POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

KIERUNEK INFORMATYKA

Praca dyplomowa magisterska

Klasyfikacja dźwięków perkusyjnych przy użyciu jednokierunkowych sztucznych sieci neuronowych

Autor: Aron Mandrella

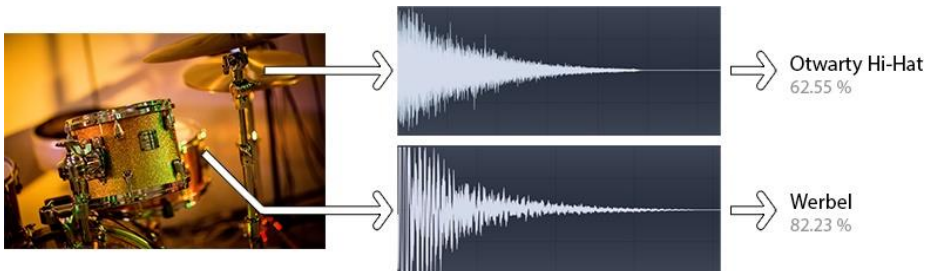
Kierujący pracą: Dr hab. inż. Adam Domański

Gliwice, Wrzesień 2020

Spis treści

1 Wstęp

Od ostatnich kilkunastu lat, obserwować można nieustający wzrost popularności badań, nad szeroko pojmowaną sztuczną inteligencją. Najwięcej czasu i środków, jest jednak prawdopodobnie poświęcanych, na rozwój zaledwie jednej z jej gałęzi. Jak donosi portal natureindex.com, według Google Scholar najczęściej cytowanym czasopismem naukowym wydanym w 2019 roku było Nature [1], przy czym spośród wszystkich opublikowanych w nim artykułów, cztery te do których odwołano się największą liczbę razy w innych pracach, są związane z *uczeniem głębokim* (ang. *deep learning*) będącym jedną z podkategorii *uczenia maszynowego* (ang. *machine learning*). Uczenie głębokie to technika wykorzystywana do tworzenia algorytmów, operujących na skomplikowanych strukturach danych, z których analizą na ogół nie radzą sobie dobrze klasyczne metody stosowane w uczeniu maszynowym. Do takich skomplikowanych struktur danych, zaliczyć można między innymi: obrazy, filmy, dźwięki, oraz tekst w wybranym języku.



Rys. 1 Ilustracja idei rozpoznawania dźwięków perkusyjnych.
Algorytm stara się określić typ dźwięku perkusyjnego po brzmieniu.

Jednym z problemów, który w kontekście uczenia głębokiego nie został jeszcze dobrze przebadany, jest rozpoznawanie dźwięków perkusyjnych (Rys. 1). Powstało na ten temat kilka prac [2] [3], lecz autorzy ograniczali

się w nich do wyodrębnienia zaledwie trzech różnych instrumentów perkusyjnych znacząco różniących się brzmieniem. Taki dobór przykładów znacząco upraszcza problem, więc algorytmy uczenia głębokiego w tych pracach zapewniły dokładność niewiele wyższą niż dużo prostsze algorytmy uczenia maszynowego. Istnieją również starsze prace, w których badano metody rozróżniania bardzo podobnych instrumentów perkusyjnych [4] [5], jednak w nich z kolei nie próbowano stosować uczenia głębokiego.

W niniejszej pracy wykazano, że gdy należy rozróżnić większą liczbę instrumentów perkusyjnych, a część z nich ma podobne brzmienie, to wykorzystywane w uczeniu głębokim sztuczne sieci neuronowe (ang. *artificial neural networks*), a w szczególności splotowe sieci neuronowe (ang. *convolutional neural networks*), są dużo lepszym narzędziem niż klasyczne algorytmy.

1.1 Cel i zakres pracy

Celem niniejszej pracy było pokazanie, że mechanizmy sztucznej inteligencji, nadają się do klasyfikacji relatywnie trudnego zbioru dźwięków perkusyjnych. Otrzymane wyniki wskazują również, że sieci neuronowe a zwłaszcza sieci konwolucyjne, mogą być potencjalnie lepszym wyborem niż klasyczne algorytmy uczenia maszynowego. By zrealizować cel pracy, porównano dokładność 156 sieci neuronowych o różnych architekturach lub stosujących na wejściu różne reprezentacje dźwięku, z dokładnością 340 różnie skonfigurowanych lub stosujących na wejściu różne reprezentacje dźwięku klasyfikatorów k najbliższych sąsiadów (kNN). Klasyfikator kNN został wybrany jako przykład klasycznego algorytmu uczenia maszynowego, gdyż w literaturze wykazano, iż jest dobrym algorytmem do klasyfikacji dźwięków perkusyjnych i zapewnia wyższą lub zbliżoną dokładność co algorytmy las losowy czy maszyna wektorów nośnych [5] [2]. Zarówno dla sieci neuronowych jak i dla klasyfikatorów kNN dokładność klasyfikacji wyznaczano przy pomocy 10-częściowej walidacji częściowej. Najlepsze i

najgorsze dwa wyniki odrzucano, a pozostałe sześć uśredniano. Sumarycznie wytrenowano więc ponad półtora tysiąca sieci neuronowych i niemal trzy i pół tysiąca klasyfikatorów kNN. Nie licząc jednego wyjątku, każdej sieci neuronowej udało się osiągać wyższą średnią dokładność niż najlepszemu spośród wszystkich przebadanych klasyfikatorów kNN dla którego średnia dokładność wyniosła ok. 87%. Sieci neuronowe które osiągnęły najwyższą dokładność (ok. 95%) stosują architekturę konwolucyjną, regularyzację L2, dropout, oraz wynik transformacji ze stałym Q jako wejściową reprezentację dźwięku. Otrzymane wyniki nie wskazują jakoby wysoka dokładność wynikała z nadmiernego dopasowania sieci do zbioru uczącego. Stworzone sieci porównywano również między sobą. W podrozdziałach 5.6.1 - 5.6.6 oraz w podrozdziale 5.8 szczegółowo pokazano jak dobór architektury przekłada się na dokładność sieci neuronowych. Przeprowadzenie powyżej opisanych badań wymagało odpowiedniego zestawu danych do testów, lecz nie wydaje się by taki który spełniał postawione wymagania istniał, lub by był publicznie dostępny. Pośrednim celem pracy było dlatego również skompletowanie autorskiego zbioru dźwięków do trenowania klasyfikatorów i klasyfikacji. Stworzony zbiór składa się z 7640 dźwięków i jest podzielony na 10 niezbalansowanych grup, jednak przy pomocy technik augmentacji zwiększono go tak, by składał się z 14500 dźwięków podzielonych na 10 zbalansowanych grup. Dzięki temu podczas walidacji krzyżowej do testów wykorzystywanych jest zawsze 1450 dźwięków (po 145 dźwięków na grupę), a do uczenia 13 050 dźwięków. Bardziej szczegółowo stworzony zbiór omówiono w rozdziale 5. Trenowanie oraz walidację klasyfikatorów przeprowadzono przy pomocy języka Python oraz pakietów obliczeniowych takich jak TensorFlow [6] oraz SciPy [7]. Do przetwarzania dźwięku użyto pakietu Librosa [8].

Analizie poddawane były wyłącznie wyizolowane perkusyjne sygnały dźwiękowe, a nie takie w kontekście złożonego utworu muzycznego. Praca ta podobnie jak prace [2] [4] [5] skupia się jedynie na aspekcie identyfikacji typu dźwięku perkusyjnego po brzmieniu i jej celem nie jest stworzenie systemu automatycznej transkrypcji perkusji. Jest to odrębny kierunek badań, gdyż wiąże się również z koniecznością filtracji zakłóceń w postaci

innych instrumentów. System taki można by oczywiście stworzyć, odpowiednio modyfikując prezentowane rozwiązania, jednak i bez tego, algorytm rozróżniający instrumenty perkusyjne, ma wiele zastosowań praktycznych. Współcześnie w dużej części muzyki słyszanej w radiu perkusja nie jest nagrywana na żywo, a generowana cyfrowo przy pomocy cyfrowych sekwencerów [9]. Program taki na podstawie sygnału MIDI oraz pojedynczych wyizolowanych perkusyjnych sygnałów dźwiękowych, jest w stanie wygenerować ścieżkę perkusyjną. Osoby zajmujące się tworzeniem muzyki posiadają zazwyczaj dużą liczbę takich sygnałów dźwiękowych co sprawia, że znalezienie tych o odpowiednim brzmieniu dla danej kompozycji może być czasochłonne. W ostatnich latach popularność zyskuje dlatego oprogramowanie do organizacji plików perkusyjnych [10] [11] [12]. Rozwiązania prezentowane w niniejszej pracy mogły by posłużyć do udoskonalenia części z takich programów.

1.2 Zawartość rozdziałów

Zawartość kolejnych rozdziałów jest następująca:

- W rozdziale 2 wprowadzono podstawowe pojęcia z dziedziny uczenia maszynowego i uczenia głębokiego w kontekście klasyfikacji dźwięku, oraz ogólnie opisano stosowane rozwiązania.
- W rozdziale 3 przeprowadzono krótkie wprowadzenie teoretyczne z zakresu dźwięku i sztucznych sieci neuronowych.
- W rozdziale 4 zawarto przegląd literatury tematycznie powiązanej z niniejszą pracą.
- W rozdziale 5 podano wykorzystane narzędzia, omówiono autorski zbiór danych, przedstawiono metodykę badawczą i oraz przeprowadzono analizę otrzymanych wyników.
- W rozdziale 6 podsumowano przebieg i efekty niniejszej pracy.

2 Analiza tematu

W niniejszym rozdziale krótko wytłumaczono czym jest klasyfikacja w kontekście uczenia maszynowego, oraz omówiono dwie metody klasyfikacji dźwięków perkusyjnych.

2.1 Klasyfikacja

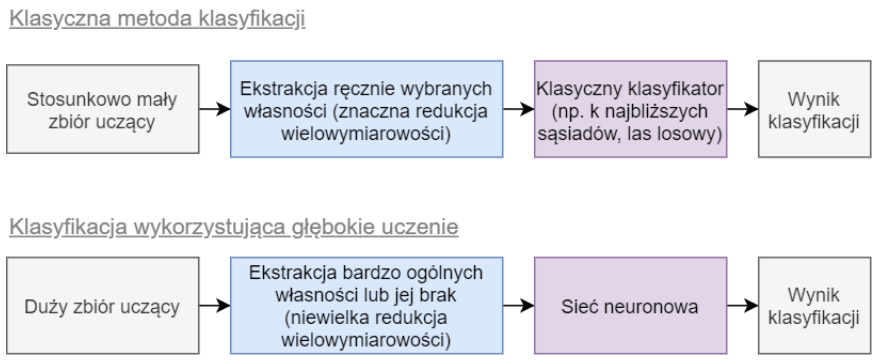
W kontekście uczenia maszynowego, mianem *klasyfikacja* [13] określa się zazwyczaj proces, w którym algorytm zwany *klasyfikatorem,* analizuje pewne dane, a następnie stara się do tych danych przypisać zero, jedną lub kilka z etykiet, których wcześniej został nauczony. Uczenie odbywa się poprzez analizę wielu przykładów danych w tzw. zbiorze uczącym (ang. *training data*) wraz z *etykietami referencyjnymi* przypisanymi do tych danych przez człowieka. Końcowym celem jest by algorytm, analizując dane których jeszcze nie widział, był wstanie wybrać takie etykiety, by możliwie najlepiej te dane opisać, tj. w podobny sposób jak zrobiłby to człowiek. Takie podejście do klasyfikacji, to tzw. *uczenie nadzorowane* (ang. *supervised learning*). Można również klasyfikować stosując *uczenie nienadzorowane* (ang. *unsupervised learning*) poprzez klasteryzację, lecz w niniejszej pracy stosowane będzie wyłącznie podejście pierwsze.

Do praktycznych przykładów klasyfikacji, zaliczyć można identyfikację obiektów na obrazie, określenie cyklu koniunkturalnego w oparciu o historię zmian na giełdzie walut, wykrywanie w portalach społecznościowych spamu, oraz oczywiście identyfikację różnych dźwięków perkusyjnych.

Jako ciekawe kamienie milowe w badaniach dotyczących klasyfikacji przy pomocy metod uczenia głębokiego, można wymienić chociażby: splotową sieć neuronową AlexNet [14] z 2012 roku zaprojektowaną przez A.Krizhevsky, która rozpoznaje obiekty na zdjęciach ze zbioru ImageNet z błędem o ponad 10.8 p.p. (punktu procentowego) mniejszym w stosunku do poprzednich rozwiązań, oraz DeepFace [15] z 2014 roku stworzony przez zespół Facebooka, który rozpoznaje zdjęcia twarzy ze zbioru Labeled Faces in the Wild (LFW) z dokładnością na poziomie 97.25% ± 0.25%, co dorównuje ludzkiej dokładności i deklasuje najlepszy stosowany wówczas przez FBI algorytm, osiągający wyniki o około 12 p.p. gorsze.

2.2 Metody klasyfikacji dźwięku

Chociaż klasyfikacja dźwięku, podobnie jak klasyfikacja obrazów, jest stosunkowo długo znanym i dobrze przebadanym zagadnieniem – w obu przypadkach pierwsze algorytmy były testowane już w XX wieku – to i na tej płaszczyźnie metody korzystające z głębokiego uczenia, sukcesywnie wypierają klasyczne rozwiązania. Nie oznacza to jednak, iż podejścia te się



Rys. 2 Dwa podejścia do klasyfikacji dźwięku.

od siebie diametralnie różną. W obu najpierw na podstawie sygnału dźwiękowego wyznacza się szereg liczb – tzw. *własności* (ang. *features*) – opisujących różne aspekty badanego sygnału dźwięku, a następnie liczby te przepuszcza się przez wybrany klasyfikator, w celu otrzymania wyniku

końcowego. To co najbardziej odróżnia klasyczne podejście od tego bazującego na uczeniu głębokim, to inny balans pomiędzy tymi dwoma etapami. Różnicę tą zobrazowano na Rys. 2. Pojęcie *wielowymiarowość* odnosi się do ilości liczb używanych do opisu pojedynczego sygnału dźwiękowego. Im wielowymiarowość jest bardziej redukowana, tym opis dźwięku jest bardziej ogólny. Przykładowo, średnia głośność dźwięku to dane o mniejszej wielowymiarowości niż wektor głośności dźwięku w różnych chwilach czasu.

2.2.1 Podejście klasyczne

W klasycznym algorytmie klasyfikacji, w fazie projektowej należy ręcznie przeanalizować dane z zestawu uczącego i określić niewielki zbiór własności, którymi opisane zostaną jego elementy. Może to być przykładowo szereg liczb pośrednio charakteryzujących dźwięk pod kątem długości, głośności, brzmienia oraz zmienności tych cech w czasie. Przykładową często stosowaną i stosunkowo uniwersalną postacią własności spotykaną w klasycznych algorytmach klasyfikacji dźwięku są współczynniki mel-cepstralne (ang. *mel-frequency cepstral coefficients*, *MFCC*). Niezależnie jednak jaką metodą własności wydobywane są dźwięku, cel jest taki by w drugim etapie wyuczony klasyfikator był w stanie poprzez ich analizę dobrze rozpoznać typ dźwięku. Przykładową pracą w której do klasyfikacji dźwięków perkusyjnych użyto MFCC jako własności oraz algorytmu k najbliższych sąsiadów jako klasyfikatora jest [5] z 2004 roku.

2.2.2 Podejście stosujące uczenie głębokie

W przypadku klasyfikacji metodami uczenia głębokiego, realizacja pierwszego etapu ogranicza się do ekstrakcji wielu bardzo ogólnych cech, gdyż ich dokładniejsza analiza jest już pośrednio zadaniem sieci neuronowej. Sieć neuronowa dostaje na wejście więcej danych, a przy odpowiedniej architekturze sieci potrafi dość dobrze określić co w nich jest najważniejsze, więc w teorii jest w stanie osiągać lepsze wyniki, niż wcześniej stosowane metody, wymagające dużej ingerencji ludzkiej.

Jedną z prac w których porównano klasyczne metody klasyfikacji dźwięku, z tymi bazującymi na głębokim uczeniu, jest artykuł dr. Karola Piczaka [16] z 2015 roku. Otrzymane wyniki były co prawda porównywalne do tych uzyskiwanych przy pomocy ówczesnych najlepszych rozwiązań, jednak jak zostało później udowodnione w wielu innych pracach - chociażby w [17] - jeśli zastosowana zostanie dobra architektura sieci neuronowej oraz dostępny będzie dostatecznie duży i dobry zbiór uczący, możliwe jest osiągniecie dokładności o kilkanaście, a nawet o kilkadziesiąt p.p. wyższej, od tej uzyskiwanej przez klasyczne metody.

2.3 Dalsza treść pracy

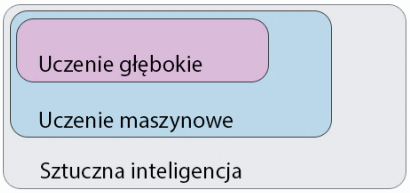
By omówić i porównać najlepsze opisane w literaturze rozwiązania dla problemu klasyfikacji dźwięków perkusyjnych (tzw. *state-of-the-art*) należy najpierw wprowadzić kilka podstawowych pojęć z zakresu ucznia maszynowego, uczenia głębokiego oraz przetwarzania dźwięku. Dalsza przebieg pracy jest więc następujący: wprowadzenie teoretyczne (rozdział 3), przegląd literatury (rozdział 4), omówienie badanych rozwiązań i prezentacja wyników (rozdział 5).

3 Wprowadzenie teoretyczne

W niniejszym rozdziale krótko wprowadzono podstawowe pojęcia z zakresu uczenia maszynowego, uczenia głębokiego, oraz przetwarzania dźwięku. Prezentowane tu informacje są szeroko dostępne w literaturze, dlatego skupiono się wyłącznie na przytoczeniu faktów niezbędnych przy interpretacji wyników badań prezentowanych w rozdziale 5.

3.1 Uczenie maszynowe

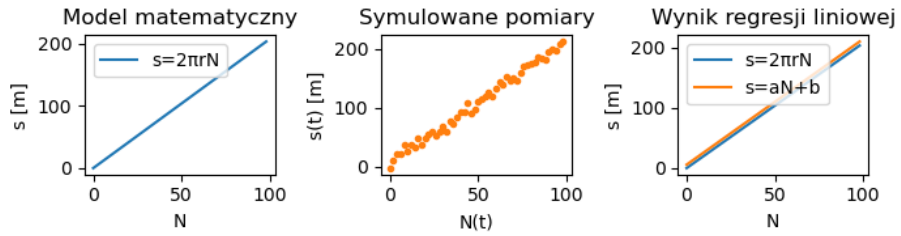
Uczenie maszynowe wchodzące w skład metod sztucznej inteligencji (Rys. 3), to dziedzina algorytmiki obejmująca algorytmy, które poprzez analizę wskazanego zestawu danych uczących są w stanie wypracować pewne wnioski oraz zauważyć korelacje, pomimo braku wiedzy dotyczącej źródła i natury tych danych [18]. Korzyści płynące z takiego podejścia, najłatwiej zobrazować na prostym przykładzie.



Rys. 3 Podkategorie metod sztucznej inteligencji.

Wiedząc, że jedno z kół roweru obróciło się N razy, że promień tego koła to r, oraz pomijając ewentualny wpływ poślizgu, można z dość dużą dokładnością wyznaczyć, jaką trasę s przejechał rowerzysta, od momentu w którym zaczęto liczenie obrotów. Wystarczy podstawić odpowiednie wartości do prostego równania dwóch zmiennych: $s(r, N) \approx 2\pi r N$.

Zbierając pewien zbiór danych uczących, składający się z pomiaru liczby obrotów koła, oraz przejechanej trasy w różnych chwilach czasu t jako punkty (N_t, s_t), by rozwiązać ten problem można również wykorzystać regresję liniową – jeden z najprostszych algorytmów uczenia maszynowego. Przy jej pomocy do zbioru tych punktów wyznaczona zostanie prosta



Rys. 4 Symulacja przykładu z rowerzystą. Przyjęto promień koła N=0.33m, a sztuczne pomiary wygenerowano poprzez dodanie do wyników z modelu matematycznego szumu Gaussa z wariancją 5m i średnią 5m (błąd pomiarowy jest duży w celu lepszej prezentacji wyników)

najlepszego dopasowania, a funkcja opisująca tą prostą będzie równaniem jednej zmiennej $s(N) \approx aN + b$, gdzie a i b to stałe *parametry modelu* wyznaczonego przez regresję liniową. Jak pokazano na Rys. 4, algorytm regresji liniowej pomimo, iż nie zna promienia r ani matematycznej relacji pomiędzy zmiennymi, jest w stanie w oparciu o zestaw pomiarów stworzyć model który wiernie opisuje rzeczywistość. Dla tak prostego przykładu, oczywiście lepiej było by się posłużyć gotowym wzorem matematycznym, jednak nie zawsze taki da się niestety wyznaczyć. Często zdarzają się złożone problemy, w których zależności pomiędzy zmiennymi nie są liniowe, a ich ręczne wyznaczenie jest bardzo ciężkie lub wręcz niemożliwe. Właśnie w takich przypadkach współcześnie stosuje się uczenie maszynowe.

3.2 Uczenie głębokie

Uczenie głębokie dodatkowo rozwija ideę uczenia maszynowego. Pewne problemy są tak skomplikowane, iż nie da się dla nich stworzyć wiernego modelu, stosując algorytmy projektowane z myślą o stosunkowo niewielkiej

liczbie parametrów takie jak: regresja liniowa, drzewa decyzyjne czy las losowy. Regresja liniowa w omówionym wcześniej przykładzie z rowerzystą posiada zaledwie dwa parametry *a* i *b*, natomiast w przypadku uczenia głębokiego, algorytm posiadający setki milionów parametrów nie jest niczym niezwykłym. Oczywiście nic nie stoi na przeszkodzie, by dla złożonych danych możliwych do opisania za pomocą wielowymiarowych punktów, użyć regresji liniowej wyznaczającej miliony parametrów, lub by stworzyć drzewo decyzyjne z milionami gałęzi, jednak jak zostało wykazane w wielu pracach naukowych przytoczonych w rozdziale 4, w przypadku złożonych problemów – a rozpoznawanie dźwięków do takich należy – klasyczne algorytmy uczenia maszynowego najczęściej radzą sobie zdecydowanie gorzej, od tych stosowanych w uczeniu głębokim. Swoją zdolność budowania skomplikowanych modeli, algorytmy uczenia głębokiego zawdzięczają wykorzystywanym przez nie siecią neuronowym [19]. Minusem modeli wykorzystujące sieci neuronowe jest konieczność posiadania większych zbiorów uczących, ze względu na dużą liczbę parametrów jakie należy wyznaczyć.

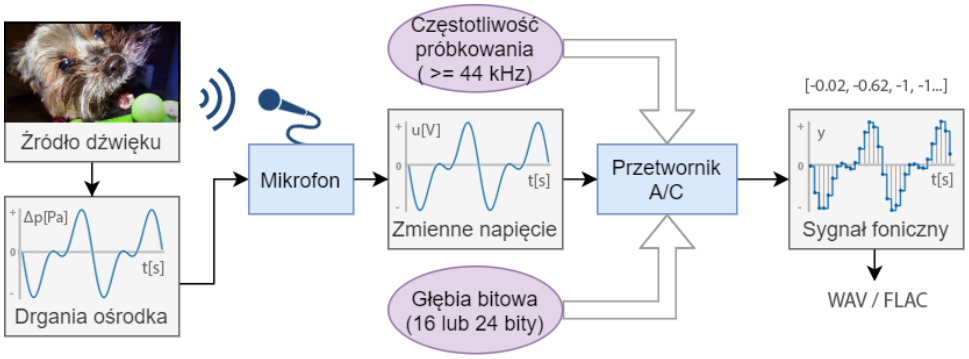
3.3 Dźwięk i jego własności

3.3.1 Sygnał audio

Ucho to narząd dzięki któremu ludzie są w stanie słyszeć. Reaguje ono na fale mechaniczne propagowane w powietrzu lub w innym ośrodku, co w ostateczności powoduje powstanie wrażenia słuchowego w mózgu – innymi słowy dźwięku. Nie każde drgania są jednak słyszalne. W przypadku ludzi, za pasmo słyszalnych fal mechanicznych – nazywanych wówczas *falami akustycznymi* – przyjmuje się zazwyczaj w literaturze orientacyjny zakres od 16 Hz do 20 kHz [20]. W praktyce jednak wiele zależy od wieku danej osoby i stanu uszkodzenia jej narządu słuchu.

By móc analizować dźwięk przy pomocy algorytmów komputerowych, w pierwszej kolejności należy przekonwertować fale akustyczne na *sygnał*

audio (ang. *audio signal*) nazywany również sygnałem fonicznym lub sygnałem dźwiękowym. Jest to reprezentacja dźwięku, przy pomocy zmiennego napięcia, lub przy pomocy serii liczb binarnych w pamięci komputera. Proces rejestracji sygnału audio jest stosunkowo prosty. Źródło dźwięku powoduje zmiany w ciśnieniu atmosferycznym, mikrofon propaguje te zmiany w postaci zmiennego w czasie napięcia, a przetwornik analogowo-cyfrowy w stałych odstępach czasu zapisuje wartości tego napięcia jako liczby binarne. Na tak zapisywane wartości napięcia zwykło mówić się *próbki*, a na proces ich zapisu *próbkowanie*. Uzyskany zbiór próbek stanowi *czasowo-amplitudową reprezentację dźwięku*, o której w niniejszej pracy często mówi się po prostu dźwięk. Przechowywać można ją w postaci pliku w formacie .wav, lub w postaci wspierającego bezstratną kompresję pliku .flac. Cały proces konwersji zobrazowano na Rys. 5.



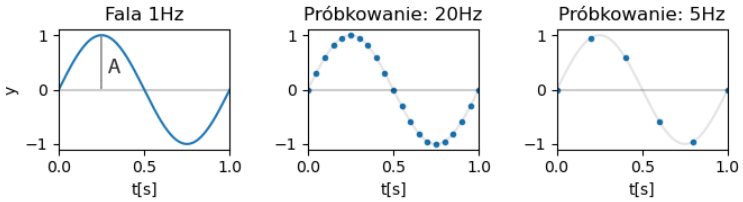
Rys. 5 Konwersja fali akustycznej na sygnał audio.
Przed przetwornikiem A/C często się również filtr dolnoprzepustowy.

Sygnał audio można również syntetycznie wygenerować. W takim wypadku wystarczy stworzyć funkcję wychylenia sygnału $y(t)$, zwracającą wychylenie dla wskazanej chwili czasu t, a następnie generować próbki z wybranym krokiem czasowym.

Z cyfrową reprezentacją dźwięku wiążą się dwa bardzo ważne pojęcia: *częstotliwość próbkowania* (ang. *sample rate*) oraz *głębia bitowa* (ang. *bit depth*). Pierwsze ma wpływ na *dyskretyzację* sygnału źródłowego – tj. zmianę sygnału ciągłego na dyskretny określony jest tylko w pewnych

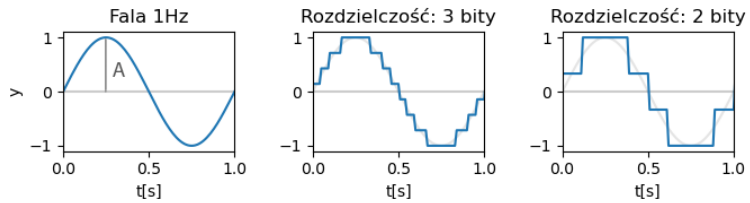
chwilach czasu – a drugie ma wpływ na *kwantyzację tego* sygnału – tj. redukcję zbioru przyjmowanych przez niego wartości wychylenia.

Częstotliwość próbkowania mówi o tym, ile próbek przypada na sekundę źródłowego sygnału (Rys. 6). Zgodnie z twierdzeniem Nyquista–Shannona o próbkowaniu [21], by móc na podstawie próbek wiernie odtworzyć próbkowany sygnał, częstotliwość próbkowania musi być większa bądź równa dwukrotności największej częstotliwości składowej tego sygnału. W przypadku fal akustycznych, minimalną częstotliwością próbkowania będzie więc: 2 · 20 kHz = 40 kHz. W praktyce najczęściej spotykane częstotliwości próbkowania to jednak 44.1 kHz i 48 kHz, co pozwala rejestrować fale mechaniczne o maksymalnych częstotliwościach równych odpowiednio: 22.05 kHz oraz 24 kHz.



Rys. 6 Wpływ częstotliwości próbkowania na sygnał audio. A to amplituda sygnału.

Głębia bitowa, inaczej rozdzielczość bitowa dźwięku, określa ile bitów jest wykorzystywanych dla binarnego zapisu jednej próbki (Rys. 7). Im głębia bitowa przetwornika jest większa, tym mniejsze różnice w zmianach wartości napięcia sygnału źródłowego można zarejestrować. Najczęściej dla dźwięków stosuje się głębię 16 bitową, 24 bitową lub 32 bitową, przy czym tą ostatnią raczej przy obróbce dźwięku, a nie przy jego rejestracji.



Rys. 7 Wpływ głębi bitowej na sygnał audio zapisany w postaci liczb całkowitych. Sygnał przeskalowano do zakresu [-1,1].

3.3.2 Dźwięki sinusoidalne

Różne rodzaje fal akustycznych, powodują różne wrażenia słuchowe. Podzielić je można na trzy kategorie [20]: *dźwięki sinusoidalne*, *dźwięki harmoniczne* oraz *dźwięki nieharmoniczne*.

Dźwięki sinusoidalne zwane również tonami, to najprostsze z dźwięków. Wywoływane są przez okresową falę akustyczną o przebiegu sinusoidalnym więc charakteryzuje je amplituda oraz częstotliwość tej fali (Rys. 8). Wzór na wychylenie ich sygnału można matematycznie zdefiniować następująco:

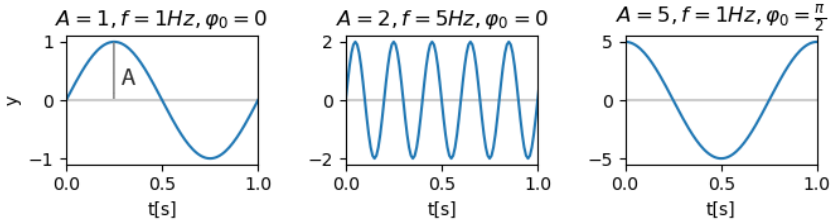
$$y_T(t, f, A, \varphi_0) = A\sin(2\pi f t + \varphi_0) \qquad (1)$$

gdzie

A amplituda sygnału (maksymalne wychylenie),

f częstotliwość sygnału (ilość oscylacji na sekundę),

φ_0 faza sygnału dla $t = 0$.



Rys. 8 Przebiegi tonu dla różnych parametrów fali.

3.3.3 Dźwięki harmoniczne

Dźwięki harmoniczne to takie które można stworzyć z połączenia wielu dźwięków sinusoidalnych nazywanych w tym kontekście *harmonicznymi*. Częstotliwość f_{Hk} kolejnej k-tej harmonicznej musi spełniać warunki:

$$f_{H1} = f_B, \qquad f_{Hk} = kf_B \;\; dla \;\; k = 2, 3, 4 \ldots \qquad (2)$$

gdzie częstotliwość f_B to tzw. *częstotliwość bazowa* dźwięku. Dźwięki zbliżone do dźwięków harmonicznych, produkowane są naturalnie między innymi przez instrumenty smyczkowe, strunowe, dęte i ludzki głos. Fala

akustyczna dźwięków harmonicznych jest falą okresową niesinusoidalną. Ogólny wzór na wychylenie sygnału audio dźwięku idealnie harmonicznego, matematycznie można zdefiniować jako:

$$y_H(t, f_B, \boldsymbol{A}, \boldsymbol{\varphi_0}) = \sum_{k=1}^{\infty} A_k \sin(2\pi f_B k t + \varphi_{k0}) \qquad (3)$$

gdzie

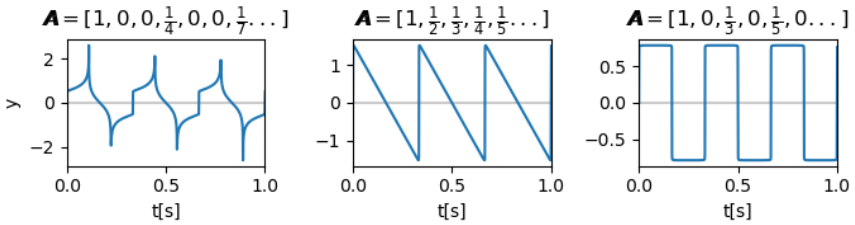
A_k amplituda k-tej harmonicznej,

φk_0 faza k-tej harmonicznej dla $t = 0$,

A wektor amplitud harmonicznych,

$\boldsymbol{\varphi_0}$ wektor faz harmonicznych.

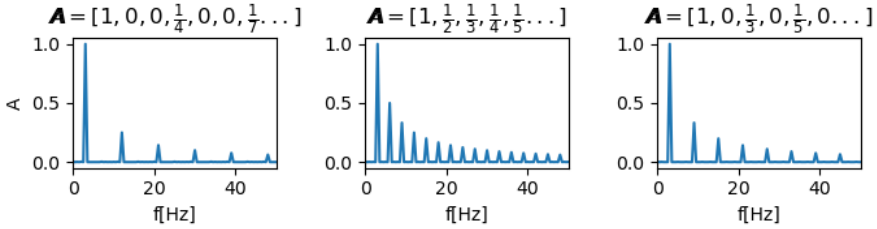
Na Rys. 9 pokazano jaki wpływ ma wektor $\boldsymbol{A}$ na przebieg generowanego sygnału audio, jeżeli przyjęte zostanie, że $\boldsymbol{\varphi_0}$ = [0,0…] oraz f_B = 3 Hz.



Rys. 9 Przykładowe przebiegi sygnałów harmonicznych.

