POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Praca dyplomowa inżynierska

Tworzenie muzyki za pomocą sztucznych sieci neuronowych

autor: Grzegorz Kazana

kierujący pracą: dr inż. Grzegorz Baron

Gliwice, styczeń 2020

załącznik nr 2 do zarz. nr 97/08/09

Oświadczenie

Wyrażam zgodę / Nie wyrażam zgody* na udostępnienie mojej pracy dyplomowej / rozprawy doktorskiej*.

Gliwice, dnia 13 stycznia 2020

.....
(podpis)

.....
(poświadczenie wiarygodności
podpisu przez Dziekanat)

* podkreślić właściwe

Oświadczenie promotora

Oświadczam, że praca „Tworzenie muzyki za pomocą sztucznych sieci neuronowych" spełnia wymagania formalne pracy dyplomowej inżynierskiej.

Gliwice, dnia 13 stycznia 2020

.....

(podpis promotora)

Spis treści

Rozdział 1

Wstęp

Współczesne zastosowania metod uczenia maszynowego są bardzo szerokie. Codziennie, często nieświadomie, korzystamy z usług i produktów których działanie nie byłoby tak skuteczne lub nawet możliwe bez sztucznej inteligencji. Z pośród niezliczonej ilości problemów praktycznie niemożliwych do rozwiązania podejściami klasycznymi, takich jak detekcja i klasyfikacja obiektów na obrazach lub rozpoznawanie mowy, duża część zostaje rozwiązana z wykorzystaniem szerokiej gamy algorytmów oraz modeli uczenia.

Zadania stawiane przed systemami sztucznej inteligencji można podzielić między innymi na problemy:

- klasyfikacji - polegające na przypisaniu obiektu do określonej kategorii, na przykład rozpoznawanie obiektów na obrazie
- grupowania - polegające na wyodrębnieniu przypadków o podobnych cechach lub współwystępujących, na przykład systemy sugerujące przedmioty w sklepach internetowych na podstawie dotychczasowych wyborów
- regresji - polegające na oszacowaniu wartości ciągłej dla poszczególnych cech, na przykład przewidywanie cen produktów.

Coraz szersze zastosowanie metod uczenia maszynowego prowadzi do powstawania pytań poruszających temat ich granic możliwości i zastosowań. Pomimo powszechności i znaczenia zastosowań przytoczonych kategorii problemów, nie można w nich dostrzec przejawów kreatywności.

Jednym z problemów, którego rozwiązanie byłoby silnym argumentem w debacie na temat kreatywności i zdolności twórczych systemów komputerowych jest komponowanie muzyki.

1.1 Cel pracy

Celem pracy jest analiza problemu generacji utworów muzycznych za pomocą sztucznych sieci neuronowych wraz z oceną wyników. Krokiem będącym fundamentem do realizacji celu jest również dobór odpowiedniego sposobu przetwarzania plików muzycznych do postaci numerycznej.

Pomyślna realizacja celów i otrzymanie akceptowalnych rezultatów oznaczałoby, że metody opracowane w procesie badawczym mają potencjał wdrożeniowy. Zbadane algorytmy oraz kroki mogą potencjalnie znaleźć zastosowanie w zadaniach ekstrakcji stylu, augmentacji istniejących zbiorów danych, lub w optymistycznym scenariuszu jako narzędzie wspierające proces twórczy muzyków.

1.2 Zakres pracy

Do zadań autora pracy należy

- dobór zbioru danych
- dobór, implementacja i analiza metod przetwarzania plików muzycznych
- dobór i parametryzacja architektury modelu uczenia maszynowego
- wykorzystanie przygotowanych danych w procesie uczenia i syntezy nowych próbek
- ocena uzyskanych wyników

1.3 Struktura pracy

Niniejsza praca w kolejnych rozdziałach prezentuje analizę różnych podejść do kolejnych etapów procesu implementacji systemu, którego celem jest generowanie utworów muzycznych.

Treścią 2. rozdziału jest przybliżenie dziedziny pracy oraz odniesienie do istniejących prac o powiązanej tematyce.

W rozdziale 3. poruszono problem wyboru formatu danych, oraz przedstawiono opis zbioru na którym były przeprowadzane eksperymenty. Przedstawiono również potencjalne trudności wiążące się z wyborem poszczególnych źródeł danych.

Rozdział 4. traktuje o różnych podejściach do zagadnienia reprezentacji wybranego formatu danych w postaci numerycznej. Rozdział zawiera również opis wad i zalet każdego z podejść.

Treścią 5. rozdziału jest przestawienie obranego typu modelu uczenia maszynowego oraz wybranej architektury, razem z opisem procesu uczenia i generowania próbek.

Rozdział 6. poświęcono technicznym aspektom związanym z implementacją poszczególnych etapów procesu przetwarzania danych i uczenia modelu.

Rozdziały 7. i 8. są poświęcone analizie wyników, wnioskom i sugestiom zmian w procesie mających na celu poprawienie otrzymywanych rezultatów.

Rozdział 2

Analiza tematu

Temat pracy wymaga realizacji dwóch etapów: opracowania danych i wykorzystania modelu uczenia maszynowego.

Celem zadania doboru i analizy metod obróbki plików muzycznych jest przekształcenie danych do postaci użytecznej dla algorytmów uczenia maszynowego. Do istotnych aspektów opracowanego procesu należy między innymi stopień kompresji danych, sposób wyrażania relacji między przykładami, możliwość bezstratnej transformacji odwrotnej.

Zadanie generowania muzyki polega na ekstrakcji pewnych cech charakterystycznych przykładowych utworów, na przykład stylu konkretnego artysty, i wykorzystaniu ich przy syntezie tworzonych próbek.

2.1 Wprowadzenie do dziedziny

Jedną z najbardziej rozpowszechnionych metod uczenia maszynowego jest zastosowanie sieci neuronowych. Zaproponowane w latach czterdziestych i rozwijane w drugiej połowie XX wieku [7], sztuczne sieci neuronowe czerpią inspirację ze sposobu funkcjonowania ludzkiego mózgu, który jest zbudowany z komórek nerwowych - neuronów. Połączenia między komórkami są modelowane poprzez wagi, reprezentujące siłę połączenia, a zjawisku aktywacji komórek w sieci odpowiada operacja ważonej sumy informacji pochodzących z połączonych neuronów oraz wag tych połączeń.

W latach siedemdziesiątych opracowano algorytm uczenia sztucznych sieci z powodzeniem wykorzystywany do dziś - propagację wsteczną. Metoda ta polega na minimalizacji błędu predykcji poprzez regulację wag w oparciu o pochodną funkcji błędu.

Zastosowania sieci neuronowych są bardzo szerokie i różnorodne. Zasadą będącą podporą uniwersalności sztucznych sieci neuronowych jest twierdzenie stanowiące o ich możliwości aproksymacji dowolnej funkcji ciągłej w zamkniętym przedziale [2].

Mimo tego, że przytoczona metoda uczenia maszynowego jest znana od wielu dekad, dopiero w ostatnich latach przeżywa swoisty renesans spowodowany wzrostem dostępnej mocy obliczeniowej oraz możliwością zastosowania w procesie uczenia akceleratorów (na przykład kart graficznych) znacząco przyśpieszających równoległe operacje matematyczne.

Ponieważ dane reprezentujące muzykę mają postać sekwencji rozłożonej w czasie, konieczne jest wykorzystanie sieci mających możliwość agregacji stanu, czyli architektur posiadających pamięć. Sieciami spełniającymi powyższy warunek są modele należące do grupy rekurencyjnych sieci neuronowych. Najprostszym przykładem rekurencyjnej sieci neuronowej jest sieć na wejście której przekazywany jest również stan wyjść z analizy poprzedniego elementu sekwencji.

Niestety, taka architektura jest narażona na wiele problemów, takich jak trudność tworzenia powiązań pomiędzy odległymi elementami sekwencji oraz zjawisko znikającego lub eksplodującego gradientu.

Architekturą rozwiązującą powyższe problemy jest architektura Long Short-term Memory Network (LSTM) [5]. Jej pomysłodawca rozszerza klasyczną rekurencyjną sieć neuronową o globalną pamięć niezależną od stanu wyjść. Dane mogą zostać wprowadzone lub wymazane z pamięci dzięki warstwom bramkującym. Kosztem większych możliwości architektury jest zwielokrotnienie ilości parametrów, co może przekładać się na dłuższy czas uczenia.

2.2 Założenia pracy

Na potrzeby pracy przyjęto definicję muzyki mówiącą, że muzyką nazywamy ciągi dźwięków tworzące kompozycyjną całość.

Utwory muzyczne można analizować pod wieloma względami, takimi jak:

- rytmiczność - organizacja dźwięków w czasie
- melodyczność - sposób zestawiania następujących po sobie dźwięków
- harmoniczność - spójność i ład występujący między dźwiękami
- dynamika - zróżnicowanie siły dźwięków

W kontekście pracy skupiono się na dwóch aspektach wynikających z powyższej definicji: rytmiczności i tonalności.

2.3 Przegląd literatury

Powstało wiele prac poruszających tematykę tworzenia muzyki za pomocą uczenia maszynowego, lecz każda z nich wyróżnia odmiennym podejściem do problemu.

Elementem wspólnym dużej części prac jest wykorzystany format danych wejściowych. W większości analizowanych artykułów, twórcy decydują się na wykorzystanie danych muzycznych w formacie midi [4, 9, 14, 15], aczkolwiek równie popularnym wyjściem jest skorzystanie z plików notacji ABC które miało miejsce w pracach [1, 13]. Opis formatu ABC ma miejsce w sekcji 3.1.2. Najrzadsze podejście opiera się na wykorzystaniu nieprzetworzonych plików audio, na przykład w formacie wave [10].

Jedną z różnic występujących między podejściami jest sposób obróbki i tworzenia wewnętrznej reprezentacji danych, gdyż praktycznie wszyscy autorzy proponują własne rozwiązanie problemu. W przypadku danych tekstowych w formacie ABC, częstym elementem jest ograniczenie lub usunięcie meta informacji zawartych w plikach [1, 13] oraz pominięcie białych znaków. Konkretne znaki zapisu są albo przedstawione jako wektory kodu 1 z N [13], lub przekształcane są kolejne wartości liczb naturalnych [1]. W pracach korzystających z plików midi, częstym podejściem do problemu reprezentacji wysokości dźwięku jest zastosowanie kodu 1 z N [15]. Większym zróżnicowaniem cechuje się podejście do reprezentacji danych

w wymiarze czasu. Duża część podejść opiera się na restrykcji możliwych wartości rytmicznych do ustalonego dyskretnego zbioru [4, 14], podczas gdy stosowane są również podejścia korzystające bezpośrednio z taktowania plików midi [9].

