WSI lab5 sprawozdanie

Małgorzata Grzanka, Weronika Jamróz

May 2024

Spis treści

1 Wstęp

Celem zadania było zaimplementowanie perceptronu wielowarstwowego oraz wybranego algorytmu optymalizacji gradientowej z algorytmem propagacji wstecznej. My zdecydowałyśmy się na algorytm stochastycznego najszybszego spadku (Stochastic Gradient Descend).

Do analizy działania stworzonego modelu wykorzystałyśmy zbiór danych The Digit Dataset → Link do datasetu ←, który jest zbiorem 1797 obrazków przedstawiających cyfry 0-9 napisane ręcznie. Obrazki są w nim przedstawione jako listy 64 pikselów, gdzie jeden piksel jest reprezentowany przez liczbę naturalną z zakresu (0., 16.).

2 Zasada działania perceptronu wielowarstwowego

2.1 Ogólna zasada działania

Perceptron wielowarstwowy składa się z wielu warstw neuronów. Każdy neuron zwraca określoną wartość, która jest wyliczana na podstawie wyjść neuronów w poprzedniej warstwie oraz dobieranych w procesie uczenia wag i biasów. Wyście neuronu nazywane jest aktywacją. Wektor aktywacji na warstwie L wyliczany jest według wzoru:

$$a^L = \sigma(w^L \cdot a^{L-1} + b^L)$$

gdzie σ to funkcja aktywacji warstwy neuronów, w^L to macierz wag dla warstwy L, b^L to wektor biasów dla warstwy L, a a^L i a^{L-1} to odpowiednio wektor wyjść (aktywacji) neuronów w wartswie poprzedniej (L-1) i aktualnej (L). Zatem, mając określone wejście, sieć wylicza odpowiadające temu wejściu aktywacje na kolejnych warstwach za pomocą wag i biasów. Aktywacje na ostatniej warstwie to wynik działania sieci.

2.2 Wyliczanie wartości wag i biasów

Trenowanie sieci polega na znalezieniu odpowiednich wartości wag i biasów, na podstawie których obliczane są aktywacje neronów. Znajdywanie te odbywa się na podstawie algorytmu stochastycznego najszybszego spadku (Stochastic Gradient Descend). Zbiór danych dzielony jest na podzbiory (mini batches) o określonym rozmiarze i dla każdego z nich wyliczany jest gradient wag i gradient biasów. Gradient dla pojedynczego mini batcha to średnia gradientów każdej próbki z tego mini batcha. Aktualizacja macierzy wag i wektora biasów polega na odjęciu od aktualnej wartości gradientu dla mini batcha przemnożonego przez współczynnik learning rate.

Gradient wag i biasów dla pojedynczej próbki wyliczany jest na podstawie wzoru:

$$\frac{dC}{dw^L} = \frac{dC}{da^L} \cdot \sigma'(z^L) \cdot a^{L-1}$$

$$\frac{dC}{dB^L} = \frac{dC}{da^L} \cdot \sigma'(z^L)$$

Do obliczenia $\frac{dC}{da^L}$ używa się backpropagation - wykorzystuje się $\frac{dC}{da^{L+1}}$. Błąd dla ostatniej warstwy obliczany jest za pomocą danego wzoru na funkcję straty.

$$\frac{dC}{da^L} = w^{L+1} \cdot \frac{dC}{da^{L+1}}$$

3 Hiperparametry

Nasza siec neuronowa przyjmuje następujące hiperparaetry:

