Uniwersytet Jagielloński w Krakowie

Wydział Fizyki, Astronomii i Informatyki Stosowanej

Piotr Przemielewski

Nr albumu: 1155461

Analiza danych dźwięków miejskich metodami uczenia maszynowego

Praca licencjacka
na kierunku Informatyka

Praca wykonana pod kierunkiem
prof. dr hab. Elżbiety Richter-Wąs
Instytut Fizyki
Zakład Fizyki Gorącej Materii

Kraków 2021

Spis treści

1. Wstęp

Obecnie ponad 55% ludzkości żyje na obszarach zurbanizowanych [1], a do roku 2050 odsetek ten ma wzrosnąć do 68%. W Polsce liczba osób zamieszkująca miasta zwiększała do roku 1990, po którym nastał zastój, trwający blisko 20 lat, objawiający się spadkiem urbanizacji w ciągu ostatnich 10 lat [2]. Jednocześnie 58% Polaków deklaruje, że gdyby mogło wybrać miejsce zamieszkania, chciałoby mieszkać w mieście [3], a Bank Światowy prognozuje, że poziom zurbanizowania będzie sukcesywnie rosnąć.

Badania te rzutują na trend rozmieszczenia geograficznego ludzi w najbliższych dekadach. Wraz ze wzrostem populacji i urbanizacji miasta staną przed wieloma wyzwaniami związanymi z wpływem człowieka na środowisko, zapewnieniem wysokiej jakości pomocy medycznej, przemyślanym zagospodarowaniem terenu, czy skutecznym nadzorowaniem i zabezpieczaniem przestrzeni publicznej. Idee te są bliskie określeniu inteligentnego miasta.

Kai Strittmatter w swojej książce przytacza przykład wykorzystania sieci inteligentnych kamer, które były w stanie rozpoznać twarz poszukiwanego, który wybrał się na koncert i został rozpoznany wśród 60 000 uczestników [4]. Wszystko wskazuje na to, że w ciągu najbliższych lat będziemy słyszeć o kolejnych tego typu przypadkach, a systemy nadzorowania życia publicznego będą się dalej rozwijać.

Systemy nadzoru mogą przybierać odmienne formy i różnić się pozyskiwanymi danymi. W połączeniu stanowią potężne narzędzia, jednak i osobne ich zainstalowanie może służyć jako źródło cennych informacji. Czujniki będące w stanie rozróżnić dźwięki różnych kategorii mogą pozytywnie wpłynąć na bezpieczeństwo i zapewnić szybką reakcję służb w razie potrzeby.

1.1 Cel pracy

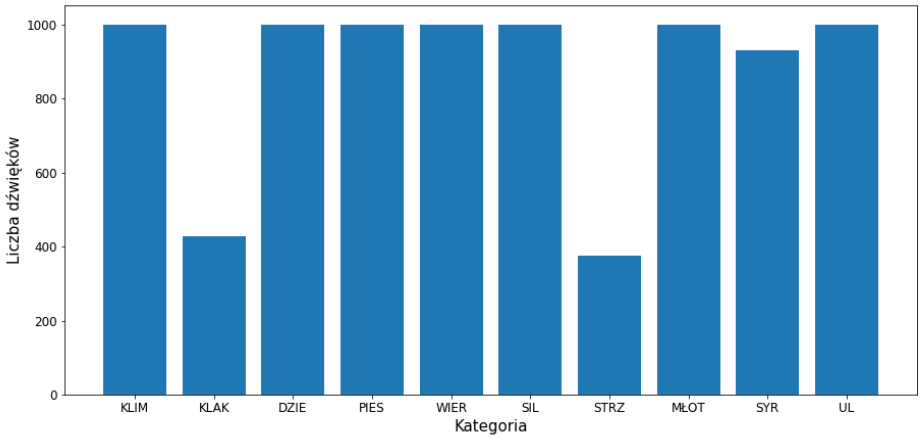
Celem pracy jest dokonanie analizy danych dźwięków miejskich należących do odpowiednich kategorii i przedstawienie modeli sieci neuronowych będących w stanie je sklasyfikować. Zaprezentowane zostaną powszechne transformacje plików audio oraz możliwości ich deformowania. Modele zostaną porównane pod względem uzyskanej dokładności i parametrów z nią powiązanymi.

2. Dane

Zbiór danych użyty w pracy pochodzi ze strony urbansounddataset.weebly.com [5] i zawiera 8732 pliki audio w formacie WAV. Każdy plik należy do jednej z dziesięciu klas (skróty w nawiasach użyte są w późniejszej części pracy):

- klimatyzacja (KLIM),
- klakson (KLAK),
- bawiące się dzieci (DZIE),
- szczekający pies (PIES),
- wiercenie (WIER),
- silnik na biegu jałowym (SIL),
- strzały z broni palnej (STRZ),
- młot pneumatyczny (MŁOT),
- syrena (SYR),
- muzyka uliczna (UL).

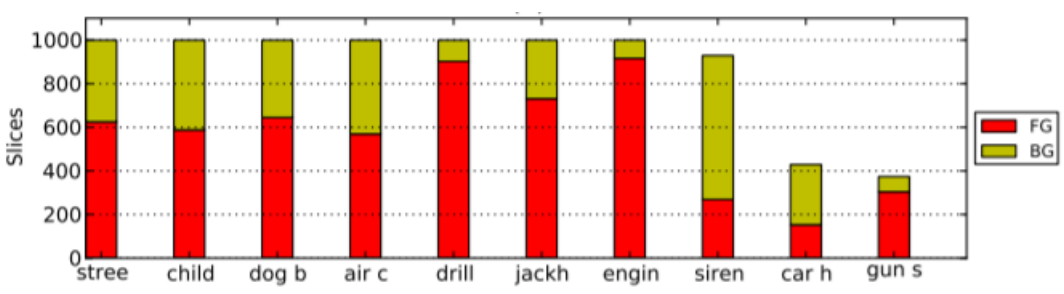
Zbiór danych powstał na bazie dźwięków udostępnionych na stronie freesound.org [6]. Pliki audio rozdzielone są na 10 folderów. Dołączony został plik CSV umożliwiający lokalizację poszczególnych dźwięków oraz wskazujący ich klasę.



Rys.1: Ilościowy rozkład dźwięków należących do poszczególnych klas.

Wszystkie dźwięki, z wyjątkiem klaksonu samochodowego oraz strzałów z broni palnej, występują w zbiorze porównywalną liczbę razy.

Pliki, prócz przyporządkowanych klas, można również rozważać w zależności od odległości od źródła, tj. czy zdarzenie przedstawiane jako dźwięk odbywało się na pierwszym planie (*ang. FR – foreground)*, czy w tle (*ang. BG – background*).



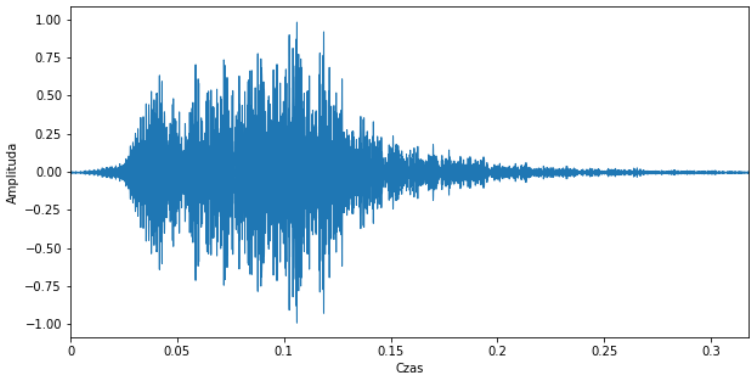
Rys.2: Liczba próbek na klasę w zależność od odległości od źródła. [7]

2.1 Reprezentacja dźwięków

Sygnał dźwiękowy może zostać przedstawiony na różne sposoby, a jego ostateczna forma w znacznej mierze decyduje o poziomie trudności stawianym przed klasyfikatorem.