Kluczowym aspektem wszystkich prac jest dobór modelu uczenia maszynowego. Najbardziej popularnym wyborem są rekurencyjne sieci neuronowe [1, 9, 12, 13], ale opisywane są również inne podejścia, na przykład:

- modele kontekstowe, takie jak Continuous Bag-of-Words [1],
- modele generatywno-adwersyjne (Generative Adversial Networks) [1, 9]
- modele oparte o sieci konwolucyjne [10, 15].

Aspektem wartym zwrócenia uwagi, są również poczynione założenia i nałożone ograniczenia. Niektóre metody przytoczone przez autorów prac wymagają:

- pominięcia elementów polifonicznych utworów [4, 13],
- ograniczenia zbioru danych do utworów będących w ustalonej tonacji i/lub ustalonym metrum [13, 14],
- ograniczenia występujących wartości rytmicznych [4].

2.4 Odniesienie do istniejących prac

Niniejsza praca wpisuje się w kanon prac w których zdecydowano się korzystać z plików typu midi, lecz w przeciwieństwie do dużej części dostępnych publikacji zdecydowano się położyć duży nacisk na opis wewnętrznej postaci danych i procesu ich otrzymywania.

Podjęcie powyższej decyzji wynika z trudności znalezienia szczegółowych opisów tego kroku procesu w artykułach i źródłach dostępnych online. Kolejnym powodem motywującym powyższy wybór jest ufność względem poglądu stwierdzającego i podkreślającego istotę kroku przygotowania danych w procesie uczenia maszynowego.

2.5 Opis narzędzi

Podczas implementacji korzystano z narzędzi:

- Google Colab - usługa będąca środowiskiem obliczeniowym przystosowanym do uruchamiania skryptów w języku Python, z dostępem do akceleratorów obliczeniowych w postaci kart graficznych i jednostek tensorowych,
- Tensorflow - biblioteka języka Python służąca do wydajnych obliczeń, ułatwiająca tworzenie i uczenie głębokich sieci neuronowych,
- Jupyter notebook - środowisko uruchomieniowe języka Python, umożliwiające przejrzystą ilustrację wykonywanych operacji,
- mido - biblioteka ułatwiająca otwieranie i przetwarzania plików w formacie midi.

Rozdział 3

Dobór danych

Jednym z kluczowych aspektów każdego procesu analizy danych i uczenia maszynowego jest odpowiedni dobór danych. Decyzje podjęte na tym etapie pociągają za sobą konsekwencje w kolejnych etapach procesu. W zależności od postaci danych wejściowych, określany jest konieczny nakład pracy w procesie ich obróbki. Ponadto, odpowiedni dobór danych ma znaczący wpływ na otrzymywane wyniki.

W przypadku danych muzycznych należy zwrócić uwagę między innymi na cechy takie jak:

- dostępność danych treningowych,
- sposób reprezentacji upływu czasu,
- sposób wyrażenia wysokości dźwięku,
- jasność przedstawiania zależności między dźwiękami,
- stopień kompresji,
- złożoność samego formatu oraz możliwość istnienia niepoprawnych reprezentacji.

3.1 Porównanie formatów danych

Wybór odpowiedniego formatu danych został poprzedzony analizą zbioru popularnych formatów muzycznych. Zbiór porównanych rodzajów plików ograniczono do plików audio, notacji ABC oraz midi. Opisane formaty zostały dobrane na podstawie ich odmienności względem siebie.

3.1.1 Pliki audio

Najmniej restrykcyjnymi formatami plików audio są formaty będące zapisem amplitudy fali akustycznej. Przykładami plików tego rodzaju są pliki WAV oraz AIFF. Kluczowym parametrem plików audio jest częstotliwość próbkowania. Od niej zależy zakres możliwych do wyrażenia częstotliwości. Typową wartością dla plików WAV jest 44100Hz, co według twierdzenia Nyquista-Shannona przekłada się na zakres częstotliwości ograniczony wartością 22050Hz [11], będącą bliską granicy słyszalności ludzkiego aparatu słuchowego.

W przeciwieństwie do pozostałych opisanych formatów, wysokości dźwięków wyrażone są jedynie poprzez sekwencje zmian amplitudy sygnału, przez co ich ekstrakcja wymagałaby wykorzystania transformaty Fouriera.

Największą zaletą i jednocześnie wadą tego formatu jest jego swoboda. Postać fali akustycznej pozwala na przedstawienie każdego dźwięku będącego w paśmie przenoszenia, zarówno złożonych wieloinstrumentowych kompozycji muzycznych, prostych melodii, jak i dźwięków nie podlegających pod definicję muzyki. Prowadzi to do niskiego stopnia kompresji informacji, wymuszającego złożone metody obróbki i skomplikowane modele uczenia.

3.1.2 Notacja ABC

Notacja ABC jest jedną z cyfrowych postaci klasycznego zapisu nutowego. Dane w plikach ABC przedstawione są za pomocą znaków ASCII.

Ponieważ jest to odpowiednik zapisu nutowego, oznacza to że zarówno wysokości dźwięków, jak i wartości rytmiczne należą do dyskretnego zbioru. Fakt ten nakłada spore ograniczenie na treść plików, lecz w kontekście muzyki są to ograniczenia bardzo pomocne.

Poza informacją o samym dźwięku i jego wartości rytmicznej, format zawiera również metadane utworu takie jak autor, sygnatura i tempo. Przykładowy fragment utworu w notacji ABC przedstawia rysunek 3.1.

Dużą zaletą tego formatu jest jego wysoki stopień kompresji. Za pomocą relatywnie krótkich ciągów znaków możliwe jest opisanie znacznej część utworu muzycznego. Głównym mankamentem notacji ABC w kontekście niniejszej pracy jest fakt, że nie wszystkie ciągi znaków są poprawnymi zapisami w notacji ABC. Ozna-

```
X:1
T:Speed the Plough
M:4/4
C:Trad.
K:G
|:GABc dedB|dedB dedB|c2ec B2dB|c2A2 A2BA|
  GABc dedB|dedB dedB|c2ec B2dB|A2F2 G4:|
|:g2gf gdBd|g2f2 e2d2|c2ec B2dB|c2A2 A2df|
  g2gf g2Bd|g2f2 e2d2|c2ec B2dB|A2F2 G4:|
```

Rysunek 3.1: Przykładowy zapis w notacji ABC

cza to, że pojedyncze błędy w generowanych sekwencjach mogą powodować niepoprawność całego utworu. W przypadku niemożliwości wyuczenia modelu tworzenia całkowicie poprawnych sekwencji niemożliwa byłaby transformacja dowolnej sekwencji do pliku midi. Chcąc zapewnić możliwość generacji utworów muzycznych konieczna byłaby walidacja i implementacja obsługi i poprawiania błędów w zapisie.

3.1.3 Format midi

Pliki w formacie midi są zapisem wiadomości przesyłanych protokołem komunikacyjnym o ten samej nazwie. Każdy plik midi składa się z kanałów oraz ścieżek. Poszczególne kanały reprezentują poszczególne instrumenty biorące udział w nagraniu. Na poszczególnych ścieżkach znajdują się wiadomości reprezentujące zdarzenia w utworze. Do najczęstszych z nich należą: note_on, note_off, set_tempo.

W przypadku wiadomości związanych z rozpoczęciem lub zakończeniem dźwięku dołączony jest również numer dźwięku z zakresu 0-127. Wartości tego parametru odpowiadają kolejnym dźwiękom skali dwunastotonowej poczynając od C-1 i kończąc na G9. Dostępne dźwięki są nadzbiorem dźwięków dostępnych na 88 klawiszowej klawiaturze klasycznego fortepianu.

Dodatkowym parametrem każdej wiadomości jest wartość time, będąca ilością taktów (ang. 'ticks') które upłynęły od poprzedniej wiadomości. Oznacza to, że długość trwania dźwięku można wyznaczyć poprzez zsumowanie parametrów time wiadomości występujących pomiędzy odpowiadającymi sobie zdarzeniami note_on i note_off.

Możliwa jest konwersja wyznaczonej wartości zarówno na czas w milisekundach i wartości rytmiczne. Stałe potrzebne do tych przekształceń są naczęściej zawarte w meta wiadomościach znajdujących się na początku utworu.

Zaletą formatu midi jest dyskretna reprezentacja wysokości dźwięku oraz pozwalająca na elastyczną interpretację postać upływu czasu.

3.1.4 Podsumowanie

Wszystkie omówione formaty danych mają wady i zalety. Ponieważ dostępne są zbiory utworów w każdym z przytoczonych formatów, wybór nie był ograniczony dostępnością danych, a decyzja została podjęta na postawie ich cech.

W dalszej części pracy opisywane podejścia będą tyczyć się jedynie plików w formacie midi, na którego wybór zdecydowano się drogą eliminacji. Pliki audio odrzucono przez wymaganą złożoność wstępnej obróbki, a notację ABC z powodu obaw związanych z trudnościami zagwarantowania syntaktycznej poprawności generowanych ciągów znaków.

3.2 Opis wybranego zbioru danych

Do dalszych eksperymentów, opierających się na różnych sposobach tworzenia numerycznej reprezentacji danych wybrano stosunkowo mały zbiór zawierający utwory z serii gier Pokemon. Zbiór składa się z dziesięciu utworów o średniej długości wynoszącej 90 sekund. Kompozycje zawierają zarówno elementy monofoniczne, jak i polifoniczne. Główną motywacją wyboru tego zbioru była jego mała objętość, co pozwalało na szybsze testowanie metod obróbki danych i procesu uczenia.

Rozdział 4

Sposoby reprezentacji danych

Znaczna część modeli uczenia maszynowego, w tym sieci neuronowe, wymagają danych w postaci numerycznej. Odpowiednio dobierając proces przekształcania danych możliwe jest nie tylko dostosowanie danych do metody uczenia, ale również uwypuklenie informacji prawdziwie w nich istotnych, co potencjalnie skróci i ułatwi proces uczenia modelu.

Poniższy rozdział zawiera analizę różnych podejść do przedstawionego problemu.

4.1 Reprezentacja wysokości dźwięków

Pierwszym z wymiarów danych muzycznych są wysokości dźwięków. W przypadku plików midi oryginalnie są to liczby z przedziału 0-127. Są to już dane numeryczne, które można by teoretycznie wykorzystać bezpośrednio do uczenia modelu. W takim przypadku zadanie sprowadziłoby się do problemu regresji, ponieważ model próbowałby oszacować wartość numeryczną, którą następnie trzeba by ograniczyć do liczb całkowitych z dozwolonego zakresu.

