Projekt: Implementacja i analiza sieci neuronowych MLP

Porównanie implementacji ręcznej i frameworkowej

Jakub Sornat
Maciej Tajs
Bartłomiej Sadza

16 listopada 2025

Spis treści

1 Wstęp

1.1 Cel projektu

Celem projektu jest implementacja wielowarstwowego perceptronu (MLP, Multi-Layer Perceptron) w dwóch wariantach:

- **Implementacja ręczna** – napisana od podstaw w NumPy, z pełną kontrolą nad forward propagation, backward propagation i optymalizacją
- **Implementacja frameworkowa** – wykorzystująca bibliotekę TensorFlow/Keras

Projekt ma na celu:

1. Zrozumienie wewnętrznego działania sieci neuronowych poprzez implementację od podstaw
2. Porównanie wydajności i dokładności obu podejść
3. Praktyczne zastosowanie MLP do problemów klasyfikacji i regresji
4. Przeprowadzenie systematycznych eksperymentów z różnymi hiperparametrami

1.2 Zakres prac

W ramach projektu zrealizowano:

- Implementację ręczną sieci MLP z warstwami Dense, aktywacjami ReLU i Softmax
- Implementację modelu MLP w Keras/TensorFlow
- Preprocessing i przygotowanie 4 zbiorów danych (2 klasyfikacyjne, 2 regresyjne)
- Przeprowadzenie eksperymentów z grid search po hiperparametrach
- Analizę wyników i wizualizację (learning curves, confusion matrices, wykresy porównawcze)
- Dokumentację techniczną projektu

1.3 Użyte technologie

- **Python 3.x** – główny język programowania
- **NumPy** – operacje na macierzach i implementacja ręczna
- **Pandas** – przetwarzanie danych
- **TensorFlow/Keras** – implementacja frameworkowa
- **Matplotlib** – wizualizacja wyników
- **scikit-learn** – metryki ewaluacji i preprocessing
- **OpenPyXL** – eksport wyników do Excel

2 Zbiory danych

W projekcie wykorzystano cztery zbiory danych: dwa do klasyfikacji binarnej i dwa do regresji.

2.1 Classification: Adult Income Dataset

Źródło: UCI Machine Learning Repository

Opis: Zbiór danych zawierający informacje o dochodach osób dorosłych. Celem jest przewidzenie, czy osoba zarabia powyżej 50K$ rocznie.

Charakterystyka:

- Liczba rekordów: 32,561
- Liczba cech (po preprocessingu): 21
- Klasy: 2 (>50K, 50K)
- Typ problemu: Klasyfikacja binarna

Przykładowe cechy:

- Wiek (age)
- Wykształcenie (education)
- Stan cywilny (marital_status)
- Zawód (occupation)
- Godziny pracy tygodniowo (hours_per_week)

2.2 Classification (Our): Loan Approval Dataset

Źródło: [Podać źródło jeśli znane]

Opis: Zbiór danych dotyczący zatwierdzania kredytów bankowych.

Charakterystyka:

- Liczba rekordów: 50,000
- Liczba cech (po preprocessingu): 9
- Klasy: 2 (Approved, Rejected)
- Typ problemu: Klasyfikacja binarna

Przykładowe cechy:

- Dochód roczny (annual_income)
- Wynik kredytowy (credit_score)
- Liczba niespłaconych kredytów (defaults_on_file)

2.3 Regression: Stock Market Dataset

Źródło: [Podać źródło jeśli znane]
Opis: Dane z rynku akcji zawierające szeregi czasowe cen i wskaźników.
Charakterystyka:

- Liczba rekordów: 29,900
- Liczba cech: [liczba po preprocessingu]
- Zmienna docelowa: Cena zamknięcia (closing price)
- Typ problemu: Regresja

2.4 Regression (Our): Student Performance Dataset

Źródło: [Podać źródło jeśli znane]
Opis: Zbiór danych o wynikach egzaminów uczniów.
Charakterystyka:

