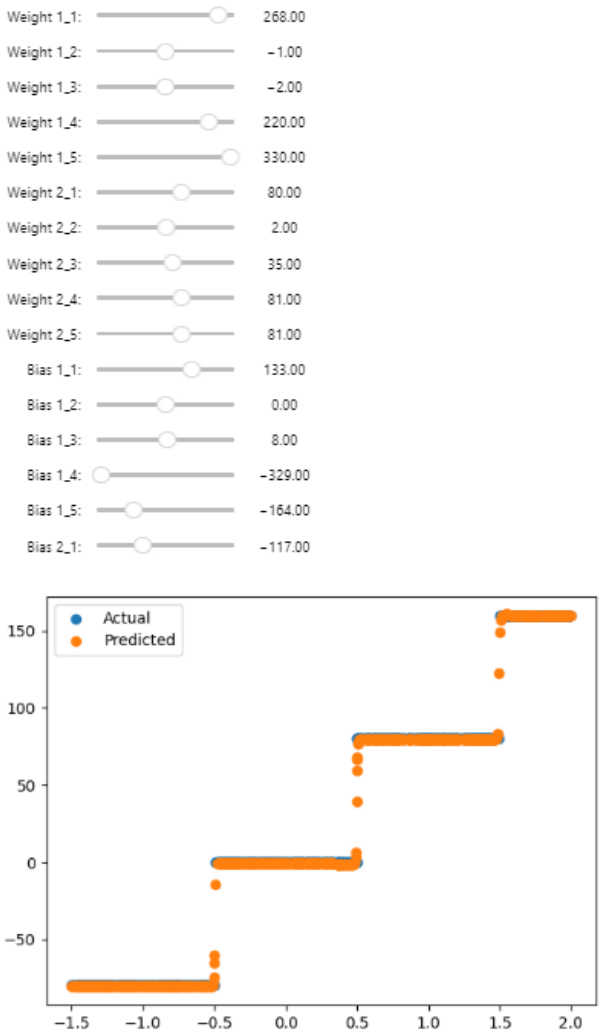
Architektura [1,5,1] została wybrana w celu dopasowania modelu do danych. Zostało opracowane narzędzie do manualnego dostosowywania wag i biasów, co pozwoliło na maksymalizację dopasowania do danych oraz obliczenie MSE (Mean Squared Error) dla zbioru testowego.




Rysunek 1: Porównanie danych zbioru square-simple z predykcjami sieci neuronowej (MSE: 4.50092863)

### 2.2.2 Analiza Zbioru steps-large

Podobne podejście zostało zastosowane do zbioru steps-large, wykorzystując architekturę [1,5,1].




Rysunek 2: Porównanie danych zbioru steps-large z predykcjami sieci neuronowej (MSE: 5.5821970035)

# 3 NN2: Implementacja Backpropagation

## 3.1 Cel Eksperymentu

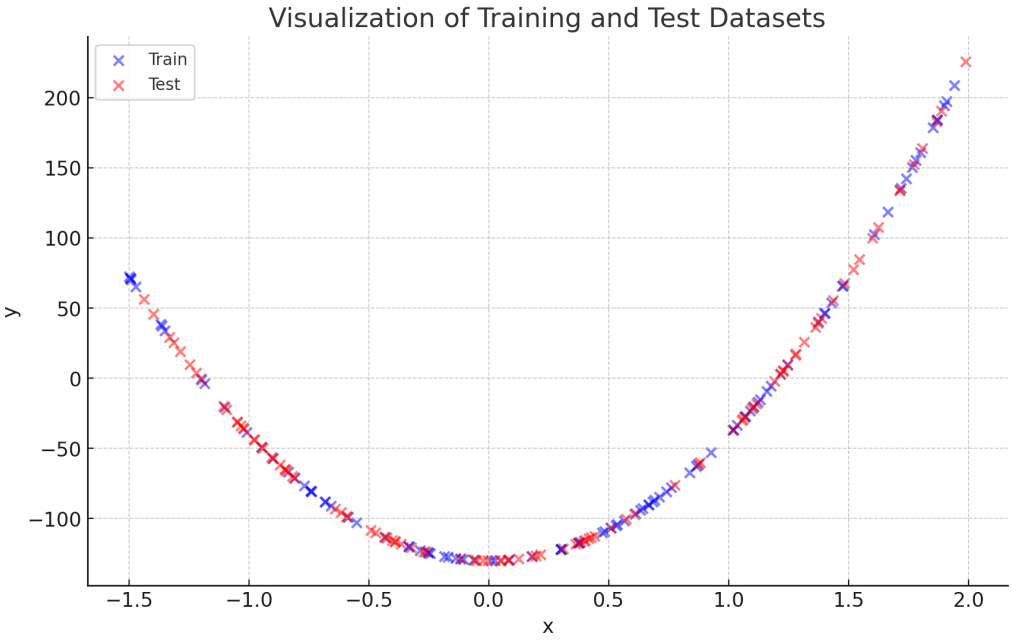
Integracja mechanizmu propagacji wstecznej umożliwiła analizę adaptacyjnych właściwości sieci neuronowych poprzez bezpośrednie obserwacje zmian wag w procesie uczenia, oferując wgląd w proces optymalizacji sieci.

## 3.2 Eksperymenty

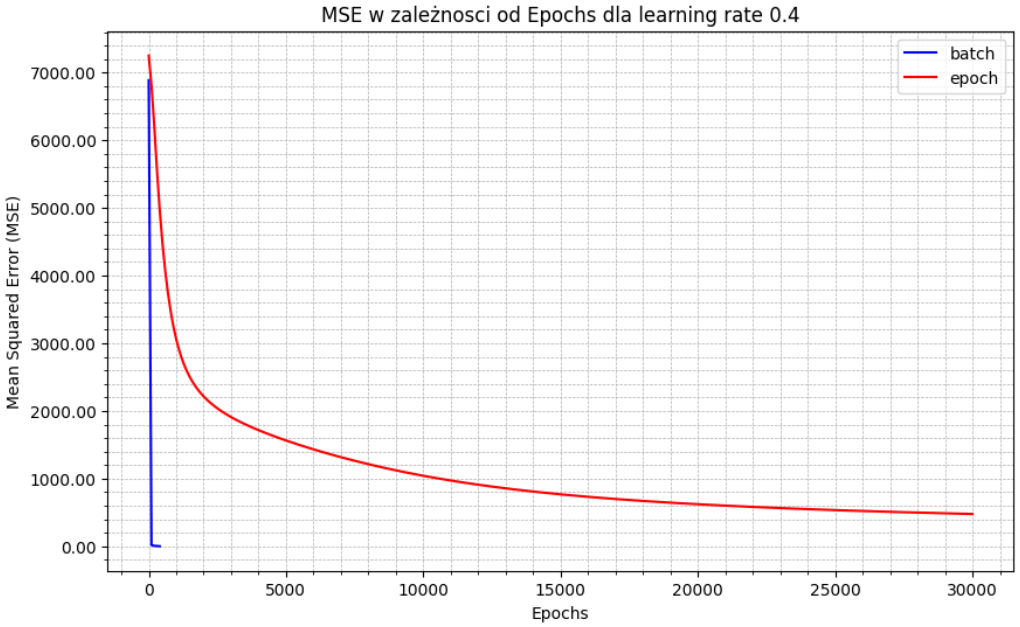
W eksperymentach skupiono się na analizie procesu uczenia przy użyciu propagacji wstecznej, zbadano wpływ wielkości batcha na szybkość i efektywność uczenia.

### 3.2.1 Zbiór square-simple

Model szybko adaptował się do prostego zbioru, co zostało zademonstrowane na serii eksperymentów.



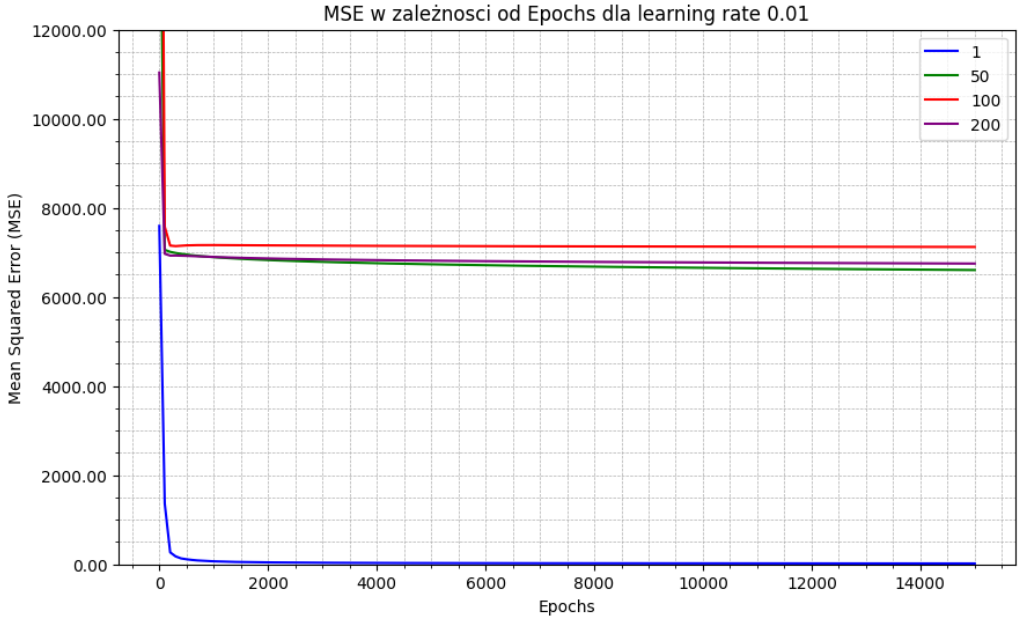

Rysunek 3: Analiza zbioru danych square-simple



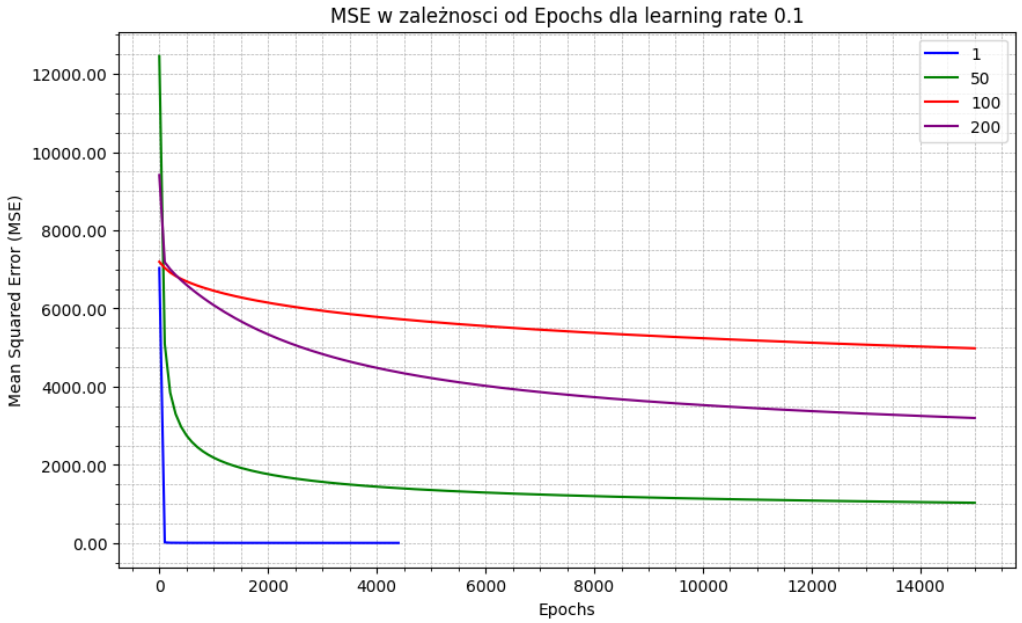

Rysunek 4: Analiza wpływu batch vs epoch na proces uczenia dla zbioru square-simple

Eksperymenty porównywały różne wartości współczynnika uczenia przy stałej architekturze sieci [1,5,5,1].