Jedną z wad tego rozwiązania, jest przedstawienie fałszywych relacji między dźwiękami. Dźwięki znajdujące się w bliższym sąsiedztwie byłyby traktowane jako bardziej sobie podobne. Takie założenie w muzyce nie zawsze jest prawdą. Przykładowo, dźwięki 60 i 66 (C4 i F#4), znajdują się relatywnie blisko, mimo tego że występuje między nimi interwał trytonu, będący jednym z najsilniejszych dyso-

nansów w skali dwunastotonowej. Z drugiej strony, dźwięki 60 i 84 (C4 i C6) dzieli spora odległość, mimo tego że jest to ten sam dźwięk zagrany dwie oktawy wyżej.

Z niniejszymi problemami można się mierzyć poprzez stosowanie poniższych podejść.

4.1.1 Kody 1 z N i M z N

Klasycznym sposobem na rozwiązanie powyższego problemu jest zastosowanie kodu 1 z N. W przypadku informacji o dźwiękach midi, oznaczałoby to wypełnienie wektora 128 zerami i postawienie jedynki na pozycji reprezentującej dany dźwięk. Zaletą tego rozwiązania jest możliwość reprezentacji melodii polifonicznych, co przekształciłoby powyższy kod w kod M z N, gdzie M to ilość dźwięków wybrzmiewających jednocześnie w danym momencie sekwencji. Mimo tego, że w taki sposób można uniknąć problemu fałszywych zależności między sąsiednimi dźwiękami, to nie są reprezentowane również zależności mogących być użytecznymi w procesie uczenia, takich jak w przypadku reprezentacji dźwięków odległych o oktawy. Kolejnym problemem tego podejścia jest również znaczący wzrost wymiarowości danych i spadek ich kompresji, co przełoży się na wolniejszy proces uczenia.

4.1.2 Wektory zanurzone

Problem dużej wymiarowości i niemożliwości wyrażenia relacji między obiektami nie jest charakterystyczny tylko i wyłącznie dla analizy dźwięku. Zagadnienie to występuje również w dziale przetwarzania języka naturalnego. Również w kontekście modeli operujących na fragmentach języka każdy wyraz stanowi jeden element z N, gdzie N jest rozmiarem słownika - zbioru wszystkich wyrazów.

Rozwiązaniem cieszącym się dużą popularnością jest algorytm Word2Vec zaproponowany w roku 2013 [8]. Jest to algorytm służący do wyznaczenia ukrytej reprezentacji kodów 1 z N w postaci wektorów wartości ciągłych o dużo mniejszym wymiarze. Metoda opiera się na założeniu twierdzącym, że wektory o podobnym znaczeniu występują w sąsiedztwie podobnych lub nawet tych samych wyrazów. Ponieważ to założenie jest również prawdziwe w kontekście muzyki przyjęto użycie algorytmu Word2Vec za podstawne.

Jedyną modyfikacją potrzebną do wykorzystania tej metody do przykładów polifonicznych, było traktowanie całych wielodźwięków jako pojedynczych obiektów w słowniku. Pozwoliła na to transformacja wektorów kodu M z N na ciągi znaków reprezentujących wybrzmiewające dźwięki.

4.2 Reprezentacja czasu

Drugim wymiarem danych muzycznych jest czas. Ponownie, wartość oznaczająca upływ czasu mogłaby być użyta bezpośrednio, znów sprowadzając problem do regresji. Dodatkowo, rozsądnym krokiem byłoby znormalizowanie wartości w impulsach midi do arbitralnie wybranej wartości reprezentującej konkretną wartość rytmiczną. Przykładowym mapowaniem mogłoby być przyjęcie wartości 1 dla ćwierćnut.

Głównym mankamentem tego rozwiązania, jest spowodowane naturą regresji ryzyko niemożliwości precyzyjnego wyuczenia dokładnych wartości rytmicznych, a jedynie oscylowanie między wartościami. W takim przypadku generowana muzyka mogłaby sprawiać wrażenia arytmicznej, gdyż następujące dźwięki nie tworzyłyby spójnych taktów.

4.2.1 Próbkowanie

Alternatywną metodą wyrażenia upływu czasu jest próbkowanie. Z przyjętą częstotliwością próbkowania, możliwe byłoby tworzenie listy aktualnie wybrzmiewających dźwięków i zapełnienie macierzy o wymiarach $t \times n$, gdzie n to wymiarowość reprezentacji wysokości dźwięku, a t to ilość pobranych próbek w danym fragmencie utworu.

Ponieważ dźwięki zawsze trwają wielokrotność pewnej wartości, można twierdzić, że łatwiejsze byłoby odtworzenie rytmicznego charakteru muzyki.

Wadą tego podejścia jest konieczność doboru częstotliwości próbkowania. Zbyt niskie wartości uniemożliwiłyby precyzyjne przestawienie utworu. Natomiast, wraz z wyborem większej częstotliwości próbkowania, te same dźwięki byłyby reprezentowane przez większą ilość wektorów, co powodowały oddalenie następujących po sobie dźwięków w sekwencji i utrudnienie uczenia.

4.2.2 Grupowanie długości dźwięków

Podejściem łączącym zalety obydwu rozwiązań, jest przeprowadzenie grupowania znormalizowanych wartości parametru czasu w odniesieniu do ustalonej wartości rytmicznej. W ten sposób możliwe byłoby odzwierciedlenie wartości rytmicznych występujących w zbiorze danych i przedstawienie ich w dyskretny sposób bez arbitralnego określania zbioru dozwolonych wartości. Po podziale wartości rytmicznych na niezależne klasy problem zostanie przekształcony do problemu klasyfikacji 1 z N.

W celu otrzymania grup odpowiadających najczęstszym wartościom rytmicznym można skorzystać z algorytmów klasteryzujących takich jak k-Means [6] lub DBSCAN [3].

4.3 Podsumowanie

Na podstawie zalet i wad każdej z metod przedstawionych w powyższej analizie oraz eksperymentach nieudokumentowanych w sposób formalny, zdecydowano się na reprezentacje dźwięków w postaci wektorów zanurzonych, a wybraną reprezentacją czasu zostały dyskretne wartości uzyskane poprzez grupowanie.

Rozdział 5

Proces uczenia

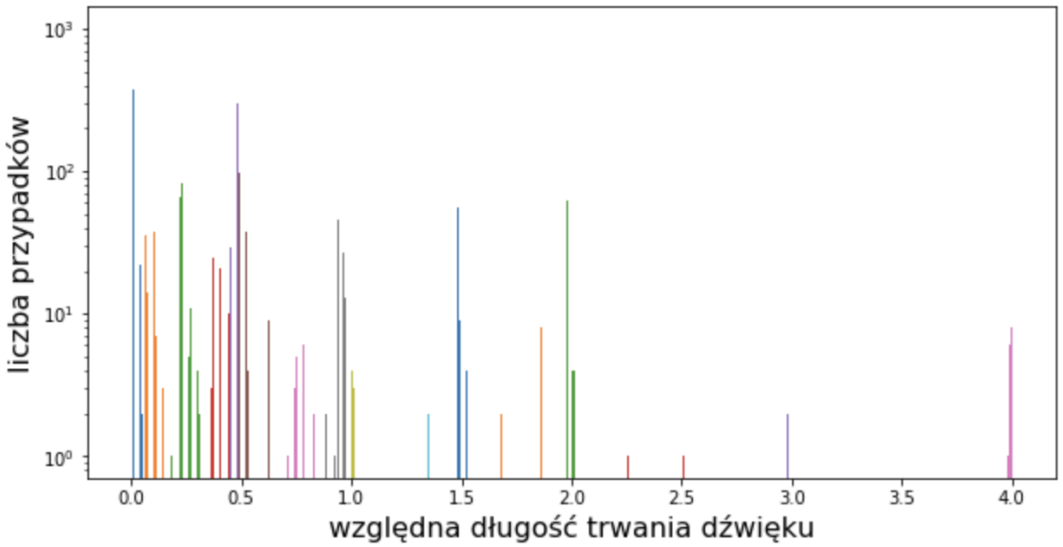
Do przetworzenia danych do wybranej postaci zostały wykorzystane modele wymagające uczenia. Ich opis został zawarty w sekcji 5.1 i 5.2. Sekcja 5.3 oraz następujące poświęcone są opisowi procesu uczenia głównego modelu, czyli sieci neuronowej której zadaniem jest ekstrakcja cech utworów treningowych.

5.1 Grupowanie wartości rytmicznych - k-Means

Model k-Means został wykorzystany w celu wyznaczenia klastrów długości trwania dźwięków. Powyższy zabieg umożliwił ograniczenie wszystkich występujących wartości rytmicznych do dyskretnego zbioru.

Model dzieli występujące przypadki na grupy w taki sposób, aby zminimalizować odległość między przypadkami należącymi do jednej grupy. Do głównych parametrów tego modelu należy docelowa ilość klastrów, ilość iteracji oraz niezależnych uruchomień modelu mająca na celu rozwiązania problemu zatrzymywania uczenia w lokalnym minimum.

Przykładowy wynik grupowania przedstawia rysunek 5.1.



Rysunek 5.1: Wykres przedstawiający rozkład długości trwania dźwięków oraz wynik grupowania. Dźwięki przydzielone do tej samej klasy są oznaczone tym samym kolorem.

5.2 Wyznaczanie wektorów zanurzonych - Word2Vec

Kolejnym zastosowaniem uczenia maszynowego w etapie przetwarzania danych było użycie algorytmu Word2Vec. Algorytm został użyty do wyznaczenia ciągłej reprezentacji rzadkich wektorów reprezentujących dźwięki.

Uczenie algorytmu Word2Vec polega na wyznaczaniu wag ukrytej warstwy wielowarstwowego perceptronu uczonego wektorami reprezentującymi kolejne słowa należące do kontekstu. Model wymagał określenia docelowej wymiarowości wektorów i rozmiaru kontekstu.

Użycie powyższego modelu pozwoliło przekształcić wektory kodu M z N, gdzie M to ilość jednocześnie wybrzmiewających dźwięków a N to ilość wsystkich możliwych dźwięków, na wektory wartości ciągłych o wymiarze 1×20.

5.3 Architektura modelu sieci neuronowej

Otrzymawszy dane w postaci sekwencji par wektorów opisujących dźwięki i ich czas trwania, przystąpiono do projektu architektury modelu. Z powodu dwuelementowej postaci danych zdecydowano się na zaprojektowanie modelu dwuwejściowego i dwuwyjściowego.

