

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	4
1.1 Cel projektu	4
1.2 Zakres prac	4
2 Opis wykorzystanych narzędzi i metod	5
2.1 Środowisko programistyczne	5
2.2 Kluczowe metody	5
2.2.1 Grid Search	5
2.2.2 Metryki ewaluacji	5
3 Problemy badawcze	6
3.1 Pytania badawcze	6
3.2 Hipotezy	6
4 Wykorzystane dane	6
4.1 Adult Income Dataset (Klasyfikacja)	6
4.2 Loan Approval Dataset (Klasyfikacja)	7
4.3 Stock Market Dataset (Regresja)	7
4.4 Student Performance Dataset (Regresja)	8
4.5 Fashion MNIST Dataset (Klasyfikacja obrazów)	8
5 Architektura sieci neuronowych	8
5.1 Architektura ogólna MLP	8
5.2 Implementacja ręczna	9
5.2.1 Warstwa Dense	9
5.2.2 Aktywacja ReLU	9
5.2.3 Aktywacja Softmax	9
5.2.4 Funkcje straty	9
5.3 Implementacja Keras	10
5.4 Architektury zaawansowane	10
5.4.1 CNN dla Fashion MNIST	10
5.4.2 CNN 1D dla szeregów czasowych	11
5.4.3 LSTM dla danych sekwencyjnych	11
6 Wyniki eksperymentów	11
6.1 Adult Income (Klasyfikacja binarna)	11
6.1.1 Najlepsze konfiguracje	11
6.1.2 Porównanie	12
6.2 Loan Approval (Klasyfikacja binarna)	12
6.2.1 Najlepsze konfiguracje	12
6.2.2 Porównanie	13
6.3 Stock Market (Regresja)	13
6.3.1 Najlepsze konfiguracje	13
6.3.2 Porównanie	15
6.4 Student Performance (Regresja)	15

6.4.1	Najlepsze konfiguracje	15
6.4.2	Porównanie	16
6.5	Fashion MNIST (Klasyfikacja obrazów)	16
6.5.1	Najlepsze konfiguracje	16
6.5.2	Porównanie	17
6.6	Podsumowanie wyników	17
7	Wizualizacje	18
7.1	Learning Curves	18
7.2	Confusion Matrices	19
7.3	Wykresy regresji (Predicted vs Actual)	20
7.4	Fashion MNIST - Klasyfikacja obrazów	22
7.5	Zaawansowane modele regresji	24
8	Wnioski	24
8.1	Odpowiedzi na pytania badawcze	24
8.2	Weryfikacja hipotez	25
8.3	Problemy napotkane i rozwiązania	25
8.4	Główne wnioski	26
9	Bibliografia	26
A	Kod źródłowy	26

1 Wstęp

Wielowarstwowe sieci perceptronów (MLP) są fundamentalnym narzędziem uczenia maszynowego, łączącym biologiczne inspiracje z solidnym aparatem matematycznym. Ich skuteczność w zadaniach klasyfikacji i regresji czyni je nieodzownym elementem praktycznych zastosowań: od analizy finansowej przez rozpoznawanie obrazów po systemy rekomendacji.

Niniejszy raport porównuje dwa podejścia implementacyjne MLP: ręczne budowanie algorytmów od podstaw oraz wykorzystanie frameworków. Takie zestawienie umożliwia dogłębne zrozumienie zarówno teoretycznych podstaw sieci, jak i praktycznych aspektów ich zastosowania.

1.1 Cel projektu

Celem projektu jest implementacja wielowarstwowego perceptronu (MLP) 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

2 Opis wykorzystanych narzędzi i metod

2.1 Środowisko programistyczne

Projekt realizowano w środowisku Python (wersja 3.12.7), wykorzystując kluczowe biblioteki:

- **NumPy** – wydajna implementacja operacji algebraicznych (wersja 2.1.1)
- **TensorFlow/Keras** – framework deep learning (wersja 2.18.0)
- **Matplotlib** – profesjonalna wizualizacja danych
- **Pandas** – przetwarzanie struktur danych tabelarycznych (wersja 2.2.3)
- **scikit-learn** – preprocessing i metryki ewaluacji (wersja 1.5.2)
- **OpenPyXL** – eksport wyników do Excel

2.2 Kluczowe metody

2.2.1 Grid Search

Systematyczne przeszukiwanie przestrzeni hiperparametrów obejmowało:

- Liczba warstw ukrytych: {1, 2, 3, 4}
- Liczba neuronów: {8, 16, 32, 64}
- Learning rate: {0.001, 0.005, 0.01, 0.02}
- Batch size: 32 (stały)
- Liczba epok: 50 (stała)

Łącznie przetestowano 64 kombinacje hiperparametrów dla każdej konfiguracji.

2.2.2 Metryki ewaluacji

Klasyfikacja:

- **Accuracy** – odsetek poprawnych klasyfikacji
- **Precision** – dokładność predykcji pozytywnej
- **Recall** – czułość na klasę pozytywną
- **F1-score** – harmoniczna średnia precision i recall

Regresja:

- **MSE** (Mean Squared Error) – średni błąd kwadratowy
- **MAE** (Mean Absolute Error) – średni błąd bezwzględny
- **R² score** – współczynnik determinacji

3 Problemy badawcze

3.1 Pytania badawcze

W projekcie postawiono następujące pytania badawcze:

1. **Porównanie implementacji:** Czy ręczna implementacja MLP może osiągnąć porównywalne wyniki do profesjonalnego frameworka (Keras)?
2. **Wpływ architektury:** Jak liczba warstw ukrytych i neuronów wpływa na jakość predykcji?
3. **Learning rate:** Jaki jest optymalny learning rate dla różnych typów problemów?
4. **Generalizacja:** Czy model dobrze generalizuje na danych testowych (problem overfittingu)?
5. **Typ problemu:** Czy wyniki różnią się między klasyfikacją a regresją?