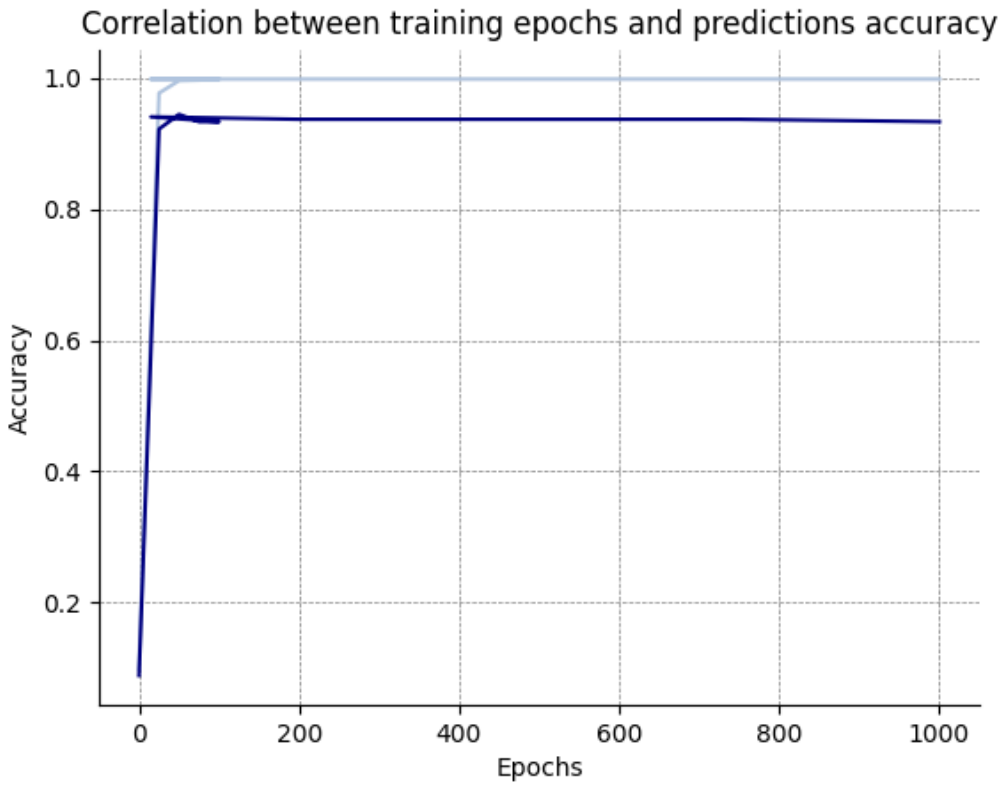
- struktura warstw - lista reprezentująca ile neuronów ma znajdować się w kolejnych warstwach

- funckje aktywacji - lista nazw funckji aktywacji, które będą stosowane na kolejnych warstwach w sieci
- funkcja straty - funkcja służąca do wyliczania funkcji starty w stochastic gradient desced
- learning rate - współczynnik wykorzystywany w Stochastic Gradient Descend, przez który mnożony jest gradient wag i biasów.
- epochs - liczba iteracji podczas trenowania sieci (ile razy ma one aktualizować wagi i biasy przechodząc przez cały dataset)
- rozmiar mini-batchy - ile elementów jest w jednym mini-batchu
- scale_data_flag - flaga czy program ma skalowac dane

Zbadałyśmy wpływ niektórych hiperparametrów nadziałanie sieci. Experymenty zostały przeprowadzone dla hiperparametrów *learning_rate*, *mini_batch_size* oraz *epochs*. Sieć testowana była dla powłok [64, 16, 16, 10] oraz funkcji aktywacji ['sigmoid', 'sigmoid', 'sigmoid'].