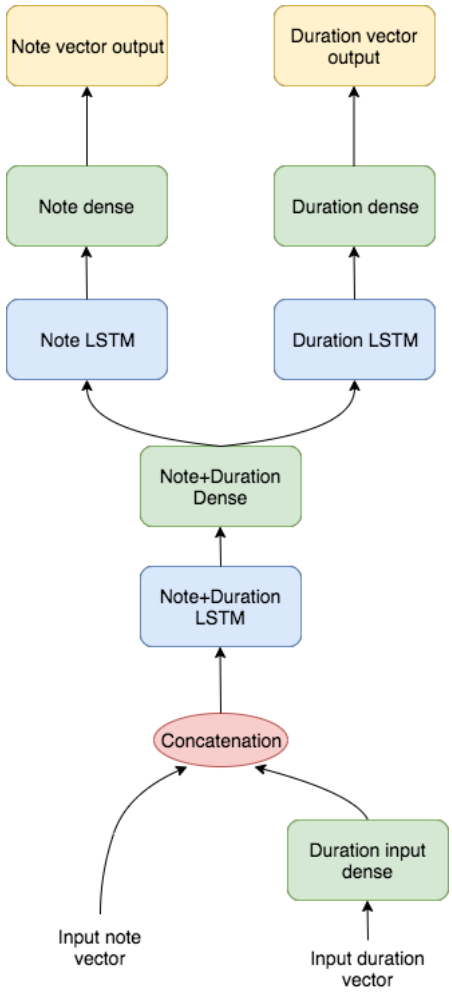
Ponieważ charakter wektorów w sekwencji jest różny - wartości ciągłe opisujące dźwięki i kategoryczny kod 1 z N opisujący długość dźwięku - przed złączeniem wektorów postanowiono przekształcić wektor 1 z N przez warstwę gęstą o wymiarze mniejszym niż N. Celem tej operacji było wprowadzenie konieczności przez sieć kompresji informacji, co przełoży się na wyznaczenie ciągłej reprezentacji kodu o mniejszej wymiarowości.

Następnie wektory zostają złączone, i ich sekwencje trafiają do warstw rekurencyjnej sieci LSTM, w której model ma szanse ekstrahować wiedzę o zależnościach między kolejnymi elementami sekwencji. W tym kroku analizowane są dane o wysokości i długości trwania dźwięków jednocześnie.

Otrzymywane wektory reprezentacji ukrytej są połączone z osobnymi, mniejszymi sieciami LSTM odpowiedzialnymi za przekształacnie reprezentacji ukrytej zawierającej za równo informacje o wysokościach i długościach trwania dźwięków, do postaci w której nacisk jest kładziony jedynie na jeden z dwóch powyższych aspektów. Wektory otrzymywane na wyjściach osobnych sieci LSTM są przekształcane przez warstwy gęste będące wyjściami modelu. Wspomniane przekształcenie przez warstwę gęstą umożliwia silniejsze uniezależnienie kodu wyjściowego od wewnętrzej reprezentacji danych na których przeprowadzane są operacje w sieciach rekurencyjnych, co pozostawia większą swobodę przy optymalizacji wag i wewnętrznej postaci danych.

Celem wprowadzenia dodatkowych sieci LSTM do procesu uczenia modelu był podział zadania pozwalający na lepszą specjalizację modeli, które były uczone na podstawie jednej części danych w izolacji. Korzystnym efektem ubocznym tej operacji jest uzyskanie zaplanowanej struktury modelu dwuwyjściowego.

Opracowaną strukturę modelu przedstawia rysunek 5.2.



Rysunek 5.2: Diagram przestawiający architekturę modelu.

Z powodu różnego charakteru otrzymywanych wektorów, wyjścia modelu były uczone odzielnymi funkcjami błędu. Wyjście odpowiedzialne za dźwięki uczone jest funkcją błędu odpowiednią do problemu regresji - błędem średniokwadratowym, wyjście klasyfikujące długość dźwięku funkcją odpowiednią dla zadania klasyfikacji 1 z N - kategorycznej entropii krzyżowej.

5.4 Parametry modelu

Podczas procesu uczenia najistotniejszymi parametrami były:

- rozmiar okna, czyli długość uczonych sekwencji,
- ilość przykładów w wiązce, czyli ile sekwencji było uczonych jednocześnie poprzedzając pojedyncze przeprowadzenie algorytmu propagacji wstecznej,
- rozmiar głównej sieci LSTM.

5.5 Proces uczenia

Podczas uczenia modelu zdecydowano się na dynamiczny rozmiar okna, będący liczbą losową z zakresu <15, 50>. Motywacją tego wyboru były testowe uruchomienia modelu potwierdzające zdroworozsądkową intuicję mówiącą, że większe wartości umożliwią uczenie dłuższych powtarzających się motywów, a krótsze wartości umożliwią uczenie lokalnego następstwa nut.

Dodatkowo, warto przybliżyć sposób tworzenia przypadków testowych. Naiwnym podejściem byłoby jednokrotne utworzenie zbioru danych poprzez utworzenie wszystkich możliwych podciągów o zadanej długości (rozmiarze okna) i dalsze używanie ich jako statycznej listy przypadków. Takie rozwiązanie nie umożliwiałoby dynamicznej zmiany długości sekwencji, i wymagałoby dodatkowego kroku w procesie przetwarzania danych. Alternatywnym podejściem, na które się zdecydowano jest dynamiczne wybieranie losowych podciągów z utworów należących do zbioru danych, które odbywa się równolegle z uczeniem modelu. Główną obawą jaką można posiadać do takiego podejścia jest jego narzut czasowy podczas uczenia modelu, aczkolwiek zaobserwowano że tempo przygotowania i kolejkowania kolejnych przypadków testowych znacznie przewyższa tempo ich uczenia.

Podczas testowania różnych wartości pozostałych parametrów, zaobserwowano interesujący wpływ ilości przykładów w wiązce. Większe wartości, rzędu 16 lub 32 miały negatywny wpływ na jakość generowanych sekwencji. Mniejsze wartości prowadziły do przetrenowania modelu.

Rozmiar głównej sieci miał spodziewany efekt, zbyt duży prowadził do przetrenowania. Dla obecnego zbioru danych zdecydowano się zachować wartość 64 jednostek.

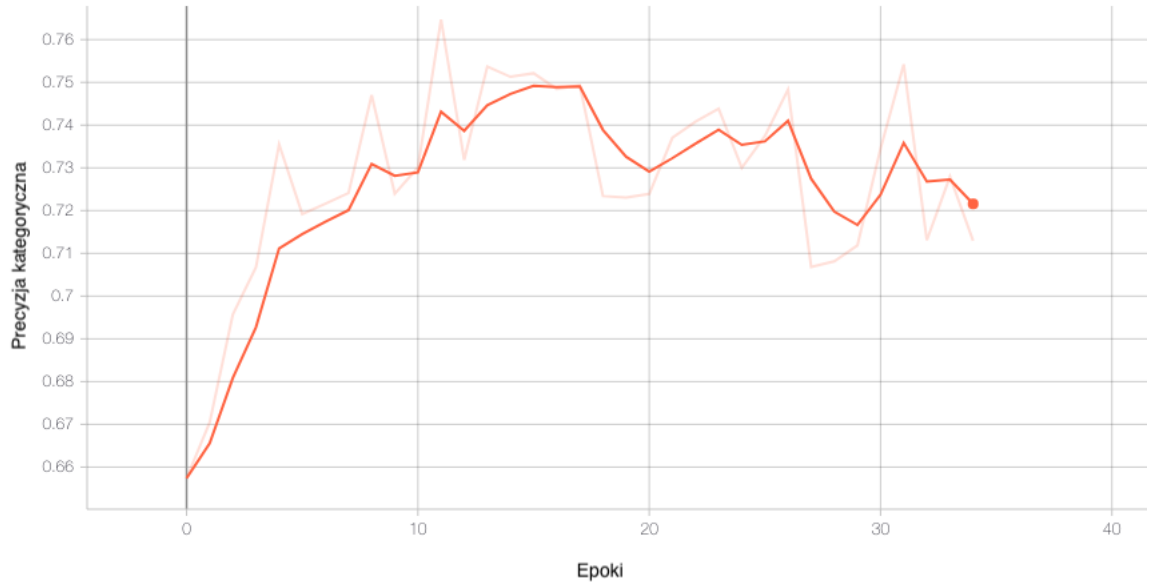
Podczas uczenia monitorowano metryki opisujące jakość predykcji i postęp uczenia modelu. Do mierzonych metryk należą funkcje błędu dla poszczególnych wyjść modelu, średni błąd kwadratowy wyjścia reprezentującego wektory dźwięków oraz kategoryczna dokładność stwierdzająca czy najwyższa wartość wektorów znajduje się na tym samym indeksie.

Proces uczenia podzielone był na epoki. W klasycznych zastosowaniach przez jedną epokę rozumie się jednokrotne wykorzystanie każdego przykładu z zbioru treningowego. Jednakże z powodu opisanego powyżej sposobu otrzymywania przypadków testowych, niemożliwe było wykorzystanie tej definicji. Zdecydowano się na arbitralne przyjęcie konkretnej ilości przypadków testowych jako jednostkę jednej epoki. Obraną wartością był jeden tysiąc.

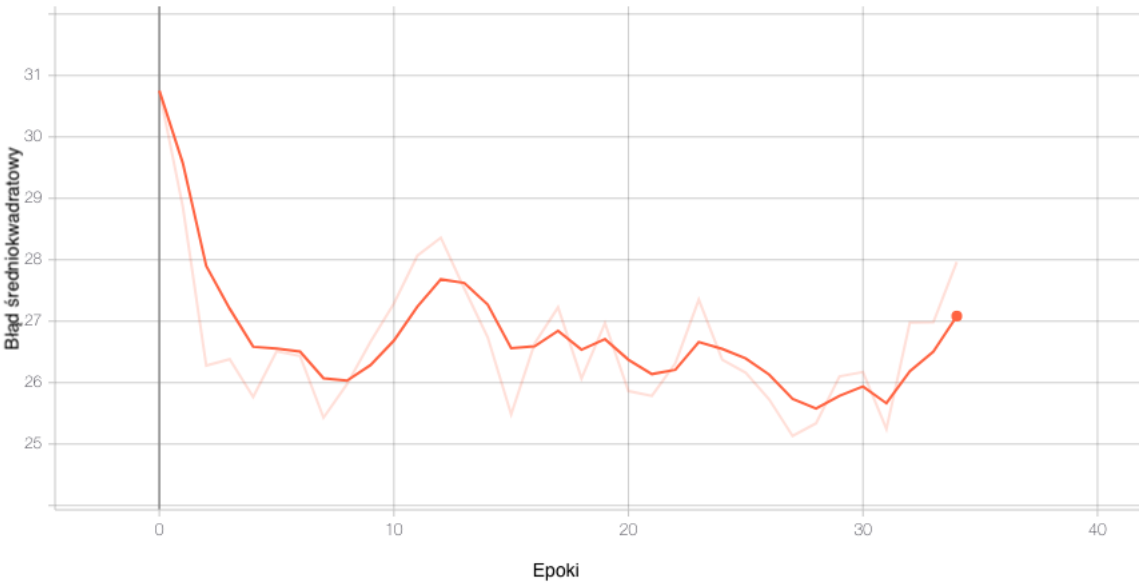
Po ukończeniu każdej epoki przeprowadzane były pomiary skuteczności modelu na danych wyjętych spoza korpusu danych treningowych. Ponieważ problematycznym byłoby wydzielenie sekwencji walidacyjnych z części środkowych utworów, zdecydowano się zarezerwowanie ostatnich 25 elementów sekwencji z każdego utworu. Na podstawie wyników powyższych miar, można wyciągać wnioski dotyczące postępu uczenia modelu.

