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

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 niniejszego projektu jest kompleksowa analiza trzech wybranych problemów uczenia maszynowego, obejmujących zagadnienia klasyfikacji, regresji oraz analizy obrazów. Projekt ma na celu porównanie różnych architektur sieci neuronowych w szczególności perceptronu wielowarstwowego, konwolucyjnych sieci neuronowych (CNN) oraz rekurencyjnych sieci neuronowych (RNN, w tym LSTM) z uwzględnieniem ich efektywności, stabilności oraz wpływu parametrów modelu na jakość uzyskiwanych wyników. W projekcie zastosowano zarówno implementacje własne, jak i biblioteczne (TensorFlow), co umożliwia przeprowadzenie analizy porównawczej między rozwiązaniami od podstaw a architekturami wysoko zoptymalizowanymi.

1.2 Zakres prac

Projekt został zrealizowany zespołowo, z wyraźnym podziałem ról odpowiadających kluczowym etapom procesu badawczego. Podział obowiązków został zaprojektowany tak, aby uwzględnić zarówno kompetencje członków zespołu, jak i specyfikę poszczególnych komponentów projektu.

- **Jakub Sornat** odpowiadał za implementację autorskiej sieci neuronowej opartej na perceptronie wielowarstwowym. Jego zadania obejmowały zaprojektowanie struktury sieci, przygotowanie algorytmów propagacji wstecznej, implementację funkcji kosztu oraz mechanizmów uczenia. Implementacja została przygotowana bez użycia zewnętrznych frameworków, co pozwoliło na pełną analizę działania perceptronu na poziomie matematycznym i algorytmicznym.
- **Maciej Tajs** był odpowiedzialny za przygotowanie oraz przetwarzanie danych wykorzystywanych we wszystkich częściach projektu. Jego praca obejmowała wstępne czyszczenie danych, standaryzację i normalizację zmiennych, konstrukcję odpowiednich podziałów na zbiory treningowe, walidacyjne i testowe. Zastosowane procedury przygotowania danych zapewniały poprawność eksperymentów i minimalizację błędów wynikających z niejednorodności zbiorów.
- **Bartłomiej Sadza** odpowiadał za implementację modeli wykorzystujących biblioteki uczenia maszynowego, w szczególności TensorFlow/Keras. Obejmowało to przygotowanie perceptronu bibliotecznego, konwolucyjnych sieci neuronowych (CNN) oraz rekurencyjnych sieci neuronowych (LSTM). Jego zadania obejmowały również dobór odpowiednich hiperparametrów, przygotowanie pętli uczenia, integrację modeli z pipeline'em danych oraz realizację eksperymentów porównawczych.

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

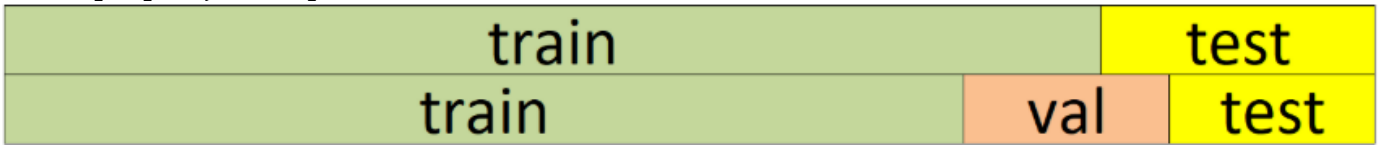
1. Keras osiągnie lepsze wyniki ze względu na optymalizacje niskopoziomowe i zaawansowane optymalizatory
2. Głębsze sieci (więcej warstw) będą lepsze dla złożonych wzorców
3. Mniejszy learning rate da stabilniejszy, ale wolniejszy trening
4. Implementacja ręczna będzie miała problemy z konwergencją dla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

4 Wykorzystane dane

Wykorzystane zostało 5 zbiorów danych – 2 dla problemu klasyfikacji, 2 dla problemu regresji oraz 1 dla problemu rozpoznawania obrazów. Wszystkie zbiory zostały poddane obróbce, tak by sieci neuronowe działały wydajnie, oraz by zwiększyć poprawność ich predykcji. W tym celu dokonano selekcji zmiennych objaśniających oraz ich transformacji takich jak:

- One-hot-encoding
- Target-encoding
- Grupowanie klas
- Transformacja logarytmiczna
- Standaryzacja

Dane zostały podzielone na podzbiory w dwóch wariantach – train + test oraz train + validation + test w proporcjach odpowiednio 80:20 oraz 70:15:15.



4.1 Adult Income Dataset (Klasyfikacja)

Dane pochodzą z roku 1994, dotyczą one zarobków w Stanach Zjednoczonych, celem wykorzystania sieci neuronowej była predykcja, czy dana kombinacja cech socjo-ekonomicznych oznacza zarobki mniejsze bądź równe 50 tysiącom, lub większe.

Zmienne:

- **age** *(int)* – wiek osoby (w latach).
- **workclass** *(cat)* – typ zatrudnienia, np. *Private, Self-emp-not-inc, Government, Without-pay, Never-worked*.
- **fnlwgt** *(int)* – tzw. *final weight*, liczba reprezentowanych osób w populacji przez ten rekord (waga próby).
- **education** *(cat)* – poziom wykształcenia, np. *Bachelors, HS-grad, Some-college, Masters*.
- **education_num** *(int/ordinal)* – liczba odpowiadająca poziomowi wykształcenia (np. 1–16).
- **marital_status** *(cat)* – status cywilny, np. *Married-civ-spouse, Divorced, Never-married, Widowed*.
- **occupation** *(cat)* – rodzaj wykonywanego zawodu, np. *Tech-support, Craft-repair, Sales, Exec-managerial*.
- **relationship** *(cat)* – relacja w gospodarstwie domowym, np. *Husband, Wife, Not-in-family, Own-child*.
- **race** *(cat)* – kategoria rasowa, np. *White, Black, Asian-Pac-Islander, Amer-Indian-Eskimo, Other*.
- **sex** *(cat)* – płeć (*Male* / *Female*).
- **capital_gain** *(int)* – zyski kapitałowe w ciągu roku (np. ze sprzedaży akcji, inwestycji).
- **capital_loss** *(int)* – straty kapitałowe w ciągu roku.
- **hours_per_week** *(int)* – liczba przepracowanych godzin tygodniowo.
- **native_country** *(cat)* – kraj pochodzenia, np. *United-States, Mexico, Philippines, Germany, Canada*.
- **income** *(cat, binarna)* – **zmienna objaśniana**, czy osoba zarabia więcej niż $50,000 rocznie (*<=50K* / *>50K*).

