Sprawozdanie

Przedmiot	WSI	**Grupa**	103	**Prowadzący**	mgr. inż. Kacper Kania
Numer ćwiczenia	5	**Temat ćwiczenia**		Wielowarstwowy perceptron	
Numery albumów	336369, 318407			**Skład sekcji**	Jakub Włodarz, Mikołaj Wewiór
Data zadania	23.04.2024			**Data wykonania**	21.05.2024

1. Treść zadania

Celem ćwiczenia było stworzenie modelu wielowarstwowego perceptronu, wykorzystującego przejście w przód, propagacje wsteczną oraz algorytm SGD do przeprowadzenia treningu na zbiorze *digits*. Perceptron powinien wspierać funkcje aktywacji tj. ReLU czy sigmoid. Do realizacji zadania wykorzystano język programowania Python w wersji 3.12 wraz z stosownymi bibliotekami (Numpy, Sklearn).

2. Opis programu

Stworzony program został podzielony na dwie klasy:

- *MLP*, zarządzający procesem uczenia na podanych warstwach,
- *Layer,* podstawową warstwę obliczeniową, która przetwarza podane paczki uczenia.

Uruchomienie instancji klasy *MLP* wymaga przekazania wyłącznie tablicy warstw, stanowiącej strukturę naszego modelu. Trening odbywa się poprzez wywołanie metody *fit()*, która pobiera dane wejściowe, etykiety oraz parametry bezpośrednio związane z procesem uczenia: liczbę epok i learning rate. Wielkość paczki uczenia ustalona zostaje poprzez warstwy.

Wizualizację analizy założonego zbioru danych wykonano w oddzielnym pliku *.ipynb,* umożliwiającym czytelną prezentację poszczególnych etapów jak i wyników pracy modelu - dane wejściowe przekazywane są w wersji znormalizowanej tablicy wartości zmiennoprzecinkowych, natomiast dla etykiet wykonywany jest *one-hot encoding*.

3. Wykonane eksperymenty

W ramach eksperymentów przeprowadzonych na stworzonym perceptronie założono trenowanie modelu przez 250 iteracji, zwanych potocznie epokami. Wartość *learning_rate* została ustalona jako 0.0005. W przypadku funkcji aktywacji na ostatniej warstwie umieszczono funkcję sigmoid, podczas gdy pozostałe warstwy wykorzystywały ReLU bądź sigmoid.

Podział danych został wykonany zgodnie z poleceniem - 70% należy do części do treningowej, a 30% do testowej. Ilość przykładów jest wystarczająco mała, aby pominąć podział danych na część walidacyjną. Badania przeprowadzono dla ziarna liczb pseudolosowych o wartości *‘20240521’*.

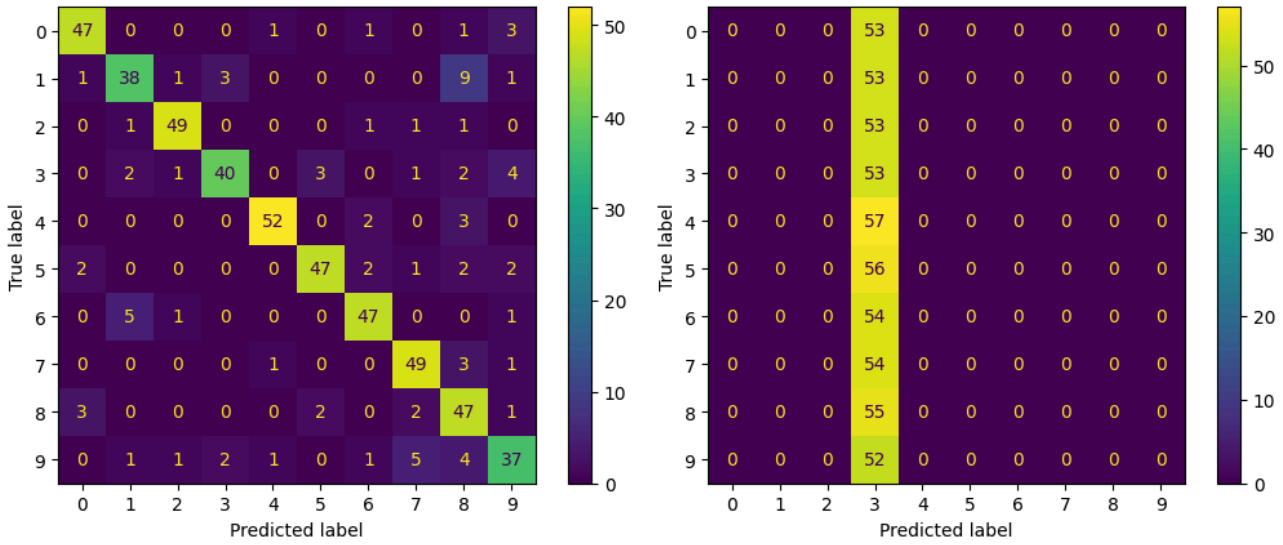
Tab. 1 – Wyniki analizy perceptronu wielowarstwowego dla zestawu danych ‘digits’

Liczba warstw ukrytych	Funkcja aktywacji	Rozmiar paczki uczenia	Accuracy	F1-score (micro)

