Głębokie uczenie i inteligencja obliczeniowa
Automatyka i Robotyka II stopień
2023/2024

Klasyfikacja gatunków muzyki z wykorzystaniem modeli konwolucyjnych sieci neuronowych

Skład zespołu

Imię i nazwisko	Obowiązki
Łukasz Gakan	Tworzenie dokumentacji, testowanie modeli i szukanie zbiorów danych testowych
Piotr Mamos	Model MFCC i MFCC na STFT (własna sieć), pomoc przy testowaniu, pomoc przy łączeniu endpointów
Dawid Maziarski	Tworzenie dokumentacji, przygotowanie własnej małej bazy utworów do testów praktycznych, analiza działania wariantów modeli MFCC, “ekspertyza muzyczna”
Julia Nowak	Tworzenie GUI, łączenie endpointów i modeli z inputami graficznymi, testowanie GUI
Michał Rola	Stworzenie skryptu przetwarzającego dane z plików audio na melspektrogramy, stworzenie wzoru dokumentacji oraz zebranie literatury.
Jakub Szczypek	Tworzenie i formatowanie dokumentacji, aktywny udział w cotygodniowych spotkaniach
Szymon Szewczyk	Model na podstawie spektrogramu (transfer learning), trenowanie, walidacja (28 i 3 sekundowe kawałki), pomoc w debugowaniu oraz pisaniu dokumentacji
Łukasz Szyszka	Zarządzanie projektem, prowadzenie cotygodniowych spotkań projektowych, rejestrowanie postępów, przydzielanie zadań, konsultowanie projektu z prowadzącą, tworzenie i formatowanie dokumentacji
Adam Złocki	Połączenie backendu (z uwzględnieniem wyboru modelu) z GUI, redakcja i formatowanie dokumentacji

Opiekun projektu: Prof. Joanna Kwiecień

1. Wstęp i opis badanego problemu

Uczenie maszynowe jest jedną z najszybciej rozwijających się technologii według wielu źródeł, dla której można znaleźć zastosowanie we wszystkich dziedzinach życia.

Jedną z kluczowych zalet tej technologii jest zdolność do automatycznego wykrywania wzorców w danych treningowych, co pozwala na tworzenie precyzyjnych modeli prognostycznych. Przykłady zastosowań obejmują personalizację reklam na stronach internetowych, rekomendacje treści na platformach streamingowych oraz optymalizację tras w transporcie.

Niniejszy projekt skupia się na stworzeniu oraz porównaniu różnych modeli klasyfikatorów muzyki, wykorzystując w tym celu techniki sieci konwolucyjnych CNN (ang. Convolutional Neural Networks). Modele te przyjmują na wejściu spektrogramy oraz parametry MFCC (ang. Mel-Frequency Cepstral Coefficients) generowane z plików audio utworów, a na wyjściu dokonują klasyfikacji gatunku muzycznego prezentowanego na nagraniu.

Klasyfikacja gatunków muzyki to zadanie polegające na przyporządkowaniu utworu muzycznego do jednego z predefiniowanych gatunków muzycznych.
Jest to istotny problem w dziedzinie przetwarzania dźwięku i rozpoznawania wzorców, mający zastosowanie m.in. w platformach streamingowych, systemach rekomendacyjnych oraz archiwizacji i katalogowaniu zasobów muzycznych.

Problem klasyfikacji gatunków muzyki jest złożony ze względu na różnorodność i złożoność danych muzycznych. Różne gatunki mogą posiadać podobne cechy akustyczne, co sprawia, że rozróżnienie ich bywa trudne. Dodatkowo, utwory w ramach jednego gatunku mogą znacznie się różnić, co jeszcze bardziej komplikuje proces klasyfikacji.

2. Wykorzystane technologie i narzędzia

Narzędzia oraz biblioteki dostępne na rynku bardzo pomogły w przyśpieszeniu pracy nad projektem. Poniższy rozdział poświęcony został przybliżeniu wykorzystanych narzędzi oraz technik obróbki danych. Poruszony zostanie również temat technologii wykorzystanych przy uczeniu modelu.

2.1. Mel-Spektrogramy

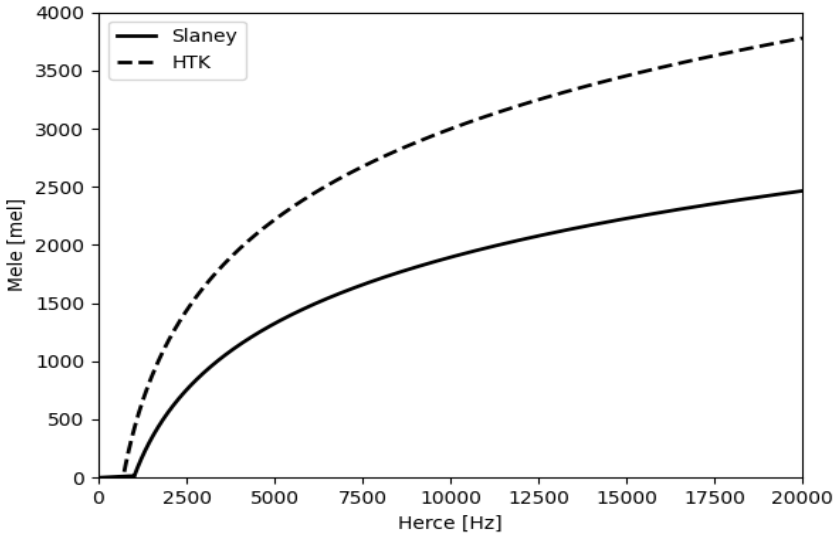
Mel-spektrogramy to spektrogramy posiadające skale Melową na osi OY zamiast liniowo rosnącej częstotliwości. Skala Melowa powstała dzięki badaniom polegającym na ocenie przez słuchaczy do jakiej częstotliwości należy dany ton. Jest więc to miara subiektywna i posiada wiele równań zależnie od interpretacji tych danych, jednak znacznie lepiej przedstawia w jaki sposób ludzkie ucho postrzega dźwięki [1].

Biblioteka Librosa pozwala na uzyskanie mel-spektrogramów z wykorzystaniem wzoru Slaney'a (2.1) oraz HTK (2.2) [2].

$$m(f) = \begin{cases} \frac{3f}{200}, & f < 1000 \\ 1527 \log_{6,4}\left(\frac{f}{1000}\right), & f \geq 1000 \end{cases}$$

$$m(f) = 2595 * \log_{10}\left(\frac{1+f}{700}\right)$$

Wykresy przedstawiające porównanie tych skali na osi częstotliwości wyskalowanej w Hercach przedstawia rysunek 1.



Rys. 1. Porównanie skali Slaney oraz HTK na osi częstotliwości wyskalowanej w Hercach.

2.2. MFCC

Mel-Frequency Cepstral Coefficients (MFCC) to jedne z najważniejszych cech wykorzystywanych w przetwarzaniu mowy i rozpoznawaniu dźwięków. Ich głównym zadaniem jest reprezentowanie sygnału dźwiękowego w sposób zgodny z ludzkim postrzeganiem dźwięku. Główne kroki w procesie obliczania MFCC obejmują:

1. Pre-emphasis (Preemfaza) – operacja polegająca na zwiększeniu energii wysokich częstotliwości. Pomaga to zrównoważyć spektrum sygnału, ponieważ w naturalnym dźwięku energia jest zwykle skoncentrowana w niższych częstotliwościach. Typowy wzór dla preemfazy to:

$$x'(n) = x(n) - \alpha \cdot x(n-1)$$

gdzie α jest współczynnikiem preemfazy, zazwyczaj równym 0.97.

2. Ramkowanie i Okienkowanie – proces dzielenia sygnału na krótkie fragmenty zwane ramkami (zazwyczaj 20-40 ms), które mogą się na siebie nakładać. Każda ramka jest następnie okienkowana (najczęściej za pomocą okna Hamminga) w celu zmniejszenia efektu przecieków:

$$w(n) = 0.54 - 0.46 \cdot \cos\left(\frac{2\pi n}{N-1}\right)$$

3. Transformata Fouriera (FFT) – każda okienkowana ramka jest przekształcana za pomocą szybkiej transformaty Fouriera (FFT), aby uzyskać widmo mocy:

$$P = |FFT(x)|^2$$

4. Skala Mel – Widmo mocy jest mapowane na skalę Mel za pomocą banku trójkątnych filtrów rozmieszczonych równomiernie na skali Mel. Skala Mel jest zdefiniowana jako:

$$M(f) = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right)$$

5. Logarytmowanie – Wartości amplitud z filtrów Mel są przekształcane na skalę logarytmiczną:

$$\log(E)$$

6. Dyskretna Transformata Kosinusowa (DCT) – Logarytmowane wartości są następnie przekształcane za pomocą DCT, aby uzyskać współczynniki cepstralne. Zazwyczaj tylko pierwsze 12-13 współczynników jest zachowywanych, ponieważ wyższe współczynniki są mniej istotne:

$$c_k = \sum_{n=1}^{N} \log(E_n) \cos\left(\frac{\pi k(2n-1)}{2N}\right)$$

Po obliczeniu MFCC dla każdego pliku dźwiękowego, współczynniki te są wykorzystywane jako wektory cech wejściowych dla modelu klasyfikacyjnego (np. sieci neuronowej), który rozpoznaje gatunek muzyczny.

2.3. Sieci neuronowe

Sztuczne sieci neuronowe (ANN - Artificial Neural Networks) są uproszczonym modelem mózgu ludzkiego, zdolnym do odwzorowania bardzo złożonych funkcji. Składają się z dużej liczby jednostek - neuronów, które posiadają zdolność przetwarzania informacji. Główne zalety sztucznych sieci neuronowych to:

- Adaptacja i samoorganizacja – zdolność do uczenia się i adaptacji na podstawie dostarczonych danych,
- Generalizacja – umiejętność uogólniania wiedzy na podstawie przykładów,
- Odporność na błędy – mała wrażliwość na uszkodzenia poszczególnych elementów,
- Szybkie przetwarzanie – możliwość równoległej pracy przy dużej liczbie neuronów.

Sztuczny neuron (AN) jest modelem biologicznego neuronu. Każdy neuron w sieci otrzymuje sygnały wejściowe, które są sumowane i przetwarzane za pomocą funkcji aktywacji. Jeśli suma ważona przekracza pewną wartość progową, neuron zostaje pobudzony i przekazuje sygnał dalej. Model matematyczny sztucznego neuronu można opisać wzorem:

$$y = \phi\left(\sum_{i=0}^{n} u_i \cdot w_i\right),$$

gdzie:

- u_i to sygnał wejściowy,
- w_i to waga przypisana do tego sygnału,
- ϕ to funkcja aktywacji.

