POLITECHNIKA BIAŁOSTOCKA

WYDZIAŁ INFORMATYKI

PROJEKT ZESPOŁOWY – BADAWCZY

TEMAT: Rozpoznawanie stylów muzycznych z wykorzystaniem deep learningu

WYKONAWCY: Paweł Marczak, Szymon Górski

BIAŁYSTOK, 16.01.2022

Spis treści

1 Motywacje

Projekt był realizowany w ramach przedmiotu „Projekt zespołowy - badawczy" na studiach II. stopnia kierunku informatyka. Motywacjami przy wyborze tematu projektu były:

- potencjalne wykorzystanie efektów pracy i zdobytej wiedzy przez jednego z autorów przy pisaniu pracy magisterskiej o spokrewnionej tematyce
- chęć poszerzenia wiedzy o modelach klasyfikujących opartych na głębokim uczeniu
- zainteresowanie autorów tematami muzycznymi

2 Tło projektu

Rozpoznawanie gatunków muzycznych jest procesem wymagającym szerokiego rozeznania na poziomie technicznym – oprócz wiedzy dotyczącej cech charakteryzujących dany gatunek muzyczny należy posiadać wiedzę na temat ich ekstrakcji, przetwarzania i rozpoznawania. Istotną rolę pełni tu zarówno dobór metod otrzymywania informacji na temat cech danego utworu (z zakresu: barwy dźwięku, brzmienia całości, zawartości rytmicznej, wysokości tonów/dźwięków [5]), jak też utworzenie optymalnego modelu rozpoznawania gatunku (radzącego sobie dobrze z obecnymi wśród sygnałów dźwiękowych zależnościami nieliniowymi [2]).

W przypadku ekstrakcji cech istotne jest zwrócenie uwagi na pozyskanie danych dotyczących cech widmowych oraz związanych z nimi parametrów mel-cepstralnych (MFCC) ([5], [6]). Otrzymany zestaw nie jest jednak kompletny – brakuje w nim informacji odnośnie zawartości rytmicznej, możliwych do wyekstrahowania poprzez analizę pod względem metrum, tempa, rytmu i jego intensywności [7].

W celu określenia postępów w pracach nad problemem algorytmicznej klasyfikacji gatunków muzycznych wsparto się artykułami przeglądowymi ([1], [11]). Wywnioskowano z nich, że najrozsądniejsze jest testowanie utworzonej metody na zbiorze GTZAN, pozostałe zestawy cieszą się mniejszą popularnością ([20], [21]). Dzięki temu możliwe jest zestawienie jej (nie tylko przez autorów) ze zdecydowanie większą liczbą prac opartych na różnych podejściach.

Autorzy	Zestaw danych	Model	Dokładność
Tzanekis, Cook [12]	GTZAN	Gaussian Mixture Mode	61%
Benetos, Kontropoulos [13]	GTZAN	Non-negative Tensor Factorization	75%
Medhat et.al [18]	Homburg	MCLNN	61%
Sturm [10]	GTZAN	Sparse Representation Classification	83%
Choi et.al [17]	Navier music	CNN	75%
Lopes et.al [3]	LMD subset	SVM	60%
Pelchat, Gelowitz [22]	Private dataset	CNN	85%
Yang et. al [23]	GTZAN	Parallel CNN-RNN	92%
Ghosal, D., Kolekar, M.H. [25]	GTZAN	CNN Average Pooling with MLP	94%

Tablica 1: Zestawienie wybranych modeli klasyfikacji gatunków muzycznych

Rozpatrzone rozwiązania testowane na GTZAN ([12], [13], [10]) podobnie konstruują wektory cech, tzn. wybierają pewien zestaw cech wyrażonych jako wektor n-wymiarowy, następnie

uzależniają go od momentu trwania utworu, tworząc tym samym wektor n+1 wymiarowy (n to liczba wybranych do ekstrakcji cech). Dobór samych cech i sposobu ich ekstrakcji różni się dla każdej z prac, ale Tzanekis i Benetos ([12], [13]) podkreślają wysoką informatywność współczynników MFCC pod kątem klasyfikacji stylów (silnie skorelowanych ze spektrogramami, wykorzystywanymi powszechnie jako input w nowszych metodach opartych na głębokim uczeniu).

Wszystkie wspomniane prace korzystają z modeli opartych na tradycyjnych algorytmach. Nie jest to kwestią przypadku - zbiór GTZAN zawiera jedynie 1000 utworów (po 100 dla każdego stylu). Jest to stosunkowo mało by wytrenować sieć neuronową, choć nowoczesne metody głebokiego uczenia radzą sobie dobrze również na tak małych zestawach ([23]).

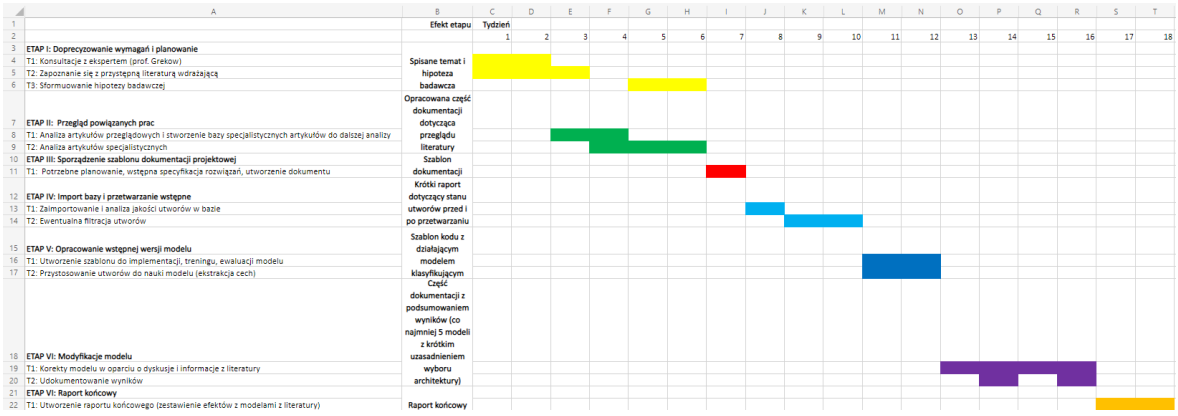
W poszukiwaniu inspiracji przy implementacji modelu, przejrzano kilka propozycji ([17], [18], [19], [22], [23]). Szczególną uwagę poświęcono metodom opartym o konwolucyjne sieci neuronowe (CNN) ([17], [22]), będących podstawą większości „głębokich" podejść do problemu ([23], [19], [18]). Łączą w sobie niezłe wyniki oraz prostotę implementacji i modyfikacji, co sprzyja późniejszemu rozwijaniu modelu.

W tej pracy pokazane zostanie, że opracowany przez autorów model oparty na CNN osiąga dokładność klasyfikacji nie gorszą od tradycyjnych metod testowanych na zbiorze GTZAN.

3 Skrótowa analiza wymagań

Projekt zakłada implementację kilku modeli klasyfikacji gatunków muzycznych opartych na sieciach neuronowych, wytrenowanie i przetestowanie ich na jednym z publicznie dostępnych zbiorów danych, udokumentowanie wyników i zestawienie ich z innymi modelami z literatury, testowanymi na tych samych danych.

