POLITECHNIKA WARSZAWSKA

WYDZIAŁ ELEKTRONIKI I TECHNIK INFORMACYJNYCH

Rozprawa doktorska

mgr Karol Jerzy Piczak

Klasyfikacja dźwięku za pomocą splotowych sieci neuronowych

Promotor

dr hab. inż. Jarosław Arabas, prof. PW

WARSZAWA 2018

Streszczenie

Niniejsza rozprawa skupia się na temacie wykorzystania splotowych sieci neuronowych do klasyfikacji dźwięku. Jej celem jest wykazanie, że modele tego typu, których efektywność została wcześniej potwierdzona w licznych zagadnieniach rozpoznawania obrazów, można z powodzeniem zastosować również w zadaniach klasyfikacji dźwięków o ogólnym charakterze i to pomimo występującej między tymi obszarami dysproporcji w dostępności etykietowanych zbiorów danych.

Rozprawa prezentuje jedne z pierwszych opublikowanych w literaturze przykładów użycia splotowych sieci neuronowych do klasyfikacji dźwięków środowiskowych i rozpoznawania gatunków ptaków śpiewających. Zaproponowana w tym celu metoda opiera się na przetwarzaniu spektrogramów wyrażonych w skali melowej za pomocą sieci wykorzystujących wertykalne filtry w pierwszej warstwie splotowej. Podejście takie zapewnia połączenie dobrej dokładności klasyfikacji z korzystnymi parametrami wydajnościowymi w porównaniu do architektur splotowych typowych dla przetwarzania obrazów. Zarówno wyniki przeprowadzonych eksperymentów, jak i pozytywny odbiór koncepcji przez szerszą społeczność zajmującą się tą tematyką, potwierdzają, że splotowe sieci neuronowe są obiecującym narzędziem w obszarze rozumienia dźwięku.

Poza wyczerpującym omówieniem literaturowym tematyki splotowych sieci neuronowych i klasyfikacji dźwięku, rozprawa zawiera również szczegółową analizę wrażliwości zaproponowanych modeli na zmiany wartości hiperparametrów. Zestawienie to jest jednym z najszerszych porównań tego typu przeprowadzonych dotychczas w literaturze przedmiotu.

Integralnym efektem prac badawczych podjętych w ramach rozprawy jest także utworzenie zbioru nagrań środowiskowych „ESC-50”, mające na celu poprawę sytuacji ograniczonej publicznej dostępności zasobów z etykietowanymi danymi tego typu. Znaczenie tej inicjatywy potwierdzają liczne publikacje innych autorów wykorzystujące ten zbiór jako punkt odniesienia w przeprowadzanych eksperymentach.

Słowa kluczowe: *splotowe sieci neuronowe, klasyfikacja dźwięku, spektrogram, ESC-50*

Abstract

This thesis focuses on the use of convolutional neural networks in sound classification. Its goal is to show that such models, whose effectiveness has been confirmed earlier in numerous image recognition problems, can be successfully applied in general sound classification tasks, and it is possible despite a pronounced gap between these areas with regard to the availability of labeled datasets.

This work presents some of the first published examples of using convolutional neural networks in environmental sound classification and singing bird recognition. The proposed method is based on processing mel-spectrograms through networks operating with vertical filters in the first convolutional layer. Such an approach combines good classification accuracy and favorable performance metrics when compared with convolutional architectures common in image processing. Both the experimental results and subsequent recognition by a broader community engaged in this subject confirm that convolutional neural networks are a promising tool in the area of sound recognition.

Apart from a thorough literature review pertaining to convolutional neural networks and sound classification, this thesis also contains a detailed analysis evaluating the sensitivity of proposed models to changes in hyperparameter values. This study is one of the broadest comparisons of this kind performed to date.

An integral outcome of the performed research work is the creation of the „ESC-50" dataset of environmental recordings, which tries to address the limited availability of publicly accessible resources with labeled data of this kind. The significance of this initiative is confirmed through numerous publications of other authors employing it as a reference dataset in their experiments.

Keywords: *convolutional neural networks*, *sound classification*, *spectrogram*, *ESC-50*

Spis treści

Wykaz stosowanych oznaczeń

Zmienne

a	skalar (liczba naturalna lub rzeczywista)
$\boldsymbol{a} = (a_1, \ldots, a_n)$	wektor
$\boldsymbol{A}$	macierz
$\boldsymbol{\mathsf{A}}$	tensor
a	skalarna zmienna losowa
$\mathbf{a}$	wektorowa zmienna losowa
$\mathbf{A}$	macierzowa zmienna losowa

Zbiory

$\mathbb{A}$	zbiór
$\{0, 1\}$	zbiór zawierający liczby 0 i 1
$\{0, 1, \ldots, n\}$	zbiór liczb całkowitych od 0 do n
$[a, b]$	przedział rzeczywisty zawierający a i b
$[a, b)$	przedział rzeczywisty z wyłączeniem b
$\lvert\mathbb{K}\rvert$	liczebność (moc) zbioru $\mathbb{K}$

Indeksowanie

a_i	element i wektora $\boldsymbol{a}$, indeksowanie od 1
a_{-i}	wszystkie elementy wektora $\boldsymbol{a}$ z wyłączeniem elementu i
$A_{i,j}$	element (i, j) macierzy $\boldsymbol{A}$
$\boldsymbol{A}_{i,\bullet}$	wiersz i macierzy A
$\boldsymbol{A}_{\bullet,i}$	kolumna i macierzy A
$A_{i,j,k}$	element (i, j, k) tensora $\boldsymbol{\mathsf{A}}$
$\boldsymbol{\mathsf{A}}_{\bullet,\bullet,i}$	przekrój przez tensor $\boldsymbol{\mathsf{A}}$
a_i	element i wektorowej zmiennej losowej $\mathbf{a}$

Funkcje

$f(\boldsymbol{x};\boldsymbol{\theta})$	funkcja $\boldsymbol{x}$ parametryzowana przez $\boldsymbol{\theta}$
$f * g$	splot (konwolucja) funkcji f i g
$f \star g$	korelacja wzajemna funkcji f i g
$\log x$	logarytm naturalny
$\log_{10} x$	logarytm dziesiętny
$\exp(x)$	funkcja eksponencjalna
$\sigma(x)$	sigmoidalna funkcja logistyczna, $\frac{1}{1+\exp(-x)}$
$\zeta(x)$	softplus, $\log(1+\exp(x))$

Funkcja f operująca na wartościach skalarnych może być zastosowana również do wektorów, macierzy lub tensorów: $f(\boldsymbol{x})$, $f(\boldsymbol{X})$, $f(\boldsymbol{\mathsf{X}})$. Zapis taki oznacza zastosowanie funkcji f dla każdego elementu oddzielnie: $\boldsymbol{A} = \sigma(\boldsymbol{X}) \implies A_{i,j} = \sigma(X_{i,j})$.

Pozostałe

$\frac{\mathrm{d}y}{\mathrm{d}x}$	pochodna y względem x
$\frac{\partial y}{\partial x}$	pochodna cząstkowa y względem x
$\nabla_{\boldsymbol{x}} y$	gradient y względem $\boldsymbol{x}$
$P(\mathrm{a})$	rozkład prawdopodobieństwa zmiennej dyskretnej
$p(\mathrm{a})$	rozkład prawdopodobieństwa zmiennej ciągłej lub zmiennej, której typ nie został określony
$\mathrm{a} \sim P$	zmienna losowa a o rozkładzie P
$\mathbb{E}_{X\sim P}[f(x)]$ lub $\mathbb{E}f(x)$	wartość oczekiwana $f(x)$ względem $P(\mathrm{x})$
$\mathcal{N}(\mu, \sigma^2)$	rozkład normalny ze średnią μ i wariancją σ^2
$\mathcal{U}(a, b)$	rozkład jednostajny
$\boldsymbol{A} \odot \boldsymbol{B}$	iloczyn Hadamarda (po współrzędnych) macierzy $\boldsymbol{A}$ i $\boldsymbol{B}$ (analogicznie dla wektorów)

$\left\|x\right\|_p$	norma L^p
x^+	część dodatnia
$\lfloor x \rfloor$	część całkowita (podłoga)
$\|x\|$	wartość bezwzględna, moduł liczby zespolonej
$\arg(x)$	argument liczby zespolonej
$\operatorname{sgn}(x)$	signum

Oznaczenia dla sieci neuronowych

$\mathbb{X}$	zbiór danych trenujących
$\boldsymbol{x}^{[i]}$	przykład i ze zbioru trenującego
$\boldsymbol{X}$	macierz przykładów uczących, przykład $\boldsymbol{x}^{[i]}$ w wierszu $\boldsymbol{X}_{i,\bullet}$
$y^{[i]}$ lub $\boldsymbol{y}^{[i]}$	klasa lub wektor etykiet powiązany z przykładem $\boldsymbol{x}^{[i]}$
$h_j^{(l)}$	wartość sumy wejść neuronu j w warstwie l (przed aktywacją)
$a_j^{(l)}$	wartość aktywacji neuronu j w warstwie l
$W_{i,j}^{(l)}$	waga połączenia między neuronem j w warstwie l i neuronem i w warstwie $l-1$ (warstwa 0 oznacza warstwę wejścia)
$\varphi(h)$	funkcja aktywacji (przejścia) neuronu
$\boldsymbol{\theta}_{(t)}$	wektor wag (parametrów) sieci w kroku t uczenia

Rozdział 1

Wprowadzenie

1.1 Motywacja

Nagłówki gazet od kilku lat ogłaszają, że nastała era *big data* i czeka nas kolejna rewolucja w praktycznie każdym aspekcie życia (Marr, 2015). Chociaż można się spierać, ile w tych hasłach jest nadmiernej fascynacji przeplatanej marketingowym koloryzowaniem, to niezaprzeczalnie dane składowane i przetwarzane w formie cyfrowej pomnażają się lawinowo. Według szacunków *International Data Corporation*, ich łączny rozmiar w roku 2025 przekroczy próg 163 zettabajtów (miliardów terabajtów) (IDC, 2017). Coraz częściej wykorzystywane będą też systemy w pełni autonomiczne i urządzenia codziennego użytku połączone w ramach rozwoju koncepcji *Internet of things*. Wzrośnie tym sposobem udział danych o charakterze krótkotrwałym, których rozmiar już teraz istotnie przekracza możliwości ich archiwizowania. Oznacza to nie tylko ogromną ilość danych w obiegu, ale też w dużej części bardzo ograniczony czas ich potencjalnego wykorzystania.

Jednocześnie, jednym z często spotykanych mitów pojawiających się w dyskusjach na temat fenomenu *big data* jest idea, że sama ilość dostępnych danych przekłada się automatycznie na wymierne korzyści. Cytując profesora Gary'ego Kinga, „*to jednak nie ilość danych jest rewolucyjna. Rewolucja big data polega na tym, że teraz możemy coś z tymi danymi zrobić*" (Shaw, 2014). Patrząc chociażby na mnogość zasobów multimedialnych zamieszczanych w takich serwisach jak YouTube czy Spotify, rola coraz lepszego rozumienia i docierania do interesujących treści staje się kluczowa. Sama ich potencjalna dostępność nie musi się bowiem przekładać na rzeczywiste wykorzystanie. Dla przykładu, według statystyk Spotify, około 20% spośród ponad 20 milionów utworów dostępnych w serwisie nie zostało odtworzone ani razu (Aguilar, 2013). Podkreśla to skalę zjawiska i fakt, że to nie sama ilość danych jest wartością. Wartością jest ich rozumienie, możliwość odpowiedniej selekcji i zaspokojenia konkretnych potrzeb użytkownika poprzez efektywne odnajdywanie i syntezę treści według zadanych przez niego kryteriów.

Mimo ciągłego rozwoju, systemy rekomendujące i wyszukujące treści multimedialne wciąż opierają się przede wszystkim na przygotowywanych odręcznie metadanych – opisach, oznaczeniach kategorii i innych atrybutach reprezentujących zwięźle zawartość poszczególnych nagrań. Proces tworzenia tego typu metadanych jest niezwykle żmudny. Dalszy wzrost skali problemu, zarówno pod kątem ilości obrabianych treści, jak i pożądanej szczegółowości opisu, napotyka więc szybko na ograniczenia w zakresie dostępnych zasobów ludzkich. Z tego powodu konieczne staje się generowanie tych atrybutów w sposób zautomatyzowany. Aby jednak było to możliwe, systemy komputerowe muszą być w stanie analizować pliki multimedialne na poziomie ich faktycznej treści.

Rozumienie zawartości nagrań otwiera też drogę do wielu zastosowań wykraczających poza wyszukiwanie odpowiednich utworów czy filmów o charakterze rozrywkowym. Wszędzie tam, gdzie występuje potrzeba skupienia się wyłącznie na fragmentach zawierających specyficzne i stosunkowo rzadkie zdarzenia, metody automatycznej analizy treści byłyby nieocenionym narzędziem przesiewania coraz większych cyfrowych archiwów. Nawet ograniczając się do klasyfikacji występujących źródeł dźwięku, na którym to obszarze skupia się niniejsza rozprawa, można wskazać wiele możliwości wykorzystania tego typu usprawnień.

Chyba najszerszym z nich jest ogólnie rozumiany monitoring, gdzie zdarzenia wymagające natychmiastowej reakcji operatora stanowią niewielki ułamek całości pozyskiwanego materiału, siłą rzeczy przekraczającego możliwości percepcyjne pojedynczej osoby. Mowa tu zarówno o dozorze w skali miast (wystrzały, krzyki, agresywne zachowania i zakłócanie porządku), pojedynczych obiektów (odgłosy włamania, wandalizmu) i pomieszczeń (upadanie i wołanie o pomoc osób starszych w domach opieki, pacjentów w szpitalach), ale też bardziej odległych od cywilizacji obszarów (oznaki kłusownictwa, wycinki drzew w rezerwatach przyrody). Dopełnieniem tego typu reaktywnych rozwiązań są systemy pasywne, które pozwalają na wielkoskalową i długofalową ocenę zmian zachodzących w środowisku – na przykład w celu monitorowania bioróżnorodności (migracje zwierząt) czy tworzenia bardziej dokładnych map akustycznych w obszarach zurbanizowanych (z podziałem nie tylko na poziom występującego hałasu, ale też jego dokładne źródła).

Rozumienie otoczenia dźwiękowego pozwoli również na tworzenie bardziej inteligentnych aparatów słuchowych czy chociażby telefonów komórkowych, które dostosowują ustawienia dzwonka nie tylko do poziomu hałasu (ruchliwa ulica, pomieszczenie biurowe), ale też konkretnego kontekstu (ławka w parku, kościół). Także w przypadku systemów autonomicznych – robotów, inteligentnych samochodów – kluczowym elementem ich skutecznego działania jest świadomość i orientacja w środowisku, w którym się znajdują. Wykrywanie anomalii w paśmie słyszalnym może być również wykorzystywane do diagnostyki obiektów technicznych (kontrola pracy silników i innych urządzeń, próby szczelności zbiorników),

w zastosowaniach medycznych (diagnostyka chrapania, rozpoznawanie bezdechu, szmerów płucnych i sercowych), a nawet wojskowych (wczesne ostrzeganie na rozległych obszarach, detekcja i neutralizacja dronów). Ten bardzo zdawkowy przegląd uzmysławia, jak wiele konsekwencji może mieć postęp w dziedzinie automatycznego rozumienia dźwięku.

Niestety, patrząc realnie na obecny stopień rozwoju techniki, faktyczne zdolności percepcji maszynowej są cały czas dosyć ograniczone i dalekie od wyobrażeń kreowanych przez popkulturę w filmach i powieściach z gatunku science-fiction. Chociaż systemy komputerowe są bardzo efektywne w gromadzeniu i przetwarzaniu ogromnych ilości danych, to cały czas daleko im do takiej łatwości i kompletności rozumienia zjawisk stojących za tymi danymi, jak ma to miejsce w przypadku ludzi. To co nam wydaje się często proste i naturalne, dla systemu komputerowego może być nie lada wyzwaniem. Ta wyraźna przepaść zmniejsza się jednak z roku na rok.

Jednym z motorów innowacji w tym zakresie jest zachodzący postęp sprzętowy. Przełomową zmianą w tym aspekcie okazała się na pewno popularyzacja wykorzystania procesorów graficznych do zastosowań ogólnych, otwierająca drogę dla tworzenia modeli o dużo większej złożoności obliczeniowej. Pozwoliło to na popularyzację gałęzi uczenia maszynowego określanej obecnie mianem głębokiego uczenia (*deep learning*). Sama idea wykorzystania modeli sieci neuronowych o wielu warstwach nie jest nowa, gdyż jej początków można się doszukiwać już w latach 70. poprzedniego stulecia **(Widrow & Lehr, 1990)**. Jednak dopiero jednoczesne połączenie pojawiających się możliwości obliczeniowych i kilku nowych, pozornie drobnych pomysłów pozwoliło na skok jakościowy. Było to o tyle ważne, że w wielu obszarach wzrosła praktyczna zastosowalność tworzonych rozwiązań, które przestały mieć charakter wyłącznie akademickich ciekawostek.

Kamieniem milowym na tej drodze było wprowadzenie do użycia modeli bazujących na splotowych sieciach neuronowych[1]. Ponownie, sam pomysł wykorzystania sieci o ograniczonym lokalnie przetwarzaniu danych nie był czymś całkowicie nowatorskim. Jego źródła można znaleźć już w pracach Fukushimy **(1980)** i Rumelharta et al. **(1986)**. Niemniej dopiero Krizhevsky et al. **(2012)** swoim modelem splotowym *AlexNet* skokowo poprawili wyniki osiągane w najbardziej znanym benchmarku z zakresu rozpoznawania obrazów *ImageNet Large-Scale Visual Recognition Challenge* **(Russakovsky et al., 2015)** i tym sposobem wyraźnie wytyczyli nowy kierunek na kolejne lata. Zmiana ta przyczyniła się do szybkiego powstania wielu rozwiązań, które wprowadziły nową jakość w zakresie dokładności komputerowych systemów rozpoznawania obrazów. Miejscami zaczęły one nawet przewyższać ludzkie zdolności w tym zakresie **(Cireşan et al., 2012)**. O istotności tego momentu świadczy fakt, że model splotowy *AlexNet* był w 2012 roku jedynym i pierwszym

[1] Termin *convolutional neural networks* będzie tłumaczony w rozprawie jako splotowe sieci neuronowe. Często spotykaną alternatywną formą są konwolucyjne sieci neuronowe.

modelem tego typu, który pojawił się w ramach *ImageNet*. Natomiast już rok później, w kolejnej edycji, praktycznie wszystkie rozwiązania zgłoszone przez uczestników opierały się na modyfikacjach tej koncepcji.

W zbliżonym czasie modele splotowe, wraz z innymi wariantami głębokich sieci neuronowych, zaczęły być wykorzystywane w dziedzinach wykraczających poza rozpoznawanie obrazów. Rozkwit tego typu technik bardzo widoczny okazał się w systemach rozumienia mowy. Badania prowadzone przez potentatów technologicznych, napędzane wyraźnie presją komercyjną w związku z wprowadzaniem inteligentnych asystentów osobistych pokroju *Apple Siri*, doprowadziły do zmiany w zakresie praktycznie stosowanych modeli na hybrydy sieci neuronowych z ukrytymi modelami Markowa, podążając później w kierunku sieci rekurencyjnych. Pokazało to, że tego typu zasady uczenia złożonych hierarchicznych reprezentacji danych i ich lokalnego przetwarzania mają sens nie tylko w przypadku danych wizualnych, ale też dźwiękowych.

Patrząc szerzej na okres lat 2012–2015 można więc z jednej strony powiedzieć, że w tym czasie dokonał się istotny postęp w rozumieniu treści multimedialnych przez systemy komputerowe. Z drugiej strony, trzeba też zauważyć, że był to postęp dosyć zawężony. Mimo bardzo szybkiego transferu wiedzy między dziedzinami i rosnącej popularności głębokiego uczenia, innowacje w zakresie rozpoznawania dźwięku w tym okresie ograniczały się przede wszystkim do zadań rozumienia mowy, pozostawiając obszar dźwięków o bardziej ogólnym charakterze jakby na uboczu. Sytuacja ta była bezpośrednią motywacją podjętych w ramach rozprawy prac badawczych związanych z nasuwającym się pytaniem – co może być powodem takiego stanu rzeczy? Czy modele opierające się na splotowych sieciach neuronowych można wykorzystać też w szerzej pojętych zadaniach klasyfikacji nagrań dźwiękowych?

Potencjalną odpowiedź na te pytania można rozpatrywać w ramach występowania trzech możliwych stanów: „*nie jest to w ogóle możliwe*”, „*nie jest to obecnie możliwe*”, „*nie jest to interesujący problem badawczy*”.

Zacznijmy od rozważenia wersji najbardziej wykluczającej. Hipotetycznie możliwe jest, że charakter całego świata dźwięków, wykraczającego poza bardzo specyficzny i rządzący się swoimi regułami podzbiór jakim jest ludzka mowa, może być na tyle odmienny i różnorodny, że bezpośrednie zaadaptowanie splotowych sieci neuronowych do tego typu zadań może okazać się niewystarczające. Konieczne w takim wypadku byłoby rozwijanie innych podejść, na przykład wykorzystujących atrybuty tworzone na bazie specjalistycznej wiedzy dziedzinowej w zakresie przetwarzania sygnałów.

Drugą możliwością jest występowanie innych czynników ograniczających rozpowszechnienie się tego typu modeli w obszarze rozpoznawania dźwięku. Obrazowałby to stan, gdzie sama metodyka jest adekwatna i może przynieść oczekiwane rezultaty, ale aktualny

etap rozwoju dziedziny nie pozwala na jej wykorzystanie – na przykład z powodu niedostępności tak licznych etykietowanych zbiorów danych jak w przypadku klasyfikacji obrazów.

Trzecia sytuacja wiązałaby się z bardziej fundamentalnym pytaniem – po co w ogóle zajmować się rozpoznawaniem dźwięku w szerszym zakresie? Być może jest to problem na tyle akademicki i odległy od realnych zastosowań, że nie ma on żadnego znaczenia praktycznego? Taka ewentualność też mogłaby tłumaczyć wolniejszy rozwój tej dziedziny wiedzy.

Ustalenie stopnia prawdziwości tych wariantów jest o tyle ważne, że od tego w dużej mierze zależy dalsze postępowanie. O ile pierwsza i trzecia możliwość dosyć wyraźnie stawiają pod znakiem zapytania kwestię zastosowania sieci splotowych do klasyfikacji dźwięku, to już w drugim przypadku występujący stan ma ewidentnie charakter tymczasowy. Wskazywałby on raczej na to, że odpowiednie inicjatywy badawcze mogłyby doprowadzić do istotnego postępu w tej dziedzinie. Być może stan ten w takim wypadku należałoby wręcz zdefiniować jako „*jest to możliwe, ale jeszcze nie doczekało się realizacji*". Byłby to argument potwierdzający, że jest to ciekawy i otwarty problem badawczy.

Uprzedzając trochę wywód prowadzony w dalszej części rozprawy, zdaniem autora właśnie taki wariant najlepiej opisuje rzeczywistość. Zaprezentowane w rozprawie wyniki prac eksperymentalnych **(Piczak, 2015a)**, a także następujący po roku 2016 skokowy wzrost liczby publikacji powielających i rozszerzających zaproponowane podejście, zdają się potwierdzać tezę, że zastosowanie splotowych sieci neuronowych w zadaniach klasyfikacji dźwięku jest możliwe i stanowi obiecujący kierunek badań. Co więcej, możliwe jest pomimo występowania czynników ograniczających związanych z dostępnością etykietowanych zbiorów nagrań.

Jeśli chodzi o kwestię oceny przydatności tego typu badań, to jest to oczywiście sprawa dosyć subiektywna. Wydaje się jednak, że wymienione we wstępie obszary zastosowań rozumienia dźwięku pokazują, że to raczej nie ich brak jest głównym czynnikiem hamującym rozwój tej dziedziny. Możliwości wykorzystania są liczne i wiele z nich miałoby duże znaczenie, gdyby udało się je przekuć na systemy działające w skali realnie spotykanej w praktyce. W tym aspekcie należałoby się natomiast zgodzić, że w porównaniu z problemem automatycznego rozpoznawania mowy, droga do wdrożeń przemysłowych w zakresie szerszego rozumienia dźwięku jest na tę chwilę faktycznie dłuższa. Nie powinno to jednak zniechęcać, a raczej motywować do bardziej wytężonej pracy.

1.2 Cel rozprawy

Nakreślony dotychczas motyw przewodni zbadania możliwości klasyfikacji dźwięku za pomocą splotowych sieci neuronowych można doprecyzować w formie bardziej konkretnych zamiarów, których osiągnięcie jest celem sporządzonej rozprawy, mianowicie:

1. Przede wszystkim, odpowiedzieć na pytanie: „***Czy splotowe sieci neuronowe mogą być z powodzeniem zastosowane w zadaniach klasyfikacji dźwięków niebędących mową**, w szczególności mając na uwadze ograniczoność etykietowanych zbiorów danych dostępnych w tej dziedzinie*”?

2. Zaprezentować działanie systemów opartych na tej metodzie w formie studium przypadku dla przykładowych zadań klasyfikacji dźwięków środowiskowych i śpiewu ptaków.

3. Przeanalizować wpływ zmian w architekturze i procesie uczenia sieci na skuteczność klasyfikacji.

4. Udostępnić etykietowany zbiór nagrań środowiskowych adekwatny dla zadań klasyfikacji wraz z poziomem odniesienia w postaci oceny skuteczności rozpoznawania osiąganej przez ludzi.

5. Pokazać możliwości głębszego rozumienia działania splotowych sieci neuronowych w zastosowaniach dźwiękowych poprzez wizualizację efektów uczenia sieci bazujących na przetwarzaniu spektrogramów.

1.3 Omówienie układu rozprawy

Rozprawa podzielona została na 8 rozdziałów. Rozdziały 2–3 zawierają wprowadzenie teoretyczne połączone z przeglądem literatury przedmiotu. Opis oryginalnych prac badawczych przeprowadzonych przez autora zawarty został w rozdziałach 4–7.

- **Rozdział 2** stanowi ogólne wprowadzenie do zagadnienia sztucznych sieci neuronowych i głębokiego uczenia, ze szczególnym naciskiem na modele splotowe. Oprócz omówienia funkcjonowania splotowych sieci neuronowych i głównych założeń uczenia modeli głębokich zawiera on przegląd najbardziej znaczących architektur splotowych występujących w literaturze, a także wybranych najnowszych koncepcji w zakresie konstruowania sztucznych sieci neuronowych.

- **Rozdział 3** skupia się na problematyce klasyfikacji dźwięku jako zagadnieniu maszynowego uczenia. Poza wprowadzeniem porównującym typy zadań rozpoznawania

dźwięku i sposoby oceny jakości uzyskiwanych rozwiązań, rozdział ten zawiera opis metod reprezentacji sygnałów dźwiękowych i przegląd literaturowy podejść wykorzystywanych w klasyfikacji dźwięku.

- **Rozdział 4** przedstawia efekt prac badawczych związanych ze stworzeniem udostępnionego publicznie zbioru nagrań środowiskowych „ESC". Obok rozszerzonego względem oryginalnej publikacji **(Piczak, 2015b)** opisu zawartości zbioru i analizy wyników uzyskanych przez podstawowe klasyfikatory i ludzi, w rozdziale tym zawarte zostało również zestawienie licznych prac odnoszących się do tego zbioru od czasu jego publikacji.

- **Rozdział 5** zawiera wyniki eksperymentalne związane z zastosowaniem splotowych sieci neuronowych do klasyfikacji dźwięków środowiskowych, stanowiące autorski wkład w rozwój tego obszaru badawczego. W stosunku do publikacji konferencyjnej **(Piczak, 2015a)** mocno poszerzona została część omówienia wyników klasyfikacji i wizualizacji efektów uczenia.

- **Rozdział 6** przedstawia możliwość klasyfikacji gatunków ptaków śpiewających za pomocą splotowych sieci neuronowych na przykładzie zadania konkursowego *BirdCLEF 2016* **(Piczak, 2016)**.

- **Rozdział 7** dopełnia przedstawione we wcześniejszych rozdziałach wyniki eksperymentalne poprzez pogłębioną analizę zachowania modeli splotowych sieci neuronowych w zadaniach klasyfikacji dźwięku. W ramach tego rozdziału oceniany jest wpływ kilkunastu czynników na dokładność klasyfikacji uzyskiwaną przez modele będące przedstawicielami czterech typów architektur splotowych. Analiza ta jest jednym z najszerszych zestawień tego typu przeprowadzonych dotychczas w literaturze odnoszącej się do problematyki rozpoznawania dźwięku.

- **Rozdział 8** stanowi podsumowanie rozprawy.

1.4 Powiązane publikacje

Wyniki oryginalnych prac eksperymentalnych przeprowadzonych przez autora, których omówienie znajduje miejsce w ramach rozprawy, zostały przedstawione w następujących publikacjach:

- K. J. PICZAK. ***Environmental Sound Classification with Convolutional Neural Networks***. Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP). Boston, USA. IEEE, 2015 - *oznaczana jako* **(Piczak, 2015a)**.
- K. J. PICZAK. ***ESC: Dataset for Environmental Sound Classification***. Proceedings of the ACM International Conference on Multimedia (ACMMM). Brisbane, Australia. ACM, 2015 - *oznaczana jako* **(Piczak, 2015b)**.
- K. J. PICZAK. ***Recognizing Bird Species in Audio Recordings Using Deep Convolutional Neural Networks***. Working Notes of CLEF. Évora, Portugalia. CLEF, 2016 - *oznaczana jako* **(Piczak, 2016)**.
- K. J. PICZAK. ***The Details That Matter: Frequency Resolution of Spectrograms in Acoustic Scene Classification***. Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), Monachium, Niemcy. TUT, 2017 - *oznaczana jako* **(Piczak, 2017)**.

Rozdziały 4, 5 i 6 zawierają rozszerzone i ujednolicone omówienie tych treści wraz z pogłębioną analizą wyników eksperymentów.

Rozdział 2

Splotowe sieci neuronowe

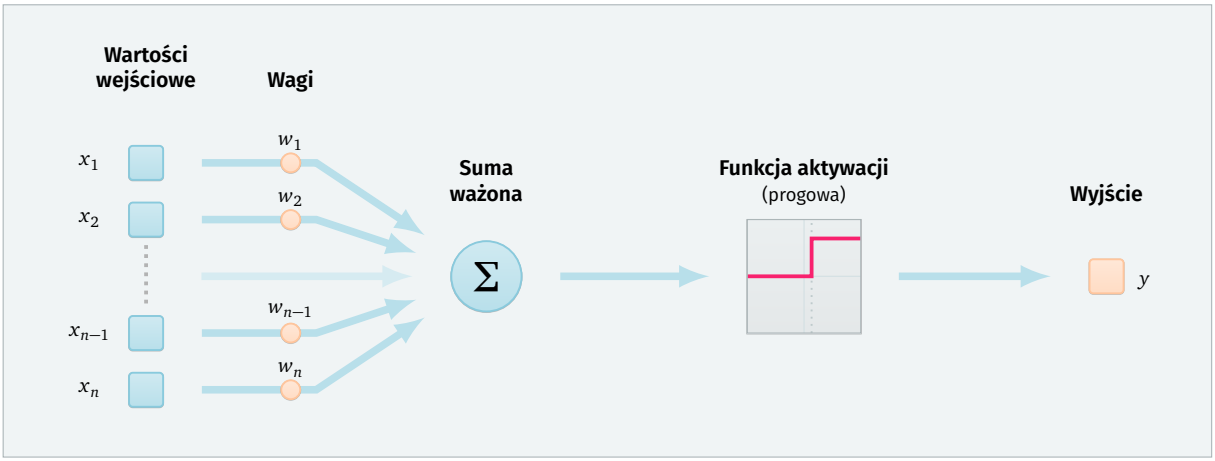
2.1 Podstawy funkcjonowania sztucznych sieci neuronowych

Sztuczne sieci neuronowe są rodziną modeli matematycznych, których funkcjonowanie inspirowane jest budową i działaniem naturalnych układów nerwowych, w szczególności ludzkiego mózgu. Chociaż z dzisiejszej perspektywy modele te stały się już nieodzownym i miejscami dominującym elementem uczenia maszynowego i nic nie wskazuje na to, by miały być w najbliższym czasie wyparte przez inne podejścia, to historia ich akceptacji jako jednej z obiecujących metod sztucznej inteligencji jest mocno burzliwa. Celem wprowadzenia zawartego w niniejszym rozdziale nie jest całościowe omawianie tej bogatej w wydarzenia drogi ich rozwoju, ale zwrócenie uwagi na kilka wybranych kluczowych koncepcji, które na przestrzeni lat ustanowiły podwaliny dla dzisiejszego stanu wiedzy i były przełomami, dzięki którym sztuczne sieci neuronowe stały się tak popularne. Szersze przedstawienie tematu, zarówno od strony historycznej, jak i teoretycznej, zawierają na przykład monografie Haykina (2004), Kriesela (2007), Hagana et al. (2014) czy w bardziej przystępnej i skondensowanej formie artykuł Kurenkova (2015).

2.1.1 Perceptron i model sztucznego neuronu McCullocha-Pittsa

Głównym celem wykorzystywanych obecnie modeli sztucznych sieci neuronowych nie jest wierne symulowanie działania struktur ludzkiego mózgu, ale raczej zastosowanie pewnych upraszczających analogii w tworzeniu możliwie efektywnych matematycznych metod rozwiązywania złożonych problemów. Na początkowym etapie rozwoju tej koncepcji bezpośrednie inspiracje naturą były jednak wyraźniejsze.

Punktem wyjściowym dla stosowanych dziś złożonych sieci neuronowych był zaproponowany w 1943 roku, na bazie ówczesnej wiedzy z zakresu neurofizjologi, uproszczony matematyczny model neuronu przedstawiony przez McCullocha i Pittsa (1943). W modelu tym zobrazowany na rysunku 2.1 sztuczny neuron jest jednostką przetwarzającą pewien



Rys. 2.1: Schemat funkcjonowania neuronu McCullocha-Pittsa (1943).

zbiór wartości sygnałów otrzymywanych na jego wejściu na pojedynczą wartość wyjściową. Składa się on z trzech elementów:

- sygnałów wejściowych $(x_1, \ldots, x_n)$, które ważone są wagami $(w_1, \ldots, w_n)$ (odpowiedniki połączeń synaptycznych),
- funkcji sumującej (analogia do potencjału komórkowego) postaci:

$$h = \sum_{i=1}^{n} w_i x_i \tag{2.1}$$

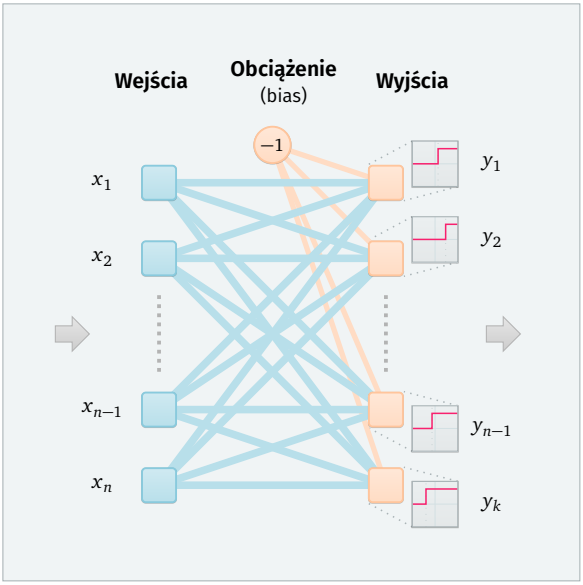
- funkcji aktywacji φ (przejścia), która określa wyjściową odpowiedź neuronu dla sumy wartości otrzymywanych na wejściu.

W przypadku modelu McCullocha-Pittsa funkcja ta była funkcją progową określoną dla progu aktywacji θ:

$$y = \varphi(h) = \begin{cases} 1 & \text{dla } h \geq \theta \\ 0 & \text{dla } h < \theta \end{cases} \tag{2.2}$$

Model ten jest bardzo prosty, ale McCulloch i Pitts pokazali w swojej pracy, że neurony takie potrafią z powodzeniem modelować podstawowe funkcje logiczne i sieć takich jednostek przetwarzających można wykorzystać do skonstruowania dużo bardziej złożonych systemów, które będą teoretycznie w stanie rozwiązywać skomplikowane zadania sztucznej inteligencji. Problemem tej koncepcji był jednak brak odniesienia do tego, jak w praktyce miałby przebiegać proces tworzenia i uczenia sieci takich neuronów.

Na bazie tej idei Rosenblatt (1957) przedstawił algorytm uczenia nadzorowanego dla klasyfikatorów binarnych określany mianem *perceptronu*. Jego implementacja wykorzystana została do stworzenia sprzętowego systemu rozpoznawania prostych kształtów na obrazach o rozmiarze 20 × 20 pikseli. Przedstawiony na rysunku 2.2 klasyfikator Rosenblatta stanowi w istocie zgrupowanie kilku neuronów McCullocha-Pittsa. Jedyną istotną modyfikacją wcześniejszej koncepcji jest wprowadzenie dodatkowego sztucznego wejścia o stałej wartości (np. −1) i wag obciążających (*bias*), które pozwalają na płynne przesuwanie wartości progu aktywacji. Samo programowanie perceptronu zachodzi poprzez uczenie funkcji klasyfikującej na bazie dokonywanych przez model obserwacji danych, jest więc typowym przykładem techniki uczenia maszynowego. Algorytm uczenia perceptronu można przedstawić w następujących krokach:



Rys. 2.2: Schematyczne przedstawienie modelu perceptronu Rosenblatta (1957).

- zainicjalizuj perceptron losowymi wagami (małe dodatnie lub ujemne wartości),
- dla każdego przykładu ze zbioru trenującego oblicz wartości wyjścia perceptronu,
- jeżeli wyjście jest różne od oczekiwanego:
  - gdy neuron na wyjściu wygenerował 1, a oczekiwano 0, to zmniejsz jego wagi dla wejść, które miały wartość 1,
  - gdy neuron na wyjściu wygenerował 0, a oczekiwano 1, to zwiększ jego wagi dla wejść, które miały wartość 1,
- powtórz proces do momentu, gdy nie zachodzą żadne zmiany lub przekroczono założoną liczbę kroków.

Dla bardziej uogólnionego przypadku, gdy wartości na wejściu należą do zbioru liczb rzeczywistych, regułę aktualizacji wag można rozszerzająco zdefiniować jako:

$$\widetilde{W}_{i,j} \leftarrow W_{i,j} - \eta(y_j - \mathring{y}_j)x_i \tag{2.3}$$

gdzie $W_{i,j}$ oznacza wagę neuronu j dla wejścia x_i, $\widetilde{W}_{i,j}$ jest odpowiednio wagą po aktualizacji, $\mathring{y}_j$ określa oczekiwaną poprawną wartość na wyjściu neuronu j, y_j natomiast wartość rzeczywiście wygenerowaną, a parametr η wyznacza tempo uczenia.

Głównym problemem tego modelu jest podkreślany w „*Perceptrons*”, szeroko komentowanej książce Minsky'ego i Paperta (1969), zakres ograniczeń perceptronu składającego się z pojedynczej warstwy neuronów. W szczególności, autorzy ci wykazali, że model taki nadaje się wyłącznie do klasyfikacji problemów liniowo separowalnych i z tego powodu nie jest możliwe nauczenie tego typu modeli np. odwzorowania funkcji logicznej XOR. Utarło się, że pojawienie się tej publikacji było historycznym momentem odwrócenia się całkowicie od idei perceptronu i autorom często mylnie przypisywano też rozszerzającą interpretację, jakoby nie tylko modele składające się z pojedynczej warstwy miały takie ograniczenia, ale też te wielowarstwowe obciążone były takimi defektami. W istocie to właśnie ograniczenie ekspresyjności pojedynczej warstwy wymusza zastosowanie modeli głębszych, z pośrednią warstwą neuronów. Kwestią sporną do dzisiaj jest, na ile wniosek ten był już w ówczesnym czasie powszechnie akceptowany, ale niewątpliwie głównym ograniczeniem był fakt, że nieznany na tamten moment był jeszcze żaden sposób uczenia sieci składających się z więcej niż 2 warstw (wejścia i wyjścia).

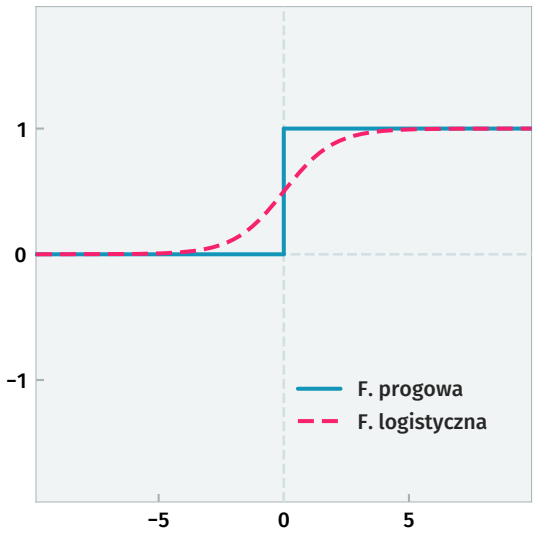
2.1.2 Perceptron wielowarstwowy i propagacja wsteczna

Z problemem uczenia sieci neuronowych składających się z kilku warstw zmagało się wielu badaczy (Werbos, 1974; Parker, 1985; LeCun, 1985), ale dopiero przedstawienie przez Rumelharta et al. (1988) odpowiednio przystępnego omówienia koncepcji uczenia za pomocą propagacji wstecznej błędu (*backpropagation*) sprawiło, że metoda ta została powszechnie przyjęta jako dominujący do dzisiaj sposób uczenia sztucznych sieci neuronowych. Jej znaczenie potwierdza fakt, że po blisko trzydziestu latach, mimo różnych modyfikacji, na fundamentalnym poziomie pozostaje ona wciąż jedynym podejściem, które wykorzystywane jest w codziennej praktyce.

Na czym więc opiera się idea propagacji wstecznej błędu? Punktem wyjścia jest zauważenie, że jeżeli omawianą wcześniej progową funkcję aktywacji zastąpimy przez możliwie jej bliską, ale różniczkowalną postać, na przykład przedstawioną na rysunku 2.3 funkcję logistyczną postaci:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

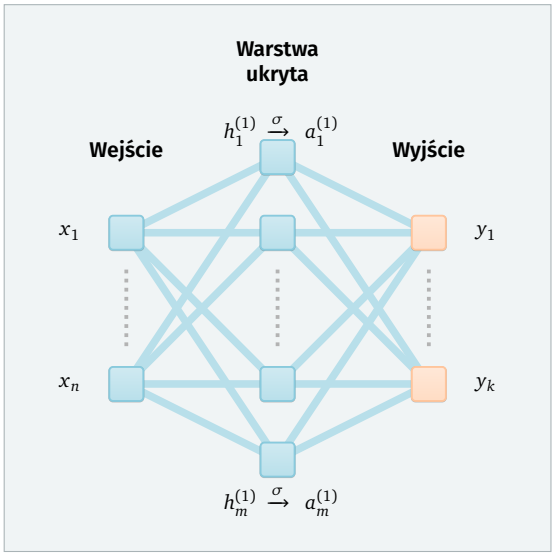
to możemy wykorzystać dobrze znaną z analizy matematycznej regułę łańcuchową do rozłożenia błędu otrzymywanego na wyjściu sieci na



Rys. 2.3: Porównanie progowej funkcji aktywacji z funkcją logistyczną.

poszczególne neurony w poprzedzających warstwach i odpowiednio pojedyncze wagi, przypisując im ich jednostkowy udział w zmianach wartości funkcji straty. Na bazie tej informacji możliwe staje się uczenie poprzez minimalizację błędu sieci za pomocą metody spadku gradientu (*gradient descent*).

Rozpatrując przykład przedstawionego na rysunku 2.4 perceptronu wielowarstwowego (*multi-layer perceptron*) z jedną warstwą ukrytą, uczenie sieci będzie przebiegało w dwóch krokach. W pierwszym (tzw. *forward pass*) dla zadanego zbioru wartości wejściowych $(x_1, \ldots, x_n)$ wyznaczane są wartości aktywacji poszczególnych neuronów warstwy ukrytej przy uwzględnieniu obciążeń $\boldsymbol{b}$:



Rys. 2.4: Schematyczne przedstawienie perceptronu wielowarstwowego.

$$a_j^{(1)} = \sigma(h_j^{(1)}) = \sigma\left(\sum_i x_i W_{i,j}^{(1)} + b_j^{(1)}\right) \quad (2.5)$$

i odpowiednio dla warstwy wyjścia:

$$y_j = a_j^{(2)} = \sigma(h_j^{(2)}) = \sigma\left(\sum_i a_i^{(1)} W_{i,j}^{(2)} + b_j^{(2)}\right) \quad (2.6)$$

Jeżeli teraz przyjmiemy funkcję straty, której wartość, zależna pośrednio od wag $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}$ i obciążeń $\boldsymbol{b}^{(1)}, \boldsymbol{b}^{(2)}$, określana jest formułą:

$$e = \frac{1}{2}\sum_j (y_j - \mathring{y}_j)^2 \quad (2.7)$$

to możliwe staje się wyrażenie wpływu zmian poszczególnych parametrów sieci na otrzymywaną wartość funkcji straty przez wsteczną propagację błędu (tzw. *backward pass*):

$$\frac{\partial e}{\partial y_j} = y_j - \mathring{y}_j \quad (2.8)$$

$$\frac{\partial e}{\partial h_j^{(2)}} = \frac{\partial e}{\partial y_j}\frac{\partial y_j}{\partial h_j^{(2)}} = \frac{\partial e}{\partial y_j}\sigma(h_j^{(2)})(1-\sigma(h_j^{(2)})) = \frac{\partial e}{\partial y_j} y_j(1-y_j) \quad (2.9)$$

$$\frac{\partial e}{\partial W_{i,j}^{(2)}} = \frac{\partial e}{\partial h_j^{(2)}}\frac{\partial h_j^{(2)}}{\partial W_{i,j}^{(2)}} = \frac{\partial e}{\partial h_j^{(2)}} a_i^{(1)} \quad (2.10)$$

$$\frac{\partial e}{\partial b_j^{(2)}} = \frac{\partial e}{\partial h_j^{(2)}}\frac{\partial h_j^{(2)}}{\partial b_j^{(2)}} = \frac{\partial e}{\partial h_j^{(2)}} \quad (2.11)$$

Wpływ wartości aktywacji neuronu w warstwie ukrytej na funkcję straty dokonuje się przez wszystkie neurony w kolejnej warstwie, które są z nim połączone, stąd:

$$\frac{\partial e}{\partial a_i^{(1)}} = \sum_j \frac{\partial e}{\partial h_j^{(2)}} \frac{\partial h_j^{(2)}}{\partial a_i^{(1)}} = \sum_j \frac{\partial e}{\partial h_j^{(2)}} W_{i,j}^{(2)} \tag{2.12}$$

$$\frac{\partial e}{\partial h_j^{(1)}} = \frac{\partial e}{\partial a_j^{(1)}} \frac{\partial a_j^{(1)}}{\partial h_j^{(1)}} = \frac{\partial e}{\partial a_j^{(1)}} \sigma(h_j^{(1)})(1-\sigma(h_j^{(1)})) = \frac{\partial e}{\partial a_j^{(1)}} a_j^{(1)}(1-a_j^{(1)}) \tag{2.13}$$

$$\frac{\partial e}{\partial W_{i,j}^{(1)}} = \frac{\partial e}{\partial h_j^{(1)}} \frac{\partial h_j^{(1)}}{\partial W_{i,j}^{(1)}} = \frac{\partial e}{\partial h_j^{(1)}} x_i \tag{2.14}$$

$$\frac{\partial e}{\partial b_j^{(1)}} = \frac{\partial e}{\partial h_j^{(1)}} \frac{\partial h_j^{(1)}}{\partial b_j^{(1)}} = \frac{\partial e}{\partial h_j^{(1)}} \tag{2.15}$$

Analogiczne postępowanie pozwala na uogólnienie tej metody dla większej liczby warstw. Tym sposobem możliwe staje się uczenie sieci o dowolnej głębokości. Na podobnej zasadzie jak w przypadku dowodzonej przez Cybenkę (1989) tezy, że sieć neuronowa z co najmniej jedną warstwą ukrytą jest w stanie aproksymować dowolną funkcję ciągłą przy wykorzystaniu skończonej liczby neuronów, stwierdzenia tego typu nie dają jednak praktycznych gwarancji, że takowy efekt uda się osiągnąć. Raczej pokazują tylko, że z teoretycznego punktu widzenia nie jest to niemożliwe.

W praktycznych zastosowaniach szybko okazało się bowiem, że specyfika łańcuchowego wyznaczania składowych gradientu za pomocą metody propagacji wstecznej błędu w przypadku zwiększania liczby warstw prowadzi do problemów ze stabilnością numeryczną. Wymnożony wielokrotnie w kolejnych warstwach sygnał gradientowy zanika całkowicie (*vanishing gradient*) lub przyjmuje ogromne wartości (*exploding gradient*), co powoduje, że uczenie sieci jest wyjątkowo powolne lub staje się całkowicie niemożliwe. Problem ten był jednym z głównych powodów, dla których przez wiele lat architektury głębokie nie cieszyły się zbytnią popularnością.

Propagacja wsteczna błędu sprawdzała się jednak w praktyce, jeśli chodzi o modele płytsze. Zaproponowana przez LeCuna et al. (1989) architektura składająca się z 3 warstw ukrytych, choć z dzisiejszej perspektywy bardzo prosta, w owym czasie była przykładem udanego wykorzystania sztucznej sieci neuronowej do rozwiązania bardzo konkretnego i realnego problemu – maszynowego odczytywania kodów pocztowych na kopertach. Z perspektywy tematyki podejmowanej w rozprawie ważne jest też to, że była to jedna z pierwszych architektur o charakterze splotowym. Przed jej bardziej szczegółowym omówieniem w podrozdziale 2.3.1, zajmiemy się jednak najpierw wprowadzeniem ogólnych zasad funkcjonowania modeli splotowych.

2.1.3 Modele splotowe

Wykorzystywane dzisiaj architektury splotowe przyjmują bardzo zróżnicowane postaci, ale w większości przypadków składają się z kombinacji kilku podstawowych elementów, które opisuje niniejszy podrozdział. Ich omówienie będzie miało charakter dwutorowy – poprzez przedstawienie na czym konkretnie polega zasada ich funkcjonowania oraz pokazanie motywacji, wyjaśniającej w jakim celu wprowadzenie tego typu modyfikacji do typowej sieci składającej się z kilku warstw w pełni połączonych może się okazać przydatne.

Splot

W analizie matematycznej splot (konwolucja) jest działaniem określonym dla dwóch funkcji (f, g), oznaczanym znakiem $*$, którego wynikiem jest trzecia funkcja (s) rozumiana jako zmodyfikowana wersja jednej z oryginalnych funkcji, taka że:

$$s(t) = (f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\,\mathrm{d}\tau = \int_{-\infty}^{\infty} f(t-\tau) g(\tau)\,\mathrm{d}\tau \tag{2.16}$$

a w wariancie dyskretnym odpowiednio:

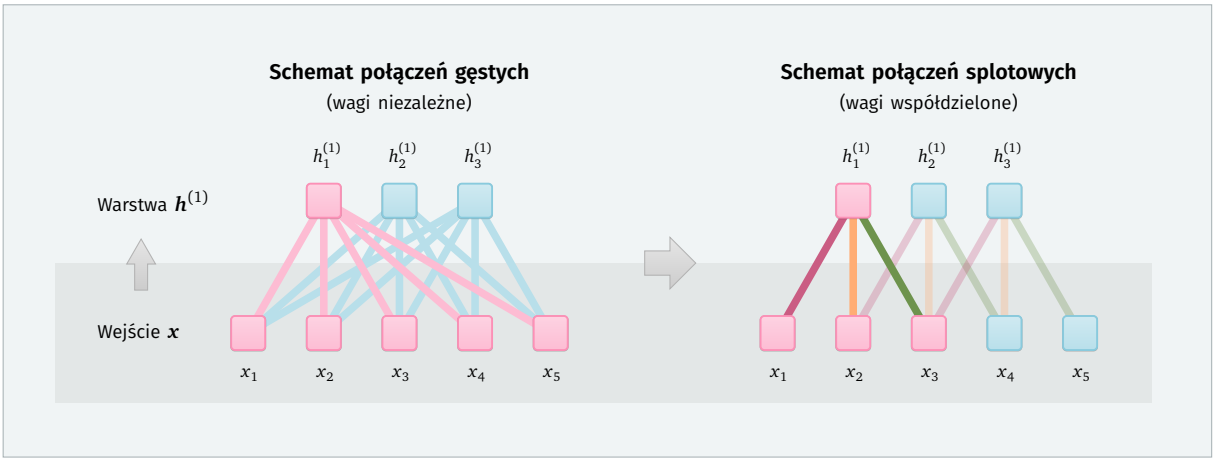
$$s(t) = (f * g)(t) = \sum_{\tau=-\infty}^{\infty} f(\tau) g(t-\tau) \tag{2.17}$$

W nomenklaturze wykorzystywanej do opisywania sieci neuronowych przyjęło się określać $f(\tau)$ jako wejście, $g(t-\tau)$ jako jądro lub filtr (*kernel, filter*), a wynik samej operacji jako mapę atrybutów lub aktywacji (*feature map*, *activation map*). Praktyka uczenia maszynowego ogranicza wykorzystanie splotu do przetwarzania danych dyskretnych z dodatkowym założeniem, że poza pewnym skończonym obszarem zarówno wejście, jak i filtr przyjmują wartości zerowe, co pozwala na ich przedstawienie za pomocą tensorów. Jednocześnie też bardzo częstym przypadkiem w tym kontekście jest wykorzystanie splotu w większej liczbie wymiarów. Dla typowego zadania obrazowego, operującego na dwuwymiarowym obrazie wejściowym $\boldsymbol{M}$ przy pomocy filtra (jądra) $\boldsymbol{K}$, splot można sformułować jako:

$$S(i,j) = (M * K)(i,j) = \sum_k \sum_l M(k,l) K(i-k, j-l) \tag{2.18}$$

lub korzystając z jego przemienności i odpowiednio zmieniając znaczenie indeksów k i l:

$$S(i,j) = (K * M)(i,j) = \sum_k \sum_l M(i-k, j-l) K(k,l) \tag{2.19}$$



Rys. 2.5: Porównanie warstwy gęstej (w pełni połączonej) z warstwą splotową.

Często dla wygody implementacje zawarte w popularnych bibliotekach zamiast splotu stosują operację o podobnym charakterze, którą w obszarze przetwarzania sygnałów formalnie wyróżnia się jako korelację wzajemną (*cross-corelation*):

$$\widetilde{S}(i,j) = (K \star M)(i,j) = \sum_k \sum_l M(i+k, j+l) K(k,l) \tag{2.20}$$

Odwrócenie kierunku przetwarzania w praktyce nie jest jednak istotne w zakresie w jakim operacje te znajdują zastosowanie przy tworzeniu sztucznych sieci neuronowych, gdyż w ramach właściwego uczenia sieci ustalane są wartości parametrów odpowiednie dla przyjętego wariantu (z odwróconym jądrem lub bez jego odwracania). Zazwyczaj więc, w kontekście sieci splotowych, operacje te określa się zbiorczo jednym terminem bez ich rozróżniania.

Sama definicja splotu jest jednak mało intuicyjna jeśli chodzi o obrazowanie zasady funkcjonowania sieci splotowych. Właściwie też na początkowym etapie rozwoju tej rodziny sieci termin ten nie pojawiał się raczej w powszechnym użyciu. Częściej mowa była o sieciach z współdzieleniem wag i warstwach o połączeniach ograniczonych lokalnie, gdyż w istocie na tym polegają różnice między warstwami splotowymi a standardowymi warstwami gęstymi (w pełni połączonymi). Sytuację tę obrazuje rysunek 2.5.

W typowej dla omawianego wcześniej perceptronu wielowarstwowego gęstej warstwie ukrytej, każdy neuron połączony jest z każdym neuronem warstwy poprzedzającej (w przypadku przedstawionym na rysunku będą to wartości wejścia). Jest to podejście szczególnie sensowne, gdy warstwa wejścia jest zbiorem wielu różnorakich atrybutów charakteryzujących zbiorczo daną obserwację. Natomiast w wariancie splotowym wartość aktywacji poszczególnych neuronów znajdujących się w warstwie ukrytej zależy wyłącznie od

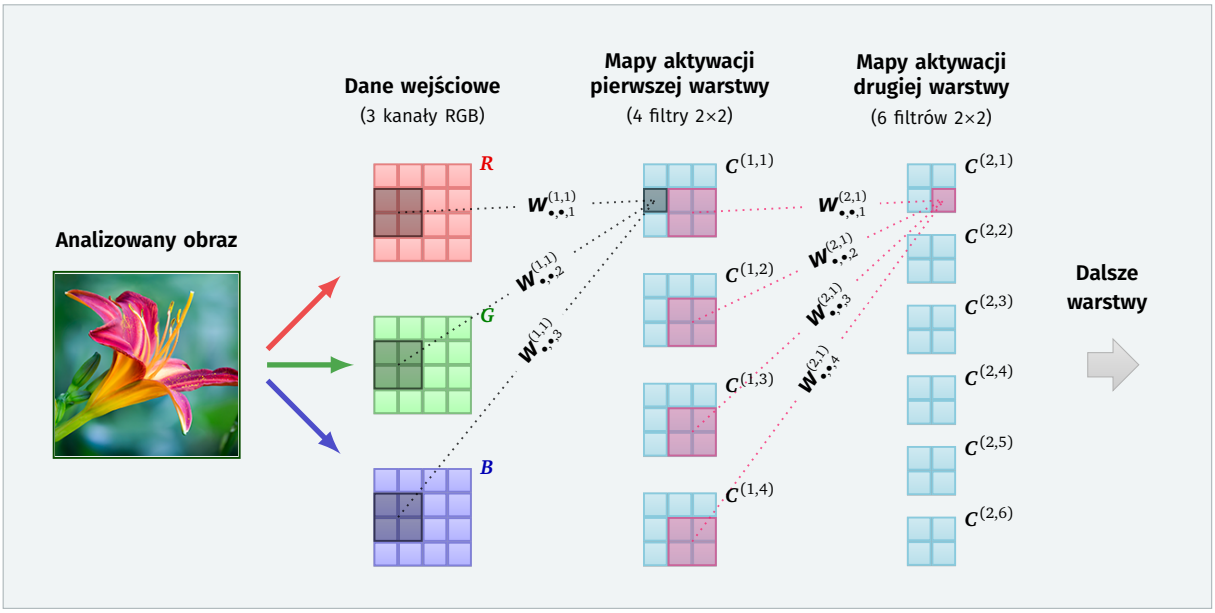
kilku neuronów z warstwy poprzedzającej, znajdujących się w pewnym lokalnym otoczeniu określanym jako pole postrzegania neuronu (*receptive field*). Dla przedstawionego przykładu, neuron $h_1^{(1)}$ opiera się wyłącznie na wartościach (x_1, x_2, x_3). Co więcej, uzyskiwane przez warstwę $\boldsymbol{h}^{(1)}$ wartości wyjścia są wynikiem zastosowania jądra o stałych wagach $(w_1^{(1)}, w_2^{(1)}, w_3^{(1)})$, które jest sukcesywnie wymnażane dla kolejnych podciągów sekwencji wejścia $(x_1, \ldots, x_5)$. W przypadku gdy analizowane dane mają strukturę, którą da się wyrazić za pomocą topologii siatki (szeregi czasowe dla danych 1D, obrazy w przypadku 2D), wprowadzenie tego typu ograniczeń liczby połączeń wynikających z założonej struktury modelu umożliwia uzyskanie kilku ważnych skutków.

Po pierwsze, w istotny sposób redukowana jest liczba parametrów koniecznych do reprezentacji sieci, co pozwala usprawnić proces jej uczenia i jest pewną pośrednią formą regularyzacji, zapobiegającą zbytniemu dopasowaniu do danych. Ważne jest także łatwiejsze wykrywanie wzorców i zależności statystycznych występujących w danych poprzez analizowanie ich dużo mniejszych fragmentów.

Druga kwestia również wiąże się ze specyfiką operowania przez neurony splotowe na niewielkich wycinkach poprzedniej warstwy. Zastosowanie tych samych wag dla różnych fragmentów danych, co można równoważnie rozumieć jako przesuwanie pojedynczego filtra względem zmieniającego się podciągu sekwencji wejściowej, skutkuje tworzeniem się reprezentacji danych odpornych na translację. Przykładowo, w przypadku przesunięcia krawędzi na obrazie, odpowiedź detektora splotowego wyczulonego na ich wykrywanie będzie dalej taka sama co do wartości, ale w odpowiednio innym miejscu generowanej na wyjściu mapy aktywacji. Własność ta powoduje, że warstwy splotowe bardzo dobrze spisują się jako detektory prostych cech, których charakter nie jest zależny od miejsca ich występowania. Natomiast kaskadowe wykorzystanie takich warstw pozwala uzyskać detektory cech wyższego rzędu, które mimo stosunkowo niewielkiej liczby wykorzystywanych wag zachowują wysoką ekspresyjność.

Zobrazowaniem koncepcji tworzenia detektorów splotowych w warunkach bliższych praktycznym implementacjom (warstwy splotowe operujące na wielokanałowych danych 2D) jest rysunek 2.6. W przykładzie tym kolorowy obraz rozdzielany jest na 3 kanały składowe reprezentowane przez macierze $\boldsymbol{R}$, $\boldsymbol{G}$, $\boldsymbol{B}$ stanowiące dane wejściowe dla sieci. Pierwsza warstwa splotowa $\boldsymbol{C}^{(1)}$ składa się z 4 filtrów o rozmiarach 2×2, które są sukcesywnie aplikowane dla kolejnych fragmentów obrazu. Wartością uzyskiwaną na wyjściu warstwy $\boldsymbol{C}^{(1)}$ dla filtra k na pozycji (i, j) i funkcji aktywacji φ jest:

$$C_{i,j}^{(1,k)} = \varphi\left(\sum_{r=1}^{2}\sum_{c=1}^{2} W_{r,c,1}^{(1,k)} R_{i+r-1,j+c-1} + \sum_{r=1}^{2}\sum_{c=1}^{2} W_{r,c,2}^{(1,k)} G_{i+r-1,j+c-1} + \sum_{r=1}^{2}\sum_{c=1}^{2} W_{r,c,3}^{(1,k)} B_{i+r-1,j+c-1}\right) \tag{2.21}$$



Rys. 2.6: Przykład działania warstw splotowych dla danych obrazowych.

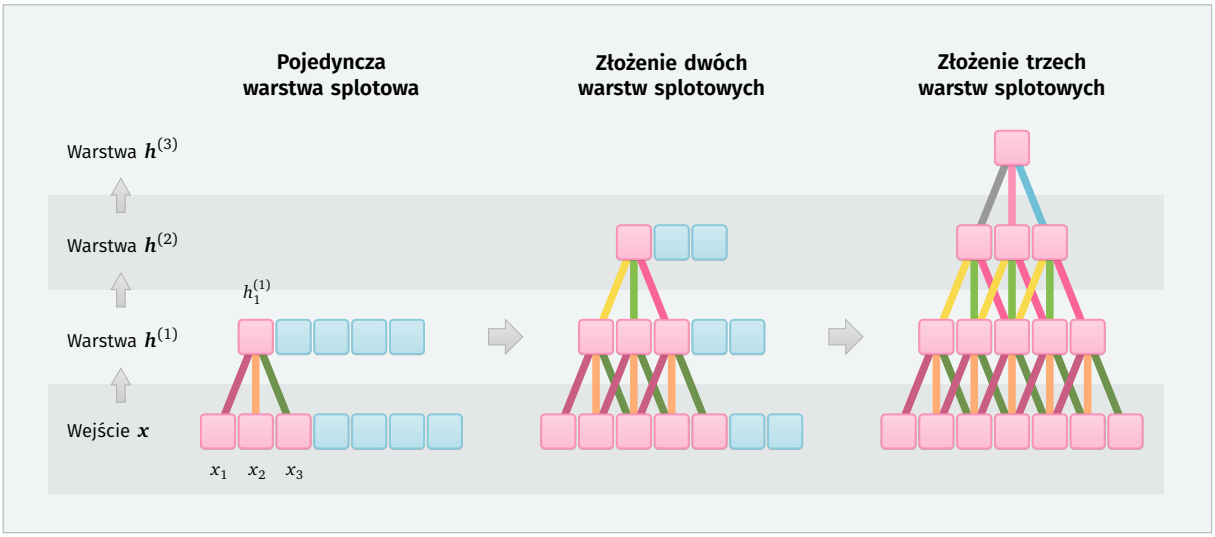
Wariant ten zakłada, że wynik splotu jest określony tylko w przypadku gdy filtr całkowicie pokrywa się z zakresem danych wejściowych, stąd generowana mapa aktywacji ma rozmiar odpowiednio mniejszy od macierzy wejścia. Możliwe są również alternatywne podejścia, w których brakujące wartości uzupełnia się zerami lub wartościami odbitymi od krawędzi.

Każda z tak wygenerowanych przez pierwszą warstwę map aktywacji staje się odrębnym kanałem dla kolejnej warstwy, tak że:

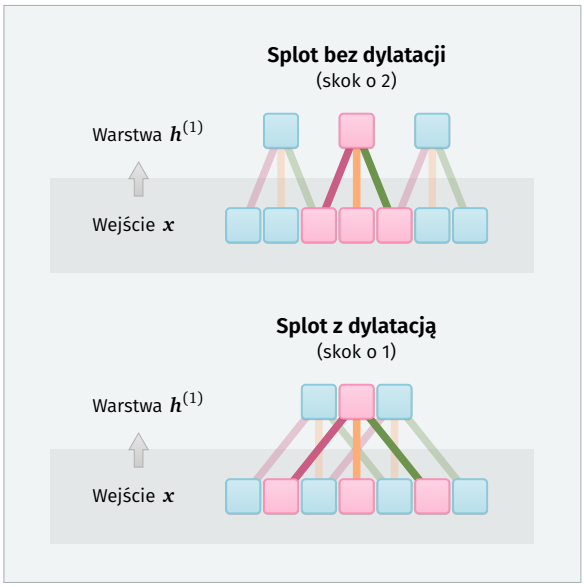
$$C_{i,j}^{(2,k)} = \varphi\left(\sum_{r=1}^{2}\sum_{c=1}^{2}\sum_{l=1}^{4} W_{r,c,l}^{(2,k)} C_{i+r-1,j+c-1}^{(1,l)}\right) \tag{2.22}$$

Taka struktura pozwala, a właściwie w pewien sposób odgórnie wymusza wytwarzanie się w ramach architektur splotowych reprezentacji hierarchicznych, w których kolejne warstwy wyrażają cechy jako odpowiednią kombinację cech wykrywanych przez warstwę poprzedzającą.

Specyficzną właściwością takich złożeń warstw splotowych jest to, że w typowych konfiguracjach ich pole postrzegania rośnie stopniowo. Efekt ten widoczny jest na rysunku 2.7. Dla sekwencji danych wejściowych $(x_1, \ldots, x_7)$, neuron $h_1^{(1)}$ jest uzależniony wyłącznie od wartości (x_1, x_2, x_3). Dopiero w warstwie $\boldsymbol{h}^{(3)}$ pole postrzegania neuronu obejmuje pośrednio cały zakres danych wejściowych. Działanie to wykazuje duże podobieństwo do ogólnej zasady funkcjonowania kory wzrokowej ludzkiego mózgu, w którym to następujące po sobie zgrupowania neuronów tworzą hierarchię w dużej mierze sekwencyjną, zajmującą się przetwarzaniem informacji na coraz wyższym poziomie agregacji.



Rys. 2.7: Zmiana pola postrzegania wraz z głębokością warstwy.

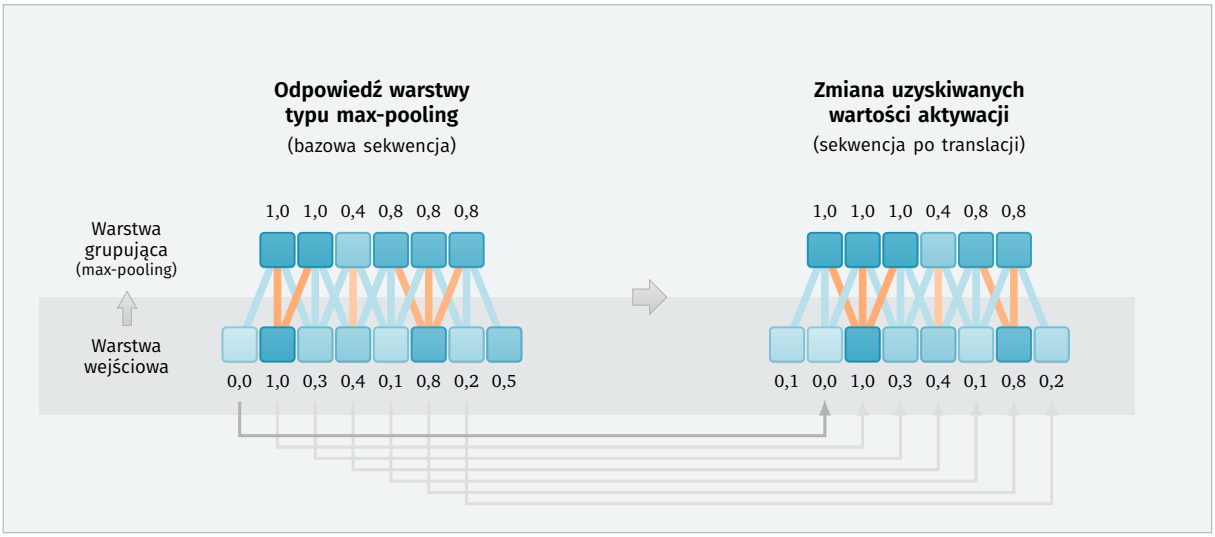


Rys. 2.8: Splot ze skokiem i z dylatacją.

Opisany dotychczas sposób postępowania zakładał, że filtr splotowy wymnażany jest sukcesywnie przez elementy wejścia na zasadzie przesuwającego się o jedną pozycję okna obejmującego sąsiadujące ze sobą elementy. Założenie to można jednak zmodyfikować. Zwiększanie skoku (*stride*) warstwy splotowej jest jednym z często stosowanych sposobów zmniejszania rozdzielczości uzyskiwanych map aktywacji, a tym samym ograniczania złożoności całego modelu. Inną stosunkowo nową koncepcją, znajdującą w szczególności zastosowanie w modelach, w których wartości wyjściowe mają charakter gęstej siatki predykcyjnej (np. w zadaniach semantycznej segmentacji obrazów), jest wykorzystanie splotu z dylatacją **(Chen et al., 2016)** (występującego zamiennie pod terminami *atrous convolution* lub *dilated convolution*), przedstawionego na rysunku 2.8.

Pooling

Odmiennym sposobem redukowania liczby przetwarzanych aktywacji jest operacja grupowania (*pooling*). Polega ona na agregacji informacji zawartej w polu postrzegania (oknie) neuronu za pomocą zbiorczej statystyki - najczęściej funkcji maksimum lub średniej, ale sporadycznie w wykorzystaniu można spotkać też inne funkcje grupujące. W najprostszej

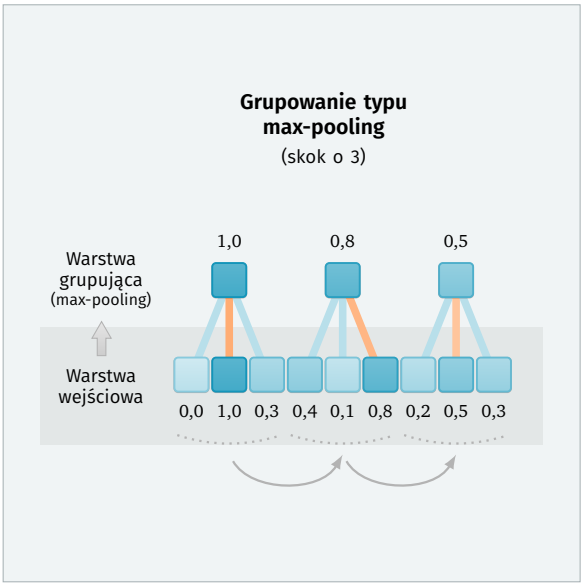


Rys. 2.9: Zobrazowanie funkcjonowania warstwy grupującej przy translacji danych wejściowych. Kolorem pomarańczowym zaznaczone zostały połączenia do maksymalnych wartości aktywacji występujących w polu postrzegania określonego neuronu grupującego.

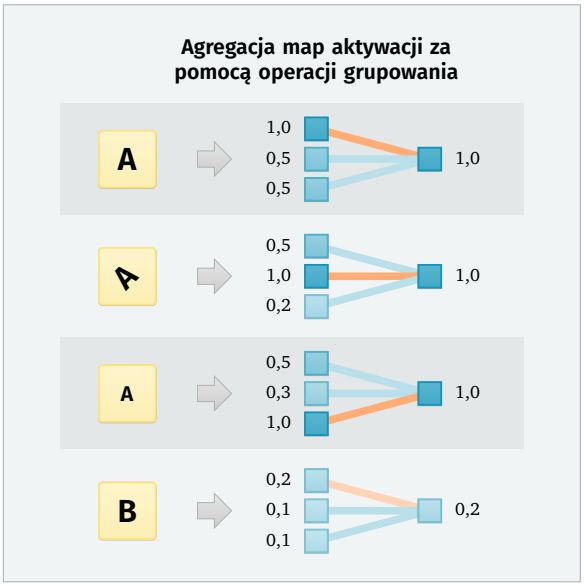
postaci warstwa grupowania wprowadza niezmienność w zakresie niewielkich translacji danych wejściowych tak, że dla małych przesunięć wartości aktywacji za warstwą grupującą pozostają w większości niezmienione, co ilustruje rysunek 2.9.

Najczęściej spotykaną odmianą poolingu jest przedstawiony na rysunku 2.10 wariant ze skokiem większym niż 1 (zazwyczaj równym rozmiarowi okna, jeżeli nie zaznaczono inaczej). Na podobnej zasadzie jak skok warstwy splotowej, modyfikacja ta pozwala znacząco ograniczyć rozmiar tworzonego modelu. Redukcja ta następuje w tym przypadku jednak już po etapie wyznaczenia aktywacji warstwy poprzedzającej z pełną rozdzielczością. Zmniejszenie rozmiaru wyjścia dokonuje się więc nie poprzez całkowite odrzucenie pewnej części informacji, jak w przypadku wariantu warstwy splotowej ze skokiem, ale poprzez jej uśrednienie za pomocą jednej z wymienionych funkcji agregujących.

Rzadziej spotykanym zastosowaniem grupowania jest agregacja dokonywana nie na poziomie przestrzennym, ale względem kanałów (generowanych map aktywacji), która ułatwia uczenie reprezentacji odpornych na deformacje. Koncepcja ta zakłada, że dla występujących w danych uczących zróżnicowanych wystąpień tej samej klasy semantycznej, poszczególne neurony w części detekcyjnej będą reagowały z różną siłą, specjalizując się przykładowo w wykrywaniu wariantów zrotowanych czy występujących w innej skali. Zazwyczaj na dalszym etapie przetwarzania fakt wystąpienia konkretnego wariantu nie jest jednak tak ważny, jak określenie klasy obiektu (np. że była to konkretnie litera „*A*”). Neuron grupujący odpowiedzi wielu neuronów niższego rzędu pozwala w takim przypadku na zagregowanie tego typu szczegółowej informacji w pojedynczą wartość bliższą znaczeniu semantycznemu. Działanie to obrazuje rysunek 2.11.



Rys. 2.10: Pooling ze skokiem.

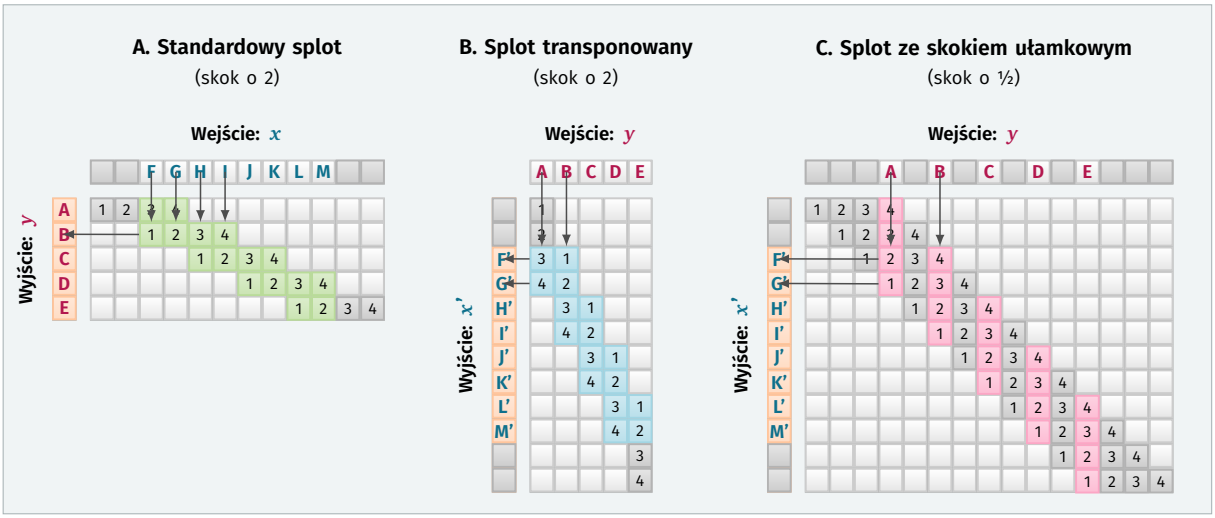


Rys. 2.11: Pooling po mapach aktywacji.

Splot transponowany

Warstwy splotowe i grupujące stanowią trzon większości wykorzystywanych w praktyce architektur splotowych i właściwie ich omówienie wystarczałoby do opisania przedstawianych w dalszej części rozprawy modeli. Dla kompletności opisu warto jednak bardzo zwięźle nadmienić, że trzecim modułem spotykanym w architekturach splotowych, szczególnie tych omawianych w najświeższych artykułach, jest operacja splotu transponowanego lub splotu ze skokiem ułamkowym (*transposed convolution* lub *fractional stride convolution*), często mylnie nazywana w skrócie rozplotem (dekonwolucją) **(Shi et al., 2016)**.

Termin rozplotu, zdefiniowany formalnie w kontekście przetwarzania sygnałów, wskazywałby na operację odwrotną do splotu. Celem przedstawionego na rysunku 2.12 splotu transponowanego nie jest jednak dokładne odwrócenie przetworzenia w kontekście uzyskania tych samych wartości liczbowych wejścia $\boldsymbol{x}$, ale taka zmiana kierunku projekcji, aby możliwe stało się przejście z reprezentacji o mniejszym rozmiarze (wektor $\boldsymbol{y}$) z powrotem do reprezentacji dłuższej (wektor $\boldsymbol{x}$). Co istotne, wagi warstwy standardowej i transponowanej nie muszą być ze sobą w żaden sposób powiązane. Zazwyczaj dalszy proces uczenia zakłada co prawda, że celem wprowadzenia splotu transponowanego jest takie ustalenie wag, aby aproksymowały przekształcenie odwrotne, ale nie jest to warunek konieczny. Aspekt, na który warto również zwrócić uwagę, to fakt, że zwizualizowane na rysunku 2.12.B przetworzenie formalnie nie jest splotem, dlatego często przedstawia się je jako równoważną operację splotu przeprowadzanego na danych z odpowiednimi lukami wypełnianymi wartościami zerowymi (rys. 2.12.C).



Rys. 2.12: Porównanie operacji splotu, splotu transponowanego i splotu ze skokiem ułamkowym. Cyframi zaznaczone zostały indeksy wag. Adaptacja własna na podstawie (Shi et al., 2016).

Warstwy wykorzystujące splot transponowany znajdują zastosowanie między innymi w rodzinach modeli określanych jako autokodery splotowe (*convolutional autoencoders*). W modelach tych oczekiwane wyjście jest równoważne z wejściem, a celem modelu jest możliwie wierna rekonstrukcja danych wejściowych przy założonych ograniczeniach architektury (najczęściej model zawiera przewężenie wymuszające przedstawienie zawartych informacji za pomocą odpowiednio skompresowanej reprezentacji).

Inną grupą modeli, w których pojawia się splot transponowany, są sieci generatywne z adwersarzem (*generative adversarial networks*) (Radford et al., 2015) i autokodery wariacyjne (*variational autoencoders*) (Pu et al., 2016). Koncepcje te zyskują w ostatnim czasie popularność przede wszystkim w związku z uzyskiwanymi przez nie interesującymi wynikami empirycznymi w zakresie generowania treści obrazowych. Nieco szersze omówienie sieci generatywnych z adwersarzem zostało zawarte w podrozdziale 2.3.7.

Splot 1 × 1

Osobliwą operacją, której należy się jeszcze słowo komentarza ze względu na jej mocno nieintuicyjny charakter, jest splot z rozmiarem filtra 1 × 1. Wariant ten jest o tyle nietypowy, że jedną z głównych zalet warstw splotowych jest odporne na przesunięcia reagowanie na określone wzorce, które mają pewną lokalną strukturę przestrzenną. W przypadku splotu 1 × 1 aspekt ten jest ignorowany, gdyż każdy piksel wejściowy przetwarzany jest niezależnie i przyporządkowywany jest mu dokładnie jeden piksel wyjściowy. Sensowność takiej operacji wynika jednak z dodatkowego wymiaru danych wejściowych wprowadzanego przez użycie w poprzedzającej warstwie wielu filtrów splotowych generujących odrębne mapy aktywacji (kanały), które są następnie sumowane z wagami określonymi przez filtr 1 × 1. Przekształcenie takie uzupełniane jest zazwyczaj o nieliniową funkcję aktywacji.

Wykorzystanie warstw splotowych typu 1×1 spełnia dwa główne cele. Po pierwsze, pozwalają one na redukcję wymiarowości danych generowanych przez warstwy zawierające dużą liczbę filtrów splotowych. Umieszczony po takich warstwach splot typu 1×1 zapewnia zmniejszenie liczby map aktywacji dostarczanych do dalszego przetwarzania, jednocześnie nie zmieniając rozmiaru samych map. Drugim aspektem jest wprowadzanie tą drogą kolejnych nieliniowości umożliwiających transformację generowanych cech. W tym kontekście decyzja o wykorzystaniu warstwy splotowej 1×1 jest odpowiednikiem rozbudowywania perceptronu o dodatkowe warstwy ukryte.

2.2 Modele głębokie

2.2.1 Deep learning - zmiana paradygmatu czy kwestia nazewnictwa?

Wspominany już w poprzedzających podrozdziałach model LeCuna et al. (1989) składał się z kilku warstw ukrytych. Chociaż w kolejnych latach udawało się zastosować inne modele splotowe zawierające większą liczbę warstw, w tym omawiany w dalszej części LeNet-5 (LeCun et al., 1998b), to przez długi czas przeważała opinia, że sztuczne sieci neuronowe składające się z wielu warstw są trudne do uczenia. Twierdzenie to dotyczyło szczególnie wielowarstwowych perceptronów zbudowanych z warstw gęsto połączonych. Co więcej, wraz z upowszechnieniem się rozwiązań opierających się na maszynie wektorów nośnych (*support vector machine*), które uzyskiwały dużo lepsze rezultaty empiryczne w zadaniach klasyfikacji, uczenie architektur głębokich wydawało się niezbyt obiecującą ścieżką.

Krokiem w nowym kierunku była przełomowa praca Hintona i Salakhutdinova (2006), w której autorom udało się skutecznie wytrenować głęboki model poprzez wyjście od procedury uczenia bez nadzoru, w iteracyjny sposób budując coraz głębszy model składający się z ograniczonych maszyn Boltzmanna (*restricted Boltzmann machines*). Parametry tak uzyskanego modelu były następnie wykorzystywane jako punkt inicjalizacji wag dla analogicznego modelu uczonego już w pełni nadzorowany sposób. Mimo że metoda wstępnego przyuczania sieci (*unsupervised pre-training*) z czasem straciła na znaczeniu, to w tym przełomowym momencie pokazała, że uczenie modeli głębokich jest problemem optymalizacyjnym, z którym można sobie przy odpowiednim podejściu realnie poradzić.

Mniej więcej w tym właśnie czasie uczenie modeli o głębokich architekturach zaczęło się wyodrębniać jako obszar uczenia maszynowego określany mianem głębokiego uczenia (*deep learning*). Kwestią subiektywnej oceny jest określenie, na ile był to zabieg mający na celu próbę sztucznego odcięcia się od dotychczasowej opinii, którą sztuczne sieci neuronowe wyrobiły sobie w poprzedzających latach, a na ile modele głębokie były rzeczywiście kompletnie nową jakością. Niewątpliwie architektury te szybko stały się bezkonkurencyjne pod kątem uzyskiwanych wyników empirycznych i z tym faktem trudno polemizować. Czy

jednak zmiany na poziomie fundamentalnym miały charakter bardziej ewolucyjny czy rewolucyjny? Tu już trudniej o jednoznaczną odpowiedź. Przywołując uwagi przedstawione w tej kwestii przez Hintona **(Kurenkov, 2015b)** oraz Glorota i Bengio **(2010)**, nagły postęp w tej dziedzinie możliwy był w związku z wystąpieniem kilku uzupełniających się efektów:

- w znaczący sposób zwiększyły się rozmiary dostępnych etykietowanych baz danych (przykładowo, pełny zbiór *ImageNet* zawiera kilkanaście milionów obrazów),
- równocześnie istotnie wzrosły dostępne moce obliczeniowe, w szczególności w wyniku zastosowania procesorów graficznych do obliczeń równoległych,
- wykorzystywane dotychczas funkcje aktywacji zastąpiono innymi ich rodzajami,
- zaczęto zwracać dużo większą uwagę na kwestię odpowiedniej inicjalizacji modelu,
- pojawiły się nowe sposoby regularyzacji i normalizacji sieci,
- proces uczenia gradientowego rozwinięto przez wprowadzenie metod adaptacyjnych.