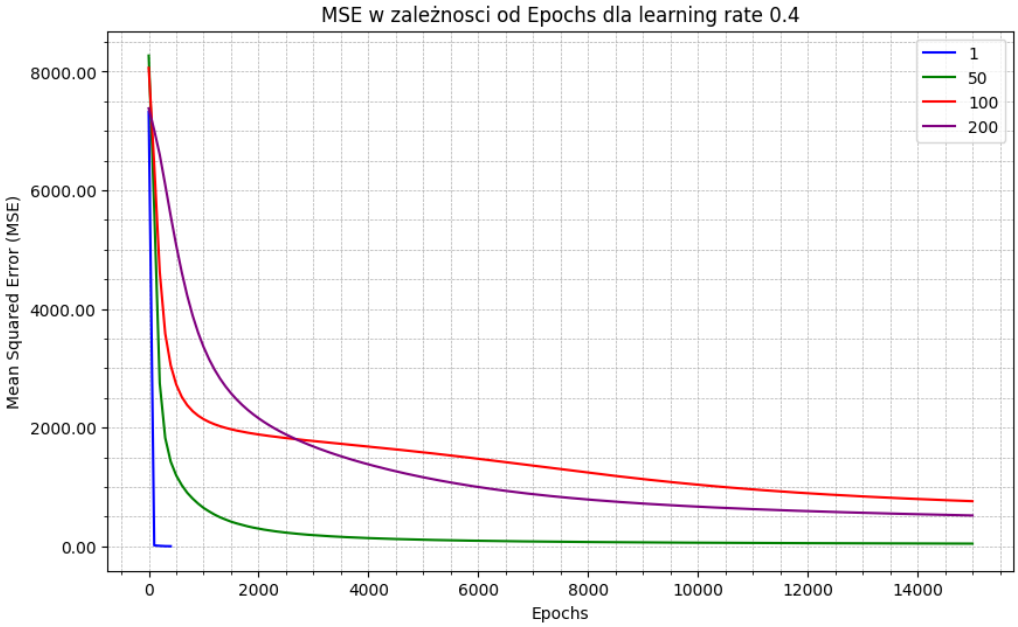
Następnie skupiłem się na porównaniu szybkości zbiegania MSE przy różnych learning rate i dla różnych rozmiarów batcha.




Rysunek 5: Analiza uczenia przy współczynniku 0.01 (MSE: 19.66143110519924)



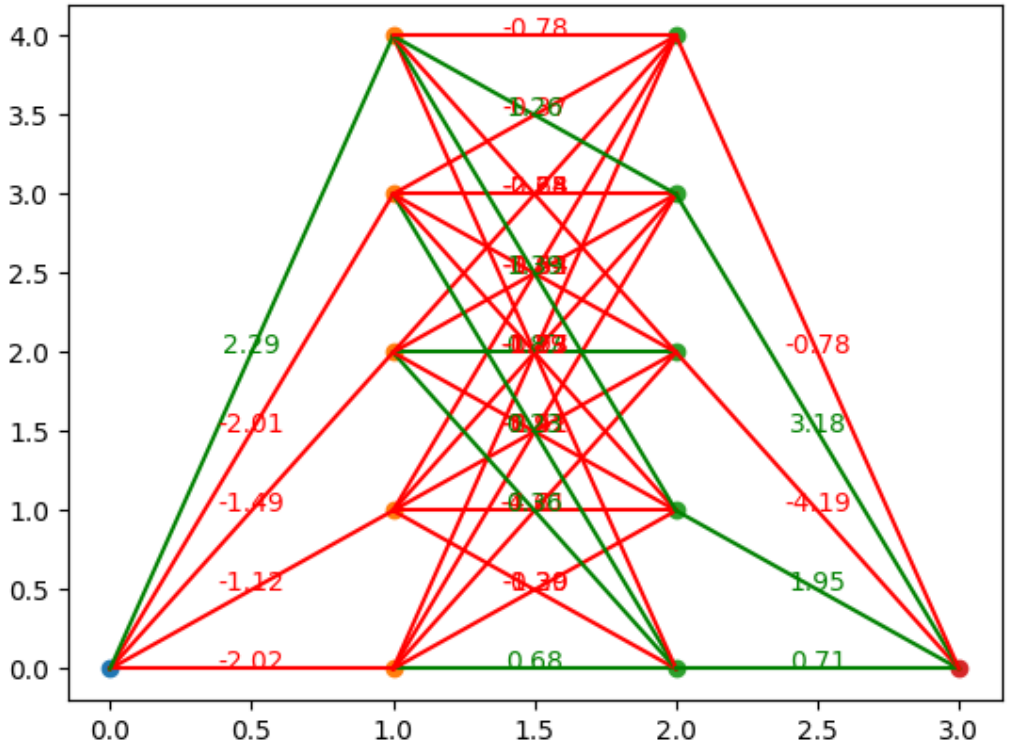

Rysunek 6: Analiza uczenia przy współczynniku 0.1 (MSE: 3.956566)




Rysunek 7: Analiza uczenia przy współczynniku 0.4 (MSE: 3.23299)

Na podstawie analizy wyników i wykresów uzyskanych z przeprowadzonych eksperymentów, można wyciągnąć kilka istotnych wniosków dotyczących efektywności uczenia sieci neuronowych. Podejście oparte na aktualizacjach batchowych wykazuje lepszą efektywność w porównaniu do aktualizacji po każdej epoce, co sugeruje, że częstsze aktualizacje wag na mniejszych podzbiorach danych mogą przyczyniać się do efektywniejszego uczenia. Wynika to z lepszego unikania minimum lokalnych, co jest szczególnie widoczne przy mniejszych wartościach współczynnika uczenia, gdzie sieć często utyka, nie osiągając lepszych wyników. Ponadto, odpowiedni dobór rozmiaru batcha jest kluczowy, ponieważ zbyt mały rozmiar może prowadzić do niestabilności w procesie uczenia, podczas gdy zbyt duży może maskować użyteczne szczegóły w danych. Najlepszy wynik błędu średniokwadratowego (MSE), który osiągnięto w badaniach, wynosił 3.23299, podkreślając znaczenie optymalizacji hiperparametrów dla osiągnięcia maksymalnej wydajności modelu.

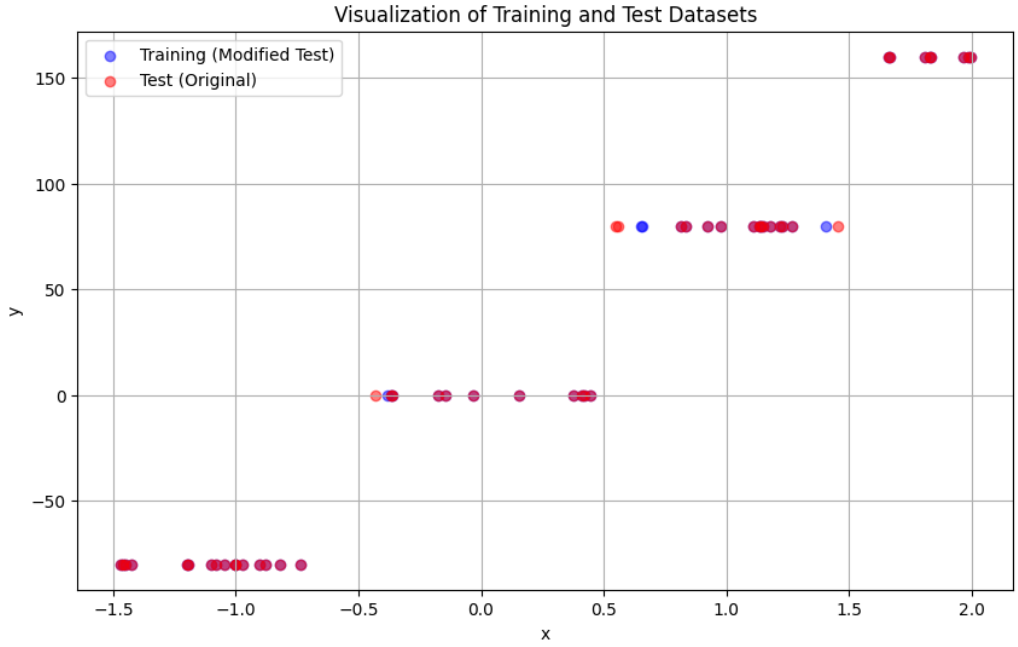
Wizualizacja najlepiej dopasowanej sieci:



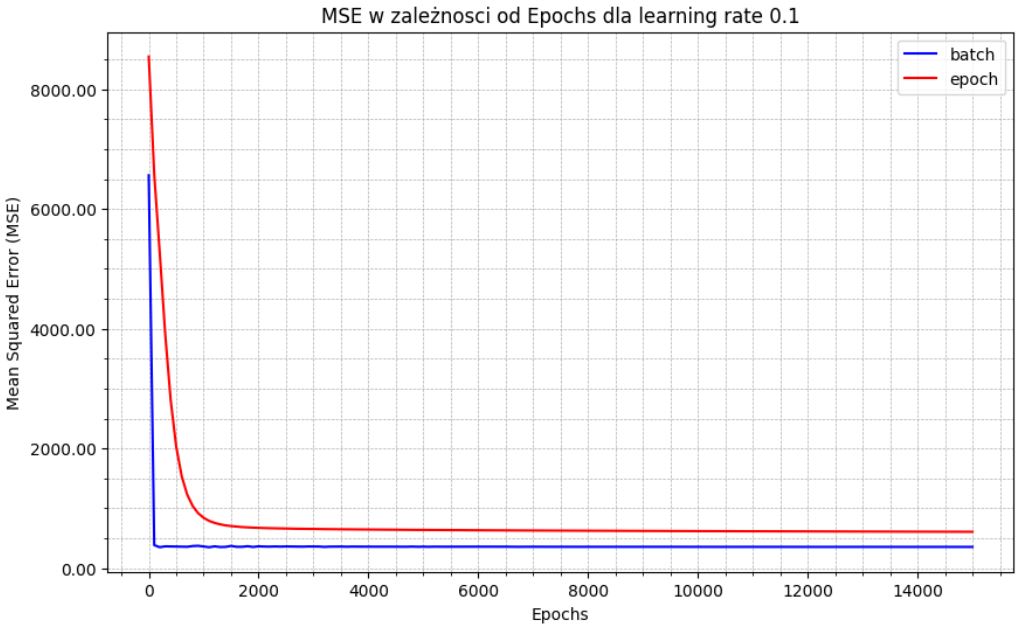

Rysunek 8: Wizualizacja optymalnej sieci (MSE: 3.23299)

### 3.2.2 Zbiór steps-small

Analiza wykazała, że zbiór steps-small stanowił wyzwanie dla modelu, co uwidoczniło ograniczenia w adaptacji przy zastosowanej konfiguracji.

