Sieci Neuronowe 2, Projekt

Temat: Sieć Neuronowa Rozpoznająca Instrumenty Muzyczne

Filip Żmuda 263631

Grupa środa 9:15

Prowadząca: mgr inż. Monika Wasilewska

Link do repozytorium z projektem:

https://zts.ita.pwr.wroc.pl/gitlab/filip.zmuda_263631/rozpoznawanie-instrumentow-muzycznych-sieci-neuronowe-2-projekt.git

1. Wprowadzenie

Projekt dotyczy opracowania i wyuczenia sieci neuronowej do rozpoznawania instrumentów muzycznych na podstawie analizy dźwięku. Projekt ten wykorzystuje bibliotekę *librosa* do przetwarzania sygnałów audio oraz *tensorflow* do budowy i treningu modelu sieci neuronowej. Jako typ sieci wybrano sieć konwolucyjną. Bazą danych użytą w projekcie jest baza *Audioset* (Google). Dane audio są przetwarzane w celu wyodrębnienia cech, które następnie są używane do szkolenia modelu.

2. Cel projektu

Celem projektu jest zbudowanie skutecznego modelu, który potrafi klasyfikować pięć rodzajów instrumentów muzycznych:

- Brass instrument - Instrumenty dęte blaszane
- Drum kit - perkusja o nieokreślonej wysokości dźwięku
- Guitar - Gitara (w tym gitary klasyczne, akustyczne i elektryczne)
- Piano - instrumenty klawiszowe (Fortepian, pianino, keyboard, wykluczono organy)
- Violin/fiddle - instrumenty smyczkowe

na podstawie nagrań audio. W projekcie zaimplementowano także techniki augmentacji danych, aby poprawić wydajność modelu i radzić sobie z ograniczoną liczbą próbek.

3. Baza danych

Bazą danych użytą w projekcie jest baza *Audioset* stworzona przez firmę Google. Jest to baza danych stworzona ręcznie przez użytkowników platformy *YouTube*, którzy oznaczają dostępnymi tagami 10-sekundowe fragmenty nagrań dostępnych w serwisie. Oznacza to, że jest ona obarczona ludzkim błędem. Stąd część nagrań zawiera instrumenty podobne do tych, których nazwa widnieje w tagu, jednak nie takich same (np. banjo, czy mandolina są oznaczane jako gitara akustyczna, mimo, że ich zakres częstotliwości dźwięków, czy sama charakterystyka dźwięku jest inna, niż w gitarze), lub fragmenty, w których otagowany instrument wcale nie stanowi głównej części słyszalnego audio. Nagrania są tworzone przez różne osoby, używające różnych instrumentów i sprzętu nagrywającego, różne są też warunki, szum itd. Z jednej strony może to sprawiać problemy z uczeniem sieci, a z drugiej może nadać jej cechę bycia elastyczną w rozpoznawaniu nagrań o różnej charakterystyce. Jednak największe wyzwanie stanowił fakt, że omawiana baza danych nie istnieje jako rzeczywiste pliki audio. Jest ona wyłącznie zbiorem odnośników do 10-sekundowych fragmentów nagrań w serwisie YouTube zapisanym w pliku .csv. Dodatkowo, część nagrań, które znalazły się w bazie na przestrzeni czasu zdążyła zostać usunięta, lub filmy zostały ustawione jako prywatne. Taki stan rzeczy wymusił konieczność stworzenia skryptu, który pozwoli na stabilne i automatyczne pobieranie danych z bazy (jest on załączony w repozytorium poświęconym projektowi). Skrypt znajduje w pliku .csv z linkami identyfikator (lub identyfikatory, jeśli klasa jest złożona z więcej niż jednego instrumentu, np. klasa gitara składa się m.in. z klas gitara akustyczna, gitara elektryczna, które są różnymi instrumentami z tej samej grupy, brzmiącymi podobnie) danego tagu, który określa nazwę klasy, a następnie pobiera wszystkie dostępne pliki należące do tej klasy. Szereg trudności przy tym procesie poskutkował maksymalną liczbą **3962** 10-sekundowych plików audio na każdą z pięciu klas (Rozmiar bazy: 5 x 2,35 GB ≈ 12 GB).

