

Klasyfikacja liter polskiego języka migowego - porównanie metod uczenia maszynowego z wykorzystaniem punktów charakterystycznych dłoni

Jakub Wasilewski, Igor Włodarczyk

Politechnika Wrocławска, Wydział Informatyki i Telekomunikacji
Nazwa kursu: Uczenie Maszynowe
Prowadzący: mgr. Szymon Wojciechowski

Streszczenie W pracy przedstawiono eksperymentalne porównanie metod klasyfikacji statycznych gestów polskiego języka migowego (PJM). Głównym celem badania było porównanie skuteczności różnych klasyfikatorów w rozpoznawaniu liter alfabetu PJM. Zbadano cztery podejścia: Random Forest, regresję logistyczną, wielowarstwową sieć neuronową oraz konwolucyjną sieć neuronową. Jako reprezentację danych wykorzystano 78-wymiarowy wektor cech ekstrahowanych z punktów charakterystycznych dloni przy użyciu biblioteki MediaPipe. Przeprowadzono eksperymenty z i bez augmentacji danych na zbiorze 2549 próbek obejmujących 22 litery alfabetu PJM. Najlepsze wyniki uzyskał Random Forest, z dokładnością 98.87% przy użyciu augmentacji danych. Praca zawiera szczegółową analizę metryk, macierzy pomyłek oraz wpływu augmentacji na wydajność modeli.

Keywords: język migowy · klasyfikacja gestów · uczenie maszynowe · MediaPipe · augmentacja danych

1 Wstęp

Polski język migowy (PJM) stanowi naturalny środek komunikacji osób niesłyszących w Polsce. Rozpoznawanie gestów języka migowego za pomocą technik wizji komputerowej i uczenia maszynowego może znacząco ułatwić komunikację między osobami słyszącymi a niesłyszczącymi, stanowiąc podstawę systemów tłumaczących w czasie rzeczywistym.

Niniejsza praca koncentrowała się na klasyfikacji statycznych gestów alfabetu PJM z wykorzystaniem cech ekstrahowanych z punktów charakterystycznych dloni. W przeciwieństwie do podejść wykorzystujących bezpośrednio obrazy RGB, użycie punktów charakterystycznych znacząco redukowało wymiarowość problemu ($224 \times 224 \times 3 = 150528$ do 78 cech), co pozwalało na bardziej efektywne trenowanie klasyfikatorów przy ograniczonej liczbie danych treningowych.

1.1 Cel pracy

Głównym celem badania było eksperymentalne porównanie skuteczności różnych klasyfikatorów uczenia maszynowego w zadaniu klasyfikacji liter PJM. Praca miała odpowiedzieć na pytanie, które z metod najlepiej radziły sobie z tym zadaniem, oraz jaki wpływ miała augmentacja danych na wydajność poszczególnych modeli. Dodatkowo weryfikowano, czy proste klasyfikatory mogły konkurować z głębszymi sieciami neuronowymi przy wykorzystaniu odpowiednio zaprojektowanych cech.

1.2 Dostęp do kodu źródłowego

Kompletny kod źródłowy projektu, skrypty eksperymentalne oraz szczegółowe wyniki dostępne są w publicznym repozytorium GitHub: <https://github.com/wasilewskij/polish-sign-language>. Ze względu na ograniczenia rozmiaru repozytorium GitHub, surowe zdjęcia nie zostały dołączone do repozytorium.

Bardziej szczegółowy przegląd wyników, zawierający rozszerzone metryki per-class oraz dodatkowe macierze pomyłek, dostępny jest pod adresem: https://github.com/wasilewskij/polish-sign-language/blob/master/results_cv/report_cv_comparison.md.

2 Przegląd literatury

Rozpoznawanie języka migowego jest aktywnie badanym obszarem na styku wizji komputerowej i uczenia maszynowego [1,2]. Większość prac koncentruje się na amerykańskim języku migowym (ASL), podczas gdy badania nad polskim językiem migowym są stosunkowo niewielkie.

Współczesne podejścia do rozpoznawania gestów można podzielić na dwie główne kategorie: metody oparte na bezpośredniej analizie obrazów oraz metody wykorzystujące punkty charakterystyczne (ang. *hand landmarks*).