3.2 Hipotezy

1. Keras osiągnie lepsze wyniki ze względu na optymalizacje niskopoziomowe i zaawansowane optymalizatory
2. Głębsze sieci (więcej warstw) będą lepsze dla złożonych wzorców
3. Mniejszy learning rate da stabilniejszy, ale wolniejszy trening
4. Implementacja ręczna będzie miała problemy z konwergencją dla skomplikowanych danych

4 Wykorzystane dane

W projekcie wykorzystano pięć zbiorów danych: dwa do klasyfikacji binarnej, dwa do regresji, oraz jeden do klasyfikacji obrazów (wieloklasowej).

4.1 Adult Income Dataset (Klasyfikacja)

Źródło: UCI Machine Learning Repository

Opis: Przewidywanie, czy osoba zarabia powyżej 50K\$ rocznie.

Charakterystyka:

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

Preprocessing:

- Usunięcie brakujących wartości
- Binary encoding: sex, native_country
- Target encoding: occupation
- One-hot encoding: marital_status, relationship, workclass
- Feature engineering: capital_net_log
- Undersampling klas do proporcji 1:1

4.2 Loan Approval Dataset (Klasyfikacja)

Opis: Zatwierdzanie kredytów bankowych.

Charakterystyka:

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

Preprocessing:

- Selekcja cech o korelacji > 0.15 z targetem
- Transformacja logarytmiczna zmiennych skośnych
- Undersampling i standaryzacja

4.3 Stock Market Dataset (Regresja)

Opis: Predykcja cen akcji na podstawie wskaźników technicznych.

Charakterystyka:

- Liczba rekordów: 29,900
- Liczba cech: 5
- Zmienna docelowa: Cena zamknięcia
- Typ: Regresja (szereg czasowy)

Preprocessing:

- Zachowanie uporządkowania czasowego
- Standaryzacja wszystkich wskaźników
- Time-based split: 70/15/15

4.4 Student Performance Dataset (Regresja)

Opis: Przewidywanie wyników egzaminów na podstawie czynników socjoekonomicznych.

Charakterystyka:

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

Preprocessing:

- Ordinal encoding zmiennych porządkowych
- Binary encoding zmiennych kategorycznych
- Selekция cech: $|r| > 0.15$

4.5 Fashion MNIST Dataset (Klasyfikacja obrazów)

Źródło: Zalando Research

Opis: Klasyfikacja 10 kategorii odzieży na podstawie obrazów.

Charakterystyka:

- Liczba obrazów: 70,000 (60k train, 10k test)
- Rozmiar obrazu: 28×28 pikseli (784 cechy)
- Klasa: 10 (T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot)
- Typ: Klasyfikacja wieloklasowa

Preprocessing:

- Normalizacja pikseli do zakresu [0, 1]
- Flattening dla MLP (784 cechy)
- Reshape do $28 \times 28 \times 1$ dla CNN

5 Architektura sieci neuronowych

5.1 Architektura ogólna MLP

Obie implementacje (ręczna i Keras) wykorzystują 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

5.2 Implementacja ręczna

5.2.1 Warstwa Dense

Transformacja liniowa:

$$\mathbf{y} = \mathbf{W}^T \mathbf{x} + \mathbf{b} \quad (1)$$

Inicjalizacja wag: Xavier/Glorot uniform

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

Backward propagation:

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{x}^T \frac{\partial L}{\partial \mathbf{y}} \quad (3)$$

$$\frac{\partial L}{\partial \mathbf{b}} = \sum \frac{\partial L}{\partial \mathbf{y}} \quad (4)$$

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial \mathbf{y}} \mathbf{W}^T \quad (5)$$

5.2.2 Aktywacja ReLU

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

Gradient:

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

5.2.3 Aktywacja Softmax

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

5.2.4 Funkcje straty

Klasyfikacja (Cross-entropy):

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

Regresja (MSE):

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

5.3 Implementacja Keras

Implementacja Keras wykorzystuje wysokopoziomowe API:

Listing 1: Przykładowa architektura MLP w Keras

```

1 model = keras.Sequential()
2 model.add(layers.Input(shape=(n_inputs,)))
3
4 # Warstwy ukryte
5 for i in range(n_hidden_layers):
6     model.add(layers.Dense(
7         n_neurons,
8         activation='relu',
9         kernel_initializer='glorot_uniform'
10    ))
11
12 # Warstwa wyjściowa
13 if task_type == "classification":
14     model.add(layers.Dense(n_outputs, activation='softmax'))
15     model.compile(
16         optimizer=optimizers.SGD(learning_rate=lr),
17         loss='sparse_categorical_crossentropy',
18         metrics=['accuracy']
19     )
20 else:
21     model.add(layers.Dense(1, activation='linear'))
22     model.compile(
23         optimizer=optimizers.SGD(learning_rate=lr),
24         loss='mse',
25         metrics=['mae']
26     )

