WYDZIAŁ
MATEMATYKI
I FIZYKI STOSOWANEJ
POLITECHNIKI RZESZOWSKIEJ

Rozpoznawanie Pozycji Ciała na Podstawie Map Nacisku Stopy przy Użyciu Sieci Neuronowych

Marcin Przybylski

1 maja 2025

Spis treści

1 Wstęp

Niniejszy dokument stanowi sprawozdanie z projektu badawczego dotyczącego zastosowania sieci neuronowych do klasyfikacji pozycji ciała człowieka (stanie, skłon, przysiad) na podstawie danych z map rozkładu nacisku stóp. Analiza opierała się na danych wejściowych w postaci obrazów o wymiarach 18x34 pikseli, reprezentujących rozkład nacisku na podłoże, co stanowi interesujące wyzwanie w dziedzinie rozpoznawania wzorców.

Głównym celem projektu było nie tylko zbudowanie działających klasyfikatorów, ale przede wszystkim porównanie efektywności dwóch fundamentalnie różnych architektur sieci neuronowych w tym specyficznym zadaniu klasyfikacyjnym:

- W pełni połączonej sieci neuronowej (Fully Connected Network - FCN), która traktuje dane wejściowe jako płaski wektor cech, ignorując ich przestrzenną strukturę.
- Konwolucyjnej sieci neuronowej (Convolutional Neural Network - CNN), która jest specjalnie zaprojektowana do wykorzystywania lokalnych zależności przestrzennych w danych obrazowych poprzez zastosowanie filtrów konwolucyjnych i operacji poolingu.

Dodatkowo, projekt obejmował eksplorację wpływu popularnych technik optymalizacyjnych na wydajność i zdolność generalizacji modelu CNN, jako że to właśnie ta architektura jest standardem w przetwarzaniu obrazów. Zbadano zastosowanie normalizacji wsadowej (Batch Normalization) połączonej ze zwiększeniem liczby filtrów (co potencjalnie pozwala na naukę bardziej złożonych cech), wykorzystanie augmentacji danych (Data Augmentation) w celu sztucznego powiększenia zbioru treningowego i poprawy odporności modelu na niewielkie transformacje, oraz implementację dynamicznej redukcji współczynnika uczenia (Learning Rate Scheduling) dla potencjalnego ulepszenia procesu zbieżności do optymalnego minimum funkcji kosztu.

Prace zostały zrealizowane w elastycznym i interaktywnym środowisku Google Colaboratory, co umożliwiło łatwe eksperymentowanie i wykorzystanie zasobów chmurowych (GPU). Wykorzystano przy tym standardowy stos technologiczny języka Python dla uczenia maszynowego: bibliotekę TensorFlow wraz z jej wysokopoziomowym API Keras do definicji, kompilacji i treningu modeli neuronowych; NumPy do efektywnych operacji na tablicach danych; Scikit-learn do podziału zbiorów danych i obliczania metryk ewaluacyjnych; oraz Matplotlib i Seaborn do tworzenia czytelnych wizualizacji wyników i procesu uczenia. Do pobierania danych wykorzystano bibliotekę ‘requests‘.

Cały proces badawczy, implementacja kodu oraz szczegółowe wyniki pośrednie zostały udokumentowane w notatniku Google Colab. Dodatkowo, przygotowano dokumentację opisującą kroki postępowania i założenia projektu. Oba zasoby są dostępne publicznie pod poniższymi linkami:

- Główny notatnik roboczy Google Colab (zawierający kod i wyniki): Link do Colab
- Dokumentacja / Instrukcje projektu (Google Docs): Link do Instrukcji

W niniejszym sprawozdaniu przedstawiono syntetyczne podsumowanie przeprowadzonych prac, obejmujące metodykę przygotowania danych, opis zastosowanych architektur sieci neuronowych, przebieg procesu treningu i ewaluacji, prezentację kluczowych wyników wraz z ich interpretacją oraz wnioski płynące z przeprowadzonych eksperymentów porównawczych.

2 Metodologia

Proces badawczy obejmował następujące kroki:

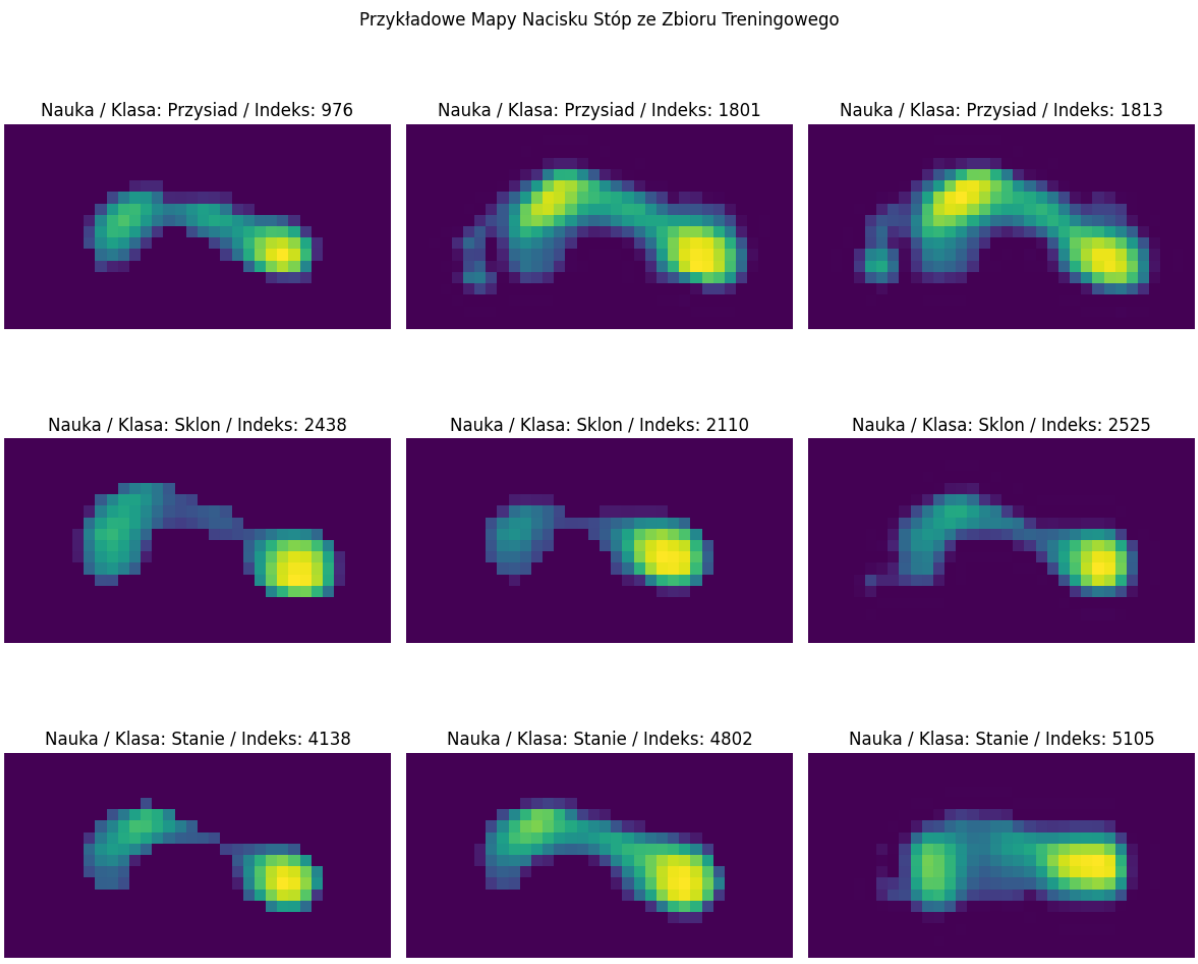
2.1 Wczytywanie i Przygotowanie Danych

Dane wejściowe, składające się z map nacisku stóp oraz odpowiadających im etykiet pozycji ciała, zostały pobrane ze zdalnego serwera (`bigbang.prz.edu.pl`) w formacie plików `.npy`. Dostępne były oddzielne zbiory treningowy i testowy.

Kształty danych:

- Treningowe (cechy): `(5416, 18, 34)` po transpozycji
- Testowe (cechy): `(744, 18, 34)` po transpozycji
- Treningowe (etykiety): `(5416, 1)`
- Testowe (etykiety): `(744, 1)`

Klasy mapowano następująco: 0 - Przysiad, 1 - Skłon, 2 - Stanie.



Rysunek 1: Przykładowe mapy nacisku stóp ze zbioru treningowego dla każdej z trzech klas (Przysiad, Skłon, Stanie).

Kluczowe kroki przetwarzania wstępnego obejmowały:

- **Transpozycja wymiarów:** Zmiana kolejności wymiarów w tablicach cech, aby uzyskać format (liczba_obrazów, wysokość, szerokość).
- **Normalizacja:** Przeskalowanie wartości pikseli z zakresu [0, 255] do zakresu [0, 1] poprzez podzielenie przez 255.0.
- **Spłaszczenie (dla FCN):** Przekształcenie obrazów 18x34 w jednowymiarowe wektory o długości $18 \times 34 = 612$ cech.
- **Zmiana kształtu (dla CNN):** Dodanie wymiaru kanału (1 dla skali szarości), uzyskując kształt wejściowy (18, 34, 1).
- **Kodowanie One-Hot Etykiet:** Konwersja etykiet klas (0, 1, 2) do postaci wektorów binarnych (np. $0 \rightarrow [1, 0, 0]$, $1 \rightarrow [0, 1, 0]$, $2 \rightarrow [0, 0, 1]$).
- **Podział na zbiór treningowy i walidacyjny:** Podział oryginalnego zbioru treningowego na właściwy zbiór treningowy (85%) i zbiór walidacyjny (15%) przy użyciu funkcji `train_test_split` z biblioteki Scikit-learn (z parametrem `random_state=173201` i stratyfikacją). Zbiór testowy pozostał nienaruszony do końcowej ewaluacji.

2.2 Architektury Modeli

Zdefiniowano i porównano kilka modeli:

2.2.1 Model FCN (W Pełni Połączony)

Prosta sieć składająca się z:

- Warstwy wejściowej (niejawnej, przyjmującej 612 cech).
- Dwóch ukrytych warstw gęstych (Dense) po 7 neuronów z aktywacją ReLU, każda poprzedzona warstwą Dropout (współczynnik 0.1).
- Warstwy wyjściowej Dense z 3 neuronami (liczba klas) i aktywacją Softmax.

Podsumowanie Architektury FCN:

```
Model: "Fully_Connected_Network"

 Layer (type)                Output Shape              Param #

 dense (Dense)               (None, 7)                   4,291

 dropout (Dropout)           (None, 7)                       0

 dense_1 (Dense)             (None, 7)                      56

 dropout_1 (Dropout)         (None, 7)                       0

 dense_2 (Dense)             (None, 3)                      24

 Total params: 4,371 (17.07 KB)
 Trainable params: 4,371 (17.07 KB)
 Non-trainable params: 0 (0.00 B)
```

2.2.2 Model CNN (Bazowy)

Podstawowa sieć konwolucyjna:

- Warstwa Conv2D (4 filtry, kernel 3x3, ReLU, padding 'valid') → MaxPooling2D (2x2).
- Warstwa Conv2D (4 filtry, kernel 3x3, ReLU, padding 'valid') → MaxPooling2D (2x2).
- Warstwa Conv2D (4 filtry, kernel 3x3, ReLU, padding 'valid').
- Warstwa Flatten.
- Warstwa Dense (16 neuronów, ReLU).
- Warstwa Dropout (0.2).
- Warstwa wyjściowa Dense (3 neurony, Softmax).