- Liczba rekordów: 6,607
- Liczba cech (po preprocessingu): 5
- Zmienna docelowa: Wynik egzaminu (exam score)
- Typ problemu: Regresja

Cechy:

- Godziny nauki (study_hours)
- Zaangażowanie rodziców (parental_involvement)
- Dostęp do zasobów edukacyjnych (access_to_resources)

3 Preprocessing danych

3.1 Ogólna procedura

Dla każdego zbioru danych przeprowadzono następujące kroki preprocessingu:

1. **Obsługa brakujących wartości** – usuwanie lub imputacja
2. **Kodowanie zmiennych kategorycznych**
    - One-hot encoding dla zmiennych nominalnych
    - Target encoding dla zmiennych o wysokiej kardynalności
    - Binary encoding dla zmiennych binarnych
    - Ordinal encoding dla zmiennych porządkowych

3. **Feature engineering** – tworzenie nowych cech (np. logarytmy, ilorazy)
4. **Selekcja cech** – analiza korelacji, usuwanie cech o niskiej korelacji z targetem
5. **Balansowanie klas** (dla klasyfikacji) – undersampling klasy większościowej
6. **Standaryzacja** – normalizacja cech numerycznych (mean=0, std=1)
7. **Podział danych** – train/test (80/20) oraz train/val/test (70/15/15)

3.2 Adult Income Dataset

Specyficzne operacje:

- Usunięto brakujące wartości w kolumnach: workclass, occupation, native_country
- Binary encoding: sex, native_country
- Target encoding: occupation (ze względu na wysoką kardynalność)
- One-hot encoding: marital_status, relationship, workclass
- Feature engineering: capital_net_log = log(capital_gain - capital_loss + 1)
- Undersampling: zbalansowano klasy do proporcji 1:1

3.3 Loan Approval Dataset

Specyficzne operacje:

- Selekcja cech: usunięto cechy o korelacji < 0.15 z targetem
- Transformacja logarytmiczna: annual_income, defaults_on_file
- Undersampling: zbalansowano klasy
- Standaryzacja wszystkich cech numerycznych

3.4 Stock Market Dataset

Specyficzne operacje:

- Zachowanie uporządkowania czasowego (time-based split)
- Standaryzacja wszystkich wskaźników
- Podział czasowy: 80/20 i 70/15/15

3.5 Student Performance Dataset

Specyficzne operacje:

- Ordinal encoding: Parental_Involvement, Access_to_Resources
- Binary encoding zmiennych kategorycznych
- Selekcja cech: $|r| > 0.15$ z targetem
- Standaryzacja tylko cech numerycznych (zachowanie interpretacji cech ordinalnych)

4 Architektura sieci neuronowych

4.1 Architektura ogólna

Obie implementacje (ręczna i Keras) wykorzystują tę samą architekturę MLP:

- **Warstwa wejściowa:** liczba neuronów = liczba cech
- **Warstwy ukryte:** N warstw Dense z aktywacją ReLU
- **Warstwa wyjściowa:**
  - Klasyfikacja: Dense(n_classes) + Softmax
  - Regresja: Dense(1) + aktywacja liniowa

4.2 Implementacja ręczna

4.2.1 Warstwa Dense

Warstwa Dense implementuje transformację liniową:

$$\mathbf{y} = \mathbf{W}^T\mathbf{x} + \mathbf{b} \tag{1}$$

Inicjalizacja wag: Xavier/Glorot uniform

$$W \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}}\right) \tag{2}$$

Backward propagation:

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{x}^T\frac{\partial L}{\partial \mathbf{y}} \tag{3}$$

$$\frac{\partial L}{\partial \mathbf{b}} = \sum \frac{\partial L}{\partial \mathbf{y}} \tag{4}$$

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial \mathbf{y}}\mathbf{W}^T \tag{5}$$

4.2.2 Aktywacja ReLU

$$\text{ReLU}(x) = \max(0, x) \tag{6}$$

Gradient:

$$\frac{\partial \text{ReLU}(x)}{\partial x} = \begin{cases} 1 & \text{jeśli } x > 0 \\ 0 & \text{w przeciwnym razie} \end{cases} \tag{7}$$

4.2.3 Aktywacja Softmax (klasyfikacja)

$$\text{Softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{8}$$

4.2.4 Funkcja straty

Klasyfikacja: Cross-entropy loss

$$L = -\frac{1}{N} \sum_{i=1}^{N} \log(\hat{y}_{i,c_i}) \tag{9}$$

gdzie c_i to prawdziwa klasa dla próbki i.