2.1.1 Postać fali dźwiękowej

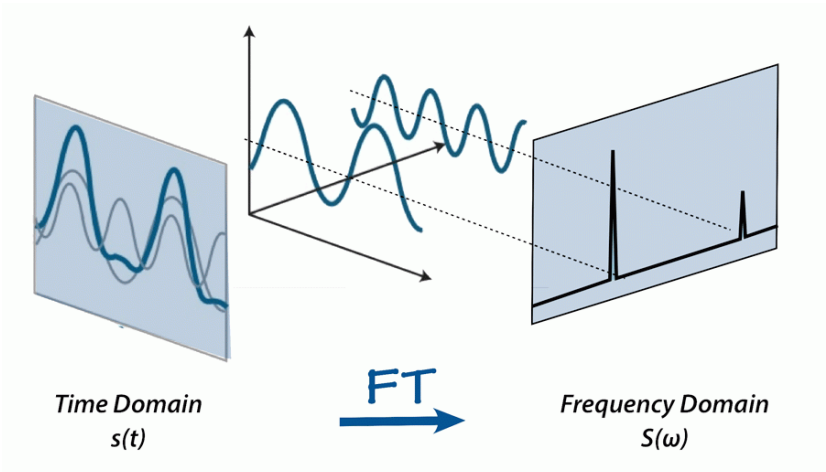
Sygnały dźwiękowe w swojej pierwotnej reprezentacji znajdują się w dziedzinie czasu. Zmiana w ciśnieniu powietrza w danym czasie jest graficznie reprezentowana przez wykres w postaci fali dźwiękowej *(*ang. *audio waveform*). Amplituda znajduje się w przedziale (-1;1) i odnosi się do głośności lub słyszalności. Jest to forma mocno wymagająca względem klasyfikatora, zważając na gęstość danej reprezentacji – przykładowo może zawierać 220 500 wartości dla 5 sekundowego nagrania.



Rys.3: Wykres w postaci fali dźwiękowej przykładowego pliku audio ze zbioru.

2.1.2 Transformacja Fouriera

Sygnały audio złożone są z kilku fal dźwiękowych o pojedynczych częstotliwości. W czasie pobierania próbek sygnałów rejestrowane są jedynie wynikowe amplitudy. Matematycznym sposobem pozwalającym na dekompozycję pojedynczego sygnału do oddzielnych częstotliwości i ich amplitud jest **transformacja Fouriera**. Zachodzi więc przejście z dziedziny czasu na dziedzinę częstotliwości.



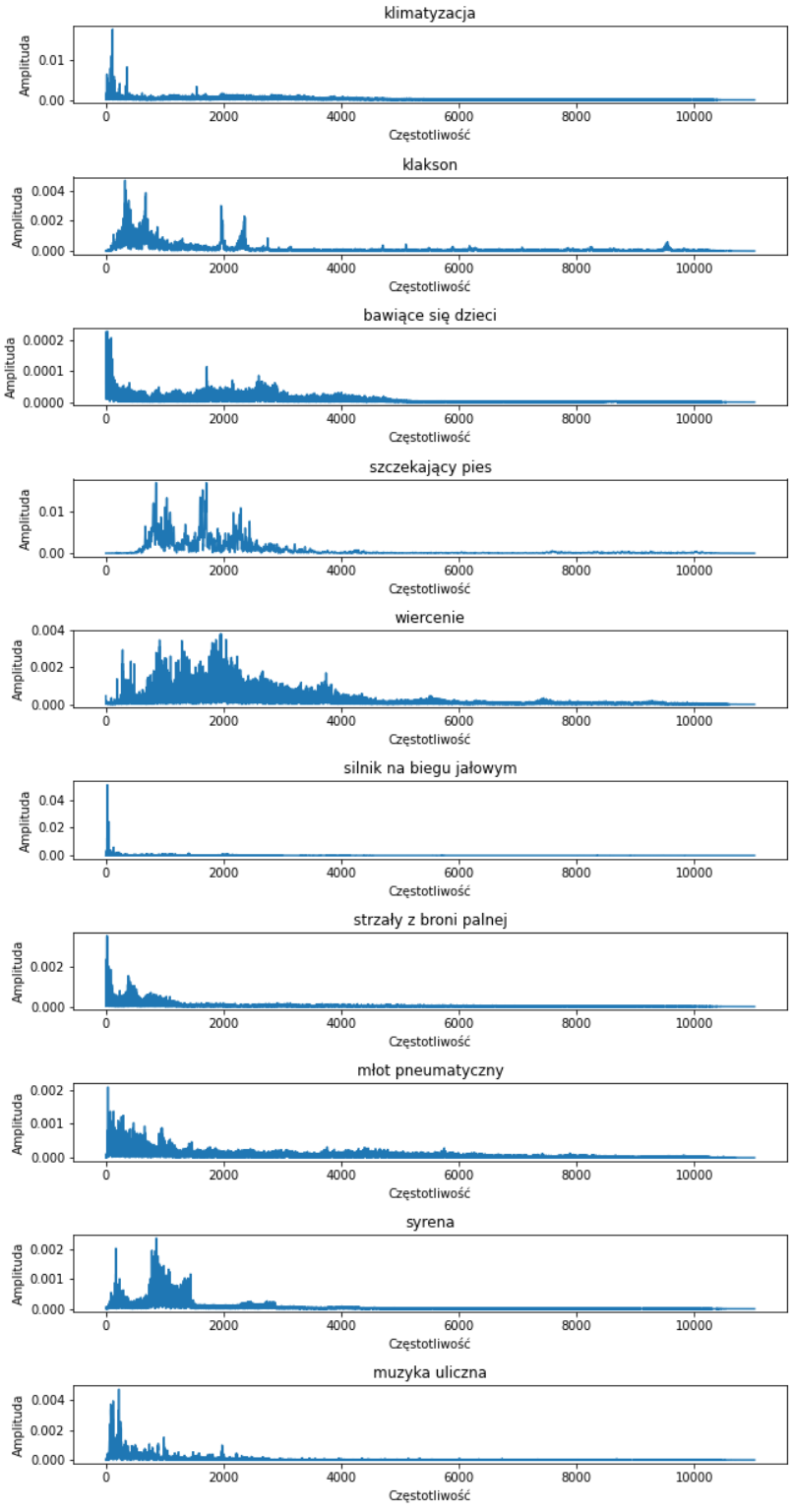
Rys.4: Reprezentacja sygnałów audio w trójwymiarowej przestrzeni. [8]

Algorytmem pozwalającym na obliczenie transformaty Fouriera jest **szybka transformacja Fouriera** (ang. *fast Fourier tranform (FFT)*).

