POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka (AIR)
SPECJALNOŚĆ: Robotyka (ARR)

PRACA DYPLOMOWA MAGISTERSKA

Klasyfikacja gatunków muzyki z wykorzystaniem uczenia maszynowego

Classifying Music Genres
Using Machine Learning

AUTOR:
Krzysztof Danielak

PROWADZĄCY PRACĘ:
Dr inż. Witold Paluszyński,
Katedra Cybernetyki i Robotyki

WROCŁAW 2020

Dziękuję Rodzicom

Spis treści

Wstęp

Tematem niniejszej pracy magisterskiej jest klasyfikacja gatunków muzyki z wykorzystaniem uczenia maszynowego. Postawionym zadaniem było zbadanie możliwości klasyfikacji muzyki przy użyciu algorytmów k najbliższych sąsiadów, maszyny wektorów nośnych (SVM) i konwolucyjnych sieci neuronowych.

Pośrednim celem pracy jest zapoznanie się autora z algorytmami uczenia maszynowego - zarówno klasycznymi jak i głębokiego uczenia - oraz ich analiza, co zostało opisane w dalszej części pracy.

Motywacją podejmowania tematów prac w dziedzinie uczenia maszynowego jest ich duża aktualność. Maszynowe uczenie (a w szczególności głębokie uczenie, które jest kategorią uczenia maszynowego) jest dziedziną nauki, która rozwija się coraz szybciej w ostatnich latach. Jednym z zastosowań algorytmów uczenia maszynowego jest pozyskiwanie informacji o dźwiękach czy też utworach muzycznych (ang. *music information retrieval*). Możliwość automatycznej klasyfikacji muzyki na poszczególne gatunki (ang. *automatic music genre classification*; w skrócie AMGC) jest szczególnie istotna dla serwisów streamingowych, których popularność i jakość ciągle rośnie. Także z punktu widzenia użytkownika uporządkowanie piosenek według gatunków jest przydatne, zwłaszcza w kontekście rosnących zbiorów muzyki w zasobach internetowych. W związku z tym autor uznał problem klasyfikacji muzyki za interesujący i postanowił go zgłębić w niniejszej pracy.

Rozdział 1

Klasyfikacja gatunków muzyki z użyciem uczenia maszynowego

1.1 Definicja problemu

W dzisiejszych czasach reprezentacja sygnału muzycznego jest głównie cyfrowa a nie analogowa [Hac00]. Sygnał jest poddawany próbkowaniu (kilkukrotnie na sekundę) i przekształcany przez przetwornik analogowo-cyfrowy wartości numerycznych w dogodnej skali. Sekwencja ta reprezentuje cyfrowy sygnał audio muzyki i może być użyta do odtworzenia dźwięku.
Z powyższego wynika, że cyfrową reprezentację muzyki można przedstawić za pomocą wektora S [Hac00]:

$$S =< s_1, s_2, ..., s_N >$$

gdzie s_i jest i-tą próbką a N jest całkowitą liczbą próbek sygnału. Sekwencja ta zawiera w sobie dużo informacji muzycznej i cechy powiązane z barwą dźwięku, rytmem i wysokością dźwięku. Początkowo cechy akustyczne są wyodrębniane z krótkich ramek sygnału dźwiękowego. Następnie są łączone w bardziej abstrakcyjne cechy na poziomie segmentów. Zatem można wygenerować wektor cech:

$$\overline{X} =< x_1, x_2, ..., x_D >,$$

gdzie cecha x_j jest wyodrębniana z S (lub pewnej jego części) za pomocą odpowiedniej procedury ekstrakcji. Teraz możliwe jest formalne zdefiniowanie problemu automatycznej klasyfikacji gatunków muzyki (AMGC). Jest to problem klasyfikacji wzorców przy użyciu cech na poziomie segmentów jako wejścia. Dla skończonego zbioru gatunków G należy wybrać jedną klasę g, która najlepiej reprezentuje gatunek muzyki przypisany sygnałowi S.

$$\hat{g} = \arg\max_{g \in G} P(g|\overline{X})$$

gdzie prawdopodobieństwo $P(g|\overline{X})$ jest prawdopodobieństwem a posteriori, że muzyka należy do gatunku g dla danych cech wyrażonych przez $\overline{X}$. Na mocy twierdzenia Bayesa można zapisać:

$$\hat{g} = \arg\max_{g \in G} \frac{P(\overline{X}|g).P(g)}{P(\overline{X})}$$

gdzie $P(\overline{X}|g)$ jest prawdopodobieństwem, dla którego wektor cech $\overline{X}$ występuje w klasie g. P(g) jest prawdopodobieństwem a priori gatunku g muzyki a $P(\overline{X})$ jest prawdopodobieństwem

wystąpienia wektora cech $\overline{X}$. Ostatnie z prawdopodobieństw jest w ogólnym przypadku nieznane. Jeśli klasyfikator oblicza prawdopodobieństwa dla całego zbioru gatunków muzyki, wtedy $\sum_{g \in G} P(g|\overline{X}) = 1$ i można obliczyć prawdopodobieństwo dla każdego gatunku muzyki $g \in G$ poprzez:

$$P(g|\overline{X}) = \frac{P(\overline{X}|g).P(g)}{\sum_{g \in G} P(\overline{X}|g).P(g)}$$

[SKK08a]

1.2 Reprezentacja utworu muzycznego

MP3 jest techniką kompresji dźwięku. [Hac00] Jest to inne podejście do problemu kompresji danych niż zip. MP3 dostarcza środki analizy wzorców w strumieniu dźwięku i porównywania ich do modeli ludzkiego słuchu i postrzegania. Ponadto MP3 redukuje dużą ilość danych (w porównaniu z zip) zachowując jedynie dane absolutnie niezbędne do odtworzenia zrozumiałego sygnału. Jest to kompresja stratna. Ilość zachowywanych danych jest dostosowywana przez osobę dokonującą kompresji danych, więc można osiągnąć balans między rozmiarem pliku i jakością. Narzędzie do kompresji nazywa się „koderem" zaś oprogramowanie do odtwarzania zwane jest „dekoderem" albo „odtwarzaczem MP3".[Hac00]

Po przetworzeniu pliku przez koder, rozmiar pliku zmniejsza się około dziesięciokrotnie przy zachowaniu zbliżonej jakości. Przy nieco mniejszej kompresji (do $\frac{1}{8}$ oryginalnego rozmiaru) plik MP3 może być praktycznie nierozróżnialny od materiału źródłowego. Na przykład 3-minutowa piosenka może być skompresowana do pliku o rozmiarze 3MB, co oznacza że płyta DVD o pojemności 5.12 GB może zmieścić do 5 dni nieprzerwanej muzyki.[Hac00]

Plik MP3 składa się z serii bardzo krótkich ramek. Każda ramka danych jest poprzedzona nagłówkiem, który zawiera dodatkowe informacje o danych. W niektórych kodowaniach te ramki mogą wchodzić ze sobą w interakcje. Na przykład jeśli jedna ma pozostawioną pojemność, a następna nie ma, to ramki mogą zostać ułożone optymalnie. Na początku albo na końcu pliku MP3 znajdują się dodatkowe informacje o całym pliku takie jak nazwisko wykonawcy, gatunek muzyczny czy tytuł utworu.

Jakość nieskompresowanego dźwięku zależy w dużej jakości zależy od częstotliwości próbkowania, które jest miarą tego, ile razy na sekundę sygnał jest reprezentowany cyfrowo w ostatecznym strumieniu. Jest ona wyrażana najczęściej w kilohercach. Im jest ona wyższa, tym lepsza jakość dźwięku i większy jest ostateczny plik. Płyty CD miały częstotliwość próbkowania wynoszącą 44.1 kHz, zaś płyty DVD mają 48 kHz lub 96 kHz.[Hac00]

Z kolei WAV jest to standard nieskompresowanych plików audio szeroko używanych w systemach Windows.[Hac00] Zwykle są to pliki dźwiękowe PCM z nagłówkiem WAV. Jeśli taki plik zostanie poddany kompresji, będzie niewiele mniejszy od pliku początkowego.

1.3 Wstępne przetwarzanie danych

Sygnał audio może być przetworzony na spektrogramy w skali melowej. Jest to powszechnie używana metoda ekstrakcji cech dźwięku ze względu na to, że dobrze reprezentuje sposób słyszenia człowieka (tj. częstotliwość w skali logarytmicznej) [HSP].

W celu konwersji surowego dźwięku na spektrogram melowy (oryg. *mel-spectrogram*) należy zastosować krótkoczasową transformatę Fouriera (STFT) na kolejnych oknach czasowych

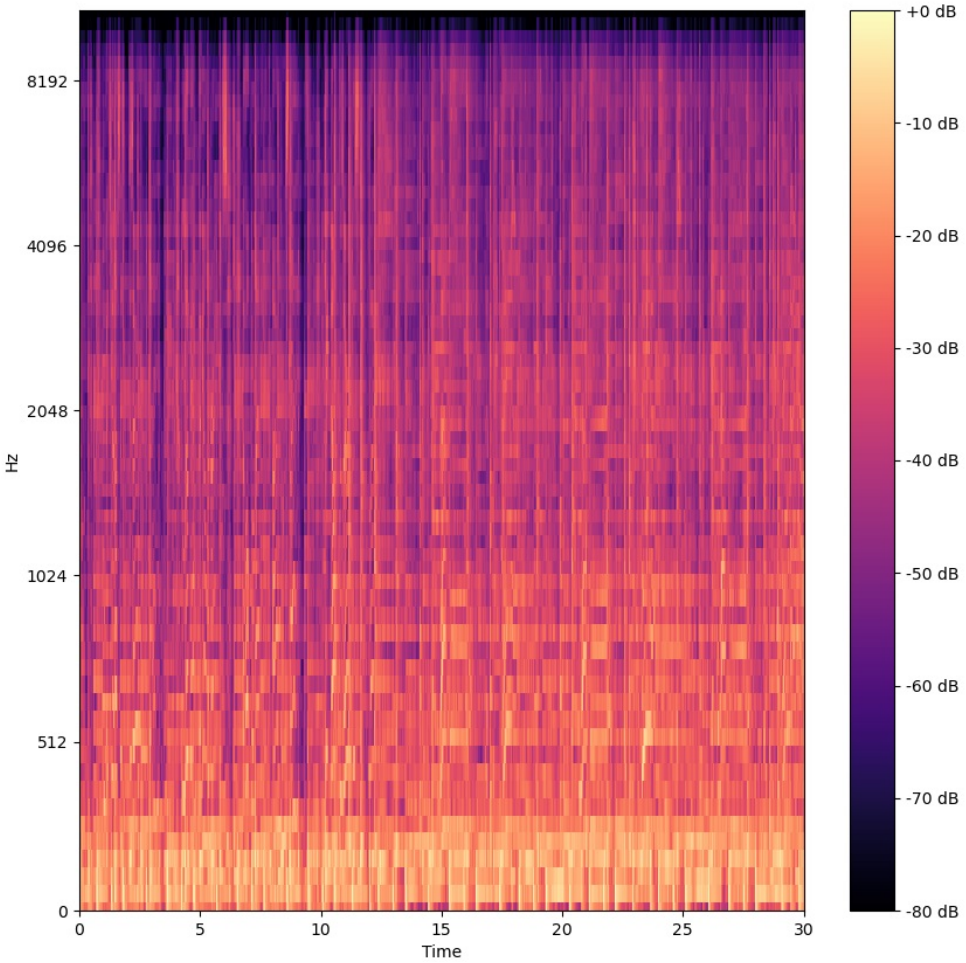
dźwięku, zwykle o szerokości ok. 20 ms. Dla danego sygnału $x[n]$, okna $w[n]$, osi częstotliwości ω i przesunięcia m jest ona obliczana następująco [HSP]:

$$\mathbf{STFT}\{x[n]\}(m,\omega) \sum_{n} x[n]w[n-m]e^{-j\omega n}$$

W praktyce można powyższe obliczać szybciej stosując przesuwne algorytmy DFT. Następnie mapuje się uzyskany wynik do skali melowej poprzez następującą transformację częstotliwości f [O'S87]:

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

Powyższy wzór nie jest jedynym istniejącym, jednakże jest on najpopularniejszy. Następnie stosuje się dyskretną transformację kosinusową, która jest powszechna w przetwarzaniu sygnałów. W ten sposób otrzymuje się współczynniki częstotliwości melowej [RE17]. Można je wykorzystać by otrzymać spektrogram melowy, taki jak na przykładzie poniżej.