Podczas wstępnej pracy z danymi odrzucone zostały niektóre zmienne objaśniające ze względu na wysokie współczynniki korelacji między sobą lub niski współczynnik korelacji ze zmienną objaśnianą. Finalny zbiór danych zawiera zmienne: **age, education_num, occupation, sex, hours_per_week, marital_status, relationship, workclass, capital_net_log, income.** Zmienne **capital_gain** oraz **capital_loss** zostały połączone w zmienną **capital_net**, a następnie poddane transformacji logarytmicznej. Zbiór danych nie zawiera istotnego odsetka outlierów.

Braki danych nie stanowiły istotnej części całego zbioru, zostały usunięte. Ze względu na nierówny rozkład klasy income (3:1) zbiór został poddany undersaplingowi. W wyniku tych działań liczba obserwacji zbioru zmniejszyła się z początkowych 32.6 tysięcy do 15 tysięcy. Źródło danych: https://www.kaggle.com/datasets/jainaru/adult-income-census-dataset

4.2 Loan Approval Dataset (Klasyfikacja)

Zbiór danych o kredytach zawiera informacje o klientach ubiegających się o pożyczki oraz o tym, czy kredyt otrzymali.
Zmienne:

- **customer_id** *(string)* – unikalny identyfikator klienta w zbiorze danych.
- **age** *(int)* – wiek klienta w latach.
- **occupation_status** *(cat)* – status zawodowy klienta (np. *Employed*, *Self-employed*, *Unemployed*).
- **years_employed** *(float)* – liczba lat przepracowanych u obecnego lub ostatniego pracodawcy.
- **annual_income** *(float)* – roczny dochód brutto klienta w jednostkach walutowych.
- **credit_score** *(int)* – punktowa ocena zdolności kredytowej (im wyższa, tym mniejsze ryzyko kredytowe).
- **credit_history_years** *(float)* – długość historii kredytowej klienta w latach.
- **savings_assets** *(float)* – wartość posiadanych oszczędności i aktywów finansowych.
- **current_debt** *(float)* – aktualny poziom zadłużenia klienta.
- **defaults_on_file** *(int)* – liczba wcześniejszych przypadków niespłaconych kredytów lub zaległości finansowych.
- **delinquencies_last_2yrs** *(int)* – liczba opóźnień w spłacie zobowiązań w ciągu ostatnich dwóch lat.
- **derogatory_marks** *(int)* – liczba negatywnych wpisów w historii kredytowej (np. windykacje, bankructwa).
- **product_type** *(cat)* – rodzaj produktu finansowego (np. *Personal Loan*, *Mortgage*, *Auto Loan*).
- **loan_intent** *(cat)* – cel pożyczki (np. *Debt Consolidation*, *Education*, *Home Improvement*).
- **loan_amount** *(float)* – kwota wnioskowanego kredytu.
- **interest_rate** *(float)* – oprocentowanie pożyczki w ujęciu rocznym.
- **debt_to_income_ratio** *(float)* – stosunek całkowitego zadłużenia do rocznego dochodu (miara obciążenia finansowego).
- **loan_to_income_ratio** *(float)* – stosunek kwoty kredytu do rocznego dochodu klienta.
- **payment_to_income_ratio** *(float)* – stosunek miesięcznej raty kredytu do miesięcznego dochodu klienta.
- **loan_status** *(binary)* – zmienna objaśniana oznaczająca czy kredyt został przyznany.

Z tego zbioru danych na wstępnie odrzucone zostało customer_id, które oczywiście nie ma wpływu na spłatę kredytu, następnie do pracy z siecią neuronową wyselekcjonowano zestaw zmiennych objaśniających na podstawie macierzy korelacji. Finalny zbiór danych zawiera zmienne: age,annual_income, credit_score, defaults_on_file, delinquencies_last_2yrs, derogatory_marks, debt_to_income_ratio, loan_to_income_ratio, payment_to_income_ratio, loan_status – wszystkie te zmienne są zmiennymi liczbowymi.
Ze względu na dużą liczbę liczbę outlierów dla zmiennych annual_income, defaults_on_file, deliquencies_last_2yrs, derogatory_marks zbadano, czy rozkład klasy income jest podobny do tego z całego zbioru (Dla 56% obserwacji wartość income wynosiła 1).

```
age                    →   112 outlierów (  0.22%) | klasa 1:   91 (81.25%)
annual_income          →  2355 outlierów (  4.71%) | klasa 1: 1754 (74.48%)
credit_score           →   349 outlierów (  0.70%) | klasa 1:  169 (48.42%)
defaults_on_file       →  2674 outlierów (  5.35%) | klasa 1:    0 (  0.00%)
delinquencies_last_2yrs →  1761 outlierów (  3.52%) | klasa 1:    0 (  0.00%)
derogatory_marks       →  6393 outlierów ( 12.79%) | klasa 1: 1686 ( 26.37%)
debt_to_income_ratio   →   332 outlierów (  0.66%) | klasa 1:    0 (  0.00%)
loan_to_income_ratio   →     0 outlierów (  0.00%) | klasa 1:    0 (  0.00%)
payment_to_income_ratio →    0 outlierów (  0.00%) | klasa 1:    0 (  0.00%)
```

Rozkłady klasy income w przypadku outlierów znacznie różniły się od tego dla całego zbioru, outliery nie zostały więc usunięte by nie tracić potencjalnie istotnej wartości informacyjnej.
Finalna liczba rekordów dla tego zbioru to około 45 tysięcy rekordów.
Źródło danych: https://www.kaggle.com/datasets/parthpatel2130/realistic-loan-approval-dataset-us-and-canada

4.3 Stock Market Dataset (Regresja)

Zbiór danych przedstawia szereg czasowy notowań kryptowaluty w interwałach godzinowych. Dane dotyczą okresu od 15.10.2020 do 14.03.2024, składają się z około 30 tysięcy rekordów.
Zmienne:

- **datetime** – moment rozpoczęcia świecy (godzina).
- **open** – cena otwarcia w tej godzinie.
- **high** – najwyższa cena w tej godzinie.
- **low** – najniższa cena w tej godzinie.
- **close** – cena zamknięcia.
- **volume** – wolumen obrotu

Ten zbiór danych nie wymagał szczególnego przygotowywania – nie zawierał braków a dane były posortowane już na wstępie. Oczywiście sama sieć nie otrzymała zmiennej datetime jako inputu, przed trenowaniem modelu zmienne poza close - zmienną objaśnianą zostały poddane standaryzacji.