Podsumowanie Architektury CNN (Bazowy):

```
Model: "Convolutional_Neural_Network"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 16, 32, 4)	40
max_pooling2d (MaxPooling2D)	(None, 8, 16, 4)	0
conv2d_1 (Conv2D)	(None, 6, 14, 4)	148
max_pooling2d_1 (MaxPooling2D)	(None, 3, 7, 4)	0
conv2d_2 (Conv2D)	(None, 1, 5, 4)	148
flatten (Flatten)	(None, 20)	0
dense (Dense)	(None, 16)	336
dropout (Dropout)	(None, 16)	0
dense_1 (Dense)	(None, 3)	51

```
Total params: 723 (2.82 KB)
Trainable params: 723 (2.82 KB)
Non-trainable params: 0 (0.00 B)
```

2.2.3 Modele CNN (Optymalizacje)

Na bazie modelu CNN stworzono trzy kolejne wersje, wprowadzając modyfikacje:

- **CNN v2 (BatchNorm, More Filters):** Dodano warstwy Batch Normalization po każdej warstwie Conv2D i Dense (przed aktywacją lub Dropout). Zwiększono

liczbę filtrów w warstwach Conv2D (8, 16, 32) i neuronów w warstwie Dense (32). Zwiększono Dropout do 0.3.

- **CNN v3 (Augmentation):** Bazując na architekturze v2, dodano na początku modelu warstwy augmentacji danych (`RandomFlip`, `RandomRotation`, `RandomZoom`).
- **CNN v4 (LR Scheduling):** Architektura identyczna jak v3, ale podczas treningu zastosowano callback `ReduceLROnPlateau` do dynamicznego zmniejszania współczynnika uczenia.

Podsumowania architektur v2, v3 i v4 zostały przedstawione w logach wykonania skryptu i są podobne do siebie pod względem kolejności warstw (z dodaniem BN i augmentacji). Ich podsumowania 'model.summary()' wyglądają tak samo jak w poprzednim przebiegu, ponieważ 'random$_s$*tate*'*niewpywanasamarchitektur*.

2.3 Proces Treningu

Wszystkie modele kompilowano przy użyciu:

- Optymalizatora: `Adam`.
- Funkcji straty: `categorical_crossentropy` (odpowiednia dla klasyfikacji wieloklasowej z etykietami one-hot).
- Metryki: `accuracy`.

Trening przeprowadzono w dwóch etapach dla modeli FCN i bazowego CNN:

1. **Trening wstępny:** Dłuższy trening (500 epok) na zbiorze treningowym z monitorowaniem wydajności na zbiorze walidacyjnym. Celem było zidentyfikowanie optymalnej liczby epok, minimalizującej stratę na zbiorze walidacyjnym (`val_loss`), aby uniknąć przeuczenia.