Pojawiające się po drodze liczne propozycje zmian w zakresie stosowanych metod konstruowania i uczenia sieci były zazwyczaj rozwiązaniami komplementarnymi, ale zdarzały się też przypadki, gdzie nowe pomysły stanowiły konkurencyjne próby poradzenia sobie z komplikacjami wynikającymi z uczenia głębokich modeli na dużych zbiorach danych. Stąd też część koncepcji nie przechodziła próby czasu i po stosunkowo krótkim okresie stosowania bez echa ustępowała miejsca nowszym podejściom. Ogólna idea pozostawała jednak wspólna – jak uzyskać model sieci neuronowej, który w czasie uczenia będzie posiadał odpowiednią ekspresyjność do poradzenia sobie z coraz bardziej skomplikowanymi problemami praktycznymi, będzie w stanie w odpowiedni sposób rozpropagować sygnał uczący na wszystkie warstwy modelu unikając problemu zanikania lub nadmiernego rozrostu wartości gradientu, a jednocześnie też zachowa odpowiednią zdolność do generalizacji dla nowych danych?

Kolejne cztery podrozdziały zawierają krótkie omówienie wybranych koncepcji, które są próbą odpowiedzi na tak postawione pytanie. Chociaż z dzisiejszej perspektywy spotyka się je już powszechnie w modelach zaliczanych do obszaru *deep learning*, to można je uznać za pewnego rodzaju nowość w stosunku do rozwiązań stosowanych przed wyodrębnieniem się tej dziedziny. Kwestia oceny, w jakim stopniu znaczenie tych zmian uzasadnia powstanie tej wyraźnej cezury, jest natomiast na tyle subiektywna, że zostanie pozostawiona czytelnikowi.

2.2.2 Funkcje aktywacji neuronów

Tradycyjnie wykorzystywaną funkcją aktywacji w modelach perceptronu wielowarstwowego była omawiana w podrozdziale 2.1.2 **funkcja logistyczna** postaci:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.23}$$

Funkcja ta generuje na wyjściu wartość z zakresu $(0,1)$. Alternatywę dla niej stanowi **tangens hiperboliczny**, będący odpowiednim przeskalowaniem funkcji logistycznej do przedziału wartości wyjściowych $(-1,1)$:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{2e^x - e^x - e^{-x}}{e^x + e^{-x}} = \frac{2e^x}{e^x + e^{-x}} - 1 = \frac{2}{1+e^{-2x}} - 1 \ = 2\sigma(2x) - 1 \tag{2.24}$$

Posiada on tę przydatną cechę, że w zakresie argumentów bliskich 0 funkcja ta jest zbliżona do przekształcenia tożsamościowego - dla odpowiednio małych wartości aktywacji i wag uczenie sieci przypomina uczenie modelu liniowego. Co więcej, w przeciwieństwie do funkcji logistycznej, większe jest prawdopodobieństwo, że średnia wartość aktywacji warstwy, stanowiącej wejście dla warstwy kolejnej, będzie oscylować wokół 0. Zdaniem LeCuna et al. (1998) oraz Glorota i Bengio (2010) oznacza to w praktyce, że zbieżność procesu uczenia sieci wykorzystujących tangens hiperboliczny następuje zazwyczaj szybciej niż w przypadku zastosowania standardowej funkcji logistycznej, chociaż prawdą jest, że na poziomie teoretycznym zachowanie neuronów z logistyczną funkcją aktywacji jest bliższe modelom biologicznym.

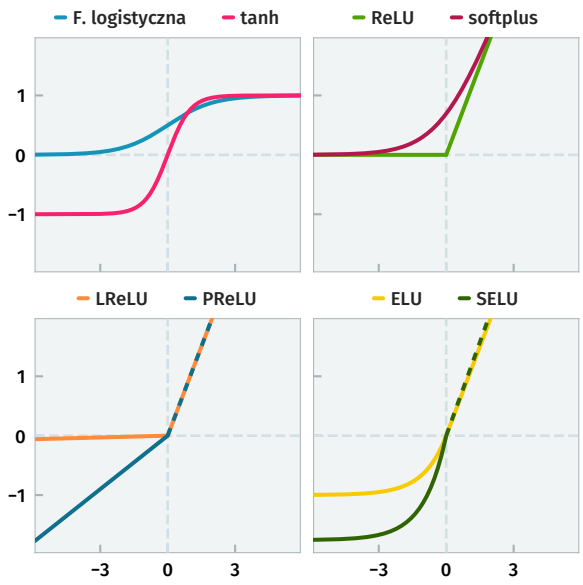
Niestety, obydwie z omawianych funkcji posiadają te samą właściwość, która okazuje się mocno niepożądana w uczeniu modeli głębokich zawierających dużą liczbę neuronów – dla wartości argumentów oddalonych od 0 dosyć szybko dochodzi do nasycenia, skutkującego zanikaniem gradientu. Uczenie staje się trudne po wejściu w ten obszar, gdyż duże zmiany parametrów w niewielkim stopniu wpływają na wartości aktywacji otrzymywane w poszczególnych warstwach.

W odpowiedzi na tę kwestię sigmoidalne funkcje aktywacji zostały zastąpione[1] przez neurony określane jako **ReLU** (*rectified linear units*) (Glorot & Bengio, 2010), których funkcja aktywacji, przedstawiona na rysunku 2.13, przybiera postać:

$$\varphi(x)_{\text{ReLU}} = x^+ = \max(0, x) \tag{2.25}$$

[1] Mowa tu o kontekście warstw splotowych i gęsto połączonych. Sigmoidalne funkcje aktywacji spotyka się dalej w modelach rekurencyjnych i w warstwach wyjścia.

Funkcja ta jest w istocie funkcją tożsamościową dla dodatnich argumentów, co powoduje, że jej pochodna w obszarze aktywnym przez cały czas nie tylko nie zanika, ale przyjmuje stałą wartość 1. Pozwala to na przyspieszenie obliczeń przez unikanie względnie kosztownych operacji wyliczania wartości funkcji eksponencjalnej. Przy skali uzyskiwanej w ramach uczenia głębokiej sieci na dużym zbiorze danych może się to okazać zauważalną oszczędnością czasową. Usprawnienie uczenia przy wykorzystaniu neuronów typu *ReLU* nie ogranicza się jednak tylko do kwestii mniejszej złożoności obliczeniowej, ale też szybszego zbiegania procesu uczenia liczonego liczbą iteracji wykonanych na zbiorze uczącym. W pracy Krizhevsky'ego et al. (2012) autorzy pokazują, że przy tej samej architekturze splotowej zamiana z tangensu hiperbolicznego na ReLU powoduje sześciokrotne zmniejszenie liczby epok potrzebnych do uzyskania takiego samego błędu uczenia.



Rys. 2.13: Porównanie funkcji aktywacji.

Wprowadzenie ReLU ma jeszcze kilka innych zalet. Zerowa odpowiedź dla ujemnych argumentów powoduje, że neurony te są bliższe rzeczywistemu funkcjonowaniu neuronów organicznych poprzez wyraźne rozróżnienie stanu pobudzonego i niepobudzonego. Występowanie dla znaczącego obszaru wejściowego zerowej wartości na wyjściu powoduje, że aktywacja warstw zbudowanych z ReLU ma charakter rzadki. Glorot i Bengio (2010) podkreślają, że skutkuje to zdolnością sieci do rozplątywania informacji (*information disentangling*). W przypadku gęstych reprezentacji czynniki wyjaśniające zmienność występującą w danych są związane - każda pojedyncza zmiana na wejściu powoduje zmiany wielu wartości wyjścia. Dla reprezentacji rzadkich wpływ ten natomiast ogranicza się do pewnego podzbioru wyjściowego, co pozwala sieci na płynne dostosowanie długości stosowanej reprezentacji wyjściowej do konkretnego przykładu danych wejściowych.

Bardzo istotnym praktycznym skutkiem zastosowania neuronów typu ReLU jest brak konieczności wstępnego przyuczania, gdyż sieci wykorzystujące tę funkcję aktywacji udaje się z sukcesem uczyć z nadzorem w zwyczajny sposób (Glorot & Bengio, 2010). Rzeczywiście, z perspektywy ostatnich lat trzeba przyznać, że rozpowszechnienie się tej koncepcji bardzo uprościło uczenie modeli głębokich i wyznaczanie bardziej obiecującej inicjalizacji parametrów za pomocą metod nienadzorowanych straciło istotnie na znaczeniu.

Jakie problemy mogą w takim razie wynikać z zastosowania ReLU? Z pozoru może się wydawać, że taka postać funkcji aktywacji przeczy przedstawionym w podrozdziale 2.1.2 założeniom odnośnie różniczkowalności. Fakt, że pochodna nie jest określona w punkcie $x = 0$ ma jednak w praktyce niewielkie znaczenie, gdyż sytuacja, w której wartość bezpośrednio przed aktywacją przyjmuje dokładnie 0, zdarza się bardzo rzadko. Mając na uwadze występujące przybliżenia numeryczne, za wartość pochodnej można w takim wypadku przyjąć subpochodną ze zbioru $[0, 1]$.

Dużo większym mankamentem neuronów typu ReLU jest zerowa wartość pochodnej występująca dla ujemnych argumentów. Możliwe więc staje się doprowadzenie w wyniku dokonywanych aktualizacji wag do takiej sytuacji, w której dany neuron dla każdego przykładu ze zbioru trenującego będzie generował zerową wartość aktywacji. Jednocześnie całkowity zanik gradientu wyznaczanego względem wag neuronu powoduje w rezultacie, że neuron ten przestaje się uczyć, a jego odpowiedź jest zawsze zerowa niezależnie od wartości wejścia. Problem ten określa się mianem wymierających neuronów (*dead ReLU*) i spotyka się go zwłaszcza przy zbyt dużym tempie uczenia. Wyjście z takiego miejsca jest szczególnie trudne, gdy wymieranie dotknie znaczącą część neuronów w poszczególnych warstwach, co prowadzi często do nieoczekiwanej redukcji efektywnej wielkości sieci.

Próbą zaradzenia tego typu trudnościom jest zastosowanie funkcji aktywacji, które nie skutkują wyzerowaniem pochodnej dla wartości ujemnych. Neurony typu ***Leaky ReLU*** (*LReLU*) **(Maas et al., 2013)** wprowadzają niewielki stały spadek dla obszaru nieaktywnego:

$$\varphi(x)_{\mathrm{LReLU}} = \begin{cases} x & \text{dla } x > 0 \\ 0{,}01x & \text{dla } x \leq 0 \end{cases} \tag{2.26}$$

Jeszcze dalej idącą modyfikacją jest ReLU parametryczny (***parametric ReLU*** lub *PReLU*) **(He et al., 2015)**, w którym nachylenie części nieaktywnej jest parametrem zmienianym indywidualnie dla każdego neuronu w ramach procesu uczenia:

$$\varphi(x)_{\mathrm{PReLU}} = \begin{cases} x & \text{dla } x > 0 \\ \alpha x & \text{dla } x \leq 0 \end{cases} \tag{2.27}$$

W szczególnym przypadku, gdy $\alpha = -1$, to $\varphi(x) = |x|$. Funkcję tę, określaną mianem prostownika wartości bezwzględnej (*absolute value rectification*), Jarrett et al. **(2009)** wykorzystali wcześniej w obszarze rozpoznawania obrazów w celu wprowadzenia niezależności od polaryzacji oświetlenia (jasny kontur na ciemnym tle i ciemny kontur na jasnym tle).

Jeszcze jedną z podnoszonych wad ReLU jest nieujemna wartość aktywacji, która prowadzi do podobnej sytuacji, jak w przypadku omawianego zastąpienia funkcji logistycznej tangensem hiperbolicznym - średnia wartość aktywacji warstwy jest zawsze dodatnia, co

nie jest najbardziej pożądanym efektem. Zaproponowany przez Cleverta et al. (2015) wariant **ELU** (*Exponential Linear Units*) pozwala na przesunięcie średniej aktywacji bliżej zera, a jednocześnie w porównaniu do LReLU i PReLU ma charakter jednostronnie wysycający, co uodparnia go na występowanie szumu w zakresie nieaktywnym:

$$\varphi(x)_{\mathrm{ELU}} = \begin{cases} x & \text{dla } x > 0 \\ \alpha(e^x - 1) & \text{dla } x \leq 0 \end{cases} \tag{2.28}$$

Jednocześnie, zdaniem autorów tej koncepcji, zaletą ELU w porównaniu do opisywanych w dalszej części metod normalizacji (np. warstw typu *batch normalization*) jest istotnie mniejsza złożoność obliczeniowa. W praktycznych zastosowaniach wydaje się jednak, że warstwy normalizujące cieszą się dużo większą popularnością.

Inne propozycje, o których jeszcze warto wspomnieć to stosowany w autokoderach progowany ReLU (*thresholded ReLU*) (Konda et al., 2014), zaproponowane przez Goodfellowa et al. (2013) warstwy typu *maxout* czy gładka aproksymacja ReLU za pomocą funkcji *softplus*:

$$\zeta(x) = \log(1 + e^x) \tag{2.29}$$

Oceniając obecny stan głębokiego uczenia, wydaje się jednak, że chociaż publikacje wprowadzające te koncepcje wskazywały na empiryczne zalety zastosowania bardziej rozwiniętych form neuronów typu ReLU, to nie ma obecnie konsensusu co do tego, czy faktycznie korzyść z tytułu ich stosowania jest istotna i niezależna od rozważanego problemu. W praktyce większość publikacji ogranicza się więc do stosowania aktywacji typu ReLU, rzadziej LReLU lub PReLU. W ostatnim czasie Klambauer et al. (2017) zaproponowali modyfikację w postaci sieci samonormalizujących (*self-normalizing neural networks*), w których obszernie dowodzą, że aktywacje typu **SELU** (*scaled exponential linear units*)[2]:

$$\varphi(x)_{\mathrm{SELU}} = \begin{cases} \lambda x & \text{dla } x > 0 \\ \lambda\alpha(e^x - 1) & \text{dla } x \leq 0 \end{cases} \tag{2.30}$$

przy odpowiednio małej skali inicjalizowanych wag gwarantują istnienie górnej i dolnej granicy wariancji uzyskiwanych aktywacji, tym sposobem całkowicie eliminując problem zanikania lub nadmiernego rozrostu wartości gradientu. Jednocześnie też w przeprowadzanych eksperymentach autorom udało się uzyskać bardzo dobre wyniki przy wykorzystaniu głębokich sieci składających się wyłącznie z warstw w pełni połączonych – wynik o

[2] Parametry założone przez Klambauera et al. (2017) to: $\alpha \approx 1{,}6733$, $\lambda \approx 1{,}0507$.

tyle osobliwy, że rzeczywiście, co podkreślają autorzy, większość bardzo głębokich architektur ogranicza się dziś do modeli splotowych i rekurencyjnych. Na tę chwilę jest jednak zbyt wcześnie, aby stwierdzić, czy przedstawiane przez nich propozycje się upowszechnią.

Na sam koniec tego omówienia warto jeszcze przywołać funkcję *softmax*, która jest o tyle specyficzna, że opiera się na całej warstwie, przeskalowując wartości wektora wyjściowego do zakresu $[0, 1]$ tak, że suma aktywacji całej warstwy n neuronów wynosi 1:

$$\text{softmax}(\boldsymbol{z})_j = \frac{\exp(z_j)}{\sum_{k=1}^{n} \exp(z_k)} \tag{2.31}$$

Ta właściwość powoduje, że tego typu funkcja aktywacji sprawdza się w problemach klasyfikacji w ostatniej warstwie (warstwie wyjścia) jako reprezentacja prawdopodobieństwa, że dany przykład należy do jednej z n możliwych klas.

2.2.3 Inicjalizacja wag w modelach głębokich

Cytowana już wcześniej praca Hintona wyraźnie pokazała, jak istotna dla skutecznego uczenia modeli głębokich jest odpowiednia inicjalizacja parametrów. Goodfellow et al. **(2016)** wskazują na trzy aspekty w jakich dobór początkowych wag może mieć wpływ na końcowy rezultat uczenia. Uczenie modeli głębokich jest ich zdaniem na tyle trudne, że w najgorszym przypadku nieodpowiedni punkt startowy może w ogóle zadecydować o braku zbieżności procesu uczenia. W bardziej pozytywnym scenariuszu, w którym taki problem nie wystąpi, sposób inicjalizacji parametrów może wydłużać czas potrzebny do zakończenia tego procesu, a także wpływać na końcową wartość funkcji kosztu. Trzecią kwestią, najbardziej dotkliwą z punktu widzenia dalszych rozważań prowadzonych w rozprawie, jest to, że błąd generalizacji może się istotnie zmieniać dla tej samej architektury i stałych hiperparametrów w zależności od konkretnej realizacji inicjalizacji losowej.

Chociaż okazało się że wpływ konkretnego sposobu inicjalizacji wag na efekt uczenia jest nie do zbagatelizowania, to obecna wiedza w tym obszarze ma raczej charakter pewnego rodzaju heurystyk i dobrych intuicji niż mocno sformalizowanych dowodów. Dokładne rozumienie tych zależności pozostaje póki co otwartym problemem.

Na pewno jednym z twardych wymagań w tej kwestii jest konieczność tzw. „łamania symetrii". Różne neurony tej samej warstwy powinny cechować się innymi wartościami początkowymi wag. W przeciwnym wypadku połączenie determinizmu algorytmu uczenia, funkcji kosztu i stosowanego modelu doprowadziłoby do sytuacji, w której w kolejnych iteracjach wszystkie neurony podlegałyby identycznym modyfikacjom. Z tego względu najczęstszym sposobem ustalania wartości początkowych wag jest inicjalizacja losowa rozkładem normalnym lub jednostajnym. Czasem też spotyka się rozkład normalny obcięty do wartości w zakresie dwóch odchyleń standardowych od średniej.

O ile w praktyce nie ma zbytnich wątpliwości, że stosowane wagi powinny skupiać się wokół zera, to już kwestia przyjmowanej wariancji rozkładu jest bardziej dyskusyjna. Najprostszym sposobem jest ustalenie dla wszystkich neuronów stałej skali o niewielkiej wartości. Goodfellow et al. (2016) podkreślają, że chociaż wprowadzany przez większe wartości wag silniejszy efekt „łamania symetrii" i braku zanikania sygnału uczącego mógłby być pożądany, to na niekorzyść tego podejścia przemawia szereg innych negatywnych efektów, takich jak: ryzyko nadmiernego wzrostu wartości gradientu (tzw. *exploding gradient*), zbytnia czułość wyjściowych predykcji na małe zmiany parametrów wejściowych, wchodzenie funkcji aktywacji w obszary nasycone, a także konieczność długiego dostosowywania wag jeżeli ich docelowy rozkład w punkcie zbieżności uczenia jest znacząco inny od tego założonego przy inicjalizacji.

Z tego powodu przez ostatnie lata w literaturze przedmiotu pojawiło się kilka powszechnie stosowanych heurystyk, opierających się na inicjalizacji wag uwzględniającej liczbę neuronów w poszczególnych warstwach. Metody te stanowią kompromis między próbą uzyskania stałej wariancji wartości aktywacji dla kolejnych warstw i odpowiednio stałej wariancji otrzymywanych gradientów. Jeżeli przez f_{in} oznaczymy liczbę połączeń wchodzących (*fan-in*) neuronu, a przez f_{out} liczbę połączeń wychodzących (*fan-out*), to Goodfellow et al. (2016) za LeCunem et al. (1998) proponują początkowy rozkład postaci:

$$\mathrm{W}_{i,j} \sim \mathcal{U}\left(-\sqrt{\frac{1}{f_{\text{in}}}}, \sqrt{\frac{1}{f_{\text{in}}}}\right) \tag{2.32}$$

Popularne implementacje[3] za sposób określany jako ***LeCun uniform*** rozumieją jednak:

$$\mathrm{W}_{i,j} \sim \mathcal{U}\left(-\sqrt{\frac{3}{f_{\text{in}}}}, \sqrt{\frac{3}{f_{\text{in}}}}\right) \tag{2.33}$$

Bardzo często wykorzystywane są również propozycje He et al. (2015) – ***He uniform***:

$$\mathrm{W}_{i,j} \sim \mathcal{U}\left(-\sqrt{\frac{6}{f_{\text{in}}}}, \sqrt{\frac{6}{f_{\text{in}}}}\right) \tag{2.34}$$

oraz Glorota i Bengio (2010) zamiennie określane jako ***Glorot*** lub ***Xavier uniform***:

$$\mathrm{W}_{i,j} \sim \mathcal{U}\left(-\sqrt{\frac{6}{f_{\text{in}} + f_{\text{out}}}}, \sqrt{\frac{6}{f_{\text{in}} + f_{\text{out}}}}\right) \tag{2.35}$$

[3] Mowa o *TensorFlow* i *Keras*: https://www.tensorflow.org/api_docs/python/tf/contrib/keras/initializers/lecun_uniform.

Dla inicjalizacji rozkładem normalnym $\mathcal{N}(\mu, \sigma^2)$ warianty te przyjmują odpowiednio postać:

$$\textbf{\textit{LeCun normal}}: \qquad \mathrm{W}_{i,j} \sim \mathcal{N}\left(0, \frac{1}{f_{\text{in}}}\right) \tag{2.36}$$

$$\textbf{\textit{He normal}}: \qquad \mathrm{W}_{i,j} \sim \mathcal{N}\left(0, \frac{2}{f_{\text{in}}}\right) \tag{2.37}$$

$$\textbf{\textit{Glorot (Xavier) normal}}: \qquad \mathrm{W}_{i,j} \sim \mathcal{N}\left(0, \frac{2}{f_{\text{in}} + f_{\text{out}}}\right) \tag{2.38}$$

Istotność doboru odpowiedniej skali wag podkreślają szczególnie Saxe et al. **(2013)** i Sussillo **(2014)**. Ich prace pokazują, że zwrócenie uwagi na ten aspekt pozwala na efektywne uczenie również w przypadku bardzo głębokich sieci. Do problemu można jednak podejść też od drugiej strony, nie przejmując się zbytnio samym szukaniem pieczołowicie dobranego punktu startowego dla wag modelu, ale starając się kontrolować kwestię propagacji sygnału gradientowego w nieco inny sposób. Przykładem tego typu postępowania jest zastosowanie opisywanych w podrozdziale 2.2.4 warstw normalizujących czy wprowadzenie koncepcji samonormalizacji na poziomie funkcji aktywacji, jak w przypadku cytowanej wcześniej pracy Klambauera et al. **(2017)**. Chociaż trudno w takim gąszczu możliwości wyłonić jednoznacznie faworyta, to najważniejsze jest zauważenie pewnej prawidłowości. Niezależnie od szczegółów obranego sposobu, cel jest zawsze podobny – stworzyć warunki odpowiednie dla uczenia metodą wstecznej propagacji błędu przy dużej liczbie pośrednich kroków (głębokości sieci).

Warto w tym miejscu zwrócić jeszcze uwagę na jedną kwestię wynikającą ze stosowania modeli, które stają się nie tylko coraz głębsze, ale też coraz szersze. Przytaczane dotychczas sposoby inicjalizacji skutkują ustalaniem w takim wypadku wag o bardzo małych wartościach. Zupełnie odmiennym podejściem jest zaproponowany przez Martensa **(2010)** sposób inicjalizacji rzadkiej (*sparse initialization*), w którym niezależnie od szerokości warstwy poprzedzającej, tylko k wag dla każdego neuronu przyjmuje wartość początkową różną od 0. Zazwyczaj za k przyjmuje się niewielką wartość – w przypadku cytowanej pracy było to 15. Pozwala to na zastosowanie większych wartości wag bez nadmiernego zwiększania sumy generowanej na wyjściu neuronu. Efekt ten jest szczególnie istotny w związku z wykorzystaniem przez Martensa **(2010)** wysycającej funkcji aktywacji w postaci tangensa hiperbolicznego. Natomiast z perspektywy obecnie stosowanych neuronów typu *ReLU* zaletą tego sposobu jest przede wszystkim duże początkowe zróżnicowanie ich odpowiedzi. Jak podkreślają Goodfellow et al. **(2016)**, przyjęcie tak mocno określonego rozkładu a priori może jednak powodować problemy i wydłużać proces uczenia w przypadku stosowania neuronów, które muszą działać w bardzo dokładnie skoordynowany sposób, np. neuronów typu *maxout*.

Mniejsza uwaga poświęcona została kwestii ustalania początkowych wartości wag obciążających (*biases*), gdyż, poza szczególnymi przypadkami, dobre rezultaty przynosi inicjalizacja wartościami zerowymi. Wartości większe od zera mogą być użyteczne w przypadku neuronów ReLU – gwarantując, że w początkowej fazie uczenia wszystkie neurony będą znajdować się w obszarze aktywnym (dodatnim). Jest to jeden ze sposobów radzenia sobie ze zbyt wczesnym efektem wymierania. Inną sytuacją, w której manipulacja początkowymi wagami obciążającymi może okazać się przydatna, jest klasyfikacja danych cechujących się bardzo nierównomiernym rozkładem prawdopodobieństwa występowania poszczególnych etykiet. Zmiana obciążeń neuronów w warstwie wyjścia pozwala wtedy na przybliżenie rozkładu początkowego generowanego przez sieć do tego oczekiwanego na podstawie zbioru uczącego, co może być szczególnie pomocne, gdy w jego ramach występują duże dysproporcje w częstości występowania poszczególnych klas.

Z pewnością kwestia ustalania początkowych wartości wag w modelach głębokich pozostaje cały czas otwartym problemem badawczym. Okres przejściowy, w którym wyłaniała się obecna forma nurtu określanego głębokim uczeniem, pokazał jednak, że jest to kwestia, której nie należy lekceważyć. Z drugiej strony, prawdą jest też, że proponowane w ostatnich latach różnego rodzaju modyfikacje w zakresie procesu uczenia, metod regularyzacji i normalizacji coraz bardziej uodparniają tworzone modele na problem złej inicjalizacji. Mimo to odwołanie się do sprawdzonych heurystyk w tym zakresie jest dalej sensowne z dwóch powodów – odpowiednia inicjalizacja wag nie jest w stanie zaszkodzić, a tylko pomóc w procesie uczenia sieci i jednocześnie wciąż duża część stosowanych modeli radzi sobie tylko w umiarkowanym stopniu z bardzo złymi punktami startowymi.

2.2.4 Metody regularyzacji

Jedną z charakterystycznych cech modeli głębokich jest wysoka ekspresyjność, która chociaż jest ich największą zaletą, to może się też okazać główną bolączką. Konstruowanie dużych sieci składających się z wielu połączonych ze sobą szerokich warstw wiąże się z występowaniem wolnych parametrów liczonych często w dziesiątkach milionów. W tej sytuacji nawet względnie duże zbiory uczące mogą się okazać niewystarczające, bardzo szybko prowadząc do przeuczenia (*overfitting*) sieci, które objawia się idealnym dopasowaniem do danych występujących w zbiorze uczącym, połączonym ze spadkiem jej zdolności do generalizacji. Problem ten jeszcze bardziej potęguje się w przypadku wykorzystywania niewielkich zbiorów danych. Niestety, poza pojedynczym wyjątkiem w postaci udostępnionego w 2017 roku przez Google *AudioSetu*[4], do takiej grupy należałoby raczej zaliczyć wszystkie etykietowane zbiory nagrań dostępne w zadaniach klasyfikacji dźwięku.

[4] Krótkie omówienie tego zbioru danych zawarte zostało w podrozdziale 3.3.2.

Próbą odpowiedzi na ten problem jest zastosowanie różnego rodzaju metod regularyzujących, które starają się odgórnie narzucić pewną strukturę modelu lub, rozumiejąc ten obszar szerzej, zmodyfikować jego funkcjonowanie w taki sposób, aby zmniejszyć uzyskiwany błąd generalizacji. Trzeba w tym miejscu podkreślić fakt, że wybór architektur o charakterze splotowym już sam w sobie jest formą bardzo silnej regularyzacji, mocno narzucającą założenia co do struktury stosowanej przez model reprezentacji danych. Niestety, w praktyce okazuje się, że ograniczenie to jest nadal niewystarczające i musi być uzupełniane przez zastosowanie innych technik, które zostaną pokrótce przedstawione w niniejszym podrozdziale.

Regularyzacja typu L_2 i L_1

W bazowym wariancie uczenie sieci polega na takim dostrajaniu jej parametrów, aby zminimalizować uzyskiwaną przez model wartość założonej funkcji kosztu. Do wartości tej może zostać jednak dołączony dodatkowy element kary, będący funkcją tych parametrów, który wyraża stopień zgodności wag z założeniami co do ich pożądanego kształtu. Zazwyczaj tym oczekiwanym stanem jest, aby wartości stosowanych wag nie były zbyt duże. Jeżeli więc przez $J(\boldsymbol{\theta}; \boldsymbol{X}, \mathring{\boldsymbol{y}})$ oznaczymy wartość funkcji kosztu osiąganą przez model o parametrach $\boldsymbol{\theta}$ na zbiorze danych $\boldsymbol{X}$ i etykietach $\mathring{\boldsymbol{y}}$, to w wyniku dodania kary normalizującej (*norm penalty*) $\Omega(\boldsymbol{\theta})$ otrzymujemy zregularyzowaną funkcję celu $\tilde{J}$:

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \mathring{\boldsymbol{y}}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \mathring{\boldsymbol{y}}) + \alpha\Omega(\boldsymbol{\theta}) \tag{2.39}$$

w której $\alpha \in [0, \infty)$ określa siłę wprowadzanej regularyzacji. Zazwyczaj przy określaniu kary ze zbioru uwzględnianych parametrów wyłącza się wagi obciążające. W związku z tym w dalszej części tego podrozdziału przez $\boldsymbol{w}$ oznaczany będzie wektor tylko tych wag, które podlegają regularyzacji. Przyjmując w takim wypadku karę postaci:

$$\Omega_{L_2}(\boldsymbol{\theta}) = \frac{1}{2}\left\|\boldsymbol{w}\right\|_2^2 = \frac{1}{2}\sqrt{\sum_i w_i^2}^2 = \frac{1}{2}\sum_i w_i^2 \tag{2.40}$$

uzyskujemy **regularyzację typu L_2** określaną jako rozpad lub zanikanie wag (*weight decay*). Poza obszarem maszynowego uczenia spotyka się ją również pod terminem regresji grzbietowej (*ridge regression*) lub regularyzacji Tikhonova. Wprowadzenie do funkcji kosztu tak sformułowanej kary ogranicza ustalanie się wag o dużych wartościach. Regularyzacja L_2 faworyzuje wykorzystanie wszystkich wag po trochu zamiast kilku w dużym stopniu, stąd tworzona z jej użyciem reprezentacja jest gęsta.

Nieco odmienny charakter ma **regularyzacja typu L_1**, w której kara normalizująca przybiera postać:

$$\Omega_{L_1}(\boldsymbol{\theta}) = \|\boldsymbol{w}\|_1 = \sum_i |w_i| \tag{2.41}$$

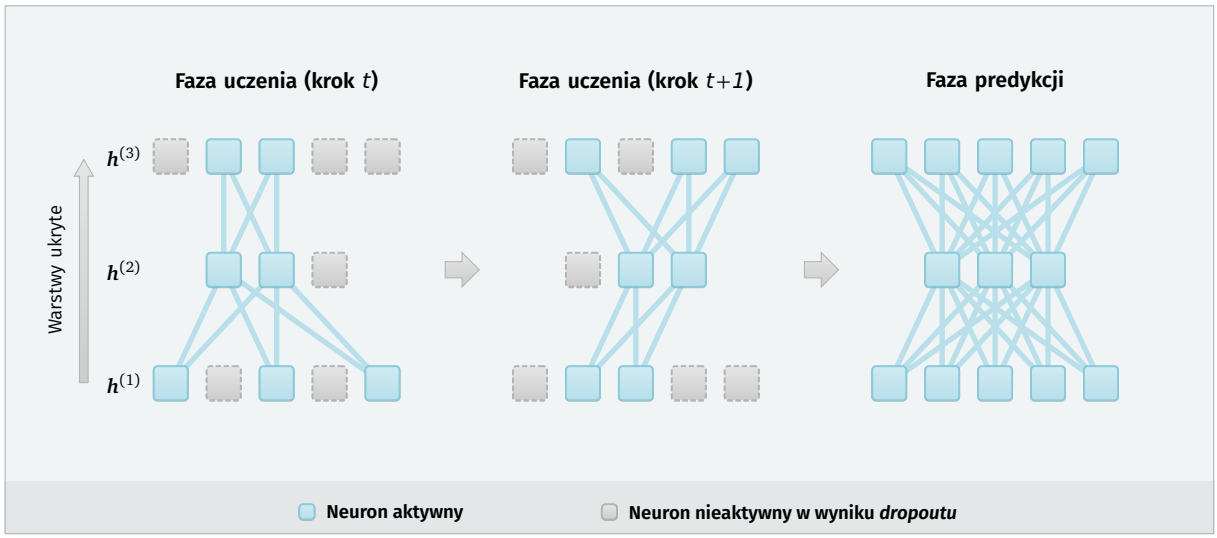
Chociaż celem regularyzacji typu L_1 jest również uzyskanie wag o odpowiednio małych wartościach, to w porównaniu do L_2 prowadzi ona do parametryzacji, w której wektor wag jest rzadki – przy odpowiednio dużej sile regularyzacji α część wag utrzymywana jest jako 0, natomiast te o największym znaczeniu są korygowane w istotnie mniejszym stopniu niż w przypadku L_2. Z tego powodu regularyzacja L_1 używana jest jako selektor atrybutów i wraz z neuronami typu ReLU pozwala na wprowadzenie do modelowania założenia, że pożądanym sposobem przedstawienia danych w ramach tworzonej sieci jest reprezentacja rzadka.

Kary normalizujące L_2 i L_1 w pośredni sposób ograniczają zakres wartości, jakie wagi mogą przyjmować w trakcie uczenia. Możliwe jest też nałożenie twardych warunków ograniczających, które bezpośrednio wyznaczają nieprzekraczalny próg c (*max norm*), powyżej którego wagi są odpowiednio przeskalowywane, np. tak że $\|\boldsymbol{w}\|_2^2 < c$. Efekt korygowania pojawia się w takim wypadku tylko w momencie wykroczenia wag poza dozwolony zakres, natomiast nie ma wpływu na kształtowanie się ich wartości poza tą sytuacją. Hinton et al. w pracy wprowadzającej technikę *dropout* (2012) wykorzystali właśnie twarde ograniczenia ustalane niezależnie dla każdego neuronu w warstwie ukrytej, uzasadniając tę decyzję możliwością dużo lepszej eksploracji przestrzeni rozwiązań poprzez zastosowanie bardzo dużego tempa uczenia, które wygasa dopiero z czasem. Wprowadzony warunek pozwolił w początkowym stadium uczenia na utrzymanie wag w ryzach niezależnie od wielkości zmian proponowanych przy tak dobranym tempie uczenia.

Dropout

Całkiem innym pomysłem radzenia sobie z przeuczeniem sieci jest zaproponowana w cytowanej już pracy Hintona et al. (2012) koncepcja określana jako ***dropout***. Chociaż może wydawać się nieskomplikowana ideowo, to w praktyce okazała się wyjątkowo efektywna. Co też istotne, *dropout* ma charakter komplementarny i bez większego problemu może być stosowany razem z innymi metodami, w tym z opisywaną już regularyzacją L_2 i L_1.

Zasada działania *dropoutu* polega na zobrazowanym na rysunku 2.14 losowym usuwaniu części neuronów w trakcie uczenia sieci. Motywacją takiego postępowania jest chęć „oduczenia” sieci bazowania na bardzo szczegółowych zależnościach między pobliskimi neuronami (*spurious co-adaptations*), które do swojej użyteczności potrzebują współdziałania wielu innych detektorów cech. Intencją autorów tej metody jest uczenie neuronów,



Rys. 2.14: Schemat uczenia z wykorzystaniem *dropoutu*.

które będą generować cechy użyteczne w ogólnym kontekście – bez gwarancji, że będzie możliwe ich sparowanie z wartościami aktywacji konkretnie określonych innych neuronów.

W każdej iteracji uczenia z wykorzystaniem *dropoutu* wybierane są neurony, które w danym kroku zostaną odrzucone. Selekcja ta następuje z prawdopodobieństwem określanym przez hiperparametr p. W kolejnej iteracji postępowanie to powtarza się, ponownie wychodząc od modelu zawierającego wszystkie neurony. Model stosowany w fazie predykcji jest z kolei uśredniany w ten sposób, że choć posługuje się już wszystkimi neuronami, to wartości wag są odpowiednio przeskalowywane, aby uwzględnić większą w porównaniu do fazy uczenia efektywną liczbę wejść, na których operuje każdy neuron.

Postępowanie to może być rozumiane jako pewna analogia do *baggingu* (uczenia wielu wariantów modelu na podzbiorach i uśredniania ich predykcji), którego zastosowanie w pełnej formule jest jednak zbyt kosztowne obliczeniowo dla modeli głębokich o czasie uczenia liczonym często w dniach, a nawet tygodniach. Główną różnicą w przypadku *dropoutu* jest to, że generowane podsieci nie są całkowicie niezależnymi modelami, ale dzielą między sobą wspólny zbiór parametrów.

Dobór konkretnej wartości p jest mocno uzależniony od przyjętej architektury, wykorzystywanego zbioru danych uczących i innych hiperparametrów, stąd trudno jednoznacznie zaproponować rozwiązanie optymalne w tym zakresie. Przyjęcie zbyt dużego odsetka odrzucanych neuronów może skutkować bardzo powolnym uczeniem lub jego całkowitym zatrzymaniem. Wykorzystanie *dropoutu* może też prowadzić do niepożądanego tworzenia się neuronów redundantnych, które uczą się tylko najbardziej dyskryminujących cech w kilku zbliżonych kopiach – tak, aby przynajmniej jedna z nich była zawsze dostępna. W takiej sytuacji bardzo maleje efektywna wielkość uczonej sieci w porównaniu do tej założonej nominalnie, stąd przy bardzo agresywnym stosowaniu *dropoutu* konieczne może się okazać zastosowanie modelu szerszego niż pierwotnie przewidywany.

W tym miejscu warto jeszcze zwrócić uwagę na możliwość wystąpienia niejednoznaczności w samym rozumieniu parametru p. W ramach literatury przedmiotu można spotkać go zarówno jako określenie części neuronów pozostawianych w czasie uczenia sieci, jak i odwrotnie – jako prawdopodobieństwo ich odrzucenia[5]. Być może powstanie tego problemu jest związane z tym, że w pracy Hintona et al. (2012) wprowadzającej tę metodę, autorzy usuwali dokładnie połowę neuronów w każdym kroku, stąd rozróżnienie to nie miało większego znaczenia. W ramach dalszej części rozprawy prawdopodobieństwo p w odniesieniu do *dropoutu* będzie używane w tym drugim znaczeniu, rozumianym jako ta część neuronów, która jest odrzucana.

Normalizacja

Chociaż ustalenie odpowiednich wag początkowych jest pomocne w uzyskiwaniu odpowiedniej propagacji sygnału uczącego w głębokich sieciach, to nie gwarantuje ono, że wraz z postępującym uczeniem sieci te korzystne właściwości nie zostaną utracone. Próba bardziej trwałego wymuszenia stałych rozkładów wartości wejść poszczególnych warstw była motywacją zaproponowanej przez Ioffe'a i Szegedy'ego (2015) metody normalizacji partiami (***batch normalization***). Autorzy tego pomysłu zauważają, ze przy standardowym konstruowaniu głębokich sieci neuronowych wraz z uczeniem danej warstwy następuje ciągła zmiana rozkładu generowanych przez nią wartości wyjściowych. Jest to równoznaczne ze zmianą rozkładu wejść dla kolejnej następującej po niej warstwy, która dużą część uczenia musi z tego powodu poświęcić na dostosowywanie się do zachodzącego wewnątrz modelu przesunięcia (*internal covariate shift*). Efekt ten może się szczególnie potęgować na zasadzie łańcuchowej w przypadku występowania długich sekwencji warstw spotykanych w modelach głębokich. Skutkuje to koniecznością ustalania odpowiednio niskiego tempa uczenia i zważania na podnoszoną już kwestię inicjalizacji.

Ioffe i Szegedy (2015) sugerują by w związku z tym kwestię normalizacji odpowiedzi włączyć bezpośrednio do modelu poprzez wymuszenie odpowiedniej korekty dla każdej mini-partii (*mini-batch*) zbioru uczącego. Zaproponowane rozwiązanie pozwoliło autorom na uzyskanie dokładności klasyfikacji *ImageNet* porównywalnej do innych przodujących modeli rozpoznawania obrazu przy czternastokrotnym ograniczeniu liczby iteracji na zbiorze uczącym. Nie jest więc wielkim zaskoczeniem, że technika ta szybko stała się bardzo popularna, o czym świadczy chociażby liczba cytowań, która w ciągu dwóch lat od publikacji przekroczyła dwa tysiące.

[5] Dla przykładu, implementacja w popularnej bibliotece *Keras* jako wymagany parametr dla *dropoutu* rozumie prawdopodobieństwo odrzucenia neuronu. Nieco starszy *pylearn2* w analogicznej sytuacji posługuje się prawdopodobieństwem włączenia neuronu do uczenia.

Metoda normalizacji partiami dokonuje niezależnej standaryzacji każdej wartości aktywacji $a_j^{(l)[i]}$ (w dalszej części oznaczenie warstwy $^{(l)}$ będzie pomijane dla większej przejrzystości) osiąganej dla konkretnego przykładu uczącego i, w oparciu o statystyki wyznaczane na bazie partii b takich przykładów $\mathbb{B} = \{\boldsymbol{x}^{[1]}, \ldots, \boldsymbol{x}^{[b]}\}$ w ten sposób, że:

$$\mu_{\mathbb{B}_j} = \frac{1}{b}\sum_{i=1}^{b} a_j^{[i]} \tag{2.42}$$

$$\sigma^2_{\mathbb{B}_j} = \frac{1}{b}\sum_{i=1}^{b}(a_j^{[i]} - \mu_{\mathbb{B}_j})^2 \tag{2.43}$$

$$\hat{a}_j^{[i]} = \frac{a_j^{[i]} - \mu_{\mathbb{B}_j}}{\sqrt{\sigma^2_{\mathbb{B}_j} + \epsilon}} \tag{2.44}$$

gdzie ϵ jest niewielką wartością dodawaną dla celów stabilności numerycznej. W wyniku tej operacji wartość oczekiwana skorygowanej aktywacji wynosi 0 z wariancją 1. Tak uzyskany rozkład nie musi być optymalny z punktu widzenia efektywności uczenia, więc dodatkowo wprowadza się parametry skalujące γ i β, które są dostrajane przez sieć i pozwalają jej na wykorzystanie szerszego spektrum reprezentacji:

$$\varphi_{\mathrm{BN}}(\hat{a}_j^{[i]}) = \gamma_j \hat{a}_j^{[i]} + \beta_j \tag{2.45}$$

Wprowadzenie normalizacji partiami pozwala na użycie większej wartości tempa uczenia. Efekt ten jest szczególnie widoczny w przypadku głębokich sieci z sigmoidalną funkcją aktywacji, w których tego typu normalizacja ułatwia utrzymanie neuronów w obszarze nienasyconym. Innym pozytywnym skutkiem jest uodpornienie sieci na zmianę skali parametrów. Ioffe i Szegedy (2015) wskazują też, że koncepcja ta ma podobne właściwości regularyzujące jak *dropout*, gdyż wartości aktywacji są korygowane za każdym razem na podstawie statystyk z losowo tworzonej mini-partii. Podobny pomysł proponowali też wcześniej Gülçehre i Bengio (2013) w postaci warstw standaryzujących (*standarization layer*), jednak jedną z różnic w tych podejściach jest miejsce dokonywanej standaryzacji – po zastosowaniu nieliniowej funkcji aktywacji (warstwy standaryzujące) lub bezpośrednio przed nią (normalizacja partiami).

Trudności z normalizacją partiami pojawiają się w sytuacji, gdy liczba przykładów uczących w partii jest bardzo mała lub nie są one niezależne od siebie. Renormalizacja partiami (***batch renormalization***) jest rozszerzeniem zaproponowanym przez Ioffe'a (2017), które ma na celu poradzenie sobie z takimi przypadkami. Ba et al. (2016) dodatkowo podkreślają, że problemem normalizacji partiami jest nie tylko zależność od rozmiaru partii, ale też trudność w jej zastosowaniu do sieci rekurencyjnych. W związku z tym przedstawili

normalizację warstw (***layer normalization***) poprzez wyliczanie statystyk korygujących na bazie wszystkich aktywacji uzyskiwanych przez daną warstwę dla pojedynczego przykładu uczącego (zamiast dla każdego neuronu indywidualnie, ale na podstawie całej mini-partii). Pozwala to na łatwe zastosowanie w modelach rekurencyjnych i wykonywanie identycznych obliczeń w fazie uczenia i predykcji. Autorzy tej techniki przyznają jednak, że normalizacja warstwami zakłada, że poszczególne neurony w warstwie mają mniej więcej podobny wkład w łączną predykcję, stąd normalizacja ta nie jest adekwatna dla warstw splotowych, w których stan taki zazwyczaj nie zachodzi.

Salimans i Kingma (2016) wprowadzili z kolei reparametryzację określaną jako normalizacja wag (***weight normalization***), która również nie tworzy zależności między przykładami uczącymi i nadaje się do modeli typu *long short-term memory* (*LSTM*) czy zastosowań w obrębie głębokiego uczenia ze wzmocnieniem (*deep reinforcement learning*) i modeli generatywnych. Normalizacja wag dokonuje rozdzielenia długości wektorów wag od ich kierunku w ten sposób, że dla każdego neuronu wektor wag $\boldsymbol{w}$ zastępowany jest przez:

$$\boldsymbol{w} = \frac{g}{\left\|\boldsymbol{v}\right\|_2}\boldsymbol{v} \tag{2.46}$$

i następnie przeprowadza się optymalizację metodą spadku gradientu (*gradient descent*) względem parametrów $\boldsymbol{v}$ i g. Wcześniejsze rozwiązania, które sugerowały inne formy normalizacji wag, operowały cały czas na parametryzacji modelu bezpośrednio względem wektora $\boldsymbol{w}$. W tym wypadku zaproponowane rozdzielenie skali $\left\|\boldsymbol{w}\right\|_2 = g$ przyspiesza konwergencję i chociaż nie dorównuje osiąganą dokładnością modelom z normalizacją partiami, to jest mniej kosztowne obliczeniowo, zwłaszcza w przypadku sieci splotowych, w których liczba wag jest istotnie mniejsza od liczby uzyskiwanych aktywacji. Brak wprowadzanego szumu związanego z wyznaczaniem statystyk na losowo dobranych przykładach uczących pozwala też na uczenie w problemach, gdzie normalizacja partiami się nie sprawdza. Salimans i Kingma (2016) pokazują to na przykładzie rozszerzenia modelu głębokiej Q-sieci (*deep Q-network*) (Mnih et al., 2015) o część normalizacyjną i wykorzystania jej w problemie uczenia ze wzmocnieniem.

Krąg możliwych podejść nie zamyka się jednak wyłącznie na przedstawianych dotychczas technikach. Dla przykładu, Luo et al. (2017) wyszli w ostatnim czasie z pomysłem normalizacji cosinusowej (*cosine normalization*). Z drugiej strony, wykorzystywane dużo wcześniej rozwiązania normalizujące lokalną odpowiedź (*local response normalization*) (Jarrett et al., 2009) zostały już praktycznie całkowicie wyparte przez omawiane w tej części nowsze techniki. Widać więc, że kwestia normalizacji modeli głębokich jest obszarem aktywnego poszukiwania nowych rozwiązań. Na tę chwilę dominującym sposobem jest z pewnością metoda normalizacji partiami, która sprawdza się dosyć uniwersalnie w większości spotykanych problemów. Trudno jednak prognozować, jak długo taki stan się utrzyma.

Inne techniki

Uczenie maszynowe opiera się na budowaniu modeli na bazie prezentowanych przykładów uczących. Naturalnie więc większa różnorodność i liczebność zbioru uczącego wspomaga generalizację osiąganego rozwiązania. Niestety, zebranie odpowiednio dużej ilości wysokiej jakości etykietowanych danych jest często niemożliwe ze względów praktycznych. O ile omawiane dotychczas techniki starały się w takiej sytuacji zwiększyć prawdopodobieństwo uczenia się przez model rzeczywistego procesu różnicującego dane i ograniczyć zbytnie dopasowanie do szumu, to możliwe jest podejście do problemu od drugiej strony. Zamiast odgórnie ograniczać ekspresyjność modelu, spróbujmy mimo wszystko po prostu zwiększyć ilość danych, z którymi ma on szansę się zapoznać. W przypadku gdy pozyskanie dodatkowych rzeczywistych obserwacji nie jest możliwe, pewną pomocą jest stosowane często w praktyce syntetyczne powiększanie zbioru uczącego (***dataset augmentation***). Proces ten polega na generowaniu nowych danych trenujących poprzez poddawanie dostępnych w zbiorze uczącym przykładów różnego rodzaju transformacjom. Wprowadzane do uczenia deformacje mają na celu uodpornić sieć na te formy zróżnicowania danych, których spodziewamy się w fazie predykcji, np. rotacje, zmiany oświetlenia i skali wykrywanych obiektów. Niestety, wraz ze wzrostem skomplikowania tych transformacji może się okazać, że zdolność syntetycznego generowania tego typu przykładów może być tożsama z uzyskaniem odpowiedzi na bazowy problem klasyfikacji. Przykładowo, możliwość wytworzenia z jednego nagrania szczekania wariantów dla psów każdej innej rasy oznaczałaby, że właściwie znany jest fizyczny proces generujący, któremu można nadać semantyczną etykietę szczekania.

Innym sposobem radzenia sobie ze zbytnim dopasowaniem modelu do szumów występujących w danych jest wcześniejsze zatrzymanie procesu uczenia (***early stopping***). Chociaż rozwiązanie to nie należy do bardzo wyrafinowanych, to potrafi przynosić dobre rezultaty. Zatrzymanie może bazować na odgórnie ustalonej liczbie iteracji lub na monitorowaniu wyników uzyskiwanych na odrębnym podzbiorze walidacyjnym. Wcześniejsze zatrzymanie uczenia jest formą regularyzacji, która przesuwa uzyskiwane rozwiązanie z miejsca potencjalnie osiąganej zbieżności do punktu bliższego inicjalizacji modelu, stąd zakłada ono, że początkowy rozkład wag ma pewne cechy pożądane z punktu widzenia generalizacji. W przypadku modeli z odpowiednio silnym *dropoutem* ryzyko wyraźnego przeuczenia mocno maleje i ustalanie długości uczenia jest w pierwszej kolejności kwestią uwzględnienia korzyści i kosztów – na ile istotnie wydłużenie uczenia wpływa na poprawę końcowego modelu.

Często spotykaną metodą poprawiania zdolności generalizacji jest też tworzenie komitetów (***model ensembles***) składających się z kilku wariantów modelu lub jego różnych inicjalizacji losowych i uśrednianie generowanych predykcji. Sugeruje to, że uzyskiwane

rozwiązania są suboptymalne i w zależności od konkretnego punktu inicjalizacji czy stosowanej architektury mogą lepiej dostosowywać się do konkretnych aspektów analizowanych danych. Główną zaletą zastosowania komitetów jest dosyć uniwersalna poprawa uzyskiwanych wyników przy właściwie zerowym wysiłku koncepcyjnym. Z tego powodu często spotyka się je w różnego rodzaju konkursach i zadaniach typu „pobij benchmark" jako ostatni krok pozwalający choćby w niewielkim stopniu polepszyć końcowy rezultat. Zazwyczaj jednak nie są to różnice na tyle istotne jakościowo, by uzasadnić wdrożenie komitetów w systemach produkcyjnych, gdzie koszt wielokrotnego wyliczania wariantów predykcji dla pojedynczej obserwacji okazuje się być aspektem, którego nie da się pominąć.

Próbą uogólnienia koncepcji *dropoutu* i tworzenia komitetów jest zaproponowany przez Singha et al. **(2016)** ***swapout***, który „*dokonuje próbkowania z bogatego zbioru możliwych architektur, w tym dropoutu, stochastycznej głębokości i połączeń rezydualnych*". Podstawowa wersja *swapoutu* zakłada architekturę, w której neurony są gęsto połączone nie tylko z bezpośrednio poprzedzającą warstwą, ale również ze wszystkimi wcześniejszymi. Na podobnej zasadzie jak w *dropoucie*, w każdym kroku losowo usuwane są poszczególne połączenia, co pozwala na wytwarzanie się architektur z przeróżnymi schematami przeskoków (*skip connections*) i omawianych nieco dokładniej w podrozdziale 2.3.6 połączeń rezydualnych (*residual connections*). Chociaż pomysł takiego tworzenia całej rodziny sieci o współdzielonych parametrach jest z pewnością ciekawy, to wprowadza niestety dodatkową złożoność związaną z koniecznością wnioskowania stochastycznego. Próbkowanie wielu realizacji architektur wprowadza ten sam rodzaj problemów co zastosowanie dużych komitetów, stąd rozwiązanie to nadaje się tylko do niektórych sytuacji.

Z praktycznego punktu widzenia istotniejsze wydaje się zastosowanie pomysłów, które można zbiorczo określić jako uczenie przez transfer wiedzy (***transfer learning***). Podejście to opiera się na spostrzeżeniu, że głębokie modele tworzone w oparciu o odpowiednio liczne zbiory danych tworzą reprezentacje, które są na tyle ogólne, że można je zastosować również w innych sytuacjach, gdzie dostęp do danych uczących jest dużo bardziej ograniczony. Naturalnie najlepsze efekty przynosi zastosowanie w zadaniach odpowiednio zbliżonych, ale jak pokazali Razavian et al. **(2014)** na przykładzie wykorzystania sieci *OverFeat* **(Sermanet et al., 2013)** jako ekstraktora cech, nawet w problemach mocno odległych od bazowego, korzyści z takiego postępowania mogą być znaczące. Dogłębniej tę kwestię analizowali Yosinski et al. **(2014)**.

Wykorzystanie gotowych sieci splotowych wyuczonych na bardzo dużych zbiorach danych jest popularne również z innego powodu – odtworzenie od podstaw rozwiązań konkurencyjnych do najlepszych stosowanych obecnie modeli rozpoznawania obrazów może wymagać nawet kilku tygodni obliczeń. W związku z tym użycie już dostępnego ogólnego modelu i zaadaptowanie go do bardziej szczegółowego zadania jest nie tylko kwestią

poprawy zdolności generalizacji, ale też bardzo istotną oszczędnością czasową. Uczenie poprzez transfer wiedzy polega w takim wypadku na usunięciu ostatniej warstwy modelu ogólnego i wprowadzeniu niewielkiego klasyfikatora operującego na tak uzyskanym wyjściu. Możliwe jest również dostrajanie wag całego modelu (*fine-tuning*), traktując otrzymany wcześniej model po prostu jako punkt bardzo dobrej inicjalizacji wag.

Ciekawym kierunkiem badań są też próby pozyskiwania dodatkowej wiedzy z treści multimedialnych przez modele wizyjne w celu jej dalszego wykorzystania do uczenia modeli akustycznych **(Aytar et al., 2016; Arandjelović & Zisserman, 2017a,b)**.

2.2.5 Algorytmy optymalizacji

Po przygotowaniu zbioru danych, określeniu architektury sieci, sposobu inicjalizacji jej parametrów i regularyzacji, przychodzi czas na właściwy proces uczenia. W tym ostatnim etapie kluczową decyzją jest wybranie algorytmu, który posłuży do minimalizacji funkcji kosztu i ustalenie odpowiedniego tempa uczenia.

W problemach klasyfikacji typu 1 z N (predykcja pojedynczej etykiety z N możliwych kategorii), do których ogranicza się niniejsza rozprawa, najczęściej spotykaną funkcją kosztu jest średnia wartość odstępstwa prognozowanego rozkładu prawdopodobieństwa od rzeczywistych wartości etykiet mierzona entropią krzyżową (*categorical cross-entropy*). Przyjmując w dalszej części następujące oznaczenia:

$$\mathring{\boldsymbol{y}}^{[i]} = (\mathring{y}_1^{[i]}, \ldots, \mathring{y}_N^{[i]}) \quad - \quad \text{etykieta dla przykładu } i \text{ w kodowaniu 1 z } N \text{ (\textit{one-hot})}, \tag{2.47}$$

$$\mathring{\boldsymbol{Y}} \quad - \quad \text{macierz } M \times N \text{ etykiet zbioru uczącego } (\mathring{\boldsymbol{Y}}_{i,\bullet} = \mathring{\boldsymbol{y}}^{[i]}), \tag{2.48}$$

$$\boldsymbol{y}^{[i]} = f(\boldsymbol{x}^{[i]}; \boldsymbol{\theta}) \quad - \quad \text{predykcja } \boldsymbol{y}^{[i]} = (y_1^{[i]}, \ldots, y_N^{[i]}) \text{ dla przykładu } i \text{ generowana za pomocą modelu określonego parametrami } \boldsymbol{\theta}, \tag{2.49}$$

$$\mathcal{L}^{[i]} = \mathcal{L}(\boldsymbol{y}^{[i]}, \mathring{\boldsymbol{y}}^{[i]}) \quad - \quad \text{wartość funkcji kosztu dla pojedynczego przykładu } i, \tag{2.50}$$

$$J(\boldsymbol{\theta}; \boldsymbol{X}, \mathring{\boldsymbol{Y}}) \quad - \quad \text{łączna wartość funkcji kosztu modelu (dla } M \text{ przykładów uczących), } J(\boldsymbol{\theta}; \boldsymbol{X}, \mathring{\boldsymbol{Y}}) = \tfrac{1}{M} \textstyle\sum_{i=1}^{M} \mathcal{L}^{[i]}, \tag{2.51}$$

entropię krzyżową $\mathcal{L}_{\mathsf{cross-entropy}}$ dla przykładu i można przedstawić jako:

$$\mathcal{L}^{[i]}_{\mathsf{cross-entropy}} = -\sum_{j=1}^{N} \mathring{y}_j^{[i]} \log(y_j^{[i]} + \epsilon) \tag{2.52}$$

gdzie ϵ jest niewielką wartością dodawaną dla celów stabilności numerycznej.

Podstawowym sposobem minimalizacji tak określonej funkcji kosztu jest optymalizacja za pomocą metody spadku gradientu, która dokonuje wyznaczenia gradientu na bazie całego zbioru uczącego (***batch gradient descent*** lub *vanilla gradient descent*). Jedynym hiperparametrem w tym wypadku jest tempo uczenia η, a pojedyncza iteracja takiego dostrajania wag wyraża się przez operację:

$$\boldsymbol{\theta}_{(t+1)} \leftarrow \boldsymbol{\theta}_{(t)} - \eta \nabla_{\boldsymbol{\theta}_{(t)}} J(\boldsymbol{\theta}_{(t)}; \boldsymbol{X}, \mathring{\boldsymbol{Y}}) \tag{2.53}$$

Niestety, chociaż wyliczanie wartości gradientu na podstawie wszystkich przykładów uczących jest najdokładniejsze, to przy dużych zbiorach danych staje się kompletnie niepraktyczne. Koniecznością w takiej sytuacji staje się jego estymowanie na podstawie mini-partii składającej się z b przykładów:

$$\hat{\boldsymbol{g}}_{(t)} = \frac{1}{b} \nabla_{\boldsymbol{\theta}_{(t)}} \sum_{i}^{b} \mathcal{L}(f(\boldsymbol{x}^{[i]}; \boldsymbol{\theta}_{(t)}), \mathring{\boldsymbol{y}}^{[i]}) \tag{2.54}$$

Krok aktualizujący jest analogiczny jak w przypadku operowania na całym zbiorze:

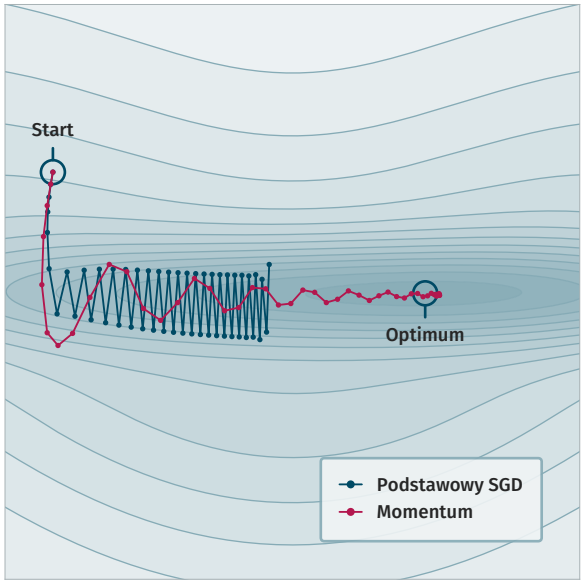
$$\boldsymbol{\theta}_{(t+1)} \leftarrow \boldsymbol{\theta}_{(t)} - \eta \hat{\boldsymbol{g}}_{(t)} \tag{2.55}$$

Technikę tę określa się jako optymalizację spadkiem gradientu mini-partiami (***mini-batch gradient descent***). Często zamiennie pojawia się też nazwa spadku stochastycznego gradientu (***stochastic gradient descent***, ***SGD***), choć właściwie związana jest ona z sytuacją uczenia typu *online* ($b = 1$).

Ustalenie odpowiedniej wartości b jest niestety kwestią niejednoznaczną, najczęściej podejmowaną na zasadzie prób i błędów. Duża liczebność partii powoduje, że wyznaczany gradient jest stabilniejszy, ale kosztem wykonywania dla każdej iteracji większej liczby obliczeń. Poza niepotrzebnym wydłużeniem czasu uczenia zwiększone zostają też wymagania pamięciowe, co jest szczególnie istotne w przypadku wykorzystywania procesorów graficznych. Bardzo głębokie modele mogą potrzebować w fazie uczenia wielu gigabajtów dostępnej pamięci, nawet przy ograniczeniu rozmiaru partii do kilkunastu przykładów. Z tego powodu dobór tej wielkości wynika często po prostu z ograniczeń sprzętowych. Z kolei zbyt mały rozmiar partii nie tylko wprowadza dużą ilość szumu w wyznaczanie kierunku uczenia, ale też nie pozwala w pełni wykorzystać przyspieszenia z tytułu przetwarzania równoległego. Pewnym mało intuicyjnym zachowaniem jest to, że odpowiednia ilość szumu wprowadzanego przez estymację gradientu może wpływać pozytywnie na proces uczenia, ułatwiając eksplorację przestrzeni możliwych rozwiązań i przyspieszając tym sposobem dochodzenie do punktu zbieżności na niższych poziomach funkcji kosztu. Optymalny rozmiar partii równoważący te wszystkie aspekty zależy niestety od konkretnej architektury, zbioru danych i dostępnych rozwiązań sprzętowych, więc trudno w praktyce znaleźć lepszy sposób

na jego ustalenie niż wyjście od w miarę bezpiecznej wielkości kilkudziesięciu przykładów i korektę na bazie obserwacji zachowania procesu uczenia.

Dużą wadą uczenia za pomocą podstawowej wersji spadku gradientu jest jej powolność. Nawet mimo wprowadzania stopniowego wygasania (*learning rate decay*), które pozwala na przyjęcie większej wartości tempa uczenia w początkowej fazie eksploracji, metoda ta ma tendencję do zatrzymywania się na obszarach potocznie określanych jako wąskie doliny lub kaniony, przedstawionych na rysunku 2.15. Zaproponowana przez Polyaka (1964) technika ***momentum*** wprowadza do procesu optymalizacji bezwładność, która dobrze sprawdza się w takich sytuacjach. *Momentum* zakłada, że aktualizacja wag następuje w sposób pośredni poprzez korektę wektora prędkości $\boldsymbol{v}$ tak, że w każdym kroku:



Rys. 2.15: Problem powolnego uczenia za pomocą spadku gradientu i technika momentum.

$$\boldsymbol{v}_{(t)} \leftarrow \alpha \boldsymbol{v}_{(t-1)} - \eta \hat{\boldsymbol{g}}_{(t)} \tag{2.56}$$

i następnie:

$$\boldsymbol{\theta}_{(t+1)} \leftarrow \boldsymbol{\theta}_{(t)} + \boldsymbol{v}_{(t)} \tag{2.57}$$

Hiperparametr $\alpha \in [0, 1]$ określany jako *momentum decay* decyduje o stopniu bezwładności. Ciekawe interaktywne wyjaśnienie jego wpływu na uczenie przedstawia Goh (2017).

Dalszym krokiem usprawniającym jest metoda przyspieszonego gradientu Nesterova (***Nesterov's Accelerated Gradient***) (1983), często spotykana po prostu jako *Nesterov momentum*, która uwzględnia dodatkowo wpływ wprowadzonej bezwładności na zmianę gradientu wyliczanego w danym kroku (tzw. *look-ahead gradient*, $\dot{\boldsymbol{g}}$):

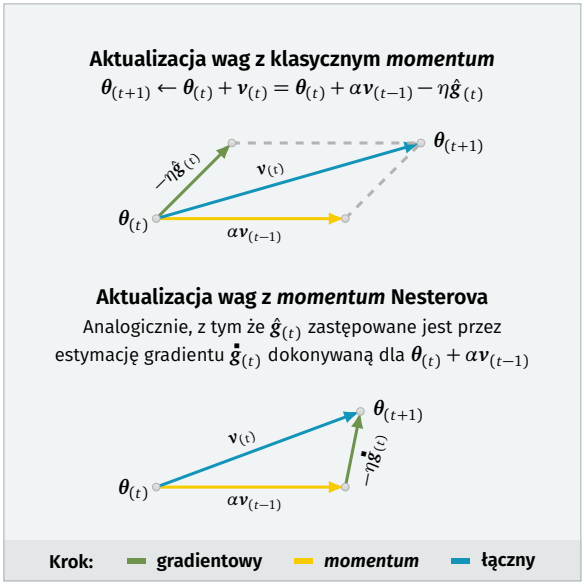
$$\dot{\boldsymbol{\theta}}_{(t)} = \boldsymbol{\theta}_{(t)} + \alpha \boldsymbol{v}_{(t-1)} \tag{2.58}$$

$$\boldsymbol{v}_{(t)} \leftarrow \alpha \boldsymbol{v}_{(t-1)} - \eta \dot{\boldsymbol{g}}_{(t)} = \alpha \boldsymbol{v}_{(t-1)} - \eta \frac{1}{b} \nabla_{\dot{\boldsymbol{\theta}}_{(t)}} \sum_{i}^{b} \mathcal{L}(f(\boldsymbol{x}^{[i]}; \dot{\boldsymbol{\theta}}_{(t)}), \mathring{\boldsymbol{y}}^{[i]}) \tag{2.59}$$

$$\boldsymbol{\theta}_{(t+1)} \leftarrow \boldsymbol{\theta}_{(t)} + \boldsymbol{v}_{(t)} \tag{2.60}$$

Wizualnie sytuację tę obrazuje schemat na rysunku 2.16. Przedstawiona wersja wzorów 2.59 i 2.60 jest zgodna z formułą przyjętą przez Sutskevera et al. (2013), ale spotykane są również lekko odmienne implementacje oparte na wersji Bengio et al. (2013).

Bardzo ważną kwestią w zaprezentowanych algorytmach jest ustalenie odpowiedniego tempa uczenia. Jego adekwatna wartość zależy w dużym stopniu od przyjętej architektury modelu i charakterystyki zbioru danych, a większe wyczulenie modeli głębokich na ten hiperparametr tym bardziej tworzy miejsce potencjalnych niepowodzeń. Postęp związany z wprowadzaniem technik głębokiego uczenia dosyć naturalnie wiązał się więc również z próbą usprawnienia tego aspektu. Jednocześnie pojawiła się wątpliwość, czy zastosowanie jednakowego tempa uczenia dla wszystkich wag modelu jest najlepszym rozwiązaniem. Tą drogą do użytku weszły warianty metod optymalizacyjnych, które zbiorczo można określić jako algorytmy adaptacyjne.



Rys. 2.16: Wyliczanie kroku aktualizacji wag dla *momentum* i wariantu Nesterova.

Najwcześniejszą z przedstawianych tutaj metod jest algorytm ***AdaGrad*** (Duchi et al., 2011). Jego założeniem jest szybkie wygaszanie tempa uczenia w kierunkach, dla których pochodna cząstkowa osiąga duże wartości, co pozwala na bardziej wyrównaną eksplorację również w obszarach jej mniejszego nachylenia. *AdaGrad* osiąga ten cel w wyniku korekty opierającej się na skumulowanych kwadratach wartości gradientu $\boldsymbol{r}$ przez iterację:

$$\boldsymbol{r}_{(t)} \leftarrow \boldsymbol{r}_{(t-1)} + \hat{\boldsymbol{g}}_{(t)} \odot \hat{\boldsymbol{g}}_{(t)} \tag{2.61}$$

$$\boldsymbol{\theta}_{(t+1)} \leftarrow \boldsymbol{\theta}_{(t)} - \frac{\eta}{\sqrt{\boldsymbol{r}_{(t)}} + \epsilon} \odot \hat{\boldsymbol{g}}_{(t)} \tag{2.62}$$

gdzie ϵ jest niewielką wartością dodawaną dla celów stabilności numerycznej, a operacje pierwiastkowania, dzielenia i wymnażania są dokonywane oddzielnie dla każdego elementu wektora.

Sumowanie wartości gradientów od samego początku uczenia powoduje, że wygaszanie występujące w ramach *AdaGrad* może się okazać zbyt szybkie i nie ma sposobu, żeby ten proces odwrócić. Zaproponowana nieformalnie przez Hintona (2012) metoda ***RMSprop*** zamiast tego akumuluje historyczny gradient za pomocą wykładniczej średniej kroczącej:

$$\boldsymbol{r}_{(t)} \leftarrow \rho\boldsymbol{r}_{(t-1)} + (1-\rho)\hat{\boldsymbol{g}}_{(t)} \odot \hat{\boldsymbol{g}}_{(t)} \tag{2.63}$$

$$\boldsymbol{\theta}_{(t+1)} \leftarrow \boldsymbol{\theta}_{(t)} - \frac{\eta}{\sqrt{\boldsymbol{r}_{(t)}} + \epsilon} \odot \hat{\boldsymbol{g}}_{(t)} \tag{2.64}$$

Nowo wprowadzany hiperparametr ρ kontroluje w tym wypadku tempo wygaszania pamięci (domyślnie $\rho = 0{,}9$), pozwalając na uniknięcie całkowitej stagnacji uczenia. *RMSprop* można również rozszerzyć o *momentum Nesterova* lub uzyskać technikę opisaną niezależnie przez Zeilera (2012) jako ***AdaDelta*** poprzez dodatkowe uwzględnienie historii dokonywanych zmian wag.

Chociaż empiryczne wyniki zastosowania różnych algorytmów optymalizacji nie wskazują na jednoznacznego faworyta, to względnie dużą popularnością wśród podejść adaptacyjnych zaczęła się w ostatnim czasie cieszyć metoda ***Adam*** (*adaptive moments*), której krok aktualizacji wag rozbudowany jest do postaci (Kingma & Ba, 2014):

$$\boldsymbol{s}_{(t)} \leftarrow \rho_1 \boldsymbol{s}_{(t-1)} + (1-\rho_1)\hat{\boldsymbol{g}}_{(t)}, \qquad \hat{\boldsymbol{s}}_{(t)} \leftarrow \boldsymbol{s}_{(t)}/(1-\rho_1^t) \tag{2.65}$$

$$\boldsymbol{r}_{(t)} \leftarrow \rho_2 \boldsymbol{r}_{(t-1)} + (1-\rho_2)\hat{\boldsymbol{g}}_{(t)} \odot \hat{\boldsymbol{g}}_{(t)}, \qquad \hat{\boldsymbol{r}}_{(t)} \leftarrow \boldsymbol{r}_{(t)}/(1-\rho_2^t) \tag{2.66}$$

$$\boldsymbol{\theta}_{(t+1)} \leftarrow \boldsymbol{\theta}_{(t)} - \frac{\eta \hat{\boldsymbol{s}}_{(t)}}{\sqrt{\hat{\boldsymbol{r}}_{(t)}} + \epsilon} \tag{2.67}$$

Dołożenie w miejsce wyznaczanego gradientu jego wykładniczej średniej kroczącej ma charakter wygładzający. Ważne jest też uzyskiwane tym sposobem ograniczenie od góry efektywnego rozmiaru kroku, co pozwala nieco inaczej rozumieć tempo uczenia — w metodzie *Adam* można je traktować jako wyznaczany wokół aktualnych wartości parametrów obszar zaufania, w którym pojedyncza aktualizacja może się maksymalnie poruszać. Adaptacyjny charakter tego zachowania powoduje, że nie jest konieczne tak uważne dostrajanie tego hiperparametru, jak w przypadku optymalizacji za pomocą standardowej metody spadku gradientu. Kingma i Ba (2014) zauważają też, że inicjalizacja średnich kroczących wartościami zerowymi prowadzi do wystąpienia obciążenia, którego korektę przewiduje wyznaczanie momentów $\hat{\boldsymbol{s}}$ i $\hat{\boldsymbol{r}}$. Wykładnik potęgi t we wzorach 2.65 i 2.66 oznacza w tej sytuacji numer danego kroku.

Autorzy przedstawianej metody przewidują również w rozszerzeniu określanym jako ***AdaMax*** możliwość zastąpienia normy L_2 normą L_∞. Zachowując dotychczasową postać momentu $\hat{\boldsymbol{s}}$ ze wzoru 2.65, iteracja algorytmu przyjmuje wtedy formę:

$$\boldsymbol{r}_{(t)} \leftarrow \max(\rho_2 \boldsymbol{r}_{(t-1)}, |\hat{\boldsymbol{g}}_{(t)}|) \tag{2.68}$$

$$\boldsymbol{\theta}_{(t+1)} \leftarrow \boldsymbol{\theta}_{(t)} - \frac{\eta \hat{\boldsymbol{s}}_{(t)}}{\boldsymbol{r}_{(t)}} \tag{2.69}$$

Do obydwu wariantów możliwe jest też dołączenie metody wyznaczania przyspieszonego gradientu Nesterova zgodnie z zaprezentowanym przez Dozata (2016) opisem algorytmów określanych odpowiednio jako ***Nadam*** i ***NadaMax***.

Czy w takim razie w tym gąszczu możliwych do zastosowania modyfikacji podstawowego algorytmu optymalizacyjnego można wskazać jakieś bardziej ogólne zalecenia? Różnorodność uzyskiwanych empirycznie rezultatów niestety nie ułatwia odpowiedzi na to pytanie, tym bardziej, że pominięte w dotychczasowym omówieniu zostały jeszcze metody optymalizacji rozproszonej i te korzystające z pochodnych drugiego rzędu.

Jeśli chodzi o tę pierwszą grupę, to uczenie rozproszone może się okazać ważnym tematem w niedalekiej przyszłości, ale w kontekście mniejszych problemów jest o tyle nieistotne, że są to głównie próby zaadaptowania obecnych już rozwiązań do przetwarzania równoległego. Zestawienie kilku strategii wykorzystywanych obecnie w tym zakresie przedstawia między innymi Ruder **(2016)**.

Z kolei prezentowane w bardzo obszernym omówieniu Bottou et al. **(2016)** metody optymalizacji opierające się na estymowaniu pochodnych drugiego rzędu są o tyle niepraktyczne, że w większości przypadków ich złożoność pamięciowa ogranicza możliwości realnego zastosowania dla sieci neuronowych składających się z milionów parametrów. Chociaż warianty starające się uwzględnić ten aspekt, jak na przykład *L-BFGS*, udawało się wprowadzić z dobrymi rezultatami dla wybranych architektur splotowych **(Le et al., 2011)**, to w codziennej praktyce dalej są one ewidentną rzadkością.

Dosyć częstym postępowaniem, które można uznać za bezpieczną heurystykę, jest wybór metody optymalizacji na podstawie porównania wstępnych rezultatów uczenia modelu za pomocą tradycyjnej metody gradientu stochastycznego (*SGD*), rozszerzonej ewentualnie o *momentum*, i jednej z metod adaptacyjnych. Najczęściej dobór tej ostatniej dokonywany jest na zasadzie subiektywnej preferencji podyktowanej stopniem znajomości konkretnego wariantu, ułatwiającym ewentualne dalsze dostrajanie hiperparametrów. Z drugiej strony, wprowadzanie odpowiedniej normalizacji sieci potrafi na tyle uprościć proces uczenia, że coraz częstszą tendencją jest po prostu ograniczanie się do wykorzystania najbardziej podstawowego wariantu optymalizacji ze względu na znikomą poprawę przy zastosowaniu metod adaptacyjnych. Z tego powodu w eksperymentach prezentowanych w ramach rozprawy wykorzystywana będzie przede wszystkim metoda gradientu stochastycznego z momentum, względnie algorytm *Adam*.

2.3 Rozwój architektur splotowych

Omówienie zawarte w podrozdziale 2.2 przedstawiło szereg istotnych elementów związanych z uczeniem głębokich sieci neuronowych i pokazało też główne tendencje rozwojowe w tym obszarze. Aspektem szczególnie ważnym z punktu widzenia analiz prezentowanych w rozprawie, a dotychczas zaznaczanym tylko na bardzo ogólnym poziomie, jest wybór odpowiedniej architektury sieci splotowej – schematu połączeń między warstwami, ich liczby,

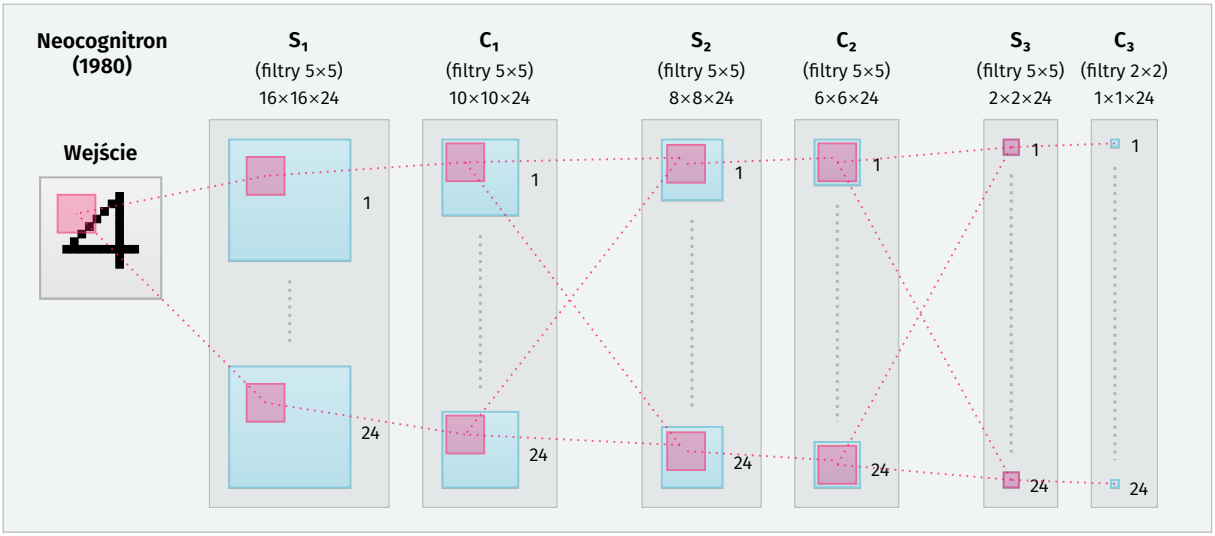
kształtu stosowanych filtrów. Jako że w przeciągu ostatnich lat w zakresie tym proponowane były setki szczegółowych wariantów, celem tego podrozdziału będzie skupienie się raczej na ogólnym procesie zmian zachodzących w stosowanych modelach poprzez chronologiczne omówienie flagowych architektur i stojących za nimi pomysłów.

2.3.1 Neocognitron i LeNet – splotowe początki

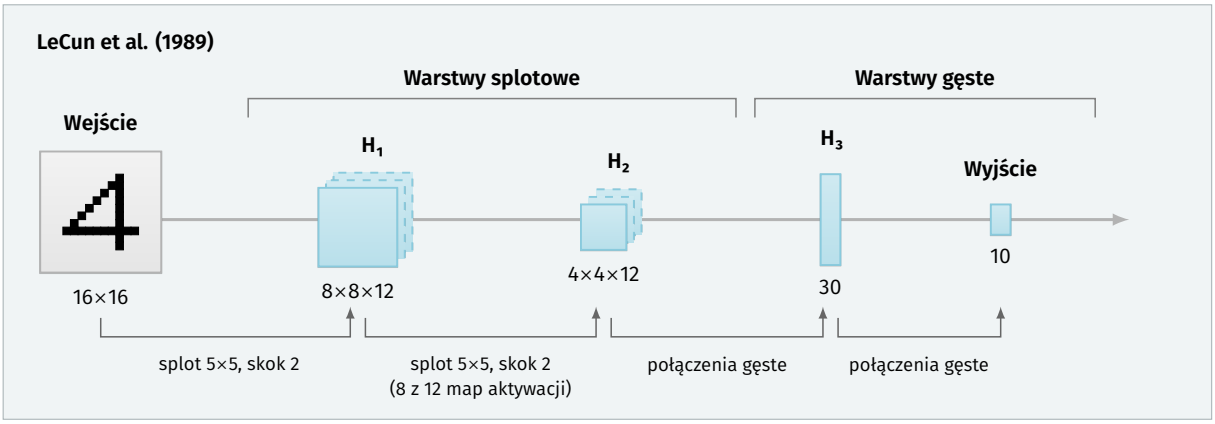
Jedną z najwcześniejszych architektur splotowych jest model Fukushimy (1980) określany jako ***neocognitron***. Celem tego modelu jest wyuczenie bez nadzoru samoorganizującej się sieci neuronowej, która będzie w stanie rozpoznawać wzorce według geometrycznego podobieństwa ich kształtu, niezależnie od ich pozycji.

Neocognitron, przedstawiony schematycznie na rysunku 2.17, zawiera warstwę wejścia (zespół fotoreceptorów) połączoną kaskadowo z trzema modułami składającymi się kolejno z jednej warstwy neuronów prostych (*S-cells*) i złożonych (*C-cells*). Założeniem Fukushimy jest, że tylko połączenia przychodzące do warstwy prostej są plastyczne i ulegają modyfikacjom w procesie organizacji sieci. Efektem tego procesu jest wytworzenie selektywnej odpowiedzi w neuronach ostatniej warstwy, które reagują na wystąpienie na wejściu konkretnego kształtu, ale nie są podatne na jego niewielkie zmiany lub przesunięcia. Aspekt ten odróżnia *neocognitron* od jego poprzednika, *cognitronu* (Fukushima, 1975), który nie miał charakteru splotowego, stąd jego odpowiedź zależała od umiejscowienia wzorca.

Niemodyfikowalny charakter mają z kolei neurony warstwy złożonej. Są one tak zdefiniowane, że aktywują się, gdy w ich bezpośrednim polu postrzegania chociaż jeden z neuronów typu „S” generuje mocny sygnał. Chociaż szczegóły implementacji odróżniają model *neocognitronu* od stosowanych obecnie sieci neuronowych, to warstwy „C” można w uproszczeniu uznać za analogię do operacji grupowania typu *max-pooling*. W takim wypadku



Rys. 2.17: Model *neocognitronu* zaproponowany przez Fukushimę (1980).



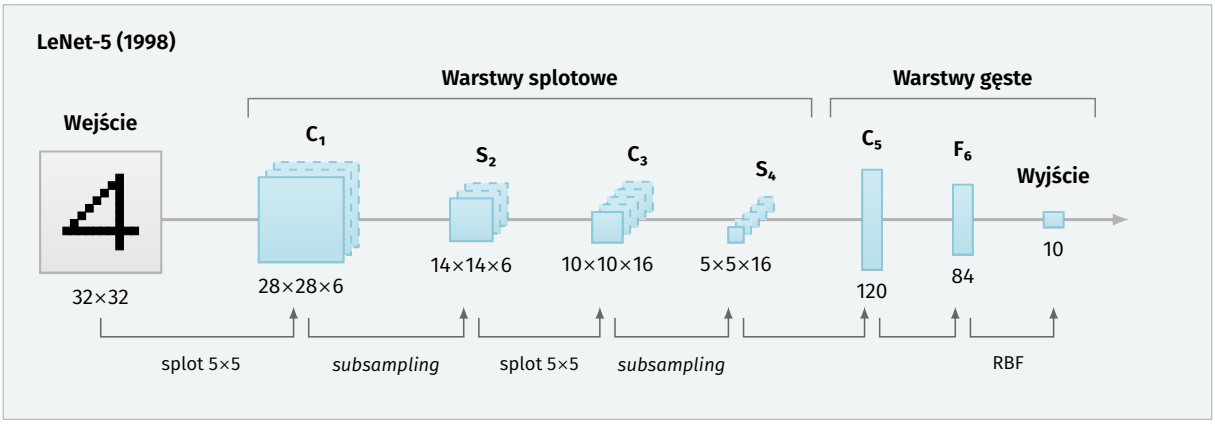
Rys. 2.18: Model rozpoznawania kodów pocztowych LeCuna et al. (1989).

neocognitron w aktualnej nomenklaturze można opisać jako sieć splotową składającą się z sześciu warstw (nie wliczając wejścia) w postaci trzech modułów *splot-pooling*. Rozmiar wejścia, na którym operuje model Fukushimy, to 16×16 pikseli, a każda warstwa splotowa zawiera 24 filtry o rozmiarze 5×5. W kolejnych warstwach, wraz ze zwiększaniem się pośredniego pola postrzegania, zmniejszana jest natomiast gęstość stosowanej siatki neuronów – w ostatnim module każda z 24 map aktywacji generowanych przez warstwę C_3 zawiera tylko jeden neuron wyjścia, aktywujący się dla specyficznego kształtu występującego w danych wejściowych.