4 Zakres projektu i harmonogram



Rysunek 1: Diagram Gantta opisujący planowany przebieg prac

Prace zostały podzielone na 7 etapów, każdy z nich powinien przynieść odpowiedni efekt:

- ETAP I: Doprecyzowanie wymagań i planowanie (planowany termin zakończenia: 16.11.21) – Spisane: temat i hipoteza badawcza
- ETAP II: Przegląd powiązanych prac (planowany termin zakończenia: 16.11.21) – Opracowana część dokumentacji dotycząca przeglądu literatury
- ETAP III: Sporządzenie szablonu dokumentacji projektowej (planowany termin zakończenia: 23.11.21) – Szablon dokumentacji
- ETAP IV: Import bazy i przetwarzanie wstępne (planowany termin zakończenia: 7.12.21) – Krótki raport dotyczący stanu utworów przed i po ewentualnym przetwarzaniu (z podjętymi krokami)
- ETAP V: Opracowanie wstępnej wersji modelu (planowany termin zakończenia: 21.12.21) – Szablon kodu z działającym modelem klasyfikującym
- ETAP VI: Modyfikacje modelu (planowany termin zakończenia: 4.01.22 i 18.01.22) – Część dokumentacji z podsumowaniem wyników (co najmniej 5 modeli z krótkim uzasadnieniem wyboru architektury). Raport aktualizowany 2-etapowo, co 2 tygodnie.
- ETAP VII: Raport końcowy (planowany termin zakończenia: 21.12.21) – Raport końcowy

5 Szczegółowa specyfikacja projektu i opis wymagań

- Baza treningowo - testowa: Do treningu i testowania modeli wykorzystana została baza utworów GTZAN. Składa się ona z 1000 utworów, każdy o długości 30s. Utwory zostały skatalogowane gatunkami (10 gatunków, po 100 utworów w każdym). Badania ograniczyły się do pracy z tym jednym zbiorem danych, ze względu na jego dużą popularność i ograniczone zasoby zespołu projektowego.
- Środowisko deweloperskie - projekt rozwijany był Środowisku chmurowym Google Colab. Środowisko to spełniało wszystkie potrzeby projektu: łatwa dostępność potrzebnych paczek, możliwość korzystania ze współdzielonych zasobów Google Drive oraz ze zdalnych zasobów GPU potrzebnych do obliczeń. Chmurowe środowisko wykonawcze bardzo usprawniło współpracę pomiędzy autorami.
- Ekstrakcja cech - wektorami cech przekazywanymi do modelu klasyfikującego były spektrogramy pozyskiwane z próbek z wykorzystaniem biblioteki Pythonowskiej Librosa, cieszącej się dużą popularnością wśród osób realizujących podobne projekty.
- Model sieci neuronowych - wybranym typem klasyfikatora zostały konwolucyjne sieci neuronowe (CNN) oparte na bibliotece Keras udostępnionej przez Python. Powodami tego wyboru były wcześniejsze doświadczenie autorów w pracy z Keras oraz powszechność i niezła skuteczność CNN w problemach związanych z klasyfikacją dźwięków.
- Metodologia modyfikacji i ewaluacji modelu - model był cyklicznie modyfikowany (w oparciu o wiedzę z litearatury, konsultacje z ekspertami i spostrzeżenia autorów) i ewaluowany. Ewaluacja wyników oparta była na kilkukrotnym powtórzeniu procesu uczenia

i ocenę średniej dokładności klasyfikacji oraz odchylenia standardowego na wyizolowanym zbiorze testowym.

- Zestawienie wyników - zestawienie modeli zostało na porównaniu procentowej dokładności klasyfikacji, zebranej w tabeli 1 w ramach przeglądu literatury. Wyniki zestawiano z pracami Tzanekisa [12], Benetosa [13] i Sturma [10] opartymi na tradycyjnych klasyfikatorach. Konkretniej, celem tego projektu było osiągnięcie precyzji klasyfikacji nie mniejszej niż 83% [10] na zbiorze testowym. Osiągnięte wyniki zostały poparte testami statystycznymi (dwumianowym testem proporcji).

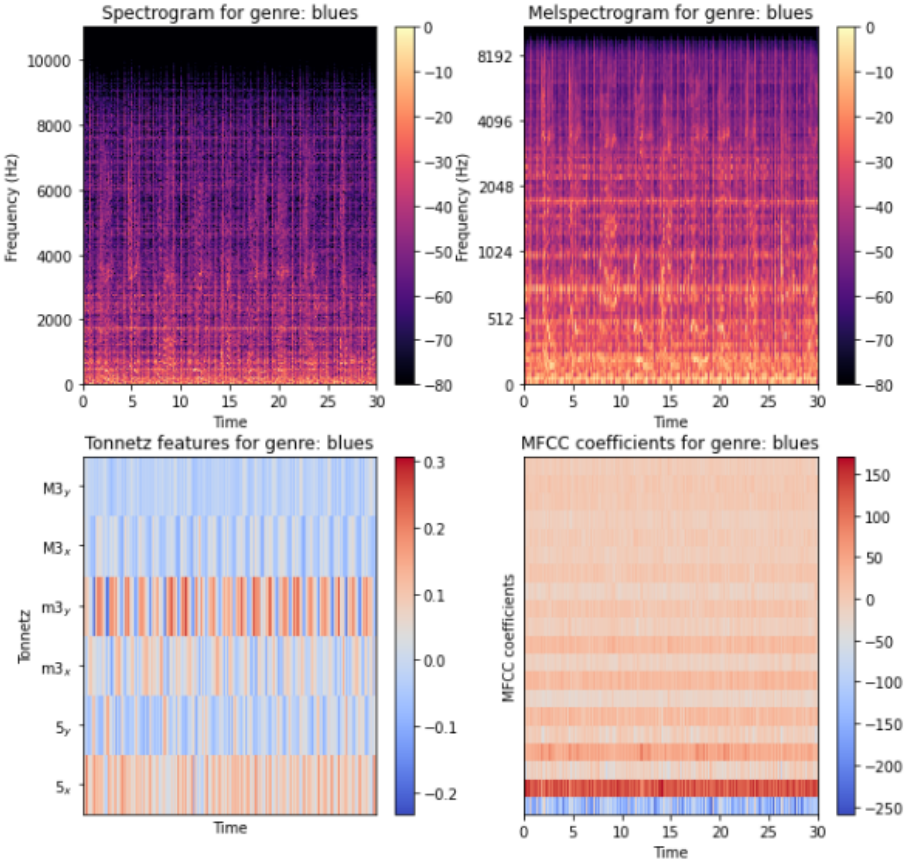
6 Opis postępu i wyników prac

6.1 Przygotowania i eksploracja bazy

W ramach przygotowań do projektu, dopracowane zostały szczegóły dotyczące wymagań, dokonano przeglądu powiązanych prac, sporządzono zestawienie wybranych wyników i wyszukano materiały pomocne przy implementacji.

Kolejnym krokiem było zaimportowanie bazy GTZAN, dokonanie wymaganych analiz i przetworzenia wstępnego.

Wygenerowano kilka przykładowych spektrogramów i innych obrazów charakteryzujących wybrane utwory.



Rysunek 2: Przykładowa analiza cech gatunku

Dokonano także przesłuchania wybranych plików dźwiękowych zawartych w bazie GTZAN. Jakość plików zawartych w bazie była średnia, głównie ze względu na niską częstotliwość próbkowania (22050 Hz) oraz reprezentację sygnału jedynie za pomocą jednego kanału audio (mono). Stwierdzono także wystąpienie jednego wadliwego pliku w bazie (o nazwie: 'jazz.00054.wav') – niezależnie od wykorzystanego software'u plik nie był możliwy do odtworzenia, przez co nie był brany pod uwagę przy rozwoju prac.