2. **Trening ostateczny:** Resetowanie wag modelu i ponowny trening na zbiorze treningowym przez liczbę epok wyznaczoną w kroku 1.

Dla modeli optymalizacyjnych (v2, v3, v4) zastosowano liczbę epok bazującą na optymalnej liczbie dla modelu bazowego CNN (v2 użyło tej samej liczby, v3 i v4 podwojonej liczby epok, wyznaczonej dla nowego 'random_state'). Rozmiar wsadu (`batch_size`) ustawiono na 32. W modelu v3 augmentacja była stosowana w locie, a w v4 działał callback redukcji LR.

2.4 Ewaluacja Modeli

Ostateczna ocena modeli została przeprowadzona na **zbiorze testowym**, który nie był używany podczas treningu ani walidacji. Główne metryki oceny to:

- **Dokładność (Accuracy):** Odsetek poprawnie sklasyfikowanych próbek.
- **Strata (Loss):** Wartość funkcji straty na zbiorze testowym.
- **Macierz Pomyłek (Confusion Matrix):** Tabela pokazująca liczbę próbek z każdej prawdziwej klasy, które zostały przypisane do każdej przewidywanej klasy. Pozwala zidentyfikować, które klasy są najczęściej mylone.

Dodatkowo, wizualizowano krzywe uczenia (dokładność i strata vs epoka) dla zbiorów treningowego i walidacyjnego dla każdego etapu treningu, aby ocenić proces nauki i zidentyfikować potencjalne problemy (np. przeuczenie).

3 Uzasadnienie Wyboru Metod i Narzędzi

3.1 Wybór Metod (FCN vs. CNN)

Porównanie FCN i CNN jest standardowym podejściem przy wprowadzaniu do problemów klasyfikacji obrazów. FCN traktuje obraz jako płaski wektor cech, ignorując jego dwuwymiarową strukturę, podczas gdy CNN jest specjalnie zaprojektowany do wykorzystywania lokalnych zależności przestrzennych w obrazach za pomocą warstw konwolucyjnych i poolingu. Oczekiwano, że CNN okaże się skuteczniejszy. Wybrane techniki optymalizacji (BN, Augmentacja, LR Scheduling) są powszechnie stosowanymi metodami poprawy wydajności i stabilności treningu modeli CNN.

3.2 Wykorzystane Narzędzia

Wybrano standardowy i popularny stos technologiczny dla uczenia głębokiego w Pythonie:

- **TensorFlow/Keras:** Potężna i elastyczna biblioteka do budowy i trenowania sieci neuronowych. Keras API upraszcza definicję modeli.
- **NumPy:** Fundamentalna biblioteka do obliczeń numerycznych, zwłaszcza operacji na tablicach.