W kontekście klasyfikacji języków migowych, Ameen i Vadera [4] zastosowali CNN do klasyfikacji gestów ASL z obrazów głębi i RGB, osiągając dokładność około 98-99%. Oguntimilehin i Balogun [5] przedstawili system rozpoznawania Nigerian Sign Language w czasie rzeczywistym wykorzystując architekturę CNN, osiągając dokładność testową około 93%. Kołodziej i in. [3] opisują system wykorzystujący architekturę DenseNet do rozpoznawania 24 znaków alfabetu ASL, osiągając dokładność bliską 100% dla użytkowników widzianych podczas treningu oraz 83-85% dla nowych użytkowników.

W kontekście polskiego języka migowego, Piskozub i Strumiłło [6] proponują rękawice z czujnikami do rozpoznawania liter alfabetu PJM. Filipowska i in. [7] prezentują system rozpoznawania gestów PJM na podstawie sygnałów EMG, osiągając wysoką dokładność klasyfikacji. Brak jest jednak otwartoźródłowych projektów wykorzystujących standardowe kamery RGB do klasyfikacji statycznych liter PJM.

Dla znaków dynamicznych, obserwuje się w literaturze częstszy wybór konwolucyjnych sieci neuronowych w zadaniu klasyfikacji. Natomiast podejścia oparte

na punktach charakterystycznych dloni, ze względu na ich znaczącą redukcję wymiarowości danych oraz łatwiejszą interpretację przez model, wydają się być rozsądny wyborem dla znaków statycznych.

3 Zbiór danych

3.1 Metodologia zbierania danych

Zbiór danych został ręcznie zebrany przy użyciu kamery internetowej i skryptu `collect_dataset.py`. Proces zbierania obejmował:

1. Zapisywanie obrazów RGB o rozdzielcości 224×224 pikseli
2. Automatyczną detekcję dloni za pomocą MediaPipe Hand Landmarker
3. Odrzucenie próbek, w których nie wykryto dloni

3.2 Charakterystyka zbioru

- **Liczba próbek:** 2549 (po filtracji MediaPipe)
- **Liczba klas:** 22 litery PJM (A-Z bez J, Q, V, X)
- **Rozkład klas:** quasi-zbalansowany, 101-119 próbek na klasę
- **Format:** Obrazy RGB $224 \times 224 \times 3$
- **Pominięte próbki:** 294 (10.3%), MediaPipe nie wykrył dloni

Rozkład klas przedstawia Tabela 1. Zbiór jest stosunkowo zbalansowany, co minimalizuje potrzebę stosowania technik ważenia klas.

Tablica 1. Rozkład próbek w zbiorze danych

Klasa	A	B	C	D	E	F	G
Liczba	113	119	119	112	109	113	112
<hr/>							
Klasa	H	I	K	L	M	N	O
Liczba	117	116	119	119	119	117	119
<hr/>							
Klasa	P	R	S	T	U	W	Y
Liczba	115	119	101	115	119	119	119
<hr/>							
Klasa	Z						
Liczba	119						

3.3 Ekstrakcja cech - punkty charakterystyczne dloni

Dla każdego obrazu RGB, MediaPipe Hand Landmarker ekstrahuje 21 punktów charakterystycznych dloni, każdy opisany przez współrzędne (x, y, z) . Aby zwiększyć odporność na translacje i skalowanie, zastosowano normalizację względem nadgarstka:

Listing 1.1. Ekstrakcja i normalizacja punktów charakterystycznych

```
def extract_landmarks(image):
    results = hands.process(cv2.cvtColor(image,
                                         cv2.COLOR_BGR2RGB))
    if not results.multi_hand_landmarks:
        return None

    hand = results.multi_hand_landmarks[0]
    landmarks = [(lm.x, lm.y, lm.z) for lm in hand.landmark]

    wrist = np.array(landmarks[0])
    landmarks_norm = [np.array(lm) - wrist
                      for lm in landmarks]

    return np.array(landmarks_norm).flatten()
```

Ostatecznie, każda próbka jest reprezentowana przez wektor 78-wymiarowy ($21 \times 3 = 63$ współrzędne + 15 relacji geometrycznych między punktami).