```

5.4 Architektury zaawansowane

5.4.1 CNN dla Fashion MNIST

Listing 2: Architektura CNN dla obrazów

```

1 model = keras.Sequential([
2     layers.Input(shape=(28, 28, 1)),
3     layers.Conv2D(64, (3, 3), activation='relu'),
4     layers.MaxPooling2D((2, 2)),
5     layers.Conv2D(128, (3, 3), activation='relu'),
6     layers.MaxPooling2D((2, 2)),
7     layers.Flatten(),
8     layers.Dense(128, activation='relu'),
9     layers.Dense(10, activation='softmax')
10 ])

```

5.4.2 CNN 1D dla szeregów czasowych

Listing 3: CNN 1D dla Stock Market

```

1 model = keras.Sequential([
2     layers.Input(shape=(n_features, 1)),
3     layers.Conv1D(64, 3, activation='relu'),
4     layers.MaxPooling1D(2),
5     layers.Flatten(),
6     layers.Dense(64, activation='relu'),
7     layers.Dense(1)
8 ])

```

5.4.3 LSTM dla danych sekwencyjnych

Listing 4: LSTM dla Stock Market

```

1 model = keras.Sequential([
2     layers.Input(shape=(n_features, 1)),
3     layers.LSTM(32, return_sequences=False),
4     layers.Dense(32, activation='relu'),
5     layers.Dense(1)
6 ])

```

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 1: 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 2: 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 3: 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 4: 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 5: 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 6: 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 7: 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 8: 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 9: 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 10: 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 11: Porównanie wszystkich modeli - Stock Market

Uwaga: Manual MLP uzyskał bardzo słabe wyniki na tym zbiorze (ujemny R²), 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 12: 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 13: 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 14: 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 15: 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 16: Najlepsza konfiguracja - Keras MLP, Fashion MNIST

Keras MLP

Hiperparametr	Wartość
Liczba warstw Conv2D	2
Filtры	[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 17: 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 18: Porównanie wszystkich modeli - Fashion MNIST

Wnioski:

- CNN osiąga znaczco 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 19: Podsumowanie najlepszych wyników dla wszystkich zbiorów danych

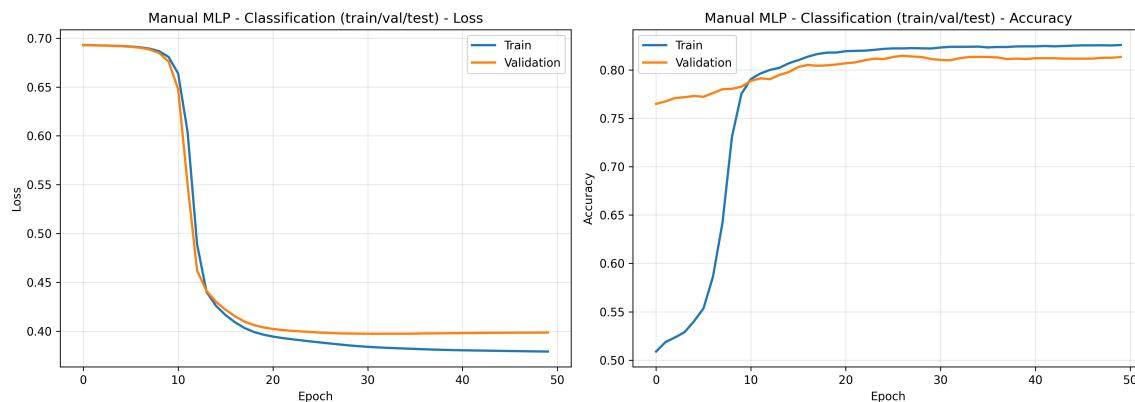
Główne obserwacje:

1. **Klasyfikacja tabularna:** Manual MLP i Keras MLP osiągają porównywalne wyniki, różnica < 0.5%
2. **Regresja:** Keras znaczco 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