Wektory $\boldsymbol{\varphi_0}$ oraz $\boldsymbol{A}$ opisujące parametry interferujących (nakładających się na siebie) fal składowych, oprócz wpływu na kształt końcowego przebiegu, mają przede wszystkim wpływ na postrzegane przez człowieka brzmienie dźwięku, zwane także *barwą*. Zestaw wszystkich częstotliwości wchodzących w skład złożonych dźwięków, określa się mianem [22] *widma akustycznego*. Stwierdzenie, że dwa dźwięki mają podobną barwę w praktyce oznacza, że mają podobne widmo akustyczne. Rozgraniczyć można dodatkowo *widmo amplitudowe* – tz. takie na którym pokazano amplitudy składowych, oraz *widmo fazowe* – tz. takie na którym pokazano fazy

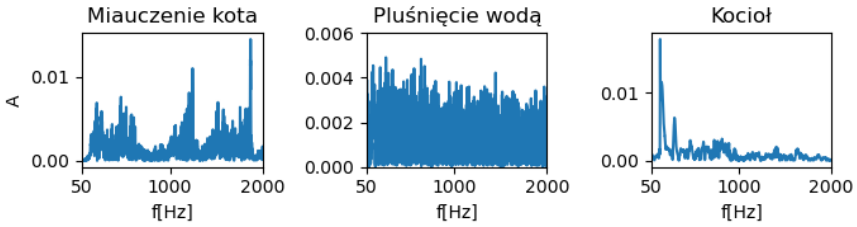
składowych. Amplitudowe widma dla dyskretnych przebiegów z Rys. 9, pokazano na Rys. 10.



Rys. 10 Przykładowe amplitudowe dyskretne widma akustyczne dla dźwięków idealnie harmonicznych.

3.3.4 Dźwięki nieharmoniczne

Za dźwięki nieharmoniczne uznać można wszystkie dźwięki które nie zaliczają się do żadnej z dwóch pierwszych grup. Są to między innymi szumy, szmery, huki oraz uderzenia (Rys. 11). Połączenie dźwięków sinusoidalnych których częstotliwości nie spełniają wzoru (2), również będzie dźwiękiem nieharmonicznym. Fala akustyczna dźwięków nieharmonicznych jest falą nieokresową.

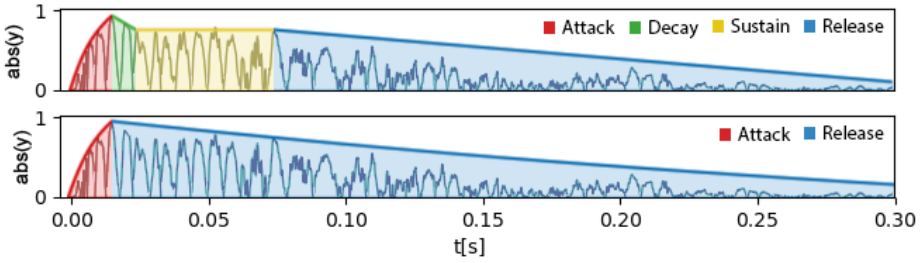


Rys. 11 Przykładowe amplitudowe dyskretne widma akustyczne dla dźwięków nieharmonicznych, wyznaczone na całej długości dźwięku.

3.3.5 Dźwięki perkusyjne

W literaturze przy opisie dźwięków produkowanych przez instrumenty muzyczne, często spotkać można się z pojęciem obwiedni *ADSR*. Nazwa ADSR to akronim pochodzący od angielskich słów: *attack*, *decay*, *sustain*, oraz *release*, a każde z nich odnosi się do innej charakterystycznej fazy dźwięku. Są to kolejno [23]: faza narastania amplitudy do wartości

maksymalnej, faza opadania amplitudy do poziomu podtrzymania (utrzymującej się stałej wartości), faza podtrzymania amplitudy, oraz faza opadania amplitudy. Takie cztery fazy dźwięku dość łatwo rozgraniczyć można np. dla dźwięku skrzypiec oraz dźwięku głosu ludzkiego, jednak w przypadku dźwięków perkusyjnych, wstępowanie fazy decay i sustain, stanowi wyjątek a nie regułę. Z tego powodu, czasem spotkać można się również z obwiedniami *AR* lub *AD*. Przykładowe obwiednie ADSR i AR naniesione na dźwięk stopy perkusyjnej pokazano na Rys. 12.



Rys. 12 Przykłady obwiedni ADSR (wyżej) i AR (niżej).
Dla lepszej prezentacji wzięto wartość bezwzględną sygnału.

Dźwięki perkusyjne neutralnie powstają w wyniku uderzenia instrumentu perkusyjnego, tudzież innego obiektu którego drgania nie są tłumione. Początkowo uderzenie powoduje duże wibracje obiektu, co skutkować będzie dużymi zniekształceniami w ciśnieniu ośrodka wokół obiektu, a tym samym dużą amplitudą rozchodzącej się fali akustycznej. Później drgania będą stopniowo wygasać, co w rezultacie doprowadzi również do zmniejszenia amplitudy fali akustycznej. Cały proces jest więc w przybliżeniu klasycznym przykładem odpowiedzi układu dynamicznego – tj. uderzanego obiektu – na pobudzenie impulsowe – tj. uderzenie. Oprócz amplitudy zazwyczaj w czasie zmienia się również charakter wibracji. Początkowo są one bardzo szybkie i chaotyczne, co powoduje powstanie wielu wysokich częstotliwości w widmie akustycznym, ale później drgania stają się bardziej wolne i równomierne, przez co w widmie akustycznym zaczynają dominować niższe częstotliwości. Prosty dźwięk perkusyjny można więc łatwo syntetycznie wygenerować, np. odpowiednio modulując w czasie częstotliwość i amplitudę dźwięku sinusoidalnego. Przykładowy

wzór na wychylenie sygnału audio bardzo prostego dźwięku perkusyjnego, matematycznie można zdefiniować jako:

$$y_P(t, f, \Delta f) = e^{-2t} A \sin(2\pi f t + e^{-20t} \Delta f) \quad (4)$$

gdzie Δf to wartość początkowego wychylenia częstotliwości od końcowej częstotliwości *f*. Wygenerowany przy pomocy tego wzoru sygnał audio, dla wartości $f = 60\ Hz$ i $\Delta f = 6\ kHz$, brzmiałby jak typowa stopa perkusyjna w Rolandzie TR-808 [24]. Jest to jeden z wielu klasycznych automatów perkusyjnych lat 80-tych. Tego typu syntetyczne dźwięki, również zawarte zostały w zbiorze danych stworzonym na potrzeby niniejszej pracy.

Dźwięki perkusyjne mogą posiadać cechy zarówno cechy dźwięków harmonicznych jak i nieharmonicznych. Można to dostrzec na ostatnim wykresie z Rys. 11. Zazwyczaj dominujące są cechy nieharmoniczne, jednak dla niektórych instrumentów perkusyjnych – np. werbli i kotłów – w fazach sustain i release gdy wibracje nie są już tak chaotyczne, widmo akustyczne nabiera także cech dźwięków harmonicznych.

3.3.6 Poziom głośności

W literaturze przyjmuje się, że głośność fali akustycznej zależy od jej natężenia [22]. Tak zdefiniowana głośność, nie ma jednak przełożenia na wrażenie głośności odczuwane przez ludzi. Dobrze tłumaczy to analogia do zmiany masy. Dużo łatwiej zauważyć zmianę z 1 kg na 2 kg, niż zmianę z 49 kg na 50 kg pomimo, iż w obu przypadkach różnica wynosi dokładnie 1 kg, gdyż pierwszym wypadku masa zwiększa się o 100%, a w drugim o zaledwie 2%. Wpływ na odczuwalną zmianę ciężaru, ma więc w praktyce względna zmiana masy, przez co wykładnicze zmiany wartości są dużo bardziej zauważalne. Zależność pomiędzy odczuwalną głośnością, a zmianą natężenia fali jest dokładnie taka sama. Z tego powodu definiuje się bardziej miarodajny wskaźnik odczuwalnej głośności o nazwie *poziom głośności* podawany w decybelach [22]. Jest on przedstawiany przy pomocy funkcji logarytmicznej, dzięki czemu liniowa zmiana poziomu głośności, mniej więcej przekłada się na liniową zmianę głośności odczuwalnej przez człowieka. Zwiększenie poziomu głośności o 3 dB przekłada się zawsze na

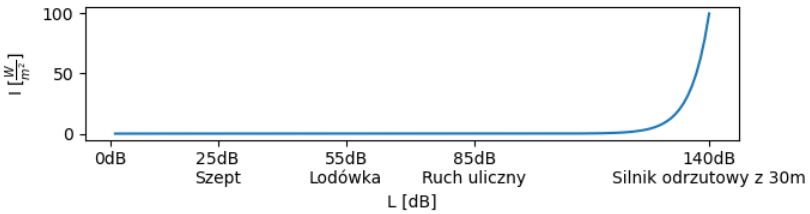
ok. dwukrotny wzrost odczuwalnej głośności. Zależność między natężeniem dźwięku, a poziomem głośności pokazano na Rys. 13. Wzór na poziom głośności fali akustycznej jest następujący:

$$L = 10 \log_{10} \frac{I}{I_0} \, [dB] \tag{5}$$

gdzie

I natężenie fali akustycznej,

I_0 pewne natężenie odniesienia dla którego $L = 0dB$, najczęściej próg słyszalności: $10^{-12} \frac{W}{m^2}$.



Rys. 13 Zależność poziomu głośności od natężenia dźwięku. Informacje o poziomie głośności dźwięków wzięto z [82].

Analogiczny wzór stosuje się na ogół wewnątrz algorytmów klasyfikacji dźwięku. Gdy znane są amplitudy składowych widma akustycznego, to w większości rozwiązań przedstawia się je postaci logarytmu dziesiętnego, by algorytm postrzegał zmiany w głośności podobnie jak człowiek.

3.3.7 Odczuwalna wysokość dźwięku

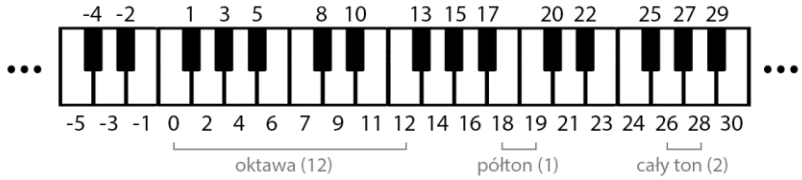
Dźwięki sinusoidalne o różnych częstotliwościach *f*, będą dla człowieka różnie brzmieć – mówi się wówczas, że mają różną *wysokość*. Podobnie jednak jak w zależności między natężeniem dźwięku i odczuwalną głośnością, relacja między częstotliwością, a odczuwalną wysokością jest wykładnicza. Można to wykazać omawiając zasadę strojenia instrumentów muzycznych na której bazuje jeden z opisanych w dalszej części pracy algorytmów konwersji dźwięku na postać czasowo częstotliwościową.

Klawiatura fortepianu składa się *k* klawiszy, a naciśnięcie każdego z nich, powoduje uderzenie wybranej struny, która drgając wygeneruje dźwięk w

przybliżeniu harmoniczny o częstotliwości bazowej f_{Bk}. Jeżeli klawisze te będą ponumerowane tak jak na Rys. 14, to w powszechnie stosowanym systemie strojenia równie temperowanego, wartości f_{Bk} określa wzór [25]:

$$f_{Bk} = f_0 \cdot 2^{\frac{k}{12}} \tag{6}$$

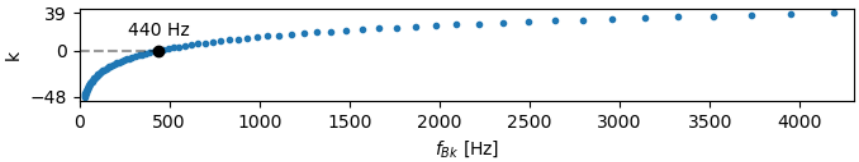
gdzie f_0 to pewna arbitralna częstotliwość przyjęta dla klawisza $k = 0$. Naciskanie od lewej po kolei klawiszy na nastrojonym fortepianu, będzie powodować w przybliżeniu liniowy wzrost odczuwalnej wysokości granego dźwięku, pomimo wykładniczego wzrostu częstotliwości fal akustycznych. Ogólnie można powiedzieć, że zmiana wysokości pomiędzy dowolną parą dźwięków o częstotliwościach bazowych f_{Bk} i $f_{Bk+\mathbb{N}}$ dla pewnego stałego $\mathbb{N}$, będzie dla człowieka odczuwalna bardzo podobnie pod warunkiem, że częstotliwości f_{Bk} i $f_{Bk+\mathbb{N}}$ nie są zbyt wysokie (o czym więcej w podrozdziale 3.4.3). Z powyższego wzoru wynika również, iż dla klawiszy oddalonych od siebie o $k = 12$, częstotliwość bazowa wzrośnie lub zmaleje dwukrotnie. Stosując terminologię muzyczną, powiedzieć można wówczas, że wysokość dźwięku wzrośnie lub zmaleje o *oktawę*. Jest to powszechnie stosowany w literaturze termin. Nazwy innych często stosowanych miar odległości pomiędzy wysokościami dźwięków zapisano na Rys. 14.



Rys. 14 Numeracja klawiszy na pianinie i podstawowe odległości między wysokościami dźwięków.

Nieliniowe postrzegane wysokości przez człowieka uwzględniane jest w mniejszym bądź większym stopniu, w większości algorytmów klasyfikacji dźwięku. Warto również zwrócić uwagę na fakt, iż zmieniając we wzorze (6) mianownik wykładnika można podzielić oktawę na dowolną liczbę stopni.

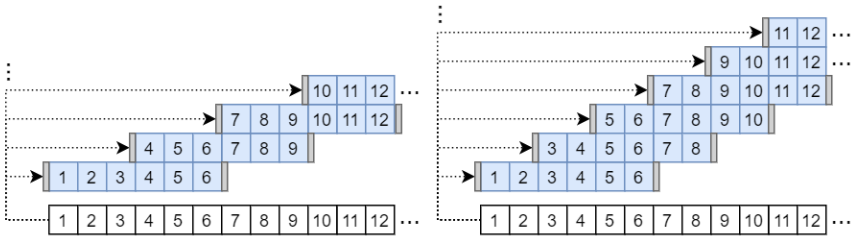
Na Rys. 15 pokazano częstotliwości f_{Bk} dla wszystkich 88 klawiszy typowo nastrojonego fortepianu. Najniższa wartość to 27.5 Hz, a najwyższa to 4186.01 Hz.



Rys. 15 Zależność f_{Bk} od k dla $f_0 = 440Hz$ w przypadku podziału oktawy na 12 części.

3.3.8 Dzielenie sygnału audio na części

W klasycznych algorytmach klasyfikacji dźwięku często stosowane jest np. tzw. *zero-crossing rate*. Statystyka ta określa średnio jak często wartości próbek zmieniają znak na przeciwny w jednostce czasu, a więc pośrednio opisuje wysokość dźwięku. Dla dźwięków perkusyjnych częstotliwość zmian znaku będzie się na ogół zmniejszać w czasie, wiec można się spodziewać, że dla różnych typów dźwięków perkusyjnych, te zmiany w czasie będą różnie przebiegać. Wyznaczając zero-crossing rate dla całego dźwięku, nie da się jednak tego zaobserwować, wobec czego tą jak i różne inne własności (np. widmo akustyczne) zazwyczaj wyznacza się niezależnie dla wielu mniejszych fragmentów badanego dźwięku.



Rys. 16 Wizualizacja podziału spróbkowanego sygnału audio na części.
Z lewej: długość okna = 6 próbek, skok = 3 próbki (nakładanie 50%)
Z prawej: długość okna = 6 próbek, skok = 2 próbki (nakładanie 66.6%)

Sygnał audio zazwyczaj dzielony jest na równo od siebie oddalone w czasie części o stałej długości. Należy więc określić dwa parametry podziału: liczba próbek przypadająca na fragment – tzw. *długość okna* (ang. *window*

length, *WL*), oraz liczbę próbek dzieląca początki sąsiadujących fragmentów - tzw. *skok okna* (ang. *hop length*, *HL*). Drugi parametr można również zdefiniować jako procentowy stopień nakładania się na siebie przedziałów – tzw. *nakładanie* (ang. *overlap*, *O*). Ideę takiego równomiernego podziału zilustrowano Rys. 16.

Znając częstotliwość próbkowania dzielonego sygnału, parametry podziału można również przedstawić w postaci czasowej. Tabela 1 zestawia często stosowane długości przedziałów w próbkach, z odpowiadającymi im długościami przedziałów w milisekundach, dla częstotliwości próbkowania 44100 Hz stosowanej podczas badań w niniejszej pracy.

Próbki	256	512	1024	2048	4096	8192
Czas [ms]	5.80	11.61	23.22	46.44	92.87	185.76

Tabela 1 Przelicznik liczby próbek na długość sygnału, dla częstotliwości próbkowania 44100 Hz.

3.3.9 Okno czasowe

Po podzieleniu sygnału audio na części, czasem aplikuje się również tzw. [26] *okno czasowe* (ang. *window function*, *WF*). Każdej próbce w każdym stworzonym przedziale, przypisuje się wówczas nową wartość wg. wzoru:

$$\check{y}_D(n) = y_D(n)WF(n) \quad dla \quad n = 0{,}1{,}2 \ldots \tag{7}$$

gdzie

$y_D(n)$ wartość wychylenia sygnału spróbkowanego dla *n*-tej próbki licząc od początku danego przedziału,

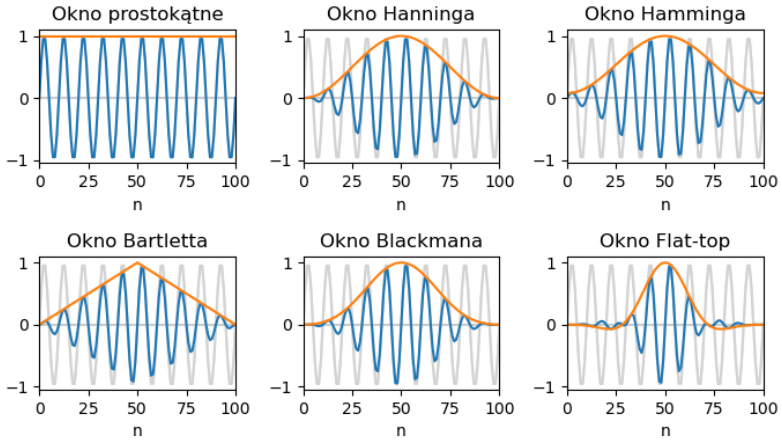
WF pewna wybrana funkcja okna czasowego,

W literaturze, w zależności od kontekstu spotkać się można z wieloma różnymi propozycjami dla definicji funkcji *WF*, jednak najczęściej stosowane wydaje się być tzw. *okno Hanninga* zdefiniowane dla spróbkowanego sygnału wzorem:

$$WF_{hann}(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{WL - 1}\right)\right) \tag{8}$$

gdzie WL określa rozmiar tego okna. W przypadku WF_{hann} będzie to rozmiar przedziału w którym $\check{y}_D(n)$ może przyjąć wartości niezerowe.

Wpływ różnych podstawowych funkcji okna czasowego na przebieg sygnału audio pokazano na Rys. 17. Jak można zauważyć, prostokątne okno czasowe $WF_{\square}(n) = 1$ pozostawia wartości sygnału bez zmian.



Rys. 17 Wpływ funkcji okna czasowego na sygnał.
Na szaro zaznaczono $y_D(n)$, na niebiesko $\check{y}_D(n)$, na pomarańczowo $WF(n)$.

3.4 Czasowo-częstotliwościowa reprezentacja dźwięku

Czasowo-amplitudowa postać dźwięku to pewna funkcja wychylenia sygnału audio. Na jej podstawie można wyliczyć własności takie jak: uśredniony poziom głośności, czasy segmentów obwiedni ADSR nałożonej na dźwięk, czy zero-crossing-rate. Wszystkie wymienione własności opisują pewne cechy dźwięku, jednak żadna z nich dobrze nie charakteryzuje jego barwy. Szczegółowe informacje o brzmieniu dźwięku, można uzyskać poprzez analizę widma akustycznego, czyli tzw. *częstotliwościową postać dźwięku*. Jest to pewna funkcja częstotliwości $\Omega(f)$ niezależna od czasu, zwracająca informację o amplitudach i fazach poszczególnych składowych sygnału. Przykłady częstotliwościowej postaci dźwięku dla amplitudy

pokazano już na Rys. 10 i Rys. 11. Znając funkcję $\Omega(f)$ można np. określić jaki zakres częstotliwości jest w danym dźwięku dominujący. Jako, że w praktyce barwa dla większości dźwięków nie jest czymś stacjonarnym, to i taka reprezentacja nie jest jednak w pełni wystarczająca.

Własności dźwięku na cele klasyfikacji, liczy się wobec tego najczęściej w oparciu o tzw. *postać czasowo-częstotliwościową*. Oddaje ona zmienność dźwięku zarówno w domenie czasu i jak i domenie częstotliwości. Taka postać we współczesnych algorytmach klasyfikacji traktowana jest często jako gotowy zbiór własności dla klasyfikatora – są to wówczas wspominane w podrozdziale 2.2.2 „ogólne cechy dźwięku". W literaturze opisano wiele metod wyznaczania postaci czasowo-częstotliwościowej. Najpopularniejszą oraz te które wykorzystano podczas badań opisano poniżej.

3.4.1 Krótko-czasowa transformacja Fouriera

Najczęściej stosowaną w literaturze metodą wyznaczania czasowo-częstotliwościowej postaci dźwięku jest tzw. *krótko-czasowa transformacja Fouriera* (ang. *short-time Fourier transform, STFT*). Jak implikuje nazwa, STFT to algorytm w którym dla wielu krótkich fragmentów sygnału audio, stosowana jest *transformacja Fouriera* (ang. *Fourier transform, FT*), a dokładniej *dyskretna transformacja Fouriera* (ang. *discrete Fourier transform, DFT*) dedykowana sygnałom spróbkowanym. DFT jest metodą wyznaczania postaci częstotliwościowej dźwięku, więc zastosowanie jej dla fragmentów sygnału rozłożonych w czasie, da w rezultacie postać czasowo-częstotliwościową.

Wynikiem krótko-czasowej dyskretnej transformaty Fouriera jest pewna macierz liczb zespolonych [26]. Jej kolumny odnoszą się do kolejnych chwil czasu, a wiersze do kolejnych prążków (pasm częstotliwości) na widmie akustycznym. Częstotliwość centralna dla i-tego wiersza to:

$$f_{STFT\,(i+1)} = i\frac{SR}{WL} \quad dla \quad i = 0,1,\ldots\left\lfloor\frac{WL}{2}\right\rfloor + 1 \tag{9}$$

gdzie SR to częstotliwość próbkowania sygnału, a operacja $\lfloor ... \rfloor$ oznacza zaokrąglenie w dół do najbliższej liczy całkowitej. Górny limit wierszy wynika z odrzucenia części prążków – tzw. częstotliwości ujemnych [26].

Z powyższej zależności wynikają trzy ważne fakty. Po pierwsze, prążki są liniowo rozmieszczone na osi częstotliwości z krokiem $\Delta f = SR/WL$. Po drugie, liczba prążków zależy bezpośrednio od długości okna. Po trzecie, częstotliwości prążków są niezależne od sygnału. Jeżeli analizowany jest sygnał o częstotliwości 5 Hz, a prążki mają częstotliwości kolejno: 0 Hz, 2Hz, 4Hz, 6Hz… to skoro nie istnieje prążek 5 Hz, korelacja sygnału „przeniknie" na prążki sąsiednie. Właśnie z tego względu stosuje się omówione wcześniej okna czasowe. Pozwalają one do pewnego stopnia ograniczyć w takiej sytuacji rozmycie energii na prążki sąsiednie [26]. Okno Hanninga uznawane jest za stosunkowo dobre dla większości zastosowań [26], więc od tego momentu jeżeli nie więc zostanie sprecyzowane jakiego okna czasowego użyto przy wyznaczaniu postaci częstotliwościowej, oznaczać to będzie, że użyto okna Hanninga.

Na podstawie otrzymanej macierzy liczb zespolonych można wyznaczyć macierz amplitud $\boldsymbol{A}_{STFT}$ (widmo akustyczne) oraz macierz faz $\boldsymbol{\varphi}_{STFT}$ (widmo fazowe). Pierwszą poprzez obliczenie z każdej liczby zespolonej wartości bezwzględnej, a drugą poprzez obliczenie z każdej liczby zespolonej kąta na płaszczyźnie zespolonej.

Jak można wywnioskować z literatury cytowanej w rozdziale 4, widmo fazowe w procesie klasyfikacji dźwięku zazwyczaj nie jest wykorzystywane, gdyż barwę lepiej opisuje widmo amplitudowe. Na jego podstawie poprzez podniesienie każdego elementu w i-tym wierszu i j-tej kolumnie do kwadratu można wyznaczyć również *widmową gęstość mocy* [27]:

$$GWM(\boldsymbol{A}_{STFT})_{\,i,j} = \left(\boldsymbol{A}_{STFT\;i,j}\right)^2 \qquad (10)$$

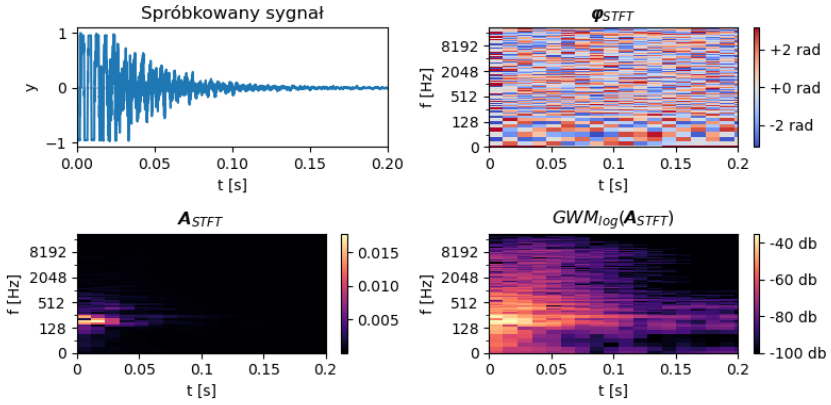
oraz widmową gęstość mocy w postaci logarytmicznej stanowiąca analogię do poziomu głośności:

$$GWM_{log}(\boldsymbol{A}_{STFT})_{\ i,j} = 10\log\left(\frac{GWM(\boldsymbol{A}_{STFT})_{\ i,j}}{GWM_{referencyjne}}\right) \qquad (11)$$

gdzie $GWM_{referencyjne}$ najczęściej przyjmuje wartość 1.

Podniesienie widma akustycznego do kwadratu we wzorze (10) ma po pierwsze podstawy fizyczne (natężenie fali akustycznej jest proporcjonalne do kwadratu amplitudy tej fali ($I \sim A^2$) [22]), a po drugie podobnie jak zresztą logarytmowanie, potęgowanie pozwala lepiej dopasować liczbową reprezentację dźwięku do realnych wrażeń słuchowych.

Przykładowy wynik STFT dla dźwięku werbla w postaci graficznej pokazano na Rys. 18. Dwa dolne wykresy ilustrujące zmiany w amplitudzie składowych, to tzw. *spektrogramy*. Dla przyjętej palety barw, barwa biała odpowiada zawsze największej wartości w macierzy, a barwa czarna wartości najmniejszej.



Rys. 18 Przykładowy wynik STFT. Parametry: SR=44.1kHz, WL=2048, O=50%.

3.4.2 Rozdzielczość czasowa i rozdzielczość częstotliwościowa

Omawiając czasowo-częstotliwościowe reprezentacje dźwięku należy odnieść się do związku pomiędzy *rozdzielczością czasową* i *rozdzielczością częstotliwościową*. By go wyjaśnić przeanalizowana zostanie sytuacja, w której dla sygnału spróbkowanego z częstotliwością $SR = 44.1$ kHz, wyznaczane jest STFT w oknach o długości WL z nakładaniem $O = 0\%$.

Znając *SR* i *WL* można w oparciu o wzór (9) i definicję częstotliwości próbkowania, wyrazić różnicę częstotliwości Δf_{STFT} pomiędzy sąsiednimi prążkami, oraz różnice czasu Δt pomiędzy początkami sąsiednich fragmentów sygnału. Zależności opisujące te różnice to odpowiednio:

$$\Delta t = \frac{WL}{SR} \quad i \quad \Delta f_{STFT} = \frac{SR}{WL} = \frac{1}{\Delta t} \ . \tag{12}$$

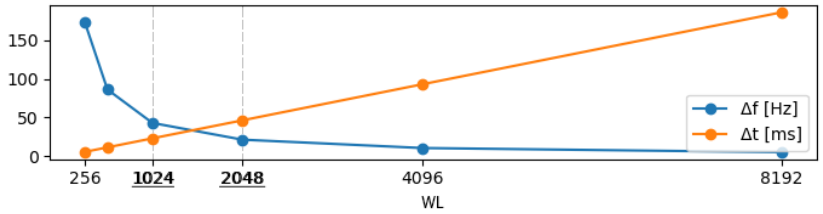
Interwały czasowe Δt dla często stosowanych długości *WL* i wskazanego *SR*, podane zostały już w podrozdziale 3.3.8 (Tabela 1). Przykładowo gdy $WL = 2048$ to $\Delta t = 46.44\, ms$, co w praktyce oznacza, że dla STFT liczonego z takim oknem czasowym, wynik to uśrednione widmo akustyczne dla przedziału o długości 46ms. Bardzo krótkie zmiany w realnym widmie akustycznym – np. takie trwające ok. 2ms – mogą zostać „przegapione", gdyż w niewielkim stopniu wpłyną na widmo uśrednione. Inną konsekwencją jest niemożność określenia punktu wystąpienia zdarzenia w badanym sygnale z dokładnością czasową poniżej Δt. Wartość Δt jest proporcjonalna do *WL*, więc im *WL* będzie mniejsze, tym wyższa będzie rozdzielczość czasowa.

Różnice częstotliwości Δf_{STFT} dla często stosowanych wartości *WL* i wskazanego *SR* opisuje z kolei Tabela 2. Przykładowo gdy $WL = 2048$ to $\Delta f_{STFT} = 21.53$ Hz, co analogicznie do tego co powiedziano powyżej oznaczać będzie, że dla STFT liczonego z takim oknem czasowym, wynik to widmo akustyczne uśrednione w przedziałach częstotliwości o szerokości 21 Hz. Jeżeli w sygnale będą występować pewne rezonujące składowe o dużych amplitudach, to ich częstotliwość można określić jedynie z dokładnością do Δf_{STFT}. Wartość Δf_{STFT} jest odwrotnie proporcjonalna do *WL*, więc im *WL* będzie mniejsze, tym niższa będzie rozdzielczość częstotliwościowa.

WL	256	512	1024	2048	4096	8192
Δf_{STFT}	172.27	86.13	43.07	21.53	10.77	5.38

Tabela 2 Przelicznik liczby próbek na różnicę w częstotliwości sąsiednich prążków wyznaczonych przez DFT dla próbkowania 44100 Hz.

Pomiędzy rozdzielczością czasową i rozdzielczością częstotliwościową zachodzi więc konflikt. Dobierając długość okna dla STFT należy liczyć się z koniecznością dokonania kompromisu pomiędzy dokładnością temporalną i dokładnością spektralną. Zależność pomiędzy Δt i Δf_{STFT} dla różnych WL przy SR=44.1 kHz ilustruje wykres na Rys. 19. Wartości WL dla których wspomniany kompromis wydaje się być dobry oznaczono grubszą czcionką.

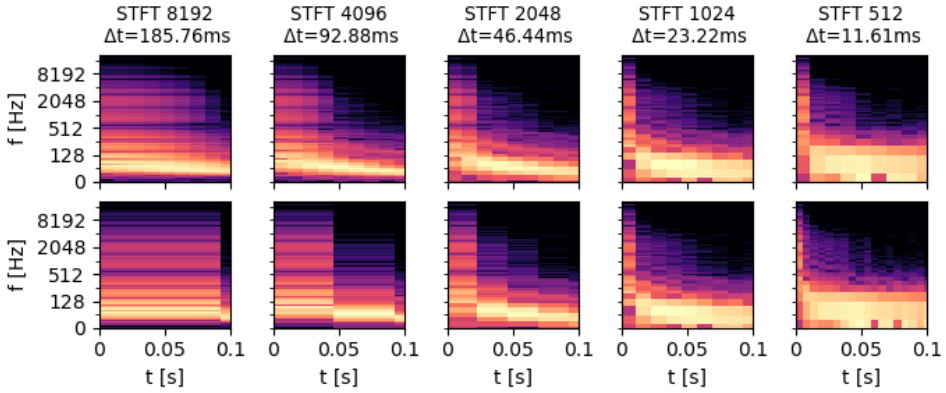


Rys. 19 Zależność Δt i Δf_{STFT} od WL dla SR=44 100 Hz.

Rozdzielczość czasową można sztucznie poprawić stosując nakładające się na siebie okna czasowe. Dla O procentowego nakładania okien, STFT zwróci $1/(1-O)$ razy więcej równomiernie rozłożonych przedziałów czasowych, dzięki czemu spektrogramy wizualnie wyglądać będą dużo lepiej i „gładziej". Pozwoli to również dokładniej określić ramy czasowe pewnych zdarzeń mimo, że każdy z przedziałów nadal obejmować będzie dokładanie WL próbek w interwale czasowym Δt. Sąsiednie przedziały stosują jednak wówczas częściowo te same próbki, co prowadzi to do generowania nadmiarowych danych. W literaturze przyjmuje się, że dobrym stopniem nakładania się okien na cele klasyfikacji dźwięku jest ok. 50% [28].

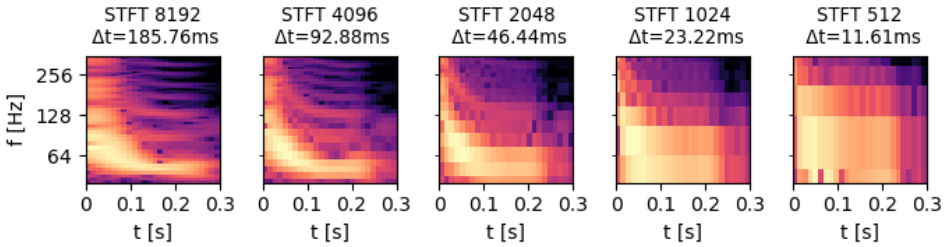
Wpływ długości okna oraz stopnia nakładania się okien STFT dla dźwięku stopy perkusyjnej zapisanej w próbkowaniu SR=44.1 kHz pokazano na Rys. 21 i Rys. 20.