4 Augmentacja danych

Aby zwiększyć rozmiar zbioru treningowego i poprawić generalizację modeli, zastosowano augmentację danych. Ze względu na statyczny charakter gestów PJM oraz semantyczne znaczenie orientacji dloni, wybrano następujące transformacje:

- **Obrót:** $\pm 15^\circ$
- **Przesunięcie:** $\pm 10\%$ szerokości i wysokości
- **Skalowanie:** $90 - 110\%$ rozmiaru
- **Jasność:** $\pm 20\%$
- **Kontrast:** $\pm 20\%$

4.1 Proces augmentacji

Augmentacja została przeprowadzona w dwóch etapach, zgodnie z zasadą separacji generowania danych od treningu modeli:

1. **Wstępne generowanie augmentowanych obrazów:** Napisano skrypt `generate_augmented_data.py`, który dla każdego oryginalnego obrazu generuje 5 augmentowanych wersji przy użyciu biblioteki Albumentations. Wygenerowane obrazy zapisywane są w katalogu `data/augmented/` z nazwami odpowiadającymi oryginalnym plikom (np. `A1_aug0.jpg`, `A1_aug1.jpg`, ...).
2. **Dobieranie danych podczas treningu:** W czasie walidacji krzyżowej, dla każdej próbki należącej do zbioru treningowego danego foldu, ładowane są zarówno oryginalne, jak i odpowiadające im augmentowane obrazy. Z wszystkich tych obrazów ekstrahowane są punkty charakterystyczne, które następnie trafiają do zbioru treningowego. Dzięki temu augmentacja jest wykonywana raz, a następnie wielokrotnie wykorzystywana.

Kluczowe jest, że zbiór walidacyjny zawsze składa się wyłącznie z oryginalnych próbek, aby zapewnić uczciwe porównanie modeli i uniknąć przecieków informacji między zbiorami.

5 Metodologia eksperymentów

5.1 Stratified K-Fold Cross-Validation

Wszystkie eksperymenty przeprowadzono z użyciem 5-krotnej stratyfikowanej walidacji krzyżowej. Stratyfikacja zapewnia, że każda klasa ma proporcjonalną reprezentację w każdym foldzie, co jest szczególnie istotne przy małych zbiorach danych.

Proces validacji:

1. Podział danych na 5 foldów ze stratyfikacją
2. Dla każdego foldu:
 - 4 foldy jako zbiór treningowy
 - 1 fold jako zbiór walidacyjny
 - Trening modelu na zbiorze treningowym
 - Ewaluacja na zbiorze walidacyjnym
3. Agregacja wyników: średnia i odchylenie standardowe metryk

5.2 Modele klasyfikacyjne

Zbadano cztery różne podejścia:

Random Forest Ensemble 100 drzew decyzyjnych z maksymalną głębokością 20. Hiperparametry:

```
RandomForestClassifier(
    n_estimators=100,
    max_depth=20,
    random_state=42
)
```

Logistic Regression Wieloklasowa regresja logistyczna z regularizacją L2:

```
LogisticRegression(
    max_iter=1000,
    random_state=42,
    multi_class='multinomial'
)
```

Keras MLP Wielowarstwowa sieć neuronowa z następującą architekturą:

```
model = Sequential([
    Dense(256, activation='relu', input_dim=78),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dense(22, activation='softmax')
])
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy'
)
```

Trenowanie przez 50 epok z early stopping (patience=10).

CNN Prosta konwolucyjna sieć neuronowa operująca bezpośrednio na obrazach RGB o wymiarach $224 \times 224 \times 3$:

```
model = Sequential([
    Conv2D(32, (3,3), activation='relu',
           input_shape=(224,224,3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(22, activation='softmax')
])
```

Ze względu na błąd implementacyjny, CNN nie został włączony do porównania z augmentacją danych.

5.3 Metryki ewaluacji

Do oceny modeli wykorzystano następujące metryki:

- **Accuracy:** Procent poprawnie sklasyfikowanych próbek
- **Balanced Accuracy:** Średnia recall per-class (uwzględnia niebalansowanie)
- **F1-macro:** Średnia harmoniczna precision i recall, uśredniona po klasach
- **F1-weighted:** F1 ważone liczebnością klas
- **Precision/Recall per-class:** Analiza błędów dla każdej litery
- **Macierz pomyłek:** Wizualizacja błędnych klasyfikacji

6 Wyniki eksperymentów

6.1 Eksperymenty bez augmentacji

Tabela 2 przedstawia wyniki dla modeli trenowanych na oryginalnym zbiorze danych.