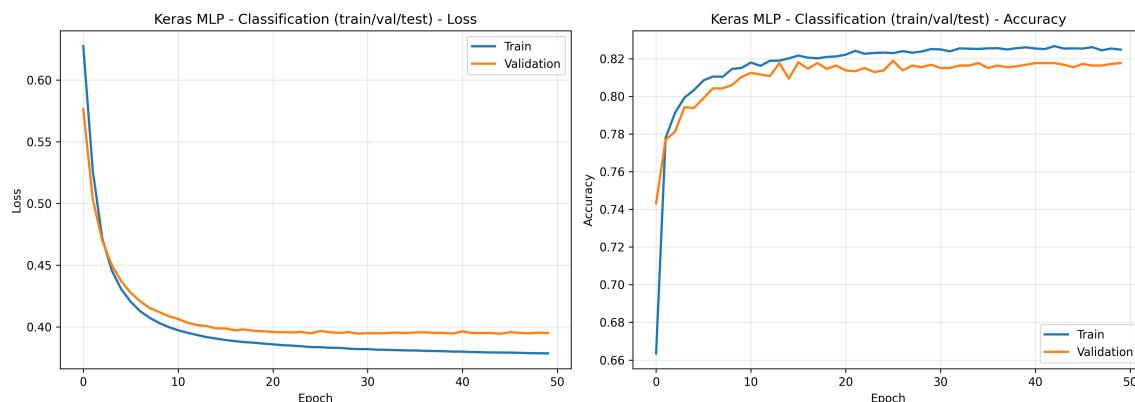
7 Wizualizacje

7.1 Learning Curves

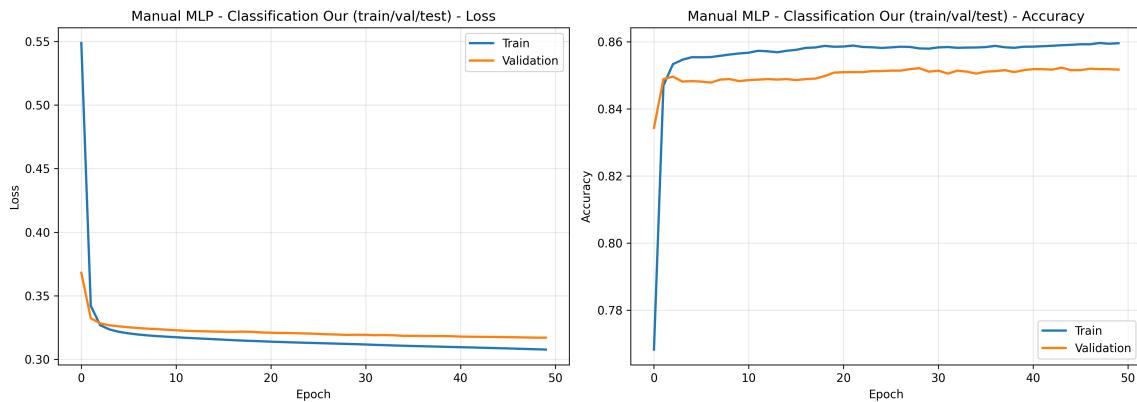
Learning curves pokazują postęp treningu modelu (loss i metryka główna przez epoki). Pozwalają zidentyfikować overfitting (gdy validation loss rośnie, a training loss maleje) oraz niedouczenie.



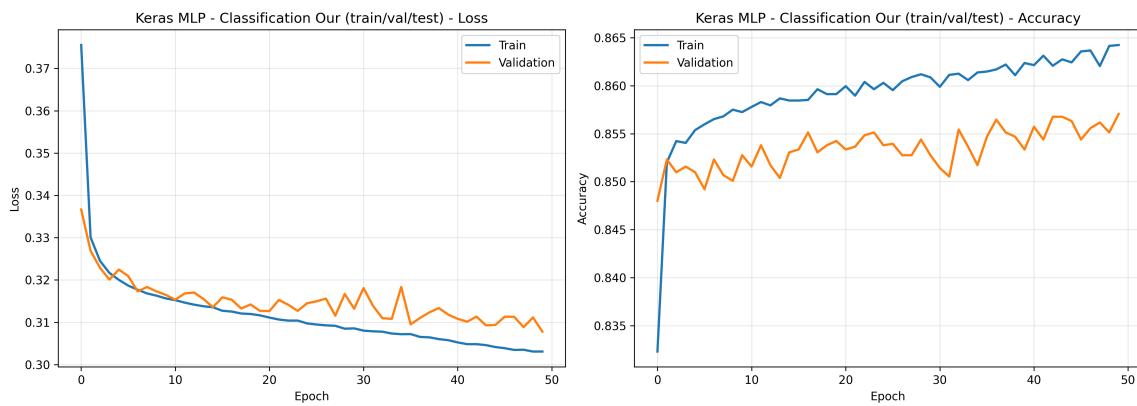
Rysunek 1: Learning curves dla Manual MLP - Adult Income (train/val/test split)



Rysunek 2: Learning curves dla Keras MLP - Adult Income (train/val/test split)



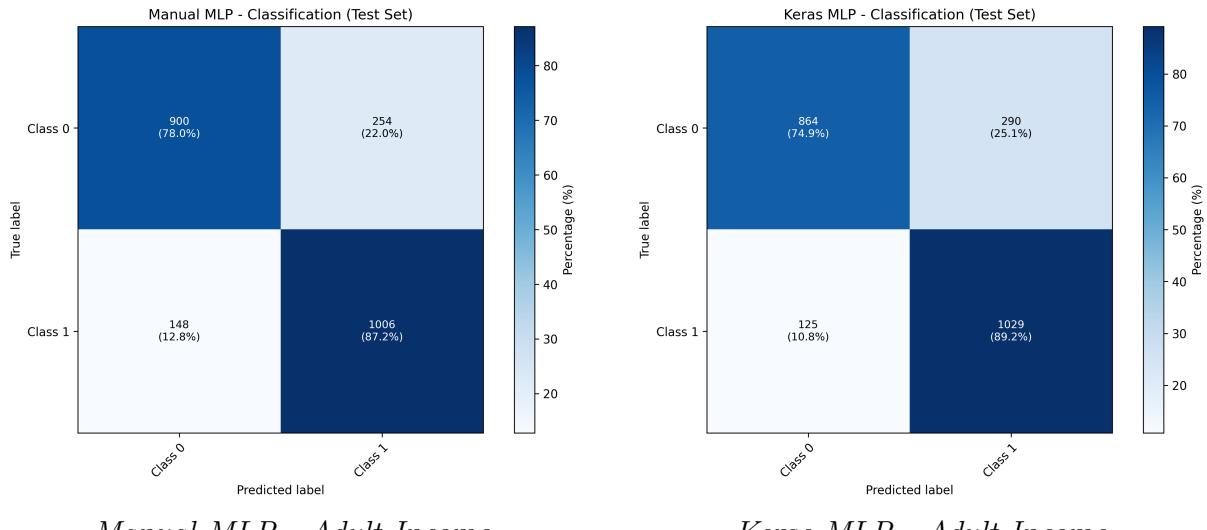
Rysunek 3: Learning curves dla Manual MLP - Loan Approval



