Rozpoznawanie symboli muzycznych przy użyciu konwolucyjnych sieci neuronowych

Kornel Dzieża

January 15, 2026

1 Wstęp

Celem projektu było stworzenie systemu rozpoznawania symboli muzycznych (nut, pauz oraz kluczy muzycznych) na obrazach o rozmiarze 128x128 pikseli. Obrazki te stanowią wycinki większych partytur muzycznych zawierających pięciolinię. Projekt ten jest częścią szerszego zagadnienia, jakim jest automatyczny odczyt zapisu nutowego z zeskanowanych materiałów muzycznych.

Drugim, równie istotnym celem projektu było praktyczne zapoznanie się z zagadnieniami uczenia maszynowego oraz głębokiego uczenia. W trakcie realizacji projektu autor zdobył doświadczenie w projektowaniu, trenowaniu oraz analizie konwolucyjnych sieci neuronowych (CNN). Wiedza zdobyta podczas wcześniejszych eksperymentów z rozpoznawaniem cyfr pisanych ręcznie została wykorzystana jako punkt wyjścia do pracy nad symbolami muzycznymi.

Na początkowym etapie projektu podjęto próbę zastosowania klasycznej architektury **AlexNet**. Model ten jednak okazał się nieodpowiedni do pracy z małymi obrazami oraz ograniczoną liczbą danych treningowych. W rezultacie zaprojektowano uproszczoną architekturę **AlexNet-light**, lepiej dopasowaną do charakteru problemu.

2 Opis teoretyczny rozwiązania

2.1 Konwolucyjne sieci neuronowe

Jednym z pierwszych problemów napotkanych podczas realizacji projektu był brak odpowiedniego zbioru danych zawierającego pojedyncze symbole muzyczne. W celu jego rozwiązania wykorzystano istniejącą czcionkę symboli muzycznych, na podstawie której wygenerowano syntetyczne dane uczące. Obrazy różniły się położeniem symboli, niewielkimi przesunięciami oraz losowymi artefaktami.

Konwolucyjne sieci neuronowe (CNN) są powszechnie stosowane w zadaniach analizy obrazu. Składają się z warstw konwolucyjnych odpowiedzialnych za ekstrakcję cech, warstw normalizacji, warstw redukujących rozmiar danych (pooling) oraz warstw gęstych realizujących klasyfikację. Ich struktura pozwala na wykrywanie cech niezależnie od położenia obiektu w obrazie.

2.2 Architektura AlexNet-light

Architektura AlexNet-light została zaprojektowana jako odpowiedź na ograniczenia klasycznego modelu AlexNet w kontekście analizy niewielkich obrazów symboli muzycznych. Oryginalny AlexNet został stworzony z myślą o dużych obrazach (224x224 piksele) oraz bardzo rozbudowanych zbiorach danych. W przypadku niniejszego projektu zastosowanie tak dużej sieci prowadziło do nadmiernego przeuczenia oraz niestabilnego procesu trenowania.

Model AlexNet-light zachowuje główne założenia architektury AlexNet, takie jak hierarchiczna ekstrakcja cech oraz stopniowe zwiększanie liczby filtrów konwolucyjnych, jednak w znacznie uproszczonej i dostosowanej do problemu formie.

2.2.1 Wejście i wstępne przetwarzanie danych

Na wejściu sieci znajduje się obraz w skali szarości o rozmiarze 128x128 pikseli. Wybór skali szarości pozwala na ograniczenie liczby parametrów oraz skupienie się na kształcie symboli, który jest kluczowy w rozpoznawaniu zapisu nutowego.

Przed właściwym przetwarzaniem obrazu stosowana jest augmentacja danych, obejmująca niewielkie rotacje, translacje, zmiany skali oraz kontrastu. Zabieg ten pozwala na zwiększenie różnorodności danych treningowych oraz poprawę odporności modelu na niewielkie deformacje symboli, które mogą występować w rzeczywistych skanach partytur.