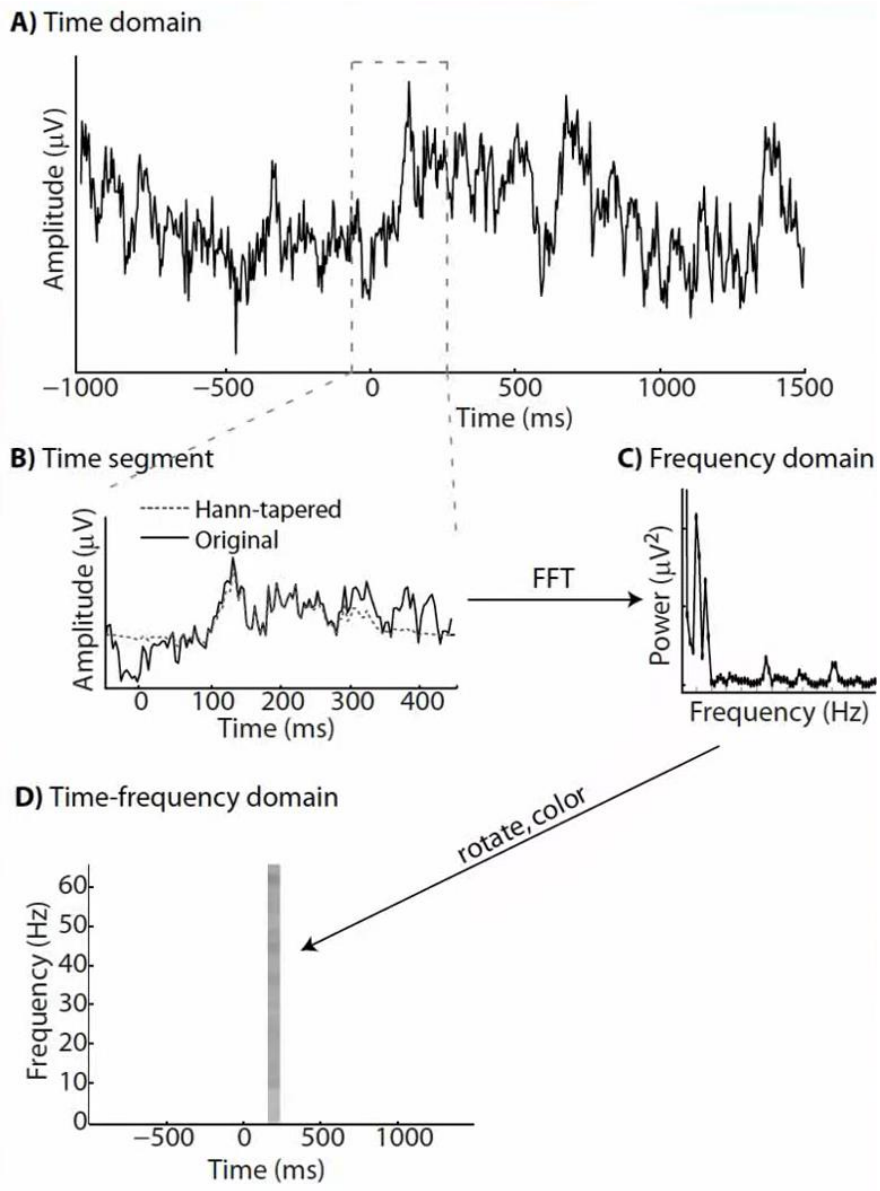
2.1.3 Spektrogram

Transformacja Fouriera pozwala na analizowanie częstotliwości sygnału i daje ogląd na sygnały niezmieniające się w czasie lub zmieniające się niewiele. Takie sytuacje są mniejszością, szczególnie w przypadku muzyki, czy ludzkiej mowy, które nazywane są sygnałami nieokresowymi.

Zamiast stosować transformację Fouriera na całym sygnale, stosuje się ją na kolejnych segmentach sygnału. Sekwencję takich transformacji nazywa się krótkoczasową transformatą Fouriera (ang. *STFT – Short-Time Fourier Transform*).



Rys.5: Krótkoczasowa transformata Fouriera dla przykładowych dźwięków z poszczególnych klas.



Rys 6: Przebieg procesu powstawania spektrogramu. [9]

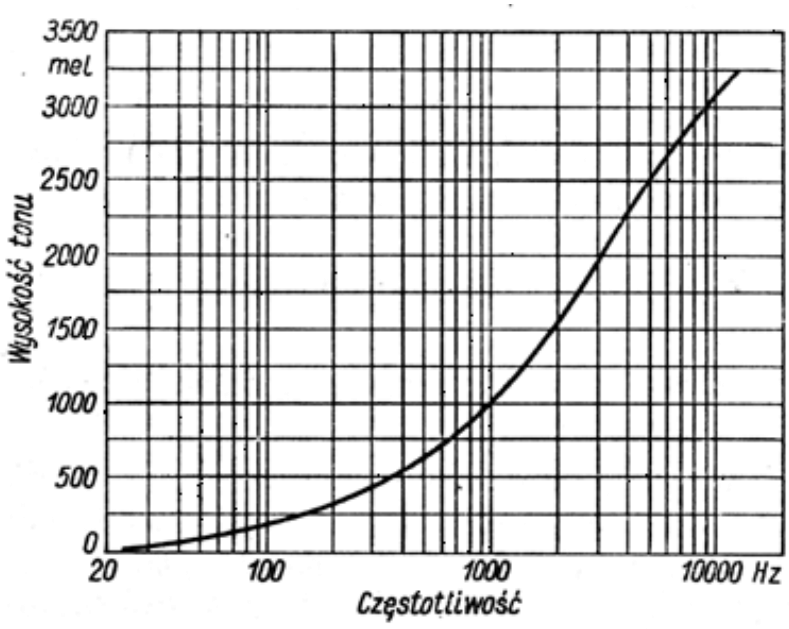
Rys.6 można rozumieć w następujący sposób:

1. Wybierany jest fragment sygnału, który będzie analizowany.
2. Za pomocą szybkiej transformacji Fouriera obliczana jest transformata Fouriera dla wskazanego segmentu.
3. Skala amplitudy jest zmieniana w skalę mocy.
4. Uzyskane widmo jest obracane w taki sposób, że częstotliwość stanowi wartość y na osi, środkowa wartość czasu na osi wycinka sygnału stanowi wartość x na osi, natomiast moc, stanowiącą trzeci wymiar, dla określonej częstotliwości w określonym czasie, reprezentuje się za pomocą kolorów.

W taki sposób otrzymuje się jedno pasmo na wykresie **spektrogramu**. Przesuwając okno wyboru segmentu o określoną jednostkę czasową i postępując zgodnie z określonymi krokami sukcesywnie uzupełnia się pozostałe fragmenty wykresu.

2.1.4 Spektrogram w skali melowej

Mel stanowi jednostkę wysokości tonu, umożliwiającą określenie położenia tonu na skali częstotliwości. Skala melowa określa stosunek subiektywnej skali wysokości tonu i obiektywnej skali częstotliwości [10].



Rys.7: Skala melowa. [10]

Zależności między skalą mel i Hz określa się wzorem:

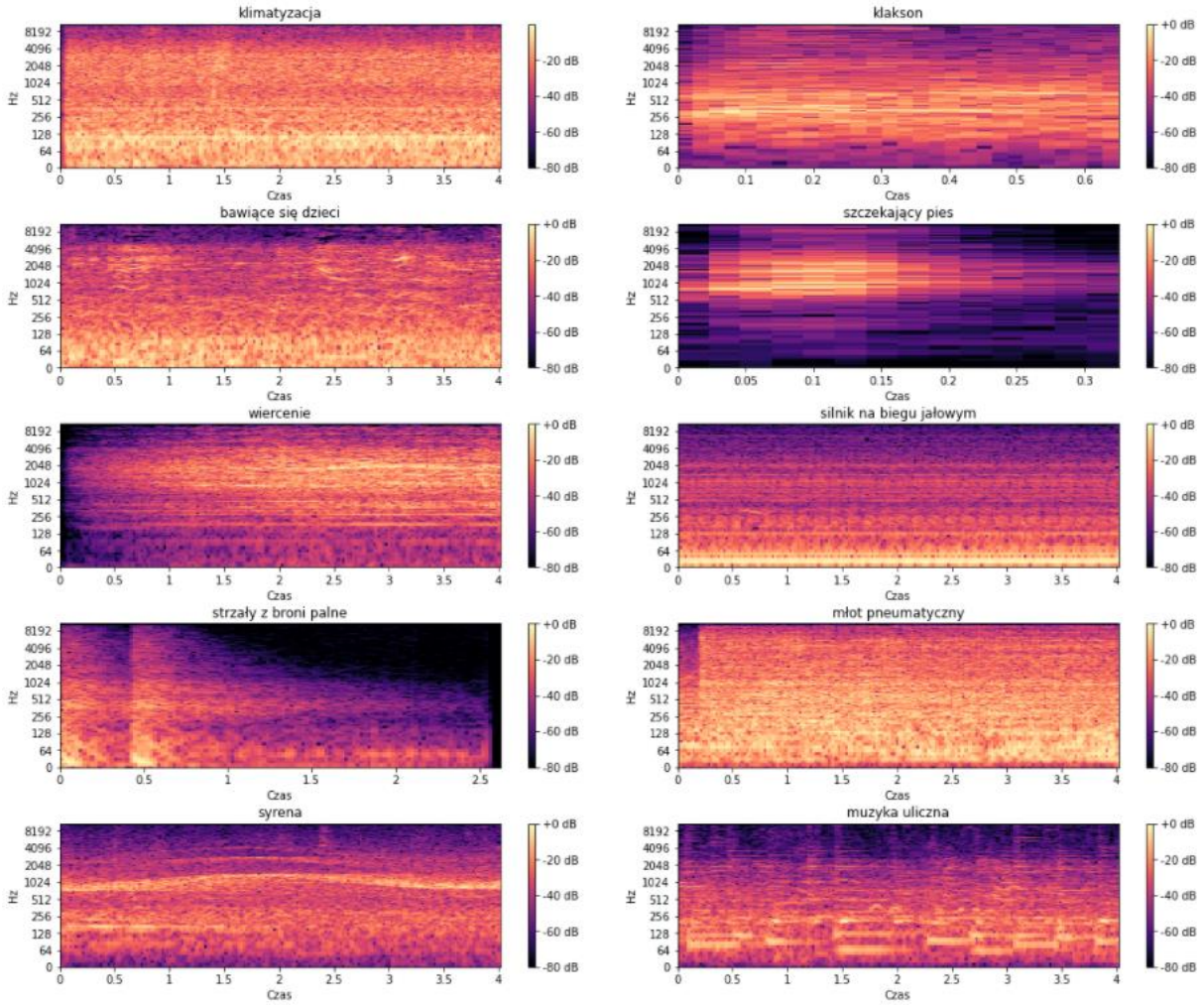
$$\mathrm{mel} = 2595\log_{10}(1 + \frac{\mathrm{f}}{700}) \tag{2.1}$$