- **Scikit-learn:** Zapewnia narzędzia do podziału danych (`train_test_split`) oraz metryki oceny (`accuracy_score`, `confusion_matrix`).
- **Matplotlib/Seaborn:** Biblioteki do tworzenia wizualizacji (wykresy uczenia, mapy nacisku, macierze pomyłek).
- **Requests:** Użyto do niezawodnego pobierania plików danych w środowisku Colab.

Wybór ten podyktowany był ich dojrzałością, obszerną dokumentacją, wsparciem społeczności oraz efektywnością w realizacji postawionych zadań.

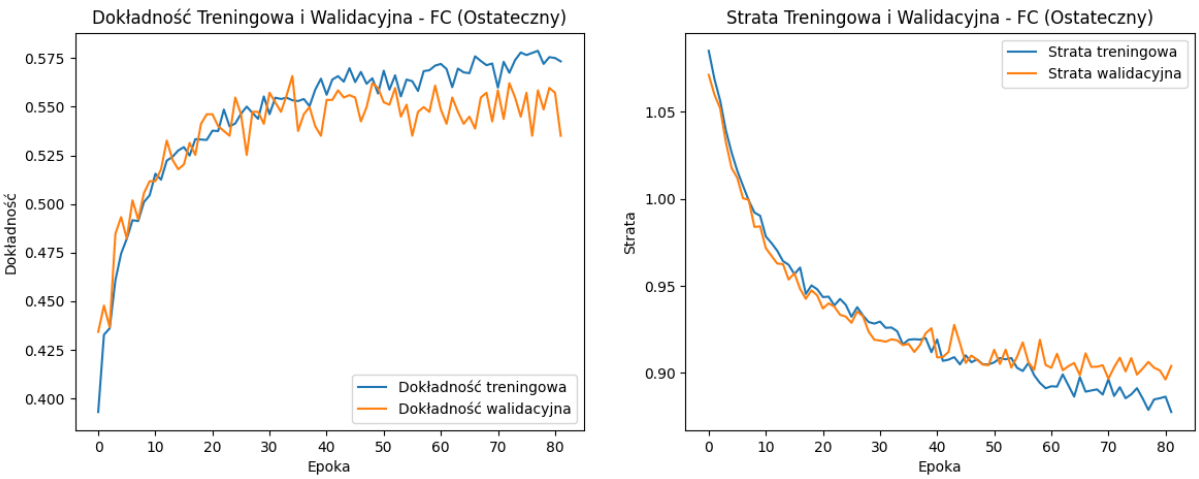
4 Wyniki i Interpretacja

W tej sekcji przedstawiono wyniki uzyskane dla poszczególnych modeli na zbiorze testowym oraz ich interpretację, bazując na przebiegu z `random_state=173201`.

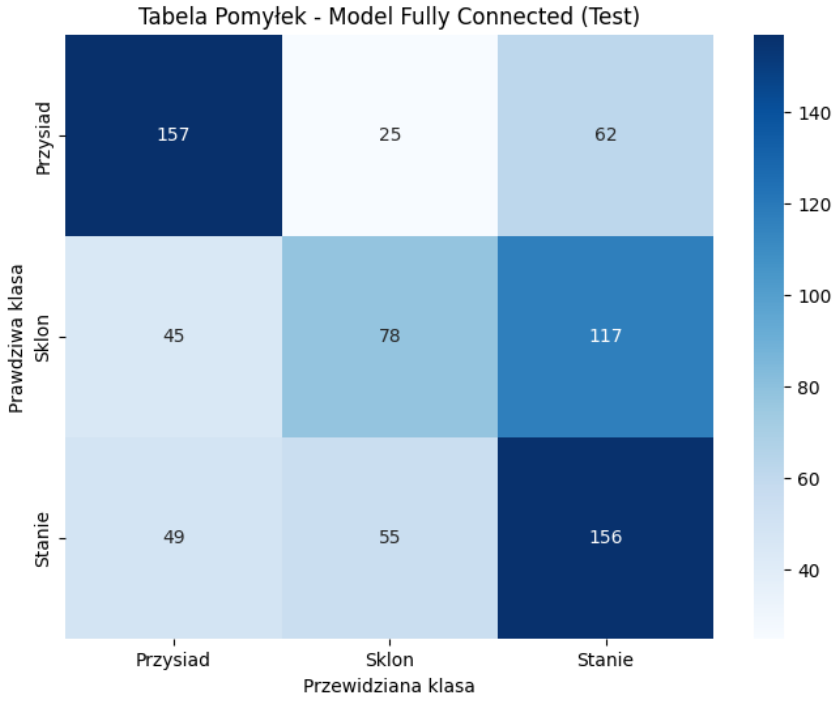
4.1 Model FCN

Ostateczny model FCN, wytrenowany przez 82 epoki (optymalna liczba zidentyfikowana na podstawie `val_loss`=0.8645 w treningu wstępnym), osiągnął na zbiorze testowym:

- **Dokładność: 52.55%**
- **Strata: 1.0120**



Rysunek 2: Krzywe uczenia (dokładność i strata) dla ostatecznego treningu modelu FCN (82 epoki).



Rysunek 3: Macierz pomyłek dla modelu FCN na zbiorze testowym.

Interpretacja macierzy pomyłek FCN (Rysunek 3): Model FCN, mimo osiągnięcia lepszej dokładności niż w poprzednim przebiegu, nadal wykazuje problemy z klasyfikacją. Najczęstsze błędy dotyczą:

- Mylenie Skłonu (1) ze Staniem (2): 117 przypadków.
- Mylenie Przysiadu (0) ze Staniem (2): 62 przypadki.
- Mylenie Stania (2) ze Skłonem (1): 55 przypadków.
- Mylenie Stania (2) z Przysiadem (0): 49 przypadków.

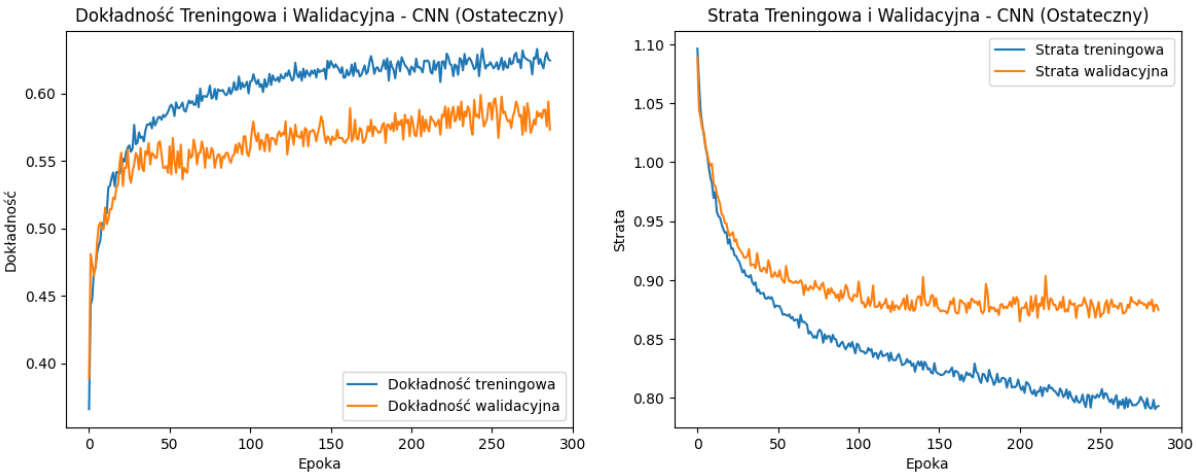
- Mylenie Skłonu (1) z Przysiadem (0): 45 przypadków.
- Mylenie Przysiadu (0) ze Skłonem (1): 25 przypadków.

Model nadal najczęściej myli Skłon ze Staniem. Ogólna dokładność 53% jest niska, ale zauważalnie lepsza niż poprzednio.

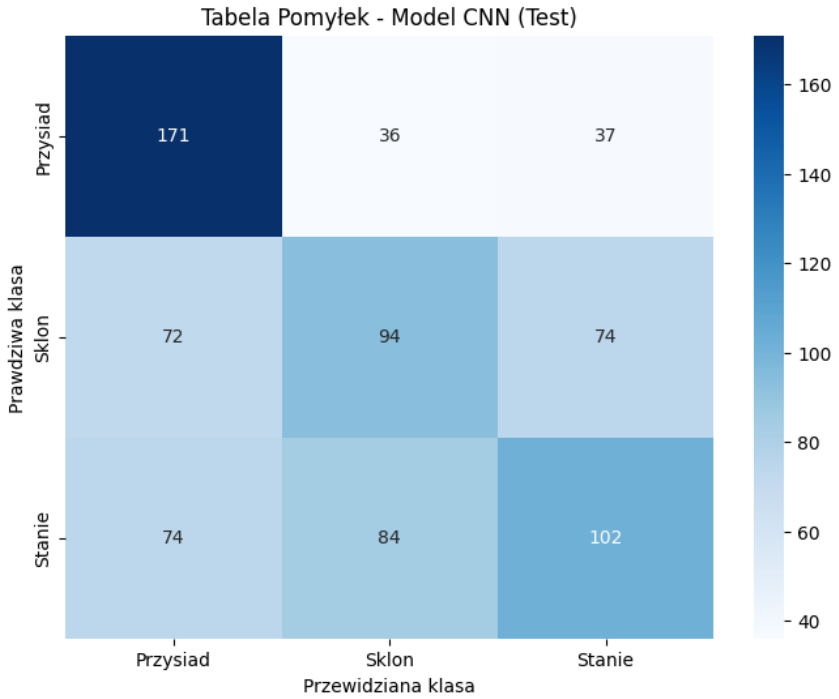
4.2 Model CNN (Bazowy)

Ostateczny model CNN, wytrenowany przez 287 epok (optymalna liczba zidentyfikowana na podstawie `val_loss`=0.8940 w treningu wstępnym), osiągnął na zbiorze testowym:

- **Dokładność: 49.33%**
- **Strata: 1.0117**



Rysunek 4: Krzywe uczenia (dokładność i strata) dla ostatecznego treningu modelu CNN (287 epok).