Istnieje kilka różnych funckji aktywacji stosowanych w sztucznych neuronach. O to kilka z nich:

- Liniowa
- Sigmoidalna (logistyczna)
- Tangens hiperboliczny (tanh):
- ReLU (Rectified Linear Unit)

Sieci neuronowe składają się z wielu warstw. Można wyróżnić trzy podstawowe rodzaje warstw:

- Wejściowe – odbierają dane wejściowe.
- Ukryte – przetwarzają sygnały wydobywając cechy pośrednie.
- Wyjściowe – generują końcowy wynik przetwarzania.

Proces uczenia sieci neuronowych polega na automatycznym doborze wag na podstawie zbioru uczącego. Proces ten obejmuje następujące kroki:

- Inicjacja – losowe przypisanie wag początkowych.
- Propagacja sygnału – obliczanie wartości wyjściowych dla każdej warstwy.

- Obliczanie błędu – porównanie wartości wyjściowej z wartością wzorcową i wyznaczenie błędu.
- Propagacja wsteczna – korekcja wag za pomocą metod gradientowych, aby minimalizować błąd.

Rodzaje sieci neuronowych obejmują:

- Konwolucyjne Sieci Neuronowe (CNN): Wykorzystywane głównie w rozpoznawaniu obrazów. Struktura sieci obejmuje warstwy konwolucyjne, które ekstraktują cechy obrazu, oraz warstwy pooling, które redukują wymiarowość.
- Rekurencyjne Sieci Neuronowe (RNN): Idealne do przetwarzania danych sekwencyjnych, takich jak tekst czy sygnały czasowe. Modele takie jak LSTM (Long Short-Term Memory) rozwiązują problem znikającego gradientu.
- Generatywne Sieci Adwersarialne (GAN): Składają się z dwóch rywalizujących sieci - generatora i dyskryminatora, używane do generowania realistycznych danych.

W projekcie wykorzystano sieci Konwolucyjne Sieci Neuronowe (CNN). Poniżej opisano strukturę oraz działanie tych sieci:

1. Warstwa Konwolucyjna.
   Warstwy konwolucyjne są podstawowym elementem CNN. Każda warstwa zawiera zestaw filtrów (jąder), które przesuwają się po całym obrazie, wykonując operacje splotu. Filtry te pozwalają na wykrywanie różnych cech obrazu, takich jak krawędzie, tekstury czy bardziej złożone struktury. Operacja konwolucji można opisać matematycznie jako:

   $$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau,$$

   gdzie *f* to wejście, a g to filtr.

2. Warstwa Pooling
   Warstwy pooling (np. max-pooling) redukują wymiarowość danych, co zmniejsza liczbę parametrów i obciążenie obliczeniowe sieci. Pooling polega na wybieraniu maksymalnej (lub średniej) wartości w zadanym obszarze. Max-pooling można przedstawić wzorem:

   $$y = \max(x_1, x_2, \ldots, x_n)$$

   gdzie x_i to wartości w danym obszarze.

3. Warstwa Aktywacji
   Warstwy aktywacji wprowadzają nieliniowość do modelu. Najczęściej stosowaną funkcją aktywacji w CNN jest ReLU (Rectified Linear Unit), która przekształca wejście x według wzoru:

   $$ReLU(x) = \max(0, x)$$

4. Warstwa Pełnego połączenia
   Warstwy te działają podobnie do tradycyjnych warstw w sztucznych sieciach neuronowych, gdzie każdy neuron jest połączony z każdym neuronem w poprzedniej warstwie. Służą one do końcowej klasyfikacji po przetworzeniu cech przez warstwy konwolucyjne i pooling.

Sieci neuronowe mogą być wykorzystywane w wielu dziedzinach nauki, ale także codziennego życia. Możemy spotykać się z nimi na każdym kroku nie będąc nawet tego świadomi. O to kilka przykładów zastosowań sieci neuronowych:

- Rozpoznawanie obrazów i obiektów,
- Synteza mowy na podstawie tekstu,
- Rozpoznawanie języka i automatyczne tłumaczenie,
- Sterowanie robotami i samochodami autonomicznymi,
- Generowanie i synteza sygnałów,
- Chatboty i wiele innych.

Podsumowując, sieci neuronowe są potężnym narzędziem w dziedzinie sztucznej inteligencji, zdolnym do rozwiązywania złożonych problemów dzięki swojej zdolności do uczenia się i adaptacji. Ich wszechstronność i skuteczność sprawiają, że znajdują zastosowanie w wielu różnych dziedzinach, od analizy obrazów po przetwarzanie języka naturalnego.

2.4. Zbiór danych

Wybrana baza danych (GTZAN Dataset - Music Genre Classification) pochodzi ze strony kaggle.com. GTZAN to jedna z najbardziej znanych baz danych używanych do klasyfikacji gatunków muzycznych. Składa się z 1000 utworów, z których każdy trwa 30 sekund. Utwory są podzielone na 10 różnych gatunków muzycznych: blues, klasyczna, country, disco, hip-hop, jazz, metal, pop, reggae i rock. Każdy gatunek jest reprezentowany przez 100 utworów. Poniżej przedstawiono kilka powodów, dla których wybrano bazę danych GTZAN:

1. **Popularność i Szerokie Zastosowanie:**

   GTZAN jest powszechnie używany w literaturze naukowej i projektach badawczych, co ułatwia porównywanie wyników naszego modelu z wynikami innych badaczy. Dzięki szerokiemu zastosowaniu tej bazy możemy korzystać z istniejących badań i benchmarków.

2. **Struktura i Przystępność:**

   Baza danych jest dobrze zorganizowana, a jej rozmiar jest wystarczający do przeprowadzenia różnorodnych eksperymentów. 1000 utworów o długości 30 sekund zapewnia zrównoważoną próbkę, która pozwala na efektywne trenowanie i testowanie modeli bez konieczności przetwarzania ogromnych ilości danych.

3. **Różnorodność Gatunków:**

   GTZAN oferuje szeroką gamę gatunków muzycznych, co jest kluczowe dla rozwijania modeli klasyfikacji, które mogą być stosowane w różnych kontekstach muzycznych. Dzięki reprezentacji takich gatunków jak blues, klasyczna, country, disco, hip-hop, jazz, metal, pop, reggae i rock, możemy trenować modele na zróżnicowanych danych.

4. **Dostępność i Dokumentacja:**

   Dane są łatwo dostępne na platformie Kaggle, co ułatwia ich pobranie i integrację z naszymi narzędziami analitycznymi. Dodatkowo, dobra dokumentacja i wsparcie społeczności sprawiają, że praca z tą bazą danych jest bardziej efektywna.

5. **Uznane Standardy:**

   Jako że GTZAN jest uznawanym standardem w dziedzinie klasyfikacji muzyki, wykorzystanie tej bazy danych w naszym projekcie dodaje wiarygodności naszym badaniom i wynikom. Użycie standardowej bazy danych pozwala na przejrzyste i zrozumiałe przedstawienie wyników.

Każdy plik audio w bazie danych GTZAN został przetworzony na spektrogram, co jest wizualną reprezentacją sygnału audio w dziedzinie czasu i częstotliwości. Spektrogramy te są używane jako dane wejściowe do modeli uczenia maszynowego, co umożliwia analizę i klasyfikację muzyki na podstawie jej cech dźwiękowych. Dzięki temu możliwe jest wykorzystanie zaawansowanych technik przetwarzania sygnałów i głębokiego uczenia, aby zwiększyć dokładność klasyfikacji. Zbiór danych został podzielony w proporcjach:

- 70% - zbiór treningowy
- 15% - zbiór walidacyjny
- 15% - zbiór testowy

Każdy z powyższych zbiorów zawiera poszczególne gatunki muzyczne w równych proporcjach.

2.5 Python

Język programowania Python pozwolił na szybką oraz względnie prostą implementację kodu. Do stworzenia niniejszego projektu wykorzystano wersję 3.11. W tabeli 3.1 znajdują się informacje na temat wykorzystanych bibliotek.

Tabela 1. Użyte biblioteki wraz z opisami oraz informacjami o ich zastosowaniach w projekcie.

Nazwa biblioteki	**Wersja**	**Opis i zastosowanie w projekcie dyplomowym**
Keras	2.10.0	Głównym zadaniem tej biblioteki jest pozwolenie użytkownikowi na proste tworzenie modelów uczenia maszynowego i do tego została wykorzystana w tym projekcie [3].
Librosa	0.10.1	Biblioteka ta służy do analizy plików audio. Zapewnia również elementy niezbędne do tworzenia systemów wyszukiwania informacji muzycznych [4]. W tym projekcie dyplomowym została wykorzystana do wczytywania plików audio oraz przetwarzaniu ich na mel-spektrogramy.
Matplotlib	3.8.0	Biblioteka ta pozwala na szybkie i łatwe tworzenie wizualizacji danych [5]. Pomogła ona przy tworzeniu wizualizacji wykorzystanych w tym projekcie oraz przy zapisie i wczytywaniu mel-spektrogramów.
NumPy	1.26.3	Biblioteka ta umożliwia przeprowadzanie wszelkiego rodzaju działań na macierzach w prosty sposób, zarówno numerycznych, jak i ich przetwarzania (dodawanie wierszy/rzędów/wymiarów, zamiana ich miejscami). Pozwala również na zapis i wczytywanie macierzy, co również zostało wykorzystane w tym projekcie, w celu stworzenia zbioru danych [6].
scikit-learn	1.3.0	Biblioteka ta powstała z wykorzystaniem takich bibliotek, jak NumPy, SciPy oraz Matplotlib i pozwala na szybki dostęp do najpopularniejszych technik wykorzystywanych w uczeniu maszynowym [7], [8]. W trakcie prac biblioteka ta została wykorzystana przede wszystkim do wstępnego przetworzenia danych oraz oceny modelu.