Gdzie:

f – częstotliwość w [Hz].

Korzystając z matematycznych przekształceń częstotliwości są konwertowane to skali melowej. Wyniki to z tego, że ludzkie ucho inaczej odbiera zmianę różnych częstotliwości. Przykładowo łatwiej zostanie zauważona przez człowieka zmiana częstotliwości z niskiej na wysoką niż z wysokiej na wyższą, mimo, iż różnica może być jednakowa. Ilustruje to Rys.7.

Spektrogram w skali melowej jest spektrogramem, w którym częstotliwości zostały przekonwertowane do skali melowej. W ostatniej fazie przetwarzania wartości spektrogramu są przeskalowywane ze skali mocy do skali decybelowej. Jest to domyślna reprezentacja dźwięków, która będzie używana w pracy, później nazywana **spektrogramem przeskalowanym w skali melowej**.



Rys.8: Spektrogramy przeskalowane w stali melowej dla przykładowych dźwięków danych klas.

2.2 Reprezentacja dźwięków w postaci tablicowej

Każdy spektrogram przeskalowany w stali melowej, reprezentujący pojedynczy dźwięk, zapisany jest w formie tablicy o wymiarach 128x128. Po przekształceniu wszystkich dźwięków do danej postaci otrzymuje się 8732 tablic 128x128. Zostaną one wykorzystane do trenowania i testowania utworzonych modeli uczenia maszynowego. Klasa reprezentująca poszczególny dźwięk zapisana jest w wektorze o wymiarach 1x10 z wykorzystaniem techniki **kodowania „gorącojedynkowego”**.

2.2.1 Kodowanie „gorącojedynkowe”

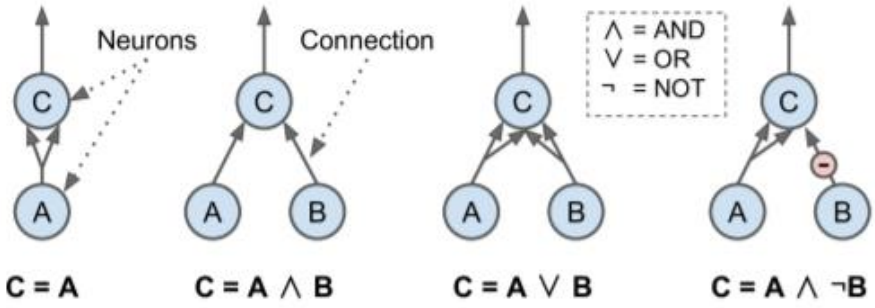
Idea kodowania „gorącojedynkowego” polega na wprowadzeniu sztucznej cechy (*ang. dummy feature*) dla każdej unikalnej wartości w kolumnie cechy nominalnej. W przedstawianej analizie zastosowanie tej techniki powoduje przekonwertowanie wektora przynależności dźwięków do klas do tablicy, gdzie każdą kolumnę identyfikuje się z poszczególną klasą dźwięku. Wykorzystywane są wartości binarne w celu wyszczególnienia rodzaju danego zapisu audio. Każdy plik audio przypisany jest do jednej z dziesięciu klas.

3. Sztuczne sieci neuronowe

Sztuczne neurony stanowią elementy używane przy budowaniu wielowarstwowych sztucznych sieci neuronowych. Koncepcje i założenia dotyczące tego zagadnienia inspirowane są występującymi w naszych mózgach sieciami neuronów biologicznych.

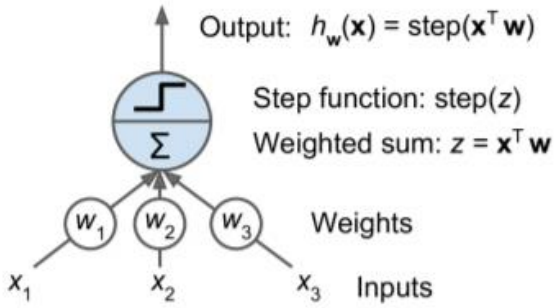
3.1 Sztuczny neuron

Model sztucznego neuronu po raz pierwszy zaproponowany został w 1943 roku przez neurofizjologa Warrena McCullocha i matematyka Waltera Pittsa (tzw. neuron McCullocha-Pittsa) [11]. Prezentowana przez nich komórka nerwowa przedstawiona została jako bramka logiczna posiadająca co najmniej jedno binarne wejście i jedno binarne wyjście.



Rys.9: Operacje logiczne wykonywane przez sztuczne sieci neuronowe. [12]

W odstępnie paru lat od tego wydarzenia Frank Rosenblatt upublicznił koncepcję reguły uczenia **perceptronu** [13], stanowiącą jedną z najprostszych architektur sztucznych sieci neuronowych. **Progowa jednostka logiczna** (*ang. Threshold Logic Unit*) stanowi jego podstawę i jest lekko zmodyfikowanym sztucznym neuronem. Wartościami wejść/wyjść, w przeciwieństwie do sztucznego neuronu, są liczby, gdzie każdemu połączeniu przypisana jest waga. Progowa jednostka logiczna oblicza ważona sumę sygnałów wejściowych, a następnie zostaje użyta funkcja aktywacji wobec otrzymanej sumy, dająca tym samym ostateczny wynik.



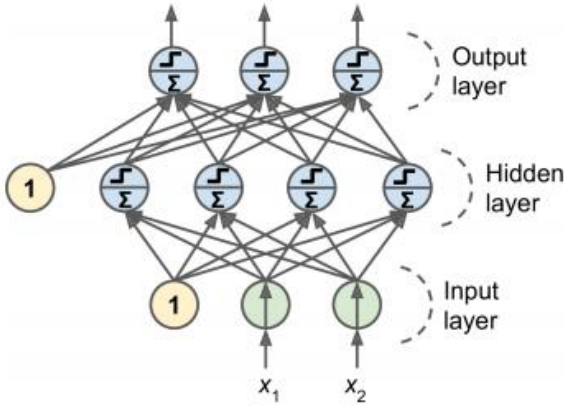
Rys.10: Liczenie sumy ważonej sygnałów wejściowych i zastosowanie funkcji aktywacji. [12]

3.2 Perceptron wielowarstwowy

Na perceptron wielowarstwowy składają się:

- jedna warstwa wejściowa (przechodnia),
- co najmniej jedna warstwa progowych jednostek logicznych (warstwy ukryte),
- warstwa progowych jednostek logicznych (warstwa wyjściowa).