Podział na podzbiory został dokonany chronologicznie – w przypadku podziału na zbiór uczący i testowy pierwsze 80% obserwacji trafiło do zbioru train a ostatnie 20% do test, a dla podziału na train validation i test pierwsze 70% obserwacji do zbioru uczącego, następne 15% do walidacyjnego i ostatnie 15% do testowego.

4.4 Student Performance Dataset (Regresja)

Ten zbiór danych skupia się na czynnikach, które mogą wpływać na wynik studenta na egzaminie. Spośród pierwotnych 19 zmiennych objaśniających wyselekcjonowane zostały zmienne liczbowe: Previous_Scores, Hours_Studied, Attendance oraz zmienne porządkowe: Access_to_Resources, Parental_Involvement. Dokonano tego na podstawie macierzy korelacji.

Exam_Score	Access_to_Resources	Previous_Scores	Hours_Studied	Parental_Involvement	Attendance
67	3	73	23	1	84
61	2	59	19	1	64
74	2	91	24	2	98
71	2	98	29	1	89
70	2	65	19	2	92

Zbiór danych nie zawierał braków, wszystkie zmienne liczbowe oprócz zmiennej objaśnianej zostały poddane standaryzacji, natomiast zmienne porządkowe przekształcone w odpowiednie wartości liczbowe. Zbiór został podzielony w sposób losowy na odpowiednie podzbiory zgodnie z konwencją opisaną na początku rozdziału.

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)
- Klasy: 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 cech)
- Reshape do 28×28×1 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:

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

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:

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

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

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

5.2.2 Aktywacja ReLU

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

Gradient:

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

5.2.3 Aktywacja Softmax

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

5.2.4 Funkcje straty

Klasyfikacja (Cross-entropy):

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

Regresja (MSE):

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

5.3 Implementacja Keras

Implementacja Keras wykorzystuje wysokopoziomowe API:

Listing 1: Przykładowa architektura MLP w Keras

```
model = keras.Sequential()
model.add(layers.Input(shape=(n_inputs,)))

# Warstwy ukryte
for i in range(n_hidden_layers):
    model.add(layers.Dense(
        n_neurons,
        activation='relu',
        kernel_initializer='glorot_uniform'
    ))

# Warstwa wyjsciowa
if task_type == "classification":
    model.add(layers.Dense(n_outputs, activation='softmax'))
    model.compile(
        optimizer=optimizers.SGD(learning_rate=lr),
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
    )
else:
    model.add(layers.Dense(1, activation='linear'))
    model.compile(
        optimizer=optimizers.SGD(learning_rate=lr),
        loss='mse',
        metrics=['mae']
    )
```

5.4 Architektury zaawansowane

5.4.1 CNN dla Fashion MNIST

Listing 2: Architektura CNN dla obrazów

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

5.4.2 CNN 1D dla szeregów czasowych

Listing 3: CNN 1D dla Stock Market

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

5.4.3 LSTM dla danych sekwencyjnych

Listing 4: LSTM dla Stock Market

```
model = keras.Sequential([
    layers.Input(shape=(n_features, 1)),
    layers.LSTM(32, return_sequences=False),
    layers.Dense(32, activation='relu'),
    layers.Dense(1)
])
```

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 R2	-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 R2	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 R2	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 R2	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 R2	-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 R2	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 R2	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 R2	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
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 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 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 19: Podsumowanie najlepszych wyników dla wszystkich zbiorów danych

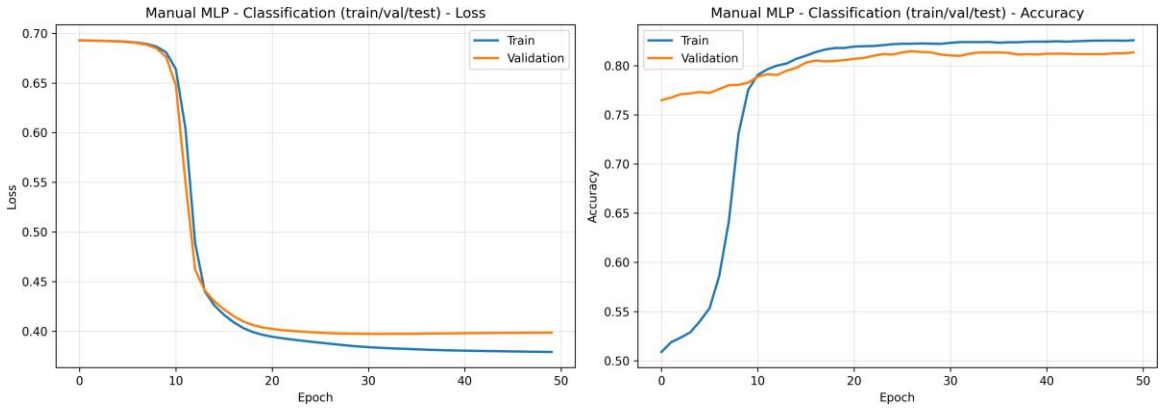
Główne obserwacje:

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

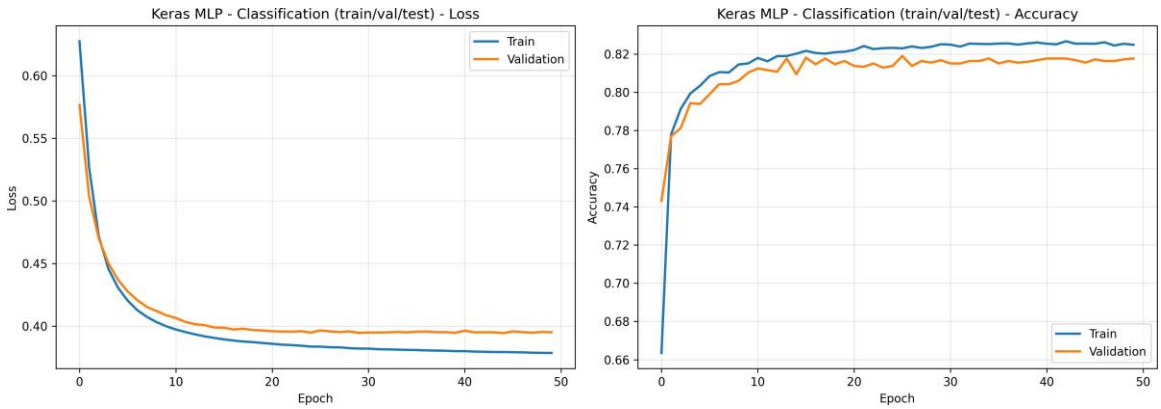
7 Wizualizacje

7.1 Learning Curves

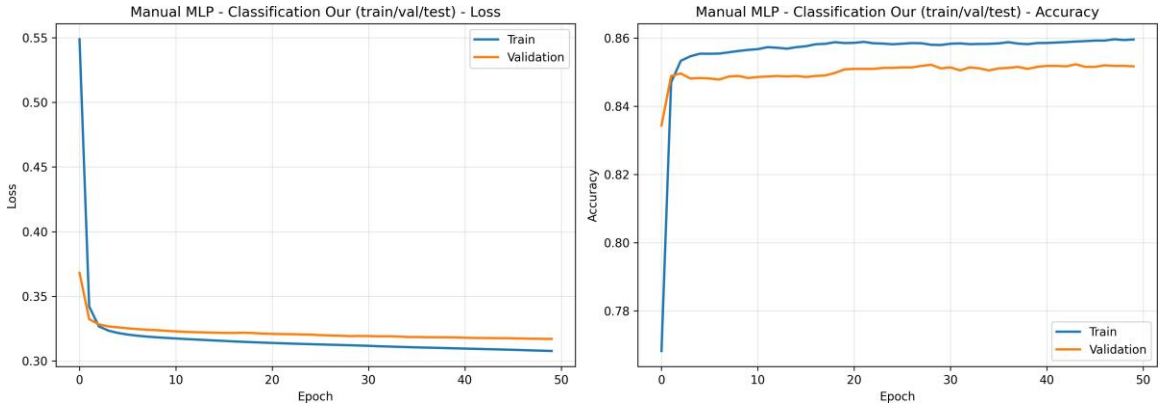
Learning curves pokazują postęp treningu modelu (loss i metryka główna przez epoki). Pozwalają zidentyfikować overfitting (gdy validation loss rośnie, a training loss maleje) oraz niedouczenie.



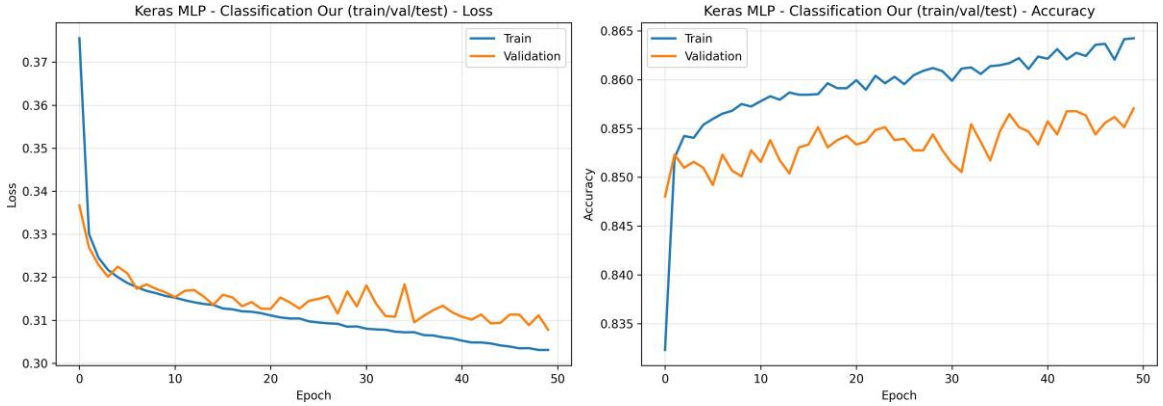
Rysunek 1: Learning curves dla Manual MLP - Adult Income (train/val/test split)



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



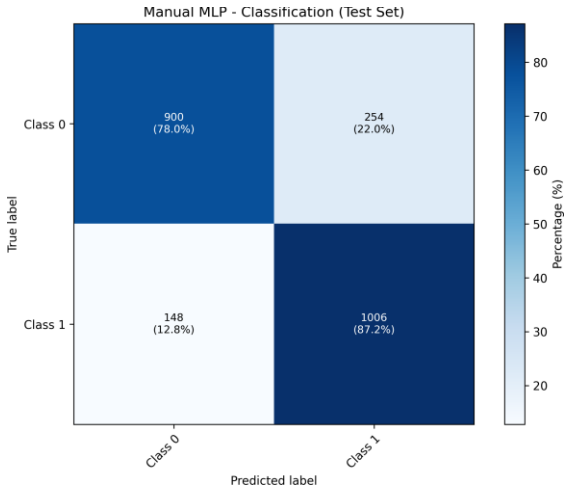
Rysunek 3: Learning curves dla Manual MLP - Loan Approval

