Podstawy Sieci Neuronowych	
Kierunek *Informatyczne Systemy Automatyki*	**Termin** *środa TP* $13^{15} - 15^{55}$
Imię, nazwisko, numer albumu *Mikołaj Nowak 280082, Jakub Jasiński 280109*	**Data** *22.01.2026*
Link do projektu `https://github.com/RareIcubu/Podstawy-AI`	

Projekt: Biblioteka MLP

Spis treści

1 Wstęp

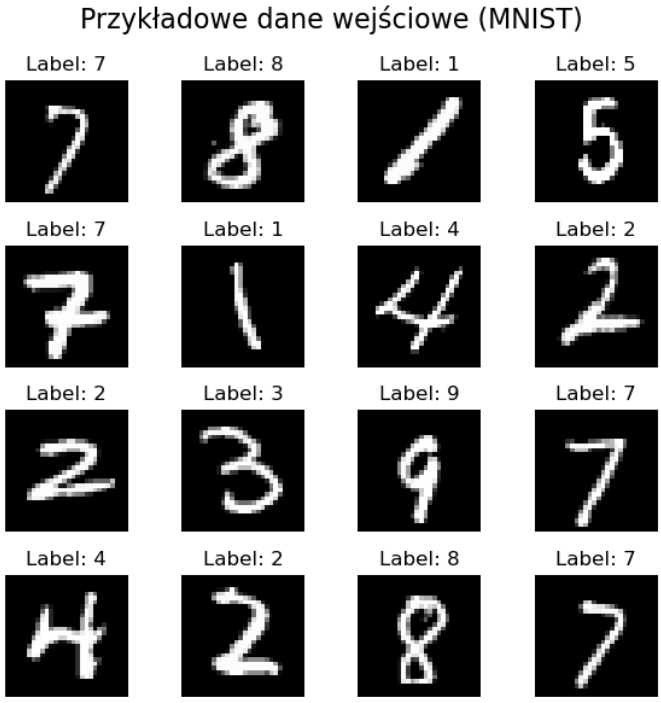
W ramach projektu wybraliśmy wariant nr 2. Naszym zadaniem było napisanie własnej biblioteki do sieci neuronowych MLP z algorytmem uczenia. Następnie należało zaimplementować i wytrenować sieć do rozwiązania zadania klasyfikacji obrazów MNIST oraz aproksymacji funkcji Ackleya.

2 Analiza Problemu

2.1 Analiza i Wizualizacja Danych

2.1.1 MNIST

Zbiór danych składa się z obrazów cyfr 0-9. Poniższa rycina przedstawia losową próbkę danych wejściowych po normalizacji (wartości [0, 1]).



Rysunek 1: Przykładowe obrazy ze zbioru uczącego MNIST

Do klasyfikacji obrazów użyta została baza MNIST z serwisu Kaggle (`https://www.kaggle.com/datasets/hojjatk/mnist-dataset`), zawierająca 60 tys. zdjęć treningowych oraz 10 tys. zdjęć testowych. Dane są spłaszczane do wektora 784 elementów. Etykiety kodowane są metodą one-hot encoding.

2.1.2 Funkcja Ackleya

Do aproksymacji użyto danych generowanych losowo (20 tys. próbek), aby zapewnić lepszą generalizację w dziedzinie $x, y \in [-2, 2]$.

2.2 Wejścia i Wyjścia

2.2.1 MNIST

Obrazy MNIST mają wymiary 28x28 pikseli. Są one "spłaszczane" do jednowymiarowej listy o długości 784 elementów. Na wyjściu sieci znajduje się 10 neuronów (klasyfikacja wieloklasowa).

2.2.2 Funkcja Ackleya

Wzór analityczny:

$$f(x,y) = -20\exp\left[-0.2\sqrt{0.5(x^2+y^2)}\right] - \exp\left[0.5(\cos(2\pi x)+\cos(2\pi y))\right] + e + 20 \tag{1}$$

Zastosowano **Feature Engineering**. Wektor wejściowy został rozszerzony o wartości cosinusów:

$$Input = [x, y, \cos(2\pi x), \cos(2\pi y)] \tag{2}$$

Podejście to linearyzuje problem okresowości funkcji w wyższym wymiarze, co znacząco przyspiesza zbieżność.

2.3 Dobór Funkcji Aktywacji

2.3.1 MNIST - ReLU

Wybrano **ReLU** w warstwach ukrytych ze względu na rzadkość aktywacji (sparsity) oraz eliminację problemu zanikającego gradientu. Na wyjściu zastosowano **Softmax**.