Rysunek 5: Macierz pomyłek dla bazowego modelu CNN na zbiorze testowym.

Interpretacja macierzy pomyłek CNN (Rysunek 5): W tym przebiegu bazowy model CNN osiągnął dokładność niższą niż FCN (49.33% vs 52.55%) i niższą niż w poprzednim przebiegu. Model ma duże trudności z rozróżnianiem klas. Najczęstsze błędy to:

- Mylenie Stania (2) ze Skłonem (1): 84 przypadki.
- Mylenie Stania (2) z Przysiadem (0): 74 przypadki.
- Mylenie Skłonu (1) ze Staniem (2): 74 przypadki.
- Mylenie Skłonu (1) z Przysiadem (0): 72 przypadki.
- Mylenie Przysiadu (0) ze Staniem (2): 37 przypadków.
- Mylenie Przysiadu (0) ze Skłonem (1): 36 przypadków.

Wyraźnie widać, że model bardzo często myli klasy Stanie i Skłon w obie strony, a także Stanie z Przysiadem. Wynik poniżej 50% wskazuje na słabą wydajność tej konkretnej architektury przy tym podziale danych.

4.3 Optymalizacje CNN

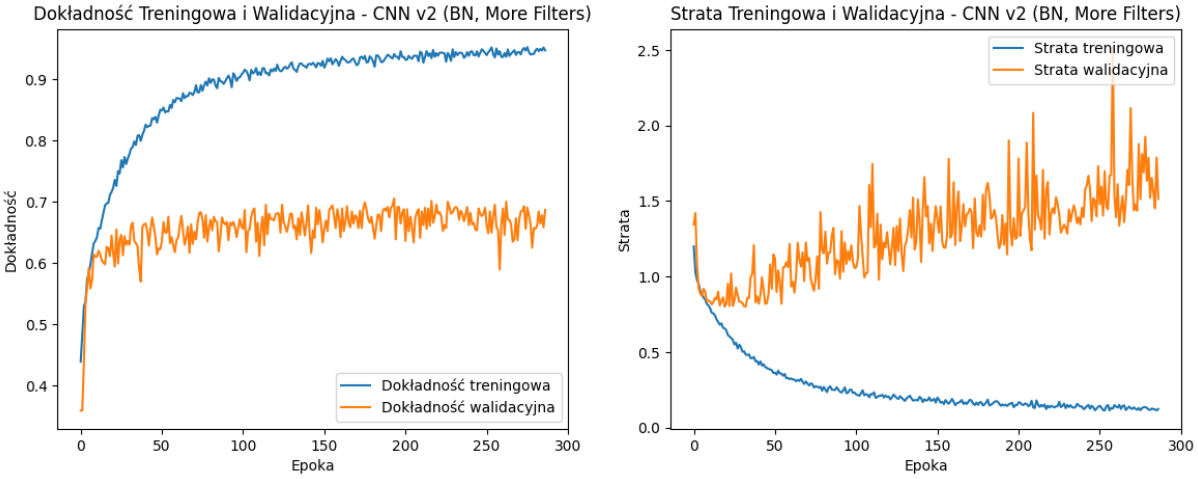
Przeprowadzono trzy eksperymenty mające na celu poprawę bazowego modelu CNN, trenując je na danych podzielonych z `random_state=173201`.

4.3.1 CNN v2 (BatchNorm, More Filters)

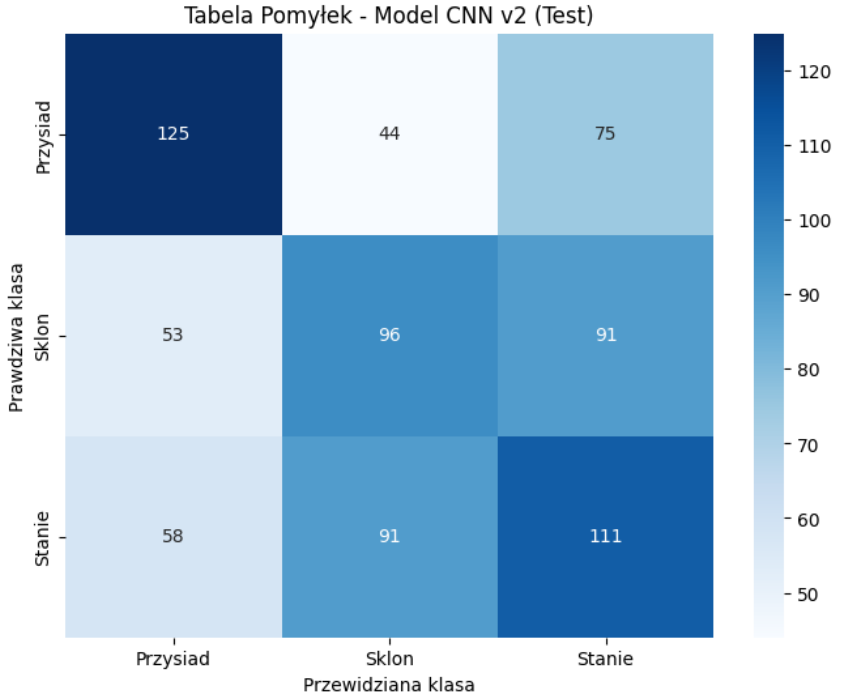
Model z dodaną normalizacją wsadową i większą liczbą filtrów/neuronów (trenowany przez 287 epok) osiągnął na zbiorze testowym:

- **Dokładność: 44.62%**
- **Strata: 3.7451**

Wynik ten jest **najgorszy** spośród wszystkich modeli w tym przebiegu i gorszy niż w poprzednim uruchomieniu dla v2. Mimo że krzywe uczenia (Rysunek 6) mogły pokazywać lepszą dokładność walidacyjną (nawet >70% w pewnym momencie), model fatalnie zgeneralizował na zbiór testowy. Strata jest bardzo wysoka.



Rysunek 6: Krzywe uczenia dla modelu CNN v2 (BN, More Filters, 287 epok).



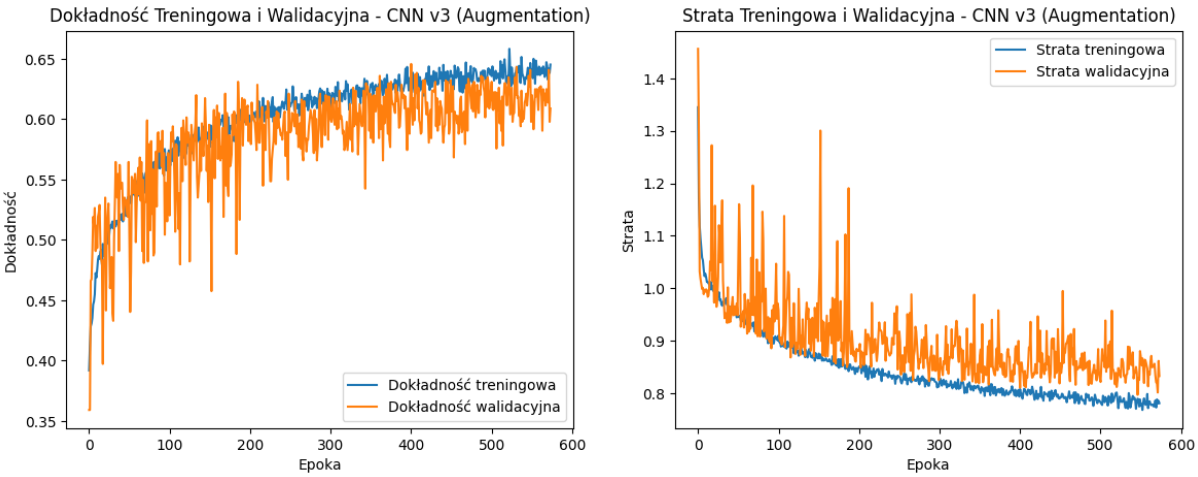
Rysunek 7: Macierz pomyłek dla modelu CNN v2 na zbiorze testowym.

4.3.2 CNN v3 (Augmentation)

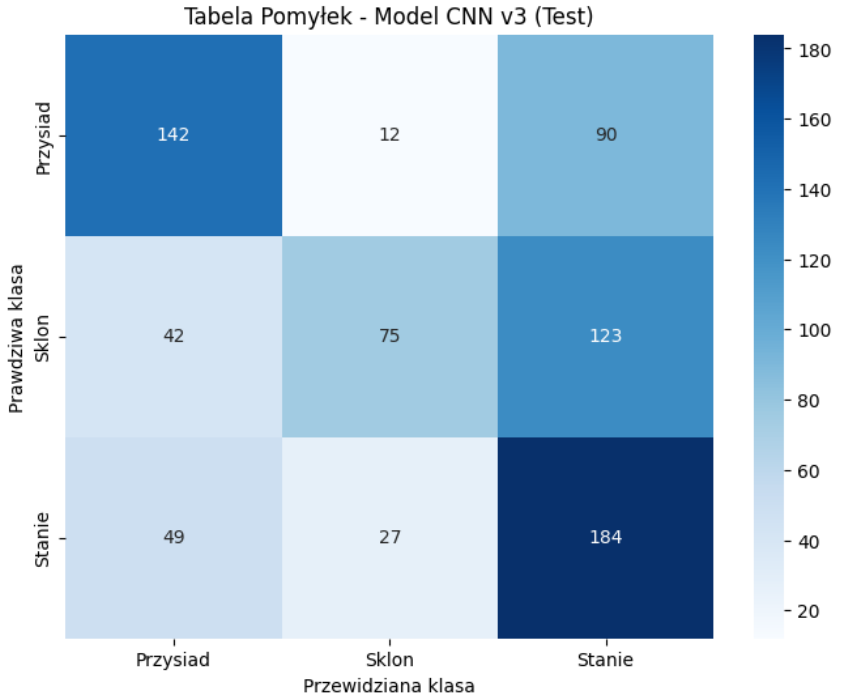
Model v2 wzbogacony o augmentację danych (trenowany przez 574 epoki = 2 * 287) osiągnął na zbiorze testowym:

- **Dokładność: 53.90%**
- **Strata: 1.0241**