3. Rozwiązanie

3.1. Model Szymka

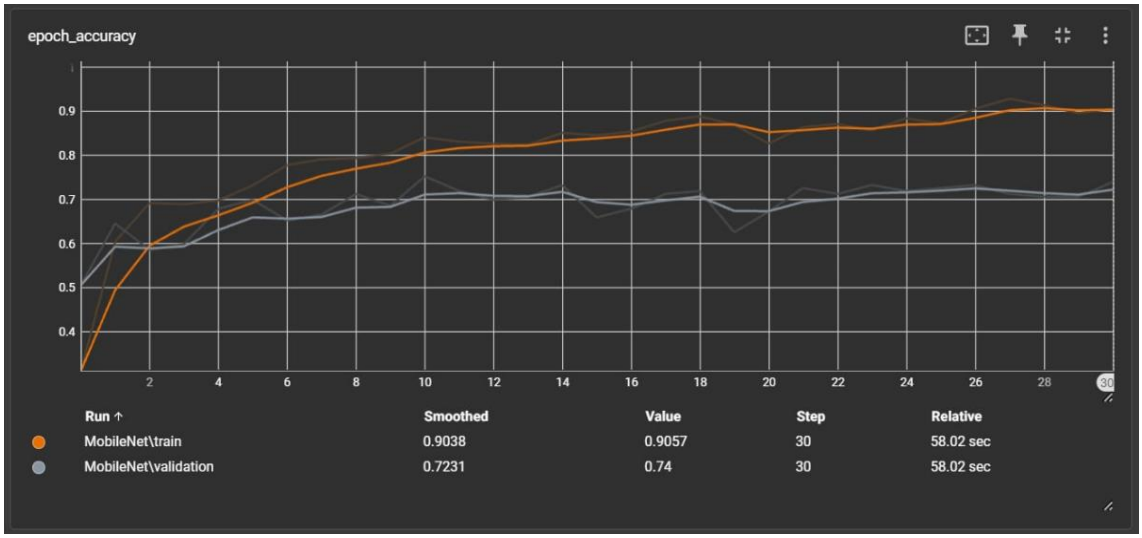
Model oparty na spektrogramach wykorzystuje architekturę MobileNet (wstępnie przetrenowaną z pomocą ImageNet) do klasyfikowania gatunków muzycznych dostosowaną do naszego problemu. Głównym powodem wykorzystania modelu MobileNet jest jego wydajność i dokładność, ponieważ jest on przeznaczony do mobilnych i wbudowanych aplikacji wizyjnych. Dzięki temu dobrze radzi sobie ze złożonością obliczeniową danych opartych na obrazach. Dodatkowo, dzięki wykorzystaniu "transfer learning'u" model może szybko dostosować się do nowo rozpoznanych wzorców na spektrogramach, które reprezentują natężenie poszczególnych częstotliwości sygnału audio w czasie. Pozwala to na dokonanie klasyfikacji muzyki na poszczególne gatunki z dużą dokładnością.

Architektura modelu:

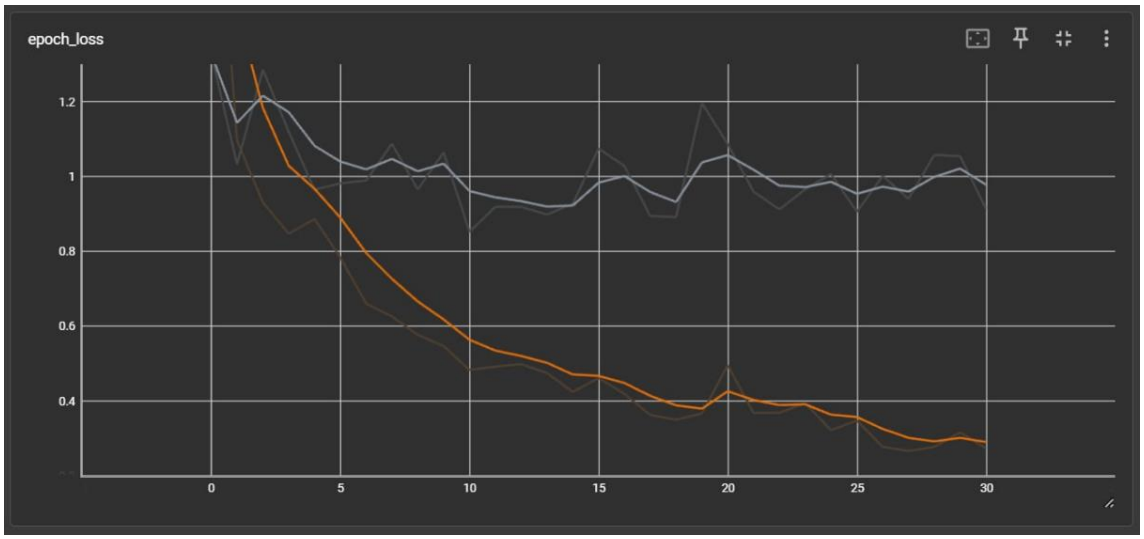
- Input Layer: Kształt wejściowy (2412, 128, 3) dla czarno-białych (w skali szarości) obrazów spektrogramu,
- Convolutional Layer 1: 32 filtry, jądro 3x3, aktywacja ReLU, 'same' padding,
- Pooling Layer 1: 2x2 Max pooling,
- Convolutional Layer 2: 64 filtry, jądro 3x3, aktywacja ReLU, 'same' padding,
- Pooling Layer 2: 2x2 Max pooling,
- Flatten Layer: Spłaszczenie danych wejściowych,
- Dense Layer 1: 128 neuronów, aktywacja ReLU,
- Output Layer: Aktywacja Softmax za pomocą 10 jednostek (dla 10 gatunków muzycznych),

Proces trenowania:

1. Wszystkie warstwy modelu bazowego były zablokowane, uczono jedynie warstwy wyjściowe:
    - Loss Function: sparse_categorical_crossentropy,
    - Optimizer: SGD z learning_rate=0.01, momentum=0.9,
    - Batch Size: 32,
    - Epochs: 150,
    - Early Stopping: Monitoruj „val_loss” z tolerancją 20 epok.



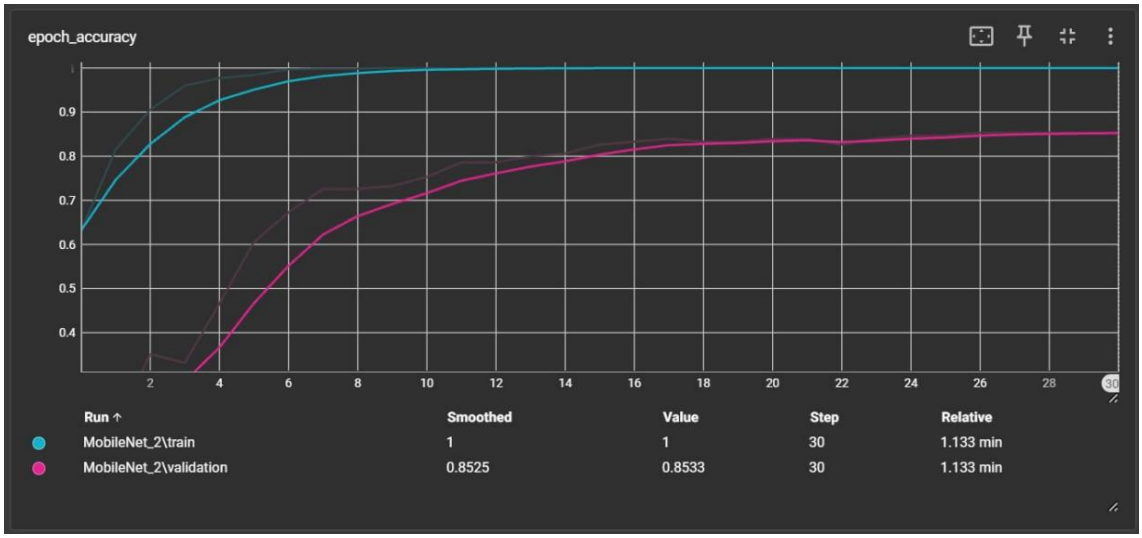
Rys. 2. Wykres uzyskanej dokładności modelu dla pierwszej próby.



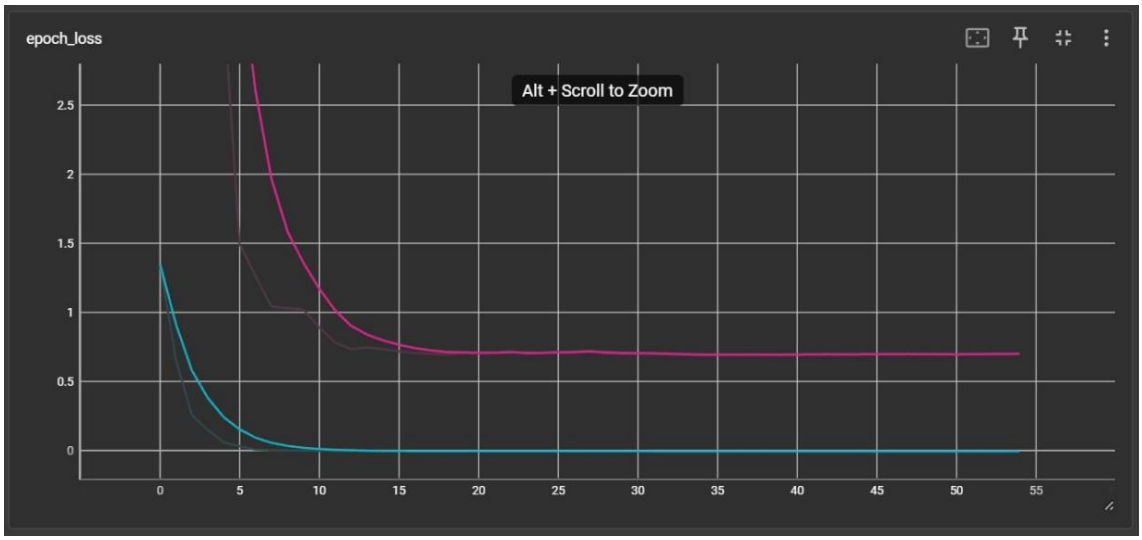
Rys. 3. Wykres uzyskanej wartości dla funkcji straty dla pierwszej próby.

Jak widać trenowanie modelu tym sposobem nie dało rezultatów, które można uznać a zadowalające. Walidacja modelu pokazała, że mimo stosunkowo dobrych wyników dla danych treningowych, wartość dla funkcji straty była bardzo wysoka w przypadku danych walidacyjnych, a dokładność także nie była nadmiernie wysoka.

2. Połowa warstw modelu bazowego była zablokowana
   - Loss Function: sparse_categorical_crossentropy,
   - Optimizer: SGD z learning_rate=0.01, momentum=0.9,
   - Batch Size: 32,
   - Epochs: 100,
   - Early Stopping: Monitoruj „val_loss” z tolerancją 20 epok.



Rys.4. Wykres uzyskanej dokładności dla drugiej próby.



Rys. 5. Wykres uzyskanej wartości dla funkcji straty dla drugiej próby.

Jak widać, dla drugiej próby uzyskane wyniki są już dużo lepsze (dokładność zbioru walidacyjnego na poziomie 0.8533 i wartość dla funkcji straty na poziomie około 0.65). Należy pamiętać, że te dane zostały uzyskane na zbiorze walidacyjnym, a nie testowym, ale ze względu na uzyskanie stosunkowo dobrych wyników wykorzystano tak przetrenowany model.

Ocena modelu:

Wynikiem wytrenowania sieci na 28-sekundowych fragmentach utworów, przy zachowaniu powyższych parametrów, było uzyskanie dla zbioru testowego dokładności na poziomie 0.88, zaś wartość dla funkcji straty wyniosła 0.5021. Ze względu na stosunkowo małą liczbę danych treningowych to właśnie architektura MobileNet pozwoliła na osiągnięcie wysokiej dokładności, przy zachowaniu akceptowalnie niskiej wartości dla funkcji straty.