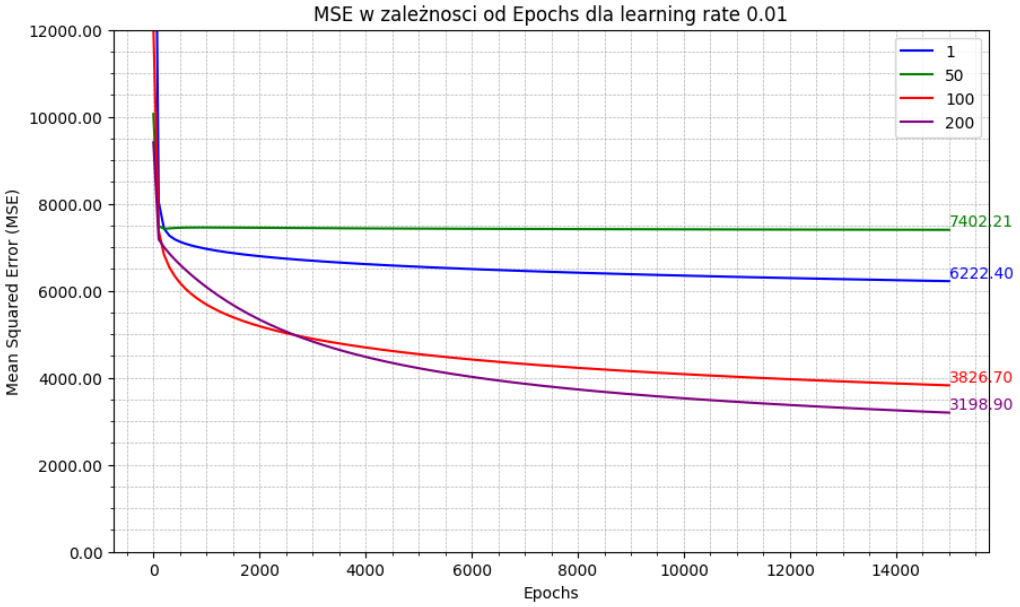



Rysunek 9: Analiza zbioru danych steps-small

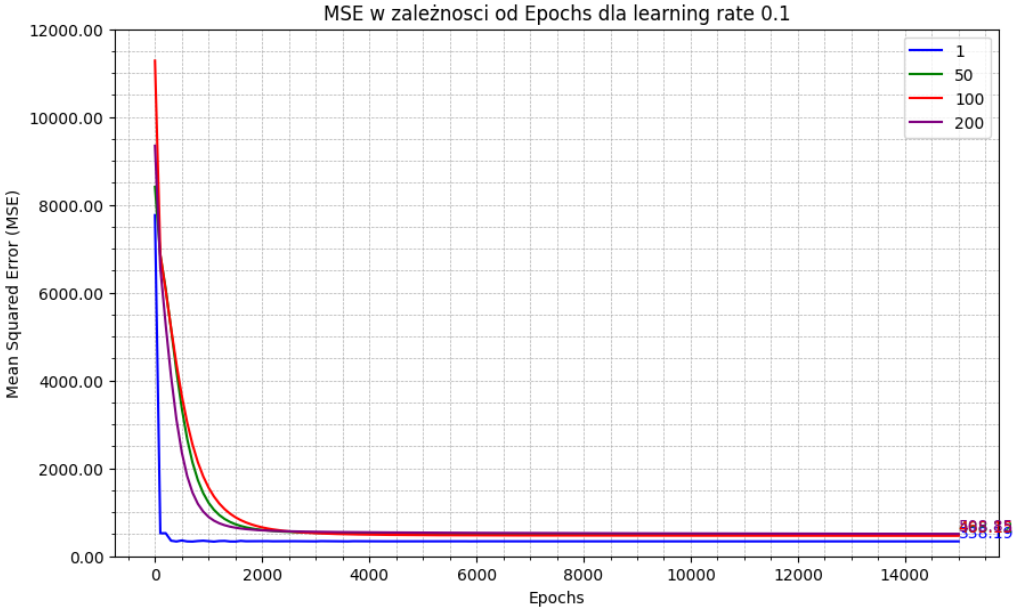



Rysunek 10: Analiza wpływu batch vs epoch na proces uczenia dla zbioru steps-small

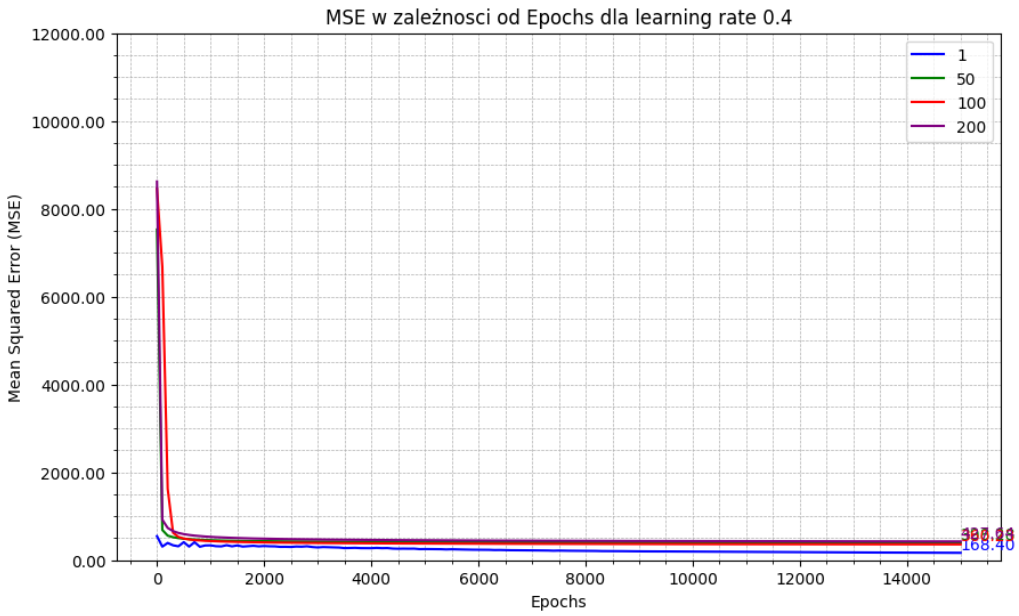
Rezultaty eksperymentów dla zbioru steps-small ukazały stabilizację MSE, sugerując ograniczenia w zdolności sieci do dalszej adaptacji.




Rysunek 11: Analiza uczenia przy współczynniku 0.01




Rysunek 12: Analiza uczenia przy współczynniku 0.1



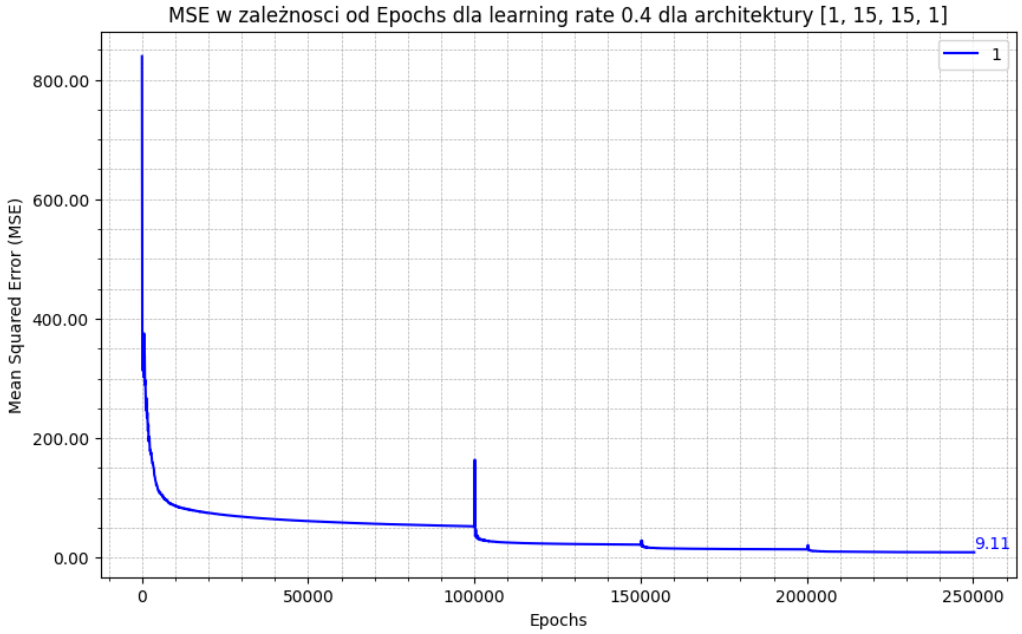

Rysunek 13: Analiza uczenia przy współczynniku 0.4

Eksperymenty z bardziej zaawansowaną architekturą [1, 15, 15, 1] wykazały ograniczone możliwości adaptacji modelu do zbioru steps-small.

Analiza wyników dla zestawu danych steps-small wskazuje na obiecującą tendencję spadkową błędu podczas procesu uczenia, co sugeruje, że sieć neuronowa skutecznie adaptuje się do danych. Wykresy pokazują, że odpowiednio dobrana kombinacja rozmiaru batcha, architektury sieci oraz współczynnika uczenia ma kluczowe znaczenie dla optymalizacji wyników. Najlepszy uzyskany wynik MSE, który wynosi 9.11, został osiągnięty przy specyficznej konfiguracji tych parametrów, co zostało wyraźnie zademonstrowane na wykresach.

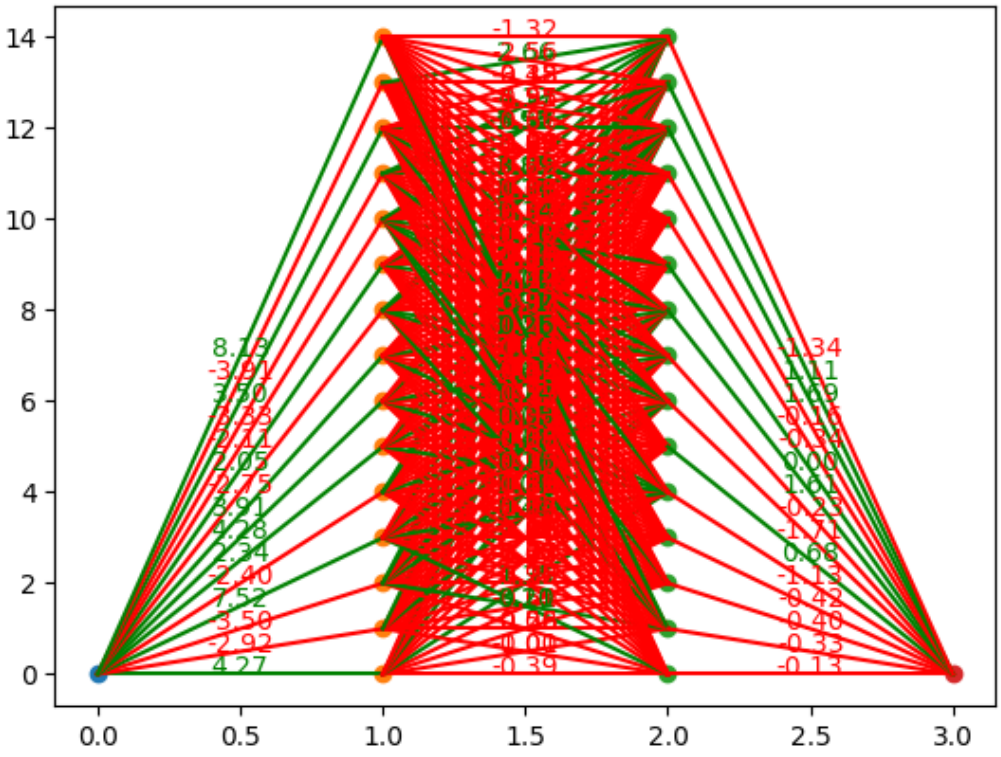
Ponadto, eksperymenty potwierdzają, że dobór odpowiedniego rozmiaru batcha jest krytycznym elementem w procesie uczenia, wpływającym na szybkość i stabilność konwergencji. Zbyt duży lub zbyt mały rozmiar batcha może negatywnie wpłynąć na proces uczenia, co podkreśla potrzebę precyzyjnej kalibracji tego parametru w zależności od specyfiki danych i architektury sieci.

Poniżej wykres finalnego uczenia, które przyniosło najlepsze rezultaty.




Rysunek 14: Analiza uczenia z architekturą [1, 15, 15, 1] (MSE: 9.11)

Wizualizacja sieci o zaawansowanej architekturze ilustruje skomplikowaną strukturę modelu, co jest wynikiem zastosowania znaczącej liczby neuronów. Prezentacja ta ma na celu jednorazowe ukazanie graficznej reprezentacji sieci neuronowej zwiększonej złożoności, co pozwala na wizualną ocenę jej konstrukcji.