Ponadto każda warstwa, poza warstwą wyjściową, posiada neuron obciążający (*ang. bias neuron*), który cały czas wysyła na wyjście wartość 1.



Rys.11: Diagram perceptronu wielowarstwowego. [12]

W momencie, gdy sztuczna sieć neuronowa zawiera wiele warstw ukrytych, nazywa się ją **głęboką siecią neuronową**.

3.3 Funkcja kosztu

Skuteczność modeli uczenia maszynowego, dla konkretnych danych, może być mierzona za pomocą funkcji kosztu. Określa ona różnicę pomiędzy przewidzianą wartością, a wartością rzeczywistą, w postaci liczby. Najczęściej dąży się do tego, by odnaleźć optymalne parametry modelu, dla których funkcja kosztu będzie zwracała wartości jak najmniejsze.

Funkcję kosztu można określić za pomocą sumy kwadratów błędów (*ang. Sum of Squared Errors*), między otrzymanymi wynikami, a rzeczywistymi wartościami:

$$J(w) = \frac{1}{2}\sum_i (y^{(i)} - \varphi(z^{(i)}))^2 \qquad (3.1)$$

Gdzie:

$y^{(i)}$ – wartość rzeczywista dla próbki i,

$\varphi(z^{(i)})$ – wartość przewidziana dla próbki i.

Wielowymiarowa entropia krzyżowa (*ang. Categorical Cross-Entropy*) stanowi funkcję kosztu używaną w klasyfikacji wieloklasowej. W tym przypadku próbka może należeć do jednej z wielu kategorii, a zadaniem modelu jest przydzielenie jej prawidłowej etykiety.

$$L^i_{cross-entropy} = -\sum_j \dot{y}^i_j \log\left(y^i_j + \epsilon\right) \tag{3.2}$$

Gdzie:

$\dot{y}^i = (\dot{y}^i_1, \ldots, \dot{y}^i_N)$ – etykieta dla przykładu i w kodowaniu 1 z N (kodowanie „gorącojedynkowe”),

$y^i = f(x^i;\ \theta)$ - predykcja $y^i = (y^i_1, \ldots, y^i_N)$ dla przykładu i generowana za pomocą modelu określonego parametrami θ,

ϵ – niewielka wartość dodawana dla celów stabilności numerycznej.

3.4 Propagacja wsteczna

Algorytm propagacji wstecznej służy do efektywnego wyliczenia pochodnych funkcji kosztu. Pochodne te są następnie używane w celu aktualizowania współczynników wag sieci neuronowej. Po otrzymaniu gradientów realizowany jest klasyczny algorytm gradientu prostego, a proces wykonywany jest do momentu zbieżności z rozwiązaniem. W algorytmie korzysta się z metody odwrotnego różniczkowania automatycznego, która bardzo dobrze sprawuje się w problemach, w których występuje wiele zmiennych.

3.5 Porzucanie

Porzucanie (*ang. Dropout*) w danej warstwie odnosi się do tymczasowego pominięcia neuronu w procesie uczenia. Każdy neuron ma jednakowe prawdopodobieństwo, najczęściej 50%, na pominięcie w danym przebiegu treningowym. Używane jest w celu zapobiegnięcia przetrenowania w sieciach głębokich.

3.6 Epoka

Epoką nazywa się jednorazowe przejście sieci neuronowej przez cały zbiór treningowy. Podstawowo wyznaczana jest jedynie jedna epoka, co najczęściej nie wystarcza modelowi do uzyskania zbieżności. W celu uniknięcia trenowania przez kolejne epoki, bez wyraźnych skutków, stosowane są metody wczesnego zatrzymania.

3.7 Funkcje aktywacji

Funkcja aktywacji na wejściu otrzymuje sumę kombinacji określonych wartości wejściowych i powiązanego z nimi wektora wag. Zwrócona wartość stanowi wyjście neuronu.

3.7.1 Funkcja sigmoidalna

Funkcja sigmoidalna jest funkcją aktywacyjną o charakterystycznym S-kształcie. W swojej podstawowej postaci konwertuje wejściowe wartości na wartości z zakresu (0;1), jednak istnieją jego odmiany, takie jak tangens hiperboliczny, które przyjmują wartości (-1;1).

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.3}$$

3.7.2 Funkcja ReLU

Funkcja ReLU stanowi funkcję ciągłą, nieróżniczkowalną w punkcie z = 0, a której pochodna dla z < 0 wynosi 0. Za sprawą szybkości przetwarzania, wynikającej m.in. z tego, że pochodna funkcji przyjmuje stałą wartość 1 dla z > 0, i dobrego sprawdzania się w praktyce stała się funkcją domyślną. Brak maksymalnej wartości wyjściowej minimalizuje niektóre problemy związane z metodą gradientu prostego, takie jak problem zanikających gradientów (*ang. vanishing gradients*).

$$\sigma(z) = \max(0, z) \tag{3.4}$$

3.7.3 Funkcja softmax

Funkcja softmax jest używana w klasyfikacji wieloklasowej. Stosuje się ją w warstwie wyjściowej w celu przekonwertowania wyników do znormalizowanego rozkładu prawdopodobieństwa. Wartości te mieszczą się w przedziale [0;1].

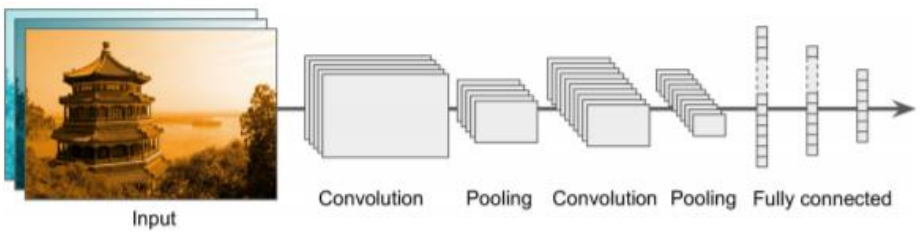
$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.5}$$

3.8 Optymalizator

Algorytmy optymalizacyjne są standardowym sposobem minimalizacji funkcji kosztu za pomocą metody spadku gradientu i jego wariacji. Wykorzystywanym hiperparametrem jest współczynnik uczenia (*ang. learning rate*), który określa prędkość dopasowywania się do danych. Po zastosowaniu takiego optymalizatora wagi w procesie uczenia zostają zaktualizowane.

3.9 Konwolucyjne sieci neuronowe

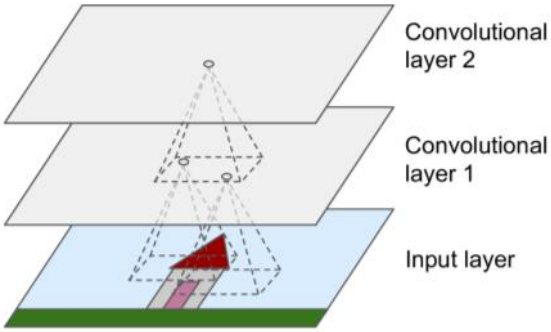
Konwolucyjne (splotowe) sieci neuronowe są powszechnie wykorzystywane do przetwarzania obrazów. Stanowią podstawę programów wykorzystywanych w celu rozpoznawania określonych obiektów. Jednocześnie okazują się skuteczne również w zadaniach innego typu, takich jak przetwarzanie języka naturalnego, czy rozpoznawanie mowy i dźwięków. Zważając na to ostatnie, stanowią doskonałe narzędzie do podejmowanego w pracy problemu.