Na bazie tak ustalonej architektury Fukushima (1980) przeprowadził udane eksperymenty dla rozpoznawania pięciu cyfr (0–4) i czterech liter „X”, „Y”, „T”, „Z”. Chociaż rozszerzenie wachlarza możliwości wejściowych do dziesięciu cyfr okazało się już nieco problematyczne dla *neocognitronu*[6], to mając na względzie okres jego powstania, zaskakujący jest stopień podobieństwa tej architektury do występujących dużo później wariantów. Wydaje się jednak, że praca Fukushimy pozostała nieco na uboczu i dopiero LeCun et al. byli w stanie spopularyzować koncepcję modeli splotowych, dochodząc do znanej dziś rodziny rozwiązań określanych jako ***LeNet*** (LeCun et al., 1998b).

Jednak już na wcześniejszym etapie LeCun et al. (1989) wykorzystali zobrazowaną na rysunku 2.18 architekturę splotową do odczytywania zapisywanych odręcznie kodów pocztowych. Jej wejściem są obrazy o rozmiarze 16×16 pikseli przetwarzane następnie przez dwie warstwy splotowe H_1 i H_2 o skoku 2 i zawierające 12 filtrów 5×5 pikseli każda. Nieco nietypowo, neurony warstwy H_2 operują na podzbiorach tylko 8 z 12 map aktywacji generowanych przez warstwę H_1. Dalsze przetwarzanie opiera się na neuronach gęsto połączonych – 30 w warstwie ukrytej H_3 i 10 w warstwie wyjścia. Model ten wprowadza nieliniowość poprzez zastosowanie przeskalowanego tangensu hiperbolicznego. Łącznie sieć zawiera 1256 neuronów, 64 660 połączeń i tylko 9760 parametrów.

[6] Fukushima tłumaczył występujące trudności zbyt małą liczbą filtrów, której zwiększenie nie było możliwe ze względu na bardzo ograniczone zasoby pamięciowe ówczesnego sprzętu komputerowego.



Rys. 2.19: Schemat architektury sieci *LeNet-5* (LeCun et al., 1998b).

Istotnie większym modelem (340 908 połączeń, 60 000 parametrów) jest zobrazowany na rysunku 2.19 ***LeNet-5*** (LeCun et al., 1998b), składający się z 7 warstw. Danymi wejściowymi w jego przypadku są fragmenty pisma odręcznego o rozmiarze 32×32 pikseli, na których operuje warstwa splotowa C_1 składająca się z 6 filtrów 5×5. Warstwa S_2 (*subsampling layer*) dokonuje zmniejszenia rozdzielczości map aktywacji na podobnej zasadzie jak grupowanie uśredniające[7] na oknie 2×2. Analogicznie funkcjonuje warstwa S_4. Z kolei warstwa C_3 (16 filtrów 5×5) tworzy połączenia tylko do wybranych map z S_2, w zbliżony sposób jak w opisywanym już wcześniej modelu LeCuna et al. (1989). Autorzy decyzję tę uzasadniali koniecznością ograniczenia liczby tworzonych połączeń i wymuszonym złamaniem symetrii filtrów splotowych. Wybiórczość połączeń nie występuję już natomiast w warstwie C_5, która składa się ze 120 filtrów 5×5. Ze względu na rozmiar tworzonych map aktywacji (1×1) można ją rozumieć jako warstwę w pełni połączoną. Ekspresyjność modelu dodatkowo wzbogaca warstwa gęsta F_6 z 84 neuronami, na którą nałożona jest warstwa wyjścia 10 neuronów z euklidesową radialną funkcją bazową (*Euclidean Radial Basis Function units*).

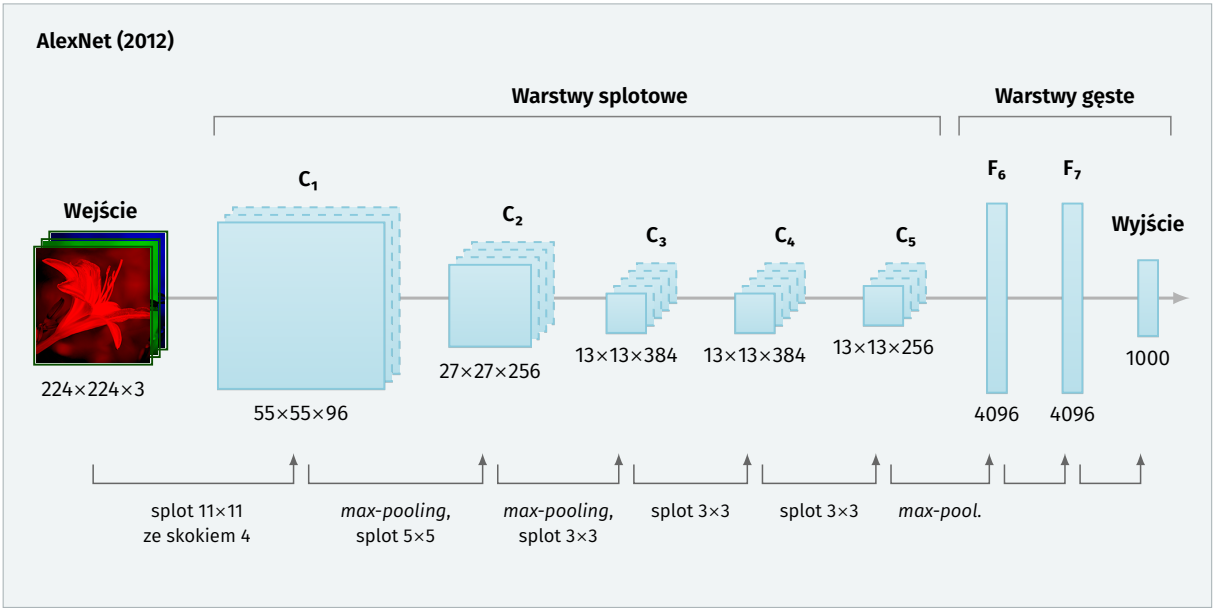
Architektura *LeNet-5* jest istotna z dwóch powodów. Po pierwsze, osiągając dla bardzo praktycznego zastosowania rozpoznawania pisma odręcznego dokładność lepszą od innych technik, w tym bardzo cenionych w owym czasie maszyn wektorów nośnych, pokazała, że uczenie wielowarstwowych sieci neuronowych jest możliwą i obiecującą ścieżką rozwoju. Było to szczególnie ważne ze względu na dosyć powszechną w tym okresie nieufność w kierunku sztucznych sieci neuronowych. Drugim aspektem jest wprowadzenie przez *LeNet-5* pewnego sprawdzonego schematu, jak taki model splotowy mógłby wyglądać. Chociaż późniejsze architektury powiększały się zarówno na szerokość, jak i na głębokość, to w dużej mierze powielały schemat powtarzanych operacji *splot–pooling* zakończony przetwarzaniem warstwami gęstymi.

[7] W odróżnieniu od operacji typu *average pooling*, warstwy „S" dokonują jeszcze wymnożenia przez wyuczony współczynnik, dodania obciążenia i aktywacji sigmoidalnej.

2.3.2 AlexNet – nowa era w rozpoznawaniu obrazów

Pojawienie się przywoływanej wielokrotnie sieci Krizhevsky'ego et al. (2012), ***AlexNet***, było przełomowym momentem popularyzującym zastosowanie sieci splotowych. Chociaż Cireşan et al. (2011) już rok wcześniej opisali model splotowy uczony na *GPU*, to przeprowadzone przez nich eksperymenty ograniczone były do niewielkich zbiorów (*MNIST*, *NORB*, *CIFAR-10*). Dopiero uzyskanie przez *AlexNet* pierwszego miejsca w *ImageNet Large Scale Visual Recognition Challenge 2012* (Russakovsky et al., 2015) pokazało, że sieci splotowe mają ogromny potencjał również w zakresie klasyfikacji obiektów na dużą skalę.

Architektura *AlexNet* składa się z ośmiu warstw zobrazowanych na rysunku 2.20 i jest w dużej mierze podobna do *LeNet*, natomiast różni się przede wszystkim wielkością modelu, licząc 60 milionów parametrów. Pierwsza warstwa splotowa przetwarza fragmenty obrazów składające się z 224 × 224 pikseli (×3 kanały RGB) za pomocą 96 filtrów o rozmiarze 11 × 11 × 3 i skoku 4. *Max-pooling* dokonywany jest w sposób nachodzący na siebie (w polu 3 × 3 ze skokiem 2) i dalej odpowiedź sieci jest lokalnie normalizowana. Sieć Krizhevsky'ego jest o tyle specyficzna, że ze względu na jej znaczący rozmiar w porównaniu do dostępnych zasobów pamięciowych konieczne było podzielenie jej przetwarzania na dwie karty graficzne, co znalazło też odzwierciedlenie w samej architekturze – druga, czwarta i piąta warstwa splotowa są połączone tylko z tymi filtrami poprzedniej warstwy, które znajdują się na tym samym *GPU*. Kolejne warstwy stosują malejące okno przetwarzania (5 × 5 i 3 × 3), zwiększając liczbę wykorzystywanych filtrów do 256 i 384. Gros parametrów sieci znajduje się natomiast w dwóch warstwach gęstych liczących 4096 neuronów każda.



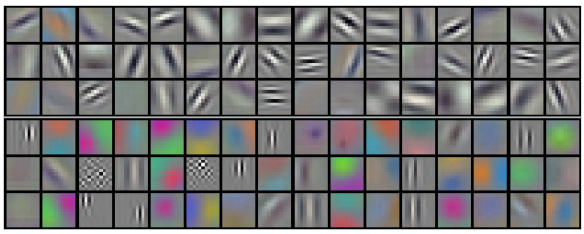
Rys. 2.20: Schemat architektury sieci Krizhevsky'ego (2012).

Z punktu widzenia rozwoju modeli splotowych ważne jest kilka faktów związanych z *AlexNet*. Sieć ta potwierdziła przede wszystkim zasadność wykorzystania aktywacji typu ReLU, wskazując na uzyskiwaną tym sposobem dużo szybszą zbieżność procesu uczenia. Podobnie rzecz się miała z zastosowaniem *dropoutu*, który okazał się konieczny, aby uniknąć przetrenowania występującego pomimo bazowania na bardzo dużym zbiorze uczącym. Nowością było też wprowadzenie kolejno kilku warstw splotowych, które nie były przedzielone operacją grupowania. Krizhevsky et al. **(2012)** zawarli też w swojej pracy praktyczne uwagi wskazujące na potrzebę sztucznego zwiększania zbioru danych przez translacje obrazów i ich odbicia, nawet w przypadku tak dużego zbioru jak *ILSVRC*. Poprawę końcowego wyniku zapewniło w ich przypadku również uśrednianie predykcji uzyskiwanej dla kilku mniejszych fragmentów i stworzenie komitetu pięciu sieci splotowych.

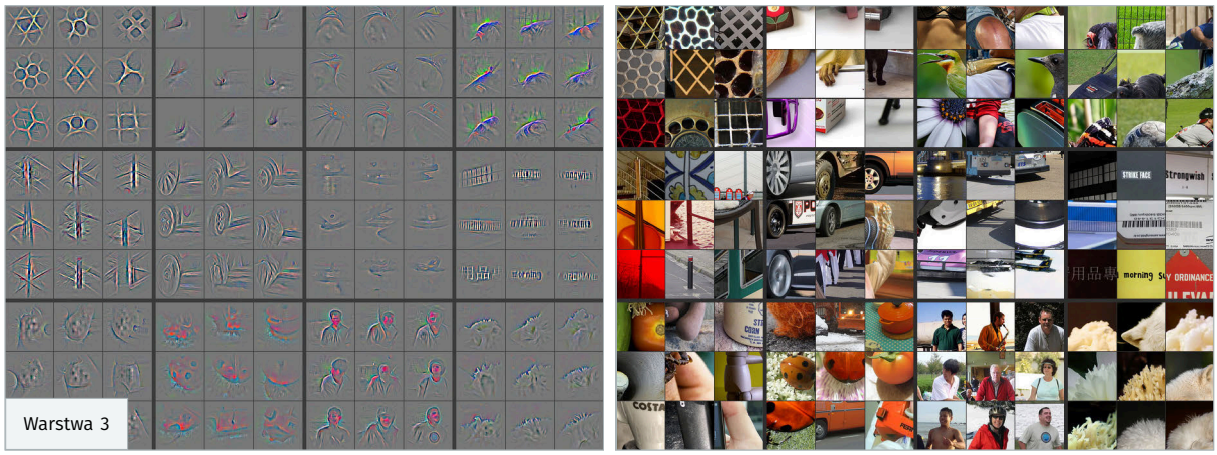
2.3.3 ZFNet – wizualizacja sieci splotowych

Głębokie sieci neuronowe spotykały się z zarzutem, że mają charakter „czarnej skrzynki", której sposób działania jest kompletnie ukryty, a usprawnienia modelu dokonywane są tylko na zasadzie prób i błędów. Praca Zeilera i Fergusa **(2014)** pokazała, że stwierdzenie to nie musi być wcale prawdą. Autorzy ci wprowadzili nową technikę wizualizacji funkcjonowania sieci neuronowych i zademonstrowali, jak pozyskane tą drogą informacje mogą posłużyć do usprawnienia konstruowanego modelu w bardziej ukierunkowany sposób.

Podejściem standardowo pojawiającym się we wcześniejszych pracach była analiza pierwszej warstwy splotowej, którą można stosunkowo łatwo zobrazować poprzez bezpośrednie wyświetlenie wag wyuczonych filtrów. Rysunek 2.21 przedstawia efekt takiego postępowania na bazie modelu *AlexNet*. Dużo bardziej skomplikowana jest niestety interpretacja funkcjonowania warstw występujących głębiej. Techniką stosowaną w tym celu może być utworzenie sztucznego przykładu maksymalizującego wartość aktywacji danego neuronu za pomocą optymalizacji gradientowej **(Erhan et al., 2009)**, ale zdaniem Zeilera i Fergusa **(2014)** sposób ten jest szukaniem po omacku, wymaga bardzo uważnej inicjalizacji i trudno na jego podstawie wnioskować na temat stopnia niezmienności neuronów. Dlatego ich propozycją jest powrotne odwzorowanie uzyskiwanych aktywacji w przestrzeni obrazów wejściowych za pomocą sieci rozplotowej (*deconvolutional network*) **(Zeiler et al., 2011)**. Sieć taka, wykorzystując splot transponowany, stara się w przybliżony sposób zrekonstruować wartości aktywacji uzyskiwane w poprzedzających warstwach przy założeniu, że wzbudzany jest tylko



Rys. 2.21: Zobrazowanie wybranych filtrów splotowych pierwszej warstwy *AlexNet*. Reprodukcja za zgodą **(Krizhevsky et al., 2012)**.



Rys. 2.22: Wizualizacja trzeciej warstwy *ZFNet*. Obrazy na szarym tle przedstawiają rekonstrukcję wzorców za pomocą sieci rozplotowej dla dziewięciu przykładów ze zbioru walidacyjnego generujących największe wartości aktywacji dla wybranych neuronów. Odpowiadające im oryginalne obrazy umieszczone są po prawej stronie. Reprodukcja za zgodą (Zeiler & Fergus, 2014) udzieloną poprzez Copyright Clearance Center, Inc.

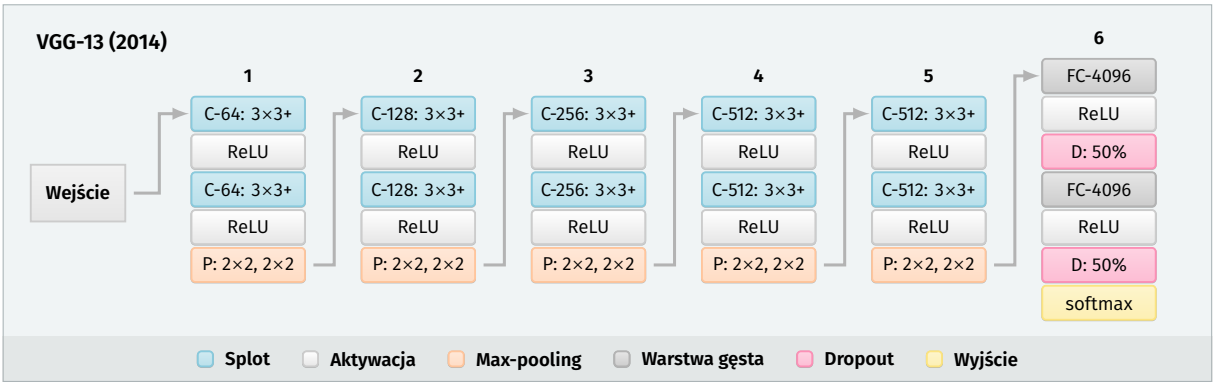
jeden neuron w warstwie analizowanej. Metoda ta pozwala na ekstrakcję wzorców, które najmocniej aktywują konkretne neurony i odpowiadających im obrazów spośród zbioru walidacyjnego, przedstawionych na rysunku 2.22.

Zeiler i Fergus (2014) zauważają, że przeprowadzenie takiej analizy na pierwszych dwóch warstwach modelu *AlexNet* pozwala wskazać pewne mankamenty tej architektury. Jedną kwestią jest widoczne na rysunku 2.21 skupianie się przez pierwszą warstwę splotową na wzorcach albo o wysokiej, albo niskiej częstotliwości, bez uwzględniania wariantów pośrednich. Innym problemem tej warstwy jest jej duży skok (4), który skutkuje wystąpieniem na dalszym etapie widocznych artefaktów aliasingu. W związku z tym model ***ZFNet*** koryguje te błędy poprzez zmniejszenie wielkości filtrów pierwszej warstwy do rozmiaru 7×7 i ograniczenie skoku do 2. Pozytywny efekt tych zmian widoczny jest nie tylko w wizualizacjach tak usprawnionego modelu, ale także w poprawie jego zdolności klasyfikacyjnej (błąd walidacji typu *top-5* zmniejszony zostaje z 18,1% do 16,5%).

2.3.4 VGGNet – głębokość ma znaczenie

Rok 2014 był rokiem znaczącego zwiększania się głębokości wykorzystywanych architektur splotowych. Czołowe rezultaty uzyskane w ramach *ILSVRC 2014* należały do modeli *VGGNet* i *GoogLeNet* składających się odpowiednio nawet z 19 i 22 warstw.

VGGNet jest architekturą przedstawioną przez Simonyana i Zissermana z *Visual Geometry Group* (2014) w pracy, która porównywała modele o rosnącej głębokości przy zastosowaniu filtrów splotowych o bardzo małych rozmiarach (3×3). Skok splotu ustalony jest w ich przypadku na 1 piksel i podobnie wypełnienie (*padding*) również ma szerokość 1 piksela. Pozwala to uzyskać mapy aktywacji, które mają taką samą rozdzielczość jak dane na



Rys. 2.23: Schemat architektury *VGGNet* (Simonyan & Zisserman, 2014) w wariancie „B" składającym się z 13 warstw.

Oznaczenia: **(C-K: $H\times W$)** – warstwa splotowa z K filtrami o wysokości H i szerokości W (znak „+" wskazuje na *padding*, w tym wypadku o wielkości 1), **(P: $H\times W$, $S_H\times S_W$)** – *max-pooling* z oknem $H\times W$ i skokiem $S_H\times S_W$, **(FC-N)** – warstwa w pełni połączona (gęsta) z N neuronami, **(D: p%)** – *dropout* z prawdopodobieństwem odrzucenia p%.

wejściu warstwy splotowej. Tym sposobem możliwe staje się łączenie takich warstw przetwarzających w dowolnie długi ciąg. Redukcja rozdzielczości następuje wyłącznie poprzez grupowanie typu *max-pooling* z oknem 2×2 i krokiem o wartości 2, które jest stosowane co kilka warstw, zgodnie ze schematem na rysunku 2.23.

Autorzy *VGGNet* wskazują, że moduł składający się z 3 warstw 3×3 ma efektywne pole postrzegania jak pojedyncza warstwa 7×7, ale używa dużo mniejszej liczby parametrów i zawiera w sobie trzy nieliniowości zamiast jednej. Z tego powodu zdaniem Simonyana i Zissermana (2014) lepsze jest właśnie stosowanie małych rozmiarów filtrów. Jednocześnie też w swojej pracy empirycznie pokazują, że stopniowe zwiększanie liczby warstw z 11 do 19 pozwala obniżyć błąd klasyfikacji *top-5* uzyskiwany na zbiorze walidacyjnym *ILSVRC 2014* z 10,4% do 9,0%, a w przypadku operowania na dwóch rozdzielczościach obrazów wejściowych nawet do 8,0%. Niestety, mimo bardzo dobrych wyników klasyfikacji, istotną wadą tej architektury jest bardzo duża liczba parametrów (144 miliony). Niemniej sam pomysł stosowania małych filtrów splotowych stał się bardzo popularny poprzez przykład *VGGNet*, a z czasem też okazało się, że warstwy gęsto połączone mają przy takim podejściu niewielki wpływ na klasyfikację i model ten można istotnie pomniejszyć poprzez ich wyeliminowanie.

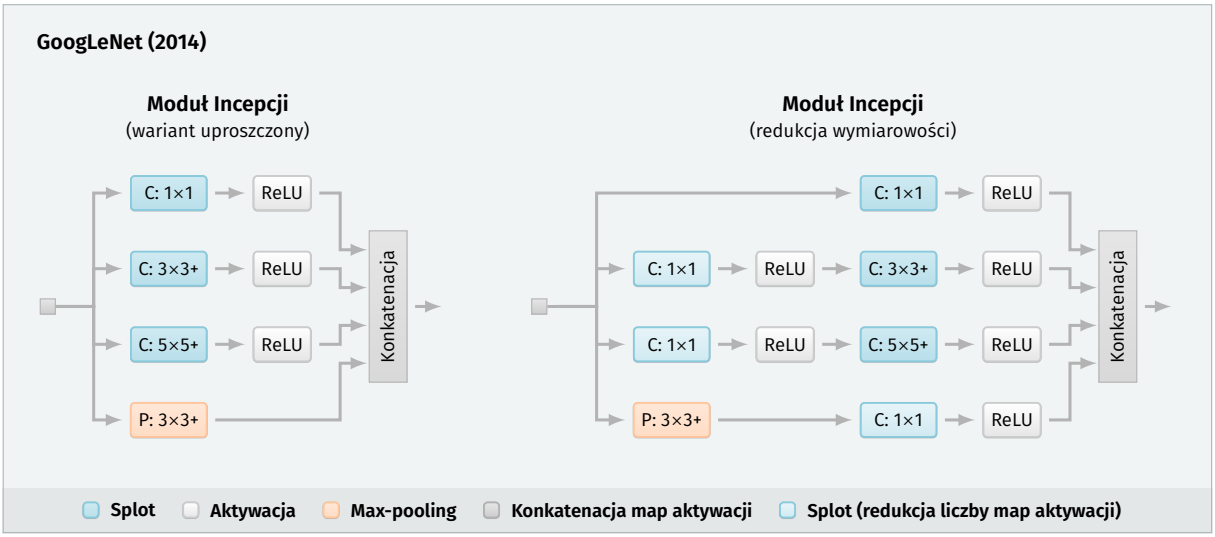
2.3.5 GoogLeNet – moduł Incepcji

Drugi czołowy model w ramach *ILSVRC 2014*, ***GoogLeNet***, jest siecią jeszcze głębszą, bo zawierającą 22 warstwy przetwarzające. Dość naturalnie można by oczekiwać, że będzie tym sposobem jeszcze bardziej wymagający obliczeniowo od *VGGNet*. Zespołowi z Google udało się jednak dokonać rzeczy zgoła odwrotnej – sieć *GoogLeNet* (Szegedy et al., 2015) zawiera 12 razy mniej parametrów niż *AlexNet*, a mimo to jej błąd klasyfikacji *top-5* na

danych z *ILSVRC* jest redukowany, w porównaniu do sieci Krizhevsky'ego et al. (2012), z 15,3% do 6,67%[8]. Cytując autorów, wynikającą z tego „pocieszającą wiadomością jest to, że większość postępu [w klasyfikacji i detekcji obiektów] nie jest tylko wynikiem mocniejszego sprzętu, większych zbiorów danych i modeli, ale przede wszystkim konsekwencją nowych pomysłów, algorytmów i usprawnień architektur sieci" (Szegedy et al., 2015).

Główną motywacją Szegedy'ego et al. jest właśnie zauważenie, że dotychczas najczęstszym sposobem tworzenia dokładniejszych modeli było zwiększanie ich rozmiaru i wykorzystanie liczniejszych zbiorów uczących. Jeżeli nie jest to jednak możliwe, to większa liczba parametrów staje się problemem, bo łatwiejsze staje się przeuczenie sieci. Tym bardziej, że liniowe zwiększanie liczby filtrów w połączonych ze sobą warstwach powoduje kwadratowe zwiększanie się złożoności obliczeniowej. Odpowiedzią na te problemy byłoby wykorzystanie reprezentacji rzadkich, za którymi dodatkowo przemawiają prace o charakterze teoretycznym (Arora et al., 2014) i analogie biologiczne, ale niestety obecne rozwiązania sprzętowe są bardzo nieefektywne obliczeniowo operując na tego typu strukturach. Dlatego autorzy *GoogLeNet* wprowadzają sposób budowania sieci określany jako moduł Inception (*Inception module*), który w ich zamiarze ma aproksymować lokalnie rzadkie struktury przetwarzania za pomocą standardowych modułów gęsto połączonych.

Przedstawiony na rysunku 2.24 moduł Incepcji składa się z połączenia operacji splotowych o różnych rozmiarach filtrów (1×1, 3×3, 5×5), których wynik jest scalany na wyjściu danej warstwy. Różne rozmiary filtrów pozwalają na pokrycie skupisk cech o rosnącym zróżnicowaniu przestrzennym. Szegedy et al. zauważają, że kolejne warstwy operujące na wyższym poziomie abstrakcji przetwarzanych cech będą zmniejszały stopień koncentracji



Rys. 2.24: Moduł Incepcji w wariancie uproszczonym i z redukcją wymiarowości. Adaptacja własna na podstawie (Szegedy et al., 2015). Oznaczenia jak na rysunku 2.23.

[8] W obydwu przypadkach podane wyniki dotyczą komitetów 7 sieci splotowych.

Tab. 2.1: Architektura *GoogLeNet* przedstawiona przez (Szegedy et al., 2015).

Moduł	Okno / skok	Wymiary wyjścia	Głęb.	1×1	3×3 red.	3×3	5×5 red.	5×5	MP	Param.	Oper.
Splot	7×7/2	112×112×64	1							2,7K	34M
Max-pooling	3×3/2	56×56×64	0								
Splot	3×3/1	56×56×192	2		64	192				112K	360M
Max-pooling	3×3/2	28×28×192	0								
Inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
Inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
Max-pooling	3×3/2	14×14×480	0								
Inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
Inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
Inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
Inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
Inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
Max-pooling	3×3/2	7×7×832	0								
Inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
Inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
Avg-pooling	7×7/1	1×1×1024	0								
Dropout (40%)		1×1×1024	0								
W. gęsta		1×1×1000	1							1000K	1M
Softmax		1×1×1000	0								

Poszczególne kolumny tabeli przedstawiają w kolejności: typ modułu (warstwy), rozmiar okna wraz ze skokiem, wymiary wyjściowe uzyskiwanej mapy aktywacji, głębokość modułu wyrażoną liczbą warstw, liczbę filtrów poszczególnych elementów modułu Incepcji (rys. 2.24, wersja z redukcją wymiarowości) – wartości w kolumnach „3×3 red.", „5×5 red." i „MP" odpowiadają liczbie filtrów warstwy splotowej 1×1 redukującej wymiarowość odpowiednio przed warstwami 3×3+, 5×5+ i po operacji *max-poolingu*. Kolumny „Param." i „Oper." zawierają przybliżoną liczbę parametrów i złożoność obliczeniową poszczególnych modułów.

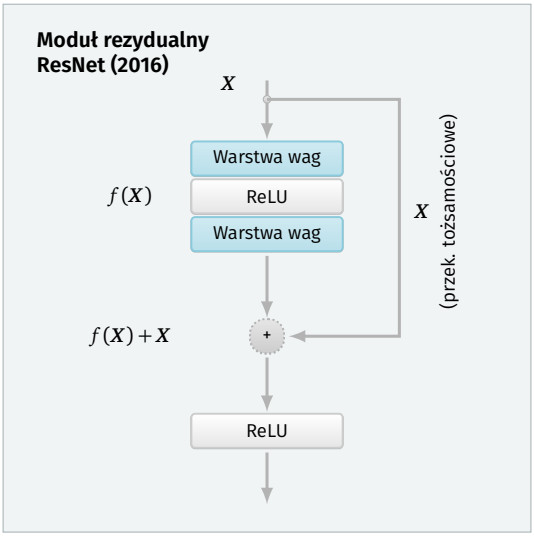
i z tego powodu proporcjonalny udział splotu 3×3 i 5×5 powinien wzrastać dla kolejnych warstw modelu. Dużym problemem jest natomiast fakt, że rosnąca liczba wykorzystywanych filtrów splotowych 5×5 bardzo szybko staje się zaporowa obliczeniowo, dlatego drugi wariant modułu Incepcji rozszerzony jest o reduktory wymiarowości w postaci splotu 1×1, które jednocześnie wprowadzają też dodatkową nieliniowość.

Architektura sieci określana jako *Inception network* jest właśnie złożeniem wielu takich modułów uzupełnianych co jakiś czas operacjami typu *max-pooling* ze skokiem 2, które redukują rozdzielczość reprezentacji. *GoogLeNet* z kolei jest konkretnym wdrożeniem tej koncepcji składającym się z 22 warstw (27 wliczając grupowanie) przedstawionych w tabeli 2.1. Jedną z obaw autorów był wpływ głębokości sieci na efektywną propagację błędu. Z tego powodu wprowadzili oni pomocnicze klasyfikatory (niewielkie sieci splotowe), które operują na wyjściach wcześniejszych warstw oznaczonych jako *Inception 4a* i *4d*, a wartość ich funkcji kosztu dodawana jest z wagą 0,3 do całkowitego kosztu modelu. Przeprowadzone eksperymenty pokazały jednak, że wpływ tego rozwiązania na poprawę klasyfikacji jest niewielki (0,5%) i dodatkowo użycie nawet tylko jednego klasyfikatora pomocniczego jest wystarczające do uzyskania tego efektu.

Koncepcja *Inception network* doczekała się z czasem kilku udoskonaleń. Główną modyfikacją wprowadzoną w kolejnych iteracjach tej architektury, *Inception-v2* i *Inception-v3* (Szegedy et al., 2016), jest rozbicie dużych filtrów (5×5, 7×7) na odpowiednie złożenia operacji 3×3 lub moduły tworzone przez połączenie splotu $1 \times n$ oraz $n \times 1$. *Inception-v4* (Szegedy et al., 2017) stała się z kolei bazą wyjściową do stworzenia modelu hybrydowego *Inception-ResNet*, wdrażającego pomysły przedstawione w sieciach z połączeniami rezydualnymi.

2.3.6 ResNet – połączenia rezydualne

Koncepcja sieci rezydualnych opiera się na spostrzeżeniu, że chociaż zwiększanie liczby stosowanych warstw wielokrotnie wskazywane było jako kluczowy element usprawniania tworzonych modeli, to w przypadku przekroczenia pewnej granicznej wartości dochodzi do zaskakującego zjawiska degradacji. Jest to efekt o tyle osobliwy, że po początkowym wysycaniu dokładności modelu i spadku krańcowej użyteczności dodawania kolejnych warstw, w pewnym momencie dalsze zwiększanie głębokości zaczyna wręcz mocno negatywnie wpływać na skuteczność modelu. O ile dosyć naturalnie można by wytłumaczyć ten fakt przeuczeniem sieci, to jak wskazują He et al. (2016), spadek dokładności widoczny jest nie tylko na zbiorze walidacyjnym, ale ujawnia się również zwiększeniem błędu uczenia na zbiorze trenującym. Oznacza to, że zbyt głębokie modele stają się nadmiernie trudne w optymalizacji, mimo że teoretycznie powinny osiągać rezultaty co najmniej nie gorsze niż ich płytsze odpowiedniki. Ta ostatnia teza wynika z przedstawionego przez He et al. (2016) rozwiązania przez rozbudowę – biorąc za punkt wyjścia model płytszy, dodawane do niego kolejne warstwy w najgorszym wypadku (gdy nie jest możliwe znalezienie lepszej reprezentacji) powinny po prostu wprowadzić przekształcenie tożsamościowe. W praktyce jednak znalezienie tak teoretycznie trywialnego rozwiązania okazuje się zbyt trudne dla stosowanych optymalizatorów, dlatego He et al. (2016) wprowadzają tę koncepcję odgórnie do zaproponowanej architektury, wymuszając uczenie funkcji rezydualnych (resztowych).



Rys. 2.25: Moduł z połączeniem rezydualnym (He et al., 2016a).

Główną częścią składową sieci rezydualnej (***ResNet***) jest przedstawiony na rysunku 2.25 moduł, który do typowego sekwencyjnego złożenia warstw dodaje połączenie zachowujące niezmienioną wartość wejściową $\boldsymbol{X}$. W porównaniu do klasycznego zgrupowania warstw sieci moduł taki uczy się, zamiast pewnego przekształcenia $h(\boldsymbol{X})$, odpowiednio funkcji $f(\boldsymbol{X}) = h(\boldsymbol{X}) - \boldsymbol{X}$. Powoduje to, że gdyby w skrajnym wypadku okazało się, że przekształcenie tożsamościowe jest optymalne, to łatwiej jest przy takim podejściu zrównać część rezydualną do zera niż wyuczyć ciąg standardowych warstw takiego przekształcenia od podstaw.

He et al. **(2016)** w swojej pracy wskazują na kilka ważnych wniosków eksperymentalnych. Po pierwsze, bardzo głębokie sieci rezydualne są łatwe w optymalizacji, czego nie można powiedzieć o analogicznych modelach bez połączeń rezydualnych. Analizowane przez cytowanych autorów standardowe modele składające się z 18 i 34 warstw wskazują, że większa liczba warstw pogarsza wyniki uczenia. W przypadku sieci rezydualnych efekt jest odwrotny. Dodatkowo, sieć 34-warstwowa nie tylko cechuje się mniejszym błędem od jej płytszego odpowiednika, ale obydwa modele rezydualne zachowują się lepiej od swoich tradycyjnych wariantów.

Drugą kwestią jest możliwość łatwego uzyskiwania poprawy przez dalsze zwiększanie głębokości. He et al. **(2016)** w ramach *ILSVRC 2015* rozpatrują również modele składające się z 50, 101 i nawet 152 warstw, które cechują się odpowiednio malejącym błędem walidacji. Jednocześnie, co podkreślają autorzy, *ResNet-152* ma mniejszą złożoność niż *VGGNet*, mimo ogromnej dysproporcji w głębokości. Kierunek ten jest podtrzymywany w dalszych pracach **(He et al., 2016b)**, gdzie rozszerzenie propagacji przekształcenia tożsamościowego z pojedynczych modułów rezydualnych na całą sieć pozwala tworzyć modele składające się nawet z ponad 1000 warstw. Zgodnie ze stanem na koniec 2017 roku są to najgłębsze i najskuteczniejsze architektury w zastosowaniach rozpoznawania obrazów, ale ich złożoność obliczeniowa jest niestety zaporowa[9].

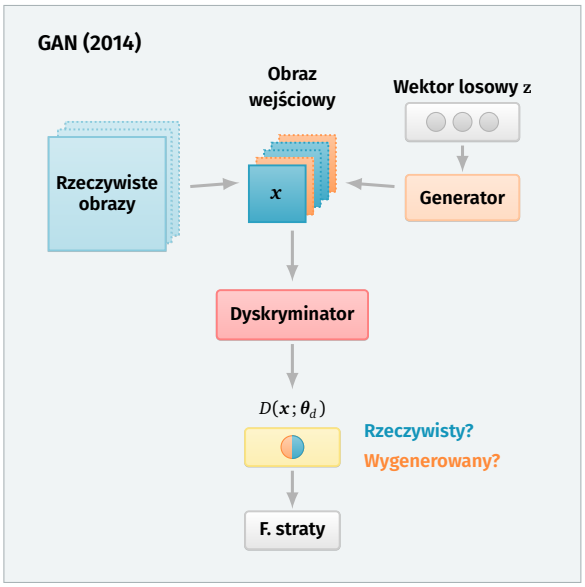
2.3.7 GAN - sieci generatywne z adwersarzem

W tym samym czasie gdy architektury splotowe ewoluowały w kierunku wariantów składających się z coraz większej liczby warstw, w szeroko pojętym obszarze uczenia modeli głębokich wyłaniała się jeszcze jedna ważna tendencja wprowadzania do użycia modeli określanych mianem sieci generatywnych z adwersarzem (*generative adversarial networks* – ***GANs***). Koncepcja ta nie jest związana z konkretną architekturą (rozumianą jako schemat warstw i połączeń), ale podejmuje proces uczenia na wyższym poziomie abstrakcji. Chociaż tym sposobem nie do końca wpisuje się w prezentowany w całości podrozdziału

[9] Czas trenowania na zbiorze *ImageNet* dla sieci rezydualnych składających się z kilkuset warstw może być liczony w miesiącach, nawet przy wykorzystaniu kilku kart graficznych.

chronologiczny rozwój implementacji modeli splotowych, to należałoby jej poświęcić parę zdań chociażby dlatego, że zdaniem LeCuna (2016) jest to „*najbardziej interesujący pomysł ostatnich dziesięciu lat w obszarze uczenia maszynowego*”, co potwierdza też rosnąca wykładniczo liczba publikacji zajmujących się tym tematem (Hindupur, 2017).

Sieci generatywne z adwersarzem są nowym sposobem uczenia modeli o charakterze generatywnym zaproponowanym przez Goodfellowa et al. (2014). W podejściu tym występują dwa modele schematycznie przedstawione na rysunku 2.26 – generator (G) i dyskryminator (D). Celem generatora jest losowe tworzenie danych, które swoim rozkładem brzegowym odpowiadałyby rozkładowi występującemu w rzeczywistym zbiorze trenującym. Rolą dyskryminatora jest z kolei ocena, czy przedstawiany mu przykład pochodzi rzeczywiście ze zbioru uczącego, czy został sztucznie wygenerowany. Zależnie od kontekstu takie duety modeli określa się czasami także jako podrabiacz–śledczy (*forger–investigator*) lub aktor–krytyk (actor–critic), choć ten ostatni termin związany jest z konkretną koncepcją pochodzącą z obszaru uczenia ze wzmocnieniem, która choć bardzo zbliżona, to jest nieco szerszym pojęciem (Pfau & Vinyals, 2016).



Rys. 2.26: Schemat interakcji sieci generatora i dyskryminatora.

Zgodnie z oznaczeniami zastosowanymi oryginalnie przez Goodfellowa et al. (2014), jeśli za $G(\mathbf{z};\boldsymbol{\theta}_g)$ przyjmiemy dokonywane za pomocą sieci neuronowej parametryzowanej przez $\boldsymbol{\theta}_g$ przekształcenie do przestrzeni danych szumu losowego $\mathbf{z}$ o rozkładzie a priori $p_{\mathbf{z}}(\mathbf{z})$, a rezultatem $D(\boldsymbol{x};\boldsymbol{\theta}_d)$ jest skalar określający prawdopodobieństwo, że dane wejściowe $\boldsymbol{x}$ pochodzą z rozkładu rzeczywistego p_{data} a nie generowanego p_g, to uczenie sieci typu *GAN* można wyrazić jako grę między D i G przy strategii *minimax*:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{2.70}$$

W praktyce oznacza to, że proces ten przebiega iteracyjnie, tak aby zmaksymalizować prawdopodobieństwo prawidłowego rozróżnienia między przykładami rzeczywistymi i syntetycznymi przez dyskryminator, a jednocześnie uczenie generatora stara się zminimalizować szansę wykrycia: $\log(1 - D(G(\mathbf{z})))$.

Uzyskanie odpowiedniej równowagi między generatorem i dyskryminatorem jest złożonym problemem i w ramach rozwiązań typu *GAN* nie udało się dotychczas formalnie dowieść istnienia sposobu postępowania, który zagwarantowałby, że uczenie sieci z adwersarzem zawsze dojdzie do stanu stabilnego. Widać to po licznych eksperymentach, które pokazują, że uzyskanie pożądanych wyników jest na obecnym etapie często sztuką i niewielkie zmiany hiperparametrów okazują się kluczowo wpływać na końcowy rezultat. Dlatego też udoskonalanie technik uczenia sieci generatywnych z adwersarzem jest bardzo świeżym i otwartym problemem badawczym **(Salimans et al., 2016)**. Przykładowo, Durugkar et al. **(2016)** sugerują, że wykorzystanie sieci z wieloma dyskryminatorami (*GMAN – Generative Multi-Adversarial Network*) pozwala na usprawnienie procesu uczenia bez konieczności przeformułowywania funkcji celu.

Odrębną kwestią jest dobór architektury generatora i dyskryminatora. Chociaż teoretycznie w zakresie tym istnieje duża dowolność, to znacząca część obecnych modeli bazuje na zaproponowanej przez Radforda et al. **(2015)** koncepcji *DCGAN* (*Deep Convolutional Generative Adversarial Networks*), która łączy architektury w pełni splotowe (bez operacji grupowania, natomiast ze splotem transponowanym w przypadku generatora) z normalizacją partiami i wykorzystaniem optymalizatora *Adam*. Często spotykanym rozszerzeniem jest też wykorzystanie modeli warunkowych (*conditional GAN*) **(Mirza & Osindero, 2014)**, które pozwalają na uzależnienie generowanego rozkładu od dodatkowych danych wprowadzanych na wejściu, np. pożądanej etykiety.

Kluczowym pytaniem jednak pozostaje ustalenie, jakie możliwe zastosowania kryją się za modelami typu *GAN*, że zaczęły się one cieszyć tak dużą popularnością. Jest to pytanie tym bardziej ważne, że sam Goodfellow twierdzi, iż „*zastosowane do obrazów, modele te zdają się tylko generować jeszcze więcej obrazów, a światu nie brakuje obrazów*” **(Goodfellow, 2016)**. Krok dalej wylicza jednak całą listę zastosowań, w tym do zadań, których głównym celem jest właśnie generowanie nowych obrazów. Grupa ta zawiera zarówno sztuczne zwiększanie rozdzielczości zdjęć **(Ledig et al., 2016)**, tworzenie i szczegółową edycję obrazów na bazie zgrubnych szkiców **(Zhu et al., 2016; Brock et al., 2016)**, generowanie terenu w światach wirtualnych **(Guérin et al., 2017)**, jak i imitowanie stylu **(Jing et al., 2017; Isola et al., 2016; Choi et al., 2017)**. Modele generatywne sprawdzają się także w zadaniach, gdzie oczekiwane wyjście może z definicji przybierać wiele poprawnych wartości, na przykład w predykcji kolejnych klatek w filmie **(Lotter et al., 2015)**. Goodfellow podkreśla, że implementacje modeli typu *GAN* bardzo skutecznie weryfikują też faktyczną zdolność do generowania i operowania na wielowymiarowych rozkładach danych, stąd są dobrym testem, czy stosowane modele i architektury pozwalają rzeczywiście uchwycić w wytworzonej reprezentacji elementy istotne dla danej dziedziny problemowej.

Z punktu widzenia tematyki klasyfikacji dźwięku, najciekawsze zastosowania tej grupy modeli wiążą się z ich wykorzystaniem w uczeniu z częściowym nadzorem (*semi-supervised learning*). Modele generatywne umożliwiają w tym wypadku uczenie z niepełnymi danymi lub syntetyczne zwiększanie zbiorów uczących, inkorporując przy generowaniu przykładów dodatkową wiedzę pozyskaną z analizy przykładów nieetykietowanych, których dostępność jest nieporównywalnie większa. O ile w obszarze przetwarzania obrazów postępowanie takie staje się coraz częstsze, to w zastosowaniach dźwiękowych wykorzystanie podejść typu *GAN* właściwie nie zostało jeszcze szczegółowo zbadane i wydaje się szczególnie ciekawym kierunkiem potencjalnych przyszłych badań.

Rozdział 3

Klasyfikacja dźwięku jako zagadnienie uczenia maszynowego

3.1 Definicja problemu klasyfikacji dźwięku

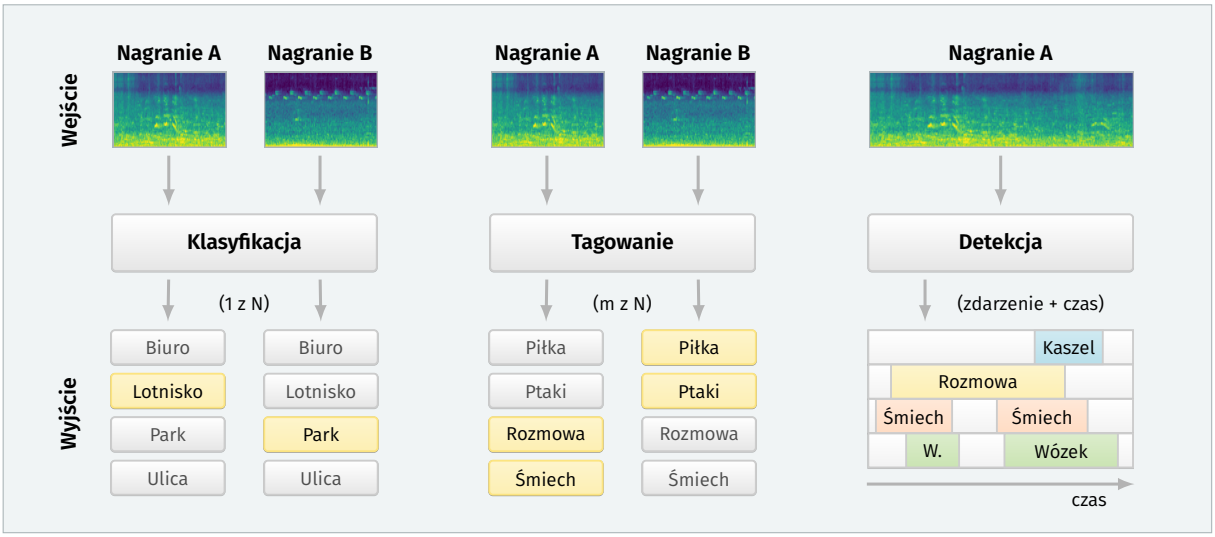
3.1.1 Porównanie zadań w ramach analizy nagrań dźwiękowych

Analiza nagrań dźwiękowych obejmuje wiele zagadnień o mocno zróżnicowanej specyfice. Nawet ograniczając się w zakresie tylko do samego rozpoznawania dźwięku[1], zadania w tym obszarze można podzielić na kilka bardziej szczegółowych wariantów. W powszechnym użyciu często nie są one tak precyzyjnie rozróżniane, ale dla bardziej spójnego opisu adekwatne wydaje się zastosowanie terminologii wykorzystywanej przez Virtanena et al. (2018), która wyróżnia trzy główne typy problemów.

Pierwszym z nich jest **klasyfikacja dźwięku** (*sound classification*), będąca głównym obszarem zainteresowań niniejszej rozprawy. W ramach problemu klasyfikacji zadanie postawione przed modelem polega na wybraniu, która ze zdefiniowanych uprzednio etykiet (klas) najlepiej opisuje przedstawiony fragment. Dla klasyfikacji scen akustycznych rezultatem może być przykładowo określenie, czy kontekst nagrania wskazuje na biuro, ruchliwą ulicę czy łąkę. Wybór jednego z tych kontekstów wyklucza inne, stąd jest to problem typu *single-label*.

Szerszym zadaniem niż klasyfikacja jest **tagowanie** (*audio tagging*), którego celem jest wskazanie wszystkich etykiet adekwatnych dla danego nagrania. Podejście to jest określane również jako *multi-label classification*, stąd występujące czasem niejednoznaczności, gdy mowa ogólnie o klasyfikacji dźwięku. Przykładem tagowania dla fragmentu pozyskanego z rezerwatu przyrody byłoby wskazanie wszystkich występujących w nim gatunków ptaków.

[1] Jeżeli nie zostało to inaczej zaznaczone, to pojawiające się w tym rozdziale odwołania do „dźwięku” obejmują obszar wszelkich treści akustycznych, które nie są mową lub muzyką.



Rys. 3.1: Typy zadań rozpoznawania dźwięku według systematyki Virtanena et al. (2018).

Zarówno klasyfikacja jak i tagowanie mają charakter globalny. Generowana w ich przypadku etykieta wskazuje jedynie na fakt wystąpienia danego zjawiska, natomiast nie określa jego dokładnego umiejscowienia w czasie. W niektórych zastosowaniach informacja ta nie jest tak ważna, podczas gdy dla innych może się okazać kluczowa. Rozpatrując tę kwestię na bazie podanych przykładów – kontekst sceny akustycznej jest zazwyczaj stały dla całości lub przynajmniej dłuższego fragmentu nagrania, ale już zmiany w zestawieniu śpiewających w danym momencie gatunków ptaków są zazwyczaj dużo częstsze. Dopiero **detekcja zdarzeń dźwiękowych** (*sound event detection*) stara się nie tylko odnotować wystąpienie konkretnego zdarzenia, ale też oznaczyć jego czas początkowy i końcowy. Wizualnie różnicę między tymi trzema podejściami przedstawia rysunek 3.1. Warto w tym miejscu zaznaczyć, że w praktyce rozgraniczenie to nie jest jednak takie ostre. Przy założeniu odpowiednio gęstej siatki (np. rzędu kilkuset milisekund), problem detekcji można w dużej mierze sprowadzić do klasyfikacji lub tagowania dokonywanego na stałej długości oknie przesuwającym się po całości nagrania. Chociaż jest to często spotykane podejście, to może się ono nie sprawdzić w niektórych zastosowaniach, dla których bardzo ważna jest wysoka precyzja umiejscowienia zdarzenia w czasie.

Samo pojęcie detekcji zdarzeń dźwiękowych zawiera w sobie wiele konkretnych wariantów zadań o bardzo zróżnicowanym poziomie trudności. Jest to szczególnie widoczne w obszarze bioakustyki, gdzie klasyfikacja binarna na zasadzie „wystąpienie” lub „brak wystąpienia zdarzenia” stanowi zazwyczaj dopiero początkowy krok odsiewający interesujące fragmenty z gąszczu danych pozyskanych w procesie monitoringu pasywnego. Wyniki końcowej analizy mogą natomiast opierać się na bardziej szczegółowych aspektach sygnału, które uwidaczniają się na przykład w formie konkretnego kształtu występującego na spektrogramie. Zależności te dobrze obrazuje systematyka zadań zaproponowana przez Stowella et al. (2016) przedstawiona na rysunku 3.2. Złożoność zadań w tym wypadku

Format wyjścia (złożoność zadania)		Typowe podejścia, zastosowanie	Zalety i wady
(1)	Binarne etykiety (wystąpienie/brak) 	Różne formy klasyfikatorów Oznaczenie wystąpienia gatunku w nagraniu	+ Prosta ocena i oznaczanie danych - Brak umiejscowienia w czasie
(1)	Momenty nagłosu 	Detektory nagłosu (krzywa energii, klasyfikacja ramek) Oznaczenie początków zdarzeń	+ Obsługa zachodzących zdarzeń - Brak czasu trwania (wygłosu)
(2)	Segmentacja monofoniczna aktywności 	Progowanie energii, HMM Oznaczenie aktywności (VAD)	- Scalanie zachodzących zdarzeń
(3)	Segmentacja polifoniczna (multi-mono) 	Faktoryzacja macierzy (NMF) Oznaczenie gatunków	+ Możliwe zachodzenie gatunków - Scalanie w ramach gatunku
(3)	Segmentacja polifoniczna (pełna) 	Uwzględniające zachodzenie Oznaczenie gatunków	+ Obsługa zachodzących zdarzeń
(4)	Prostokąty czas-częst. 	Korelacja wzajemna spektrogramów Wyznaczenie ROI	+ Ograniczenie do obszarów zainteresowania
(5)	Ślady czas-częst. 	Metody detekcji obrazowej Dokładniejsze ROI	- Możliwe rozdzielanie elementów harmonicznych
(5)	Sinusoidy czas-częst. 	Śledzenie wysokości dźwięku Dokładny nasłuch, np. wielorybów	- Tylko dla tonów prostych

Rys. 3.2: Systematyka zadań w zakresie bioakustyki. Adaptacja własna za **(Stowell et al., 2016)**.

rośnie wraz z przechodzeniem od wariantów monofonicznych do polifonicznych (wykrywanie kilku różnych zdarzeń dźwiękowych występujących w tym samym czasie) i ze zwiększaniem stopnia szczegółowości detekcji (od określenia wyłącznie punktów początkowych w czasie do oznaczania konkretnych fragmentów spektrogramu – za pomocą prostokątów lub fragmentów krzywych). Chociaż tego typu dokładna analiza jest zadaniem niezwykle ciekawym pod kątem badawczym, ze względu na istotnie większą złożoność w porównaniu do zadania klasyfikacji, nie będzie szczegółowiej rozpatrywana w dalszej części rozprawy.

3.1.2 Ocena jakości systemu rozpoznawania dźwięku

Występujące zróżnicowanie zadań rozpoznawania dźwięku powoduje, że również prawidłowa ocena i wzajemne porównywanie efektywności systemów analizujących sceny i zdarzenia akustyczne staje się problemem złożonym. Sytuację tę komplikuje fakt, że nawet w przypadku ograniczenia się wyłącznie do pojedynczego typu zadań możliwe jest wykorzystanie wielu potencjalnych miar jakości, które koncentrują się na odmiennych aspektach funkcjonowania systemów rozpoznawania dźwięku. Najbardziej ogólnym i zarazem skomplikowanym z rozpatrywanych zadań jest detekcja zdarzeń dźwiękowych i dlatego na jej przykładzie omówione zostaną główne sposoby opisu jakości uzyskiwanych wyników.

Ocena ilościowa rezultatów działania systemu detekcji dźwięku polega na porównaniu danych referencyjnych z predykcją wygenerowaną przez model. Porównanie to może być dokonywane na jeden z dwóch sposobów **(Mesaros et al., 2016a)**: poprzez analizę następujących po sobie fragmentów nagrania (***segment-based metrics***, czasami też określane jako *frame-based metrics*) lub rozpatrywanie każdego zdarzenia osobno (***event-based metrics***). Rozróżnienie między tymi podejściami obrazują rysunki 3.3 i 3.4.

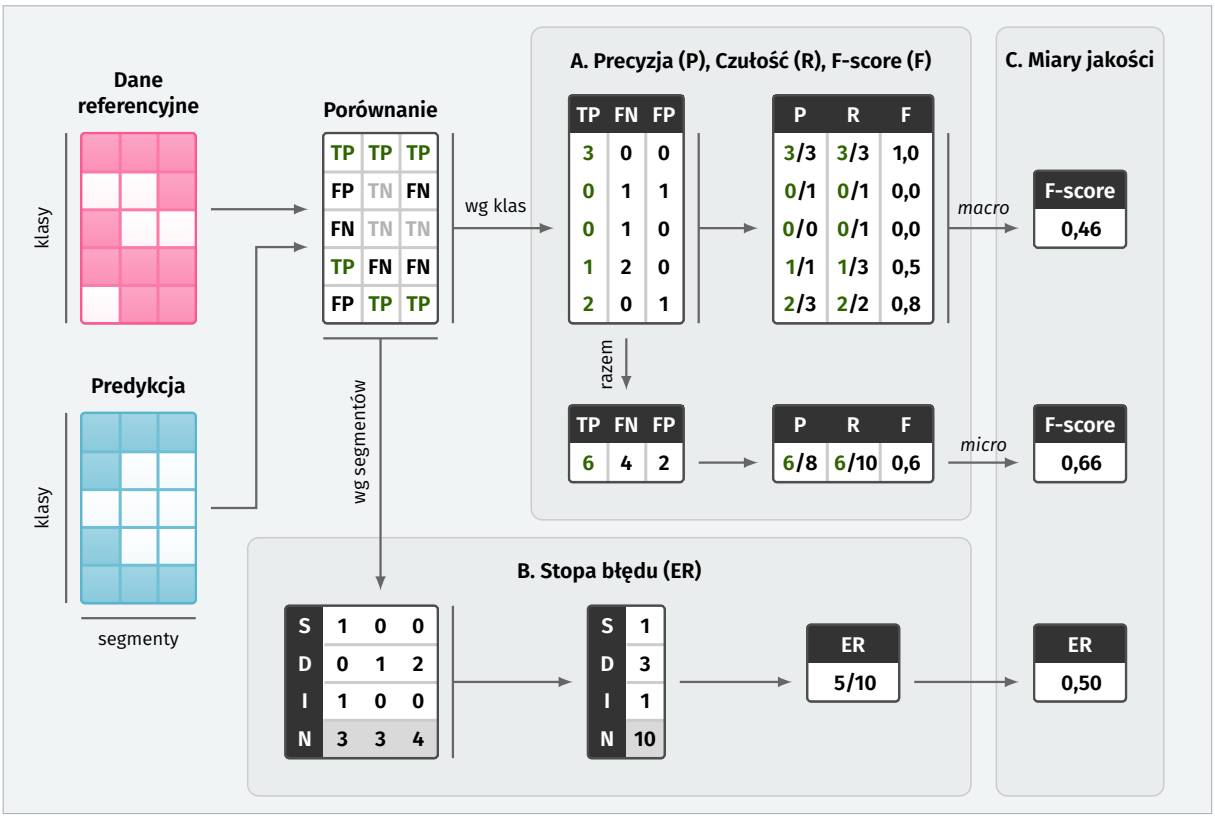
W przypadku analizy segmentów nagranie dzielone jest na fragmenty o zadanej długości (np. 100 ms), którym odpowiadają poszczególne kolumny macierzy wejściowej przedstawionej na rysunku 3.3. Wiersze macierzy przedstawiają z kolei binarne predykcje systemu dla poszczególnych klas zdarzeń dźwiękowych (zdarzenie wystąpiło lub nie wystąpiło w danym segmencie). Porównanie wygenerowanych predykcji z wartościami rzeczywistymi pozwala na wyodrębnienie następujących sytuacji:

- wynik prawdziwie pozytywny (***true positive***, *TP*) - zarówno model i dane referencyjne wskazują na występowanie danego zdarzenia w analizowanym segmencie,
- wynik fałszywie pozytywny (***false positive***, *FP*) - model przewiduje wystąpienie zdarzenia dźwiękowego, które nie jest odzwierciedlone w danych referencyjnych,
- wynik fałszywie negatywny (***false negative***, *FN*) - zdarzenie dźwiękowe, które zostało oznaczone w danych referencyjnych, nie zostało wykryte przez model,
- wynik prawdziwie negatywny (***true negative***, *TN*) - model i dane referencyjne jednocześnie wskazują na brak zdarzenia dźwiękowego (wyniki te są rzadko wykorzystywane w dalszym obliczaniu miar jakości).

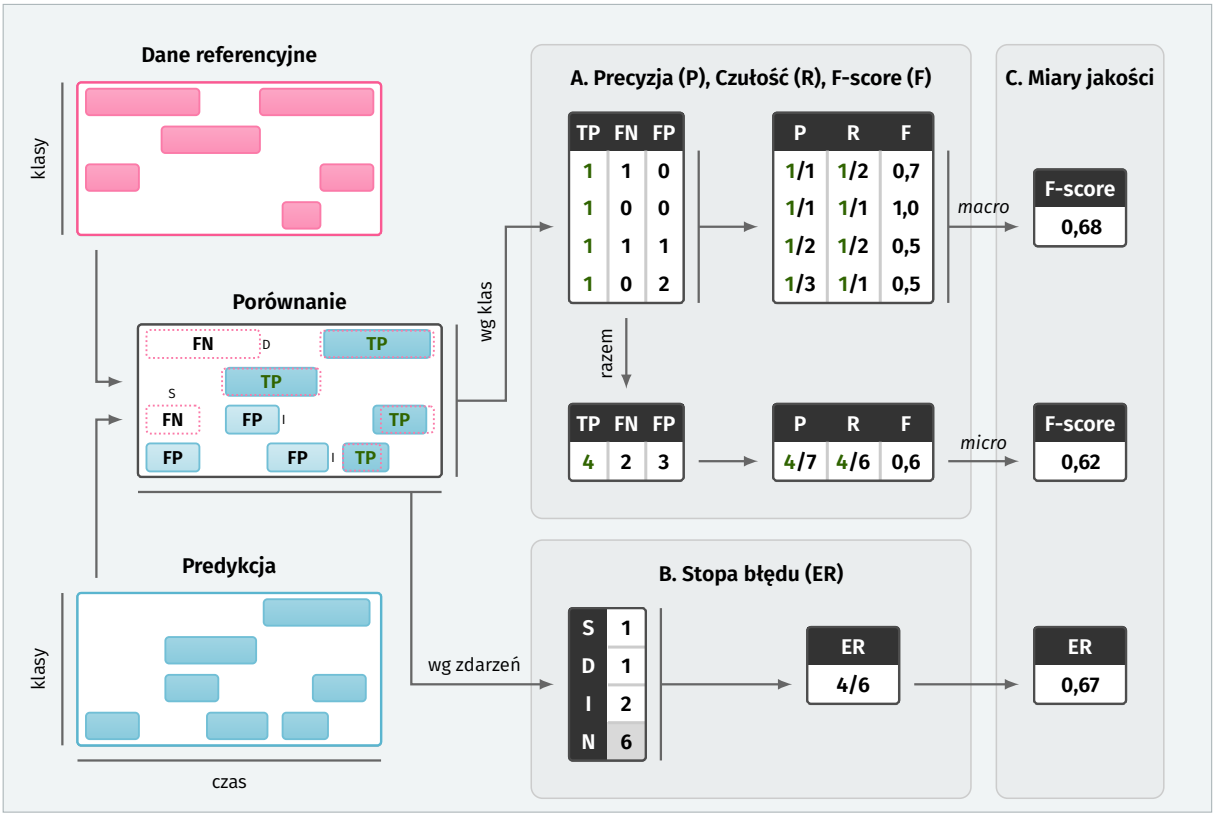
Dla przedstawionego na rysunku 3.4 wariantu opartego na zdarzeniach (*event-based*) predykcja systemu i dane referencyjne porównywane są kolejno dla każdego pojedynczego zdarzenia dźwiękowego, stąd znaczenie poszczególnych scenariuszy jest następujące:

- wynik prawdziwie pozytywny (***true positive***, *TP*) - zdarzenie dźwiękowe wyznaczone przez model występuje w danych referencyjnych z tą samą etykietą i czasem wystąpienia (w ramach ustalonej odgórnie tolerancji czasowej),
- wynik fałszywie pozytywny (***false positive***, *FP*) - zdarzenie dźwiękowe wyznaczone przez model nie odpowiada w danych referencyjnych zdarzeniu z tą samą etykietą i czasem,
- wynik fałszywie negatywny (***false negative***, *FN*) - zdarzenie dźwiękowe oznaczone w danych referencyjnych nie ma swojego odpowiednika w predykcji generowanej przez model.

Ze względu na specyfikę tego wariantu trudno mówić w jego przypadku o wynikach prawdziwie negatywnych (*true negative*). Jeśli chodzi o zakres tolerancji czasowej przyjmowany do oceny, to Mesaros et al. **(2016)** sugerują, że wartości rzędu od ± 100 ms do ± 250 ms



Rys. 3.3: Wyznaczanie miar jakości na podstawie analizy segmentów (*segment-based metrics*). Adaptacja własna na bazie (Mesaros et al., 2016a).



Rys. 3.4: Wyznaczanie miar jakości na podstawie poszczególnych zdarzeń (*event-based metrics*). Adaptacja własna na bazie (Mesaros et al., 2016a).

można uznać za standardowe. W przypadku dłuższych zdarzeń rozsądne może być też proporcjonalne rozszerzenie tolerancji znacznika końca zdarzenia, np. o 50% jego całkowitej długości.

Jeżeli przez *TP*, *FP*, *FN*, *TN* oznaczymy odpowiednio liczby wystąpień wyników prawdziwie pozytywnych, fałszywie pozytywnych, fałszywie negatywnych i prawdziwie negatywnych, to na ich podstawie możliwe jest wyliczenie najczęściej wykorzystywanych miar końcowych w postaci precyzji (***precision***) i czułości (***recall***)[2]:

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \tag{3.1}$$

Precyzja opisuje, w jakim stopniu predykcje generowane przez model są trafne, z kolei czułość określa, jak duża część zdarzeń dźwiękowych jest rzeczywiście wychwytywana przez system. Miarą, która łączy te dwa aspekty ewaluacji jest ***F-score***:

$$F = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{3.2}$$

Zdaniem Mesaros et al. **(2016)** istotną zaletą tej miary jest jej rozpowszechnienie w różnych dziedzinach badawczych. Ważne jest jednak zwrócenie uwagi na dokładny sposób jej wyliczania. Agregacja wyników pośrednich może być przeprowadzona dwojako. Pierwszym podejściem określanym jako *instance-based averaging* lub *micro-averaging* jest zliczanie wszystkich wystąpień wyników prawdziwie pozytywnych dla całego zbioru danych łącznie. W wariancie tym każdy przykład ma taką samą wagę, co powoduje, że klasy częściej występujące mają większy wpływ na uzyskiwany wynik ewaluacji. W przypadku gdy celem jest lepsze oddanie działania systemu dla klas o niewielkiej liczebności, stosowane jest uśrednianie typu *macro* (*class-based averaging*), w którym miary jakości są wyliczane dla każdej klasy osobno i uśredniane dopiero na końcowym etapie. Rozróżnienie to jest o tyle istotne, że wartości liczbowe uzyskiwane w obydwu wariantach mogą się znacząco różnić. Fakt ten dobrze obrazuje panel *C* na rysunku 3.3.

Innym sposobem pomiaru skuteczności działania systemu rozpoznawania dźwięku jest stopa błędu (***error-rate***) **(Mesaros et al., 2016a)**, która zlicza liczbę różnic między predykcją a danymi referencyjnymi:

- *S* (*substitutions*) - liczba przykładów, dla których system wskazał inną etykietę niż poprawna,
- *D* (*deletions*) - liczba przykładów, dla których system nie wskazał zdarzenia, choć powinien (wyniki fałszywie negatywne po odliczeniu substytucji),

[2] W zależności od kontekstu precyzję określa się też jako *positive predictive value* a czułość jako *true positive rate* lub *sensitivity*.

- I (*insertions*) - liczba przykładów, dla których system wskazał zdarzenie, chociaż żadne w rzeczywistości nie wystąpiło (wyniki fałszywie pozytywne po odliczeniu substytucji),
- N - ogólna liczba aktywnych przykładów.

Error rate w takim wypadku wyraża się jako:

$$ER = \frac{S + D + I}{N} \tag{3.3}$$

Przykład wyliczenia tej miary dla wariantów opartych na segmentach i zdarzeniach obrazują odpowiednio panele B rysunków 3.3 i 3.4.

Popularną metodą oceny klasyfikatora, zwłaszcza w zastosowaniach medycznych, jest też analiza za pomocą krzywej ***ROC*** (*receiver operating characteristic*), która dla różnych progów odcięcia zestawia ze sobą udział wyników prawdziwie pozytywnych (*true positive rate*, *TPR*) z udziałem wyników fałszywie pozytywnych (*false positive rate*, *FPR*)[3]:

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{TN + FP} \tag{3.4}$$

Konkretne klasyfikatory wygodniej porównuje się jednak nie za pomocą wykresów, ale w postaci skondensowanej do pojedynczych wartości liczbowych. Do tego celu może służyć wyliczanie obszaru pod krzywą *ROC* (*Area Under Curve*, ***AUC***) lub określanie równej stopy błędu (*Equal Error Rate*, ***EER***), czyli miejsca, w którym udział wyników fałszywie negatywnych ($1 - TPR$) równoważy się liczbowo z udziałem wyników fałszywie pozytywnych (FPR). Tego typu miary oparte na analizie *ROC* są jednak stosowane w ograniczonym stopniu do oceny systemów rozpoznawania dźwięku ze względu na nietrywialne adaptowanie do problemów wieloklasowych **(Fieldsend & Everson, 2005)**, a także fakt, że w wariancie oceny opartej na zdarzeniach nie jest możliwe wyznaczenie wyników prawdziwie negatywnych. Jednym z upraszczających podejść może być potraktowanie rozpoznawania wielu klas zdarzeń dźwiękowych jako kilku problemów binarnych i uśrednienie uzyskanych wartości *AUC* **(Foster et al., 2015)**.

Ostatnią spośród omawianych miar oceny jakości modeli predykcyjnych jest dokładność (***accuracy***), która wyraża stosunek prawidłowych odpowiedzi do łącznej liczby przykładów:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.5}$$

Główną wadą dokładności jest to, że z jednakową siłą uwzględnia wyniki prawdziwie pozytywne i prawdziwie negatywne. W przypadku zadania detekcji, w którym analizowane

[3] Porównanie wartości *TPR* i *FPR* określa się też jako zestawienie *czułość* i $1 - swoistość$.

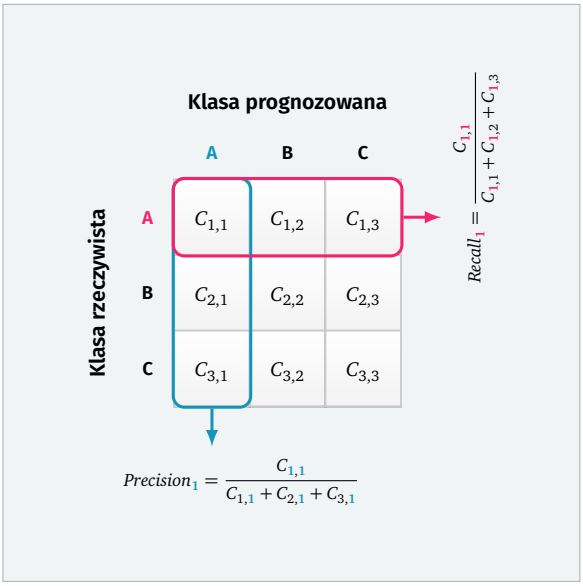
zdarzenia mogą występować stosunkowo rzadko, stan ten jest niepożądany – model, który generuje wyłącznie predykcje typu „brak zdarzenia" może cechować się bardzo wysoką dokładnością przy zerowej użyteczności. Jednym ze sposobów korygowania tej nierównowagi jest wprowadzenie odpowiednich mnożników dla obydwu wariantów w ramach wyliczania dokładności zrównoważonej (***balanced accuracy***, *BACC*) **(Mesaros et al., 2016a)**:

$$BACC = w \cdot \frac{TP}{TP+FN} + (1-w) \cdot \frac{TN}{TN+FP} = w \cdot TPR + (1-w) \cdot (1-FPR) \tag{3.6}$$

Sytuacja jest dużo prostsza w typowym zadaniu klasyfikacji, gdy dla każdego z rozważanych przykładów możliwe jest udzielenie binarnej odpowiedzi, czy został on poprawnie sklasyfikowany. Dokładność wyraża wtedy po prostu stosunek liczby poprawnie udzielonych odpowiedzi (N_{correct}) do całkowitej liczby przykładów (N):

$$ACC = \frac{N_{\text{correct}}}{N}. \tag{3.7}$$

W zadaniach klasyfikacji miara ta jest wykorzystywana najczęściej właśnie ze względu na intuicyjną interpretację. Jej mankamentem jest natomiast przekazywanie ograniczonej ilości informacji. Na dokładniejszą ocenę działania klasyfikatora operującego na ciągu klas $(y_1, \ldots, y_n)$ pozwala dopiero analiza przedstawionej na rysunku 3.5 macierzy pomyłek $\boldsymbol{C}$ (*confusion matrix*), której elementy $C_{i,j}$ oznaczają liczbę przykładów należących do klasy y_i, dla których prognozowaną klasą jest y_j. Wartości na głównej przekątnej odpowiadają prawidłowym predykcjom, natomiast analiza kolumn macierzy $\boldsymbol{C}$ umożliwia wyznaczenie precyzji modelu dla poszczególnych klas:



Rys. 3.5: Macierz pomyłek dla klasyfikatora operującego na zbiorze 3 klas danych.

$$Precision_k = \frac{C_{k,k}}{\sum_{i=1}^{n} C_{i,k}}, \qquad k \in \{1, 2, \ldots, n\}. \tag{3.8}$$

Analogiczne postępowanie przeprowadzone dla wierszy wyznacza odpowiednie czułości:

$$Recall_k = \frac{C_{k,k}}{\sum_{j=1}^{n} C_{k,j}}, \qquad k \in \{1, 2, \ldots, n\}. \tag{3.9}$$

3.2 Metody reprezentacji sygnałów dźwiękowych

Klasyfikator dźwięku można określić jako funkcję, która sygnałowi dźwiękowemu $\boldsymbol{x}$, przedstawionemu w postaci wektora próbek lub atrybutów, przyporządkowuje odpowiadającą mu etykietę y ze zdefiniowanego a priori zbioru klas $\mathbb{Y}$:

$$f_{\text{klas.}}(\boldsymbol{x}) = y\,, \qquad y \in \mathbb{Y}\,. \tag{3.10}$$

Analizowany sygnał dźwiękowy może być reprezentowany na różne sposoby, od których w praktyce zależy trudność zadania stawianego przed klasyfikatorem. Przez analogię do zasady działania układu słuchowego, reprezentacje te można określić jako bardziej lub mniej przetworzone. Pierwotna forma przedstawienia nagrania dźwiękowego w dziedzinie czasu (jako dyskretny przebieg amplitudy sygnału) jest szczególnie wymagająca dla klasyfikatora ze względu na gęstość uzyskiwanej reprezentacji. Przy częstotliwości próbkowania na poziomie 44,1 kHz, względnie krótkie 5-sekundowe nagranie staje się w takim wariancie ciągiem aż 220 500 wartości.

Jedną z możliwości ograniczenia wymiarowości danych wejściowych jest wykorzystanie pośrednich reprezentacji w dziedzinie częstotliwości lub czasu i częstotliwości. Chociaż pozwalają one przybliżyć sygnał dźwiękowy do sposobu jego percepcyjnego rozumienia **(Sottek & Genuit, 2005)**, to zachowują dalej bardzo ogólny charakter i czasem wprowadzane ograniczenie wymiarowości okazuje się być niewystarczające. Z tego powodu przez wiele lat dominującym sposobem postępowania było dalsze kondensowanie zawartej w nagraniu informacji poprzez tworzenie specjalistycznych atrybutów dopasowywanych do konkretnego zadania. Kolejne podrozdziały zajmą się omówieniem obydwu z tych podejść.

3.2.1 Ogólna reprezentacja w dziedzinie czasu i częstotliwości

Dyskretna transformacja Fouriera (*discrete Fourier transform*, ***DFT***) pozwala na przekształcenie sygnału dźwiękowego $\boldsymbol{x} = (x_0, \ldots, x_{N-1})$, próbkowanego z częstotliwością f_s, z dziedziny czasu do dziedziny częstotliwości:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot \exp(-2\pi\mathrm{i}\frac{kn}{N}), \qquad k \in \{0, \ldots, N-1\}\,. \tag{3.11}$$

Dla $k \in \{0, \ldots, \lfloor \frac{N}{2} \rfloor\}$ elementy X_k opisują kolejne składowe widmowe do częstotliwości Nyquista. Ich częstotliwość wyraża się jako $f_k = k \cdot f_\text{s}/N$, amplitudę określa $|X_k|$, natomiast fazę $\arg(X_k)$. Zakładając brak efektów związanych z aliasingiem, powrót do dziedziny czasu możliwy jest za pomocą przekształcenia odwrotnego (*inverse DFT*, ***IDFT***):

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot \exp(2\pi\mathrm{i}\frac{kn}{N}), \qquad n \in \{0, \ldots, N-1\}\,. \tag{3.12}$$

Na analizę zmian widma w czasie pozwala natomiast krótkoczasowa transformacja Fouriera (*short-time Fourier transform*, ***STFT***), polegająca na zastosowaniu DFT na przesuwającym się wzdłuż sygnału oknie czasowym o długości L. Składową k dla ramki m wyznacza w takim wypadku:

$$X_{k,m} = \sum_{n=0}^{L-1} w_n \cdot x_{mL+n} \cdot \exp(-2\pi \mathrm{i} \frac{kn}{L}), \qquad k \in \{0, \ldots, L-1\} \wedge m \in \{0, \ldots, M-1\}, \quad (3.13)$$

gdzie w_n jest mnożnikiem wynikającym z zastosowania wybranej funkcji okna (np. Hanna lub Blackmana), a M liczbą ramek uzyskanych dla sygnału $\boldsymbol{x}$ o długości N: $M = \lfloor \frac{N}{L} \rfloor$. Postępowanie to zakłada, że przeskok (*hop size*) między kolejnymi ramkami jest równy długości okna, ale często stosuje się też wartości mniejsze, w których zachodzenie ramek na siebie umożliwia wygładzenie uzyskiwanego wyniku. Jeżeli za macierz $\boldsymbol{S}$ przyjmiemy:

$$S_{k,m} = |X_{k,m}|^2, \qquad k \in \{0, \ldots, \lfloor \tfrac{L}{2} \rfloor\} \wedge m \in \{0, \ldots, M-1\}, \quad (3.14)$$

to jej wizualizację określa się jako spektrogram mocy (***power spectrogram***). Pozwala ona na reprezentację nagrania względem czasu i częstotliwości. Wartości macierzy $\boldsymbol{S}$ mogą zostać przedstawione w skali decybelowej względem wielkości odniesienia θ_{ref}, tworząc spektrogram przeskalowany[4] (***log-scaled spectrogram***):

$$S_{\mathrm{dB}_{k,m}} = 10 \cdot \log_{10} \left(\frac{S_{k,m}}{\theta_{\mathrm{ref}}} \right). \quad (3.15)$$

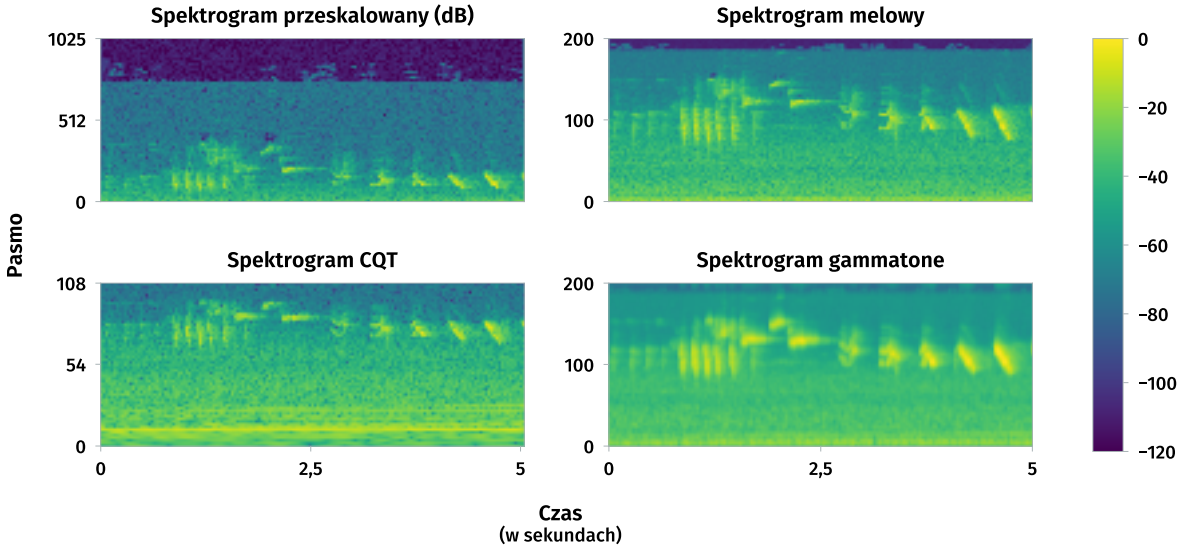
Alternatywnie możliwe jest też zastosowanie krzywej korekcyjnej A wyznaczanej za pomocą funkcji $A(k)$, która pozwala otrzymać spektrogram ważony percepcyjnie (***perceptually weighted spectrogram***)[5]:

$$S_{\mathrm{dBA}_{k,m}} = A(k) + 10 \cdot \log_{10} \left(\frac{S_{k,m}}{\theta_{\mathrm{ref}}} \right). \quad (3.16)$$

W standardowej formule tworzenia spektrogramu poszczególne częstotliwości rozłożone są równomiernie. Jest to o tyle niefortunne, że nie oddaje zbyt dobrze procesu słyszenia u ludzi, który opiera się na nieliniowych filtrach o szerokości pasma rosnącej wraz z częstotliwością. W celu częściowego zniwelowania tej rozbieżności można się posłużyć przekształceniami do logarytmicznej skali częstotliwości (***log-frequency spectrograms***) lub metodami stricte percepcyjnymi. Najczęściej wykorzystywane w zadaniach rozpoznawania dźwięku są reprezentacje oparte na skali melowej, transformacie *CQT* lub filtrach *gammatone*, których wizualne porównanie zawiera rysunek 3.6.

[4] W dalszej części rozprawy, jeśli nie zaznaczono inaczej, w takim właśnie znaczeniu będzie używany termin „spektrogram".

[5] Celem tej korekty jest uwzględnienie występujących w układzie słuchowym różnic w odbiorze głośności dźwięku w zależności od jego częstotliwości. Dokładny sposób korekty definiuje standard IEC 61672.



Rys. 3.6: Porównanie reprezentacji nagrania za pomocą różnych wariantów spektrogramów.

W przypadku pierwszej z nich zamiana częstotliwości wyrażonej w hercach (f_k) na wysokość dźwięku w skali melowej dokonuje się według formuły (O'Shaughnessy, 1999):

$$f_{\text{mel}}(f_k) = 2595 \cdot \log_{10}(1 + f_k/700)\,. \tag{3.17}$$

CQT (*Constant-Q transform*) zakłada natomiast, że częstotliwości środkowe poszczególnych pasm spektrogramu odzwierciedlają stały stosunek między interwałami charakterystyczny dla współczesnej muzyki zachodniej o stroju równomiernym:

$$f_k = f_0 \cdot 2^{k/b}\,, \tag{3.18}$$

gdzie f_0 jest częstotliwością środkową pierwszego pasma, a wartość b określa liczbę interwałów w oktawie.

W sytuacji zastosowania filtrów gammatone wprowadzane jest dodatkowe przetworzenie sygnału opierające się na odpowiedzi impulsowej aproksymującej filtrację zachodzącą w ślimaku kociego układu słuchowego (de Boer & de Jongh, 1978):

$$g(t) = at^{\gamma-1} \cdot \exp(-2\pi\lambda t) \cdot \cos(2\pi f_c t + \phi_c)\,, \tag{3.19}$$

gdzie a jest czynnikiem skalującym amplitudę, t wyznacza czas (w sekundach), γ to rząd filtra (zazwyczaj przyjmuje wartość 4), λ reguluje czas trwania odpowiedzi impulsowej, natomiast f_c i ϕ_c opisują częstotliwość i fazę fali nośnej związanej z poszczególnymi filtrami. Odstępy między częstotliwościami środkowymi mogą być rozłożone zgodnie ze skalą ERB (*equivalent rectangular bandwidth*) (Shao et al., 2009), która odzwierciedla występującą w układzie słuchowym rosnącą szerokość pasm kolejnych filtrów, według zależności:

$$\beta_{\text{ERB}}(f_k) = 0{,}108 \cdot f_k + 24{,}7\,. \tag{3.20}$$

3.2.2 Opis zawartości nagrań za pomocą atrybutów szczegółowych

Celem wprowadzania bardziej szczegółowych atrybutów jest takie ograniczenie wymiarowości danych, aby możliwe było skupienie się na wybranych aspektach różnicujących poszczególne klasy dźwiękowe przy jednoczesnym odrzuceniu jak największej ilości informacji, które nie mają istotnego znaczenia dla danego zadania klasyfikacyjnego. Cel ten jest o tyle wymagający, że aspekty sygnału nieistotne w jednym przypadku, mogą się okazać kluczowe dla uzyskania zadowalającej mocy predykcyjnej w zadaniu o odmiennej specyfice.

W związku z tym na przestrzeni lat zaproponowano dziesiątki, a właściwie setki wariantów szczegółowych atrybutów opisujących nagrania dźwiękowe, których nawet pobieżne omówienie wykraczałoby znacząco poza założone ramy rozprawy. Z tego powodu niniejszy podrozdział przedstawi tylko kilka najistotniejszych przykładów tego typu deskryptorów. Wyczerpujące omówienie tej tematyki zawiera natomiast rozprawa Eybena (2015) i monografia Mitrovicia (2010). W bardziej skondensowanej i praktycznej formie z zasadami tworzenia różnych atrybutów szczegółowych można się też zapoznać poprzez projekty *openSMILE* (Eyben et al., 2010, 2013), *Essentia* (Bogdanov et al., 2013), *pyAudioAnalysis* (Giannakopoulos, 2015) i w ramach standardu *MPEG-7* (Kim et al., 2005).

Atrybuty tworzone na podstawie dziedziny czasu (*temporal features*)

Energia sygnału

Podstawowym i często stosowanym atrybutem opisującym sygnał dźwiękowy na podstawie dziedziny czasu jest jego średnia energia. Zgodnie z oznaczeniami wprowadzonymi w podrozdziale 3.2.1, dla ramki m sygnału o długości L wyraża się ona przez:

$$E_m = \frac{1}{L}\sum_{n=0}^{L-1} x_{mL+n}^2\,, \qquad E_{\mathrm{RMS}_m} = \sqrt{E_m}\,, \qquad m \in \{0,\ldots,M-1\}\,. \tag{3.21}$$

Użycie modeli psychoakustycznych uwzględniających nieliniowe aspekty percepcji pozwala na zamianę energii na poziom głośności, który lepiej opisuje sposób odbierania nagrania przez ludzi.