2.3.2 Funkcja Ackleya - Tanh

Wybrano tangens hiperboliczny (**Tanh**) ze względu na gładkość funkcji (C^∞) oraz centrowanie wartości w zerze, co lepiej pasuje do aproksymacji pofalowanej powierzchni niż ReLU. Na wyjściu zastosowano funkcję liniową.

3 Przygotowanie Architektury Problemu

3.1 Biblioteka MLP

Tworzenie sieci polega na dodawaniu kolejnych warstw (‘Dense‘) do modelu. Biblioteka obsługuje w pełni konfigurowalne warstwy, funkcje aktywacji oraz algorytm Stochastic Gradient Descent (SGD) z Backpropagation.

4 Wybór Architektury sieci

4.1 Liczba Warstw i Neuronów

4.1.1 Architektura dla MNIST

Przyjęto architekturę testową z jedną warstwą ukrytą [64] dla szybkiej weryfikacji biblioteki, oraz testowano głębsze struktury w szerszych eksperymentach.

4.1.2 Architektura dla Ackleya

Testowano sieci o różnej głębokości (od 2 do 3 warstw ukrytych) i szerokości (64, 128, 256 neuronów), aby zbadać wpływ pojemności modelu na dokładność aproksymacji.

5 Trenowanie Sieci i Opis Eksperymentów

5.1 Testy

Przeprowadzono Grid Search dla następujących parametrów:

5.1.1 MNIST

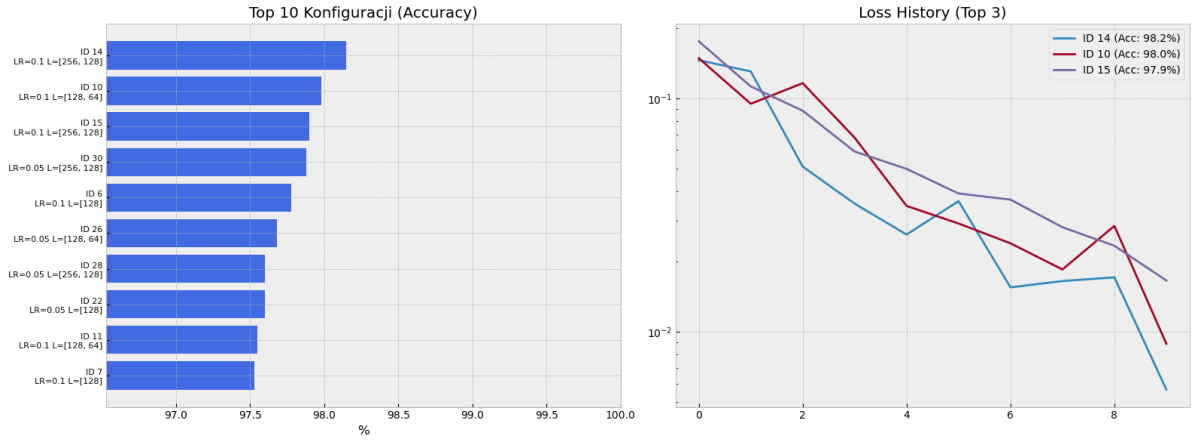
- Współczynnik uczenia (LR): 0.1, 0.05, 0.01
- Warstwy ukryte: [64], [128], [64, 64], [128, 64], [256, 128]
- Epoki: 5, 10
- Batch size: 32, 64