Rys. 21 dobrze oddaje destrukcyjny wpływ długiego okna czasowego na rozdzielczość czasową. Na początku dźwięku stopy w fazie attack, w widmie akustycznym jest dużo wysokich częstotliwości. Trwa to jednak bardzo krótko (sądząc po wykresie w prawym dolnym rogu krócej niż 6 ms). Dla najdłuższego okna czasowego, informacja o wysokich częstotliwościach jest jednak „rozmyta" na przedział trwający aż 185 ms.



Rys. 21 Wpływ długości okna WL na rozdzielczość czasową.
W pierwszym wierszu przyjęto stały skok okna HL=512 próbek,.
a w drugim wierszu przyjęto zalecany procentowy skok okna O=50%.

Rys. 20 z kolei ukazuje destrukcyjny wpływ krótkiego okna czasowego na rozdzielczość częstotliwościową. Stopa perkusyjna jest instrumentem perkusyjnym, którego brzmienie mocno definiuje aktywność widma akustycznego w rejonie ok. 30 Hz - 300 Hz. Dla długich okien czasowych dokładnie widać na jakich częstotliwościach dźwięk stopy rezonuje. W tym przypadku dotrzeć można nawet częstotliwość bazową i jej harmoniczne. Patrząc na wynik dla najkrótszego okna czasowego, można jedynie stwierdzić, że w dolnych częstotliwościach nagromadzone jest dużo energii, ale nie sposób określić jak jest ona dokładnie rozłożona.

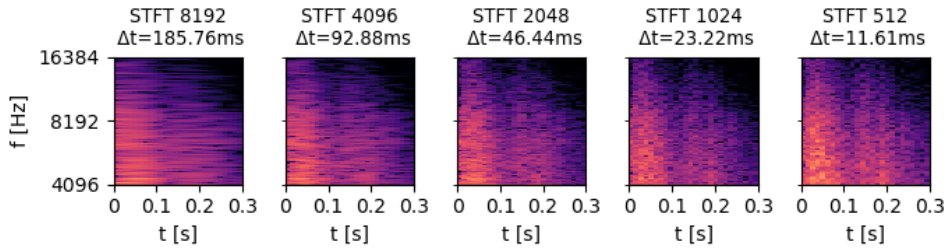


Rys. 20 Wpływ długości okna WL na rozdzielczość częstotliwościową.
Przyjęto stały skok okna HL=512 próbek.

Powyższe przykłady potwierdzają, wnioski z Rys. 19. Dobry kompromis pomiędzy dokładnością spektralną i temporalną zdają się dawać okna o długości 1024 lub 2048 próbek. Są to również wartości często polecane w

literaturze [28]. Oczywiście można również użyć okna o dowolnej długości pomiędzy 2048 a 1024, lecz w praktyce zwykło stosować się okna których długość jest potęgą dwójki, gdyż algorytm *FFT* (ang. *fast Fourier transform*) implementujący DFT, został zaprojektowany z myślą o takich wartościach.

Warto przypomnieć, że DFT rozmieszcza prążki liniowo w dziedzinie częstotliwości. Na Rys. 22 przedstawiono STFT dźwięku klaśnięcia dla *SR*=44.1 kHz. Przebadano tu analogiczny scenariusz do tego z Rys. 20, jednak dla dźwięku w którego widmie akustycznym dominują wysokie częstotliwości. Tym razem wyższa rozdzielczość częstotliwościowa nie poprawia znacząco wyniku, a można wręcz powiedzieć, że szkodzi gdyż powoduje generowanie nadmiarowych danych. Ilustruje to techniczny problem algorytmu STFT. Z punktu widzenia klasyfikacji, w spektrogramie STFT dla niskich częstotliwości prążków jest zazwyczaj za mało, a dla wysokich zbyt dużo.



Rys. 22 Wpływ długości okna WL na rozdzielczość częstotliwościową. Przyjęto stały skok okna HL=512 próbek.

3.4.3 Spektrogram w skali melowej

Spektrogram w skali melowej (ang. *mel-spectrogram, MEL-S*) jest generowany przy pomocy lekko zmodyfikowanego algorytmu STFT. Wynik STFT jest tak korygowany, by prążki w domenie częstotliwości nie były rozmieszczone liniowo, a logarytmicznie – tj. zgodnie z ludzką percepcją wysokości. Jest to obecnie prawdopodobnie najczęściej stosowana podczas klasyfikacji czasowo-częstotliwościowa reprezentacja dźwięku. W wielu różnych pracach wykazano, że klasyfikator jest w stanie dokładniej klasyfikować dźwięki, gdy zbiorem własności nie będzie zwykły spektrogram, a spektrogram w skali melowej [29].

Zależność pomiędzy wartością mel w skali melowej, a wartością f $[Hz]$ w liniowej skali częstotliwości, bywa definiowana różnie lecz najczęściej stosowana funkcja przejścia to [28]:

$$mel = 2595 \log\left(1 + \frac{f}{700}\right) \tag{13}$$

Liniowa zmiana odczuwalnej wysokości dźwięku w całym zakresie słyszalnym przekłada się wówczas na linowy wzrost w skali melowej. Wzór (13) wyznaczono w oparciu o jedno z badań, dotyczących nieliniowości ludzkiego postrzegania wysokości dźwięku na przykładzie dźwięków sinusoidalnych. Osoby poddawane testom proszone były o ręczne wybranie częstotliwości dźwięku, który odczuwalnie jest oktawę niższy od pewnego dźwięku sinusoidalnego o częstotliwości f_{sin} [30]. Dla $f_{sin} = 440\,Hz$ badani z wykształceniem muzycznym wskazywali poprawną wartość tj. ok. $220\,Hz$, jednak dla $f_{sin} = 8\,kHz$ ci sami badani wskazywali już częstotliwość o wartości ok. $1.8\,Hz$, czyli o ponad dwa razy mniejszą od tej której powinni.

Modyfikacja STFT by spektrogramy miały prążki na osi częstotliwości rozmieszczone w skali melowej, polega na lewostronnym przemnożeniu macierzy $GWM(\boldsymbol{A}_{STFT})$ przez macierz filtrów $\boldsymbol{F}_{Mel}$ o wymiarach $[N_{Mel}, \lfloor WL/2 \rfloor + 1]$ gdzie N_{Mel} jest mniejsze niż $\lfloor WL/2 \rfloor + 1$ i określa liczbę prążków (liczbę wierszy) w MEL-S opisanym macierzą:

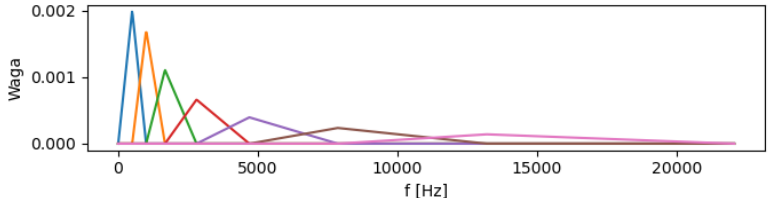
$$\boldsymbol{GWM}_{Mel} = \boldsymbol{F}_{Mel} \times GWM(\boldsymbol{A}_{STFT}) \tag{14}$$

co następnie zgodnie ze wzorem (11) przekształcane jest na postać $GWM_{log}(\boldsymbol{GWM}_{Mel})$ oznaczaną dalej jako $GWM_{log}(\boldsymbol{A}_{Mel})$. Wynikowym rozmiarem macierzy $GWM_{log}(\boldsymbol{A}_{Mel})$ będzie $[N_{Mel}, T]$ gdzie T to liczba fragmentów w jakich przetwarzano sygnał.

Każda wartość w kolumnie j-tej końcowej macierzy $\boldsymbol{GWM}_{Mel}$, będzie kombinacją liniową wszystkich wartości z kolumny j-tej macierzy $GWM(\boldsymbol{A}_{STFT})$, gdzie wartości w wierszach macierzy $\boldsymbol{F}_{Mel}$ stanową różne wersje parametrów tej kombinacji. Wiersze te to więc pewnego rodzaju filtry. Współczynniki filtrów oblicza się przy pomocy nakładających się w

skali melowej funkcji okien trójkątnych, jednak w literaturze można się spotkać z różnymi metodykami ich określania.

Na Rys. 23 na liniową oś częstotliwości naniesiono przykładowy zestaw filtrów $\boldsymbol{F}_{Mel}$ stworzony przy pomocy biblioteki Librosa [8]. Ta konkretna implementacja normalizuje wagi filtrów tak, by suma każdego wiersza w macierzy $\boldsymbol{F}_{Mel}$ była jednakowa. Punkty przecięcia czubków filtrów trójkątnych z osią częstotliwości, to nowe częstotliwości centralne prążków dla stworzonego MEL-S. Im częstotliwość jest większa, tym rzadziej pojawiają się kolejne prążki, co zgodnie z tym co zostało powiedziane w podrozdziale 3.4.2 jest efektem pożądanym. Przykładową postać MEL-S przedstawiono na Rys. 33 w rozdziale 5.



Rys. 23 Przykładowy zestaw filtrów w macierzy $\boldsymbol{F}_{Mel}$ wygenerowany przez bibliotekę Librosa. Każdy wiersz to jedna krzywa. Parametry: $N_{Mel} = 7$, $WL = 2048$ i $SR = 44.1\ kHz$, $f_{max} = SR/2$.

3.4.4 Współczynniki mel-cepstralne

Współczynniki mel-cepstralne (ang. *mel-frequency cepstral coefficients, MFCC*) to kolejna metoda reprezentacji dźwięku bazująca na STFT. Współczynniki te uzyskuje się poprzez dodatkowe przetworzenie MEL-S. Jak wyjaśniono w pracy [16] MFCC stosowane jest najczęściej w klasycznych algorytmach klasyfikacji dźwięku, gdyż pozwala zmniejszyć wielowymiarowość MEL-S.

By uzyskać MFCC należy wszystkie kolumny w MEL-S przetworzyć przy pomocy *dyskretnej transformaty cosinusowej* (ang. *discrete cosine transform, DCF*). Dzięki takiemu zabiegowi, wartości pomiędzy sąsiadującymi wierszami będą mniej skorelowane [31]. Istnieje wiele definicji DCT, jednak najczęściej stosowane przy wyliczania MFCC jest tzw.

DCT-II. Przyjmując, że element w i-tym wierszu i j-tej kolumnie macierzy $GWM_{log}(\boldsymbol{A}_{Mel})$ o rozmiarach $[N_{Mel},\ T]$ to $MEL_{\,i,j}$, wzór na element w i-tym wierszu i j-tej kolumnie macierzy $\boldsymbol{M}_{MFCC}$ przetworzonej przy pomocy algorytmu DCT-II będzie następujący [32]:

$$M_{MFCC\ i,j}\ =\ \sum_{n=1}^{N_{Mel}} MEL_{\,n,j} \cos\left[(i-1)\frac{\pi}{N_{Mel}}\left(n-\frac{1}{2}\right)\right] \tag{15}$$

Przy wyznaczaniu MFCC wyniki często ogranicza się również do N_{MFCC} górnych wierszy (tj. takich dla których $i \leq N_{MFCC} \leq N_{Mel}$) gdyż opisują one najbardziej zauważalne zmiany w widmie danego dźwięku [31]. Pierwszy wiersz przykładowo zawiera sumę wszystkich prążków w danej chwili, więc opisuje głośność dźwięku w czasie. Dolne wiersze będą z kolei zawierać więcej informacji związanych z detalami takimi jak szum tła. Wynik MFCC w którym odrzucono część wierszy można interpretować jako MEL-S poddane pewnej formie stratnej kompresji. Wynikowym rozmiarem macierzy $\boldsymbol{M}_{MFCC}$ będzie $[N_{MFCC},\ T]$. Dla MFCC bezpośrednia zależność między częstotliwością, a numerem wiersza nie występuje. Przykładową postać MFCC przedstawiono na Rys. 33 w rozdziale 5.

3.4.5 Transformacja ze stałym Q

Stosunkowo rzadko używaną w literaturze metodą uzyskiwania czasowo-częstotliwościowej reprezentacji dźwięku, jest *transformacja ze stałym Q* (ang. *constant-Q transform*, *CQT*). Częstotliwości centralne dla kolejnych prążków w CQT wyznaczane są zgodnie z metodą opisaną w podrozdziale 3.3.7 tz.: [33]

$$f_{CQT\ k} = f_{\mathrm{CQT0}} \cdot 2^{\frac{k}{BPO}} \quad dla \quad k = 0,1,\dots,N_{CQT} \tag{16}$$

gdzie BPO to liczba prążków przypadająca na oktawę (od ang. *bins per octave*), f_{CQT0} to arbitralnie wybierana częstotliwość pierwszego prążka, a N_{CQT} to liczba wszystkich prążków. Ze wzoru (16) wynikają zależności:

$$\Delta f_{CQT\ k} =\ f_{CQT\ k+1} - f_{CQT\ k} = f_{CQT\ k} \cdot (2^{\frac{1}{BPO}} - 1) \tag{17}$$

$$Q = \frac{f_{CQT\,k}}{\Delta f_{CQT\,k}} = \frac{1}{2^{\frac{1}{BPO}} - 1} \tag{18}$$

co oznacza, iż Q będące ilorazem częstotliwości centralnej i rozdzielczości prążka jest stałe, jednak szerokość pasm prążków $\Delta f_{CQT\,k}$ już nie.

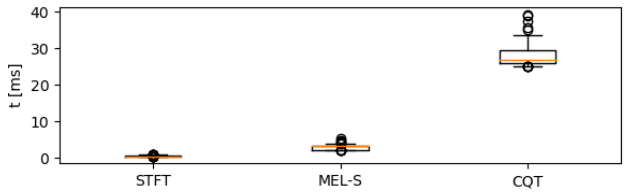
Podobnie jak w MEL-S, rozdzielczość częstotliwościowa CQT zmniejsza się wraz ze wzrostem częstotliwości. Rozłożenie prążków w obu metodach jest jednak inne. W literaturze CQT jest głównie używane jest podczas analizy materiału harmonicznego lub muzycznego, lecz pojawiają się również prace w których autorzy wykazują, że dla klasyfikacji dźwięków nieharmonicznych bazującej na uczeniu głębokim, wyniki otrzymywane gdy CQT jest macierzą własności, czasem są zauważalnie lepsze niż gdy tą macierzą będzie szerzej polecane MEL-S [34]. W rozdziale 5 niniejszej pracy wykazano, iż dla dźwięków perkusyjnych również jest to prawda.

W przeciwieństwie do MFCC i MEL-S, by otrzymać CQT nie wystarczy przetworzyć wyników otrzymanych przy pomocy STFT. Konieczne jest zastosowanie odmiennego algorytmu, aczkolwiek pokrewnego do STFT [33]. Parametry transformacji CQT to: f_{CQT0}, N_{CQT}, BPO oraz skok okna HL tudzież procentowe nakładanie okien O.

Główna różnica pomiędzy CQT i STFT polega na innym rozkładzie prążków w domenie częstotliwości i innej metodzie podziału sygnału na fragmenty. W algorytmie CQT należy określić wyłącznie skok okna HL pomiędzy kolejnymi przedziałami, gdyż ich długość zostanie dobierana automatycznie podczas obliczeń. Dla wysokich częstotliwości branych jest mniej próbek, a dla niskich więcej próbek [33]. Rozdzielczość czasowa dla wysokich częstotliwości jest więc wyższa niż dla niskich częstotliwości, dzięki czemu transformacja CQT znacząco redukuje oba opisane w podrozdziale 3.4.2 problemy transformacji STFT. Przykładową postać CQT przedstawiono na Rys. 33 w rozdziale 5.

Pewną istotną wadą CQT jest długi czas obliczeń. Wykresy pudełkowe na Rys. 24 pokazują czasy wyznaczania spektrogramów różnego typu o zbliżonym rozmiarze dla 100 różnych prób. Podczas wyznaczania STFT i

MEL-S użyto FFT, a do wyznaczenia CQT użyto rekursywnej implementacji opisanej w pracy [35]. Pomarańczowa linia w pudełkach to mediana, a rozmiary pudełek i próg dla wyników odstających określono zgodnie z bazową definicją wykresów pudełkowych Tukey-y. Kolejne wykresy pudełkowe w niniejszej pracy również stosują tą samą konwencję.



Rys. 24 Pomiarowe porównanie czasowej złożoności STFT, MEL-S i CQT.

3.4.6 Współczynniki cq-cepstralne

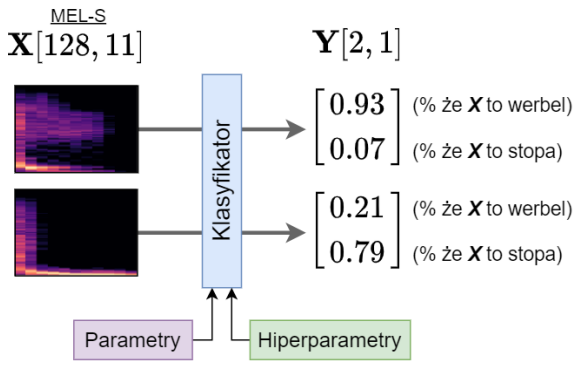
Współczynniki cq-cepstralne (ang. *Constant-Q Cepstral Coefficients, CQCC*) stanowią analogię do współczynników MFCC opisanych w podrozdziale 3.4.4. Wyznacza się je więc przy pomocy wzoru (15) z tym, że w miejsce macierzy $GWM_{log}(\boldsymbol{A}_{Mel})$ brana jest analogiczna macierz dla CQT. W literaturze spotkać się można również z innymi definicjami CQCC [36], lecz w niniejszej pracy stosowana będzie powyższa intuicyjna metoda. Przykładową postać CQCC przedstawiono na Rys. 33 w rozdziale 5.

3.5 Sztuczne sieci neuronowe

3.5.1 Sieć neuronowa jako klasyfikator

Klasyfikacja z matematycznego punktu widzenia to przekształcanie zbioru własności $\boldsymbol{X}$, w pewien wektor prawdopodobieństw $\boldsymbol{Y}$. W przypadku współczesnych metod klasyfikacji dźwięku, zbiór $\boldsymbol{X}$ będzie na ogół wynikiem przekształcenia STFT, MEL-S lub CQT, a wektor $\boldsymbol{Y}$ zawierał będzie N wartości zmiennoprzecinkowych z zakresu [0 - 1] gdzie N to liczba możliwych do przypisania etykiet. Każda wartość w wektorze $\boldsymbol{Y}$ jest

prawdopodobieństwem tego, że dana etykieta dobrze pasuje do danych opisanych własnościami $\boldsymbol{X}$. Ostatnim etapem klasyfikacji jest wybór najlepszych etykiet. Gdy do ma zostać przypisana tylko jedna etykieta, to najczęściej brana jest ta z najwyższym prawdopodobieństwem. Przebieg przekształcenia $\boldsymbol{X}$ w $\boldsymbol{Y}$ zależy od wyboru algorytmu uczenia maszynowego, parametrów modelu, oraz tzw. *hiperparametrów* które są z góry ustalane w fazie projektowej. Proces klasyfikacji zilustrowano na Rys. 25.



Rys. 25 Klasyfikacja jako przekształcenie własności $\boldsymbol{X}$ w prawdopodobieństwa $\boldsymbol{Y}$.

Jak zostało powiedziane w rozdziale 2, gdy klasyfikacji podlegają skomplikowane struktury danych takie jak dźwięk, współcześnie role klasyfikatorów pełnią najczęściej ANN (sztuczne sieci neuronowe). To co najbardziej wyróżnia ANN na tle większości klasycznych algorytmów uczenia maszynowego, to architektura warstwowa. Gdy ANN przekształca $\boldsymbol{X}$ w $\boldsymbol{Y}$ – czyli gdy realizuję *propagację w przód* danych z *warstwy wejściowej* (ang. *input layer*) do *warstwy wyjściowej* (ang. *output layer*) – to po drodze wykonuje również szereg dodatkowych przekształceń pośrednich, który gdy nie występują rozwidlenia i sprzężenia zwrotne ma postać: $\boldsymbol{X} \rightarrow \boldsymbol{U}_1 \rightarrow \cdots \rightarrow \boldsymbol{U}_{N_{HL}} \rightarrow \boldsymbol{Y}$ gdzie N_{HL} to liczba tzw. *warstw ukrytych* sieci neuronowej (ang. *hidden layers*). Pojęciem warstwa określany jest zestaw: (przekształcenie, wynik przekształcenia) toteż kolejnymi warstwami przekształcenia są: $(\boldsymbol{X}), (\rightarrow \boldsymbol{U}_1), \ldots, (\rightarrow \boldsymbol{U}_{HL}), (\rightarrow \boldsymbol{Y})$. Jak można zauważyć, warstwa wejściowa jest jedyną warstwą nie mającą własnego przekształcenia, natomiast liczba wszystkich warstw ANN jest równa $NNL = N_{HL} + 2$.

Ze względu na architekturę wyróżnić można wiele różnych rodzajów sieci neuronowych. W niniejszej pracy badane będą jednak wyłącznie *jednokierunkowe sieci neuronowe* (ang. *feedforward neural network, FNN*) bez równoległych rozwidleń. W sieciach tych przepływ informacji jest jednokierunkowy [37], więc nie mogą w nich występować przykładowo sprzężenia zwrotne. Jest to obecnie prawdopodobnie najczęściej stosowana w literaturze architektura gdy celem jest klasyfikacja danych.

3.5.2 Jednokierunkowe sieci neuronowe bez rozwidleń

Stosując konwencję z pracy [38], działanie FNN bez rozwidleń można matematycznie zilustrować jako ciąg $NNL-1$ *przekształceń warstwowych* LT na przestrzeni NNL warstwowej sieci:

$$\boldsymbol{Y} = LT_{NNL-1}(\dots LT_2(LT_1(\boldsymbol{X}, \boldsymbol{\beta}_1), \boldsymbol{\beta}_2), \boldsymbol{\beta}_{NNL-1}) \qquad (19)$$

gdzie

LT_k przekształcenie definiowane przez k-tą warstwę tz.
$\boldsymbol{U}_1 = LT_1(\boldsymbol{X}, \boldsymbol{\beta}_1)$,
$\boldsymbol{U}_k = LT_k(\boldsymbol{U}_{k-1}, \boldsymbol{\beta}_k)\ \ dla\ k = 2,3, \dots, NNL-2$,
$\boldsymbol{Y} = LT_{NNL-1}(\boldsymbol{U}_{NNL-1}, \boldsymbol{\beta}_{NNL-1})$.

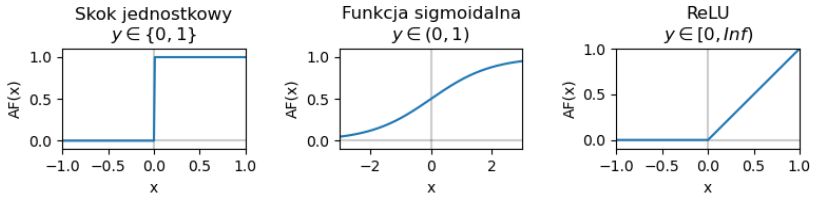
$\boldsymbol{\beta}_k$ zbiór parametrów przekształcenia k-tej warstwy.

W zależności od rodzaju danej warstwy, przypisane do niej przekształcenie LT_k może być różnie zdefiniowane. Uczenie takiej sieci polega na odpowiednim dobraniu wszystkich parametrów we wszystkich zbiorach $\boldsymbol{\beta}_k$.

3.5.3 Działanie warstw w sieciach neuronowych

Wyróżnić można wiele rodzajów warstw neuronowych. Trzy najczęściej stosowane podczas klasyfikacji to: *warstwy gęsto połączone* (ang. *densely connected layer*), *warstwy konwolucyjne* (ang. *convolutional layers*) oraz *warstwy poolingowe* (ang. *pooling layers).*

W warstwach gęsto połączonych przekształcenie wewnątrz warstwy polega na przemnożeniu jednokolumnowej macierzy wejściowej przez macierz wag i dodaniu do wyniku jedno kolumnowej macierzy przesunięć [39]. Wartości w obu macierzach to parametry warstwy. Liczba wierszy w



Rys. 26 Przykładowe funkcje aktywacji.

macierzy wag odpowiada liczbie neuronów warstwy. By przetworzyć przy pomocy warstwy gęsto połączonej wielowymiarową macierz, należy ją uprzednio sprowadzić do postaci wektorowej przy pomocy standardowej procedury algebraicznej [40]. Po powyżej opisanym przekształceniu, dla każdej wartości w wynikowym wektorze aplikowana jest również zazwyczaj pewna funkcja aktywacji. Może ona być dowolnym przekształceniem $\mathbb{R} \rightarrow \mathbb{R}$, jednak na ogół będzie miała charakter nieliniowy. Pozwala to wprowadzić nieliniowość do inaczej liniowego przekształcenia warstwy, co w efekcie sprawia że sieć neuronowa jako całość jest w stanie lepiej modelować złoże nieliniowe problemy [41]. Fragmenty przebiegów kilku często stosowanych funkcji aktywacji przedstawiono na Rys. 26.

To która z funkcji aktywacji zostanie wybrana, będzie zależało od kontekstu tj. od roli jaką dana warstwa pełni w sieci neuronowej. Przykładowo podczas klasyfikacji jedno-etykietowej dla wyjściowej warstwy sieci zazwyczaj wybiera się funkcję przejścia $softmax$ o postaci zdefiniowanej następująco:

$$\boldsymbol{Y}_{softmax} = softmax(\boldsymbol{Y}) = \left[\frac{e^{y_1}}{\sum_{k=1}^{Y_y} e^k} \quad \cdots \quad \frac{e^{y_{Y_y}}}{\sum_{k=1}^{Y_y} e^k} \right]^T. \qquad (20)$$

Dzięki takiemu przekształceniu wyjście sieci będzie miało formę rozkładu prawdopodobieństwa etykiet (suma wyjść będzie równa 1, a każde wyjście

przyjmować będzie wartości 0-1), a zastosowanie zależności wykładniczej wzmocni różnice pomiędzy prawdopodobieństwami etykiet. Taką funkcję w warstwie wyjściowej zastosowano między innymi w pracach [42] i [38].

Do warstw ukrytych próbowano przypisywać różne funkcje aktywacji, jednak w ostatnich latach najczęściej stosowane jest prawdopodobnie funkcja *ReLU* (ang. *rectified linear unit*). Dotyczy to oczywiście również prac zajmujących się klasyfikacją dźwięku takich jak chociażby [16] [2] [43] [29] [44]. Definicja ReLU jest następująca:

$$AF_{ReLU}(x) = \max(0, x)\,. \tag{21}$$

ReLU po raz pierwszy zaproponowane zostało w pracy [45] z 2010 roku i od tego czasu zyskało duże uznanie w środowisku naukowym. Autorzy wspomnianej pracy badali jak zmiana funkcji aktywacji, wpływa na dokładność wyników dla różnych rodzajów ANN. Okazało się, iż spośród wielu przebadanych często stosowanych funkcji aktywacji takich jak (np. tanh i sigmoid) funkcja ReLU pozwalała uzyskać wyniki obarczone najniższym błędem.

Warstwy konwolucyjne różnią się od warstw gęsto-połączonych tym, że stosowaną w nich operacją matematyczną nie jest mnożenie macierzy, a splot macierzy [46]. Warstwy te stanowią podstawę dla sieci konwolucyjnych. Na wejście takiej sieci podawana jest wielowymiarowa macierz, dla której wyznaczanych jest równolegle od kilku do kilkunastu splotów z różnymi macierzami parametrów (maskami) o takich rozmiarach, by wynikiem splotu była macierz dla której ostatni wymiar jest jednostkowy. Do wyniku każdego ze splotu dodawane jest także przesuniecie. Finalnie wszystkie wyniki łączone są w nową macierz, tym razem tak by rozmiar ostatniego wymiaru macierzy był równy liczbie użytych masek. Podobnie jak w warstwie gęsto-połączonej, dla otrzymanych wartości zazwyczaj aplikowana jest również pewna funkcji aktywacji. Czasem w warstwach konwolucyjnych określa się również *krok splotu* (ang. *stride*). Określa on o ile elementów przesuwana jest zawsze maska splotu. Najczęściej w literaturze stosuje się krok jednostkowy.

Warstwy poolingowe działają podobnie jak warstwy konwolucyjne z tym, że skok maski będzie równy rozmiarowi maski, a wartości w masce nie są parametrami lecz wynikają z przebiegu algorytmu. Mogą to być przykładowo jedynki podzielone przez liczbę elementów w masce, lub maska z zerami i jedynką w miejscu w którym obecnie znajduje się największy element w przetwarzanym fragmencie macierzy wejściowej. W pierwszym przypadku przeprowadzany będzie tzw. *average-pooling* (zawsze zostanie zwrócona średnia z wartości obejmowanych przez maskę), a w drugim przypadku będzie to tzw. *max-pooling* (zawsze zostanie zwrócone maksimum z wartości obejmowanych przez maskę). Warstwy poolingowe stosuje się zazwyczaj w celu zmniejszenia rozmiaru macierzy wejściowej (im większa maska poolingowe tym większa redukcja wielowymiarowości).

3.5.4 Wyznaczanie parametrów sieci

Odpowiedni wybór architektury jest ważny, jednak kluczowy dla poprawnego działania sieci neuronowej jest dobór zbiorów parametrów tj. wartości w macierzach i wektorach przesunięć stosowanych w warstwach gęsto-połączonych tudzież konwolucyjnych. Gdy sieć jest uczona, to w zerowej iteracji zazwyczaj wszystkim parametrom przypisywane są losowe wartości [47]. Algorytmy uczenia ANN są więc niedeterministyczne. To z jakim rozkładem i z jakich zakresów wartości te zostaną wybrane, zależeć będzie już od architektury uczonej sieci oraz od użytej funkcji inicjalizującej. Po takiej inicjalizacji sieć wykorzystywana do klasyfikacji danych na N klas, najczęściej zaledwie 1 na N przykładów sklasyfikuje poprawnie. Jest to więc dokładność na poziomie klasyfikatora losowego który etykiety „zgaduje”.

By ten wynik poprawić definiuje się *funkcję straty* (ang. *loss function*). Wyraża ona dla zbioru wszystkich parametrów błąd pomiędzy $\boldsymbol{Y}$ i $\boldsymbol{Y}_{ref}$ uśredniony dla wszystkich przykładów znajdujących się w zbiorze uczącym gdzie $\boldsymbol{Y}_{ref}$ to poprawne wyjście dla danego $\boldsymbol{X}$. Minimalizując funkcję straty, można znaleźć takie parametry które sprawią, iż algorytm będzie dokładniej klasyfikować dane w zbiorze uczącym.

Do minimalizowania funkcji straty, a więc do znajdywania możliwie najlepszego zestawu parametrów sieci, wykorzystuje się współcześnie głównie *algorytmy gradientowe*. W oparciu o definicję funkcji straty oraz przekształceń warstwowych, przy pomocy *propagacji wstecznej* w każdej iteracji wyznaczają one wartość pochodnej cząstkowej z funkcji straty po każdym z parametrów sieci. Dzięki temu algorytm jest w stanie określić w jaki sposób należy zmienić wartość danego parametru, by zmniejszyć wartość funkcji straty. To jak bardzo wartość pochodnej ma się przekładać na zmianę wartości parametru w kolejnej iteracji określa się za pomocą hiperparametru zwanego *współczynnik nauki* (ang. *learning rate*).

Wyliczenie wartości funkcji straty dla wszystkich przykładów w zbiorze uczącym jest czasochłonne dlatego zazwyczaj uśrednienie realizuje się wyłącznie dla części losowo wybranych przykładów zwanej w literaturze „*mini-seria*" (ang. *mini-batch*). Przykłady losowane są tak by nie powtarzały się pomiędzy mini-seriami. Po wykorzystaniu wszystkich przykładów, mija tzw. epoka nauki, a losowanie rozpoczynane jest od nowa.

Najprostszym algorytmem gradientowym jest *metoda gradientu prostego* (ang. *stochastic gradient descent, SGD*) [47]. Działa ona tak jak opisano powyżej. Czasem stosuje się również modyfikację zwaną: *metoda gradientu prostego ze współczynnikiem momentum* (ang. *gradient descent with momentum*) [48]. Od zwykłego SGD odróżnia ją to, iż podczas obliczeń w kolejnych iteracjach uwzględniane są również wyniki z poprzednich iteracji w postaci ruchomej średniej wykładniczej. W literaturze można się również spotkać z modyfikacją w której stosowane jest *momentum Nesterova* [49].

Oprócz SGD i SGD ze współczynnikiem momentum, istnieje bardzo wiele innych algorytmów uczenia sieci neuronowych. Dwa warte uwagi ze względu na swoją popularność to *RMSprop* oraz *ADAM* z 2014 roku. Podobnie jak SGD ze współczynnikiem momentum, wykorzystują one wykładniczą średnią ruchomą gradientu funkcji straty, lecz implementują dodatkowo automatyczną regulację współczynnika nauki (ang. *adaptive learning rate*) [50] [51]. Dla każdego parametru główny współczynnik nauki będzie w kolejnych iteracjach odpowiednio skalowany. W metodach tych

ważniejszą rolę od dokładnych wartości pochodnych cząstkowych w gradiencie funkcji straty, pełnią znaki tych wartości oraz historia zmian tych znaków.

3.5.5 Zapobieganie nadmiernemu dopasowaniu

Z algorytmami uczenia sieci neuronowych wiąże się pewien problem. Wytrenowany klasyfikator może mieć wysoką dokładność dla przykładów które wykorzystano podczas nauki, lecz niską dla nieznanych przykładów. Siec może „zapamiętać" jaki powinien być w pewnych przypadkach wynik, lecz nie „dostrzec" ogólnych zależności pomiędzy danymi o wspólnych etykietach – nie zgeneralizować problemu. W takiej sytuacji stwierdza się, iż zaszło *nadmierne dopasowanie* modelu do zbioru uczącego (ang. *overfit*). Klasyczne algorytmy uczenia maszynowego również są na nie podatne, jednak w przypadku uczenia głębokiego jest to bardziej powszechny problem, gdyż sieci neuronowe posiadające miliony parametrów potrafią dużo dokładniej „zapamiętywać" przykłady niż dużo prostsze klasyczne modele.