2.2.2 Bloki konwolucyjne

Główna część modelu składa się z trzech bloków konwolucyjnych. Każdy blok zawiera następujące elementy:

- warstwę konwolucyjną (`Conv2D`) z funkcją aktywacji ReLU,
- normalizację wsadową (`BatchNormalization`),
- warstwę redukującą rozmiar danych (`MaxPooling`),
- warstwę regularyzującą (`Dropout`).

W kolejnych blokach liczba filtrów konwolucyjnych wzrasta (32, 64 oraz 128), co umożliwia modelowi przechodzenie od wykrywania prostych cech (takich jak krawędzie i linie) do bardziej złożonych struktur odpowiadających całym symbolom muzycznym.

Zastosowanie normalizacji wsadowej stabilizuje proces uczenia oraz przyspiesza zbieżność modelu. Warstwy `Dropout` redukują ryzyko przeuczenia poprzez losowe wyłączanie części neuronów w trakcie treningu.

2.2.3 Warstwy gęste i klasyfikacja

Po zakończeniu ekstrakcji cech mapa aktywacji jest spłaszczana przy użyciu warstwy `Flatten`, a następnie przekazywana do warstw gęstych. Pierwsza warstwa gęsta składa się z 256 neuronów z funkcją aktywacji ReLU i pełni rolę klasyfikatora wysokiego poziomu, łącząc wykryte cechy w spójną reprezentację symbolu.

Ostatnia warstwa gęsta wykorzystuje funkcję aktywacji `softmax`, która zwraca rozkład prawdopodobieństwa przynależności obrazu do jednej z klas symboli muzycznych. Klasa o najwyższym prawdopodobieństwie jest traktowana jako końcowa predykcja modelu.

2.2.4 Zalety zaproponowanej architektury

Architektura AlexNet-light charakteryzuje się korzystnym kompromisem pomiędzy złożonością modelu a jego skutecznością. Zmniejszona liczba parametrów pozwala na stabilne trenowanie nawet przy ograniczonej liczbie danych, jednocześnie zachowując wysoką skuteczność klasyfikacji.

Dzięki modularnej budowie model może być łatwo rozszerzany lub modyfikowany, na przykład poprzez dodanie kolejnych bloków konwolucyjnych lub zmianę liczby klas. Architektura ta stanowi solidną podstawę do dalszych prac nad automatycznym rozpoznawaniem zapisu nutowego.

```
def build_light_cnn(input_shape=(128,128,1), num_classes=10):
    model = models.Sequential([
        layers.Input(shape=input_shape),
        data_augmentation,
        layers.Conv2D(32,(3,3),activation='relu',padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2,2)),
        layers.Dropout(0.25),
        layers.Conv2D(64,(3,3),activation='relu',padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2,2)),
        layers.Dropout(0.3),
        layers.Conv2D(128,(3,3),activation='relu',padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2,2)),
        layers.Dropout(0.4),
        layers.Flatten(),
        layers.Dense(256,activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(num_classes,activation='softmax')
    ])
    return model
```

3 Problemy z pięciolinią

W kolejnym etapie projektu podjęto próbę bezpośredniego rozpoznawania symboli wyciętych z obrazów zawierających pięciolinię. Podejście to okazało się nieskuteczne, ponieważ linie pięciolinii znacząco zaburzały proces klasyfikacji.

Po analizie problemu ustalono, że konieczne są dwa dodatkowe etapy:

1. wykrycie i usunięcie linii pięciolinii,
2. klasyfikacja symboli na obrazach pozbawionych linii.

Usuwanie pięciolinii realizowano przy użyciu operacji morfologicznych, co prowadziło do powstawania charakterystycznych artefaktów.