4. Struktura Projektu

Projekt składa się z kilku kluczowych kroków:

1. **Ładowanie i augmentacja danych:**
    - Dane są ładowane z określonego folderu, gdzie każdy podfolder reprezentuje różne klasy instrumentów.
    - Próbki są selekcjonowane losowo z każdej klasy, aby zbalansować dane.
    - Do augmentacji zastosowano zmianę wysokości dźwięku (pitch shift), aby sztucznie zwiększyć różnorodność danych treningowych.
    - Dane są dzielone na zbiór uczący i testowy.
2. **Ekstrakcja cech:**
    - Z każdej próbki audio wyodrębniane są współczynniki cepstralne częstotliwości Melowej (Mel-frequency cepstral coefficients - MFCC), które są stosowane w zadaniach związanych z przetwarzaniem mowy i dźwięku.
3. **Budowa i trening modelu:**
    - Model sieci neuronowej jest budowany przy użyciu biblioteki tensorflow.
    - Model jest trenowany, a następnie testowany, aby ocenić jego wydajność.

4.1 Ładowanie i augmentacja danych

Ze względu na nierównomierną ilość pobranych z bazy plików, z każdej klasy wybierane są losowo 3962 pliki. Ponieważ trudno wyuczyć sieć z zadowalającą dokładnością przy takiej ilości danych, przeprowadzany jest proces augmentacji, polegający na utworzeniu kopii plików, które na całej długości przestrojono zakresie -1:1 (co oznacza obniżenie i podwyższenie częstotliwości o pół tonu i cały ton w każdą ze stron). W ten sposób uzyskiwanych jest pięciokrotnie więcej próbek do uczenia. Do realizacji augmentacji użyto funkcji *librosa.effects.pitch_shift*. Zbiór danych jest dzielony na zbiory uczący i testowy w proporcji 80% uczenie, 20% test.

4.2 Ekstrakcja cech

Zasadniczym etapem przygotowania danych jest wydobycie charakterystycznych cech poszczególnych instrumentów z nagrań. Te cechy zawarte są w spektrogramach. Ze względu na to, że muzyka tworzona jest w zakresie częstotliwości słyszalnych dla człowieka, spektrogramy „przestraja” się według skali melowej (*mel* od *melody*), która bazuje na modelach psychoakustycznych, aby ułożyć charakterystykę dźwięku tak, jak słyszy ją ludzkie ucho. Z tak utworzonego spektrogramu oblicza się współczynniki cepstralne częstotliwości melowej (mel-frequency cepstral coefficients - MFCC), które są składowymi spektrogramu melowego. W praktyce MFCC oblicza się bezpośrednio z nagrania funkcją pakietu librosa *librosa.feature.mfcc*.

4.3 Budowa, kompilacja i uczenie modelu

4.3.1 Budowa modelu

Model opiera się na konwolucyjnej sieci neuronowej (CNN). CNN są szczególnie efektywne w przetwarzaniu danych, które mają powtarzalną strukturę, takich jak obrazy czy sygnały audio. Poniżej omówiono szczegółową strukturę modelu, w tym wyjaśnienie poszczególnych warstw oraz ich roli.

- Warstwa Wejściowa (Conv1D(64, 3, activation='relu', input_shape=(20, 20))):

Kształt Wejściowy: (20,20)

Cel: Przyjęcie danych wejściowych w formie przekształconych MFCC. Wymiary danych wejściowych oznaczają, że przekazywanych jest 20 próbek, z których każda zawiera 20 współczynników. Warstwa zawiera 64 filtry konwolucyjne o rozmiarze 3. Filtry konwolucyjne przesuwają się wzdłuż osi czasu, co pozwala na wykrywanie cech takich jak krawędzie, zmiany w amplitudzie dźwięku itp. Aktywacja ReLU (Rectified Linear Unit) wprowadza nieliniowość, co umożliwia sieci naukę bardziej złożonych reprezentacji.

- Pierwsza Warstwa Poolingowa (MaxPooling1D(2)):

Ta warstwa wykonuje operację max pooling z rozmiarem okna 2.

Cel: Redukcja wymiarów danych poprzez pobieranie maksymalnej wartości z każdego okna o rozmiarze 2, zmniejszenie liczby parametrów i obciążenia obliczeniowego. Warstwa wprowadza również zmniejszenie wrażliwości sieci na przesunięcia sygnału.

- Druga Warstwa Konwolucyjna (Conv1D(128, 3, activation='relu')):

Ta warstwa zawiera 128 filtrów konwolucyjnych o rozmiarze 3.

Cel: Uczenie się bardziej złożonych i wyższych poziomów cech na podstawie wyjść z poprzednich warstw. Zwiększenie liczby filtrów pozwala na wykrycie większej ilości i różnorodności cech.

- Druga Warstwa Poolingowa
- Warstwa Flatten

Przekształca wielowymiarowe dane wyjściowe z warstw konwolucyjnych i poolingowych w jednowymiarowy wektor.