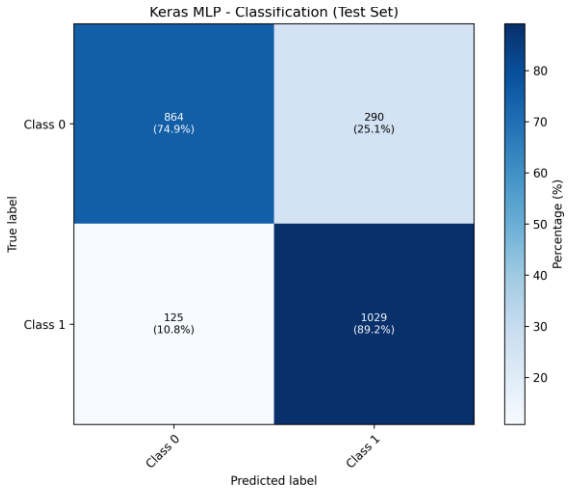


Rysunek 4: Learning curves dla Keras MLP - Loan Approval

7.2 Confusion Matrices

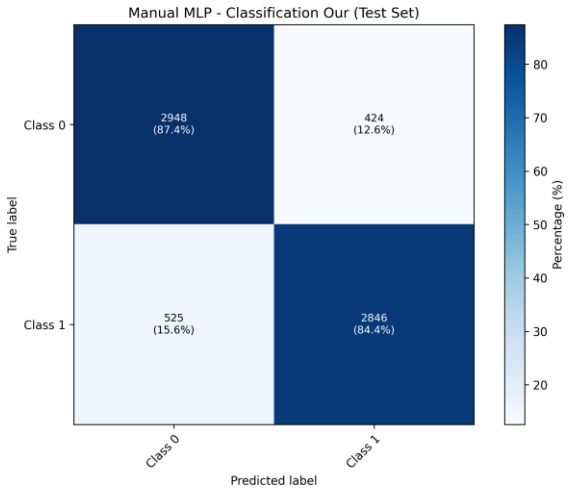
Macierze pomyłek pokazują szczegółowy rozkład predykcji dla każdej klasy.

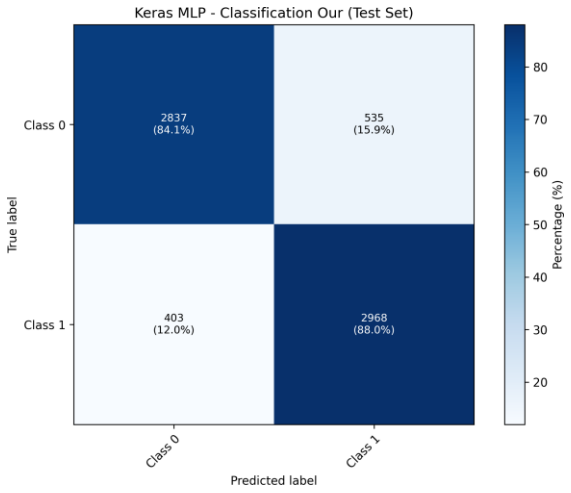




Manual MLP - Adult Income *Keras MLP - Adult Income*

Rysunek 5: Porównanie confusion matrices dla Adult Income Dataset



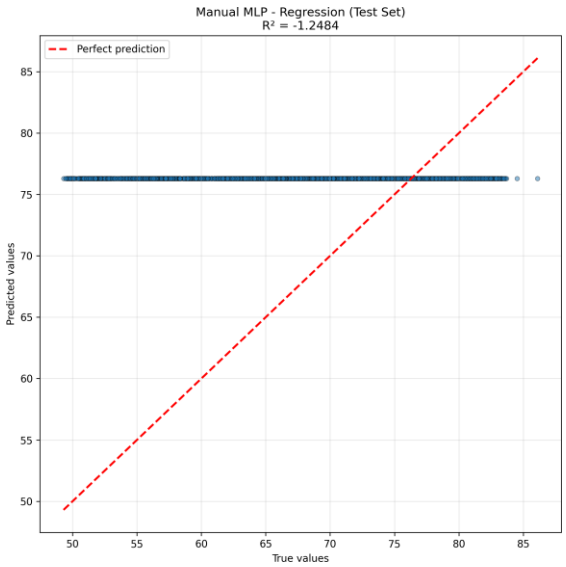


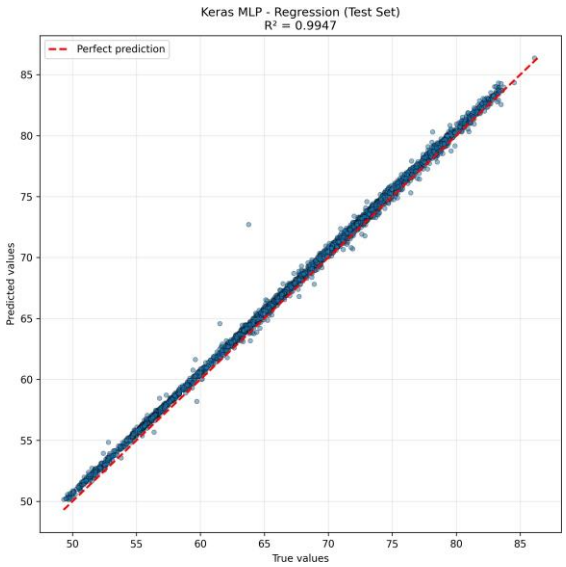
Manual MLP - Loan Approval *Keras MLP - Loan Approval*

Rysunek 6: Porównanie confusion matrices dla Loan Approval Dataset

7.3 Wykresy regresji (Predicted vs Actual)

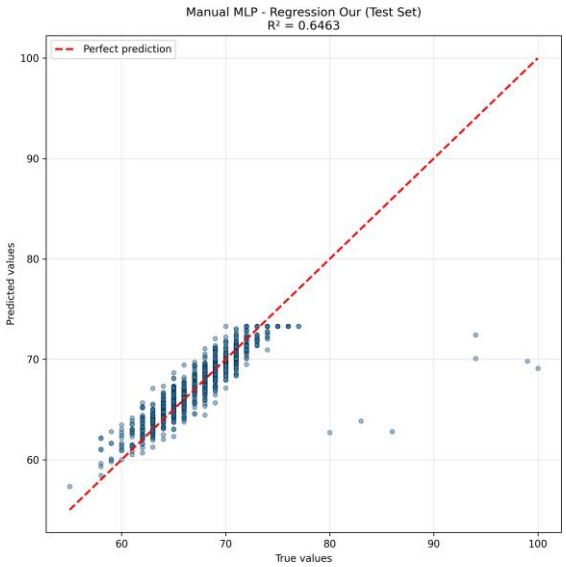
Scatter plots pokazują korelację między predykcjami a rzeczywistymi wartościami. Idealna predykcja to linia $y = x$.