Rys.12: Typowa architektura konwolucyjnych sieci neuronowych. [12]

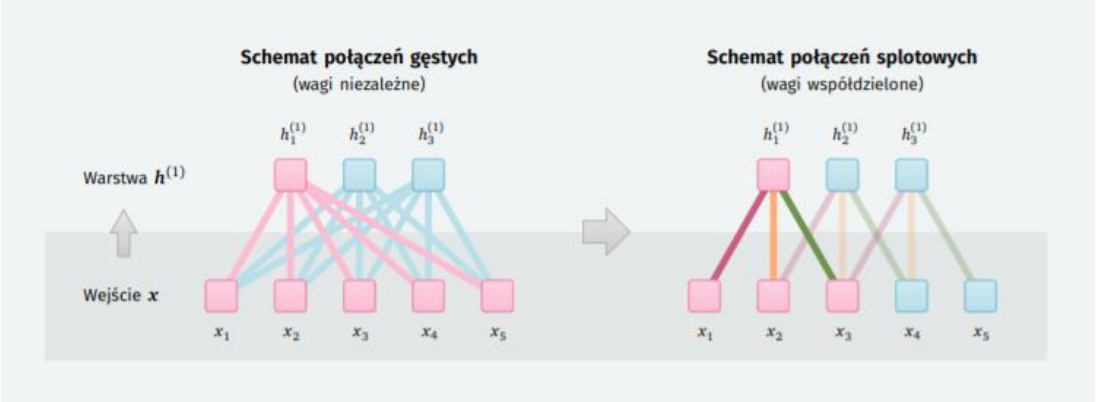
3.9.1 Warstwy splotowe (konwolucyjne)

Warstwy splotowe stanowią najistotniejszy składnik konwolucyjnych sieci neuronowych. Ich działanie stanowi analogię do działania neuronów składających się na korę wzrokową, która wyznacza lokalne pola recepcyjne, tj. reaguje wyłącznie na bodźce wzrokowe, które mieszczą się w określonym rejonie pola wzrokowego. Przykładając to na problem rozpoznania obiektów na obrazie - neurony umieszczone w I warstwie konwolucyjnej nie są połączone z każdym pikselem obrazu otrzymanego na wejściu, natomiast są połączone z pikselami umieszczonymi w ich polu recepcyjnym. Idąc dalej, każdy neuron w II warstwie jest połączony jedynie z neuronami z określonego obszaru I warstwy. Postępując w ten sposób można sukcesywnie wyodrębniać poszczególne cechy obrazów.



Rys.13: Warstwy konwolucyjne. [12]

Zasadnicza różnica między warstwą splotową, a warstwą gęstą występującą w perceptronie jest to, że w warstwie gęstej każdy neuron jest połączony z każdym neuronem warstwy poprzedniej, a nie tylko z tymi znajdującymi się w ich polu recepcyjnym. Obrazuje to Rys.14.



Rys.14: Warstwa gęsta w pełni połączona w porównaniu do warstwy splotowej. [14]

3.9.2 Filtry

Filtrami nazywa się zbiory wag neuronu, ukazane w postaci niedużych obrazów o rozmiarze pola recepcyjnego. Dana warstwa, w której wszystkie neurony korzystają z tego samego filtra, daje mapę cech, która to z kolei pozwala na zaobserwowanie fragmentów najbardziej zbliżonych to danego filtra. Filtry przeważnie są wybierane przez sieć konwolucyjną w trakcie uczenia.

3.9.3 Stosy map cech

Warstwy splotowe składają się z paru map cech o jednakowych rozmiarach – odmiennych filtrów, które są stosowane na wejściach. W każdej mapie cech neurony posiadają takie same parametry, różne od parametrów w innych mapach. Obrazy, które sieć otrzymuje na wejściu, składają się również z warstw, nazywanych kanałami barw.

Ostateczny wzór na wartość wyjściową w warstwie splotowej wygląda następująco:

$$z_{i,j,k} = b_k \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k} \quad \text{gdzie} \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases} \tag{3.6}$$

Gdzie:

$z_{i,j,k}$ – wyjście neuronu będącego w rzędzie i, kolumnie j oraz mapie cech k warstwy splotowej l,

s_h, s_w – krok pionowy i poziomy, f_h, f_w – wysokość i szerokość pola recepcyjnego, $f_{n'}$ - liczba map cech w poprzedzającej warstwie l-1,

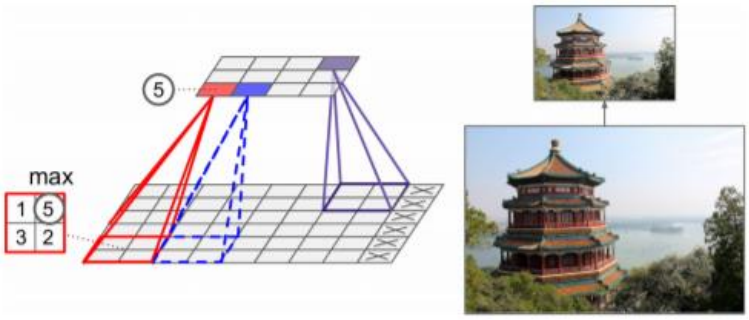
$x_{i',j',k'}$ - wyjście neuronu zlokalizowanego w warstwie l-1, rzędzie i', kolumnie j', mapie cech k (lub kanale k', w przypadku, gdy poprzednia warstwa była warstwą wejściową),

b_k- człon obciążenia dla mapy cech k (w warstwie l); interpretowany jako „pokrętło jasności" mapy cech k,

$w_{u,v,k',k}$ – waga połączenia między dowolnym neuronem w mapie cech k warstwy l, a jego wejściem mieszczącym się w wierszu u, kolumnie v (względem pola recepcyjnego neuronu), a mapą cech k'.

3.9.4 Warstwa łącząca

Warstwa łącząca (*ang. pooling layer*) odpowiada za zmniejszenie ilość obliczeń, pamięci oraz parametrów. Definiowane jest jądro łączące (*ang. pooling kernel*) o podanych rozmiarach, przesuwające się po neuronach warstwy poprzedzającej z określonym krokiem, które przesyła dane na wejście za pomocą określonej funkcji grupującej. Najpopularniejszym typem jest maksymalizująca warstwa łącząca *(ang. max pooling layer)*.



Rys.15: Maksymalizująca warstwa łącząca. [12]

W takim przypadku jak na Rys.15 do następnej warstwy zostanie przekazana jedynie maksymalna wartość z określonego jądra. Mogłoby się wydawać, że skoro aż tyle pozostałych informacji zostanie odrzuconych, lepszym pomysłem byłoby użycie innej funkcji agregującej, np. uśredniającej warstwy łączącej (*ang. average pooling layer*), która przekazałaby średnią. Praktyka pokazuje jednak, że podejście takie jest mniej wydaje i nie skutkuje przekazaniem najistotniejszych informacji.