Istnieją różne metody zapobiegania nadmiernemu dopasowaniu sieci neuronowych. Obecnie bardzo dużą popularnością cieszy się tzw. *dropout* [52]. Na początku każdej iteracji uczenia (mini-batcha) dla części losowo wybranych neuronów usuwane są wszystkie połączenia wychodzące. To ile neuronów zostanie w ten sposób dezaktywowanych, zależy hiperparametu określającego prawdopodobieństwo dezaktywacji neuronów w obrębie danej warstwy. Po zakończeniu iteracji, usunięte połączenia są resetowane i cały proces zaczyna się od nowa. Uczenie stosujące dropout można więc interpretować, jako uczenie wielu mniejszych sieci ze współdzielonymi parametrami. Uczona sieć nie może polegać na obecności żadnego z połączeń toteż zapamiętywanie przykładów jest utrudnione. Sieć przy której uczeniu stosowano dropout, jest wiec poniekąd uśrednieniem wielu innych mniejszych sieci. W bibliotekach dropout jest implementowany poprzez pewną specjalną pośrednią warstwę, która jest aktywna tylko podczas uczenia [53]. Warstwa ta w każdej iteracji losowo zeruje wyjścia z warstwy

ją poprzedzającej z określonym prawdopodobieństwem. W literaturze zazwyczaj radzi się by dla warstw gęsto połączonych stosować dropout o wartości 50% [54], a dla warstw konwolucyjnych by dropoutu nie używać, lub by nie był większy niż 10% - 20% [55].

Nadmierne dopasowanie można również ograniczyć odpowiednio modyfikując funkcję straty przy pomocy *regularyzacji* [56]. Istnieją różne jej warianty, jednak najczęściej stosowana w kontekście klasyfikacji dźwięku przy pomocy sieci neuronowych, wydaje się być regularyzacja L2 stosowana między innymi w pracach [16], [38] i [29]. Regularyzacja L2 sprawia, iż w modelu preferowane są parametry o małych wartościach. Pozwala to ograniczyć sytuacje w której pojedyncza wejście neuronu ma bardzo dużą wagę i bardzo duży wpływ na sygnał wyjściowy. Model dzięki temu stanie się mniej czuły na pewne wyróżniające się w danych detale, co w efekcie pomoże w generalizacji problemu.

Ograniczyć „zapamiętywanie" przykładów można również poprzez duży zbiór uczący. Gdy przykładów i ich etykiet jest wiele, to by zminimalizować funkcję straty konieczne jest znalezienie takiego zbioru parametrów, który dla wielu różnych przykładów uwydatni najistotniejsze informacje. Oczywiście takie duże zbiory danych uczących są rzadko dostępne, więc jest to mało praktyczne podejście. Dla niewielkich zbiorów uczących nadmierne dopasowanie ograniczyć można za to poprzez *augmentację zbioru danych* (ang. *augmentation*). Jest to technika bardzo często stosowana podczas trenowania sieci neuronowych służących do klasyfikacji obrazów [57]. Każde zdjęcie można wykonać i obrobić na wiele różnych sposobów. Czasem fotografowany obiekt będzie zajmować więcej kadru, czasem mniej, czasem będzie obrócony, a czasem na fotografii pojawi się szum, lub naniesiona zostanie korekcja kolorów. Jeżeli fotografowanym obiektem będzie przykładowo pies, to człowiek w każdym przypadku będzie w stanie to stwierdzić. Gdy celem jest stworzenie sieci neuronowej wykrywającej na obrazach zwierzęta i dostępny jest niewielki zbiór uczący, to na podstawie każdego przykładu można poprzez różne operację skalowania, rotacji czy zaszumienia wygenerować dziesiątki nowych przykładów). Dla dźwięku

również można stosować podobne augmentacje. Niewielka zmiana wysokości lub długości, czy dodanie szumu lub pogłosu nie utrudni człowiekowi identyfikacji dźwięku. Najprostszą metodą na wykonanie augmentacji dźwięku w postaci próbek jest jednowymiarowa interpolacja. Przy jej pomocy można próbki „zagęścić" tak by dźwięk trwał krócej, lub „rozrzedzić" tak by dźwięk trwał dłużej. Jako, iż zwiększy lub zmniejszy to ilość okresów fali przypadających na sekundę sygnału, to oprócz długości zmieni się również odczuwalna wysokość dźwięku. Przyjmując, że s opisuje szybkość odtwarzania sygnału po interpolacji – tz. $s = 2$ dla dwa razy większego zagęszczenia i $s = 0.5$ dla dwa razy mniejszego zagęszczenia – zależność pomiędzy zmianą wysokości dźwięku w pół tonach Δst i zmianą szybkości Δs jest następująca:

$$\Delta s = 2^{-\frac{\Delta st}{12}} \tag{22}$$

Zmiana w brzmieniu dźwięków perkusyjnego po takiej augmentacji, jest bardzo podobna do zmiany w brzmieniu jaką powoduje fizyczna zmiana strojenia instrumentów perkusyjnych. Przy pomocy augmentacji można również zbalansować niezbalansowany zbiór uczący (taki w którym ilość przykładów dla różnych etykiet nie jest równa) [58]. Dla niezbalansowanego zbioru uczącego, sieć mogła by mieć wysoką dokładność dla etykiet o wielu przykładach i niską dla pozostałych. Wynika to oczywiście z uśredniania wartości funkcji straty dla mini-serii.

Ostatnią metodą zapobiegania nadmiernemu dopasowania badaną w tej pracy jest normalizacją *mini-batchy* (ang. *batch normalization*) [59]. Polega ona na normalizowaniu rozkładu statystycznego wejścia wybranej warstwy w oparciu o średnią ruchomą z średniej i wariancji wejść w kolejnych mini-seriach. Po takiej normalizacji wartość oczekiwana wejść będzie równa 0, a wariancja 1. Autorzy pracy [59] wykazują, że taki zabieg pomaga w generalizacji. Normalizacje mini-serii realizuje się poprzez dodanie do sieci specjalnych warstw, przed warstwami dla której wejścia mają zostać znormalizowane.

3.6 Walidacja krzyżowa

By poprawnie przeprowadzić walidację klasyfikatora, należy do testów wykorzystać inne przykłady od tych użytych w procesie jego uczenia. W ten sposób sprawdzone zostanie czy model generalizuje problem, oraz czy nie zaszło nadmierne dopasowanie. Jeżeli dla przykładów uczących oraz dla przykładów testowych średni błąd klasyfikacji ma podobną niską wartość, to oznacza, iż klasyfikator dobrze spełnia swoją funkcję. Oczywiście zarówno przykładów do uczenia jak i tych i do testów powinno być jak najwięcej. Pierwszych by dobrze wytrenować model, a drugich by z dużą dokładnością go ocenić. *Walidacja krzyżowa* (ang. *cross-validation*) [60] to jedna z najczęściej używanych w literaturze metod walidacji, która spełnia powyższe założenia. Zbiór wszystkich *N* przykładów dzielony jest w niej na $k > 2$ równych części (lub prawie równych jeżeli *N* nie jest podzielne bez reszty przez *k*), a następnie przeprowadzanych jest *k* cykli uczenia i walidacji tak by każda część została użyta raz do walidacji i *N-1* razy do uczenia. Wynikiem *k*-częściowej walidacji krzyżowej będzie więc *k* zestawów wskaźników jakości, wyznaczonych dla modelu uczonego i walidowanego *k* razy, przy pomocy *k* różnych zbiorów uczących i testowych. Najczęściej stosowane w pracach naukowych wartości *k* to 5 lub 10, dla których proporcja pomiędzy rozmiarami zbiorów uczących i testowych to odpowiednio 1:4 i 1:9.

3.7 Wskaźniki jakości

3.7.1 Dokładność klasyfikacji

Przyjmując, że dla pewnego N_{Train} elementowego zbioru przykładów testowych $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_{N_{Train}}\}$ wektor $\boldsymbol{\phi} = [\phi_1, \phi_2, \cdots, \phi_{N_{Train}}]$ zawiera etykiety wyznaczone przez klasyfikator, natomiast wektor $\boldsymbol{\phi}_{ref} = [\phi_{ref\ 1}, \phi_{ref\ 2}, \cdots, \phi_{ref\ N_{Train}}]$ zawiera etykiety jakie klasyfikator powinien zwrócić

dla tych przykładów, oraz że operator $a == b$ symbolizuje porównanie dwóch liczb którego wynikiem jest liczba binarna o wartości 1 gdy $a = b$ i o wartości 0 gdy $a \neq b$, *dokładność klasyfikacji* (ang. *accuracy*) można zdefiniować następująco [61]:

$$ACC = \frac{1}{N_{Test}} \sum_{i=1}^{N_{Test}} \phi_i == \phi_{ref\,i} \tag{23}$$

ACC przyjmuje wartości z zakresu [0,1] i określa dla jakiej części przykładów w zbiorze testowym klasyfikator wyznaczył prawidłową etykietę. Większe wartości ACC oznaczają wyższą dokładność. Powyższy wzór stosuje się zazwyczaj w sytuacji gdy zbiór uczący jest zbalansowany gdyż wówczas nie zachodzi tzw. *paradoks dokładności* [62].

3.7.2 Tablica pomyłek i wskaźniki pochodne

Wskaźniki ACC i $BACC$ pozwalają ocenić dany model za pomocą zaledwie jednej liczby, więc dobrze nadają się do szybkiego porównywania różnych klasyfikatorów. Niestety by dokładniej klasyfikatory porównać należy już posłużyć się bardziej złożonymi wskaźnikami. By przekonać się pomiędzy którymi etykietami dany klasyfikator najczęściej się myli, można wykorzystać np. *tablicę pomyłek* (ang. *confusion matrix*) [63]. Jest to pewna macierz $\boldsymbol{CONF}$ o rozmiarach $[N_{Lab}, N_{Lab}]$ w której wartości elementów $CONF_{i,j}$ w *i*-tym wierszu i *j*-tej kolumnie wyznaczane są jako:

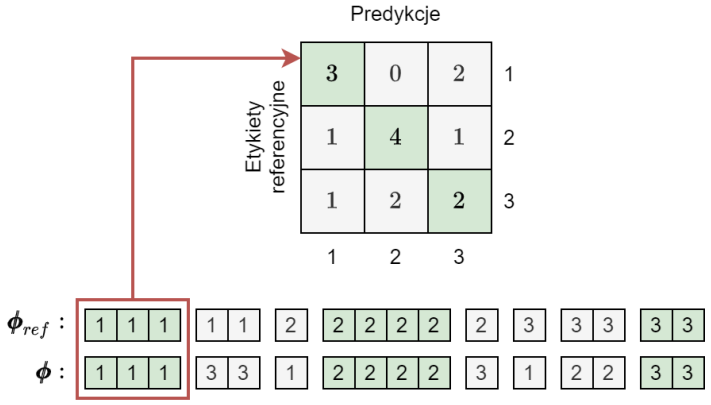
$$CONF_{i,j} = \sum_{k=1}^{N_{Test}} \left(\phi_{ref\,k} == i\right) \,\&\, (\phi_k == j) \tag{24}$$

gdzie & to operator boolowskiej operacji „i”.

By stworzyć tablicę pomyłek należy więc zliczyć wystąpienia wszystkich $(N_{Lab})^2$ permutacji: (etykieta poprawna $\phi_{ref} \in \{1, \cdots, N_{Lab}\}$, etykieta przewidziana $\phi \in \{1, \cdots, N_{Lab}\}$), a następnie wyniki przedstawić w postaci macierzy, w której wiersze odpowiadają wynikowi referencyjnemu, natomiast kolumny wynikowi predykcji. Główna przekątna zawiera

wówczas liczbę wszystkich poprawnych klasyfikacji. Przykładową tablicę pomyłek dla $\mathrm{N_{Lab}} = 3$ przedstawiono na Rys. 27.

Czasem stosuje się również znormalizowaną wersję tablicy pomyłek. Wartości są wówczas tak modyfikowane, by suma wszystkich elementów w każdym wierszu wynosiła 1.



Rys. 27 Przykładowa tablica pomyłek dla $N_{Lab} = 3$.

Na podstawie tablicy pomyłek można wyznaczyć również inne często stosowane wskaźniki jakości. Są to pewne wektory o długości N_{Lab}, w których *k*-ta wartość (oznaczana dalej indeksem dolnym *k*) pod pewnym kątem opisuje wynik walidacji z punktu widzenia *k*-tej etykiety. Najczęściej stosowane wskaźniki jakości tego typu wymieniono poniżej [63]:

- *True positives* (*TP*) – liczba poprawnych pozytywnych predykcji.

$$TP_k = CONF_{k,k} \tag{25}$$

- *False positives* (*FP*) – liczba błędnych pozytywnych predykcji.

$$FP_k = \left(\sum_{i=1}^{N_{Lab}} CONF_{i,k}\right) - TP_k \tag{26}$$

- *False negatives* (*FN*) – liczba błędnych negatywnych predykcji.

$$FN_k = \left(\sum_{i=1}^{N_{Lab}} CONF_{k,i}\right) - TP_k \tag{27}$$

- *True negatives* (*TN*) – liczba poprawnych negatywnych predykcji.

$$TN_k = \left(\sum_{i=1}^{N_{Lab}} \sum_{j=1}^{N_{Lab}} CONF_{i,j} \right) - (TP_k + FP_k + FN_k) \qquad (28)$$

- *Czułość* (ang. *sensitivity*) zwana również *true positive rate* (*TPR*), *hit-rate* lub *recall* – określa jaka część spośród predykcji która miała być pozytywna jest pozytywna. Jest to więc dokładność z perspektywy jednej wybranej etykiety.

$$TPR_k = \frac{TP_k}{TP_k + FN_k} \qquad (29)$$

- *Specyficzność* (ang. *specificity*) zwana również *true negative rate* (*TNR*) – określa jaka część spośród predykcji która miała być negatywna jest negatywna.

$$TNR_k = \frac{TN_k}{TN_k + FP_k} \qquad (30)$$

- *Precyzja* (ang. *precision*) zwana również *positive predictive value* (*PPV*) – określa jaka część spośród wszystkich pozytywnych predykcji jest poprawna.

$$PPV_k = \frac{TP_k}{TP_k + FP_k} \qquad (31)$$

- *Współczynnik* $F\beta$ [64] – można interpretować jako harmoniczną średnią [65] czułości i precyzji, przy czym im wartość β będzie większa, tym czułość będzie bardziej wpływać na końcowy wynik. Oznaczenie współczynnika jako F2 oznacza, iż przyjęto $\beta = 2$.

$$F\beta_k = (1 + \beta^2) \frac{PPV_k \cdot TPR_k}{(\beta^2 \cdot PPV_k) + TPR_k} \qquad (32)$$

4 Rozwiązania znane z literatury

Ogólny schemat klasyfikacji dźwięku dla podejścia stosującego uczenie głębokie, przedstawiono na Rys. 2 w podrozdziale 2.2. W pierwszym etapie wyznaczana jest wybrana postać czasowo częstotliwościowa, którą następnie w drugim etapie podaje się na wejście pewnej sieci neuronowej. By znaleźć wysokiej jakości rozwiązanie dla zadanego problemu, należy więc zbadać różne kombinacje algorytmów i hiperparametrów dla tych dwóch etapów, a następnie wybrać kombinację najlepszą. Jest to jednak dość trudne gdyż liczba wszystkich kombinacji jest bardzo duża. Warto więc ograniczyć wszystkie kombinacje do tych potencjalnie dobrych, poprzez analizę rozwiązań z literatury o tej samej tematyce.

Zbiór uczący stworzony na potrzeby niniejszej pracy został omówiony dokładniej w rozdziale 5, jednak w skrócie mówiąc zawiera on dźwięki perkusyjne podzielone na 10 kategorii, z czego w obrębie jednej kategorii dźwięki mogą się bardzo różne, a różnice pomiędzy kategoriami czasem są dość niewielkie aczkolwiek możliwe do zauważenia przez człowieka. Do klasyfikacji używane będą zgodnie z tematem pracy jednokierunkowe sieci neuronowe. Niestety nie wydaje się by w literaturze można było obecnie znaleźć chociażby jedno badanie które również można opisać w powyższy sposób. Istnieje jednak wiele badań w których tematyka jest zbliżona. W szczególności dobrym punktem odniesienia wydają się być popularne w ostatnich latach badania dotyczące klasyfikacji dźwięków miejskich i dźwięków związanych z różnymi zdarzeniami w otoczeniu (rozbicie szkła, wiercenie, czy szum klimatyzacji). Dźwięki te podobnie jak dźwięki perkusyjne mają w głównej mierze nieharmoniczne widmo akustyczne, część dźwięków ma podobną barwę (np. głośny silnik i wiercenie), a część posiada nawet typowy dla dźwięków perkusyjnych impulsowy przebieg

amplitudy (np. młot pneumatyczny). Ze względu na te podobieństwa istnieje duża szansa, iż rozwiązania które sprawdziły się podczas klasyfikacji takich dźwięków, sprawdzą się również przy klasyfikacji dźwięków perkusyjnych.

W dalszej części rozdziału krótko opisano więc kilka rozwiązań znanych z literatury. Mniej istotne informacje będą pomijane. Dla skrócenia opisu, stosowane będą oznaczenia zgodnie z poniższymi schematami:

- $Dense(150, ReLU)$ – warstwa gęsto połączona o 150 neuronach i funkcji aktywacji ReLU.
- $Conv(3{,}3{,}10, stride = [2{,}2], ReLU)$ – warstwa konwolucyjna o funkcji aktywacji ReLU, mająca 10 różnych masek których pierwsze dwa rozmiary to [3,3], a skok maski to [2,2]. Jeżeli stride nie jest określony to znaczy, że ma rozmiary [0,0].
- $MaxPool(3{,}3, stride = [2{,}2])$ – warstwa wykonująca max-pooling z maską o rozmiarze [3,3] i skokiem [2,2]. Jeżeli stride nie jest określony to znaczy, że jest taki sam jak rozmiar maski.
- $Dropout(50\%)$ – warstwa dropoutowa dodająca 50% dropoutu do warstwy ją poprzedzającej.
- $BatchNorm()$ – warstwa wykonująca normalizację mini-serii.
- $STFT(1024, 50\%)$ – spektrogram $GWM_{log}(\boldsymbol{A}_{STFT})$ wyznaczony dla okna o długości 1024 próbek i nakładaniu 50%. Użyta funkcja okna to albo okno Hanninga albo w pracy nie sprecyzowano.
- $MelS(1024, 50\%, 80)$ – spektrogram $GWM_{log}(\boldsymbol{A}_{Mel})$ wyznaczony dla okna o długości 1024 próbek i nakładaniu 50% oraz banku filtrów $\boldsymbol{F}_{Mel}$ mającego 80 wierszy. Użyta funkcja okna to albo okno Hanninga albo w pracy nie sprecyzowano.
- $CQT(80{,}12{,}512)$ – spektrogram ze stałym Q $GWM_{log}(\boldsymbol{A}_{CQT})$ liczony co 512 próbek sygnału. Na każdą oktawę przypada 12 prążków, a suma wszystkich prążków to 80. Użyta funkcja okna to albo okno Hanninga albo w pracy nie sprecyzowano.

4.1 Przegląd literatury dotyczącej klasyfikacji dźwięku

4.1.1 Dźwięki perkusyjne o podobnej barwie

Uderzając ten sam instrument perkusyjny w różny sposób w różnych miejscach, można uzyskać szeroki zakres brzmień. W pracy [5] z 2004 roku próbowano klasyfikować dźwięki powstałe w wyniku uderzeń w werbel, ze względu na sposób w jaki te dźwięki wytworzono. Zdaniem autorów jest to trudniejsze zadanie od klasyfikacji dźwięków dla różnych instrumentów perkusyjnych, gdyż różnice w brzmieniu są mniej wyraźne. Na potrzeby badań stworzony został zbiór uczący zawierający 1260 przykładów podzielonych na 7 równych grup. Wejściem klasyfikatorów zawsze był ręcznie wyselekcjonowany zbiór wielu różnych własności. Na podstawie reprezentacji czasowej wyznaczono np. zero-crossing-rate i czas fazy attack, a na podstawie reprezentacji częstotliwościowej wyznaczono np. MFCC oraz centroid spektrogramu (jest to średnia ważona częstotliwości prążków gdzie wagi to ich GWM_{log}). Przetestowano trzy algorytmy klasyfikujące: bardzo prostą sieć neuronową z jedną ukrytą warstwą mającą sześć neuronów trenowaną przez 1000 epok, SVM (ang. *support vector machine*) oraz algorytm k-najbliższych sąsiadów (ang. *k-nearest neighbor, kNN*). Najlepsze wyniki dla różnych klasyfikatorów były podobne, jednak konsekwentnie wysoką dokładność zapewniał jedynie kNN. Rozmiar okna użyty przy wyznaczaniu własności, okazał się nie mieć dużego wpływu na wyniki. Dokładność klasyfikacji dla algorytmu kNN wynosiła 89.3% gdy klasyfikowano dźwięki ze wszystkich siedmiu kategorii, lub 99.8% gdy zbiór uczący ograniczono do trzech kategorii o najbardziej różnym brzmieniu. Drugi wynik jest niezaprzeczalnie bardzo dobry jednak należy zaznaczyć, iż dźwięki testowe zostały nagrane w tym samym pomieszczeniu, przy użyciu tego samego mikrofonu, stojącego zawsze w tym samym miejscu. Dodatkowo wykorzystano zaledwie trzy różne werble, uderzane na zmianę trzema różnymi pałeczkami. Dźwięki w obrębie danej kategorii miały więc

zapewne bardzo podobne brzmienie, toteż ciężko przewidzieć jaka była by dokładność modelu, dla trudniejszego zbioru uczącego.

Duże podobieństwo w brzmieniu może również występować pomiędzy rożnymi instrumentami perkusyjnymi podobnego typu. Dobrym przykładem są chociażby talerze perkusyjne, które oczywiście podobnie jak werble mogą być także różnie uderzane. W pracy [4] z 2015 roku klasyfikowano dźwięki pięciu talerzy perkusyjnych („china", „crash", „hi-hat", „ride", „splash") tak by przypisana pojedyncza etykieta identyfikowała instrument i wykorzystaną technikę grania na nim. Zbiór uczący podzielony był w sumie na 12 kategorii, przy czym na każdą przypadało od 36 do 152 przykładów. Autorzy udostępnili link do pobrania zbioru uczącego, jednak na czas pisania niniejszej pracy nie jest on już aktywny. W pracy przebadano różne metody generowania własności i różne algorytmy klasyfikacji takie jak: naiwny klasyfikator Bayesa, C4.5, las losowy, kNN oraz SVM. Dla wszystkich algorytmów testowano wiele kombinacji hiperparametrów. Stosując jako wejście MFCC o 40 współczynnikach (dla takiej liczby współczynników wyniki były najlepsze), a jako klasyfikator SVM, udało się osiągnąć wskaźnik jakości F2 na poziomie 74.93% gdy wykorzystano do uczenia wszystkie kategorie, oraz na poziomie 94.01% gdy przykłady podzielono tylko ze względu na rodzaj talerza perkusyjnego. Są to wyniki o odpowiednio 11.56 p.p. i 2.58 p.p. mniejsze od najlepszych jakie uzyskano gdy na wejście SVM podano LSF (ang. *linear spectra frequencies*) nie opisane w niniejszej pracy. Wyniki dla MFCC i LSF są dość podobne, toteż można uznać, iż MFCC jest równie dobrą reprezentacją dźwięków perkusyjnych. Najlepszym klasyfikatorem po SVM okazał się być w tym przypadku las losowy.

4.1.2 Dźwięki perkusyjne

Najbardziej podobna pod względem zakresu badań do niniejszej pracy, jest prawdopodobnie praca [2] z 2016 roku. Klasyfikowano w niej dźwięki perkusyjne na zaledwie trzy kategorie („stopa", „werbel" i „hi-hat") jednak podobnie jak w niniejszej pracy, autorzy próbowali w tym celu stosować CNN. Zbiór testowy składał się 3713 przykładów (na każdą kategorię

przypadało od 1124 do 1320 przykładów) których średnia długość to około jednej sekundy. Wszystkie dźwięki przekonwertowano do próbkowania 44.1 kHz i wyznaczano MelS(1024, 50%, 80) znormalizowane do zakresu [0-1]. Końcowa macierz własności miała 8 kolumn (chwil czasu) co nie wynika z parametrów MEL-S, długości plików i próbkowania, jednak nie opisano w pracy w jaki sposób liczba kolumn została zmniejszona. Przetestowano klasyfikację przy pomocy algorytmu kNN, lasu losowego oraz następującej CNN: Conv(2, 4, 16, ReLU) → MaxPool(3, 1) → BatchNorm() → Conv(3, 3, 32, ReLU) → MaxPool(3, 1) → BatchNorm() → Conv(3, 3, 64, ReLU) → MaxPool(3, 1) → BatchNorm() → Dense(256, Tanh) → Dropout(50%) → Dense(3, Softmax) przy czym warstwy BatchNorm() i Dropout() były dodawane opcjonalnie. Sieć uczono przez 300 epok pewnym algorytmem gradientowym ze współczynnikiem nauki 0.005 (więcej szczegółów o metodzie nie podano). Do testu użyto 20% wylosowanych przykładów bez walidacji krzyżowej. Dla kNN oraz CNN w wariantach bez warstw BatchNorm() osiągnięto dokładność 96%, dla CNN z warstwami BatchNorm() dokładność 97%, natomiast dla lasu losowego osiągnięto dokładność 91%. W każdym przypadku dokładność była więc bardzo wysoka, a algorytm kNN okazał się lepszy od lasu losowego i niemal równie efektywny co CNN. Niestety zbiór uczący nie jest udostępniony do wglądu, co uniemożliwia określenie czy wysoka dokładność dla algorytmu kNN nie wynika aby z prostych przykładów, lub dużych podobieństw pomiędzy zbiorem uczącym i walidującym. Powyżej opisana sieć neuronowa w dalszej części niniejszej pracy będzie nazywana GajhedeCNN (od nazwiska pierwszego wymienionego w publikacji autora).

4.1.3 Dźwięki miejskie i związane ze zdarzeniami

Praca [16] z 2015 roku to jedna z pierwszych prac dotycząca klasyfikacji dźwięków miejskich przy pomocy CNN. Do testów wykorzystano trzy publicznie dostępne zbiory, przy czym dwa ostatnie stworzył sam autor omawianej pracy. Są to kolejno: UrbanSounds8k [66] (8732 dźwięków trwających poniżej 4 sekund, podzielonych na 10 klas: „klimatyzator”, „bawiące się dzieci”, „klakson samochodowy”, „wiercenie”, „szczekanie

psa", „silnik", „wystrzał z pistoletu", „młot pneumatyczny", „alarm", „muzyka grana na ulicy"), ESC-50 (2000 dźwięków trwających 5 sekund podzielonych na 50 równych grup z 5 kategorii: „zwierzęta", „natura", „miasto", „mieszkanie", „ludzkie dźwięki nie związane z mową") i ESC-10 (400 dźwięków wyodrębnionych z ESC-50 podzielonych na 10 grup: „szczekanie psa", „deszcz", „fale morskie", „płaczące dziecko", „tykający zegar", „kichnięcie", „lecący helikopter", „piła mechaniczna", „ognisko", „kogut"). Dokładność klasyfikacji przeprowadzonej przez człowieka dla zbiorów ESC-50 i ESC-10 to odpowiednio: 81% i 96%. By zwiększyć rozmiar zbiorów ESC-50 i ESC-10 zastosowano augmentację odpowiednio cztero lub dziesięciokrotną (różne kombinacje opóźnień, przyśpieszeń i zmiany wysokości). Dla największego zbioru UrbanSounds8k augmentacji nie zastosowano, gdyż poprawa wyników była niewielka, a proces uczenia stawał się znacząco dłuższy. Podczas wyznaczania własności wszystkie dźwięki podzielono na mniejsze fragmenty o równych długościach, przy czym wyodrębniono dwa warianty tego podziału: 950 ms z nakładaniem 50% i 2.3s z nakładaniem 90%. Następnie fragmenty przekonwertowano do próbkowania 22.05 kHz i wyznaczano dla każdego macierz własności o rozmiarach [60, 41, 2] gdzie pierwszy kanał to MelS(1024, 50%, 60) znormalizowany do zakresu [0-1], a drugi kanał to tzw. delta z tego mel-spektrogramu (od każdej kolumny odejmowana jest kolumna poprzednia by wyznaczyć zmiany częstotliwości w czasie). Końcowa etykieta dźwięku wyznaczana jest w oparciu o uśrednione prawdopodobieństwa etykiet wyznaczone dla każdego fragmentu danego dźwięku. Taki zabieg jest konieczny gdyż dla większości dźwięków miejskich nie można łatwo wskazać czasu początku i końca. Przykładowo dźwięk ogniska to przedzielone ciszą pojedyncze „strzały". Dla kolejnych omawianych prac metodologia jest zazwyczaj analogiczna. Do klasyfikacji fragmentów użyto CNN którego architekturę wybrano w obtarciu o wstępne testy: Conv(57, 6, 80, ReLU) → MaxPool(4, 3, stride=[1, 3]) → Dropout(50%) → Conv(1, 3, 80, ReLU) → MaxPool(1, 3, stride=[1, 3]) → Dense(5000, ReLU) → Dropout(50%) → Dense(5000, ReLU) → Dropout(50%) → Dense(10 lub 50, Softmax) Do każdej warstwy posiadającej parametry dodano również

regularyzację L2 ze współczynnikiem kary 0.001. Sieć uczono przy pomocy SGD ze współczynnikiem momentum Nesterova 0.9, w mini-batchach po 1000 przykładów, przez 300 epok ze współczynnikiem nauki 0.002 gdy użyto krótkich fragmentów, oraz przez 150 epok ze współczynnikiem nauki 0.01 gdy użyto długich fragmentów. Dokładność klasyfikacji w zależności od zbioru uczącego wyznaczono przy pomocy 5-cio lub 10-cio częściowej walidacji krzyżowej. Nie licząc jednego przypadku, lepsze wyniki otrzymano dla dłuższych fragmentów dźwięków. Najwyższe dokładności po uśrednieniu wyników walidacji krzyżowej to: dla UrbanSounds8k 73.1%, dla ESC-50 64.5% i dla ESC-10 79%. Pomimo, iż dla ESC-50 i ESC-10 otrzymana dokładność jest aż o ok. 20 p.p. mniejsza od tej jaką osiąga człowiek, to są to wyniki zadowalajcie. Dla porównania stosując klasyczne podejście w którym macierz wejściowa zawiera MFCC i zero-crossing-rate (w pracy nie podano rozmiarów tej macierzy), a klasyfikator to las losowy, średnie dokładności będą w stosunku do powyższych wyników: dla UrbanSounds8k mniejsze o 5.1 p.p, dla ESC-50 mniejsze o 22.5 p.p i dla ESC-10 mniejsze o 7 p.p.. Dokładność klasyfikacji dla zbioru ESC-50 zawierającego aż 50 różnych grup uległa dużej poprawie co sugeruje, że sieci neuronowe najlepiej nadają się do złożonych problemów. Średnia dokładność uzyskana dla UrbanSounds8k była na poziomie średniej dokładności uzyskanej wówczas już w innych pracach stosujących klasyczne rozwiązania, jednak wariancja okazała się dużo niższa. W tabeli pomyłek można było zauważyć, że z najniższą dokładnością sklasyfikowane zostały dźwięki mocno zmienne w czasie (np. młot pneumatyczny) co zdaniem autora można by poprawić zmniejszając szerokość masek (liczbę kolumn) w warstwach konwolucyjnych. Reasumując, autorowi udało się więc wykazać, iż konwolucyjne sieci neuronowe nadają się do klasyfikacji różnych dźwięków otoczenia. Zwraca on jednak uwagę, na duży minus jakim jest długi proces uczenia CNN.

Wyniki z omówionej powyżej pracy [16] próbowano poprawić między innymi w pracy [38] z 2017 roku. Jej autorzy w szczególności skupili się na zbadaniu wpływu jaki augmentacja ma na otrzymywane wyniki, gdyż ich zdaniem to właśnie brak dużych zbiorów uczących sprawia, iż dokładność

dla klasyfikacji stosującej uczenie głębokie jest czasem porównywalna do dokładności dla klasyfikacji stosującej klasyczne algorytmy. Do testów wybrano zbiór UrbanSounds8k dla którego w pracy [16] augmentacja nie była stosowana. Metody augmentacji jakie przetestowano to: wydłużanie w zakresie ±20%, zmiana wysokości w zakresie: ±2 półtonu oraz ±3.5 półtonu, kompresja dynamiki (wyrównywanie amplitudy w czasie) oraz zaszumienie. Dźwięki są klasyfikowano przy pomocy wielu 3 sekundowych fragmentów oddalonych od siebie o 23 ms. Sygnał audio konwertowano do próbkowania 44.1 kHz i wyznaczano dla niego MelS(1024, 0%, 128). Finalne macierze własności były o rozmiarach [128,128,1]. Autorzy nie sprecyzowali czy macierze te później znormalizowano. Architektura testowanej sieci była następująca: Conv(5, 5, 24, ReLU) → MaxPool(2, 4) → Conv(5, 5, 48, ReLU) → MaxPool(2, 4) → Conv(5, 5, 48, ReLU) → Dropout(50%) → Dense(64, ReLU) → Dropout(50%) → Dense(10, Softmax). Do warstw gęsto połączonych dodano również regularyzację L2 ze współczynnikiem kary 0.001. Zdaniem autorów zastosowanie większej ilości warstw konwolucyjnych z mniejszymi maskami, pozwoli sieci lepiej „dostrzec" różne lokalne cechy dźwięków. Sieć uczono przy pomocy zwykłego SGD z logarytmiczną funkcją straty, w mini-batchach po 100 przykładów, przez 50 epok ze współczynnikiem nauki 0.01. Dokładność klasyfikacji wyznaczono przy pomocy 10-cio częściowej walidacji krzyżowej, jednak do uczenia wykorzystano jedynie 8 z 9 części. Za pomocą ostatniej części określano dla której epoki dokładność była najlepsza. Sieć osiągnęła średnią dokładność 74% bez augmentacji (8732 oryginalnych przykładów), oraz średnią dokładność 79% gdy zastosowano wszystkie techniki augmentacji na raz ($8732 + 5 \cdot 4 \cdot 8732 = 183\,372$ przykładów). W obu przypadkach uzyskano więc średnią dokładność wyższą niż w pracy [16] (warto jednak odnotować, iż również wariancja była nieznacznie większa). Analizują tablicę pomyłek zauważyć można było, że dla pewnych kategorii augmentacja pogorszyła dokładność, gdyż dźwięki zapewne zbyt różniły się wówczas od oryginału. Autorzy radzą by techniki augmentacji dobierać do problemu, a być może nawet by dla różnych typów dźwięków w zbiorze uczącym stosować różne augmentacje. Najlepsze efekty okazała się dawać augmentacji przy pomocy

zmiany wysokości, gdyż najkorzystniej wpłynęła na średnią dokładność, nie pogarszając jednocześnie wyników dla żadnej etykiety. Powyżej opisana sieć neuronowa w dalszej części niniejszej pracy będzie nazywana SalamonCNN.