Cel: Umożliwienie połączenia danych z warstwami gęstymi (Dense). Przygotowanie danych do pełnego połączenia w warstwach Dense, gdzie każdy neuron w warstwie jest połączony z każdym neuronem w warstwie poprzedzającej.

- Pierwsza Warstwa Gęsta (Dense(256, activation='relu', kernel_regularizer=l2(0.001))):

  Ta warstwa zawiera 256 neuronów.

  Cel: Uczenie się globalnych wzorców w danych. Regularyzacja L2 pomaga w zapobieganiu nadmiernemu dopasowaniu (overfitting) poprzez karanie dużych wag.

- Dropout(0.5)

  Dropout wyłącza losowo 50% neuronów podczas treningu.

Cel: Zapobieganie przeuczeniu poprzez zmuszanie sieci do uczenia się bardziej odpornej reprezentacji danych. Pomoc w generalizacji modelu.

- Druga Warstwa Gęsta Dense(128, activation='relu', kernel_regularizer=l2(0.001)):

Opis: Ta warstwa zawiera 128 neuronów.

Cel: Kontynuacja uczenia się globalnych wzorców. Regularyzacja L2 i aktywacja ReLU działają podobnie jak w pierwszej warstwie gęstej. Zmniejszenie liczby neuronów w porównaniu do poprzedniej warstwy, co pomaga w stopniowym przekształcaniu i redukcji wymiarów danych przed warstwą wyjściową.

- Dropout

- Warstwa Wyjściowa (Dense(5, activation='softmax'))

Ta warstwa zawiera 5 neuronów, odpowiadających liczbie klas (instrumentów muzycznych).

Cel: Przekształcenie wyjścia sieci w rozkład prawdopodobieństwa dla każdej z klas. Aktywacja softmax zapewnia, że suma wszystkich wyjść wynosi 1, co umożliwia interpretację wyników jako prawdopodobieństwa przynależności do poszczególnych klas.

Podsumowanie:

Model ten jest zaprojektowany w celu efektywnej ekstrakcji cech z sygnałów audio oraz ich klasyfikacji na różne instrumenty muzyczne. Zastosowanie warstw konwolucyjnych 1D i poolingowych pozwala na wykrywanie lokalnych wzorców i redukcję wymiarów danych. Warstwy gęste umożliwiają integrację i klasyfikację wyodrębnionych cech, a regularyzacja L2 i dropout pomagają w zapobieganiu przeuczeniu. Warstwa wyjściowa z aktywacją softmax przekształca wyniki w prawdopodobieństwa klas, co umożliwia łatwą interpretację wyników.

4.3.2 Kompilacja i uczenie modelu

Kompilacja:

- Optymalizator

Optymalizatorem zastosowanym w kompilacji modelu jest optymalizator Adam z niską szybkością uczenia (learning rate=0.00001). Wynika to z potrzeby dokładnego dostosowania wag modelu w trakcie uczenia. Niski learning rate jest szczególnie przydatny w przypadku problemów, gdzie dane mają wysoką złożoność i wymagają delikatniejszego dostosowania wag. Minimalizuje on funkcję straty poprzez skuteczne aktualizowanie wag modelu, co prowadzi do lepszej zdolności rozpoznawania różnych instrumentów muzycznych.

- Funkcja straty 'categorical_crossentropy':

Model ma za zadanie rozpoznawać wiele klas, dlatego zastosowano funkcję straty odpowiednią dla problemów klasyfikacji wieloklasowej. Ta funkcja mierzy odległość między rozkładem prawdopodobieństwa przewidywanym przez model a rzeczywistymi etykietami. Minimalizacja tej funkcji sprawi, że model będzie dokładniej dopasowany do rzeczywistych etykiet wyjściowych.

Uczenie:

- Funkcja Early Stopping z monitorem 'val_loss', patience=10, restore_best_weights=True:

Użycie Early Stopping pozwala na automatyczne zatrzymanie treningu, gdy funkcja straty na zbiorze walidacyjnym (val_loss) przestaje się poprawiać przez określoną liczbę epok (patience=10). Przywrócenie najlepszych wag modelu (restore_best_weights=True) pozwala zachować najbardziej wydajny model. Zapobiega to przetrenowaniu modelu i poprawa jego zdolności generalizacji, co jest kluczowe aby uniknąć nadmiernego dopasowania do danych treningowych.