Należy zaznaczyć, iż wykorzystywana instancja bazy GTZAN pochodzi z portalu Kaggle (link: https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification).

Gatunek	Ilość plików
Blues	100
Muzyka poważna	100
Country	100
Disco	100
Hip-hop	100
Jazz	100
Metal	100
Pop	100
Reggae	100
Rock	100
Ilość wszystkich plików	**1000**

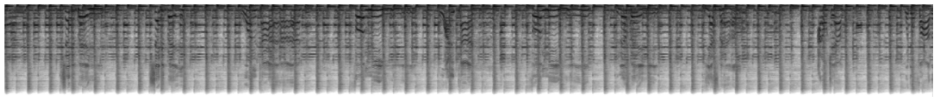
Tablica 2: Zestawienie podstawowych statystyk dotyczących bazy GTZAN

Cecha	Wartość
Ilość działających plików	999
Ilość błędnych plików	1
Częstotliwość próbkowania	22050 Hz
Ilość kanałów	1 (mono)
Minimalna długość pliku	29.93 s
Maksymalna długość pliku	30.65 s
Średnia długość pliku	30.02 s

Tablica 3: Zestawienie rozszerzonych statystyk dotyczących bazy GTZAN

6.2 Ekstrakcja cech i przygotowanie zbiorów treningowego i testowego

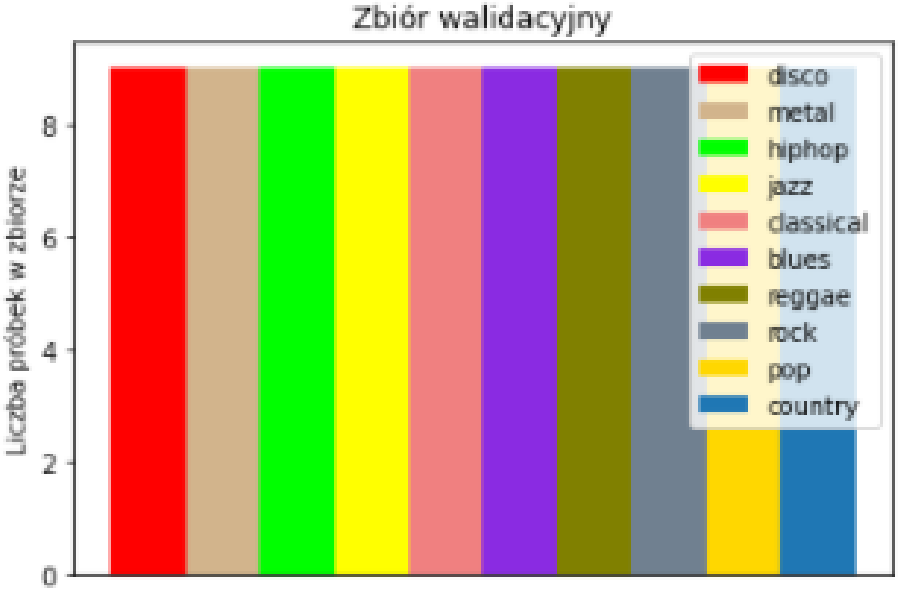
Na wejście klasyfikatora zdecydowano się przekazywać melspektrogramy w skali decybelowej. Do wygenerowania ich wykorzystano moduły Librosy. Ustalono zakres amplitudy do -80 dB. Następnie wartości spektrogramów podzielono przez -80, przez co wszystkie wartości należały do zakresu od 0 do 1. Wynikowo otrzymano spektrogram taki jak na rysunku 3. Z racji, że utwory trwały aż 30 s, wynikowe spektrogramy były mocno niewymiarowe. Wszystkie zostały dopasowane do rozmiaru 128 x 1300 pikseli i otrzymały etykiety.



Rysunek 3: Przykładowy znormalizowany melspektrogram w skali decybelowej

Zdecydowano się podzielić zbiór na treningowy i testowy ze stratyfikacją klas w stosunku 90% - 10%. Następnie ze zbioru testowego odseparowano część walidacyjną, ponownie w stosunku 90% - 10%. Proporcje te wydają się niesprawiedliwe, jednak ze względu na małe rozmiary bazy, takie podejście jest dość popularne dla zbioru GTZAN, również wśród prac do których się przyrównywano.

Rysunek 4: Rozkład zbioru treningowego

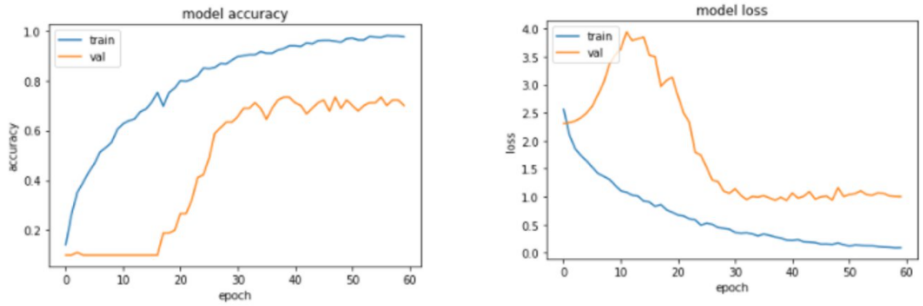


Rysunek 5: Rozkład zbioru walidacyjnego

Rysunek 6: Rozkład zbioru testowego

6.3 Utworzonie prototypu klasyfikatora

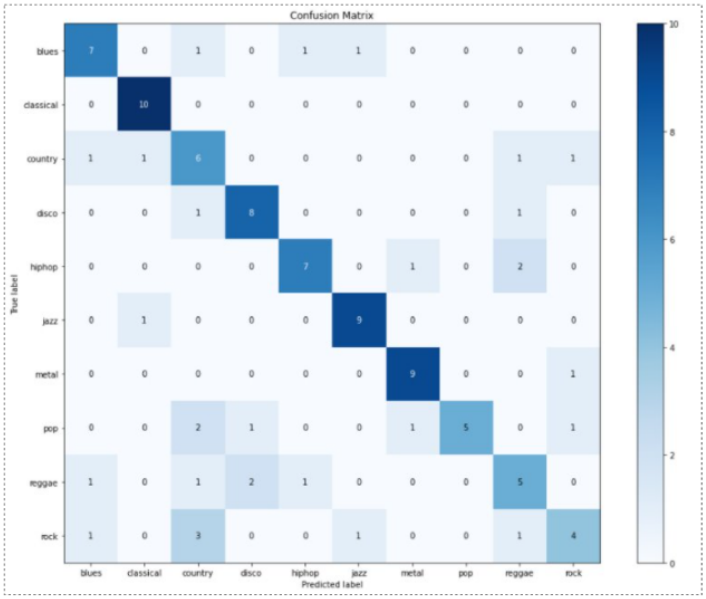
Utworzono wstępną wersję modelu sieci CNN w oparciu o zgromadzone materiały i wskazówki prowadzącego. Po dopasowaniu parametrów i rozwiązaniu kilku mniejszych problemów udało się przygotować działającą sieć klasyfikującą spektrogramy. Przy pięciokrotnym powtórzeniu procedury uczenia i testowania na zbiorze testowym osiągnął on średnią 71.6% dokładności i 1.62% odchylenia standardowego - wynik zdecydowanie gorszy od docelowego.