Jak można zauważyć w pracy [2], [16] i [38] dane podane na wejście CNN były mel-spektrogramem, lub bezpośrednio na nim bazowały. W pracy [29] z 2017 roku zbadano więc jak na dokładność klasyfikacji dźwięku za pomocą CNN wpłynie również wybór innej, rzadziej stosowanej w literaturze reprezentacji dźwięku. Do testów użyto zbiorów ESC-50 i UrbanSounds8K, a pomiary wykonano dla czterech różnych sieci. Sieć dla której wyniki w większości przypadków były najlepsze ma następującą architekturę: Conv(3, 3, 180, ReLU) → MaxPool(4, 4) → Dropout(50%) → Dense(800, Liniowa) → Dropout(50%) → Dense(10 lub 50, Softmax). Do każdej warstwy posiadającej parametry dodano również regularyzację L2 (wartości współczynnika kary nie podano). Sieć trenowano algorytmem Adam z logarytmiczną funkcją straty w mini-batchach po 100 przykładów przez 200 epok dla zbioru ESC-50, oraz przez 100 epok dla zbioru UrbanSounds8k. Badana była również głębsza sieć podobna do tej w pracy [38], jednak otrzymane wyniki były wówczas gorsze. Zdaniem autora nie jest to aż tak „trudny problem", więc dla złożonych sieci zachodzi nadmierne dopasowanie. Dźwięki klasyfikowano jako całość, a nie jak w poprzednich dwóch cytowanych pracach przy pomocy wielu fragmentów. Każdy dźwięk przekonwertowano do próbkowania 22.05 kHz i skrócono lub wydłużono (dodając zera) tak by trwał 4 sekundy. Na podstawie tak przygotowanych serii próbek, wyznaczono kolejno: STFT□(2048, 50%), STFT□(512, 50%), MelS□(2048, 50%, 512), MelS□(512, 50%, 128), CQT□(1024, 128, 1024) i CQT□(256, 32, 256). Wyniki zawsze normalizowano do zakresu [-1, 1], a by porównanie było bardziej sprawiedliwe, różne macierze własności są interpolowane następnie dwuwymiarowym algorytmem Lanczosa [67] do jednego z dwóch stałych rozmiarów: [154, 12] gdy indeks dolny to □, lub [37, 50] gdy indeks dolny to □. Przebadano również MFCC i CWT (ang. *continuous wavelet transform*), jednak dla tych reprezentacji dźwięku wyniki okazały się najgorsze i dokładne hiperparametry nie zostały w pracy zawarte.

Zazwyczaj najlepsze wyniki zapewniały własności MelS□ i CQT□. Dla podanej sieci najwyższe osiągnięte mediany dokładności oraz ich odchylenia standardowe to: 74.66±3.39% dla MelS□ i zbioru UrbanSounds8k oraz 55.00±1.63% dla MelS□ i zbioru ESC-50. Dokładności przedstawiono przy pomocy median, więc niestety nie można ich porównać ze średnimi dokładnościami w pracach poprzednich. Według autora z tablic pomyłek wynika, że własności w wersjach □ i □ nadają się lepiej do klasyfikacji odmiennych typów dźwięków. Pierwsze lepiej klasyfikują dźwięki o wysokich częstotliwościach, gdyż dostępnych jest więcej prążków, a drugie lepiej klasyfikują dźwięki mocno zmienne w czasie gdyż dostępnych jest więcej chwil czasu. Są to oczywiście wnioski zgodne z tym co opisano w niniejszej pracy już w podrozdziale 3.4.2 . Powyżej opisana sieć neuronowa w dalszej części niniejszej pracy będzie nazywana HuzaifahCNN.

CQT do klasyfikacji próbowano również stosować w pracy [34] z 2016 roku. Autorom udało się wówczas wykazać, iż dla proponowanej sieci oraz dla testowanego przez nich zbioru DCASE2016 [68] zawierającego dźwięki otoczenia, CQT lepiej nadaje się do wyznaczenia macierzy własności niż MEL-S. Do klasyfikacji użyto CNN o architekturze podobnej do tej w pracy [29], jednak część konwolucyjna składała się z dwóch równoległych gałęzi. W pierwszej gałęzi maski miały wiele wierszy, a w drugiej wiele kolumn. Zamysłem autorów było by jedna gałąź uwydatniała temporalny charakter dźwięku, a druga spektralny. Zamiast funkcji aktywacji ReLU użyto również funkcji aktywacji Leaky ReLU. Architektura nie zostanie jednak dokładnie omówiona, gdyż w niniejszej pracy badane będą jedynie sieci bez rozwidleń. Wszystkie wyniki uzyskane przy pomocy CQT / MEL-S oraz CNN okazały się lepsze od tych uzyskanych gdy użyto MFCC i klasyfikator GMM (ang. *gaussian mixture models*). Średnie dokładności dla 4-ro częściowej walidacji krzyżowej jakie otrzymano to kolejno: dla MelS(1024, 50%, 40) 76.23%, dla MelS(1024, 50%, 80) 76.55%, dla CQT(80, 12, 512) 80.25%, dla CQT(84, 12, 512) 78.11%, oraz dla CQT(126, 12, 512) 79.39%. Dźwięki trwały około sekundy, przetwarzano je w całości, a rozmiar macierzy własności CQT dla której osiągnięto najlepsze wyniki klasyfikacji to [80, 82].

Podobne badania do [29] [16] [38] przeprowadzono również w pracy [44] z 2019 roku. Proponowana sieć neuronowa przypominająca mniejszą wersję sieci z pracy [38] osiągnęła jednak dla zbiorów ESC-10 i ESC-50 wyniki mniej dokładne od tych już przedstawionych.

Ciekawą pracą o której warto wspomnieć jest również [43] z 2017 roku, w której badano jak pewne bardzo głębokie sieci konwolucyjne stworzone z myślą o przetwarzaniu obrazów, poradzą sobie z identyfikacją dźwięków. Na podstawie 70 mln filmów pochodzących z serwisu YouTube, stworzono zbiór uczący składający się z pojedynczych przykładów trwających po 960 ms, a których łączny czas trwania to 5.24 mln godzin. Do każdego przykładu przypisana mogła być jedna lub więcej etykiet spośród 30871 możliwych. Referencyjne etykiety do przykładów pochodzących z tego samego filmu przypisano w oparciu o metadane takie jak opis, tytuł oraz komentarze pod filmem (przykładowe etykiety to np.: „piosenka", „gitara", „strona internetowa", „sport" i „gra"). Celem było stworzenie algorytmu który potrafi określić zawartość filmu po jego ścieżce dźwiękowej. Jako reprezentację dźwięku użyto: MelS(25ms, 60%, 64) co w efekcie dało macierze własności o rozmiarach [64, 96]. W pracy nie określono użytego próbkowania. Przetestowane sieci konwolucyjne to: AlexNet [14], VGG [69], InceptionV3 [70] oraz ResNet-50 [71] przy czym autorzy pracy zawsze nieznacznie modyfikowali początkowe i końcowe warstwy sieci. Przebadano również 25 wariantów sieci DNN. Do uczenia wykorzystano algorytm Adam, a sam proces uczenia każdej sieci realizowany był przy użyciu od 10 do 40 kart graficznych i trwał od 32 do 356 godzin. Najwyższą dokładność wieloetykietowej klasyfikacji zapewniła sieć ResNet-50, jednak co ważniejsze z perspektywy niniejszej pracy, każda sieć konwolucyjna osiągnęła wyższą dokładność od dowolnego przebadanego wariantu DNN. Potwierdza to więc, iż odpowiednio wykorzystane warstwy konwolucyjne, mogę zwiększyć dokładność sieci. Więcej wniosków które pomogły by w niniejszej pracy wybrać architekturę sieci do testów niestety nie można wysnuć, gdyż badane problemy zbyt różnią się skalą. W powyższej opisanej pracy autorzy nie musieli dla przykładu wcale zapobiegać nadmiernemu

dopasowaniu, gdyż takowe ze względu na bardzo duży zbiór uczący nie występowało.

4.2 Wnioski z literatury

We wszystkich omówionych badaniach dotyczących klasyfikacji dźwięku metodami uczenia głębokiego stosowana jest podobna metodyka badawcza. Wybierany jest pewien zbiór uczący, dźwięki konwertowane są na jedną lub kilka różnych macierzy własności, a następnie zazwyczaj przy pomocy 10-cio częściowej walidacji krzyżowej wyznaczana jest dla jednej lub kilku różnych sieci neuronowych średnia dokładność klasyfikacji. Badane są oczywiście wszystkie kombinacje: wejście - architektura sieci. Ze względu na długie czasy uczenia i walidację krzyżową, autorzy prac najczęściej ograniczają się często do wyboru zaledwie jednej postaci macierzy własności i jednej sieci. By dla otrzymanych wyników był pewien punkt odniesienia, dla badanego zbioru wykonuje się również klasyfikację przy pomocy wysoko przetworzonych własności takich jak MFCC i klasycznego klasyfikatora takiego jak kNN lub las losowy. Pozwala to określić czy dla danego problemu warto używać sieci neuronowych, czy może lepiej stosować prostsze algorytmy.

Spośród wszystkich przedstawionych sieci GajhedeCNN, HuzaifahCNN i SalamonCNN wydają się być najbardziej odpowiednie do rozwiązania problemu postawionego w niniejszej pracy.

Analizę prac [2] [16] [38] [29] [34] można podsumować następującą listą spostrzeżeń i wniosków dotyczących wyboru algorytmów oraz hiperparametrów na potrzeby klasyfikacji dźwięku za pomocą uczenia głębokiego:

- Najczęściej klasyfikowane są sygnały trwające ok. 1 sekundy.
- Macierze własności dla sieci najlepiej wyznaczyć w oparciu o CQT lub MEL-S. Dla MFCC osiągana dokładność jest niższa. Najczęściej

stosowaną reprezentacją dźwięku wydaje się być MelS(1024, 50%, 60 do 128) znormalizowane do zakresu [-1, 1] lub do zakresu [0, 1]. Obiecująca wydaje się również postać CQT(80 do 126, 12, 512).

- Sieci zawierają 1,2 lub 3 cykle konwolucyjne z opcjonalnym max-poolingiem oraz 1 lub 2 ukryte warstwy gęsto połączone.
- Dla warstw konwolucyjnych i poolingowych warto stosować maski kwadratowe (o równych wymiarach) lub prawie kwadratowe.
- Gdy zbiór uczący nie jest duży (tz. ma kilkaset lub kilka tysięcy przykładów) to sieć nie powinna mieć zbyt wiele parametrów.
- Najczęściej stosowana funkcja aktywacji to ReLU.
- Nadmiernego dopasowanie najczęściej niwelowane jest przy pomocy dropoutu (ok. 50%) i regularyzacji L2 (wsp. kary ok. 0.001).
- Sieci trenowane są algorytmem Adam lub SGD bez momentum. Współczynniki nauki należą do zakresu [0.005 - 0.1], a rozmiary mini-batchy do zakresu [100 - 1000]. Gdy jedna z tych wartości jest mniejsza to druga zazwyczaj jest większa.
- Sieci uczone są przez 50 do 300 epok. Dla mniejszych zbiorów uczących liczba epok jest zazwyczaj większa.

5 Klasyfikacja dźwięków perkusyjnych za pomocą sieci neuronowych

W niniejszym rozdziale wymieniono wykorzystane narzędzia, opisano stworzony zbiór danych do testów, oraz wskazano i scharakteryzowano badane algorytmy, a następnie przedstawiono wyniki przeprowadzonych dla tych algorytmów badań.

5.1 Wykorzystane narzędzia

Podczas realizacji niniejszej pracy wykorzystano głównie język Python w wersji 3.7.7 [72] oraz dedykowane dla tego języka biblioteki. Niezbędne obliczenia wykonane zostały częściowo lokalnie przy pomocy procesora Intel i7-4790 i karty graficznej GeForce GTX 970, a częściowo przy pomocy serwisu Google Colab [73] na którym użytkownikowi zasoby przydzielane są dynamicznie. Lokalne obliczenia prowadzono wewnątrz środowiska Spyder 4 [74] z dystrybucji WinPython64-3.7.7.1 [75].

Najważniejsze spośród wykorzystanych bibliotek wraz z rolą jaką pełniły w trakcie badań to kolejno:

- Librosa [8] – wczytywanie dźwięku, przetwarzanie dźwięku oraz wyznaczanie jego postaci czasowo częstotliwościowej
- TensorFlow [6] – tworzenie i uczenie sieci neuronowych

- sklearn [76] – tworzenie i trenowanie klasycznych klasyfikatorów oraz wyznaczanie wskaźników jakości klasyfikacji
- NumPy [7] – podstawowe obliczenia matematyczne
- Matplotlib [7] – tworzenie wykresów
- Pandas [7] – zbieranie, sortowanie i przetwarzanie danych

5.2 Opis autorskiego zbioru danych

5.2.1 Zawartość zbioru

Zbiór danych przygotowany na potrzeby niniejszej pracy zawiera 7640 dźwięków perkusyjnych trwających od 20 ms do 30 s. Składa się on z nagrań

Rys. 28 Przykładowe instrumenty perkusyjne w zbiorze danych: 1. ride, 2. kotły, 3. werbel, 4. crash, 5. hi-hat, 6.bongosy , 7. kongi, 8. krowie dzwonki, 9. Stopa perkusyjna, 10. djembe, 11. trójkąt, 12. agogo, 13. dzwonki, 14. klaśnięcie. Pojedyńcze grafiki zaczerpnięto z [83].

prawdziwych instrumentów perkusyjnych (Rys. 28) oraz z syntetycznie wygenerowanych sygnałów (przykładową metodę generowania takiego sygnału opisano podrozdziale 3.3.5). Dźwięki pozyskano z różnych źródeł

takich jak: prywatne nagrania, paczki dźwięków z serwisu Splice [77], oraz paczki dźwięków tworzone i udostępniane przez użytkowników portalu Reddit [78]. Zebrane dane podzielone zostały na 10 kategorii:

- (0) „Clap" – dźwięki o brzmieniu zbliżonym do klaśnięcia
- (1) „Conga" – dźwięki o brzmieniu zbliżonym do instrumentów perkusyjnych z Afryki: np. uderzenia w bongosy oraz dejembe.
- (2) „Crash" – dźwięki o brzmieniu zbliżonym do talerza crash
- (3) „HHatC" – dźwięki o brzmieniu zbliżonym do zamkniętego talerza hi-hat (ang. *closed hi-hat*)
- (4) „HHatO" – dźwięki o brzmieniu zbliżonym do otwartego talerza hi-hat (ang. *open hi-hat*)
- (5) „Kick" – dźwięki o brzmieniu zbliżonym do stopy perkusyjnej
- (6) „Metal" – dźwięki o brzmieniu metalicznym: np. uderzenia w trójkąty, agogo, dzwonki, oraz pręty deski do prasowania
- (7) „Ride" – dźwięki o brzmieniu zbliżonym do talerza ride
- (8) „Snare" – dźwięki o brzmieniu zbliżonym do werbla
- (9) „Tom" – dźwięki o brzmieniu zbliżonym do kotła

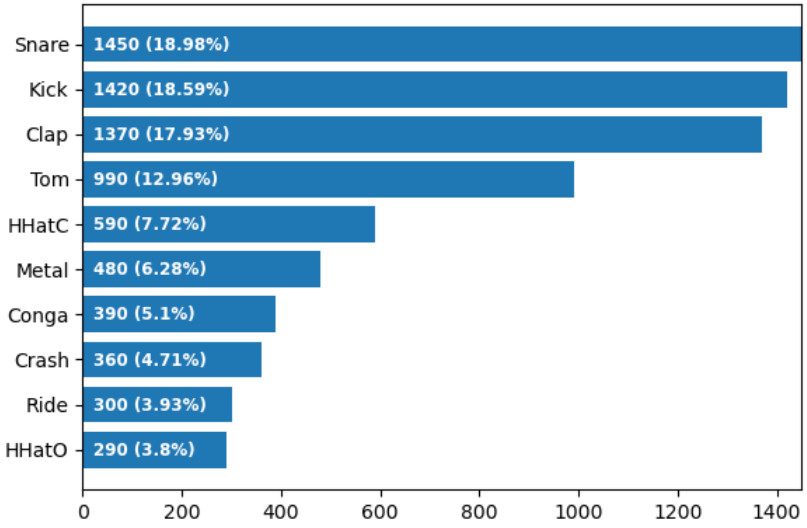
Kategorie wybrano tak, by dobrze podzielić najczęściej stosowane we współczesnej muzyce rozrywkowej dźwięki perkusyjne, ze względu na rolę jaką zazwyczaj pełnią w utworze. Podobne kategorie wyróżniono w klasycznym analogowym sekwencerze Roland TR-808 z lat 80 [24]. Pod względem liczby przykładów oraz liczby kategorii, niniejszy zbiór był najbardziej inspirowany zbiorem dźwięków miejskich UrbanSounds8k [66]. Cyfry podane w nawiasach odnoszą się do numeru indeksu danej etykiety w wektorach prawdopodobieństw zwracanych przez trenowane klasyfikatory

Dźwięki w obrębie pojedynczej kategorii czasem znacząco różnią się brzmieniem. Przykładowo kategoria „Clap" zawiera nagrania pojedynczych klaśnięć, nagrania kilkudziesięciu osób klaszczących jednocześnie, oraz

klaśnięcia wygenerowane syntetyczne z białego szumu. Przebieg amplitudy, oraz widmo amplitudowe w każdym przypadku będą się więc różnić, jednak dla człowieka dostrzeżenie podobieństw pomiędzy dźwiękami z danej kategorii nie powinno stanowić problemu.

Pomiędzy różnymi kategoriami mogą natomiast zachodzić znaczące podobieństwa. Przykładowo granice pomiędzy dźwiękami w kategoriach „Crash", „Ride" i „HHatO" wynikają wyłącznie z drobnych różnic pomiędzy rozkładem mocy w wysokich częstotliwościach. Dla człowieka osłuchanego z różnymi dźwiękami perkusyjnymi dostrzeżenie tych drobnych różnic pomiędzy nie powinno jednak stanowić problemu.

Liczebność poszczególnych kategorii przedstawiono na Rys. 29.



Rys. 29 Liczebność poszczególnych kategorii przygotowanego zbioru danych. Wartość procentowa określa jaką część całego zbioru stanowi dana kategoria.

Pliki w zbiorze danych są w formacie .FLAC, zajmują ok. 497 MB i sumarycznie trwają ok. 156 minut. Dodatkowe statystyki dotyczące zbioru zawarto w poniższych tabelach (Tabela 3, Tabela 4 i Tabela 5).

Liczba kanałów	1 kanał	2 kanały
Liczba plików	1234 (16.15%)	6406 (83.85%)

Tabela 3 Liczebność plików w zbiorze danych ze względu na liczbę kanałów.

Głębia bitowa	16 bitów	24 bity
Liczba plików	7281 (95.3%)	359 (4.7%)

Tabela 4 Liczebność plików w zbiorze danych ze względu na głębie bitową.

Próbkowanie	44.1 kHz	48 kHz	Inne
Liczba plików	7458 (97.62%)	151 (1.98%)	31 (0.41%)

Tabela 5 Liczebność plików w zbiorze danych ze względu na próbkowanie.

5.2.2 Wstępne przetwarzanie

Przed wykonaniem jakichkolwiek innych operacji, dźwięki ze zbioru danych poddawane są zawsze wstępnemu przetwarzaniu. Kolejne kroki tego przetwarzania to:

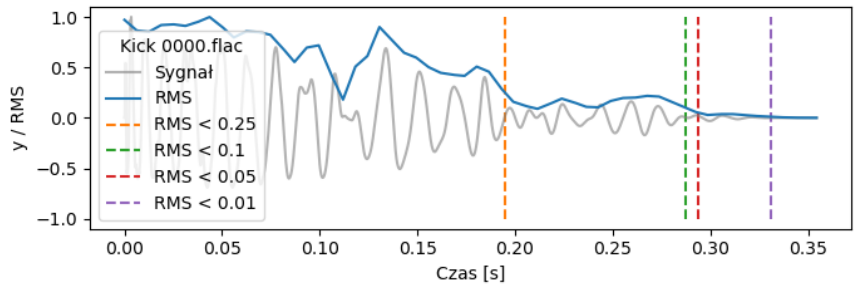
- Konwersja dźwięków dwukanałowych na jednokanałowe (nowe próbki to średnia próbek z obu kanałów)
- Konwersja dźwięków do próbkowania 44.1 Hz
- Skalowanie sygnałów tak by po normalizacji wartość próbki o maksymalnej wartości bezwzględnej wynosiła 1.
- Usunięcie szumu na początku sygnału (próbek dla których wartość bezwzględna jest mniejsza niż 0.1). Na pozostałe próbki nakładana jest obwiednia narastania amplitudy trwająca przez 10 próbek, o przebiegu takim jak funkcja cosinus w przedziale od 0 do $\pi/2$.

5.2.3 Przebieg wartości skutecznej dźwięków

Ze względu na długi czas wygasania amplitudy lub duży pogłos najdłuższe dźwięki w zbiorze danych trwają aż kilkadziesiąt sekund. Średnia długość trwania dźwięków jest jednak znacznie mniejsza. By określić jaki fragment dźwięku wystarczy do klasyfikacji, przeanalizowano jak z upływem czasu zmienia się amplituda dźwięków.

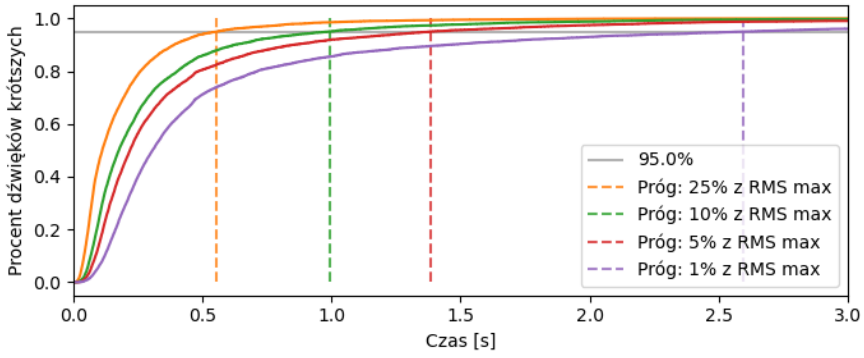
Często stosowaną miarą głośności spróbkowanych sygnałów audio jest *wartość skuteczna sygnału* (ang. *root mean square*, *RMS*). Czasowy przebieg RMS sygnału łatwo wyznaczyć w oparciu o macierz $GWM(\boldsymbol{A}_{STFT})$.

Wystarczy zsumować wszystkie wiersze, a następnie wartość w każdej kolumnie spierwiastkować i podzielić przez początkową liczbę wierszy [79]. Wynikowa macierz $\boldsymbol{RMS}$ będzie mieć jeden wiersz i tyle kolumn (chwil czasu) co miała macierz $GWM(\boldsymbol{A}_{STFT})$. Otrzymany wynik można następnie znormalizować tak by $\boldsymbol{RMS}$ przyjmowało jedynie wartości z zakresu [0,1]. Przykładowy przebieg znormalizowanego $\boldsymbol{RMS}$ pokazano na Rys. 30. Pionowe przerywane linie symbolizują punkty czasu, po których wartość RMS nie wzrasta już ponownie powyżej zadanego progu.



Rys. 30 Przykładowy, znormalizowany wynik $\boldsymbol{RMS}$ wyznaczony z STFT o parametrach $WL = 512$ i $O = 75\%$, oraz interpolowany dla wszystkich oryginalnych chwil czasu sygnału y_D.

Na Rys. 31 pokazano jak przybiegają zmiany RMS dla wszystkich dźwięków w zbiorze uczącym. Oś pionowa procentowo określa dla ilu dźwięków po danej chwili czasu znormalizowane RMS nie wzrośnie już powyżej zadanego progu. Pionowe przerywane linie wyznaczają chwilę czasu w których dla danego progu wartość na osi pionowej osiąga 95%. Jak



Rys. 31 Zmiany RMS dla wszystkich dźwięków w zbiorze uczącym. RMS wyznaczono jak na Rys. 30.

można zauważyć, dla ponad 95% dźwięków w zbiorze uczącym, RMS na stałe opada poniżej 10% maksymalnej wartości już w pierwszej sekundzie. Na podstawie tego można więc założyć, iż wyłącznie w oparciu o analizę dynamicznych zmian widma akustycznego w pierwszej sekundzie, powinno się dać z dużą dokładnością rozróżnić różne typy dźwięków.

Doświadczenia empiryczne zdają się potwierdzać powyższe założenia. Słuchając jednosekundowych fragmentów człowiekowi dość łatwo określić do jakiej kategorii dany dźwięk należy

5.2.4 Podział zbioru na części i augmentacja

Zbiór danych został losowo podzielony na dziesięć równych części zawierających po 764 przykładów, tak by dźwięki z każdej kategorii zostały równo rozmieszczone pomiędzy częściami.

Każdą część niezależnie zbalansowano za pomocą metody augmentacji opisanej podrozdziale 3.5.5. Dla najbardziej licznej kategorii „Snare" na każdą część przypada dokładnie 145 przykładów, więc z dźwięków w pozostałych kategoriach wygenerowano tyle nowych przykładów, by finalnie w każdej kategorii, w każdej części było ich tyle samo. Informację o tym ile nowych przykładów z pojedynczego dźwięku maksymalnie mogło zostać wygenerowanych zawiera Tabela 6. Dla dźwięków w kategorii „Snare" nie były generowane żadne nowe przykłady.

Clap	Conga	Crash	HHatC	HHatO	Kick	Metal	Ride	Tom
1	3	4	2	4	1	3	4	1

Tabela 6 Maksymalna liczba augmentacji przykładu w danej kategorii..

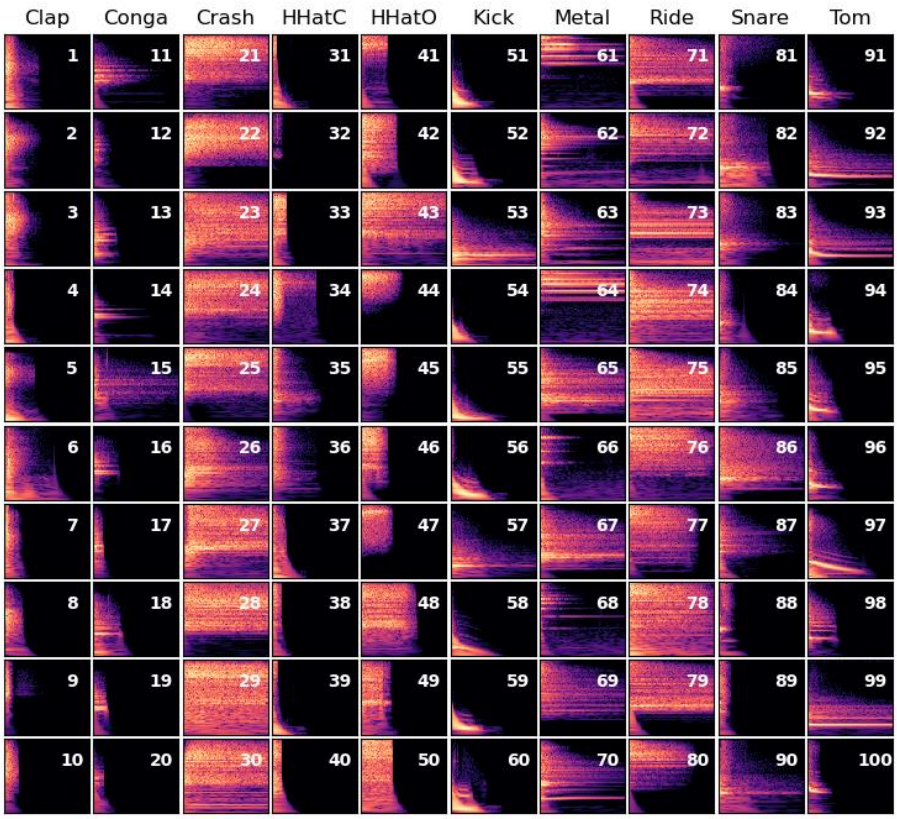
Zmiana wysokości dźwięku w półtonach Δst jest zawsze losowana. Dla przykładów z kategorii: „HHatC" i „HHatO" brane są maksymalnie cztery różne wartości ze zbioru $\{-1, -0.5, 0.5, 1\}$, a dla dźwięków z pozostałych kategorii ze zbioru $\{-2, -1, 1, 2\}$. Dla kategorii „HHatC" i „HHatO" losowane są mniejsze wartości, gdyż dźwięki pomiędzy tymi kategoriami różnią się głównie długością. Dla zbyt dużych/małych wartości Δst dźwięki

mogły by zostać zbyt skrócone/wydłużone, co utrudniło by klasyfikatorowi zauważenie właściwych różnic pomiędzy tymi kategoriami.

Po augmentacji sumarycznie wszystkich przykładów jest 14500, a na każdą część przypada ich dokładnie 1450. Podczas walidacji krzyżowej na zbiór uczący przypada ich więc 13050, natomiast na zbiór testowy 1450.

5.2.5 Przykładowe dźwięki w zbiorze

Na Rys. 32 graficznie przedstawiono po dziesięć losowych dźwięków z każdej kategorii. Do wyznaczenia spektrogramów użyto algorytmu CQT skonfigurowanego jak podano podrozdziale 5.3.



Rys. 32 Przykładowe dźwięki w zbiorze uczącym.

Zgodnie z tym co powiedziano wcześniej, na powyższym zestawieniu można dostrzec cechy charakterystyczne dźwięków dla każdej kategorii, podobieństwa pomiędzy dźwiękami z różnych kategorii (np. 27-43-76 oraz

18-95), oraz różnice pomiędzy dźwiękami w obrębie jednej kategorii (np. 51-53 oraz 91-99).

5.3 Wybrane metody konwersji dźwięków na macierze własności

Do wyznaczania macierzy własności podawanych następnie na wejście klasyfikatorów, wykorzystano następujące reprezentacje dźwięku: MEL-S, CQT, MFCC oraz CQCC. Jest to wybór poparty wnioskami z literatury opisanej w rozdziale 4.

Wybraną konfigurację algorytmów MEL-S i CQT opisuje Tabela 7 i Tabela 8. W obu przypadkach wynikowe macierze mają po 108 wierszy, maksymalna częstotliwość prążka to ok. 16.74 kHz, a skok okna to 512 próbek (11.61 ms). Główną różnicą pomiędzy tymi dwoma reprezentacjami dźwięku, jest toteż inne rozłożenie prążków w dziedzinie częstotliwości. Zarówno MEL-S jak i CQT przeliczane są oczywiście na GWM_{log}. W otrzymanych wynikach pozostawianych jest wyłącznie 86 pierwszych kolumn, dzięki czemu spektrogramy opisują tylko pierwszą sekundę dźwięku (sygnały trwające krócej niż sekundę przed wyznaczeniem postaci czasowo-częstotliwościowej zostały prawostronnie dopełniane zerami).

Okno	HL	BPO	N_{CQT}	f_{CQT0}
Hanninga	512	12	108	$2^{\frac{-45}{12}} \cdot 440\ Hz \approx 32.70\ Hz$

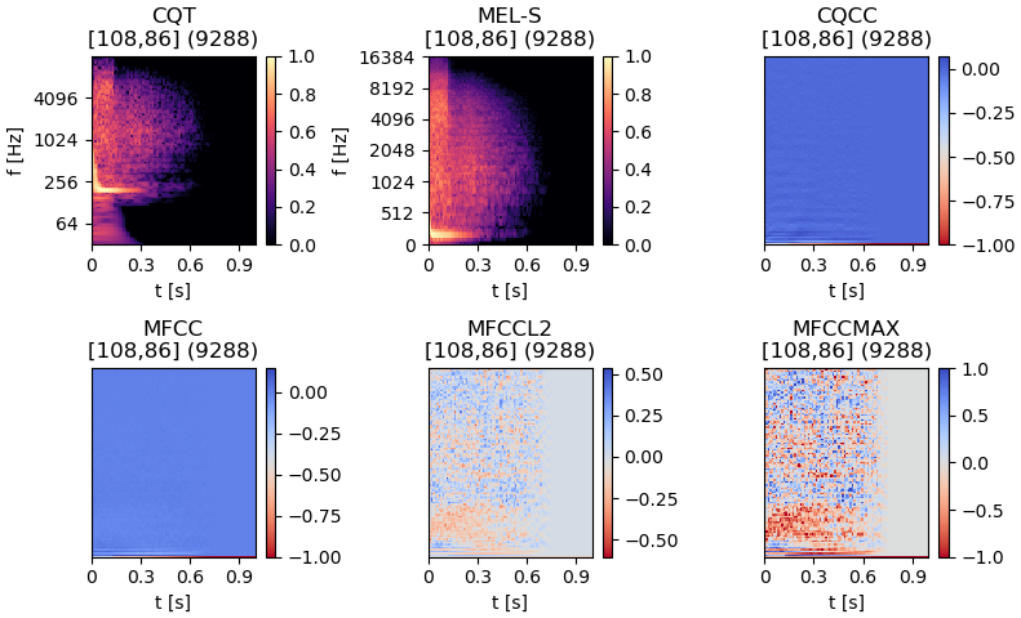
Tabela 7 Parametry użyte podczas generowaniu CQT.