4 Wpływ epok na accurancy

4.1 Wyniki



Rys. 1: Dobranie najlepszej wartości parametru epochs

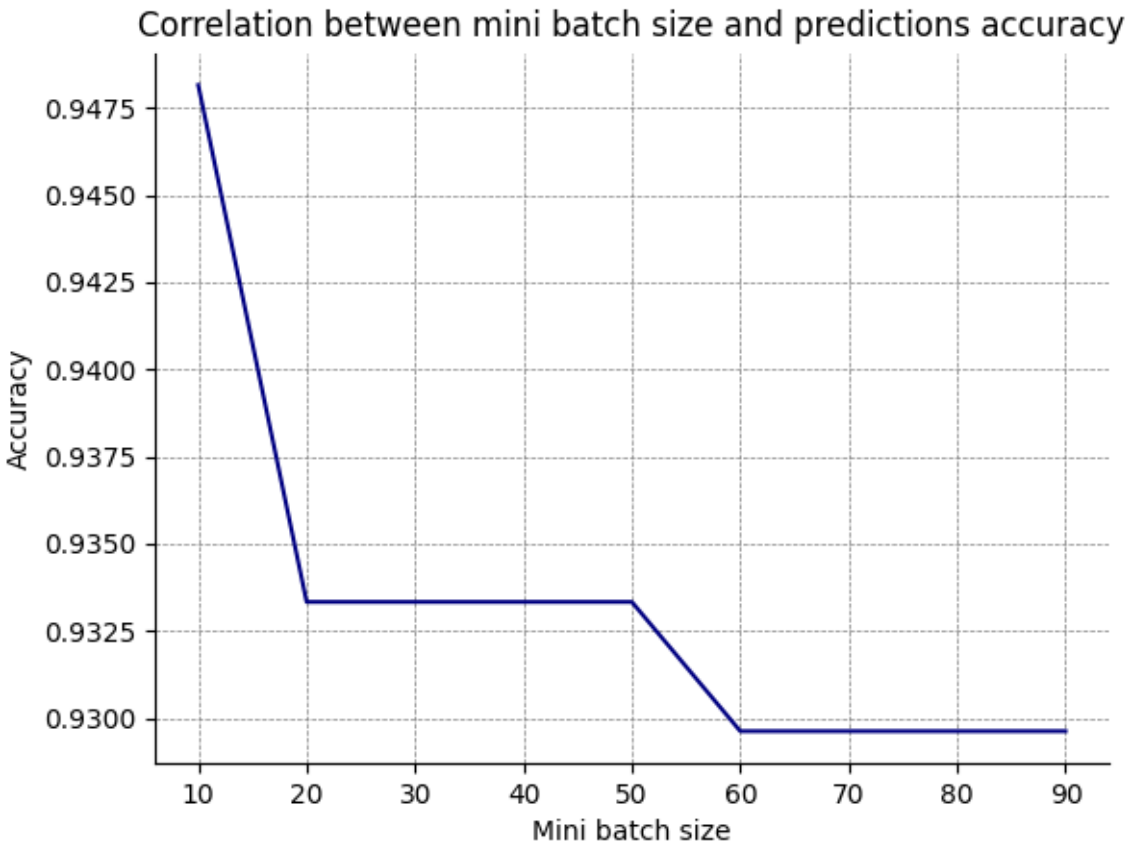
4.2 Wnioski

Można zauważyć że sieć uczy się przy stosunkowo małej liczbie powtórzeń, i już przy około 50 osiąga satysfakcjonujące wyniki. Można również zauważyć że po około 50 epokach algorytm nie uczy się już bardziej i jest to spowodowane maksymalnym dopasowaniem parametrów sieci - wag

i obciążeń - do zbioru treningowego. Nie jest to jednak przeuczenie, ponieważ wyniki dla zbioru validacyjnego nie maleją.