Ten model uzyskał **najlepszą dokładność** spośród wszystkich testowanych modeli w tym przebiegu, nieznacznie przewyższając FCN (52.55%). Augmentacja danych ponownie okazała się pomocna w generalizacji. Krzywe uczenia (Rysunek 8) pokazują oczekiwane zachowanie dla augmentacji.



Rysunek 8: Krzywe uczenia dla modelu CNN v3 (Augmentation, 574 epoki).



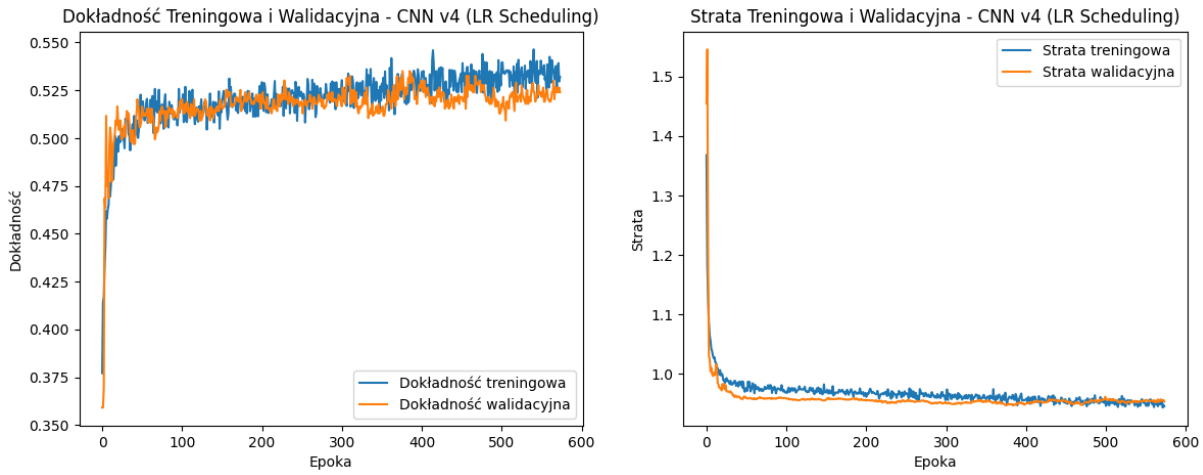
Rysunek 9: Macierz pomyłek dla modelu CNN v3 na zbiorze testowym.

4.3.3 CNN v4 (LR Scheduling)

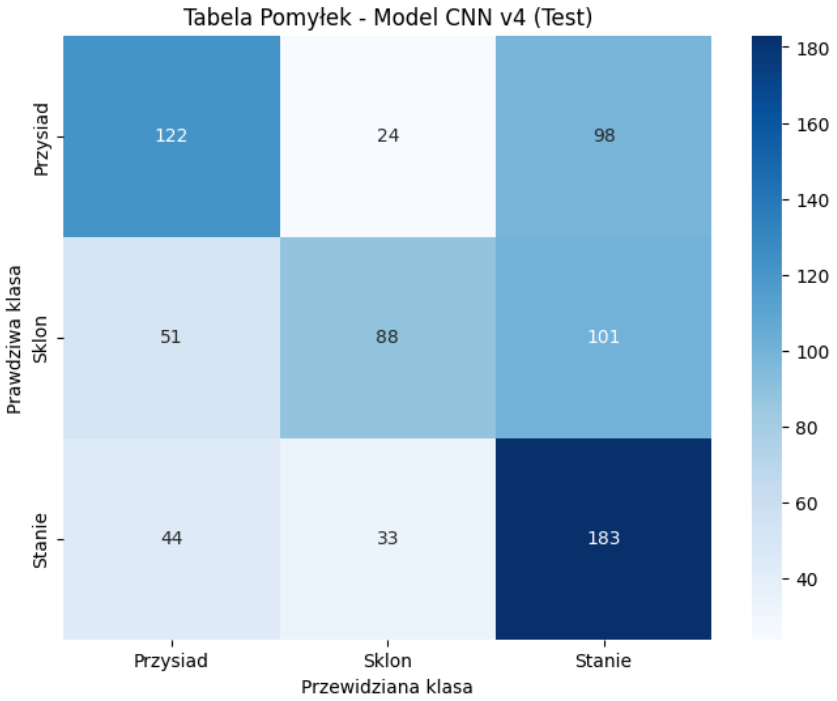
Model v3 trenowany z dynamiczną redukcją współczynnika uczenia (przez 574 epoki) osiągnął na zbiorze testowym:

- **Dokładność: 52.82%**
- **Strata: 0.9833**

Ten model osiągnął wynik bardzo zbliżony do FCN i nieco gorszy od modelu v3 (bez LR Scheduling). Callback redukcji LR zadziałał, ale nie przełożyło się to na poprawę dokładności w porównaniu do v3.



Rysunek 10: Krzywe uczenia dla modelu CNN v4 (LR Scheduling, 574 epoki).



Rysunek 11: Macierz pomyłek dla modelu CNN v4 na zbiorze testowym.

4.4 Porównanie Modeli

Tabela 1 podsumowuje końcowe dokładności uzyskane przez wszystkie modele na zbiorze testowym dla przebiegu z `random_state=173201`.

Tabela 1: Porównanie dokładności modeli na zbiorze testowym.

Model	Dokładność na zbiorze testowym (%)
FCN (bazowy)	52.55
CNN (bazowy)	49.33
CNN v2 (BN, More Filters)	44.62
CNN v3 (Augmentation)	**53.90**
CNN v4 (LR Scheduling)	52.82

W tym przebiegu eksperymentu, w przeciwieństwie do poprzedniego, prosty model FCN (52.55%) okazał się lepszy niż bazowy model CNN (49.33%). Najlepszy wynik ponownie osiągnął model CNN v3 wykorzystujący augmentację danych (53.90%), jednak jego przewaga nad modelem FCN jest niewielka (ok. 1.35 punktu procentowego). Model CNN v4 z redukcją LR osiągnął wynik porównywalny z FCN (52.82%). Najsłabiej wypadł model CNN v2 z Batch Normalization i większą liczbą filtrów (44.62%).

Wyniki te potwierdzają, że augmentacja danych była najbardziej skuteczną techniką optymalizacji w tym przypadku. Pokazują również dużą wrażliwość wyników, zwłaszcza dla prostego CNN, na konkretny podział danych treningowych/walidacyjnych (wpływ 'random_state'). Mimo zastosowania optymalizacji, ogólna uzyskana dokładność (maksymalnie 54%) pozostaje stosunkowo niska, co sugeruje, że zadanie jest trudne, a dane mogą być zaszumione lub nie zawierać wystarczająco wyraźnych cech różnicujących klasy.