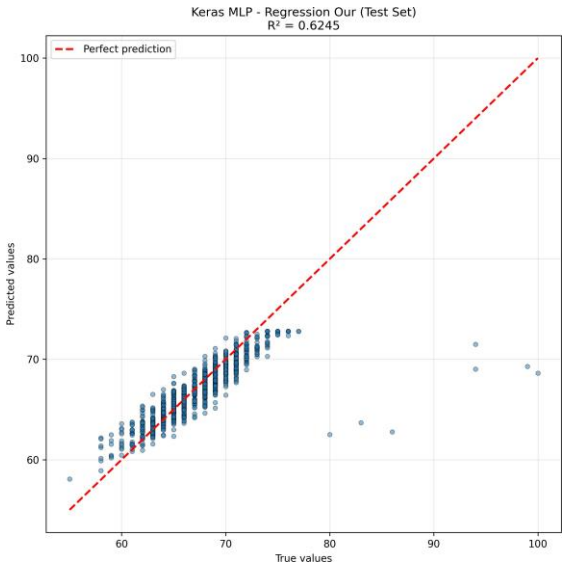




Manual MLP - Stock Market *Keras MLP - Stock Market*

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

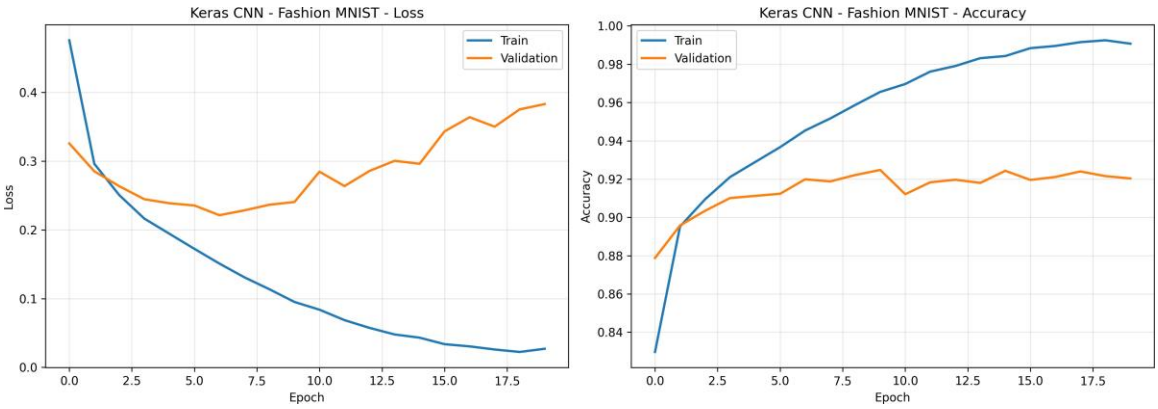




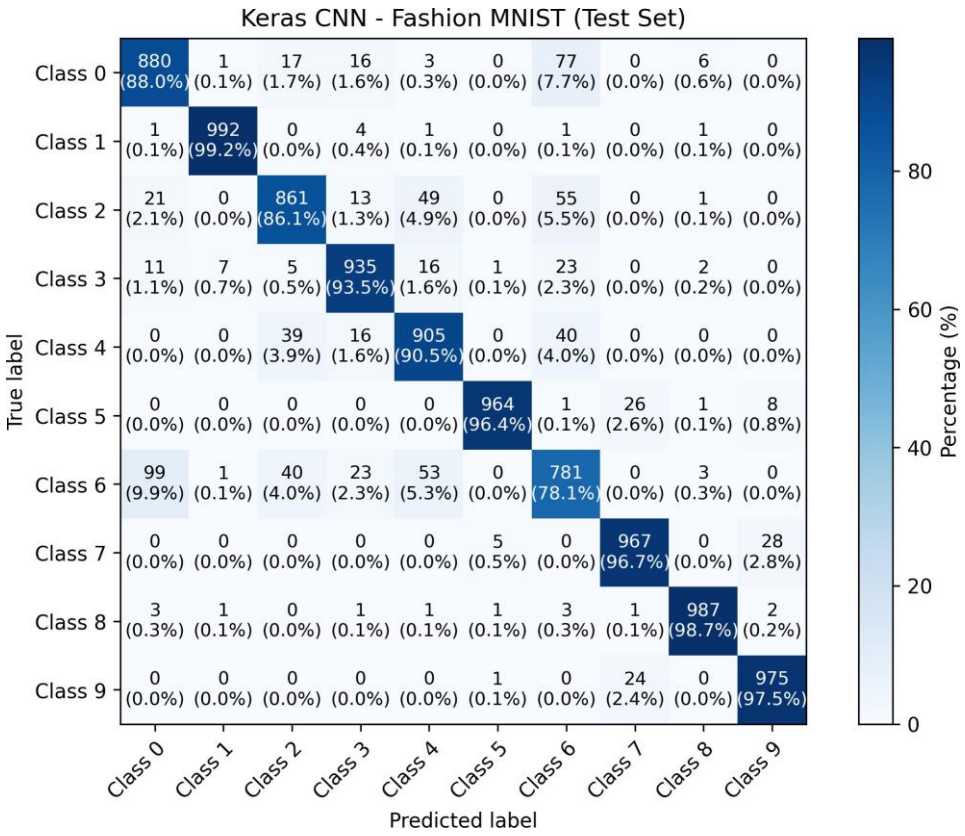
Manual MLP - Student Performance *Keras MLP - Student Performance*