5 Wpływ rozmiaru mini-batchy na accurancy

5.1 Wyniki



Rys. 2: Dobranie najlepszej wartości parametru mini batch size

5.2 Wnioski

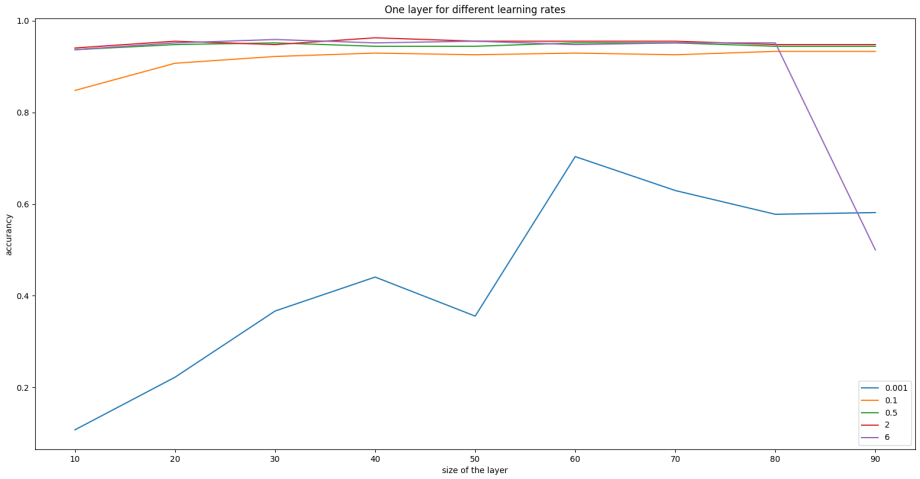
z wykresu wynika, że dla mniejszych rozmiarów mini-batchy sieć uczy się lepiej i moż być to specyfikă tego zbioru danych. optymalna wielkość mini-batchy to w tym przypadku 10 - 20

6 Struktura i learning rate do accurancy

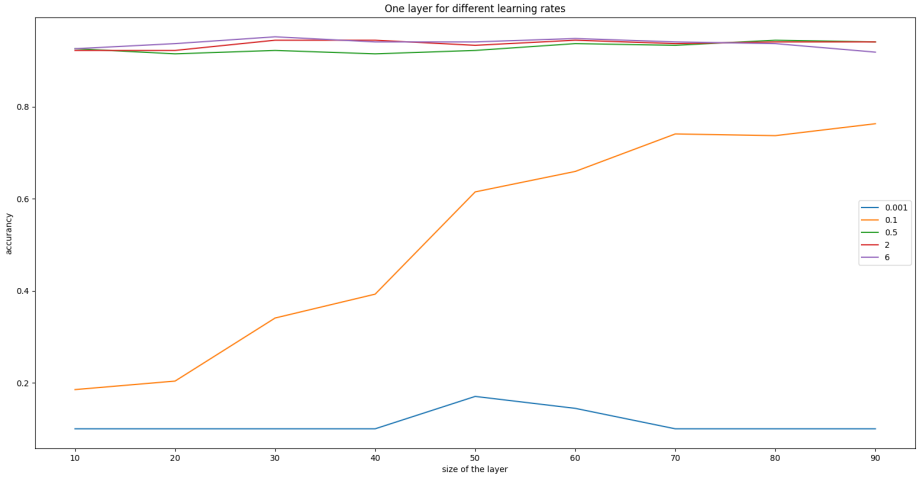
W tym doświadczeniu przetestowano wpływ wielkości warstw w sieci na wynik jej działania dla różnych learning rate dla różnych funkcji aktywacji.