- Batch size = 2:

Ustawienie małego rozmiaru batcha (mini-batch gradient descent) pozwala na dokładniejsze aktualizowanie wag modelu i potencjalnie poprawia stabilność uczenia. Dla złożonych danych takich jak dźwięki instrumentów muzycznych, małe batche mogą pomóc w uniknięciu zakleszczeń w minimum lokalnym. Celem jest poprawa skuteczności uczenia poprzez bardziej stabilną optymalizację wag modelu, co prowadzi do lepszych wyników w rozpoznawaniu instrumentów muzycznych.

Podsumowując, konfiguracja kompilacji i uczenia modelu sieci neuronowej do rozpoznawania instrumentów muzycznych jest dostosowana w celu zapewnienia precyzyjnego uczenia modelu, minimalizacji funkcji straty i efektywnego rozpoznawania wielu klas instrumentów, jednocześnie zapobiegając przetrenowaniu poprzez zastosowanie Early Stopping i odpowiedniego batch size.

5. Wyniki uczenia

Najlepsze wyniki uczenia, które udało się osiągnąć dostosowując poszczególne parametry:

```
Epoch 1/50
7924/7924 [==============================] - 44s 5ms/step - loss: 0.9447 - accuracy: 0.6535 - precision: 0.7942 - recall: 0.5051 - val_loss: 1.3452 - val_accuracy: 0.5399 - v
al_precision: 0.6453 - val_recall: 0.3945
Epoch 2/50
7924/7924 [==============================] - 41s 5ms/step - loss: 0.9419 - accuracy: 0.6554 - precision: 0.7950 - recall: 0.5064 - val_loss: 1.3648 - val_accuracy: 0.5422 - v
al_precision: 0.6415 - val_recall: 0.4033
Epoch 3/50
7924/7924 [==============================] - 39s 5ms/step - loss: 0.9329 - accuracy: 0.6574 - precision: 0.7923 - recall: 0.5104 - val_loss: 1.3701 - val_accuracy: 0.5379 - v
al_precision: 0.6426 - val_recall: 0.4043
Epoch 4/50
7924/7924 [==============================] - 39s 5ms/step - loss: 0.9247 - accuracy: 0.6617 - precision: 0.7917 - recall: 0.5133 - val_loss: 1.3904 - val_accuracy: 0.5379 - v
al_precision: 0.6354 - val_recall: 0.4081
Epoch 5/50
7924/7924 [==============================] - 39s 5ms/step - loss: 0.9177 - accuracy: 0.6641 - precision: 0.7951 - recall: 0.5186 - val_loss: 1.3904 - val_accuracy: 0.5416 - v
al_precision: 0.6357 - val_recall: 0.4061
Epoch 6/50
7924/7924 [==============================] - 39s 5ms/step - loss: 0.9083 - accuracy: 0.6694 - precision: 0.7999 - recall: 0.5252 - val_loss: 1.4097 - val_accuracy: 0.5419 - v
al_precision: 0.6292 - val_recall: 0.4094
Epoch 7/50
7924/7924 [==============================] - 39s 5ms/step - loss: 0.9015 - accuracy: 0.6731 - precision: 0.8023 - recall: 0.5308 - val_loss: 1.4198 - val_accuracy: 0.5401 - v
al_precision: 0.6275 - val_recall: 0.4137
Epoch 8/50
7924/7924 [==============================] - 40s 5ms/step - loss: 0.8902 - accuracy: 0.6764 - precision: 0.8028 - recall: 0.5367 - val_loss: 1.4280 - val_accuracy: 0.5374 - v
al_precision: 0.6334 - val_recall: 0.4200
Epoch 9/50
7924/7924 [==============================] - 39s 5ms/step - loss: 0.8834 - accuracy: 0.6796 - precision: 0.8061 - recall: 0.5451 - val_loss: 1.4721 - val_accuracy: 0.5414 - v
al_precision: 0.6228 - val_recall: 0.4359
Epoch 10/50
7924/7924 [==============================] - 39s 5ms/step - loss: 0.8781 - accuracy: 0.6799 - precision: 0.8038 - recall: 0.5440 - val_loss: 1.4469 - val_accuracy: 0.5369 - v
al_precision: 0.6263 - val_recall: 0.4218
Epoch 11/50
7924/7924 [==============================] - 39s 5ms/step - loss: 0.8666 - accuracy: 0.6873 - precision: 0.8098 - recall: 0.5545 - val_loss: 1.4656 - val_accuracy: 0.5326 - v
al_precision: 0.6229 - val_recall: 0.4240
```

Rys. 1 Wyniki uczenia modelu

Model notuje wzrost dokładności na zbiorze uczącym w kolejnych epokach, ale z góry można przewidzieć problemy z generalizacją na nieznanych danych z uwagi na spadek dokładności i precyzji walidacyjnej (val_accuracy, val_precision). Aby uniknąć nadmiernego przeuczenia model zatrzymuje uczenie na 11 epoce ze względu na funkcję Early Stopping.