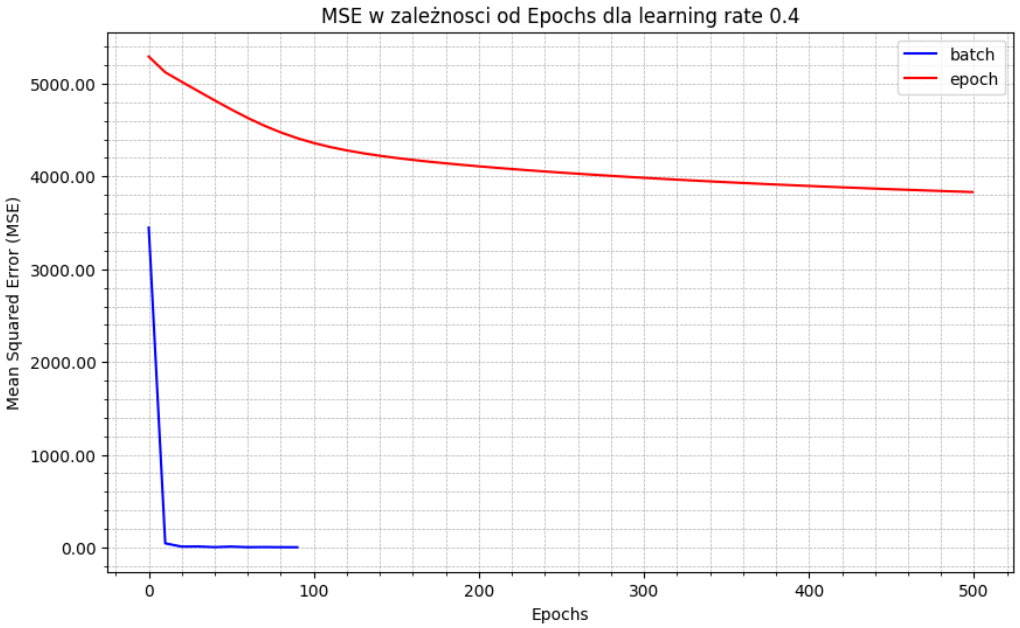



Rysunek 15: Wizualizacja zaawansowanej sieci

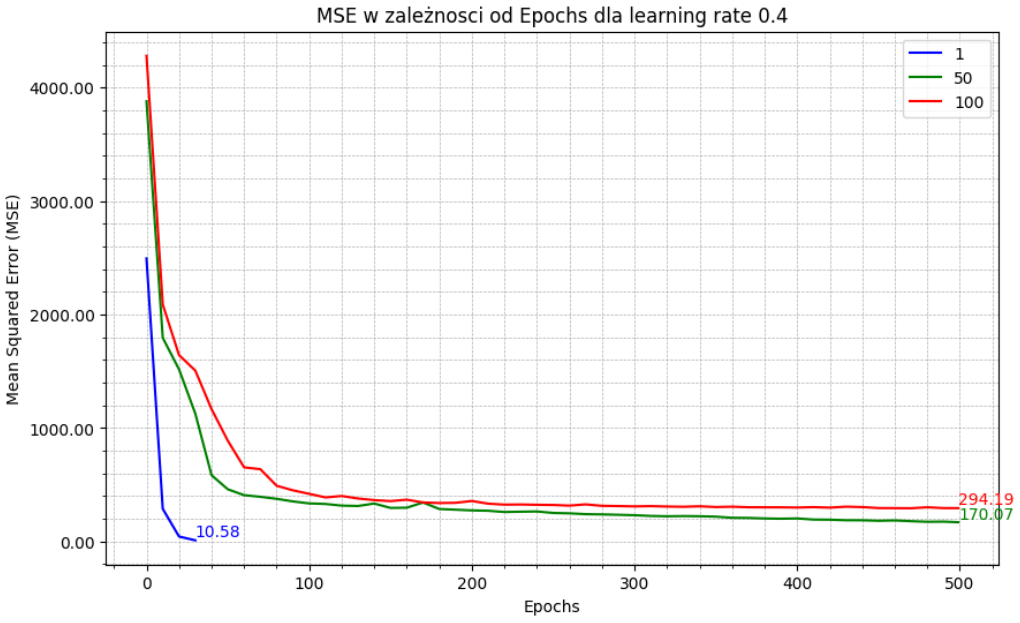
### 3.2.3 Zbiór multimodal-large

Eksperymenty na zbiorze multimodal-large prezentowały wyzwania adaptacyjne dla modelu, co zostało zademonstrowane w serii testów.

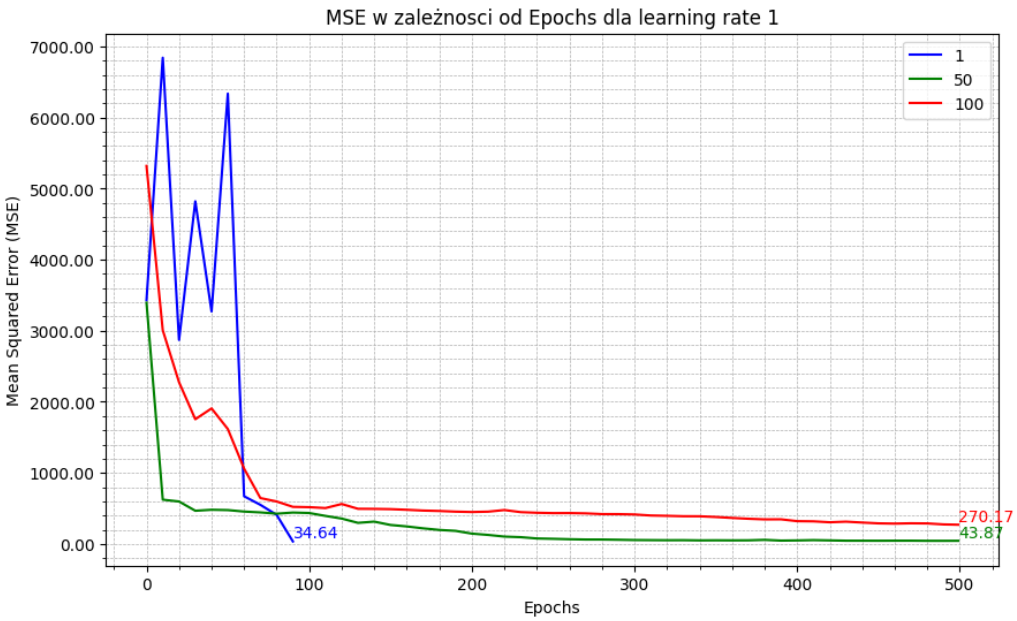
Wyniki eksperymentów dla różnych konfiguracji sieci ukazały różnice w szybkości i efektywności uczenia.




Rysunek 16: Analiza wpływu batch vs epoch na proces uczenia dla zbioru multimodal-large

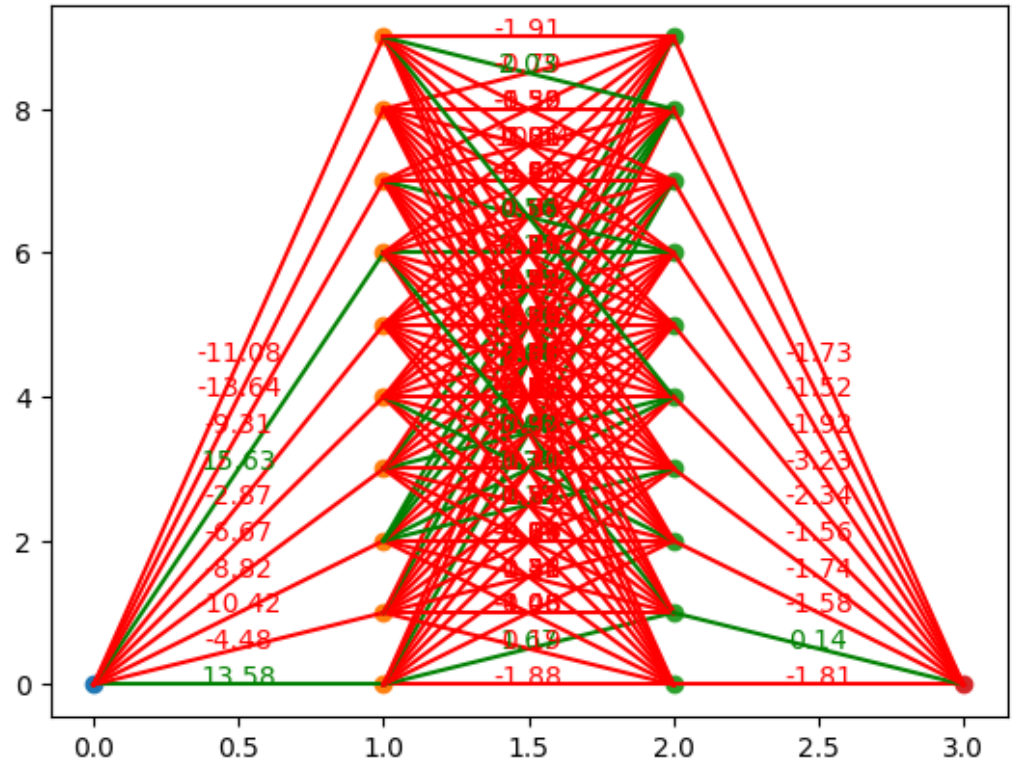



Rysunek 17: Analiza uczenia przy współczynniku 0.4




Rysunek 18: Analiza uczenia przy współczynniku 1

Optymalna konfiguracja sieci o architekturze [1, 10, 10, 1] zastosowana do zbioru danych multimodal-large pozwoliła osiągnąć wartość błędu średniokwadratowego (MSE) równą 10.88. Na górnym wykresie zaprezentowano proces uczenia tej sieci, natomiast na poniższej ilustracji przedstawiono wizualizację samej architektury sieci. Obie grafiki razem ilustrują efektywność oraz dynamikę uczenia dla wybranej konfiguracji sieci neuronowej zastosowanej do tego zestawu danych.




Rysunek 19: Wizualizacja sieci, która osiągneła najlepsze rezultaty (MSE: 10.88)

# 4 NN3: Implementacja Technik Optymalizacji Momentu i RMSProp

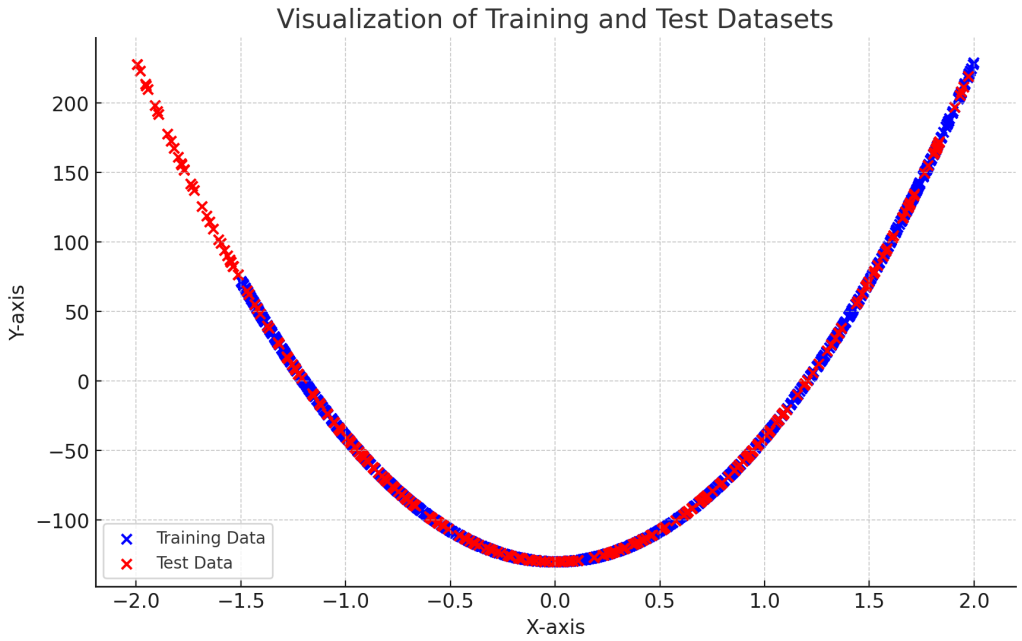
## 4.1 Opis Eksperymentu

W ramach niniejszego badania podjęliśmy analizę zaawansowanych metod optymalizacyjnych, takich jak moment (Momentum) oraz RMSProp, mających na celu usprawnienie procesu uczenia sieci neuronowych. Przeanalizowano wpływ tych metod na szybkość zbiegania do minimum funkcji kosztu, korzystając z różnorodnych zestawów danych, w celu oceny skuteczności wprowadzonych modyfikacji.