6.1 Sigmoida

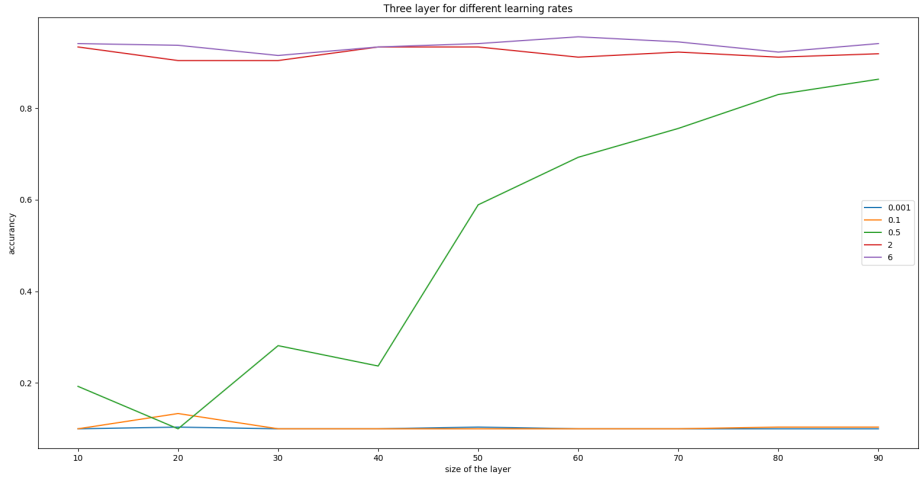
6.1.1 Wyniki



Rys. 3: Sigmoida, 1 warstwa



Rys. 4: Sigmoida, 2 warstwy



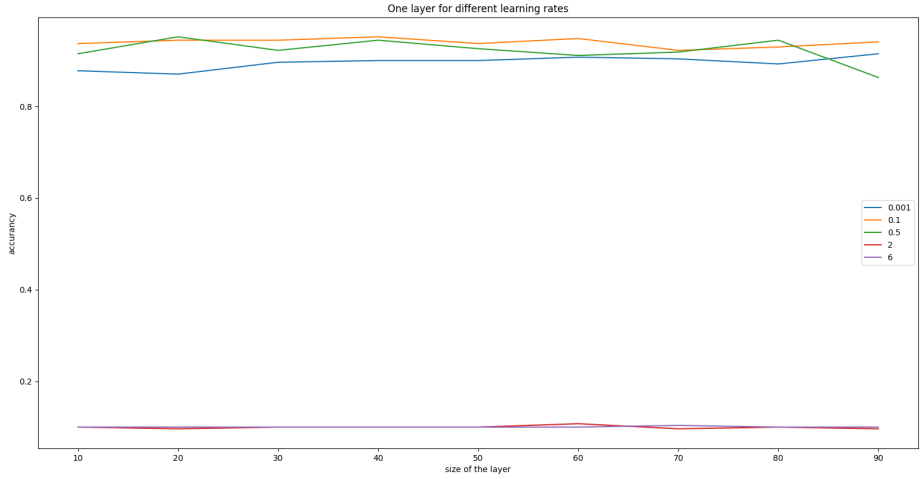
Rys. 5: Sigmoida, 3 warstwy

6.1.2 Wnioski

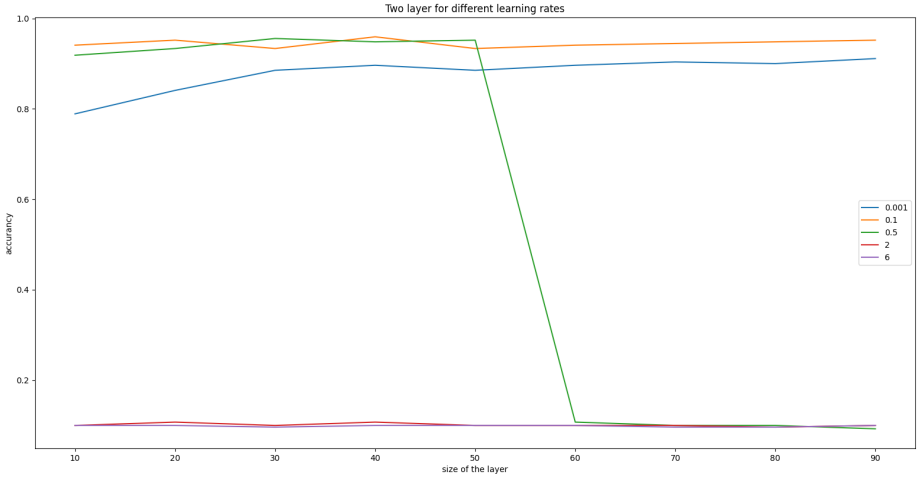
Dla sigmoidy, lepiej sprawdzają się większe learning rate - dla learning rate 2 oraz 6 sieć osiągała najlepszą precyzję. Learning rate o niskich wartościach były bardzo nieskuteczne w predykcjach dla każdej ilości warstw. Im więcej warstw sieci, tym learning rate powinno być większe. Jednak dla mniejszej ilości warstw (1 lub 2), zbyt duże learning rate okazuje się nieskuteczne przy wzroście liczby neuronów w warstwach. Learning rate 2 okazał się lepszy dla mniejszej ilości warstw, natomiast 6 - dla większej ilości warstw.