Rysunek 1.1: Przykładowy spektrogram melowy dla utworu z gatunku rock z bazy GTZAN

Ze względu na dyskretną transformację kosinusową spektrum melowe jest zazwyczaj reprezentowane na skali logarytmicznej. Skutkuje to sygnałem, który w dziedzinie cepstralnej ma pik quefrencji odpowiadający wysokości sygnału oraz liczbą formantów reprezentujących

niskie piki quefrencji. Ze względu na to, że większość informacji sygnału jest reprezentowana przez pierwsze kilka współczynników mel-cepstralnych (MFCC) ignoruje się lub ucina się składowe wyższego rzędu transformacji kosinusowej. Ostatecznie MFFC są obliczane następująco [RE17]:

$$c(n) = \sum_{m=0}^{M-1} \log_{10}(s(m)) \cos(\frac{\pi n(m-0.5)}{M}); n = 0,1,2,\ldots,C-1$$

gdzie c(n) to cepstralne współczynniki oraz C to liczba współczynników MFCC.

1.4 Algorytm PCA

W celu redukcji wymiaru danych dla modeli KNN (K najbliższych sąsiadów) i SVM (maszyna wektorów nośnych) można wykorzystać algorytm analizy głównych składowych (PCA). Jest to rodzaj analizy czynników, która próbuje aproksymować dane poprzez rzutowanie ich na podprzestrzeń o niższym wymiarze. Aby to zrobić najpierw dokonuje się transformacji spektrogramów melowych poprzez normalizację ich średnich i wariancji. W celu zachowania jak największej wariancji dla m próbek $x^{(i)}$, długości jednostki u algorytm PCA maksymalizuje: [HSP]

$$\frac{1}{m}\sum_{i=1}^{m}(x^{(i)^T}u)^2 = \frac{1}{m}\sum_{i=1}^{m}x^{(i)}x^{(i)^T}$$

1.5 Algorytmy uczenia maszynowego

1.5.1 K najbliższych sąsiadów

Jest to jedna z najstarszych i najprostszych metod klasyfikacji wzorców[HSP]. Najważniejszą zasadą klasyfikacji jest przypisanie nowej próbce klasy k sąsiadów ze zbioru uczącego. Wydajność algorytmu zależy od metryki odległości używanej do zidentyfikowania danych próbek jako najbliższych sąsiadów. [WBS06]

W przypadku braku wcześniejszej wiedzy, klasyfikatory KNN mogą używać prostej odległości euklidesowej, aby zmierzyć różnice między próbkami reprezentowanymi jako wektor wejścia. W idealnym przypadku metryka powinna być dostosowana do rozwiązywanego problemu. [WBS06]. W przypadku klasyfikatora gatunków muzyki proponuje się wykorzystanie odległości euklidesowej.[HSP]

I tak dla k = 10 niech próbki będą oznaczone jako $x^{(1)}$, ..., $x^{(10)}$, które będą k najbliższymi sąsiadami próbki x, tj. wyrażenie $||x-x^{(i)}||_2$ jest dla nich najmniejsze - gdzie $||\cdot||_2$ oznacza odległość euklidesową między próbkami. Następnie możliwe jest zastosowanie wariantu algorytmu z wagami dobierając wagi w_i takie, że [HSP]:

$$w_i \propto ||x-x^{(i)}||_2, \sum_{i=1}^{10} w_i = 1$$

Ostatecznie algorytm zwraca

$$\arg\max_y \sum w_i \mathbb{1}(y = y^{(i)})$$

czyli klasę, która przeważa wśród sąsiadów (po uwzględnieniu wag, jeśli rozpatrywana jest wersja ważona algorytmu).

1.5.2 Maszyna wektorów nośnych (Support Vector Machine)

Jest to klasyfikator optymalny pod względem marginesów[HSP]. Klasyfikator SVM wykorzystuje funkcje jądra, aby znaleźć bardziej złożone relacje w danych wejściowych. Ze względu na to, że muzyka nie jest wejściem, które dałoby się rozdzielić liniowo, działanie klasyfikatora sprowadza się do znalezienia[HSP]

$$\min_{\gamma,w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \xi_i$$

takiego, że

$$y^{(i)}(\sum_{j} \alpha_j K(x^{(j)}, x^{(i)}) + b) \geqslant 1 - \xi_i, \xi_i \geqslant 0$$

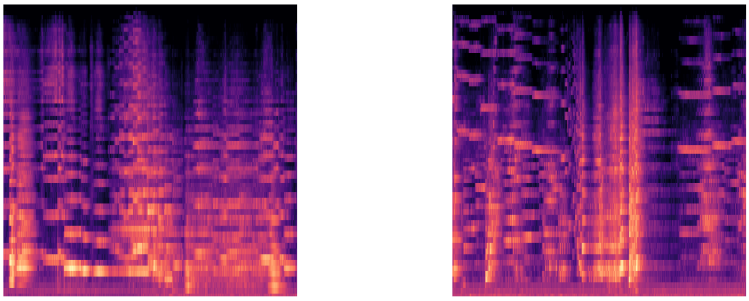
gdzie $x^{(i)}$ to próbki, m to liczba próbek, α_j i b to wagi i polaryzacja, C to parametr kary a $1 - \xi_i$ jest to margines funkcyjny dla próbki i. K: $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ jest funkcją jądra. Tradycyjnie funkcją jądra jest iloczyn skalarny wektorów $x^{(i)}$ oraz $x^{(j)}$, ale stosuje się też np. radialną funkcję bazową jako funkcję jądra:

$$K(x^{(j)}, x^{(i)}) = \exp(-\frac{||x^{(j)} - x^{(i)}||_2^2}{2\sigma^2})$$

Powyższe jądro jest zwane także funkcją jądrową Gaussa. Odpowiada nieskończonej wymiarowo przestrzeni cech i jest powiązane z metryką euklidesową. Funkcja jądra często jest minimalizowana używając minimalnej optymalizacji sekwencyjnej. [HSP]

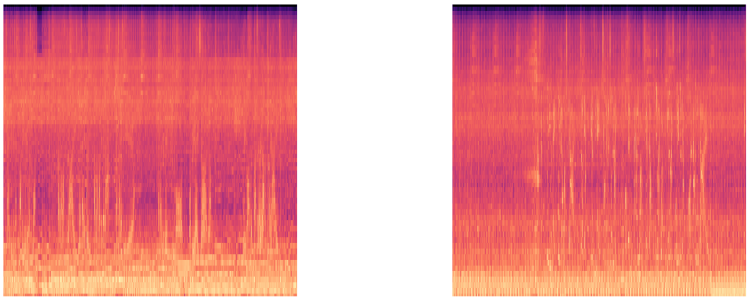
1.6 Głębokie sieci neuronowe

Głębokie uczenie pozwala na realizację zadania klasyfikacji sieci neuronowych bez potrzeby ręcznego doboru cech.[Bah18] Konwolucyjne sieci neuronowe (ang. *Convolutional Neural Network*; w skrócie CNN) są wykorzystywane w zadaniu klasyfikacji obrazów. Reprezentacja obrazu w postaci trójkanałowej (ze względu na model barw RGB) macierzy jest wprowadzana do konwolucyjnej sieci neuronowej. Sieć jest trenowana, aby przewidzieć klasę obrazu. Dźwięk może być reprezentowany w postaci spektrogramu melowego. W związku z tym problem klasyfikacji utworów muzycznych można sprowadzić do problemu rozpoznawania obrazów, w którym sieć wykorzystując spektrogram musi określić gatunek muzyki, do którego należy utwór użyty do wygenerowania spektrogramu.



(a) (b)

Rysunek 1.2: 2 przykładowe spektrogramy dla muzyki klasycznej (GTZAN)



(a) (b)

Rysunek 1.3: 2 przykładowe spektrogramy dla muzyki metalowej (GTZAN)

Jak widać na 1.2 i 1.3 w ramach danego gatunku można zauważyć pewne wzorce, zatem podawanie spektrogramów melowych na wejście sieci konwolucyjnych w kontekście problemu klasyfikacji gatunków muzyki jest zasadne [Bah18]. Każda sieć konwolucyjna składa się z różnych typów warstw neuronów. Każda warstwa sieci neuronowej realizuje pewną operację:

- warstwa konwolucyjna - zawiera zbiór filtrów o określonym rozmiarze. Filtry przesuwają się po zbiorze danych (początkowo po obrazie wprowadzonym na wejściu sieci) i przetwarzają go fragment po fragmencie.[Pal] Wartości wag filtrów są modyfikowane w trakcie uczenia sieci przez algorytm propagacji wstecznej[Bah18].

- warstwa łącząca (ang.*pooling layer*) - służy redukcji wymiarowości mapy cech otrzymanych z warstwy konwolucyjnej.[Bah18]

- warstwa *dropout* - wykorzystuje technikę *dropout*. Jest to metoda regularyzacji, która polega na tymczasowym losowym usuwaniu wybranych neuronów w czasie uczenia sieci. Po zakończonej fazie uczenia przywraca się usunięte neurony. W następnej fazie uczenia na nowo losuje się neurony do tymczasowego usunięcia. Kiedy sieć jest wytrenowana, pracuje wykorzystując wszystkie początkowo istniejące neurony.[Pal]

- warstwa gęsta (w pełni połączona) (eng.*dense layer*) albo *fully connected layer*) - warstwa, w której wszystkie neurony są połączone ze wszystkimi neuronami tej warstwy. Zwykle jest to przedostatnia warstwa sieci neuronowej.[Don18]

Operacja konwolucji jest liniowa, dlatego też aby sieć była skuteczniejsza, funkcja aktywacji w neuronach danej warstwy jest nieliniowa.[Bah18] Jako funkcję aktywacji wykorzystuje się funkcje takie jak ReLU [HSP]:

$$\mathrm{ReLU}(x) = \begin{cases} x & \text{dla } x \geqslant 0 \\ 0 & \text{dla } x < 0 \end{cases}$$

W warstwie wyjściowej wykorzystywana funkcja aktywacji to na przykład funkcja *softmax*:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Przy korzystaniu z wcześniej nauczonej sieci możliwe są dwa podejścia:

- *transfer learning* - sieć wykorzystuje wagi wcześniej nauczonej sieci. Wagi w części warstw konwolucyjnych sieci są niezmienne w czasie uczenia sieci. Pozwala się jednak, by ostatnia warstwa w pełni połączona uczyła się przewidywać prawidłową klasę [Bah18].

- *fine tuning* - zaczyna się uczenie sieci z wagami wcześniej nauczonej sieci. Natomiast w przeciwieństwie do metody *transfer learning* możliwe jest zmienianie wag na wszystkich warstwach[Bah18].

1.7 Dotychczasowe prace

Silla Jr[SKK08b] badał podejście do klasyfikacji muzyki polegające na wykorzystaniu zbioru klasyfikatorów (zgodnie ze schematami dekompozycji przestrzeni i czasu). Wykorzystano algorytmy klasycznego maszynowego uczenia takie jak naiwny klasyfikator bayesowski, drzewo decyzyjne, SVM czy k najbliższych sąsiadów. Zbiór utworów o nazwie Latin Music Database, na którym wykonywano eksperymenty został stworzony przez autora pracy [SKK08b]. Zbiór LMD zawierał 3160 utworów w formacie MP3, które należały do 10 klas (Axé, Bachata, Bolero, Forró, Gaúcha, Merengue, Pagode, Salsa, Sertaneja, i Tango). Utwory dzielono na trzy części trwające po 30 sekund. Ponadto przeprowadzono eksperymenty związane z wyborem cech przy użyciu wzorca algorytmu genetycznego. Okazało się, że podejście zbiorowe jest lepsze w większości przypadków niż klasyfikatory globalne lub pojedyncze segmenty utworu muzycznego. Z przytoczonych badań wynikło, że wybór cech zależy od momentu, z którego pochodzi segment utworu (początek, środek lub końcówka utworu). Na przykład dla algorytmu 3 najbliższych sąsiadów przy dekompozycji One-Against All uzyskano następujące dokładności: 45.83%, 56.26% i 48.43% odpowiednio dla początku, środka i końcowego segmentu utworu). Najlepsze rezultaty osiągano dla segmentów środkowych.