## 4.2 Przeprowadzone Eksperymenty

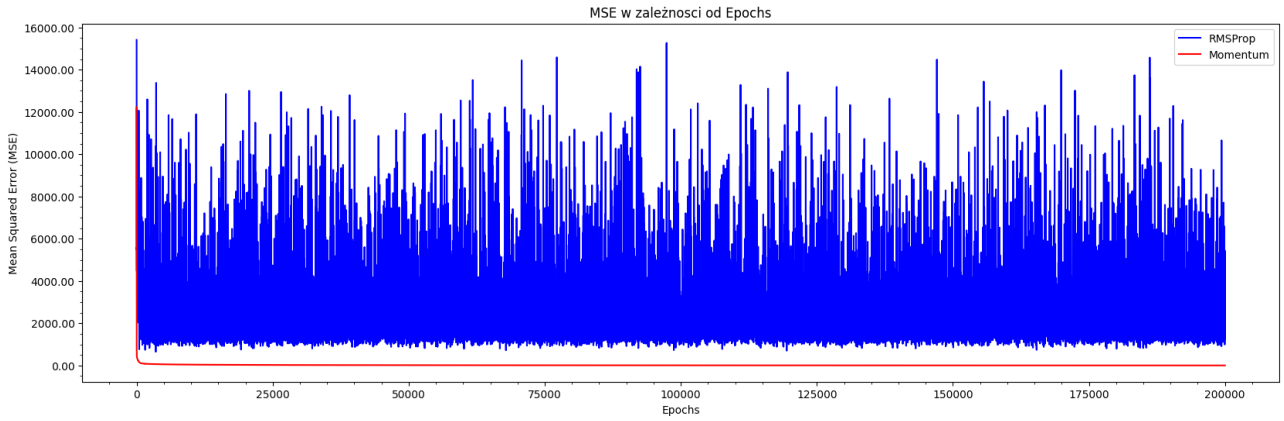
Wyniki analizy porównawczej dwóch omawianych technik optymalizacji przedstawiono poniżej, koncentrując się na ich wpływie na proces uczenia:

### 4.2.1 Analiza Zestawu Danych square-large




Rysunek 20: Analiza zbioru danych square-large

Dla konfiguracji sieci [1, 5, 1] z szybkością uczenia 0.4 i rozmiarem batcha 10, technika Momentu wykazała wyższą stabilność i skuteczność, osiągając wartość MSE równe 0.7565299005584123. Technika RMSProp w tym kontekście wykazała się niestabilnością, co objawiało się fluktuacjami wartości MSE.

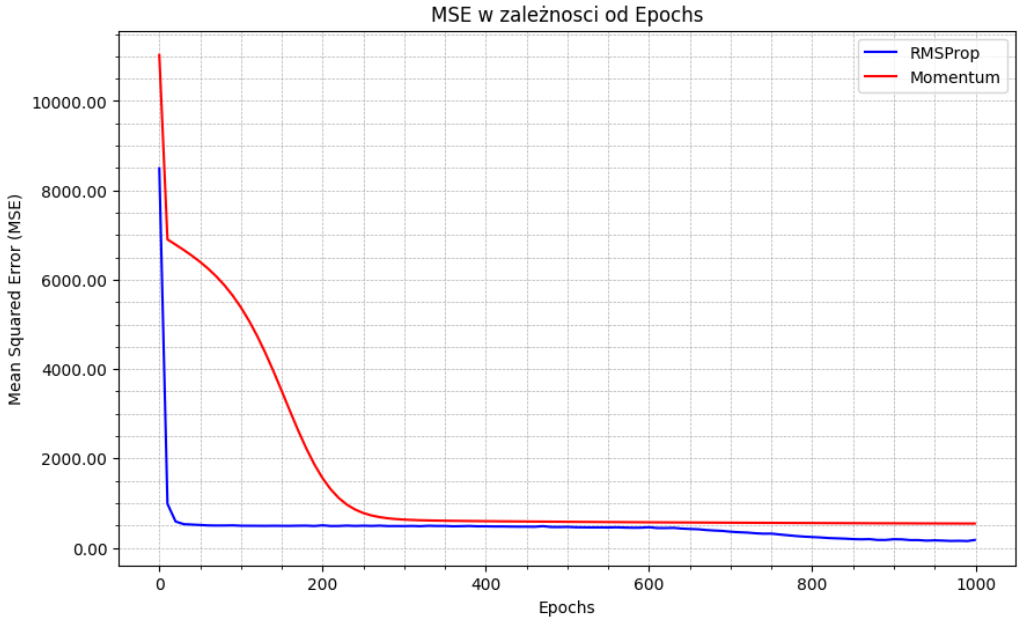



Rysunek 21: Dynamika zmian wartości MSE dla zestawu danych square-large

### 4.2.2 Analiza Zestawu Danych steps-large

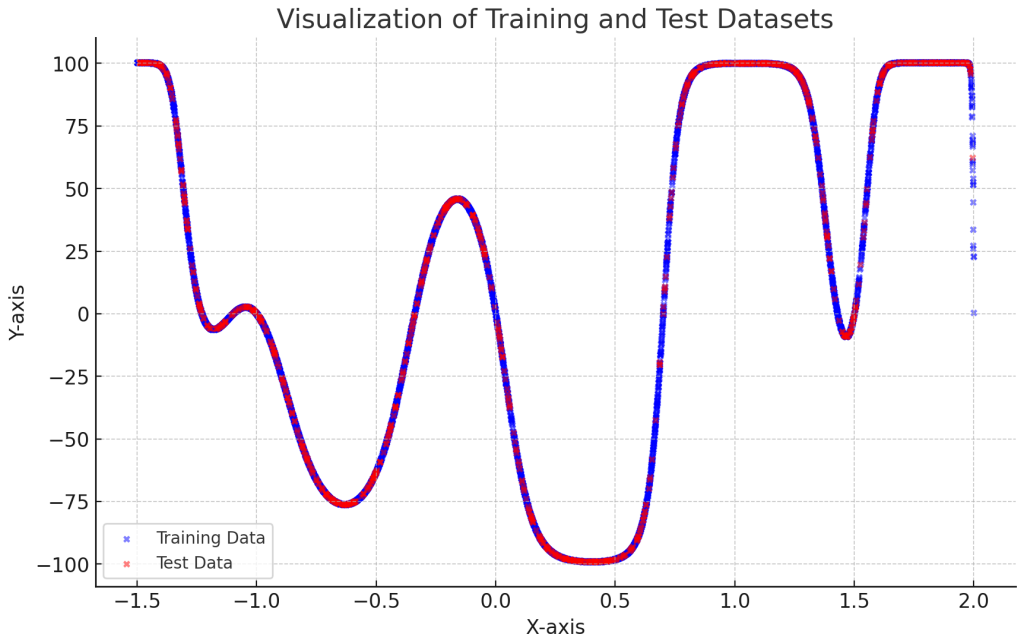

Rysunek 22: Analiza zbioru danych steps-large

Zestaw danych steps-large, ze względu na swoją specyfikę, stanowił wyzwanie dla procesu uczenia. Obie metody optymalizacji osiągnęły porównywalne rezultaty, jednakże finalne wartości MSE, na poziomie 175.54604196125257, wskazywały na trudności w dopasowaniu modelu. Eksperyment przeprowadzono dla sieci o architekturze [1, 5, 5, 1], z szybkością uczenia 0.01 i rozmiarem batcha stanowiącym 10% wielkości zbioru danych.



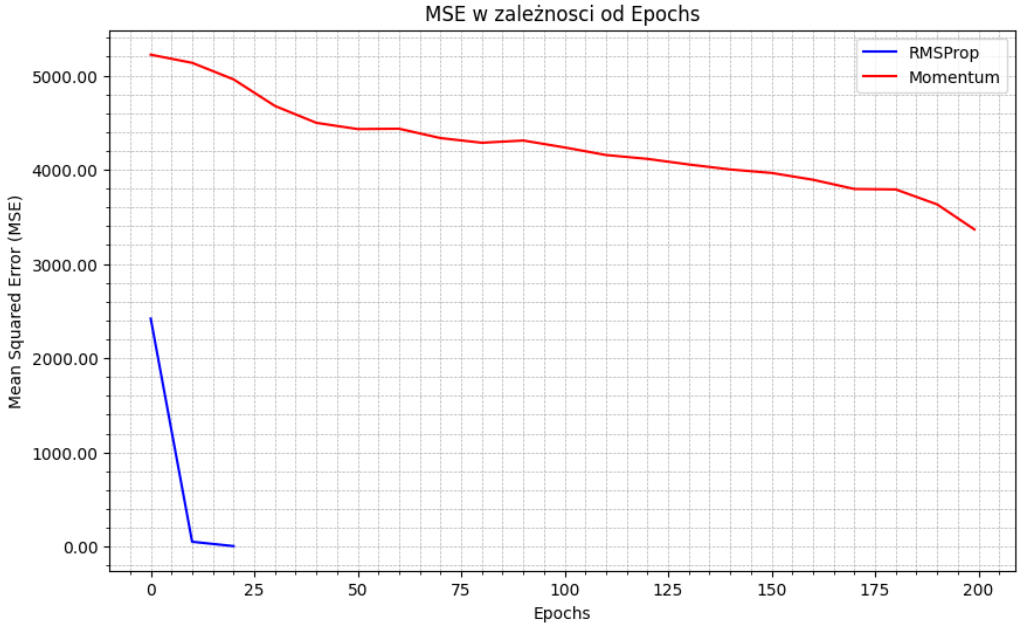

Rysunek 23: Dynamika zmian wartości MSE dla zestawu danych steps-large

### 4.2.3 Analiza Zestawu Danych multimodal-large




Rysunek 24: Analiza zbioru danych multimodal-large

W przypadku zestawu multimodal-large, zastosowanie RMSProp znacząco przyspieszyło proces uczenia, osiągając wartość MSE równą 6.118067625089367 już po około 20 iteracjach. W porównaniu, technika Momentu nie była w stanie konkurować z takim wynikiem efektywności. Analiza została przeprowadzona dla sieci o konfiguracji [1, 64, 32, 32, 1], przy szybkości uczenia 0.01 i rozmiarze batcha 10.



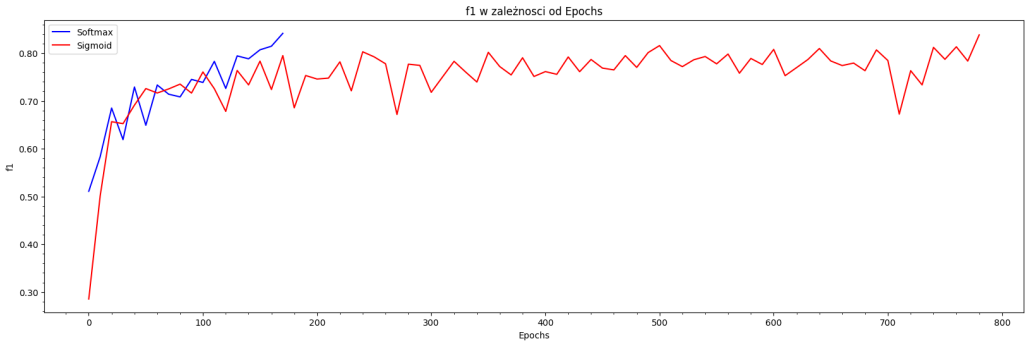

Rysunek 25: Dynamika zmian wartości MSE dla zestawu danych multimodal-large

# 5 NN4: Zadanie Klasyfikacji

## 5.1 Cel Eksperymentu

Implementacja funkcji softmax na warstwie wyjściowej pozwoliła na adaptację modelu do zadań klasyfikacji, co umożliwiło porównanie skuteczności sieci z różnymi funkcjami aktywacji. Eksperymenty na zbiorach rings3-regular, easy i xor3 ukazały przewagi podejścia z funkcją softmax na ostatniej warstwie w kontekście efektywności i szybkości uczenia.

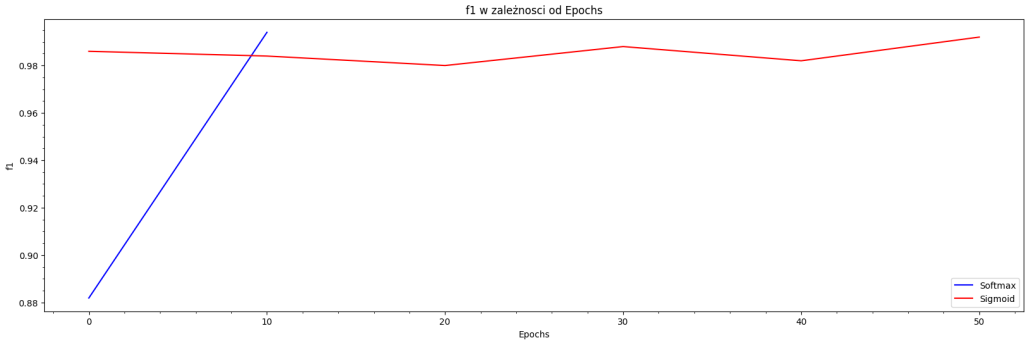
## 5.2 Zbiór rings3-regular




Rysunek 26: Porównanie efektywności klasyfikacji dla zbioru rings3-regular

Wykres przedstawia porównanie skuteczności funkcji aktywacji Softmax i Sigmoid w kontekście klasyfikacji zbioru rings3-regular. Z wykresu wynika, że funkcja Softmax osiąga wyższe wartości miary f1 szybciej niż Sigmoid, co sugeruje jej większą efektywność w tym zadaniu. Ponadto, Softmax wydaje się zachowywać stabilniej w kolejnych epokach uczenia, co może wskazywać na lepsze właściwości tej funkcji w klasyfikacji wieloklasowej dla danego zbioru danych. Najlepszy osiągnięty wynik f1 to 0.841376.