Tablica 2. Wyniki eksperymentów bez augmentacji (5-fold CV)

Model	Accuracy	Balanced Acc	F1-macro	F1-weighted
Random Forest	0.9847 ± 0.0038	0.9848 ± 0.0039	0.9848 ± 0.0039	0.9847 ± 0.0038
LogisticRegression	0.9823 ± 0.0061	0.9823 ± 0.0062	0.9823 ± 0.0061	0.9823 ± 0.0061
Keras MLP	0.9839 ± 0.0064	0.9830 ± 0.0072	0.9839 ± 0.0074	0.9839 ± 0.0065
CNN	0.1687 ± 0.2209	0.1685 ± 0.2213	0.1323 ± 0.2365	0.1323 ± 0.2364

Wszystkie trzy modele oparte na landmarkach osiągnęły bardzo wysoką dokładność ($> 98\%$), co potwierdza skuteczność reprezentacji opartej na punktach charakterystycznych dloni. Random Forest wykazał najwyższą średnią accuracy oraz najmniejsze odchylenie standardowe, co świadczy o jego stabilności.

Natomiast model CNN osiągnął znaczco słabsze wyniki ($\sim 17\%$ accuracy), co było prawdopodobnie spowodowane zbyt prostą architekturą sieci oraz niewystarczającą ilością danych treningowych. Wysokie odchylenie standardowe ($\sigma = 0.22$) wskazuje na dużą niestabilność wyników między foldami.

6.2 Eksperymenty z augmentacją

Tabela 3 przedstawia wyniki dla modeli trenowanych z augmentacją danych.

Tablica 3. Wyniki eksperymentów z augmentacją danych (5-fold CV)

Model	Accuracy	Balanced Acc	F1-macro	F1-weighted
Random Forest	0.9887 ± 0.0036	0.9877 ± 0.0056	0.9875 ± 0.0052	0.9887 ± 0.0036
LogisticRegression	0.9847 ± 0.0061	0.9832 ± 0.0082	0.9823 ± 0.0086	0.9847 ± 0.0061
Keras MLP	0.9851 ± 0.0042	0.9845 ± 0.0051	0.9824 ± 0.0053	0.9851 ± 0.0042

6.3 Porównanie wpływu augmentacji

Tabela 4 przedstawia bezpośrednie porównanie wyników z i bez augmentacji.

Augmentacja danych przyniosła nieznaczną poprawę dla wszystkich modeli, przy czym największy wzrost odnotowano dla Random Forest (+0.40%). Tak niewielki wpływ augmentacji wynika z już bardzo wysokiej bazowej dokładności modeli.

Tablica 4. Wpływ augmentacji danych na wydajność modeli

Model	Baseline Acc	Augmented Acc	Δ Acc (%)
Random Forest	0.9847	0.9887	+0.40
LogisticRegression	0.9823	0.9847	+0.24
Keras MLP	0.9839	0.9851	+0.12

6.4 Analiza per-class

Tabela 5 przedstawia szczegółową analizę dla wybranych klas (Random Forest z augmentacją).

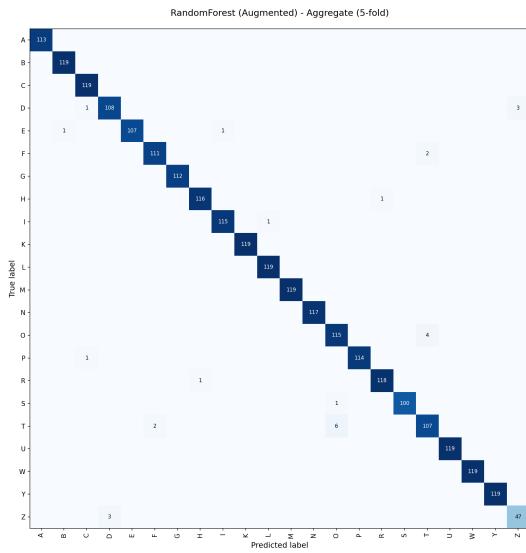
Tablica 5. Metryki per-class dla Random Forest (augmented)