Yandre M.G. Costa [COSJ17] sprawdzał czy można poprawić rezultaty klasyfikacji poprzez wykorzystanie uczenia cech. Porównano wyniki uzyskane przez konwolucyjną sieć neuronową i klasyfikator SVM. Ponadto wykonano eksperymenty łączące wyuczone cechy i cechy wybrane

ręcznie. Badania przeprowadzono na trzech bazach danych (były to ISMIR2004[1] [CWH18], LMD opisany wyżej i zbiór afrykańskiej muzyki ludowej [2]). Lepszym klasyfikatorem w porównaniu z innymi okazała się konwolucyjna sieć neuronowa dla niektórych scenariuszy; dla zbioru LMD najlepsze wyniki dawało połączenie sieci CNN i klasyfikatora SVM zaproponowane przez autorów pracy - $(92.0 \pm 1.6)\%$ dokładności.

Bahuleyan[Bah18] również porównywał głębokie uczenie z ręcznym wyborem cech (zarówno w dziedzinie czasu jak i częstotliwości) na zbiorze Audioset, który zawierał 10-sekundowe fragmenty dźwięku z 2.1 mln filmów z serwisu YouTube. Pliki dźwiękowe należące do gatunków muzyki podzielono na 7 klas (pop, rock, hip-hop, techno, rhythm blues, vocal, reggae). W ten sposób zbiór wykorzystany w pracy liczył 40540 utworów. Wyznaczono także cechy, które są najbardziej przydatne w klasyfikacji gatunków muzyki. Przedstawiono listę 20 najlepszych cech (m.in. tempo muzyki). Badania wykazały, że użycie konwolucyjnej sieci neuronowej może poprawić wyniki klasyfikacji gatunków muzyki. Algorytmy wykorzystujące ręczny dobór cech, takie jak maszyna wektorów nośnych (dokładność 59%) czy algorytm wzmacniania gradientowego (dokładność 59%, co było najlepszym osiągniętym wynikiem w tej grupie algorytmów) klasyfikowały gatunki gorzej niż sieci neuronowe. Sieć neuronowa VGG16 z wykorzystaniem metody *fine tuning* osiągnęła 64% dokładności, co udało się poprawić na 65% gdy połączono w jeden klasyfikator sieć neuronową VGG16 i algorytm wzmacniania gradientowego.

Huang[HSP] sprawdzał efektywność algorytmów klasyfikacji dla dwóch typów wejść - surowej muzyki (dane o amplitudzie) i muzyki przetworzonej na spektrogramy melowe. Badano algorytmy SVM, k najbliższych sąsiadów, sieć neuronową typu feed-forward i konwolucyjną sieć neuronową. Badania przeprowadzono na zbiorze GTZAN[3][Tza]. Okazało się, że spektrogramy melowe oraz zastosowanie algorytmu PCA pozwoliło uzyskać lepsze rezultaty na wszystkich klasyfikatorach, przy czym najlepszym z klasyfikatorów była konwolucyjna sieć neuronowa (dużo lepszy wynik niż osiągane przez ludzi w badaniach Dong[Don18] na tym samym zbiorze muzycznym). W walidacji krzyżowej osiągnęła ona dokładność 84% w przypadku gdy dane były wstępnie przetwarzane. Bez wstępnego przetwarzania wszystkie algorytmy osiągnęły wyniki gorsze od osiąganych przez ludzi w zadaniu klasyfikacji muzyki [Don18].

[1] Zbiór bardziej szczegółowo został opisany w rozdziale 2.2

[2] Zbiór był podzielony na klasy wg różnych kategorii. 10 klas kraju pochodzenia, 11 klas wg grup etnicznych, 20 klas zastosowania muzyki (np. muzyka żałobna czy rozrywkowa) i 9 klas wg instrumentów wykorzystywanych w utworze. Długość utworów w zbiorze nie została podana, ponieważ nagrywano je w ciągu wielu lat bez żadnego ujednolicenia metody nagrywania

[3] Zbiór bardziej szczegółowo został opisany w rozdziale 2.2

Rozdział 2

Środowisko eksperymentalne i zbiory danych

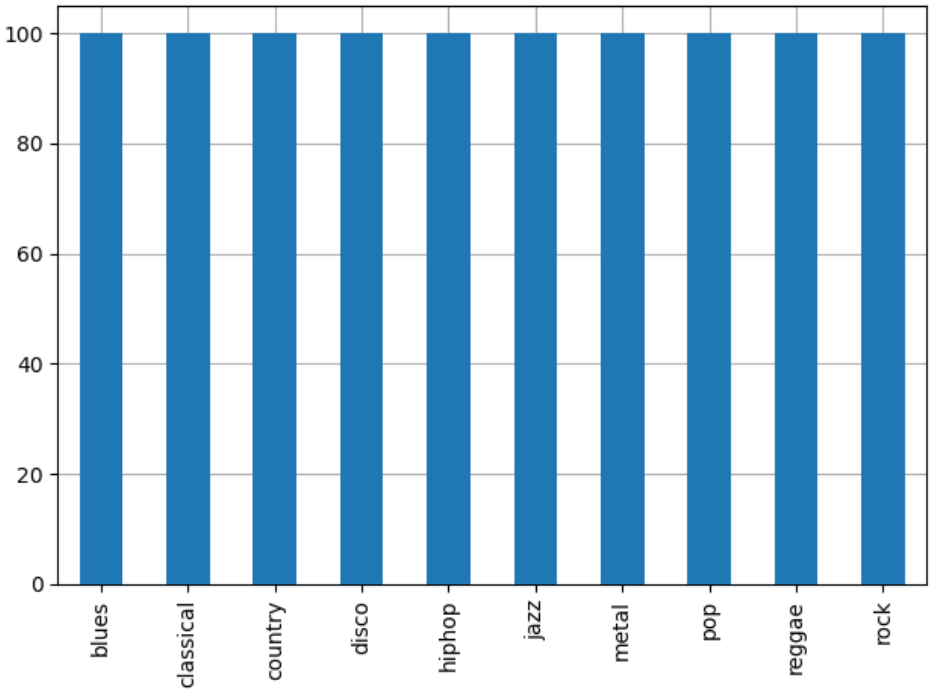
2.1 Środowisko eksperymentalne

Algorytmy zaimplementowane zostały w języku Python (wersja 3.8) przy użyciu bibliotek *scikit-learn*[PVG+11], *keras*[C+15] z *TensorFlow*[AAB+15] w wersji 2.2.0 jako backend. W celu odczytu dźwięku wykorzystano bibliotekę *Librosa*[MLM+20]. Do uczenia sieci neuronowych wykorzystano komputer klasy PC z kartą graficzną GeForce GTX 970 i z zainstalowanym systemem Windows 10.

2.2 Zbiory danych

W pracy wykorzystano dwa zbiory danych: GTZAN[Tza] oraz ISMIR2004[CWH18].

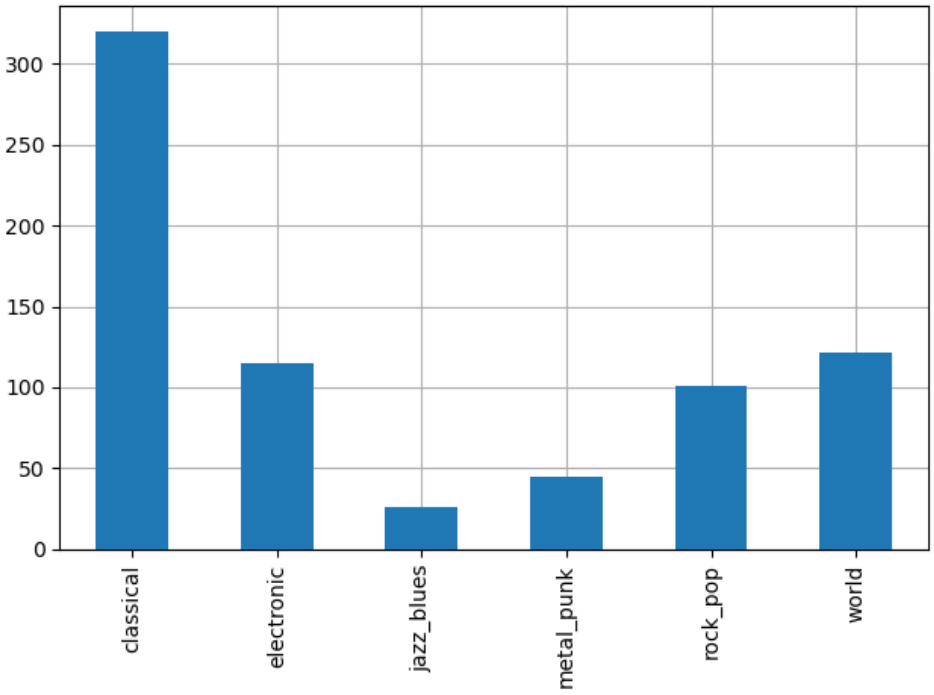
GTZAN to zbiór 1000 utworów muzycznych w formacie WAV trwających po 30 sekund. Są one równomiernie podzielone na 10 klas: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae i rock. W przypadku sieci neuronowych stworzono własny zbiór spektrogramów utworzonych ze zbioru GTZAN. Spektrogramy zostały zapisane jako obrazy w formacie PNG.



Rysunek 2.1: Wykres rozkładu klas w zbiorze GTZAN

ISMIR2004[CWH18] jest to zbiór utworów muzycznych w formacie MP3 należących do 8 klas: classical, electronic, jazz, blues, metal, punk, rock, pop i world. Jednakże w eksperymentach w niniejszej pracy korzystano z pliku opisu zbioru, w którym niektóre klasy zostały połączone przez autorów zbioru. Ostatecznie w ten sposób pliki miały przypisane następujące klasy: classical, electronic, jazz-blues, metal-punk, rock-pop i world. Korzystano wyłącznie ze zbioru treningowego ze zbioru ISMIR2004, który zawierał 729 utworów o następującym rozkładzie klas.

- classical: 320 plików
- electronic: 115 plików
- jazz-blues: 26 plików
- metal-punk 45 plików
- rock-pop: 101 plików
- world: 122 pliki



Rysunek 2.2: Wykres rozkładu klas w zbiorze ISMIR2004

2.3 Jakość klasyfikacji

W ocenie eksperymentów zostały zastosowane następujące miary oceny jakości klasyfikacji:

- dokładność (oznaczana w tabelach jako CV ze względu na stosowanie walidacji krzyżowej) - procent poprawnie sklasyfikowanych próbek na zbiorze testowym[Bah18]. Ze względu na zastosowanie walidacji krzyżowej jest to wynik poprawnej klasyfikacji utworów średnio uzyskiwany na zbiorach walidacyjnych w ciągu całego procesu walidacji krzyżowej.