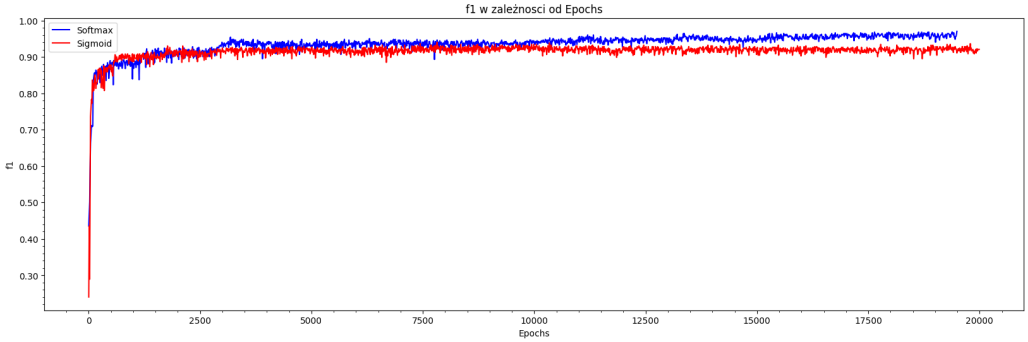
## 5.3 Zbiór easy




Rysunek 27: Porównanie efektywności klasyfikacji dla zbioru easy

Podobnie jak w przypadku poprzedniego zbioru danych, wykres dla zbioru easy ukazuje, że funkcja aktywacji Softmax szybko osiąga wysoką wartość miary f1 i utrzymuje stabilność przez cały proces uczenia. W porównaniu, Sigmoid pokazuje wolniejszą konwergencję, co ponownie podkreśla przewagę Softmax w efektywności klasyfikacji dla prezentowanego zbioru. Najlepszy osiągnięty wynik f1 to 0.9940000720025921.

## 5.4 Zbiór xor3




Rysunek 28: Porównanie efektywności klasyfikacji dla zbioru xor3

Na wykresie dla zbioru xor3 obserwujemy, że obie funkcje aktywacji, Softmax i Sigmoid, osiągają podobne wartości miary f1 i utrzymują stabilność na wysokim poziomie przez większość procesu uczenia. Oba podejścia konwergują do podobnej wydajności, co wskazuje na to, że dla tego konkretnego zbioru danych różnica między tymi funkcjami aktywacji nie jest znacząca.

# 6 NN5: Testowanie różnych funkcji aktywacji

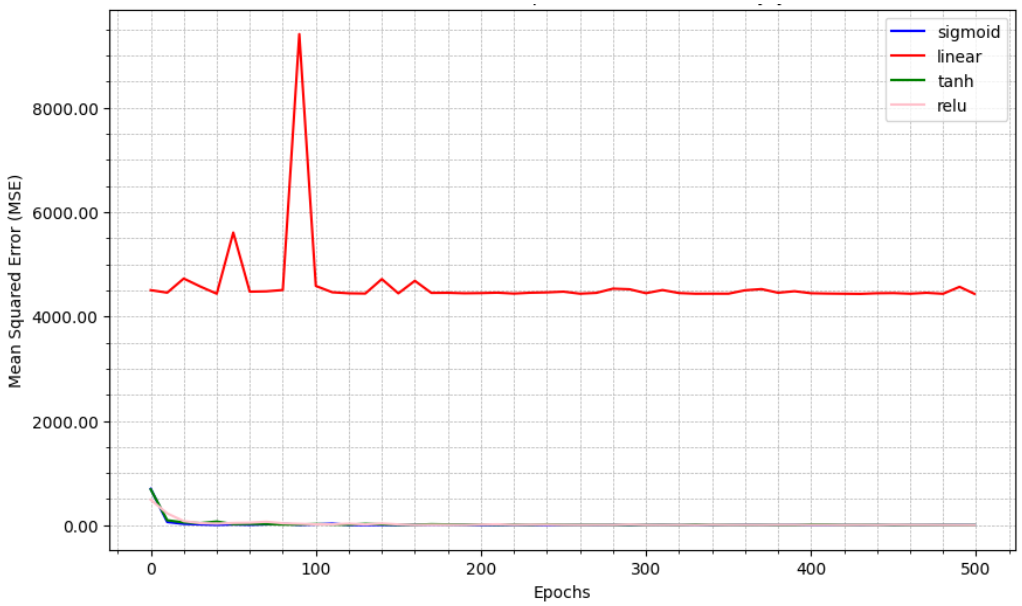
## 6.1 Opis ćwiczenia

Analiza różnych funkcji aktywacji, takich jak sigmoidalna, liniowa, tanh, oraz ReLU, pozwoliła na głębsze zrozumienie ich wpływu na proces uczenia sieci neuronowych. Poprzez porównanie szybkości uczenia i skuteczności modelu w zależności od wybranej funkcji aktywacji, zyskaliśmy wgląd w optymalne strategie konstruowania sieci dla różnych zadań regresji i klasyfikacji.

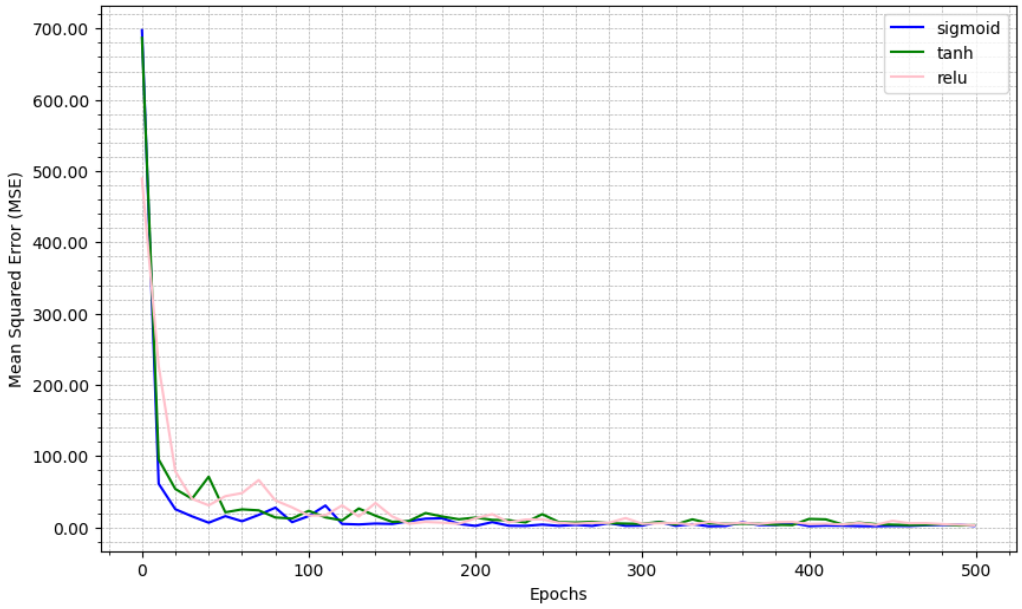
## 6.2 Zbiór multimodal-large

W tym zbiorze zrobiłem porównanie różnych funkcji aktywacji, dla trzech, dwóch i jednej warstwy ukrytej. Poniżej porównania:

### 6.2.1 3 warstwy ukryte



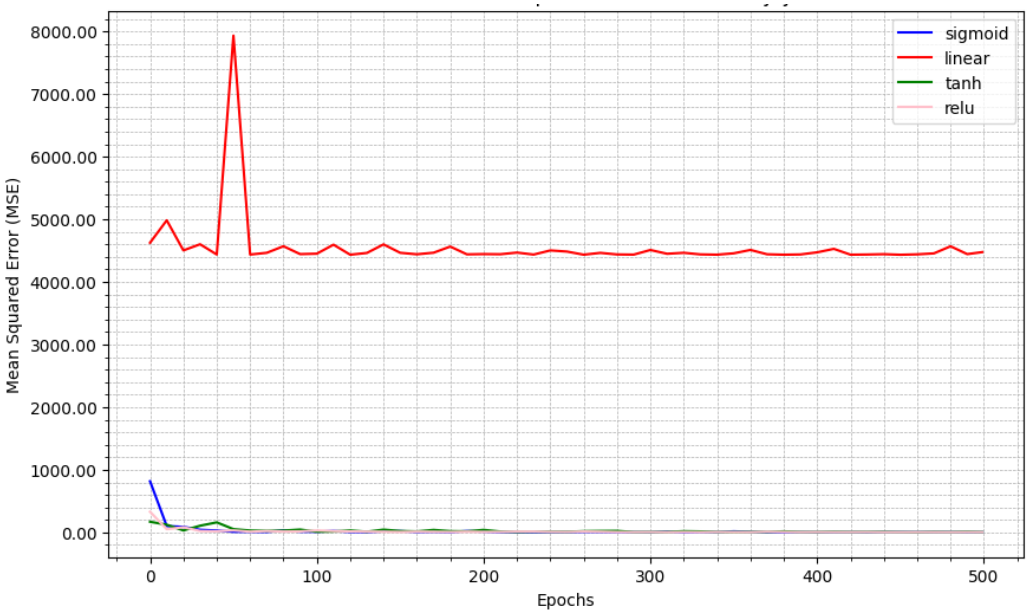

Rysunek 29: Porównanie szybkości uczenia w zależności od funkji aktywacji



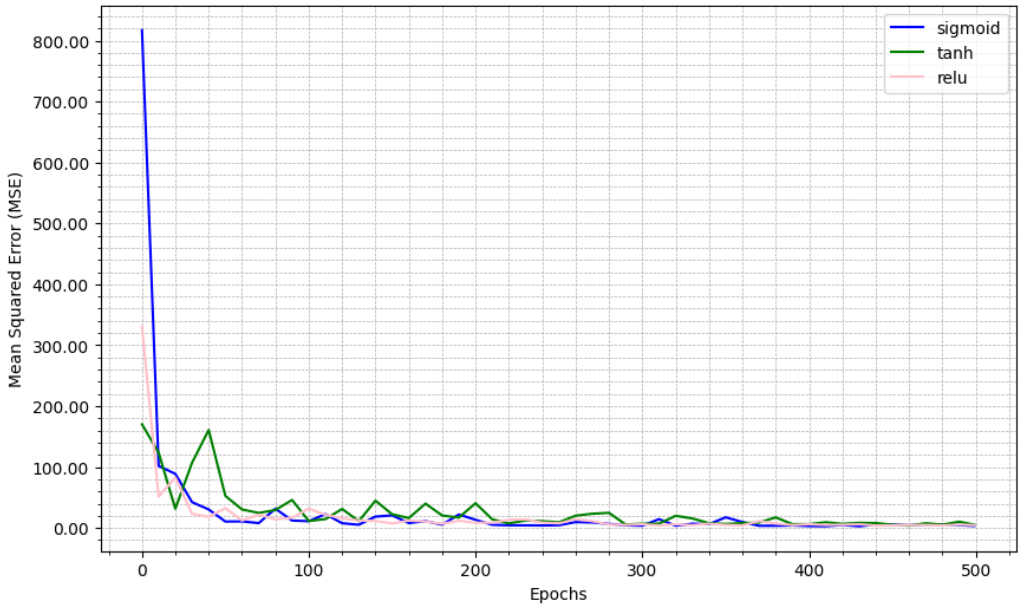

Rysunek 30: Porównanie szybkości uczenia w zależności od funkji aktywacji dla trzech najlepszych wariantów

Wykresy prezentują dynamikę błędu średniokwadratowego (MSE) dla sieci neuronowej o architekturze [1, 64, 32, 32, 1] z trzema warstwami ukrytymi, uczącej się z użyciem różnych funkcji aktywacji: sigmoidalnej, liniowej, tanh i ReLU. Na górnym wykresie widoczna jest wyraźna niestabilność w przypadku funkcji liniowej, gdzie występują skoki MSE, co może sugerować nadmierne dopasowanie do danych treningowych lub wpływ anomalii. Pozostałe funkcje aktywacji prezentują znacznie stabilniejszy przebieg błędu w procesie uczenia. Na dolnym wykresie obserwujemy, że wszystkie trzy nieliniowe funkcje aktywacji – sigmoid, tanh i ReLU – wykazują podobną i szybką zbieżność do niskiego błędu MSE, z ReLU osiągając nieco lepsze rezultaty w początkowych epokach. Najlepszy uzyskany wynik MSE dla tych eksperymentów to 2.2039491592926326, co wskazuje na efektywność zastosowanych metod w procesie uczenia sieci neuronowej.