4 Rozszerzenie zbioru danych

Po zidentyfikowaniu źródła problemu zdecydowano się na stworzenie nowego zbioru danych. Zawierał on symbole muzyczne z artefaktami imitującymi efekty czyszczenia pięciolinii. Dane te generowano w analogiczny sposób jak pierwotny zbiór, celowo wprowadzając zakłócenia strukturalne.

Do trenowania wykorzystano tę samą architekturę AlexNet-light, zmieniając jedynie zbiór danych treningowych.

5 Proces trenowania i wyniki

Dane podzielono na zbiór treningowy (80%) oraz walidacyjny (20%). Model trenowano przez maksymalnie 50 epok z użyciem mechanizmów `EarlyStopping` oraz `ModelCheckpoint`.

5.1 Wyniki dla danych bez artefaktów

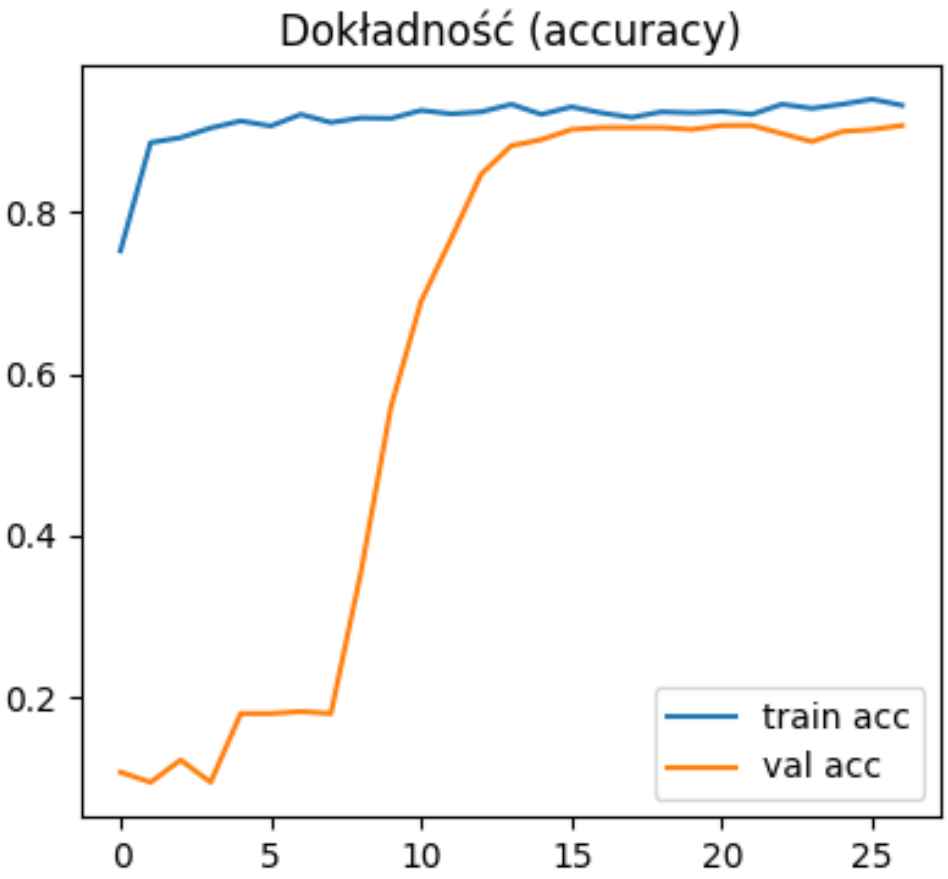

Figure 1: Dokładność modelu trenowanego na czystych danych syntetycznych.