- f-score - znając macierz pomyłek można obliczyć precyzję (ang.*precision*) i czułość (ang. *recall*). Przyjmuje wartości od 0 do 1, gdzie 1 to najlepszy klasyfikator. Miara F1 zwana inaczej f-score jest średnią harmoniczną precyzji i czułości[Bah18]:

$$2 * \frac{\mathrm{precision} * \mathrm{czulosc}}{\mathrm{precyzja} + \mathrm{czulosc}} [\mathrm{skl}]$$

- ROC AUC - jest to skrót od *Area Under Curve* oraz *Receiver Operator Chacteristics*. Jest to pole pod krzywą ROC, która łączy wierzchołki (0,0) i (1,1) jednostkowego kwadratu [KWGS08] i która przedstawia zależność między prawdziwymi i fałszywymi pozytywami (odpowiednio *true positives* i *false positives*)[Bah18]. Przyjmuje się, że dla podstawowego modelu losowo przypisującego klasy z rozkładem jednostajnym AUC wynosi 0.5. W związku z tym oczekuje się, że tworzone klasyfikatory mają AUC powyżej 0.5, a więc są lepsze od klasyfikatora losowego. W bibliotece scikit-learn wyniki są obliczane dla każdej z klas i następnie wyliczana jest nieważona średnia.

- współczynnik kappa κ - określa zgodność klasyfikacji dwóch klasyfikatorów. W praktyce określa, o ile lepszy jest klasyfikator od klasyfikatora losowego. κ może przyjmować wartości od -1 do 1, gdzie 1 to pełna zgodność. Wartości powyżej 0.5 wskazują na znacząco poprawną klasyfikację[Pal]. W bibliotece scikit-learn jest on definiowany jako[skl]:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

gdzie p_o jest to empiryczna zgodność klasy przypisanej próbce z rzeczywistą klasą próbki. Natomiast p_e jest oczekiwaną zgodnością między dwoma klasyfikatorami, gdyby oba klasyfikatory klasyfikowały próbki losowo.

Rozdział 3

Wyniki eksperymentów praktycznych

3.1 KNN

3.1.1 Eksperyment nr 1

Badane parametry
wagi
metryka
wartość k

Celem tego eksperymentu było odtworzenie eksperymentu z pracy Huang[HSP]. Oryginalny eksperyment badał znalezienie najlepszych hiperparametrów dla 4 klasyfikatorów dla problemu klasyfikacji muzyki. Eksperyment nr 1 miał odtwarzać część eksperymentu z artykułu dotyczącą algorytmu KNN. Eksperyment polegał zatem na znalezieniu najlepszych parametrów dla algorytmu K najbliższych sąsiadów dla badanego problemu. Walidacja krzyżowa była 10-krotna i po pierwotnym przemieszaniu elementów zbioru z utworami w każdym przypadku badano taki sam układ podzbiorów. W eksperymencie wykorzystano bazę GTZAN (pliki WAV) [Tza], tak jak zrobiono to w oryginalnym eksperymencie. Nie stosowano algorytmu PCA. Podane wyniki to średnie dla wszystkich kroków walidacji krzyżowej.

Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu. Najwyższy wynik w tabeli otrzymano dla wersji algorytmu z wagami dla metryki manhattanu dla k = 7 i 21, dlatego uruchomiono dla tych przypadków klasyfikator z większą dokładnością podawania czynników oceny i otrzymano następujące wyniki. Dla k = 21: CV = 0.567, F-score= 0.548 oraz κ = 0.516. Dla K = 7: CV = 0.571, F-score 0.552 oraz κ = 0.521.

Po ponownym uruchomieniu eksperymentu z innym układem podzbiorów okazało się, że otrzymywane wartości miar jakości algorytmu zmieniają się. W związku z tym eksperyment został powtórzony w inny sposób, a wyniki zamieszczono poniżej.

wagi	metryka	k	CV	F-score	κ
bez wag	euklidesowa	3	0.51	0.48	0.45
		7	0.52	0.50	0.47
		10	0.49	0.47	0.44
		21	0.49	0.47	0.43
	manhattan	3	0.54	0.52	0.49
		7	0.56	0.54	0.51
		10	0.53	0.51	0.48
		21	0.54	0.52	0.48
odwrotność odległości	euklidesowa	3	0.53	0.51	0.48
		7	0.55	0.53	0.50
		10	0.53	0.51	0.48
		21	0.51	0.49	0.46
	Manhattan	3	0.56	0.54	0.50
		7	**0.57**	**0.55**	**0.52**
		10	0.55	0.53	0.50
		21	**0.57**	**0.55**	**0.52**

3.1.2 Eksperyment nr 2

Badane parametry
wagi
metryka
wartość k

Eksperyment polegał na znalezieniu najlepszych parametrów dla algorytmu K najbliższych sąsiadów dla badanego problemu. Walidacja krzyżowa była 10-krotna. Klasyfikator uruchamiano 10-krotnie (dokonywano pierwotnego przemieszaniu elementów zbioru) uzyskując czynniki oceny takie jak dokładność walidacji krzyżowej, F-score, AUC i współczynnik κ. Ze względu na to, że po każdym uruchomieniu klasyfikatora pracował on na inaczej ułożonych zbiorach (losowość przemieszania) uzyskiwane wyniki dla danej iteracji różniły się. Stąd dane przedstawione w tabeli są to średnie uzyskane wyniki dla każdego przebiegu. W eksperymencie wykorzystano bazę GTZAN (pliki wav) [Tza].

wagi	metryka	k	CV	F-score	AUC	κ
bez wag	euklidesowa	3	0.50 ± 0.01	0.49 ± 0.01	0.82 ± 0.01	0.45 ± 0.01
		7	0.51 ± 0.01	0.50 ± 0.02	0.85 ± 0.01	0.46 ± 0.01
		10	0.50 ± 0.01	0.49 ± 0.02	0.86 ± 0.01	0.45 ± 0.02
		21	0.48 ± 0.01	0.46 ± 0.01	0.87 ± 0.01	0.42 ± 0.01
	manhattan	3	0.54 ± 0.01	0.53 ± 0.01	0.83 ± 0.01	0.48 ± 0.01
		7	0.56 ± 0.01	0.54 ± 0.02	0.87 ± 0.00	0.51 ± 0.01
		10	0.55 ± 0.01	0.53 ± 0.01	0.88 ± 0.00	0.49 ± 0.01
		21	0.53 ± 0.01	0.51 ± 0.01	0.89 ± 0.01	0.47 ± 0.01
odwrotność odległości	euklidesowa	3	0.53 ± 0.01	0.52 ± 0.02	0.82 ± 0.01	0.48 ± 0.01
		7	0.54 ± 0.02	0.53 ± 0.02	0.87 ± 0.01	0.49 ± 0.01
		10	0.53 ± 0.01	0.52 ± 0.02	0.87 ± 0.01	0.48 ± 0.01
		21	0.51 ± 0.01	0.49 ± 0.01	0.88 ± 0.00	0.46 ± 0.02
	manhattan	3	$\mathbf{0.57 \pm 0.01}$	0.56 ± 0.01	0.83 ± 0.01	$\mathbf{0.52 \pm 0.01}$
		7	$\mathbf{0.57 \pm 0.01}$	$\mathbf{0.56 \pm 0.02}$	0.88 ± 0.01	$\mathbf{0.52 \pm 0.01}$
		10	0.56 ± 0.01	0.55 ± 0.02	0.89 ± 0.01	0.51 ± 0.01
		21	0.55 ± 0.01	0.54 ± 0.01	$\mathbf{0.90 \pm 0.00}$	0.50 ± 0.01

Tablica 3.1: Wyniki eksperymentu nr 2

Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu. Ze względu na występowanie różnych wyników dla różnych uruchomień klasyfikatora (wynikające z różnego przemieszania zbioru) przyjęto metodologię uruchamiania klasyfikatora więcej niż raz również w następnych eksperymentach.

3.2 SVM

3.2.1 Eksperyment nr 3

Badane parametry
funkcja jądra
stopień wielomianu funkcji wielomianowej
funkcja kary C

Analogicznie jak w eksperymencie nr 1 i 2 celem było zbadanie i znalezienie najlepszych parametrów, tym razem dla algorytmu SVM dla badanego problemu. Miało to również odtwarzać eksperyment z pracy Huang[HSP], tym razem dla klasyfikatora SVM. Ponownie walidacja krzyżowa była 10-krotna. Klasyfikator uruchamiano 10-krotnie (dokonywano pierwotnego przemieszaniu elementów zbioru) uzyskując czynniki oceny takie jak dokładność walidacji krzyżowej, F-score, AUC i współczynnik κ. Ze względu na to, że po każdym uruchomieniu klasyfikatora pracował on na inaczej ułożonych zbiorach (losowość przemieszania) uzyskiwane wyniki dla danej iteracji różniły się. Stąd dane przedstawione w tabeli są to średnie uzyskane wyniki dla każdego przebiegu. W eksperymencie wykorzystano bazę GTZAN (pliki wav). [Tza]

funkcja jądra		C	CV	F-score	AUC
radialna funkcja jądra (RBF)		1	0.10±0.05	0.09±0.05	0.59±0.10
		10	0.11±0.06	0.09±0.08	0.61±0.09
		100	0.11±0.05	0.09±0.06	0.61±0.09
		1000	0.10±0.05	0.09±0.07	0.64±0.05
wielomianowa	stopień wielomianu				
	2	1	0.58±0.07	0.56±0.09	0.91±0.03
		10	**0.60±0.05**	**0.59±0.05**	0.91±0.03
		100	0.59±0.07	0.58±0.06	0.91±0.03
		1000	0.58±0.09	0.56±0.08	0.91±0.03
	3	1	0.57±0.09	0.56±0.09	0.91±0.03
		10	0.56±0.11	0.55±0.12	0.89±0.04
		100	0.59±0.09	0.57±0.08	**0.90±0.02**
		1000	0.57±0.10	0.56±0.10	0.90±0.03
sigmoidalna		1	0.05±0.02	0.02±0.01	0.56±0.05
		10	0.06±0.03	0.01±0.02	0.56±0.05

Tablica 3.2: Wyniki eksperymentu nr 3

Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu.

3.3 Algorytm PCA

3.3.1 Eksperyment nr 4

Badane parametry
użycie algorytmu PCA

Celem eksperymentu było porównanie wpływu zastosowania algorytmu PCA na klasyfikację utworów za pomocą klasycznych metod maszynowego uczenia, ponieważ w pracy Huang [HSP] również badano wpływ przetwarzania wstępnego na jakość klasyfikacji. Ponownie walidacja krzyżowa była 10-krotna i dokonywano pierwotnego przemieszania utworów w zbiorze. Następnie każdy klasyfikator uruchamiano 10-krotnie uzyskując czynniki oceny takie jak dokładność walidacji krzyżowej, F-score, AUC i współczynnik κ. Ze względu na to, że po każdym uruchomieniu klasyfikatora pracował on na inaczej ułożonych zbiorach (losowość przemieszania) uzyskiwane wyniki dla danej iteracji różniły się. Stąd dane przedstawione w tabeli są to średnie uzyskane wyniki dla każdego przebiegu. Każdy z klasyfikatorów był uruchamiany z wyznaczonymi wcześniej najlepszymi parametrami.

klasyfikatory	CV	F-score	AUC	κ
KNN	0.57 ± 0.01	0.56 ± 0.01	0.87 ± 0.01	0.52 ± 0.02
KNN + PCA	0.60 ± 0.01	0.59 ± 0.02	0.88 ± 0.01	0.55 ± 0.01
SVM (f.liniowa) + PCA	0.56 ± 0.02	0.55 ± 0.02	0.90 ± 0.02	brak danych
SVM (RBF)	0.11 ± 0.01	0.09 ± 0.01	0.61 ± 0.02	0.05 ± 0.01
SVM (RBF) + PCA	$\mathbf{0.62 \pm 0.01}$	$\mathbf{0.60 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	$\mathbf{0.57 \pm 0.01}$
SVM (f.wielomianowa 2 st)	0.58 ± 0.02	0.57 ± 0.02	0.91 ± 0.00	0.53 ± 0.03
SVM (f.wielomianowa) + PCA	0.61 ± 0.01	0.59 ± 0.01	0.92 ± 0.00	0.56 ± 0.01

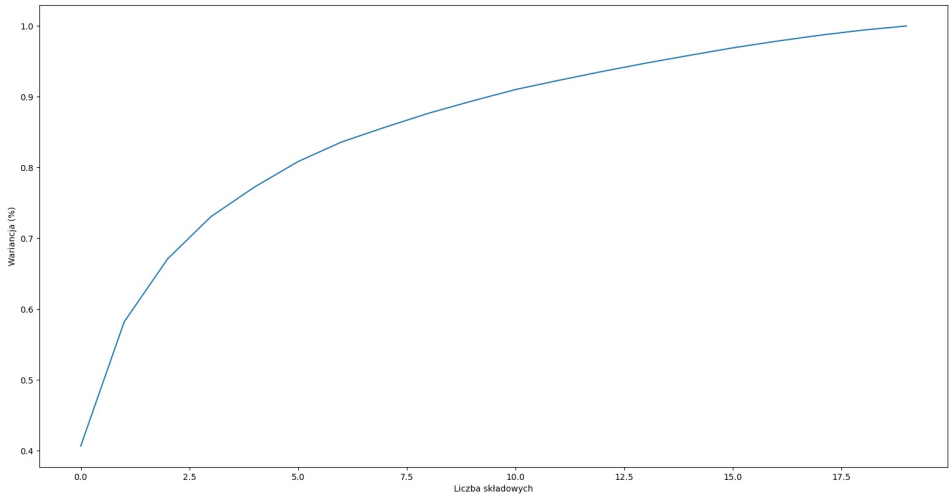
Tablica 3.3: Wyniki eksperymentu nr 4

Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu.