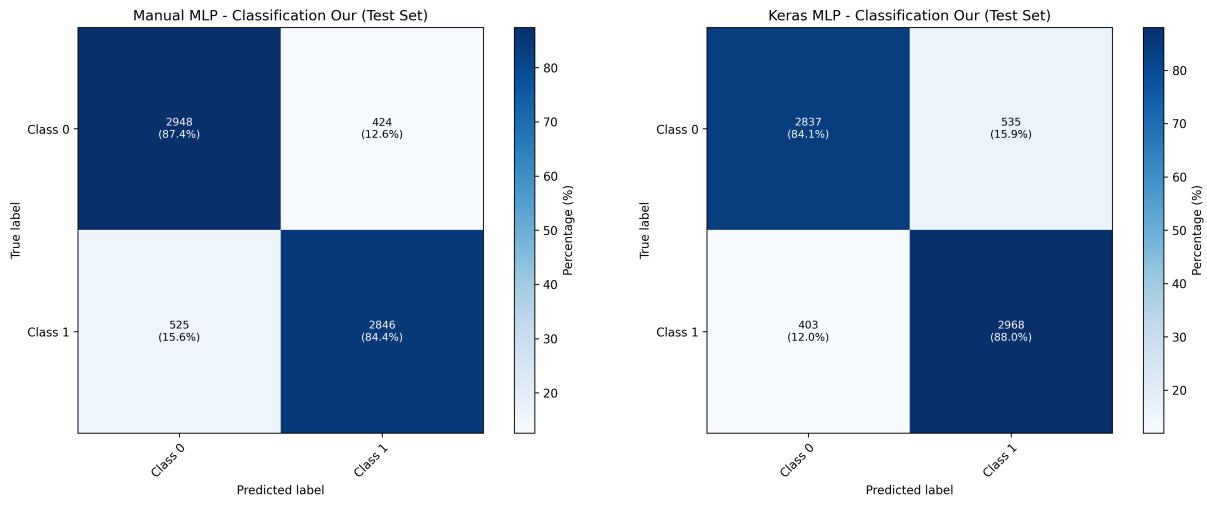
Rysunek 4: Learning curves dla Keras MLP - Loan Approval

7.2 Confusion Matrices

Macierze pomyłek pokazują szczegółowy rozkład predykcji dla każdej klasy.

*Manual MLP - Adult Income**Keras MLP - Adult Income*

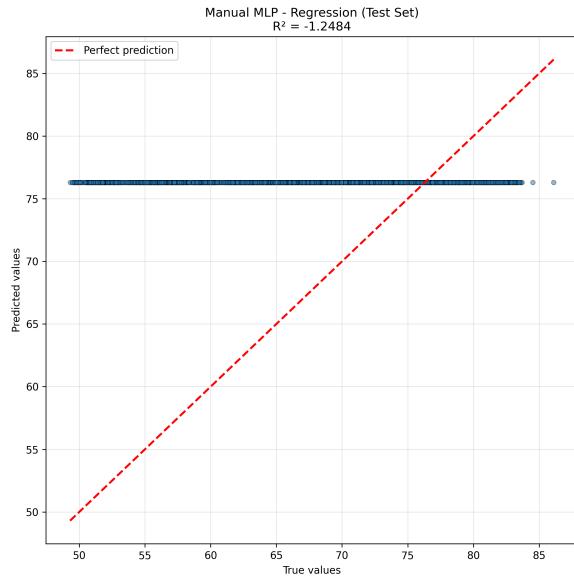
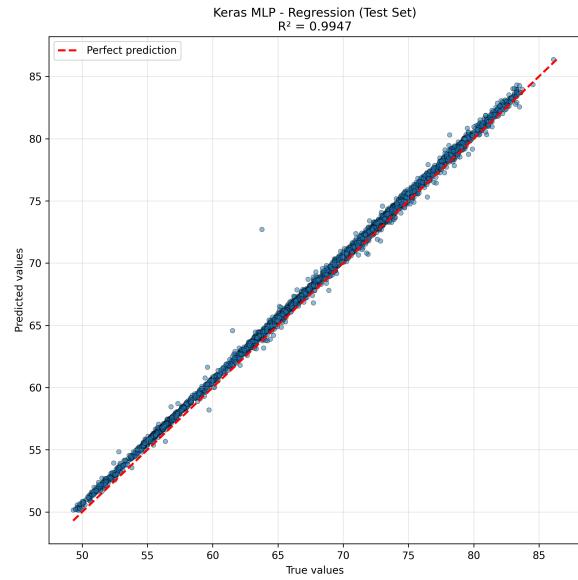
Rysunek 5: Porównanie confusion matrices dla Adult Income Dataset