4. Tworzenie modeli sieci neuronowych

4.1 Wstępne przetwarzanie dźwięków

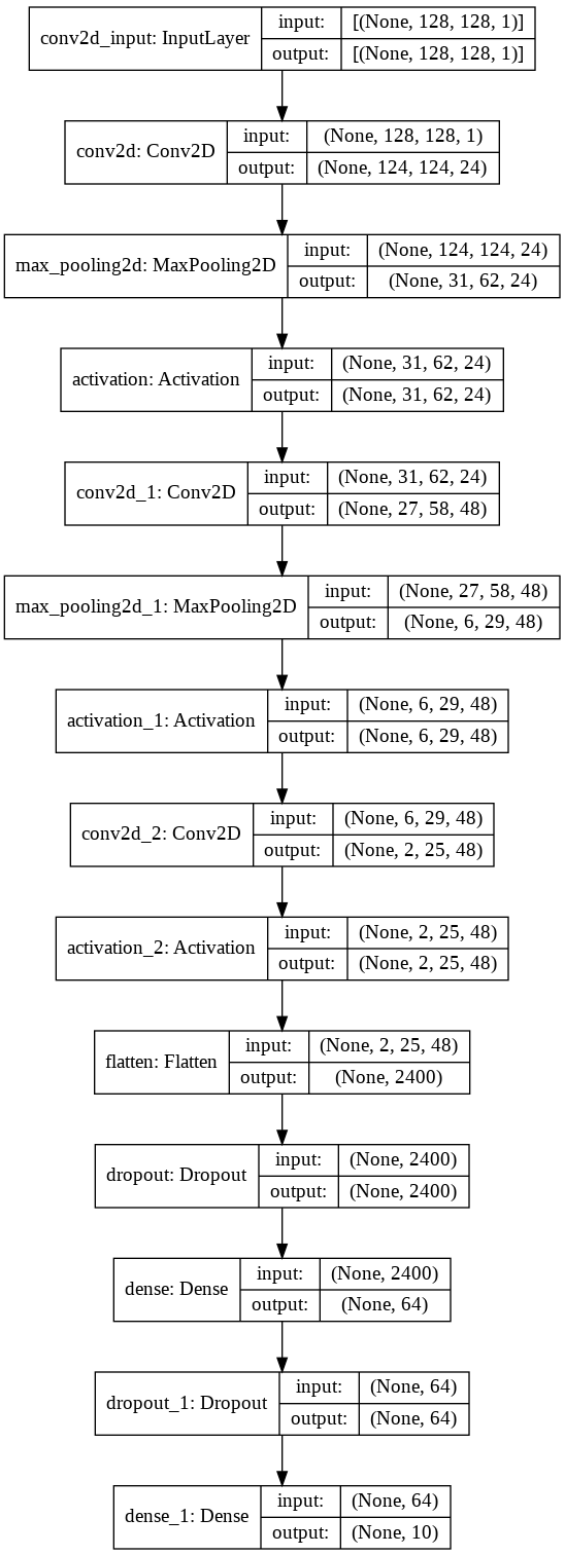
W celu ujednolicenia przetwarzanych dźwięków określone zostają wartości atrybutów konwertera na przeskalowany spektrogram w skali melowej. Zakres częstotliwości zostaje określony przez słyszalny ludzkim uchem przedział 0-22050 Hz, długość okna spektrogramu wynosi 23 ms i tyle samo wynosi przeskok (*ang. hop-size*) między kolejnymi ramkami. Oznacza to, że szybka transformacja Fouriera zostanie zastosowana na przesuwającym się wzdłuż sygnału oknie czasowym o długości 23 ms i tyle będzie wynosić przeskok między kolejnymi ramkami. Długość dźwięków zostaje określona na 3 sekundy (128 ramek), zatem odpowiednie spektrogramy są dopełniane lub ich części są odcinane, w zależności od przypadku. Służy to temu, by pliki audio były reprezentowane przez tablice o takich samych wymiarach 128x128.

4.2 10-krotna walidacja krzyżowa

Podczas trenowania modeli zastosowana została 10-krotna walidacja krzyżowa. Tak jak zostało już opisane w pierwszym rozdziale, pliki audio rozdzielone są na 10 folderów. Zatem w każdym z 10 eksperymentów, dla konkretnego modelu, inny folder z dźwiękami stanowi dane testowe, natomiast pozostałe foldery z plikami stanowią zbiór treningowy. Jest to swego rodzaju wymaganie, nałożone przez autorów zbioru danych [5], by wyniki uzyskane na stworzonym zbiorze danych były uznane za reprezentatywne. Wynika to z faktu, że niektóre dźwięki pochodzą z tego samego, oryginalnego nagrania. Nie możemy zatem zebrać wszystkich danych, wymieszać ich i losowo rozdzielić na zbiór testowy i treningowy, ponieważ istnieje wówczas szansa, że wycinki pochodzące z tego samego źródła znajdą się zarówno w jednym, jak i drugim zbiorze. Skutkowałoby zawyżonymi efektywnościami klasyfikacji. Ponadto ze zbioru treningowego jest losowo wybierany folder, który będzie stanowił zbiór walidacyjny podczas treningu.

4.3 Model bazowy

Pierwszy stworzony w ramach pracy model swoją architekturą bazuje na modelu zaproponowanym przez autorów używanego zbioru danych [15]. Model posiada trzy warstwy splotowe z dwoma warstwami łączącymi między nimi, po których występują dwie warstwy gęste w pełni połączone. Porzucanie z prawdopodobieństwem 50% jest stosowane do wejścia dwóch ostatnich warstw. Ten model, jak i wszystkie następne, wykorzystują implementację wczesnego zatrzymywania po 15 epokach, tj. w razie braku poprawy skuteczności na zbiorze walidacyjnym, trwającym 15 epok, trenowanie jest przerywane, a model wraca do najlepszej otrzymanej wersji.

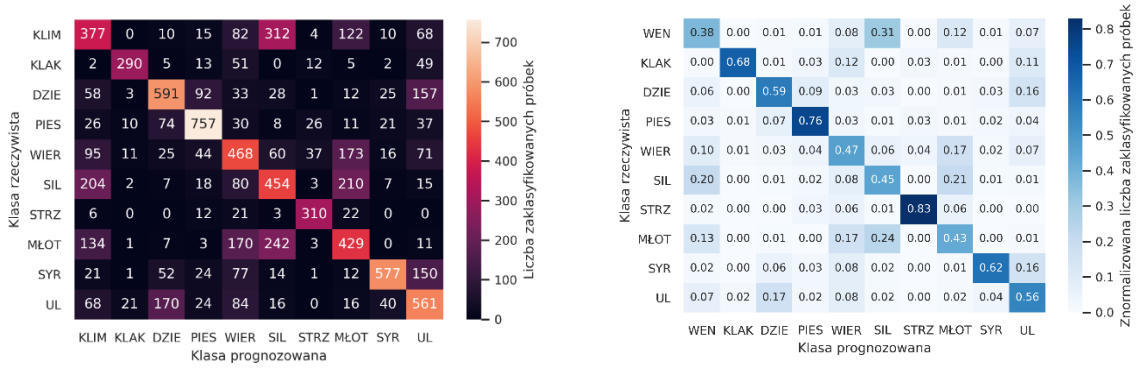


Rys.16: Architektura modelu bazowego.

Stosowaną funkcją aktywacyjną jest ReLU, z wyłączeniem ostatniej warstwy, gdzie użyta została funkcja softmax, w celu przedstawienia wyników jako prawdopodobieństwa przynależności do poszczególnych klas.

Model jest trenowany przez 50 epok, posiada 241 434 parametrów, a jako optymalizator zastosowany został optymalizator spadku stochastycznego gradientu (*ang. stochastic gradient descent*) o stałym współczynniku uczenia 0,01.

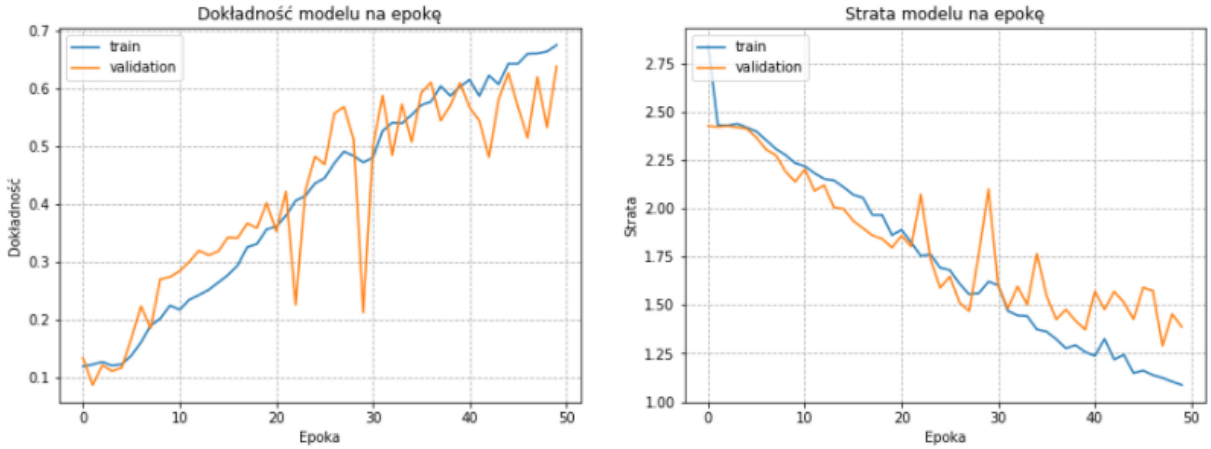
Po odtworzeniu modelu 10 razy, za każdym razem biorąc inny folder jako folder testowy, średnia dokładność modelu wyniosła **56,2%** (przy losowym przypisaniu wyniosłaby 10%).