Dla porównania przy wykorzystaniu architektury InceptionResNetV2 dokładność osiągnęła poziom 0.8243, zaś wartość funkcji straty 0.6206. O ile jeszcze dokładność nie spadła znacząco, o tyle bardzo wzrosła wartość dla funkcji straty, w porównaniu do architektury MobileNet. Kolejne testowane architektury (EfficientNetB7, DenseNet121, Xception) wykazały jeszcze gorsze wyniki. Jak zostało to wspomniane wyżej, aby zmienić tą tendencję, a więc aby architektury o bardziej skomplikowanej budowie zwracały lepsze wyniki, należałoby zmienić lub znacznie rozbudować zbiór danych treningowych. Jednakże na potrzeby tego projektu osiągnięta dokładność i wartość dla funkcji straty były w pełni zadowalające. Przy takich wynikach nie było także zbytniego zagrożenia przeuczenia modelu.

W celach porównawczych model trenowano także na krótszych (3-sekundowych) fragmentach utworów i co ciekawe, wyniki osiągnięte dla takich danych treningowych wydawałyby się lepsze (patrząc jedynie na wartości metryk). Wytrenowany dla takich parametrów model (bazowany na architekturze MobileNet) osiągnął bowiem wyniki odpowiednio: 0.9418 dla dokładności oraz 0.2735 dla funkcji straty. Te wartości są dużo lepsze w porównaniu do tych osiągniętych przez model wytrenowany na dłuższych odcinkach czasowych. Przyczyną takiego stanu rzeczy jest najprawdopodobniej fakt, że przy „pocięciu" dłuższych fragmentów na krótsze, tych drugich będzie naturalnie dużo więcej, a więc sieć będzie miała dużo większą bazę danych treningowych. To rozwiązanie nie zostało jednak wykorzystane z dwóch zasadniczych powodów:

- Przy takiej wysokiej dokładności możemy zbliżyć się do granicy, gdzie model będziemy uznawać za przeuczony. Co prawda wartość dla funkcji straty na to nie wskazuje, ale z drugiej strony nie zastosowano też dodatkowych środków ostrożności w celu uniknięcia przeuczenia modelu, poza podstawowymi funkcjonalnościami sieci MobileNet.
- Drugim, a zarazem dużo bardziej istotnym kryterium oceny była merytoryczność takiego rozwiązania. Przy 3-sekundowych odcinkach ciężko mówić o właściwym rozpoznawaniu gatunku muzycznego. Taki odcinek czasowy jest zwyczajnie za krótki, aby wykorzystać go do rzetelnego uczenia. Jest to raczej zgadywanie na podstawie kilku zarejestrowanych dźwięków. Dodatkowo, dużo zależałoby tutaj od tego jaki fragment utworu zostałby zarejestrowany. Stąd, takie rozwiązanie należy odrzucić, gdyż nie spełnia podstawowego warunku merytoryczności.

3.2. Model MFCC

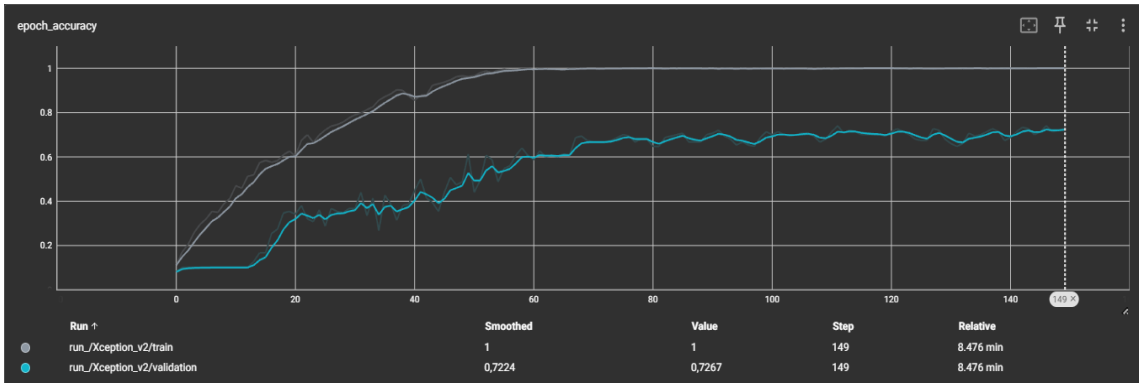
Model oparty na MFCC to custom'owa konwolucyjna sieć neuronowa zaprojektowana specjalnie do klasyfikacji gatunków muzycznych przy użyciu Mel-Frequency Cepstral Coefficients (MFCCs). MFCCs stanowią zwartą reprezentację sygnału audio, wychwytującą jego charakterystyczne aspekty brzmienia i barwy, które są kluczowe dla rozróżnienia poszczególnych gatunków. Model składa się z kilku warstw konwolucyjncyh i łączących, które pomagają we właściwym wyodrębnieniu odpowiednich cech oraz warstw gęstych (dense layers) służących do przeprowadzenia klasyfikacji. Takie podejście pozwala modelowi uczyć się skomplikowanych wzorców w ramach MFCC, dzięki czemu dobrze radzi sobie z rozpoznawaniem gatunków. Custom'owa konstrukcja zapewnia elastyczność i możliwość dostosowania się modelu do specyficznych (charakterystycznych) cech analizowanych danych audio. Zasadniczym powodem wykorzystania custom'owej konstrukcji jest fakt, że za pomocą MFCC wylicza się 13 miar, przez co kształt wejściowy trudno przerobić na kształt wykorzystywany przez standardowe architektury sieci. Jest to też powód dlaczego nie jest wykorzystany transfer learning.

Architektura modelu:

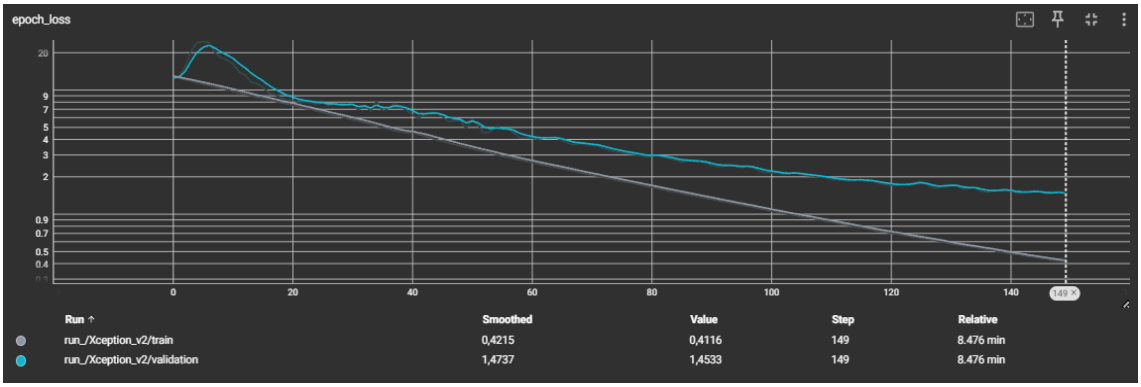
- Input Layer: Kształt wejściowy (6*n_mfcc, n_frames, 1), gdzie n_mfcc to liczba miar MFCC (=13), a n_frames to liczba ramek czasowych,
- Convolutional Layer 1: 32 filtry, jądro 3x3, aktywacja ReLU, 'same' padding,
- Pooling Layer 1: Max pooling 2x2,
- Convolutional Layer 2: 64 filtry, jądro 3x3, aktywacja ReLU, 'same' padding,
- Pooling Layer 2: Max pooling 2x2,
- Convolutional Layer 3: 128 filtrów, jądro 3x3, aktywacja ReLU, 'same' padding,
- Pooling Layer 3: Max pooling 2x2,
- Flatten Layer: spłaszcza dane wejściowe,
- Dense Layer 1: 128 neuronów, aktywacja ReLU,
- Dropout Layer: współczynnik porzucania wynoszący 0,5, aby zapobiec nadmiernemu dopasowaniu (przetrenowaniu),
- Dense Layer 2: 64 neurony, aktywacja ReLU,
- Output Layer: aktywacja Softmax z 10 jednostkami (dla 10 gatunków muzycznych).

Proces trenowania:

- Loss Functiony: sparse_categorical_crossentropy,
- Optimizer: SGD z learning_rate=0.01, momentum=0.9,
- Batch Size: 128,
- Epochs: 150,
- Early Stopping: Monitoruj „val_loss" z tolerancją 20 epok,
- Checkpointing: Zapisywanie najlepszej wersji modelu na podstawie utraty wartości walidacji.



Rys. 6. Wykres uzyskanej dokładności modelu opartego na MFCC.



Rys. 7. Wykres uzyskanej wartości funkcji straty modelu opartego na MFCC.

Ocena modelu:

Wynikiem wytrenowania sieci na 28-sekundowych fragmentach utworów, przy zachowaniu powyższych parametrów, było uzyskanie dla zbioru testowego dokładności na poziomie 0.72, wartość dla funkcji straty wyniosła 1.48. Model trenowano także na krótszych (3-sekundowych) fragmentach utworów i podobnie jak w przypadku pierwszego modelu, wyniki osiągnięte dla takich danych treningowych były lepsze w porównaniu do tych osiągniętych przez model wytrenowany na dłuższych odcinkach czasowych (patrząc jedynie na wartości metryk). Przyczyną tego ponownie była większa ilość danych treningowych (z uwagi na krótsze fragmenty). Podczas testów praktycznych (polegających na testowaniu za pomocą utworów spoza bazy danych) okazało się jednak, że model oparty na fragmentach 28-sekundowych radzi sobie trochę lepiej od modelu 3-sekundowego. Jest to spowodowane prawdopodobnie:

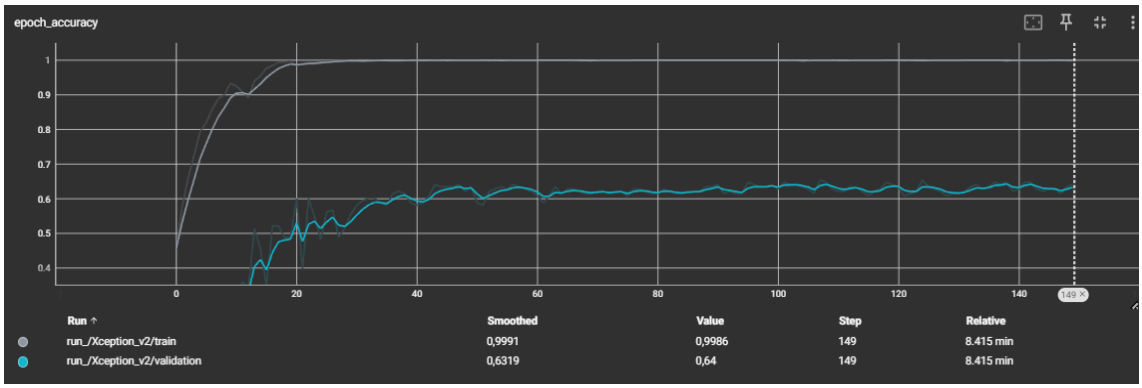
- Większymi informacjami kontekstowymi utworu muzycznego zawartymi w 28-sekundowych fragmentach (jak opisano w punkcie 4.1).
- Gorszym dopasowaniem do podobnych do siebie utworów z bazy danych. Utwory z bazy danych pochodzą z podobnego okresu czasu, są bardzo podobne do siebie i różnią się od współczesnych utworów reprezentujących te same gatunki muzyczne (szczególnie jeżeli artysta realizuje dany gatunek w eksperymentalny sposób). Gorsze dopasowanie do utworów z bazy może prowadzić do lepszych rezultatów w praktycznym zastosowaniu.