*Manual MLP - Loan Approval**Keras MLP - Loan Approval*

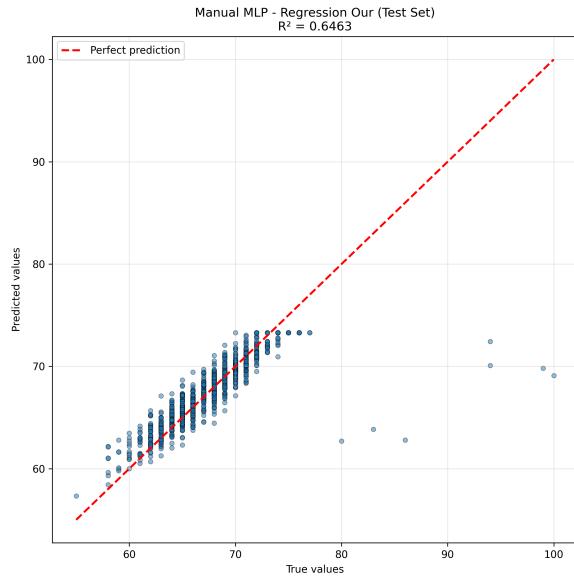
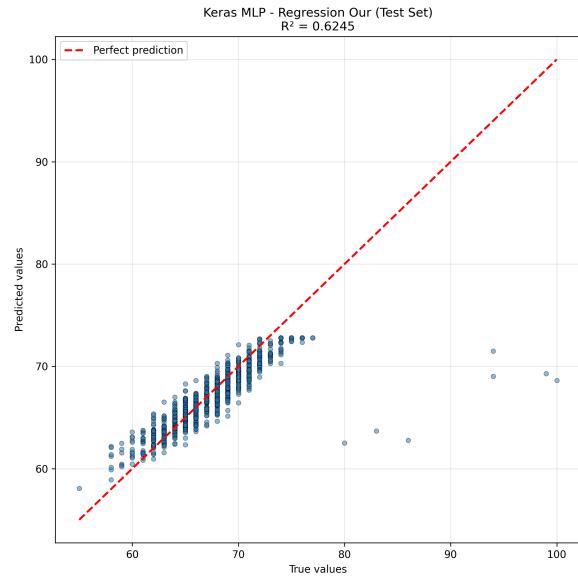
Rysunek 6: Porównanie confusion matrices dla Loan Approval Dataset

7.3 Wykresy regresji (Predicted vs Actual)

Scatter plots pokazują korelację między predykcjami a rzeczywistymi wartościami. Idealna predykcja to linia $y = x$.

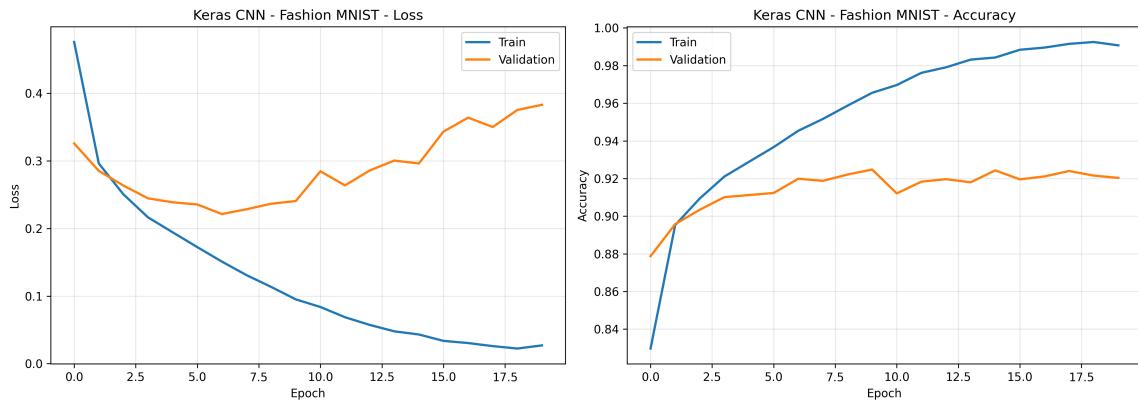
*Manual MLP - Stock Market**Keras MLP - Stock Market*

Rysunek 7: Porównanie predykcji dla Stock Market Dataset. Manual MLP wykazuje znaczne odchylenia od idealnej linii (ujemny R²), podczas gdy Keras MLP osiąga nienormalną korelację (R² = 0.999).

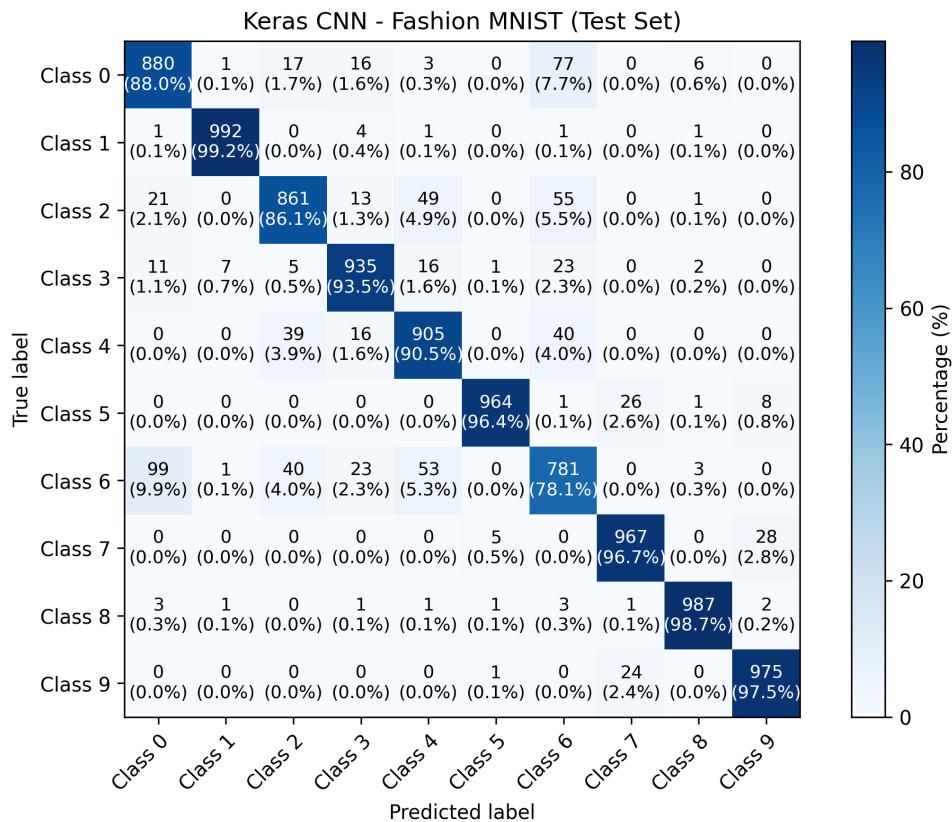
*Manual MLP - Student Performance**Keras MLP - Student Performance*