### 6.2.2 2 warstwy ukryte



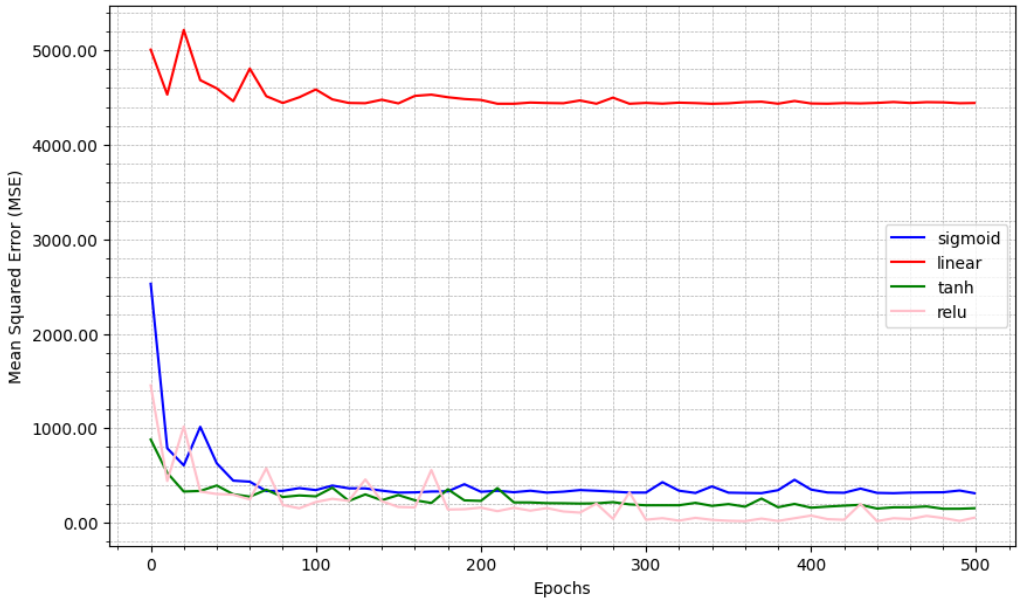

Rysunek 31: Porównanie szybkości uczenia w zależności od funkji aktywacji



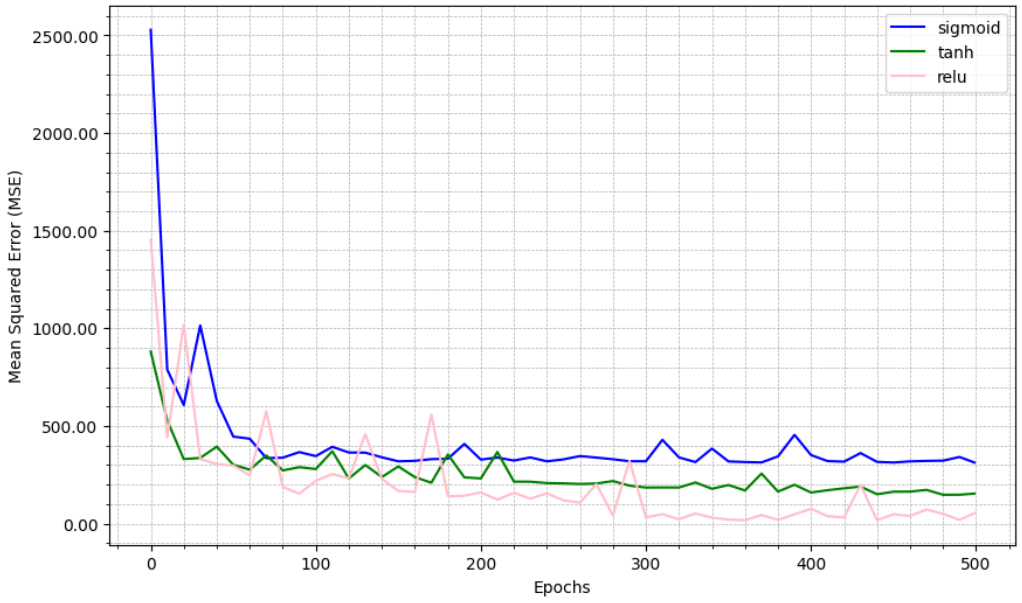

Rysunek 32: Porównanie szybkości uczenia w zależności od funkji aktywacji dla trzech najlepszych wariantów

Analizując wykresy dla sieci o architekturze [1, 64, 32, 1] z dwiema warstwami ukrytymi, zaobserwowano, że wszystkie funkcje aktywacji osiągają zbieżność w procesie uczenia, z najniższym błędem MSE równym 3.334245858731005. Funkcja liniowa wykazuje dużą niestabilność, co może być powiązane z jej ograniczeniami w nieliniowej separacji danych. Nieliniowe funkcje aktywacji – sigmoid, tanh, i ReLU – prezentują znacznie bardziej stabilne zachowanie, z tendencją do szybszego zmniejszania błędu MSE, co sugeruje ich przydatność w modelowaniu złożonych zależności w danych. Funkcja ReLU, podobnie jak na wcześniejszym wykresie, wydaje się oferować największą szybkość uczenia w początkowych epokach.

### 6.2.3 1 warstwa ukryta




Rysunek 33: Porównanie szybkości uczenia w zależności od funkji aktywacji



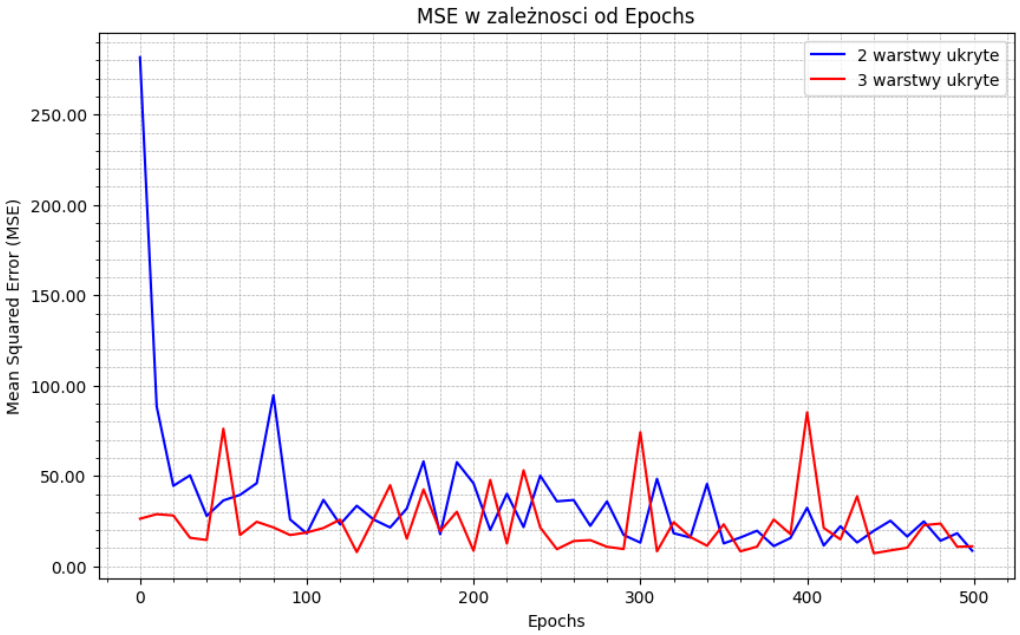

Rysunek 34: Porównanie szybkości uczenia w zależności od funkji aktywacji dla trzech najlepszych wariantów

Na przedstawionych wykresach dla sieci o architekturze [1, 64, 1] z jedną warstwą ukrytą obserwujemy wyraźny wpływ wyboru funkcji aktywacji na szybkość zmniejszania błędu średniokwadratowego (MSE). Wykresy ukazują, że mimo prób z różnymi funkcjami aktywacji, sieć osiąga najwyższy błąd MSE spośród trzech porównywanych konfiguracji architektury sieci, osiągając wartość 19.54497591094832. ReLU, podobnie jak w poprzednich konfiguracjach, początkowo szybko zniża wartość MSE, jednakże stabilizuje się na wyższym poziomie niż w przypadkach z większą liczbą warstw ukrytych. Wynika z tego, że większa głębokość sieci może przyczyniać się do lepszego modelowania złożoności danych.

Optymalne rezultaty w kontekście minimalizacji błędu średniokwadratowego (MSE) zostały zaobserwowane przy wykorzystaniu funkcji aktywacji sigmoidalnej, zarówno dla konfiguracji sieci z dwoma warstwami ukrytymi, jak i dla sieci z trzema warstwami ukrytymi. Wyniki te podkreślają efektywność funkcji aktywacji sigmoidalnej w ekstrakcji i modelowaniu cech w strukturach danych o większym stopniu skomplikowania.

## 6.3 Zbiór steps large

W następnym etapie eksperymentów zdecydowano się poddać weryfikacji dwie najbardziej obiecujące konfiguracje sieci neuronowej – [1, 64, 32, 1] oraz [1, 64, 32, 32, 1] – aplikując je do zbioru danych steps-large. Celem tego testu jest zbadanie skuteczności tych specyficznych architektur sieci w kontekście bardziej złożonego zadania regresji, co może dostarczyć cennych wskazówek odnośnie ich zdolności do generalizacji w różnorodnych scenariuszach uczenia maszynowego.




Rysunek 35: Porównanie dwóch najlepszych wariantów z poprzedniego zbioru dla steps-large

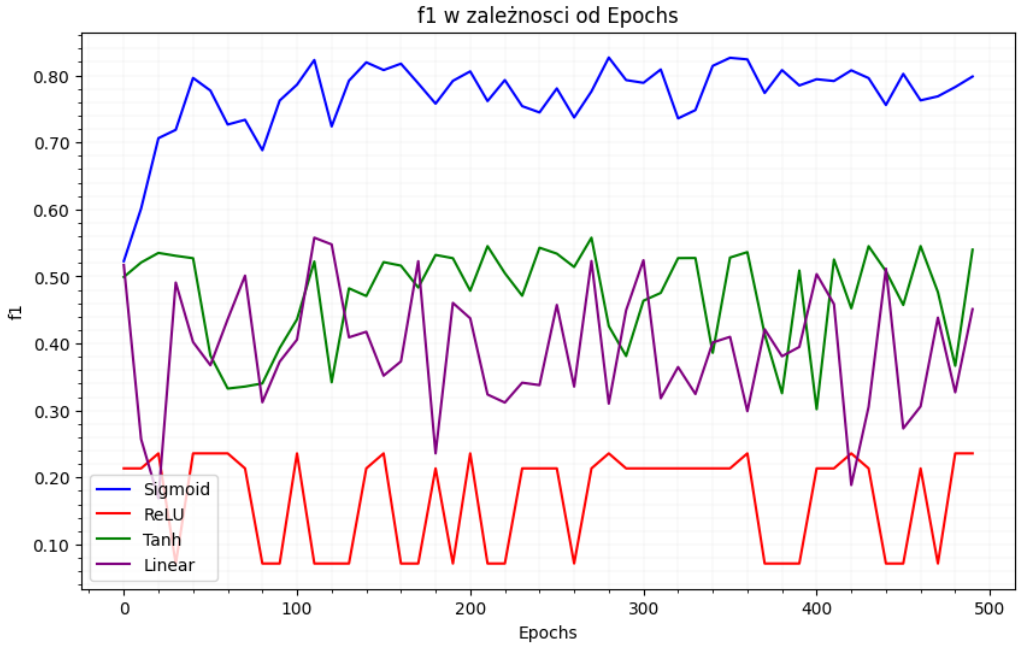
Najlepsze osiągnięte MSE to 8.635490801989514 dla dwóch warstw ukrytych oraz 8.628056960717736 dla trzech warstw ukrytych.