Uczenie zaplanowano na okres 35 epok, co przełożyło się na czas bliski 20 minut. Dobrana ilość epok wynikała z obserwacji spadku trafności predykcji następującej po tej liczbie powtórzeń. Do uczenia wykorzystano akceleracją sprzętową w postaci karty graficznej dostępnej w usłudze chmury obliczeniowej Google Colab.

Rysunki 5.3 i 5.4 zawierają wykresy opisujące miary postępu uczenia podczas kroków walidacyjnych. Na wykresy naniesiono krzywe wygładzające wartości pomiarów w celu wyraźniejszego ukazania występujących tendencji.



Rysunek 5.3: Wartości miary precyzji predykcji długości trwania dźwięków



Rysunek 5.4: Wartości błędu średniokwadratowego predykcji wektora dźwięków

Na rysunkach można dostrzec trafność predykcji klasy długości trwania dźwięku, oraz błąd średniokwadratowy predykcji wektorów dźwięków. W pierwszej połowie uczenia na obydwu wykresach można zaobserwować pożądaną tendencję, czyli wzrost trafności i spadek błędu. Po kolejnych epokach postęp uczenia malał, lub nawet zachodził jego regres.

5.6 Generowanie próbek

Opierając się na założeniu, że wyuczony model posiada pewną wiedzę możliwe jest przejście do procesu generowania próbek.

5.6.1 Opis podejścia

Proces otrzymywania generowanych sekwencji zależny jest od rodzaju modelu. W przypadku modeli typu sequence-to-sequence, możliwe byłoby podanie losowego wektora na część modelu w którym dokonuje się przekształcenie jednowymiarowej postaci ukrytej na sekwencję. Przy zastosowaniu architektury opisanej w poprzednim rozdziale, konieczne będzie rozpoczynanie generacji sekwencją, dalej nazywaną ziarnem.

Ponieważ opracowany model przekształca sekwencje danych wejściowych na sekwencje o tej samej długości, sekwencja otrzymana po zadaniu ziarna będzie tej samej długości. Proponowanym rozwiązaniem tego problemu jest łączenie sekwencji wejściowych z sekwencjami wyjściowymi, i ponowne wprowadzenie jej do sieci. Krok ten jest powtarzany aż do otrzymania sekwencji o pożądanej długości.

5.6.2 Opisy ziaren

Kolejną kwestią, którą rozważono, jest opracowanie sposobów generowania ziaren. Oczywistym pomysłem jest zastosowanie wektorów o wartościach zerowych lub losowych. Ponadto, opracowano ziarna zawierające pojedynczy dźwięk, sekwencję losowych dźwięków i sekwencję dźwięków często współwystępujących. Dźwięki współwystępujące są obierane poprzez porównanie odległości między składowymi ich wektorów i wybranie wektorów o zbliżonych składowych.

Rozdział 6

Implementacja

Do implementacji wymaganych narzędzi oraz funkcjonalności skorzystano z języka Python. Głównym powodem tłumaczącym wybór tego języka programowania jest jego prostota, przenośność oraz szeroki zbiór wysokopoziomowych bibliotek często opartych na wydajnych implementacjach algorytmów napisanych w językach niskiego poziomu, takich jak C i C++.

Kolejną zaletą języka Python jest dostępność systemu zarządzania bibliotekami - *pip*, oraz narzędzi kontrolujących środowisko uruchomieniowe gwarantujące użycie poprawnej wersji interpretera i zależności na różnych systemach - *pipenv*. Dodatkowo, wykorzystano popularne w dziedzinie analizy danych środowisko zastępujące standardowy interaktywny interpreter - *Jupyter Notebook*. Pozwala ono między innymi na jednoczesną wizualizację i tworzenie procesu przetwarzania danych.

6.1 Wykorzystane biblioteki

Podczas implementacji projektu, korzystano z następujących bibliotek programistycznych:

- Tensorflow - biblioteka umożliwiająca przeprowadzanie zaplanowanych procesów obliczeń na tensorach, będących fundamentem procesu uczenia sieci neuronowych. Podczas pracy skorzystano z wysokopoziomowego interfejsu tensorflow.keras, zawierającego predefiniowane architektury i warstwy sieci.

Za pomocą interfejsu biblioteki możliwe było utworzenie oraz uczenie modelu o architekturze opisanej w rozdziale 5.3.

- numpy - biblioteka rozszerzająca funkcjonalność języka Python o wydajne i łatwe w obsłudze operacje numeryczne na n-wymiarowych tablicach. Zastosowanie biblioteki miało miejsce w procesie opracowania danych. Dodatkowo, przetworzone dane były zapisywane na dysk w formacie natywnym dla biblioteki numpy, ponieważ modele utworzone za pomocą narzędzia Tensorflow przystosowane są do współpracy z takim rodzajem danych.
- mido - biblioteka służąca do dekodowania plików w formacie midi. Umożliwia wczytywanie plików z dysku oraz odczyt zdarzeń zapisanych na poszczególnych ścieżkach pliku. Uzyskane zdarzenia posiadają wszystkie niezbędne atrybuty, takie jak: rodzaj wiadomości, wysokość dźwięku oraz ilość jednostek czasu które upłynęły od poprzedniego zdarzenia.
- scipy - biblioteka zawierająca implementację algorytmu k-Means użytego w procesie grupowania wartości rytmicznych dźwięków (4.2.2).
- gensim - biblioteka zawierająca implementację modelu Word2Vec wymaganego do wyznaczenia wektorów zanurzonych (4.1.2).

6.2 Specyfikacja zewnętrzna

Korzystanie z dostępnych metod obróbki danych umożliwiają skrypty znajdujące się w folderze *scripts*. Z powodu chęci zachowania jak największej elastyczności i możliwości wielokrotnego używania tego samego kroku procesu, wynikiem działania części skryptów są dane w postaci pośredniej lub metadane konieczne w innych krokach procesu.

Poniższa lista zawiera opis poszczególnych skryptów oraz ich przypadków użycia:

- **create_sparse_notes_sampled_time.py** - przeprowadza transformację plików midi do postaci w której dźwięki są reprezentowane za pomocą kodu M z N a upływ czasu poprzez próbkowanie (postać opisana w rozdziałach 4.1.1 i 4.2.1). Do parametrów wywołania należy folder plików źródłowych, folder docelowy oraz okres próbkowania. Możliwe jest również przeprowadzenie transformacji odwrotnej.

- **create_sparse_clustered_time.py** - transformuje pliki midi do postaci kodu M z N z grupowaną wartością rytmiczną (4.1.1 i 4.2.2). Podczas pracy skryptu uczony jest model k-Means odpowiedzialny za wyznaczenie zadanej ilości grup wartości rytmicznych. Parametrami wejściowymi jest folder źródłowy, folder docelowy oraz kierunek transformacji.
- **prepare_for_embedding.py** - jako dane wejściowe przyjmuje utwory w których informacja o wysokości dźwięku jest wyrażona za pomocą kodu M z N i generuje metadane konieczne do przeprowadzenia procesu wyznaczania wektorów zanurzonych (4.1.2). Do tworzonych danych należy przekształcony zbiór wejściowy pozbawiony powtórzeń bezpośrednio następujących po sobie dźwięków, słownik zawierający występujące w zbiorze wielodźwięki oraz ich liczność.

Pełną listę argumentów każdego z skryptów można uzyskać poprzez jego uruchomienie z flagą *–help*.

W przypadku procesów których automatyzacja była kłopotliwa lub istotna była wizualizacja przebiegu zadania, skorzystano z plików *Jupyter Notebook*.

Do najistotniejszych należą:

- **Word2Vec.ipynb** - zawiera proces uczenia modelu Word2Vec służącego do wyznaczania wektorów zanurzonych odpowiadających zapisowi M z N wielodźwięków. Umożliwia ręczne ustalenie ilości koniecznych powtórzeń uczenia na podstawie postępu treningu. Wynikiem działania procesu jest plik zawierający mapowanie konkretnych wielodźwięków na wektory zanurzone.
- **LSTM_clustered_time_sparse.ipynb** - zawiera proces uczenia sieci neuronowej za pomocą danych w postaci kodu M z N i grupowanych wartości rytmicznych. W trakcie uczenia oraz po jego zakończeniu możliwe jest zapisanie modelu na dysk.
- **LSTM_clustered_time_with_embedded.ipynb** - rozszerza funkcjonalność powyższego procesu o etap konwersji danych wejściowych do postaci w której wysokości dźwięków są w postaci wektorów zanurzonych. Konieczne jest wskazanie pliku wygenerowanego za pomocą procesu *Word2Vec.ipynb*.
- **LSTM_generating_clustered_time_embedded.ipynb** - zawiera proces generacji i transformacji próbek do postaci plików midi. Wymaga wska-

zania ścieżki do pliku modelu sieci neuronowej oraz słownika wektorów zanurzonych.

- **MeasuringDataset.ipynb** - przeprowadza proces wyznaczania miar opisujących zbiór danych treningowych. Miary zostały opisane w rozdziale 7.1.
- **MeasuringGenerated.ipynb** - przeprowadza proces wyznaczania miar dla wygenerowanych próbek za pomocą wskazanego modelu.