6. Ocena modelu

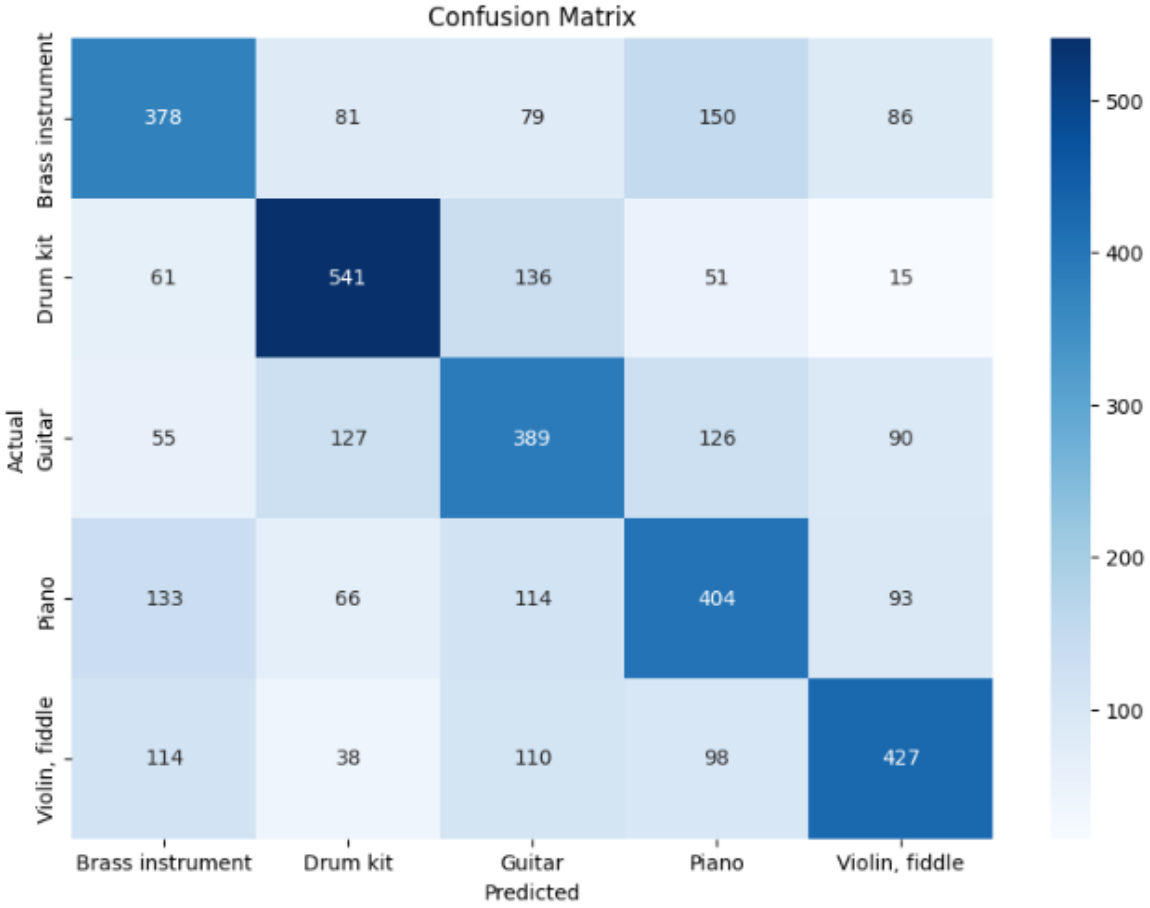
6.1 Wyniki dla wszystkich klas

Problemy z generalizacją są widoczne w wyraźnym spadku dokładności na zbiorze testowym:

```
124/124 [==============================] - 1s 4ms/step - loss: 1.3452 - accuracy: 0.5399 - precision: 0.6453 - recall: 0.3945
Test Loss: 1.3451589345932007
Test Accuracy: 0.5398788452148438
Test Precision: 0.6453344225883484
Test Recall: 0.39449772238731384
124/124 [==============================] - 0s 3ms/step
```

Rys. 2 Wyniki modelu na zbiorze testowym

Aby zobaczyć dlaczego tak się dzieje, wyrysowano macierz pomyłek:



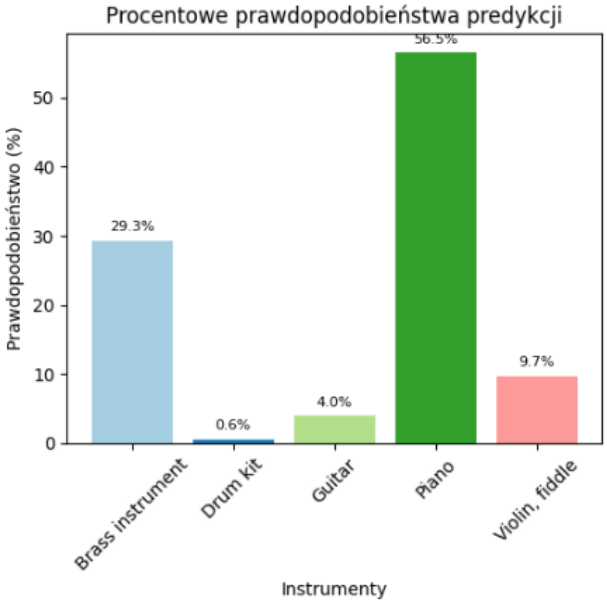
Rys. 3 Macierz pomyłek

Macierz pokazuje, że najlepiej rozpoznawaną klasą jest perkusja. Najprawdopodobniej wynika to z naturalnej rytmiczności tego instrumentu, do której jest przeznaczony – te same schematy powtarzają się wielokrotnie, co jest łatwo wykrywalne dla sieci konwolucyjnej. Podobną rytmiczność w swojej charakterystyce posiada gitara (dźwięk z gitary najczęściej wydobywany jest przez uderzanie strun w określonym, powtarzającym się schemacie, lub przez tzw. *arpeggio* – szybkie i wielokrotne szarpanie strun w tej samej kolejności zmieniając w międzyczasie wysokość dźwięku). Jest to dość prawdopodobny powód, dla którego tworzy się w macierzy charakterystyczny, nieco ciemniejszy od reszty kwadrat, właśnie między

klasami gitary i perkusji. Podobny kwadrat zależności można zauważyć dla trzech klas w prawym dolnym rogu – Gitara, instrumenty klawiszowe, oraz instrumenty smyczkowe. Trudno bez szczegółowej analizy wytłumaczyć te zależność, a pozostałe niedopasowania są dość nieregularne, więc aby je zbadać należy przeanalizować wyniki pojedynczych, specjalnie dobranych przypadków testowych.

6.2 Pojedyncze przypadki

6.2.1 Fortepian i skrzypce

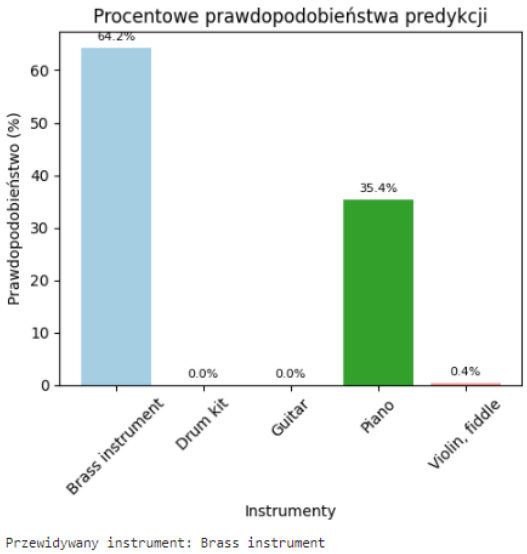



```
Przewidywany instrument: Piano
```

Rys. 4 Nagranie duetu fortepianowo-skrzypcowego, oraz procentowy rozkład prawdopodobieństwa predykcji modelu

Model bardzo dobrze rozpoznał grający fortepian, ale skrzypce będące jednym z dwóch grających instrumentów przewidział dopiero jako trzeci instrument, a trzykrotnie większe prawdopodobieństwo uzyskały instrumenty dętę, niewystępujące w nagraniu. Daje to sygnał, że model ma problem z rozpoznawaniem instrumentów smyczkowych, być może ze względu na dane w bazie.