Następnie przeprowadzono analogiczną analizę, aby zidentyfikować, która funkcja aktywacji osiąga najwyższą efektywność w kontekście zadania klasyfikacji. Ta kolejna faza badań pozwoliła na ocenę i porównanie wpływu różnych funkcji aktywacji na wydajność sieci neuronowych w segregowaniu i klasyfikowaniu danych do odpowiednich kategorii.

## 6.4 Analiza Zestawów Danych ring3-regular i rings5-regular

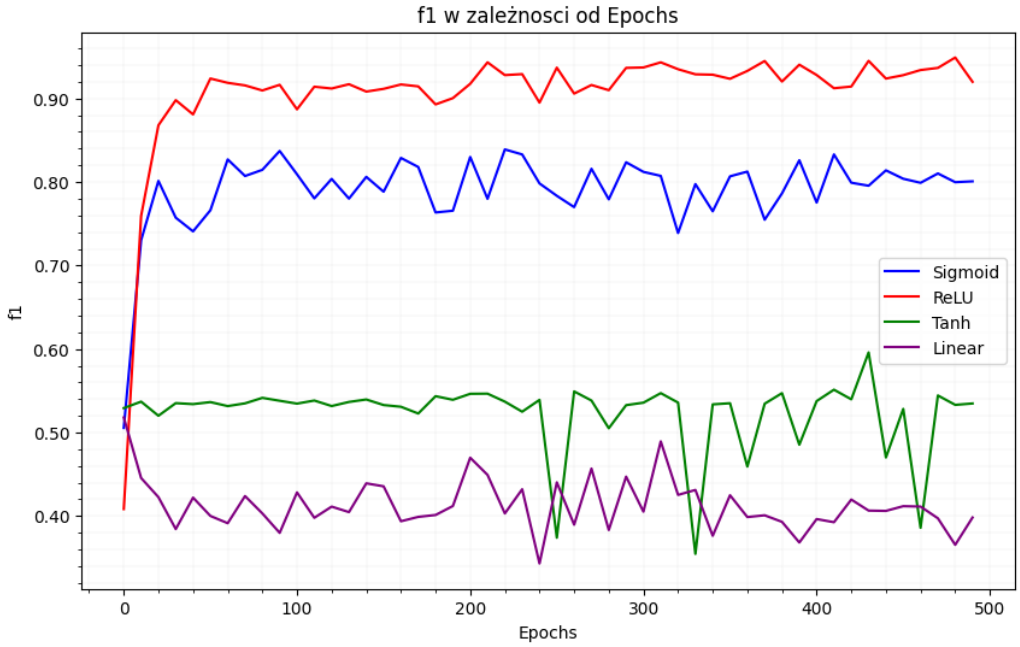
### 6.4.1 Zbiór ring3-regular - Badanie Modelu z Architekturą [2, 10, 10, 3]

W ramach przeprowadzonych eksperymentów na zestawie danych ring3-regular, analizowano model z architekturą [2, 10, 10, 3]. Przyjęto współczynnik uczenia na poziomie 0.01 oraz rozmiar batcha wynoszący 64. Celem badania było określenie wpływu konfiguracji na proces uczenia się modelu z wykorzystaniem różnych funkcji aktywacji.




Rysunek 36: Efektywność różnych funkcji aktywacji dla modelu na zestawie danych ring3-regular przy współczynniku uczenia 0.01 i rozmiarze batcha 64.

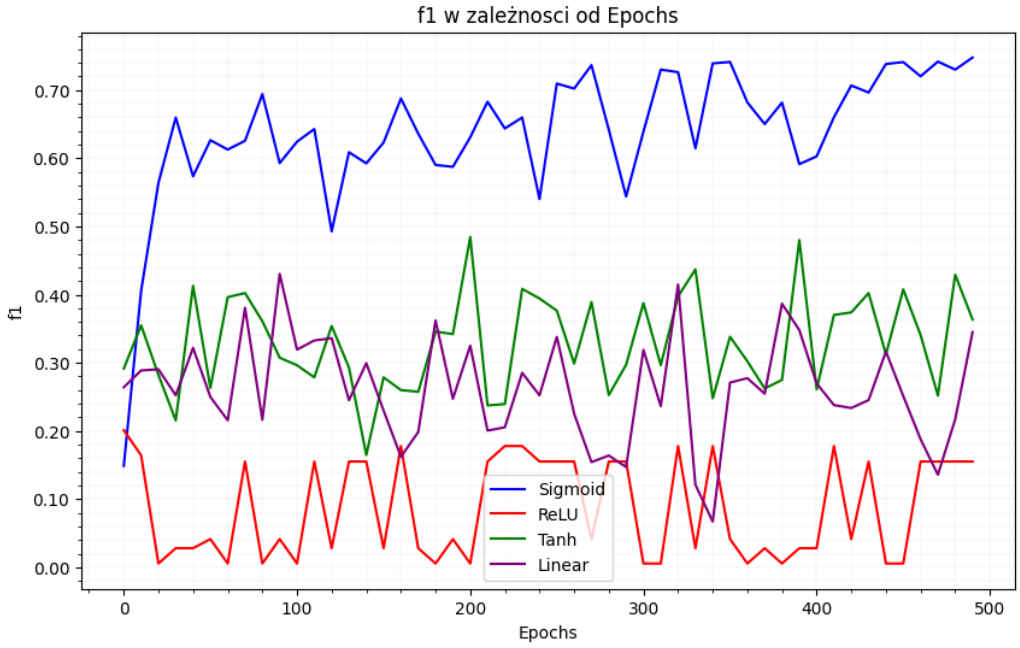
Następnie dokonano redukcji współczynnika uczenia do 0.001 oraz zmniejszono rozmiar batcha do 20, co pozwoliło zaobserwować wyraźną poprawę efektywności funkcji aktywacji ReLU w porównaniu do pozostałych funkcji.



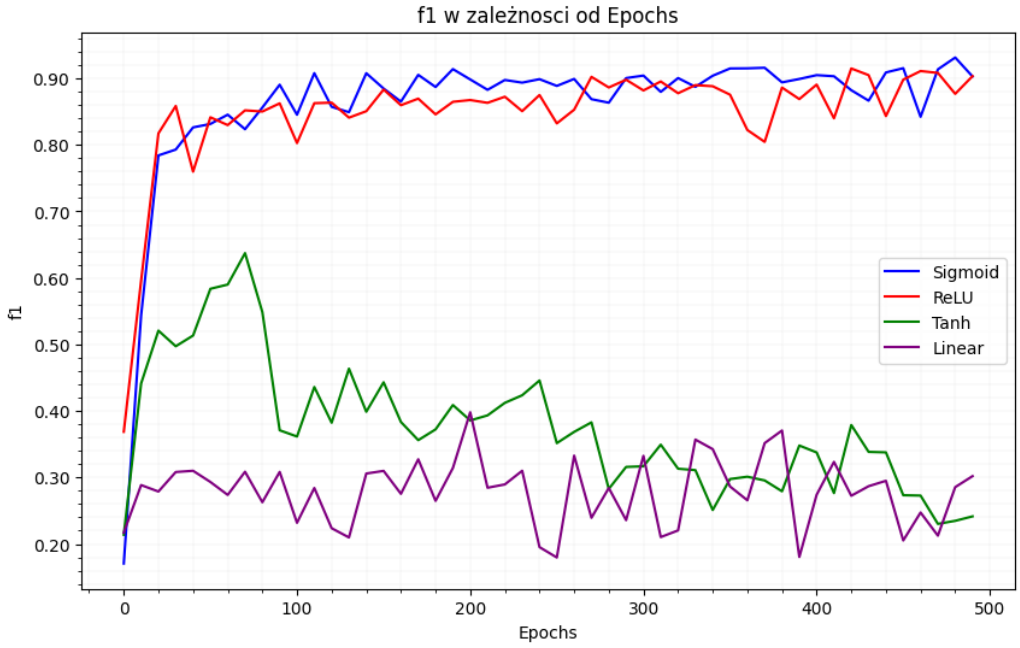

Rysunek 37: Efektywność różnych funkcji aktywacji dla modelu na zestawie danych ring3-regular przy zmniejszonym współczynniku uczenia 0.001 i rozmiarze batcha 20.

### 6.4.2 Zbiór ring5-regular - Badanie Modelu z Architekturą [2, 10, 10, 5]

Podobną metodologię badawczą zastosowano dla zestawu danych rings5-regular, testując model z architekturą [2, 10, 10, 5]. W tym przypadku również zaobserwowano, że funkcja aktywacji ReLU osiąga lepsze rezultaty przy niższych współczynnikach uczenia.




Rysunek 38: Porównanie wydajności funkcji aktywacji dla modelu na zestawie danych rings5-regular przy współczynniku uczenia 0.01 i rozmiarze batcha 64.




Rysunek 39: Porównanie wydajności funkcji aktywacji dla modelu na zestawie danych rings5-regular przy zmniejszonym współczynniku uczenia 0.001 i rozmiarze batcha 20.

Wnioski z eksperymentów wskazują na to, że dla modeli z architekturami [2, 10, 10, 3] i [2, 10, 10, 5] funkcja aktywacji ReLU, zwłaszcza przy niższym współczynniku uczenia, wykazuje zwiększoną skuteczność w zadaniu klasyfikacji.

# 7 NN6: Zjawisko Przeuczenia i Regularyzacja

W trakcie badań nad prostymi zbiorami danych, takimi jak multimodal-sparse, rings5-sparse, rings3-balance i xor3-balance nie udało się zaobserwować znaczącego zjawiska przeuczenia (overfittingu). Przyczyną tego może być niska złożoność samych zbiorów, które nie wymagają zastosowania skomplikowanych modeli do osiągnięcia wysokiej skuteczności klasyfikacji. Overfitting jest zjawiskiem częstszym w przypadku, gdy modele są zbyt złożone w stosunku do prostoty danych, co prowadzi do nauczenia się przez model nie tylko ogólnych trendów, ale i szumu oraz wyjątków charakterystycznych tylko dla danych treningowych, co negatywnie wpływa na zdolność generalizacji modelu.

Możliwą przyczyną braku overfittingu w przeprowadzonych eksperymentach jest także odpowiednia regularyzacja modelu, zastosowanie technik, takich jak wczesne zatrzymywanie procesu uczenia (early stopping) lub użycie technik augmentacji danych, które nie były potrzebne lub użyte w tym przypadku. Ponadto, w eksperymentach użyto ograniczonej liczby epok uczenia, co także mogło zapobiec przeuczeniu modelu.

# 8 Wnioski Końcowe

Z przeprowadzonych eksperymentów wynika, że odpowiednia konfiguracja sieci neuronowej, w tym wybór architektury oraz hiperparametrów, jest kluczowa dla efektywności modelu w różnych zadaniach predykcyjnych. Funkcje aktywacji takie jak sigmoid i ReLU wykazały się szczególną efektywnością w kontekście klasyfikacji, co podkreśla znaczenie ich dobrego dopasowania do specyfiki danych. Warto zauważyć, że najniższe wartości MSE zostały osiągnięte przy użyciu funkcji aktywacji sigmoidalnej dla architektur sieci z większą liczbą warstw ukrytych. Ponadto, aktualizacje oparte na batchach okazały się bardziej efektywne niż aktualizacje po każdej epoce, co wskazuje na ich preferencyjne zastosowanie w praktycznych implementacjach.

Nie zaobserwowano zjawiska przeuczenia w prostych zestawach danych, co może wynikać z niewystarczającej złożoności danych lub ograniczonego zakresu przeprowadzonych eksperymentów. Przyszłe badania powinny uwzględniać bardziej złożone zbiory danych, aby lepiej zrozumieć dynamikę i ryzyko przeuczenia w bardziej skomplikowanych scenariuszach.

Uzyskane wyniki podkreślają wagę starannej kalibracji hiperparametrów oraz architektury sieci, zwracając uwagę na ich wpływ na zdolność modelu do generalizacji i unikania przeuczenia. Eksperymenty te dostarczają wartościowych wskazówek dla przyszłych badań nad sieciami neuronowymi, ich optymalizacji i zastosowania w różnorodnych dziedzinach uczenia maszynowego.