5.1.2 Funkcja Ackleya

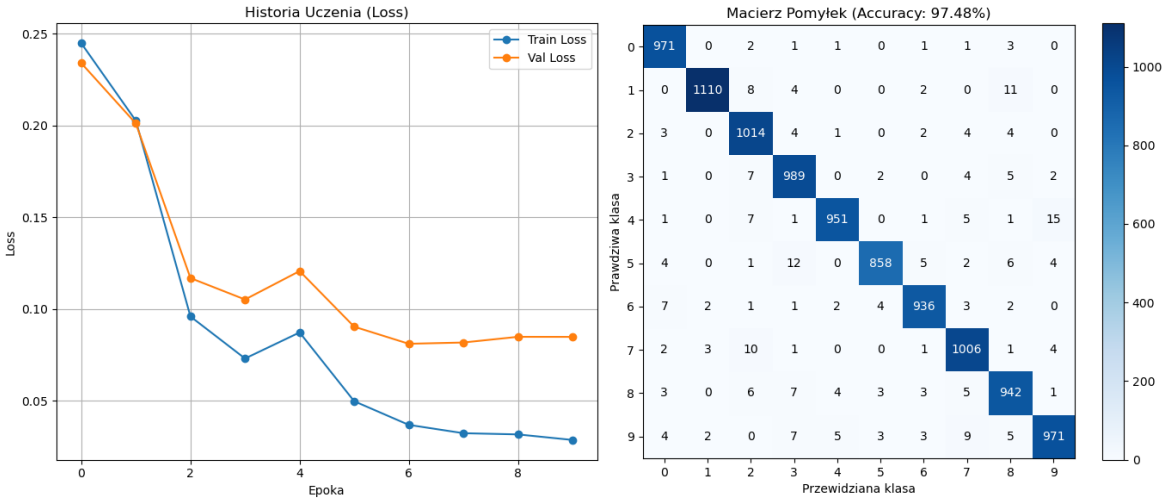
- Współczynnik uczenia (LR): 0.05, 0.01, 0.005
- Architektura: [64, 64], [128, 128], [128, 128, 128], [256, 256]
- Epoki: 500, 1000
- Batch size: 64, 128

6 Analiza rozwiązania i wyników

6.1 MNIST



Rysunek 2: Wyniki eksperymentów MNIST



Rysunek 3: Przykładowe wyniki dla pojedynczej sieci MNIST

Tabela 1: Najlepsze 10 wyników dla klasyfikacji MNIST

ID	LR	Warstwy	Epoki	Batch	Accuracy	Czas [s]
14	0.10	[256, 128]	10	32	**98.15%**	248.61
10	0.10	[128, 64]	10	32	97.98%	118.90
15	0.10	[256, 128]	10	64	97.90%	244.75
30	0.05	[256, 128]	10	32	97.88%	248.20
6	0.10	[128]	10	32	97.78%	110.05
26	0.05	[128, 64]	10	32	97.68%	118.49
28	0.05	[256, 128]	5	32	97.60%	124.93
22	0.05	[128]	10	32	97.60%	110.07
11	0.10	[128, 64]	10	64	97.55%	117.08
7	0.10	[128]	10	64	97.53%	108.13

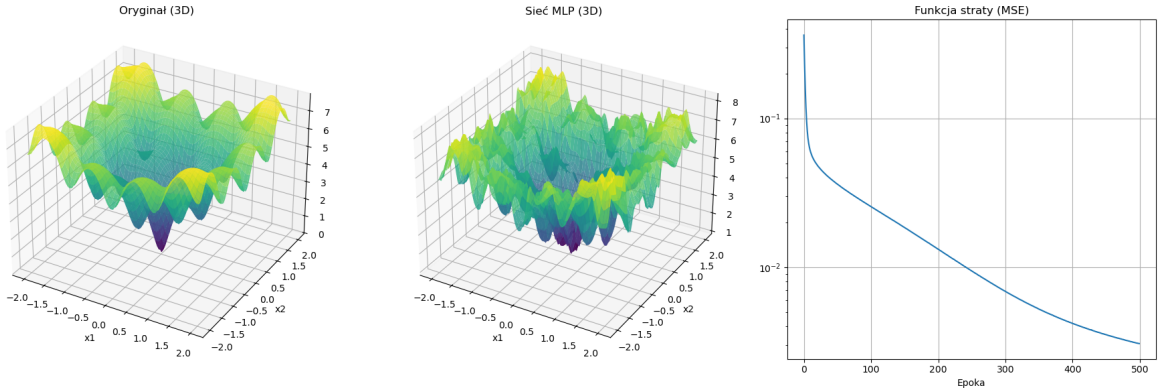
Tabela 2: Najgorsze 10 wyników dla klasyfikacji MNIST

ID	LR	Warstwy	Epoki	Batch	Accuracy	Czas [s]
33	0.01	[64]	5	64	91.86%	27.13
37	0.01	[128]	5	64	92.15%	54.30
32	0.01	[64]	5	32	93.13%	27.93
39	0.01	[128]	10	64	93.46%	108.51
35	0.01	[64]	10	64	93.53%	54.44
36	0.01	[128]	5	32	93.65%	55.38
41	0.01	[128, 64]	5	64	93.90%	58.34
45	0.01	[256, 128]	5	64	94.19%	121.84
34	0.01	[64]	10	32	94.80%	55.04
40	0.01	[128, 64]	5	32	95.00%	59.17