Rysunek 7: Krzywe z historii uczenia prototypu

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 1303, 32)     320

 max_pooling2d (MaxPooling2D  (None, 15, 20, 32)       0
 )

 batch_normalization (BatchN  (None, 15, 20, 32)       128
 ormalization)

 conv2d_1 (Conv2D)           (None, 15, 20, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 7, 10, 64)        0
 2D)

 batch_normalization_1 (Batc  (None, 7, 10, 64)        256
 hNormalization)

 conv2d_2 (Conv2D)           (None, 7, 10, 64)         36928

 max_pooling2d_2 (MaxPooling  (None, 3, 5, 64)         0
 2D)

 batch_normalization_2 (Batc  (None, 3, 5, 64)         256
 hNormalization)

 flatten (Flatten)           (None, 960)               0

 dense (Dense)               (None, 64)                61504

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 64)                4160

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 10)                650

=================================================================
Total params: 122,698
Trainable params: 122,378
Non-trainable params: 320
_________________________________________________________________
```

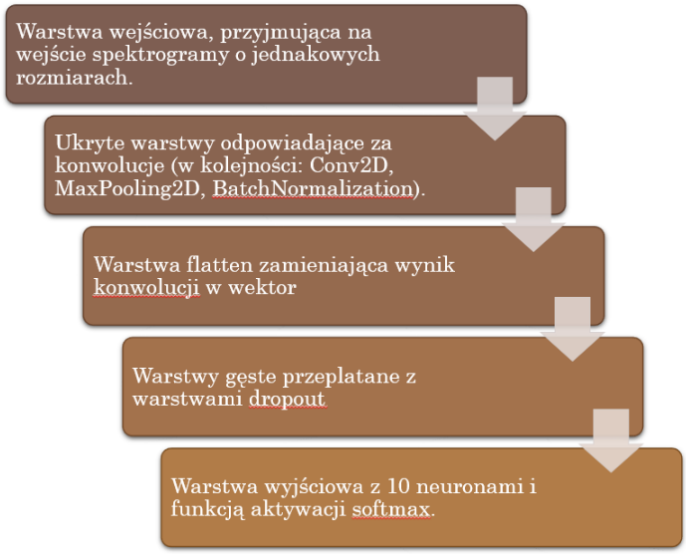
Rysunek 8: Struktura sieci prototypu



Rysunek 9: Macierz pomyłek prototypu

6.4 Dostrajanie klasyfikatora

W oparciu o różne źródła, ustalono, że każdy rozpatrywany model sieci powinien być wzorowany na schemacie z rysunku 10. W celu znalezienia optymalnego zestawienia hiperparametrów utworzono 10 wersji modelu. Wszystkie trenowano przez 60 epok (zaobserwowano, że dłuższe uczenie w zdecydowanej większości przypadków niczego nie zmieniało) – wagi z najlepszymi rezultatami na zbiorze walidacyjnym otrzymane w trakcie tego procesu zapisywane były za pomocą callbacku *ModelCheckpoint*, natępnie wracano do nich po zakończeniu uczenia. Dla każdego zestawu hiperparametrów procedura uczenia była powtarzana 5 razy, natępnie zapisywano średnią i odchylenie standardowe dokładności klasyfikacji na zbiorze testowym. Informacje o testowanych parametrach i otrzymanych rezultatach przedstawiono w ramach tabeli 4. Kolejne testowane zestawy parametrów były dobierane na podstawie spostrzeżeń z poprzednich prób dotyczących otrzymanych wyników i krzywych uczenia.



Rysunek 10: Topologia sieci klasyfikujących

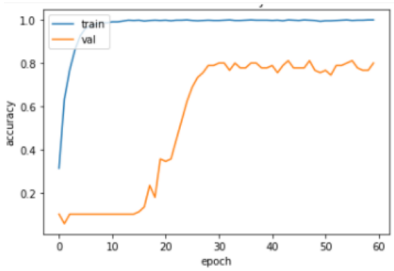
	Architektura			Parametry		Rezultaty	
Nazwa modelu	Liczba filtrów (Conv2D)	Rozmiary filtrów (MaxPooling2D)	Neurony w warstwie gęstej (Dense)	Współczynnik uczenia	Zmienność współczynnika	Średnia dokładność	Odchylenie standardowe
Model 1	32, 32, 64, 64	(1, 8), (4, 4), (2, 2), (2, 2)	128, 128, 10	0.0001	Nie	0.75400	0.02653
Model 2	32, 32, 64, 64	(1, 8), (3, 3), (2, 2), (2, 2)	256, 128, 10	0.0001	Nie	0.76000	0.01789
Model 3	32, 32, 64, 64	(1, 8), (2, 2), (2, 2), (2, 2)	128, 128, 10	0.0001	Nie	0.76000	0.03162
Model 4	32, 32, 64, 64	(1, 8), (2, 2), (2, 2), (2, 2)	512, 512, 10	0.0001	Nie	0.75400	0.04964
Model 5	32, 32, 64, 64	(1, 8), (2, 2), (2, 2), (2, 2)	512, 512, 10	0.001	Tak, zmniejszany 2-krotnie co 10 epok	0.78200	0.04167
Model 6	32, 32, 64, 64	(1, 8), (2, 2), (2, 2), (2, 2)	512, 512, 10	0.002	Tak, zmniejszany 2-krotnie co 10 epok	0.72200	0.04167
Model 7	32, 32, 64, 64	(1, 6), (2, 2), (2, 2), (2, 2)	1024, 1024, 10	0.001	Tak, zmniejszany 2-krotnie co 10 epok	0.77400	0.02871
Model 8	128, 128, 128, 128	(1, 8), (2, 2), (2, 2), (2, 2)	512, 512, 10	0.001	Tak, zmniejszany 2-krotnie co 10 epok	0.75000	0.18974
Model 9	32, 64, 64, 128	(1, 8), (2, 2), (2, 2), (2, 2)	512, 512, 10	0.001	Tak, zmniejszany 2-krotnie co 10 epok	0.77400	0.02871
Model 10	32, 64, 128, 256	(1, 8), (2, 2), (4, 4), (2, 2)	1024, 1024, 1024, 10	0.001	Tak, zmniejszany 2-krotnie co 10 epok	0.78800	0.01327

Tablica 4: Zestawienie architektur, parametrów i rezultatów działania poszczególnych modeli klasyfikacji gatunków muzycznych utworzonych na potrzeby projektu

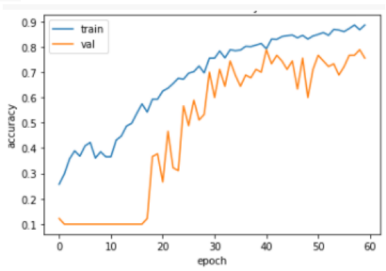
Najlepszym modelem pod kątem skuteczności okazał się model 10. Był to największy z testowanych modeli (charakteryzował się ponad 20 milionami trenowalnych parametrów). Dało się zauważyć, że jego rozmiary odbijały się na czasie uczenia. Oprócz tego, duża liczba parametrów zwiększała ryzyko nadmiernego dopasowania modelu do danych treningowych. Osiągnał jednak najlepsze i najstabilniejsze wyniki na zbiorze walidacyjnym oraz całkowicie odizolowanych danych testowych, więc tak dużą liczbę trenowalnych parametrów, w tym przypadku można uznać za uzasadnioną.