Regresja: Mean Squared Error (MSE)

$$L = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{10}$$

4.2.5 Optymalizator

Stochastic Gradient Descent (SGD):

$$W \leftarrow W - \eta \frac{\partial L}{\partial W} \tag{11}$$

gdzie η to learning rate.

4.3 Implementacja Keras

Implementacja Keras wykorzystuje wysokopoziomowe API:

Listing 1: Przykładowa architektura w Keras

```
1
2
3
4
5
6
7
8
9
10
```

5 Metodologia eksperymentów

5.1 Grid Search

Przeprowadzono systematyczne przeszukiwanie przestrzeni hiperparametrów (grid search):

Hiperparametr	**Wartości**
Liczba warstw ukrytych	{1, 2, 3, 4}
Liczba neuronów na warstwę	{8, 16, 32, 64}
Learning rate	{0.001, 0.005, 0.01, 0.02}
Batch size	32 (stałe)
Liczba epok	50 (stałe)
Liczba kombinacji	64 ($4 \times 4 \times 4$)

Tabela 1: Przestrzeń hiperparametrów

5.2 Procedura eksperymentu

Dla każdej kombinacji hiperparametrów:

1. Przeprowadzono **3 powtórzenia** (runs) z różną inicjalizacją
2. Wybrano **najlepszy run** na podstawie metryki na zbiorze treningowym
3. Zapisano wyniki (metryki train, validation, test)

Łącznie: $64 \times 3 = 192$ treningi dla każdego zbioru danych i typu podziału.

5.3 Metryki ewaluacji

5.3.1 Klasyfikacja

- **Accuracy:** $\text{Acc} = \frac{TP+TN}{TP+TN+FP+FN}$
- **Precision:** $\text{Prec} = \frac{TP}{TP+FP}$
- **Recall:** $\text{Rec} = \frac{TP}{TP+FN}$
- **F1-score:** $F_1 = 2 \cdot \frac{\text{Prec} \cdot \text{Rec}}{\text{Prec}+\text{Rec}}$
- **Cross-entropy loss**

5.3.2 Regresja

- **MSE:** $\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
- **MAE:** $\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$
- **R² score:** $R^2 = 1 - \frac{\sum(y_i-\hat{y}_i)^2}{\sum(y_i-\bar{y})^2}$

5.4 Podziały danych

Dla każdego zbioru zastosowano dwa warianty podziału:

1. **Train/Test (80/20):**
   - 80% danych do treningu
   - 20% danych do testowania
2. **Train/Validation/Test (70/15/15):**
   - 70% danych do treningu
   - 15% danych do walidacji (wybór najlepszego modelu)
   - 15% danych do testowania końcowego

6 Wyniki eksperymentów

6.1 Adult Income (Klasyfikacja binarna)

6.1.1 Najlepsze konfiguracje

Hiperparametr	Wartość
Liczba warstw ukrytych	4
Liczba neuronów	16
Learning rate	0.0100
Test Accuracy	0.829723
Test Precision	0.832371
Test Recall	0.829723
Test F1-score	0.831044

Tabela 2: Najlepsza konfiguracja - Manual MLP, Adult Income

Manual MLP

Hiperparametr	Wartość
Liczba warstw ukrytych	3
Liczba neuronów	16
Learning rate	0.0050
Test Accuracy	0.827123
Test Precision	0.829648
Test Recall	0.827123
Test F1-score	0.828384