6.1.1 Analiza Wyników (MNIST)

- **Najlepszy wynik:** Model ID 14 osiągnął dokładność **98.15%**. Jest to głęboka sieć ([256, 128]), uczona przez 10 epok ze stosunkowo wysokim współczynnikiem uczenia ($LR = 0.1$).
- **Wpływ Learning Rate:** W przeciwieństwie do wstępnych testów, pełne eksperymenty pokazały, że wyższy Learning Rate (0.1 oraz 0.05) daje lepsze rezultaty niż niski (0.01), który dominował w tabeli najgorszych wyników (ok. 91-94%).
- **Wpływ Architektury:** Większe sieci ([256, 128] oraz [128, 64]) konsekwentnie zajmowały najwyższe miejsca w rankingu, co potwierdza, że większa pojemność modelu pomaga w ekstrakcji cech z obrazów MNIST.
- **Batch Size:** Mniejszy rozmiar batcha (32) występował częściej w czołówce (6 z 10 najlepszych modeli), co sugeruje, że częstsze aktualizacje wag pomagają w szybszej zbieżności.

6.2 Funkcja Ackleya



Rysunek 4: Wyniki eksperymentów funkcji Ackleya (MSE Loss i Przebieg Uczenia)

Tabela 3: Top 10 najlepszych konfiguracji dla funkcji Ackleya (posortowane wg MSE)

ID	LR	Warstwy	Epoki	Batch	Loss (MSE)	Czas [s]
10	0.050	[128, 128, 128]	1000	64	**0.000836**	945.11
42	0.005	[128, 128, 128]	1000	64	0.000836	940.23
14	0.050	[256, 256]	1000	64	0.000860	1664.86
6	0.050	[128, 128]	1000	64	0.001031	530.38
2	0.050	[64, 64]	1000	64	0.001090	188.24
12	0.050	[256, 256]	500	64	0.001163	835.15
8	0.050	[128, 128, 128]	500	64	0.001189	472.97
38	0.005	[128, 128, 128]	1000	128	0.001423	948.33
15	0.050	[256, 256]	1000	128	0.001643	1653.34
4	0.050	[128, 128]	500	64	0.001700	266.93

6.2.1 Analiza Wyników (Ackley)

- **Wpływ liczby epok:** Najlepsze wyniki (MSE < 0.001) osiągnięto niemal wyłącznie przy **1000 epokach**. Krótszy trening (500 epok) często nie wystarczał do pełnej konwergencji.
- **Złożoność sieci:** Sieć trójwarstwowa [128, 128, 128] okazała się optymalnym kompromisem między dokładnością (MSE 0.000836) a czasem obliczeń. Sieci płytsze [64, 64] radziły sobie gorzej.
- **Batch Size:** Mniejszy rozmiar batcha (64) dominował w czołówce rankingu. Wprowadza on więcej stochastyczności do gradientu, co pomaga uciec z licznych minimów lokalnych funkcji Ackleya.
- **Stabilność:** Zarówno wysoki (0.05), jak i niski (0.005) Learning Rate pozwolił na osiągnięcie świetnych wyników przy odpowiedniej liczbie epok.

7 Wnioski Końcowe

7.1 Ograniczenia MLP w przetwarzaniu obrazu

Choć sieć MLP jest w stanie klasyfikować cyfry, brak inwariancji przestrzennej (niszczenie struktury 2D przez spłaszczanie obrazu) ogranicza jej potencjał w porównaniu do sieci konwolucyjnych (CNN).

7.2 Podsumowanie projektu

- Zaimplementowana biblioteka poprawnie realizuje proces uczenia (Backpropagation).
- **Feature Engineering** (dodanie cosinusów) był kluczowy dla sukcesu aproksymacji funkcji Ackleya.
- **Czas obliczeń:** Pythonowa implementacja na CPU jest stosunkowo wolna (najdłuższy eksperyment trwał ponad 27 minut), co uzasadnia użycie frameworków GPU w większych projektach.
- Przeprowadzone testy (łącznie 48 konfiguracji dla Ackleya i 48 dla MNIST) pozwoliły wyznaczyć optymalne hiperparametry, osiągając błąd MSE rzędu $8 \cdot 10^{-4}$ oraz dokładność klasyfikacji $> 98\%$.