6.3 Specyfikacja wewnętrzna

Kod źródłowy został podzielony na moduły odpowiadające za poszczególne funkcjonalności projektu. Lista modułów projektu:

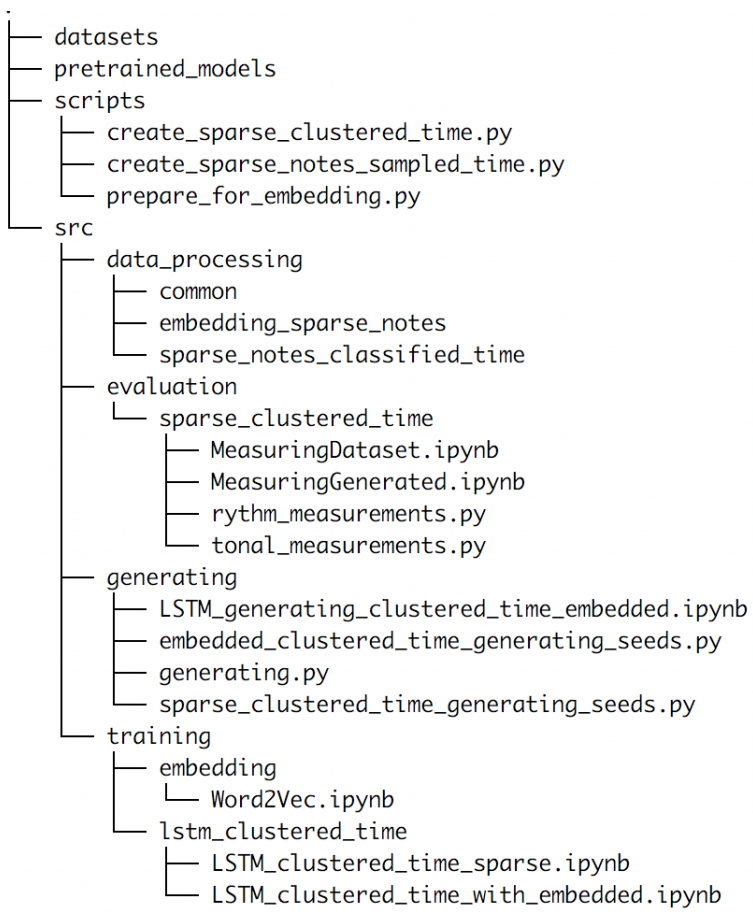
- **data_processing**
- **evaluation**
- **training**
- **generating**

Strukturę projektu przedstawia rysunek 6.1.

6.3.1 Moduł data_processing

Zawiera definicje funkcji odpowiedzialnych za przetwarzanie danych w postaci plików midi do postaci numerycznej i z numerycznej do midi. Moduł został podzielony na podfoldery ze względu na rodzaj numerycznej reprezentacji danych. Zaimplementowane w module metody umożliwiają uzyskanie danych w postaci:

- Kod M z N + próbkowanie
- Wektory zanurzone + próbkowanie
- Wektory zanurzone + czas jako zmienna ciągła
- Kod M z N + pogrupowane długości dźwięków
- Wektory zanurzone + pogrupowane długości dźwięków



Rysunek 6.1: Hierarchia folderów i modułów projektu

6.3.2 Moduł evaluation

Zawiera miary opisane w rozdziale 7.1. Funkcje wyznaczające wartości miar przystosowane są do współpracy z danymi wejściowymi w postaci kodu M z N dla wysokości dźwięków oraz kodu 1 z N w przypadku informacji o czasie jego trwania. Oznacza to, że w celu dokonania pomiarów próbek w których wysokości dźwięków opisują wektory zanurzone, konieczne jest uprzednia transformacja przy użycia słownika uzyskanego za pomocą modelu Word2Vec.

6.3.3 Moduł training

Główną zawartością modułu są *Notebooki* służące do uczenia modeli, przeznaczone do uruchomienia lokalnie lub w usłudze Google Colab.

6.3.4 Moduł generating

Zawiera funkcje konieczne do przeprowadzenia procesu generacji próbek będących w postaci danych opracowanych w module **data_processing**. W pliku *generating.py* zawarto implementację głównej idei podejścia do procesu generacji próbek opisanej w rozdziale 5.6.1. W pozostałych plikach modułu zdefiniowano funkcje ziaren służących do początkowania procesu generowania. Ziarna podzielono ze względu na wewnętrzną postać danych.

6.4 Walidacja kodu

Duża część modułu *data_processing* oraz *evaluation* została pokryta testami jednostkowymi sprawdzającymi poprawność ich implementacji. Testy okazały się pomocnym narzędziem, które pozwoliło wcześnie dostrzec błędy w logice aplikacji.

Do implementacji testów wykorzystano bibliotekę *unittest*.

6.5 Uruchamianie

Do uruchomienia narzędzi konieczne jest posiadanie interpretera języka Python, menadżera zależności *pip* oraz środowiska *pipenv*. Po wykonaniu poniższej komendy w głównym katalogu projektu, powinna zostać uruchomiona konsola w środowisku zawierającym wszystkie wymagane zależności.

```
pipenv install && pipenv shell
```

Z uruchomionej konsoli można uruchomić serwer *Jupyter*, wykonać skrypt lub uruchomić konsolę interpretera języka Python.

Poniżej przedstawiono przykładowe użycie skryptu *create_sparse_clustered_time.py*:

```
python scripts/create_sparse_clustered_time.py
    --mode=mid2np
    --src=./path/to/midi
    --dst=./output
```

Rozdział 7

Analiza wyników

Jednym z napotkanych problemów podczas implementacji procesu przetwarzania danych i projektu architektury modelu był czasochłonny krok oceny wyników. Po każdorazowym uczeniu modelu konieczne było generowanie próbek, konwersja ich postaci numerycznej na pliki midi, manualny odsłuch i subiektywna ocena.

W celu formalizacji wyników i uzyskania możliwości wygodniejszego sposobu oceny modelu, konieczne było wyznaczenie miar opisujących generowane próbki.

Wyniki w postaci wygenerowanych plików muzycznych zostały dołączone do płyty CD zawierającej kod źródłowy programu.

7.1 Miary i ich definicje

Na podstawie artykułu 'On the evaluation of generative models in music'[16] zdecydowano się zaimplementować następujące miary:

- Rozpiętość tonalna - odległość między najniższym i najwyższym dźwiękiem, w przypadku plików midi ograniczona zakresem <0,127>,
- Histogram tonalny - rozkład występowania poszczególnych dźwięków z dwunastotonowej skali, niezależnie od oktawy,
- Macierz przejść dźwięków - dwuwymiarowa macierz przedstawiająca występowanie konkretnych par dźwięków następujących bezpośrednio po sobie. Za równo kolumny, jak i rzędy odpowiadają dźwiękom skali dwunastotonowej. Wartość znajdująca się na przecięciu odpowiada częstości następowania po

sobie dźwięków określonych kolumną i rzędem. Uwagi wymaga kwestia nanoszenia na macierz informacji o następujących po sobie wielodźwiękach. W celu oddania również takich zdarzeń, na macierz nanoszone są pary będące wynikami iloczynu kartezjańskiego między składowymi wielodźwięków.

- Rozpiętość rytmiczna - odległość między najkrótszą i najdłuższą wartością rytmiczną występującą w mierzonej próbce. Ponieważ wartości kodu 1 z N są posortowane według długości trwania, może ona być wyrażona poprzez różnicę indeksów między klasami kodu 1 z N. Przyjmuje wartości z zakresu <0, ilość klas rytmicznych>.
- Histogram rytmiczny - rozkład występowania poszczególnych klas wartości rytmicznych.
- Macierz przejść wartości rytmicznych - opisuje częstość przejść między konkretnymi wartościami rytmicznymi.

Dodatkowo, opracowano dwie miary mierzące stopień kompresji generowanych ciągów dźwięków i wartości rytmicznych. Uznanie stopnia kompresji jako istotny parametr wynika z intuicji wynikającej z zasady działania dużej części algorytmów kompresji, w których powtarzające się ciągi znaków są zastępowane krótszym znacznikiem wskazującym na pierwsze wystąpienie ciągu. Oznacza to, że sekwencje charakteryzujące się większą powtarzalnością mogą zostać skompresowane w większym stopniu. Główną zaletą tej miary jest jej zdolność wykrywania powtarzających się ciągów rozsuniętych w czasie, co stanowi uzupełnienie bardziej lokalnej miary, jaką jest macierz przejść.

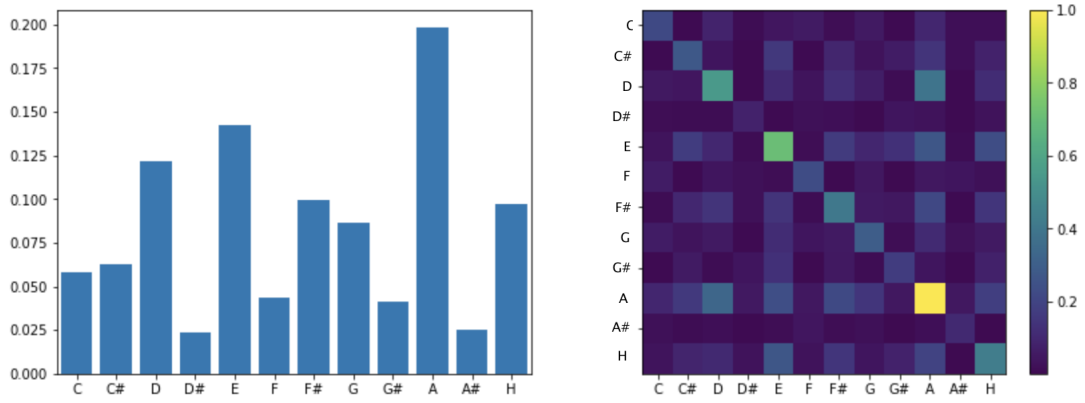
W celu możliwości skorzystania z dostępnych algorytmów, sekwencje dźwięków i wartości rytmicznych zostały przekształcone na ciągi znaków. W przypadku wektorów opisujących dźwięki zdecydowano się na zapis symboli kolejnych dźwięków. Wartości rytmiczne przedstawione były przez indeksy odpowiadających im klas.

Do kompresji wykorzystano bibliotekę zlib, korzystającą z algorytmu delfate.

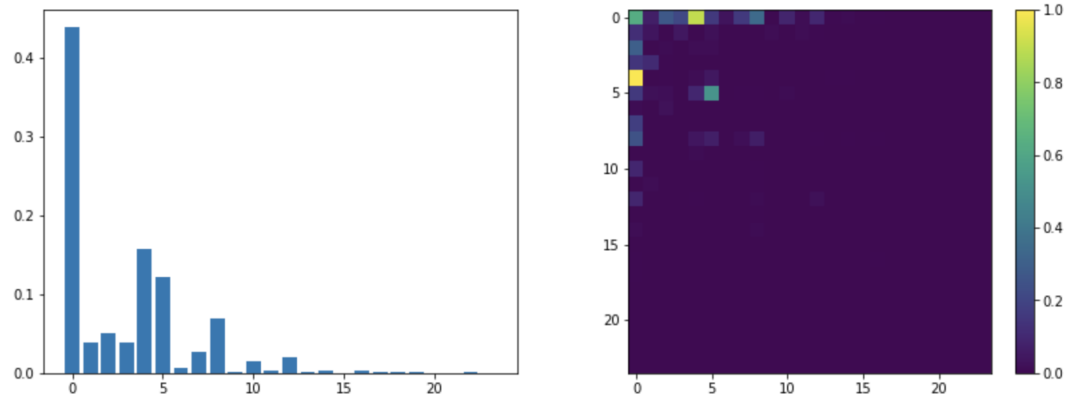
7.2 Pomiar zbioru treningowego

Ponieważ wartości miar będą różne dla każdego zbioru danych, w procesie oceny generowanych próbek powinno się je stosować w odniesieniu do wartości wyznaczonych dla całego zbioru danych treningowych.