Zero-crossing rate

Zero-crossing rate (***ZCR***) określa liczbę zmian znaku sygnału $\boldsymbol{x}$ w określonej jednostce czasu. Dla ramki m o długości L przyjmuje postać (Virtanen et al., 2018):

$$ZCR_m = \frac{1}{2}\sum_{n=0}^{L-1} \left|\mathrm{sgn}\left(x_{mL+n}\right) - \mathrm{sgn}\left(x_{mL+n-1}\right)\right|\,, \qquad m \in \{0,\ldots,M-1\} \wedge mL+n > 0\,. \tag{3.22}$$

ZCR pozwala na przykład na odróżnianie typowych sygnałów harmonicznych (małe wartości ZCR) od szumu (wysokie wartości ZCR).

Log attack time

Logarytmiczny czas narastania (*log attack time*) wyznaczany jest jako logarytm dziesiętny z czasu, w którym natężenie dźwięku wzrasta z 20% do 90% maksymalnej wartości (lub inaczej ustalonych progów[6]). Atrybut ten wykorzystywany był często w klasyfikacji instrumentów muzycznych **(Yang & Chen, 2011)**.

Atrybuty tworzone na podstawie widma (*spectral features*)

Spectral centroid

Centroid widma (*spectral centroid*) wyznacza środek ciężkości widma:

$$S_{\text{centroid}_m} = \frac{\sum_{k=1}^{K} f(k) \cdot S_{k,m}}{\sum_{k=1}^{K} S_{k,m}}, \qquad m \in \{0, \ldots, M-1\} \wedge K = \left\lfloor \frac{L}{2} \right\rfloor, \tag{3.23}$$

gdzie $f(k)$ oznacza częstotliwość środkową (w hercach) pasma k, natomiast $S_{k,m}$ jest wartością spektrum wyznaczaną zgodnie ze wzorem 3.14. Centroid widma jest powiązany z takimi własnościami percepcyjnymi jak jasność i ostrość dźwięku, stąd również ten atrybut znajdował szerokie zastosowanie w klasyfikacji instrumentów muzycznych **(Kim et al., 2005)**.

Spectral spread

Drugi moment centralny widma określany jest jako wariancja widmowa (*spectral variance*) lub w standardzie *MPEG-7* jako rozpiętość widmowa (*spectral spread*) **(Kim et al., 2005)**:

$$S_{\text{spread}_m} = \sqrt{\frac{\sum_{k=1}^{K} (f(k) - S_{\text{centroid}_m})^2 \cdot S_{k,m}}{\sum_{k=1}^{K} S_{k,m}}}, \qquad m \in \{0, \ldots, M-1\} \wedge K = \left\lfloor \frac{L}{2} \right\rfloor. \tag{3.24}$$

Atrybut ten informuje o stopniu koncentracji widma wokół centroidu i może pomóc w rozróżnieniu sygnałów tonalnych od szumu. Na analogicznej zasadzie możliwe jest też wyznaczenie momentów wyższego rzędu (*spectral skewness* i *spectral kurtosis*).

Spectral roll-off

Punkt spadku widma (*spectral roll-off point*) definiowany jest jako częstotliwość, poniżej której zawarta jest określona część energii całego widma. Typowe progi to 95%, 90%, 75% i 50% całości energii **(Eyben, 2015)**.

[6] http://essentia.upf.edu/documentation/reference/streaming_LogAttackTime.html.

Spectral flatness

Płaskość widma (*spectral flatness*) jest stosunkiem średniej geometrycznej do średniej arytmetycznej widma **(Eyben, 2015)**:

$$S_{\text{flatness}_m} = \frac{\sqrt[K]{\prod_{k=1}^{K} S_{k,m}}}{\frac{1}{K}\sum_{k=1}^{K} S_{k,m}}, \qquad m \in \{0,\ldots,M-1\} \wedge K = \left\lfloor \frac{L}{2} \right\rfloor. \tag{3.25}$$

Zgodnie z nazwą atrybut ten stara się uchwycić występowanie dużej liczby szczytów w widmie – większe wartości będzie przyjmował między innymi dla sygnałów mocno harmonicznych lub łączących wiele wyraźnych indywidualnych tonów.

Spectral flux

Strumień widmowy (*spectral flux*) jest atrybutem dynamicznym wyrażającym poziom zmiany widma między sąsiadującymi ramkami **(Eyben, 2015)**:

$$S_{\text{flux}_m} = \sum_{k=1}^{K} \left(\frac{S_{k,m}}{\mu_m} - \frac{S_{k,m-1}}{\mu_{m-1}} \right)^2, \qquad m \in \{1,\ldots,M-1\} \wedge K = \left\lfloor \frac{L}{2} \right\rfloor. \tag{3.26}$$

Zależnie od obranych współczynników normalizujących μ_m możliwe jest uzyskanie strumienia widmowego nieznormalizowanego ($\mu_m = 1$) oraz znormalizowanego L_1 lub L_2:

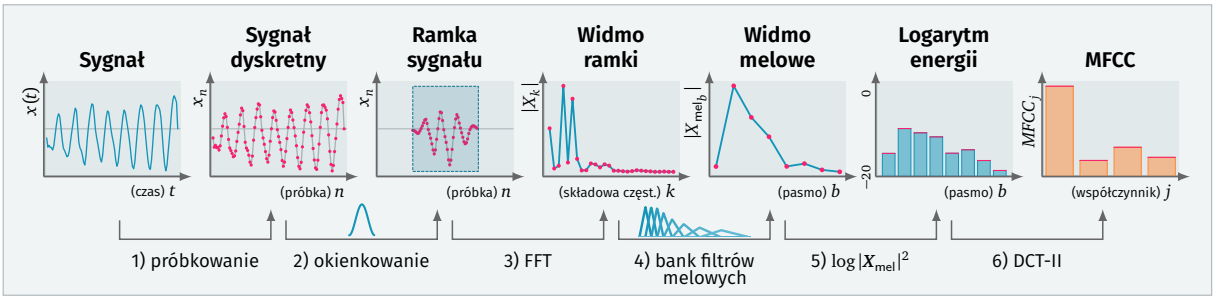
$$L_1 : \ \mu_m = \sum_{k=1}^{K} |S_{k,m}|, \qquad L_2 : \ \mu_m = \sqrt{\sum_{k=1}^{K} S_{k,m}^2}. \tag{3.27}$$

Atrybuty tworzone na podstawie cepstrum (*cepstral features*)

Cepstrum definiowane jest jako wynik zastosowania odwrotnej transformacji Fouriera (w analizowanym kontekście będzie to IDFT) na logarytmie z estymowanego widma sygnału $\boldsymbol{x}$:

$$\boldsymbol{c} = DFT^{-1}\{\log |DFT(\boldsymbol{x})|^2\}. \tag{3.28}$$

Osobliwą cechą takiego przedstawienia jest własność, że cepstrum splotu dwóch sygnałów jest sumą wartości cepstrum wyznaczonych dla każdego z sygnałów osobno. Z tego powodu analiza cepstralna znalazła liczne zastosowania w obszarze przetwarzania mowy, pozwalając na łatwiejszą separację sygnału na część związaną ze wzbudzeniem strun głosowych i z filtracją w ramach traktu głosowego (zgodnie z modelem *source–filter*). Inną przydatną cechą tej reprezentacji jest możliwość skondensowanego przedstawienia kształtu obwiedni widma poprzez współczynniki cepstralne.



Rys. 3.7: Schemat wyznaczania współczynników mel-cepstralnych (MFCC). Adaptacja własna na podstawie (Kim et al., 2005).

Analiza cepstralna łączona jest zazwyczaj z zastosowaniem percepcyjnie motywowanych skal częstotliwości. Przykładem takiego postępowania jest wyznaczanie współczynników mel-cepstralnych (*mel-frequency cepstral coefficients*, ***MFCC***), które przez wiele lat były dominującym sposobem parametryzacji w zadaniach rozpoznawania mowy. Schematyczne przedstawienie tego procesu ilustruje rysunek 3.7.

W postępowaniu tym, w kroku czwartym, widmo wyznaczone dla poszczególnych ramek sygnału jest redukowane za pomocą zespołu trójkątnych filtrów rozłożonych zgodnie ze skalą melową (*mel filter bank*). W kroku szóstym zlogarytmowane wartości energii w pasmach melowych podlegają odwróconej transformacji, analogicznie do wzoru 3.28, z tą różnicą, że zamiast IDFT zwykło się do tego celu wykorzystywać dyskretną transformację kosinusową (***DCT-II***) (Eyben, 2015):

$$MFCC_{j,m} = \sqrt{\frac{2}{B}} \sum_{b=0}^{B-1} \log(S_{\mathrm{mel}_{b,m}}) \cdot \cos\left(\frac{\pi j}{B}\left(b + \frac{1}{2}\right)\right), \tag{3.29}$$

gdzie j i m oznaczają odpowiednio indeks współczynnika (zazwyczaj wykorzystuje się kilkanaście pierwszych MFCC) i ramki, $\boldsymbol{S}_{\mathrm{mel}}$ jest spektrogramem (wzór 3.14) poddanym przetworzeniu przez zespół filtrów melowych, natomiast B określa liczbę pasm (wierszy macierzy $\boldsymbol{S}_{\mathrm{mel}}$). Zależnie od konkretnej implementacji, uzyskane współczynniki mogą być jeszcze dodatkowo filtrowanie w dziedzinie cepstrum (tzw. *liftering*) z odpowiednim współczynnikiem liftrowania λ:

$$\widetilde{MFCC}_{j,m} = MFCC_{j,m} \cdot \left(1 + \frac{\lambda}{2} \sin\frac{\pi j}{\lambda}\right). \tag{3.30}$$

MFCC są zdecydowanie najczęściej spotykaną odmianą współczynników cepstralnych, ale na podobnej zasadzie tworzone mogą być też warianty bazujące na filtrach gammatone (*gammatone cepstral coefficients*, ***GTCC***) (Valero & Alias, 2012), CQT (*constant-Q cepstral coefficients*, ***CQCC***) (Todisco et al., 2016) lub predykcji liniowej (*linear prediction cepstral coefficients*, ***LPCC***) (Reynolds, 1994).

3.3 Rozwój metod klasyfikacji dźwięku

Detekcja i klasyfikacja scen i zdarzeń akustycznych[7], mimo względnie mniejszej popularności niż zagadnienia rozpoznawania mowy czy wyszukiwania informacji w muzyce (*music information retrieval*), jako problem badawczy podejmowana była na przestrzeni lat niejednokrotnie. Z punktu widzenia niniejszej rozprawy istotną cezurą jest rok 2015, w którym opublikowane zostały pierwsze wyniki badań autora **(Piczak, 2015a)**. Wraz z pracami Zhanga et al. **(2015)** i Espiego et al. **(2015)** były one najwcześniejszymi zastosowaniami splotowych sieci neuronowych do klasyfikacji dźwięków środowiskowych. Podejście to spotkało się z na tyle dobrym przyjęciem, że w kolejnych latach w ramach tego obszaru badawczego wyraźnie widoczne stało się odchodzenie od szczegółowych deskryptorów dźwiękowych na rzecz reprezentacji bardziej ogólnych – spektrogramów, a nawet przedstawienia w dziedzinie czasu – i łączenie takich reprezentacji wejściowych z różnymi rozwinięciami modeli splotowych.

Z tego powodu omówienie pokrewnych prac naukowych zostanie przedstawione właśnie z podziałem na dwa etapy. Pierwszy z nich obejmuje okres przed rokiem 2015, wskazując na stan zastany, główne inspiracje i zauważone luki. Drugi z nich odnosi się do prac z okresu 2015–2017, osadzając osiągnięte wyniki w kontekście bieżących osiągnięć dziedziny badawczej i jednocześnie potwierdzając zasadność założonej w rozprawie metody klasyfikacji dźwięku za pomocą splotowych sieci neuronowych poprzez wskazanie na liczne prace rozwijające zaproponowaną koncepcję.

3.3.1 Okres przed rokiem 2015 – istotna rola atrybutów szczegółowych

Dominujące przez lata podejście do klasyfikacji dźwięku można opisać jako proces dwufazowy. W pierwszym kroku zakłada on konwersję nagrania z surowej postaci do przedstawienia jako sekwencji ramek, dla których tworzone są wektory szczegółowych atrybutów mających w możliwie skondensowany i ekspresyjny sposób opisać zawartość danego segmentu. Następnie wybrany klasyfikator uczony jest na bazie tak utworzonych danych wejściowych. Dla każdego etapu, zarówno jeśli chodzi o dobór atrybutów, jak i konkretnego klasyfikatora, możliwe jest skorzystanie z wielu potencjalnych wariantów. Co więcej, wybór szczegółowego zestawu atrybutów jest względnie niezależny od decyzji wykorzystania danego typu klasyfikatora, stąd w literaturze pojawiło się wiele różnych kombinacji tych dwóch elementów. Obrazują to między innymi przeglądowe artykuły Barchiesiego et al. **(2015)**, Chachady i Kuo **(2014)** czy Cowlinga i Sitte **(2003)**.

[7] Zwrot ten utarł się jako dominujące określenie tej dziedziny w związku z tworzeniem się społeczności wokół cyklicznych warsztatów tematycznych – *Detection and Classification of Acoustic Scenes and Events*.

MFCC i deskryptory niskopoziomowe

W rozpatrywanym okresie zdecydowanie najczęściej stosowaną metodą było wykorzystanie atrybutów w postaci współczynników mel-cepstralnych (MFCC) rozszerzonych potencjalnie o inne omawiane w podrozdziale 3.2.2 deskryptory (określane w literaturze często zbiorczo jako *LLD*, *low-level descriptors*), na bazie których w kolejnym kroku dokonywana jest klasyfikacja. Wybieranym rodzajem klasyfikatora zazwyczaj okazuje się być model mieszanin gaussowskich (*Gaussian mixture model*, *GMM*), metoda najbliższych sąsiadów (*k-nearest neighbors*, *k-NN*) lub maszyna wektorów nośnych (*support vector machine*, *SVM*). Alternatywnie możliwe jest też zastosowanie ukrytych modeli Markowa (*hidden Markov models*, *HMM*), które pozwalają dodatkowo na uwzględnienie sekwencyjnego charakteru zmian atrybutów na przestrzeni czasu.

Chu et al. **(2006)** wykorzystali tego typu postępowanie do rozpoznawania 5 kontekstów akustycznych w mobilnych robotach. W swojej pracy porównali działanie klasyfikatorów k-NN, GMM i SVM na bazie zestawu atrybutów, który obejmował 12 MFCC, deskryptory widmowe (*spectral centroid*, *bandwidth*, *assymetry*, *flatness* i *roll-off*), zero-crossing rate i odchylenia standardowe wybranych atrybutów. Modele oparte na SVM okazały się najdokładniejsze, chociaż kosztowne obliczeniowo. Jednocześnie autorzy wskazywali też, że ograniczenie zestawu cech do najbardziej istotnych (*feature selection*) poprawia działanie systemu niezależnie od doboru klasyfikatora.

Na większej liczbie kontekstów akustycznych (24 klasy) skupiali się Eronen et al. **(2006)** porównując efektywność reprezentacji za pomocą atrybutów MFCC, energii w pasmach, predykcji liniowej cepstrum (*LP-cepstrum* i *LPCC*) i wybranych *LLD* (*spectral bandwidth*, *roll-off*, *flux*, *ZCR* i energii) kojarzonych z GMM i metodą k-NN. Z wyjątkiem MFCC, poszczególne typy atrybutów osiągnęły lepsze wyniki w wariancie k-NN, co zdaniem autorów wynika z ich skomplikowanych rozkładów, które nie są zbyt dobrze modelowane przez stosunkowo prosty GMM. Łatwiejsze modelowanie i duża moc predykcyjna MFCC powodują z kolei, że w połączeniu z obranym klasyfikatorem GMM osiągnęły one największą dokładność spośród wszystkich atrybutów.

Mesaros et al. **(2010)** przedstawili natomiast model sekwencyjny opierający się na 3-stanowych HMM. Podejmowany przez nich problem był o tyle trudniejszy, że polegał na detekcji 61 klas zdarzeń w nagraniach izolowanych pochodzących z zewnętrznej bazy efektów dźwiękowych i w rzeczywistych nagraniach środowiskowych przeprowadzonych przez autorów. Do klasyfikacji wykorzystali 16 MFCC wraz z ich zmianami w czasie (Δ-MFCC i Δ^2-MFCC). W przypadku zdarzeń izolowanych dokładność uzyskanego modelu wyniosła 54%, ale dla nagrań w dużo mniej ograniczonych warunkach spadała do 24%, co wskazuje jak trudne dla systemu detekcji zdarzeń dźwiękowych jest poradzenie sobie z szumem i innymi zakłóceniami. Zespół ten kilka lat później rozszerzył tę koncepcję poprzez uczenie

indywidualnych HMM tworzonych zależnie od wykrytej w nagraniu ogólnej sceny akustycznej **(Heittola et al., 2013)**.

Do odmiennych wniosków odnośnie ukrytych modeli Markowa doszli Wang et al. **(2006)**. Ich zdaniem, w przypadku bardzo ograniczonego zestawu atrybutów (tylko *spectral centroid*, *spread* i *flatness*), to wcale nie HMM, ale klasyfikator hybrydowy łączący SVM i k-NN może osiągać lepsze wyniki.

Istotność problemu odpowiedniej selekcji atrybutów omawiali również Muhammad et al. **(2010)**. W przeprowadzanych przez nich eksperymentach ograniczenie zbioru wejściowego z pełnego zestawu MPEG-7 do wybranych 13 atrybutów poprawiało zdolność klasyfikacji za pomocą GMM. Ponownie jednak najefektywniejsze okazywało się połączenie tak wybranego podzbioru z MFCC.

MFCC do klasyfikacji scen akustycznych wykorzystywali też Geiger et al. **(2013)**, pokazując, że współczynniki te są bardzo silne same w sobie i dodanie innych atrybutów (między innymi 35 LLD opartych na widmie, 6 odnoszących się do energii i 3 związanych z tonem podstawowym) poprawia skuteczność klasyfikacji tylko w niewielkim stopniu. Jednocześnie też kolejny raz podkreślone było bardzo dobre działanie SVM, co potwierdzały też wyniki Karbasi et al. **(2011)**, w których metoda ta okazywała się dokładniejsza niż GMM i k-NN. Co więcej, trochę mało zaskakująco, wektor rozszerzony o dodatkowe atrybuty poza MFCC osiągał niewiele lepsze wyniki od samych współczynników mel-cepstralnych. Istotne znaczenie miało natomiast modelowanie trendu zmian w czasie za pomocą zaproponowanych przez autorów *spectral dynamic features* **(Karbasi et al., 2011)**.

Innym rozszerzeniem pomysłu wykorzystania współczynników cepstralnych jest zaproponowane przez Valero i Alíasa **(2012)** zastosowanie filtrów gammatone. Uzyskane tym sposobem współczynniki gammatone-cepstralne (GTCC) charakteryzują się łagodniejszymi przejściami między poszczególnymi filtrami, nieco większą korelacją wzajemną i lepszym odwzorowaniem niskich częstotliwości w porównaniu ze skalą melową. Zdaniem autorów zwłaszcza ten ostatni fakt wpływa na poprawę dokładności klasyfikacji dźwięków środowiskowych względem MFCC. Koncepcyjnie metoda ta jednak odbiega tylko nieznacznie od wykorzystania samych MFCC.

W tym miejscu warto przywołać komentarz Virtanena et al. **(2018)**, który wyraża zdziwienie co do tak dużej popularności MFCC w zakresie klasyfikacji dźwięków środowiskowych. Współczynniki te zostały oryginalnie stworzone w całkiem innym celu, jakim było rozumienie mowy ludzkiej w oparciu o model *source–filter*. MFCC w dużej mierze ignorują część źródła i są względnie niezależne od zmian samej wysokości dźwięku, gdyż bardziej pożądane przy transkrypcji mówców o zróżnicowanych aparatach głosowych jest skupienie się na samej artykulacji. Rozpiętość i znaczenie wysokości dźwięku są natomiast dużo większe w przypadku muzyki i zdarzeń akustycznych spotykanych w środowisku. MFCC

potrafią jednak w zwięzły sposób dosyć dobrze opisać globalną charakterystykę obwiedni częstotliwości, co zdaniem Virtanena et al. (2018) może być główną przyczyną takiego rozpowszechnienia tych współczynników również w obszarze analizy dźwięków środowiskowych.

Metody słownikowe

Odmienne podejście do klasyfikacji dźwięku reprezentują metody, które można zbiorczo określić jako słownikowe. Ich głównym założeniem jest próba reprezentacji nagrania jako złożenia prostszych elementów (atomów lub szablonów)[8]. Dopiero uzyskana tym sposobem forma ograniczonego wymiarowo przedstawienia dźwięku stanowi podstawę do dalszej klasyfikacji.

Wczesnym przykładem takiego postępowania jest automatyczne rozpoznawanie sylab ptasiego śpiewu w nagraniach ciągłych (Anderson et al., 1996), opierające się na jednoetapowym algorytmie DTW (*dynamic time warping*). W metodzie tej problem rozumienia sygnału wejściowego reformułowany jest jako próba jego dopasowania do sekwencji składającej się z określonych szablonowych fragmentów spektrogramu, które mogą podlegać niejednorodnym deformacjom na osi czasu. Pozwala to na dostosowanie działania systemu do zróżnicowanej rozciągłości czasowej poszczególnych wokalizacji. Niestety, praca Andersona et al. ograniczała się do analizy tylko dwóch gatunków ptaków, a pożądane szablony wokalizacji wyszukiwane były ręcznie przez ekspertów.

Chu et al. (2009) do rozpoznawania dźwięków środowiskowych wykorzystali z kolei technikę określaną jako poszukiwanie dopasowujące (*matching pursuit*, *MP*), opierającą się na przedstawieniu sygnału dźwiękowego jako ważonej sumy możliwie małej liczby elementów z ustalonego ponadzupełnego słownika. Taka rzadka reprezentacja upraszcza zadanie, które jest na dalszym etapie stawiane przed klasyfikatorem. Chu et al. wykazali, że w połączeniu z metodami GMM i k-NN atrybuty utworzone na bazie *matching pursuit* uzyskują wyniki lepsze niż MFCC, szczególnie w przypadku zdarzeń dźwiękowych zachodzących w bardzo wąskich pasmach częstotliwości (np. odgłosy owadów), które rozmywają się w szerokich pasmach wyznaczanych przez standardowe implementacje MFCC. Jednocześnie reprezentacje te mają charakter komplementarny, gdyż ich łączne wykorzystanie ze współczynnikami mel-cepstralnymi jeszcze bardziej podnosi dokładność klasyfikacji. Chociaż w ramach cytowanej pracy rozbicie sygnału odbywało się w dziedzinie czasu, to możliwe jest też podobne wykorzystanie poszukiwania dopasowującego dla przedstawień w dziedzinie czasu i częstotliwości (Ghoraani & Krishnan, 2011).

[8] W zależności od konkretnej pracy szablony te mogą być wyrażone w dziedzinie czasu lub czasu i częstotliwości.

Innym podejściem słownikowym jest zastosowanie nieujemnej faktoryzacji macierzy (*non-negative matrix factorization*, *NMF*), która ma na celu wyrażenie wejściowego spektrogramu $\boldsymbol{S}$ jako iloczynu dwóch nieujemnych macierzy $\tilde{\boldsymbol{S}} = \boldsymbol{WH}$, tak aby zminimalizować odległość między $\tilde{\boldsymbol{S}}$ i $\boldsymbol{S}$. Macierz $\boldsymbol{W}$ zawiera kolumnowe wektory bazowe określające wzorce w dziedzinie częstotliwości, natomiast macierz $\boldsymbol{H}$ przedstawia ich aktywacje w czasie. Specyfika tej metody zakłada, że wymnażane elementy są nieujemne, co zazwyczaj skutkuje wytworzeniem się reprezentacji rzadkiej. W odróżnieniu od pracy Chu et al. **(2009)**, elementy słownika w przypadku NMF ustalane są w ramach procesu uczenia. Podejście to popularne było między innymi w transkrypcji muzyki polifonicznej **(Smaragdis & Brown, 2003)**, ale Cotton i Ellis **(2011)** rozszerzyli je w przypadku dźwięków środowiskowych o tworzenie szablonów z większych fragmentów spektrogramów zamiast pojedynczych kolumn (tzw. *convolutive NMF*).

Reprezentacje wieloskalowe

Transformata Fouriera zapewnia informację o lokalizacji w dziedzinie częstotliwości, natomiast nie określa umiejscowienia w czasie. Chociaż jej krótkoczasowy wariant (STFT) pozwala na uzyskanie przedstawienia w dziedzinie czasu i częstotliwości, to rozdzielczość tej reprezentacji jest zależna od obranego rozmiaru okna, który jest stały dla całej analizy. Dłuższe okno zapewnia dokładniejsze odwzorowanie w dziedzinie częstotliwości okupione gorszym umiejscowieniem w czasie.

W przypadku analizy falkowej (*wavelet analysis*) jądrem transformacji jest funkcja (falka) o skończonej energii pozwalająca na analizę czasowo-częstotliwościową. Zastąpienie STFT transformacją falkową umożliwia zastosowanie rozmiarów okna, które są dłuższe dla niskich częstotliwości i odpowiednio krótsze dla częstotliwości wyższych, pozwalając utrzymać relatywny błąd odwzorowania częstotliwości na stałym poziomie **(Loughlin & Cohen, 2010)**.

Lepsza możliwość umiejscowienia w czasie i wykrywania zdarzeń dźwiękowych o krótkotrwałym charakterze są szczególnie adekwatne dla zadań monitoringu akustycznego, stąd Istrate et al. **(2006)** wykorzystali w kontekście telemonitoringu medycznego właśnie dyskretną transformację falkową z macierzystą falką Daubechies do dokładnej detekcji zdarzeń, które w dalszym etapie są klasyfikowane przez system oparty na MFCC. Podobnie Rabaoui et al. **(2008)** zaproponowali nadzór akustyczny opierający się na klasyfikacji atrybutów wywodzących się z analizy falkowej za pomocą jednoklasowych SVM **(Schölkopf et al., 2000)**. Według Valero i Alíasa **(2012)** w tego typu zastosowaniach jeszcze lepsze efekty przynosi wykorzystanie falek opartych o funkcje gammatone, które trafniej oddają charakter funkcjonowania układu słuchowego.

Ciekawe zdanie odnośnie analizy falkowej w klasyfikacji dźwięku prezentują z kolei Chachada et al. (2014), twierdząc, że atrybuty oparte na falkach nie są wcale lepsze od MFCC, lecz tylko porównywalne. Chociaż połączenie obydwu typów reprezentacji poprawia efektywność tworzonych systemów, to zwiększenie złożoności obliczeniowej może być nieuzasadnione w niektórych zastosowaniach. Niemniej zastrzeżenie to nie dotyczy wariantu opartego na falkach gammatone, które zdaniem autorów są szczególnie adekwatne do reprezentowania eksplozywnych zdarzeń dźwiękowych (np. wystrzałów, odgłosów kroków, zamykania drzwi).

Rozwinięciem koncepcji MFCC i analizy falkowej jest transformacja rozpraszająca (*scattering transform*) (Mallat, 2011), którą Andén i Mallat (2014) wykorzystali do klasyfikacji muzyki i fonemów. Metoda ta opiera się na kaskadowym zastosowaniu transformacji falkowych w celu uzyskania reprezentacji, która jest lokalnie niezmienna względem translacji i odporna na deformacje (*time-warping*), zwłaszcza w zakresie wysokich częstotliwości. Ciąg operacji i nieliniowości wprowadzanych przez transformację rozpraszającą można rozumieć jako formę zbliżoną do sieci splotowej, której filtry są z góry założonymi falkami. Cytując autorów: „*scattering transform, podobnie jak MFCC, zapewnia niskopoziomową niezmienną reprezentację sygnału, bez uczenia. Bazuje na uprzedniej informacji odnośnie typu niezmienności, który musi być wyliczony, w tym wypadku względem translacji i deformacji w czasie [...]. Gdy taka informacja nie jest dostępna lub gdy źródła zmienności są dużo bardziej złożone, konieczne staje się uczenie na bazie przykładów, które jest zadaniem odpowiednim dla głębokich sieci neuronowych. W tym sensie obydwa podejścia są komplementarne*" (Andén & Mallat, 2014).

Inspiracje z obszaru rozpoznawania mowy

W czasie gdy głównym podejściem do klasyfikacji zdarzeń akustycznych były systemy oparte na MFCC, to właśnie sieci neuronowe, w tym ich warianty splotowe, zaczynały zdobywać popularność w obszarze rozpoznawania mowy.

Jedną z istotnych prac rozpoczynających ten trend zaprezentowali Lee et al. (2009), skupiając się na problemie uczenia atrybutów z danych audio przy pomocy splotowych sieci przekonań (*convolutional deep belief network*, *CDBN*). Zaproponowany przez autorów CDBN składał się z dwóch splotowych ograniczonych maszyn Boltzmanna (*convolutional restricted Boltzmann machines*, *CRBM*), które operowały na spektrogramach poddanych redukcji wymiarowości za pomocą analizy głównych składowych (*PCA whitening*). Atrybuty wygenerowane w nienadzorowany sposób przez CDBN były następnie wykorzystywane w niezależnych zadaniach klasyfikacji mówców i muzyki. Modele oparte na tego typu reprezentacji okazywały się być dokładniejsze niż te bazujące na MFCC.

Konkretną architekturę splotowej sieci neuronowej (*CNN*) w zadaniu rozpoznawania mowy zweryfikowali natomiast Abdel-Hamid et al. **(2012)**. Była to bardzo prosta z obecnego punktu widzenia sieć, dla której wejściem jest kontekst 15 sąsiadujących ramek o długości 25 ms z przeskokiem co 10 ms. Każdą ramkę opisuje wektor energii rozłożonych w 40 pasmach według skali melowej, wraz z ich różnicą pierwszego i drugiego rzędu po czasie (zmianą między ramkami). Zaproponowana sieć składała się z warstwy dokonującej splotu w dziedzinie częstotliwości (wielkość filtra ustalona została na 8 pasm), warstwy grupowania (*max-pooling*), pojedynczej warstwy gęsto połączonej i warstwy wyjścia typu softmax. Sekwencje generowanych predykcji były następnie modelowane przez HMM. Wniosek, który wyciągnęli autorzy, sprowadzał się do pokazania, że w ramach hybryd z ukrytymi modelami Markowa, to właśnie sieci splotowe (CNN/HMM) cechują się mniejszą stopą błędu niż warianty oparte na głębokich niesplotowych sieciach neuronowych (DNN/HMM).

Dopiero w kolejnych pracach **(Abdel-Hamid et al., 2013, 2014)** można odnaleźć rozszerzenie splotu na obydwa wymiary (w dziedzinie częstotliwości i czasu), co przybliżyło wykorzystywaną architekturę do rozwiązań stosowanych w rozpoznawaniu obrazów. Według Abdel-Hamida et al. **(2013)** istotną różnicą w przetwarzaniu spektrogramów jest odmienne znaczenie osi, które w przypadku standardowych obrazów pozostają równoważne. Występujące na spektrogramach wzorce są charakterystyczne dla poszczególnych zakresów częstotliwości, dlatego zdaniem autorów bardziej adekwatne jest takie ograniczenie współdzielenia wag, aby filtry operujące na różnych pasmach mogły być uczone niezależnie od siebie. Szczegółem, na który warto również zwrócić uwagę, jest zastosowanie wstępnego przyuczania sieci za pomocą ograniczonych maszyn Boltzmanna (*RBM pre-training*) mimo względnie małej głębokości (analizowane sieci składały się maksymalnie z trzech warstw splot–pooling).

Równoległy kierunek obrał w owym czasie Microsoft **(Deng et al., 2013a,b)**, opierając się na regularyzowanych przez dropout sieciach składających się z warstwy splotowej i trzech warstw w pełni połączonych. Wyniki uzyskane w cytowanych artykułach potwierdzały kierunek dalszych badań, wskazując, że w zadaniach rozpoznawania mowy głębokie sieci neuronowe operujące bezpośrednio na wartościach energii widma są trochę dokładniejsze niż te same modele bazujące na MFCC i dużo dokładniejsze niż warianty opierające się na kombinacji MFCC z GMM lub HMM. Do bardzo zbliżonych wniosków o wyższości architektur splotowych nad dotychczasowymi rozwiązaniami doszedł w podobnym czasie również zespół z IBM **(Sainath et al., 2013)**. Podobnie też van den Oord et al. **(2013)** wykazali, że architektury splotowe wykorzystujące spektrogramy mogą uzyskiwać dobre wyniki w predykcji czynników ukrytych (*latent factors*) dla generowania rekomendacji muzycznych opartych na treści utworów.

Sieci neuronowe w klasyfikacji dźwięków środowiskowych

Mimo obiecujących wyników uzyskiwanych przez modele splotowe w obszarze przetwarzania mowy i muzyki, architektury tego typu nie były w ogóle spotykane w rozpoznawaniu dźwięków środowiskowych i bioakustyce. Próby zastosowania sieci neuronowych w tym zakresie ograniczały się do wykorzystania sieci typu *feedforward* i to też zazwyczaj w oparciu o atrybuty bardziej przetworzone, jak MFCC.

Przykładowo, Kons i Toledo-Ronen **(2013)** zajmowali się klasyfikacją 4 rodzajów zdarzeń akustycznych (*tłum, ruch uliczny, aplauz, muzyka*) w nagraniach ulicznych za pomocą głębokiej sieci neuronowej. Wejściem do sieci były wektory 192 atrybutów opisujących 5-sekundowe fragmenty nagrań za pomocą MFCC, odchylenia standardowego MFCC i odchylenia standardowego wartości spektrogramu w poszczególnych pasmach melowych. Sama sieć składała się z 3 warstw ukrytych składających się z 200 neuronów z logistyczną funkcją aktywacji. Autorzy zastosowali również zmodyfikowane przyuczanie za pomocą RBM. Chociaż osiągnięte za pomocą tak skonstruowanej sieci wyniki wykazywały dużą poprawę względem GMM, to nie były już istotnie lepsze od klasyfikacji za pomocą SVM.

Innym przykładem wykorzystania sieci neuronowych jest praca Asgariego et al. **(2014)** zajmująca się predykcją kontekstów akustycznych z nagrań dokonywanych urządzeniem noszonym przez użytkownika w trybie ciągłym (tzw. *life-log*). Celem postawionego zadania było jednoczesne oznaczenie lokalizacji (trzy warianty) i aktywności (cztery możliwości) wykonywanej w danym momencie przez uczestnika badania. Asgari et al. wykorzystali sieć neuronową składającą się z dwóch warstw ukrytych po 1024 neurony sigmoidalne i 2 niezależnych warstw wyjścia typu softmax. Reprezentacją wejściową były w tym przypadku MFCC, standardowy zestaw atrybutów określony w ramach Interspeech 2010 Paralinguistic Challenge **(Schuller et al., 2010)** lub zaproponowane przez autorów atrybuty starające się przedstawić nagranie za pomocą parametrów modelu harmonicznego. W odróżnieniu od poprzedników, Asgari et al. zastosowali przyuczanie nie za pomocą RBM, ale rzadkich autokoderów (*sparse autoencoders*). Omawiana sieć neuronowa charakteryzowała się lepszą dokładnością niż klasyfikator oparty na SVM, zwłaszcza gdy wybranym zestawem atrybutów były parametry harmoniczne.

Ravanelli et al. **(2014)** rozważali z kolei poprawę klasyfikacji treści dźwiękowych za pomocą hierarchicznego uczenia perceptronów wielowarstwowych (*MLP*). W ich przypadku pierwszy etap przetwarzania opierał się na perceptronie składającym się z trzech warstw ukrytych (po 2000 neuronów każda), który generował predykcje dla 40 klas, operując na kontekście 49 ramek wejściowych o długości 25 ms. Atrybuty MFCC wyznaczone dla analizowanej sekwencji ramek były kondensowane za pomocą dyskretnej transformacji kosinusowej (DCT). Predykcje wygenerowane dla różnych fragmentów nagrania agregował następnie z dużo większym przeskokiem czasowym drugi MLP. Można w tym postępowaniu

doszukać się pewnej wczesnej analogii do omawianego na stronie 33 splotu z dylatacją – zaproponowane zestawienie pozwala na zwiększenie skali czasowej, na której operuje druga podsieć, bez nadmiernego zwiększania liczby parametrów całego modelu. Chociaż uczenie hierarchiczne polepszało efektywność systemu, to dokładność klasyfikacji na poziomie ramek pozostawała stosunkowo niska (37%).

Eskperymenty Gencoglu et al. **(2014)** i Khunarsal et al. **(2013)** są najbliższe koncepcyjnie do rozwiązań zaproponowanych w rozprawie w tym sensie, że rezygnowały z reprezentacji na bazie MFCC na rzecz fragmentów spektrogramów. Gencoglu et al. **(2014)** wykorzystali 2- i 5-warstwowe perceptrony jako klasyfikatory, pokazując, że sieć 5-warstwowa z wstępnym przyuczeniem za pomocą RBM osiąga lepszą dokładność niż sieć 2-warstwowa i model łączący GMM/HMM. Co ciekawe, brak takiego przyuczenia powodował, że stosowana sieć osiągała dokładność klasyfikacji dużo gorszą nawet od GMM/HMM, mimo stosunkowo niewielkiej wielkości sieci (70 neuronów na warstwę). Prawdopodobnie mogło to być spowodowane ograniczonym i mocno niezrównoważonym zbiorem uczącym, który przy 61 klasach liczył łącznie tylko 1325 przykładów o bardzo zróżnicowanym czasie trwania. Istotnym ograniczeniem zaproponowanej metody jest też krótki kontekst analizy – klasyfikacja opierała się na fragmentach nagrań o długości 150 ms. Na jeszcze mniejszych fragmentach spektrogramów (16 ms) operowali domyślnie Khunarsal et al. **(2013)**. Dopiero w dodatkowych eksperymentach wykazali rosnącą skuteczność klasyfikacji za pomocą perceptronu wielowarstwowego przy rozszerzaniu okna aż do 512 ms.

To właśnie możliwość wydłużenia kontekstu analizy nagrań była jedną z motywacji podjęcia w rozprawie tematu wykorzystania modeli splotowych do klasyfikacji dźwięków środowiskowych. Chociaż teoretycznie nic nie stoi na przeszkodzie, aby długie sekwencje ramek próbować przetwarzać za pomocą głębokich perceptronów wielowarstwowych, to brak współdzielenia wag wprowadzanego przez warstwy splotowe powoduje w ich przypadku, że modele te zaczynają się znacząco rozrastać wszerz. Poza kwestiami czysto obliczeniowymi utrudnia to też uczenie na małych zbiorach danych. Samo zwiększenie czasowej rozpiętości analizy jest natomiast pożądane, gdyż pozwala na uchwycenie niestacjonarnych aspektów sygnału, które są potencjalnie istotne znaczeniowo (np. charakterystycznej modulacji syreny alarmu powietrznego). Obiecujące wyniki zastosowania sieci splotowych do rozpoznawania mowy pokazały również, że architektury tego typu mogą z dużym prawdopodobieństwem efektywnie operować bezpośrednio na spektrogramach. Powinno to umożliwić zastąpienie dotychczasowych podejść, które wyraźnie rozgraniczały fazę ekstrakcji atrybutów i uczenia klasyfikatora, jednym zintegrowanym modelem.

Problem zbiorów danych i porównywalności

Na zakończenie warto odnieść się do jeszcze jednego wyraźnego problemu występującego w dziedzinie detekcji i klasyfikacji scen i zdarzeń akustycznych, jakim była w omawianym okresie trudność w porównywaniu uzyskiwanych wyników. Cytowane dotychczas prace opierały się na zbiorach danych, które albo były tworzone ad hoc wyłącznie na potrzeby konkretnego eksperymentu, albo pochodziły z zewnętrznych komercyjnych baz efektów dźwiękowych, co w obydwu przypadkach skutkowało mocnym ograniczeniem ich dostępności. Nie zdarzało się również zbyt często, aby rozpowszechniane były implementacje opisywanych algorytmów. Publikowane kolejno artykuły powtarzały ten cykl, stąd wiedza na temat rzeczywistego funkcjonowania poszczególnych podejść, w zależności od zmieniających się warunków, była bardzo fragmentaryczna.

Chociaż na przestrzeni czasu ukazało się kilka zbiorów nagrań dostępnych publicznie, to były one obarczone różnymi mankamentami, które nie pozwoliły na wytworzenie się w tej materii podobnego konsensusu jak w przypadku *ImageNet* dla obszaru rozpoznawania obrazów. Głównymi wadami udostępnianych kolekcji nagrań, poza ich niewielkim rozmiarem, był specjalistyczny charakter lub ograniczenie do pojedynczego kontekstu akustycznego.

W pierwszy nurt wpisują się zbiory *MIVIA* **(Foggia et al., 2014)**, *INRIA NAR* **(Maxime et al., 2014)**, *Dares Amstel* **(Zajdel et al., 2007)** i *RWCP* **(Nakamura et al., 2000)**. *MIVIA* specjalizuje się wyłącznie w dwóch klasach zdarzeń związanych z wypadkami drogowymi. *INRIA NAR* obejmuje wprawdzie 852 przykłady (42 klasy) zdarzeń akustycznych zarejestrowanych w warunkach domowych, ale z celowo ograniczoną jakością odzwierciedlającą rozwiązania sprzętowe dostępne w mobilnych robotach. Z kolei *Dares Amstel* zawiera wyłącznie nagrania inscenizowanych przykładów agresywnych zachowań. Liczebnie największą różnorodnością spośród tej listy cechuje się *RWCP* (9722 przykładów), ale są to dalej nagrania względnie monotematyczne (odgłosy upadania, uderzania przedmiotów zarejestrowane w warunkach bezechowych).

W drugiej grupie (pojedynczy kontekst) znalazły się zbiory *Freiburg-106* **(Stork et al., 2012)**, *DCASE 2013 Events* **(Stowell et al., 2015)** i *TU Dortmund* **(Plinge et al., 2014)**. *Freiburg-106* gromadzi 1479 nagrań aktywności domowych w 22 klasach. *DCASE 2013 Events* koncentruje się na środowisku biurowym (16 klas, 205 zdarzeń), podobnie jak *TU Dortmund* (pomieszczenie konferencyjne).

Kolekcją teoretycznie pozbawioną obydwu niedoskonałości jest *Dares G1* **(Van Grootel et al., 2009)**, który zawiera 120 jednominutowych fragmentów nagrań dokonanych w różnych miejscach publicznych. Niestety, oznaczenia konkretnych zdarzeń akustycznych zostały w jego przypadku przeprowadzone w sposób mocno nieusystematyzowany (blisko 800 klas, wiele synonimicznych, średnio tylko 4 wystąpienia na klasę), na co również zwracają uwagę Mesaros et al. **(2013)**.

Wyraźny był więc brak zbioru, który byłby stosunkowo ogólny w zastosowaniu, a jednocześnie zawierał pliki dźwiękowe odzwierciedlające zróżnicowane konteksty akustyczne i sprzętowe. Interesującym źródłem pozyskania tego typu materiałów okazał się zainicjowany na Uniwersytecie Pompeu Fabra projekt *Freesound* **(Font et al., 2013)**, który gromadzi udostępniane na zasadzie licencji Creative Commons nagrania użytkowników wraz z wprowadzonymi przez nich opisami. Na możliwości płynące z takiego rozwiązania wskazali Stowell i Plumbley **(2013)** przedstawiając *freefield1010* zawierający 10-sekundowe fragmenty dźwiękowe pochodzące właśnie ze strony *Freesound*. Niestety, załączone w jego ramach metadane bazowały bezpośrednio na podanych przez użytkowników opisach i nie były dalej weryfikowane, co skutkowało jakością niewystarczającą do typowych celów badawczych. Dopiero opublikowany pod koniec 2014 roku *UrbanSound8K* **(Salamon et al., 2014)** był w stanie lepiej wykorzystać ten potencjał.

Zbiór *UrbanSound8K* został sporządzony przez ręczne przeszukanie i oznaczenie około 3000 plików źródłowych (60 godzin materiału) pod kątem występowania 10 klas zdarzeń akustycznych spotykanych w warunkach ulicznych. Wystąpienia te zostały następnie podzielone na około 4-sekundowe fragmenty, co w rezultacie zapewniło bazę 8732 przykładów. Chociaż miejscami dalej można mieć pewne zastrzeżenia co do oznaczeń zgromadzonych w nim nagrań, to trzeba przyznać, że *UrbanSound8K* był istotnym przełomem jakościowym i mocno wspomógł porównywalność w tej dziedzinie badawczej. Na bardzo podobnej zasadzie powstał też zbiór *ESC-50*, sporządzony w ramach prac autora **(Piczak, 2015b)**. Szczegółowym jego omówieniem zajmie się rozdział 4.

3.3.2 Lata 2015–2017 – uczenie na reprezentacjach ogólnych

Okres lat 2015–2017 charakteryzował się dynamicznym ożywieniem w obszarze rozpoznawania dźwięku. Główne zmiany, które zostały zapoczątkowane w tym czasie, można podsumować w następujących punktach omawianych w niniejszym podrozdziale:

- wyodrębnienie i integracja społeczności badawczej zajmującej się tematyką rozpoznawania dźwięku wokół warsztatów *Detection and Classification of Acoustic Scenes and Events*,
- poprawa sytuacji w zakresie dostępności zbiorów danych i porównywalności metod,
- popularyzacja wykorzystania sieci splotowych do klasyfikacji dźwięku połączona z rozwojem nowych koncepcji.

Proces skupiania się społeczności badawczej

Tematyka rozpoznawania dźwięku do niedawna nie mogła się pochwalić ani jednym miejscem, które można by określić jako zajmujące się tylko tym obszarem badawczym. Dlatego

też szczególnie wartościowa okazała się inicjatywa organizacji warsztatów *Detection and Classification of Acoustic Scenes and Events*[9] podjęta przez Tampere University of Technology we współpracy z Queen Mary University of London, University of Surrey i IRCCyN.

Pierwsze kroki w tym kierunku poczynione zostały w roku 2013 poprzez zaproponowanie otwartego zadania benchmarkowego *IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events* (DCASE 2013) **(Stowell et al., 2015)**, które polegało na porównaniu nadsyłanych metod klasyfikacji scen i zdarzeń akustycznych na ujednoliconym zestawie zbiorów danych. Ze względu na ich bardzo ograniczony charakter nie było możliwe wyciągnięcie jednoznacznych konkluzji na temat efektywności poszczególnych metod[10], ale sama inicjatywa przetarła szlak do organizacji kolejnej edycji konkursu, tym razem połączonej z częścią warsztatową – DCASE 2016 **(Mesaros et al., 2018)**.

W momencie finalizowania treści rozprawy odbyły się już dwie edycje tego wydarzenia (w roku 2016 i 2017), a kolejna została zapowiedziana na koniec roku 2018. Po poprawieniu pierwszych niedociągnięć, inicjatywa ta stała się miejscem gromadzącym społeczność skupioną wokół tematu rozpoznawania dźwięku, zarówno od strony akademickiej, jak i przemysłu. Poza umożliwieniem bardziej transparentnego porównywania metod klasyfikacji scen i zdarzeń dźwiękowych, warsztaty DCASE przyczyniły się również do popularyzacji samej dziedziny badawczej, czego efektem jest wydanie specjalne *IEEE/ACM Transactions on Audio, Speech, and Language Processing* poświęcone wyłącznie tej tematyce **(Richard et al., 2017)**, czy pierwsza monografia opisująca ten obszar naukowy **(Virtanen et al., 2018)**.

Zwiększenie dostępności zbiorów danych

Kolejnym przejawem poprawy sytuacji w dziedzinie klasyfikacji dźwięku jest stopniowe wypełnianie luki w publicznej dostępności zbiorów nagrań umożliwiających porównywanie systemów rozpoznawania dźwięku między sobą. Istotną rolę w tym procesie odegrały wspomniane wcześniej *UrbanSound8K* **(Salamon et al., 2014)** i *ESC-50* **(Piczak, 2015b)**. Chociaż zbiory te nie są pozbawione mankamentów związanych z ograniczoną ilością materiału źródłowego, to z perspektywy czasu można ocenić, że spotkały się one z dobrym przyjęciem społeczności, stając się nieformalnymi poziomami odniesienia dla kolejnych publikacji[11].

[9] http://dcase.community.
[10] Oceny modeli w przypadku klasyfikacji scen dokonywane były na łącznie 100 przykładach, po 10 na klasę.
[11] Na koniec roku 2017 zestawienie modeli walidowanych na zbiorze *ESC-50* liczyło kilkadziesiąt pozycji: https://github.com/karoldvl/ESC-50.

Do grona zasobów dostępnych publicznie[12] dołączyły w międzyczasie również kolekcje *CICESE Sound Events* **(Beltrán et al., 2015)**, *LITIS Rouen* **(Rakotomamonjy & Gasso, 2015)**, przygotowane przez zespół z Tampere University of Technology zbiory konkursowe DCASE (*TUT Acoustics Scenes*, *TUT Sound Events*) **(Mesaros et al., 2016b)** oraz AudioSet **(Gemmeke et al., 2017)**. Szczególnie tej ostatniej inicjatywie należałoby się słowo komentarza.

AudioSet jest podjętym przez Google przedsięwzięciem o skali dotychczas niespotykanej w rozpoznawaniu dźwięków. Zbiór ten składa się z hierarchicznej ontologii 635 klas akustycznych wykorzystanych do oznaczenia blisko 2 milionów 10-sekundowych fragmentów filmów z serwisu YouTube. Powstanie takiego zasobu, przewyższającego liczebnością wszystkie dotychczasowe kolekcje o kilka rzędów wielkości, otwiera całkowicie nowe możliwości dla rozwoju dziedziny. AudioSet posiada jednak kilka mankamentów, które nie pozwalają uznać go za zbiór doskonały.

Po pierwsze, ze względów licencyjnych dostarczany jest on wyłącznie w formie metadanych i linków do fragmentów filmów, bez surowej treści dźwiękowej. Rozwiązanie takie mocno utrudnia rozpoczęcie korzystania z tego zbioru i jednocześnie powoduje, że podlega on ciągłej „erozji" związanej z wygasaniem części linków[13].

Drugi aspekt związany jest z samą ontologią. AudioSet jest wyjątkowo niezrównoważony – najbardziej ogólne kategorie, „muzyka" i „mowa", liczą ponad milion przykładów każda, w czasie gdy znajdująca się na końcu „szczoteczka do zębów" zawiera tylko 127 fragmentów.

Trzecim problemem tego zbioru jest słaba jakość adnotacji. Proces oznaczania tak dużej ilości materiału dosyć naturalnie wymusza pewne kompromisy, ale w przypadku AudioSetu znacząca część klas nie spełnia założonego odgórnie kryterium jakości (więcej niż 50% poprawnych etykiet), do czego otwarcie przyznają się autorzy, zapowiadając poprawioną wersję.

Mimo to, pierwsze prace bazujące na tym zbiorze pokazały już, że problem słabych etykiet (*weak labels*) można zrekompensować samą ilością danych **(Kumar et al., 2017)**. Możliwe też, że w przyszłości uda się poprawić wiele z wymienionych braków poprzez inicjatywę Music Technology Group z Uniwerystetu Pompeu Fabra, autorów *Freesound*, określaną jako *Freesound Datasets* **(Fonseca et al., 2017)**. Jej pierwszym założonym celem jest zorganizowanie dużego przedsięwzięcia crowdsourcingowego do oznaczania treści występujących w ramach *Freesound* wedug ontologii przyjętej przez *AudioSet*. Istnieje więc realna

[12] Aktualne zestawienie zbiorów danych w dziedzinie rozpoznawania dźwięku prowadzi Toni Heittola z Tampere University of Technology: http://www.cs.tut.fi/~heittolt/datasets.

[13] Według nieoficjalnych szacunków podanych przez Dana Ellisa, jednego z głównych autorów, dotyczy to nawet do 1% filmów miesięcznie.

szansa, że w perspektywie kilku najbliższych lat w dziedzinie rozpoznawania dźwięku uda się wypracować konsensus podobny do sytuacji z ImageNet.

Upowszechnienie wykorzystania sieci splotowych

Prace autora **(Piczak, 2015a)**, Zhanga et al. **(2015)** i Espiego et al. **(2015)**, opublikowane równolegle w 2015 roku, stanowiły pierwsze propozycje wykorzystania sieci splotowych do klasyfikacji dźwięków środowiskowych. Dokładniejsze omówienie samej koncepcji przedstawionej w artykule autora zawiera rozdział 5, ale w tym miejscu warto odnieść się pokrótce do głównych podobieństw i różnic między zaprezentowanymi podejściami.

Wszystkie z omawianych modeli opierały się na połączeniu splotu i grupowania, po których następowało dalsze przetwarzanie za pomocą warstw gęstych (w pełni połączonych). Nie wliczając warstwy wyjścia, są to: dwie warstwy splot–max-pooling i dwie gęste w modelu autora, dwie warstwy splot–subsampling i jedna gęsta dla Zhanga et al., jedno złożenie splot–max-pooling i cztery warstwy w pełni połączone w eksperymencie Espiego et al. Patrząc przez pryzmat dzisiejszych standardów wizji komputerowej były to więc sieci względnie płytkie, ale z pewnością nie odbiegały znacząco od pierwszych prób w obszarze przetwarzania mowy.

W miarę zbliżona w każdym przypadku była też wejściowa reprezentacja danych. Model autora operował na fragmentach spektrogramów o rozmiarze 60×41, Zhang et al. przyjęli wymiary 40×52, natomiast Espi et al. analizowali wariant 129×30.

W porównaniu do prac Zhanga et al. i Espiego et al., główne różnice pracy autora sprowadzają się do wykorzystania filtrów wertykalnych (57×6) zamiast kwadratowych (od 5×5 do 9×9), a także wprowadzenia dropoutu oraz aktywacji typu ReLU. Charakterystyczną cechą jest również, inspirowane artykułem Abdel-Hamida et al. **(2013)**, wykorzystanie drugiego kanału „*delta*”, zawierającego różnicę spektrogramu po czasie.

Potencjał modeli splotowych w klasyfikacji dźwięku został szybko dostrzeżony, co bardzo dobrze obrazują statystyki głównego zadania DCASE 2016 – klasyfikacji scen akustycznych[14]. Spośród 49 rozwiązań zgłoszonych w roku 2016, ponad połowa opierała się na sztucznych sieciach neuronowych, z mniej więcej równym podziałem na warianty splotowe i niesplotowe. W kolejnej edycji (DCASE 2017[15]) udział ten był jeszcze wyraźniejszy – splotowe sieci neuronowe można było odnaleźć w więcej niż połowie z 97 nadesłanych modeli. Co ciekawe, tylko jedno zgłoszenie w pierwszej piętnastce rankingu obyło się bez zastosowania jakiejkolwiek formy sieci splotowych.

Podobny proces przemiany dotknął też obszaru bioakustyki, gdzie w ramach zadania BirdCLEF2016 **(Goëau et al., 2016)**, zajmującego się klasyfikacją śpiewu 999 gatunków

[14] http://www.cs.tut.fi/sgn/arg/dcase2016/task-results-acoustic-scene-classification.
[15] http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-acoustic-scene-classification-results.

ptaków z Ameryki Południowej, modele splotowe pojawiły się po raz pierwszy **(Sprengel et al., 2016; Piczak, 2016; Tóth & Czeba, 2016)**. Architektury tego typu odegrały dominującą rolę również w późniejszym *Bird Audio Detection Challenge 2016–2017* **(Stowell et al., 2016)**.

Przy tak dużej liczbie nowych publikacji w obszarze wykorzystania splotowych sieci neuronowych do klasyfikacji dźwięku trudno jest odnieść się do wszystkich prac bardziej szczegółowo. Można jednak w bardzo skondensowany sposób wskazać kilka trendów badawczych i płynących z nich wniosków:

- prostsze architektury mogą się dobrze sprawować w przypadku mniejszych zbiorów danych **(Phan et al., 2016)**,
- dla skali osiąganej przez AudioSet widoczny jest już wzrost dokładności głębszych modeli (kolejno AlexNet, VGG, Inception, ResNet-50) **(Hershey et al., 2017)**,
- najdokładniejsze systemy zazwyczaj tworzą komitety klasyfikatorów, również uzupełniając sieci splotowe innymi metodami **(Eghbal-Zadeh et al., 2016)**,
- chociaż uczenie modeli rekurencyjnych nie jest łatwe, to mogą być one rozszerzeniem wariantów splotowych **(Parascandolo et al., 2016, 2017)**,
- ważnym etapem uczenia jest umiejętne syntetyczne zwiększanie ilości danych (*data augmentation*) **(Salamon & Bello, 2017; Takahashi et al., 2016)**,
- sieci splotowe można wykorzystać na jeszcze ogólniejszych reprezentacjach niż spektrogramy (tzw. uczenie *end-to-end*), ale jest to zadanie trudniejsze **(Tokozume & Harada, 2017; Dai et al., 2017)**, mimo że modele splotowe z powodzeniem potrafią aproksymować przekształcenie do spektrogramu **(Tax et al., 2017)**,
- nowym zagadnieniem jest wykorzystanie informacji przestrzennej z nagrań wielokanałowych, dotychczas ograniczone dostępnością danych **(Adavanne et al., 2017)**,
- ze względu na specyfikę zbiorów danych ważnym praktycznym problemem badawczym będzie poprawa efektywności uczenia na zaszumionych etykietach (*weak/noisy labels*) **(Kumar & Raj, 2017b)**,
- uczenie modeli akustycznych jest też możliwe poprzez innowacyjne wykorzystanie transferu wiedzy z modeli wizyjnych bez konieczności pozyskiwania nowych danych etykietowanych **(Aytar et al., 2016; Arandjelović & Zisserman, 2017a,b)**,
- w praktycznych zastosowaniach ważnym aspektem będzie też wykorzystanie lekkich wariantów modeli w sieciach inteligentnych sensorów z lokalnym przetwarzaniem **(Abeßer et al., 2017; Mydlarz et al., 2017)**[16].

[16] Innym przykładem takiego zastosowania jest praca autora typu *proof-of-concept*, implementująca ograniczony model splotowy na komputerach jednopłytkowych *Raspberry Pi* dla celów monitoringu rezerwatów przyrody: https://github.com/karoldvl/EARS, https://sensingclues.com/serval/.

Głównym wnioskiem płynącym z podsumowania ostatnich lat w dziedzinie rozpoznawania dźwięku jest podkreślenie jej dynamicznego dojrzewania do stania się pełnoprawnym obszarem badawczym. Jednocześnie też można już z dużą dozą pewności powiedzieć, że splotowe sieci neuronowe również w tym obszarze przyjęły się jako ważne narzędzie. Fakt ten dodatkowo potwierdza tezę stawianą w rozprawie, że modele splotowe mogą być z powodzeniem wykorzystane do klasyfikacji dźwięków środowiskowych mimo ich odmiennego, mniej ustrukturyzowanego charakteru niż w przypadku mowy i muzyki.

Rozdział 4

ESC – zbiór nagrań środowiskowych[1]

4.1 Wprowadzenie

Jedną z przeszkód w rozwoju metod klasyfikacji scen i zdarzeń akustycznych jest przedstawiona w podrozdziale 3.3.1 silna fragmentacja wysiłków badawczych i duża trudność w ich porównywaniu czy odtworzeniu. Większość eksperymentów, do których odnosiły się przeglądowe prace Barchiesiego et al. (2015) oraz Chachady i Kuo (2014), przeprowadzana była na bardzo specyficznych zbiorach danych o małej wielkości lub (częściowo) komercyjnym charakterze. Ograniczona liczba dostępnych publicznie zbiorów, a także trudność z uzyskaniem kodu źródłowego do większości publikacji powodują, że odtworzenie prezentowanych wyników jest trudniejsze, niż powinno być. Sytuacja ta kontrastuje mocno z obszarem przetwarzania obrazów, gdzie zbiory takie jak *MNIST*[2], *CIFAR*[3], a z czasem też *ImageNet*[4], stały się standardowymi wyznacznikami dla porównywania efektywności poszczególnych metod klasyfikacji obrazów między sobą.

Z tego też względu celem przyświecającym tworzeniu zbioru nagrań omawianego w niniejszej rozprawie było ułatwienie prowadzenia bardziej transparentnych badań w dziedzinie klasyfikacji dźwięków środowiskowych poprzez:

- publiczne udostępnienie zbioru nagrań, który mógłby posłużyć jako odniesienie dla porównywania efektywności metod klasyfikacji dźwięku,
- zaprezentowanie szacunkowego punktu odniesienia w postaci dokładności klasyfikacji nagrań pochodzących z tego zbioru uzyskiwanej w eksperymentach z udziałem ludzi,

[1] Wyniki prac autora związane z przygotowaniem i analizą opisywanego zbioru nagrań środowiskowych zostały przedstawione w ramach publikacji konferencyjnej „*ESC: Dataset for Environmental Sound Classification*” (Piczak, 2015b) oraz towarzyszących jej materiałów uzupełniających (https://github.com/karoldvl/paper-2015-esc-dataset/). Niniejszy rozdział rozszerza to omówienie o pogłębioną część eksploracji zbioru danych, a także uaktualnia je o zestawienie publikacji wykorzystujących zbiór *ESC* jako zbiór referencyjny.

[2] http://yann.lecun.com/exdb/mnist/.

[3] https://www.cs.toronto.edu/~kriz/cifar.html.

[4] http://www.image-net.org/.

- porównanie tak otrzymanych wyników z dokładnością klasyfikacji uzyskiwaną przez typowe klasyfikatory uczenia maszynowego wykorzystujące najczęściej spotykane atrybuty,
- udostępnienie kodu źródłowego w formacie *Jupyter Notebook* z rozszerzoną analizą zbioru danych, pozwalającego na łatwą replikację uzyskanych wyników.

4.2 Metodyka tworzenia zbioru

Założenie o możliwie nieograniczonym licencyjnie rozpowszechnianiu tworzonego zbioru wymuszało pozyskanie materiału źródłowego na jeden z dwóch sposobów: albo poprzez jego nagranie we własnym zakresie, albo poprzez przetworzenie dostępnych w Internecie treści o odpowiednio otwartej licencji. W przypadku pierwszego rozwiązania ewidentną zaletą jest pełna kontrola nad procesem pozyskiwania materiałów źródłowych, ich jakością i wykorzystywanym sprzętem, a także całkowita dowolność dalszego licencjonowania. Główną przeszkodą w tym wypadku jest niestety pracochłonność uzyskania tą metodą zbioru, który byłby odpowiednio zróżnicowany pod kątem typów zdarzeń i warunków nagrywania (geograficznych, sprzętowych).

Dużo bliższe rzeczywistym sytuacjom spotykanym przez typowego użytkownika jest wykorzystanie nagrań z baz tworzonych i nadzorowanych właśnie przez społeczność. Przykładem takie przedsięwzięcia jest projekt *Freesound*[5], który udostępnia nagrania na bazie licencji typu *Creative Commons*. W związku z tym prezentowany zbiór nagrań środowiskowych ***ESC*** (*Environmental Sound Classification*), podobnie jak *UrbanSound8K*[6], opiera się właśnie na treści dźwiękowej pochodzącej z bazy *Freesound*.

ESC udostępniany[7] jest w trzech częściach: *ESC-50*, *ESC-10* i *ESC-US*. Główną z nich jest *ESC-50* – etykietowany zbiór 50 klas dźwięków środowiskowych licencjonowany na warunkach *Creative Commons Attribution Non-Commercial*. Klasy zdarzeń zawarte w jego ramach zostały wybrane arbitralnie, starając się zachować równowagę między głównymi kategoriami dźwięków, a także biorąc pod uwagę ograniczenia w liczebności i różnorodności nagrań źródłowych, jak i subiektywną ocenę użyteczności i wyrazistości poszczególnych klas. Pozyskanie materiałów dźwiękowych z bazy *Freesound* polegało na wykonaniu zapytań o treści związanej z konstruowanymi klasami. Tak wygenerowane wyniki były następnie ręcznie oceniane i weryfikowane przez autora poprzez oznaczanie fragmentów zawierających zdarzenia dźwiękowe należące rzeczywiście do danej klasy. Na podstawie sporządzonych oznaczeń utworzone zostały 5-sekundowe fragmenty (krótsze zdarzenia były

[5] https://freesound.org/.
[6] https://serv.cusp.nyu.edu/projects/urbansounddataset/urbansound8k.html.
[7] http://dx.doi.org/10.7910/DVN/YDEPUT. *ESC-50* również przez: https://github.com/karoldvl/ESC-50.

odpowiednio uzupełniane ciszą), które podlegały konwersji do ujednoliconego formatu[8]. Całość zbioru została następnie podzielona na pięć równolicznych podzbiorów walidacyjnych w ten sposób, aby zagwarantować, że fragmenty nagrań pochodzące z tego samego materiału źródłowego umieszczone zostaną w tym samym podzbiorze.

4.3 Opis zawartości zbioru

ESC-50 składa się z 2000 etykietowanych nagrań należących do 50 klas (40 przykładów na klasę). Klasy te tworzą pięć luźno zdefiniowanych kategorii głównych (po 10 klas na kategorię):

- odgłosy zwierząt,
- odgłosy środowiska naturalnego, wody,
- odgłosy ludzkie (z wyłączeniem mowy),
- dźwięki wewnętrzne (domowe),
- dźwięki zewnętrzne (hałas miejski).

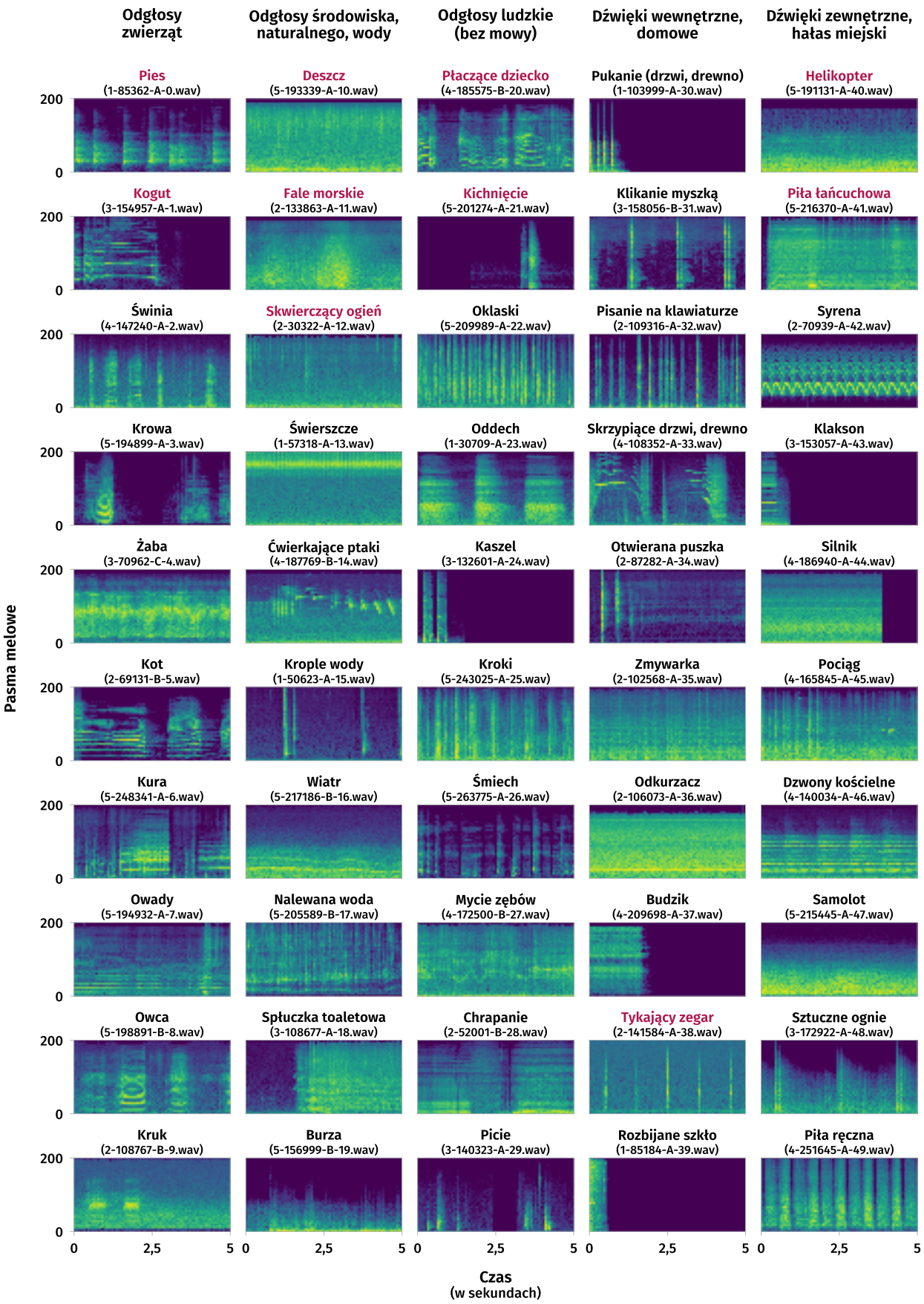
Rysunek 4.1 zawiera wizualizację przykładowych fragmentów dla poszczególnych klas należących do zbioru *ESC-50*.

Celem procesu oznaczania i selekcji materiału źródłowego było utrzymanie zdarzeń dźwiękowych danej klasy jako dominujących we fragmencie, z możliwie ograniczonym hałasem tła. Wykorzystane nagrania rejestrowane były jednak w warunkach dalekich od sterylności studia nagraniowego, stąd możliwe jest występowanie niewielkiego nakładania się zdarzeń w warstwie tła.

ESC-50 zapewnia ekspozycję na zróżnicowany zestaw dźwięków. Niektóre z nich są bardzo powszechnie spotykane (*śmiech*, *miauczenie kota*, *szczekanie psa*), niektóre mocno charakterystyczne (*tłuczenie szkła*, *mycie zębów*), a jeszcze inne cechują się subtelniejszymi różnicami między sobą (np. hałas *helikoptera* i *samolotu*).

Jednym z potencjalnych niedostatków tworzonego zbioru jest stosunkowo ograniczona liczba przykładów dla pojedynczej klasy. Poza kosztem ręcznego oznaczania i selekcji odpowiednich fragmentów, czynnikiem ograniczającym w tym wypadku było również założenie o chęci utrzymania ścisłej równowagi w liczebnej reprezentacji poszczególnych klas (każda z nich zawiera po 40 przykładów). O ile w przypadku stosunkowo powszechnych zdarzeń dźwiękowych odszukanie setek odpowiednich nagrań źródłowych nie stanowi większego problemu, to zadanie to jest dużo trudniejsze dla klas bardziej egzotycznych. Niemniej

[8] Oryginalny format: częstotliwość próbkowania 44,1 kHz, monofoniczny, *Ogg Vorbis* z kompresją 192 kbit/s. W późniejszej wersji zbioru, dla większej kompatybilności odczytywania, pliki udostępnione zostały w formacie *WAVE*.



Rys. 4.1: Wizualizacja wybranych nagrań zbioru *ESC-50* dla poszczególnych klas. Klasy należące do podzbioru *ESC-10* zaznaczono kolorem ciemnoróżowym.

wydaje się, że liczba ta zapewnia już wystarczającą użyteczność zbioru i może być potencjalnie rozszerzona w przypadku wystąpienia odpowiedniego zainteresowania tego typu kolekcją.

W zastosowaniach uczenia bez nadzoru lub z wykorzystaniem słabych etykiet częściową odpowiedzią na braki ilościowe nagrań może być też wykorzystanie zbioru *ESC-US* (*unsupervised*), który zawiera dodatkowe 250 000 fragmentów nagrań w takim samym formacie jak *ESC-50*. Chociaż *ESC-US* formalnie został sporządzony jako zbiór nieetykietowany, to zachowane w nim zostały metadane (opisy zawartości, tagi) wprowadzone dla materiałów źródłowych przez społeczność *Freesound*. Nie były one jednak weryfikowane indywidualnie przez autora, stąd ich jakość opiera się wyłącznie na procesie moderacji treści zapewnianym przez *Freesound*.

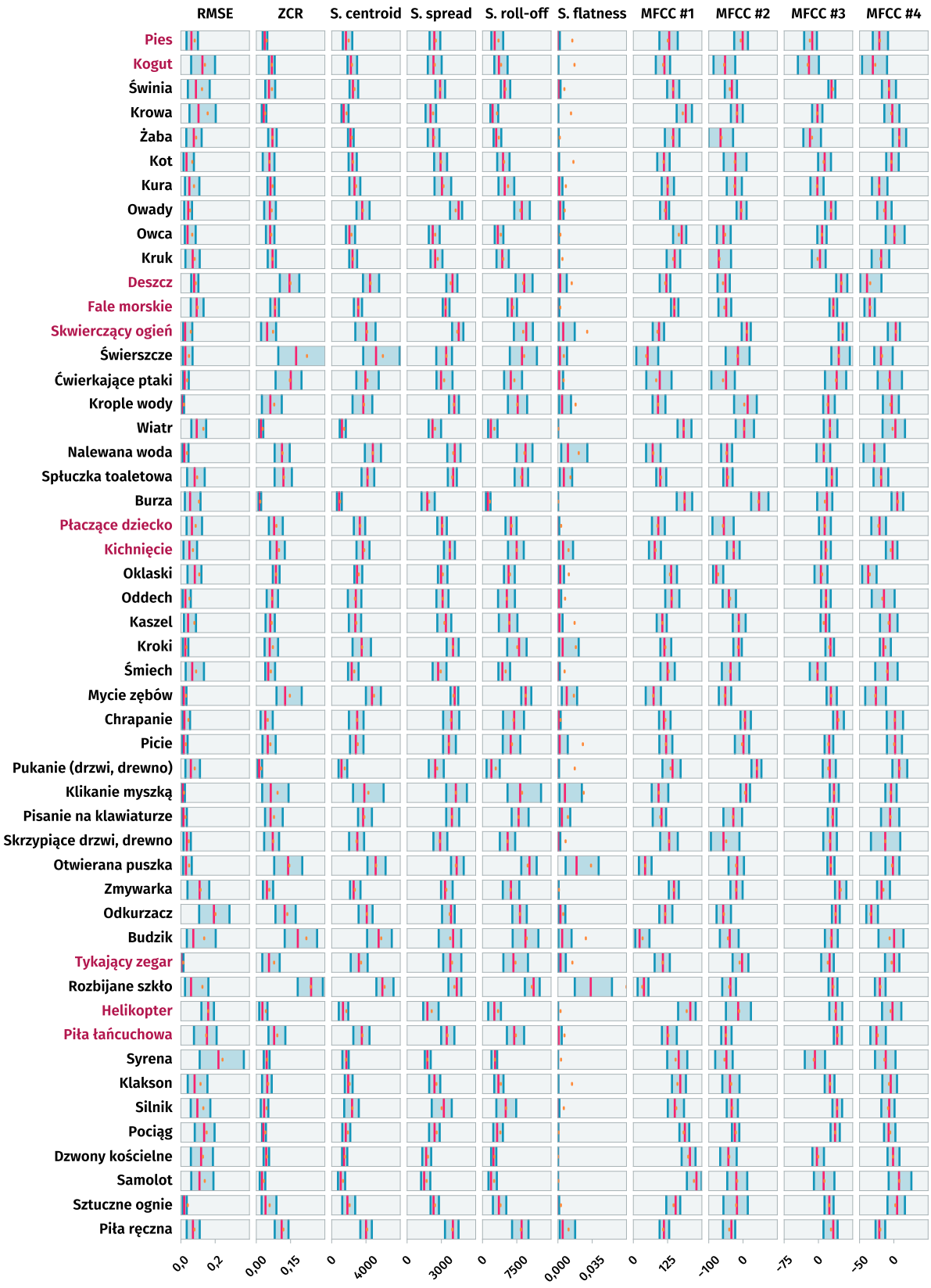
Sam *ESC-50* powstał natomiast jako rozszerzenie *ESC-10*, mniejszego zbioru stworzonego w początkowej fazie jako praca typu *proof-of-concept*, zawierającego 10 klas dźwiękowych przedstawiających trzy główne grupy dźwięków:

- dźwięki perkusyjno-wybuchowe, w tym z bardzo znaczącymi wzorcami czasowymi (*kichnięcie*, *szczekający pies*, *tykający zegar*),
- dźwięki z silną zawartością harmoniczną (*płaczące dziecko*, *piejący kogut*),
- tekstury tła akustycznego z mniej lub bardziej wyraźną strukturą (*deszcz*, *fale morskie*, *skwierczący ogień*, *helikopter*, *piła łańcuchowa*).

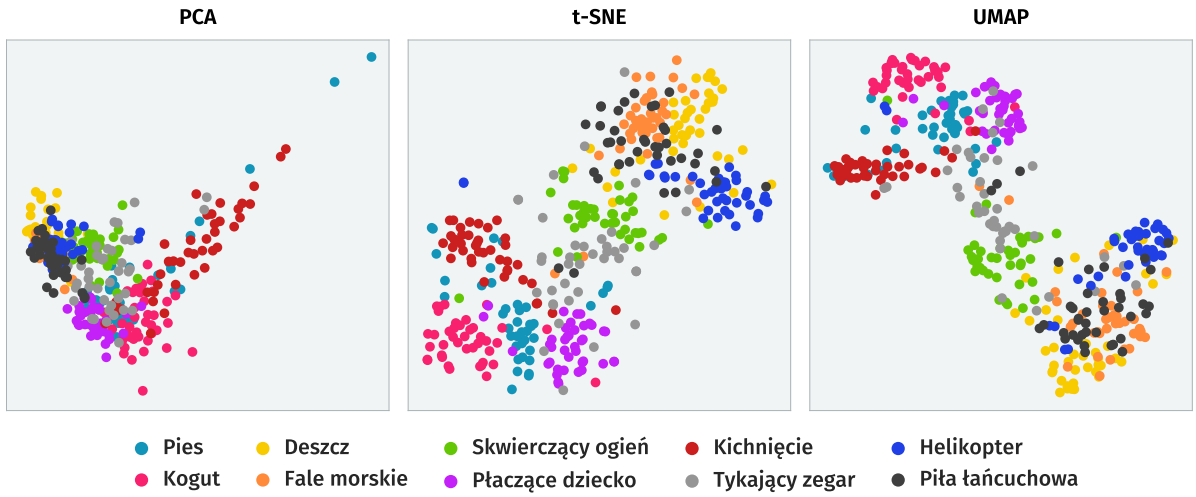
Podzbiór ten, udostępniany bez ograniczeń co do zastosowań komercyjnych na zasadzie licencji *Creative Commons Attribution*, zapewnia dużo prostszy problem klasyfikacyjny. Rozpoznawanie dźwięków z tak ograniczonego zestawu zdarzeń, będące zadaniem trywialnym z perspektywy ludzkiej, stawia więc bardzo wysoko poprzeczkę oczekiwanej dokładności klasyfikacji względem automatycznych systemów rozpoznawania dźwięku. Większa wyrazistość różnic pomiędzy klasami powoduje, że podzbiór ten może faworyzować inne, prostsze metody klasyfikacji niż cały *ESC-50*. Widać to przy analizie separowalności poszczególnych klas przedstawionej na rysunkach 4.2, 4.3 i 4.4.

Pierwszy z nich ilustruje rozkłady empiryczne typowych atrybutów niskopoziomowych opisywanych w podrozdziale 3.2.2 wyznaczonych dla zbioru *ESC-50*[9]. Przy uwzględnieniu wszystkich 50 klas dostrzeżenie wyraźnych wzorców nie jest tak łatwe, można jednak wskazać na kilka prawidłowości. Analiza średniego poziomu energii (*RMSE*) pozwala na oddzielenie dźwięków o charakterze ciągłego hałasu tła (*odkurzacz*, *syrena* jak i większość klas z kategorii dźwięków zewnętrznych) od dźwięków środowiska domowego (*klikanie myszką*, *pisanie na klawiaturze*) czy naturalnego (*ćwierkające ptaki*, *krople wody*). Atrybut *zero-crossing rate*, wykorzystywany między innymi do rozróżniania dźwięków perkusyjnych

[9] Wartości atrybutów wyznaczane były z odrzuceniem ramek, w których dominowała cisza (RMSE $\leq$ 0,005).



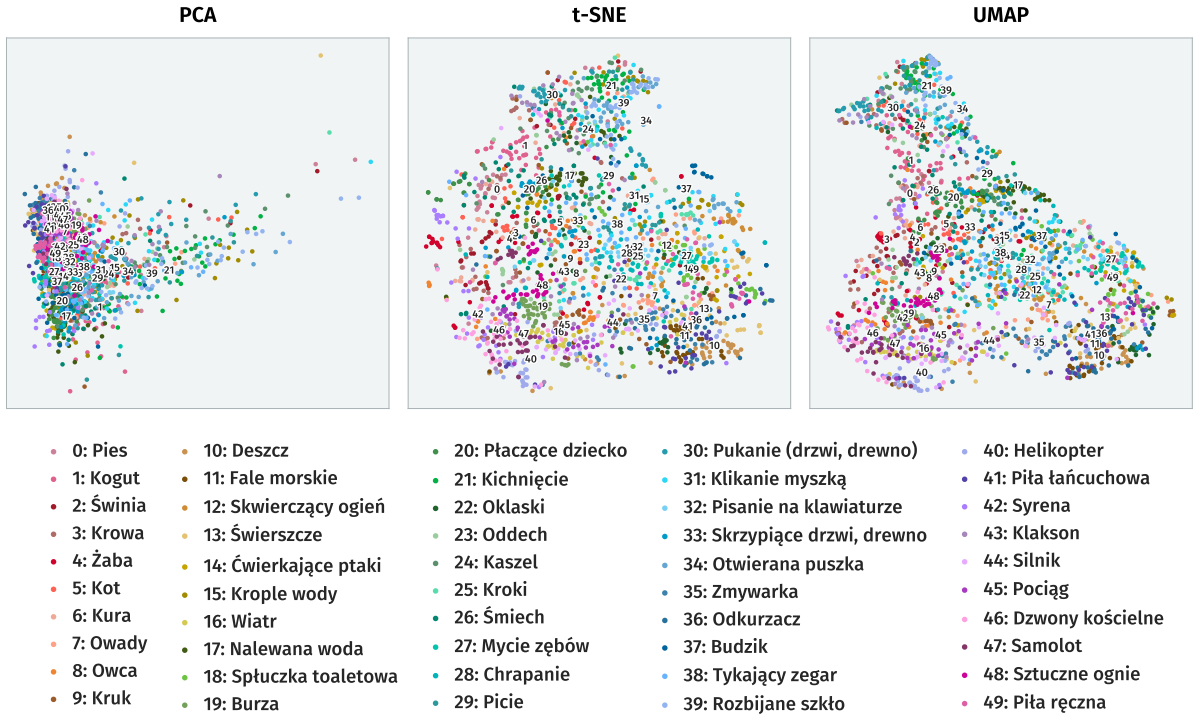
Rys. 4.2: Rozkłady empiryczne wybranych atrybutów niskopoziomowych dla nagrań pochodzących ze zbioru *ESC-50*. Kolorem ciemnoróżowym zaznaczono klasy należące do podzbioru *ESC-10*. Poszczególne pola ilustrują kwartyle i wartość średnią (pomarańczowy punkt) kolejno dla atrybutów: średniej energii, *zero-crossing rate*, centroidu widma, rozpiętości widmowej, punktu spadku widma, płaskości widma oraz czterech pierwszych *MFCC* (z wyłączeniem współczynnika zerowego).



Rys. 4.3: Zobrazowanie klas zbioru *ESC-10* w przestrzeni dwuwymiarowej za pomocą technik *Principal Component Analysis* (PCA), *t-distributed Stochastic Neighbor Embedding* (t-SNE) i *Uniform Manifold Approximation and Projection* (UMAP).

(między werblem a bębnem basowym) (Gouyon et al., 2000), także w przypadku *ESC-50* wyraźnie oddziela dźwięki na podstawie dominujących częstotliwości. Zachowując się podobnie do centroidu widmowego (*spectral centroid*), różnicuje on między odgłosami *burzy*, *wiatru* czy *pukania* a np. *rozbijaniem szkła* czy dźwiękiem *budzika*. W dużej mierze podobnie zachowują się wartości rozpiętości widmowej (*spectral spread*) i punktu spadku widma (*spectral roll-off*). Płaskość widma (*spectral flatness*) dla większości klas jest atrybutem słabo dyskryminującym, funkcjonuje jednak jako dobry detektor *rozbijanego szkła*. Jest to o tyle ważne, że klasę tę można określić jako zdarzenie rzadkie i zazwyczaj niebezpieczne, stąd ma ona szczególne znaczenie praktyczne z punktu widzenia systemów nadzoru i ochrony przeciwwłamaniowej. Interpretacja wartości *MFCC* jest kwestią trudniejszą, tym bardziej, że powinna być dokonywana wielowymiarowo dla wszystkich współczynników łącznie.

Uzyskanie bardziej intuicyjnego rozumienia tych zależności jest natomiast możliwe poprzez wizualizację zbioru w przestrzeni dwuwymiarowej za pomocą popularnych technik mapowania: *PCA* – analizy głównych składowych, *t-SNE* – *t-distributed Stochastic Neighbor Embedding* (Maaten & Hinton, 2008) i *UMAP* – *Uniform Manifold Approximation and Projection* (McInnes & Healy, 2018). Rysunek 4.3 porównuje wyniki takiego przedstawienia wygenerowanego dla *ESC-10* na podstawie omawianych dotychczas atrybutów rozszerzonych o wartości 12 współczynników mel-cepstralnych. O ile w przypadku *PCA* jest to trochę gorzej widoczne, to już dla *t-SNE* i *UMAP* większość klas udaje się osadzić w stosunkowo dobrze oddzielonych skupiskach. Największą trudność odseparowania sprawiają nagrania *piły łańcuchowej*, *tykającego zegara* i *szczekania psa*.



Rys. 4.4: Zobrazowanie klas zbioru *ESC-50* w przestrzeni dwuwymiarowej za pomocą technik *PCA*, *t-SNE* i *UMAP*. Liczby odpowiadające poszczególnym klasom umieszczone zostały w środkach ciężkości punktów należących do danej klasy.