Próbowano jeszcze zwiększać liczbę parametrów, bez całej procedury 5 - krotnego powtarzania, jednak zauważono wyraźne pogorszenie wyników. Manipulacje learning ratem również nie przynosiły zamierzonych efektów. Należy zauważyć, że wykres uczenia modelu 10 (rysunek 12) odbiega od podręcznikowych przykładów (krzywe treningowa i walidacyjna mocno od siebie oddalone - typowe dla zbyt małych zbiorów treningowych, niski learning rate w końcowych fazach wygładza krzywą walidacyjną i zmniejsza szanse na poprawę wyników w późniejszych epokach).

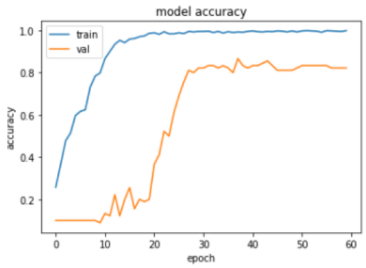
Odnosząc kolejne porażki w próbach poprawienia osiąganych wyników i przebiegu uczenia się modelu poprzez dalszą manipulację parametrami, autorzy postanowili spróbować czegoś innego.



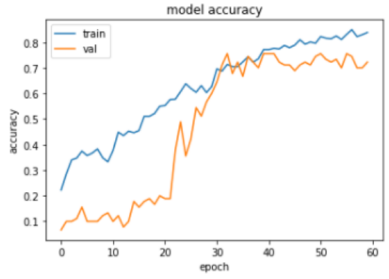
Przykładowy wykres uczenia dla modelu 4



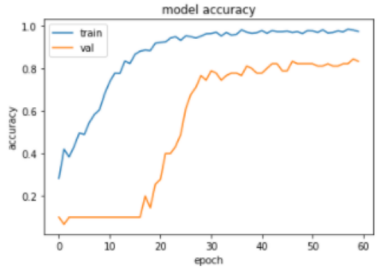
Przykładowy wykres uczenia dla modelu 6



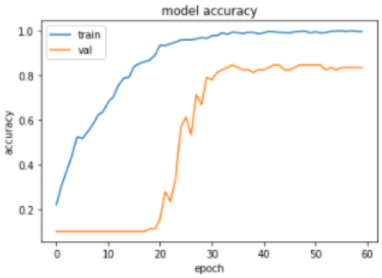
Przykładowy wykres uczenia dla modelu 7



Przykładowy wykres uczenia dla modelu 8



Przykładowy wykres uczenia dla modelu 9



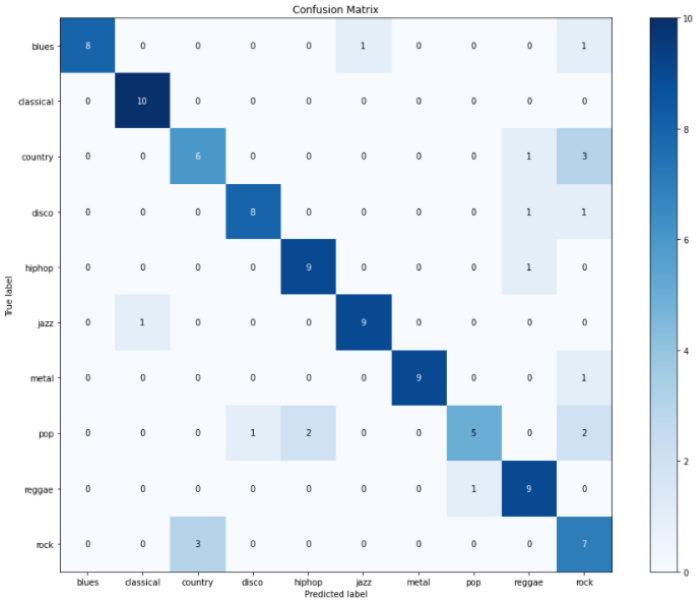
Przykładowy wykres uczenia dla modelu 10

Rysunek 11: Historia uczenia wybranych modeli

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 1298, 32)	320
max_pooling2d (MaxPooling2D)	(None, 126, 162, 32)	0
batch_normalization (BatchNormalization)	(None, 126, 162, 32)	128
conv2d_1 (Conv2D)	(None, 126, 162, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 63, 81, 64)	0
batch_normalization_1 (BatchNormalization)	(None, 63, 81, 64)	256
conv2d_2 (Conv2D)	(None, 63, 81, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 15, 20, 128)	0
batch_normalization_2 (BatchNormalization)	(None, 15, 20, 128)	512
conv2d_3 (Conv2D)	(None, 15, 20, 256)	295168
max_pooling2d_3 (MaxPooling2D)	(None, 7, 10, 256)	0
batch_normalization_3 (BatchNormalization)	(None, 7, 10, 256)	1024
flatten (Flatten)	(None, 17920)	0
dense (Dense)	(None, 1024)	18351104
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 1024)	1049600
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 1024)	1049600
dropout_2 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 10)	10250

Total params: 20,850,314
Trainable params: 20,849,354
Non-trainable params: 960

Rysunek 12: Struktura modelu 10

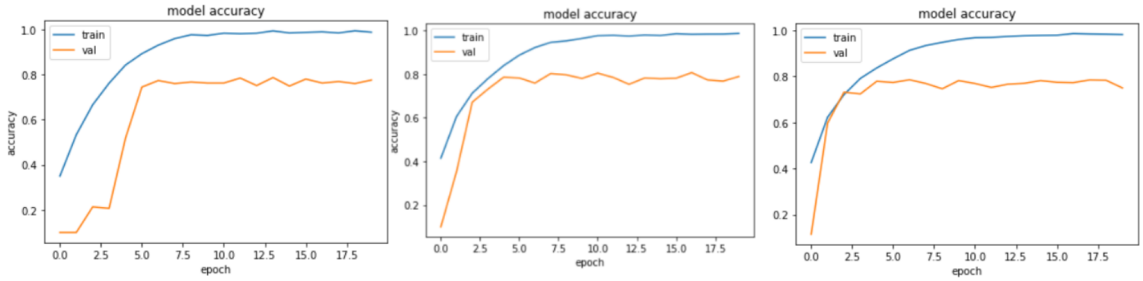


Rysunek 13: Macierz pomyłek na zbiorze testowym modelu 10

6.5 Pocięcie utworów

Popularną praktyką wśród badaczy pracujących na GTZAN ([13], [19]) jest sztuczne zwiększanie objętości zbioru, przez cięcie trzydziesto - sekundowych utworów na więcej krótszych fragmentów.

Licząc, że powiększenie zbioru treningowego przybliży do siebie krzywe uczenia autorzy przeprowadzili eksperyment. Zachowując strukturę modelu 10, pocięli nagrania na fragmeny o długości 6s, 3s i 2s, dla każdej długości próbki wytrenowali sieć i zestawili ze sobą wykresy uczenia (rysunek 20).



Przykładowy wykres uczenia na nagraniach 6s

Przykładowy wykres uczenia na nagraniach 3s

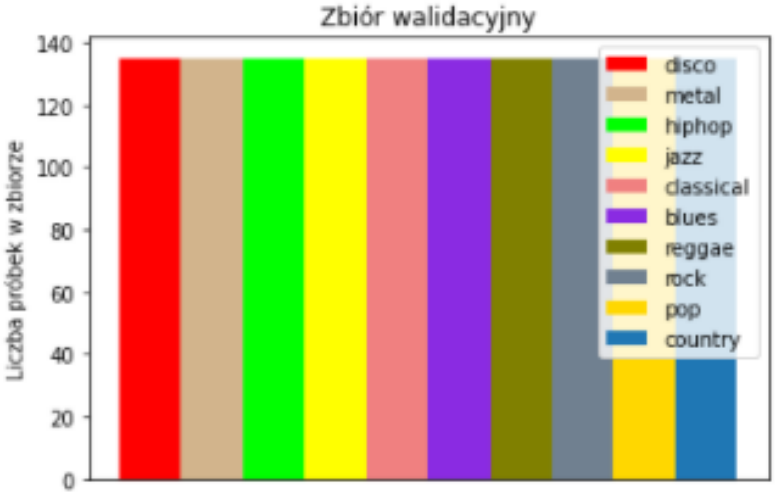
Przykładowy wykres uczenia na nagraniach 2s

Rysunek 14: Zestawienie wykresów uczenia dla różnych długości próbek

Zgodnie z oczekiwaniami, zwiększenie liczby próbek przybliżało do siebie krzywe i zmniejszało liczbę epok potrzebnych na pełne wyszkolenie sieci. Niestety, dla nowego kształtu spektrogramów wejściowych, hiperparametry modelu znowu były niedostrojone. Postanowiono

pozostać przy rozmiarach spektrogramów, dla którego krzywe uczenia się przecieły (długość próbki 2s) i wrócono do kroku dostrajania hiperparametrów.

Rysunek 15: Rozkład próbek w zbiorze treningowym



Rysunek 16: Rozkład próbek w zbiorze walidacyjnym

Rysunek 17: Rozkład próbek w zbiorze testowym

6.6 Dostrajanie modelu do pociętych utworów

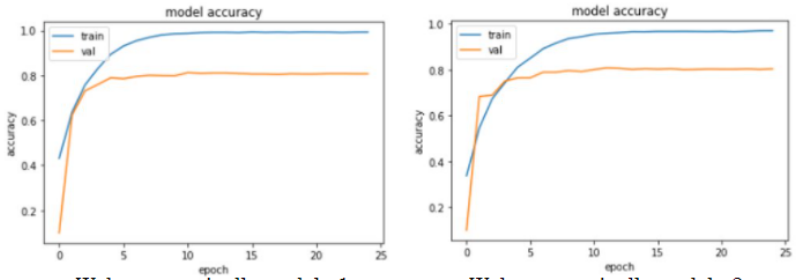
Metodyka dostrajania z paragrafu 6.4 nie była zła, ale pochłaniała mnóstwo czasu i zasobów. Ze względu na problemy z zasobami Colaba (paragraf 6.8), zrealizowanie całej procedury było mocno utrudnione. Z tego względu zdecydowano się wesprzeć narzędziem tuner dodanym do bibliotek Keras w wariancie opartym na algorytmie Hyperband [24].

W uproszczeniu, pozwala on przeszukać przestrzeń zdefiniowanych przez badacza zestawów hiperparametrów zgodnie z założeniami algorytmu, w celu znalezienia najlepszego zestawu pod względem wybranej metryki (np. dokładności na zbiorze walidacyjnym) osiąganej przez wybraną liczbę epok. Do działania tunera dodano callback EarlyStopping w celu oszczędzania zasobów udostępnionych przez Google i zmniejszający się w czasie learning rate ze względu na dobre rezultaty, jakie dostarczał na niepociętym zbiorze.

Znaleziono 3 najlepsze zestawy i przetestowano je zgodnie z procedurą z paragrafu 6.4 (modele 1,2,3; tabela 5). Obserwując krzywe uczenia zdecydowano się zredukować współczynnik degradacji learning rate dla jednego modelu, licząc, że da mu to większe szanse na wskoczenie na wyższy procent dokładności w późniejszych epokach (modele 4,5; tabela 5).