Okno	HL	WL	N_{Mel}	f_{min}	f_{max}
Hanninga	512	1024	108	0 Hz	$2^{\frac{-45+108}{12}} \cdot 440\ Hz$

Tabela 8 Parametry użyte podczas generowania MEL-S.
f_{min} i f_{max} to parametry określające górną i dolną granicę częstotliwości prążków dla macierzy filtrów trójkątnych.

Na podstawie MEL-S i CQT wyznaczano następnie MFCC oraz CQCC. W obu przypadkach liczba wierszy nie została w żaden sposób ograniczona, by podczas badań możliwe było sprawdzenie jak różna liczba wierszy wpływa na dokładność klasyfikacji.

MEL-S i CQT znormalizowano do zakresu [0,1] (skalowanie i przesunięcie), natomiast dla MFCC i CQCC do zakresu [0,1] wyłącznie przeskalowano. Dla MFCC wygenerowano również dwa alternatywne wyniki: MFCCMAX ze skalowaniem do zakresu [0,1] niezależnym dla każdego wiersza, oraz MFCCL2 z takim skalowaniem dla każdego wiersza by jego druga norma wynosiła 1.



Rys. 33 Reprezentacje dźwięku badane w niniejszej pracy wyznaczone dla dźwięku z kategorii „Snare”.

Wszystkie znormalizowane macierze własności testowane w niniejszej pracy przedstawiono wraz z ich rozmiarami na Rys. 33. Sumaryczna liczba własności w każdej macierzy to $108 \cdot 86 = 9288$.

5.4 Wybrane algorytmy klasyfikacji

5.4.1 Jednokierunkowe sieci neuronowe

W oparciu o wnioski z literatury opisanej w rozdziale 4, do testów wybrano cztery główne warianty sieci neuronowych:

- Sieci bazujące na GajhedeCNN z pracy [2]
- Sieci bazujące na HuzaifahCNN z pracy [29]
- Sieci bazujące na SalamonCNN z pracy [38]
- Proste sieci DNN z taką samą liczbą neuronów w każdej warstwie analogiczne do sieci testowanych w pracy [43]

Każdy wariant został przebadany dla różnych macierzy własności, a czasem testowano nawet różne zestawy hiperparametrów sieci, dlatego dalsze szczegóły dotyczące architektury zostaną podane już bezpośrednio w trakcie omawiania wyników. Do uczenia sieci neuronowych wybierano zawsze jeden lub dwa z poniższych sposobów:

- Algorytmem SGD bez współczynnika momentum, ze stratą logarytmiczną i współczynnikiem nauki 0.01, w mini-seriach po 100 przykładów przez maksymalnie 250 epok
- Algorytmem ADAM ze stratą logarytmiczną, współczynnikiem nauki 0.001, współczynnikami $\beta 1 = 0.9$ i $\beta 2 = 0.999$, w mini-seriach po 100 przykładów przez maksymalnie 150 epok

Jedynymi odstępstwami od powyższej reguły są sieci typu DNN opisane w podrozdziale 5.6.2 dla których nie zastosowano żadnej metody zapobiegania nadmiernemu dopasowaniu, oraz sieci SalamonCNN z ostatniej serii w podrozdziale 5.6.6. Pierwsze uczono algorytmem ADAM przez 50 epok, a drugie przez 300 epok.

Algorytm SGD przetestowano wyłącznie dla sieci typu SalamonCNN gdyż to właśnie tego algorytmu używali jej autorzy. Dla innych sieci

stosowany był zawsze algorytm ADAM. Sumarycznie na systemie lokalnym i w usłudze Google Colab by wytrenować wszystkie sieci neuronowe poświęcono ok. 71 godzin czasu obliczeniowego

5.4.2 Klasyfikator k najbliższych sąsiadów

By dla wyników otrzymywanych przy pomocy sieci neuronowych był znany pewien punkt odniesienia, do testów wybrano również klasyfikator kNN, który w pracy [5] i [2] zapewnił wysoką dokładność klasyfikacji.

Działanie klasyfikatora kNN nie zostanie dokładnie opisane gdyż nie stanowi to meritum niniejszej pracy, jednak klasyfikacja ta sprowadza się do: konwersji macierzy własności na wielowymiarowe punkty, znalezienia k punktów w zbiorze uczącym leżących najbliżej punktu klasyfikowanego (k jego najbliższych sąsiadów), oraz przydzielenia klasyfikowanego punktu do kategorii do której należało najwięcej znalezionych sąsiadów [80]. Dla hiperparametrów $k = 1$ będzie to kategoria najbliższego sąsiada. Proces trenowania klasyfikatora kNN polega jedynie na zapisaniu danych ze zbioru uczącego.

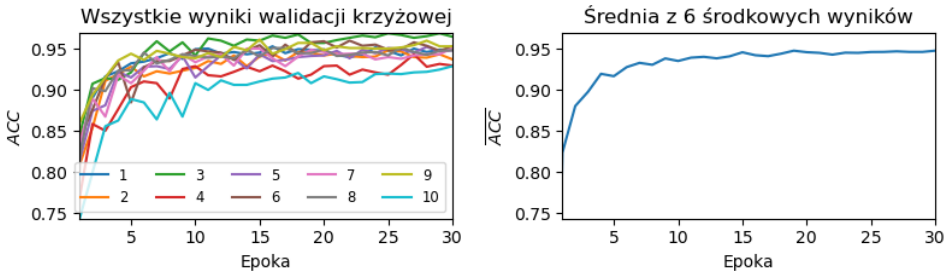
5.5 Metodyka badawcza

Celem niniejszej pracy było sprawdzenie czy do klasyfikacji dźwięków perkusyjnych warto wykorzystać sieci neuronowe, oraz ocenienie jakie macierze własności i architektura sieci sprawdzą się najlepiej. Metodyka badawcza została więc skonstruowana tak, by możliwe było badanie jak różne hiperparametry wpływają na dokładność.

5.5.1 Trenowanie i walidacja modeli

Każdy badany model (klasyfikator dla wybranych hiperparametrów i wybranej macierzy własności) trenowany i testowany był dziesięciokrotnie, zawsze za pomocy innej pary zbiorów uczących i testowych (tworzonych jak opisano w podrozdziale 3.6). Po zakończeniu każdej epoki uczenia,

wyznaczano i zapisywano następujące statystyki: dokładność klasyfikacji ACC dla zbioru testowego i zbioru uczącego, wartość funkcji straty ($LOSS$) dla zbioru uczącego i zbioru testowego, oraz czasy poświęcone na uczenie modelu i wyznaczenie predykcji. Dokładność i wartość funkcji straty dla zbioru testowego liczono w oparciu o predykcje generowane po zakończeniu epoki za pomocą modelu w którym wyłączono dropout i regularyzację, natomiast dokładność i wartość funkcji straty dla zbioru uczącego, była średnią ruchomą tych z statystyk wyznaczonych dla każdej mini-serii nauki, gdy w modelu aktywny był dropout i regularyzacja [81]. By możliwe było wyznaczenie innych statystyk, zawsze zapisywano również otrzymane wektory prawdopodobieństw dla zbioru testowego.



Rys. 34 Przebieg uśrednionej dokładności klasyfikacji zbioru testowego dla modelu SalamonCqtCNN omówionego w podrozdziale 5.7.

Sumarycznie dla każdej epoki, każdego badanego modelu, zebrano po dziesięć odrębnych zestawów statystyk (jeden dla każdego zbioru uczącego). Zestawy te następnie uśredniono, po wcześniejszym odrzuceniu dwóch zestawów statystyk w których dokładność klasyfikacji dla zbioru testowego w danej epoce była najwyższa, oraz dwóch zestawów w których dokładność ta była najniższa. Otrzymane statystyki uśrednione oznaczane są dalej jako $\overline{ACC}$ i $\overline{LOSS}$. Na Rys. 34 porównano przebiegi dokładności klasyfikacji ACC zbiorów testowych, z przebiegiem dokładności uśrednionej $\overline{ACC}$. Jak można zauważyć, na wykresie z lewej strony zależność pomiędzy dokładnością klasyfikacji zbioru testowego, a liczbą epok przez jaką uczony był model przy pomocy danego zbioru uczącego jest dość chaotyczna. Wynika to oczywiście z niedeterministycznych aspektów uczenia sieci neuronowych takich jak: losowa inicjalizacja wag, losowanie mini-serii, oraz losowa

dezaktywacja neuronów gdy włączony jest dropout. Wektor parametrów czasem może zostać również tymczasowo przesuwany w kierunku dla którego dokładność klasyfikacji danego zbioru testowego okaże się być bardzo wysoka lub bardzo niska. Przebieg dokładności uśrednionej na wykresie znajdującym się z prawej strony, dużo lepiej oddaje ogólny trend zmian. Szczególnie usunięcie części najlepszych i najgorszych wyników, pozwala zredukować wpływ wyjątkowo szczęśliwych lub niefortunnych zmian parametrów trenowanych sieci na ogólny wynik modelu w danej epoce. Uśrednienie statystyk uwydatnia więc niejako wpływ wartości hiperparametrów na osiąganą dokładność.

Po zakończeniu trenowania modelu za pomocą wszystkich zbiorów uczących, oraz po wyznaczeniu uśrednionych statystyk dla wszystkich epok, jako finalny zestaw liczb reprezentujący działalnie danego modelu wybierany jest zestaw statystyk uśrednionych z epoki dla której uśredniona dokładność klasyfikacji $\overline{ACC}$ zbioru testowego okazała się największa. Gdy więc w podrozdziale 5.6 w tekście lub na wykresie będzie podawana dokładność modelu, będzie to uśredniona dokładność klasyfikacji zbioru testowego dla najlepszej epoki. Wyniki podawane są po zaokrągleniu do dwóch cyfr po przecinku.

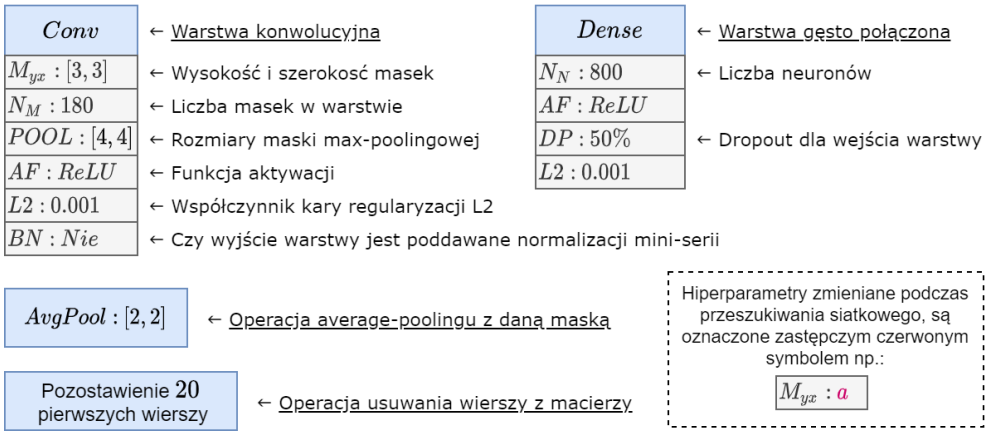
W podrozdziale 5.7 i 5.8 w procesie uśredniania pomijany będzie już etap odrzucania dwóch najlepszych i dwóch najgorszych zestawów statystyk. Podawane czułości tudzież specyficzności dla danej etykiety są więc liczone w oparciu o wyniki predykcji ze wszystkich dziesięciu zbiorów uczących, a w tablicach pomyłek prezentowane będą pomyłki dla wszystkich 14500 dźwięków z dziesięciu 1450 dźwiękowych zbiorów testowych.

5.5.2 Dobór hiperparametrów modeli

Hiperparametry klasyfikatorów wymienionych w podrozdziale 5.4 dobierano zgodnie z ideą *przeszukiwania siatkowego* (ang. *grid search*), tj. dla części hiperparametrów określono zawsze od kilku do kilkudziesięciu możliwych wartości, a następnie badano wszystkie możliwe kombinacje. Przykładowo, jeżeli w pewnym modelu więcej niż jedną wartość mogą

przyjąć hiperparametry $a = \{1\ lub\ 2\}$ i $b = \{3\ lub\ 4\}$, to przebadane zostaną kombinacje: $(1,3), (1,4), (2,3), (2,4)$. Postać wejściowej macierzy własności również traktowana będzie jako jeden z wielu hiperparametrów klasyfikatora.

Architekturę trenowanych modeli, oraz to które hiperparametry mogą przyjmować różne wartości zilustrowano w podrozdziałach 5.6.1 - 5.6.6 za pomocą schematów blokowych. Przyjęte oznaczenia wyjaśniono na Rys. 35. Zerowa regularyzacja L2 lub zerowy dropout oznacza oczywiście brak regularyzacji/dropoutu. Regularyzację dodawano do macierzy wag i masek lecz nie do przesunięć.



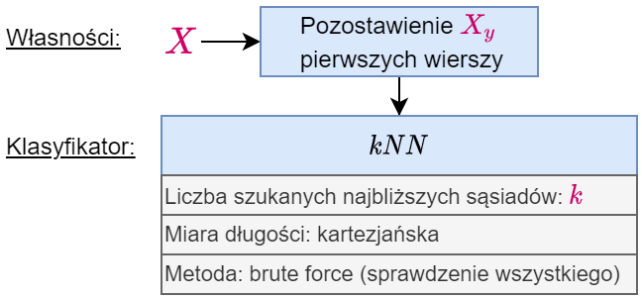
Rys. 35 Objaśnienie symboli pojawiających się na schematach blokowych.

To jakie wartości mogą przyjmować dane hiperparametry podczas przeszukiwania siatkowego opisano za pomocą tabel. Kolejne możliwe wartości będą w komórkach rozdzielane przecinkami. By podczas opisu móc w łatwy sposób odnieść się do pewnej wybranej partii przebadanych modeli, wykonywane przeszukiwania siatkowe podzielono na serie i ponumerowano.

5.6 Wpływ hiperparametrów na dokładność różnych modeli

5.6.1 Modele typu kNN

Jako pierwsze przedstawione zostaną wyniki otrzymane dla klasyfikatora kNN, gdyż stanowią one punkt odniesienia dla pozostałych wyników. Schemat blokowy klasyfikatora przedstawiono na Rys. 36. Możliwe wartości dla zmienianych hiperparametrów zawiera Tabela 9. Przebadano 340 różnych modeli tego typu.



Rys. 36 Schemat blokowy klasyfikatora kNN.

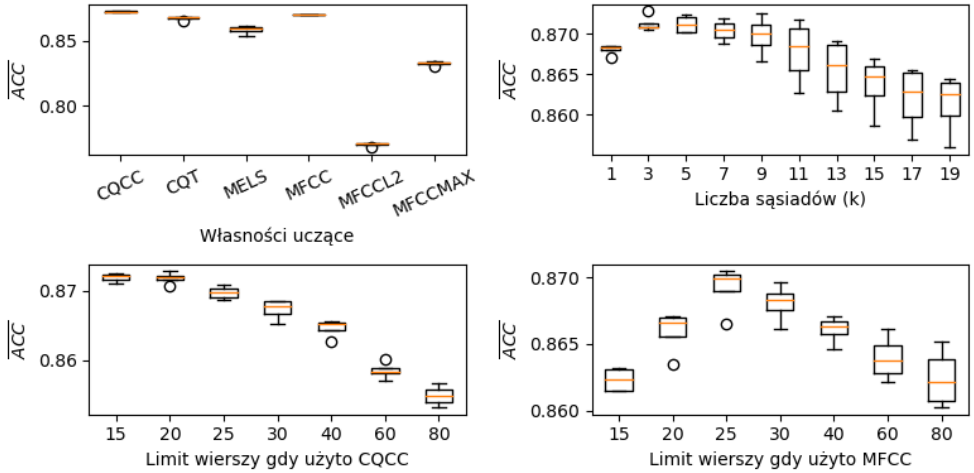
Seria	X (własności uczące)	X_y (limit wierszy)	k
1a	MELS, CQT	Wszystkie	1, 3, … , 21
2b	MFCC, CQCC, MFCCMAX, MFCCL2	15, 20, 25, 30, 40, 60, 80	1, 3, … , 21

Tabela 9 Testowane wartości hiperparametrów dla klasyfikatora kNN.

Wpływ hiperparametrów na dokładność zilustrowano na Rys. 37. Każde pudełko na wykresach opisuje dokładnie cztery wartości $\overline{ACC}$ dla czterech najlepszych modeli w których wybrany hiperparametr przyjmuje wartość wskazaną na osi poziomej. Kolejne wykresy tego typu tworzone są według tej samej reguły.

W przypadku klasyfikatora kNN najlepsze rezultaty zapewniły wysoce przetworzone macierze własności takie jak CQCC i MFCC, co potwierdza teorię omówioną podrozdziale 2.2.1. Warto jednak zwrócić uwagę na

zauważalnie niższą dokładność dla MFCCL2 i MFCCMAX. Normalizacja każdego wiersza osobno okazała się nie być dobrym rozwiązaniem. Podobne rezultaty we wstępnych testach uzyskano również dla sieci neuronowych, więc w dalszych badaniach postacie MFCCL2 i MFCCMAX nie były już wykorzystywane.



Rys. 37 Wpływ hiperparametrów na dokładność klasyfikatora kNN.

CQCC zazwyczaj okazywało się lepszą reprezentacją dźwięków perkusyjnych niż MFCC. Zarówno dla CQCC jak i MFCC najlepsze wyniki otrzymano gdy macierz własności ograniczono do ok. 20 - 40 wierszy co potwierdza wnioski z pracy [4].

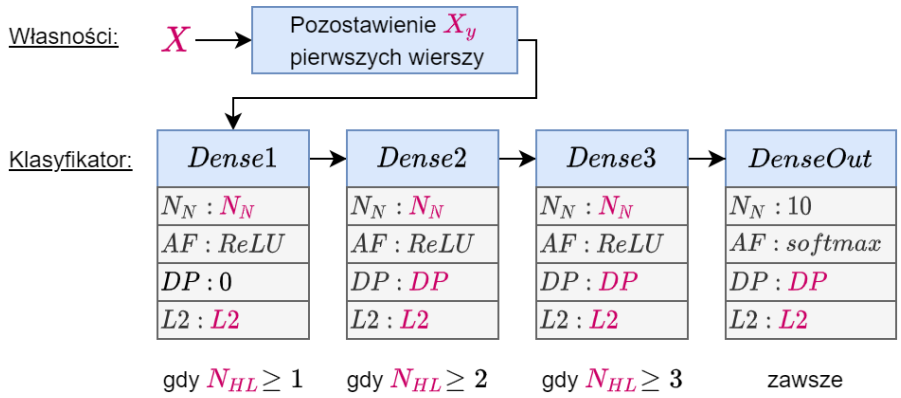
Najlepszy wynik dla klasyfikatora kNN uzyskano dla $X = CQCC$, $X_y = 20$ i $k = 3$. Uśredniona dokładność była wówczas równa $\overline{ACC} = 87.28\%$.

5.6.2 Modele typu DNN

Schemat blokowy modeli typu DNN przedstawiono na Rys. 38. Możliwe wartości dla zmienianych hiperparametrów zawiera Tabela 10. Przebadano 114 różnych modeli tego typu, a na trenowanie wszystkich 1140 sieci przeznaczono ok. 9 godzin.

Przy pomocy pierwszej serii modeli sprawdzono jak na dokładność klasyfikacji wpływa liczba warstw, liczba neuronów we warstwach oraz postać wejściowej macierzy własności. Otrzymano dość zaskakujące wyniki,

gdyż najlepszymi reprezentacjami dźwięku podobnie jak w przypadku klasyfikatora kNN okazały się być CQCC i MFCC (Rys. 40). Jest to rezultat sprzeczny z informacjami jakie można znaleźć w literaturze. Zdolność analizy danych w badanych prostych sieci gęsto połączonych okazałą się najwidoczniej niewystarczająca, skoro to uprzednio przetworzone własności pozwalają otrzymać wyższą dokładność.

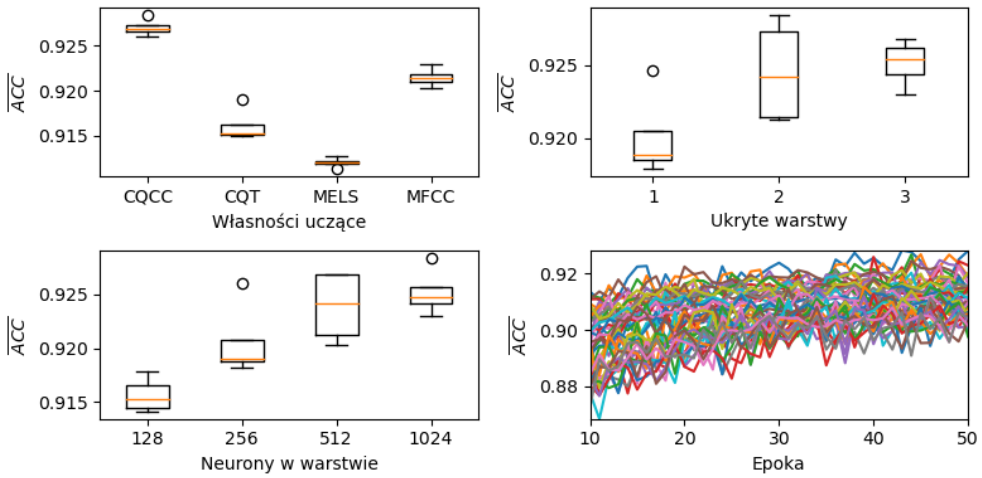


Rys. 38 Schemat blokowy modeli typu DNN.
N_{HL} to hiperparametr określający liczbę ukrytych warstw sieci.

Seria	X	X_y	N_{HL}	N_N	$L2$	DP
1a	CQT, MELS	Wszystkie	1, 2, 3	128, 256, 512, 1024	0	0%
1b	CQCC, MFCC	30	1, 2, 3	128, 256, 512, 1024	0	0%
2	CQCC, MFCC	10, 15, 20, 25, 40, 60, 80	2	256, 512, 1024	0	0%
3	CQCC, MFCC	20, 40	2	512, 1024	0, 0.001	0, 50%

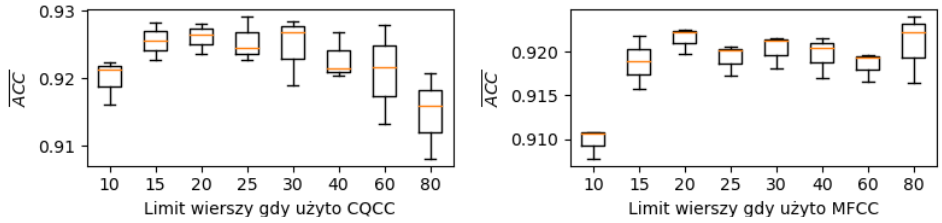
Tabela 10 Testowane wartości hiperparametrów dla modeli typu DNN.

Dla sieci o większej liczbie warstw/neuronów uzyskano zazwyczaj wyższą dokładność klasyfikacji. Na podstawie zebranych wyników dość dobrym wyborem dla badanego problemu wydają się sieci o 2 warstwach ukrytych i co najmniej 512 neuronach w każdej warstwie (Rys. 40).



Rys. 40 Wpływ hiperparametrów na dokładność modeli typu DNN. Na wykresie dokładności od epoki narysowano przebiegi dla wszystkich modeli z serii 1a i 1b.

Jako, że własności CQCC i MFCC okazały się najlepsze, to przy pomocy drugiej serii modeli sprawdzono jaki jest związek pomiędzy liczbą wierszy w macierzy własności a dokładnością. Tym razem najlepsze wyniki również otrzymywano zazwyczaj dla ok. 20 - 40 wierszy (Rys. 39).

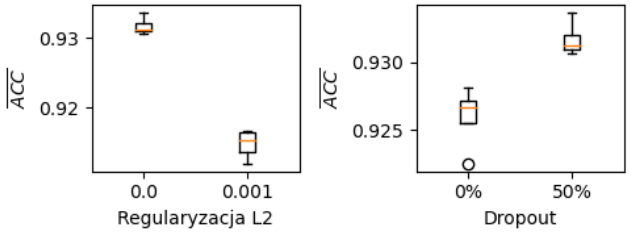


Rys. 39 Wpływ rozmiaru CQCC i MFCC na dokładność w modelach typu DNN.

Ostatecznie przy pomocy trzeciej serii modeli sprawdzono jeszcze jak na dokładność klasyfikacji najbardziej obiecujących modeli, wpłynie dodanie regularyzacji L2 i/lub dropoutu. Najlepsze okazały się być modele z dropoutem lecz bez regularyzacji (Rys. 41).

By móc wyniki dla DNN uczciwie porównać z wynikami dla sieci neuronowych omówionych w kolejnych podrozdziałach, wszystkie modele z trzeciej serii poddano dalszemu trenowaniu tak by sumarycznie każdy model był uczony algorytmem ADAM przez maksymalnie 150 epok. Najlepszy model typu DNN uzyskano dla: $X = CQCC$, $X_y = 20$, $N_{HL} = 2$, $N_N = 1024$, $L2 = 0$, $DP = 50\%$ i 122 epoki. Posiada on ok. 3 miliony

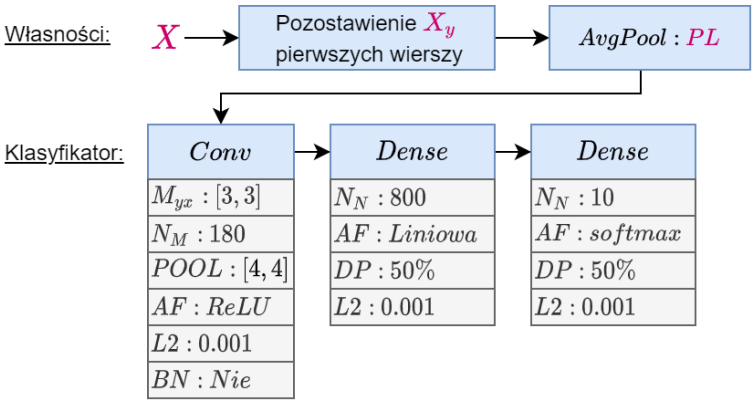
parametrów. Uśredniona dokładność otrzymana dla zbioru testowego jest równa $\overline{ACC} = 93.89\%$, a dla zbioru uczącego $\overline{TrainACC} = 97.87\%$.



Rys. 41 Wpływ dropoutu i regularyzacji L2 na dokładność w modelach typu DNN.

5.6.3 Modele typu HuzaifahCNN

Schemat blokowy modeli typu HuzaifahCNN przedstawiono na Rys. 42. Możliwe wartości dla zmienianych hiperparametrów zawiera Tabela 11. Przebadano 6 różnych modeli tego typu, a na trenowanie wszystkich 60 sieci przeznaczono ok. 9 godzin. Na wejściu dodano average-pooling by liczba własności w macierzy danych, była podobna jak w oryginalnej pracy z której zaczerpnięto HuzaifahCNN.

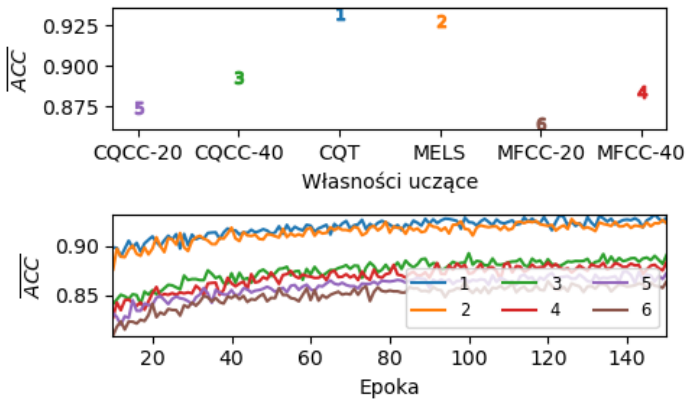


Rys. 42 Schemat blokowy modeli typu HuzaifahCNN.

Seria	X	X_y	PL
1a	CQT, MELS	Wszystkie	[1,4]
1b	CQCC, MFCC	20, 40	[1,2]

Tabela 11 Testowane wartości hiperparametrów dla modeli typu HuzaifahCNN.

Ze względu na dość długie czasy uczenia sieci konwolucyjnych, liczba modeli badanych w tym i kolejnych podrozdziałach jest mniejsza niż w podrozdziałach poprzednich. Wyniki na wykresach reprezentowane będą więc za pomocą pojedynczych punktów, gdzie każdy punkt to dokładność dla innego modelu (Rys. 43). Punkty są numerowane według dokładności w obrębie danej serii (punkt „1" to model o najwyższej dokładności w serii).



Rys. 43 Wpływ macierzy własności na dokładność modeli typu HuzaifahCNN. Numery oznaczają liczbę pozostawionych wierszy.

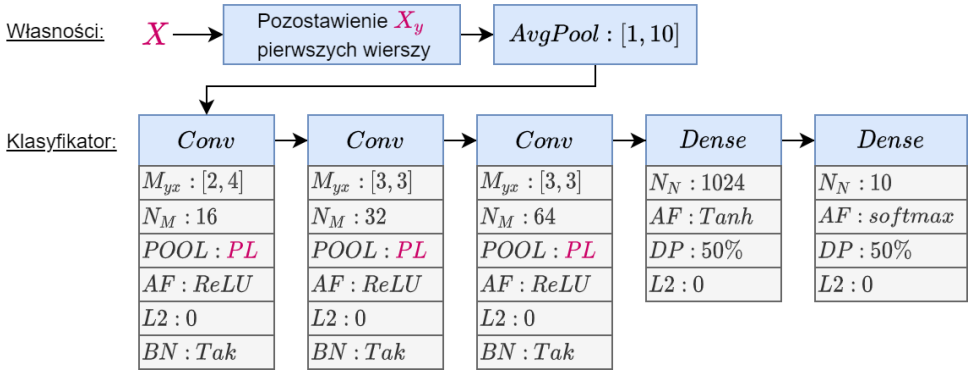
Dla modeli typu HuzaifahCNN wyniki okazały się być dość zgodne z oczekiwaniami. Nieprzetworzone spektrogramy CQT i MELS pozwoliły osiągnąć wyższą dokładność niż ich przetworzone odpowiedniki.

Najlepszy model typu HuzaifahCNN uzyskano dla $X = CQT$ i 147 epoki. Posiada on ok. 15 milionów parametrów. Uśredniona dokładność otrzymana dla zbioru testowego jest równa $\overline{ACC} = 93.26\%$, a dla zbioru uczącego $\overline{TrainACC} = 92.60\%$. Stosunkowo niska dokładność dla zbioru uczącego sugeruje, iż istnieje duży potencjał na poprawę otrzymanego wyniku w kolejnych epokach. Tak duża liczba parametrów wydaje się jednak być nadmiarowa dla problemu klasyfikacji dźwięków perkusyjnych i może potencjalnie prowadzić do zapamiętywania przykładów.

5.6.4 Modele typu GajhedeCNN

Schemat blokowy modeli typu GajhedeCNN przedstawiono na Rys. 44. Możliwe wartości dla zmienianych hiperparametrów zawiera Tabela 12.

Przebadano 6 różnych modeli tego typu, a na trenowanie wszystkich 60 sieci przeznaczono ok. godziny. Na wejściu dodano average-pooling gdyż w pracy z której zaczerpnięto GajhedeCNN macierze własności miały zaledwie 8 kolumn (chwil czasu). Maski max-poolingowe zmieniano tak, by wyjście ostatniej warstwy konwolucyjnej zawsze miało podobne rozmiary.



Rys. 44 Schemat blokowy modeli typu GajhedeCNN.

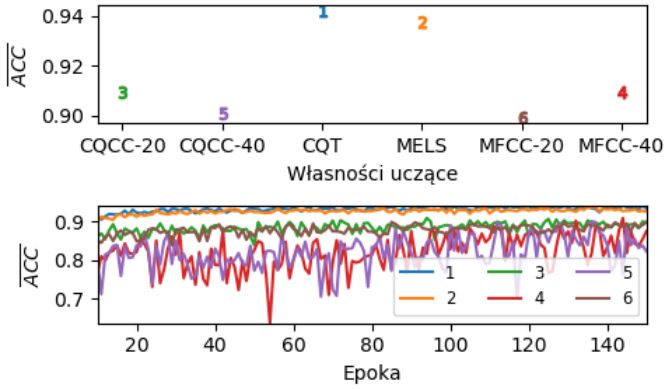
Seria	X	X_y	PL
1a	CQT, MELS	Wszystkie	[5,1]
1b	CQCC, MFCC	40	[3,1]
1c	CQCC, MFCC	20	[2,1]

Tabela 12 Testowane wartości hiperparametrów dla modeli typu GajhedeCNN.

Uzyskano wyniki zbliżone do tych otrzymanych dla modeli typu HuzaifahCNN, lecz uśrednione dokładności były wyższe (Rys. 45). Może to wynikać zarówno z mniejszej liczby parametrów trenowanych (przez co 150 epok wystarczyło by sieć osiągnęła wysoką dokładność) jak i bardziej odpowiedniej architektury dla postawionego problemu. Warto przypomnieć, iż GajhedeCNN jest jedyną spośród sieci neuronowych zaczerpniętych z literatury, która była już badana dla dźwięków perkusyjnych (z tym że podzielonych na zaledwie trzy kategorie i dla prostego zbioru danych).