Sytuacja mocno komplikuje się dla całości zbioru *ESC-50*, przedstawionego na rysunku 4.4. Poza tworzeniem się niewielkich lokalnych skupisk specyficznych dla pojedynczej klasy (widać to wyraźnie na przykładzie klasy 48 – *sztuczne ognie* czy 47 – *samolot*), można również zauważyć pewną tendencję bliższego występowania klas należących do tej samej głównej kategorii (zaznaczonych na rysunku poprzez zastosowanie zbliżonych odcieni do ich reprezentacji). Niewątpliwie jednak wizualizacje te potwierdzają, że problem klasyfikacji nagrań należących do zbioru *ESC-50* jest dużo bardziej złożony od *ESC-10*, co widoczne jest również w wynikach osiąganych przez modele opisywane w kolejnym podrozdziale.

4.4 Wyniki uzyskiwane przez ludzi i podstawowe klasyfikatory

4.4.1 Klasyfikacja przez ludzi

Ludzki układ słuchowy radzi sobie bardzo dobrze z jednoczesnym rozpoznawaniem wielu bodźców słuchowych, nawet w bardzo hałaśliwym otoczeniu. Dlatego też przy tak uproszczonym zadaniu, jakie przedstawia klasyfikacja nagrań ze zbioru *ESC*, można oczekiwać, że nie będzie ono dla niego żadnym wyzwaniem. Faktycznym celem postawionym w ramach

badania było więc stwierdzenie, na ile proste jest to zadanie. Do odpowiedzi na to pytanie wykorzystana została platforma crowdsourcingowa *CrowdFlower*[10], której uczestnicy podjęli się próby klasyfikacji nagrań należących do etykietowanej części zbioru *ESC*.

Przeprowadzony eksperyment polegał na udostępnieniu każdemu uczestnikowi pewnej liczby nagrań do odsłuchania i wybrania przez niego jednej z, odpowiednio, 10 lub 50 klas, która najlepiej opisywałaby zaprezentowane nagranie. Uczestnicy otrzymywali stałą stawkę wynagrodzenia za jednostkę wykonanej pracy (oznaczenie 10 nagrań). Kontrola jakości całego procesu zapewniona została przez dostępne w ramach platformy procedury (preselekcję uczestników i bieżący monitoring za pomocą pytań kontrolnych z oczekiwaną odpowiedzią). Surowe wyniki były następnie zweryfikowane przez autora celem odfiltrowania obserwacji skrajnych – w procesie tym odrzucona została nieznaczna część uzyskanych wyników. Dla każdego zbioru danych zebrano około 4000 prób klasyfikacji (w przybliżeniu dwie niezależne oceny dla każdego nagrania zbioru *ESC-50* i dziesięć w przypadku *ESC-10*). Chociaż ze względu na strukturę eksperymentu trudno o formalną statystyczną interpretację wyników, to badanie to zapewnia szacunkową ocenę ludzkich zdolności w zakresie rozpoznawania dźwięków środowiskowych zawartych w zbiorach *ESC*.

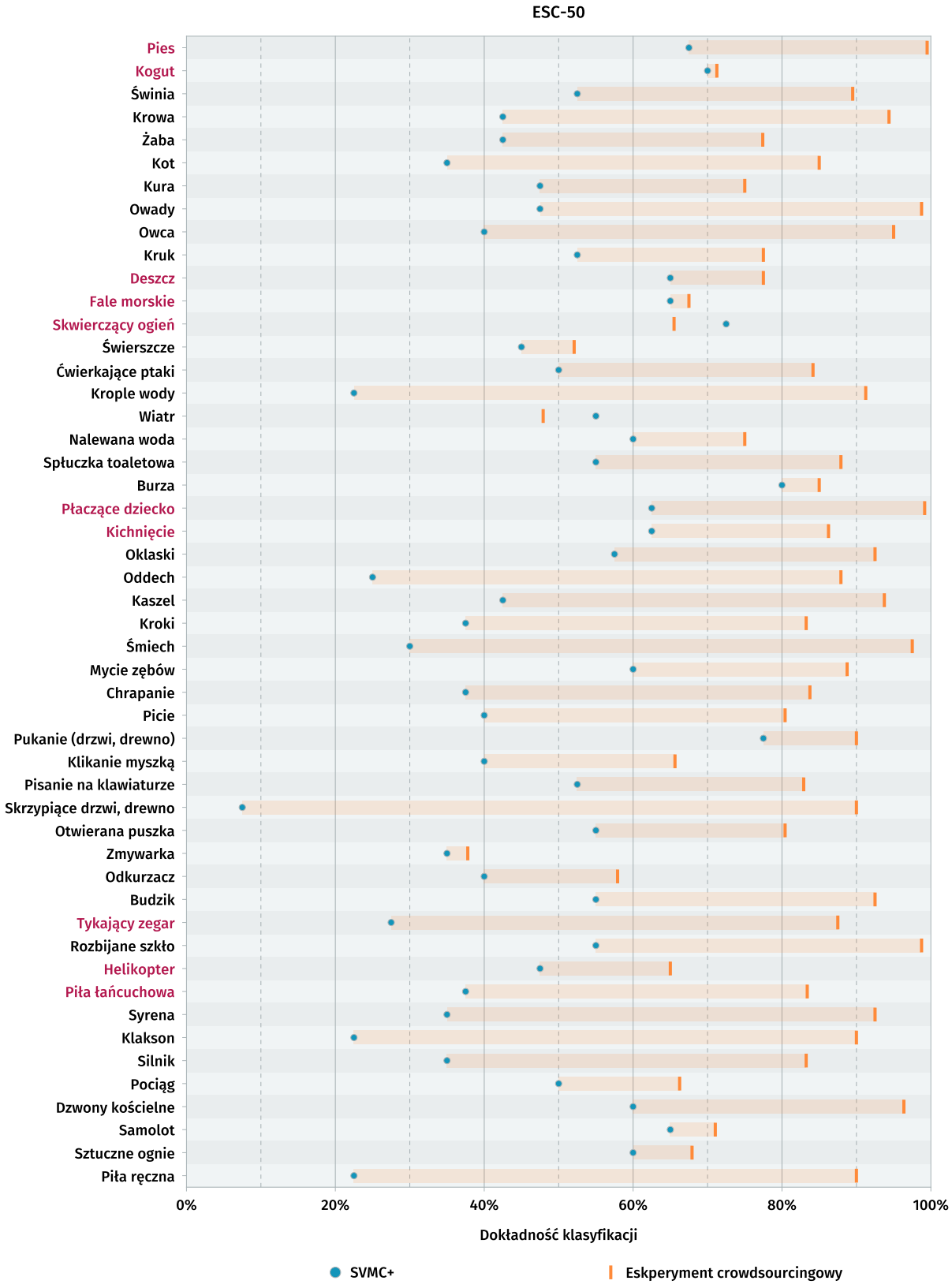
Średnia dokładność klasyfikacji w przeprowadzonym eksperymencie wyniosła 95,7% dla podzbioru *ESC-10* i 81,3% dla całego *ESC-50*. Analiza przedstawionych na rysunku 4.5 szczegółowych wyników dla pojedynczych klas wskazuje na znaczące wahania w zależności od typu zdarzenia dźwiękowego – od 37,8% dla odgłosów *zmywarki* do blisko 100% w przypadku *płaczącego dziecka* i *szczekającego psa*. Patrząc bardziej zbiorczo, można wskazać na trzy grupy dźwięków różniące się pod kątem trudności rozpoznawania przez uczestników eksperymentu:

- klasy łatwe (większość odgłosów ludzkich, niektóre zwierzęta, charakterystyczne dźwięki jak *rozbijane szkło*),
- klasy trudne (głównie tekstury tła akustycznego, hałasy mechaniczne),
- klasy średnie (pozostałe).

Podział ten wydaje się dosyć spójny z funkcjonowaniem układu słuchowego – klasy sprawiające trudność w rozróżnieniu zawierają głównie elementy monotonnego lub wręcz irytującego hałasu, które są w życiu codziennym bezwiednie odfiltrowywane ze świadomej percepcji, natomiast dużo łatwiejsze jest wychwycenie odgłosów interakcji z innymi ludźmi lub sygnałów potencjalnego niebezpieczeństwa.

Wraz ze wzrostem liczby rozważanych klas, jednym z problemów w tego typu eksperymentach staje się też pojawiająca się wśród uczestników trudność w wizualnej orientacji w gąszczu możliwych do wyboru opcji. W przypadku 50 klas zdarzeń dźwiękowych zbioru

[10] *CrowdFlower* funkcjonuje obecnie pod nazwą *Figure Eight*: https://www.figure-eight.com.



Rys. 4.5: Porównanie dokładności klasyfikacji poszczególnych klas zbioru *ESC-50* osiąganych przez model *SVM* w wariancie „*C*+” i uczestników eksperymentu crowdsourcingowego. Obszar zaciemniowany przedstawia różnicę dokładności między tymi podejściami. Klasy należące do podzbioru *ESC-10* zaznaczone zostały kolorem ciemnoróżowym.

ESC-50 kwestię wyboru odpowiedniej kategorii udało się przedstawić w formie jednej spójnej tabeli (podzielonej dla ułatwienia na pięć głównych kategorii), bez sięgania po systematykę zagnieżdżoną. Jest to jednak liczba bliska granicznej przy zastosowaniu tego typu interfejsu.

Osoby biorące udział w badaniu nie były też w żaden szczególny sposób przeszkalane (poza pojedynczymi pytaniami przykładowymi na starcie eksperymentu). Dlatego można oczekiwać, że wyniki uważnych słuchaczy wcześniej dobrze zaznajomionych z całą listą dostępnych klas byłyby jeszcze wyższe. W przypadku *ESC-10* można więc uznać, że klasyfikacja dokonywana przez ludzi jest właściwie bezbłędna, natomiast dla *ESC-50* niewielki margines musi być pozostawiony na bardziej niejednoznaczne odgłosy mechaniczne i naturalnego tła akustycznego. Niemniej przekroczenie przez systemy automatyczne dokładności klasyfikacji na poziomie 90% byłoby z pewnością równoznaczne z uzyskaniem podobnych możliwości do ludzkich, przynajmniej w aspekcie zbioru *ESC-50*.

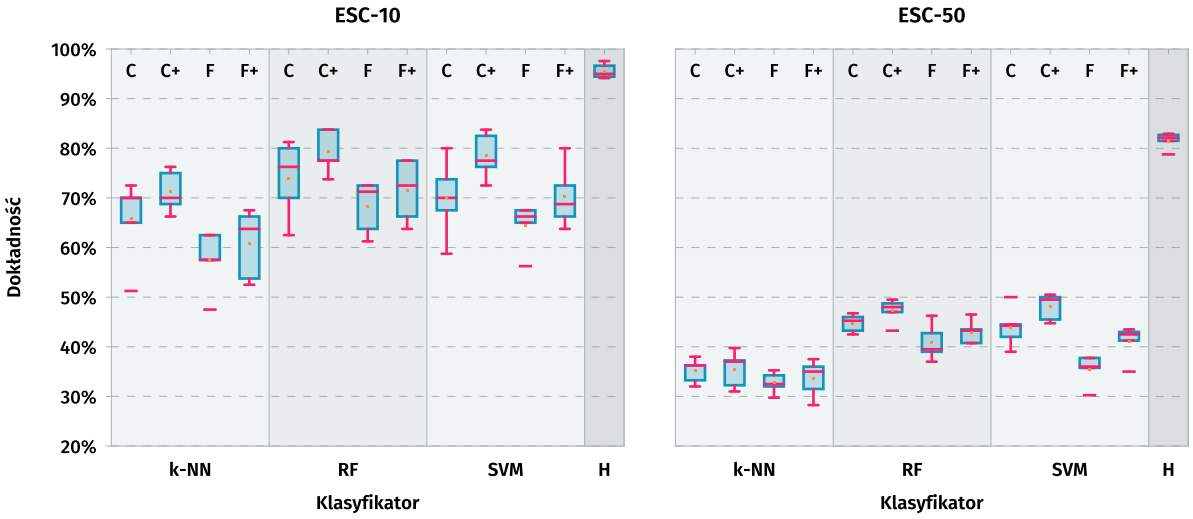
4.4.2 Klasyfikacja za pomocą podstawowych klasyfikatorów

Drugim krokiem po ustaleniu szacunkowej dokładności klasyfikacji przez ludzi jest zweryfikowanie poziomu uzyskiwanego przez podstawowe podejścia wykorzystujące techniki uczenia maszynowego do klasyfikacji dźwięków środowiskowych. Celem analizy podejmowanej w tym miejscu nie jest skonstruowanie możliwie najskuteczniejszego systemu klasyfikacji, ale ocena zachowania zbioru przy zastosowaniu typowych metod.

Eksperyment przeprowadzany w tym zakresie opierał się na wyznaczeniu dla każdego nagrania wartości 12 współczynników mel-cepstralnych (MFCC), a także 6 deskryptorów niskopoziomowych (średniej energii – RMSE, *zero-crossing rate*, centroidu widmowego, rozpiętości widmowej, punktu spadku widma i jego płaskości) na bazie okien analizy o długości 2205 próbek (50 ms) z przeskokiem (*hop length*) 441 próbek (10 ms). Wartości w poszczególnych ramkach były następnie zbiorczo opisywane za pomocą średniej, odchylenia standardowego, skośności i kurtozy. Agregacja następowała na poziomie pojedynczych nagrań (5 sekund) lub ich krótszych fragmentów (co pół sekundy). Na wyznaczonych tym sposobem wektorach cech[11] przeprowadzone zostało uczenie klasyfikatorów k-najbliższych sąsiadów (*k-nearest neighbors*, *k-NN*), lasu losowego (*random forest ensemble*, *RF*) i maszyny wektorów nośnych (*support vector machine*, *SVM*) z walidacją krzyżową z podziałem na 5 podzbiorów[12]. Zestawienie wyników uzyskanych w eksperymencie przez poszczególne metody prezentuje rysunek 4.6.

[11] W wariancie pełnym był to pojedynczy wektor $4 \times 18 = 72$ atrybutów, w krótszym 10 wektorów trenujących z każdego nagrania.

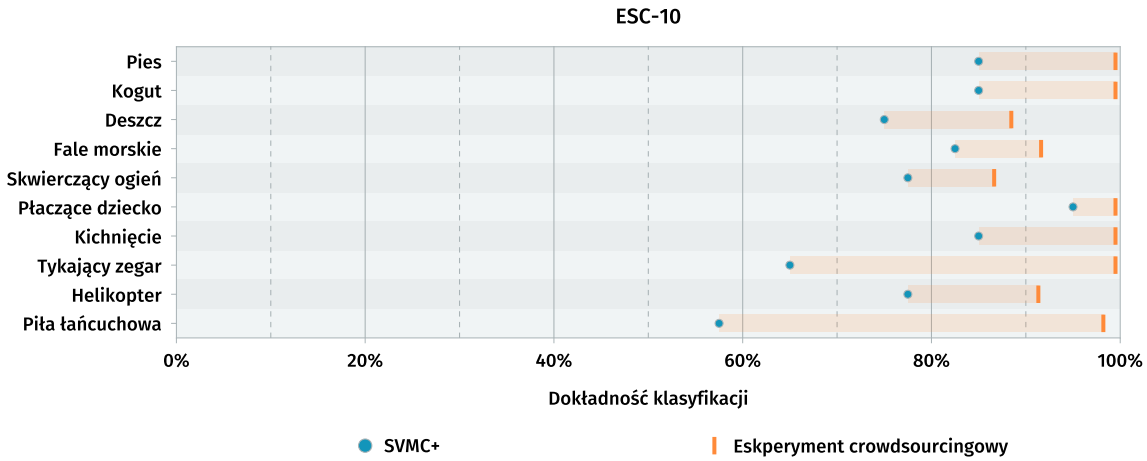
[12] Modele uczone były na bazie implementacji *scikit-learn* z następującymi zmianami względem domyślnych wartości hiperparametrów: 8 sąsiadów dla *k-NN*, 500 drzew dla *RF*, liniowe jądro i wartość kary $C = 0{,}1$ dla *SVM* (*SVC*).



Rys. 4.6: Dokładność klasyfikacji zbiorów *ESC-10* i *ESC-50* uzyskiwana przez poszczególne klasyfikatory: metodę k-najbliższych sąsiadów (*k-NN*), las losowy (*RF*) i maszynę wektorów nośnych (*SVM*). Dla porównania zaznaczono też wyniki eksperymentu crowdsourcingowego (*H*). Dla każdego typu klasyfikatora przedstawione są cztery warianty uczenia: z agregacją na poziomie całych nagrań (*C*) lub krótszych fragmentów (*F*) i odpowiednio na bazie wszystkich dostępnych atrybutów („+”) lub tylko na podstawie MFCC (bez „+”).

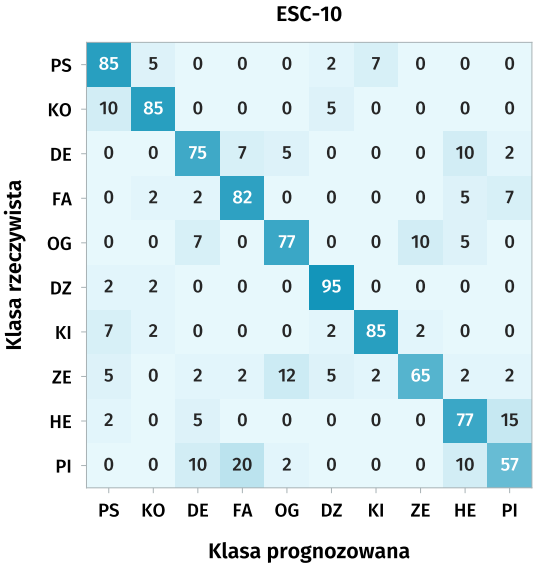
Niezależnie od obranego klasyfikatora najlepszą dokładnością klasyfikacji cechują się warianty „*C+*” wykorzystujące cały dostępny zbiór atrybutów (MFCC i deskryptory niskopoziomowe) przy agregacji cech na poziomie pojedynczych nagrań. Dla zbioru *ESC-10* ich średnia dokładność wynosi od 71,2% dla *k-NN*, przez 78,5% dla *SVM*, do 79,2% dla *RF*. Mediana dokładności jest równa dla lasu losowego i maszyny wektorów nośnych (77,5%). Mała liczba przykładów *ESC-10* uwidacznia się jednak w sporym zróżnicowaniu wyników między poszczególnymi podzbiorami walidacyjnymi. Pod tym względem pełny zbiór *ESC-50* prezentuje się nieco lepiej. Osiągany w jego przypadku średni poziom dokładności spada natomiast do 35,4% (*k-NN*), 47,3% (*RF*) i 48,0% (*SVM*).

Dokładniejsze spojrzenie na przedstawioną na rysunku 4.7 skuteczność klasyfikacji poszczególnych klas zbioru *ESC-10* uwidacznia, że dla większości zdarzeń dźwiękowych występujących w ramach tego zbioru metoda „*SVM C+*” wprowadza mniej więcej stałe pogorszenie wyników w porównaniu z rezultatami osiąganymi przez uczestników eksperymentu opisanego w podrozdziale 4.4.1. Wyraźny wyjątek stanowią klasy *tykający zegar* i *piła łańcuchowa*, które stwarzają temu modelowi nieproporcjonalnie większe problemy (dla ludzi były to klasy bardzo łatwe). Fakt ten potwierdza wcześniejsze spostrzeżenie z rysunku 4.3, że klasy te najtrudniej odseparować od reszty przykładów. *Piła łańcuchowa* była najczęściej mylona z dźwiękiem *fal morskich*, *deszczu* lub *helikoptera* (rysunek 4.8). Z kolei *tykający zegar* był rozpoznawany błędnie najczęściej jako *skwierczący ogień*.

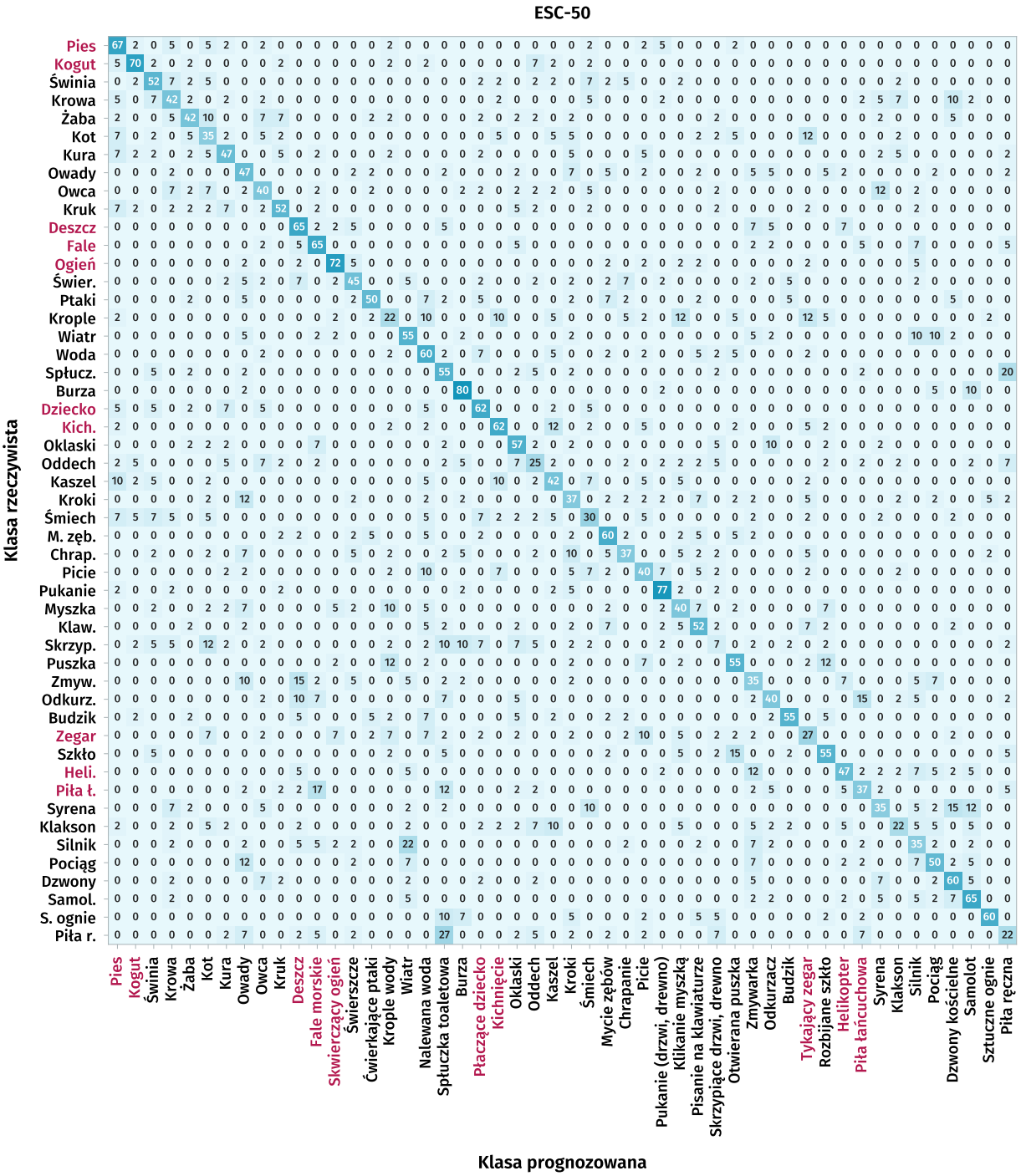


Rys. 4.7: Porównanie dokładności klasyfikacji poszczególnych klas zbioru *ESC-10* osiąganych przez model *SVM* w wariancie „*C+*” i uczestników eksperymentu crowdsourcingowego. Obszar zacieniowany przedstawia różnicę dokładności między tymi podejściami.

Analogiczna analiza dla zbioru *ESC-50* (rysunki 4.5 i 4.9) wskazuje na zestaw klas, dla których model „*SVM C+*” radzi sobie tylko niewiele gorzej niż ludzie (*kogut*, *fale morskie*, *świerszcze*, *burza*, *zmywarka*, *samolot*, *sztuczne ognie*), a nawet uzyskuje od nich nieco lepsze wyniki (*skwierczący ogień* i *wiatr*). Nie wynika to niestety z jego szczególnie wysokiej skuteczności dla tej grupy dźwięków, ale raczej względnie słabych rezultatów osiąganych przez uczestników. Do klas sprawiających ewidentne trudności należą natomiast odgłosy *skrzypiących drzwi*, *kropli wody*, *klaksonu*, *piły ręcznej*, *tykającego zegara* czy ludzkiego *oddechu* i *śmiechu*. Szukając elementów wspólnych dla tej grupy można zauważyć, że są to zdarzenia dźwiękowe o krótkim czasie trwania, dla których istotną informację znaczeniową niesie też ich cykliczna powtarzalność (jest to zwłaszcza widoczne w przypadku *tykającego zegara*, spadającej *kropli wody* czy piłowania za pomocą *piły ręcznej*). Zaprezentowane podejście do klasyfikacji i sposób agregacji ukierunkowane są bardziej na klasy dźwięków, dla których tworzone atrybuty są stacjonarne w ramach całości nagrania, stąd, dosyć zrozumiale, lepsze wyniki klasyfikacji osiągane są dla grupy dźwięków o charakterze tekstury tła akustycznego. Widoczne jest więc spore pole do poprawy osiąganych rezultatów przez bardziej złożone modele.



Rys. 4.8: Macierz pomyłek klasyfikacji zbioru *ESC-10* za pomocą klasyfikatora SVM w wariancie „*C+*”. Rozwinięcie skrótów nazw klas w kolejności jak na rysunku 4.7. Komórki przedstawiają zaokrąglone wartości procentowe.



Rys. 4.9: Macierz pomyłek klasyfikacji zbioru *ESC-50* za pomocą klasyfikatora SVM w wariancie „*C+*”. Klasy należące do podzbioru *ESC-10* zaznaczone zostały kolorem ciemnoróżowym. Komórki przedstawiają zaokrąglone wartości procentowe.

4.5 Publikacje wykorzystujące zbiór ESC

Poza wynikami prezentowanymi w niniejszej rozprawie, na przestrzeni lat 2016–2017 do zbioru *ESC* odnosiło się w swoich eksperymentach wiele publikacji – zarówno w sposób bezpośredni, porównując proponowane usprawnienia metod klasyfikacji pod kątem uzyskiwanej na tym zbiorze dokładności, jak i pośrednio wykorzystując udostępnione nagrania w innych celach, przykładowo do zadań typu *transfer learning*. Tabela 4.1 zawiera zbiorcze zestawienie[13] opublikowanych modeli wykorzystujących zbiór *ESC*.

Tab. 4.1: Zestawienie modeli poddawanych walidacji na zbiorze *ESC*.

Skróty wykorzystywane w tabeli:

CNN – splotowa sieć neuronowa, *CRNN* – splotowo-rekurencyjna sieć neuronowa, *GMM* – model mieszanin gaussowskich, *GTCC* – współczynniki gammatone-cepstralne, *GTSC* – współczynniki gammatone-spektralne, *k-NN* – metoda k-najbliższych sąsiadów, *MFCC* – współczynniki mel-cepstralne, *MLP* – perceptron wielowarstwowy, *RBM* – ograniczona maszyna Boltzmanna, *RNN* – rekurencyjna sieć neuronowa, *SVM* – maszyna wektorów nośnych, *TEO* – operator energii Teagera, *ZCR* – *zero-crossing rate*.

Publikacja	Dokładność klasyfikacji (ESC-50)	Uwagi
(Sailor et al., 2017)	86,5%	CNN z zespołem filtrów uczonych za pomocą splotowej RBM w połączeniu z atrybutami GTSC i energią w pasmach melowych
(Tokozume et al., 2017)	84,9%	architektura *EnvNet-v2* (Tokozume & Harada, 2017) z syntetycznym powiększaniem zbioru danych i uczeniem typu „*between-class*”
(Tak et al., 2017)	84,2%	CNN operująca na zespole filtrów melowych z informacją fazową i energii w pasmach melowych
(Kumar et al., 2017)	83,5%	CNN uczona wstępnie na zbiorze *AudioSet*
(Sailor et al., 2017)	83,0%	CNN z zespołem filtrów uczonych za pomocą splotowej RBM w połączeniu z atrybutami GTSC
(Agrawal et al., 2017)	82,0%	połączenie atrybutów GTSC, TEO-GTSC z siecią splotową
(Tokozume et al., 2017)	81,8%	architektura *EnvNet-v2* (Tokozume & Harada, 2017) z uczeniem typu „*between-class*”
(Piczak, 2015b)	**81,3%**	wynik uzyskany w omawianym w rozprawie eksperymencie crowdsourcingowym (podrozdział 4.4.1)
(Arandjelović & Zisserman, 2017b)	79,8%	architektura *L3* (Arandjelović & Zisserman, 2017a) ze skokiem 2, większymi partiami uczącymi i korektą tempa uczenia
(Arandjelović & Zisserman, 2017a)	79,3%	archtektura *L3* (8-warstwowa CNN) uczona wstępnie na zadaniu dopasowania ścieżki dźwiękowej z obrazem
(Agrawal et al., 2017)	79,1%	atrybuty GTSC w połączeniu z siecią splotową
(Tokozume et al., 2017)	78,8%	architektura *EnvNet-v2* (Tokozume & Harada, 2017) z syntetycznym powiększaniem zbioru danych

Ciąg dalszy na kolejnej stronie

[13] Aktualna wersja zestawienia dostępna jest w repozytorium: https://github.com/karoldvl/ESC-50.

Tab. 4.1: Zestawienie modeli poddawanych walidacji na zbiorze *ESC* (*kontynuacja*).

Publikacja	Dokładność klasyfikacji	Uwagi
(Sailor et al., 2017)	78,5%	CNN z zespołem filtrów uczonych za pomocą splotowej RBM
(Tokozume et al., 2017)	76,9%	bazowy model CNN (Piczak, 2015a) rozszerzony o normalizację partiami i uczenie typu „*between-class*”
(Agrawal et al., 2017)	74,9%	atrybuty TEO-GTSC w połączeniu z siecią splotową
(Tokozume et al., 2017)	74,4%	architektura *EnvNet-v2* (Tokozume & Harada, 2017)
(Aytar et al., 2016)	74,2%	8-warstwowa CNN z transferem wiedzy z nieetykietowanych filmów
(Tokozume et al., 2017)	73,3%	18-warstowa CNN operująca na reprezentacji w dziedzinie czasu (Dai et al., 2017) rozszerzona o uczenie typu „*between-class*”
(Tak et al., 2017)	73,3%	CNN operująca na zespole filtrów melowych z informacją fazową
(Boddapati et al., 2017)	73,2%	architektura *GoogLeNet* operująca na spektrogramach z ramką o długości 40 ms, częstotliwość prókowania zredukowana do 16 kHz
(Tokozume et al., 2017)	72,4%	bazowy model CNN (Piczak, 2015a) rozszerzony o normalizację partiami
(Agrawal et al., 2017)	72,3%	połączenie MFCC, TEO-GTCC z klasyfikatorem GMM
(Tokozume & Harada, 2017)	71,0%	połączenie CNN operującej na spektrogramach i reprezentacji w dziedzinie czasu
(Agrawal et al., 2017)	68,9%	atrybuty TEO-GTCC z klasyfikatorem GMM
—	**68,8%**	bazowy model CNN (Piczak, 2015a), uczony na 4 podzbiorach walidacyjnych (podrozdział 5.2.2)
(Boddapati et al., 2017)	68,7%	architektura *AlexNet* operująca na spektrogramach z ramką o długości 30 ms, częstotliwość prókowania zredukowana do 16 kHz
(Dai et al., 2017)	68,5%	18-warstowa CNN operująca na reprezentacji w dziedzinie czasu
(Boddapati et al., 2017)	67,8%	architektura *GoogLeNet* operująca na spektrogramach z ramką o długości 30 ms, częstotliwość prókowania zredukowana do 32 kHz
—	**67,8%**	zaktualizowany bazowy model CNN (Piczak, 2015a) (podrozdział 5.2.2)
(Jin et al., 2017)	66,3%	8-warstwowa architektura *SoundNet* (Aytar et al., 2016) ze 100-krotną redukcją liczby wykorzystywanych parametrów
(Aytar et al., 2016)	66,1%	5-warstwowa CNN z transferem wiedzy z nieetykietowanych filmów
(Jin et al., 2017)	65,8%	8-warstwowa architektura *SoundNet* (Aytar et al., 2016) ze 180-krotną redukcją liczby wykorzystywanych parametrów
(Aytar et al., 2016)	65,0%	5-warstwowa CNN uczona bezpośrednio na zbiorze ESC-50 (bez transferu wiedzy)
(Piczak, 2015a)	**64,5%**	bazowy model CNN, uczony na 3 podzbiorach walidacyjnych z 5
(Freitag et al., 2017)	64,3%	MLP operujący na atrybutach wytworzonych przez autokoder-RNN
(Boddapati et al., 2017)	63,2%	architektura *AlexNet* operująca na spektrogramach z ramką o długości 30 ms, częstotliwość prókowania zredukowana do 32 kHz
(Boddapati et al., 2017)	60,3%	architektura typu CRNN
(Huzaifah, 2017)	56,4%	3-warstwowa CNN z filtrami wertykalnymi
(Huzaifah, 2017)	54,0%	3-warstwowa CNN z filtrami kwadratowymi
(Aytar et al., 2016)	51,1%	8-warstwowa CNN uczona bezpośrednio na zbiorze *ESC-50* (bez transferu wiedzy)

Ciąg dalszy na kolejnej stronie

Tab. 4.1: Zestawienie modeli poddawanych walidacji na zbiorze *ESC* (*kontynuacja*).

Publikacja	Dokładność klasyfikacji	Uwagi
(Huzaifah, 2017)	50,9%	5-warstwowa CNN z filtrami kwadratowymi
—	48,0%	podejście bazowe (18 atrybutów + SVM, podrozdział 4.4.2)
—	47,3%	podejście bazowe (18 atrybutów + las losowy, podrozdział 4.4.2)
(Huzaifah, 2017)	46,3%	5-warstwowa CNN z filtrami wertykalnymi
(Piczak, 2015b)	44,3%	podejście bazowe (MFCC, ZCR + las losowy)
(Aytar et al., 2016)	39,9%	autokoder splotowy uczony na nieetykietowanych filmach
(Piczak, 2015b)	39,6%	podejście bazowe (MFCC, ZCR + SVM)
—	35,4%	podejście bazowe (18 atrybutów + k-NN, podrozdział 4.4.2)
(Piczak, 2015b)	32,2%	podejście bazowe (MFCC, ZCR + k-NN)
(Baelde et al., 2017)	nd.	słownik modeli dźwiękowych używany do klasyfikacji (nieporównywalna procedura walidacji)
(Elizalde et al., 2018)	nd.	przeszukiwanie materiałów internetowych na podstawie klasyfikatorów wytrenowanych na mniejszych zbiorach (w tym *ESC-50*)
(Tax et al., 2017)	nd.	aproksymacja przekształcenia reprezentacji w dziedzinie czasu do mel-spektrogramu za pomocą CNN
(Mun et al., 2017)	nd.	*transfer learning* z wykorzystaniem różnych zbiorów danych (w tym *ESC-50*)
(Kumar & Raj, 2016)	nd.	MFCC w połączeniu z GMM i SVM
(Pillos et al., 2016)	nd.	rozpoznawanie dźwięku w czasie rzeczywistym na platformie Android, walidacja na zbiorze *ESC-10*
(Hertel et al., 2016)	nd.	ocena siły dyskryminacyjnej różnych reprezentacji sygnału na zbiorach *Freiburg-106* i *ESC-10*
(Kumar & Raj, 2017a)	nd.	połączenie słabo etykietowanych danych (*YouTube*) z silnymi etykietami (*ESC-10*) w uczeniu systemów rozpoznawania zdarzeń dźwiękowych

Komentując pokrótce wymienione w tabeli modele, trzeba przede wszystkim podkreślić pojawiający się w najświeższych publikacjach istotny wzrost dokładności klasyfikacji uzyskiwanej na zbiorze *ESC-50* w porównaniu z modelem bazowym (Piczak, 2015a). Wprowadzane usprawnienia można podzielić na trzy główne nurty.

Pierwszym z nich są eksperymenty rozszerzające reprezentację wejściową do sieci splotowej o nowe typy atrybutów. Bardzo dobrze wydają się sprawować współczynniki bazujące na filtrach *gammatone* (Sailor et al., 2017; Agrawal et al., 2017) i informacji fazowej (*PEFBE – phase encoded mel filterbank energies*) (Tak et al., 2017). Można domniemywać, że częściowy powrót do bardziej wyspecjalizowanych form przetwarzania sygnału wejściowego związany jest z ograniczonym rozmiarem zbioru *ESC-50*, który nie pozwala na wyuczenie odpowiednio ogólnych reprezentacji bezpośrednio z danych. Narzucona tym

sposobem regularyzacja wynikająca z wiedzy dziedzinowej okazuje się w tym wypadku szczególnie skuteczna.

Drugim podejściem mocno poprawiającym funkcjonowanie modeli splotowych są rozszerzenia samego procesu uczenia. Wprowadzona przez Tokozume et al. **(2017)** technika „*between-class learning*" pozwala nawet w przypadku architektury bazowej **(Piczak, 2015a)** na wzrost dokładności klasyfikacji do 76,9%. W połączeniu z syntetycznym powiększaniem zbioru danych i architekturą *EnvNet-v2* możliwe staje się uzyskanie wyników na poziomie zdolności ludzkich (84,9%).

Podobnie też zainicjalizowane w odpowiedni sposób sieci splotowe pozwalają przy standardowym sposobie uczenia uzyskać wyniki konkurujące z eksperymentem crowdsourcingowym. Mowa tu przede wszystkim o wstępnym trenowaniu klasyfikatorów na zbiorze *AudioSet* **(Kumar et al., 2017)** (83,5%), wykorzystaniu części sieci z zadania dopasowania obrazu do dźwięku (*audio–visual correspondence*) **(Arandjelović & Zisserman, 2017a)** lub uczeniu na materiałach filmowych bez etykiet dźwiękowych **(Aytar et al., 2016)**.

Patrząc na te liczne modele wydaje się, że zbiór *ESC* spełnił stawiane przed nim zadanie, pozwalając na porównanie nowych architektur i metod uczenia sieci splotowych, które w latach 2016–2017 zdominowały obszar rozpoznawania dźwięku. Ostatnie prace na tyle wysoko podniosły poziom osiąganej dokładności klasyfikacji, równając się z ludzkimi możliwościami, że dosyć naturalnie nasuwa się pytanie o punkt nasycenia – czy możliwe będą dalsze postępy w tym zakresie? Prawdopodobnie tak, chociaż widoczna też staje się powoli konieczność szukania nowych trudniejszych wyzwań związana z tak dynamicznym postępem wiedzy w tej dziedzinie.

Rozdział 5

Klasyfikacja dźwięków środowiskowych za pomocą splotowych sieci neuronowych[1]

5.1 Wprowadzenie

Głównym celem rozprawy, postawionym w podrozdziale 1.2, jest odpowiedź na pytanie, czy splotowe sieci neuronowe mogą być z powodzeniem zastosowane w zadaniach klasyfikacji dźwięków niebędących mową, w szczególności mając na uwadze ograniczony charakter dostępnych etykietowanych zbiorów danych. Obecny stan rozwoju dziedziny rozpoznawania dźwięku pozwala już udzielić w tym zakresie odpowiedzi bez wątpienia twierdzącej, a nawet rozszerzyć ją na trudniejsze zagadnienia detekcji zdarzeń dźwiękowych. Eksperymenty omawiane w niniejszym rozdziale przeprowadzane były jednak oryginalnie w zdecydowanie odmiennej rzeczywistości, będąc jedną z pierwszych prób wykorzystania tego typu modeli do rozpoznawania dźwięków środowiskowych. Dlatego też przyjętym w nich założeniem badawczym było ograniczenie się w tym stadium właśnie do problemu względnie prostszego, jakim jest klasyfikacja krótkich nagrań dźwiękowych. Celem tego zadania jest wskazanie pojedynczej etykiety najlepiej opisującej elementy akustyczne dominujące w analizowanym fragmencie.

[1] Zaprezentowane w niniejszym rozdziale wyniki eksperymentów zostały oryginalnie przedstawione w ramach publikacji konferencyjnej „*Environmental Sound Classification with Convolutional Neural Networks*” **(Piczak, 2015a)** oraz towarzyszących jej materiałów uzupełniających (https://github.com/karoldvl/paper-2015-esc-convnet/). Omówienie zawarte w rozprawie rozszerza tę treść o bardziej pogłębioną wizualizację efektów uczenia sieci, a także dołącza do porównania nową wersję zaproponowanego modelu, ograniczającą liczbę wykorzystywanych parametrów poprzez zastosowanie usprawnień wprowadzonych w literaturze przedmiotu od czasu publikacji pierwszej wersji modelu. Dokładne wartości liczbowe przedstawione w niniejszym rozdziale odnoszą się do wyników uzyskanych przez implementację z wykorzystaniem biblioteki *PyTorch*. Wyniki artykułu **(Piczak, 2015a)** opierały się na implementacji z wykorzystaniem *Pylearn2/Theano*.

Zbiory danych

Do procesu uczenia i oceny zaproponowanych modeli splotowych wykorzystane zostały trzy zbiory nagrań: *ESC-50*, *ESC-10* i *UrbanSound8K*. Zbiory *ESC* wyczerpująco omawia rozdział 4, dlatego ich przedstawienie zostanie w tym miejscu pominięte. *UrbanSound8K* jest natomiast kolekcją nagrań sporządzoną według podobnej formuły przez Salamona et al. **(2014)**, zawierającą 8732 krótkie (poniżej 4 sekund) nagrania dźwięków miejskich należących do jednej z dziesięciu klas: *klimatyzacja*, *klakson*, *bawiące się dzieci*, *szczekający pies*, *wiercenie*, *silnik na biegu jałowym*, *strzały z broni palnej*, *młot pneumatyczny*, *syrena*, *muzyka uliczna*. Zbiór ten został przez twórców odgórnie podzielony na dziesięć podzbiorów walidacyjnych, co ułatwia zapewnienie porównywalności wyników uzyskiwanych w eksperymentach różnych autorów. Salamon et al. **(2014)** ustalili również wyjściowy poziom dokładności uzyskiwanej dla *UrbanSound8K* na poziomie 68%, stosując typowe podejście bazujące na współczynnikach mel-cepstralnych klasyfikowanych przez maszynę wektorów nośnych (*SVM*). W późniejszej pracy **(Salamon & Bello, 2015)** wynik ten został poprawiony przez uczenie metodą sferycznych k-średnich (*spherical k-means*) na fragmentach spektrogramów poddanych dekorelacji (*PCA whitening*), które pozwoliło na osiągnięcie dokładności 73,6%.

5.2 Metodyka eksperymentu i architektura modeli

Uczenie splotowych sieci neuronowych wymaga podjęcia wielu decyzji przy określaniu architektury modelu (format danych wejściowych, liczba i rozmiar poszczególnych warstw, wielkość filtrów i grupowania przestrzennego) a także innych hiperparametrów związanych z procesem uczenia (tempo uczenia, wartość *momentum*, wielkość partii uczących, siła regularyzacji i *dropoutu*). W związku z długim czasem potrzebnym na całkowite wyuczenie i możliwość dokładnej oceny zachowania tego typu modeli, wybór najbardziej obiecujących wartości hiperparametrów dokonuje się zazwyczaj z konieczności na bazie heurystyk i wstępnych wyników walidacji, szczególnie przy wkraczaniu na nieznane dotychczas obszary zastosowań. Na takiej właśnie zasadzie, po rozważeniu potencjalnych wariantów przedstawianej architektury, wybrany został model zilustrowany na rysunku 5.2 (określany dalej jako *E2015*). Jego dokładne funkcjonowanie opisują kolejne trzy podrozdziały.

5.2.1 Przetwarzanie danych

Pierwszym krokiem przeprowadzanym na nagraniach źródłowych pochodzących ze zbiorów *ESC* i *UrbanSound8K* jest konwersja do ujednoliconego formatu, obejmująca redukcję częstotliwości próbkowania do 22 050 Hz, normalizację wartości sygnału na poziomie poszczególnych plików, a także ustalenie długości nagrań na dokładnie 5 sekund (*ESC*) lub

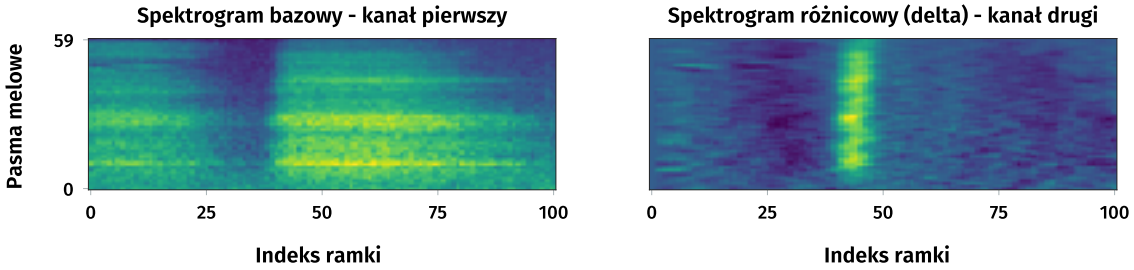
4 sekundy (*UrbanSound8K*) przez odpowiednio przycięcie lub dopełnienie ciszą fragmentów o czasie trwania odbiegającym od założonego. Następnie, za pomocą biblioteki *librosa*[2], dla każdego z nagrań generowane są spektrogramy w melowej skali częstotliwości z wartościami wyrażonymi w skali decybelowej. Do wyznaczania spektrogramów przyjęte zostały następujące wartości parametrów: długość okna FFT – 1024 próbki, długość skoku – 512 próbek, liczba pasm melowych – 60.

Tak przedstawione nagrania źródłowe mogłyby zostać wykorzystane w całości jako przykłady uczące, ale rozwiązanie to nie jest preferowane z dwóch powodów. Po pierwsze, analiza przykładów o pełnej możliwej długości niepotrzebnie zwiększa rozmiar danych wejściowych i w konsekwencji czas ich przetwarzania przez sieć neuronową. Drugim powodem przemawiającym za podzieleniem spektrogramów na mniejsze segmenty jest możliwość uzyskania liczniejszego zbioru uczącego, w którym długość pojedynczych przykładów ogranicza się do czasu wystarczającego na rozpoznanie większości klas zdarzeń dźwiękowych. Postępowanie takie zakłada jednak, że utworzone w ten sposób krótsze fragmenty będą dalej dobrze opisywane przez tę samą etykietę, która nadana została całości danego nagrania. Chociaż nie jest możliwe całkowite zagwarantowanie poprawności takiego założenia, to jest ono spełnione na tyle często, że w praktyce metoda ta uzyskuje lepsze wyniki niż wykorzystanie nagrań o pełnej długości. W związku z tym w eksperymentach przeanalizowane zostały dwa warianty sieci – w krótszym wejście modelu stanowiły segmenty o długości 41 ramek (około 950 ms), w dłuższym odpowiednio 101 ramek (około 2,3 s).

W trakcie procesu uczenia sieci poszczególne segmenty generowane są poprzez losowy wybór ciągłego fragmentu nagrania, innego w każdej epoce, co pozwala w dużym stopniu uodpornić model na przesunięcia zdarzeń dźwiękowych w czasie i jest prostą formą syntetycznego zwiększania zbioru uczącego (*data augmentation*). Klasyfikacja konkretnych przykładów ze zbioru walidacyjnego i testowego uzyskiwana jest z kolei poprzez agregację predykcji wyznaczonych po podzieleniu nagrania na sekwencję segmentów z zastosowaniem przeskoku o połowę ich długości (50% nakładania się kolejnych segmentów). W ramach eksperymentu rozważane były dwa typy agregacji – polegający na uśrednianiu wartości prawdopodobieństwa poszczególnych klas wyznaczanych dla indywidualnych segmentów (*probability voting*) i bazujący na zliczaniu najczęściej występującej predykcji, w którym każdy segment ma jeden głos wskazujący na klasę dominującą w danym nagraniu (*majority voting*).

Końcowy krok przeprowadzanego przetwarzania danych wejściowych polega na ich standaryzacji na podstawie statystyk wyznaczonych dla zbioru walidacyjnego i utworzeniu spektrogramów różnicowych (*delta spectrograms*) stanowiących drugi kanał wejścia sieci neuronowej – na podobnej zasadzie jak w splotowych modelach rozpoznawania mowy

[2] https://librosa.github.io/librosa/.



Rys. 5.1: Zobrazowanie kanałów danych wejściowych dla przykładowego segmentu nagrania *oddechu* w wariancie dłuższym (101 ramek).

zaproponowanych przez Abdel-Hamida et al. (2014). W tym celu wykorzystywana jest implementacja z biblioteki *librosa* w wersji *0.3.1*[3]. Efekt jej zastosowania można przedstawić jako splot bazowego spektrogramu (którego wiersze odpowiadają poszczególnym pasmom, a kolumny kolejnym ramkom) z jednowierszowym jądrem przekształcenia K postaci:

$$K = \begin{bmatrix} -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \end{bmatrix} \tag{5.1}$$

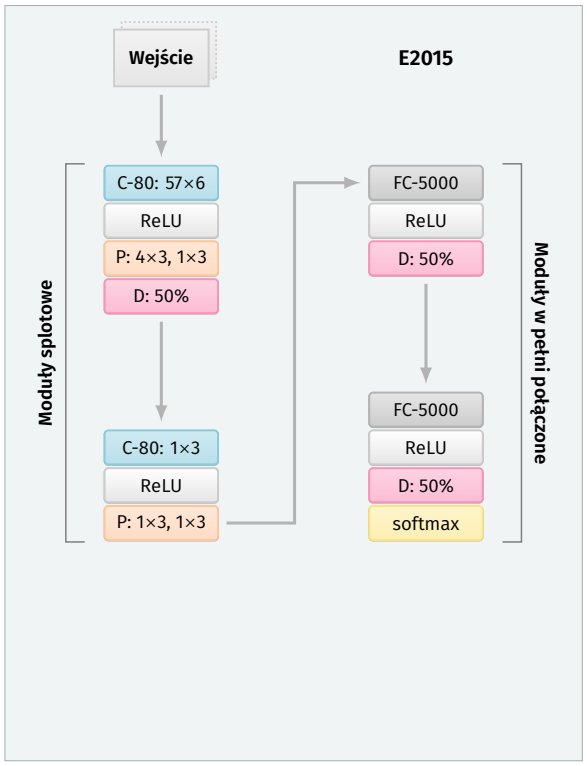
Wynik tej operacji dla przykładowego nagrania ilustruje rysunek 5.1.

5.2.2 Architektura sieci

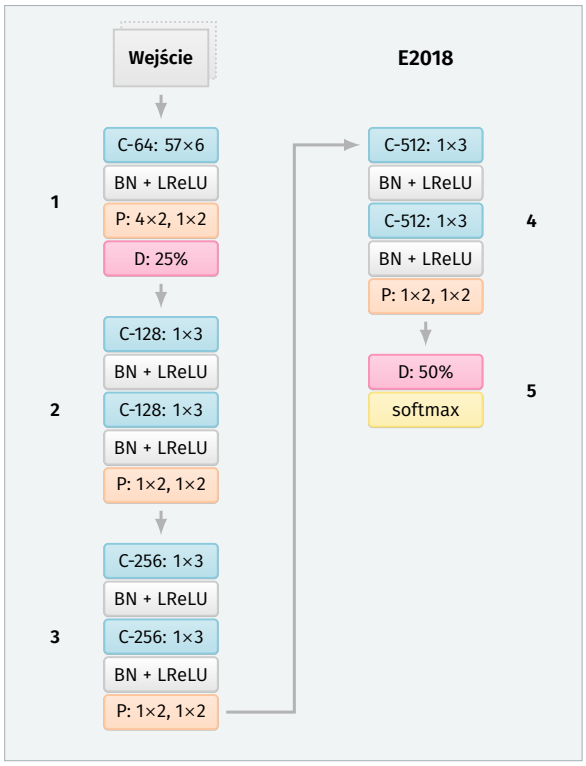
Model zaproponowany oryginalnie w artykule (Piczak, 2015a) składa się ze zilustrowanych na rysunku 5.2 dwóch warstw splotowych z grupowaniem typu *max-pooling*, dwóch warstw w pełni połączonych i warstwy wyjścia typu *softmax*. Jego wejście stanowi para spektrogramów (bazowy i różnicowy) o rozmiarach 60 pasm × L ramek, gdzie $L = 101$ lub $L = 41$ w zależności od przyjętego wariantu. Parę tę przetwarza pierwsza warstwa splotowa zawierającą 80 filtrów prostokątnych o rozmiarze 57 × 6, skoku 1 × 1 i funkcji aktywacji typu *ReLU*. Ograniczenie wysokości zastosowanych filtrów do wartości mniejszej od całkowitej liczby pasm pozwala na uzyskanie odporności na niewielkie przesunięcia w dziedzinie częstotliwości poprzez dalsze zgrupowanie odpowiedzi za pomocą warstwy typu *max-pooling* z rozmiarem okna 4 × 3 i skokiem 1 × 3. Operacja ta dokonuje także istotnej redukcji wymiarowości otrzymywanych na tym etapie map aktywacji.

Druga warstwa splotowa składa się również z 80 filtrów, tym razem o rozmiarach 1 × 3 i skoku 1 × 1, dla których wykorzystywana jest funkcja aktywacji typu *ReLU*. Grupowanie *max-pooling* po tej warstwie przeprowadzane jest z oknem 1 × 3 i skokiem 1 × 3. Uzyskana tym sposobem reprezentacja danych (dla segmentów o długości 101 ramek jest to 800 wartości skalarnych – 80 filtrów po 10 aktywacji rozłożonych w czasie) podlega dalszemu przetwarzaniu przez dwie warstwy w pełni połączone, z których każda składa się

[3] Implementacja ta różni się od dostępnej w nowszych wersjach biblioteki.



Rys. 5.2: Schematyczne przedstawienie modelu *E2015* (Piczak, 2015a), zawierającego 29 milionów parametrów.



Rys. 5.3: Architektura modelu *E2018*, redukującego liczbę parametrów do 1,7 miliona.

Oznaczenia:

(C-K: H×W) – warstwa splotowa z K filtrami o wysokości H i szerokości W, **(P: H×W, S_H×S_W)** – *max-pooling* z oknem H×W i skokiem S_H×S_W, **(FC-N)** – warstwa w pełni połączona z N neuronami, **(BN + LReLU)** – normalizacja partiami w połączeniu z aktywacją typu *Leaky ReLU*, **(D: p%)** – *dropout* z prawdopodobieństwem odrzucenia p%.

z 5000 neuronów typu *ReLU*, oraz przez warstwę wyjściową *softmax* z liczbą neuronów odpowiadającą liczbie klas (50 dla *ESC-50*, 10 dla *ESC-10* i *UrbanSound8K*). Zarówno po pierwszej warstwie splotowej, jak i po warstwach w pełni połączonych aplikowany jest *dropout* z prawdopodobieństwem 50%.

Omawiana architektura swoją konstrukcją przypomina model *LeNet-5* (LeCun et al., 1998b) z modyfikacjami wprowadzonymi przez Abdel-Hamida et al. (2014) (zastosowanie drugiego kanału w postaci spektrogramów różnicowych). Jej istotnym wyróżnikiem jest jednak zamiana małych filtrów kwadratowych na filtry o prostokątnym kształcie (wertykalne), które uczą się wzorców w dziedzinie częstotliwości. Powoduje to, że po przetworzeniu zapewnianym przez pierwszą warstwę sieci, dalsze operacje splotowe dokonują się tylko w dziedzinie czasu. Zaletą tego podejścia jest jego skalowalność względem zwiększającej się liczby analizowanych pasm częstotliwości (Piczak, 2017) – wyniki przedstawiane w rozdziale 7 potwierdzają też, że modele tego typu cechują się dobrym stosunkiem uzyskiwanej dokładności klasyfikacji do złożoności obliczeniowo-pamięciowej.

5.2.3 Procedura uczenia

Zastosowany proces uczenia sieci zakłada optymalizację wag modelu poprzez wykorzystanie metody spadku stochastycznego gradientu z regularyzacją typu L_2 o sile $\alpha = 0{,}001$, której celem jest minimalizacja wartości entropii krzyżowej. Wagi w warstwach splotowych inicjalizowane są rozkładem jednostajnym odpowiednio $\mathcal{U}(-0{,}001;\, 0{,}001)$ (pierwsza warstwa) lub $\mathcal{U}(-0{,}1;\, 0{,}1)$ (druga warstwa). Wagi dla pozostałych warstw ustalane są według rozkładu normalnego $\mathcal{N}\left(\mu = 0\,;\, \sigma^2 = 0{,}0001\right)$. Wagi obciążające we wszystkich warstwach podlegają inicjalizacji wartościami zerowymi, z wyjątkiem pierwszej warstwy splotowej, dla której początkową wartością jest 0,1.

Uczenie sieci przeprowadzane jest na partiach liczących po 1000 segmentów każda, z tempem uczenia 0,01 (dla wariantu z krótkimi segmentami odpowiednio 0,002) i *momentum* Nesterova o wartości 0,9. Ze względu na zastosowaną implementację zwiększania zbioru danych poprzez losowy wybór fragmentów nagrań, w ramach pojedynczej epoki przetwarzane jest iteracyjnie 30–35 partii przykładów (w zależności od konkretnego zbioru). Przy przyjętych limitach 150 (dłuższe segmenty) lub 300 epok (krótsze segmenty), kompletny proces uczenia zamyka się tym sposobem w przybliżeniu w 5000 lub 10 000 kroków aktualizujących parametry.

5.2.4 Porównanie z uaktualnioną wersją modelu

Omawiany w podrozdziale 2.3 proces rozwoju architektur splotowych wskazuje na dosyć istotną tendencję pojawiającą się wśród następców *VGGNet* **(Simonyan & Zisserman, 2014)**, którą jest odchodzenie od wykorzystania na końcowym etapie przetwarzania szerokich warstw w pełni połączonych i zastąpienie tego typu rozwiązań głębszymi architekturami w pełni splotowymi. Ze względu na występujące w tym aspekcie podobieństwo zaproponowanego modelu do *VGGNet*, *E2015* podziela również jej słabszą stronę – wprowadzaną przez końcowe warstwy dużą liczbę parametrów (łącznie dla *E2015* jest to około 29 milionów wartości), która przy skali analizowanych problemów okazuje się być w istotnym stopniu nadmiarowa **(Jin et al., 2017)**. Z tego powodu do porównania wyników dołączony został przedstawiony na rysunku 5.3 uaktualniony model *E2018*, który poprzez zastosowanie normalizacji partiami umożliwia zastąpienie *E2015* architekturą głębszą, w pełni splotową, o liczbie parametrów zredukowanej do około 1,7 miliona. W połączeniu z drobnymi zmianami procesu uczenia podejście to pozwala w krótszym czasie uzyskać modele, które są bardziej kompaktowe i w dużym stopniu zachowują dokładność klasyfikacji wcześniejszej architektury.

W odróżnieniu od oryginalnie zaproponowanego modelu, *E2018* przetwarza segmenty spektrogramów o rozmiarze 60×125 wygenerowane z następującymi ustawieniami: częstotliwość próbkowania – 44 100 Hz, liczba pasm melowych – 60, górna granica częstotliwości

wyznaczanych pasm – 16 000 Hz, długość okna FFT – 2205 próbek (50 ms), długość skoku – 882 próbek (20 ms). Zmiana roboczej częstotliwości próbkowania, poza zwiększeniem zakresu częstotliwości reprezentowanych na wejściowym spektrogramie, pozwala również na przetwarzanie najczęściej spotykanych zbiorów danych bez konieczności ich konwersji, bezpośrednio w wejściowym formacie.

Analogicznie do modelu *E2015*, pierwsza warstwa jest warstwą splotową z filtrami o rozmiarze 57×6 (w tym wypadku są to 64 filtry), operującą na danych dwukanałowych (spektrogram bazowy i różnicowy), po której następuje normalizacja partiami (*batch normalization*), aktywacja typu *Leaky ReLU* ($\alpha = 0{,}01$) i grupowanie typu *max-pooling* o rozmiarze 4×2 ze skokiem 1×2 zakończone *dropoutem* przeprowadzanym względem map aktywacji[4]. W dalszej części konstrukcja *E2018* opiera się na modułach składających się z ciągu: warstwa splotowa 1×3 – normalizacja – *Leaky ReLU* – warstwa splotowa 1×3 – normalizacja – *Leaky ReLU* – *max-pooling* 1×2 ze skokiem 1×2. Na całość modelu składają się następujące po sobie trzy takie moduły ze zwiększającą się liczbą filtrów w warstwach splotowych (odpowiednio 128, 256, 512), po których aplikowany jest *dropout* z prawdopodobieństwem 0,5 i wyjściowa klasyfikacja za pomocą warstwy *softmax* (z liczbą neuronów odpowiadającą liczbie klas).

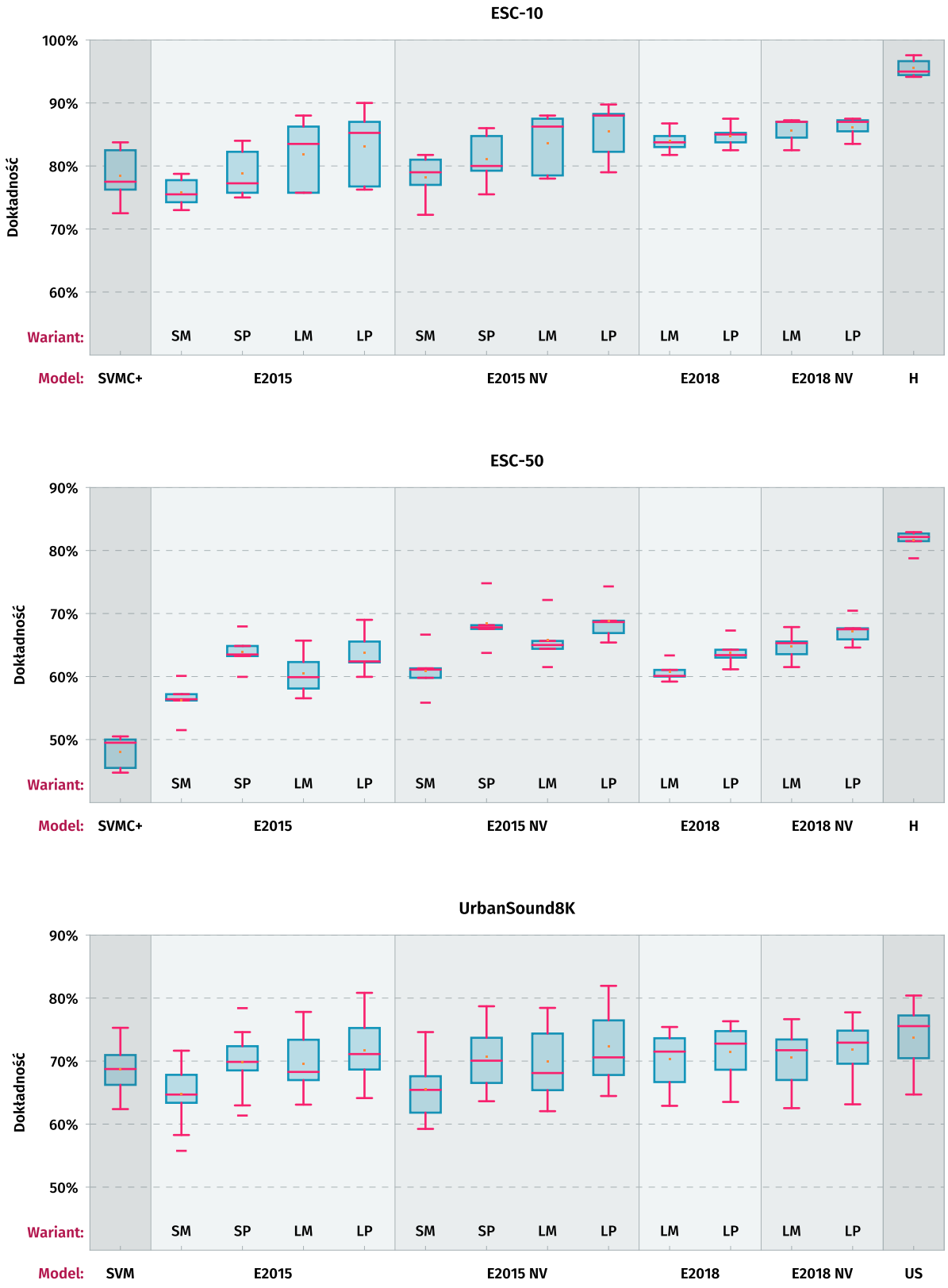
Uczenie modelu *E2018* prowadzone jest wyłącznie w wariancie odpowiadającym długim segmentom (w tym wypadku liczącym 125 ramek) partiami składającymi się ze 128 przykładów. Mniejszy rozmiar partii pozwala na redukcję liczby epok do 20 przy przybliżonym zachowaniu łącznej liczby kroków aktualizujących. Pozostałe aspekty uczenia nie różnią się od odpowiadających im ustaleń dla *E2015* z wyjątkiem wykorzystania gęstszego nakładania się segmentów (80%) przy wyznaczaniu zagregowanych predykcji dla zbioru testowego i zastosowania domyślnych dla *PyTorch* metod inicjalizacji wag.

5.3 Omówienie wyników klasyfikacji

Opisywane modele podlegały ocenie na zasadzie walidacji krzyżowej (*n-fold cross-validation*) opierającej się na pięciu (*ESC-10* i *ESC-50*) lub dziesięciu (*UrbanSound8K*) podzbiorach[5]. W oryginalnej formule przedstawionej w artykule **(Piczak, 2015a)**, w czasie gdy jeden z podzbiorów służył jako zbiór walidacyjny, na bazie którego podejmowane były decyzje co do dostrajania hiperparametrów i wyboru architektury na wczesnym etapie eksperymentów, końcowe wyniki prezentowane w publikacji obliczane były na zupełnie odrębnym zbiorze testowym. Taki sposób postępowania powodował, że w przypadku *ESC-10*

[4] Mowa o wariancie określanym jako *SpatialDropout* **(Tompson et al., 2015)** lub *Dropout2d*: https://pytorch.org/docs/stable/nn.html#torch.nn.Dropout2d.

[5] Raportowane w dalszej części wartości liczbowe są uśrednionym wynikiem eksperymentów powtórzonych dla pięciu wartości ziarna losowego.



Rys. 5.4: Porównanie rozkładów dokładności klasyfikacji względem podzbiorów testowych dla poszczególnych wariantów klasyfikatorów. W pierwszej kolumnie umieszczone zostały wyniki metod bazowych: *SVMC+* (podrozdział 4.4.2) lub *SVM* (Salamon et al., 2014). Ostatnia kolumna zawiera wyniki referencyjne: eksperymentu crowdsourcingowego (*H* – podrozdział 4.4.1) lub pracy Salamona & Bello (2015) (*US*).

Tab. 5.1: Dokładność klasyfikacji dla poszczególnych podzbiorów testowych *ESC-10* i *ESC-50*.

Wariant	Podzbiory ESC-10						Podzbiory ESC-50					
	μ_{acc}	1	2	3	4	5	μ_{acc}	1	2	3	4	5
SVMC+ (rozdz. 4.4.2)	**78,5**	72,5	83,8	76,2	82,5	77,5	**48,1**	45,5	49,5	50,5	50,0	44,8
E2015 SM	**75,9**	73,0	75,5	78,8	77,8	74,2	**56,3**	57,2	51,5	56,4	60,1	56,2
E2015 SP	**78,8**	75,8	75,0	82,2	84,0	77,2	**63,9**	63,2	60,0	64,8	68,0	63,5
E2015 LM	**81,9**	75,8	83,5	75,8	86,2	88,0	**60,5**	62,3	59,9	56,6	65,7	58,1
E2015 LP	**83,0**	76,8	85,2	76,2	87,0	90,0	**63,9**	65,6	62,3	60,0	69,0	62,4
E2015 SM NV	**78,2**	72,2	79,0	81,0	81,8	77,0	**60,9**	61,1	55,8	59,8	66,6	61,3
E2015 SP NV	**81,1**	75,5	80,0	84,8	86,0	79,2	**68,4**	67,8	63,7	67,6	74,8	68,2
E2015 LM NV	**83,6**	78,5	86,2	78,0	88,0	87,5	**65,7**	65,6	64,4	61,5	72,2	65,0
E2015 LP NV	**85,4**	79,0	88,0	82,2	89,8	88,2	**68,8**	68,8	66,9	65,4	74,3	68,6
E2018 LM	**84,0**	81,8	84,8	83,8	83,0	86,8	**60,7**	61,0	59,2	60,1	63,4	60,0
E2018 LP	**84,8**	82,5	85,0	85,2	83,8	87,5	**63,8**	63,4	61,1	64,2	67,3	63,0
E2018 LM NV	**85,6**	82,5	87,2	84,5	87,0	87,0	**64,8**	65,6	63,6	61,5	67,8	65,3
E2018 LP NV	**86,1**	83,5	87,2	85,5	87,5	87,0	**67,2**	67,5	64,6	65,9	70,4	67,6
H (rozdz. 4.4.1)	**95,5**	95,0	96,6	94,1	97,6	94,4	**81,6**	82,7	81,5	78,8	82,1	82,9

W tabeli przedstawione zostały procentowe dokładności klasyfikacji dla kolejnych podzbiorów testowych *ESC-10* i *ESC-50* oraz wartości średnie przeprowadzonej walidacji krzyżowej (μ_{acc}) uzyskane przez poszczególne warianty modeli. Dla porównania zamieszczone zostały również wyniki bazowego klasyfikatora (*SVMC+*) i eksperymentu crowdsourcingowego (*H*), opisywane w podrozdziałach 4.4.1 i 4.4.2.

i *ESC-50* uczenie modelu przebiegało efektywnie na nagraniach stanowiących tylko 60% całości zbioru. Większość późniejszych publikacji innych autorów nie przyjmowała tak restrykcyjnego założenia i stosowała wyłącznie pojedynczy zbiór testowy przy walidacji krzyżowej. W związku z tym, dla celów lepszej porównywalności wyników, prezentowane zestawienie zostało rozszerzone o warianty uczone według takiej właśnie zasady. Oznaczane są one w dalszej części skrótem *NV* (*no validation*).

Porównanie dokładności klasyfikacji osiąganej przez poszczególne modele przedstawiają wykresy na rysunku 5.4, dla których szczegółowe dane liczbowe zawarte zostały w tabelach 5.1 i 5.2. Do oznaczenia poszczególnych wariantów wykorzystane zostały skróty:

- *SM* (*short/majority*) – krótkie segmenty z agregacją typu *majority voting*,
- *SP* (*short/probability*) – krótkie segmenty z agregacją uśredniającą,
- *LM* (*long/majority*) – długie segmenty z agregacją typu *majority voting*,
- *LP* (*long/probability*) – długie segmenty z agregacją uśredniającą.

Analiza wyników pozwala na poczynienie kilku spostrzeżeń. Po pierwsze, efektem występującym uniwersalnie jest wzrost dokładności klasyfikacji związany z zastąpieniem agregacji typu *majority voting* przez wykorzystanie predykcji uśrednionej (*probability voting*). W większości przypadków lepsze rezultaty przynosi również operowanie na segmentach dłuższych. Dysproporcja między modelami o różnej wielkości segmentów jest szczególnie widoczna dla agregacji typu *majority voting*.

Tab. 5.2: Dokładność klasyfikacji dla poszczególnych podzbiorów testowych *UrbanSound8K*.

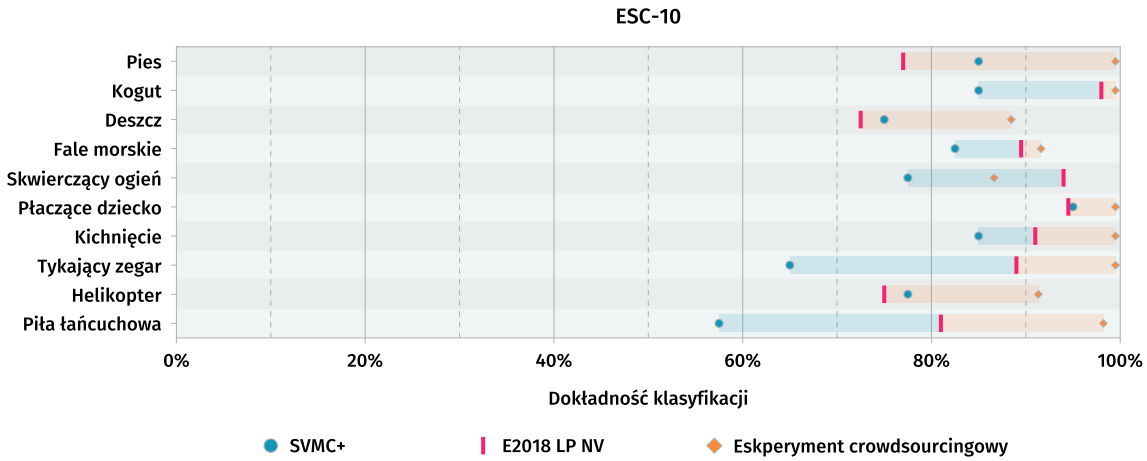
Wariant	Podzbiory UrbanSound8K										
	μ_{acc}	1	2	3	4	5	6	7	8	9	10
SVM (Salamon et al., 2014)	**68,7**	*62,4*	*66,2*	*68,7*	*71,0*	*75,3*	–	–	–	–	–
E2015 SM	**64,7**	63,6	58,3	55,8	65,6	71,6	63,3	65,8	63,8	68,5	71,0
E2015 SP	**69,9**	68,1	63,0	61,4	69,9	78,4	69,9	70,7	69,8	72,9	74,6
E2015 LM	**69,6**	67,6	64,8	63,1	67,7	77,8	69,0	66,8	68,9	75,2	74,9
E2015 LP	**71,7**	70,3	67,1	64,1	69,3	80,8	72,6	68,4	71,9	76,1	76,4
E2015 SM NV	**65,5**	61,3	59,5	59,2	64,7	71,7	66,2	66,8	63,3	67,9	74,6
E2015 SP NV	**70,6**	65,8	63,6	64,7	69,0	78,7	74,2	71,1	68,9	72,1	78,3
E2015 LM NV	**69,9**	65,1	62,0	65,2	65,9	78,4	73,4	68,7	67,5	74,7	78,1
E2015 LP NV	**72,3**	67,7	64,5	67,0	68,1	81,9	75,8	70,8	70,4	76,7	80,0
E2018 LM	**70,3**	63,4	65,3	62,9	73,7	73,5	70,7	72,3	70,7	75,1	75,4
E2018 LP	**71,4**	65,6	67,9	63,5	74,1	75,0	72,1	73,4	70,9	75,7	76,3
E2018 LM NV	**70,5**	66,4	66,0	62,5	74,7	73,3	71,5	71,9	68,8	73,5	76,6
E2018 LP NV	**71,8**	67,7	69,5	63,1	74,8	75,2	73,2	72,6	69,8	74,8	77,7
US (Salamon & Bello, 2015)	**73,7**	*64,7*	*70,5*	*75,6*	*77,2*	*80,4*	–	–	–	–	–

W tabeli przedstawione zostały procentowe dokładności klasyfikacji dla kolejnych podzbiorów testowych *UrbanSound8K* oraz wartości średnie przeprowadzonej walidacji krzyżowej (μ_{acc}) uzyskane przez poszczególne warianty modeli. Dla porównania zamieszczone zostały również wyniki bazowej metody zaproponowanej przez Salamona et al. (2014) (*SVM*) oraz podejścia opartego na uczeniu bez nadzoru (*US* – *unsupervised feature learning*) (Salamon & Bello, 2015). W związku z niedostępnością dokładnych wartości uzyskanych przez te metody dla poszczególnych podzbiorów testowych, tabela zawiera w ich przypadku wyniki przybliżone (kwartyle oraz wartości minimalne i maksymalne), nieprzypisane do indywidualnych podzbiorów.

Łatwiejszy charakter problemu stawianego przez zbiór *ESC-10* sprawia, że względna poprawa zdolności klasyfikacyjnych wynikających z zastosowania modeli splotowych nie jest w jego przypadku tak wyraźna, jak dla *ESC-50*. Wzrost dokładności przy 50 klasach (48,1% dla modelu bazowego, 68,8% dla *E2015 LP NV*) jest natomiast na tyle istotny, że zdecydowanie uzasadnia wprowadzenie podejścia splotowego w miejsce stosowanych wcześniej metod. Niemniej nawet w tym wypadku, bez wykorzystania dodatkowych zewnętrznych danych uczących, modele te nie są w stanie dorównać ludzkim możliwościom w tym zakresie.

Na tym tle wyniki dla *UrbanSound8K* kształtują się bardziej równomiernie – rozpiętość dokładności poszczególnych podejść jest niewielka: model bazowy (*SVM*) – 68,7%, najlepszy model splotowy (*E2015 LP NV*) – 72,3%[6], metoda Salamona & Bello (2015) (*US*) – 73,7%. Przegląd bieżących publikacji odnoszących się do tego zbioru zdaje się potwierdzać, że jest to jego cecha charakterystyczna i niezależnie od obranej metody następuje nasycenie dokładności klasyfikacji przy zbliżaniu się do progu 75%, któremu można zaradzić tylko poprzez odpowiednie deformacje zbioru uczącego (Salamon & Bello, 2017) lub zmiany

[6] Dla oryginalnej implementacji w *Pylearn2* (eksperyment przeprowadzany dla pojedynczej wartości ziarna losowego, bez uśredniania) model *E2015 LP* uzyskiwał dokładność na poziomie 73,1% (Piczak, 2015a).



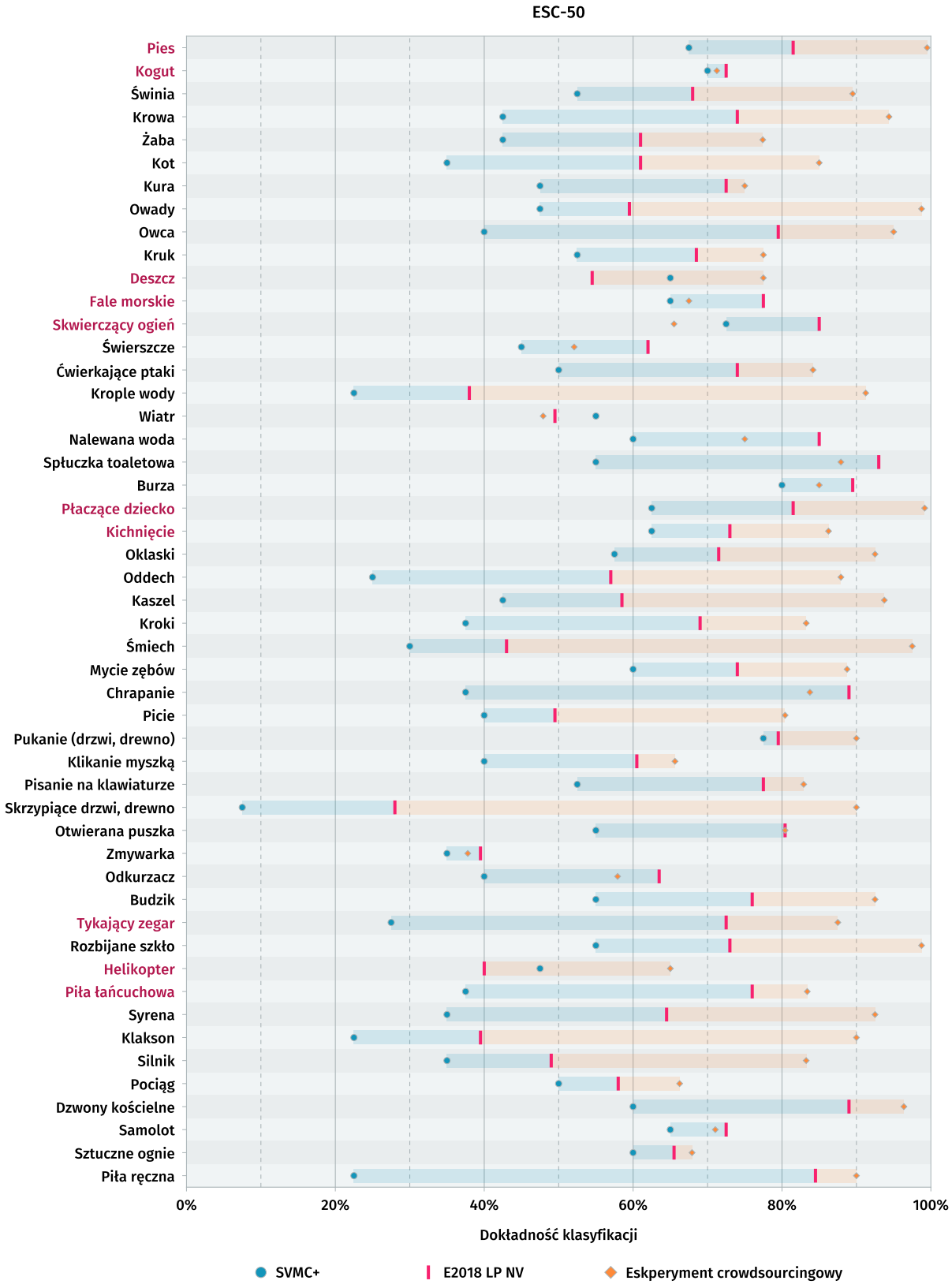
Rys. 5.5: Porównanie dokładności klasyfikacji poszczególnych klas zbioru *ESC-10* osiąganych przez model *SVM* w wariancie „*C+*”, sieć splotową *E2018 LP NV* i uczestników eksperymentu crowdsourcingowego. Obszar zacieniowany wyraża wzrost dokładności między kolejnymi metodami.

w procesie uczenia (Tokozume et al., 2017). Warto też w tym miejscu zwrócić uwagę na wpływ obciążenia modelu pod kątem konkretnych klas na uzyskiwany ostatecznie wynik. Niezbalansowany charakter *UrbanSound8K* powoduje, że ocena za pomocą przyjętej miary przywiązuje mniejszą wagę do klas z ograniczoną liczbą przykładów (głównie *strzały z broni palnej* i *klakson*).

Z kolei bliższe spojrzenie na zachowanie modeli wykorzystujących architekturę *E2018* pokazuje, że warianty te radzą sobie względnie dobrze mimo ograniczenia liczby wykorzystywanych przez nie parametrów, osiągając dokładność niewiele gorszą niż odpowiedniki *E2015*. Co więcej, wyniki *E2018* cechują się mniejszą dyspersją dla poszczególnych podzbiorów, najsilniej widoczną w przypadku *ESC-10*.

Dokładniejsza analiza skuteczności klasyfikacji dla poszczególnych klas *ESC-10* (rys. 5.5) i *ESC-50* (rys. 5.6) potwierdza wcześniejszy wniosek, że modele splotowe stają się szczególnie przydatne przy zwiększającej się liczbie rozważanych klas. W uproszczonym problemie stawianym przez *ESC-10* poprawa zdolności klasyfikacyjnych względem modelu bazowego nie jest jednorodna – w przypadku czterech z dziesięciu klas to właśnie model bazowy okazuje się nieco dokładniejszy. Dopiero po zwiększeniu złożoności zadania przewaga podejścia splotowego staje się bardzo wyraźna.

Największą poprawę dokładności modelu *E2018 LP NV* można zauważyć dla klas: *piła ręczna*, *chrapanie*, *tykający zegar*, *owca*, *piła łańcuchowa* i *spłuczka toaletowa* (różnica co najmniej 38 punktów procentowych względem *SVMC+*). Wynika ona przypuszczalnie z bardziej lokalnego w czasie przetwarzania cech przez model splotowy. Podejście bazowe, operując na statystykach agregowanych dla całego nagrania, preferuje klasy, dla których bliższe jest zachowanie założenia o stacjonarności. Właśnie dla trzech tego typu klas (*deszcz*, *helikopter*, *wiatr*) model *SVMC+* przewyższa dokładnością wariant splotowy.

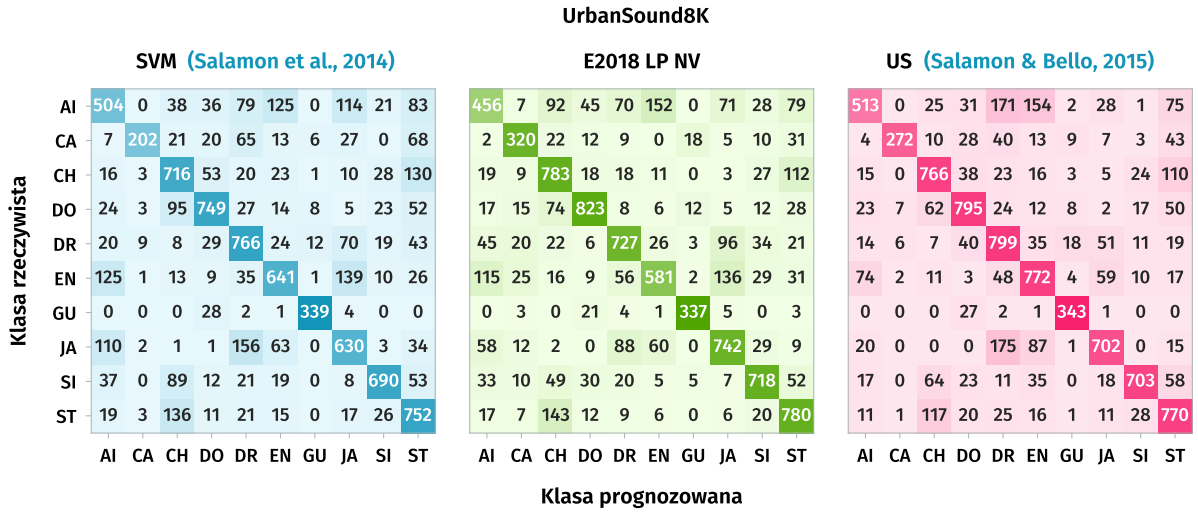


Rys. 5.6: Porównanie dokładności klasyfikacji poszczególnych klas zbioru *ESC-50* osiąganych przez model *SVM* w wariancie „*C+*”, sieć splotową *E2018 LP NV* i uczestników eksperymentu crowdsourcingowego. Obszar zacieniowany wyraża wzrost dokładności między kolejnymi metodami. Klasy należące do podzbioru *ESC-10* zaznaczone zostały kolorem ciemnoróżowym.