Klasa	Support	Precision	Recall	F1
A	113	1.0000	1.0000	1.0000
B	119	0.9920	1.0000	0.9959
D	112	0.9819	0.9822	0.9816
F	113	0.9833	0.9911	0.9870
O	119	0.9524	0.9412	0.9456
S	101	1.0000	1.0000	1.0000
T	115	0.9833	0.9739	0.9778

Warto zauważyć, że proste klasyfikatory, dysponując wyekstrahowanymi punktami charakterystycznymi dloni, radzą sobie doskonale. Wiele klas osiągnęło perfekcyjny wynik accuracy równy 1.0 (np. A, S dla Logistic Regression). Natomiast litera O, posiadająca punkty charakterystyczne zbliżone do litery T, okazuje się być klasą trudną do nauczenia.

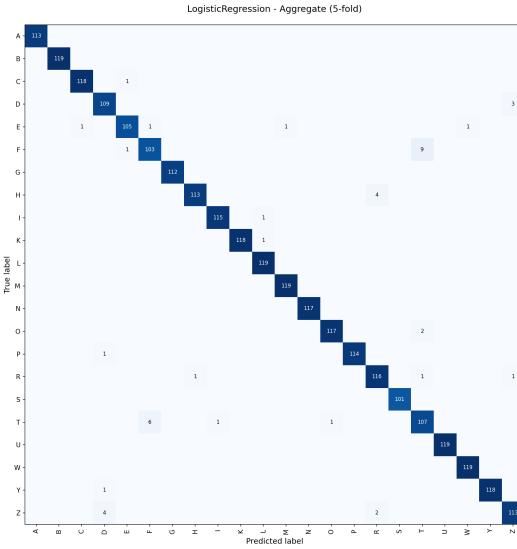
6.5 Macierze pomyłek

Rysunek 1 przedstawia zagregowaną macierz pomyłek dla Random Forest z augmentacją. Większość próbek została sklasyfikowana poprawnie (koncentracja na diagonali).



Rysunek 1. Macierz pomyłek dla Random Forest z augmentacją (agregowana z 5 foldów)

Analiza macierzy pomyłek ujawnia, że największe trudności sprawiają litery o podobnej konfiguracji punktów charakterystycznych. Rysunek 2 przedstawia szczegółową macierz dla Logistic Regression, gdzie widoczne są błędne klasyfikacje między podobnymi literami.



charakterystycznych jako cech. Sugeruje to, że dla tego typu reprezentacji złożone modele nie są konieczne.

5. **Problematyczne pary liter:** Analiza błędów klasyfikacji ujawniła, że największe trudności sprawiają litery o podobnej konfiguracji punktów (T-F, T-O, D-Z), co wskazuje na ograniczenia wynikające z podobieństwa gestów.
6. **Perfekcyjna klasyfikacja dla niektórych klas:** Proste klasyfikatory operujące na punktach charakterystycznych osiągnęły accuracy równą 1.0 dla wybranych liter (np. A, S w Random Forest), co potwierdza skuteczność tej reprezentacji dla dobrze wyodrębnionych gestów.
7. **Słabe wyniki CNN:** Prosty model CNN osiągnął znacznie gorsze wyniki ($\sim 17\%$ accuracy) w porównaniu do klasyfikatorów opartych na punktach charakterystycznych. Prawdopodobnie przydałaby się bardziej zaawansowana architektura sieci oraz znacznie większy zbiór danych treningowych. Niestety źle przeprowadzony eksperyment augmentacji nie pozwolił przetestować modelu CNN na większej ilości danych. Jest to potencjalny kierunek dalszych badań, w szczególności biorąc pod uwagę, że jest to popularny wybór w literaturze dla gestów dynamicznych. Istnieje również prawdopodobieństwo, że dla gestów statystycznych ten model jest nieopłacalny w porównaniu do klasyfikatorów, przy użyciu których można doskonale wytrenować proste modele, na niewielkim zbiorze danych.