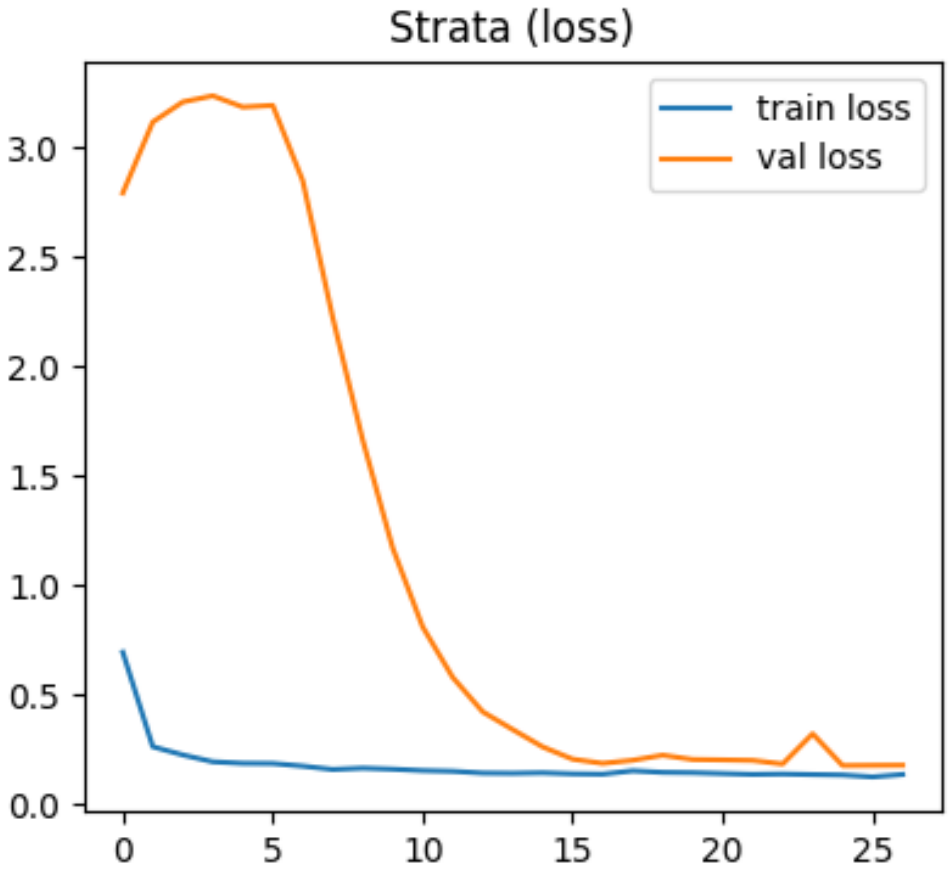

Figure 2: Strata modelu trenowanego na czystych danych syntetycznych.