Tabela 3: Najlepsza konfiguracja - Keras MLP, Adult Income

Keras MLP

6.1.2 Porównanie

Metryka	Manual MLP	Keras MLP
Test Accuracy	0.829723	0.827123
Test Precision	0.832371	0.829648
Test Recall	0.829723	0.827123
Test F1-score	0.831044	0.828384

Tabela 4: Porównanie Manual vs Keras - Adult Income

6.2 Loan Approval (Klasyfikacja binarna)

6.2.1 Najlepsze konfiguracje

Hiperparametr	Wartość
Liczba warstw ukrytych	3
Liczba neuronów	32
Learning rate	0.0200
Test Accuracy	0.863117
Test Precision	0.863121
Test Recall	0.863117
Test F1-score	0.863119

Tabela 5: Najlepsza konfiguracja - Manual MLP, Loan Approval

Manual MLP

Hiperparametr	Wartość
Liczba warstw ukrytych	2
Liczba neuronów	64
Learning rate	0.0200
Test Accuracy	0.862969
Test Precision	0.863355
Test Recall	0.862971
Test F1-score	0.863163

Tabela 6: Najlepsza konfiguracja - Keras MLP, Loan Approval

Keras MLP

6.2.2 Porównanie

Metryka	Manual MLP	Keras MLP
Test Accuracy	0.863117	0.862969
Test Precision	0.863121	0.863355
Test Recall	0.863117	0.862971
Test F1-score	0.863119	0.863163

Tabela 7: Porównanie Manual vs Keras - Loan Approval

6.3 Stock Market (Regresja)

6.3.1 Najlepsze konfiguracje

Hiperparametr	Wartość
Liczba warstw ukrytych	2
Liczba neuronów	8
Learning rate	0.0200
Test MSE	149.155077
Test MAE	9.931640
Test R^2	-1.248412

Tabela 8: Najlepsza konfiguracja - Manual MLP, Stock Market

Manual MLP

Hiperparametr	Wartość
Liczba warstw ukrytych	1
Liczba neuronów	64
Learning rate	0.0010
Test MSE	0.076797
Test MAE	0.204588
Test R^2	0.998842

Tabela 9: Najlepsza konfiguracja - Keras MLP, Stock Market

Keras MLP

Hiperparametr	Wartość
Liczba warstw Conv1D	1
Liczba filtrów	64
Learning rate	0.0010
Optimizer	rmsprop
Test MSE	0.102108
Test MAE	0.248630
Test R^2	0.998461

Tabela 10: Najlepsza konfiguracja - CNN 1D, Stock Market

Keras CNN 1D

Hiperparametr	Wartość
Liczba warstw LSTM	1
Liczba jednostek LSTM	32
Learning rate	0.0010
Optimizer	rmsprop
Test MSE	0.184879
Test MAE	0.291519
Test R^2	0.997213

Tabela 11: Najlepsza konfiguracja - LSTM, Stock Market

Keras LSTM

6.3.2 Porównanie

Metryka	Manual MLP	Keras MLP	CNN 1D	LSTM
Test MSE	149.155077	0.076797	0.102108	0.184879
Test MAE	9.931640	0.204588	0.248630	0.291519
Test R^2	-1.248412	0.998842	0.998461	0.997213

Tabela 12: Porównanie wszystkich modeli - Stock Market

Uwaga: Manual MLP uzyskał bardzo słabe wyniki na tym zbiorze (ujemny R^2), co wskazuje na problemy z konwergencją. Keras MLP osiągnął najlepsze wyniki, wyprzedzając nawet zaawansowane architektury CNN 1D i LSTM.