7.2 Ograniczenia badania

- **Statyczne gesty:** Badanie obejmowało tylko statyczne litery alfabetu, bez gestów dynamicznych.
- **Pojedynczy wykonawca:** Dane zebrano od jednej osoby, co mogło ograniczać generalizację.
- **Kontrolowane warunki:** Zdjęcia wykonano w podobnych warunkach oświetleniowych i z jednolitym tłem.
- **CNN bez augmentacji:** Błąd implementacyjny uniemożliwił pełne porównanie CNN z augmentacją danych.
- **Mały zbiór danych:** Około 2500 próbek może być niewystarczające dla głębszych sieci neuronowych.

7.3 Przyszłe kierunki badań

1. **Zaawansowane architektury CNN:** Wykorzystanie głębszych sieci (ResNet, EfficientNet) oraz transfer learning z pre-trenowanych modeli może znacząco poprawić wyniki dla obrazów RGB.
2. **Większy zbiór danych:** Rozszerzenie zbioru o więcej wykonawców i różnorodne warunki nagrywania jest kluczowe dla poprawy generalizacji, oraz bardziej szczegółowych badań nad modelami CNN.
3. Implementacja rozpoznawania gestów dynamicznych (słowa, zdania)
4. Optymalizacja dla zastosowań czasu rzeczywistego na urządzeniach mobilnych
5. Integracja z systemem tłumaczenia kontekstowego PJM

8 Podsumowanie

W pracy przeprowadzono szczegółowe badanie eksperymentalne nad klasyfikacją liter polskiego języka migowego. Wykazano, że reprezentacja oparta na punktach charakterystycznych dloni w połączeniu z klasycznymi metodami uczenia maszynowego (w szczególności Random Forest) pozwalała osiągnąć bardzo wysoką dokładność klasyfikacji (98.87%). Augmentacja danych przyczyniła się do dalszej poprawy wyników, choć jej wpływ był stosunkowo niewielki ze względu na już wysoką bazową dokładność modeli.

Rezultaty badań mogły stanowić podstawę dla praktycznych systemów wspomagających komunikację osób niesłyszących, przy czym konieczne byłoby dalsze badanie w kierunku rozszerzenia na gesty dynamiczne oraz zwiększenia różnorodności zbioru danych.

Literatura

1. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
2. A. Krizhevsky, I. Sutskever, G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pp. 1097–1105, 2012.
3. M. Kołodziej, E. Szypuła, A. Majkowski, R. Rak, “Using deep learning to recognize the sign alphabet,” *Przegląd Elektrotechniczny*, vol. 98, no. 6, 2022, doi:10.15199/48.2022.06.06.
4. S. Ameen, S. Vadera, “A convolutional neural network to classify American Sign Language fingerspelling from depth and colour images,” *Expert Systems*, vol. 34, no. 3, e12197, 2017.
5. A. Oguntimilehin, K. Balogun, “Real-Time Sign Language Fingerspelling Recognition using Convolutional Neural Network,” *International Arab Journal of Information Technology*, vol. 21, no. 1, pp. 158–165, 2024.
6. J. Piskozub, P. Strumiłło, “Data Glove for the Recognition of the Letters of the Polish Sign Language Alphabet,” in *The Latest Developments and Challenges in Biomedical Engineering*, Proc. 23rd Polish Conference on Biocybernetics and Biomedical Engineering (PCBBE 2023), Springer, 2023, pp. 351–362.
7. A. Filipowska, W. Filipowski, J. Mieszczański, K. Bryzik, M. Henkel, E. Skwarek, P. Raif, S. Sieciński, R. J. Doniec, B. Mika *et al.*, “Pattern Recognition in the Processing of Electromyographic Signals for Selected Expressions of Polish Sign Language,” *Sensors*, vol. 24, no. 18, 6710, 2024.

Użycie narzędzi AI

W trakcie realizacji projektu wykorzystano narzędzia AI w ograniczonym zakresie, co przedstawiono w Tabeli 6.

Tablica 6. Użycie narzędzi AI w projekcie

Narzędzie AI	Zastosowanie
ChatGPT	Stworzenie skryptu <code>collect_dataset.py</code> według zadanej specyfikacji
ChatGPT	Pomoc przy pisaniu skryptu generującego raport <code>results_cv/report_cv_comparison.md</code>
ChatGPT	Wyjaśnianie koncepcji i pomoc teoretyczna w zakresie uczenia maszynowego