5.2 Wyniki dla danych z artefaktami (blurry)

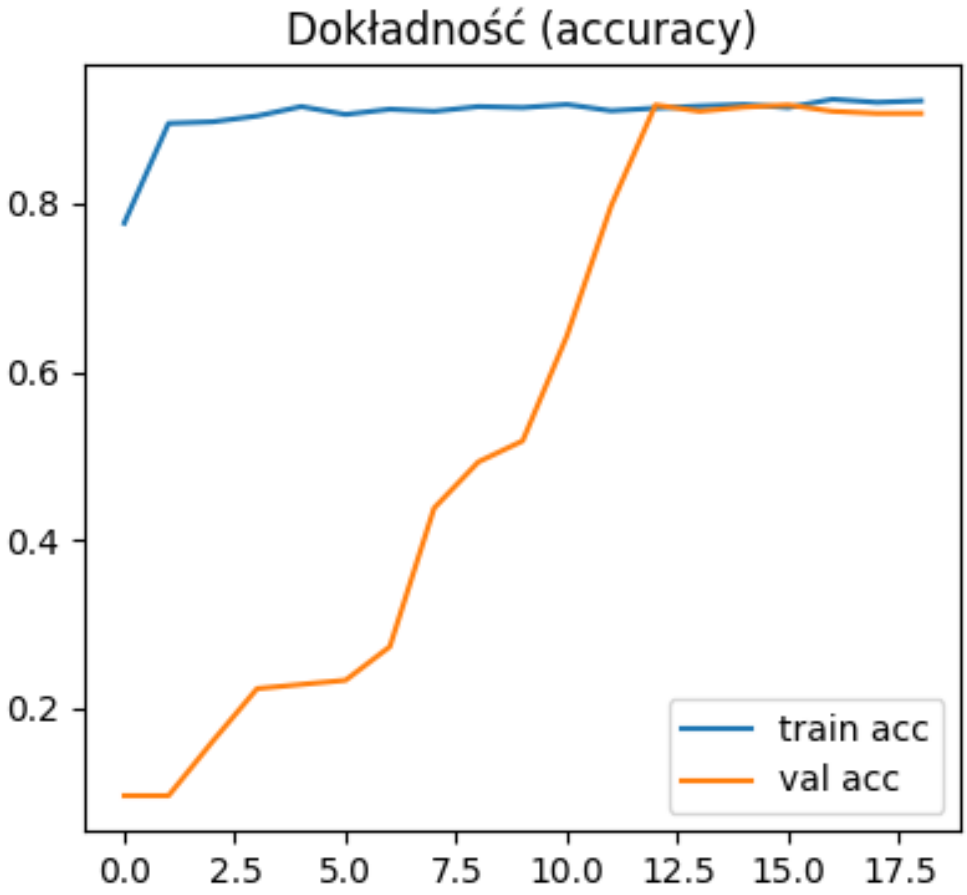

Figure 3: Dokładność modelu trenowanego na danych z artefaktami po usuwaniu pięciolinii.

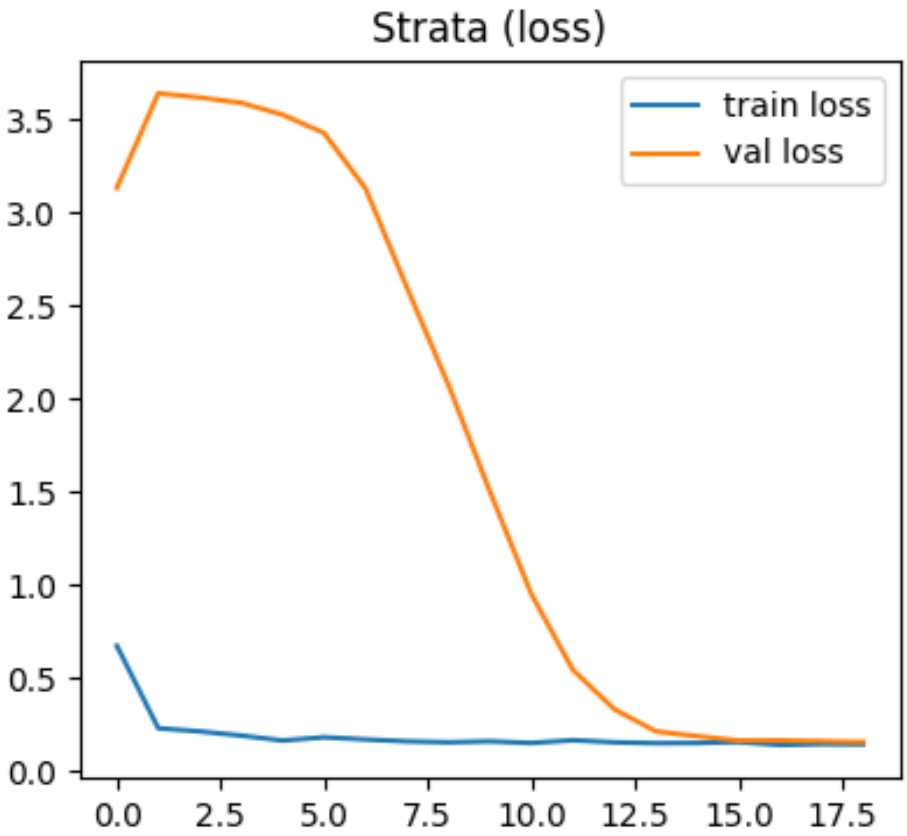

Figure 4: Strata modelu trenowanego na danych z artefaktami po usuwaniu pięciolinii.