6.4 Student Performance (Regresja)

6.4.1 Najlepsze konfiguracje

Hiperparametr	Wartość
Liczba warstw ukrytych	1
Liczba neuronów	8
Learning rate	0.0010
Test MSE	5.842659
Test MAE	1.169886
Test R^2	0.646251

Tabela 13: Najlepsza konfiguracja - Manual MLP, Student Performance

Manual MLP

Hiperparametr	Wartość
Liczba warstw ukrytych	1
Liczba neuronów	32
Learning rate	0.0200
Test MSE	5.788280
Test MAE	1.145609
Test R^2	0.649544

Tabela 14: Najlepsza konfiguracja - Keras MLP, Student Performance

Keras MLP

6.4.2 Porównanie

Metryka	Manual MLP	Keras MLP
Test MSE	5.842659	5.788280
Test MAE	1.169886	1.145609
Test R^2	0.646251	0.649544

Tabela 15: Porównanie Manual vs Keras - Student Performance

6.5 Fashion MNIST (Klasyfikacja obrazów)

6.5.1 Najlepsze konfiguracje

Hiperparametr	Wartość
Liczba warstw ukrytych	2
Liczba neuronów	128
Learning rate	0.0010
Test Accuracy	0.740300
Test Precision	0.741498
Test Recall	0.740300
Test F1-score	0.740899

Tabela 16: Najlepsza konfiguracja - Manual MLP, Fashion MNIST

Manual MLP

Hiperparametr	Wartość
Liczba warstw ukrytych	3
Liczba neuronów	128
Learning rate	0.0010
Optimizer	sgd
Test Accuracy	0.732200
Test Precision	0.735203
Test Recall	0.732200
Test F1-score	0.733698

Tabela 17: Najlepsza konfiguracja - Keras MLP, Fashion MNIST

Keras MLP

Hiperparametr	Wartość
Liczba warstw Conv2D	2
Filtry	[64, 128]
Learning rate	0.0010
Optimizer	adam
Test Accuracy	0.928800
Test Precision	0.928457
Test Recall	0.928800
Test F1-score	0.928628

Tabela 18: Najlepsza konfiguracja - Keras CNN, Fashion MNIST

Keras CNN

6.5.2 Porównanie

Metryka	Manual MLP	Keras MLP	Keras CNN
Test Accuracy	0.740300	0.732200	**0.928800**
Test Precision	0.741498	0.735203	**0.928457**
Test Recall	0.740300	0.732200	**0.928800**
Test F1-score	0.740899	0.733698	**0.928628**

Tabela 19: Porównanie wszystkich modeli - Fashion MNIST

Wnioski:

- CNN osiąga znacząco lepsze wyniki (92.88% accuracy) niż MLP (74.03% / 73.22%)
- Przewaga CNN wynika z wykorzystania lokalnych wzorców przestrzennych w obrazach

- MLP musi traktować obraz jako płaski wektor 784 cech, tracąc informację o strukturze 2D

6.6 Podsumowanie wyników

Zbiór danych	Typ	Manual MLP	Keras MLP	Najlepszy model
Adult Income	Klasyfikacja	82.97%	82.71%	Manual MLP
Loan Approval	Klasyfikacja	86.31%	86.30%	Manual MLP
Stock Market	Regresja	R^2 = -1.25	R^2 = 0.999	Keras MLP
Student Perf.	Regresja	R^2 = 0.646	R^2 = 0.650	Keras MLP
Fashion MNIST	Klasyfikacja	74.03%	73.22%	CNN (92.88%)

Tabela 20: Podsumowanie najlepszych wyników dla wszystkich zbiorów danych

Główne obserwacje:

1. **Klasyfikacja tabularna:** Manual MLP i Keras MLP osiągają porównywalne wyniki, różnica $< 0.5\%$
2. **Regresja:** Keras znacząco przewyższa Manual MLP, szczególnie dla Stock Market
3. **Obrazy:** CNN zdecydowanie lepszy od MLP dla Fashion MNIST (+18.85% accuracy)
4. **Wpływ optymalizatora:** Adam i RMSprop generalnie lepsze od SGD dla skomplikowanych zadań

7 Wnioski

7.1 Porównanie implementacji

Na podstawie przeprowadzonych eksperymentów można wyciągnąć następujące wnioski:
Implementacja ręczna (Manual MLP):