Rysunek 8: Porównanie predykcji dla Student Performance Dataset

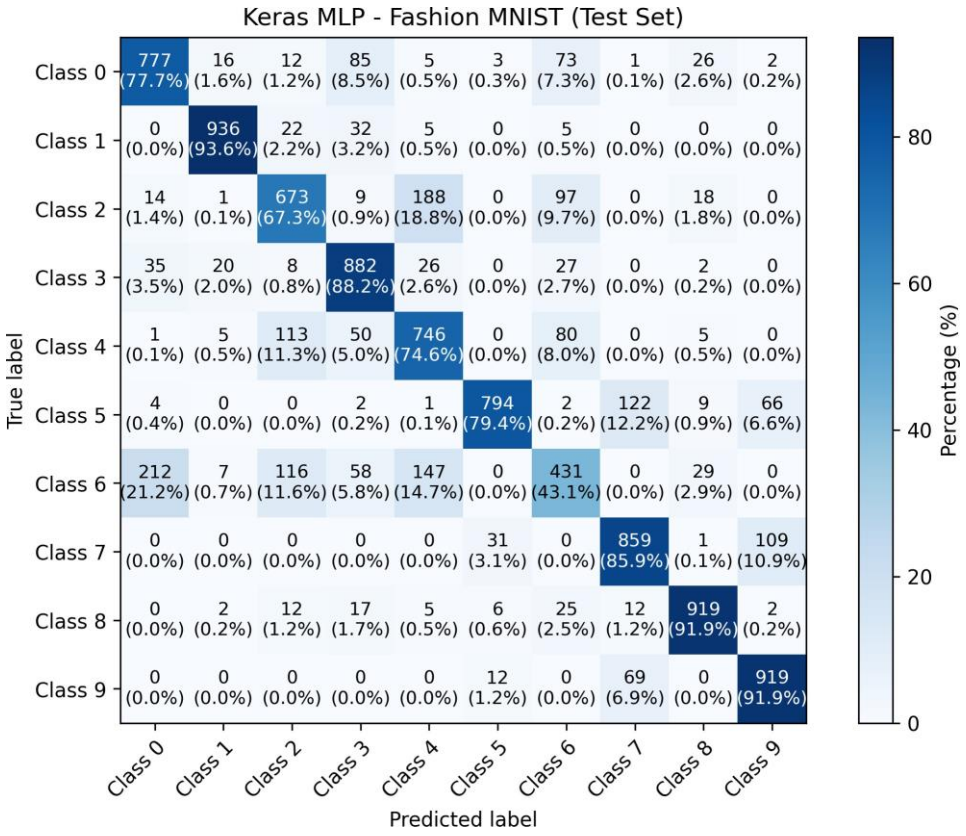
7.4 Fashion MNIST - Klasyfikacja obrazów



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

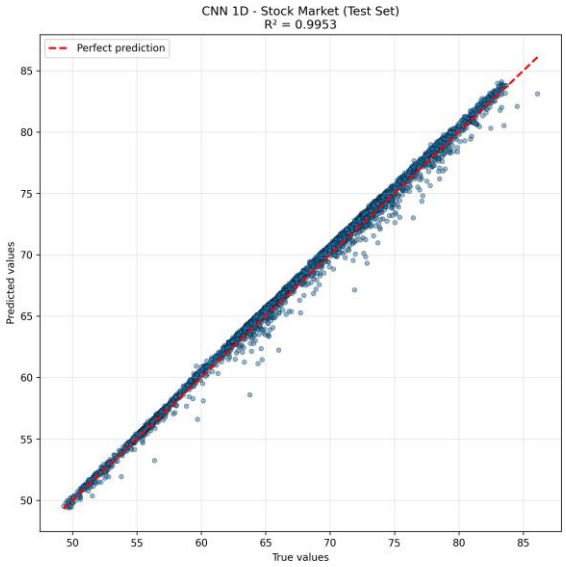


Rysunek 10: Confusion matrix dla CNN na Fashion MNIST. Model osiąga 92.88% accuracy, z najlepszą dokładnością dla klas ”Trouser” i ”Bag”, a najgorszą dla ”Shirt” (często mylony z ”T-shirt” i ”Coat”).

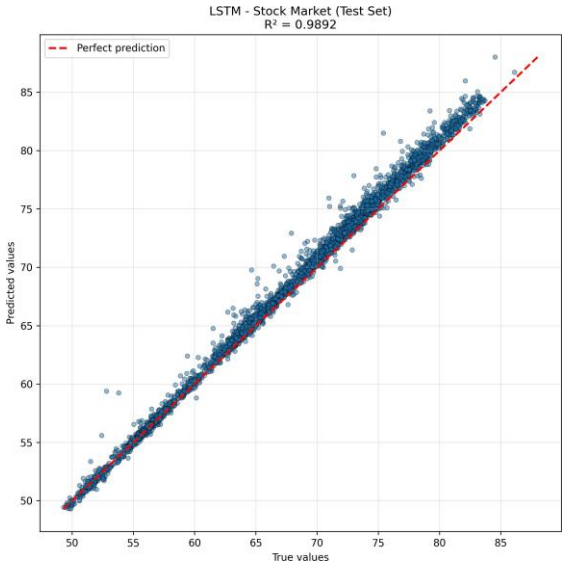


Rysunek 11: Confusion matrix dla MLP na Fashion MNIST (73.22% accuracy). Widoczne znacznie więcej błędów klasyfikacji w porównaniu do CNN.