6 Jakościowa ocena działania modelu

Oprócz klasycznej analizy metryk ilościowych, takich jak dokładność i strata, istotnym elementem oceny systemu rozpoznawania obrazów jest również analiza jakościowa wyników predykcji. W tym celu opracowano dedykowany skrypt testowy, umożliwiający wizualną ocenę działania wytrenowanych modeli konwolucyjnych.

Skrypt pozwala na wybór jednej z trzech wersji modelu:

- **CLEAN** – model trenowany na czystych danych syntetycznych,
- **FINAL** – model trenowany na danych z augmentacją,
- **BLURRY** – model trenowany na danych z artefaktami imitującymi skutki usuwania pięciolinii.

Po wyborze wersji modelu losowany jest jeden obraz testowy z każdej klasy symboli muzycznych. Następnie obrazy te są jednocześnie klasyfikowane przez model, a wyniki prezentowane są w formie jednej zbiorczej wizualizacji.

Każdy z wyświetlonych obrazów opatrzony jest informacją o klasie rzeczywistej (Ground Truth) oraz klasie przewidzianej przez model (Prediction). Kolor opisu predykcji wskazuje poprawność klasyfikacji: kolor zielony oznacza poprawną predykcję, natomiast kolor czerwony sygnalizuje błąd klasyfikacji. Takie podejście umożliwia szybką identyfikację klas problematycznych oraz ocenę, które symbole są dla modelu najbardziej mylące.

6.1 Wizualizacja wyników testów

Na rysunkach 5 oraz 6 przedstawiono przykładowe wyniki jakościowej ewaluacji dwóch wersji modelu: FINAL oraz BLURRY. Każda wizualizacja zawiera po jednym losowo wybranym przykładzie z każdej klasy, co pozwala na bezpośrednie porównanie skuteczności klasyfikacji dla całego zestawu symboli.

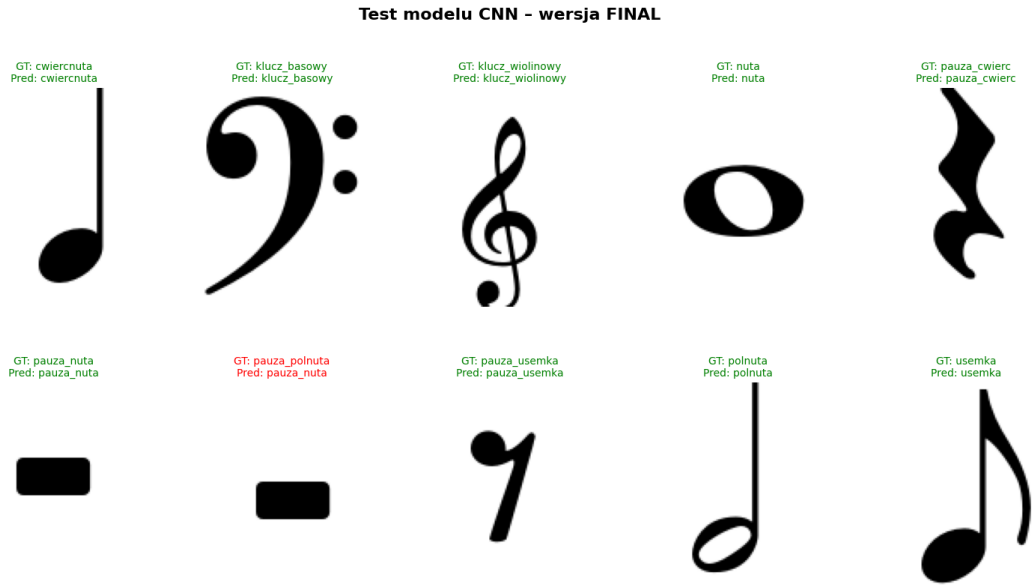

Figure 5: Wyniki jakościowej oceny modelu CNN – wersja FINAL (dane czyste z augmentacją).