Podczas testów praktycznych okazało się również, że model:

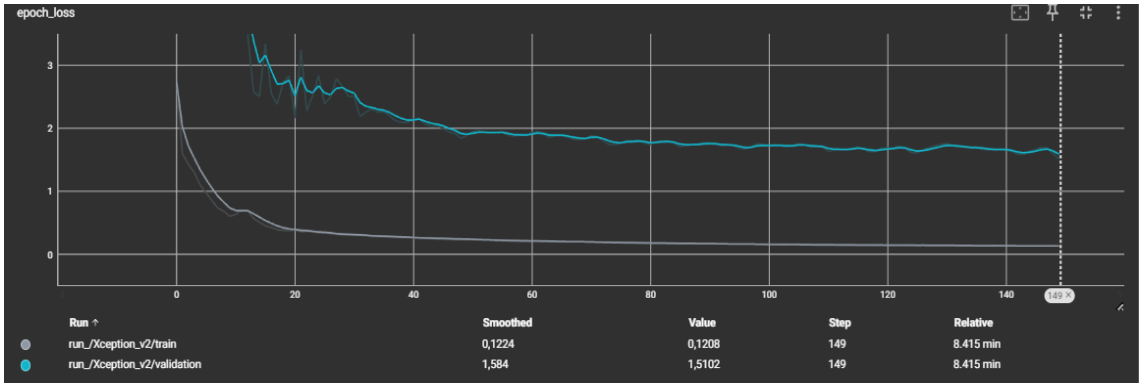
- Kompletnie nie radzi sobie z klasyfikacją współczesnego bluesa,
- Uznaje metal za rock mając duży problem z rozróżnieniem tych dwóch gatunków,
- Ma problemy z współczesną muzyką popularną klasyfikując ją do innych gatunków, do których dane utwory pop są podobne,
- Sklasyfikował jeden z najpopularniejszych utworów disco wszechczasów niepoprawnie (działając poprawnie dla współczesnego disco),
- Uznał jeden z utworów reggae za muzykę klasyczną,
- W pozostałych przypadkach mimo pomyłek działa całkiem dobrze.

Model wykorzystujący STFT:

Stworzono również drugi model oparty na MFCC, który zamiast spektrogramów jako dane wejściowe wykorzystuje STFT (Short Time Fourier Transform), czyli krótką transformatę Fouriera 28-sekundowych fragmentów utworu. Model ten był uczony przy zachowaniu tych samych parametrów co pierwszy model oparty na MFCC, ale w porównaniu osiągnął gorsze wyniki pod względem dokładności (0.64), oraz funkcji straty (1.51) na zbiorze walidacyjnym. Analiza modelu na utworach spoza bazy danych również wykazała, że jest on gorszy od oryginalnego modelu. Dokonywał on bowiem dużo większej liczby bezsensownych decyzji (takich jak uznawanie reggae za muzykę klasyczną, pop za blues itp). Jedyna zaleta modelu wykorzystującego STFT była umiejętność identyfikacji metalu, z czym oryginalny model sobie nie radził.



Rys. 8. Wykres uzyskanej dokładności modelu MFCC wykorzystującego STFT.



Rys. 9. Wykres uzyskanej wartości funkcji straty modelu MFCC wykorzystującego STFT.

Ostatecznie do klasyfikacji postanowiono użyć modelu wykorzystującego spektrogramy. Jeżeli jednak model sklasyfikuje dany utwór jako rock, to utwór ten jest dodatkowo klasyfikowany za pomocą modelu STFT, aby sprawdzić z dużo większą pewnością, czy jego gatunek to tak na prawdę metal.

4. Aplikacja

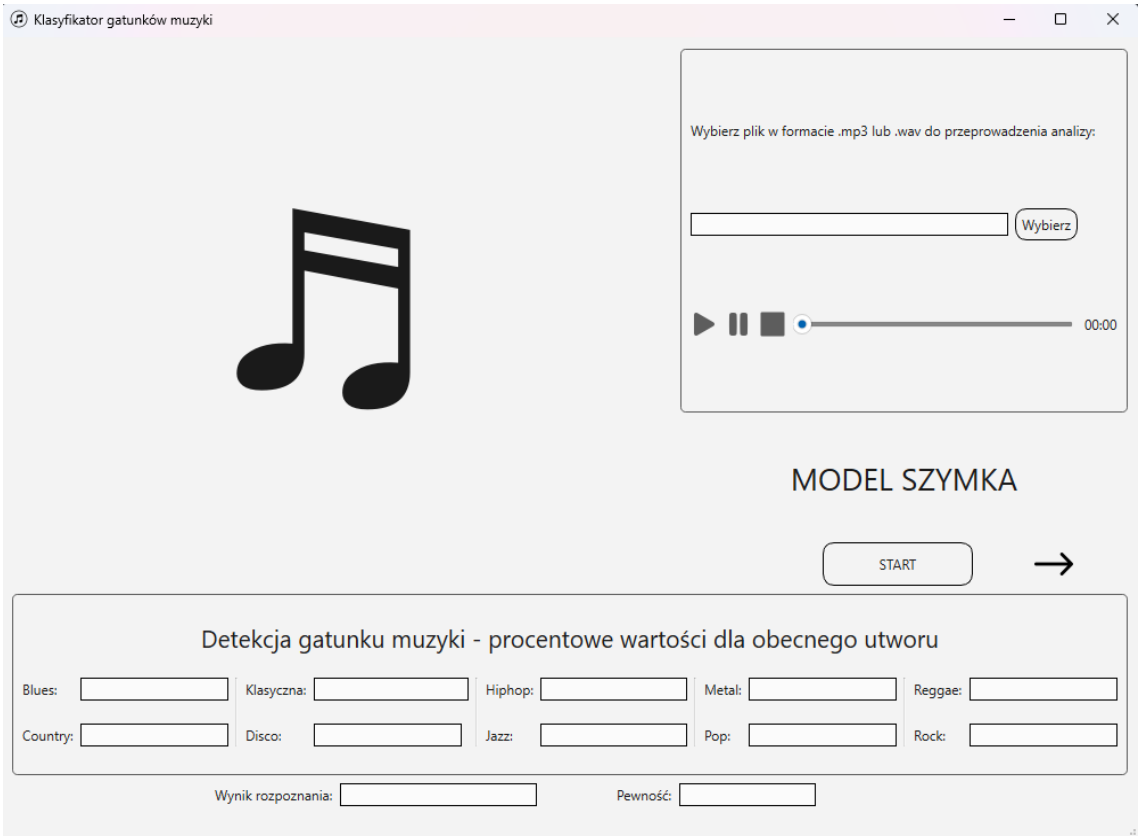
Do stworzenia aplikacji został wykorzystany język Python. Oprócz części odpowiedzialnej za funkcjonalność, utworzono również GUI, które w znacznym stopniu przyspiesza i ułatwia korzystanie z modelów.

Do stworzenia aplikacji zostały wykorzystane biblioteki:

- Matplotlib,
- Numpy,
- Pandas,
- PySide6.

Kod programu tworzony był w środowiskach Visual Studio Code oraz PyCharm, dlatego też są one zalecane do sprawnego uruchomienia aplikacji. GUI aplikacji powstało z pomocą programu Qt Creator.

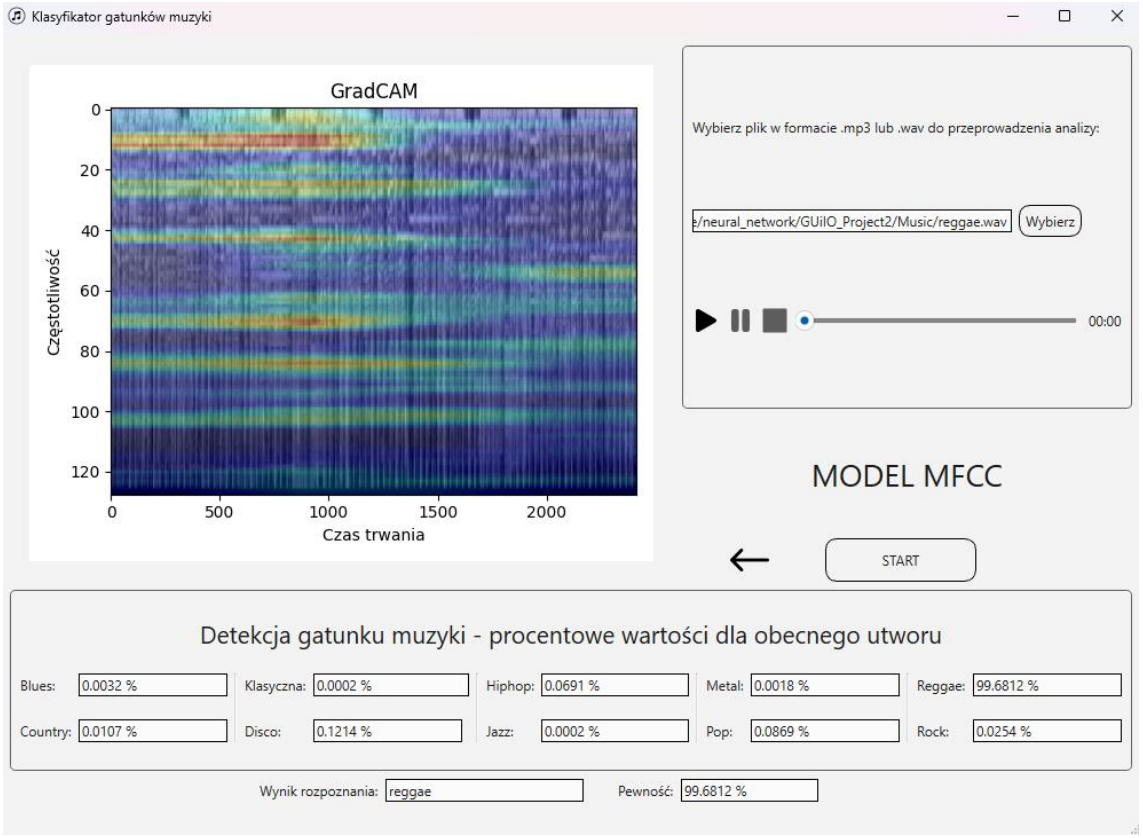
Do uruchomienia aplikacji niezbędna jest wersja Pythona co najmniej 3.10. Po zainstalowaniu wyżej wskazanych bibliotek, należy uruchomić plik *<nazwa_pliku>.py*, znajdujący się w folderze *GUI*. Po uruchomieniu programu poprzez konsolę, powinno ukazać się okno, widoczne na Rys. 10.