Tab. 5.3: Zestawienie najczęstszych błędów modelu *E2018 LP NV* dla zbioru *ESC-50*.

		Najczęstsze błędne klasyfikacje (udział)					
Klasa (dokładność klas.)		#1		#2		#3	
Skrzypiące drzwi	28,0%	Płaczące dziecko	9,0%	Oklaski	7,0%	Kot	7,0%
Krople wody	38,0%	Tykający zegar	8,5%	Otwierana puszka	8,0%	Klikanie myszką	7,5%
Zmywarka	39,5%	Pociąg	11,5%	Owady	7,0%	Helikopter	7,0%
Klakson	39,5%	Samolot	6,0%	Kot	5,0%	Helikopter	4,5%
Helikopter	40,0%	Samolot	17,0%	Wiatr	12,5%	Zmywarka	6,5%

Przy porównywaniu zachowania modelu *E2018 LP NV* z wynikami eksperymentu crowdsourcingowego, zestaw klas znacząco trudniejszych dla podejść automatycznych pozostaje w dużej części niezmieniony w stosunku do wariantu bazowego. Dalej odgłosy *skrzypiących drzwi*, *śmiechu*, *kropli wody* czy *klaksonu* wskazują na dużą dysproporcję między możliwościami klasyfikatorów splotowych a zdolnościami uczestników. Pocieszającą informacją może być fakt, że najczęstsze błędy popełniane przez rozważany model (tabela 5.3) daje się częściowo wytłumaczyć faktyczną percepcyjną bliskością między klasą rzeczywistą a niepoprawną predykcją (np. pary *helikopter–samolot*, powtarzalne padanie *kropli wody* i *tykanie zegara* czy stukot naczyń w *zmywarce* i kół *pociągu* po torach). Z drugiej strony, dosyć równomierne rozproszenie błędów na wiele potencjalnych klas w przypadku klasyfikacji nagrań *skrzypiących drzwi* czy *klaksonu* może wskazywać na trudny charakter nagrań źródłowych dla niektórych z klas lub niedoskonałość samego procesu uczenia.



Rys. 5.7: Zestawienie macierzy pomyłek klasyfikacji zbioru *UrbanSound8K* przez modele *SVM* (Salamon et al., 2014), *E2018 LP NV* i *US* (Salamon & Bello, 2015). Dla lepszej porównywalności z artykułami zachowany został oryginalny sposób prezentacji (komórki przedstawiają liczby przykładów) i skróty nazw: AI – *klimatyzacja*, CA – *klakson*, CH – *bawiące się dzieci*, DO – *szczekający pies*, DR – *wiercenie*, EN – *silnik na biegu jałowym*, GU – *strzały z broni palnej*, JA – *młot pneumatyczny*, SI – *syrena*, ST – *muzyka uliczna*.

Podobne wnioski nasuwają się przy porównywaniu macierzy pomyłek klasyfikacji zbioru *UrbanSound8K* (rys. 5.7). Najczęstsze błędy poszczególnych klasyfikatorów również i w tym wypadku skupiają się w grupach klas zbliżonych do siebie percepcyjnie. Widoczne jest natomiast w rozważanych podejściach przesunięcie granic decyzyjnych – mimo podobnej globalnej dokładności klasyfikacji, modele te w zróżnicowany sposób radzą sobie z konkretnymi typami dźwięków.

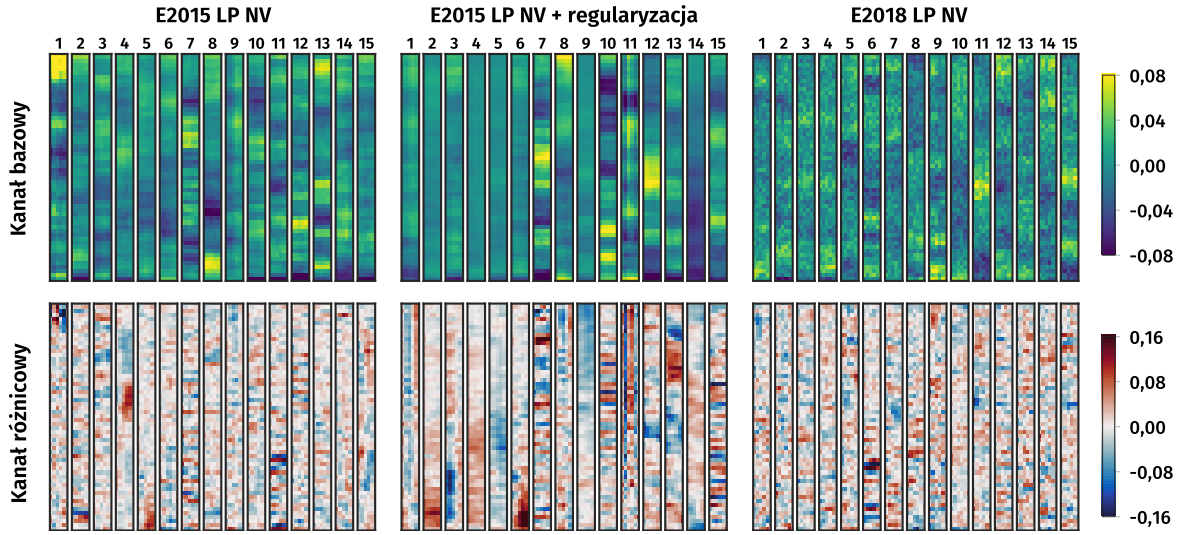
Sieć *E2018 LP NV* jest na przykład bardziej czuła na występujące rzadziej w zbiorze przykłady *klaksonu*, ale okupione to jest małą precyzją tych predykcji. Bezwzględnie lepsze wyniki uzyskuje natomiast dla klasy *szczekającego psa*. Z kolei hałas *klimatyzacji* i *silnika na biegu jałowym* są częściej mylone między sobą właśnie przez model splotowy. Dźwięki *wystrzałów z broni palnej* są na tyle charakterystyczne, że niezależnie od stosowanego podejścia notują mało błędnych klasyfikacji. Niektóre pomyłki wynikają też z samej specyfiki zbioru – bliższa analiza nagrań klas *bawiących się dzieci* i *muzyki ulicznej* pokazuje, że część z nich zawiera bardzo podobne elementy akustyczne, na podstawie których trudno dokonać jednoznacznej klasyfikacji. Wydaje się więc, że zachowanie wyuczonych modeli jest bliskie oczekiwaniom co do tego, jak taka klasyfikacja powinna przebiegać i nie wskazuje raczej na występowanie niepokojących osobliwości w sposobie ich działania.

5.4 Wizualizacja efektu uczenia

Dodatkowym krokiem analizy klasyfikatora, który może podnieść poziom zaufania co do zasady jego funkcjonowania, a także pomóc dostrzec przyczyny ewentualnych niedoskonałości, jest próba wizualizacji modelu i zrozumienia jego działania na konkretnych przykładach ze zbioru danych. W ramach tego podrozdziału zostanie ona przeprowadzona dla klasyfikatora uczonego na zbiorze *ESC-50*.

Częstym punktem wyjścia przy analizie modeli splotowych jest wizualizacja wag (filtrów) pierwszej warstwy operującej bezpośrednio na danych wejściowych. Rysunek 5.8 przedstawia porównanie wariantów *E2018 LP NV* i *E2015 LP NV* właśnie z tej perspektywy. Ewidentną różnicą między nimi jest większe zaszumienie widoczne w wagach modelu *E2018 LP NV*, w szczególności w ramach pierwszego kanału (bazowego spektrogramu). Zbliżona procedura uczenia sugeruje, że występowanie tego efektu związane jest z odmiennością samej architektury – jego prawdopodobnym źródłem może być dużo częstsze i silniejsze wykorzystanie *dropoutu* przez *E2015*, które w pośredni sposób zapewnia filtry z gładszymi przejściami.

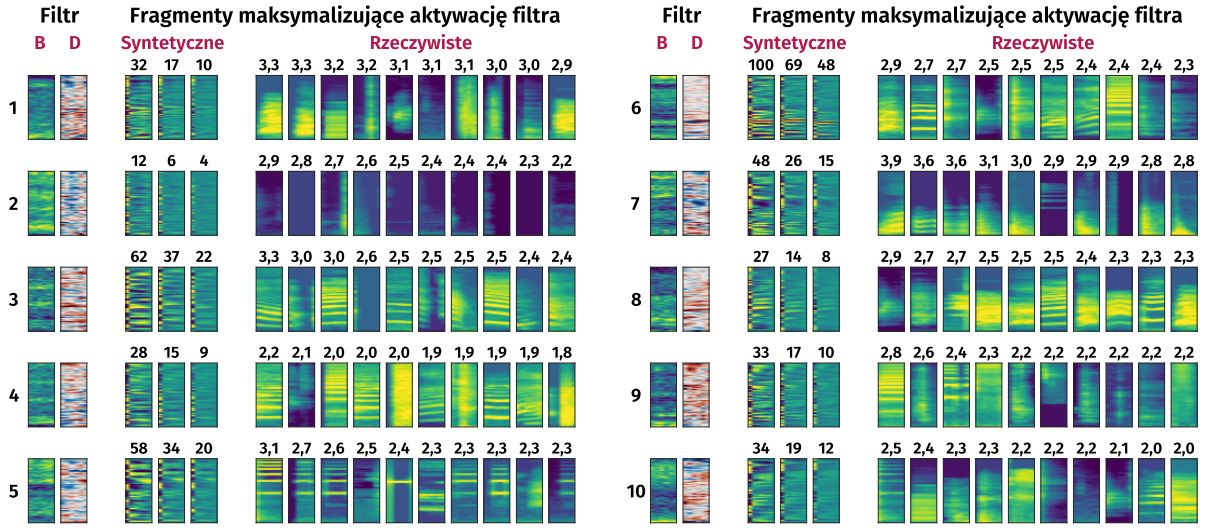
Pierwszy kanał filtrów występujących w analizowanych modelach (górny wiersz na rysunku 5.8) zawiera głównie wzorce w dziedzinie częstotliwości (różne kombinacje poziomych pasów). Dużo rzadziej pojawia się w nich aspekt zmian w czasie (takim osobliwym



Rys. 5.8: Wizualizacja pierwszej warstwy splotowej modeli uczonych na zbiorze *ESC-50*. Poszczególne wykresy przedstawiają wartości wag (dwa kanały) pierwszych piętnastu filtrów kolejno dla modelu *E2015 LP NV* ze standardowym ustawieniem regularyzacji ($\alpha = 0{,}001$), ze zwiększoną siłą regularyzacji ($\alpha = 0{,}01$) i wariantu *E2018 LP NV* ($\alpha = 0{,}001$). Wartości wag modelu ze zwiększoną siłą regularyzacji zostały podwojone dla uzyskania czytelniejszej prezentacji na wykresie.

przykładem jest filtr jedenasty dla modelu *E2015* z dodatkową regularyzacją), który z kolei jest bardzo wyraźny dla drugiego kanału różnicowego. Wartości otrzymywane dla tego kanału są jednak trudne w interpretacji – zarówno ze względu na wprowadzany dodatkowy aspekt przekształcenia różnicowego, ale też w związku z zaszumionym charakterem samych wag. Może to wskazywać na słabość wynikającą z niewielkiego rozmiaru zbioru uczącego i braku konieczności wytworzenia się filtrów gładkich o bardziej uniwersalnym charakterze. Zwiększenie siły wprowadzanej odgórnie regularyzacji typu L_2 z poziomu $\alpha = 0{,}001$ do $\alpha = 0{,}01$ (wykresy w środkowej części rysunku 5.8) co prawda poprawia wyrazistość uzyskiwanych filtrów od strony wizualnej, ale nie ma pozytywnego wpływu na dokładność klasyfikacji uzyskiwaną przez model. Porównanie zakresu wartości przyjmowanych przez wagi w poszczególnych kanałach pokazuje natomiast, że część różnicowa stanowi bardzo istotny element przetwarzania. Jednocześnie też, przy wizualizacji kompletnego zestawu filtrów poszczególnych modeli, pominiętej ze względu na ograniczoność miejsca do prezentacji, widoczne stają się przejawy typowej dla modeli splotowych redundancji. Różnice występujące między niektórymi filtrami są na tyle niewielkie, że sugerowałoby to potencjalną możliwość wprowadzenia w ich miejsce pojedynczych zastępników bez znaczącego ograniczania ekspresyjności modelu.

Trudność z bezpośrednią interpretacją niektórych aspektów rysunku 5.8 skłania do pójścia o krok dalej i przeanalizowania działania poszczególnych filtrów przez pryzmat przykładów danych, które maksymalizują ich aktywację. W tym celu dla każdego filtra utworzone zostały przykłady całkowicie syntetyczne oraz wybrane z rzeczywistych fragmentów należących do zbioru uczącego, tak aby uzyskana dla nich aktywacja filtra przyjmowała



Rys. 5.9: Zobrazowanie fragmentów spektrogramów maksymalizujących aktywację pierwszych dziesięciu filtrów pierwszej warstwy modelu *E2018 LP NV* uczonego na zbiorze *ESC-50*. Nad fragmentami umieszczone zostały wartości aktywacji uzyskiwane po przetworzeniu przez dany filtr splotowy (kanały: bazowy – B, różnicowy – D), normalizację partiami, aktywację *LeakyReLU* i operację grupowania. Fragmenty syntetyczne generowane były z rosnącą siłą regularyzacji, kolejno $\alpha_1 = 0{,}03$, $\alpha_2 = 0{,}06$, $\alpha_3 = 0{,}1$.

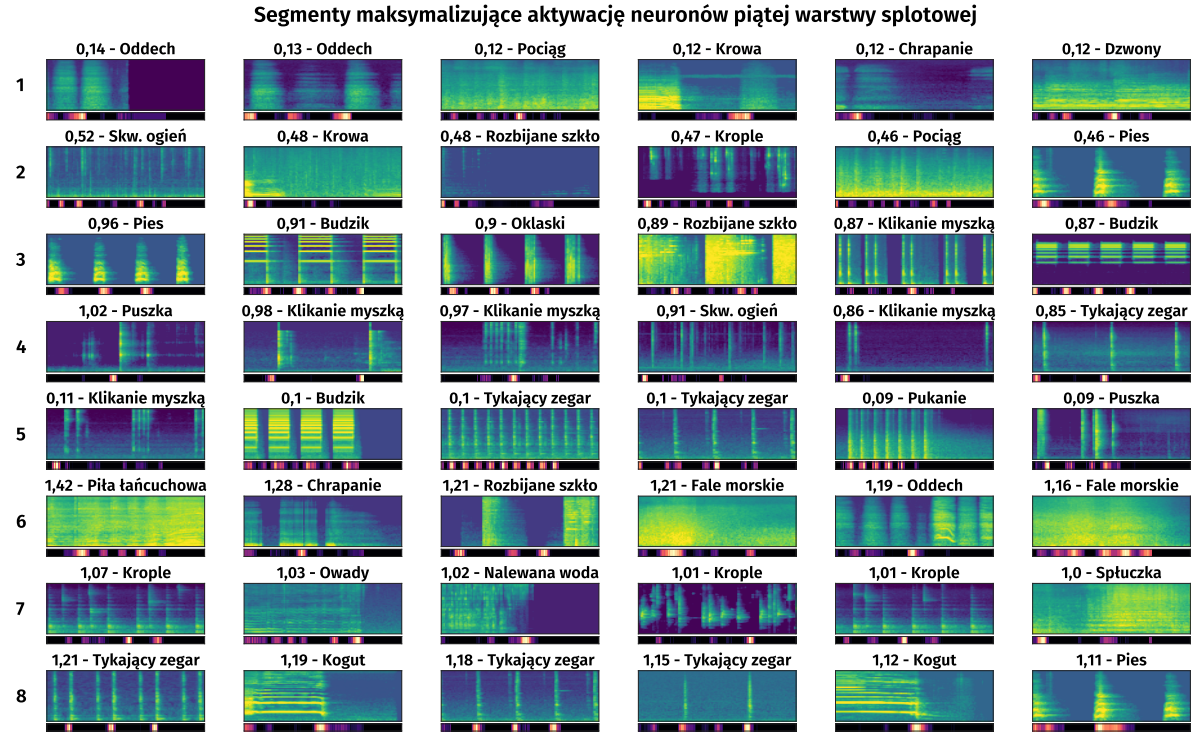
możliwie największe wartości[7]. Wizualizację tego typu zestawienia dla pierwszych dziesięciu filtrów modelu *E2018 LP NV* zawiera rysunek 5.9, na którym uwidaczniają się dwie główne tendencje.

Pierwsza z nich dotyczy fragmentów otrzymywanych w sposób syntetyczny. Chociaż wygenerowanie tego typu przykładów umożliwia uzyskanie bardzo dużych wartości aktywacji, to tworzone tym sposobem spektrogramy mają mało naturalny charakter – są to przede wszystkim impulsowe pobudzenia konkretnych kombinacji pasm częstotliwości.

Dużo ciekawszy obraz przedstawia natomiast analiza bazująca na rzeczywistych przykładach ze zbioru uczącego. Uzyskiwane w jej przypadku zgrupowanie pokazuje, że mimo miejscami mało czytelnego charakteru samych wartości wag, poszczególne filtry specjalizują się w rozpoznawaniu odmiennych elementów akustycznych. Mogą to być konkretne struktury harmoniczne (filtry 3 i 5), sygnały w szerszym paśmie (1 i 8) lub o dominujących niskich częstotliwościach (7), czy też w ogóle o mniejszym natężeniu względem tła (2).

Tego typu ocenę funkcjonowania poszczególnych filtrów można również przeprowadzić względem neuronów dalszych warstw, dla których wizualizacja samych wartości wag staje się już raczej bezcelowa. Przykład takiego zobrazowania dla piątej warstwy splotowej modelu *E2018 LP NV* zawiera rysunek 5.10. Różnicą w stosunku do wizualizacji pierwszej

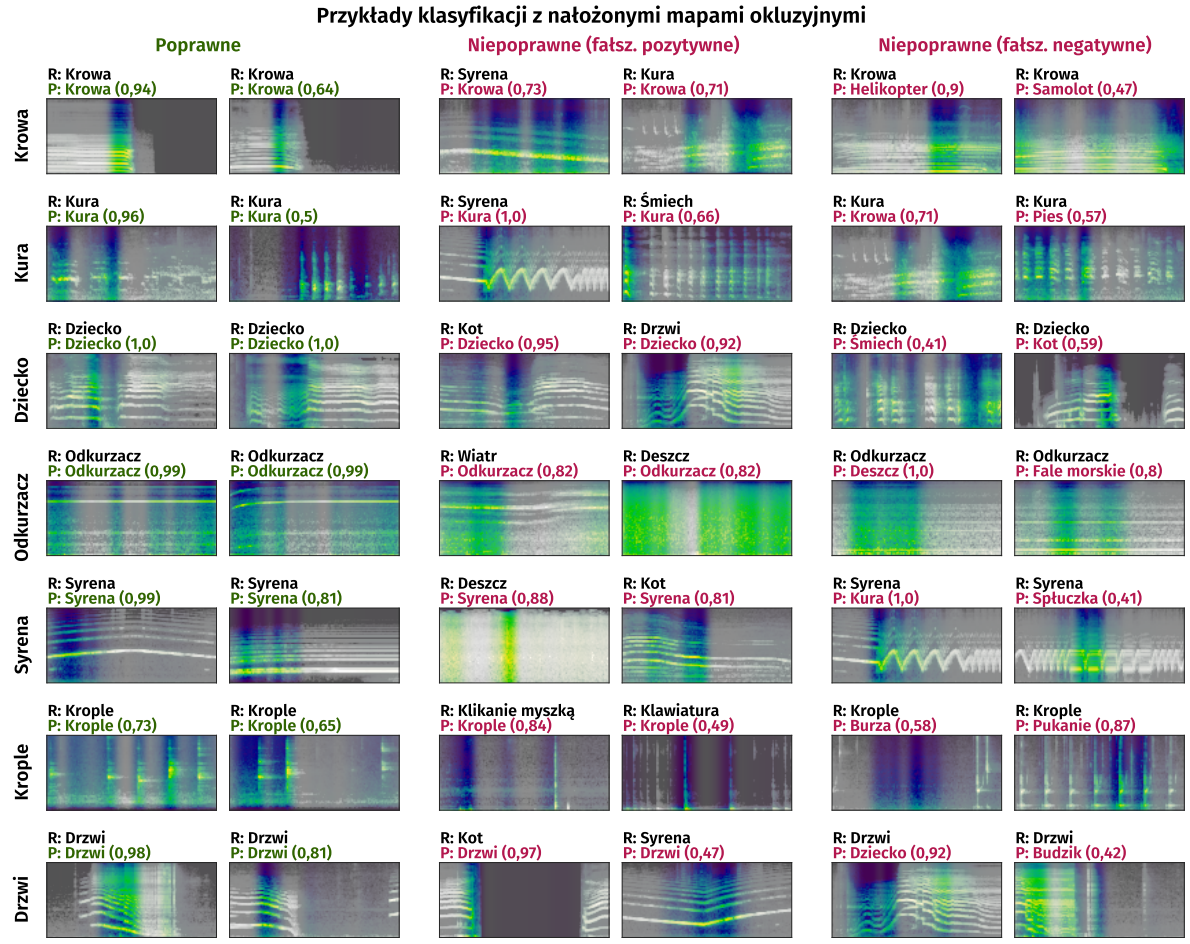
[7] Generowanie przykładów syntetycznych polegało na wyjściu od macierzy reprezentującej wejściowy fragment spektrogramu, zainicjalizowanej w sposób losowy zgodnie z rozkładem $\mathcal{N}(0\,;1)$, i takiej zmianie jej wartości, aby uzyskać maksymalną możliwą wartość aktywacji filtra. Do tego celu wykorzystana została metoda *Adam* z tempem uczenia 0,1 i trzema siłami regularyzacji typu L_2 ($\alpha_1 = 0{,}03, \alpha_2 = 0{,}06, \alpha_3 = 0{,}1$).



Rys. 5.10: Zestawienie segmentów spektrogramów maksymalizujących aktywację poszczególnych neuronów piątej warstwy splotowej modelu *E2018 LP NV* uczonego na zbiorze *ESC-50*. W kolejnych wierszach przedstawione zostały przykłady dla ośmiu pierwszych neuronów analizowanej warstwy. Każdy z segmentów ma rozmiary 60 pasm × 125 ramek. Nad segmentami umieszczone zostały wartości maksymalnej aktywacji dla danego neuronu wraz z etykietą określającą klasę, do której należy dany segment. Pod każdym ze spektrogramów jasnym kolorem na osi czasu oznaczone zostały miejsca intensywniejszego wzbudzania danego neuronu.

warstwy jest poszerzenie analizowanych przykładów do segmentów wejściowych o pełnej długości. Również i w tym wypadku zestawienie takie pozwala na dostrzeżenie pewnych specyficznych prawidłowości różnicujących działanie poszczególnych neuronów. Można więc mówić o neuronach funkcjonujących przede wszystkim jako detektory poszczególnych faz rozwoju dźwięku – nagłosu (*onset detector*) (1) lub wygłosu (*decay*) (3), także bardzo krótkiego, jak w przypadku zawartego w drugiej kolumnie przykładu *budzika*. Mocno specyficzny jest również neuron 4, który najbardziej aktywuje się dla dźwięków impulsowych w szerokim paśmie (*klikania* czy strzelania palącego się drewna). Innym ciekawym przykładem wydaje się neuron 7, który z jednej strony można interpretować jako skupiający się na elementach wygaszania dźwięku, ale specyficzne zgrupowanie klas w odpowiadających mu segmentach sugeruje również, że jest on szczególnie wyczulony na odgłosy wody.

Ostatnim krokiem analizy przeprowadzanej dla modelu *E2018 LP NV* jest ocena jego zachowania pod kątem wrażliwości klasyfikacji na występowanie konkretnych fragmentów nagrania. Rysunek 5.11 ilustruje ten aspekt zestawiając ze sobą przykłady poprawnej i błędnej klasyfikacji uzyskiwanej przez rozważany model dla siedmiu wybranych klas zbioru *ESC-50* i nakładając na te przykłady mapy okluzyjne uwypuklające kluczowe części



Rys. 5.11: Zestawienie przykładów klasyfikacji siedmiu wybranych klas zbioru *ESC-50* przez model *E2018 LP NV*. Nad spektrogramami umieszczone zostały oznaczenia faktycznej etykiety (*R*) i predykcji generowanej przez model na podstawie całości segmentu, wraz z jej prawdopodobieństwem (*P*). Większe nasycenie koloru spektrogramu obrazuje te jego fragmenty, dla których prawdopodobieństwo zwrócenia etykiety będącej predykcją dla całości segmentu jest największe.

zaprezentowanych segmentów. Mapy te zostały sporządzone w następujący sposób. Dla każdego punktu w czasie (kolumny) poszczególnych spektrogramów przeprowadzona została odrębna klasyfikacja, wyłącznie na podstawie danych zawartych w rozpoczynającym się od tego punktu oknie o długości 7, 15 lub 23 ramek. Zasłonięcie pozostałej zawartości nagrania następowało przez jej zastąpienie za pomocą wartości -1. Uzyskane tym sposobem prawdopodobieństwa wystąpienia klasy będącej jednocześnie predykcją modelu dla całości segmentu były następnie uśredniane do pojedynczej wartości dla każdego z punktów i wygładzane w czasie za pomocą okna Hamminga o długości 15 ramek. Tak utworzona jednowymiarowa mapa okluzyjna wykorzystywana była następnie przy sporządzaniu wykresów, określając stopień nasycenia koloru poszczególnych ramek spektrogramu.

Przyglądając się przykładom zilustrowanym na rysunku 5.11, widać, że dokonywana przez sieć klasyfikacja poszczególnych segmentów opiera się przede wszystkim na kluczowych fragmentach specyficznych dla danej klasy. Mnogość kolorowych obszarów na spektrogramach, występująca mimo wykorzystania krótkich okien do tworzenia map okluzyjnych, potwierdza dla wielu sytuacji możliwość dokonania predykcji z dużymi wartościami prawdopodobieństwa wystąpienia klasy już na podstawie zawartości nagrania stanowiącej tylko kilkanaście procent długości segmentu (kilkaset milisekund).

W przypadku odgłosów *krowy* będzie to koniec fazy wokalizacji z charakterystycznym opadaniem wysokości, które może być faktycznie pomylone z dźwiękiem *syreny*. Z kolei niepoprawna klasyfikacja jako *helikopter* (piąta kolumna) wynika prawdopodobnie z interpretowania krótkich przerw w sygnale jako odgłosu wirnika. Sytuacje te wskazują jednak na braki w odczytywaniu szerszego kontekstu.

Na podobnej zasadzie dźwięk *syreny* na początkowym etapie załączania można uznać za zbliżony zakresem dominujących częstotliwości i strukturą harmoniczną do odgłosów *kury*. Uwzględnienie jego cyklicznej powtarzalności w dłuższym okresie powinno natomiast pozwolić na całkowite wyeliminowanie wątpliwości klasyfikacyjnych, co w tym wypadku nie następuje.

Niejednoznaczności pojawiające się w grupie *płaczące dziecko – kot – skrzypiące drzwi* są o tyle zrozumiałe, że niektóre fragmenty nagrań mogłyby w istocie sprawić kłopot niejednemu niewystarczająco uważnemu słuchaczowi. Koty są w końcu znane z tego, że wykształciły taką charakterystykę miauczenia, aby celowo aktywować podobne obszary mózgu, jak płacz dziecka, wymuszając tym sposobem szybką reakcję opiekuna **(McComb et al., 2009)**.

Również problemy z rozróżnieniem między *kroplami wody*, *klikaniem myszką* czy naciskaniem klawiszy na *klawiaturze* pokazują, że większość błędów popełnianych przy klasyfikacji można uznać za względnie racjonalne. Ich wyeliminowanie wymagałoby raczej udoskonalenia metodyki i zwiększenia czułości sieci na czasem niewielkie różnice występujące między zbliżonymi dźwiękami niż zmiany ogólnej zasady działania modelu, do której mogłoby skłaniać pojawianie się pomyłek między klasami mocno oddalonymi w przestrzeni percepcyjnej.

5.5 Wnioski

Eksperymenty omawiane w ramach niniejszego rozdziału miały na celu zweryfikowanie, czy modele wykorzystujące splotowe sieci neuronowe mogą być odpowiednim narzędziem w zadaniach klasyfikacji dźwięków środowiskowych. Wydaje się, że w tym aspekcie można udzielić odpowiedzi jednoznacznie twierdzącej – zarówno patrząc na obiecujące względem

metod bazowych wyniki, które w przypadku *UrbanSound8K* są porównywalne z najlepszymi opublikowanymi podejściami, ale także odnosząc się do liczby powstałych na późniejszym etapie publikacji rozszerzających koncepcje zaprezentowane w tej części rozprawy.

Oryginalna architektura *E2015* okazała się w tej przestrzeni sprawdzoną bazą do implementacji modeli klasyfikujących dźwięki w oparciu o sieci splotowe z filtrami wertykalnymi i kanałem różnicowym. Jej uaktualnienie do siedmiowarstwowego modelu w pełni splotowego *E2018* zapewnia zbliżoną dokładność klasyfikacji przy istotnie zredukowanej całkowitej liczbie parametrów i wydaje się jeszcze lepszym punktem wyjścia dla nowszych implementacji tego typu modeli.

Przeprowadzona w rozdziale analiza wyników osiąganych przez omawiane sieci i wizualizacja efektu uczenia pozwalają stwierdzić, że zachowanie zaproponowanych modeli daje się określić jako w większości zgodne ze zdroworozsądkowym rozumieniem poprawnego procesu klasyfikacji dźwięku. Główne niedociągnięcia, jakich można się doszukiwać na tym polu, łączą się z silnym skupianiem się sieci na występujących w analizowanych nagraniach krótkich, specyficznych fragmentach, na bazie których dokonywana jest finalna decyzja co do proponowanej etykiety. Przewaga zastosowania dłuższych segmentów nad wersją krótszą pokazuje jednak, że sam podział nagrania zwiększający liczbę uśrednianych w czasie predykcji niekoniecznie jest najlepszym posunięciem. W tym względzie klasyfikacja uwzględniająca ten szerszy kontekst czasowy w bardziej usystematyzowany sposób i biorąca pod uwagę sekwencyjny proces kształtowania się pewnych zdarzeń dźwiękowych mogłaby okazać się obiecująca. Kierunek ten wydają się potwierdzać ostatnie prace w zakresie wykorzystania modeli z warstwami rekurencyjnymi **(Çakır et al., 2017)** lub ze skupianiem uwagi (*attention based networks*) **(Guo et al., 2017)**, a także sieci splotowych z jednoczesnym przetwarzaniem w różnych skalach czasowych (*multi-temporal resolution CNN*) **(Zhu et al., 2018)**.

Rozdział 6

Klasyfikacja gatunków ptaków za pomocą splotowych sieci neuronowych[1]

6.1 Wprowadzenie

Dobrze funkcjonujące systemy automatycznej analizy nagrań dźwiękowych, które pozwalałyby na rozpoznawanie występujących gatunków ptaków na odpowiednio dużą skalę, byłyby bardzo cennym narzędziem w rękach badaczy i organizacji publicznych zajmujących się zagadnieniami monitorowania ekosystemów i zachowania bioróżnorodności. W przeciwieństwie do obserwacji prowadzonych przez zawodowych ornitologów i hobbystów, sieci sensorów akustycznych (Cai et al., 2007; Mporas et al., 2013; Wimmer et al., 2013; Lostanlen et al., 2018) nie są ograniczone trudnymi warunkami środowiskowymi czy pojawiającym się zmęczeniem obserwatorów. Umożliwia im to niestrudzone dostarczanie nowych porcji danych, których ilość bardzo szybko zaczyna przerastać zasoby ludzkie dostępne do ich ręcznej analizy.

Przez lata inicjowane były różne przedsięwzięcia mające na celu rozwój i porównywanie modeli automatycznego rozpoznawania gatunków ptaków na podstawie danych dźwiękowych (Stowell & Plumbley, 2010). Niestety, przy ponad 500 gatunkach występujących na terenie samej Unii Europejskiej (ORNIS, 2016) i blisko 10 000 na całym świecie (IOC, 2016), większość z przeprowadzanych w tym obszarze eksperymentów była bardzo ograniczona w porównaniu z zakresem problemów spotykanych w rzeczywistych warunkach. Konkurs organizowany przy konferencji *NIPS 2013* (Glotin et al., 2013) obejmował 87 klas dźwiękowych, w czasie gdy jego odpowiedniki w ramach *ICML 2013* (Glotin, 2013) i *MLSP 2013* (Briggs et al., 2013) wypadały pod tym względem jeszcze skromniej (odpowiednio 35 i 19 gatunków).

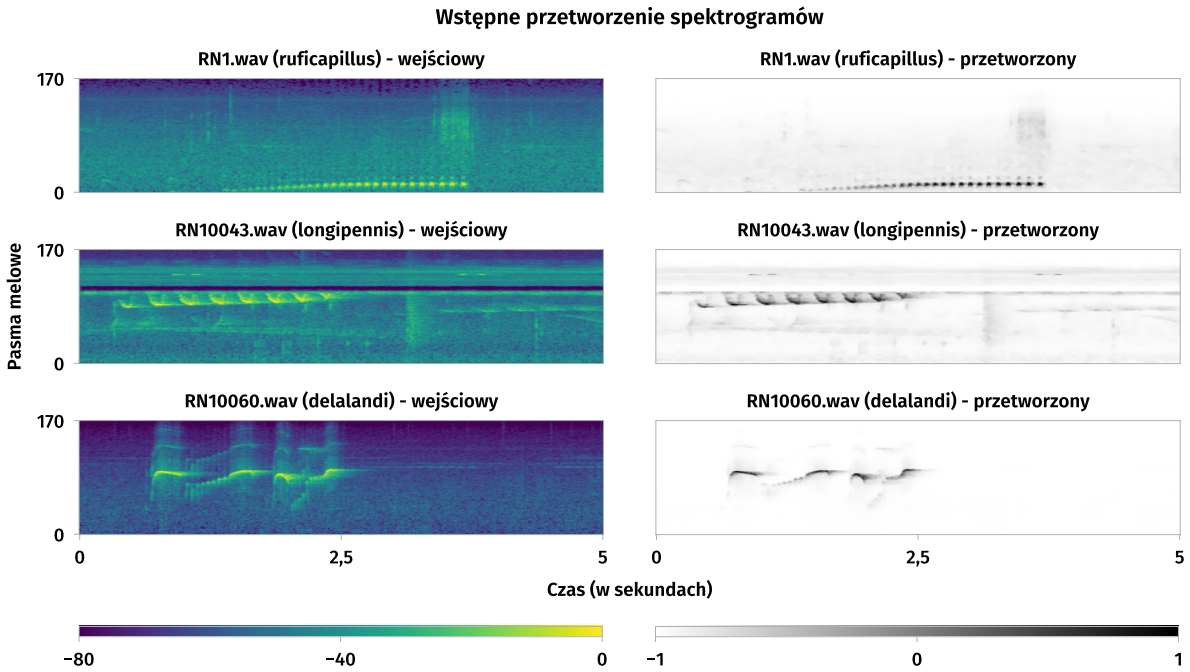
[1] Wyniki opisywanych w niniejszym rozdziale eksperymentów przeprowadzonych w ramach zadania konkursowego *BirdCLEF2016* zostały oryginalnie przedstawione w sprawozdaniu konferencyjnym „*Recognizing bird species in audio recordings using deep convolutional neural networks*" (Piczak, 2016).

Wyróżniającą się na tym tle inicjatywą jest organizowany cyklicznie przez CLEF[2] (*Conference and Labs of the Evaluation Forum*) projekt *BirdCLEF* **(Goëau et al., 2016)**. Pogłębił on tematykę rozpoznawania gatunków ptaków przez wprowadzenie zbiorów nagrań o skali bliższej rzeczywistym problemom – w edycji *BirdCLEF 2014* obejmowały one 501, a rok później już 999 gatunków ptaków śpiewających z Ameryki Południowej. Nadsyłane w ramach tego konkursu podejścia opierały się na drzewach decyzyjnych **(Lasseck, 2015)** i dopasowywaniu wzorców spektrogramów **(Lasseck, 2014)**, współczynnikach mel-cepstralnych (*MFCC*) **(Joly et al., 2014, 2015)**, atrybutach widmowych **(Leng et al., 2014)**, nienadzorowanym uczeniu atrybutów **(Stowell & Plumbley, 2014a,b; Stowell, 2015)** i głębokich sieciach neuronowych operujących na *MFCC* **(Koops et al., 2014)**. Ani razu w kontekście tym nie pojawiły się jednak architektury o charakterze splotowym. Jedyną pracą wykorzystującą sieci splotowe do rozpoznawania gatunków ptaków była publikacja Bransona et al. **(2014)**, która zajmowała się jednak zgoła odmiennym problemem klasyfikacji ptaków na zdjęciach. Z tego względu celem eksperymentów opisywanych w dalszej części rozdziału była weryfikacja, czy tego typu modele mogą być skutecznie wykorzystane również w zadaniu opierającym się na analizie nagrań dźwiękowych, jakim był problem postawiony w ramach *BirdCLEF 2016* **(Goëau et al., 2016)**.

6.2 Przetwarzanie danych

Zbiór *BirdCLEF 2016* składa się z trzech części. Część przeznaczona do uczenia zawiera 24 607 nagrań o bardzo zróżnicowanej długości (od mniej niż sekundy do blisko 45 minut). Każde z nagrań tego podzbioru zostało oznaczone pojedynczą etykietą odpowiadającą jednemu spośród 999 gatunków, który występował jako dominujący w danym nagraniu. Dodatkowo dla niektórych nagrań udostępniona została lista klas, które w subiektywnym odczuciu oceniających można dosyć wyraźnie usłyszeć w tle, lecz informacja ta nie była wykorzystywana w przeprowadzanych eksperymentach. Główna część podzbioru testowego pozostała niezmieniona w stosunku do edycji *BirdCLEF 2015* – stanowi ją 8596 nagrań, o długości od 1 sekundy do 11 minut, z pojedynczym dominującym gatunkiem występującym potencjalnie na tle innych klas. Nowością w tej edycji było rozszerzenie o podzbiór 925 nagrań tła akustycznego (w większości o długości 10 minut), które nie były rejestrowane w celu uchwycenia pojedynczego głównego gatunku, ale mogą zawierać dowolną liczbę ptaków śpiewających naprzemiennie lub jednocześnie w ramach jednego nagrania.

[2] http://www.clef-initiative.eu/.



Rys. 6.1: Przykłady wejściowych spektrogramów (po lewej) wraz z odpowiadającymi im spektrogramami po przetworzeniu mającym na celu wyizolowanie elementów pierwszego planu (po prawej).

Założeniem przeprowadzonych eksperymentów było ograniczenie rozważań wyłącznie do klasyfikatorów generujących pojedynczą etykietę, które byłyby odpowiednie dla rozpoznawania dominującego gatunku występującego w nagraniu. Proces wstępnego przetwarzania danych polegał na konwersji zbioru wejściowego do ujednoliconego formatu (*WAV*, częstotliwość próbkowania 44 100 Hz, rozdzielczość 16 bit, *mono*), na bazie którego wyliczane były spektrogramy w skali melowej z zastosowaniem biblioteki *librosa*[3] przy następujących wartościach parametrów: okno FFT o długości 2048 próbek i skoku 512, 200 pasm melowych wyznaczanych zgodnie z formułą HTK **(Young et al., 2002)** przy ograniczeniu górnego zakresu częstotliwości do 16 kHz. Następnie spektrogramy były ważone percepcyjnie i przekształcane przez operacje skalowania i progowania, odrzucenie 25 najniższych i 5 najwyższych pasm, a także wygładzenie za pomocą metody *total variation denoising* z wagą 0,1 (przy zastosowaniu algorytmu Chambolle'a **(2004)** zaimplementowanego w bibliotece *scikit-image*[4]), co miało na celu uwydatnienie elementów pierwszego planu. Przykład rezultatów takiego przetworzenia wejściowych nagrań zawiera rysunek 6.1.

80% nagrań należących do zbioru uczącego zostało następnie losowo wybrane do fazy uczenia sieci, a pozostałe 20% zachowane jako lokalny zbiór walidacyjny. Końcowym krokiem wstępnego przetwarzania danych było podzielenie pojedynczych nagrań na krótsze

[3] https://librosa.github.io/librosa/.
[4] http://scikit-image.org/.

fragmenty, odrzucając części, w których dominowała cisza. Wynikiem tego procesu utworzone zostało 85 712 segmentów uczących – każdy z nich z pojedynczą etykietą gatunku. W czasie właściwego uczenia modeli poszczególne segmenty były standaryzowane i odpowiednio przycinane lub dopełniane ciszą, tak aby zagwarantować stałe rozmiary wejścia (spektrogramy składające się z 430 ramek odpowiadających 5 sekundom nagrania) z losowym przesunięciem w czasie, co pozwalało na syntetyczne powiększenie zbioru danych uczących.

6.3 Architektury wykorzystanych modeli

W ramach przeprowadzanych eksperymentów przetestowane zostały liczne architektury opierające się częściowo o wcześniejsze doświadczenia autora z klasyfikacją dźwięków środowiskowych (rozdział 5) i pojawiające się nowe prace w zakresie rozpoznawania dźwięku za pomocą splotowych sieci neuronowych. Do konkursu *BirdCLEF 2016* zgłoszone zostały ostatecznie trzy modele schematycznie porównane w tabeli 6.1. Każdy z nich przetwarzał wejściowy spektrogram o rozmiarach 170 pasm × 430 ramek na wyjście typu *softmax* składające się z 999 neuronów, przedstawiające ocenę prawdopodobieństwa występowania danej klasy jako dominującego gatunku w analizowanym segmencie. Decyzje podjęte dla poszczególnych segmentów były w finalnym kroku uśredniane, generując zbiorcze predykcje na poziomie pojedynczych nagrań. Możliwość wystąpienia w tle wielu gatunków poza pojedynczą klasą dominującą została potraktowana w sposób uproszczony poprzez generowanie w ostatecznym zgłoszeniu konkursowym listy rankingowej predykcji tworzonych przez model, odcinanej poniżej progu prawdopodobieństwa wystąpienia równego 1%.

Zgłoszenie 1 – model A

Inspiracją dla tego modelu była praca Phana et al. **(2016)** rozważająca płytkie architektury z grupowaniem typu *1-Max*. Główną ideą stojącą za tego typu architekturami jest zastosowanie pojedynczej warstwy splotowej z odpowiednio dużą liczbą filtrów, która pozwalałyby na wyuczenie wyspecjalizowanych detektorów (wzorców) zdarzeń dźwiękowych. Do predykcji wykorzystywana jest następnie wyłącznie maksymalna wartość aktywacji uzyskiwana przez poszczególne detektory na przestrzeni całego nagrania (grupowanie *1-Max* w czasie).

Zgłoszony model składał się z pojedynczej warstwy splotowej z 600 filtrami prostokątnymi (170 × 5) z funkcją aktywacji *LeakyReLU* ($\alpha = 0{,}3$) i *dropoutem* na wejściu o prawdopodobieństwie 5%. Otrzymane przez tę warstwę wartości aktywacji grupowane były za pomocą funkcji *max* (okno grupowania rozmiaru 1 × 426) do sekwencji 600 wartości skalarnych reprezentujących maksymalną wartość aktywacji dla każdego filtra uzyskaną

Tab. 6.1: Architektury modeli zgłoszonych w ramach *BirdCLEF 2016*.

Model A	Model B	Model C
D: 5%		D: 5%
C-600: 170×5	C-80: 167×6	C-320: 167×10
LReLU	LReLU	LReLU
P: 1×426	P: 4×6, 1×3	P: 4×10, 1×5
D: 30%	C-160: 1×2	D: 5%
FC-3000	LReLU	C-640: 1×2
PReLU	P: 1×2, 1×2	LReLU
D: 30%	C-240: 1×2	P: 1×2, 1×2
SOFTMAX-999	LReLU	D: 5%
	P: 1×2, 1×2	C-960: 1×2
	C-320: 1×2	LReLU
	LReLU	P: 1×2, 1×2
	P: 1×2, 1×2	D: 5%
	D: 50%	C-1280: 1×2
	SOFTMAX-999	LReLU
		P: 1×2, 1×2
		D: 25%
		SOFTMAX-999

Oznaczenia: **(C-K: H×W)** – warstwa splotowa z K filtrami o wysokości H i szerokości W, **(P: H×W, S_H×S_W)** – *max-pooling* z oknem H×W i skokiem S_H×S_W, **(FC-N)** – warstwa w pełni połączona z N neuronami, **(D: p%)** – *dropout* z prawdopodobieństwem odrzucenia p%, **(LReLU)** – funkcja aktywacji typu *Leaky ReLU*, **(PReLU)** – funkcja aktywacji typu *Parametric ReLU*, **(SOFTMAX-N)** – warstwa wyjściowa typu *softmax* z N neuronami.

na przestrzeni całego segmentu wejściowego. Dalsze przetwarzanie zapewniała warstwa gęsto połączona zawierająca 3000 neuronów, rozszerzona o *dropout* z prawdopodobieństwem 30% i funkcję aktywacji typu *Parametric ReLU*. Przed wyjściową warstwą *softmax* (999 neuronów gęsto połączonych) została również zastosowana procedura *dropout* z prawdopodobieństwem 30%. Wszystkie wagi modelu przed uczeniem podlegały inicjalizacji według metody *He uniform* (wzór 2.34) przy początkowych obciążeniach warstwy splotowej ustalonych na 1.

Zgłoszenie 2 – model B

To zgłoszenie opierało się na modelu składającym się z czterech warstw splotowych z zastosowaniem niewielkiej regularyzacji według następującej specyfikacji:

- warstwa splotowa z 80 filtrami (167 × 6), regularyzacją typu L_1 o wadze 0,001 i funkcją aktywacji *LeakyReLU* ($\alpha = 0{,}3$),
- warstwa *max-pooling* o rozmiarze 4 × 6 i skoku 1 × 3,

- warstwa splotowa z 160 filtrami (1×2), regularyzacją typu L_2 o wadze 0,001 i funkcją aktywacji *LeakyReLU* ($\alpha = 0{,}3$),
- warstwa *max-pooling* o rozmiarze 1×2 i skoku 1×2,
- warstwa splotowa z 240 filtrami (1×2), regularyzacją typu L_2 o wadze 0,001 i funkcją aktywacji *LeakyReLU* ($\alpha = 0{,}3$),
- warstwa *max-pooling* o rozmiarze 1×2 i skoku 1×2,
- warstwa splotowa z 320 filtrami (1×2), regularyzacją typu L_2 o wadze 0,001 i funkcją aktywacji *LeakyReLU* ($\alpha = 0{,}3$),
- warstwa *max-pooling* o rozmiarze 1×2 i skoku 1×2,
- warstwa wyjściowa *softmax* (999 neuronów) z prawdopodobieństwem *dropoutu* 50% i regularyzacją typu L_2 o wadze 0,001.

Inicjalizacja wag przeprowadzana była w ten sam sposób jak dla modelu A. Zmniejszenie wysokości filtrów w stosunku do rozmiaru wejściowego spektrogramu pozwoliło natomiast na wprowadzenie niewielkiej niezmienności na przesunięcia w zakresie częstotliwości. Model B jest modelem w pełni splotowym – między ostatnią warstwą splotową a warstwą wyjścia nie ma dodatkowych warstw gęsto połączonych.

Zgłoszenie 3 – model C

Zgłoszenie to również bazowało na modelu w pełni splotowym składającym się z czterech warstw, takiej samej technice inicjalizacji wag, natomiast zwiększona w jego przypadku została istotnie liczba filtrów we wszystkich warstwach i ich szerokość w warstwie pierwszej. Dodatkowo, regularyzacja typu L_1/L_2 została zastąpiona przez *dropout* przed każdą warstwą według następującego schematu:

- warstwa splotowa z 320 filtrami (167×10), prawdopodobieństwem *dropoutu* 5% i funkcją aktywacji *LeakyReLU* ($\alpha = 0{,}3$),
- warstwa *max-pooling* o rozmiarze 4×10 i skoku 1×5,
- warstwa splotowa z 640 filtrami (1×2), prawdopodobieństwem *dropoutu* 5% i funkcją aktywacji *LeakyReLU* ($\alpha = 0{,}3$),
- warstwa *max-pooling* o rozmiarze 1×2 i skoku 1×2,
- warstwa splotowa z 960 filtrami (1×2), prawdopodobieństwem *dropoutu* 5% i funkcją aktywacji *LeakyReLU* ($\alpha = 0{,}3$),
- warstwa *max-pooling* o rozmiarze 1×2 i skoku 1×2,
- warstwa splotowa z 1280 filtrami (1×2), prawdopodobieństwem *dropoutu* 5% i funkcją aktywacji *LeakyReLU* ($\alpha = 0{,}3$),
- warstwa *max-pooling* o rozmiarze 1×2 i skoku 1×2,
- warstwa wyjściowa *softmax* (999 neuronów) z prawdopodobieństwem *dropoutu* 25%.

Zgłoszenie 4 – komitet modeli

Ostatnie ze zgłoszonych rozwiązań opierało się na prostym meta-modelu uśredniającym indywidualne predykcje wygenerowane przez poszczególne modele (A–C).

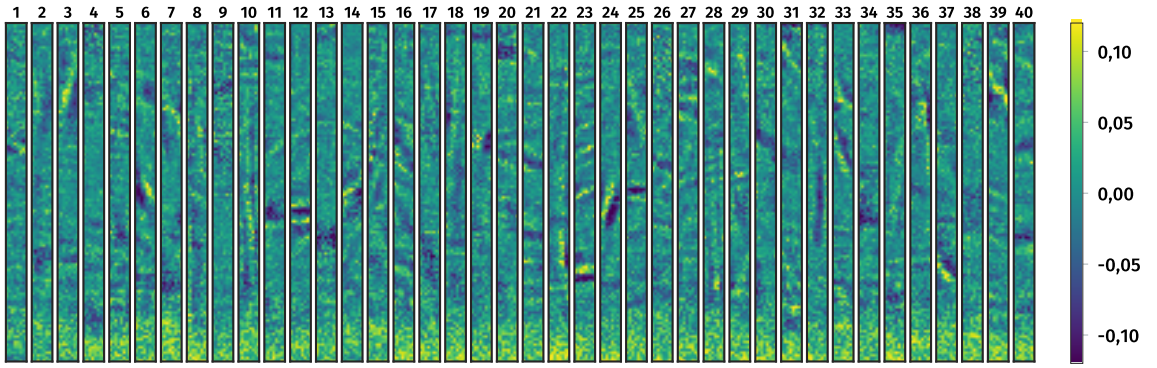
6.4 Procedura uczenia

Wszystkie modele uczone były poprzez minimalizację entropii krzyżowej za pomocą spadku stochastycznego gradientu z tempem uczenia 0,001 i *momentum* Nesterova o wartości 0,9. Każda z partii uczących składała się ze 100 segmentów. Walidacja przeprowadzana była lokalnie na wydzielonym podzbiorze (20% całości zbioru uczącego) poprzez losowe wybranie z niego w każdej epoce około 2500 nagrań, na których wyliczana była dokładność klasyfikacji. Wartość ta została przyjęta jako miara zastępcza dla oczekiwanej średniej precyzji (*mean average precision without background species, MAP_2*) - jednej z kategorii oceny przyjętej przez organizatorów we wcześniej edycji konkursu, *BirdCLEF 2015*.

Długość uczenia modeli zawierała się w zakresie 30–102 epok, co przy zastosowaniu pojedynczej karty GTX 980 Ti przekładało się czasowo na 30–60 godzin obliczeniowych w zależności od modelu. Wyniki końcowej walidacji uzyskane lokalnie przez poszczególne warianty architektur zawiera tabela 6.2, natomiast rysunek 6.2 ilustruje przykładowe filtry pierwszej warstwy splotowej uzyskane dla modelu C.

Tab. 6.2: Wyniki dokładności klasyfikacji uzyskane na lokalnym zbiorze walidacyjnym przez poszczególne warianty architektur.

	Model A	Model B	Model C
Dokładność na zbiorze walidacyjnym (miara zastępcza dla MAP_2)	45,1%	50,0%	49,5%



Rys. 6.2: Wizualizacja wybranych filtrów splotowych pierwszej warstwy modelu C.

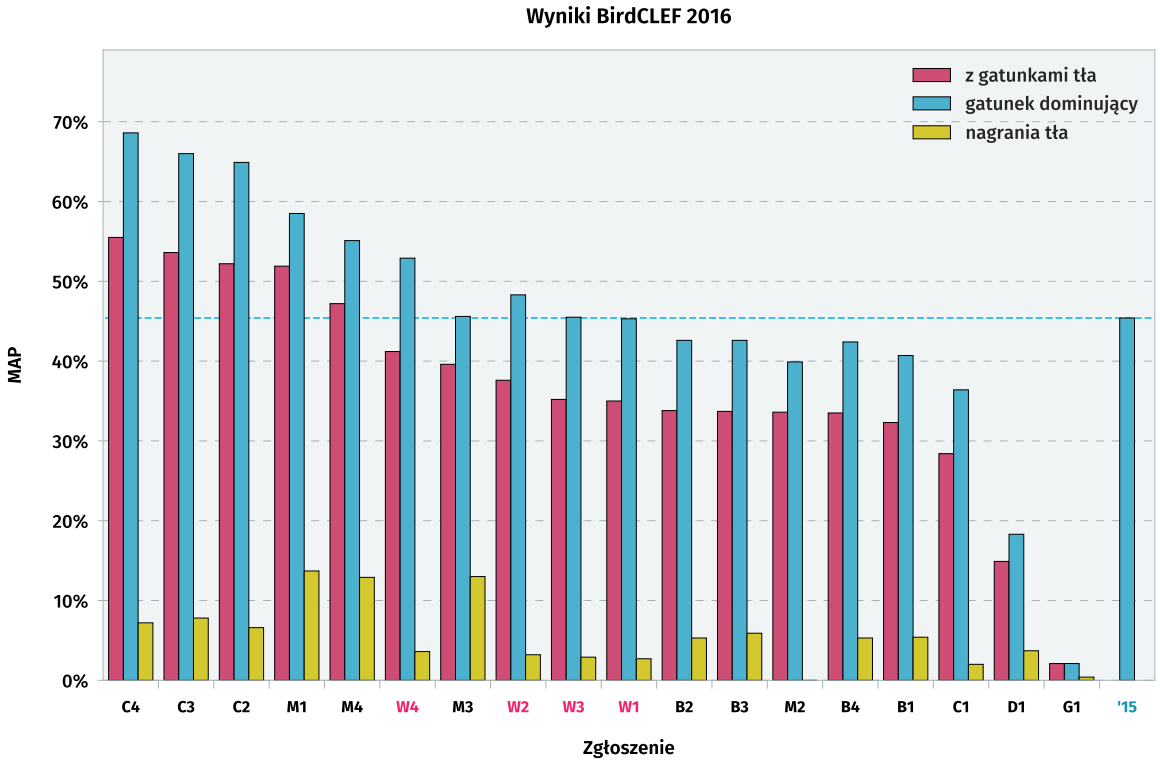
6.5 Omówienie uzyskanych wyników

Oficjalne wyniki *BirdCLEF 2016* zaprezentowane zostały w tabeli 6.3 i na rysunku 6.3. W tej edycji konkursu udział wzięło 6 zespołów zgłaszających łącznie 18 rozwiązań. Opisywane w niniejszym rozdziale zgłoszenia (zespół *WUT*) uplasowały się na 6, 8, 9 i 10 miejscu (ranking według pierwszej kolumny – *MAP* z uwzględnieniem gatunków w tle i nagrań tła). Analiza tych wyników i doświadczenia zdobyte w związku z udziałem w *BirdCLEF 2016* pozwalają na kilka spostrzeżeń.

- Przy blisko 1000 klasyfikowanych gatunków, *BirdCLEF 2016* przedstawia zadanie, którego złożoność jest wymagającym sprawdzianem dla systemów automatycznego rozpoznawania dźwięku. W tym kontekście rozwiązania oparte na splotowych sieciach neuronowych radzą sobie stosunkowo dobrze i wydają się obiecującym kierunkiem dla analizy danych bioakustycznych. Odnosząc się do porównywalnych wyników z poprzedniej edycji, uzyskanie poprawy klasyfikacji w stosunku do najlepszego wyniku z roku poprzedniego należałoby uznać za definitywny sukces. Adekwatność tego podejścia w bioakustyce potwierdza również fakt, że czołowe z przedstawianych rezultatów (zespół *Cube*) uzyskane zostały również za pomocą różnych wariantów splotowych sieci neuronowych **(Sprengel et al., 2016)**.
- Wyniki opisywanych modeli były dosyć zbliżone niezależnie od analizowanej architektury. Wydaje się więc, że większość typowych architektur splotowych dopełnionych poprawną procedurą uczenia i inicjalizacji parametrów powinna być w stanie nauczyć się w miarę dobrego przybliżenia funkcji klasyfikującej na podstawie dostarczonych danych. W wypadku tego zbioru nagrań niewielkie zmiany architektury mogą nie mieć tak dużego wpływu na końcową dokładność modelu jak sam proces przygotowania danych. Widać to również po pracy zespołu *Cube* **(Sprengel et al., 2016)**, gdzie najistotniejszym elementem rozwiązania wydaje się właśnie sama procedura wstępnego przetwarzania, w której autorzy wprowadzili dodatkowy krok odizolowywania fragmentów ze śpiewem ptaków i bez śpiewu (hałas tła) oraz zastosowali bardziej złożony sposób syntetycznego powiększania zbioru danych.
- Bardzo słabe wyniki uzyskane w kategorii nagrań tła potwierdzają, że prezentowane podejście jest mocno ukierunkowane na rozpoznawanie wyłącznie gatunku dominującego w nagraniu. Wynika to nie tylko z samego procesu uczenia na pojedynczej etykiecie, ale też zastosowanego wstępnego przetwarzania, które stara się wzmocnić elementy pierwszego planu kosztem usunięcia mniej wyraźnych odgłosów w tle. Wnioskiem płynącym z tego miejsca jest konieczność skupienia się bardziej na tym „co jest uczone” (segmentacja danych, przetwarzanie, etykietowanie, warstwy wejścia/wyjścia) niż „w jaki sposób” (wewnętrzna architektura sieci).

Tab. 6.3: Wyniki zadania konkursowego BirdCLEF 2016 (Goëau et al., 2016).

Zespół	Zgłoszenie	MAP		
		z gatunkami tła	gatunek dominujący	nagrania tła
Cube (C)	4	55,5%	68,6%	7,2%
Cube (C)	3	53,6%	66,0%	7,8%
Cube (C)	2	52,2%	64,9%	6,6%
MarioTsaBerlin (M)	1	51,9%	58,5%	13,7%
MarioTsaBerlin (M)	4	47,2%	55,1%	12,9%
WUT (W)	**4**	**41,2%**	**52,9%**	**3,6%**
MarioTsaBerlin (M)	3	39,6%	45,6%	13,0%
WUT (W)	**2**	**37,6%**	**48,3%**	**3,2%**
WUT (W)	**3**	**35,2%**	**45,5%**	**2,9%**
WUT (W)	**1**	**35,0%**	**45,3%**	**2,7%**
BME TMIT (B)	2	33,8%	42,6%	5,3%
BME TMIT (B)	3	33,7%	42,6%	5,9%
MarioTsaBerlin (M)	2	33,6%	39,9%	0,0%
BME TMIT (B)	4	33,5%	42,4%	5,3%
BME TMIT (B)	1	32,3%	40,7%	5,4%
Cube (C)	1	28,4%	36,4%	2,0%
DYNI LSIS (D)	1	14,9%	18,3%	3,7%
BIG (G)	1	2,1%	2,1%	0,4%
BirdCLEF 2015 ('15)	-	-	45,4%	-



Rys. 6.3: Wyniki zadania konkursowego BirdCLEF 2016 (Goëau et al., 2016).

- Obiecującą cechą wykorzystanego zbioru danych jest dosyć dobra zgodność między wynikami raportowanymi na utworzonym lokalnie podzbiorze walidacyjnym a oficjalnymi wynikami na zbiorze testowym ocenianym przez organizatorów. Oznacza to, że zbiór ten jest odpowiednio bogaty i jednocześnie jednorodny, co powinno ułatwiać prace nad uzyskaniem stabilnej poprawy wyników wraz z usprawnianiem wykorzystywanego rozwiązania.
- Metoda łączenia analizowanych modeli (zgłoszenie 4), mimo swojej prostoty, okazała się dosyć skutecznym sposobem poprawy uzyskiwanych wyników. Wskazuje to na potencjał związany z uśrednianiem generowanych predykcji. Jedną z takich technik rozważanych w czasie eksperymentów było progresywne zwiększanie prawdopodobieństwa *dropoutu* w trakcie uczenia, którego niestety nie udało się przetestować w finalnej formie ze względu na ograniczenia czasowe napotkane w końcowej fazie konkursu.

6.6 Wnioski

Eksperymenty przeprowadzone w ramach *BirdCLEF 2016* potwierdzają, że splotowe sieci neuronowe są obiecującym narzędziem do klasyfikacji gatunków ptaków w nagraniach dźwiękowych. Zwłaszcza biorąc pod uwagę poziom osiągnięty przez najlepsze ze zgłoszeń zespołu *Cube* (*MAP* bliski 70% przy 1000 gatunków), rozwiązanie takie można określić jako system zbliżający się do poziomu eksperckiego. O użyteczności splotowych sieci neuronowych w tej dziedzinie świadczy również fakt, że w kolejnej edycji konkursu, *BirdCLEF 2017*, wszystkie zgłoszone systemy **(Fazekas et al., 2017; Sevilla et al., 2017; Fritzler et al., 2017; Kahl et al., 2017)** należały do tej rodziny modeli, a części z nich udało się poprawić najlepszy wynik osiągnięty w *BirdCLEF 2016*.

Jednocześnie też widać jednak dużo miejsca na usprawnienia jeśli chodzi o rozpoznawanie ptaków w mniej wyraźnych nagraniach tła, które było dużym wyzwaniem dla wszystkich zgłoszonych modeli. Otwartym pytaniem pozostaje kwestia, czy słabe wyniki w tej kategorii wynikają z ograniczeń samych modeli, czy może też po części z niedoskonałości danych uczących, dla których zebranie jednoznacznych etykiet jest dużo trudniejsze i bardziej kosztowne niż w przypadku skupienia się wyłącznie na gatunku dominującym. Wydaje się, że może to być istotny kierunek, patrząc na zmiany w *BirdCLEF 2017*, w którym to właśnie wprowadzony nowy podzbiór nagrań tła pozwolił w niektórych przypadkach na poprawę średniej precyzji nawet dwu- lub trzykrotnie w porównaniu do poprzednika.

Rozdział 7

Szczegółowa analiza modeli splotowych w zadaniach klasyfikacji dźwięku

Eksperymenty omówione dotychczas w rozdziałach 5 i 6 miały charakter studiów przypadków, obrazujących zachowanie i możliwości wykorzystania konkretnych przykładów splotowych sieci neuronowych w zadaniach klasyfikacji dźwięków środowiskowych i gatunków ptaków. W ramach tych eksperymentów zestawione ze sobą zostały tylko pojedyncze warianty tego typu modeli. Chociaż liczba potencjalnych możliwości powstających przy doborze architektury i wartości hiperparametrów jest bardzo duża, to względnie długi czas uczenia głębokich sieci neuronowych powoduje, że szersze porównania w tym zakresie są dosyć rzadko spotykane w literaturze. Przykładem takiej pracy jest przeprowadzona przez Zhanga i Wallace'a (2017) analiza wrażliwości splotowych sieci neuronowych w zadaniach klasyfikacji zdań. Jej celem było rozróżnienie obszarów decyzyjnych, które mają duży wpływ na dokładność tworzonych modeli i tych, które są względnie nieistotne.

W kontekście rozumienia dźwięku tak rozległe analizy nie miały dotychczas miejsca – poszczególne prace skupiały się tylko na pojedynczych aspektach tego problemu. Przykładowo, Hershey et al. (2017) porównywali architektury *AlexNet*, *VGG*, *Inception* i *ResNet* w klasyfikacji przeprowadzanej na zbiorze *AudioSet*. Aytar et al. (2016), prezentując model *SoundNet*, dokonywali analizy ablacyjnej rozważającej zależność uzyskiwanej dokładności od liczby zastosowanych warstw. Z kolei Pons et al. (2016) oceniali wpływ różnych konfiguracji filtrów splotowych na skuteczność modeli w zadaniach klasyfikacji muzyki tanecznej.

Celem omówienia zawartego w dalszej części rozdziału jest uzupełnienie stanu wiedzy w zakresie splotowych sieci neuronowych klasyfikujących dźwięki o spostrzeżenia płynące z szeroko zakrojonej analizy wrażliwości tego typu modeli na ustalenia podejmowane przy doborze ich architektury i wartości hiperparametrów. Zgodnie z wiedzą autora jest to najszersze zestawienie tego typu przeprowadzone dotychczas w obszarze rozumienia dźwięków środowiskowych.

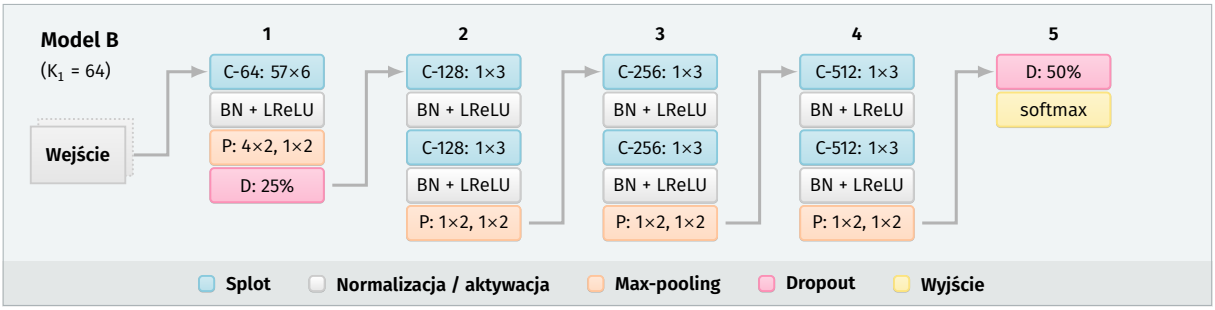
7.1 Metodyka eksperymentów

Najbardziej pożądanym sposobem przeprowadzenia takiej analizy byłoby porównanie wyników osiąganych przez szeroki zakres architektur przy różnych kombinacjach czynników (wartości hiperparametrów) mogących wpływać na dokładność modeli splotowych. Jednocześnie też takie porównanie powinno być przeprowadzone dla odpowiednio zróżnicowanego zestawu zbiorów danych, tak aby wykluczyć występowanie artefaktów związanych ze specyfiką doboru poszczególnych nagrań czy nawet konkretnych klas dźwiękowych. Rygorystyczne podejście do interpretacji wyników wymagałoby też, aby poszczególne przeliczenia powtarzane były wielokrotnie dla różnych wartości ziarna losowego, upewniając się, że pojawiające się różnice dokładności są rzeczywiście istotne, powtarzalne i wiążą się z wprowadzeniem konkretnej zmiany w procesie konstruowania i uczenia modelu. Niestety, przeprowadzenie eksperymentu w tak rozległej formule jest w praktyce niemożliwe ze względu na wymagany do tego celu zakres obliczeń. Z tego powodu konieczne staje się osiągnięcie kompromisu przez wprowadzenie pewnych odgórnych założeń ograniczających.

7.1.1 Rozważane architektury splotowe

Pierwszym przyjętym założeniem jest ograniczenie dalszej analizy do czterech przykładów architektur splotowych będących reprezentantami pewnych szerszych rodzin modeli.

Architektura bazowa (B)

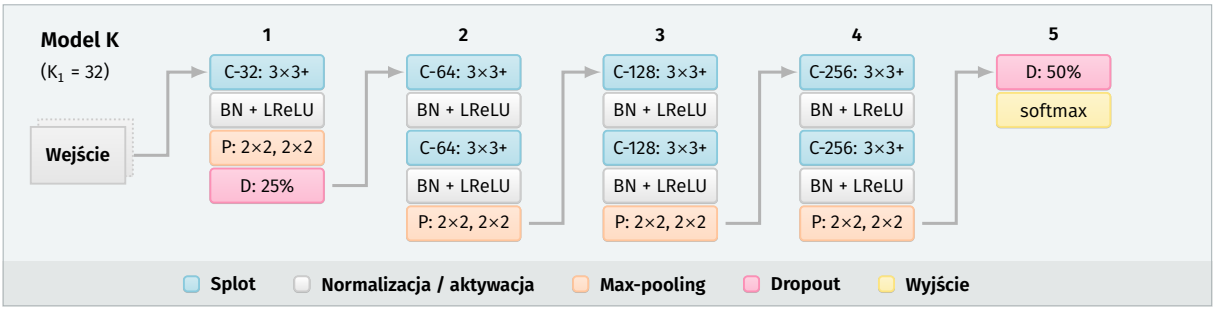


Rys. 7.1: Schemat architektury bazowej (B). Warstwy splotowe oznaczane są jako C-K: H×W, gdzie K jest liczbą filtrów, a H i W odpowiednio ich wysokością i szerokością. Skrót *BN + LReLU* oznacza normalizację partiami w połączeniu z zastosowaniem funkcji aktywacji typu *Leaky ReLU*. Dla warstwy grupowania typu *max-pooling* (P: H×W, S_H×S_W), H i W określają wysokość i szerokość okna grupowania, natomiast S_H i S_W – wysokość i szerokość przeskoku. Procentowe prawdopodobieństwo w przypadku *dropoutu* oznacza udział wejść odrzucanych.

Jako baza do dalszych porównań wybrany został opisywany w rozdziale 5 model *E2018*. Jest to sieć w pełni splotowa składająca się z 7 warstw o rosnącej wraz z głębokością liczbie filtrów, podwajanej dla każdego kolejnego modułu przedstawionego w formie kolumn na

rysunku 7.1. Charakterystyczną cechą tej architektury jest zastosowanie w pierwszej warstwie filtrów wertykalnych, które ograniczają się do przetwarzania krótkich wycinków czasowych spektrogramu, natomiast poddają analizie jednocześnie cały zakres częstotliwości. Powoduje to, że po pierwszej warstwie dalsze operacje splotowe dokonywane są wyłącznie względem czasu, stąd modele tego typu zaczęto w obszarze rozumienia dźwięku zbiorczo określać jako *1D CNN*.

Architektura z filtrami kwadratowymi (K)

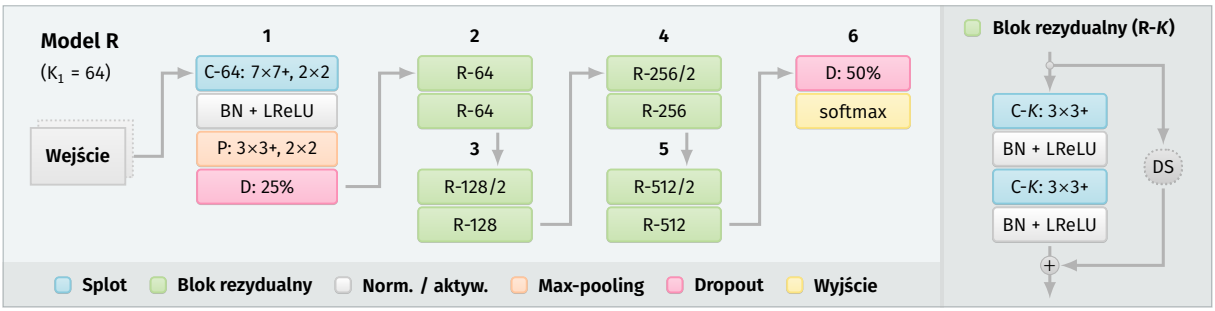


Rys. 7.2: Schemat architektury z filtrami kwadratowymi (K). Dodatkowe oznaczenie warstw splotowych przez znak „+” wskazuje na zastosowanie wypełnienia (dla filtrów 3×3 *padding* ma szerokość 1). Pozostałe oznaczenia jak na rysunku 7.1.

Architektura ta (rysunek 7.2) jest możliwie zbliżona do architektury bazowej, z tą różnicą, że warstwa pierwsza, podobnie jak wszystkie inne warstwy splotowe, wykorzystuje filtry kwadratowe o rozmiarze 3 × 3 z dodatkowym wypełnieniem (*padding*) pozwalającym zachować niezmienione rozmiary wynikowych map aktywacji. Sieć taka jest dosyć typowym przykładem rozwiązań stosowanych w obszarze klasyfikowania obrazów (*2D CNN*). Znaczące zwiększenie liczby operacji przeprowadzanych przez sieć w związku z operowaniem w dwóch wymiarach wymaga w tym wypadku redukcji liczby wykorzystywanych filtrów ($K_1 = 32$). Umożliwia to utrzymanie złożoności obliczeniowej na poziomie adekwatnym dla masowego charakteru przeprowadzanych eksperymentów.

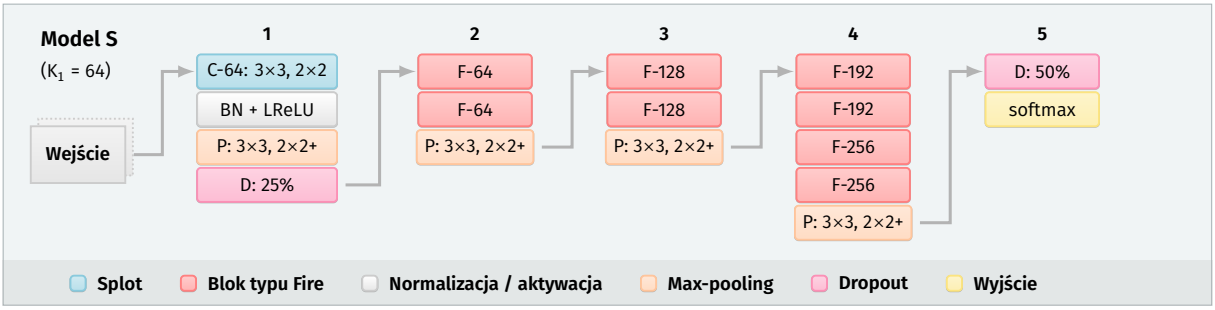
Architektura z połączeniami rezydualnymi (R)

Reprezentantem trzeciej grupy jest przedstawiona na rysunku 7.3 architektura będąca minimalną modyfikacją modelu *ResNet-18* **(He et al., 2016a)**. Bazuje ona na 17 warstwach splotowych zgrupowanych w bloki uczące się reprezentacji rezydualnej (omawianej w podrozdziale 2.3.6). Redukcja wymiarowości następuje w przypadku tego modelu przez wprowadzenie skoku wybranych warstw splotowych, gdyż w architekturze tej, poza pierwszym modułem, nie występują warstwy grupowania *max-pooling*.



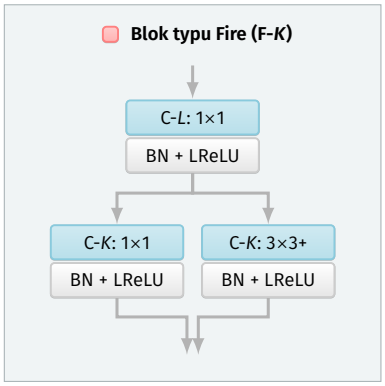
Rys. 7.3: Schemat architektury z połączeniami rezydualnymi (R). Każdy z bloków rezydualnych R-*K* składa się z dwóch warstw splotowych C-*K*: 3×3+ połączonych według schematu zaprezentowanego w prawej części rysunku. W przypadku bloków oznaczanych jako R-*K*/2, pierwsza warstwa splotowa operuje ze skokiem 2×2, natomiast na części niepoddawanej przetwarzaniu dokonywana jest redukcja wymiarowości za pomocą elementu *DS* (*downsampling*). Element *DS* jest warstwą typu C-*K*: 1×1, 2×2 połączoną z normalizacją (BN) bez nieliniowości. Wszystkie warstwy splotowe w tej architekturze nie zawierają wag obciążających. *Pooling* w pierwszym module stosuje dodatkowe wypełnienie zerami (*padding*) o szerokości 1. Pozostałe oznaczenia są zgodne z rysunkami 7.1 i 7.2.

Architektura typu SqueezeNet (S)



Rys. 7.4: Schemat architektury typu *SqueezeNet* (S). Bloki typu *Fire* są złożeniami warstw według schematu z rysunku 7.5. Znak „+” przy skoku warstw grupowania wskazuje na zaokrąglanie wymiarów wyjściowych w górę. Pozostałe oznaczenia bez zmian.

Czwartym przykładem jest zilustrowana na rysunku 7.4 architektura wzorująca się na modelu *SqueezeNet* (Iandola et al., 2016) w wersji *v1.1*[1]. *SqueezeNet* został zaproponowany jako model pozwalający utrzymać poziom dokładności sieci *AlexNet* klasyfikującej zbiór *ImageNet* przy 50-krotnej redukcji liczby parametrów. Koncepcja ta opiera się na następujących po sobie przewężeniach (*bottlenecks*) w postaci zestawień warstw splotowych 1 × 1 i 3 × 3+, określanych przez autorów jako bloki *Fire*. Schemat takiego bloku przedstawiony został na rysunku 7.5.



Rys. 7.5: Schemat bloku typu *Fire* z K filtrami, $L = K/4$.

[1] https://github.com/DeepScale/SqueezeNet.

Blok *Fire* składa się z wejściowej warstwy splotowej 1×1 o niewielkiej liczbie filtrów ($L = K/4$) z normalizacją i nieliniowością, która ma na celu zredukować efektywną liczbę kanałów (wartości dla każdego piksela) na kolejnych etapach przetwarzania, zabezpieczając przed nadmiernym rozrostem parametryzacji sieci. Wyjście z tego reduktora jest następnie niezależnie wprowadzane do warstw C-K: 1×1 i C-K: 3×3+. Odpowiedzi tych warstw podlegają złączeniu przez konkatenację, tworząc wyjście całego bloku.

Dla podstawowego wariantu architektury rozważanego w eksperymentach ($K_1 = 64$), liczba filtrów w warstwie ściskającej (*squeeze layer*) bloku *Fire* (F-K) wynosi $L = K/4$. W przypadku eksperymentów oceniających wpływ liczby filtrów na dokładność klasyfikacji (podrozdział 7.2.3) rozważane są tylko zmiany wartości K, natomiast wartości L dla poszczególnych bloków pozostają stałe, zgodnie z wariantem podstawowym.

7.1.2 Zbiory danych

Drugim założeniem co do kształtu eksperymentu jest wykorzystanie trzech zbiorów danych do oceny dokładności klasyfikacji uzyskiwanej przez poszczególne warianty modeli splotowych. Zbiorami tymi są *ESC-50* i *UrbanSound8K*, opisywane w rozdziałach 4 i 5, oraz zestaw nagrań udostępnianych w ramach *BirdCLEF 2016* (rozdział 6).

Pełny zbiór *BirdCLEF 2016* jest nie tylko znacząco większy od *ESC-50* i *UrbanSound8K*, ale też składa się z nagrań o bardzo zróżnicowanym czasie trwania. Aby umożliwić przeprowadzenie eksperymentów w założonej skali, zbiór ten został pomniejszony na potrzeby sporządzanego porównania przez wybranie nagrań o długości zawierającej się w zakresie 10–30 sekund i ograniczenie liczby klas do 100 najczęściej występujących gatunków. W ten sposób powstał podzbiór 2635 nagrań, który na potrzeby tego rozdziału będzie określany jako *Minibirds*.

Minibirds został podzielony na 5 możliwie równych podzbiorów walidacyjnych. W przypadku zbiorów *ESC-50* i *UrbanSound8K* przyjęte zostały podziały oryginalnie zaproponowane w artykułach **(Piczak, 2015b; Salamon et al., 2014)** – odpowiednio na 5 i 10 podzbiorów. Zamiast pełnej walidacji krzyżowej, w eksperymencie przeprowadzona została ocena na pojedynczym podzbiorze testowym (podzbiory o numerach 5, 10, 5 – kolejno dla *ESC-50*, *UrbanSound8K* i *Minibirds*). Jeden podzbiór (analogicznie o numerze 4, 9, 4) zachowany został również do celów dodatkowej walidacji, ale ze względu na zbliżony pod względem jakościowym charakter uzyskiwanych na nim wyników, nie będą one szczegółowo raportowane w dalszych zestawieniach. Uczenie modeli przeprowadzane było na pozostałych podzbiorach (1–3, 1–8, 1–3).

7.1.3 Aspekty poddawane analizie

Trzecim założeniem jest ustalenie następującej listy czynników mogących wpływać na dokładność klasyfikacji, które będą przeanalizowane w ramach eksperymentu:

- kształt filtrów pierwszej warstwy wykorzystywanych w modelu typu *1D CNN*,
- kształt filtrów pierwszej warstwy wykorzystywanych w modelu typu *2D CNN*,
- szerokość modelu, rozumiana jako liczba filtrów splotowych w poszczególnych warstwach,
- głębokość modelu (liczba warstw),
- zastosowanie *dropoutu*,
- zastosowanie normalizacji partiami,
- wykorzystana funkcja aktywacji neuronów,
- ustawienia procesu uczenia (optymalizatora),
- siła wprowadzanej regularyzacji,
- metoda inicjalizacji wag,
- rozmiar partii uczącej,
- długość wejściowego segmentu spektrogramu,
- rozdzielczość wejściowego spektrogramu.

Istotnym ograniczeniem wprowadzonym w analizie jest rozpatrywanie wpływu poszczególnych czynników przy założeniu *ceteris paribus*, utrzymując pozostałe hiperparametry przy domyślnie ustalonych wartościach. Chociaż takie uproszczenie jest nieodzowne dla celów przeprowadzanego eksperymentu, to należy mieć na uwadze, że brak wpływu wyizolowanych zmian czynników nie gwarantuje równocześnie ich nieistotności w potencjalnych kombinacjach.

Bazą wyjściową przy ustalaniu domyślnych wartości hiperparametrów jest kształt eksperymentu przyjęty dla uczenia modelu *E2018*, opisany w podrozdziale 5.2.4. Spektrogramy generowane są według ujednoliconych ustawień: częstotliwość próbkowania – 44 100 Hz, długość okna FFT – 2205 próbek (50 ms), przeskok – 882 próbki (20 ms), liczba pasm melowych – 60, limit górnej częstotliwości – 16 000 Hz. Długość segmentów uczących wynosi 125 ramek (2,5 s) dla zbiorów *ESC-50* i *UrbanSound8K* oraz 500 ramek (10 s) dla przykładów z *Minibirds*. Uczenie przeprowadzane jest przez minimalizację entropii krzyżowej za pomocą metody spadku stochastycznego gradientu (tempo uczenia $\eta = 0{,}01$, *momentum* Nesterova o wartości 0,9, regularyzacja L_2 o sile $\alpha = 0{,}001$) przy rozmiarze partii uczącej wynoszącym 128 przykładów. Inicjalizacja wag domyślnie dokonywana jest według reguły *LeCun uniform* (podrozdział 2.2.3). Czas uczenia na zbiorach *ESC-50* i *UrbanSound8K* pozostaje na poziomie niezmienionym względem eksperymentów opisanych w podrozdziale 5.2.4 – 20 epok odpowiadających w przybliżeniu 5000 partii uczących.

Dla zbioru *Minibirds* jest to natomiast 50 epok równoważnych około 8500 krokom aktualizującym wagi.

Miarą wykorzystywaną przy raportowaniu wyników w dalszej części rozdziału jest dokładność uzyskiwana na zbiorze testowym[2] uśredniona dla dziesięciu końcowych epok uczenia. Dla każdego ze zbiorów przeprowadzane jest uczenie zarówno w wariancie wykorzystującym dodatkowy kanał różnicowy (oznaczenie *dwa kanały*), jak i ograniczające się tylko do spektrogramu bazowego (*jeden kanał*). Wyniki przedstawione w komórkach zestawienia stanowią średnią dla pięciu powtórzeń z różnymi inicjalizacjami ziarna losowego. Odpowiadające im wartości błędu standardowego zawarte zostały w załączniku A.