- Pełna kontrola nad procesem treningu i możliwość debugowania
- Wysoka wartość edukacyjna – zrozumienie backpropagation od podstaw
- Wolniejsza ze względu na brak optymalizacji niskopoziomowych
- Dobre wyniki dla klasyfikacji tabelarycznej (82.97% Adult Income, 86.31% Loan Approval)
- Problemy z konwergencją dla skomplikowanych danych regresyjnych (ujemny R^2 dla Stock Market)

Implementacja Keras:

- Znacznie szybsza (5-10x) dzięki zoptymalizowanemu backendowi
- Łatwość implementacji różnych architektur (MLP, CNN, LSTM)
- Porównywalne wyniki dla klasyfikacji (różnica <0.5%)
- Zdecydowanie lepsze dla regresji ($R^2 = 0.999$ vs -1.25 dla Stock Market)
- Możliwość wykorzystania różnych optimizerów (Adam, RMSprop)

7.2 Wyniki eksperymentów

Klasyfikacja:

- Obie implementacje osiągnęły porównywalne wyniki (różnica <0.5%)
- Adult Income: Manual 82.97% vs Keras 82.71%
- Loan Approval: Manual 86.31% vs Keras 86.30%
- Wyniki zgodne z literaturą dla tych zbiorów danych

Regresja:

- Keras znacząco lepszy od Manual MLP
- Stock Market: Keras R^2=0.999 vs Manual R^2=-1.25
- Student Performance: Keras R^2=0.650 vs Manual R^2=0.646
- Manual MLP ma problemy z konwergencją dla danych finansowych

Fashion MNIST (obrazy):

- CNN zdecydowanie przewyższa MLP (+18.85% accuracy)
- Keras CNN: 92.88% vs Manual MLP: 74.03% vs Keras MLP: 73.22%
- Potwierdza to, że CNN lepiej nadaje się do analizy obrazów

7.3 Problemy napotkane

1. **Konwergencja Manual MLP dla regresji:** Model ręczny miał problemy z optymalizacją dla danych Stock Market. Prawdopodobna przyczyna: zbyt prosty optymalizator (SGD bez momentum).
2. **Wymiary warstw w CNN 1D:** Przy małej liczbie cech (5 dla Stock Market) pooling zbyt mocno redukował wymiary. Rozwiązanie: warunkowe stosowanie poolingu.
3. **Czas treningu:** Eksperymenty na Fashion MNIST zajmowały kilka godzin. Rozwiązanie: zwiększenie batch size, zmniejszenie liczby epok.
4. **Zarządzanie hiperparametrami:** Grid search generował setki kombinacji. Rozwiązanie: systematyczne śledzenie eksperymentów w plikach Excel.

7.4 Możliwe usprawnienia

- **Zaimplementowane:**
  - Momentum w Manual MLP (poprawia konwergencję)
  - Różne optymalizatory w Keras (Adam, RMSprop, SGD)
  - CNN dla obrazów i CNN 1D dla szeregów czasowych
  - LSTM dla danych sekwencyjnych
- **Do zaimplementowania:**
  - Regularyzacja (L1, L2, Dropout) – zapobiega overfittingowi
  - Early stopping – automatyczne zatrzymanie treningu
  - Learning rate decay – zmniejszanie LR w trakcie treningu
  - Batch normalization – stabilizuje trening
  - Data augmentation (dla obrazów) – zwiększa ilość danych treningowych

8 Bibliografia

1. Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press.
2. Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.
3. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.
4. TensorFlow Documentation:
5. Keras Documentation:
6. UCI Machine Learning Repository:

A Kod źródłowy

Pełny kod źródłowy projektu dostępny jest w repozytorium GitHub:

Struktura projektu:

- – Implementacja ręczna
- – Implementacja Keras
- – Zbiory danych i preprocessing
- – Narzędzia pomocnicze (experiment runner, visualization)
- – Wyniki eksperymentów i wizualizacje