6.2 Tangens hiperboliczny

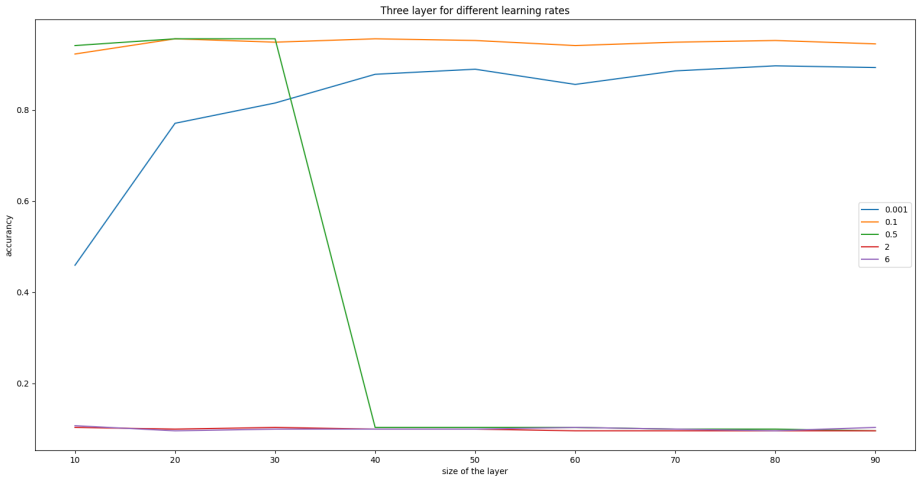
6.2.1 Wyniki



Rys. 6: Tangens hiperboliczny, 1 warstwa



Rys. 7: Tangens hiperboliczny, 2 warstwy



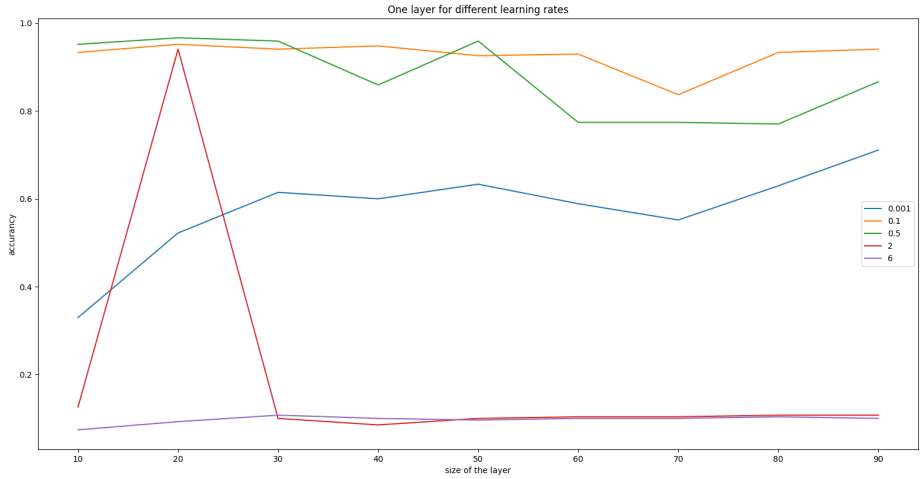
Rys. 8: Tangens hiperboliczny, 3 warstwy