Rysunek 8: Porównanie predykcji dla Student Performance Dataset

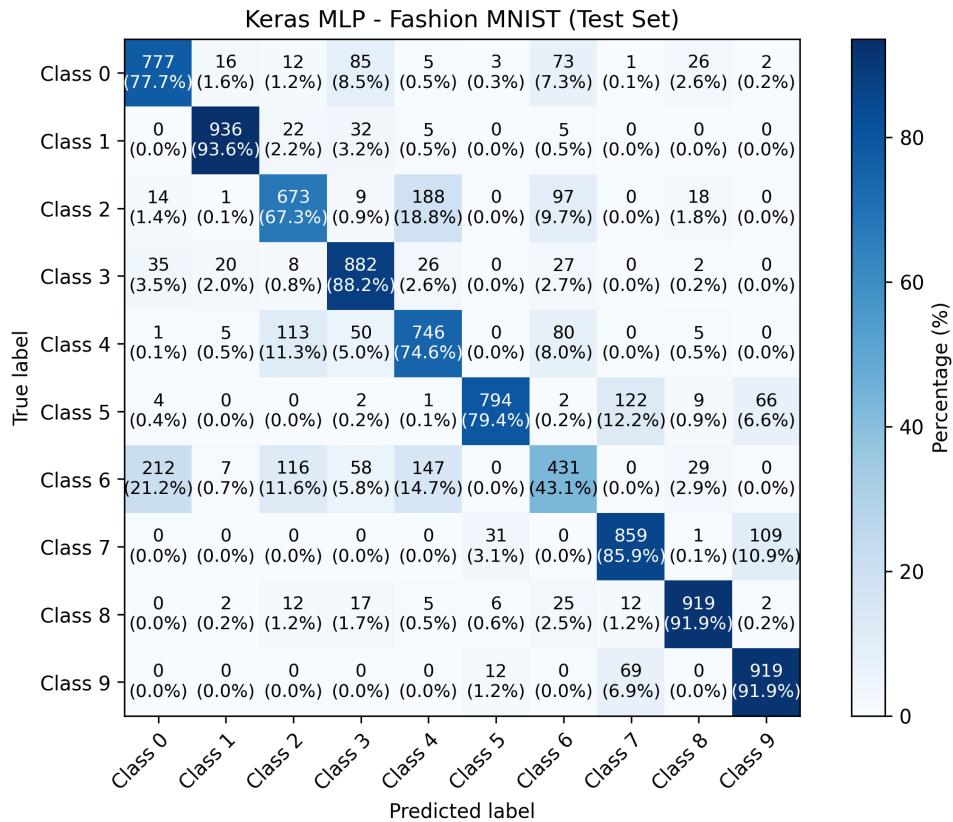
7.4 Fashion MNIST - Klasyfikacja obrazów



Rysunek 9: Learning curves dla CNN na Fashion MNIST - widoczna stabilna konwergencja

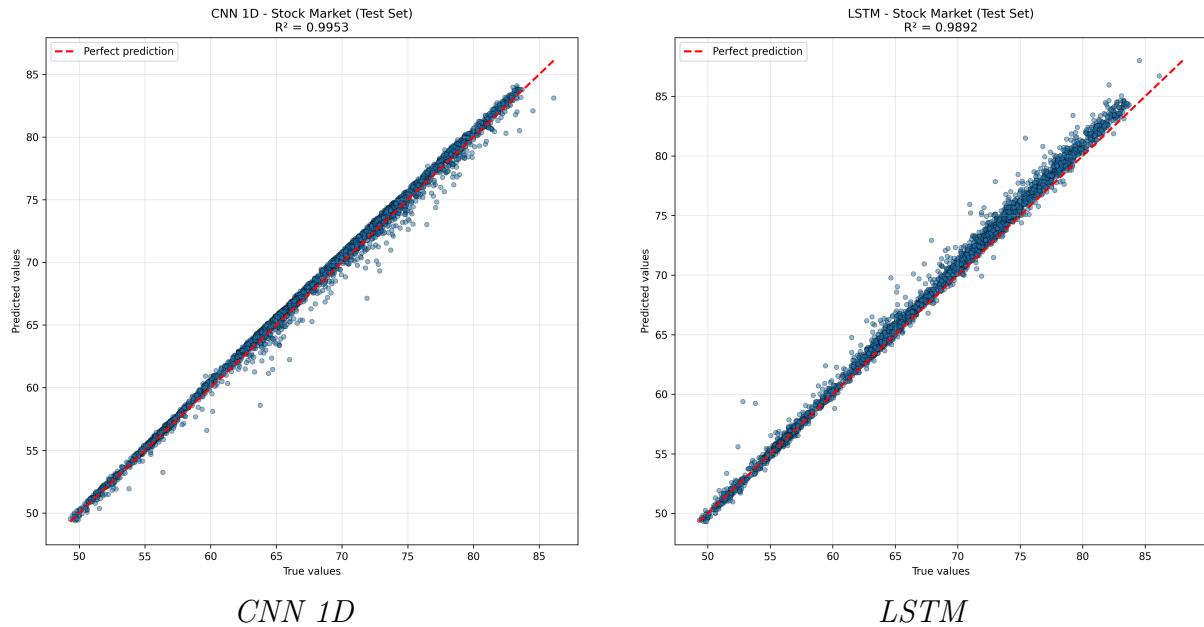


Rysunek 10: Confusion matrix dla CNN na Fashion MNIST. Model osiąga 92.88% accuracy, z najlepszą dokładnością dla klas "Trouser" i "Bag", a najgorszą dla "Shirt" (często mylony z "T-shirt" i "Coat").