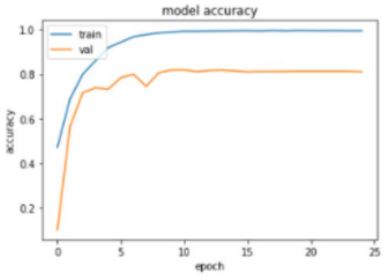
	Architektura			Parametry			Rezultaty	
Nazwa modelu	Liczba filtrów (Conv2D)	Rozmiary filtrów (MaxPooling2D)	Neurony w warstwie gęstej (Dense)	Startowy współczynnik uczenia	Zmienność współczynnika	Współczynnik dropout	Średnia dokładność	Odchylenie standardowe
Model 1	64, 128, 128, 256	(2,1), (2, 2), (2, 2), (2, 2)	384,384,384,10	0.0001	Zmniejszany co 2 epoki do 0.85^n starej wartości, gdzie n to liczba redukcji	0.3	0.783	0.002
Model 2	48, 96, 96, 192	(2,1), (2, 2), (2, 2), (2, 2)	384,384,384,10	0.0001	Zmniejszany co 2 epoki do 0.85^n starej wartości, gdzie n to liczba redukcji	0.4	0.769	0.002
Model 3	48, 96, 96, 192	(2,1), (2, 2), (2, 2), (2, 2)	256, 256, 256, 10	0.0005	Zmniejszany co 2 epoki do 0.85^n starej wartości, gdzie n to liczba redukcji	0.2	0.784	0.012
Model 4	48, 96, 96, 192	(2,1), (2, 2), (2, 2), (2, 2)	256, 256, 256, 10	0.0005	Zmniejszany co 2 epoki do 0.95^n starej wartości, gdzie n to liczba redukcji	0.2	0.782	0.013
Model 5	48, 96, 96, 192	(2,1), (2, 2), (2, 2), (2, 2)	256, 256, 256, 10	0.0007	Zmniejszany co 2 epoki do 0.97^n starej wartości, gdzie n to liczba redukcji	0.2	0.784	0.014

Tablica 5: Zestawienie architektur, parametrów i rezultatów działania poszczególnych modeli klasyfikacji gatunków muzycznych utworzonych na potrzeby projektu dla nagrań pociętych

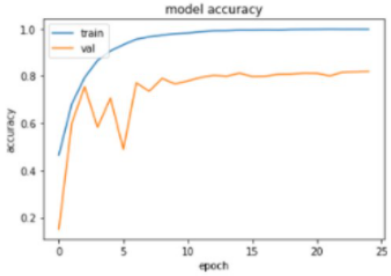


Wykres uczenia dla modelu 1
(78,3% skuteczności, $\sigma = 0.002$)

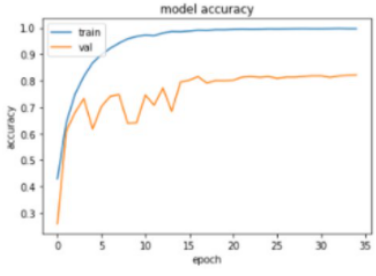
Wykres uczenia dla modelu 2
(76,9% skuteczności, $\sigma = 0.002$)



Wykres uczenia dla modelu 3
(78,4% skuteczności, $\sigma = 0.012$)



Wykres uczenia dla modelu 4
(78,2% skuteczności, $\sigma = 0.013$)



Wykres uczenia dla modelu 5
(78,4% skuteczności, $\sigma = 0.014$)

Rysunek 18: Zestawienie wykresów uczenia modeli uczonych na pociętych nagraniach

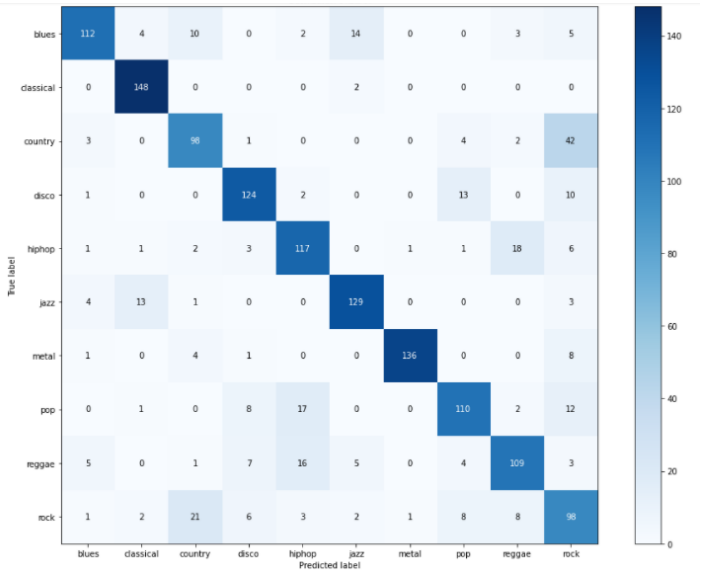
Modele 1 i 3 osiągnęły wyniki zbliżone do modelu 10 z paragrafu 6.4 (gorsze o 0.4%), ale wyniki modelu 1 miały zaledwie 0.2% odchylenia standardowego, co mogło świadczyć o dużej jego stabilności. Zważając na to, że przed strojeniem wyniki osiągane przez autorów wachały się pomiędzy 75% a 76%, trzeba stwierdzić, że tuner nie działa przypadkowo, choć ciężko powiedzieć, czy zwrócone przez niego parametry faktycznie były optymalne. Ostatecznie nie udało się poprawić wyniku z przed pocięcia. Powodami tego mogą być:

- złe dobranie programu strojenia dla tunera przez autorów (w tym callbacków)
- ograniczenia tunera i wykorzystanego algorytmu Hyperband (nieregularność wyników nauczania sieci neuronowych nie sprzyja algorytmicznemu podejściu do dostrajania)
- zbyt krótka długość trwania pociętych utworów (2 - sekundowe fragmenty mogą być zbyt mocno wyrwane z kontekstu reszty utworu)
- niewystarczająco dobra topologia sieci

Autorzy planowali dokładniej przetestować możliwości tunera, jednak zawiesili te plany ze względu na ograniczone zasoby czasowe i obliczeniowe (paragraf 6.8).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 126, 84, 64)       640

max_pooling2d (MaxPooling2D  (None, 63, 84, 64)        0
)

batch_normalization (BatchN  (None, 63, 84, 64)        256
ormalization)

conv2d_1 (Conv2D)            (None, 63, 84, 128)       73856

max_pooling2d_1 (MaxPooling  (None, 31, 42, 128)       0
2D)

batch_normalization_1 (Batc  (None, 31, 42, 128)       512
hNormalization)

conv2d_2 (Conv2D)            (None, 31, 42, 128)       147584

max_pooling2d_2 (MaxPooling  (None, 15, 21, 128)       0
2D)

batch_normalization_2 (Batc  (None, 15, 21, 128)       512
hNormalization)

conv2d_3 (Conv2D)            (None, 15, 21, 256)       295168

max_pooling2d_3 (MaxPooling  (None, 7, 10, 256)        0
2D)

batch_normalization_3 (Batc  (None, 7, 10, 256)        1024
hNormalization)

flatten (Flatten)            (None, 17920)             0

dense (Dense)                (None, 384)               6881664

dropout (Dropout)            (None, 384)               0

dense_1 (Dense)              (None, 384)               147840

dropout_1 (Dropout)          (None, 384)               0

dense_2 (Dense)              (None, 384)               147840

dropout_2 (Dropout)          (None, 384)               0

dense_3 (Dense)              (None, 10)                3850

=================================================================
Total params: 7,700,746
Trainable params: 7,699,594
Non-trainable params: 1,152
_________________________________________________________________
```