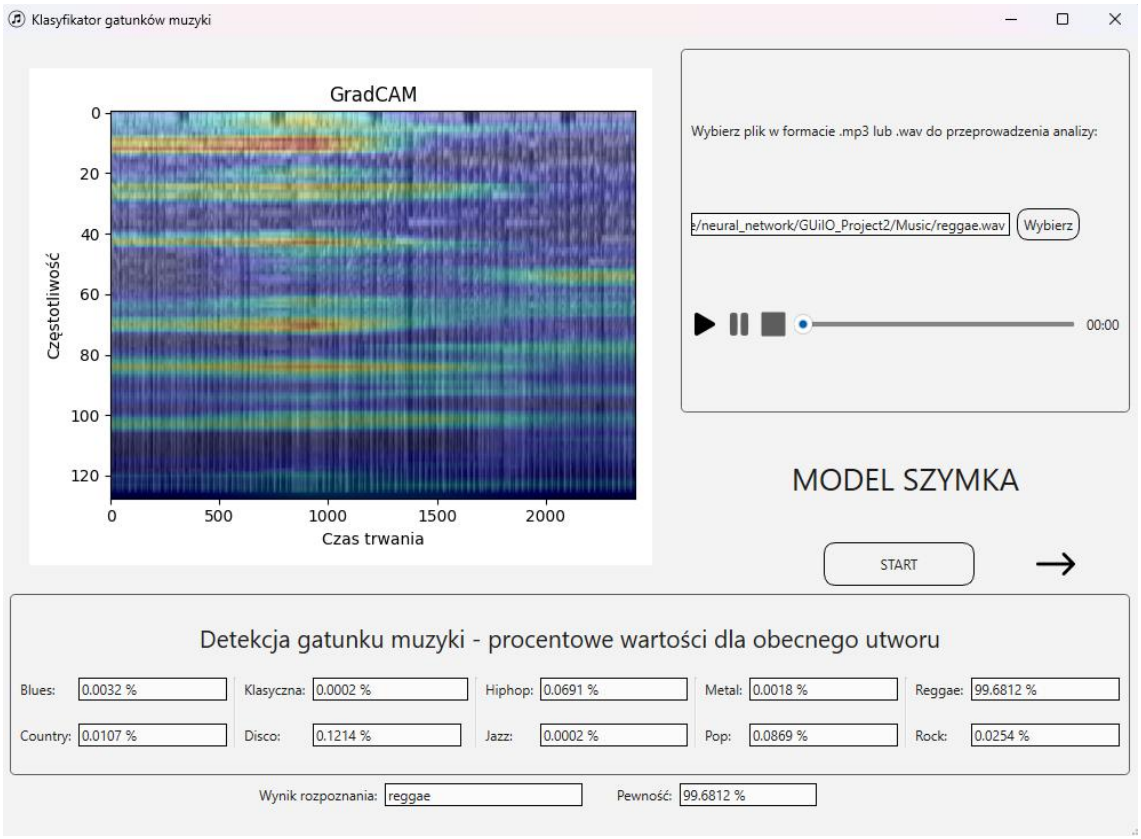
Rys. 10. Startowy widok aplikacji.

GUI podzielone jest na dwa widoki – część odpowiedzialną za model Szymka oraz część odpowiedzialną za model MFCC. Oba widoki posiadają ten sam interfejs. Prawa część okna aplikacji umożliwia wgranie pliku muzycznego w formacie .mp3 lub .wav, którego gatunek muzyczny zostanie poddany klasyfikacji przez oba modele. Wyniki umieszczone zostały w dolnej części interfejsu. Są to oceny wyrażone w procentach, mówiące, z jakim prawdopodobieństwem wgrany utwór jest danego gatunku oraz nazwa gatunku z najwyższą liczbą procentów. Aplikacja obsługuje obecnie 10 różnych gatunków widocznych na rys. 10. Warto o tym pamiętać przy ewentualnej próbie wgrania utworu muzycznego spoza obsługiwanej puli, ponieważ nie zostanie on sklasyfikowany poprawnie.

Po kliknięciu przycisku "START" w lewej części okna pojawia się spektrogram GradCAM, na którego podstawie model Szymka wykonał klasyfikację. Z powodu współdzielenia interfejsu przez oba modele, w celu sprawdzenia wyników drugiego modelu należy kliknąć jeden z przycisków znajdujących się po prawej i lewej stronie przycisku "START".



Rys. 11. Widok aplikacji dla modelu MFCC.



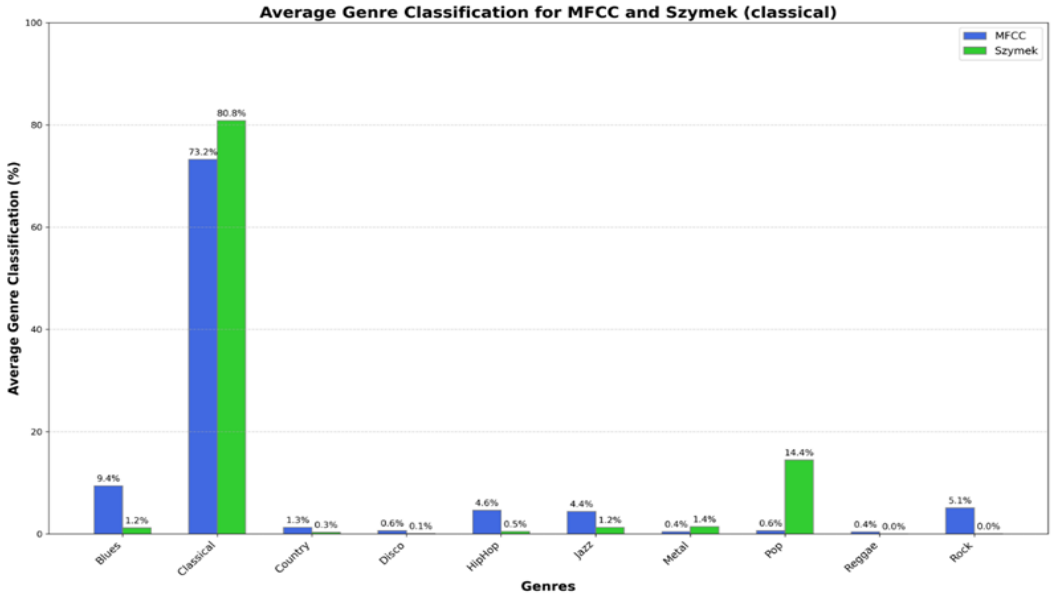
Rys. 12. Widok aplikacji dla modelu Szymka.

5. Testy, eksperymenty

Modele będące obiektem tej pracy zostały przetestowane na plikach muzycznych spoza zbioru wykorzystywanego do ich trenowania i wstępnego testowania. Wykorzystano do tego dane z popularnych zbiorów danych „The Million Song Dataset” [9] oraz „Ludwig Music Dataset” [10].

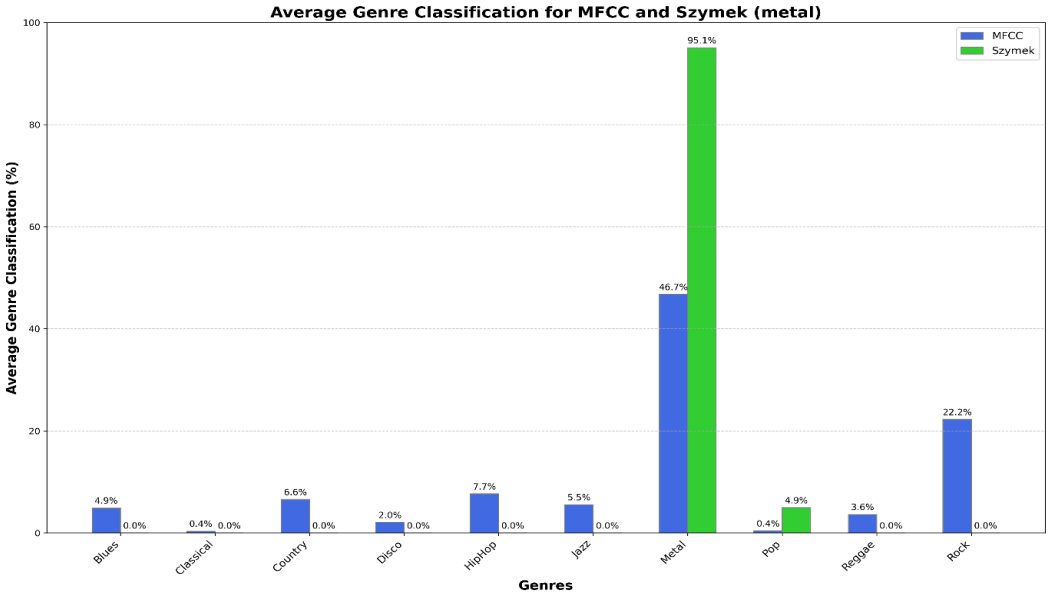
Dla każdego gatunku z tych zbiorów, który pokrywał się z listą klasyfikowanych przez modele, wybrano 20 przykładowych plików. Łącznie znaleziono 9 wspólnych gatunków (wszystkie poza disco). Pliki dla każdego gatunku podano na wejścia obu modeli. Wynikiem były pliki Excel zawierające klasyfikacje każdego zbioru testowego, wyrażone w procentowych przynależnościach do gatunków. Na podstawie zebranych wartości policzono średnie dla każdego 20-elementowego zbioru danego gatunku.

Stworzone na tej podstawie wykresy, widoczne w dalszej części rozdziału. Pokazują, że skuteczność stworzonych modeli różni się w dużym stopniu w zależności od klasyfikowanego gatunku muzyki. Co jednak ciekawe, prezentowane modele wytrenowane na tych samych danych wykazują dla niektórych gatunków duże różnice w skuteczności klasyfikacyjnej.