7.2 Wyniki analizy wrażliwości dla poszczególnych czynników

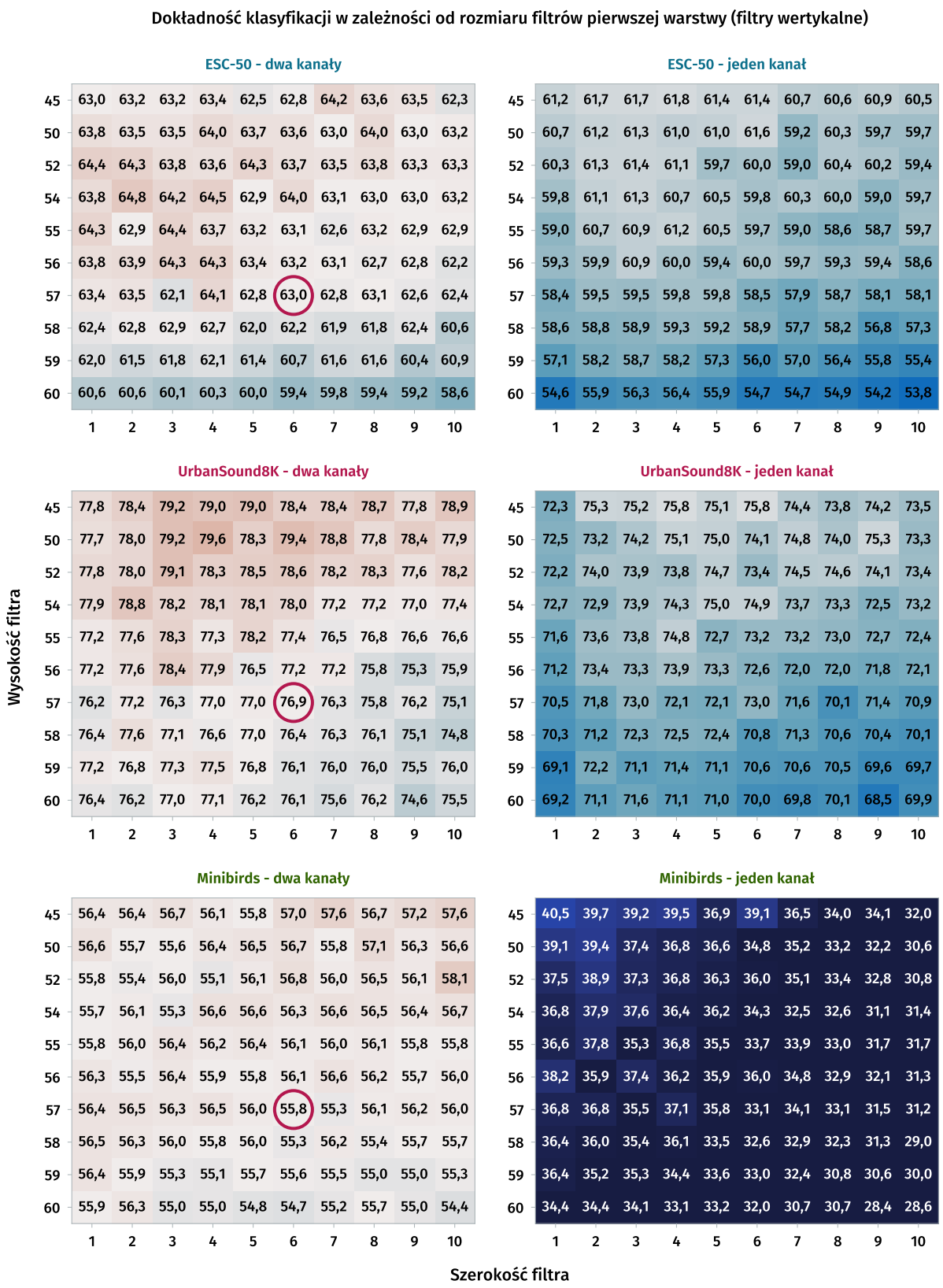
7.2.1 Kształt filtrów splotowych w modelu typu *1D CNN*

Rysunek 7.6 zawiera zestawienie wyników klasyfikacji w zależności od przyjętego kształtu filtrów pierwszej warstwy splotowej dla modelu opierającego się na architekturze bazowej (B). Utrzymane jest założenie, że przetwarzanie ma charakter typowy dla modelu *1D CNN*, dlatego zmiany wysokości filtra wertykalnego są kompensowane przez odpowiednie modyfikacje okna grupowania *max-pooling*, tak aby na wyjściu pierwszej warstwy dla każdego filtra uzyskać reprezentację jednowymiarową. Oznacza to, że mniejsze wysokości filtra wprowadzają niezmienność wobec coraz większych przesunięć w dziedzinie częstotliwości.

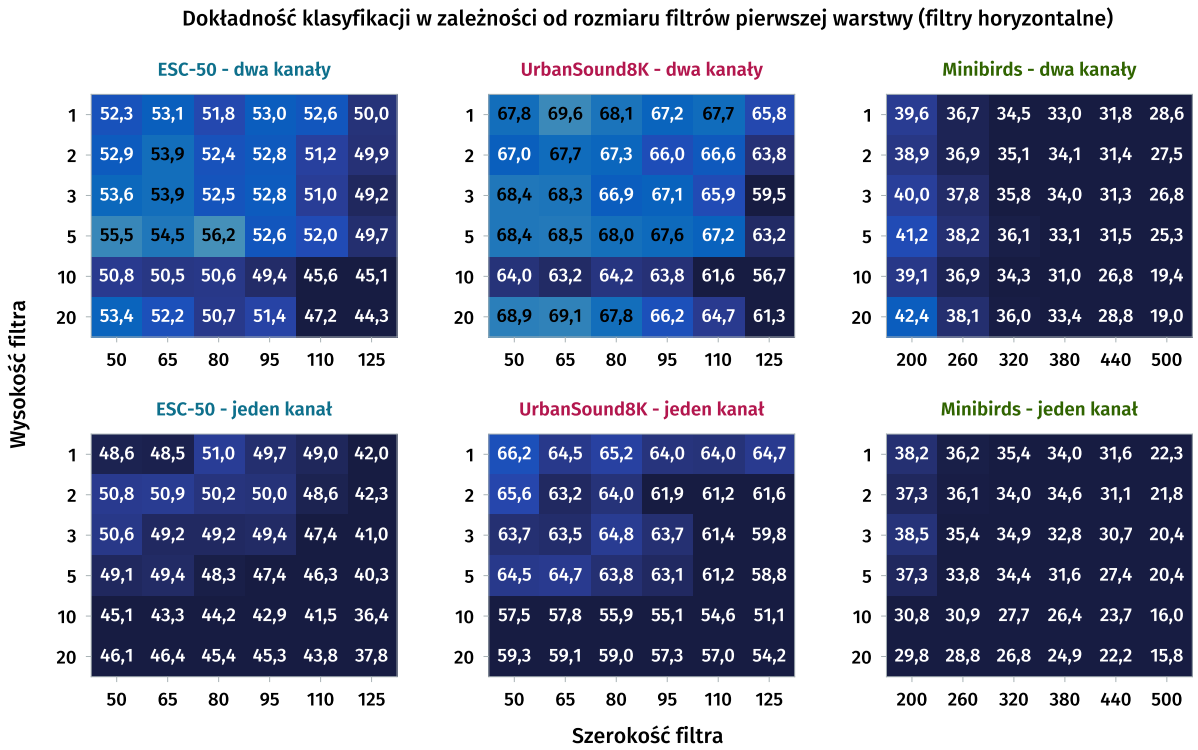
Analiza rysunku 7.6 pozwala na poczynienie kilku spostrzeżeń. Po pierwsze, modele uczone z użyciem spektrogramów różnicowych uzyskują wyniki istotnie lepsze od swoich odpowiedników opierających się tylko na spektrogramach bazowych. Zwłaszcza w przypadku zbioru *Minibirds* występująca dysproporcja zdaje się dyskwalifikować wykorzystanie modeli jednokanałowych.

Drugim interesującym aspektem jest wzrost dokładności klasyfikacji pojawiający się przy zmniejszaniu wysokości zastosowanych filtrów. Pokazuje to, że ważnym elementem polepszania zdolności generalizacyjnych modelu jest jego uodpornienie na przesunięcia w dziedzinie częstotliwości. Efekt ten jest o tyle intrygujący, że nie są to tylko niewielkie przesunięcia. Bardzo dobre wyniki uzyskują również konfiguracje opierające się na filtrach o wysokości 45 pasm (niezmienność względem przesunięć obejmujących ¼ zakresu pasm). Oznacza to, że dla klasyfikowania dźwięków środowiskowych ważniejsze okazują

[2] Predykcje końcowe uzyskiwane są zgodnie ze schematem agregacji typu *probability voting* (podrozdział 5.2.1).



Rys. 7.6: Wyniki klasyfikacji dla architektury bazowej (B) w zależności od rozmiaru filtrów pierwszej warstwy splotowej. Komórki zawierają wartości wyrażone w procentach. Wyniki dla domyślnego ustawienia filtrów zakreślone zostały ciemnoróżowym kolorem.



Rys. 7.7: Wyniki klasyfikacji dla architektury bazowej (B) w zależności od rozmiaru filtrów pierwszej warstwy splotowej w wariancie horyzontalnym (filtry operujące na dużych fragmentach czasowych, dalszy splot względem częstotliwości). Komórki zawierają wartości wyrażone w procentach.

się być konkretne wzorce w dziedzinie częstotliwości (struktury harmoniczne) niż ich dokładne umiejscowienie. Silniejsza poprawa występująca dla zbioru *UrbanSound8K* może w tym aspekcie również wynikać z mniejszej liczby klas, która powoduje, że pozytywny efekt takiego uogólnienia przeważa nad utratą możliwości dokładnego rozróżnienia dokonywanego w zakresie częstotliwości. Istotnie mniejsza skala efektu dla *Minibirds* sugeruje też, że na zależność tę może mocno wpływać typ analizowanych dźwięków.

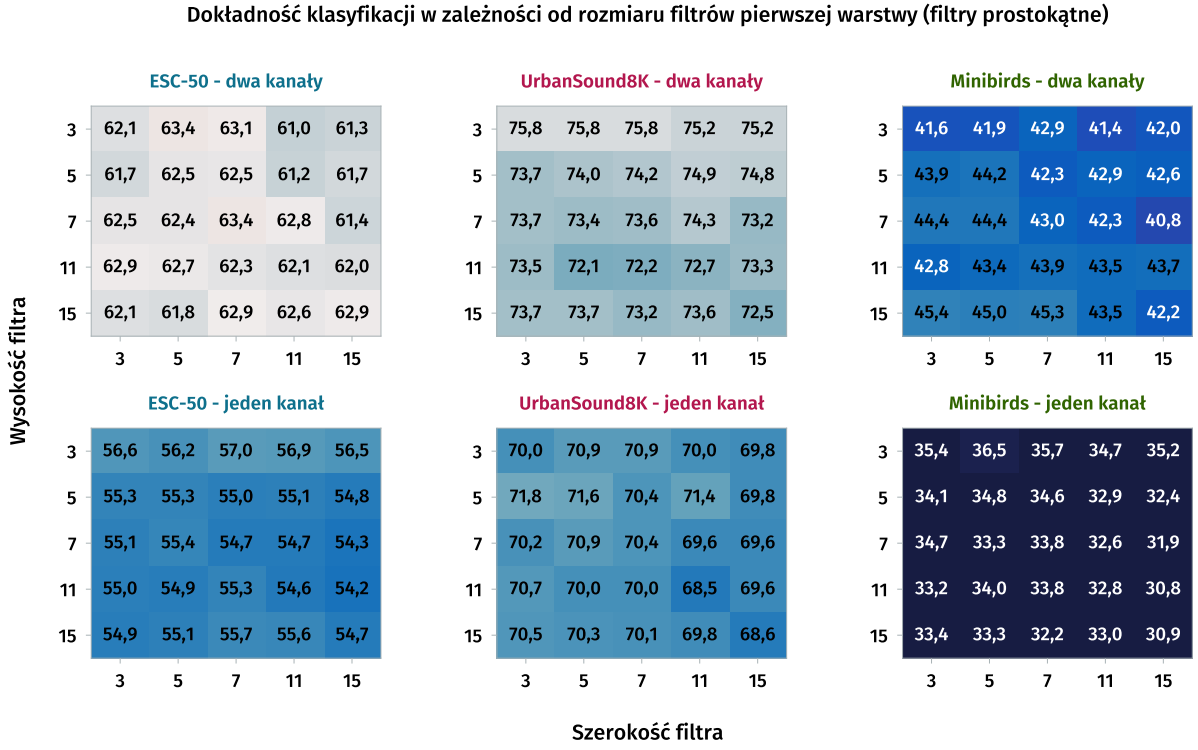
Rozważając na podstawie uzyskanych wyników aspekt regulowania szerokości filtra splotowego, nie wydaje się on być tak istotnym kryterium doboru modelu. Wnioskiem płynącym z analizy rysunku 7.6 mogłaby być mimo to sugestia co do zastosowania filtrów nie tylko o mniejszej wysokości, ale również nieco węższych. Przykładowo, redukcja rozmiaru z bazowego 57×6 do 50×4 jest w przeprowadzonym eksperymencie posunięciem jednoznacznie pozytywnym, chociaż statystycznie istotnym tylko dla zbioru *UrbanSound8K*.

Rysunek 7.7 uwidacznia z kolei, że w przypadku modeli typu *1D CNN* odwrócenie kierunku filtrów z wertykalnych na horyzontalne jest złym posunięciem[3]. Tego typu postępowanie miałoby na celu wymuszenie uczenia reprezentacji bazującej na złożeniach pewnych wzorców rytmicznych. Dalsze warstwy splotowe w takim wariancie operują już wyłącznie

[3] W wariancie horyzontalnym założono również odpowiednie odwrócenie operacji grupowania i kierunku filtrów dalszych warstw, tak aby po pierwszej warstwie uzyskać mapy aktywacji o rozmiarze 60×1.

w dziedzinie częstotliwości. Chociaż wyniki eksperymentów Ponsa et al. (2016) wskazują, że podejście takie może znaleźć uzasadnienie przy analizie nagrań muzyki tanecznej, to dużo mniejsza wyrazistość rytmiczna nagrań środowiskowych powoduje, że modele z filtrami horyzontalnymi są ewidentnie nieskuteczne dla tego typu danych.

7.2.2 Kształt filtrów splotowych w modelu typu *2D CNN*



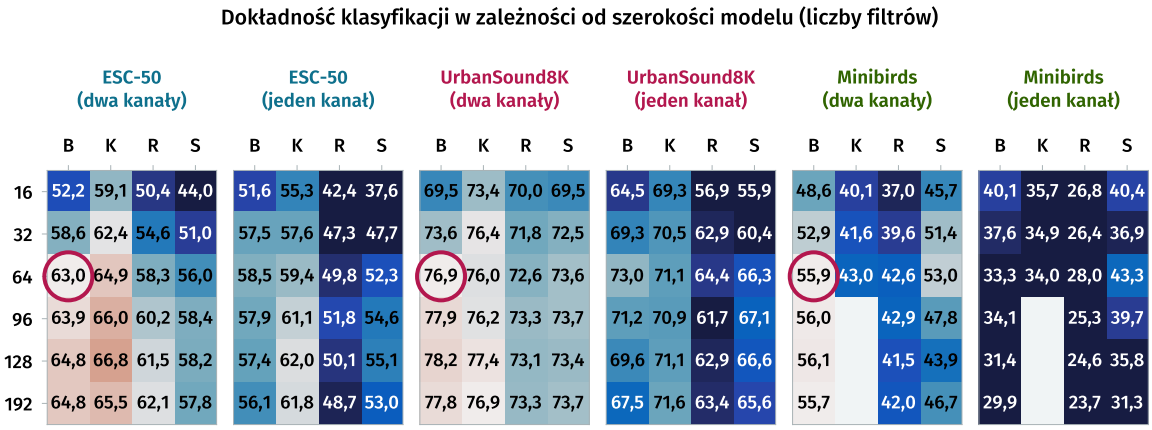
Rys. 7.8: Wyniki klasyfikacji dla architektury (K) z filtrami kwadratowymi (prostokątnymi) w zależności od rozmiaru filtrów pierwszej warstwy splotowej. *Padding* pierwszej warstwy ma wartość połowy rozmiaru filtra (po zaokrągleniu w dół). Komórki zawierają wartości wyrażone w procentach. Niewielkie różnice numeryczne w stosunku do wyników uzyskanych przez wariant 3×3 w pozostałych zestawieniach wynikają z wykorzystania do tej części przeliczeń karty graficznej o odmiennej architekturze.

Dla typowej architektury zaadaptowanej z obszaru przetwarzania obrazów (*2D CNN*), wykorzystującej na wejściu małe filtry kwadratowe (lub bliskie im prostokątne), kwestia doboru konkretnego rozmiaru tych filtrów okazuje się być względnie mało istotna. Różnice wyników na rysunku 7.8 w większości przypadków nie wykraczają poza wahania z tytułu losowej inicjalizacji parametrów (rys. A.3). Jedyną konkluzywną informację niosą rezultaty zastosowania podejścia dwukanałowego dla *UrbanSound8K*, sugerując raczej ograniczanie wysokości filtrów do możliwie małych wartości. Pod tym kątem klasyczne podejście wzorowane na *VGGNet* (filtry 3×3) wydaje się bezpiecznym rozwiązaniem.

W ogólnym rozrachunku modele bazujące na architekturze typu „K” uzyskują dokładność klasyfikacji nieznacznie słabszą niż odpowiedniki z filtrami wertykalnymi (rys. 7.6).

Duży relatywny spadek dokładności widoczny jest natomiast dla zbioru *Minibirds* w wariancie dwukanałowym. Dodatkowo na niekorzyść tej grupy modeli przemawia ich zwiększone zapotrzebowanie pamięciowo-obliczeniowe.

7.2.3 Szerokość modelu



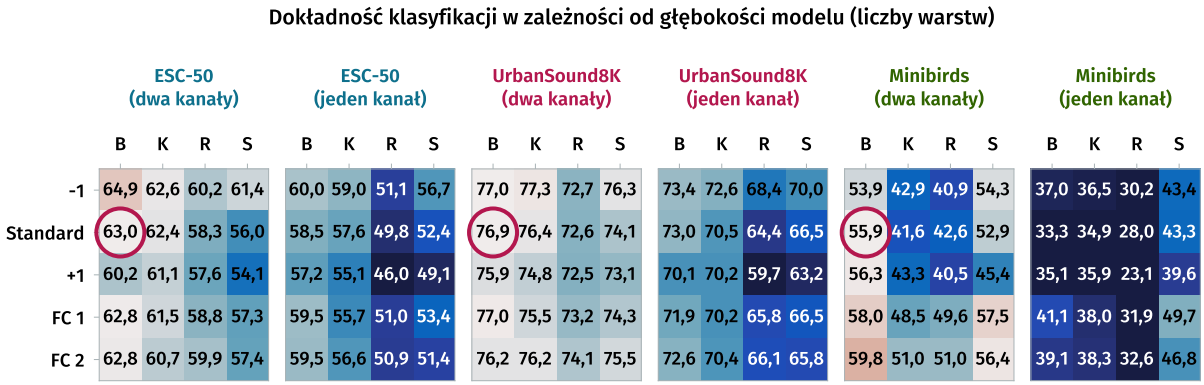
Rys. 7.9: Wyniki klasyfikacji dla rozważanych architektur (kolumny „B”, „K”, „R”, „S”) w zależności od przyjętej bazowej liczby filtrów (szerokości modelu). Podstawowe warianty architektur „B”, „R”, „S” opierają się na liczbie filtrów $K_1 = 64$. W związku z większą złożonością obliczeniową architektury „K”, przyjętą dla niej bazą jest wartość $K_1 = 32$. Również z tego powodu dla architektury tej niemożliwe było uczenie na zbiorze *Minibirds* dla wartości $K_1 > 64$. Komórki macierzy przedstawiają dokładność klasyfikacji wyrażoną w procentach. Wyniki dla modelu bazowego odpowiadającego wariantowi *E2018* zakreślone zostały kolorem ciemnoróżowym.

Nieco odmienne wnioski w kwestii architektur typu „K” przedstawia zawarte na rysunku 7.9 porównanie szerokości modeli. W zestawieniu tym analizowany jest efekt zmian hiperparametru K_1, który określa liczbę filtrów w pierwszej warstwie splotowej, a także pośrednio wpływa na szerokość kolejnych warstw, ustalaną jako odgórnie zdefiniowana dla danej architektury wielokrotność wartości K_1.

Zwiększanie liczby filtrów dla architektury typu „K” pozwala w przypadku zbioru *ESC-50* (dwa kanały) na uzyskanie wyniku istotnie lepszego od pozostałych modeli. Proces ten jest jednak bardzo kosztowny jeśli chodzi o przyrost wymagań czasowych i pamięciowych uczenia, dlatego nie jest możliwe jego zastosowanie przy sporo dłuższych segmentach zbioru *Minibirds*. Występowanie takiej poprawy dokładności wskazuje natomiast na jeden istotny fakt, jakim jest niewystarczająca ekspresyjność modelu dla domyślnych ustawień K_1, częściowo związana też z zastosowaniem wczesnego *dropoutu* (szerzej analizowanego w podrozdziale 7.2.5). Mimo ograniczonych rozmiarów zbiorów uczących okazuje się, że efekt ten jest na tyle silny, że nawet wielokrotne zwiększanie wartości K_1 nie grozi wystąpieniem problemu przeuczenia.

Podobnie jak w poprzednich zestawieniach, dokładność klasyfikacji uzyskiwana dla wariantów modeli opierających się na pojedynczym kanale jest gorsza niż dla odpowiedników dwukanałowych. Lekko zaskakujące są natomiast względnie słabe rezultaty zastosowania architektury z połączeniami rezydualnymi (R) i wzorowanej na modelu *SqueezeNet* (S) – podejść, które bardzo dobrze sprawdzają się w obszarze przetwarzania obrazów.

7.2.4 Głębokość modelu



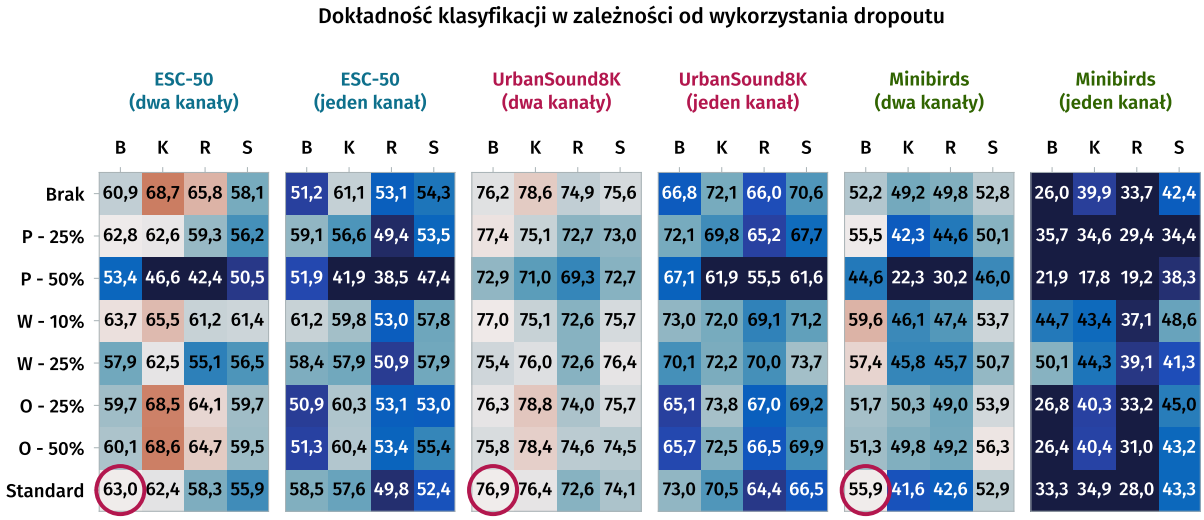
Rys. 7.10: Wyniki klasyfikacji dla rozważanych architektur (kolumny „B”, „K”, „R”, „S”) w zależności od modyfikacji liczby wykorzystanych warstw (głębokości modelu). Wiersz „Standard” zawiera wyniki dla architektur utworzonych dokładnie według schematów na rysunkach 7.1–7.4. Dla ustawień „-1” i „+1” oceniane są modele, w których została odpowiednio usunięta lub dodana jedna warstwa dla każdego z modułów 2–4 (rys. 7.1 i 7.2), jeden blok rezydualny dla modułów 2–5 (rys. 7.3) lub jeden blok *Fire* dla modułów 2–4 (rys. 7.4). W wariantach „FC 1” i „FC 2” ustawienie „Standard” rozszerzane jest o odpowiednio jedną lub dwie dodatkowe warstwy w pełni połączone (2048 neuronów, normalizacja partiami, aktywacja *Leaky ReLU*) przed warstwą *softmax*. Komórki macierzy przedstawiają dokładność klasyfikacji wyrażoną w procentach. Wyniki dla modelu bazowego odpowiadającego wariantowi *E2018* zakreślone zostały kolorem ciemnoróżowym.

Interpretacja zmian powstających wskutek modyfikacji liczby warstw modelu (rys. 7.10) zależna jest od konkretnego zbioru uczącego. W przypadku *ESC-50* ogólną tendencją jest uzyskiwanie lepszych wyników przez modele płytsze (wariant „-1”). Dodawanie kolejnych warstw do i tak już głębokich architektur przy niezmienionych parametrach uczenia (liczbie iteracji) pogarsza dokładność klasyfikacji uzyskiwaną na tym zbiorze niezależnie od typu obranej architektury. Podobny kierunek można zauważyć dla *UrbanSound8K*, chociaż efekt ten jest w jego przypadku nieco mniej wyraźny.

Osobliwie pod tym względem zachowuje się natomiast zbiór *Minibirds*. O ile liczba warstw splotowych nie odgrywa kluczowej roli w poprawie zdolności klasyfikacyjnych, to już bardzo istotnym elementem okazuje się być dołożenie przed warstwą wyjścia typu *softmax* dodatkowych warstw w pełni połączonych (warianty „FC 1”, „FC 2”). Pokazuje to, że odmienny charakter nagrań tego zbioru (dłuższy czas trwania, mniejszy stosunek treści związanej z daną klasą do całości tła akustycznego) domaga się dodatkowych możliwości

przetwarzania, których nie zapewniają modele czysto splotowe. Pewną drogą poradzenia sobie z tym niedociągnięciem mogłoby również być zastosowanie odpowiedniej wstępnej obróbki danych (na wzór rozdziału 6), tak aby uzyskać krótsze segmenty uczące zawierające więcej interesujących dla klasyfikatora treści. W celu zachowania możliwie dużej uniformizacji eksperymentu, procedury wstępnego przetwarzania nie były jednak dostosowywane pod kątem poszczególnych zbiorów.

7.2.5 Prawdopodobieństwo *dropoutu*



Rys. 7.11: Wyniki klasyfikacji dla rozważanych architektur (kolumny „B”, „K”, „R”, „S”) w zależności od ustawienia *dropoutu*. Wiersz „Standard” zakłada *dropout* z prawdopodobieństwem 25% po pierwszej warstwie i 50% przed warstwą wyjścia. Wiersz „Brak” przewiduje całkowitą rezygnację z *dropoutu*. W pozostałych wierszach oceniane są modele z *dropoutem* o określonym prawdopodobieństwie zastosowanym tylko po pierwszej warstwie (P), po wszystkich warstwach modelu (W) lub tylko przed ostatnią (O). Komórki macierzy przedstawiają dokładność klasyfikacji wyrażoną w procentach. Wyniki dla modelu bazowego odpowiadającego wariantowi *E2018* zakreślone zostały kolorem ciemnoróżowym.

Analiza zestawienia na rysunku 7.11 pokazuje, że kwestia doboru odpowiednich ustawień *dropoutu* jest bardzo ważnym obszarem decyzyjnym, mogącym w drastyczny sposób wpłynąć na poziom uzyskiwanych wyników. Co więcej, najlepsze podejście w tym obszarze zależy w dużej mierze od typu architektury.

W przypadku architektury bazowej wariant domyślny (*dropout* 25% bezpośrednio po pierwszej warstwie splotowej i 50% przed warstwą wyjścia) okazuje się być skuteczny. Rezygnacja z *dropoutu* po pierwszej warstwie („O – 50%”) wiąże się z jednoznacznym pogorszeniem wyników, więc jest to element ważny dla modeli z filtrami wertykalnymi. Zachowanie to jest o tyle kłopotliwe, że w przypadku pozostałych typów architektur decyzja o wprowadzeniu *dropoutu* na tak wczesnym etapie okazuje się mieć zgoła odwrotny efekt, będąc tym sposobem bardzo niefortunnym posunięciem. Model oparty na małych filtrach

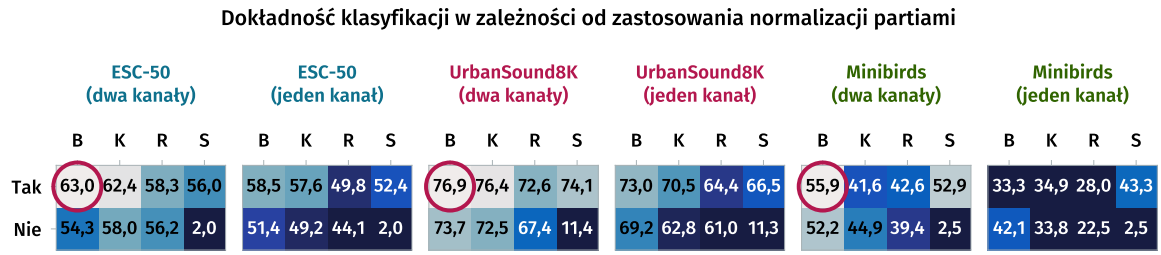
kwadratowych po całkowitym pozbawieniu *dropoutu* staje się bezkonkurencyjny w klasyfikowaniu zbiorów *ESC-50* i *UrbanSound8K*.

Różnicę w zachowaniu między architekturami można interpretować dwuaspektowo. Po pierwsze, wykorzystanie filtrów wertykalnych o bardzo dużych rozmiarach (macierz kilkuset wag) może zwiększać skłonność sieci do skupiania się już w pierwszej warstwie na bardzo specyficznych wzorcach, co pogarsza jej zdolność generalizacji. Zastosowanie częściowego *dropoutu* na tym etapie pozwala zniwelować ten efekt. W przypadku pozostałych architektur opierających się na małych filtrach wejściowych (zwykle 3×3, co najwyżej 7×7) ryzyko takie jest dużo mniejsze. Z drugiej strony, zwłaszcza po uwzględnieniu lepszego funkcjonowania modeli z większą liczbą filtrów (rys. 7.9), można domniemywać, że dla tych architektur użycie tak wczesnego *dropoutu* zbyt mocno ogranicza ich efektywną szerokość.

Ciekawym zachowaniem cechują się modele bazowe (B) uczone na zbiorze *Minibirds*. W ich przypadku wyraźnie pojawia się problem przeuczenia, któremu udaje się zaradzić przez wprowadzenie *dropoutu* na poziomie wszystkich warstw modelu („W – 10%”, „W – 25%”). Poprawa z tego tytułu jest zauważalna nie tylko dla architektury dwukanałowej, ale też w przypadku ograniczenia się wyłącznie do bazowych spektrogramów. Model typu „B” w wariancie „W – 25%” jest jedną z nielicznych kombinacji hiperparametrów, dla której udaje się przekroczyć poziom 50% przy klasyfikacji jednokanałowego *Minibirds*.

Ogólnym wnioskiem płynącym z analizy zachowania modeli przy różnych ustawieniach *dropoutu* jest bardzo silne związanie tego obszaru z konkretnymi typami architektur. Powoduje to, że formułę podobnych eksperymentów można by w przyszłości poprawić nie traktując tego aspektu jako jednego z analizowanych hiperparametrów, ale na starcie dobierając dla każdej architektury jego optymalną charakterystykę, na trwale związaną z daną rodziną modeli. W kontekście przeprowadzonego badania sugeruje to, że należałoby raczej z dużą ostrożnością podchodzić do ewentualnych wniosków wynikających z ich bezpośredniego porównywania między sobą. Można się spodziewać, że wartości absolutne raportowane dla architektur typu „K” i „R” są z tego powodu zaniżone w stosunku do możliwości występujących przy optymalnym doborze hiperparametrów i należałoby to uwzględnić przy tworzeniu ewentualnego rankingu.

7.2.6 Wykorzystanie normalizacji partiami



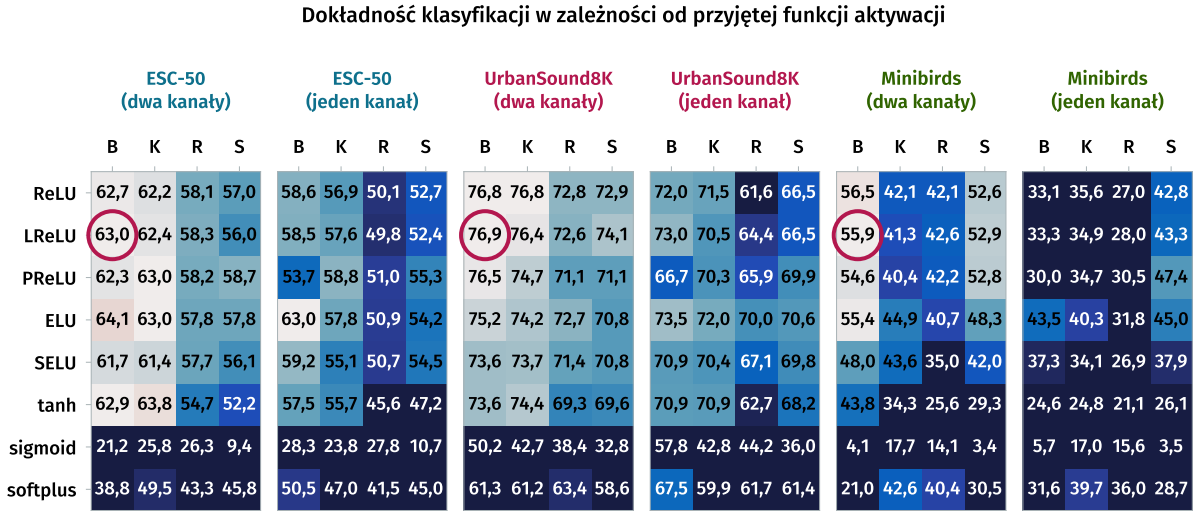
Rys. 7.12: Wyniki klasyfikacji dla rozważanych architektur (kolumny „B”, „K”, „R”, „S”) w zależności od wykorzystania normalizacji partiami. Komórki macierzy przedstawiają dokładność klasyfikacji wyrażoną w procentach. Wyniki dla modelu bazowego odpowiadającego wariantowi *E2018* zakreślone zostały kolorem ciemnoróżowym.

Wprowadzenie normalizacji partiami (*batch normalization*) jest ważnym aspektem uczenia modeli o dużej głębokości, co w większości potwierdzają wyniki zestawienia na rysunku 7.12. W przypadku architektury typu *SqueezeNet* (S) brak tego rozwiązania kompletnie uniemożliwia efektywne uczenie. Dla pozostałych modeli dysproporcje nie są tak drastyczne, ale spadek dokładności jest zdecydowanie zauważalny.

Ewenementem pod tym względem są wyniki zbioru *Minibirds* dla architektur „K” (dwa kanały) i „B” (jeden kanał). Istotnie pozytywna rola zastosowania normalizacji partiami wydaje się mimo to dosyć oczywista. Technika ta umożliwia szybsze uczenie modeli o większej głębokości. Na przykładzie eksperymentów z architekturą *E2015* (rozdział 5) widać, że uzyskanie podobnych poziomów dokładności jest co prawda możliwe również z jej pominięciem, ale wymaga zastosowania płytszej architektury w połączeniu z dużo dłuższym czasem uczenia.

7.2.7 Zastosowana funkcja aktywacji neuronów

Rysunek 7.13 zawiera porównanie modeli wykorzystujących różne funkcje aktywacji neuronów opisane w podrozdziale 2.2.2. W zakresie zestawienia ze sobą alternatywnych form aktywacji typu *ReLU* (*Leaky ReLU*, *Parametric ReLU*, *Exponential Linear Unit*, *Scaled Exponential Linear Unit*) trudno jest wskazać na różnice o charakterze systematycznym, wykraczające poza wahania związane z losową inicjalizacją. Pozostanie przy najczęściej wykorzystywanych wariantach typu *ReLU* lub *Leaky ReLU* wydaje się całkowicie uzasadnione. Przewaga nowszych sformułowań tej funkcji, podobnie jak w literaturze, nie może zostać w definitywny sposób potwierdzona w przeprowadzonym eksperymencie. Pewne obiecujące charakterystyki wykazuje funkcja aktywacji typu *ELU*, zwłaszcza przy przetwarzaniu jednokanałowym, ale zachowanie to jest zbyt niejednorodne, aby móc wyciągnąć bardziej uogólnione wnioski.



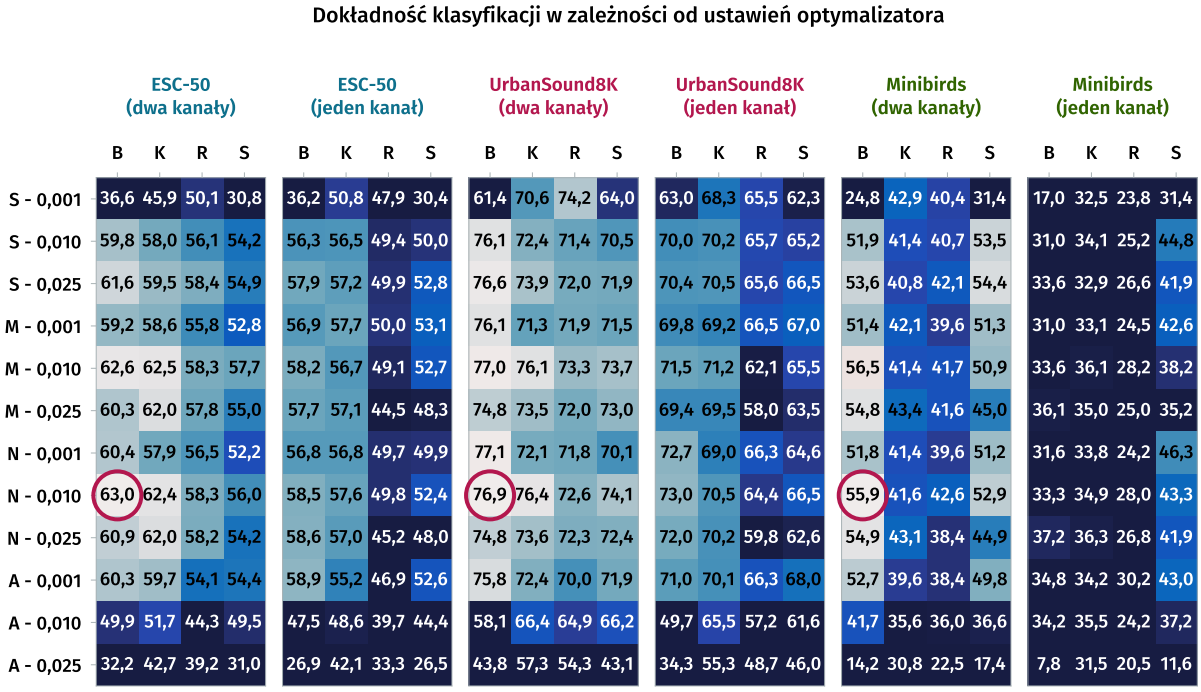
Rys. 7.13: Wyniki klasyfikacji dla rozważanych architektur (kolumny „B”, „K”, „R”, „S”) w zależności od przyjętej funkcji aktywacji neuronów. Analizowanymi funkcjami są kolejno: *ReLU*, *Leaky ReLU*, *ReLU parametryczny*, *ELU* (*Exponential Linear Unit*), *SELU* (*Scaled Exponential Linear Unit*), *tangens hiperboliczny*, funkcja *logistyczna* (*sigmoidalna*) i *softplus*. Komórki macierzy przedstawiają dokładność klasyfikacji wyrażoną w procentach. Wyniki dla modelu bazowego odpowiadającego wariantowi *E2018* zakreślone zostały kolorem ciemnoróżowym.

Ewidentnie nieskuteczne okazuje się natomiast wykorzystanie funkcji logistycznej (sigmoidalnej). Chociaż tangens hiperboliczny został w obecnie stosowanych modelach praktycznie całkowicie wyparty przez warianty *ReLU*, to trzeba przyznać, że w eksperymencie nie ustępuje on dokładnością aż w tak znaczącym stopniu, jak można by oczekiwać po jego małej popularności. Trochę dziwić może też słaba charakterystyka modeli opierających się na funkcji *softplus*, będącej gładką aproksymacją *ReLU*. Prawdopodobną przyczyną dużych różnic między nimi, mimo podobnego kształtu funkcji, jest niezerowa wartość aktywacji uzyskiwana dla sum wejścia równych zero (podobnie jak w przypadku funkcji logistycznej).

7.2.8 Ustawienia procesu uczenia (optymalizatora)

Rysunek 7.14 zawiera zestawienie porównawcze modeli splotowych z różnymi wartościami tempa uczenia i wariantem wykorzystanej metody optymalizacji. Chociaż uzyskane wyniki pokazują, że zakres ustawień przynoszących akceptowalne rezultaty jest dosyć szeroki, to na ich podstawie można dokonać kilku spostrzeżeń co do potencjalnych pułapek przy doborze tej grupy hiperparametrów.

Po pierwsze, dość oczywistym wnioskiem jest, że w przypadku stałego budżetu obliczeniowego dobór odpowiedniego tempa uczenia jest krytyczny dla podstawowej metody spadku stochastycznego gradientu. W wariancie „S – 0,001” uczenie jest zbyt wolne, by mogło się zakończyć w miejscu wysycenia przed upłynięciem limitu iteracji. Rozszerzenie



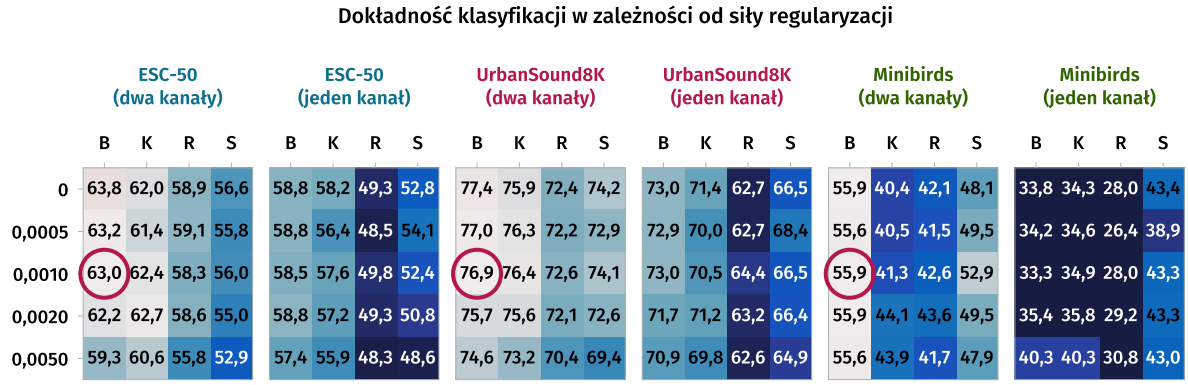
Rys. 7.14: Wyniki klasyfikacji dla rozważanych architektur (kolumny „B”, „K”, „R”, „S”) w zależności od ustawień optymalizatora. Poszczególne wiersze przedstawiają modele uczone z różnym tempem uczenia (drugi człon oznaczenia wariantu) za pomocą podstawowej metody spadku stochastycznego gradientu (S), metody rozszerzonej o wykorzystanie zwykłego *momentum* o wartości 0,9 (M) lub *momentum* Nesterova (N), a także metody *Adam* (A). Komórki macierzy przedstawiają dokładność klasyfikacji wyrażoną w procentach. Wyniki dla modelu bazowego odpowiadającego wariantowi *E2018* zakreślone zostały kolorem ciemnoróżowym.

tej metody o wykorzystanie *momentum* pozwala w dużej części uniknąć negatywnych konsekwencji zbyt małego tempa uczenia, ale trzeba też podkreślić, że dostrojenie tej wartości jest dalej ważne.

Kwestia ustalenia odpowiedniego tempa uczenia zazwyczaj przedstawiana jest jako problem specyficznie związany z kombinacją konkretnego zbioru danych (zadania) i danej architektury, stąd częstą sugestią jest po prostu weryfikacja na kilku potencjalnych wartościach. Domyślnie wykorzystane ustawienie (spadek gradientu stochastycznego z *momentum* Nesterova o wartości 0,9 i tempie uczenia $\eta = 0{,}01$) może być stosunkowo obiecującym punktem startowym do tego typu przeszukiwań.

Warto też w tym miejscu podkreślić, że optymalizacja za pomocą metody *Adam*, mimo adaptacyjnego charakteru, również wymaga określenia odpowiedniego tempa uczenia (zazwyczaj mniejszego niż w przypadku spadku stochastycznego gradientu). Jest to spostrzeżenie o tyle istotne, że można się spotkać z obiegowymi opiniami, jakoby dobór tego hiperparametru w takim wypadku miał znaczenie marginalne, czemu dobitnie zaprzeczają wyniki przedstawianego zestawienia.

7.2.9 Siła regularyzacji



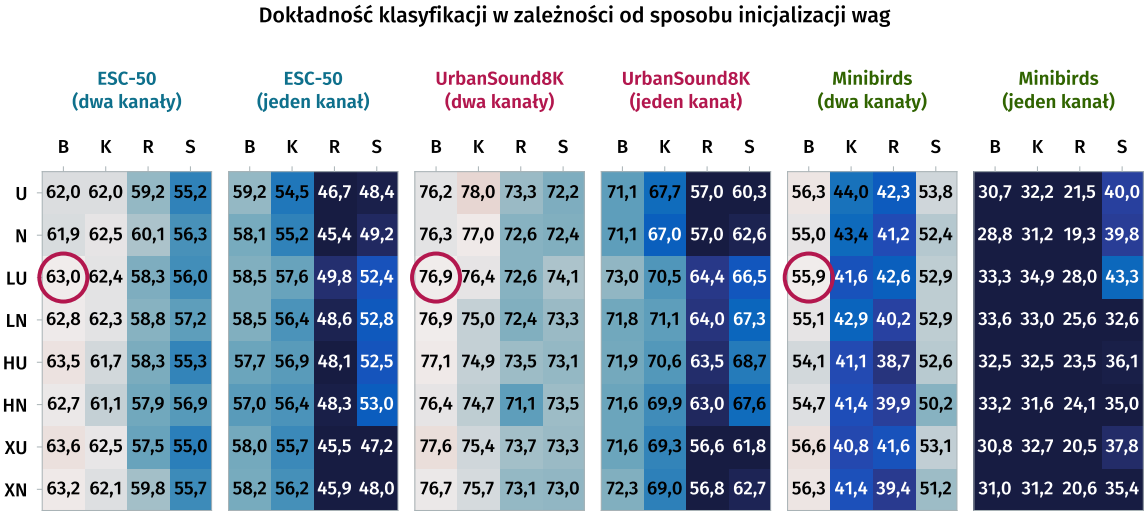
Rys. 7.15: Wyniki klasyfikacji dla rozważanych architektur (kolumny „B”, „K”, „R”, „S”) w zależności od przyjętej siły regularyzacji wag typu L_2. Komórki macierzy przedstawiają dokładność klasyfikacji wyrażoną w procentach. Wyniki dla modelu bazowego odpowiadającego wariantowi *E2018* zakreślone zostały kolorem ciemnoróżowym.

Kolejnym aspektem związanym z optymalizacją wag splotowych sieci neuronowych jest stopień narzucanej odgórnie regularyzacji. W przeprowadzonym eksperymencie rozważana jest tylko regularyzacja typu L_2 (podrozdział 2.2.4).

Rysunek 7.15 zawiera zestawienie wyników dla siły regularyzacji w zakresie od $\alpha = 0$ (brak regularyzacji) do $\alpha = 0{,}005$ (najsilniejszy przyjęty stopień). Poza ustawieniem ze skrajnie wysoką wartością, wpływ regularyzacji na dokładność klasyfikacji można uznać za kosmetyczny. Przez pryzmat oceny czysto ilościowej równie dobrym wyjściem wydaje się być całkowita z niej rezygnacja. Ewentualną korzyścią z utrzymania niewielkiego stopnia regularyzacji jest natomiast możliwość otrzymania filtrów splotowych lepiej nadających się do późniejszej interpretacji wizualnej.

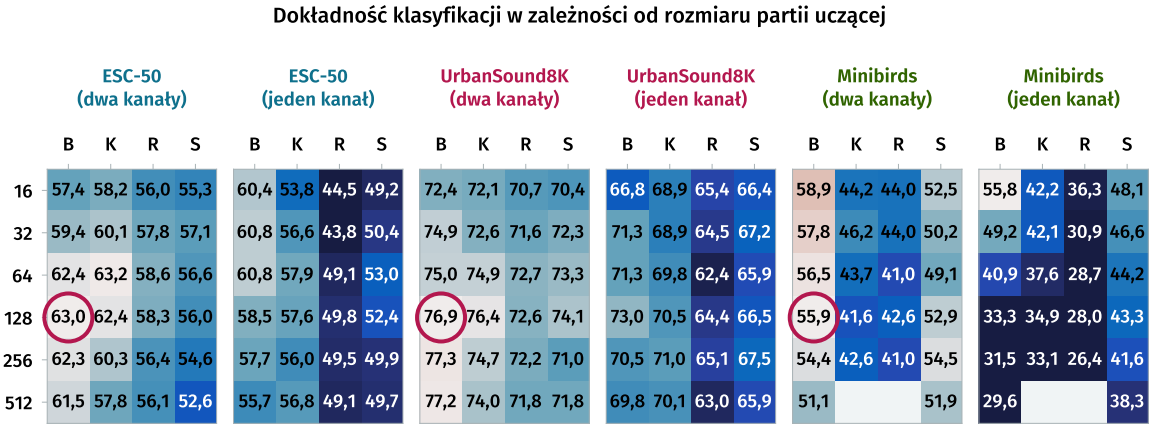
7.2.10 Metoda inicjalizacji wag

Podobne wnioski przynosi rozważenie wariantów inicjalizacji wag modelu przedstawione na rysunku 7.16. Niewielkie różnice w uzyskiwanych wynikach pokazują, że nie jest to obecnie obszar kluczowy przy uczeniu modeli splotowych. O ile sytuacja ta wyglądała zgoła inaczej jeszcze w niedalekiej przeszłości opisywanej w podrozdziale 2.2.3, to wprowadzenie procedury normalizacji partiami pozwoliło właściwie zapomnieć o tym aspekcie doboru hiperparametrów. Z tego powodu poprzestanie na domyślnych sposobach inicjalizacji proponowanych przez najpopularniejsze biblioteki wydaje się całkowicie wystarczające. Właściwie jedynym wyjątkiem w tym zakresie jest istotnie lepsze zachowanie architektury „K” w klasyfikowaniu dwukanałowych spektrogramów zbioru *UrbanSound8K* przy inicjalizacji rozkładem jednostajnym. Jest to jednak zachowanie na tyle incydentalne, że trudno z niego wywodzić ogólniejszą regułę.



Rys. 7.16: Wyniki klasyfikacji dla rozważanych architektur (kolumny „B”, „K”, „R”, „S”) w zależności od przyjętej metody inicjalizacji wag: rozkładem jednostajnym $\mathcal{U}(-0{,}01, 0{,}01)$ (U), rozkładem normalnym $\mathcal{N}(0, 0{,}0001)$ (N), metodą *LeCun uniform* (LU) lub *normal* (LN), *He uniform* (HU) lub *normal* (HN) oraz *Glorot/Xavier uniform* (XU) i *normal* (XN). Komórki macierzy przedstawiają dokładność klasyfikacji wyrażoną w procentach. Wyniki dla modelu bazowego odpowiadającego wariantowi *E2018* zakreślone zostały kolorem ciemnoróżowym.

7.2.11 Rozmiar partii uczącej



Rys. 7.17: Wyniki klasyfikacji dla rozważanych architektur (kolumny „B”, „K”, „R”, „S”) w zależności od rozmiaru partii uczącej. Porównanie zakłada w przybliżeniu stały budżet czasowy dla eksperymentów, stąd zmiana rozmiaru partii wpływa również na efektywną liczbę aktualizacji wag (iteracji w ciągu jednej epoki). W związku z przekroczeniem dostępnych zasobów pamięciowych modele opierające się na architekturach „K” i „R” nie były oceniane na zbiorze *Minibirds* dla rozmiaru partii wynoszącego 512 przykładów. Komórki macierzy przedstawiają dokładność klasyfikacji wyrażoną w procentach. Wyniki dla modelu bazowego odpowiadającego wariantowi *E2018* zakreślone zostały kolorem ciemnoróżowym.

Aspektem ściśle związanym z całościowym kształtem procesu uczenia jest ustalenie wykorzystywanego rozmiaru partii uczącej. Analiza tego hiperparametru jest o tyle trudna, że konsekwencje jego zmian są złożeniem wielu czynników.

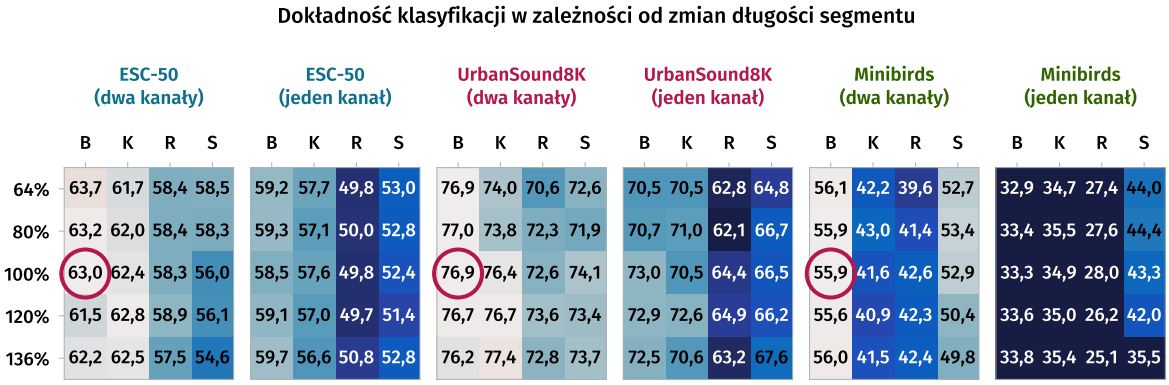
Po pierwsze, przyjęty rozmiar próby ma wpływ na stabilność wyznaczanej aproksymacji gradientu. Przy mniejszych wartościach uczenie ma bardziej nieregularny charakter, ale w praktyce okazuje się, że czasem to właśnie występowanie szumu w aktualizacjach wag może mieć pozytywny skutek.

Z drugiej strony, rozmiar partii uczącej jest nierozerwalnie związany z długością uczenia i liczbą iteracji. Zastosowanie mniej licznych partii powoduje, że przy tym samym czasie potrzebnym na wykonanie obliczeń przeprowadzana jest większa liczba aktualizacji wag modelu. Nie jest to efekt do końca liniowy w związku z narzutami komunikacyjnymi z kartą graficzną, ale mimo wszystko wymaga on ustalenia pewnego wspólnego mianownika do porównań – albo w postaci równej liczby kroków aktualizujących, albo określenia zbliżonego budżetu czasowego. W ramach eksperymentu przyjęte zostało drugie kryterium jako w praktyce istotniejsze z punktu widzenia typowego użytkownika.

Analiza rysunku 7.17 potwierdza, że przyjęty domyślnie rozmiar 128 przykładów uczących w partii jest dobrą wartością dla zbiorów *ESC-50* i *UrbanSound8K*. Słabsze wyniki w przypadku partii składających się z małej liczby przykładów (16, 32) mogłyby być związane z dużo większą liczbą iteracji i wystąpieniem efektu zbytniego dopasowania do danych, ale dokładniejsza ocena krzywych uczenia pozwala jednoznacznie odrzucić taką hipotezę. Przeważającym czynnikiem musi być niestabilność wyznaczania gradientu, gdyż modele uczone w ten sposób nie tylko wykazują się gorszą dokładnością klasyfikacji na zbiorze testowym uzyskiwaną na przestrzeni wszystkich epok, ale też finalnie osiągają wyższą wartość funkcji straty, nawet mimo znacząco większej liczby wykonanych kroków. Odmienną sytuację prezentują partie składające się z 512 przykładów, w przypadku których nie jest możliwe wykluczenie, że słabsze wyniki wynikają po prostu z zakończenia uczenia na zbyt wczesnym etapie.

Bardzo osobliwym zachowaniem cechują się wyniki architektury bazowej (B) uzyskane dla zbioru *Minibirds*. Zmniejszenie rozmiaru partii do 16 przykładów nie tylko istotnie poprawia dokładność klasyfikacji w wariancie dwukanałowym, ale też pozwala zbliżyć się dokładnością podejściu niewykorzystującemu spektrogramów różnicowych. Fakt ten jest trudny w interpretacji. Dość naturalnie mógłby się on wiązać ze stwierdzeniem, że domyślny czas uczenia na zbiorze *Minibirds* jest po prostu niewystarczający i dopiero większa liczba iteracji pozwala na osiągnięcie lepszych wyników. Z hipotezą tą nie koresponduje jednak zachowanie funkcji straty, która ostatecznie uzyskuje dla tego przypadku wartość o rząd wielkości większą (0,36) niż dla wariantu domyślnego (0,03), a mimo to wynik walidacji na zbiorze testowym jest istotnie lepszy. Można się więc w tym miejscu doszukiwać raczej problemów związanych z przeuczeniem lub specyficznym sformułowaniem samego zadania.

7.2.12 Długość segmentu



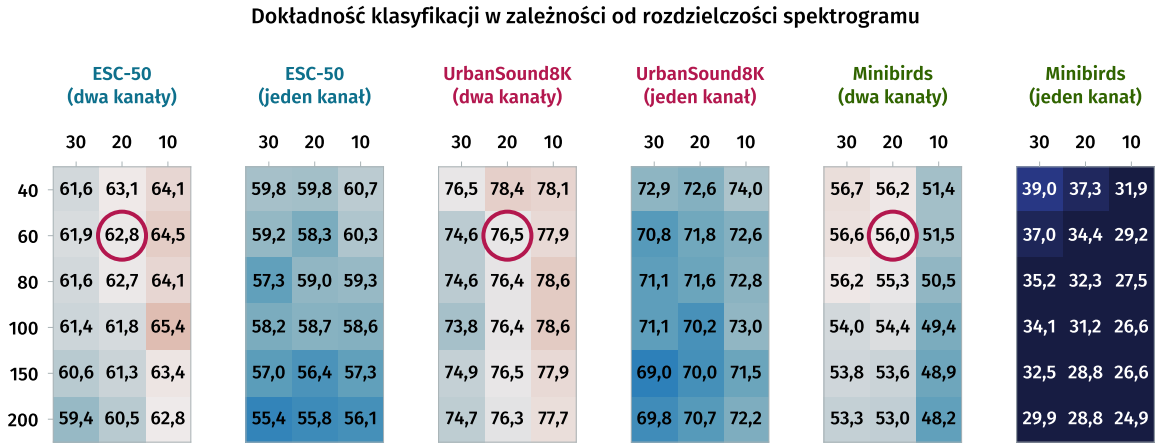
Rys. 7.18: Wyniki klasyfikacji dla rozważanych architektur (kolumny „B”, „K”, „R”, „S”) w zależności od długości segmentu uczącego wyrażonej w stosunku do ustawienia domyślnego (2,5 sekundy dla *ESC-50* i *UrbanSound8K*, 10 sekund dla *Minibirds*). Komórki macierzy przedstawiają dokładność klasyfikacji wyrażoną w procentach. Wyniki dla modelu bazowego odpowiadającego wariantowi *E2018* zakreślone zostały kolorem ciemnoróżowym.

Ostatnie dwa z analizowanych czynników związane są z formatem danych wejściowych wykorzystywanych w uczeniu splotowych sieci neuronowych. Ogólnym założeniem przyjętym w rozprawie jest stosowanie podejścia polegającego na wybieraniu krótszych fragmentów nagrań jako przykładów uczących. Celem tej części eksperymentu, której wyniki przedstawia rysunek 7.18, było stwierdzenie, na ile wybór konkretnej długości segmentu wpływa na uzyskiwaną dokładność klasyfikacji.

W przypadku *Minibirds* modyfikacje tego aspektu uczenia nie powodują znaczących zmian, co jest zrozumiałe w kontekście długiego czasu trwania segmentów wykorzystywanego domyślnie dla tego zbioru (500 ramek odpowiadających 10 sekundom nagrania). W sytuacji tej redukcja długości w wariancie „64%” do 320 ramek (6,4 sekundy) dalej generuje fragmenty, które nie powinny znacząco różnić się charakterystyką.

Nieco większe znaczenie mogłaby mieć tego typu manipulacja przeprowadzona dla zbioru *UrbanSound8K*, gdzie wariant „136%” (170 ramek – 3,4 sekundy) oznacza posługiwanie się 85% całości 4-sekundowego nagrania. Z wyjątkiem architektury „K” okazuje się to jednak nie mieć za dużego wpływu na poprawę uzyskiwanych wyników. Na przykładzie zbioru *ESC-50* można wręcz powiedzieć, że pewien potencjał leży w ograniczaniu długości segmentów. Nawet jeśli nie oddziałuje ona bezpośrednio na dokładność klasyfikacji, to ważną korzyścią krótszych reprezentacji danych wejściowych jest zmniejszenie wymagań obliczeniowo-pamięciowych, co skutkuje łatwiejszym, a przede wszystkim szybszym uczeniem modeli.

7.2.13 Rozdzielczość spektrogramu



Rys. 7.19: Wyniki klasyfikacji dla architektury bazowej (B) w zależności od zmian rozdzielczości spektrogramu. Wiersze określają liczbę pasm melowych, kolumny wartość przeskoku (w milisekundach). Najwyższą rozdzielczość dane wejściowe przyjmują dla komórek w prawym dolnym rogu macierzy. Zmiany liczby pasm częstotliwości zakładają jednakową korektę wysokości filtrów pierwszej warstwy splotowej. Poszczególne komórki przedstawiają dokładność klasyfikacji wyrażoną w procentach. Wyniki dla modelu bazowego odpowiadającego wariantowi *E2018* zakreślone zostały kolorem ciemnoróżowym.

Drugim hiperparametrem regulującym rozmiar danych wejściowych, poza absolutną długością segmentu, jest ich rozdzielczość. Jest ona w tym wypadku rozumiana jako liczba pasm melowych spektrogramu (wierszy) i liczba ramek (kolumn) przypadająca na jednostkę czasu (wynikająca z wartości przeskoku – *hop size*).

Niestety, wpływ zmian wykorzystywanej rozdzielczości spektrogramu jest czynnikiem trudnym do przeanalizowania ze względów praktycznych. Ustalenie liczby pasm na 200 przy jednoczesnym skróceniu przeskoku o połowę (z 20 ms do 10 ms) powoduje ponad sześciokrotne zwiększenie liczby wartości dostarczanych do przetworzenia przez sieć w stosunku do eksperymentów przeprowadzanych z ustawieniami domyślnymi. Poza bardzo znaczącym wydłużeniem czasu potrzebnego na uczenie modeli, główną przeszkodą w realizacji tego typu symulacji staje się w takim wypadku ilość wymaganej pamięci, która dla modeli o rozważanej głębokości zaczyna przekraczać możliwości dostępnych rozwiązań sprzętowych.

Pod tym względem osobliwym zachowaniem odznaczają się modele typu *1D CNN*, gdyż w ich przypadku zwiększenie rozdzielczości wertykalnej (liczby pasm częstotliwości) powoduje tylko niewielki przyrost złożoności obliczeniowej zamykający się na poziomie pierwszej warstwy splotowej. Z tego powodu, w ramach przeprowadzonego eksperymentu, ocenie poddana została tylko ta grupa modeli.

Analiza zestawienia na rysunku 7.19 wskazuje przede wszystkim na potencjał zastosowania reprezentacji o zwiększonej rozdzielczości czasowej (ramki o długości 10 ms) dla dźwięków środowiskowych (*ESC-50*, *UrbanSound8K*). Najlepszą kombinacją jest jej wykorzystanie w połączeniu ze 100 pasmami częstotliwości. Dalsze zwiększanie liczby pasm przynosi skutek negatywny. Odmienną konkluzję w tym aspekcie przedstawiały modele związane z klasyfikowaniem scen akustycznych **(Piczak, 2017)**, gdzie taki pozytywny efekt utrzymywał się do wartości 200 pasm. Może to więc być cecha związana ze specyfiką konkretnych nagrań.

W przypadku zbioru *Minibirds* zwiększanie rozdzielczości przynosi skutek odwrotny. Jest to o tyle nieintuicyjne, że w przeciwieństwie do klasyfikowania całych scen akustycznych, rozpoznawanie ptasich wokalizacji opiera się na wyszukiwaniu w spektrogramie bardzo specyficznych śladów, które często są tylko detalem na tle całości nagrania. W związku z tym uzyskanie wyższej rozdzielczości danych wejściowych powinno poprawiać możliwości wychwycenia takich niewielkich elementów. W analizowanym przykładzie dużo ważniejszy okazuje się jednak pozytywny efekt związany z łatwiejszym operowaniem na danych o mniejszym wymiarze.

7.3 Wnioski

Podsumowując zestawienia wyników zawarte w podrozdziałach 7.2.1–7.2.13 możliwe jest sformułowanie następujących wniosków odnośnie wpływu poszczególnych czynników na funkcjonowanie modeli splotowych w zadaniach klasyfikacji dźwięku:

- W przeanalizowanych przypadkach lepszą dokładność uzyskują modele wykorzystujące dodatkowy kanał różnicowy niż ich jednokanałowe odpowiedniki operujące wyłącznie na spektrogramach bazowych. Istotność względnej poprawy jest różna w zależności od konkretnej architektury i zbioru danych, ale można powiedzieć, że jest to efekt uniwersalnie pozytywny.
- W modelach określanych jako *1D CNN*, reprezentowanych przez architekturę bazową (B), zmniejszanie wysokości wertykalnych filtrów splotowych pierwszej warstwy połączone z odpowiadającym zwiększaniem okna grupowania zapewnia poprawę generalizacji, uodparniając modele na przesunięcia w dziedzinie częstotliwości (zmiany wysokości dźwięku). Z kolei modele jednowymiarowe uczące się wzorców rytmicznych (filtry horyzontalne zamiast wertykalnych) okazują się nieskuteczne w analizie dźwięków środowiskowych.
- Dla typowych architektur wizyjnych (model *2D CNN* - architektura „K") najpewniejszym ustawieniem jest ustalenie rozmiaru filtrów pierwszej warstwy na 3×3.

- Zwiększanie ogólnej szerokości modeli splotowych (liczby filtrów w poszczególnych warstwach) dla większości architektur przynosi zauważalną poprawę dokładności klasyfikacji. Jednocześnie też jest to jeden z czynników, który w bezpośredni sposób przekłada się na złożoność obliczeniową procesu uczenia modeli i wymagania pamięciowe, dlatego w praktyce konieczne staje się zrównoważenie tych dwóch przeciwstawnych aspektów.
- Głębokość architektur wykorzystanych w eksperymencie jest wystarczająca, a nawet można uznać, że lekko nadmiarowa. Potencjalnie pozytywny skutek może natomiast odnieść rozbudowa o dodatkowe warstwy w pełni połączone, chociaż trzeba pamiętać, że są to warstwy bardzo mocno zwiększające łączną liczbę parametrów sieci.
- Ważnym spostrzeżeniem eksperymentu jest wskazanie na kluczową rolę *dropoutu*. Jest to element, który powinien być dobierany konkretnie do danej architektury. Dla modeli z filtrami wertykalnymi ważnym aspektem jest zastosowanie *dropoutu* już na wczesnym etapie przetwarzania (spośród ocenianych wariantów najlepiej sprawdzał się *dropout* 10% po każdej warstwie). W przypadku architektur „K” i „R” okazuje się odwrotnie, że to właśnie niewprowadzanie *dropoutu* po pierwszej warstwie jest warunkiem koniecznym dla uzyskania satysfakcjonujących rezultatów.
- Normalizacja partiami (*batch normalization*) jest nieodzownym elementem uczenia modeli o konfiguracjach rozważanych w eksperymencie.
- Dobór konkretnego wariantu funkcji aktywacji zbliżonej do *ReLU* ma niewielki wpływ na dokładność klasyfikacji, dlatego pozostanie przy domyślnym wyborze (*Leaky ReLU*) jest postępowaniem uzasadnionym.
- Podobnie też optymalizacja wag za pomocą metody spadku stochastycznego gradientu z tempem uczenia $\eta = 0{,}01$ i *momentum* Nesterova o wartości 0,9 wydaje się dobrym rozwiązaniem niewymagającym dalszego dostrajania, przynajmniej w kontekście analizowanych zbiorów danych.
- Względnie niepotrzebnym zabiegiem okazała się natomiast regularyzacja wartości wag, która marginalnie wpływa na wyniki uzyskiwane w eksperymencie, można ją więc całkowicie pominąć.
- Z kolei w kwestii inicjalizacji wag nieznacznie lepszą od domyślnie obranej okazała się metoda *Glorot/Xavier uniform*. Również w tym wypadku jest to jednak aspekt o małym znaczeniu praktycznym.
- Wybrany *a priori* rozmiar partii (128 przykładów) wydaje się być dla przetwarzanych danych sensownym kompromisem między uzyskaniem stabilności procesu uczenia, jego akceptowalną szybkością i wykorzystaniem pamięci, zapewniając jednocześnie dobry poziom uzyskiwanej dokładności klasyfikacji.

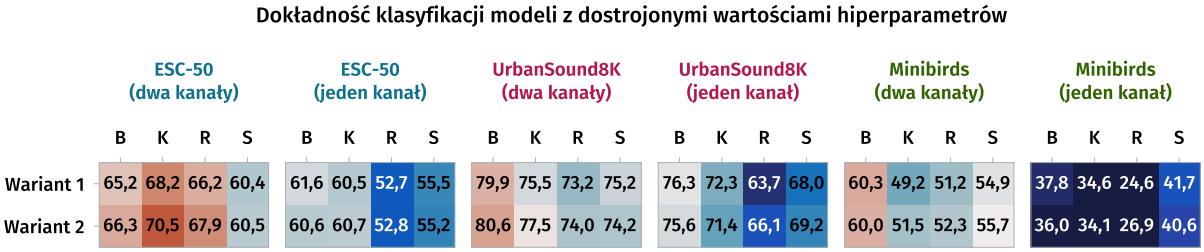
- Potencjalnym usprawnieniem związanym przede wszystkim ze zwiększeniem wydajności uczenia i działania modeli jest skrócenie długości przetwarzanych segmentów nagrań. Na bazie wyników zestawienia z rysunku 7.18 można bezpiecznie zaproponować redukcję do 64% domyślnej wartości.
- Innym obszarem umożliwiającym poprawę zdolności klasyfikowania dźwięków środowiskowych w przypadku architektury bazowej (B) jest zwiększenie rozdzielczości czasowej spektrogramów i liczby wykorzystywanych pasm częstotliwości. Wpływ ten nie został zbadany dla pozostałych architektur ze względu na skokowy przyrost zasobów wymaganych do tego typu przeliczeń.

W związku z tym, że wnioski odnośnie roli *dropoutu* w uczeniu modeli splotowych pokazały, że obrany początkowo kształt eksperymentu nie pozwala na zestawienie opisywanych architektur na równych zasadach i obiektywne porównanie uzyskiwanych wyników w kategoriach absolutnych, w ramach podsumowania przeprowadzony został dodatkowy eksperyment, którego rezultaty zestawia rysunek 7.20. Jego założeniem jest wykorzystanie dotychczas poczynionych spostrzeżeń i dostrojenie dobieranych wartości hiperparametrów dla każdej z architektur poprzez:

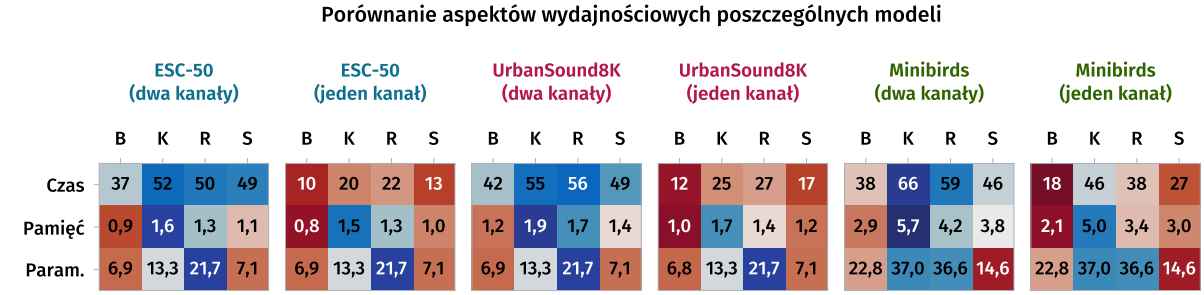
- wykorzystanie segmentów o długości zredukowanej do 64% wartości domyślnej,
- rezygnację z regularyzacji wag,
- inicjalizację wartości wag metodą *Xavier uniform*,
- dodanie dwóch warstw w pełni połączonych (wariant „FC 2” zestawienia 7.10),
- zastosowanie *dropoutu* z prawdopodobieństwem 10% po wszystkich warstwach (architektury „B” i „S”) lub 50% wyłącznie przed ostatnią warstwą („K” i „R”),
- zmianę rozmiaru filtrów pierwszej warstwy architektury „B” na 50×4.

Poza domyślnie przyjętą liczbą filtrów: $K_1 = 64$ (architektury „B”, „R”, „S”) lub $K_1 = 32$ (architektura „K”), ocenie poddane zostały również modele ze zwiększoną szerokością („Wariant 2”): $K_1 = 96$ (architektury „B”, „R”, „S”) lub $K_1 = 64$ (architektura „K”). Dla pełniejszego porównania architektur między sobą, rysunek 7.21 zestawia ze sobą również szacunkowe parametry wydajnościowe poszczególnych modeli, co powinno umożliwić lepsze umieszczenie uzyskiwanej dokładności klasyfikacji w kontekście złożoności poszczególnych wariantów.

Głównym wnioskiem płynącym z analizy tak uzyskanych wyników jest podkreślenie istotności doboru odpowiednich nastaw hiperparametrów. Porównanie najlepszych modeli przy ustawieniach domyślnych i dostrojonych wykazuje, w zależności od zbioru danych, różnice dokładności rzędu 7 (*ESC-50*), 3 (*UrbanSound8K*) lub 5 (*Minibirds*) punktów procentowych. Indywidualne poprawy poszczególnych wariantów potrafią być jeszcze większe (odpowiednio 8, 3 lub 12 punktów procentowych różnicy).



Rys. 7.20: Wyniki klasyfikacji dla rozważanych architektur przy dostrojeniu wartości hiperparametrów. „Wariant 1" zakłada domyślną szerokość poszczególnych modeli, natomiast „Wariant 2" przedstawia wyniki modeli ze zwiększoną szerokością. Komórki przedstawiają dokładność klasyfikacji wyrażoną w procentach.



Rys. 7.21: Przybliżone parametry wydajnościowe eksperymentów uzyskane dla „wariantu 1" z rysunku 7.20 uczonego przy wykorzystaniu procesora *AMD FX-8350* i karty *GeForce GTX 980 Ti*. Wiersze macierzy przedstawiają kolejno całkowity czas uczenia i walidacji dla jednej epoki (w sekundach), wykorzystaną pamięć karty graficznej (w GB) oraz łączną liczbę parametrów modelu (w milionach).

W eksperymencie potwierdza się tendencja dokładniejszego klasyfikowania przez modele dwukanałowe. W tym aspekcie największą dysproporcją względem podejścia jednokanałowego cechują się modele rezydualne (R). Zestawiając ze sobą poszczególne architektury, najlepiej ze zbiorem *ESC-50* radzi sobie rodzina modeli „K", ale rozwiązania bazowe „B" i rezydualne „R" pozostają niewiele w tyle. Z kolei dla pozostałych zbiorów najsilniej prezentują się właśnie modele typu „B".

Pozytywny efekt zwiększenia szerokości („wariant 2") widoczny jest dla modeli dwukanałowych. Uwzględnienie aspektów wydajnościowych pokazuje jednak, że wprowadzenie dodatkowego kanału jest operacją mocno kosztowną, potrafiącą zwiększyć czas uczenia nawet kilkukrotnie. W tym względzie zaproponowane w rozprawie wykorzystanie architektury bazowej jest o tyle atrakcyjne, że są to modele o relatywnie najmniejszych wymaganiach pamięciowych i najszybszym czasie uczenia. Spostrzeżenia te potwierdzają więc zasadność zaprezentowanego w rozprawie podejścia. Otwartym pytaniem pozostaje jednak kwestia, na ile uzyskiwane polepszenie zdolności klasyfikacyjnych zapewniane przez modele dwukanałowe jest istotne w stosunku do wprowadzanego tym sposobem zwiększenia ich złożoności.

Rozdział 8

Podsumowanie

Tematem, na którym skupiła się niniejsza rozprawa, jest klasyfikacja dźwięków występujących w środowisku naturalnym (niebędących mową i muzyką). Celem podejmowanych prac badawczych było zweryfikowanie hipotezy, że do zadania tego można z powodzeniem zaadaptować splotowe sieci neuronowe – modele, które sprawdziły się dotychczas w licznych zagadnieniach rozpoznawania obrazów. Co więcej, wykorzystanie tego typu podejścia powinno być możliwe pomimo ograniczonych rozmiarów zbiorów uczących dostępnych w tej dziedzinie, często mniejszych o kilka rzędów wielkości od analogicznych zasobów obrazowych.

Osiągnięcie tego zamiaru, stanowiącego główny cel rozprawy (pkt 1 podrozdziału 1.2), nastąpiło poprzez zaproponowanie konkretnych systemów klasyfikacji dźwięku bazujących na splotowych sieciach neuronowych i przeprowadzenie eksperymentalnej weryfikacji ich funkcjonowania na przykładzie zadań rozpoznawania dźwięków środowiskowych oraz gatunków ptaków śpiewających (pkt 2 listy celów). Przedstawiona w rozprawie metoda opiera się na przetwarzaniu spektrogramów wyrażonych w skali melowej za pomocą sieci wykorzystujących w pierwszej warstwie splotowej filtry o wertykalnym ułożeniu (analizujące krótkie wycinki czasowe w szerokim paśmie częstotliwości). Podejście takie skutkuje uzyskaniem dobrej dokładności klasyfikacji połączonej z poprawą wydajności uczenia względem typowych architektur wykorzystywanych w rozpoznawaniu obrazów (z małymi kwadratowymi filtrami splotowymi). Ważnym elementem ocenianych modeli okazało się również zainspirowane pracami Abdel-Hamida et al. (2014) zastosowanie dodatkowego przedstawienia danych wejściowych za pomocą spektrogramów różnicowych. Uzyskane wyniki eksperymentalne, a także następujące po ich publikacji upowszechnienie wykorzystania splotowych sieci neuronowych w obszarze rozumienia dźwięku, świadczą o dużym potencjale tej grupy modeli również na tym polu zastosowań.

W rozprawie poruszona też została kwestia wpływu zmian wartości hiperparametrów i sposobu konstruowania modeli na dokładność klasyfikacji. W tym celu przeprowadzona została bardzo rozległa weryfikacja eksperymentalna dla czterech wybranych wariantów

architektur splotowych, realizująca punkt 3 podrozdziału 1.2. Przygotowane tym sposobem zestawienie jest jednym z najszerszych porównań tego typu w dotychczasowej literaturze przedmiotu.

Cel zawarty w punkcie 4 został osiągnięty poprzez udostępnienie zbioru nagrań środowiskowych „ESC-50", którego szczegółowy opis znalazł się w rozdziale 4. Poza umożliwieniem przeprowadzenia zaplanowanych w rozprawie eksperymentów, sporządzenie takiego zasobu okazało się być ważną inicjatywą docenioną przez społeczność zajmującą się tematyką rozpoznawania dźwięku, skutkującą licznymi odniesieniami innych autorów do tego zbioru.

Opis eksperymentów zawarty w rozdziale 5 wzbogacony został o wizualizacje efektów uczenia splotowych sieci neuronowych przetwarzających spektrogramy (pkt 5 podrozdziału 1.2). Wizualizacje te demonstrują przykładowy sposób analizy pozwalającej na lepsze zrozumienie zasad działania splotowych sieci neuronowych w zastosowaniach dźwiękowych. Spojrzenie na wyniki od tej strony pokazuje, że modele te są przynajmniej w części interpretowalne, co pozwala zwiększyć poziom zaufania do generowanych przez nie predykcji.

Podsumowując te dokonania można więc stwierdzić, że wszystkie założone cele rozprawy wymienione w podrozdziale 1.2 zostały zrealizowane.

Główne rezultaty pracy

Ocena rozprawy pod kątem wkładu w rozwój dziedziny pozwala na wskazanie następujących głównych osiągnięć autorskich:

- zaprezentowanie jednego z pierwszych[1] opublikowanych zastosowań splotowych sieci neuronowych do klasyfikacji dźwięków środowiskowych **(Piczak, 2015a)**,
- zaprezentowanie jednego z pierwszych[2] opublikowanych zastosowań splotowych sieci neuronowych do rozpoznawania gatunków ptaków śpiewających **(Piczak, 2016)**,
- sporządzenie i upublicznienie etykietowanego zbioru 2000 nagrań środowiskowych „ESC-50" wraz z wynikami klasyfikacji za pomocą podstawowych metod automatycznych i osiąganych przez uczestników eksperymentu crowdsourcingowego **(Piczak, 2015b)**,
- przeprowadzenie szczegółowej analizy wpływu zmian wartości hiperparametrów na dokładność klasyfikacji dźwięku dla modeli będących przedstawicielami czterech typów architektur splotowych.

[1] Równolegle opublikowane zostały prace Zhanga et al. **(2015)** i Espiego et al. **(2015)**.
[2] Równolegle opublikowane zostały prace Sprengela et al. **(2016)** oraz Tótha i Czeby **(2016)**.

Podejmowane przez autora badania związane z klasyfikacją dźwięku za pomocą splotowych sieci neuronowych znalazły odzwierciedlenie w czterech pozycjach recenzowanych materiałów konferencji międzynarodowych, wymienionych szczegółowo w podrozdziale 1.4, a także w dostępnych publicznie kodach źródłowych i materiałach pomocniczych[3]. Prace te były wielokrotnie cytowane i ulepszane przez innych autorów, co pośrednio potwierdza przyjęcie zaproponowanej koncepcji przez środowisko naukowe.

Wnioski szczegółowe

Podsumowanie wniosków o charakterze merytorycznym, płynących z zaprezentowanych w rozprawie eksperymentów, a także z doświadczeń pozyskanych w czasie jej opracowywania, pozwala na poczynienie następujących konkluzji:

- Uczenie splotowych sieci neuronowych w zadaniach klasyfikacji dźwięku jest możliwe do przeprowadzenia z dobrym rezultatem, i to nawet przy niewielkiej skali zbiorów uczących (rzędu kilkudziesięciu przykładów na klasę), co na obecnym etapie rozwoju dziedziny ułatwia wykorzystanie tych modeli w potencjalnych problemach praktycznych.
- Splotowe sieci neuronowe opierające się na analizie spektrogramów są w momencie finalizowania treści rozprawy przodującą metodą, jeśli chodzi o dokładność klasyfikacji uzyskiwaną w zadaniach rozpoznawania dźwięku. Powoli rośnie też popularność modeli splotowych operujących bezpośrednio na przedstawieniu w dziedzinie czasu, ale na razie ustępują one możliwościami wykorzystaniu spektrogramów. Zaletą zastosowania splotowych sieci neuronowych w porównaniu do klasyfikacji na podstawie specjalistycznych atrybutów jest duża uniwersalność – modele te sprawdzają się w szerokim zakresie zadań analizy dźwięku nawet przy pominięciu etapu dostosowania do specyfiki konkretnego problemu.
- Przykładowe architektury zaprezentowane w części eksperymentalnej rozprawy stosunkowo dobrze sprawdzają się przy skali zadań osiąganej przez większość akademickich problemów w obszarze rozpoznawania dźwięku. W ich przypadku głębsze sieci nie zapewniają istotnej przewagi w porównaniu do sieci z kilkoma warstwami splotowymi. Lepszą decyzją okazuje się być zwiększanie szerokości poszczególnych warstw (liczby filtrów splotowych). Pewne oznaki pozytywnego wpływu głębokości widać dopiero przy uczeniu na milionach przykładów **(Hershey et al., 2017)**.
- Zastosowanie wertykalnych filtrów splotowych w pierwszej warstwie motywowane jest osiągnięciami obszaru rozpoznawania mowy, które zwracają uwagę na odmienny

[3] https://github.com/karoldvl/.

charakter osi spektrogramów, niż ma to miejsce dla typowych danych obrazowych. Filtry takie, analizujące krótkie wycinki czasowe w szerokim paśmie częstotliwości, zapewniają uzyskanie w przeprowadzonych eksperymentach wyników porównywalnych, a miejscami lepszych niż w przypadku małych filtrów kwadratowych. Zaletami rozwiązania zaprezentowanego w rozprawie jest większa wydajność uczenia i generowania predykcji dla tego typu modeli, skalowalność przy zwiększaniu rozdzielczości częstotliwościowej spektrogramu, a także bogatsza wartość informacyjna w przypadku pobieżnej wizualizacji wag pierwszej warstwy, do której to niestety często ogranicza się diagnostyka modeli splotowych.

- Inną decyzją, względnie unikalną pod kątem występowania w literaturze, jest ograniczenie wysokości filtrów do zakresu nieznacznie mniejszego od całkowitej liczby pasm w połączeniu z zastosowaniem odpowiedniej operacji grupowania następującej po pierwszej warstwie. Postępowanie takie pozwala uodpornić konstruowane modele na niewielkie przesunięcia w dziedzinie częstotliwości.
- Zauważalny wpływ na dokładność klasyfikacji uzyskiwaną w eksperymentach okazuje się mieć wprowadzenie dla danych wejściowych dodatkowego kanału „delta” w postaci spektrogramu różnicowego po czasie. Niestety, zastosowanie takiego wariantu przetwarzania pociąga za sobą istotne zwiększenie złożoności obliczeniowej tworzonych modeli.
- Bardzo ważnym hiperparametrem przy konstruowaniu modeli splotowych do klasyfikacji dźwięku jest prawdopodobieństwo odrzucenia neuronów dla techniki *dropout*. W połączeniu z filtrami wertykalnymi *dropout* powinien być stosowany już na wczesnym etapie przetwarzania, zaraz po pierwszej warstwie. Dla typowych architektur obrazowych (filtry kwadratowe, połączenia rezydualne) konieczne jest natomiast wprowadzenie *dropoutu* dopiero na etapie końcowym. Potrzeba zastosowania tej formy regularyzacji uczenia pokrywa się też ze spostrzeżeniami odnośnie funkcjonowania komitetów, poczynionymi w rozdziale 6 przy okazji omawiania zachowania modeli splotowych w konkursie *BirdCLEF 2016*. Uśrednianie predykcji, czy na etapie uczenia za pomocą *dropoutu*, czy końcowego łączenia kilku modeli splotowych sieci neuronowych, jest sprawdzonym sposobem poprawiania dokładności klasyfikacji uzyskiwanej na zbiorze testowym. Sugeruje to, że standardowy proces tworzenia głębokich sieci neuronowych prowadzi do zbytniego dopasowania do danych uczących i problem ten może być nasilony w obszarze dźwiękowym ze względu na niewielkie rozmiary zbiorów danych.
- Eksperymenty potwierdzają kluczową rolę zastosowania normalizacji partiami. Uczenie bez tej modyfikacji jest wprawdzie możliwe, ale wymaga zastosowania modeli z mniejszą liczbą warstw i znacznego wydłużenia samego procesu uczenia.