3.3.2 Eksperyment nr 5

Badane parametry
redukcja wymiarowości przez PCA

Celem eksperymentu było porównanie wpływu liczby cech na klasyfikację muzyki. W związku z tym za pomocą algorytmu PCA uzyskiwano różną ilość cech do klasyfikacji Ponownie walidacja krzyżowa była 10-krotna i dokonywano pierwotnego przemieszania utworów w zbiorze. Następnie każdy klasyfikator uruchamiano 10-krotnie uzyskując czynniki oceny takie jak dokładność walidacji krzyżowej, F-score, AUC i współczynnik κ. Ze względu na to, że po każdym uruchomieniu klasyfikatora pracował on na inaczej ułożonych zbiorach (losowość przemieszania) uzyskiwane wyniki dla danej iteracji różniły się. Stąd dane przedstawione w tabeli są to średnie uzyskane wyniki dla każdego przebiegu. Każdy z klasyfikatorów był uruchamiany z wyznaczonymi wcześniej najlepszymi parametrami (klasyfikator SVM z funkcją jądra RBF i C = 10).



Rysunek 3.1: Wykres zależności wariancji od liczby składowych

Na podstawie powyższego wykresu można wywnioskować do jakiego rozmiaru należałoby zredukować wektor współczynników MFCCs (których jest 20), tak by nie stracić zbyt dużej ilości informacji. W związku z tym nie ma sensu rozpatrywać ilości cech mniejszej niż 7.

liczba składowych (cech)	CV	F-score	AUC	κ
7	0.51 ±0.01	0.50 ±0.01	0.87 ±0.00	0.45 ±0.02
10	0.57 ±0.02	0.56 ±0.01	0.89 ±0.01	0.52 ±0.02
15	0.62 ±0.01	0.60 ±0.01	0.92 ±0.00	0.57 ±0.02
18	0.63 ±0.01	0.62 ±0.02	0.93 ±0.00	0.59 ±0.02
20	**0.64 ±0.01**	**0.62 ±0.01**	**0.93 ±0.00**	**0.69 ±0.01**

Tablica 3.4: Wyniki eksperymentu nr 5

Zgodnie z intuicją i wykresem na rys. 3.1 im więcej składowych, tym klasyfikator działa lepiej. Jednakże redukcja wektora cech jest pożądana m.in. z powodów czasowych oraz jest stosowana w literaturze [HSP] -np. Huang doświadczalnie ustalił, że najlepsza jest redukcja do 15 wymiarów (cech). Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu.

3.4 Reprezentacja muzyki

3.4.1 Eksperyment nr 6

Badane parametry
długość utworów w zbiorze

Celem tego eksperymentu było zbadanie wpływu długości próbki utworu na jakość klasyfikacji. Z każdego 30-sekundowego utworu ze zbioru [Tza] wybierano fragment o danej długości T sekund rozpoczynający się w losowej chwili (najpóźniej T sekund przed zakończeniem utworu). Stosowano 10-krotną walidację. Ze względu na losowość uzyskane poniżej wyniki są średnią uzyskanych wyników. Klasyfikator SVM z algorytmem PCA uruchamiano dla najlepszych wyznaczonych wcześniej parametrów (C=10, funkcja jądra RBF, redukcja do 15 cech)

T [s]	CV	F-score	AUC	κ
2	0.52 ±0.02	0.51 ±0.02	0.88 ±0.01	0.47 ±0.02
10	0.59 ±0.01	0.57 ±0.01	0.91 ±0.00	0.54 ±0.01
15	0.61 ±0.01	0.60 ±0.02	0.91 ±0.01	0.56 ±0.01
20	0.61 ±0.02	0.60 ±0.02	0.92 ±0.00	0.56 ±0.02
30	**0.62 ±0.01**	**0.60 ±0.01**	**0.92 ±0.00**	**0.57 ±0.01**

Tablica 3.5: Wyniki eksperymentu nr 6

Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu.

3.4.2 Eksperyment nr 7

Badane parametry
wielkość badanego zbioru

Celem tego eksperymentu było zbadanie wpływu wielkości zbioru muzycznego na jakość klasyfikacji, ponieważ autorzy pracy Huang [HSP] zdecydowali się na sztuczne powiększenie zbioru w sposób zbliżony do zastosowanego w niniejszym eksperymencie. Z każdego 30-sekundowego utworu ze zbioru [Tza] wybierano np. 4 fragmenty o długości 2 sekund rozpoczynające się w losowej chwili (najpóźniej 2 sekundy przed zakończeniem utworu), dzięki czemu w tym przypadku uzyskiwano zbiór 4000 próbek muzycznych, zamiast 1000. Analogicznie uzyskano zbiory o pozostałych wielkościach. Stosowano 10-krotną walidację. Ze względu na losowość uzyskane poniżej wyniki są średnią uzyskanych wyników a rozrzut uzyskanych wyników przedstawiono na wykresie. Każdy z klasyfikatorów był uruchamiany z wyznaczonymi wcześniej najlepszymi parametrami.

Wielkość zbioru	CV	F-score	AUC	κ
1000	0.50 ±0.01	0.48 ±0.01	0.86 ±0.00	0.44 ±0.02
2000	0.62 ±0.01	0.61 ±0.01	0.92 ±0.00	0.58 ±0.01
4000	0.72 ±0.01	0.72 ±0.01	0.95 ±0.00	0.69 ±0.01
8000	0.80 ±0.00	0.80 ±0.00	0.97 ±0.00	0.77 ±0.00
10000	**0.82 ±0.00**	**0.81 ±0.00**	**0.98 ±0.00**	**0.79 ±0.00**

Tablica 3.6: Wyniki eksperymentu nr 7

Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu.

3.4.3 Eksperyment nr 8

Badane parametry
format pliku muzycznego

Celem tego eksperymentu było zbadanie wpływu formatu, w którym zapisano utwory - porównanie między zbiorami GTZAN [Tza] (pliki WAV) i ISMIR 2004 [CWH18] (pliki MP3) dla każdego z klasyfikatorów. Stosowano 10-krotną walidację i po pierwotnym przemieszaniu elementów zbioru z utworami w każdym przypadku badano taki sam układ podzbiorów. Eksperyment przeprowadzono dla algorytmu SVM, z funkcją jądra RBF i C=10

Zbiór	CV	F-score	κ
GTZAN (.wav)	0.50 ±0.02	0.49 ±0.03	0.45 ±0.02
ISMIR2004 (.mp3)	**0.73 ±0.02**	**0.64 ±0.02**	**0.61 ±0.03**

Tablica 3.7: Wyniki eksperymentu nr 8

Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu.

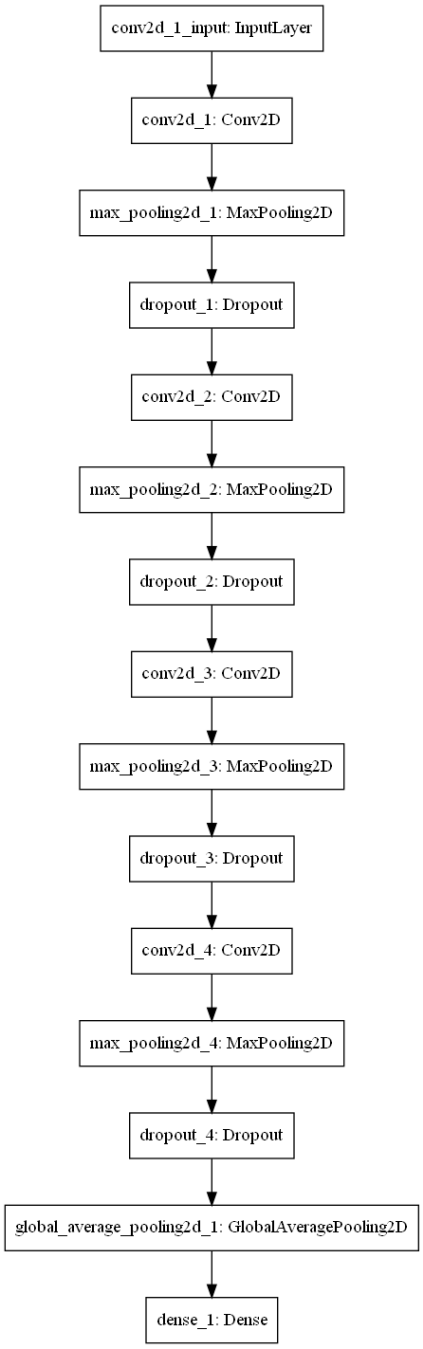
3.5 Sieci neuronowe

3.5.1 Eksperyment nr 9

Badane parametry
mini-batch
epoki

Celem eksperymentu było wyznaczenie parametrów, dla których konwolucyjna sieć neuronowa osiąga najlepsze wyniki. Architektury sieci zostały oparte na artykule Sugianto [SS19], ponieważ architektura sieci z eksperymentu Huang [HSP] została mniej dokładnie opisana w treści artykułu. Ponownie walidacja krzyżowa była 10-krotna i po pierwotnym przemieszaniu elementów zbioru z utworami w każdym przypadku badano taki sam układ podzbiorów. W eksperymencie wykorzystano bazę GTZAN (pliki wav)[Tza]. Każdą sieć uruchamiano 3 razy od początku. Ze względu na to, że po każdym uruchomieniu klasyfikatora sieć pracowała na inaczej ułożonych zbiorach (losowość przemieszania), uzyskiwane wyniki dla danej iteracji różniły się. Sieci korzystały także z warstwy *dropout*. Stąd dane przedstawione w tabeli są to średnie uzyskane wyniki dla każdego przebiegu. We wszystkich sieciach przyjęto współczynnik uczenia 0.001 i optymalizator Adam.

Sieć 4 Conv + 4 MaxPool + GAP



Rysunek 3.2: Architektura sieci 4 Conv + 4 MaxPool + GAP

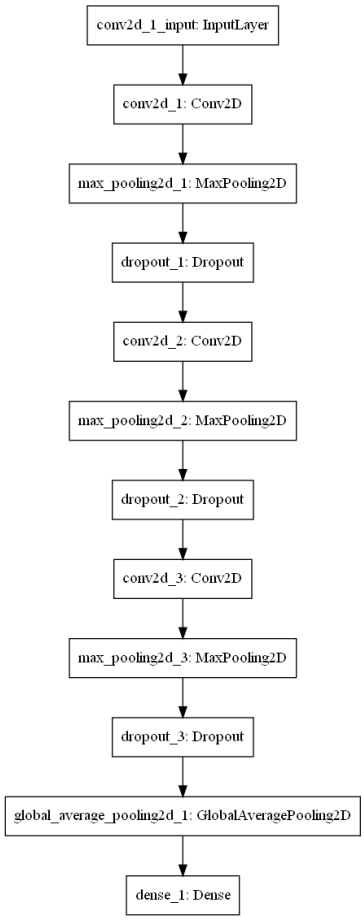
Architektura sieci jest oparta o artykuł Sugianto i Suyanto[SS19]. W artykule nie podano ilości neuronów w warstwach, w związku z czym zdecydowano, że każda kolejna warstwa konwolucyjna będzie miała dwukrotnie więcej neuronów, począwszy od 16 neuronów w warstwie Conv2d_1. Do sieci dodano także warstwy *dropout*. Warstwy konwolucyjne mają funkcję aktywacji ReLU, natomiast warstwa gęsta ma funkcję aktywacji softmax.