7.5 Zaawansowane modele regresji



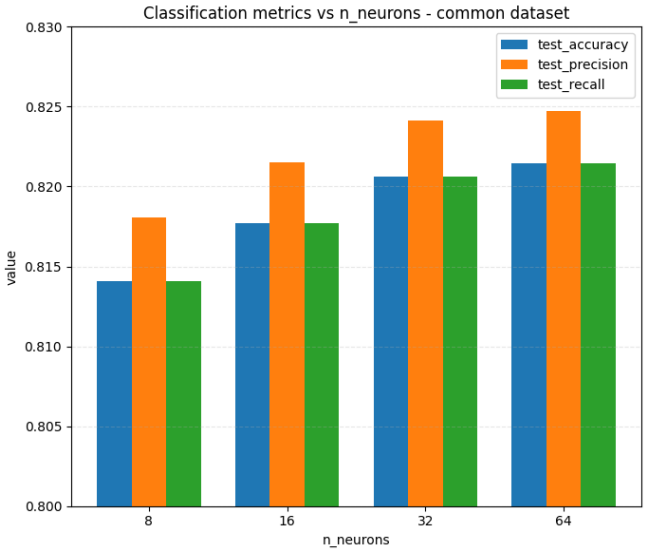
CNN 1D

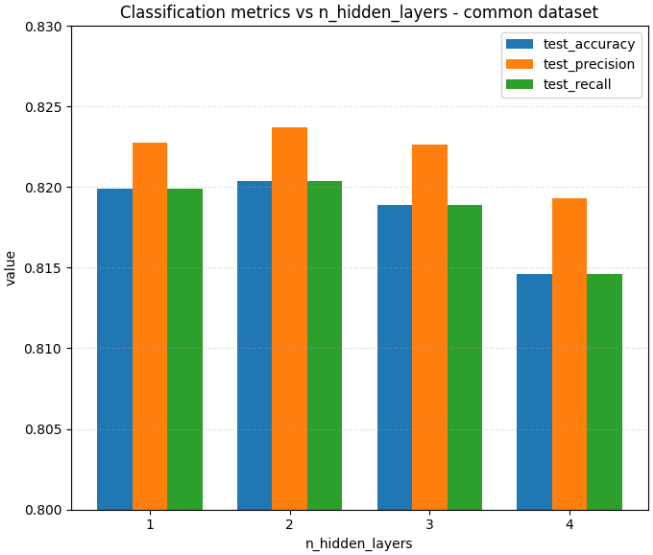


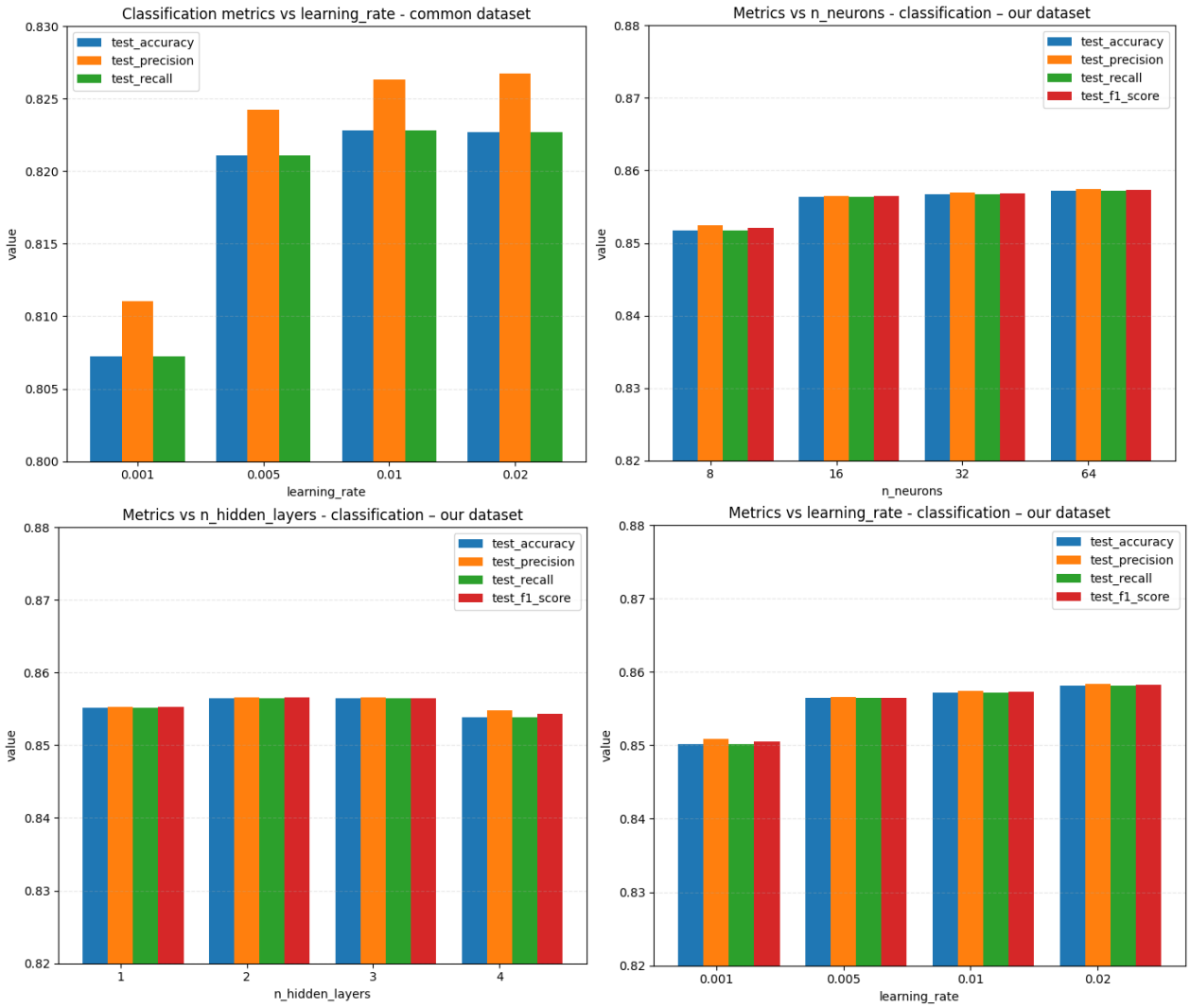
LSTM

Rysunek 12: Zaawansowane architektury dla Stock Market. Obie osiągają bardzo wysokie R2 (>0.997), ale Keras MLP wciąż najlepszy.

7.6 Skuteczność klasyfikacji w zależności od wartości parametrów







7.7 Skuteczność regresji szeregu czasowego w zależności od parametrów charakterystycznych dla LSTM oraz podziału zbioru danych – wspólny zbiór danych do regresji.

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 znacząco lepszy
- Stock Market: Keras R2=0.999 vs Manual R2=-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):** CZĘŚCIOWO POTWIERDZONA
   - Prawda dla regresji i obrazów
   - Fałsz dla prostej klasyfikacji tabelarycznej
2. **Hipoteza 2 (głębsze 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 R2 dla Stock Market
   - SGD bez momentum niewystarczający

8.3 Problemy napotkane i rozwiązania

1. **Konwergencja Manual MLP dla regresji:**
   - Problem: Ujemny R2 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