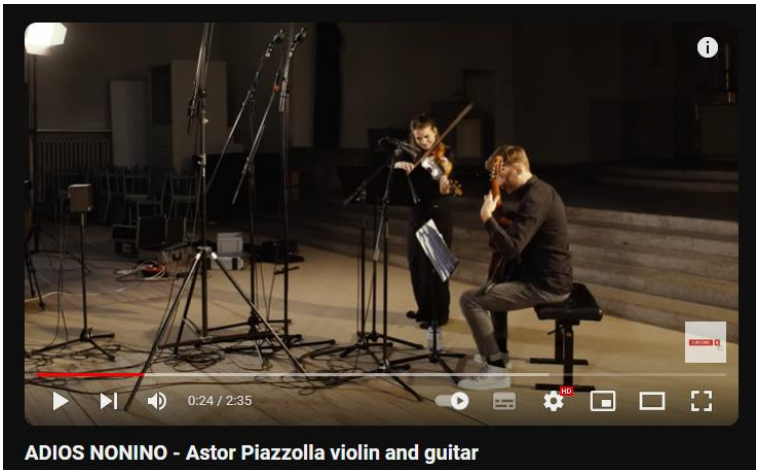
6.2.2 Fortepian i trąbka

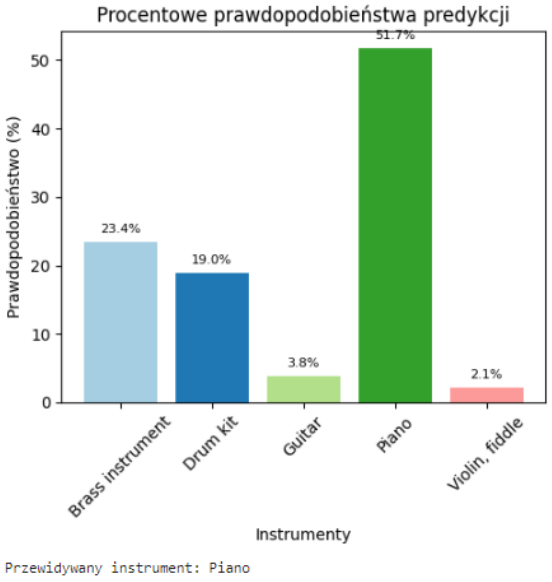


Rys. 5 Nagranie duetu fortepianu i trąbki, oraz procentowy rozkład prawdopodobieństwa predykcji modelu

Model bardzo dobrze rozpoznał trąbkę jako instrument dęty blaszany, oraz akompaniujący jej fortepian. Ze względu na to, że trąbka gra główną partię i jest w nagraniu na pierwszym planie, model przewidział ją z niemal dwukrotnie większym prawdopodobieństwem. Pozostałe instrumenty otrzymały prawdopodobieństwo 0,0%, z wyjątkiem instrumentów smyczkowych, dla których model przewidział 0,4% prawdopodobieństwa, co może być zarówno przypadkowe, jak i wskazywać na zależność między danymi w bazie.