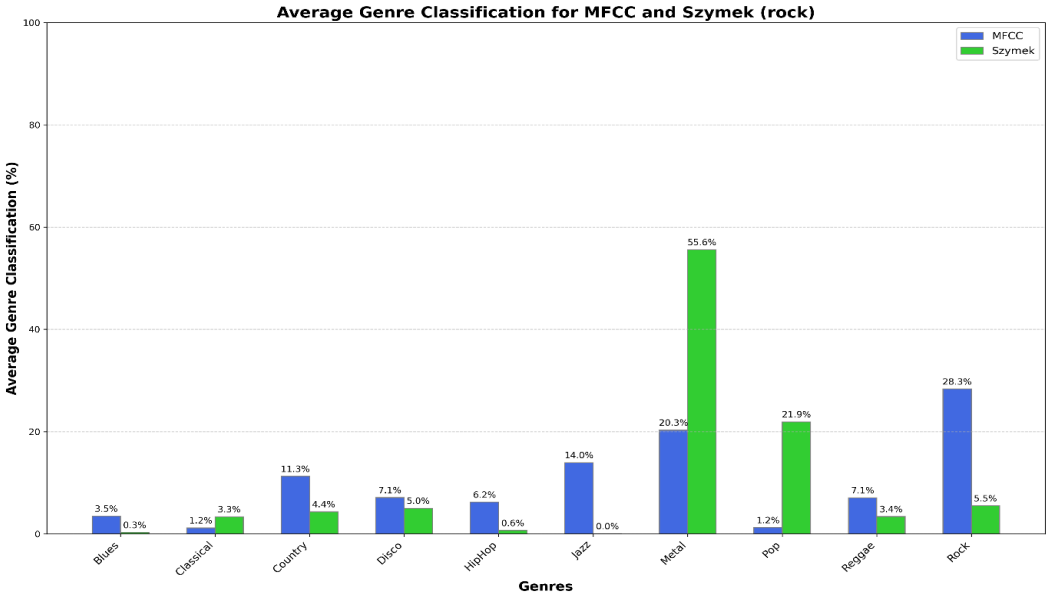


Rys. 13. Wykres procentowych wartości średnich klasyfikacji gatunków – classical.

Gatunkiem, z którym oba modele radzą sobie bardzo dobrze jest muzyka klasyczna. Jest najlepiej rozpoznawana przez modele klasyfikacyjne ze względu na unikalne i wyraziste cechy akustyczne, najwyższą jakość danych wejściowych spośród wykorzystywanych plików ora spójność wewnątrz gatunku.

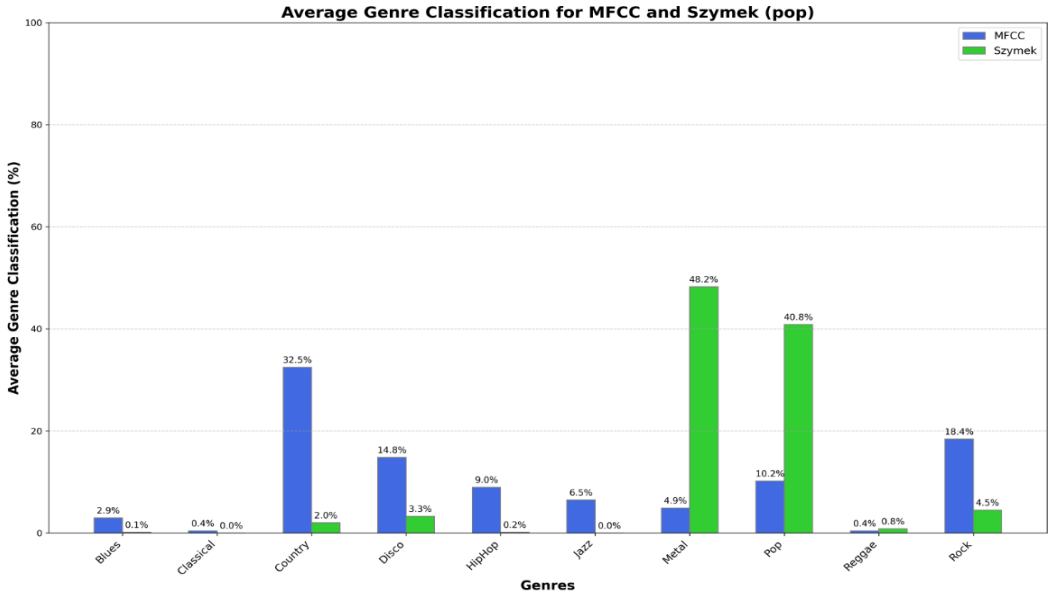


Rys. 14. Wykres procentowych wartości średnich klasyfikacji gatunków – metal.



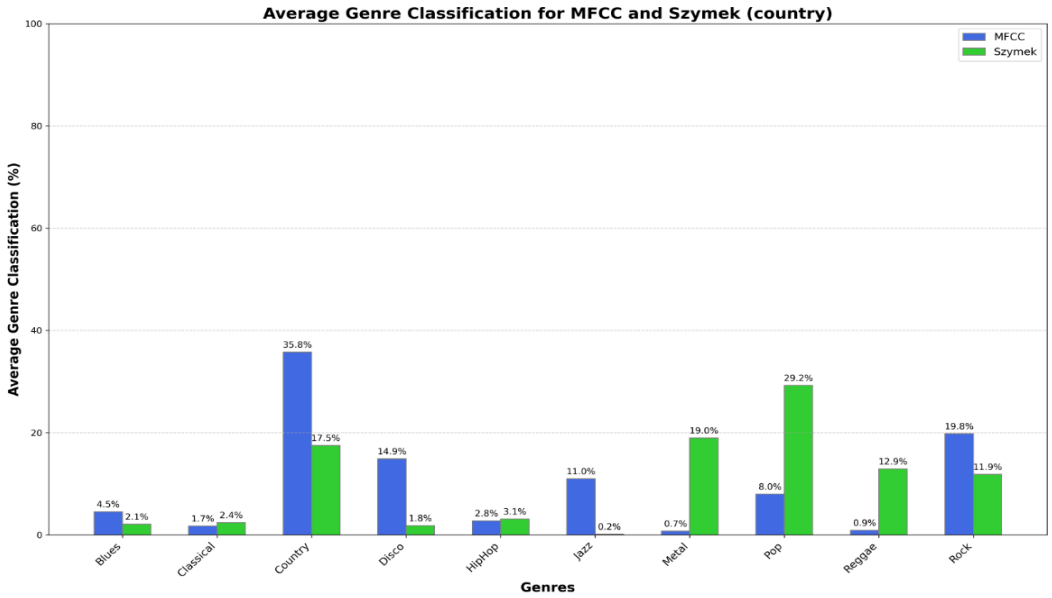
Rys. 15. Wykres procentowych wartości średnich klasyfikacji gatunków – rock.

Dla gatunków metal i rock można zauważyć, że model Szymka wykrywa, iż wgrany utwór należy do jednego z tych gatunków. Niestety, oba są traktowane jako metal. Model MFCC, analizując jedynie gatunek metal, radzi sobie znacznie gorzej. Jednak zestawiając gatunki metal i rock obok siebie, widać, że model jest w stanie je rozróżniać, co czyni go bardziej wiarygodnym. Skuteczność obu modeli była w dużej mierze zależna od wgrywanych do nich plików muzycznych tychże gatunków.



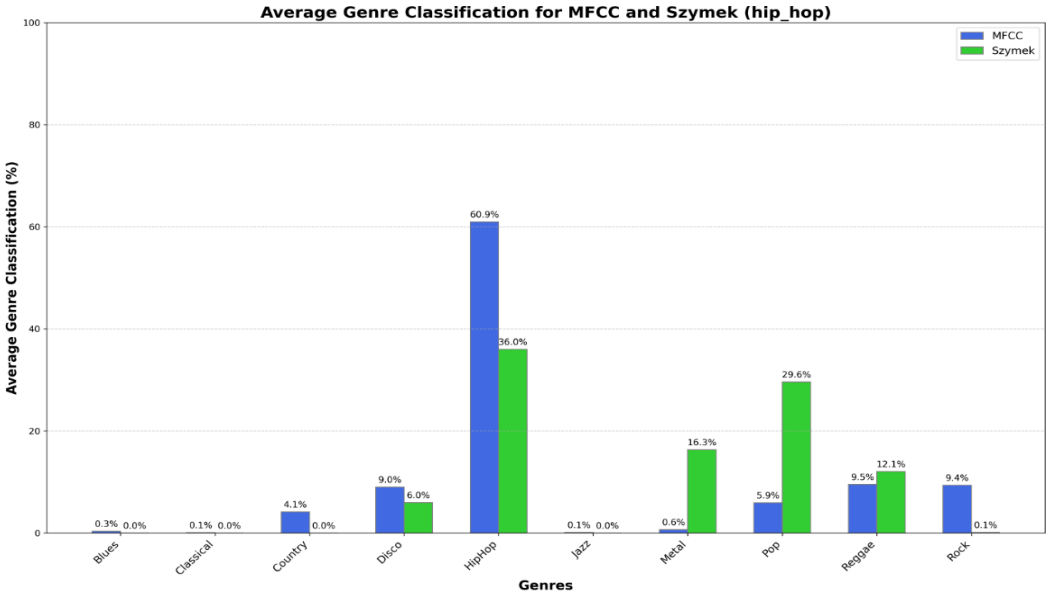
Rys. 16. Wykres procentowych wartości średnich klasyfikacji gatunków – pop.

Dla gatunku pop można zauważyć, że model Szymek rozpoznaje utwory poprawnie albo jako metal, przy czym oba te gatunki stanowią 89% jego klasyfikacji. Model MFCC natomiast radzi sobie z klasyfikacją popu gorzej. Utwory są znacznie częściej klasyfikowane jako country niż poprawnie jako pop.