Mini-batch	Epoki	CV	Strata	F-score	AUC	κ
10	20	0.65 ± 0.04	1.08 ± 0.08	0.63 ± 0.12	0.93 ± 0.03	0.61 ± 0.13
	50	0.73 ± 0.02	**0.92 ± 0.05**	0.72 ± 0.11	0.95 ± 0.03	0.69 ± 0.11
	100	**0.76 ± 0.01**	0.98 ± 0.08	**0.76 ± 0.06**	**0.96 ± 0.02**	**0.73 ± 0.06**
50	20	0.55 ± 0.01	1.33 ± 0.01	0.52 ± 0.13	0.89 ± 0.05	0.50 ± 0.14
	50	0.67 ± 0.02	1.05 ± 0.02	0.65 ± 0.12	0.93 ± 0.03	0.63 ± 0.12
	100	0.72 ±0.01	0.95 ± 0.03	0.71 ± 0.09	0.95 ± 0.02	0.69 ± 0.10
100	20	0.47 ± 0.07	1.52 ± 0.20	0.42 ± 0.21	0.86 ± 0.09	0.41 ± 0.21
	50	0.60 ± 0.02	1.19 ± 0.05	0.58 ± 0.14	0.91 ± 0.05	0.56 ± 0.15
	100	0.67 ± 0.03	1.04 ± 0.12	0.66 ± 0.11	0.94 ± 0.03	0.63 ± 0.12

Tablica 3.8: Wyniki eksperymentu nr 9 dla sieci 4 Conv + 4 MaxPool + GAP

Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu.

Sieć 3 Conv + 3 MaxPool + GAP



Rysunek 3.3: Architektura sieci 3 Conv + 3 MaxPool + GAP

Architektura sieci jest oparta o artykuł Sugianto i Suyanto[SS19]. W artykule nie podano ilości neuronów w warstwach, w związku z czym zdecydowano, że każda kolejna warstwa kon-

wolucyjna będzie miała dwukrotnie więcej neuronów, począwszy od 16 neuronów w warstwie Conv2d_1. Do sieci dodano warstwy dropout. Warstwy konwolucyjne mają funkcję aktywacji ReLU, natomiast warstwa gęsta ma funkcję aktywacji softmax.

Mini-batch	Epoki	CV	Strata	F-score	AUC	κ
25	20	0.60 ± 0.02	1.20 ± 0.02	0.57 ± 0.09	0.91 ± 0.04	0.55 ± 0.10
	50	0.70 ± 0.03	0.95 ± 0.06	0.69 ± 0.10	0.95 ± 0.03	0.67 ± 0.11
	100	**0.74 ± 0.01**	**0.93 ± 0.05**	**0.73 ± 0.09**	**0.95 ± 0.03**	**0.71 ± 0.09**
50	20	0.55 ± 0.03	1.34 ± 0.09	0.51 ± 0.13	0.89 ± 0.05	0.49 ± 0.13
	50	0.66 ± 0.03	1.06 ± 0.05	0.65 ± 0.10	0.93 ± 0.04	0.62 ± 0.10
	100	0.72 ± 0.02	0.94 ± 0.05	0.71 ± 0.07	0.95 ± 0.03	0.69 ±0.08
100	20	0.52 ± 0.02	1.43 ± 0.07	0.48 ± 0.13	0.88 ± 0.07	0.46 ±0.14
	50	0.61 ± 0.02	1.17 ± 0.05	0.59 ± 0.13	0.92 ± 0.04	0.57 ±0.12
	100	0.68 ± 0.02	1.04 ± 0.05	0.66 ± 0.13	0.94 ± 0.03	0.64 ±0.13

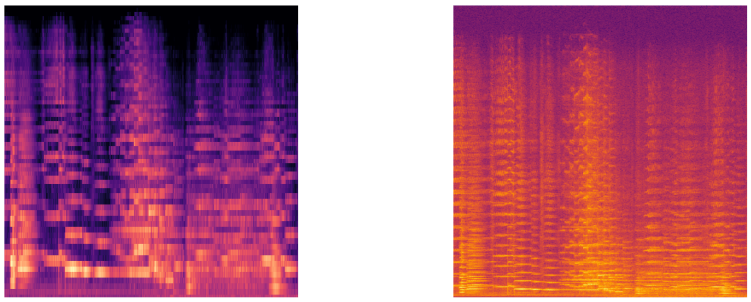
Tablica 3.9: Wyniki eksperymentu nr 9 dla sieci 3 Conv + 3 MaxPool + GAP

Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu.

3.5.2 Eksperyment nr 10

Badane parametry
reprezentacja dźwięku

Celem eksperymentu było porównanie wpływu reprezentacji cech jakimi są współczynniki mel-cepstralne oraz spektrogramy melowe na rozpoznawanie gatunków muzyki dla tej samej architektury sieci. Spektrogramy melowe otrzymywane za pomocą bibliotek Librosa i matplotlib różnią się rozdzielczością a także spektrum barw, co widać na przykładzie poniżej. W związku z tym zostały rozróżnione w eksperymencie.



(a) Librosa (b) matplotlib

Rysunek 3.4: Spektrogramy dla przykładowego utworu muzyki klasycznej (GTZAN)

W tym eksperymencie wykorzystano wcześniej pokazaną architekturę sieci 4 Conv + 4 MaxPool + GAP z wykorzystaniem warstw *dropout*.

Reprezentacja muzyki	Mini-batch	Epoki	CV	Strata	F-score	AUC	κ
współczynniki melowe	50	20	0.65 ± 0.04	1.08 ± 0.08	0.63 ± 0.12	0.93 ± 0.03	0.61 ± 0.13
		100	$\mathbf{0.76 \pm 0.01}$	$\mathbf{0.98 \pm 0.08}$	$\mathbf{0.76 \pm 0.06}$	$\mathbf{0.96 \pm 0.02}$	$\mathbf{0.73 \pm 0.06}$
	100	20	0.60 ± 0.02	1.19 ± 0.05	0.58 ± 0.14	0.91 ± 0.05	0.56 ± 0.15
		100	0.67 ± 0.03	1.04 ± 0.12	0.66 ± 0.11	0.94 ± 0.03	0.63 ± 0.12
spektrogramy melowe (Librosa)	50	20	0.42 ± 0.01	1.68 ± 0.06	0.37 ± 0.03	0.83 ± 0.01	0.35 ± 0.01
		100	0.63 ± 0.00	1.16 ± 0.06	0.61 ± 0.00	0.92 ± 0.01	0.59 ± 0.01
	100	20	0.40 ± 0.01	1.73 ± 0.03	0.35 ± 0.01	0.82 ± 0.01	0.33 ± 0.01
		100	0.59 ± 0.01	1.26 ± 0.02	0.57 ± 0.02	0.90 ± 0.01	0.54 ± 0.02
spektrogramy melowe (matplotlib)	50	20	0.35 ± 0.02	1.86 ± 0.04	0.26 ± 0.02	0.76 ± 0.01	0.27 ± 0.02
		100	0.44 ± 0.02	1.65 ± 0.07	0.39 ± 0.02	0.83 ± 0.01	0.37 ± 0.02

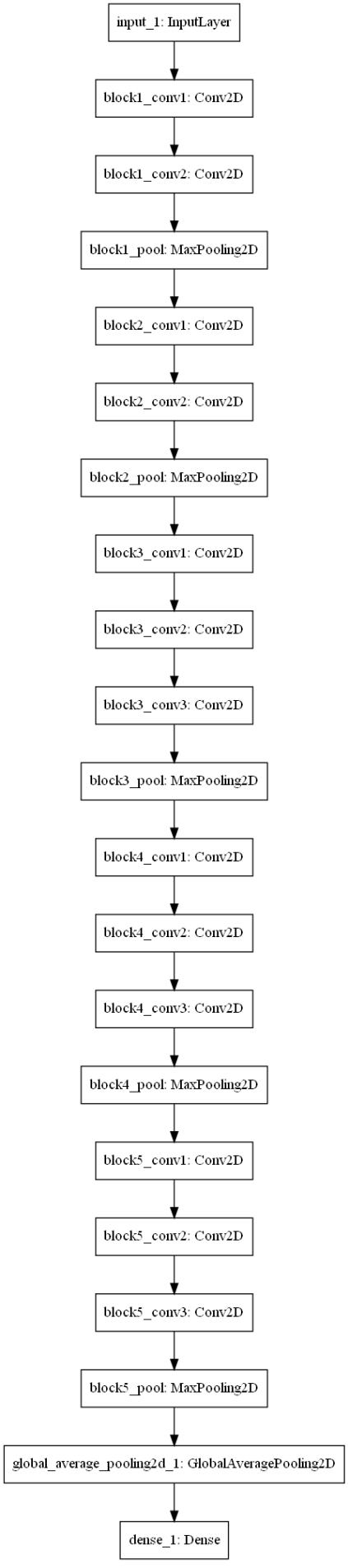
Tablica 3.10: Wyniki eksperymentu nr 10

Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu.

3.5.3 Eksperyment nr 11

Badane parametry
Transfer learning/fine tuning

W tym eksperymencie porównywano zastosowanie podejścia *transfer learning* oraz metody *fine tuning*, ponieważ była to próba odtworzenia eksperymentu Sugianto[SS19] w sensie porównania tych metod z innymi architekturami sieci neuronowych. Wyniki porównano z siecią, która osiągała najlepsze rezultaty w eksperymencie nr 10. Badana sieć ma architekturę sieci VGG16 z wagami wyuczonymi na bazie *ImageNet* (trzy ostatnie warstwy w pełni połączone zastąpiono warstwą GlobalAveragePooling i dodano własną warstwę w pełni połączoną). W przypadku fine tuningu pozwolono, by model modyfikował wagi w ostatnim bloku warstw konwolucyjnych. W eksperymencie stosowano 10-krotną walidację krzyżową. Sieci uczyły się przez 20 epok, a rozmiar mini-batcha wynosił 20. Jako cech używano współczynników mel-cepstralnych.