Najlepszy model typu GajhedeCNN uzyskano dla $X = CQT$ i 144 epoki. Posiada on 42714 parametrów (najmniej spośród wszystkich przebadanych

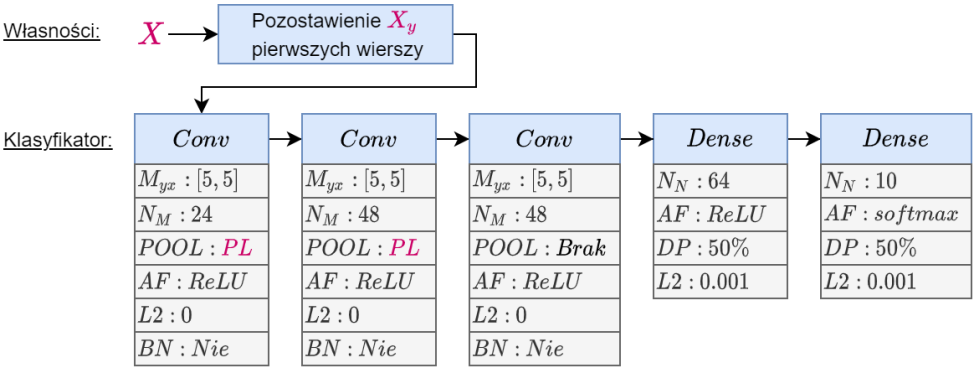
sieci). Uśredniona dokładność otrzymana dla zbioru testowego jest równa $\overline{ACC} = 94.22\%$, a dla zbioru uczącego $\overline{TrainACC} = 98.27\%$.



Rys. 45 Wpływ macierzy własności na dokładność modeli typu GajhedeCNN.

5.6.5 Modele typu SalamonCNN

Schemat blokowy modeli typu SalamonCNN przedstawiono na Rys. 46. Możliwe wartości dla zmienianych hiperparametrów zawiera Tabela 12. Przebadano 12 różnych modeli tego typu, a na trenowanie wszystkich sieci przeznaczono ok. 11 godzin. Maski max-poolingowe zmieniano tak, by wyjście ostatniej warstwy konwolucyjnej zawsze miało podobne rozmiary.



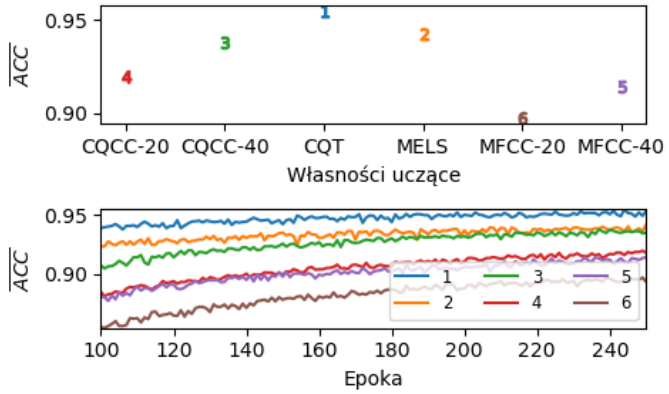
Rys. 46 Schemat blokowy modeli typu SalamonCNN.

Seria	X	X_y	PL	Metoda uczenia
1a	CQT, MELS	Wszystkie	[2,3]	SGD
1b	CQCC, MFCC	40, 20	[1,3]	SGD

2a	CQT, MELS	Wszystkie	[2,1]	ADAM
2b	CQCC, MFCC	40, 20	[1,3]	ADAM

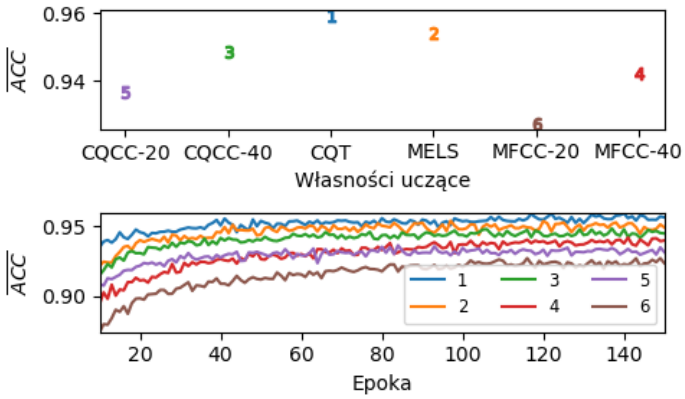
Tabela 13 Testowane wartości hiperparametrów dla modeli typu SalamonCNN.

W pierwszym podejściu (Rys. 47) próbowano trenować modele za pomocą metody SGD, gdyż to właśnie ona została wykorzystana przez autorów SalamonCNN w oryginalnej pracy.



Rys. 47 Wyniki dla modeli typu SalamonCNN trenowanych metodą SGD.

Jak jednak można zauważyć, metoda ADAM pozwoliła uzyskać jednak lepsze wyniki przy niemal dwukrotnie krótszym czasie trenowania (Rys. 48).



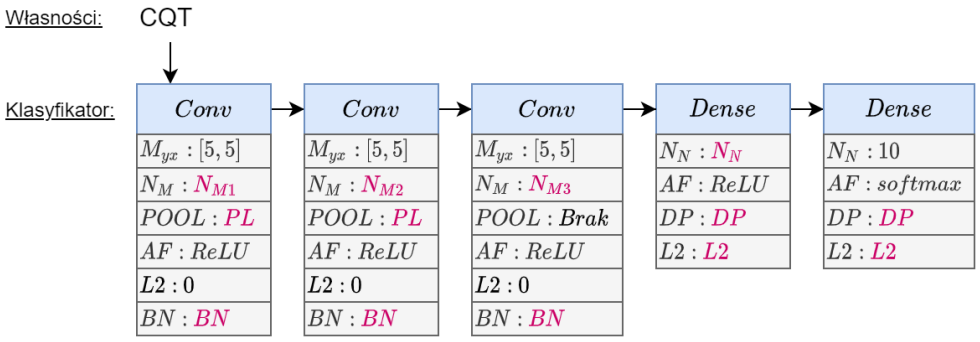
Rys. 48 Wyniki dla modeli typu SalamonCNN trenowanych metodą ADAM.

Podobnie jak w przypadku modeli typu HuzaifahCNN i GajhedeCNN, dla modeli typu SalamonCNN lepszymi reprezentacjami dźwięku okazały się być nieprzetworzone spektrogramy CQT i MELS.

Najlepszy model typu SalamonCNN uzyskano dla $X = CQT$ i 140 epoki uczenia metodą ADAM. Posiada on 272154 parametrów. Uśredniona dokładność otrzymana dla zbioru testowego jest równa $\overline{ACC} = 95.94\%$, a dla zbioru uczącego $\overline{TrainACC} = 98.45\%$.

5.6.6 Dodatkowe modele typu SalamonCNN

Omówione do tej pory modele typu HuzaifahCNN, GajhedeCNN oraz SalamonCNN różniły się od pierwowzorów głównie użytą reprezentacją dźwięku i liczbą neuronów w ostatniej warstwie. By architekturę modeli lepiej dostosować do różnych macierzy własności, czasem zmieniano również rozmiary masek poolingowych. Głównym celem było jednak najwierniejsze oddanie działania oryginalnych sieci. Dla modeli typu SalamonCNN jako, że zapewniły najwyższą dokładność, przeprowadzono dodatkowe szersze badania. Informacje o tym jakie modyfikacje sieci SalamonCNN przebadano zawiera Rys. 49 oraz Tabela 14. Dodana normalizacja mini-serii podobnie jak w sieciach typu GajhedeCNN, dotyczy wyłącznie warstw konwolucyjnych, natomiast regularyzacja L2 tak jak w oryginalnej sieci SalamonCNN, dotyczy jedynie warstw gęsto połączonych. Sumarycznie przebadano 21 dodatkowych modyfikacji, a na trenowanie sieci algorytmem ADAM przeznaczono ok. 41 godzin.

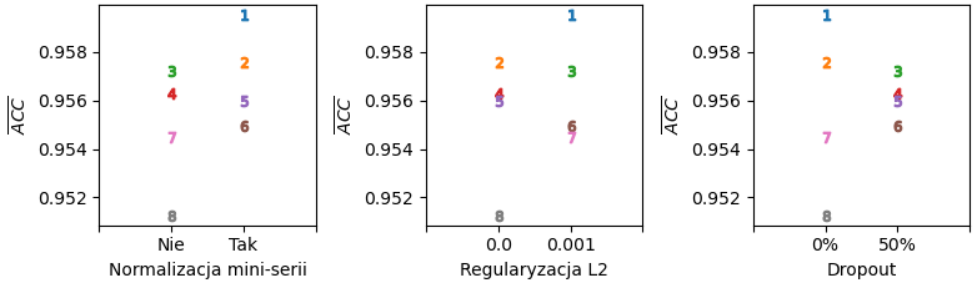


Rys. 49 Schemat blokowy dodatkowych modeli typu SalamonCNN.

Seria	N_{M1}	N_{M2}	N_{M3}	N_N	PL	$L2$	DP	BN
1	24	48	48	64	[2,3]	0, 0.001	0%, 50%	Nie, Tak
2a	12	24	48	32, 64, 128	[2,3]	0.001	0%	Tak

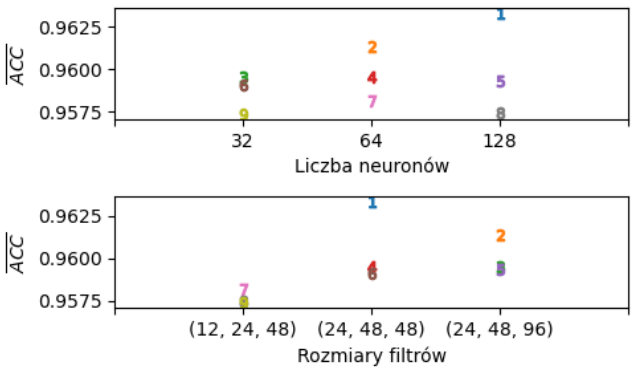
2b	24	48	48	32, 64, 128	[2,3]	0.001	0%	Tak
2c	24	48	96	32, 64, 128	[2,3]	0.001	0%	Tak
3	24	48	48	64	[3,2], [2,3], [3,3]	0.001	50%	Nie
4a	28	48	48	64, 128	[2,3], [3,3]	0.001	50%	Nie
4b	28	48	48	64, 128	[2,3], [3,3]	0.001	0%	Tak

Tabela 14 Testowane wartości hiperparametrów dla dodatkowych modeli typu SalamonCNN.



Rys. 50 Wpływ metody zapobiegania nadmiernemu dopasowaniu dla modeli typu SalamonCNN.

Przy pomocy pierwszej serii modeli zbadano jak dokładność klasyfikacji zmienia się dla różnych metod zapobiegania nadmiernemu dopasowaniu. Najlepsze okazały się modele z regularyzacją L2 i z normalizacją mini-serii lecz bez dropoutu (Rys. 50). Warto jednak zwrócić uwagę na zdecydowanie niższą wariancję pomiędzy dokładnościami różnych modeli z dropoutem. Sugerować to może lepszą zdolności generalizacji i/lub konieczności uczenia sieci z dropoutem przez dłużej niż 150 epok.



Rys. 51 Wpływ rozmiaru sieci dla modeli typu SalamonCNN.

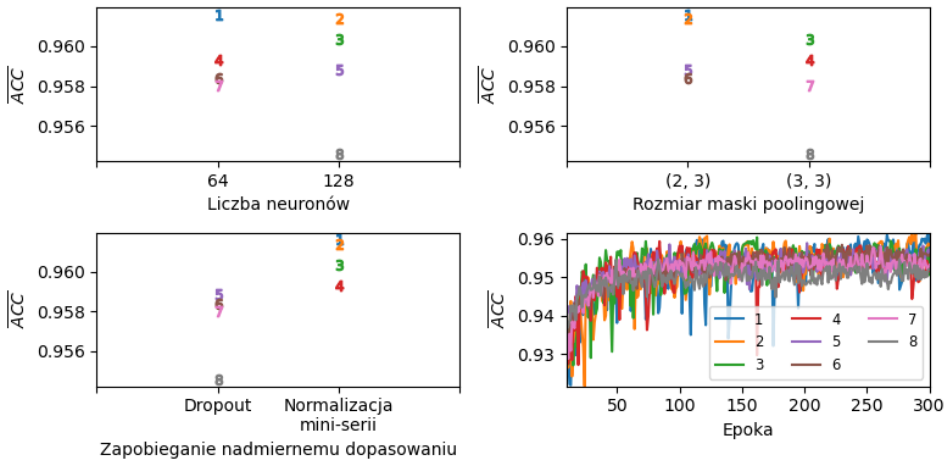
Za pomocą drugiej serii modeli zbadano jak na dokładność najlepszego modelu z pierwszej serii, wpłynie zmiana rozmiaru sieci (Rys. 51). Otrzymane wyniki są dość niejednoznaczne. Najlepszy model to ten w którym podwojono liczbę neuronów w ukrytej warstwie gęsto połączonej, jednak jego dokładność mocno odbiega od dokładności pozostałych modeli. Ciężko ocenić w jakim stopniu ta poprawa wynika z doboru hiperparametrów, a w jakim stopniu jest dziełem przypadku. Bardziej wyraźny wpływ na dokładność wydaje się mieć liczba masek w warstwach konwolucyjnych, lecz i w tym przypadku ciężko to stwierdzić z dużą pewnością dla tak małej liczby przykładów. Wszystkie wyniki zdają się jednak sugerować, iż dla postawionego problemu rozmiar bazowej sieci SalamonCNN najlepiej pozostawić bez zmian lub zwiększyć. Należy jednak pamiętać, że przez dodanie do warstwy gęsto połączonej zaledwie kilkudziesięciu kolejnych neuronów, w modelu pojawi się aż kilkaset tysięcy nowych parametrów, co oczywiście wydłuży proces uczenia i zwiększy zdolność modelu do zapamiętywania przykładów.

Rys. 52 Wpływ poolingu dla modeli typu SalamonCNN.

Duży wpływ na działanie sieci i liczbę jej parametrów ma również pooling. W trzeciej serii pomiarów badano więc, jaką dokładność osiągnie bazowa sieć dla masek max-poolingowych o różnych rozmiarach. Najlepsze wyniki otrzymano gdy kolumny (chwile czasu) i wiersze (częstotliwości) były redukowane równomiernie, lub gdy kolumny redukowano bardziej niż wiersze (Rys. 52). W zależności od wybranej maski szerokość i wysokość wyjścia z ostatniej warstwy konwolucyjnej modelu będzie inna. Dla masek z Rys. 52 będzie to od lewej kolejno: [20,3], [6,3] i [6,14]. W dwóch najlepszych modelach do warstwy gęsto połączonej docierają macierze w których wierszy jest co najmniej dwa razy więcej niż kolumn. Algorytm by

poprawnie zidentyfikować dźwięki perkusyjne, potrzebuje więc więcej informacji z dziedziny częstotliwości niż z dziedziny czasu. Jest to wynik zgodny z oczekiwaniami. Jak wyjaśniono w podrozdziale 3.3.5, czasowy przebieg amplitudy różnych dźwięków perkusyjnych o podobnej długości bywa bardzo zbliżony, dlatego by je rozróżnić należy porównać ich barwę.

W czwartej i zarazem ostatniej serii pomiarów celem było znalezienie jak najdokładniejszego modelu. Sieci trenowano przez 300 epok by zbadać czy przy dłuższym uczeniu dropout pozwoli uzyskać wyższą dokładność niż normalizacja mini-serii (Rys. 53).

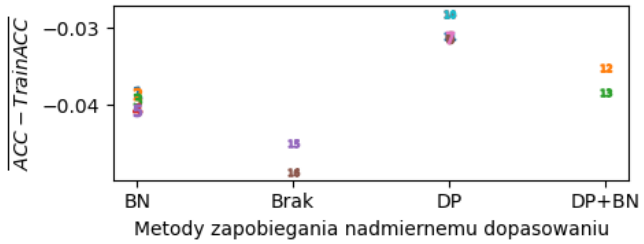


Rys. 53 Wyniki dla czwartej serii dodatkowych modeli SalamonCNN.

Wyniki w głównej mierze pokrywają się z tymi które przedstawiono już na Rys. 50, Rys. 51 i Rys. 52. Przy dłuższym uczeniu modele z normalizacją mini-serii również uzyskały wyższą dokładność od modeli z dropoutem. Różnica dokładności jest jednak rzędu zaledwie jednej dziesiątej procenta, więc nie stanowi to podstawy do wyciągania dalszych wniosków. Dla sieci neuronowych o innej architekturze być może uzyskano by wyniki przeciwne.

Finalnie najwyższą średnią dokładność uzyskano podczas drugiej serii pomiarów dla modelu o hiperparametrach: $N_M = [24, 48, 48]$, $N_N = 128$, $PL = [2, 3]$, $L2 = 0.001$, $DP = 0\%$, $BN = Tak$ oraz 88 epoki uczenia algorytmem ADAM. Model ten posiada 457418 parametrów trenowalnych i dalej nazywany będzie: BestCNN. Uśredniona dokładność otrzymana dla

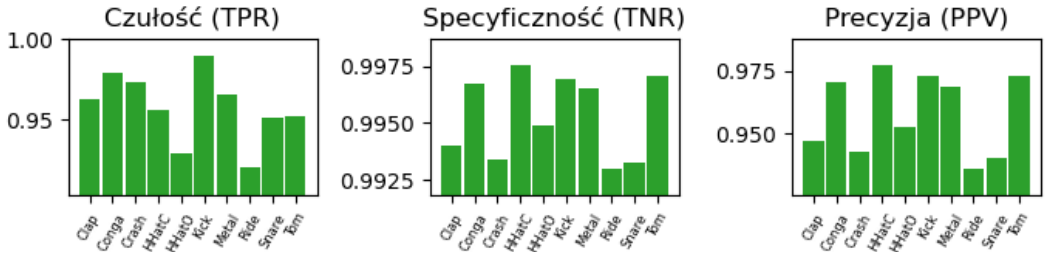
zbioru testowego jest równa $\overline{ACC} = 96.33\%$, a dla zbioru uczącego $\overline{TrainACC} = 99.9936\%$, co sugeruje, że w modelu mogło zajść nadmierne dopasowanie. Chcąc wykorzystać tą sieć w praktyce, warto było by rozważyć krótsze czasy uczenia, zmniejszenie liczby parametrów i dodanie dropoutu, gdyż wówczas dysproporcja pomiędzy dokładnością dla zbioru uczącego oraz testowego okazała się zawsze mniejsza (Rys. 54). Z tego powodu bezpieczniejszym wyborem dla najlepszego modelu od najdokładniejszego BestCNN wydaje się bazowy model typu SalamonCNN stosujący CQT podany podrozdziale 5.6.5 i dalej zwany SalamonCqtCNN.



Rys. 54 Różnica $\overline{ACC} - \overline{TrainACC}$ dla modeli z 1 i 4 serii (Tabela 14) stosujących różne metody zapobiegania nadmiernemu dopasowaniu.

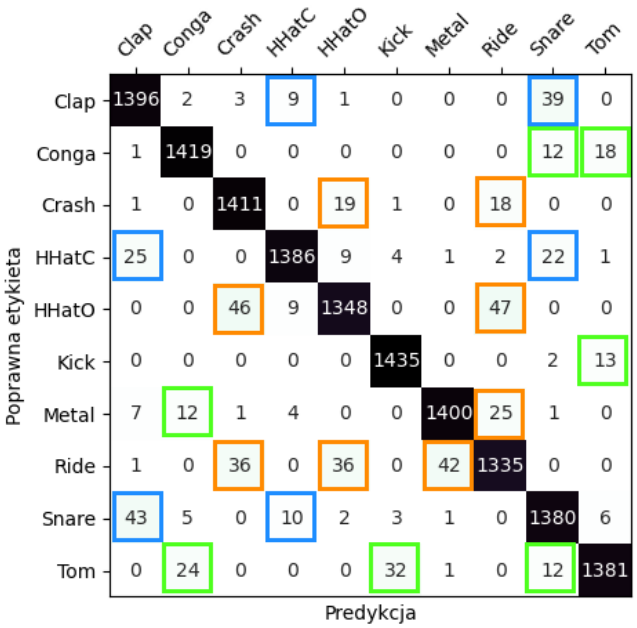
5.7 Najlepszy stworzony model

Na Rys. 55 przedstawiono czułość, specyficzność oraz precyzję modelu SalamonCqtCNN. Czułości i specyficzności dla każdej etykiety przyjmują zbliżone wartości więc nie wydaje się by w modelu zachodziła faworyzacja



Rys. 55 Czułość, specyficzność i precyzja dla SalamonCqtCNN.

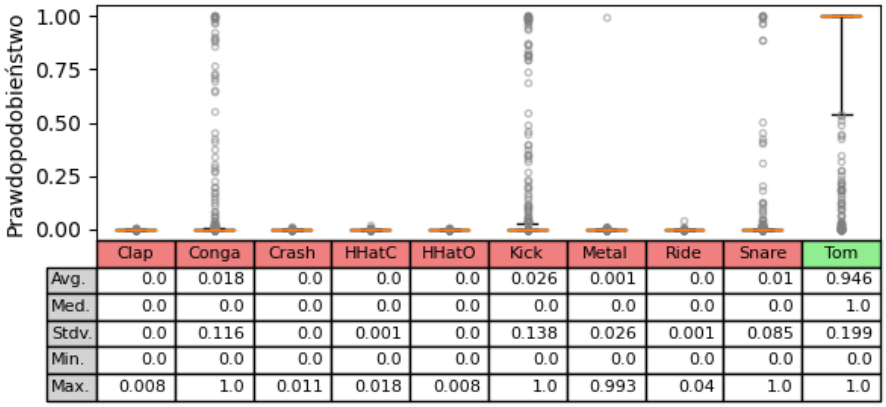
części etykiet. Warto jednak zwrócić uwagę na specyficzności etykiet „Crash”, „Ride” i „Snare”. Są one mimo wszystko zauważalnie niże od pozostałych specyficzności. Dźwięki w tych kategoriach okazały się najwidoczniej trudne do poprawnej klasyfikacji, gdyż czułość zbliżoną do czułości dla innych kategorii osiągnięto kosztem wielu fałszywie pozytywnych predykcji. Najgorzej jest w przypadku etykiety „Ride” oraz „HHatO”, gdyż zarówno specyficzność jak i czułość jest relatywnie niska.



Rys. 56 Tablica pomyłek modelu SalamonCqtCNN.

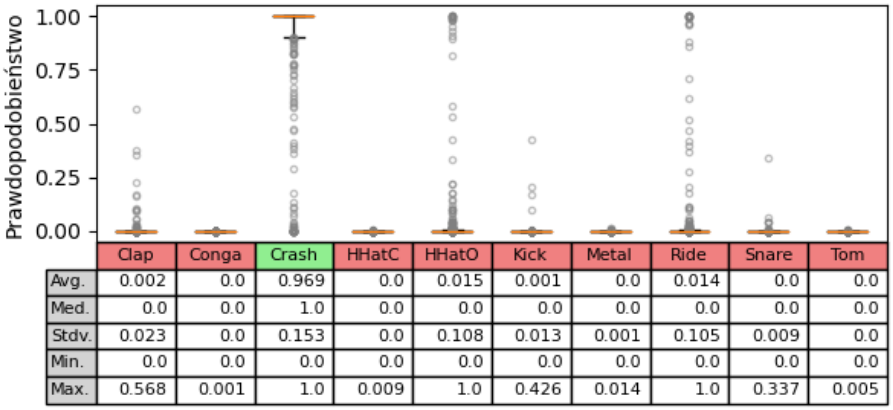
Na Rys. 56 przedstawiono tablicę pomyłek dla modelu SalamonCqtCNN. Większość pomyłek zaszła w obrębie jednej z trzech grup oznaczonych kolorowymi ramkami. Grupa pomarańczowa to długo wybrzmiewające dźwięki metaliczne („Ride”, „Metal”, „Crash”, „HHatO”), grupa niebieska to krótkie dźwięki z dużą ilością energii w okolicy ok. 2kHz („Snare”, „Clap”, „HHatC”), a grupa zielona to krótkie dźwięki z dużą ilością energii poniżej ok. 300 Hz („Kick”, „Tom”, „Conga”, „Snare”, „Metal”). Pomyłki dla pary etykiet zachodzą zazwyczaj w dwie strony. Otrzymany wynik jest całkowicie zgodny z przewidywaniami i stosunkowo łatwy do uzasadnienia. Przykładowo dźwięki dla etykiety „Kick” brzmią zazwyczaj jak niższe

wersje dźwięków dla etykiety „Tom”, a dźwięki dla etykiety „Metal” często brzmią bardzo podobnie do dźwięków dla etykiety „Ride”. Gdyby badany zbiór klasyfikował człowiek, to ewentualne pomyłki zapewne popełniłby pomiędzy tymi samymi etykietami. Jest więc duża szansa, że sieć dostrzega pewne cechy charakterystyczne dźwięków dla każdej etykiety i dokonuje klasyfikacji w oparciu o logiczne dla człowieka reguły.



	Clap	Conga	Crash	HHatC	HHatO	Kick	Metal	Ride	Snare	Tom
Avg.	0.0	0.018	0.0	0.0	0.0	0.026	0.001	0.0	0.01	0.946
Med.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
Stdv.	0.0	0.116	0.0	0.001	0.0	0.138	0.026	0.001	0.085	0.199
Min.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Max.	0.008	1.0	0.011	0.018	0.008	1.0	0.993	0.04	1.0	1.0

Rys. 57 Prawdopodobieństwa zwracane przez SalamonCqtCNN gdy poprawną etykietą jest „Tom”.

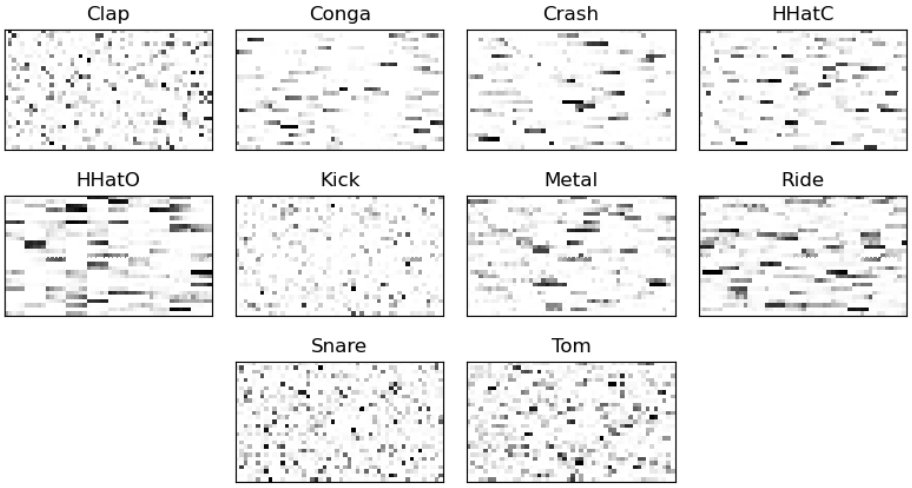


	Clap	Conga	Crash	HHatC	HHatO	Kick	Metal	Ride	Snare	Tom
Avg.	0.002	0.0	0.969	0.0	0.015	0.001	0.0	0.014	0.0	0.0
Med.	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Stdv.	0.023	0.0	0.153	0.0	0.108	0.013	0.001	0.105	0.009	0.0
Min.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Max.	0.568	0.001	1.0	0.009	1.0	0.426	0.014	1.0	0.337	0.005

Rys. 58 Prawdopodobieństwa zwracane przez SalamonCqtCNN gdy poprawną etykietą jest „Crash”.

Pomyłki dobrze ilustrują również wektory prawdopodobieństw zwracane przez klasyfikator gdy przedstawi się je w postaci wykresów pudełkowych. Przykładowe wykresy tego typu zamieszczono na Rys. 58 i Rys. 57. Jak można zauważyć, średnia prawdopodobieństwa dla niepoprawnych etykiet

są bardzo bliskie zeru. Pomyłki wynikają więc zapewne trudności części przykładów, a nie z nieposiadania wystarczająco dobrych wytycznych co do tego jak powinien brzmieć dźwięk dla wybranej etykiety. Taką hipotezę zdaje się potwierdzać również występowanie sytuacji, w których klasyfikator jest w niemal stu procentach przekonany co do poprawności niepoprawnej etykiety. By takie zdarzenie miało miejsce klasyfikowany dźwięk musi się bardzo różnić brzmieniem od wszystkich dźwięków w zbiorze uczącym, lub równie słuszną predykcją musi być zwrócona etykieta umownie traktowana jako błędna. Stworzeniem zbioru dźwięków stosowanego w niniejszej pracy zajęła się zaledwie jedna osoba, toteż istnieje szansa, że występują w nim pewne drobne przekłamania wynikające z subiektywnej oceny. Część dźwięków w zbiorze brzmieniowo znajduje się na pograniczu dwóch lub trzech etykiet, więc inna osoba mogła by uznać inne cechy za dominujące i przypisać części dźwięków inne etykiety referencyjne. Przed prowadzeniem ewentualnych kolejnych prac na tym zbiorze dobrze było by go dokładniej przebadać i potencjalnie udoskonalić.

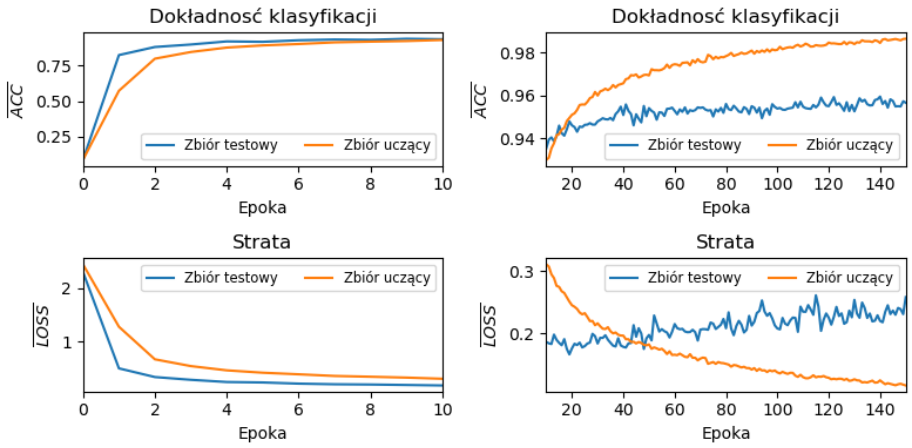


Rys. 59 Pomyłki popełniane dla poszczególnych plików przez wszystkie przebadane 164 różne modele sieci neuronowych.

Na Rys. 59 każdy mały kwadrat odpowiada dokładnie jednemu spośród wszystkich 14500 przykładów używanych podczas uczenia i walidacji sieci. Kolor kwadratu opisuje dla ilu różnych architektur sieci spośród wszystkich 164 przebadanych, poprawnie sklasyfikowało dany dźwięk, gdy ten podczas

walidacji krzyżowej był częścią zbioru uczącego (biały kolor – dla wszystkich, kolor czarny – dla żadnej). Wykres ten niejako ilustruje poziom trudności każdego z przykładów – im kolor bliższy jest czerni, tym przykład można uznać za trudniejszy. Spód wszystkich przykładów 47.31% zostało poprawnie sklasyfikowanych przez wszystkie sieci, 0.19% nie zostało poprawnie sklasyfikowanych przez żadną sieć, a ok. 4% zostało poprawnie sklasyfikowanych przez maksymalnie ok. 40% sieci. Biorąc te liczby pod uwagę, średnią dokładność na poziomie 96% uzyskaną przez BestCNN i SalamonCqtCNN można uznać za zadowalającą.

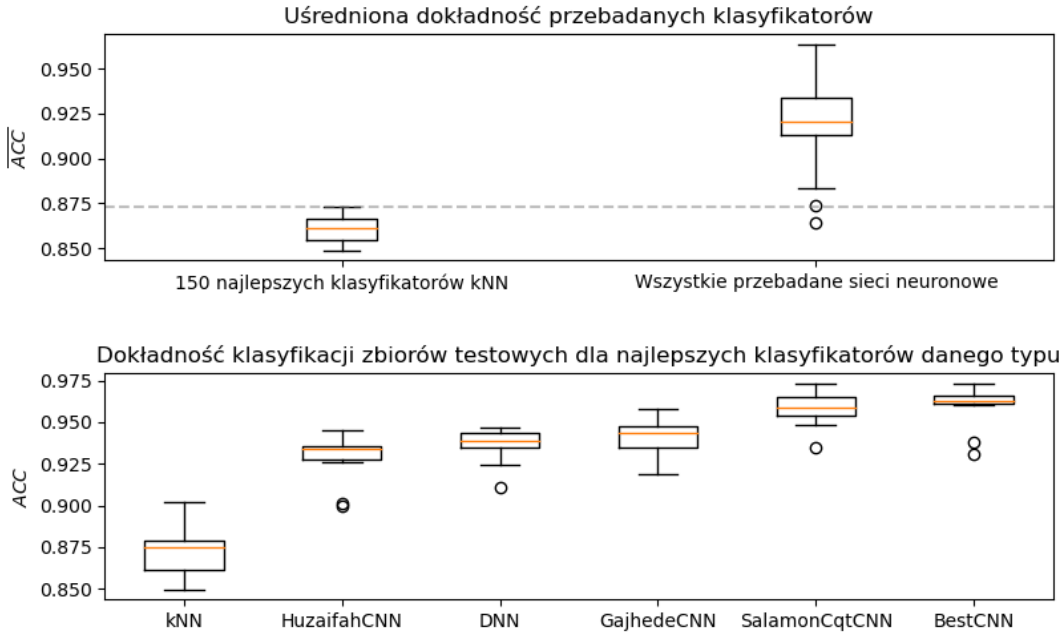
By zakończyć omawianie najlepszego modelu warto również odnieść się do kwestii czasowych. Czas uczenia pojedynczej sieci SalamonCqtCNN przez 140 epok za pomocą karty graficznej Nvidia Tesla K80 udostępnianej w serwisie Google Colab to ok. 390 sekund, a średni czas potrzebny na wykonanie jednej predykcji to ok. 0.06 ms. Czas predykcji stanowi zaledwie ułamek procenta całego czasu klasyfikacji na który składa się jeszcze wczytanie i wstępne przetwarzanie dźwięku (ok. 5.43 ms), oraz liczenie CQT (ok. 57.85 ms). Te dwa czasy zmierzono już na procesorze i7-4790 gdyż dla tych operacji użyte biblioteki nie wspierają kart graficznych.



Rys. 60 Postęp uczenia modeli SalamonCqtCNN.