Rysunek 19: Struktura modelu 1 dla pociętych nagrań



Rysunek 20: Macierz pomyłek na zbiorze testowym modelu 1 dla pociętych nagrań

6.7 Testy statystyczne

Najlepsze modele przed i po pocięciu zostały przetestowane dwumianowym testem proporcji. Traktując każdą klasyfikację jako osobne zdarzenie, dokładność klasyfikacji można traktować jako szansę tego, że losowa klasyfikacja dokonana przez model jest prawidłowa. Można założyć, że próbki w zbiorze testowym są faktycznie wybrane losowo, jeżeli klasy w ramach tego zbioru są zbalansowane (co ma miejsce w przypadku metodyki tej pracy).

Dla obu modeli 10 razy powtórzono procedurę uczenia, z innym ziarnem randomizującym podział treningowo - testowy. Następnie zsumowano liczbę prawidłowych klasyfikacji i liczbę wszystkich klasyfikacji na zbiorze testowym. Posiadając te liczby wykonano wspomniany test w celu udowodnienia, że szansa na prawidłową klasyfikację jest większa od ustalonego progu. W przypadku niepociętych spektrogramów liczba wszystkich prób wynosiła 1000, dla pociętych 15000 (dla pociętych zbiór testowy był naturalnie większy). Przyjęto poziom istotności równy 0.05.

```
stats.binom_test(success_num, n=trials_num, p=0.75, alternative='greater')
```

```
0.00014703911941796782
```

```
stats.binom_test(success_num, n=trials_num, p=0.76, alternative='greater')
```

```
0.00186135508169993
```

```
stats.binom_test(success_num, n=trials_num, p=0.77, alternative='greater')
```

```
0.015093564855410035
```

```
stats.binom_test(success_num, n=trials_num, p=0.78, alternative='greater')
```

```
0.07783880346770382
```

Rysunek 21: Wyniki testów statystycznych dla modelu 10 z pełnymi nagraniami

```
stats.binom_test(success_num, n=trials_num, p=0.75, alternative='greater')
```

```
9.367338239548623e-11
```

```
stats.binom_test(success_num, n=trials_num, p=0.76, alternative='greater')
```

```
0.00019500376269761895
```

```
stats.binom_test(success_num, n=trials_num, p=0.77, alternative='greater')
```

```
0.25193878213348614
```

Rysunek 22: Wyniki testów statystycznych dla modelu 1 z pociętymi nagraniami

Obie metody osiągnęły wynik statystycznie większy od wyniku Benetosa [13]. Model 10 osiągnął wynik statystycznie większy od 77% (był dość blsko przekroczenia 78%), model 1 przebił jedynie 76%. Należy zaznaczyć, że dla modelu 1 wykonane zostało zdecydowanie więcej prób. Mimo to model 10 poradził sobie z testami widocznie lepiej. Model 1, pomimo małego odchylenia standardowego osiągniętego w fazie dostrajania, nie poradził sobie dobrze przy różnych podziałach zbioru na część testową i treningową, tzn. był zbyt mocno dopasowany do jednego podziału. Może to być efekt uboczny działania tunera lub pocięcia nagrań.

Ostatecznie, najskuteczniejszym z opracowanych modeli okazał się model 10 operujący na niepociętych nagraniach. O ile przebił metodę Benetosa [13], do wyniku 83% osiągniętego przez Sturma trochę mu brakuje. Należy jednak zaznaczyć, że przy sprzyjających okolicznościach, opracowany model osiągał dokładność nawet 85%, co pozwala przypuszczać, że kontynuacja pracy pozwoliłaby na poprawienie wyników do oczekiwanego poziomu.

7 Podsumowanie i kontynuacja prac

Ostatecznie celu projektu nie udało się osiągnąć w stu procentach, tzn. opracowna przez autorów sieć neuronowa przewyższyła wynikami jedynie 2 z 3 prac opartych na klasycznych klasyfikatorach. Pokazane zostały jednak wyniki i przesłanki mówiące o tym, że kontynuacja prac w metodyce zaproponowanej przez autorów może przynieść poprawę rezultatuów.