Na rysunkach 7.1, 7.2 oraz w tabeli 7.1 przestawione są wartości miar całego zbioru testowego.



Rysunek 7.1: Histogram występowania poszczególnych dźwięków oraz macierz przejść między kolejnymi dźwiękami utworów zbioru treningowego



Rysunek 7.2: Histogram występowania poszczególnych długości dźwięków oraz macierz przejść między kolejnymi wartościami rytmicznymi dźwięków należących do utworów zbioru treningowego

Tablica 7.1: Zakresy i współczynniki kompresji danych

	Zakres	Współczynnik kompresji
Wysokość dźwięków	69	6.932
Długość dźwięków	23	11.364

Na postawie wykresów można sformułować następujące wnioski:

- najczęściej występującym dźwiękiem w zbiorze jest A, a najrzadszym D#,
- wyższe wartości na przekątnej macierzy przejść dźwięków wskazują, że dźwięk występujący w bieżącym wielodźwięku często występuje również w następującym
- symetria macierzy przejść dźwięków wskazuje na brak silnego wpływu kierunku odtwarzania utworu na następstwa tonalne
- klasy przedstawiające długości trwania dźwięków są silnie niezrównoważone

Pomimo że nie wszystkie miary pozwalają na bezpośrednią interpretację w kontekście melodyczności lub rytmiczności, mogą one zostać użyte w celu porównania z innymi zbiorami danych lub wygenerowanymi próbkami. Porównanie wartości miar między zbiorem treningowym a wygenerowanymi próbkami pozwoli na ocenę jakości odzwierciedlenia charakteru danych treningowych.

7.3 Wyniki

Kolejnym krokiem było wygenerowanie próbek i opisanie ich za pomocą wymienionych miar. Zestawienie wartości miar i różnic w odniesieniu do zbioru treningowego przedstawiają tabele 7.2 i 7.3. Wyniki pogrupowano na podstawie użytego ziarna.

Występujące różnice między wynikami nie pozwalają na pewne wskazanie najlepszego ziarna, lecz zachęcają do wysunięcia kilku wniosków.

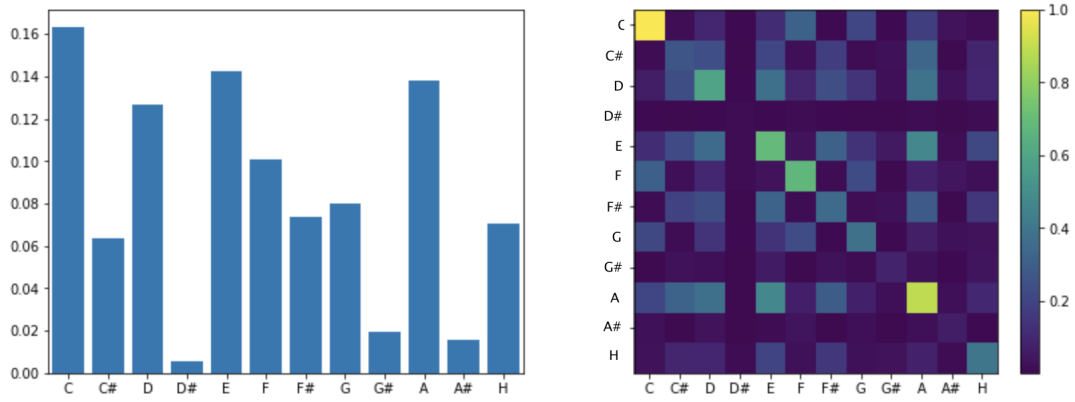
Wszystkie sekwencje poza inicjowanymi pustymi wektorami dają zbliżone różnice miedzy histogramami i tonalnymi macierzami przejść. Sekwencje zapoczątko-

Tablica 7.2: Wyniki miar tonalnych próbek generowanych różnymi ziarnami

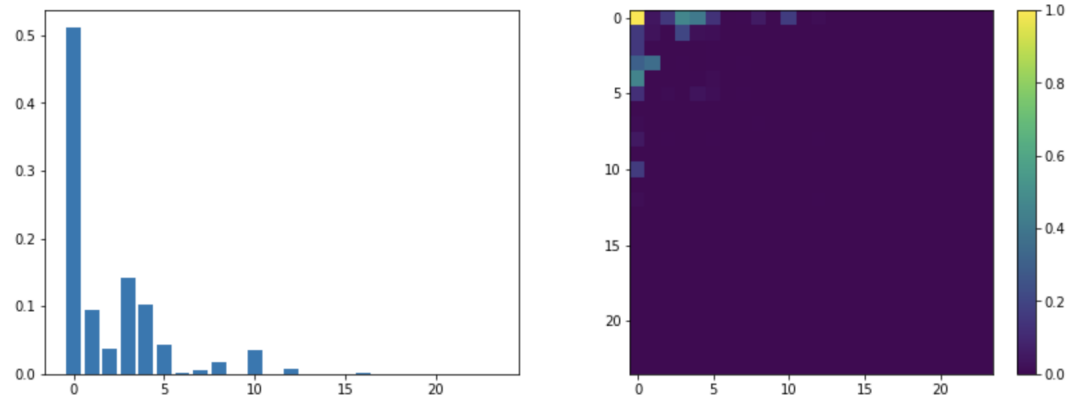
Ziarno	Rozpiętość tonalna	Średnia kwadratowa różnicy histogramu tonalnego	Średnia kwadratowa różnicy tonalnej macierzy przejść	Współczynnik kompresji dźwięków
Wartości losowe rozkładu normalnego	65	0.026	0.112	7.69
Wektory zerowe	61	0.062	0.116	9.52
Pojedynczy dźwięk	67	0.025	0.111	9.83
Sekwencja losowych dźwięków	69	0.023	0.120	8.84
Harmoniczna sekwencja dźwięków	69	0.021	0.106	8.78

Tablica 7.3: Wyniki miar rytmicznych próbek generowanych różnymi ziarnami

Ziarno	Rozpiętość rytmiczna	Średnia kwadratowa różnicy histogramu rytmicznego	Średnia kwadratowa różnicy rytmicznej macierzy przejść	Współczynnik kompresji długości dźwięków
Wartości losowe rozkładu normalnego	19	0.019	0.034	7.99
Wektory zerowe	16	0.035	0.059	10.4
Pojedynczy dźwięk	20	0.030	0.042	9.52
Sekwencja losowych dźwięków	21	0.028	0.042	8.19
Harmoniczna sekwencja dźwięków	21	0.030	0.043	7.89



Rysunek 7.3: Histogram występowania poszczególnych dźwięków oraz macierz przejść między nimi



Rysunek 7.4: Histogram występowania poszczególnych długości dźwięków i przejść między nimi

wane wartościami losowymi zgodnie z intuicją charakteryzują się niższym współczynnikiem kompresji, co może przekładać się na większe zróżnicowanie sekwencji dźwięków. Spośród zbadanych ziaren, sekwencje początkowane harmonicznym zestawem dźwięków najwierniej odwzorowały rozkłady następujących po sobie dźwięków.

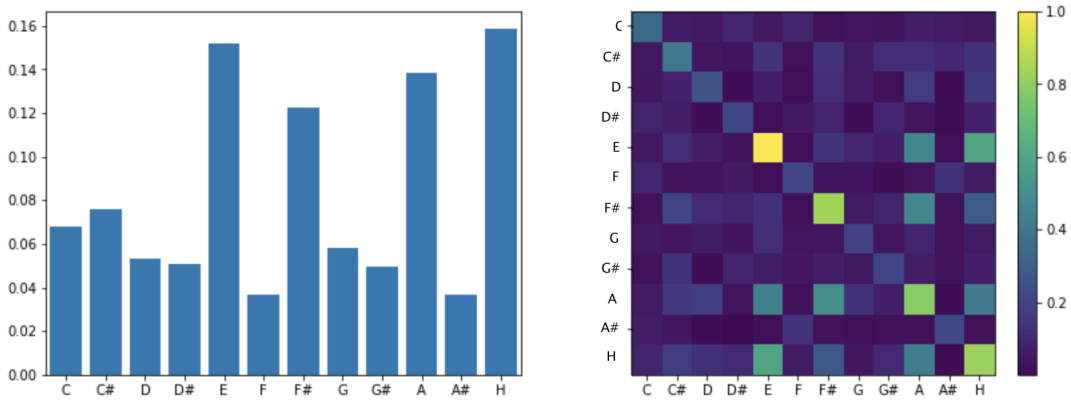
Na rysunku 7.3 oraz 7.4 zamieszczono uśrednione histogramy i macierze przejść próbek wygenerowanych przy pomocy różnych ziaren. Można zaobserwować znaczące podobieństwo względem histogramów i macierzy opisujących zbiór treiningowy, co sugeruje pomyślne odwzorowanie tych parametrów danych.

7.4 Wrażenia subiektywne

Próbki wygenerowane przez model nie dały zadowalających rezultatów. Pomimo akceptowalnego poziomu harmoniczności utworów, rytmiczność pozostawia wiele do życzenia. Zdecydowanie dominującą wartością rytmiczną była najkrótsza możliwa wartość. Najbardziej prawdopodobnym wytłumaczeniem tego zjawiska jest problem niezrównoważenia kategorii wartości rytmicznych.

Po dogłębnej analizie stwierdzono, że zdecydowana większość dźwięków o najkrótszym czasie trwania wynika z przesunięć wiadomości w plikach midi o 1 lub 2 wartości taktujące. Dla kontekstu, najczęściej stosowaną prędkością taktowania jest 120 taktów na ćwierćnutę. Źródłem tego zjawiska mógł być zamierzony zabieg wprowadzający element niedoskonałości, mający na celu wprowadzenie ludzkiego charakteru nagrania wykonywanego przez prawdziwych artystów. Niestety, zabieg ten uniemożliwił generowanie subiektywnie akceptowalnych sekwencji.