Przebieg uczenia modelu pokazano na Rys. 60. Dla lepszej czytelności wykresy podzielono na dwa zakresy epok: 0 do 10 i 10 do 150.

5.8 Porównanie różnych modeli i różnych reprezentacji dźwięku



Rys. 61 Dokładności przebadanych klasyfikatorów.

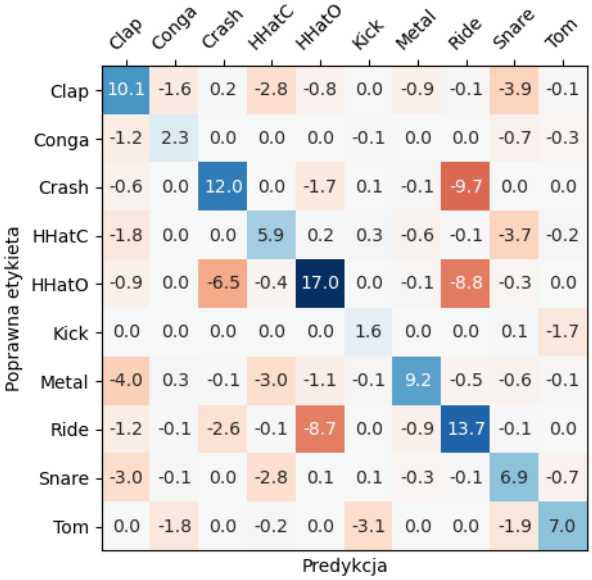
Na Rys. 61 porównano pod kątem dokładności wszystkie zbadane algorytmy kNN i wszystkie zbadane sieci neuronowe. Wyższą uśrednioną dokładność od najlepszego klasyfikatora kNN uzyskały nie tylko najlepsze sieci neuronowe danego typu, ale i niemal wszystkie sieci neuronowe przebadane w niniejszej pracy. Jedynym wyjątkiem jest sieć HuzaifahCNN stosująca dwudziestowierszowe MFCC jako wejście.

Dla najlepszych uzyskanych klasyfikatorów kNN uśredniona dokładność jest na poziomie 87%, natomiast dla najlepszych sieci neuronowych jest ona na poziomie 95%. Stosując sieci neuronowe zamiast klasyfikatora kNN udało się więc zwiększyć dokładność klasyfikacji o ok. 8 p.p. Pozornie poprawa nie wydaje się duża, jednak obu przypadkach dokładności są wysokie, więc nawet relatywnie niewielka zmiana ma znaczenie. Można to uzasadnić analizując jak często myli się klasyfikator w obu przypadkach przy pomocy zależności $1 - \overline{ACC}$. Dla $\overline{ACC_1} \approx 87\%$ stosunek pomyłek do

wszystkich klasyfikacji wynosi ≈ 13%, natomiast dla $\overline{ACC_2}$ ≈ 95% wynosi ≈ 5%. Iloraz tych dwóch liczb to:

$$\frac{100\% - \overline{ACC_1}}{100\% - \overline{ACC_2}} = \frac{100\% - 87\%}{100\% - 95\%} = \frac{13\%}{5\%} = 2.6$$

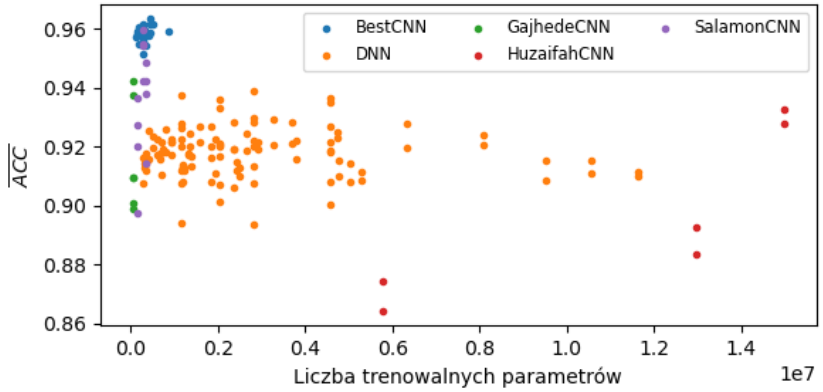
W pewnym sensie można więc powiedzieć, iż klasyfikator dla którego $\overline{ACC_2}$ ≈ 95%, jest ponad dwa razy lepszy (gdyż myli się 2.6 razy rzadziej) niż klasyfikator dla którego $\overline{ACC_1}$ ≈ 87%. Jest to oczywiście uproszczenie, jednak pozwala spojrzeć na poprawę 8 p.p. w innym świetle. Dla porównania gdy $\overline{ACC_2}$ ≈ 60% i $\overline{ACC_1}$ ≈ 40% pomimo poprawy dokładności o aż 20 p.p, lepszy klasyfikator myli się jedynie 1.5 razy rzadziej.



Rys. 62 Tablica pomyłek sieci SalamonCqtCNN po odjęciu od niej tablicy pomyłek najlepszego klasyfikatora kNN.

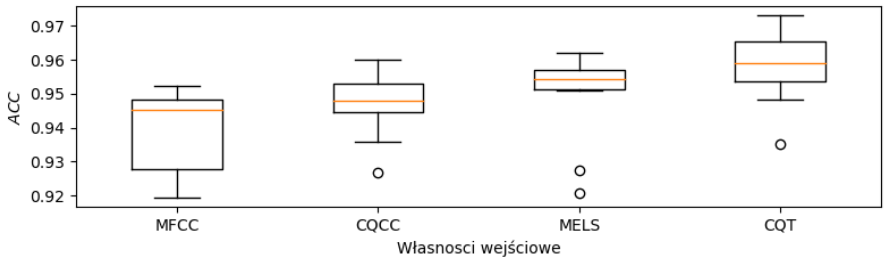
Na Rys. 62 zilustrowano różnicę pomiędzy znormalizowaną tablicą pomyłek dla sieci SalamonCqtCNN oraz dla najlepszego klasyfikatora kNN. Dodatnie wartości na przekątnej oznaczają zwiększenie czułości dla danej etykiety, a ujemne wartości dla pozostałych liczb wynikają z mniejszej ilości pomyłek pomiędzy daną parą etykiet. Wartości wyrażono w procentach. Za wyjątkiem kilku pomyłek zwiększonych o ok. 0.15%, zaobserwować można głównie poprawę wyników. W największym stopniu pomyłki zmniejszyły

się w obrębie grup wymienionych już podczas omawiania Rys. 56. Zdaje się to ponownie potwierdzać, iż w badanym zbiorze dźwięków występują trudne przykłady, z których klasyfikacją najwidoczniej lepiej radzą sobie sieci neuronowe.



Rys. 63 Wpływ liczby parametrów sieci neuronowej na dokładność.

Przeprowadzone badania wskazują, że pomiędzy liczbą parametrów sieci neuronowej i uzyskiwaną przez nią uśrednioną dokładnością, nie występuje znaczące powiązanie, gdyż modele o podobnej liczbie parametrów lecz odmiennej architekturze uzyskują bardzo różne dokładności (Rys. 63).

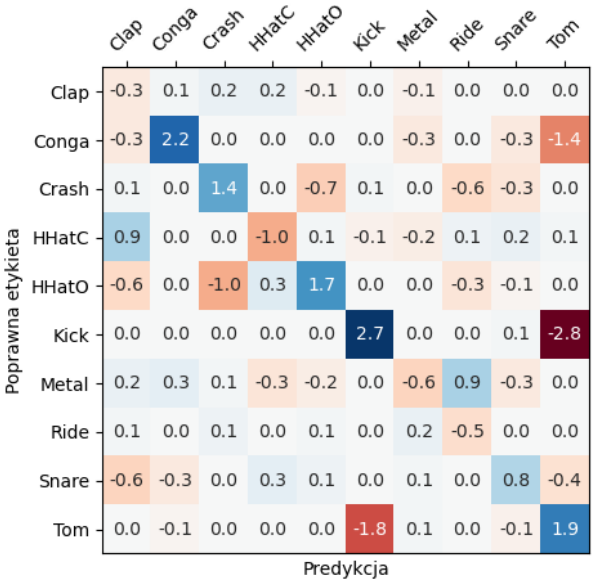


Rys. 64 Dokładność klasyfikacji zbiorów testowych przez sieci typu SalamonCNN dla różnych reprezentacji dźwięku.

Duży wpływ na dokładnosć okazała się mieć reprezentacja dźwięku podawana na wejście klasyfikatora. W większosci przypadków wyższą dokładność osiągnieto gdy zmaiast MFCC użyto CQCC, oraz gdy zamiast MELS użyto CQT. Dla sieci gęsto połączonych i klasyfikatora kNN lepsze okazały się reprezentacje CQCC i MFCC (Rys. 37, Rys. 40), natomiast dla

sieci konwolucyjnych lepsze okazały się reprezentacje CQC i MELS (Rys. 43, Rys. 45, Rys. 47, Rys. 48). Nie oznacza to jednak, iż dla sieci konwolucyjnych stosujących CQCC lub MFCC dokładność jest niska. Modele typu SalamonCNN dla każdej testowanej reprezentacji dźwięku, osiągnęły wyższą uśrednioną dokładność, niż dowolna zbadana sieć gęsto połączona tudzież klasyfikator kNN.

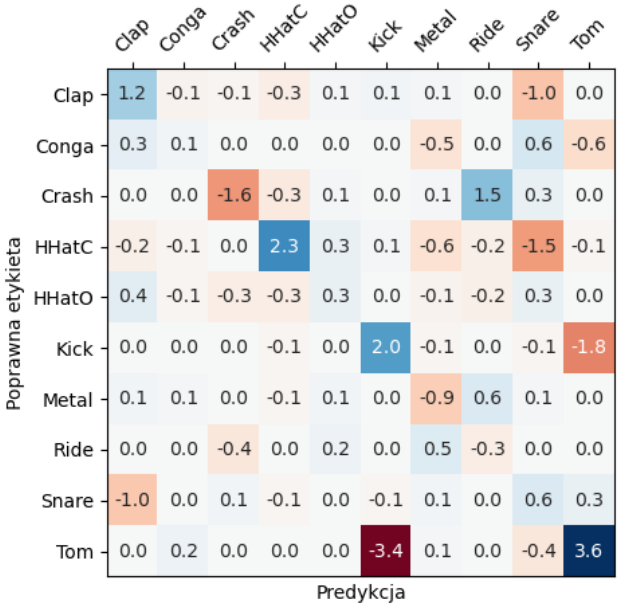
Na Rys. 64 przedstawiono dokładności uzyskane przez sieci typu SalamonCNN dla różnych reprezentacji dźwięku. Wyniki dla MFCC i CQCC ograniczono tylko do najlepszego przebadanego wariantu w którym liczba wierszy wynosi 40. Na podstawie otrzymanych wyników CQT wydaje się być najlepszym wyborem, gdy dokładność klasyfikacji ma duże znaczenie. Gdy istotny jest również sumaryczny czas klasyfikacji, to dobrą alternatywą jest szeroko polecane w literaturze MELS ze względu na średnio kilkukrotnie krótszy czas obliczeń (Rys. 24).



Rys. 65 Różnica znormalizowanych tablic pomyłek dla model typu SalamonCNN stosujących CQT oraz MELS.

W zależności od wybranej reprezentacji dźwięku, zmienia się również charakter pomyłek popełnianych przez klasyfikator. Jak pokazano na Rys. 65, zmiana macierzy własności z MELS na CQT, spowodowała zwiększenie

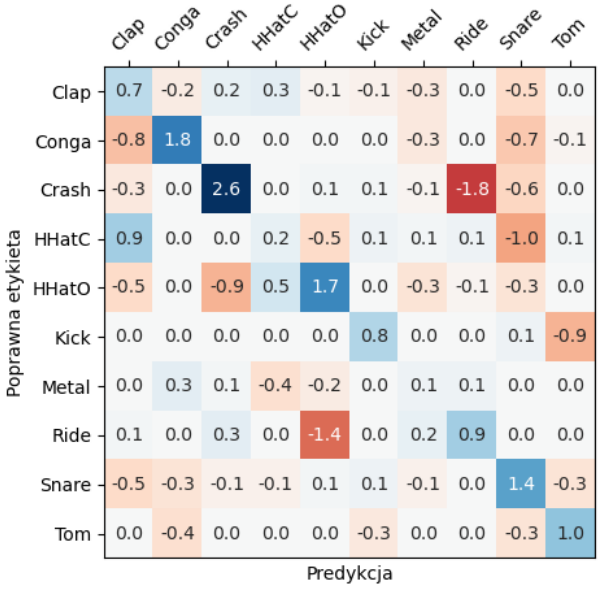
dokładności i zmniejszenie liczby pomyłek dla dźwięków zawierających dużo energii w niskich częstotliwościach (etykiety: „Kick”, „Tom”, „Conga”, „Snare”), oraz nieznaczne zmniejszenie dokładności i zwiększenie liczby pomyłek dla dźwięków zawierających dużo wysokich częstotliwości (etykiety: „Clap”, „HHatC”, „Metal”, „Ride”). Wynika to zapewne z tego, iż CQT w porównaniu do MELS posiada mniej prążków dla wysokich częstotliwości i więcej prążków dla niskich częstotliwości. Bardzo podobne zmiany w tablicy pomyłek można również zaobserwować, gdy zamiast CQCC zastosowane zostanie MFCC (Rys. 66).



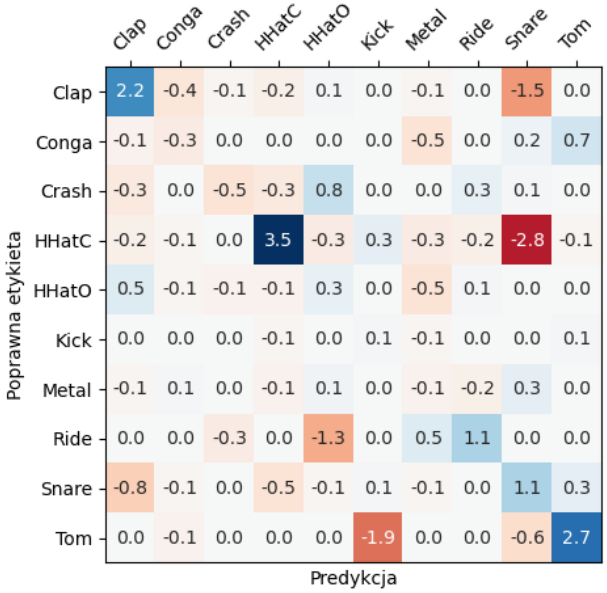
Rys. 66 Różnica znormalizowanych tablic pomyłek dla modeli typu SalamonCNN stosujących CQCC oraz MFCC. W obu przypadkach macierze ograniczono do 40 wierszy.

W przypadku zamiany MFCC na MELS (Rys. 68) lub CQCC na CQT (Rys. 67) zmiany w tablicy pomyłek również występują, jednak niemal wszystkie są na korzyść MELS tudzież CQT. Pojawia się niemniej również kilka pogorszeń. Przykładowo dla CQCC etykieta „HHatC” rzadziej jest mylona z etkietą „Clap” niż dla CQT, a dla MFCC etykieta „Conga” rzadziej jest mylona z etkietą „Tom” niż dla MELS. Ciężko określić dlaczego taka sytuacja ma miejsce. Być może po skompresowaniu macierzy własności przy pomocy DCT, usunięte zostały pewne detale brzmieniowe/szumy przez co

dla części dźwięków klasyfikator był w stanie lepiej dostrzec podobieństwo. Potwierdzenie takiej hipotezy wymagało by jednak dalszych badań i bliższej analizy przykładów dla których popełniono pomyłki.



Rys. 67 Różnica znormalizowanych tablic pomyłek dla modeli typu SalamonCNN stosujących CQT oraz CQCC o czterdziestu wierszach.



Rys. 68 Różnica znormalizowanych tablic pomyłek dla modeli typu SalamonCNN stosujących MELS oraz MFCC o czterdziestu wierszach.

6 Podsumowanie pracy

W ramach niniejszej udało się zrealizować wszystkie postawione cele, a otrzymane wyniki potwierdzają hipotezę przedstawioną we wstępie. Do klasyfikacji dźwięków perkusyjnych stosuje się w literaturze zazwyczaj klasyczne algorytmy uczenia maszynowego typu klasyfikator k najbliższych sąsiadów i las losowy, toteż dodatkowym celem pracy było określenie czy w przypadku gdy klasyfikowane są relatywnie trudne przykłady pochodzące z autorskiego zbioru danych, lepszym wyborem nie okażą się aby współcześnie stosowane sztuczne sieci neuronowe. Podczas badań wyznaczono dokładność klasyfikacji dla 156 różnych jednokierunkowych sieci neuronowych oraz 340 różnych klasyfikatorów kNN i otrzymane wyniki przemawiają jednoznacznie na korzyść sieci neuronowych. Średnia dokładność najlepszego przebadanego klasyfikatora kNN wynosi ok. 87%, natomiast średnia dokładność najlepszej przebadanej sieci neuronowej wynosi ok. 95%. Testy prowadzono na sieciach gęsto połączonych i konwolucyjnych, a wyższe dokładności udało się uzyskać dla tych drugich. Oczywiście dla innego klasycznego klasyfikatora wyniki mogły by być inne, jednak wyniki wskazują bardzo wysoki potencjał sieci neuronowych w tym obszarze.

Danymi wejściowymi dla badanych klasyfikatorów był dźwięk w postaci czasowo-częstotliwościowej wyznaczanej przy pomocy jednej z czterech metod omówionych w podrozdziałach od 3.4.3 do 3.4.6. Najwyższą dokładność klasyfikacji uzyskano gdy metodą tą była transformacja ze stałym Q. Jest to wynik ciekawy i warty uwagi zważywszy na to, iż w literaturze dotyczącej klasyfikacji dźwięku, zazwyczaj zaleca się stosowanie spektrogramu w skali melowej.

W ewentualnych dalszych pracach dotyczących klasyfikacji dźwięków perkusyjnych przy pomocy sieci neuronowych, oprócz testowania prostych sieci jednokierunkowych, można by przebadać sieci jednokierunkowe z rozwidleniami oraz sieci rekurencyjne. W niniejszej pracy nie poruszono jeszcze także takich zagadnień jak chociażby uczenie poprzez transfer wiedzy (ang. *transfer learning*). W celu potwierdzenia otrzymanych wniosków, warto było by również powtórzyć przeprowadzone już badania na innym zbiorze przykładów i z użyciem innych klasycznych algorytmów klasyfikacji takich jak las losowy czy klasyfikator Bayesa.

Inny kierunek badań mógłby dotyczyć testów na większej liczbie różnych reprezentacji dźwięku. Być może połączenie spektrogramu w skali melowej z wynikiem transformacji ze stałym Q (wysokie częstotliwości brane z pierwszej reprezentacji, niskie brane z drugiej) pozwoliło by uzyskać wyższą dokładność. Ciekawą alternatywą dla spektrogramów wydają się również osadzenia audio (ang. *audio embeddings*).

Bibliografia

[13] „Statistical classification, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Statistical_classification. [Data uzyskania dostępu: 25 04 2020].

[18] „Machine learning, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning. [Data uzyskania dostępu: 26 04 2020].

[19] „Deep learning, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Deep_learning. [Data uzyskania dostępu: 26 04 2020].

[21] „Nyquist–Shannon sampling theorem, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem. [Data uzyskania dostępu: 04 05 2020].

[23] „Envelope (music), Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Envelope_(music). [Data uzyskania dostępu: 07 05 2020].

[24] „Roland TR-808, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Roland_TR-808. [Data uzyskania dostępu: 07 05 2020].

[82] „A weighted Decibel Level Comparison Chart, Yale University,” [Online]. Available: https://ehs.yale.edu/sites/default/files/files/decibel-level-chart.pdf. [Data uzyskania dostępu: 07 05 2020].

[25] „Equal temperament, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Equal_temperament. [Data uzyskania dostępu: 08 05 2020].

[32] „Discrete cosine transform, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Discrete_cosine_transform#DCT-II. [Data uzyskania dostępu: 15 05 2020].

[37] „Feedforward neural network,” [Online]. Available: https://en.wikipedia.org/wiki/Feedforward_neural_network. [Data uzyskania dostępu: 18 05 2020].

[40] „Vectorization (mathematics), Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Vectorization_(mathematics). [Data uzyskania dostępu: 19 05 2020].

[41] „Universal approximation theorem, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Universal_approximation_theorem. [Data uzyskania dostępu: 19 05 2020].

[46] „Convolution arithmetic, deeplearning.net,” [Online]. Available: http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html. [Data uzyskania dostępu: 20 05 2020].

[48] „SGD, tensorflow,” [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/SGD. [Data uzyskania dostępu: 23 05 2020].

[53] „Dropout, Tensorflow,” [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout. [Data uzyskania dostępu: 26 05 2020].

[67] „Lanczos resampling, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Lanczos_resampling. [Data uzyskania dostępu: 29 05 2020].

[68] „DCASE2016, IEEE Audio and Acoustic Signal Processing (AASP) challenge,” 2016. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2016/index. [Data uzyskania dostępu: 29 05 2020].

[61] „Model evaluation - scikit-learn,” [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html. [Data uzyskania dostępu: 02 06 2020].

[62] „Accuracy_paradox, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Accuracy_paradox. [Data uzyskania dostępu: 02 06 2020].

[63] „Confusion_matrix, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Confusion_matrix. [Data uzyskania dostępu: 02 06 2020].

[64] „F1 score, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/F1_score. [Data uzyskania dostępu: 02 06 2020].

[65] „Harmonic mean, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Harmonic_mean. [Data uzyskania dostępu: 02 06 2020].

[57] „Image augmentation for convolutional neural networks,” [Online]. Available: https://medium.com/@ODSC/image-augmentation-for-convolutional-neural-networks-18319e1291c. [Data uzyskania dostępu: 04 06 2020].

[9] „Sekwencer (muzyka), Wikipedia,” [Online]. Available: https://pl.wikipedia.org/wiki/Sekwencer_(muzyka). [Data uzyskania dostępu: 20 06 2020].

[11] „Algonaut Atlas - Sound organization software,” [Online]. Available: https://www.algonaut.tech/. [Data uzyskania dostępu: 20 06 2020].

[12] „XLN XO - Sound organization software,” [Online]. Available: https://www.xlnaudio.com/products/xo. [Data uzyskania dostępu: 20 06 2020].

[83] „Pearl Europe,” [Online]. Available: https://www.pearleurope.com/home/. [Data uzyskania dostępu: 25 06 2020].

[77] „Splice - Royality-Free Sounds,” [Online]. Available: https://splice.com/. [Data uzyskania dostępu: 25 06 2020].

[78] „Reddit,” [Online]. Available: https://www.reddit.com/. [Data uzyskania dostępu: 25 06 2020].

[73] „Google Colab - Cloud computing,” Google, [Online]. Available: https://colab.research.google.com/. [Data uzyskania dostępu: 25 06 2020].

[72] „Python Release 3.7.7,” [Online]. Available: https://www.python.org/downloads/release/python-377/. [Data uzyskania dostępu: 25 06 2020].

[74] „Spyder IDLE - The scientific python development environment,” [Online]. Available: https://www.spyder-ide.org/. [Data uzyskania dostępu: 25 06 2020].

[75] „WinPython,” [Online]. Available: https://winpython.github.io/. [Data uzyskania dostępu: 25 06 2020].

[79] „Root mean square - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Root_mean_square. [Data uzyskania dostępu: 25 06 2020].

[80] „K-nearest neighbors algorithm - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm. [Data uzyskania dostępu: 26 06 2020].

[81] „TensorFlow metrics, accuracy,” [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/metrics/Accuracy. [Data uzyskania dostępu: 02 07 2020].

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg i X. Zheng, *TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems,* 2015.

[4] V. Alves de Souza, G. Batista i N. Souza Filho, „Automatic Classification of Drum Sounds with Indefinite Pitch,” 2015.

[17] S. Becker, M. Ackermann, S. Lapuschkin, K.-R. Müller i W. Samek, *Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals,* 2018.

[33] B. Blankertz, „The Constant Q Transform,” [Online]. Available: http://doc.ml.tu-berlin.de/bbci/material/publications/Bla_constQ.pdf. [Data uzyskania dostępu: 07 06 2020].

[22] J. Bodzenta, Wykłady z fizyki - wydanie drugie, uzupełnione i poprawione, Gliwice: Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, 2009.

[47] V. Bushaev, „How do we train neural network, towardsdatascience,” [Online]. Available: https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73. [Data uzyskania dostępu: 22 05 2020].

[49] V. Bushaev, „Stochastic gradient descent with momentum, towardsdatascience,” [Online]. Available: https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d. [Data uzyskania dostępu: 23 05 2020].

[50] V. Bushaev, „Understanding RMSprop faster neural network learning,” [Online]. Available: https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a. [Data uzyskania dostępu: 24 05 2020].

[51] V. Bushaev, „ADAM - latest trends in deep learning optimization,” [Online]. Available: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c. [Data uzyskania dostępu: 24 05 2020].

[58] S. Chatterjee, „Deep learning unbalanced training data solve it like this, towardsdatascience,” [Online]. Available: https://towardsdatascience.com/deep-learning-unbalanced-training-data-solve-it-like-this-6c528e9efea6. [Data uzyskania dostępu: 04 06 2020].

[1] B. Crew, „Google Scholar reveals its most influential papers for 2019, natureindex,” [Online]. Available: https://www.natureindex.com/news-blog/google-scholar-reveals-most-influential-papers-research-citations-twenty-nineteen. [Data uzyskania dostępu: 22 Kwiecień 2020].

[20] M. Fiałowska, B. Sagnowska i J. Salach, Z fizyką w przyszłość, Część 2 - Podręcznik sla szkół ponadgimnazjalnych, zakres rozszerzony, Kraków: ZamKor, 2013.

[39] K. Fujarewicz, *Slady z wykładu Metod Sztucznej Inteligencji, cz.1 Sieci Jednoierunkowe,* Gliwice: Wydział Automatyki, Elektroniki i Informatyki Politechniki Śląskiej, 2018.

[60] K. Fujarewicz, *Slady z wykładu Metod Sztucznej Inteligencji, cz.5 Walidacja,* Gliwice: Wydział Automatyki, Elektroniki i Informatyki Politechniki Śląskiej, 2018.

[2] N. Gajhede, O. Beck i H. Purwins, „Convolutional Neural Networks with Batch Normalization for Classifying Hi-hat, Snare, and Bass Percussion Sound Samples,” 2016.

[45] X. Glorot, A. Bordes i Y. Bengio, „Deep Sparse Rectifier Neural Networks,” 2010.

[71] K. He, X. Zhang, S. Ren i J. Sun, „Deep Residual Learning for Image Recognition,” 2016.

[43] S. Hershey, S. Chaudhuri, D. Ellis, J. Gemmeke, A. Jansen, R. Moore, M. Plakal, D. Platt, R. Saurous, B. Seybold, M. Slaney, R. Weiss i K. Wilson, „CNN architectures for large-scale audio classification,” 2017.

[54] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever i R. Salakhutdinov, „Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint,* tom arXiv, 7 2012.

[59] S. Ioffe i C. Szegedy, „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 2 2015.

[56] R. Karim, „Intuitions on L1 and L2 regularisation," [Online]. Available: https://towardsdatascience.com/intuitions-on-l1-and-l2-regularisation-235f2db4c261. [Data uzyskania dostępu: 03 06 2020].

[44] A. Khamparia, D. Gupta, N. Nhu, A. Khanna, B. Pandey i P. Tiwari, „Sound Classification Using Convolutional Neural Network and Tensor Deep Stacking Network," *IEEE Access,* tom PP, pp. 1-1, 1 2019.

[14] A. Krizhevsky, I. Sutskever i G. Hinton, „ImageNet Classification with Deep Convolutional Neural Networks," *Neural Information Processing Systems,* tom 25, 1 2012.

[42] T. Lidy i A. Schindler, *Cqt-based convolutional neural networks for audio scene classification and domestic audio tagging,* 2016.

[34] T. Lidy i A. Schindler, „Cqt-based convolutional neural networks for audio scene classification," 2016.

[8] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg i O. Nieto, „librosa: Audio and music signal analysis in python," w *In Proceedings of the 14th python in science conference, pp. 18-25. 2015.*.

[29] M. H. Md Shahrin, „Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks," 6 2017.

[55] S. Park i N. Kwak, „Analysis on the Dropout Effect in Convolutional Neural Networks," 2017.

[27] M. Pawełczyk, *Wykłady z podstaw cyfrowego przetwarzania dźwięku,* Gliwice: Instytut Automatyki Politechniki Śląskiej, Zakład Pomiarów i Systemów Sterowania, 2017.

[76] 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 i G. Louppe, „Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* tom 12, 1 2012.

[31] J. W. Picone, „Signal modeling techniques in speech recognition," *Proceedings of the IEEE,* tom 81, pp. 1215-1247, 9 1993.

[16] K. Piczak, „Environmental sound classification with convolutional neural networks," 2015.

[26] R. J. Rak i A. Majkowski, „Wirtualny analizator widma - Politechnika Warszawska, Instytut elektroniki teoretycznej i systemów informacyjno-pomiarowych," [Online]. Available: http://wazniak.mimuw.edu.pl/index.php?title=Laboratorium_wirtualne_1/Modu%C5%82_4_-_%C4%87wiczenie_4. [Data uzyskania dostępu: 11 05 2020].

[66] J. Salamon, C. Jacoby i J. Bello, „A Dataset and Taxonomy for Urban Sound Research," 2014.

[38] J. Salamon i J. Bello, „Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification," *IEEE Signal Processing Letters,* tom PP, 1 2017.

[35] C. Schörkhuber i A. Klapuri, „Constant-Q transform toolbox for music processing," *Proc. 7th Sound and Music Computing Conf.,* 1 2010.

[69] K. Simonyan i A. Zisserman, „Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv 1409.1556,* 9 2014.

[3] C. Southall, R. Stables i J. Hockman, „Automatic drum transcription for polyphonicrecordings using soft attention mechanisms andconvolutional neural networks," w *18th International Society for Music Information Retrieval Conference,* Suzhou, China, 2017.

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever i R. Salakhutdinov, „Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research,* tom 15, pp. 1929-1958, 6 2014.

[70] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens i Z. B. Wojna, „Rethinking the Inception Architecture for Computer Vision," 2016.

[28] G. Szwoch, „Analiza dźwięków muzycznych - Politechnika Gdańska, Katedra Systemów Multimedialnych, wykład z akustyki muzycznej," [Online]. Available: https://sound.eti.pg.gda.pl/student/akmuz/02-Analiza.pdf. [Data uzyskania dostępu: 06 05 2020].

[15] Y. Taigman, M. Yang, M. Ranzato i L. Wolf, „DeepFace: Closing the Gap to Human-Level Performance in Face Verification," 2014.

[10] M. Tan i K. McDonald, „The Infinite Drum Machine - Thousands of everyday sounds, organized using machine learning.," 05 2017. [Online]. Available: https://experiments.withgoogle.com/drum-machine. [Data uzyskania dostępu: 20 06 2020].

[5] A. Tindale, A. Kapur, G. Tzanetakis i I. Fujinaga, „Retrieval of percussion gestures using timbre classification techniques," 2004.

[7] P. Virtanen, R. Gommers, T. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. Walt, M. Brett, J. Wilson, K. Millman, N. Mayorov, A. Nelson, E. Jones, R. Kern i E. Larson, „SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods,* tom 17, pp. 1-12, 2 2020.

[30] C. Weihs, D. Jannach, I. Vatolkin i G. Rudolph, Music Data Analysis, Foundations and Applications, CRC Press, Chapman and Hall/CRC, 2019.

[36] J. Yang, R. Das i H. Li, „Extended Constant-Q Cepstral Coefficients for Detection of Spoofing Attacks," 2018.

Spis skrótów i symboli

Adam	algorytm gradientowy Adam
ADSR	obwiednia attack-decay-sustain-release
ANN	sztuczna sieć neuronowa
CNN	konwolucyjna sieć neuronowa
CQCC	współczynniki cq-cepstralne
CQT	transformata ze stałym Q
DCT	dyskretna transformata cosinusowa
DFT	dyskretna transformata Fouriera
DNN	gęsto połączona sieć neuronowa
FFT	szybka dyskretna transformata Fouriera
FNN	jednokierunkowa sieć neuronowa
FT	transformata Fouriera
GWM	gęstość widmowa mocy
kNN	algorytm k najbliższych sąsiadów
MEL-S	spektrogram w skali melowej
MFCC	współczynniki mel-cepstralne
RMS	wartość skuteczna sygnału
SGD	metoda gradientu prostego na mini-batchach
STFT	krótko-czasowa transformata Fouriera / spektrogram
SVM	algorytm support vector machine
A	amplituda
ACC	dokładność klasyfikacji

AF	funkcja aktywacji neuronu
BPO	liczba prążków przypadająca na oktawę
$BatchNorm$	warstwa realizująca normalizację mini-batchy
$Conv$	warstwa konwolucyjna
DP	procentowa wartość dropoutu
$Dense$	warstwa gęsto połączona
$Dropout$	warstwa dropoutowa
HL	skok okna
L	długość sygnału w próbkach
$LOSS$	wartość funkcji straty
$MaxPool$	warstwa max-poolingowa
NNL	liczba warstw sieci neuronowej
O	procentowe nakładanie okien
PPV	precyzja
SR	próbkowanie
TPR	czułość
WF	funkcja okna czasowego
WL	długość okna
f	częstotliwość
t	czas
y	wychylenie sygnału
$\boldsymbol{CONF}$	tablica pomyłek
$\boldsymbol{U}$	wyjście wybranej warstwy sieci neuronowej
$\boldsymbol{X}$	macierz własności / wejście sieci neuronowej lub warstwy
$\boldsymbol{Y}$	wyjście sieci neuronowej / wektor prawdopodobieństw
φ	faza
ϕ	indeks etykiety zwróconej przez klasyfikator

Zawartość dołączonej płyty

Na płycie DVD dołączonej do dokumentacji znajdują się następujące materiały:

- praca w formacie pdf,
- kod źródłowy wykorzystany do wygenerowania wyników,
- zbiory danych użyte w eksperymentach,
- najważniejsze wyniki.