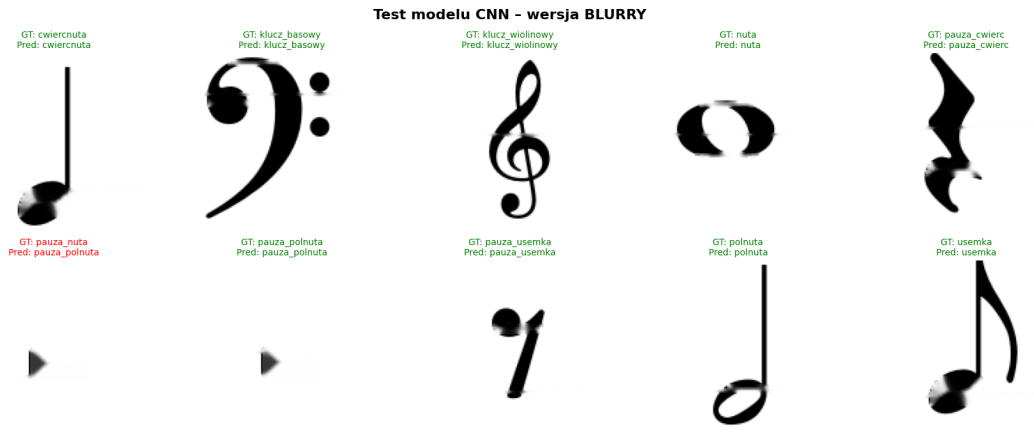

Figure 6: Wyniki jakościowej oceny modelu CNN – wersja BLURRY (dane z artefaktami po usuwaniu pięciolinii).

Na podstawie przedstawionych wyników można zaobserwować, że model trenowany na danych z artefaktami (BLURRY) wykazuje większą odporność na nieregularności kształtów symboli oraz lepiej radzi sobie z przypadkami zniekształconymi. Wersja FINAL osiąga bardzo dobre wyniki na danych czystych, jednak w niektórych przypadkach jest bardziej podatna na błędy wynikające z obecności artefaktów strukturalnych. Można

też tutaj zaobserwować wcześniej wspomniany problem podobności pauzy półnutowaej i całonutowej. Jako, że wyglądają tak samo bez pięciolini jeżeli w pryszłości byśmy implemenowali daje interpretację tych nut to trzebaby odpowiednio policzyć długości nut, pauz w takcie i odpowiednio dostosować symbol lub przypatrywać się lepiej bliskości lińz pięciolinii.

7 Eksperymenty i obserwacje

- Model trenowany na danych z artefaktami wykazuje lepszą generalizację.
- Augmentacja danych zwiększa odporność na przesunięcia i zniekształcenia.
- Najczęściej mylone klasy to pauzy o zbliżonym kształcie.

8 Ograniczenia rozwiązania

1. Dokładność modelu nie osiąga 100%.
2. Silne rozmycie obrazu prowadzi do spadku skuteczności.
3. Bardzo podobne wizualnie symbole są trudne do rozróżnienia.

9 Wnioski

Projekt potwierdził skuteczność konwolucyjnych sieci neuronowych w rozpoznawaniu symboli muzycznych. Kluczowym elementem okazało się odpowiednie przygotowanie danych, w szczególności uwzględnienie artefaktów występujących po usuwaniu pięciolinii. Zaproponowane rozwiązanie stanowi solidną podstawę do dalszego rozwoju systemów automatycznego odczytu zapisu nutowego.