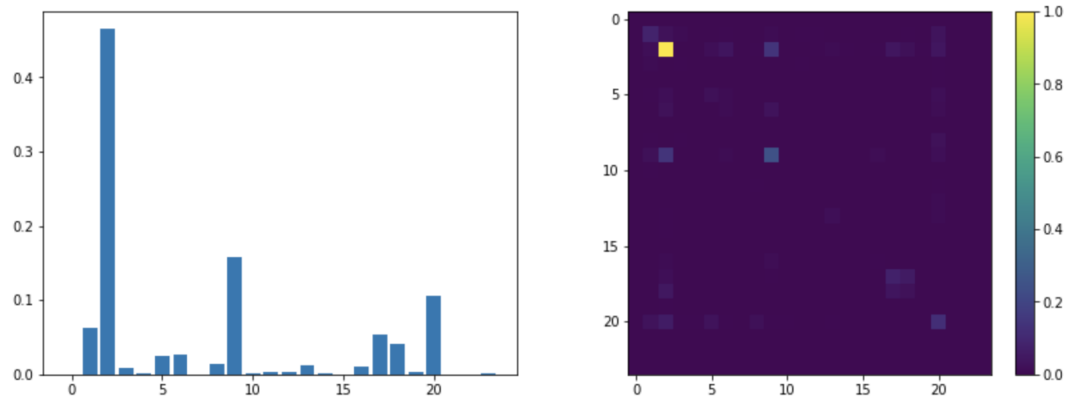
Na podstawie powyższych wniosków, podjęto ponowną próbę uczenia. Tym razem, zbiór danych został ograniczony o powyżej opisane dźwięki. Proces uczenia i generowania został przeprowadzony w ten sam sposób co poprzednio. Na wykresach 7.5 i 7.6 widać oczywiste zmiany w miarach związanych z rytmiką.



Rysunek 7.5: Histogram występowania poszczególnych dźwięków oraz macierz przejść między nimi

Jak można zaobserwować, ograniczenie zbioru danych miało również wpływ na zależności tonalne.

Tym razem wyniki, choć wciąż nie idealne, były conajmniej akceptowalne. W



Rysunek 7.6: Histogram występowania poszczególnych długości dźwięków i przejść między nimi

generowanych utworach obserwowalny był zupełnie inny rozkład długości dźwięków. Sekwencje w subiektywnym odczuciu były dużo mniej chaotyczne, a zwiększone długości dźwięków pozwalały na skupienie się na następujących po sobie dźwiękach.

Spośród sekwencji powstałych przy pomocy różnych ziaren, subiektywnie najbardziej interesujące wyniki zostały otrzymane przy początkowaniu generacji pojedynczym dźwiękiem lub pustymi wektorami. Fakt ten można próbować tłumaczyć pozostawieniem sieci największej swobody przy generowaniu sekwencji, lecz jest to jedynie spekulacja oparta na bardzo ograniczonej liczbie obserwacji.

Uzasadnione są zastrzeżenia co do tak inwazyjnej ingerencji w zbiór danych jaką jest eliminacja wszystkich przedstawicieli konkretnej klasy. Z technicznego punktu widzenia bardziej poprawnym rozwiązaniem mogłoby być dynamiczne skalowanie wartości błędu predykcji dla poszczególnych przypadków w czasie uczenia, w taki sposób, aby mniejsza waga była przywiązywana do przedstawicieli nadreprezentowanej klasy.

Rozdział 8

Wnioski

Przyjęte cele pracy, do których należał dobór, obróbka danych, przeprowadzenie procesu uczenia maszynowego i ocena wyników, zostały zrealizowane. Pomimo nie osiągnięcia w pełni satysfakcjonujących rezultatów, cały proces był niewątpliwie pouczający.

W niniejszej pracy przedstawiono jedynie jedno z możliwych podejść do rozwiązania problemu, jakim jest generowanie muzyki za pomocą metod uczenia maszynowego. Liczność oraz daty powstania dostępnych źródeł literaturowych wskazują, że problem ten jest wciąż aktualny i prowadzone są badania na temat zastosowania metod sztucznej inteligencji w zadaniach, które można określić mianem twórczych. Bogata treść studiowanych prac przedstawia szeroką gamę różnych podejść do powyższego problemu.

Kształcącym elementem pracy był również rozległy proces wstępnej analizy możliwych sposobów rozwoju projektu. Do etapów składających się na finalne efekty pracy należało porównanie odmiennych formatów danych muzycznych, opracowanie numerycznej reprezentacji danych, dobór zbioru treningowego oraz samego projektu architektury modelu. Razem z opracowywaniem metod przetwarzania danych będących intuicyjnie efektywniejszymi, tworzone również były założenia ograniczające wyniki i potencjalne zastosowania. Przykładem takiego kompromisu jest automatyczne wykrywanie najczęstszych wartości rytmicznych, pozwalające na dyskretyzację reprezentacji czasowej bez manualnego definiowania zbioru dozwolonych wartości, odbywające się kosztem całkowitej swobody jaką oferuje for-

mat midi.

Do ograniczeń opisanego procesu można zaliczyć między innymi niemożliwość reprezentowania utworów wieloinstrumentowych, ograniczony słownik wielodźwięków oraz stratność przekształcenia plików midi na macierze i ponownego przetworzenia macierzy na plik midi.

W rozdziale dotyczącym uczenia modelu zaprezentowano podstawowe kroki, jakie należy poczynić podchodząc do dowolnego problemu uczenia maszynowego. Do najważniejszych z nich należy projekt architektury modelu, dobór miar i funkcji kosztów oraz walidacja modelu. Opisywany etap pracy również pozwolił na zapoznanie się z współczesnymi narzędziami ułatwiającymi wszelkie zadania uczenia maszynowego, takimi jak Tensorflow, TensorBoard i Google Colab.

Otrzymane rezultaty nie spełniły wszystkich oczekiwań, lecz wykazywały conajmniej poprawność i częściowy sukces obranego podejścia. Jedną z przyczyn zastrzeżeń mógł być potencjalnie sam zbiór danych, nie zawierający wystarczającej różnorodności pozwalającej na skuteczną generalizację zależności opisujących powszechnie akceptowalne zasady harmonii i rytmiki.

W celu osiągnięcia rezultatów spełniających bardziej rygorystyczne oczekiwania, słusznym krokiem byłoby ponowne wybranie zbioru danych treningowych, tym razem składającego się z większej ilości próbek i nieobarczonego wadą opisaną w rozdziale poświęconym procesowi uczenia. Dodatkowo, słusznym krokiem mogłoby okazać się użycie metod augmentacji danych. W przypadku muzyki, mogłoby to być transponowanie utworów na inne tonacje, co wymusiłoby na modelu interpretacje zależności między interwałami muzycznymi, a nie konkretnymi dźwiękami. Wraz ze zmianą danych oraz ich objętości, konieczna byłaby ponowna parametryzacja modelu i procesu uczenia.

Interesującym kierunkiem, który jest podatny na dalszą eksplorację są inne rodzaje architektur sieci neuronowych. Pomimo że problem inherentnie opiera się na analizie sekwencji, co silnie sugeruje wykorzystanie rekurencyjnych sieci typu Long Short-term Memory, możliwe są konstrukcje modeli odmienne od przytoczonych. Jedną z nich są modele typu sequence-to-sequence encode-decoder, na których wyjściu i wejściu są sieci LSTM, lecz dane pomiędzy nimi są kompresowane do jednowymiarowego tensora, który następnie jest ponownie przekształcany na sekwencję. Alternatywnym podejściem szeroko stosowanym w problemach transferu

stylu i generacji obrazów, są modele generacyjno-adwersyjne (Generative-Adversial Models), składające się sieci uczącej się generowania próbek i sieci uczącej się rozróżniać oryginały od falsyfikatów.

Generowanie utworów muzycznych jest jednym z problemów przekraczających granice, które jeszcze w nieodległej przeszłości wydawały się niemożliwe do przekroczenia. Zadanie to nakierowuje na wiele pytań dotyczących definicji inteligencji, kreatywności oraz zdolności sztucznej inteligencji do tworzenia sztuki. W niedalekiej przyszłości można spodziewać się stosowania metod uczenia maszynowego na szeroką skalę również w dziedzinach, które obecnie uznaje się za wyłącznie osiągalne przez człowieka.

Bibliografia

[1] Nipun Agarwala, Yuki Inoue, Axel Sly. Music composition using recurrent neural networks. *CS 224n: Natural Language Processing with Deep Learning, Spring*, 2017.

[2] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[3] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, i in. A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd*, wolumen 96, strony 226–231, 1996.

[4] Gaëtan Hadjeres, François Pachet, Frank Nielsen. Deepbach: a steerable model for bach chorales generation. *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, strony 1362–1371. JMLR. org, 2017.

[5] Sepp Hochreiter, Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[6] James MacQueen, i in. Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, wolumen 1, strony 281–297. Oakland, CA, USA, 1967.

[7] Warren S. McCulloch, Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.

[8] Tomas Mikolov, Kai Chen, Gregory S. Corrado, Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[9] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[10] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[11] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IEEE*, 72:1192–1201, 1949.

[12] Vitor A. A. Souza, Sandra Eliza Fontes de Avila. Deep neural networks for generating music, 2018.

[13] Bob Sturm, Joao Felipe Santos, Iryna Korshunova. Folk music style modelling by recurrent neural networks with long short term memory units. *16th International Society for Music Information Retrieval Conference*, 2015.

[14] Jian Wu, Changran Hu, Yulong Wang, Xiaolin Hu, Jun Zhu. A hierarchical recurrent neural network for symbolic melody generation. *ArXiv*, abs/1712.05274, 2017.

[15] Li-Chia Yang, Szu-Yu Chou, Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *ArXiv*, abs/1703.10847, 2017.

[16] Li-Chia Yang, Alexander Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, strony 1–12, 2018.

Spis rysunków

Spis tablic

Dodatki

Dodatek A

Zawartość płyty

Na załączonym nośniku CD znajdują się:

- pełen kod źródłowy aplikacji oraz skrypty służące do automatycznego przetwarzania zbiorów danych.
- wszystkie Jupyter Notebooki (pliki .ipynb) wykorzystane podczas wizualizacji procesu przetwarzania danych oraz uczenia modeli uczenia maszynowego. Pliki te mogą być uruchamiane za równo lokalnie, jak i na platformie Google Colab.
- Przykładowy zbiór danych na którym były przeprowadzane eksperymenty.
- Zapisane wyuczone modele razem z próbkami wygenerowanymi za ich pomocą.
- Kod źródłowy pracy inżynierskiej tworzonej za pomocą oprogramowania LaTeX.
- Ostateczna wersja pracy w formacie pdf.
- Ostateczna wersja pracy w formacie pdf pozbawiona rysunków, tabel, listingów bibliografii, spisu treści oraz załączników.