Rysunek 3.5: Architektura sieci VGG16 z tranfer learningiem

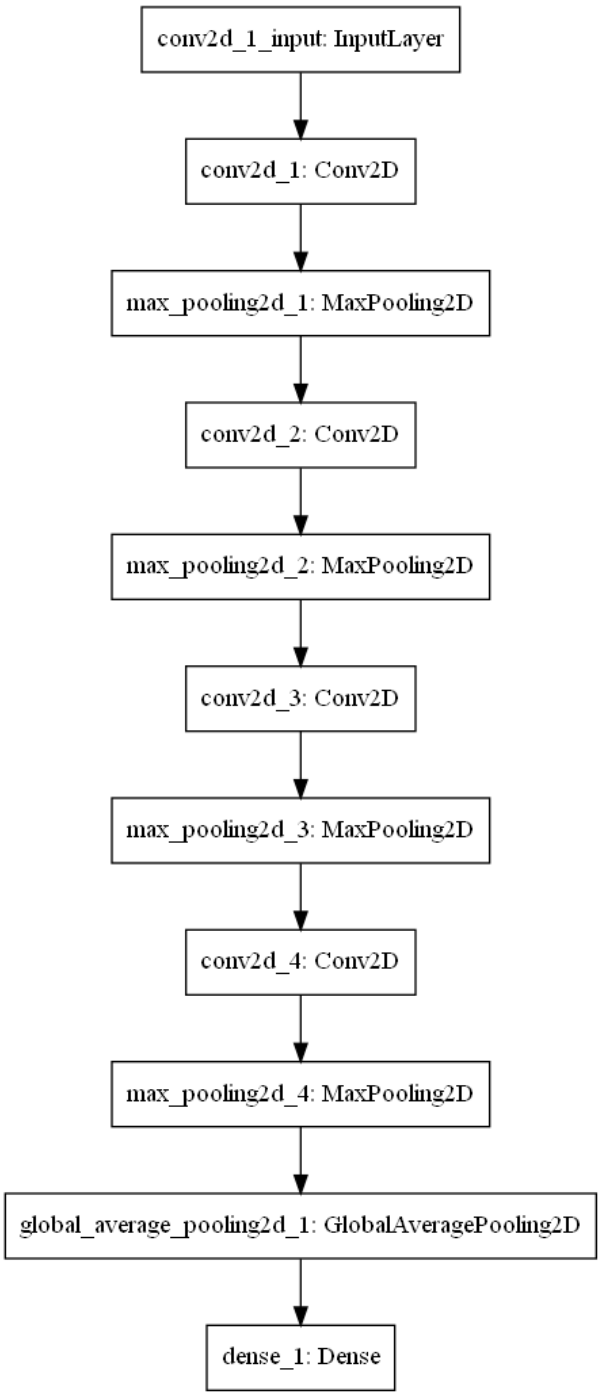
Reprezentacja muzyki	CV	Strata	F-score	AUC	κ
VGG16 (transfer learning)	0.49 ± 0.11	1.60 ± 0.37	0.45 ± 0.10	0.85 ± 0.04	0.43 ± 0.12
VGG16 (fine tuning)	0.15 ± 0.36	2.32 ± 0.14	0.08 ± 0.39	0.55 ± 0.24	0.06 ± 0.39
4 Conv + 4 MaxPool + GAP	$\mathbf{0.65 \pm 0.04}$	$\mathbf{1.08 \pm 0.08}$	$\mathbf{0.63 \pm 0.12}$	$\mathbf{0.93 \pm 0.03}$	$\mathbf{0.61 \pm 0.13}$

Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu.

3.5.4 Eksperyment nr 12

Badane parametry
użycie metody dropout

W eksperymencie nr 12 badano wpływ zastosowania warstw *dropout* dla przypadku sieci z 4 warstwami konwolucyjnymi, 4 warstwami łączącymi MaxPool, warstwą GlobalAverage Pooling (architektura na rys. 3.2) oraz warstwą gęstą. Poniżej architektura sieci w przypadku bez użycia metody *dropout*. W eksperymencie stosowano 10-krotną walidację krzyżową. Sieci uczyły się przez 20 epok, a rozmiar mini-batcha wynosił 20. Jako cech używano współczynników mel-cepstralnych. Wartość *dropout* wynosiła 0.2, co oznacza że szansa na tymczasowe usunięcie neuronu w fazie uczenia wynosiła 20%.



Rysunek 3.6: Architektura sieci 4 Conv + 4 MaxPool + GAP bez warstw Dropout

Reprezentacja muzyki	CV	Strata	F-score	AUC	κ
z warstwami dropout	**0.65 ± 0.04**	**1.08 ± 0.08**	**0.63 ± 0.12**	**0.93 ± 0.03**	**0.61 ± 0.13**
bez warstw dropout	0.57 ± 0.08	1.46 ± 0.54	0.54 ± 0.09	0.89 ± 0.04	0.51 ± 0.09

Okazuje się, że zastosowanie metody *dropout* jest zasadne. Wynik ten jest zgodny z literaturą [Pal]. Pogrubione wyniki są najlepsze dla danego wskaźnika jakości klasyfikacji w ramach eksperymentu.

Rozdział 4

Dyskusja wyników

4.1 Uczenie maszynowe

Zbadano dwa klasyczne algorytmy uczenia maszynowego KNN i SVM. Zauważono, że wartości miar jakości algorytmów zmieniają się zależnie od układu utworów w podzbiorach. Jednak zwłaszcza w przypadku algorytmu k najbliższych sąsiadów odchylenia standardowe są małe, a nawet pomijalne przy założonej dokładności podawanych wyników.

W przypadku badanego zagadnienia dobór parametrów algorytmu jest istotny, ponieważ modyfikując je można zauważyć poprawę jakości obu badanych algorytmów. Na przykład poprawiono dokładność klasyfikacji przez algorytm KNN na zbiorze walidacyjnym od 0.48 do 0.57. W artykule[HSP], z którego próbowano odtworzyć badania, autorzy napisali, że korzystali z metryki euklidesowej, a także wag będących odwrotnością odległości i najlepszy rezultat uzyskali dla k = 10. Dla 10-krotnej walidacji krzyżowej klasyfikator osiągnął dokładność 0.52. W przypadku eksperymentu nr 1 (rozdział 3.1.1) wykonanego w ramach niniejszej pracy dla tych hiperparametrów klasyfikator osiągnął wartość dokładności wynoszącą 0.53, czyli wartość zbliżoną tej z artykułu. Po ponownym uruchomieniu eksperymentu z innym układem podzbiorów okazało się, że otrzymywane wartości miar jakości algorytmu się zmieniają, dlatego też powtórzono eksperyment w wersji nr 2 (rozdział 3.1.2) i dla parametrów z artykułu[HSP] otrzymano dokładność 0.53 ± 0.01, czyli również bardzo zbliżoną.

Zwiększanie wartości k w algorytmie K najbliższych sąsiadów powoduje wzrost jakości klasyfikacji, jednakże zbyt duże wartości k pogarszają wyniki osiągane przez klasyfikator. Jedynym parametrem, który się poprawiał w wyniku uwzględniania coraz większej ilości sąsiadów był AUC. Zatem dla większej ilości wartości progu alfa (szacowanego prawdopodobieństwa, powyżej którego obserwacja klasyfikowana jest do jednej kategorii a poniżej którego – do innych) klasyfikator będzie bliższy idealnego klasyfikowania klas. W artykule[HSP] autorzy nie podali czy badali metrykę Manhattanu. Z eksperymentów 1 i 2 wynikło jednak, że metryka Manhattanu pozwalała osiągać lepsze rezultaty niż metryka euklidesowa. Można także zauważyć, że jeśli zastosuje się wagi, które są odwrotnością odległości to wyniki klasyfikacji, także się poprawiają. Przypuszczalnie dlatego, że w badanym problemie dla zbioru GTZAN sprawdza się założenie, że jeśli próbki są z jednej klasy, to są blisko siebie. Zatem osłabienie wpływu próbek, które są w większej odległości na klasyfikację utworu ma sens potwierdzony doświadczalnie.

Klasyfikator SVM dla wielomianowych funkcji jądra klasyfikował ze zbliżoną dokładnością do opisanej w eksperymencie Huang[HSP] (0.60 w artykule, 0.60 ± 0.05 w eksperymencie nr 3). Trzeba jednak pamiętać, że w przypadku klasyfikatora używanego przez Huang funkcja jądra była radialną a nie wielomianową funkcją jądra. Ponadto w przeciwieństwie do autorów

artykułu, w eksperymencie nr 3 nie używano algorytmu PCA. Widać, że radialna funkcja jądra w eksperymencie nr 3 klasyfikowała w sposób zbliżony do losowego klasyfikatora (z rozkładem jednostajnym) a sigmoidalna funkcja jądra sprawiała, że klasyfikator działał bardzo źle. Autor pracy nie potrafi wyjaśnić przyczyn, dla których tak się stało, ale w pracy Amami[AS17] (badano rozpoznawanie głosek) funkcja sigmoidalna również dawała rezultaty dużo gorsze od funkcji RBF oraz wielomianowej. Ponadto w notebooku na kaggle.com[Vas18] znajdują się wyniki pracy, w której badano klasyfikację na tym samym zbiorze utworów muzycznych i uzyskiwano rezultaty klasyfikacji dla funkcji RBF (nie badano funkcji sigmoidalnej) na tym samym poziomie dokładności (w pracy jest to 0.10 ± 0.05, a w notebooku uzyskano 0.15 ± 0.05). Rezultaty w niniejszej pracy magisterskiej mogą być gorsze od tych z Kaggle ze względu na zastosowanie innego zbioru cech, tj. w notebooku użyto nie tylko współczynników mel-cepstralnych, ale też między innymi gęstości przejść przez zero. Być może dla innych wartości parametru gamma wyniki byłyby lepsze (co nie zostało badane w eksperymencie).

Jednocześnie można zauważyć, że zwiększanie wartości funkcji kary poprawia wyniki do pewnej wartości, która różni się dla każdego ze stopni wielomianu (dla wielomianu 2 stopnia najlepsze było C = 10, natomiast dla wielomianu 3 stopnia najlepszy rezultat dało C = 100). Klasyfikator działał dość dobrze, o czym świadczy dokładność oraz pozostałe miary jakości klasyfikacji. W przypadku eksperymentu Huang [HSP] nie sprecyzowano użytej wartości funkcji kary.

Z eksperymentu nr 4 wynika, że zastosowanie dodatkowo algorytmu PCA do przetwarzania wstępnego w badanym problemie jest dobrym pomysłem, bo w każdym przypadku nastąpiła poprawa klasyfikacji. W przypadku badań Huang[HSP] również w przypadku każdego klasyfikatora zauważono poprawę wyników. Szczególnie dużą poprawę algorytm PCA dał w przypadku algorytmu SVM z funkcją RBF jako funkcją jądra. Dzięki temu najlepsze rezultaty w całym eksperymencie uzyskano stosując funkcję RBF i jednocześnie algorytm PCA. Tym samym wynik dokładności (0.62 ± 0.01) jest zbliżony do wyniku z badań Huang[HSP] (0.60). Odrobinę lepszy wynik może być skutkiem zastosowania innej funkcji kary niż autorzy artykułu, którzy nie podali przyjętej przez nich wartości tego parametru. Ponadto w pracy Amami[AS17], w której badano różne funkcje jądra w kontekście rozpoznawania głosek funkcja RBF również dała najlepsze rezultaty. W porównaniu obu klasyfikatorów SVM z zastosowaniem PCA radzi sobie lepiej od klasyfikatora KNN (z wyjątkiem przypadku zastosowania funkcji liniowej jako funkcji jądra).

Następnie badano algorytm PCA, aby ustalić wpływ redukcji liczby cech na klasyfikację. Wyniki są zgodne z intuicją - im więcej składowych, tym lepsze wyniki klasyfikacji. Niemniej redukowanie cech jest przydatne ze względów czasowych, co zaobserwowano w trakcie eksperymentu. Nie dziwi zatem, że autorzy badań Huang [HSP] również zdecydowali, że najkorzystniej będzie zastosować redukcję cech do 15 składowych.

Wpływ na jakość klasyfikacji mają także cechy plików z utworami muzycznymi. Im dłuższy jest utwór, tym bardziej klasyfikacja się polepszała. Podobnie gdy „sztucznie" powiększono ilość elementów w zbiorze okazało się, że im większy był zbiór, z którego korzystał klasyfikator, tym lepiej klasyfikowana była muzyka. Nie bez znaczenia jest także format pliku muzycznego. Algorytm SVM lepiej klasyfikował utwory ze zbioru plików MP3 niż ze zbioru plików WAV. Mimo iż MP3 jest stratną metodą kompresji, to w tym przypadku wpływ mogła mieć także długość utworów ze zbioru ISMIR2004. Utwory były pełne, a nie były to fragmenty trwające po 30 sekund jak w przypadku GTZAN. Niewykluczone też, że za znaczną poprawę jakości odpowiada fakt, że w zbiorze ISMIR2004 jest tylko 6 klas o niejednostajnym rozkładzie plików dźwiękowych - niektóre klasy połączone, niektóre inne w porównaniu ze zbiorem GTZAN, w

którym było 10 klas.