5 Podsumowanie i Wnioski

W ramach projektu przeprowadzono porównanie sieci neuronowych FCN i CNN oraz zbadano wpływ technik optymalizacyjnych (Batch Normalization, augmentacja danych, LR Scheduling) na zadanie klasyfikacji pozycji ciała (przysiad, skłon, stanie) na podstawie map nacisku stopy, wykorzystując podział danych zdeterminowany przez `random_state=173201`.

Główne wnioski:

1. **Niejednoznaczna Przewaga CNN nad FCN:** W tym konkretnym przebiegu, bazowy model CNN wypadł gorzej niż FCN. Dopiero zastosowanie augmentacji danych w modelu CNN v3 pozwoliło uzyskać nieznaczną przewagę nad FCN. Potwierdza to potencjał CNN, ale wskazuje też na wrażliwość prostych architektur na dane lub potrzebę lepszego dostrojenia.
2. **Kluczowa Rola Augmentacji Danych:** Ponownie, zastosowanie augmentacji danych (model v3) przyniosło najlepszy wynik (53.90% dokładności), co silnie sugeruje, że technika ta jest kluczowa dla poprawy generalizacji przy tym zbiorze danych.
3. **Negatywny Wpływ Innych Optymalizacji w Tej Konfiguracji:** Dodanie Batch Normalization i zwiększenie złożoności modelu (v2) drastycznie pogorszyło wynik (44.62%). Zastosowanie LR Scheduling (v4) dało wynik porównywalny z FCN,

ale gorszy niż sama augmentacja (v3). Wskazuje to, że te techniki wymagałyby dalszego, starannego dostrojenia hiperparametrów w kontekście tego problemu.

4. **Trudność Zadania Klasyfikacyjnego:** Uzyskane dokładności (maksymalnie ok. 54%) wciąż sugerują, że zadanie jest wymagające. Mapy nacisku dla różnych pozycji mogą być trudne do rozróżnienia nawet dla zoptymalizowanych modeli. Analiza macierzy pomyłek pokazała utrzymujące się problemy z myleniem klas.

5. **Znaczenie Walidacji i Wpływ 'random_state':** Proces dwuetapowego treningu i użycie zbioru walidacyjnego były istotne. Porównanie wyników między przebiegami z różnymi 'random_state' pokazuje, jak duży wpływ na wyniki może mieć losowy podział danych, co podkreśla potrzebę np. walidacji krzyżowej w bardziej zaawansowanych badaniach.