- Sieci splotowe rozważane w rozprawie skupiają się przede wszystkim na lokalnym analizowaniu krótkich fragmentów spektrogramu. Jest to ich dużą zaletą w porównaniu do typowych metod agregujących atrybuty na przestrzeni całości nagrania. Słabe rozumienie globalnego kontekstu skutkuje jednak czasem powstawaniem błędnych predykcji, które dla ludzi są dosyć proste do uniknięcia. Obiecującym usprawnieniem tego aspektu mogłaby być rozbudowa modeli splotowych o dodatkowe warstwy rekurencyjne **(Çakır et al., 2017; Parascandolo et al., 2017)**.
- Ostatnią ważną uwagą jest podkreślenie istotności samego procesu wstępnego przygotowania danych, odpowiedniego podziału na przykłady uczące i syntetycznego powiększania zbioru danych przez wprowadzane odgórnie deformacje. Nie oznacza to, że dobór architektury i dostrajanie hiperparametrów nie mają znaczenia, natomiast prawdą jest, że w konkretnych zadaniach klasyfikacyjnych różnice dokładności wynikające z tego tytułu mogą być nieznaczne w stosunku do wpływu, jaki ma samo przetworzenie nagrań. Efekt ten jest szczególnie widoczny w przypadku analizy śpiewu ptaków w ramach konkursu *BirdCLEF 2016* w związku z niejednorodną długością fragmentów, występowaniem szumu i wielu gatunków w tle, a także niewielkim procentowym udziale poszczególnych wokalizacji w całości przetwarzanych spektrogramów. Pośrednim potwierdzeniem tego aspektu jest występowanie w wynikach *BirdCLEF 2016* znaczących różnic w dokładności poszczególnych zgłoszeń bazujących na zbliżonych rozwiązaniach splotowych, dla których głównym wyróżnikiem był właśnie sposób przygotowania danych.

Dyskusja

Ciągły i otwarty charakter każdego procesu badawczego powoduje, że nie jest możliwe pozostawienie rozpatrywanego tematu bez pewnych refleksji dotykających jego słabszych stron czy pytań, na które nie udało się udzielić odpowiedzi w ramach rozprawy.

Komentarz odnośnie słabych elementów rozprawy

Konkretnym mankamentem wykorzystania splotowych sieci neuronowych jest mały stopień sformalizowania zasad ich konstruowania, dodatkowo wzmacniany dynamicznym rozwojem tego obszaru wiedzy. Decyzje co do kształtu rozwiązania dla danego problemu muszą być podejmowane częściowo poprzez próby zaadaptowania aktualnych doniesień publikacyjnych, które sprawdzają się w mniej lub bardziej oddalonych zagadnieniach, a częściowo na zasadzie weryfikacji metodą prób i błędów, wspomaganą intuicją i doświadczeniem badacza. Pokłosie tego procesu widać w omawianych w rozprawie eksperymentach, które opisują pewną ewolucję i dojrzewanie koncepcji na przestrzeni kilku lat. Przykładowo,

chybionym pomysłem w pierwotnej architekturze *E2015* okazało się wykorzystanie warstw gęstych, które mocno zwiększały złożoność modelu przy niewielkiej poprawie dokładności. Odpowiedzią na pojawiający się w tym obszarze gąszcz możliwości i próbą pewnego usystematyzowania wiedzy na temat obiecujących kierunków doboru architektury i wartości hiperparametrów jest przeprowadzona w rozprawie analiza wrażliwości. Mimo jej rozległego charakteru, liczba zaleceń, które można uznać za uniwersalnie obowiązujące, jest ograniczona.

Rozciągłość w czasie procesu badawczego powoduje również, że w roku 2018 zastosowanie splotowych sieci neuronowych do klasyfikacji dźwięku wydaje się możliwością stosunkowo oczywistą. Rzeczywiście, liczba opublikowanych w ostatnim czasie artykułów i dostępność kodu źródłowego przykładowych implementacji prowadzi do pewnej komodytyzacji, pozwalając dziś na zastosowanie tego typu modeli dla różnych problemów praktycznych przy stosunkowo małym wysiłku. W ocenie autora aspekt ten należałoby jednak odczytywać pozytywnie, jako potwierdzenie przetarcia pewnego szlaku i zasadności samego podejścia.

Prawdą jest też, że bezpośrednie porównanie wyników przedstawionych w rozdziale 5 z najnowszymi publikacjami odnoszącymi się do zbioru *ESC-50* (podrozdział 4.5) wskazuje na możliwość poprawy dokładności klasyfikacji uzyskiwanej przez zaproponowane modele. W tym kontekście warto jednak dokonać podziału potencjalnych usprawnień na trzy kategorie.

Najliczniejszą grupę stanowią podejścia opierające się na różnych formach wspomagania się dodatkowymi danymi, przykładowo poprzez transfer wiedzy pozyskanej z uczenia na zbiorze *AudioSet*, czy wykorzystanie nieetykietowanych danych w postaci bardzo dużej liczby filmów. Trudno więc bezpośrednio porównywać tego typu podejścia z zaprezentowanymi w rozprawie modelami uczonymi na samym zbiorze *ESC-50*.

Drugą kategorią są usprawnienia dotyczące procesu uczenia, które dotykają nieco odrębnego pola, nie zajmując się bezpośrednio kwestią samych splotowych sieci neuronowych. Powoduje to, że zmiany te mają charakter komplementarny i mogą być wykorzystane niezależnie od zaproponowanej architektury, co zresztą uczynili Tokozume et al. **(2017)**, poprawiając dokładność klasyfikacji architektury *E2015* dla *ESC-50* do 76,9%.

Usprawnienia w obszarze reprezentacji wejściowych, które na przykładzie zbioru *ESC-50* zastosowali Sailor et al. **(2017)**, Tak et al. **(2017)** czy Agrawal et al. **(2017)**, to z kolei jedyny typ modyfikacji, które wykluczają się z podejściem omówionym w rozprawie. W tym zakresie należy przyznać, że sposoby przedstawienia nagrań alternatywne do spektrogramów w skali melowej mogą być rzeczywiście lepszym wyborem – czy z powodu zachowania informacji fazowej, czy przez większą odporność na szum. Na tę chwilę są to jednak podej-

ścia zaprezentowane w pojedynczych publikacjach i choć wydają się bardzo obiecujące, to wymagałyby szerszej oceny w kontekście różnych zbiorów danych i architektur.

Pewna niejasność może się również pojawiać w porównaniu wyników dla zbioru *Minibirds* (rozdział 7) i dla pełnej kolekcji nagrań *BirdCLEF 2016* (rozdział 6). Chociaż *Minibirds* został celowo ograniczony do prostszego problemu klasyfikacji tylko 100 gatunków zamiast 999, to wyniki procentowe przedstawione w rozdziale 7 dla wielu analizowanych wariantów nie są wcale lepsze od dokładności klasyfikacji uzyskiwanej przy rozpoznawaniu wszystkich gatunków. Rozbieżność ta pokazuje podkreślaną już w podsumowaniu rolę odpowiedniego przygotowania i wstępnego przetworzenia danych dla analizy nagrań ptaków. Dla uproszczenia modele przedstawione w rozdziale 7 stosowały taki sam sposób wstępnej obróbki dla klasyfikacji krótkich nagrań dźwięków środowiskowych, jak i dla dużo bardziej niejednorodnych materiałów z wokalizacjami ptaków. Taka konstrukcja eksperymentów zakłada, że ogólne relacje między wynikami dla poszczególnych architektur i dobieranych wartości hiperparametrów są zachowane pomimo zaniżenia absolutnego poziomu dokładności.

Podobnie też pytaniem narzucającym się w kontekście konkursu *BirdCLEF 2016* jest kwestia braku wykorzystania spektrogramów różnicowych. Eksperymenty porównawcze w rozdziale 7 w miarę jednoznacznie wskazują na ich dużą użyteczność. Dlaczego więc nie pojawiły się w zgłoszeniach zaprezentowanych w rozdziale 6? Odpowiedź jest o tyle prozaiczna, że wynika przede wszystkim ze skali problemu przedstawionego przez pełny zbiór *BirdCLEF 2016*. Uczenie omawianych modeli nawet w wariancie jednokanałowym wymagało kilkudziesięciu godzin dla pojedynczej architektury. Wydłużenie tego procesu przez wprowadzenie kanału różnicowego nie było więc możliwe ze względu na ograniczenia czasowe udziału w konkursie.

Omówienie otwartych problemów i kierunków dalszych badań

Przeprowadzone przedstawienie koncepcji dwukanałowych splotowych sieci neuronowych z filtrami wertykalnymi zawiera dwa obszary, których nie udało się podjąć w przewidzianym w rozprawie zakresie prac. Pierwszym z nich jest próba lepszego zrozumienia znaczenia dodatkowego kanału wejściowego. Na podstawie wykonanych eksperymentów nie jest możliwe stwierdzenie, czy wykazywana poprawa dokładności klasyfikacji modeli dwukanałowych jest ściśle związana z dużą użytecznością samej reprezentacji różnicowej, czy w rzeczywistości głównym powodem zmian jest samo podwojenie liczby wag wejściowych filtrów. Jednym ze sposobów weryfikacji tego aspektu mogłoby być powtórzenie eksperymentów dla wariantu dwukanałowego w taki sposób, aby obydwa kanały powielały tę samą informację (spektrogram bazowy).

Drugą wątpliwością jest sposób konstruowania filtrów wertykalnych, które obejmując duży zakres częstotliwości, jednocześnie mocno zwiększają wymiarowość. Dla warstwy splotowej 3×3 wykorzystywane jest tylko 9 wag na kanał. Zastosowanie wejściowego filtra o kształcie 57×6 zwiększa liczbę wag blisko 40-krotnie. Jedną z przesłanek wprowadzenia sieci *VGGNet* było natomiast założenie, że objęcie tego samego pola postrzegania lepiej jest dokonywać złożeniem kilku warstw o małych rozmiarach filtrów splotowych, niż pojedynczą warstwą z dużymi filtrami. Z tego powodu pomysłem wartym rozważenia mogłoby być zastąpienie szerokopasmowych filtrów wertykalnych odpowiednim złożeniem kilku warstw z mniejszymi filtrami, które obejmowałyby efektywnie ten sam zakres danych (krótki czas, całe pasmo).

Oprócz kwestii wynikających bezpośrednio z rozwinięcia eksperymentów przeprowadzanych w rozprawie, w obszarze klasyfikacji dźwięku można wyróżnić trzy główne kierunki badań, które zdaniem autora są istotnymi wyzwaniami na drodze do stworzenia uniwersalnych klasyfikatorów dźwiękowych, które radziłyby sobie w niesprzyjających okolicznościach bardzo dużej liczby klas i niekontrolowanych warunków nagraniowych.

Pierwszym z nich są próby uodpornienia modeli na występowanie szumu i jednocześnie poprawy zdolności klasyfikacyjnych dla mniej wyraźnych dźwięków. Niedomagania splotowych sieci neuronowych widać w tej kwestii wyraźnie na przykładzie wyników klasyfikacji nagrań tła *BirdCLEF* czy zadań konkursowych *DCASE* zakładających polifoniczną detekcję wielu jednoczesnych źródeł dźwięku. O ile w przypadku klasyfikacji głównego zdarzenia czy kontekstu akustycznego uzyskiwana dokładność jest zadowalająca, to charakterystyka działania klasyfikatorów dźwiękowych bardzo pogarsza się dla dźwięków mniej wyraźnych.

Drugim kierunkiem, który jest też częściową odpowiedzią na wymienione problemy, są próby poprawy zdolności klasyfikacyjnych przez wprowadzenie modyfikacji na etapie wejścia lub wyjścia splotowych sieci neuronowych. Mogą nimi być przywoływane zastosowania innych reprezentacji danych **(Sailor et al., 2017; Tak et al., 2017; Agrawal et al., 2017)**, jednoczesne przetwarzanie w różnych skalach czasowych **(Zhu et al., 2018)** czy ze skupianiem uwagi **(Guo et al., 2017)** lub wprowadzenie dodatkowych warstw rekurencyjnych **(Parascandolo et al., 2017; Çakır et al., 2017)**.

Najobszerniejszym tematem badawczym wydaje się natomiast kwestia dostosowania modeli splotowych do zmiany skali analizowanych problemów – z dotychczasowego ograniczonego rozpoznawania kilkudziesięciu wybranych klas dźwiękowych do tysięcy możliwości, które można napotkać w rzeczywistych niekontrolowanych warunkach nagraniowych. Poza obszarem rozumienia dźwięku, tematyką tą zajmują się na przykład wyspecjalizowane warsztaty organizowane przy konferencji *NIPS*[4]. Uczenie na tak dużą skalę siłą rze-

[4] Extreme Classification: Multi-class and Multi-label Learning in Extremely Large Label Spaces. http://manikvarma.org/events/XC17/.

czy wymaga dostosowania do sytuacji, w której liczba dostępnych etykiet pociąga za sobą pogorszenie ich jakości (tzw. problem *weak/noisy labels*) **(Kumar & Raj, 2017b)**. Cenne mogą się też okazać podejścia wykorzystujące innowacyjne sposoby syntetycznego zwiększania różnorodności danych uczących, sieci typu *GAN* czy dodatkową informację zawartą w zależnościach między zapisem wizyjnym a dźwiękowym tego samego zdarzenia **(Aytar et al., 2016; Arandjelović & Zisserman, 2017a)**. Ambitnym zadaniem byłoby też stworzenie systemów klasyfikujących na bazie uczenia aktywnego (*active learning*) i przyrostowego (*incremental learning*). Przykładem zarysu takiej koncepcji jest *NELS: Never-Ending Learner of Sounds* **(Elizalde et al., 2018)**. Postępowanie to zakłada niekończące się doszkalanie modelu (ucznia) poprzez, z czasem coraz rzadsze, dopytywanie człowieka (nauczyciela) o poprawną interpretację konkretnych nagrań. W przypadku powodzenia takiego podejścia byłaby to szansa na stworzenie klasyfikatora dźwiękowego o możliwie uniwersalnym zastosowaniu. Być może na przestrzeni najbliższych kilku lat uda się taki cel osiągnąć, a z pewnością przynajmniej mocno do niego przybliżyć.

Bibliografia

Abdel-Hamid, O. et al. (2012). Applying Convolutional Neural Networks Concepts to Hybrid NN-HMM Model for Speech Recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japonia. IEEE, s. 4277–4280.

Abdel-Hamid, O. et al. (2013). Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition. *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, Francja. ISCA, s. 1173–1175.

Abdel-Hamid, O. et al. (2014). Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22.10, s. 1533–1545.

Abeßer, J. et al. (2017). Acoustic Scene Classification by Combining Autoencoder-Based Dimensionality Reduction and Convolutional Neural Networks. *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. Monachium, Niemcy. TUT, s. 7–11.

Adavanne, S. et al. (2017). Sound Event Detection Using Spatial Features and Convolutional Recurrent Neural Network. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Nowy Orlean, USA. IEEE, s. 771–775.

Agrawal, D. M. et al. (2017). Novel TEO-Based Gammatone Features for Environmental Sound Classification. *Proceedings of the European Signal Processing Conference (EUSIPCO)*. Kos, Grecja. IEEE, s. 1809–1813.

Aguilar, M. (2013). *More Than 4 Million Spotify Songs Have Never Been Played*.
URL: http://gizmodo.com/more-than-4-million-spotify-songs-have-never-been-playe-1444955615.

Andén, J. & S. Mallat (2014). Deep Scattering Spectrum. *IEEE Transactions on Signal Processing*, 62.16, s. 4114–4128.

Anderson, S. E. et al. (1996). Template-Based Automatic Recognition of Birdsong Syllables from Continuous Recordings. *The Journal of the Acoustical Society of America*, 100.2, s. 1209–1219.

Arandjelović, R. & A. Zisserman (2017a). Look, Listen and Learn. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Wenecja, Włochy. IEEE, s. 609–617.

Arandjelović, R. & A. Zisserman (2017b). Objects That Sound. arXiv:1712.06651.

Arora, S. et al. (2014). Provable Bounds for Learning Some Deep Representations. *Proceedings of the International Conference on Machine Learning (ICML)*. Pekin, Chiny. PMLR, s. 584–592.

Asgari, M. et al. (2014). Inferring Social Contexts from Audio Recordings Using Deep Neural Networks. *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. Reims, Francja. IEEE, s. 6.

Aytar, Y. et al. (2016). SoundNet: Learning Sound Representations from Unlabeled Video. *Advances in Neural Information Processing Systems (NIPS)*. Barcelona, Hiszpania. NIPS Foundation, s. 892–900.

Ba, J. L. et al. (2016). Layer Normalization. arXiv:1607.06450.

Baelde, M. et al. (2017). A Mixture Model-Based Real-Time Audio Sources Classification Method. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Nowy Orlean, USA. IEEE, s. 2427–2431.

Barchiesi, D. et al. (2015). Acoustic Scene Classification: Classifying Environments from the Sounds They Produce. *IEEE Signal Processing Magazine*, 32.3, s. 16–34.

Beltrán, J. et al. (2015). Scalable Identification of Mixed Environmental Sounds, Recorded from Heterogeneous Sources. *Pattern Recognition Letters*, 68, s. 153–160.

Bengio, Y. et al. (2013). Advances in Optimizing Recurrent Networks. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Kanada. IEEE, s. 8624–8628.

Boddapati, V. et al. (2017). Classifying Environmental Sounds Using Image Recognition Networks. *Procedia Computer Science*, 112, s. 2048–2056.

Bogdanov, D. et al. (2013). Essentia: An Audio Analysis Library for Music Information Retrieval. *Proceedings of the Conference of the International Society for Music Information Retrieval (ISMIR)*. Curitiba, Brazylia. ISMIR, s. 493–498.

Bottou, L. et al. (2016). Optimization Methods for Large-Scale Machine Learning. arXiv:1606.04838.

Branson, S. et al. (2014). Bird Species Categorization Using Pose Normalized Deep Convolutional Nets. arXiv:1406.2952.

Briggs, F. et al. (2013). The 9th Annual MLSP Competition: New Methods for Acoustic Classification of Multiple Simultaneous Bird Species in a Noisy Environment. *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. Southampton, Wielka Brytania. IEEE, s. 8.

Brock, A. et al. (2016). Neural Photo Editing with Introspective Adversarial Networks. arXiv:1609.07093.

Cai, J. et al. (2007). Sensor Network for the Monitoring of Ecosystem: Bird Species Recognition. *Proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP)*. Melbourne, Australia. IEEE, s. 293–298.

Çakır, E. et al. (2017). Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25.6, s. 1291–1303.

Chachada, S. & C.-C. J. Kuo (2014). Environmental Sound Recognition: A Survey. *APSIPA Transactions on Signal and Information Processing*, 3, s. 15.

Chambolle, A. (2004). An Algorithm for Total Variation Minimization and Applications. *Journal of Mathematical imaging and vision*, 20.1-2, s. 89–97.

Chen, L.-C. et al. (2016). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. arXiv:1606.00915.

Choi, Y. et al. (2017). StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. arXiv:1711.09020.

Chu, S. et al. (2006). Where Am I? Scene Recognition for Mobile Robots Using Audio Features. *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*. Toronto, Kanada. IEEE, s. 885–888.

Chu, S. et al. (2009). Environmental Sound Recognition With Time–Frequency Audio Features. *IEEE Transactions on Audio, Speech, and Language Processing*, 17.6, s. 1142–1158.

Cireşan, D. et al. (2011). High-Performance Neural Networks for Visual Object Classification. arXiv:1102.0183.

Cireşan, D. et al. (2012). Multi-Column Deep Neural Network for Traffic Sign Classification. *Neural Networks*, 32, s. 333–338.

Clevert, D.-A. et al. (2015). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). arXiv:1511.07289.

Cotton, C. V. & D. P. Ellis (2011). Spectral vs. Spectro-Temporal Features for Acoustic Event Detection. *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz, USA. IEEE, s. 69–72.

Cowling, M. & R. Sitte (2003). Comparison of Techniques for Environmental Sound Recognition. *Pattern recognition letters*, 24.15, s. 2895–2907.

Cybenko, G. (1989). Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2.4, s. 303–314.

Dai, W. et al. (2017). Very Deep Convolutional Neural Networks for Raw Waveforms. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Nowy Orlean, USA. IEEE, s. 421–425.

de Boer, E. & H. R. de Jongh (1978). On Cochlear Encoding: Potentialities and Limitations of the Reverse-Correlation Technique. *The Journal of the Acoustical Society of America*, 63.1, s. 115–135.

Deng, L. et al. (2013a). A Deep Convolutional Neural Network Using Heterogeneous Pooling for Trading Acoustic Invariance with Phonetic Confusion. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Kanada. IEEE, s. 6669–6673.

Deng, L. et al. (2013b). Recent Advances in Deep Learning for Speech Research at Microsoft. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Kanada. IEEE, s. 8604–8608.

Dozat, T. (2016). Incorporating Nesterov Momentum into Adam. *International Conference on Learning Representations (ICLR) - Workshop Track*. San Juan, Puerto Rico. ICLR, s. 4.

Duchi, J. et al. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12 (Jul), s. 2121–2159.

Durugkar, I. et al. (2016). Generative Multi-Adversarial Networks. arXiv:1611.01673.

Eghbal-Zadeh, H. et al. (2016). CP-JKU Submissions for DCASE-2016: A Hybrid Approach Using Binaural i-Vectors and Deep Convolutional Neural Networks. *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, s. 5.

Elizalde, B. et al. (2018). NELS - Never-Ending Learner of Sounds. arXiv:1801.05544.

Erhan, D. et al. (2009). *Visualizing Higher-Layer Features of a Deep Network*. Technical Report 1341. Université de Montréal, s. 13.

Eronen, A. et al. (2006). Audio-Based Context Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 14.1, s. 321–329.

Espi, M. et al. (2015). Exploiting Spectro-Temporal Locality in Deep Learning Based Acoustic Event Detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015.1, s. 12.

Eyben, F. (2015). *Real-Time Speech and Music Classification by Large Audio Feature Space Extraction*. Springer.

Eyben, F. et al. (2010). OpenSMILE: The Munich Versatile and Fast Open-Source Audio Feature Extractor. *Proceedings of the ACM International Conference on Multimedia (ACMMM)*. Firenze, Włochy. ACM, s. 1459–1462.

Eyben, F. et al. (2013). Recent Developments in OpenSMILE, the Munich Open-Source Multimedia Feature Extractor. *Proceedings of the ACM International Conference on Multimedia (ACMMM)*. Barcelona, Hiszpania. ACM, s. 835–838.

Fazekas, B. et al. (2017). A Multi-Modal Deep Neural Network Approach to Bird-Song Identification. *Working Notes of CLEF Conference*. Dublin, Irlandia. CLEF, s. 655–667.

Fieldsend, J. E. & R. M. Everson (2005). Visualisation of Multi-Class ROC Surfaces. *International Conference on Machine Learning (ICML) - 2nd ROCML Workshop*. Bonn, Niemcy. PMLR, s. 49–56.

Foggia, P. et al. (2014). Cascade Classifiers Trained on Gammatonegrams for Reliably Detecting Audio Events. *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. Seul, Korea Południowa. IEEE, s. 50–55.

Fonseca, E. et al. (2017). Freesound Datasets: A Platform for the Creation of Open Audio Datasets. *Proceedings of the Conference of the International Society for Music Information Retrieval (ISMIR)*. Suzhou, Chiny. ISMIR, s. 486–493.

Font, F. et al. (2013). Freesound Technical Demo. *Proceedings of the ACM International Conference on Multimedia (ACMMM)*. Barcelona, Hiszpania. ACM, s. 411–412.

Foster, P. et al. (2015). CHiME-Home: A Dataset for Sound Source Recognition in a Domestic Environment. *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz, USA. IEEE, s. 5.

Freitag, M. et al. (2017). auDeep: Unsupervised Learning of Representations from Audio with Deep Recurrent Neural Networks. arXiv:1712.04382.

Fritzler, A. et al. (2017). Recognizing Bird Species in Audio Files Using Transfer Learning. *Working Notes of CLEF*. Dublin, Irlandia. CLEF, s. 14.

Fukushima, K. (1975). Cognitron: A Self-Organizing Multilayered Neural Network. *Biological cybernetics*, 20.3-4, s. 121–136.

Fukushima, K. (1980). Neocognitron: A Self Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological cybernetics*, 36.4, s. 193–202.

Geiger, J. T. et al. (2013). Large-Scale Audio Feature Extraction and SVM for Acoustic Scene Classification. *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz, USA. IEEE, s. 4.

Gemmeke, J. F. et al. (2017). Audio Set: An Ontology and Human-Labeled Dataset for Audio Events. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Nowy Orlean, USA. IEEE, s. 776–780.

Gencoglu, O. et al. (2014). Recognition of Acoustic Events Using Deep Neural Networks. *Proceedings of the European Signal Processing Conference (EUSIPCO)*. Lizbona, Portugalia. IEEE, s. 506–510.

Ghoraani, B. & S. Krishnan (2011). Time–Frequency Matrix Feature Extraction and Classification of Environmental Audio Signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 19.7, s. 2197–2209.

Giannakopoulos, T. (2015). pyAudioAnalysis: An Open-Source Python Library for Audio Signal Analysis. *PLOS ONE*, 10.12, e0144610.

Glorot, X. & Y. Bengio (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. *Proceedings of the International Conference on Artificial Intelligence and Statistics (AIStats)*. Sardynia, Włochy. PMLR, s. 249–256.

Glotin, H. (2013). 1st Workshop on Machine Learning for Bioacoustics - ICML4B. *Proceedings of the ICML4B International Workshop Joint to ICML*. Atlanta, USA. ICML, s. 129.

Glotin, H. et al. (2013). Neural Information Processing Scaled for Bioacoustics - From Neurons to Big Data. *Proceedings of the NIPS4B International Workshop Joint to NIPS*. Lake Tahoe, USA. NIPS Foundation, s. 262.

Goëau, H. et al. (2016). LifeCLEF Bird Identification Task 2016: The Arrival of Deep Learning. *Working Notes of CLEF*. Évora, Portugal. CLEF, s. 440–449.

Goh, G. (2017). Why Momentum Really Works. *Distill*, 2017.

Goodfellow, I. (2016). NIPS 2016 Tutorial: Generative Adversarial Networks. arXiv:1701.00160.

Goodfellow, I. et al. (2013). Maxout Networks. *Proceedings of the International Conference on Machine Learning (ICML)*. Atlanta, USA. PMLR, s. 9.

Goodfellow, I. et al. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems (NIPS)*. Montréal, Kanada. NIPS Foundation, s. 2672–2680.

Goodfellow, I. et al. (2016). *Deep Learning*. MIT Press.

Gouyon, F. et al. (2000). On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds. *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*. Werona, Włochy. DAFx, s. 6.

Guérin, E. et al. (2017). Interactive Example-Based Terrain Authoring with Conditional Generative Adversarial Networks. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia*, 36.6, s. 13.

Gülçehre, Ç. & Y. Bengio (2013). Knowledge Matters: Importance of Prior Information for Optimization. arXiv:1301.4083.

Guo, J. et al. (2017). Attention Based CLDNNs for Short-Duration Acoustic Scene Classification. *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*. Sztokholm, Szwecja. ISCA, s. 469–473.

Hagan, M. T. et al. (2014). *Neural Network Design*. 2nd Edition. Martin Hagan.

Haykin, S. (2004). A Comprehensive Foundation. *Neural Networks*, 2.2004, s. 41.

He, K. et al. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Las Condes, Chile. CVF, s. 1026–1034.

He, K. et al. (2016a). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, USA. CVF, s. 770–778.

He, K. et al. (2016b). Identity Mappings in Deep Residual Networks. *Proceedings of the European Conference on Computer Vision (ECCV)*. Amsterdam, Holandia. Springer, s. 630–645.

Heittola, T. et al. (2013). Context-Dependent Sound Event Detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2013.1, s. 1.

Hershey, S. et al. (2017). CNN Architectures for Large-Scale Audio Classification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Nowy Orlean, USA. IEEE, s. 131–135.

Hertel, L. et al. (2016). Comparing Time and Frequency Domain for Audio Event Recognition Using Deep Learning. *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. Vancouver, Kanada. IEEE, s. 3407–3411.

Hindupur, A. (2017). *The GAN Zoo: A List of All Named GANs!*
URL: https://github.com/hindupuravinash/the-gan-zoo.

Hinton, G. (2012). *Neural Networks for Machine Learning - Lecture 6, Slide 29*.
URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

Hinton, G. E. & R. R. Salakhutdinov (2006). Reducing the Dimensionality of Data with Neural Networks. *science*, 313.5786, s. 504–507.

Hinton, G. E. et al. (2012). Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. arXiv:1207.0580.

Huzaifah, M. (2017). Comparison of Time-Frequency Representations for Environmental Sound Classification Using Convolutional Neural Networks. arXiv:1706.07156.

Iandola, F. N. et al. (2016). SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <0.5MB Model Size. arXiv:1602.07360.

IDC (2017). *Data Age 2025: The Evolution of Data to Life-Critical*.
URL: https://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf.

IOC (2016). *IOC World Bird List*.
URL: http://www.worldbirdnames.org/.

Ioffe, S. (2017). Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. arXiv:1702.03275.

Ioffe, S. & C. Szegedy (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the International Conference on Machine Learning (ICML)*. Lille, Francja. PMLR, s. 448–456.

Isola, P. et al. (2016). Image-to-Image Translation with Conditional Adversarial Networks. arXiv:1611.07004.

Istrate, D. et al. (2006). Information Extraction from Sound for Medical Telemonitoring. *IEEE Transactions on Information Technology in Biomedicine*, 10.2, s. 264–274.

Jarrett, K. et al. (2009). What Is the Best Multi-Stage Architecture for Object Recognition? *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Kyoto, Japonia. IEEE, s. 2146–2153.

Jin, X. et al. (2017). WSNet: Compact and Efficient Networks with Weight Sampling. arXiv:1711.10067.

Jing, Y. et al. (2017). Neural Style Transfer: A Review. arXiv:1705.04058.

Joly, A. et al. (2014). Instance-Based Bird Species Identication with Undiscriminant Features Pruning. *Working Notes of CLEF*. Sheffield, Wielka Brytania. CLEF, s. 9.

Joly, A. et al. (2015). Shared Nearest Neighbors Match Kernel for Bird Songs Identification. *Working Notes of CLEF*. Tuluza, Francja. CLEF, s. 10.

Kahl, S. et al. (2017). Large-Scale Bird Sound Classification Using Convolutional Neural Networks. *Working Notes of CLEF*. Dublin, Irlandia. CLEF, s. 14.

Karbasi, M. et al. (2011). Environmental Sound Classification Using Spectral Dynamic Features. *Proceedings of the International Conference on Information, Communications and Signal Processing (ICICS)*. Singapur. IEEE, s. 5.

Khunarsal, P. et al. (2013). Very Short Time Environmental Sound Classification Based on Spectrogram Pattern Matching. *Information Sciences*, 243, s. 57–74.

Kim, H.-G. et al. (2005). *MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval*. John Wiley & Sons.

Kingma, D. P. & J. L. Ba (2014). Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Klambauer, G. et al. (2017). Self-Normalizing Neural Networks. arXiv:1706.02515.

Konda, K. et al. (2014). Zero-Bias Autoencoders and the Benefits of Co-Adapting Features. arXiv:1402.3337.

Kons, Z. & O. Toledo-Ronen (2013). Audio Event Classification Using Deep Neural Networks. *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, Francja. ISCA, s. 1482–1486.

Koops, H. V. et al. (2014). A Deep Neural Network Approach to the Lifeclef 2014 Bird Task. *Working Notes of CLEF*. Sheffield, Wielka Brytania. CLEF, s. 634–642.

Kriesel, D. (2007). *A Brief Introduction on Neural Networks*.
URL: http://www.dkriesel.com/en/science/neural_networks.

Krizhevsky, A. et al. (2012). Imagenet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems (NIPS)*. Lake Tahoe, USA. NIPS Foundation, s. 1097–1105.

Kumar, A. & B. Raj (2016). Features and Kernels for Audio Event Recognition. arXiv:1607.05765.

Kumar, A. & B. Raj (2017a). Audio Event and Scene Recognition: A Unified Approach Using Strongly and Weakly Labeled Data. *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. Anchorage, USA. IEEE, s. 3475–3482.

Kumar, A. & B. Raj (2017b). Deep CNN Framework for Audio Event Recognition Using Weakly Labeled Web Data. arXiv:1707.02530.

Kumar, A. et al. (2017). Knowledge Transfer from Weakly Labeled Audio Using Convolutional Neural Network for Sound Events and Scenes. arXiv:1711.01369.

Kurenkov, A. (2015a). *A 'Brief' History of Neural Nets and Deep Learning, Part 1*.
URL: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/.

Kurenkov, A. (2015b). *A 'Brief' History of Neural Nets and Deep Learning, Part 4*.
URL: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning-part-4/.

Lasseck, M. (2014). Large-Scale Identification of Birds in Audio Recordings. *Working Notes of CLEF*. Sheffield, Wielka Brytania. CLEF, s. 643–653.

Lasseck, M. (2015). Improved Automatic Bird Identification through Decision Tree Based Feature Selection and Bagging. *Working Notes of CLEF*. Tuluza, Francja. CLEF, s. 12.

Le, Q. V. et al. (2011). On Optimization Methods for Deep Learning. *Proceedings of the International Conference on Machine Learning (ICML)*. Bellevue, USA. PMLR, s. 265–272.

LeCun, Y. (1985). Une Procédure d'apprentissage Pour Réseau a Seuil Asymmetrique (a Learning Scheme for Asymmetric Threshold Networks). *Proceedings of Cognitiva 85*. Paryż, Francja. CESTA, s. 599–604.

LeCun, Y. (2016). *What Are Some Recent and Potentially Upcoming Breakthroughs in Deep Learning?*
URL: https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-deep-learning.

LeCun, Y. et al. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural computation*, 1.4, s. 541–551.

LeCun, Y. et al. (1998a). Efficient BackProp. *Lecture Notes in Computer Science*, 7700, s. 9–48.

LeCun, Y. et al. (1998b). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86.11, s. 2278–2324.

Ledig, C. et al. (2016). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. arXiv:1609.04802.

Lee, H. et al. (2009). Unsupervised Feature Learning for Audio Classification Using Convolutional Deep Belief Networks. *Advances in Neural Information Processing Systems (NIPS)*. Vancouver, Kanada. NIPS Foundation, s. 1096–1104.

Leng, Y. R. et al. (2014). Bird Classification Using Ensemble Classifiers. *Working Notes of CLEF*. Sheffield, Wielka Brytania. CLEF, s. 654–661.

Lostanlen, V. et al. (2018). Birdvox-Full-Night: A Dataset and Benchmark for Avian Flight Call Detection. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, Kanada. IEEE, s. 5.

Lotter, W. et al. (2015). Unsupervised Learning of Visual Structure Using Predictive Generative Networks. arXiv:1511.06380.

Loughlin, P. & L. Cohen (2010). Wavelets: A Comparison with the Spectrogram and Other Methods for Time-Frequency Analysis. *The Journal of the Acoustical Society of America*, 127.3, s. 1936–1936.

Luo, C. et al. (2017). Cosine Normalization: Using Cosine Similarity Instead of Dot Product in Neural Networks. arXiv:1702.05870.

Maas, A. L. et al. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. *Proceedings of the International Conference on Machine Learning (ICML)*. Atlanta, USA. PMLR, s. 6.

Maaten, L. van der & G. Hinton (2008). Visualizing Data Using t-SNE. *Journal of Machine Learning Research*, 9 (Nov), s. 2579–2605.

Mallat, S. (2011). Group Invariant Scattering. arXiv:1101.2286.

Marr, B. (2015). *Big Data: 20 Mind-Boggling Facts Everyone Must Read*.
URL: https://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/.

Martens, J. (2010). Deep Learning via Hessian-Free Optimization. *Proceedings of the International Conference on Machine Learning (ICML)*. Hajfa, Izrael. PMLR, s. 735–742.

Maxime, J. et al. (2014). Sound Representation and Classification Benchmark for Domestic Robots. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Hongkong, Chiny. IEEE, s. 6285–6292.

McComb, K. et al. (2009). The Cry Embedded within the Purr. *Current Biology*, 19.13, R507–R508.

McCulloch, W. S. & W. Pitts (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Bulletin of Mathematical Biophysics*, 5.4, s. 115–133.

McInnes, L. & J. Healy (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv:1802.03426.

Mesaros, A. et al. (2010). Acoustic Event Detection in Real Life Recordings. *Proceedings of the European Signal Processing Conference (EUSIPCO)*. Aalborg, Dania. IEEE, s. 1267–1271.

Mesaros, A. et al. (2016a). Metrics for Polyphonic Sound Event Detection. *Applied Sciences*, 6.6, s. 17.

Mesaros, A. et al. (2016b). TUT Database for Acoustic Scene Classification and Sound Event Detection. *Proceedings of the European Signal Processing Conference (EUSIPCO)*. Budapeszt, Węgry. IEEE, s. 1128–1132.

Mesaros, A. et al. (2018). Detection and Classification of Acoustic Scenes and Events: Outcome of the DCASE 2016 Challenge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26.2, s. 379–393.

Mesaros, A. et al. (2013). Analysis of Acoustic-Semantic Relationship for Diversely Annotated Real-World Audio Data. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Kanada. IEEE, s. 813–817.

Minsky, M. & S. Papert (1969). *Perceptrons*. MIT.

Mirza, M. & S. Osindero (2014). Conditional Generative Adversarial Nets. arXiv:1411.1784.

Mitrović, D. et al. (2010). Features for Content-Based Audio Retrieval. *Advances in Computers*. T. 78. Elsevier, s. 71–150.

Mnih, V. et al. (2015). Human-Level Control through Deep Reinforcement Learning. *Nature*, 518.7540, s. 529–533.

Mporas, I. et al. (2013). Integration of Temporal Contextual Information for Robust Acoustic Recognition of Bird Species from Real-Field Data. *International Journal of Intelligent Systems and Applications*, 5.7, s. 9.

Muhammad, G. et al. (2010). Environment Recognition Using Selected MPEG-7 Audio Features and Mel-Frequency Cepstral Coefficients. *Proceedings of the International Conference on Digital Telecommunications (ICDT)*. Ateny, Grecja. IARIA, s. 11–16.

Mun, S. et al. (2017). Deep Neural Network Based Learning and Transferring Mid-Level Audio Features for Acoustic Scene Classification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Nowy Orlean, USA. IEEE, s. 796–800.

Mydlarz, C. et al. (2017). Noise Monitoring and Enforcement in New York City Using a Remote Acoustic Sensor Network. *Proceedings of the INTER-NOISE and NOISE-CON Congress and Conference*. Hongkong, Chiny. INCE, s. 5509–5520.

Nakamura, S. et al. (2000). Acoustical Sound Database in Real Environments for Sound Scene Understanding and Hands-Free Speech Recognition. *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*. Ateny, Grecja. ILSP, s. 4.

Nesterov, Y. (1983). A Method of Solving a Convex Programming Problem with Convergence Rate $O(1/K^2)$. *Soviet Mathematics Doklady*. T. 27, s. 372–376.

ORNIS (2016). *European Commission: The Birds Directive*.
URL: http://ec.europa.eu/environment/nature/legislation/birdsdirective/.

O'Shaughnessy, D. (1999). *Speech Communications: Human and Machine*. 2nd Edition. Wiley-IEEE Press.

Parascandolo, G. et al. (2016). Recurrent Neural Networks for Polyphonic Sound Event Detection in Real Life Recordings. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Szanghaj, Chiny. IEEE, s. 6440–6444.

Parascandolo, G. et al. (2017). Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25.6, s. 1291–1303.

Parker, D. B. (1985). *Learning-Logic: Casting the Cortex of the Human Brain in Silicon*. Technical Report 47. MIT, CCREMS, s. 47.

Pfau, D. & O. Vinyals (2016). Connecting Generative Adversarial Networks and Actor-Critic Methods. arXiv:1610.01945.

Phan, H. et al. (2016). Robust Audio Event Recognition with 1-Max Pooling Convolutional Neural Networks. arXiv:1604.06338.

Piczak, K. J. (2015a). Environmental Sound Classification with Convolutional Neural Networks. *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. Boston, USA. IEEE, s. 6.

Piczak, K. J. (2015b). ESC: Dataset for Environmental Sound Classification. *Proceedings of the ACM International Conference on Multimedia (ACMMM)*. Brisbane, Australia. ACM, s. 1015–1018.

Piczak, K. J. (2016). Recognizing Bird Species in Audio Recordings Using Deep Convolutional Neural Networks. *Working Notes of CLEF*. Évora, Portugalia. CLEF, s. 534–543.

Piczak, K. J. (2017). The Details That Matter: Frequency Resolution of Spectrograms in Acoustic Scene Classification. *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. Monachium, Niemcy. TUT, s. 103–107.

Pillos, A. et al. (2016). A Real-Time Environmental Sound Recognition System for the Android OS. *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. Budapeszt, Węgry. TUT, s. 5.

Plinge, A. et al. (2014). A Bag-of-Features Approach to Acoustic Event Detection. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florencja, Włochy. IEEE, s. 3704–3708.

Polyak, B. T. (1964). Some Methods of Speeding up the Convergence of Iteration Methods. *USSR Computational Mathematics and Mathematical Physics*, 4.5, s. 1–17.

Pons, J. et al. (2016). Experimenting with Musically Motivated Convolutional Neural Networks. *Proceedings of the IEEE International Workshop on Content-Based Multimedia Indexing (CBMI)*. Bukareszt, Rumunia. IEEE, s. 6.

Pu, Y. et al. (2016). Variational Autoencoder for Deep Learning of Images, Labels and Captions. *Advances in Neural Information Processing Systems (NIPS)*. Barcelona, Hiszpania. NIPS Foundation, s. 2352–2360.

Rabaoui, A. et al. (2008). Using One-Class SVMs and Wavelets for Audio Surveillance. *IEEE Transactions on Information Forensics and Security*, 3.4, s. 763–775.

Radford, A. et al. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv:1511.06434.

Rakotomamonjy, A. & G. Gasso (2015). Histogram of Gradients of Time-Frequency Representations for Audio Scene Classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23.1, s. 142–153.

Ravanelli, M. et al. (2014). Audio Concept Classification with Hierarchical Deep Neural Networks. *Proceedings of the European Signal Processing Conference (EUSIPCO)*. Lizbona, Portugalia. IEEE, s. 606–610.

Razavian, A. S. et al. (2014). CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. arXiv:1403.6382.

Reynolds, D. A. (1994). Experimental Evaluation of Features for Robust Speaker Identification. *IEEE Transactions on Speech and Audio Processing*, 2.4, s. 639–643.

Richard, G. et al. (2017). Introduction to the Special Section on Sound Scene and Event Analysis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25.6, s. 1169–1171.

Rosenblatt, F. (1957). *The Perceptron, a Perceiving and Recognizing Automaton*. Cornell Aeronautical Laboratory.

Ruder, S. (2016). An Overview of Gradient Descent Optimization Algorithms. arXiv:1609.04747.

Rumelhart, D. E. et al. (1986). Learning Internal Representations by Error Propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. T. 1. MIT Press, s. 318–362.

Rumelhart, D. E. et al. (1988). Learning Representations by Back-Propagating Errors. *Cognitive modeling*, 5.3.

Russakovsky, O. et al. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115.3, s. 211–252.

Sailor, H. B. et al. (2017). Unsupervised Filterbank Learning Using Convolutional Restricted Boltzmann Machine for Environmental Sound Classification. *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*. Sztokholm, Szwecja. ISCA, s. 3107–3111.

Sainath, T. N. et al. (2013). Deep Convolutional Neural Networks for LVCSR. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Kanada. IEEE, s. 8614–8618.

Salamon, J. & J. P. Bello (2015). Unsupervised Feature Learning for Urban Sound Classification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brisbane, Australia. IEEE, s. 171–175.

Salamon, J. & J. P. Bello (2017). Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. *IEEE Signal Processing Letters*, 24.3, s. 279–283.

Salamon, J. et al. (2014). A Dataset and Taxonomy for Urban Sound Research. *Proceedings of the ACM International Conference on Multimedia (ACMMM)*. Orlando, USA. ACM, s. 1041–1044.

Salimans, T. & D. P. Kingma (2016). Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. *Advances in Neural Information Processing Systems (NIPS)*. Barcelona, Hiszpania. NIPS Foundation, s. 901–909.

Salimans, T. et al. (2016). Improved Techniques for Training GANs. *Advances in Neural Information Processing Systems (NIPS)*. Barcelona, Hiszpania. NIPS Foundation, s. 2234–2242.

Saxe, A. M. et al. (2013). Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks. arXiv:1312.6120.

Schölkopf, B. et al. (2000). Support Vector Method for Novelty Detection. *Advances in Neural Information Processing Systems (NIPS)*. Denver, USA. NIPS Foundation, s. 582–588.

Schuller, B. et al. (2010). The INTERSPEECH 2010 Paralinguistic Challenge. *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*. Makuhari, Japonia. ISCA, s. 2794–2797.

Sermanet, P. et al. (2013). OverFeat: Integrated Recognition, Localization and Detection Using Convolutional Networks. arXiv:1312.6229.

Sevilla, A. et al. (2017). Audio Bird Classification with Inception-v4 Extended with Time and Time-Frequency Attention Mechanisms. *Working Notes of CLEF*. Dublin, Irlandia. CLEF, s. 8.

Shao, Y. et al. (2009). An Auditory-Based Feature for Robust Speech Recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Tajpej, Tajwan. IEEE, s. 4625–4628.

Shaw, J. (2014). *Why Big Data Is a Big Deal*.
URL: http://harvardmagazine.com/2014/03/why-big-data-is-a-big-deal.

Shi, W. et al. (2016). Is the Deconvolution Layer the Same as a Convolutional Layer? arXiv:1609.07009.

Simonyan, K. & A. Zisserman (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556.

Singh, S. et al. (2016). Swapout: Learning an Ensemble of Deep Architectures. *Advances in Neural Information Processing Systems (NIPS)*. Barcelona, Hiszpania. NIPS Foundation, s. 28–36.

Smaragdis, P. & J. C. Brown (2003). Non-Negative Matrix Factorization for Polyphonic Music Transcription. *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz, USA. IEEE, s. 177–180.

Sottek, R. & K. Genuit (2005). Models of Signal Processing in Human Hearing. *AEÜ - International Journal of Electronics and Communications*, 59.3, s. 157–165.

Sprengel, E. et al. (2016). Audio Based Bird Species Identification Using Deep Learning Techniques. *Working Notes of CLEF*. Évora, Portugal. CLEF, s. 547–559.

Stork, J. A. et al. (2012). Audio-Based Human Activity Recognition Using Non-Markovian Ensemble Voting. *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. Paryż, Francja. IEEE, s. 509–514.

Stowell, D. (2015). BirdCLEF 2015 Submission: Unsupervised Feature Learning from Audio. *Working Notes of CLEF*. Tuluza, Francja. CLEF, s. 3.

Stowell, D. & M. D. Plumbley (2010). *Birdsong and C4DM: A Survey of UK Birdsong and Machine Recognition for Music Researchers*. Technical Report C4DM-TR-09-12. Centre for Digital Music, QMUL.

Stowell, D. & M. D. Plumbley (2013). An Open Dataset for Research on Audio Field Recording Archives: Freefield1010. arXiv:1309.5275.

Stowell, D. & M. D. Plumbley (2014a). Audio-Only Bird Classification Using Unsupervised Feature Learning. *Working Notes of CLEF*. Sheffield, Wielka Brytania. CLEF, s. 12.

Stowell, D. & M. D. Plumbley (2014b). Automatic Large-Scale Classification of Bird Sounds Is Strongly Improved by Unsupervised Feature Learning. *PeerJ*, 2, e488.

Stowell, D. et al. (2016). Bird Detection in Audio: A Survey and a Challenge. arXiv:1608.03417.

Stowell, D. et al. (2015). Detection and Classification of Acoustic Scenes and Events. *IEEE Transactions on Multimedia*, 17.10, s. 1733–1746.

Sussillo, D. & L. F. Abbott (2014). Random Walks: Training Very Deep Nonlinear Feed-Forward Networks with Smart Initialization. arXiv:1412.6558.

Sutskever, I. et al. (2013). On the Importance of Initialization and Momentum in Deep Learning. *Proceedings of the International Conference on Machine Learning (ICML)*. Atlanta, USA. PMLR, s. 1139–1147.

Szegedy, C. et al. (2015). Going Deeper with Convolutions. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, USA. CVF, s. 1–9.

Szegedy, C. et al. (2016). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, USA. CVF, s. 2818–2826.

Szegedy, C. et al. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*. San Francisco, USA. AAAI, s. 4278–4284.

Tak, R. N. et al. (2017). Novel Phase Encoded Mel Filterbank Energies for Environmental Sound Classification. *Proceedings of the International Conference on Pattern Recognition and Machine Intelligence (PReMI)*. Kalkuta, Indie. Springer, s. 317–325.

Takahashi, N. et al. (2016). Deep Convolutional Neural Networks and Data Augmentation for Acoustic Event Detection. arXiv:1604.07160.

Tax, T. M. S. et al. (2017). Utilizing Domain Knowledge in End-to-End Audio Processing. arXiv:1712.00254.

Todisco, M. et al. (2016). A New Feature for Automatic Speaker Verification Anti-Spoofing: Constant Q Cepstral Coefficients. *Proceedings of the Speaker and Language Recognition Workshop (ODYSSEY)*. Bilbao, Hiszpania. UPV/EHU, s. 249–252.

Tokozume, Y. & T. Harada (2017). Learning Environmental Sounds with End-to-End Convolutional Neural Network. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Nowy Orlean, USA. IEEE, s. 2721–2725.

Tokozume, Y. et al. (2017). Learning from Between-Class Examples for Deep Sound Recognition. arXiv:1711.10282.

Tompson, J. et al. (2015). Efficient Object Localization Using Convolutional Networks. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, USA. CVF, s. 648–656.

Tóth, B. P. & B. Czeba (2016). Convolutional Neural Networks for Large-Scale Bird Song Classification in Noisy Environment. *Working Notes of CLEF*. Évora, Portugal. CLEF, s. 560–568.

Valero, X. & F. Alias (2012). Gammatone Cepstral Coefficients: Biologically Inspired Features for Non-Speech Audio Classification. *IEEE Transactions on Multimedia*, 14.6, s. 1684–1689.

Valero, X. & F. Alías (2012). Gammatone Wavelet Features for Sound Classification in Surveillance Applications. *Proceedings of the European Signal Processing Conference (EUSIPCO)*. Bukareszt, Rumunia. IEEE, s. 1658–1662.

Van den Oord, A. et al. (2013). Deep Content-Based Music Recommendation. *Advances in Neural Information Processing Systems (NIPS)*. Lake Tahoe, USA. NIPS Foundation, s. 2643–2651.

Van Grootel, M. W. W. et al. (2009). DARES-G1: Database of Annotated Real-World Everyday Sounds. *Proceedings of the NAG/DAGA International Conference on Acoustics*. Rotterdam, Holandia. NAG/DAGA, s. 4.

Virtanen, T. et al. (2018). *Computational Analysis of Sound Scenes and Events*. Springer International Publishing.

Wang, J.-C. et al. (2006). Environmental Sound Classification Using Hybrid SVM/KNN Classifier and MPEG-7 Audio Low-Level Descriptor. *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. Vancouver, Kanada. IEEE, s. 1731–1735.

Werbos, P. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Harvard University.

Widrow, B. & M. A. Lehr (1990). 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. *Proceedings of the IEEE*, 78.9, s. 1415–1442.

Wimmer, J. et al. (2013). Sampling Environmental Acoustic Recordings to Determine Bird Species Richness. *Ecological Applications*, 23.6, s. 1419–1428.

Yang, Y.-H. & H. H. Chen (2011). *Music Emotion Recognition*. CRC Press.

Yosinski, J. et al. (2014). How Transferable Are Features in Deep Neural Networks? arXiv:1411.1792.

Young, S. et al. (2002). *The HTK Book*. Version 3.2. Cambridge University Engineering Department.

Zajdel, W. et al. (2007). CASSANDRA: Audio-Video Sensor Fusion for Aggression Detection. *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. Londyn, Wielka Brytania. IEEE, s. 200–205.

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. arXiv:1212.5701.

Zeiler, M. D. & R. Fergus (2014). Visualizing and Understanding Convolutional Networks. *Lecture Notes in Computer Science*, 8689, s. 818–833.

Zeiler, M. D. et al. (2011). Adaptive Deconvolutional Networks for Mid and High Level Feature Learning. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Barcelona, Hiszpania. IEEE, s. 2018–2025.

Zhang, H. et al. (2015). Robust Sound Event Recognition Using Convolutional Neural Networks. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brisbane, Australia. IEEE, s. 559–563.

Zhang, Y. & B. C. Wallace (2017). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*. Tajpej, Tajwan. AFNLP, s. 253–263.

Zhu, B. et al. (2018). Environmental Sound Classification Based on Multi-Temporal Resolution Convolutional Neural Network Combining with Multi-Level Features. arXiv:1805.09752.

Zhu, J.-Y. et al. (2016). Generative Visual Manipulation on the Natural Image Manifold. *Proceedings of the European Conference on Computer Vision (ECCV)*. Amsterdam, Holandia. Springer, s. 597–613.

Załącznik A

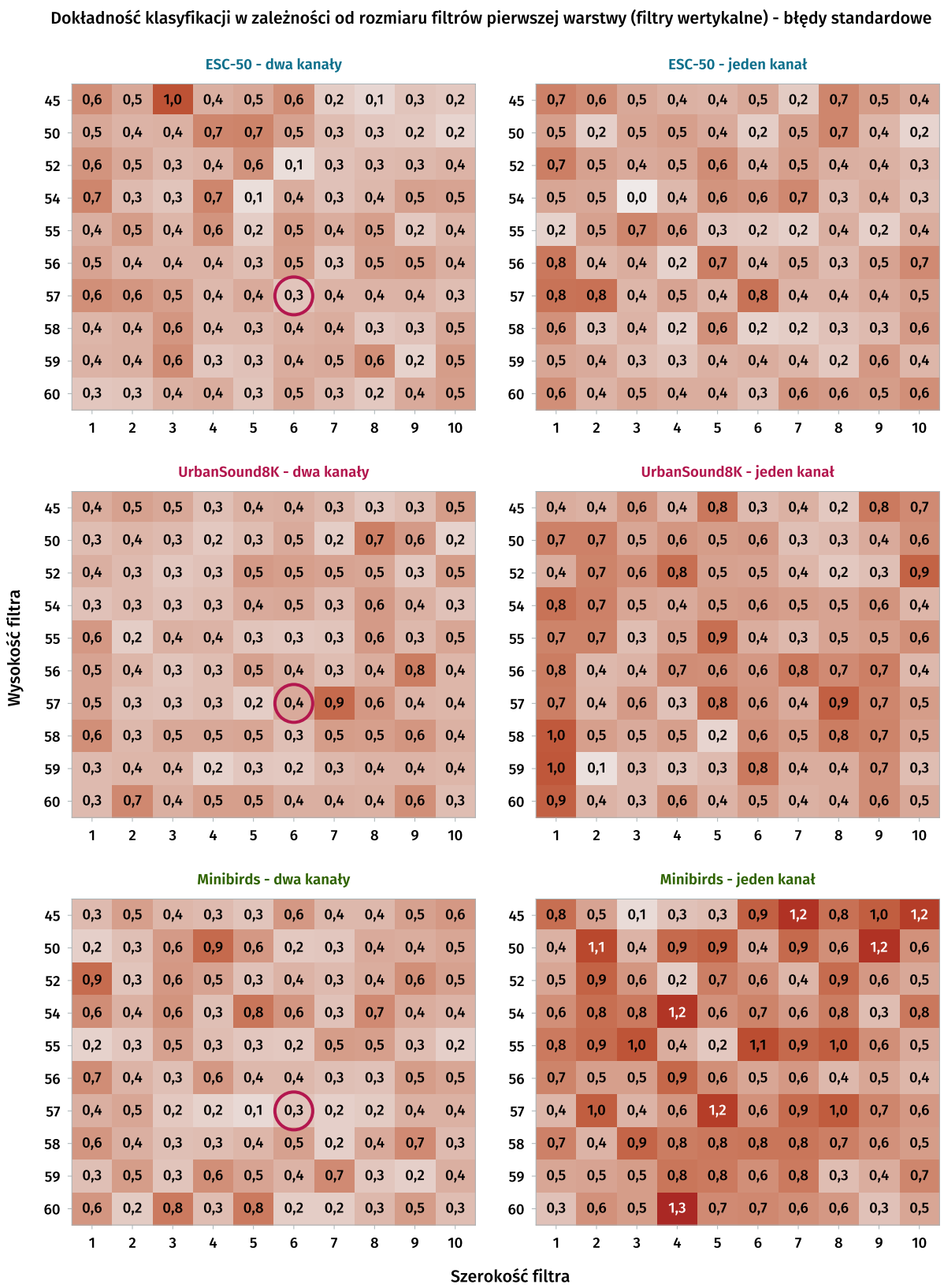
Zestawienie błędów standardowych dla analizy wrażliwości

Eksperymenty opisane w rozdziale 7 zostały przeprowadzone pięciokrotnie z różnymi wartościami ziarna losowego, aby uwzględnić aspekt niepewności kształtowania się wyników w zależności od obranego punktu inicjalizacji wag modeli czy też z tytułu występowania niewielkich rozbieżności dla symulacji przeprowadzanych na różnych architekturach kart graficznych.

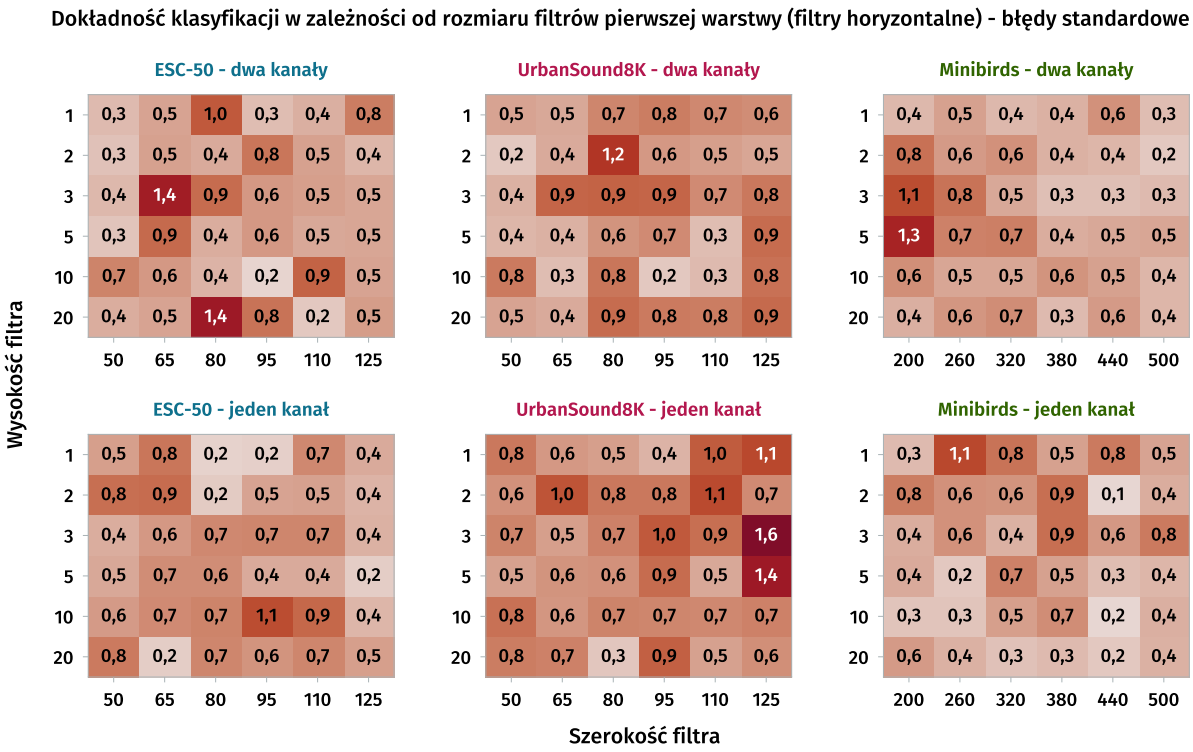
Załączone zestawienia przedstawiają wartości błędu standardowego średniej dla poszczególnych eksperymentów według wzoru:

$$SEM = \frac{\sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}}}{\sqrt{N}} \tag{A.1}$$

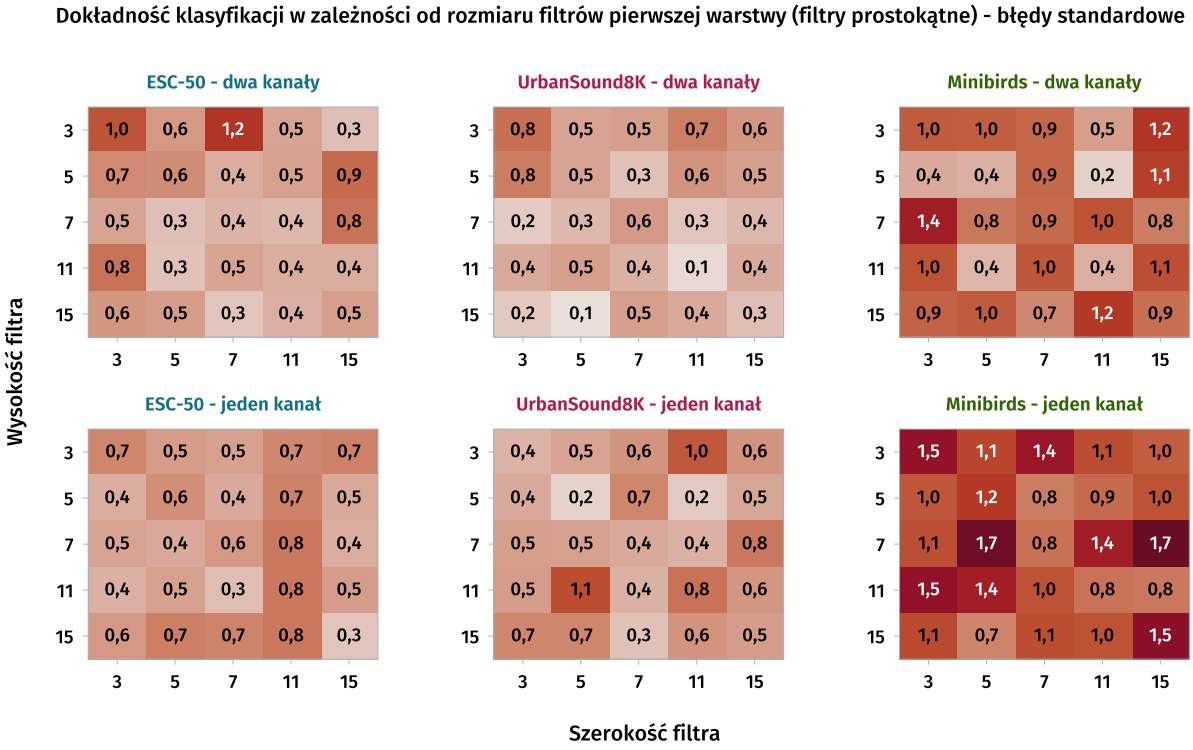
gdzie $N = 5$, $\overline{x}$ jest średnią dokładnością raportowaną w poszczególnych komórkach zestawień w rozdziale 7, natomiast x_i jest dokładnością otrzymywaną dla kolejnych powtórzeń eksperymentów.



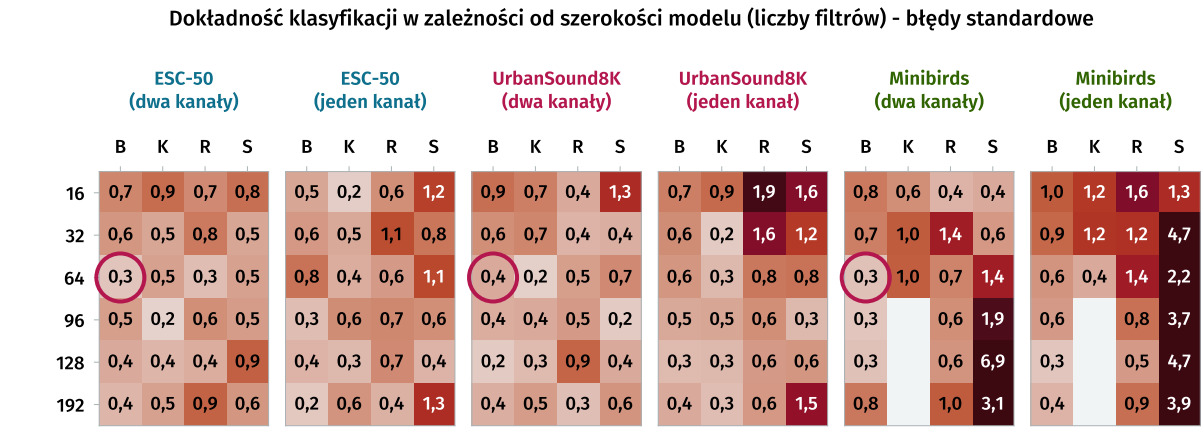
Rys. A.1: Wartości błędu standardowego dla zestawienia z rysunku 7.6 (w procentach).



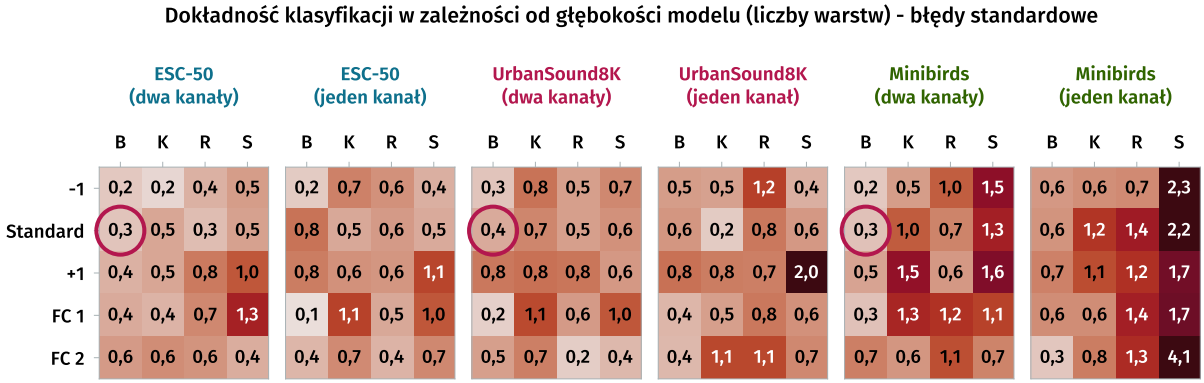
Rys. A.2: Wartości błędu standardowego dla zestawienia z rysunku 7.7 (w procentach).



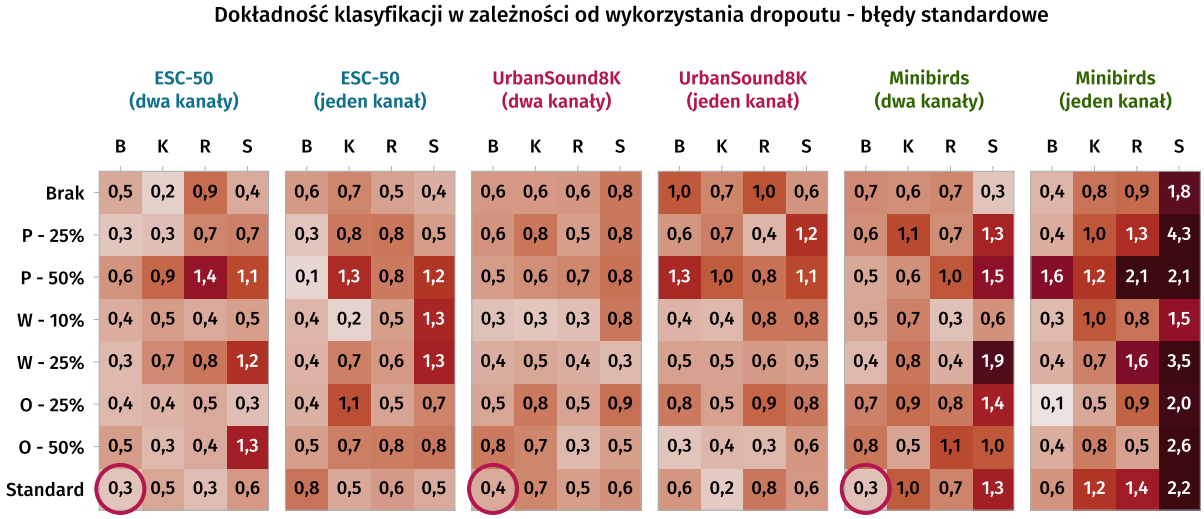
Rys. A.3: Wartości błędu standardowego dla zestawienia z rysunku 7.8 (w procentach).



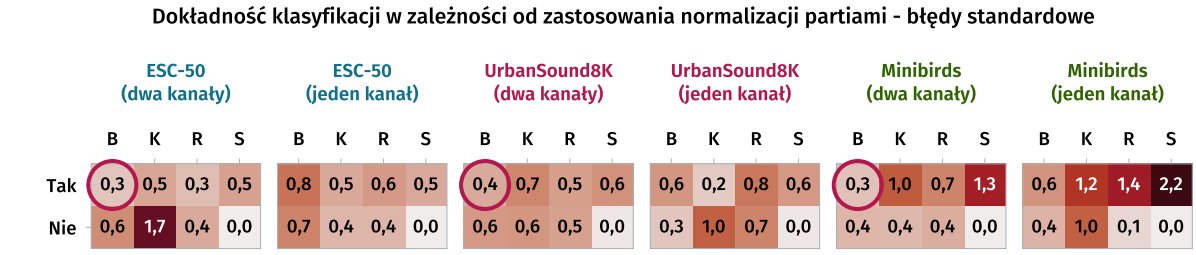
Rys. A.4: Wartości błędu standardowego dla zestawienia z rysunku 7.9 (w procentach).



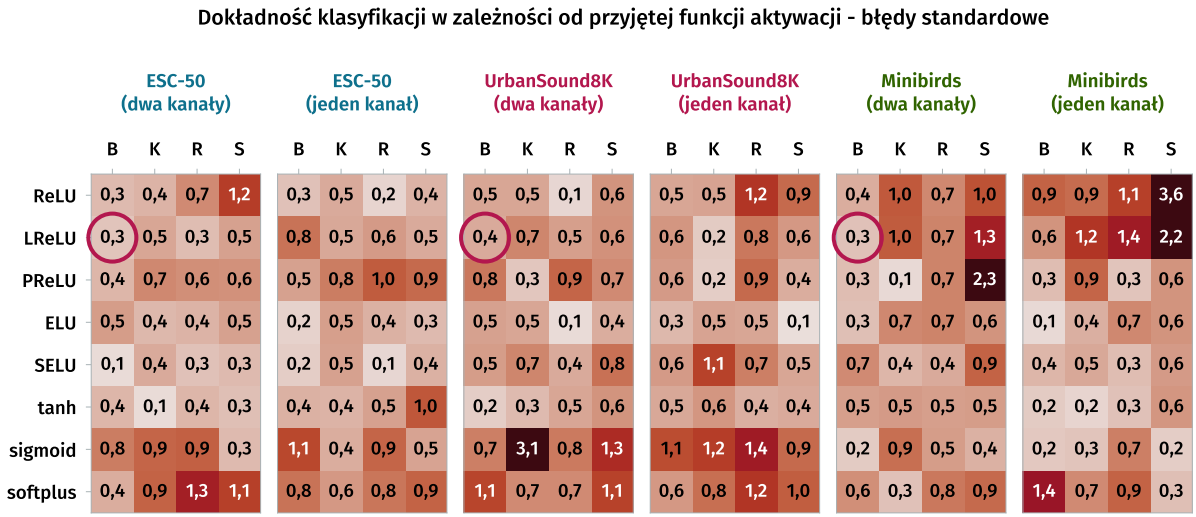
Rys. A.5: Wartości błędu standardowego dla zestawienia z rysunku 7.10 (w procentach).



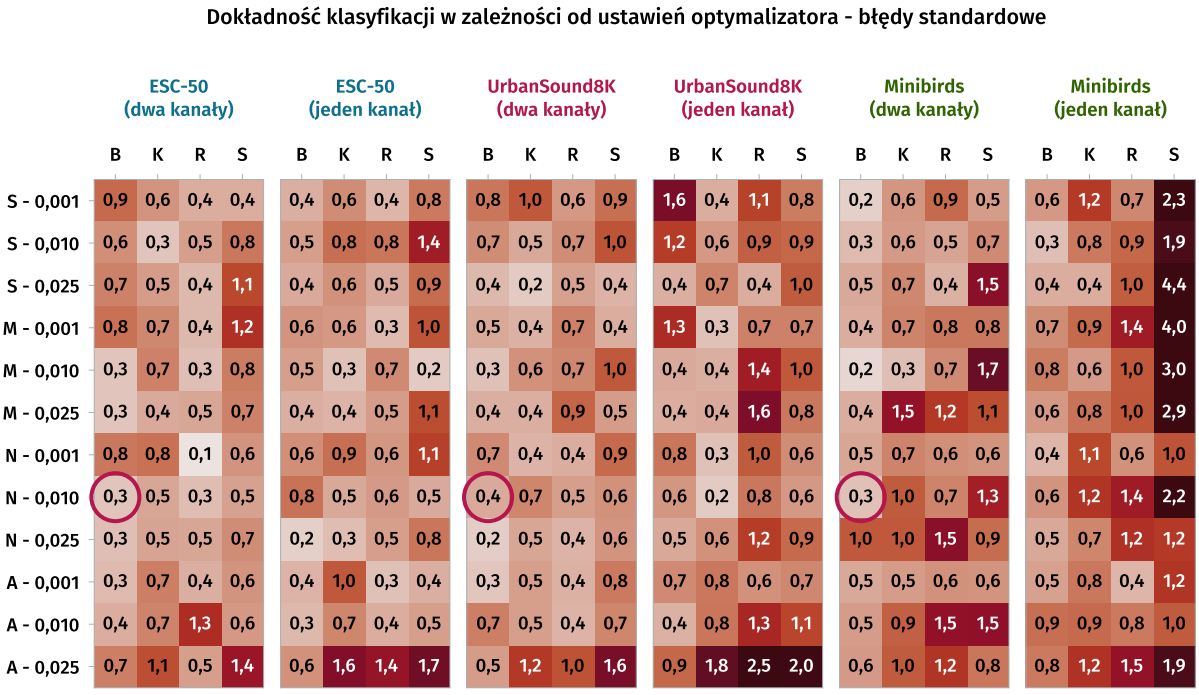
Rys. A.6: Wartości błędu standardowego dla zestawienia z rysunku 7.11 (w procentach).



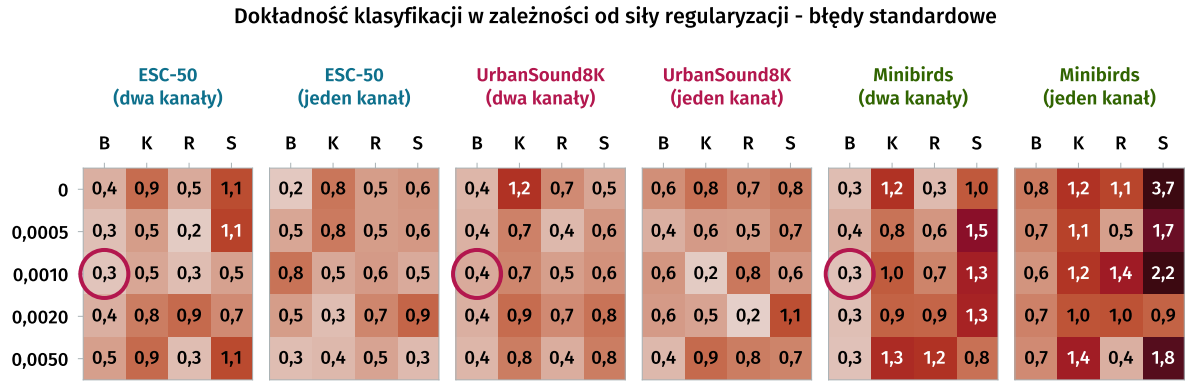
Rys. A.7: Wartości błędu standardowego dla zestawienia z rysunku 7.12 (w procentach).



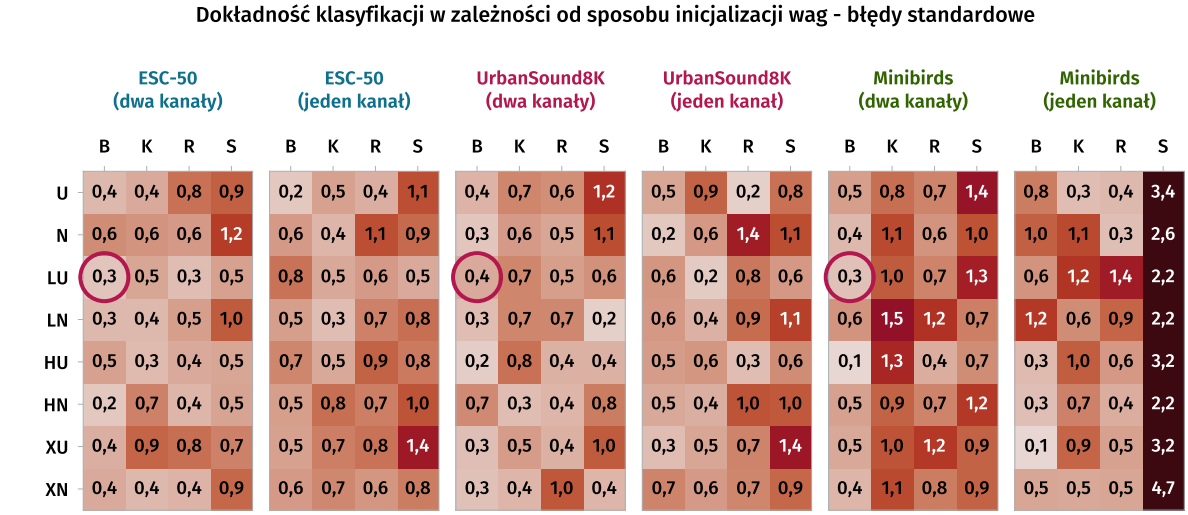
Rys. A.8: Wartości błędu standardowego dla zestawienia z rysunku 7.13 (w procentach).



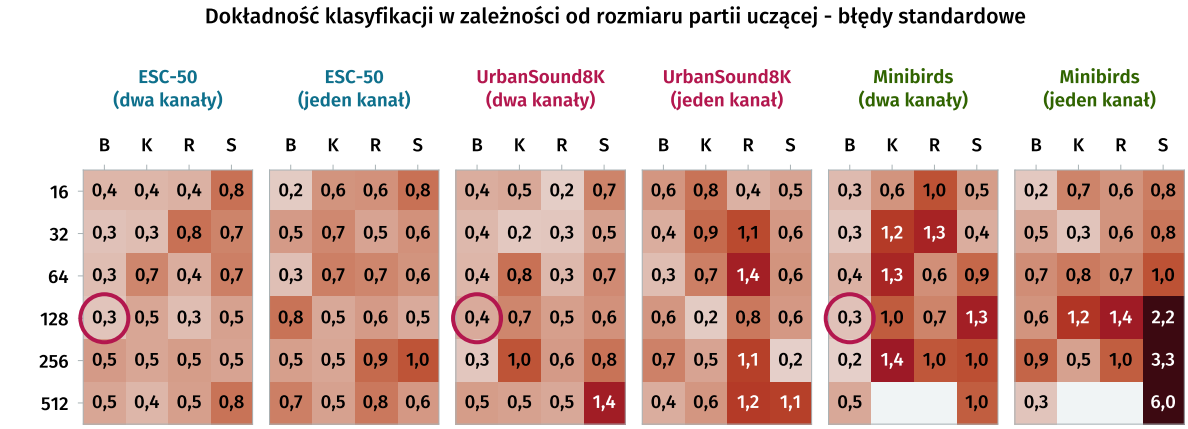
Rys. A.9: Wartości błędu standardowego dla zestawienia z rysunku 7.14 (w procentach).



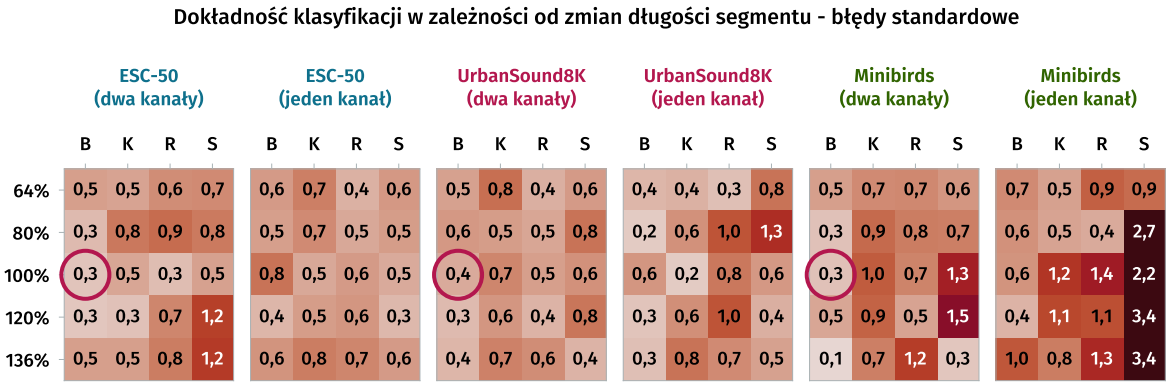
Rys. A.10: Wartości błędu standardowego dla zestawienia z rysunku 7.15 (w procentach).



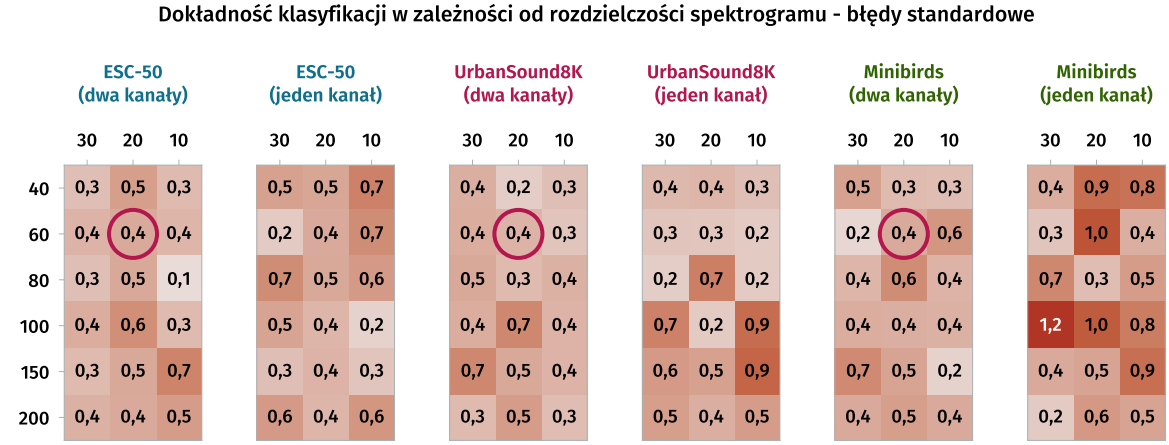
Rys. A.11: Wartości błędu standardowego dla zestawienia z rysunku 7.16 (w procentach).



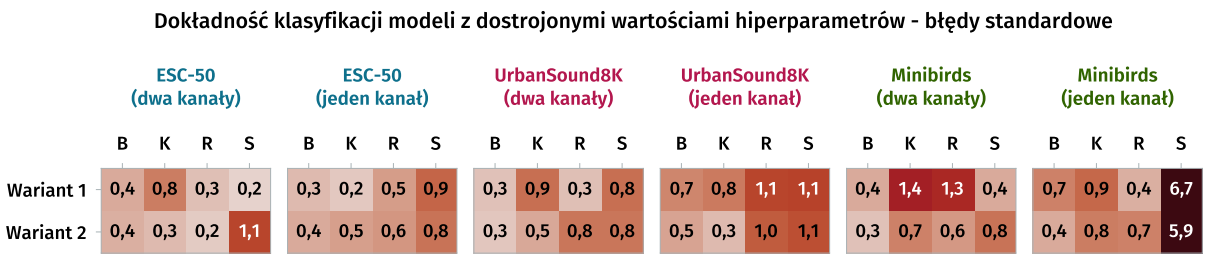
Rys. A.12: Wartości błędu standardowego dla zestawienia z rysunku 7.17 (w procentach).



Rys. A.13: Wartości błędu standardowego dla zestawienia z rysunku 7.18 (w procentach).



Rys. A.14: Wartości błędu standardowego dla zestawienia z rysunku 7.19 (w procentach).



Rys. A.15: Wartości błędu standardowego dla zestawienia z rysunku 7.20 (w procentach).