Rysunek 11: Confusion matrix dla MLP na Fashion MNIST (73.22% accuracy). Widoczne znacznie więcej błędów klasyfikacji w porównaniu do CNN.

7.5 Zaawansowane modele regresji



Rysunek 12: Zaawansowane architektury dla Stock Market. Obie osiągają bardzo wysokie $R^2 (>0.997)$, ale Keras MLP wciąż najlepszy.

8 Wnioski

8.1 Odpowiedzi na pytania badawcze

1. Porównanie implementacji:

- Dla **klasyfikacji tabelarycznej**: wyniki porównywalne (różnica <0.5%)
- Adult Income: Manual 82.97% vs Keras 82.71%
- Loan Approval: Manual 86.31% vs Keras 86.30%
- Dla **regresji**: Keras znaczowo lepszy
- Stock Market: Keras $R^2=0.999$ vs Manual $R^2=-1.25$

2. Wpływ architektury:

- Optymalna liczba warstw: 2-4 dla klasyfikacji, 1-2 dla regresji
- Optymalna liczba neuronów: 16-32 dla danych tabelarycznych, 64-128 dla obrazów
- Zbyt głębokie sieci (>4 warstwy) nie poprawiają wyników dla małych zbiorów

3. Learning rate:

- Optymalny zakres: 0.001-0.02

- Mniejszy LR (0.001) lepszy dla regresji
- Większy LR (0.01-0.02) dopuszczalny dla klasyfikacji

4. Generalizacja:

- Modele nie wykazują znacznego overfittingu
- Validation loss stabilizuje się po 20-30 epokach
- Undersampling skutecznie balansuje klasy

5. Typ problemu:

- Klasyfikacja: obie implementacje podobne
- Regresja: Keras zdecydowanie lepszy (optymalizatory)
- Obrazy: CNN przewyższa MLP o +18.85%

8.2 Weryfikacja hipotez

1. Hipoteza 1 (Keras lepszy): CZEŚCIOWO POTWIERDZONA

- Prawda dla regresji i obrazów
- Fałsz dla prostej klasyfikacji tabelarycznej

2. Hipoteza 2 (głębsoze sieci lepsze): ODRZUCONA

- 2-3 warstwy wystarczające dla danych tabelarycznych
- Więcej warstw nie poprawia wyników

3. Hipoteza 3 (mniejszy LR stabilniejszy): POTWIERDZONA

- LR=0.001 najstabilniejszy
- Większy LR może prowadzić do oscylacji

4. Hipoteza 4 (Manual problemy z konwergencją): POTWIERDZONA

- Ujemny R^2 dla Stock Market
- SGD bez momentum niewystarczający

8.3 Problemy napotkane i rozwiązania

1. Konwergencja Manual MLP dla regresji:

- Problem: Ujemny R^2 dla Stock Market
- Przyczyna: SGD bez momentum/Adam
- Rozwiązanie: Użycie Keras z zaawansowanymi optymalizatorami

2. CNN 1D wymiary:

- Problem: 5 cech to za mało dla wielu warstw Conv
- Rozwiązanie: Max 2 warstwy, warunkowy pooling

3. Fashion MNIST czas treningu:

- Problem: Kilka godzin na pełny grid search
- Rozwiązanie: Zwiększenie batch size, zmniejszenie grid

8.4 Główne wnioski

1. **Edukacyjna wartość ręcznej implementacji:** Głębokie zrozumienie backpropagation i optymalizacji
2. **Praktyczna przewaga frameworków:** Keras oferuje stabilność, szybkość i zaawansowane optymalizatory
3. **Dobór architektury:** CNN zdecydowanie lepszy dla obrazów (92.88% vs 74.03%)
4. **Znaczenie optymalizatora:** Adam/RMSprop przewyższają SGD dla złożonych problemów
5. **Preprocessing kluczowy:** Standaryzacja, balansowanie klas i selekcja cech mają duży wpływ

9 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: <https://www.tensorflow.org/>
5. Keras Documentation: <https://keras.io/>
6. UCI Machine Learning Repository: <https://archive.ics.uci.edu/ml/>
7. Xiao, H., Rasul, K., Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747

A Kod źródłowy

Pełny kod źródłowy projektu dostępny jest w repozytorium GitHub:

<https://github.com/Sornat11/NeuralNetwork>

Struktura projektu:

- `src/manual_mlp/` – Implementacja ręczna (`layers.py`, `model.py`, `metrics.py`)

- `src/models/` – Implementacja Keras (keras_mlp.py, keras_cnn.py, keras_lstm_regression.py)
- `data/` – Zbiory danych i skrypty preprocessingu
- `utils/` – Narzędzia pomocnicze (experiment_runner, visualization, results_exporter)
- `results/` – Wyniki eksperymentów (.xlsx) i wizualizacje (.png)
- `main.py` – Główny skrypt eksperymentów MLP
- `main_keras.py` – Eksperymenty Keras MLP
- `main_fashion_mnist.py` – Eksperymenty Fashion MNIST
- `main_regression_advanced.py` – CNN 1D i LSTM