1	ReLU	1	0.811	0.811
1	ReLU	4	0.104	0.104
1	ReLU	8	0.154	0.154
1	ReLU	16	0.066	0.066
1	ReLU	32	0.070	0.070
1	ReLU	64	0.104	0.104
1	ReLU	128	0.115	0.115
1	sigmoid	1	0.750	0.750
1	sigmoid	4	0.139	0.139
1	sigmoid	8	0.126	0.126
1	sigmoid	16	0.131	0.131
1	sigmoid	32	0.096	0.096
1	sigmoid	64	0.085	0.085
1	sigmoid	128	0.105	0.105
2	ReLU	1	0.813	0.813
2	ReLU	4	0.148	0.148
2	ReLU	8	0.083	0.083
2	ReLU	16	0.052	0.052
2	ReLU	32	0.076	0.076
2	ReLU	64	0.096	0.096
2	ReLU	128	0.098	0.098
2	sigmoid	1	0.415	0.415
2	sigmoid	4	0.057	0.057
2	sigmoid	8	0.137	0.137
2	sigmoid	16	0.12	0.12
2	sigmoid	32	0.104	0.104
2	sigmoid	64	0.098	0.098
2	sigmoid	128	0.098	0.098
3	ReLU	1	0.865	0.865

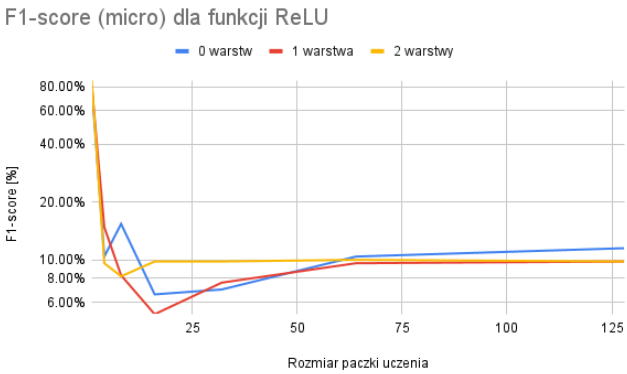
3	ReLU	4	0.096	0.096
3	ReLU	8	0.082	0.082
3	ReLU	16	0.098	0.098
3	ReLU	32	0.098	0.098
3	ReLU	64	0.1	0.1
3	ReLU	128	0.098	0.098
3	sigmoid	1	0.348	0.348
3	sigmoid	4	0.27	0.27
3	sigmoid	8	0.041	0.041
3	sigmoid	16	0.156	0.156
3	sigmoid	32	0.1	0.1
3	sigmoid	64	0.098	0.098
3	sigmoid	128	0.098	0.098

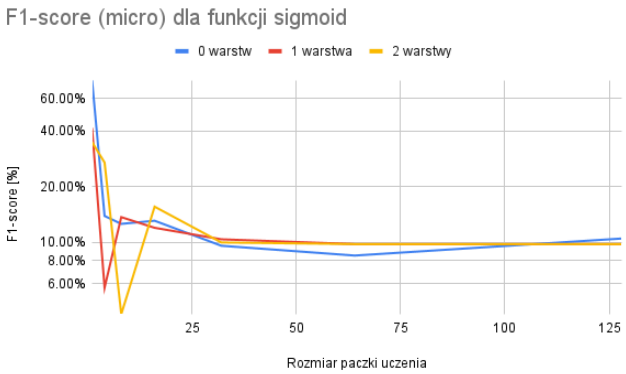
W warstwie wejściowej są 64 neurony - tyle ile pikseli bitmapy z narysowaną cyfrą. Ostatnia warstwa posiada 10 wyjść - czyli tyle ile klas do reprezentacji jest w danym zadaniu. Struktura w środku jest zmienna, zależnie od liczby warstw ukrytych. Dla przypadku z zerem warstw ukrytych warstwa wyjściowa posiada 16 wejść. W przypadku pojedynczej warstwy ukrytej posiada ona 32 wejścia i 16 wyjść. Gdy sieć składa się z dwóch warstw ukrytych mają one kolejno 32 wejścia i 24 wyjścia oraz 24 wejścia i 16 wyjść.



Rys. 1, 2 – przykładowe macierze pomyłek dla batch_size = 1 oraz 128 (2 warstwy ukryte, ReLU)

Dla przeprowadzonych pomiarów możliwe jest zauważanie pewnej ciekawej zależności - w przypadku realizacji procesu trenowania modelu z dużą paczką uczenia mamy do czynienia z zjawiskiem nadmiernego dopasowania modelu do pewnej cyfry, co z kolei przyczynia się do oscylacji wyniku końcowego w granicach 9-10%.





Rys. 3, 4 – wykresy wartości F1-score dla funkcji ReLU oraz sigmoid

4. Wnioski

Po przeprowadzonych badaniach można zauważyć, iż badane hiperparametry (*liczba warstw ukrytych*, *wielkość paczki*, *funkcja aktywacji)* nie wpływają bezpośrednio na otrzymane wyjścia. Najlepszą wartością parametru *batch_size* okazała się *batch_size* = 1. Może to wynikać z błędu w implementacji, jednak ustalenie tego faktu nie zostało możliwe.

Dla funkcji aktywacji ReLU uzyskano wyniki znacznie lepsze niż w przypadku funkcji sigmoid, co może oznaczać, iż jej stosowanie nie jest dobrym rozwiązaniem w ramach powyższego zestawu danych. W przypadku danych o liczbie warstw ukrytych zauważalna jest nieznaczna poprawa modeli końcowych dla większej ilości warstw. Niestety, taka rozbudowa modelu wiąże się z znacznym wydłużeniem czasu treningu.