6.2.2 Wnoski

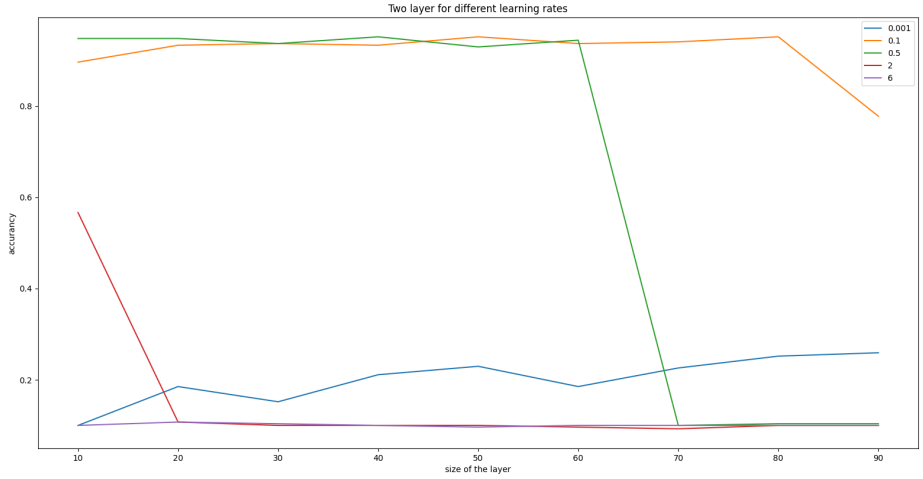
Dla tangensu hiperboliczengo, lepiej sprawdzją się mniejsze learning rate - najlepsze wyniki sieć osiąga dla learning rate równego 0.1 (stabilny wynik bez znaczenia, ile warstw i neuronów ma sieć). Zbyt mała wartość (0.001) prowadzi do nieco gorszych wyników, zwłaszcza przy mniejszej liczbie neuronów w większej ilości warstw. Duże learning rate (2 oraz 6) dają bardzo niskie wyniki dla każdej ilości warstw i neuronów w nich. Natomiast skuteczność przewidywań dla learning rate 0.5 gwałtownie spada dla większej ilości neuronów w warstwie, a granica tego spadku zmniejsza się wraz ze wzrostem liczby warstw.

6.3 Softplus

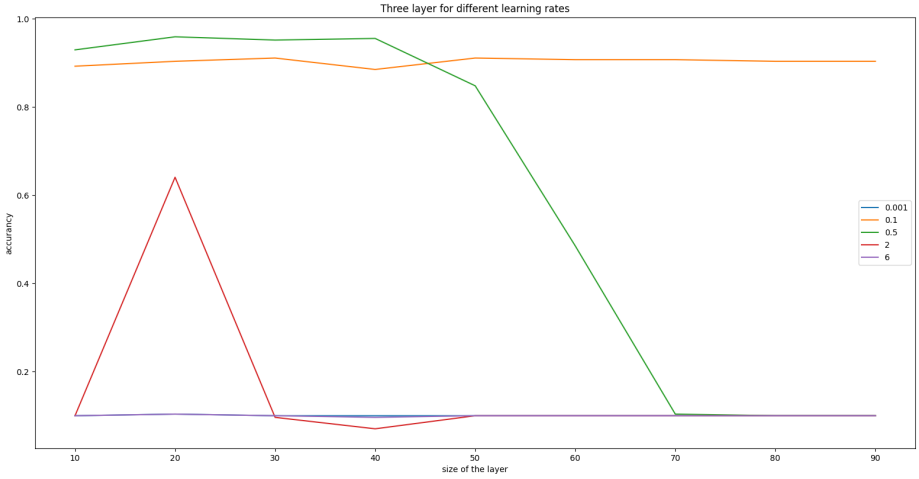
6.3.1 Wyniki



Rys. 9: Softplus, 1 warstwa



Rys. 10: Softplus, 2 warstwy



Rys. 11: Softplus, 3 warstwy

6.3.2 Wnioski

Dla softplus, najlepiej sprawdza się learning rate 0.1 - uzyskiwany jest dla niego stały, dobry wynik dla różnej ilości warstw i neuronów w nich. Z kolei bardzo duże learning rate (6) wypada bardzo słabo dla wszystkich kombinacji ilości warstw i neuronów w nich. Bardzo małe learning rate (0.001) oraz te nieco większe niż 0.1 (0.5) tracą na skuteczności wraz ze zwiększeniem się liczby warstw (0.5 dla większej liczby neurnów w warstwach, a 0.001 dla mniejszej).