Proponowane kierunki rozwoju prac to:

- dokładniejsze sprawdzenie możliwości tunera - manipulacja większą liczbą parametrów i w większym zakresie, sprawdzenie innych algorytmów przeszukujących, użycie innych callbacków (innej funkcji degradacji learning rate, wyłączenie EarlyStopping), zastosowanie tunera na niepociętych nagraniach i zestawienie z osiągniętymi wynikami modelu 10

- dodanie dodatkowych cech do procesu uczenia - analiza pomyłek pokazała, że najtrudniejszym do rozróżnienia gatunkiem muzycznym jest rock. Ekstrakcja cech, pod względem których ten gatunek się wyróżnia, a nie są dobrze widoczne z poziomu spektrogramu i dodanie ich do procesu uczenia mogłoby pomóc sieci w lepszym rozpoznawaniu tego gatunku.

- sprawdzenie modelu na innych bazach dostosowanych do MGR - konfrontacja z innymi bazami o większej pojemności zweryfikowałaby jak uniwersalna jest metodyka autorów, a proces „szlifowania" wyników mógłby przynieść pomocne wnioski również w kontekście bazy GTZAN

Niezależnie od obranej ścieżki, rozwiązania wymaga jeden problem mocno spowalniający postępy. W trakcie prac, Google Colab w pewnych momentach odcinał dostęp do GPU w ramach notatnika, informując o osiągnięciu limitu. W początkowych etapach projektu, problem ten nie występował. Z czasem, gdy projekt wchodził w fazę intensyfikacji obliczeń, przeszkoda ta pojawiała się coraz częściej, aż do momentu, gdy autorzy mieli do wykorzystania ok. 2-3 godziny dziennie, przez co ich możliwości były mocno ograniczone.

Najrozsądniejszym wyjściem z tego problemu wydaje się być przełączenie się na lokalne GPU (mimo, że możliwości tego dostarczanego przez Google są duże i prawdopodobnie przewyższają karty graficzne autorów). Należałoby sprawdzić, czy jest możliwość przekazania lokalnego GPU do Colaba, ewentualnie możnaby zaimportować projekt do środowiska lokalnego, co prawdopodobnie negatywnie wpłynęłoby na workflow.

Innym wyjściem jest wykupienie wersji premium Google Colab, jednak i ona posiada pewne ograniczenia i oczywiście wymaga dodatkowych środków pieniężnych.

Literatura

[1] *A Survey of Evaluation in Music Genre*; Bob L. Sturm; Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation (29 October 2014, p. 29-66)

[2] *Music Genre and Emotion Recognition Using Gaussian Processes*; Konstantin Markov, Tomoko Matsui; IEEE Access (volume: 2, 25 June 2014, pp. 688-697)

[3] *Selection of Training Instances for Music Genre Classification*; Miguel Lopes, Fabien Gouyon, Alessandro L. Koerich, Luiz E.S. Oliveira; 2010 20th International Conference on Pattern Recognition (conference date: 23-26 August 2010, publication date: 07 October 2010)

[4] *Analysis of Structural Complexity Features for Music Genre Recognition* ; Philipp Ginsel, Igor Vatolkin, Güter Rudolph; 2020 IEEE Congress on Evolutionary Computation (CEC) (conference date: 19-24 July 2020, publication date: 03 September 2020)

[5] *Musical genre classification of audio signals*; G. Tzanetakis, P. Cook; IEEE Transactions on Speech and Audio Processing (volume: 10, issue: 5, July 2002)

[6] *An Empirical Study of Feature Extraction Methods for Audio Classification*; Charles Parker; 2010 20th International Conference on Pattern Recognition (conference date: 23-26 August 2010, publication date: 07 October 2010)

[7] *An effective method on content based music feature extraction*; Zhanchun Gao, Yuting Liu, Yanjun Jiang; 2015 IEEE Advanced Information Technology, Electronic and Automation Control Conference (IAEAC) (conference date: 19-20 December 2015, publication date: 10 March 2016)

[8] *Modeling musical style using grammatical inference techniques: a tool for classifying and generating melodies*; Cruz-Alcáza, P.P., Vidal-Ruiz, E.:; In: Proceedings of the WEDELMUSIC, pp. 77–84, Sept 2003

[9] *Two grammatical inference applications in music processing*; Cruz-Alcáza, P.P., Vidal-Ruiz, E; Appl. Artif. Intell. 22(1/2), 53–76 (2008)

[10] *Two systems for automatic music genre recognition: what are they really recognizing?* ; Sturm, B.L; In: Proceedings of the ACM MIRUM Workshop, Nara, Japan, Nov 2012

[11] *Music Genre Classification: A Review of Deep-Learning and Traditional Machine-Learning Approaches*; Ndiatenda Ndou, Ritesh Ajoodha, Ashwini Jadhav; 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS) (conference date: 21-24 April 2021, publication date: 14 May 2021)

[12] *Musical genre classification of audio signals*; G. Tzanetakis, P. Cook; IEEE Transactions on Speech and Audio Processing (volume: 10, issue: 5, July 2002)

[13] *A tensor-based approach for automatic music genre classification*; E. Benetos, C. Kotropoulos; 2008 16th European Signal Processing Conference (conference date: 25-29 August 2008, pp. 1-4)

[14] Aggregate features and ADABOOST for music classification; J. Bergstra, N. Casagrande, D. Erhan, D. Eck, B. Kégl; Machine Learning (volume: 65, December 2006, pp. 473-484)

[15] *A comparative study on content-based music genre classification*; Tao Li, Mitsunori Ogihara, Qi Li; SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval (publication date: 28 July 2003, pp. 282-289)

[16] *Combining audio and symbolic descriptors for music classification from audio*; T. Lidy, A. Rauber, A. Pertusa, J. Iñesta (publication date: 2007)

[17] *Explaining deep convolutional neural networks on music classification*; K. Choi, G. Fazekas, and M. Sandler; (publication date: 2016).

[18] *Music Genre Classification Using Masked Conditional Neural Networks*; Medhat F., Chesmore D., Robinson J. (2017) In: Liu D., Xie S., Li Y., Zhao D., El-Alfy ES. (eds) Neural Information Processing. ICONIP 2017. Lecture Notes in Computer Science, vol 10635. Springer, Cham.

[19] *Using CNNs and RNNs for Music Genre Recognition*; Priya Dwivedi, Towards Data Science, 2018, link (visited 30.10.21): https://towardsdatascience.com/using-cnns-and-rnns-for-music-genre-recognition-2435fb2ed6af

[20] *A Benchmark Dataset for Audio Classification and Clustering*; H Homburg, I Mierswa, B Möller, K Morik - ISMIR, 2005

[21] *FMA: A Dataset For Music Analysis*; ISMIR 2017 Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, Xavier Bresson

[22] *Neural Network Music Genre Classification*; N. Pelchat and C. M. Gelowitz, Canadian Journal of Electrical and Computer Engineering, vol. 43, no. 3, pp. 170-173, Summer 2020, doi: 10.1109/CJECE.2020.2970144.

[23] *Parallel Recurrent Convolutional Neural Networks-Based Music Genre Classification Method for Mobile Devices*; R. Yang, L. Feng, H. Wang, J. Yao and S. Luo, in IEEE Access, vol. 8, pp. 19629-19637, 2020, doi: 10.1109/ACCESS.2020.2968170.

[24] *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, Ameet Talwalkar; 18(185):1 - 52, 2018.

[25] *Music Genre Recognition Using Deep Neural Networks and Transfer Learning*Ghosal, D., Kolekar, M.H. Proc. Interspeech 2018, 2087-2091, doi: 10.21437/Interspeech.2018-2045