6.2.3 Gitara i skrzypce

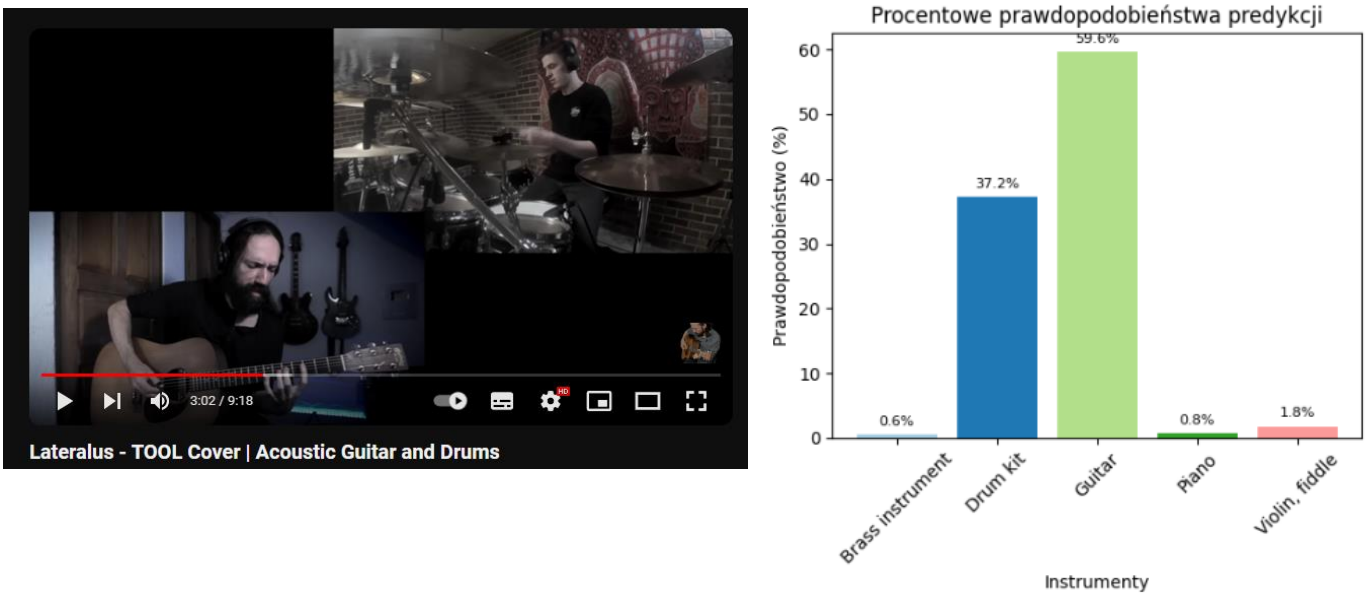




Rys. 6 Nagranie duetu gitarowo-skrzypcowego, oraz procentowy rozkład prawdopodobieństwa predykcji modelu

Pomimo nieobecności instrumentów klawiszowych w nagraniu, to właśnie dla nich model przewidział największe prawdopodobieństwo. Potwierdza to tezę o zależności między danymi, szczególnie między klasą skrzypiec i instrumentów klawiszowych. Wynika to z charakterystyki gry na skrzypcach, które rzadko występują jako instrument solowy, z uwagi na to, że ze skrzypiec wydobywa się zazwyczaj tylko jeden dźwięk jednocześnie. To stwarza potrzebę dołączenia do gry skrzypka instrumentu akompaniującego, którym w zdecydowanej większości przypadków jest fortepian, ze względu na komplementarność charakterystyk tych instrumentów. Ponieważ w bazie danych wiele nagrań instrumentów smyczkowych jest nagranych z akompaniującym fortepianem, pojawia się wewnętrzna zależność w danych, co utrudnia modelowi właściwe rozpoznawanie wzorców.

6.2.4 Gitara i perkusja



Rys. 7 Nagranie duetu gitarowy i perkusji, oraz procentowy rozkład prawdopodobieństwa predykcji modelu

Ten fragment nagrania został wybrany celowo w taki sposób, aby perksuja była na pierwszym planie, a gitara w tle grała pojedyncze dźwięki. Mimo tego gitara została przewidziana ze zdecydowanie większym prawdopodobieństwem. Ten przypadek, jak i wspominany wcześniej charakterystyczny kwadrat na macierzy pomyłek (Rys. 3) również wskazują na jakiś rodzaj zależności między danymi. Po przejrzeniu bazy okazuje się, że w wielu nagraniach z klasy gitary gra również zestaw perkusyjny. W klasie perkusji natomiast również występują nagrania z grającą gitarą, natomiast rytmiczna charakterystyka perkusji pozwala sieci konwolucyjnej na dość dobre rozpoznanie jej, co objawia się najciemniejszym polem dla tej klasy w macierzy pomyłek.

7. Podsumowanie

Głównym błędem popełnionym podczas projektowania modelu, było nieprzefiltrowanie plików podczas pobierania klas, co spowodowało wewnętrzne zależności w bazie danych (szczególnie między klasami instrumentów smyczkowych i fortepianu, oraz gitary i instrumentów perkusyjnych). Należało zastosować ten sam mechanizm, który został użyty przy pobierania klasy instrumentów klawiszowych. Podczas pobierania plików z tej klasy przekazano skryptowi identyfikator klasy organy, który jeśli również był przypisany do danego nagrania, miał odrzucać takie nagranie jako należące do zbioru danych, które miały zostać pobrane. Powodem zastosowania tego mechanizmu był fakt, że mimo, że organy należą do rodziny instrumentów klawiszowych, ich charakterystyka dźwięku zdecydowanie odbiega, od fortepianów, pianin i keyboardów. Trudno było natomiast przewidzieć zależności klasowe w instrumentach wymagających akompaniamanetu. Wymagałoby to szczegółowego rozeznania w bazie, co wydłużyłoby czas projektowania. Największą trudnością napotkaną przy projektowaniu modelu był utrudniony dostęp do danych z bazy, co skutkowało dużą iloscią czasu poświęconą na tworzenie skryptu pobierającego nagrania. Myślę, że z wiedzą nabytą podczas tworzenia tego projektu udałoby mi się stworzyć skuteczniejszy model poprzez wykluczanie zależności międzyklasowych i skupienie się na dostrojeniu parametrów modelu.