4.2 Głębokie sieci neuronowe

Skonstruowane sieci konwolucyjne klasyfikują z dokładnością lepszą niż klasyfikator losowy z rozkładem jednostajnym dla wszystkich badanych przypadków (dokładność na zbiorach walidacyjnych wyższa niż 0.1). Wartość AUC jest dużo wyższa od 0.5, a nawet dla niektórych parametrów uczenia zbliża się prawie do 1. To również świadczy o dobrej jakości klasyfikacji. Parametr κ ma często największe odchylenia standardowe. Świadczy to o tym, że może być zależny od tego, jakie elementy będą w poszczególnych podzbiorach walidacji krzyżowej. Wartości tego parametru świadczą o generalnie dobrej klasyfikacji, ponieważ dla obu sieci w jednym lub dwóch przypadkach na 9 badanych średnie wartości są nieco niższe od 0.5 oraz nawet po uwzględnieniu odchylenia standardowego ani razu nie uzyskano wartości ujemnej parametru κ.

Można zauważyć, że wraz ze wzrostem rozmiaru mini-batcha wszystkie badane miary oceny jakości klasyfikacji w obu badanych sieciach pogarszają się (dokładność, F-score, AUC i κ maleją, natomiast wartość funkcji straty rośnie). Wynika to zapewne z tego, że sieć ucząc się na większych porcjach treningowego zbioru danych w każdej fazie uczenia mniej precyzyjnie dobiera wagi neuronów. Widać także, że im dłużej sieć się uczy (im więcej epok), tym lepsze są wskaźniki jakości klasyfikacji oraz w zdecydowanej większości przypadków maleje odchylenie standardowe uzyskanych wyników. Z tego wniosek, że gdyby zwiększać ilość epok, to dla pewnej ich ilości może nie udać się uzyskać znacząco lepszej poprawy.

Porównując obie badane sieci można zauważyć, że uzyskują one zbliżone rezultaty, jednak im większy rozmiar mini-batcha, tym lepiej radzi sobie sieć z trzema warstwami konwolucyjnymi. Wynikać to może z faktu, że jest to mniej skomplikowana sieć, co jest zgodne z wynikami z badań Sugianto. [SS19] W trakcie badań zauważono także, że czas nauczenia sieci jest dużo dłuższy w przypadku uczenia ich za pomocą spektrogramów niż przy użyciu współczynników melowych.

Metoda *dropout* pozwoliła na uzyskanie lepszych wyników, co jest zgodne z literaturą [Pal]. Wynik uzyskany przez sieć o architekturze VGG16 z zastosowaniem *transfer learningu* jest gorszy(dokładność wyniosła 0.49 ± 0.11) od drugiej badanej architektury w eksperymencie 11 (4 Conv + 4MaxPool + GAP), która osiągnęła wynik dokładności 0.65 ± 0.04. Przyczyną jest prawdopodobnie zbyt skomplikowana architektura sieci VGG w zastosowaniu dla problemu klasyfikacji gatunków. Sieć konwolucyjna z artykułu Sugianto[SS19] uzyskiwała zbliżony rezultat (dokładność na poziomie 62.52%) do tej z eksperymentu nr 11. Natomiast sieć VGG16 w pracy Sugianto uzyskała lepszy wynik (56.67%) niż uzyskano w eksperymencie nr 11. Podobnie w przypadku pracy Bahuleyan[Bah18], gdzie sieci VGG16 uzyskały odpowiednio dokładność 0.63 dla sieci z *transfer learningiem* i 0.64 dla sieci z *fine tuning*. W przypadku Sugianto stosowano inny system klasyfikacji utworów oparty na klasyfikacji z głosowaniem na podstawie klasyfikacji podpróbek utworu. W przypadku badań Bahuleyan przyczyną może być dużo większy zbiór utworów (40050 utworów) niż GTZAN. Natomiast zły wynik sieci VGG16 z metodą *fine tuning* z eksperymentu 11 może wynikać z zezwolenia na uczenie zbyt dużej lub zbyt małej ilości warstw dla badanego problemu.

4.3 Wnioski

Z przeprowadzonych eksperymentów wynika, że najlepiej klasyfikować muzykę stosując mel-cepstralne współczynniki melowe. W przypadku stosowania klasycznych algorytmów maszynowego uczenia stosowanie algorytmu PCA pomaga zarówno osiągać lepsze rezultaty (niezależnie od stosowanej miary oceny jakości klasyfikacji spośród dokładności, f-score, AUC i κ) jak i optymalizuje klasyfikację pod kątem czasowym. Dobrze jest wybrać klasyfikator SVM z funkcją RBF jako funkcją jądra. Natomiast jeszcze lepiej jest zastosować sieci neuronowe (poprawa klasyfikacji z 62% na 76%; w obu przypadkach odchylenie od wyniku wyniosło 1%). Potwierdza to, dlaczego badania w kierunku rozwoju sieci neuronowych cieszą się popularnością - sieci mają potencjał lepszej klasyfikacji od tradycyjnych algorytmów uczenia maszynowego. Sieci neuronowe osiągają wyniki, których osiągnięcie dla algorytmu SVM jest możliwe dopiero dla kilkukrotnie większych zbiorów. Należy też pamiętać, że nie bez znaczenia jest format plików, które są klasyfikowane. Uczenie sieci neuronowych zajmuje jednak dużo więcej czasu w porównaniu z klasyfikacją. Projektując konwolucyjną sieć neuronową warto zastosować metodę *dropout*.

Zakończenie

W ocenie autora zadanie postawione przed autorem w ramach niniejszej pracy zostało wykonane w całości. Jak pokazały eksperymenty praktyczne w ramach niniejszej pracy magisterskiej klasyfikowanie muzyki przy użyciu uczenia maszynowego jest jak najbardziej możliwe. Nie jest to zadanie trywialne ze względu na mnogość parametrów do doboru, ilość klas (których może być dużo więcej niż 10), różne formaty plików muzycznych, czas potrzebny części algorytmów na przetworzenie klasyfikacji (zwłaszcza w przypadku uczenia sieci neuronowych; co wymusza albo posiadanie dostępu do komputera z dobrym procesorem i kartą graficzną albo wykorzystanie zasobów Google Colab, na co w przypadku niniejszej pracy się nie zdecydowano). Badano algorytmy K najbliższych sąsiadów, SVM oraz konwolucyjne sieci neuronowe. Przy najlepszych znalezionych parametrach osiągano wyniki klasyfikacji utworów muzycznych na gatunki podobne do literaturowych, zarówno gdy muzyka była reprezentowana w postaci współczynników mel-cepstralnych jak i spektrogramów melowych. Według autora pracy projekt ten można uznać za satysfakcjonujący.

Warto zwrócić uwagę, że przy wykorzystaniu biblioteki *Librosa* w wersji 0.7.2 i nowszej trzeba uwzględnić, że nie wspiera ona na moment tworzenia niniejszej pracy (lipiec 2020) możliwości otwierania plików MP3. Badania wykonywano na komputerze z systemem Windows 10 i trzeba było zainstalować framework ffmpeg. W efekcie można było otwierać pliki MP3 przy użyciu tej biblioteki, lecz odbywało się to dużo wolniej niż w przypadku plików WAV - jest to jeden z powodów, dla których w badaniach skupiono się głównie na zbiorze GTZAN.

Projekt można było poszerzyć o dodatkowe badania, które ze względów czasowych nie zostały zrealizowane. W przypadku dalszych badań warto sprawdzić potencjał innych rodzajów technik głębokiego uczenia, takie jak rekurencyjne sieci neuronowe. Można też zastosować metodę *ensemble learning* łącząc różne klasyfikatory (podobnie jak w badaniach Bahuleyan [Bah18], w których takie podejście poprawiło osiągane rezultaty). Ciekawym pomysłem jest też podejście do klasyfikacji oparte na klasyfikatorach z głosowaniem w pracy Sugianto [SS19], gdzie z utworu wybiera się pewną liczbę podpróbek kilkusekundowych z utworu, które są klasyfikowane i dopiero na podstawie głosowania większościowego wyników podpróbek klasyfikowany jest utwór. Można by było sprawdzić jaki czas podpróbki jest najlepszy, a także czy ma znaczenie z jakiej części utworu (początek, środek, koniec) pochodzi podpróbka (analogicznie do badań Silla Jr [SKK08b]).

Bibliografia

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[AS17] Rimah Amami, Abir Smiti, An incremental method combining density clustering and support vector machines for voice pathology detection. *Computers & Electrical Engineering*, 57:257–265, 2017.

[Bah18] Hareesh Bahuleyan. Music genre classification using machine learning techniques, 2018.

[C+15] Francois Chollet, i in. Keras, 2015.

[COSJ17] Yandre MG Costa, Luiz S Oliveira, Carlos N Silla Jr, An evaluation of convolutional neural networks for music classification using spectrograms. *Applied soft computing*, 52:28–38, 2017.

[CWH18] P. Cano, N. Wack, P. Herrera. Ismir04 genre identification task dataset, lipiec 2018. Licensed under a Creative Commons Attribution- NonCommercial-ShareAlike 1.0 Generic license.

[Don18] Mingwen Dong, Convolutional neural network achieves human-level accuracy in music genre classification. *2018 Conference on Cognitive Computational Neuroscience*, 2018.

[Hac00] S. Hacker, *MP3: The Definitive Guide*. O'Reilly Publishers, 2000.

[HSP] Derek A Huang, Arianna A Serafini, Eli J Pugh, Music genre classification.

[KWGS08] Mirosław Krzyśko, Waldemar Wołyński, Tomasz Górecki, Michał Skorzybut, Systemy uczące się. *Rozpoznawanie wzorców, analiza skupień i redukcja wymiarowości. WNT, Warszawa*, 2008.

[MLM+20] Brian McFee, Vincent Lostanlen, Matt McVicar, Alexandros Metsai, Stefan Balke, Carl Thomé, Colin Raffel, Ayoub Malek, Dana Lee, Frank Zalkow, Kyungyun Lee, Oriol Nieto, Jack Mason, Dan Ellis, Ryuichi Yamamoto, Scott Seyfarth, Eric Battenberg, Victor Morozow, Rachel Bittner, Keunwoo Choi, Josh Moore, Ziyao Wei, Shunsuke Hidaka, nullmightybofo, Pius Friesch, Fabian-Robert Stöter, Darío Hereñú, Taewoon Kim, Matt Vollrath, Adam Weiss. librosa/librosa: 0.7.2, styczeń 2020.

[O'S87] D. O'Shaughnessy, *Speech communication: human and machine*. Addison-Wesley series in electrical engineering. Addison-Wesley Pub. Co., 1987.

[Pal] Witold Paluszyński. Sieci neuronowe. modele głębokie. https://kcir.pwr.edu.pl/~witold/ai/ml_nndeep_s.pdf. Dostęp: 2020-07-21; kurs AREU00122 Uczenie maszynowe na Politechnice Wrocławskiej.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RE17] K. Sreenivasa Rao, Manjunath K E, *Speech Recognition Using Articulatory and Excitation Source Features*. Springer International Publishing, 2017.

[SKK08a] Carlos N. Silla, Alessandro L. Koerich, Celso A. A. Kaestner, A machine learning approach to automatic music genre classification. *Journal of the Brazilian Computer Society*, 14:7–18, 2008.

[SKK08b] Carlos N Silla, Alessandro L Koerich, Celso AA Kaestner, A machine learning approach to automatic music genre classification. *Journal of the Brazilian Computer Society*, 14(3):7–18, 2008.

[skl] scikit-learn 0.23.1 documentation. https://scikit-learn.org/. Dostęp: 2020-07-20.

[SS19] S. Sugianto, S. Suyanto, Voting-based music genre classification using melspectogram and convolutional neural network. W: *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, strony 330–333, 2019.

[Tza] George Tzanetakis. Gtzan genre collection. dostęp: 24.03.2020.

[Vas18] L. C. Vasconcelos. Svm music classification. https://www.kaggle.com/luiscesar/svm-music-classification/data, 2018. Dostęp: 2020-07-02.

[WBS06] Kilian Q Weinberger, John Blitzer, Lawrence K Saul, Distance metric learning for large margin nearest neighbor classification. W: *Advances in neural information processing systems*, strony 1473–1480, 2006.