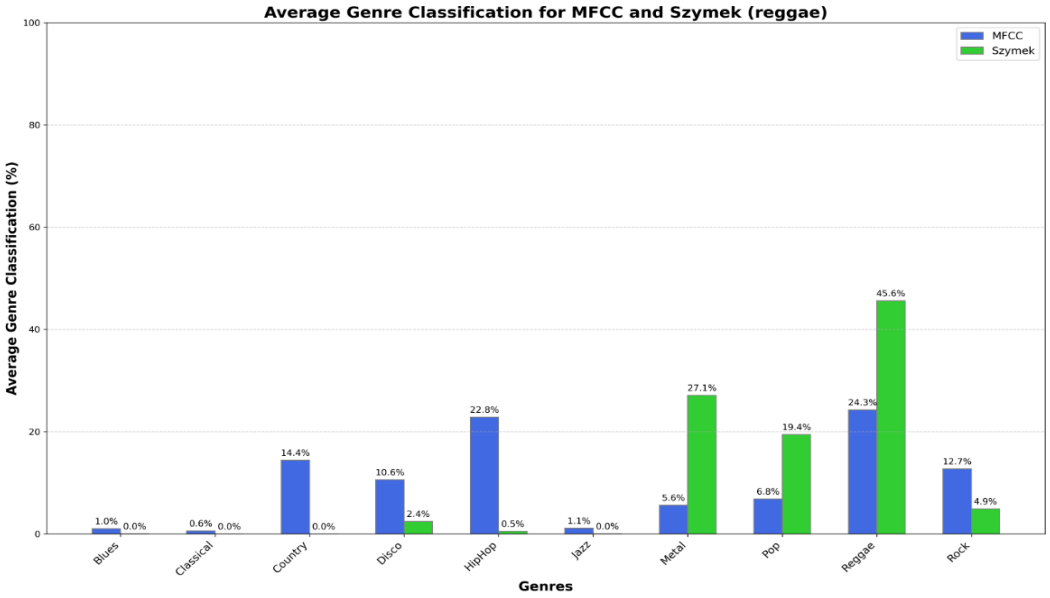


Rys. 17. Wykres procentowych wartości średnich klasyfikacji gatunków – country.

Dla gatunku country model MFCC okazuje się znacznie lepszy, osiągając dwukrotnie lepsze wyniki niż model Szymek. Model Szymek ma trudności z poprawną klasyfikacją, ponieważ utwory częściej przypisuje do gatunków metal i pop.



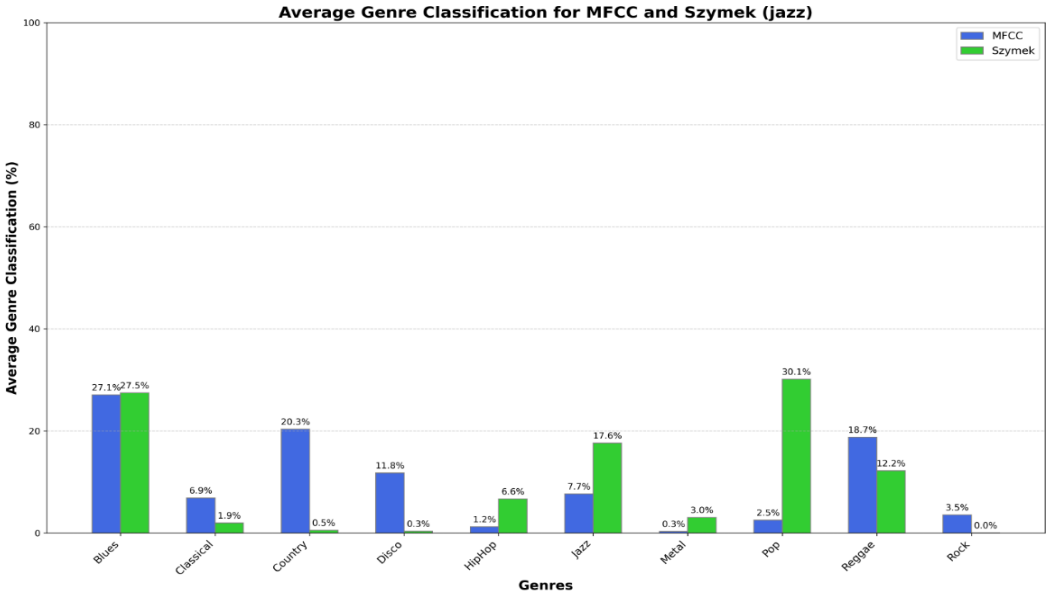
Rys. 18. Wykres procentowych wartości średnich klasyfikacji gatunków – hip hop.



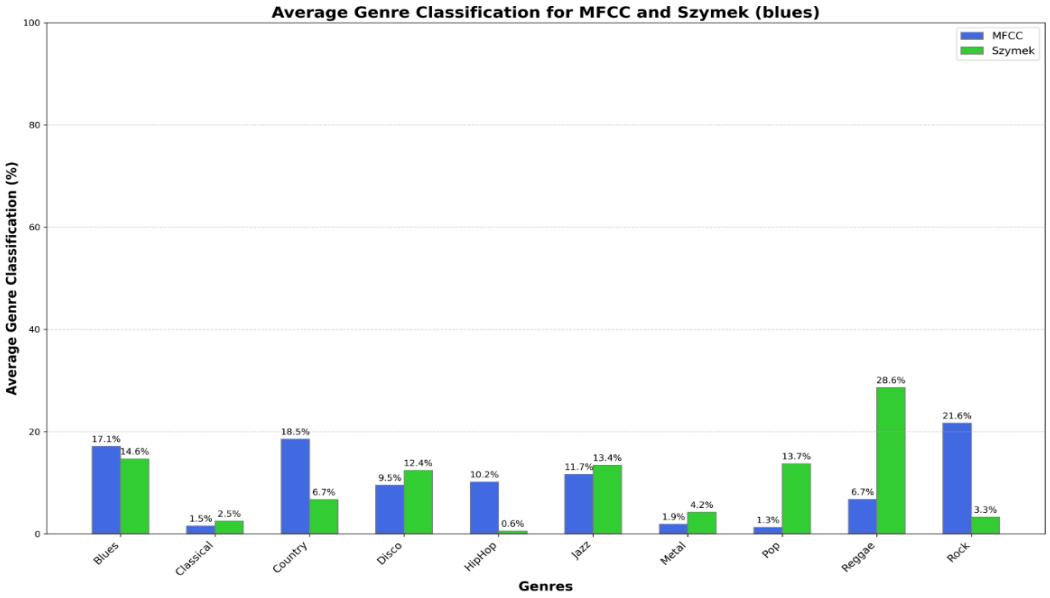
Rys. 19. Wykres procentowych wartości średnich klasyfikacji gatunków – reggae.

Dla gatunków hip hop i reggae oba modele radzą sobie dobrze, uzyskując najwyższe średnie w klasyfikacji poprawnych typów.

W przypadku hip hopu model MFCC osiąga prawie dwukrotnie lepsze wyniki niż model Szymek. Natomiast w klasyfikacji reggae model Szymek okazuje się dwukrotnie lepszy od modelu MFCC.



Rys. 20. Wykres procentowych wartości średnich klasyfikacji gatunków – jazz.



Rys. 21. Wykres procentowych wartości średnich klasyfikacji gatunków – blues.

Gatunki jazz i blues stanowią największe wyzwanie dla obu modeli, które radzą sobie z nimi najgorzej. Klasyfikowane utwory są przypisywane do różnych gatunków, a uzyskane średnie wartości skuteczności są na podobnym poziomie.

Model Szymek klasyfikuje utwory jazzowe poprawnie z skutecznością poniżej 20%. Utwory te są często klasyfikowane jako pop lub blues. Model MFCC radzi sobie z jazzem jeszcze gorzej, osiągając skuteczność poniżej 10%. Najczęściej klasyfikowane gatunki to blues, country i reggae.

Dla gatunku blues skuteczność klasyfikacji obu modeli jest bardzo niska.

6. Podsumowanie i wnioski

- Po przetestowaniu obu modeli nie da się podjąć jednoznacznej decyzji, który model jest lepszy, a który gorszy. Oba mają swoje mocne i słabe strony, co przekłada się na fakt, że niektóre gatunki muzyczne są lepiej klasyfikowane przez model Szymka, a inne przez model MFCC.
- Dobrym tego przykładem są gatunki hip-hop i reggae. W przypadku hip-hopu model MFCC radzi sobie dwa razy lepiej niż model Szymka, natomiast w przypadku reggae dwa razy lepszy okazuje się model Szymka.
- Istnieją również gatunki, z którymi oba modele radzą sobie bardzo dobrze, takie jak muzyka klasyczna, lub bardzo źle, tak jak w przypadku bluesa.
- Dokonanie klasyfikacji utworów na poszczególne gatunki wymaga uwzględnienia odpowiednio długiego odcinka czasowego, tak aby była to faktyczna, merytoryczna analiza. Z tego powodu finalnie skorzystano z odcinków o długości około 28 sekund a nie 3 sekund.
- Gatunki Blues i Country są problematyczne do rozpoznania dla obu modeli ze względu na podobieństwo występujących w nich instrumentów.
- Model MFCC stosujący SFTF (krótka transformata Fouriera) wykazał się dużo wyższą skutecznością w rozróżnianiu Rocka i Metalu, w porównaniu do MFCC wyznaczanego na podstawie spektrogramu.
- W celu poprawy skuteczności (dokładności) wykorzystanych modeli oraz aby sensownym było wykorzystanie bardziej złożonych architektur sieci neuronowych, należałoby wykorzystać większy dataset.
- Niektóre błędy w klasyfikacji współczesnych utworów mogą wynikać z faktu, że wszystkie utwory znajdujące się w bazie uczącej pochodzą z zeszłego wieku. Mogą pojawiać się różnice w utworach tych samych gatunków wynikające z ich daty (jakość dźwięku, ewolucja gatunku).

7. Literatura

[1] L. Roberts, „Understanding the Mel Spectrogram”, Analytics Vidhya.

Dostęp: 20 maja 2024 [Online]. Dostępne na: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53

[2] „librosa.mel_frequencies — librosa 0.10.1 documentation”.

Dostęp: 20 maja 2024 [Online].

Dostępne na: https://librosa.org/doc/main/generated/librosa.mel_frequencies.html

[3] „Keras: Deep Learning for humans”. Dostęp: 20 maja 2024 [Online]

Dostępne na: https://keras.io/

[4] „librosa — librosa 0.10.1 documentation”. Dostęp: 20 maja 2024 [Online].

Dostępne na: https://librosa.org/doc/latest/index.html

[5] „Matplotlib — Visualization with Python”. Dostęp: 20 maja 2024 [Online].

Dostępne na: https://matplotlib.org/

[6] „NumPy”. Dostęp: 20 maja 2024 [Online]. Dostępne na: https://numpy.org/

[7] „scikit-learn: machine learning in Python — scikit-learn 1.3.2 documentation”.

Dostęp: 20 maja 2024 [Online]. Dostępne na: https://scikit-learn.org/stable/index.html

[8] F. Pedregosa i in., „Scikit-learn: Machine Learning in Python”, J. Mach. Learn. Res., t. 12, nr 85, s. 2825–2830, 2011.

[9] „Milion Song Dataset + Spotify + Last.fm”. Dostęp: 1 czerwca 2024

Dostępne na:

https://www.kaggle.com/datasets/undefinenull/million-song-dataset-spotify-lastfm

[10] „Ludwig Music Dataset (Moods and Subgenres)”. Dostęp: 1 czerwca 2024

Dostępne na:

https://www.kaggle.com/datasets/jorgeruizdev/ludwig-music-dataset-moods-and-subgenres