6.4 Podsumowanie

Podsumowując, dobór learning rate zależy od funkcji aktywacji oraz ilości wastw i neuronów w tych warstwach. Dla simogmoidy sprawdzają się lepiej większe learning rate (bardzo duze - około 10 - dla duzej ilosci warstw, nieco mniejsze - około 2 - dla mnieszej ilosci warstw), a dla tangensa hiperbolicznego - mniejsze (około 0.1).

7 Najlepszy wynik

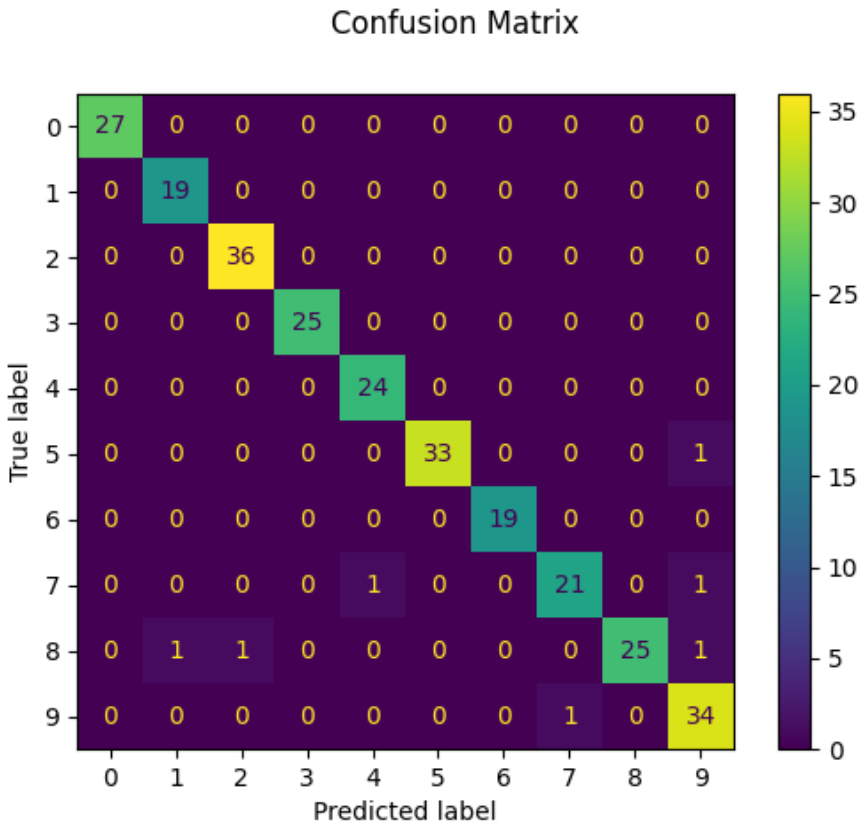
Dla hiperparametrów:

- $layers\_sizes$ = [64, 16, 16, 10]
- $activation\_functions$ = ['sigmoid', 'sigmoid', 'sigmoid']
- $loss\_function$ = 'squared_error'
- $learning\_rate$ = 0.55
- $epochs$ = 100
- $mini\_batch\_size$ = 10
- $scale\_data$ = False

udało się osiągnąć wynik dla danych testowych

$$accuracy = 0.9740740740740741$$

co przedstawione zostało na macierzy błędu 12



Rys. 12: Macierz błędu dla najlepszego uzyskanego wyniku

8 Wnioski

Udało się zaimplementować wielowarstwową sieć neuronową, która w 97% "odgaduje"cyfry na dostarczonych ilustracjach. Dokładność sieci zależy od zbiorów testowego, validacyjnego oraz treningowego. Ponieważ przy dzieleniu dane mieszane są w pseudolosowy sposób, wyniki uzyskiwane przy trenowaniu modelu przy tych samych hiperparametrach różnią się o około 1 - 3%. Zauważono również, że dobieranie hiperparametrów takich jak learning rate bardzo różni się dla różnych funkcji aktywacji dlatego dobrane hiperparametry sprawdzają się dla konkretnego ustawienia warstw neutronów oraz funkcji aktywacji.