Rys.17: Macierz pomyłek dla zbioru danych na modelu bazowym w postaci podstawowej i znormalizowanej.

Jak można zaobserwować na Rys.17, model miał spore trudności z prawidłowym przyporządkowaniem etykiet i nietrudno jest znaleźć takie klasy, dla których dokładność wynosi poniżej 50%. Najczęstsze błędy, które wystąpiły w trakcie klasyfikacji, to mylenie dźwięku wentylatora z dźwiękiem silnika, dźwięku młota pneumatycznego z dźwiękiem silnika czy dźwięku silnika z dźwiękiem młota pneumatycznego. Wszystkie te pomyłki wydają się zrozumiałe, ponieważ dźwięki te mogą być w pewien sposób do siebie podobne. Model najlepiej sprawdził się w przypadku dźwięków strzałów z broni palnej i dźwięków szczekających psów.

Przy dokonywaniu niektórych pomiarów spotkałem się z efektem, kiedy dokładność walidacyjna była sporo wyższa od dokładności treningowej, a strata walidacyjna sporo niższa od straty treningowej. Jest to efekt, który może pojawić się w przypadku małych zbiorów danych lub gdy dane treningowe i testowe/walidacyjne nie są losowo rozdzielone – oba te powody mogą występować w tym przypadku. Innym uzasadnieniem takiego efektu może być fakt, że mechanizmy optymalizacyjne, takie jak Dropout, czy regularyzacja L1/L2, są wyłączone na czas testowania. Mają zatem wpływ na stratę treningową, ale nie na testową/walidacyjną [16].

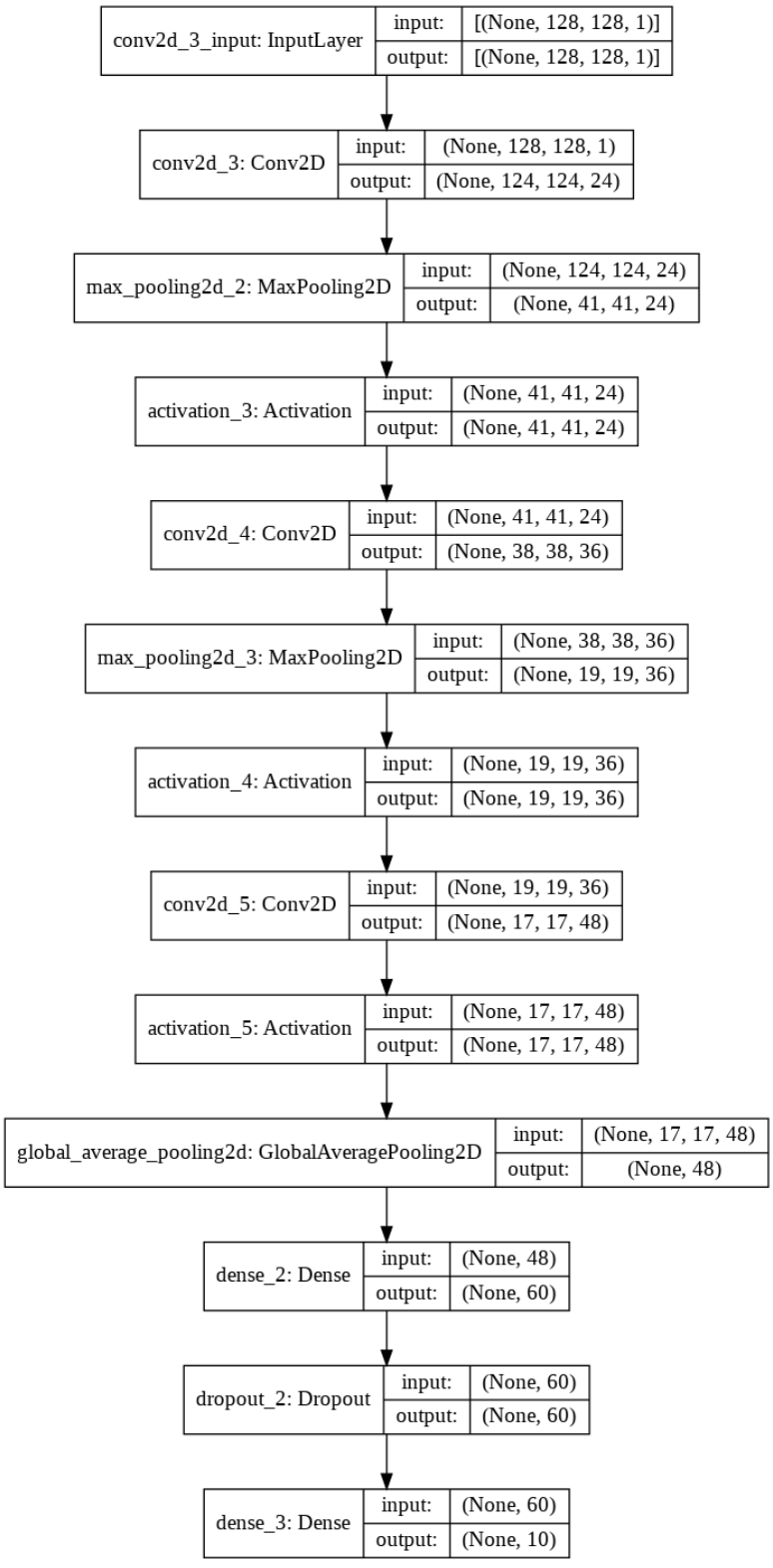


Rys.18: Przykładowy wykres zmiany dokładności i straty w funkcji liczby epok dla modelu bazowego.

Analizując lewą część Rys.18 można rozważyć przedłużenie treningu, zważając na rosnący trend dokładności w ciągu ostatnich kilku epok.

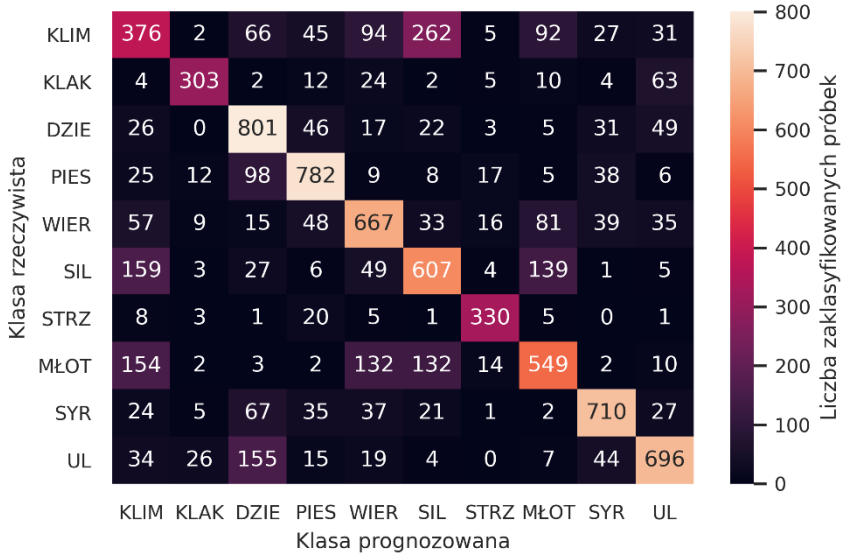
4.4 Model ulepszony

Model bazowy osiąga akceptowalne wyniki i ustala w pewien sposób dolną granicę skuteczności. Zainspirowany inną pracą [17] wprowadziłem do modelu pewne modyfikacje. Należą do nich m.in. zmiana rozmiaru jądra łączącego, zastosowanie optymalizatora Adam, zmniejszenie liczby filtrów w drugiej warstwie oraz zastosowanie porzucania wyłącznie do wejścia ostatniej warstwy. Posiada on jedynie 33 634 parametrów, jednak trenowanie trwa 100 epok, a nie 50, jak w poprzednim przykładzie.



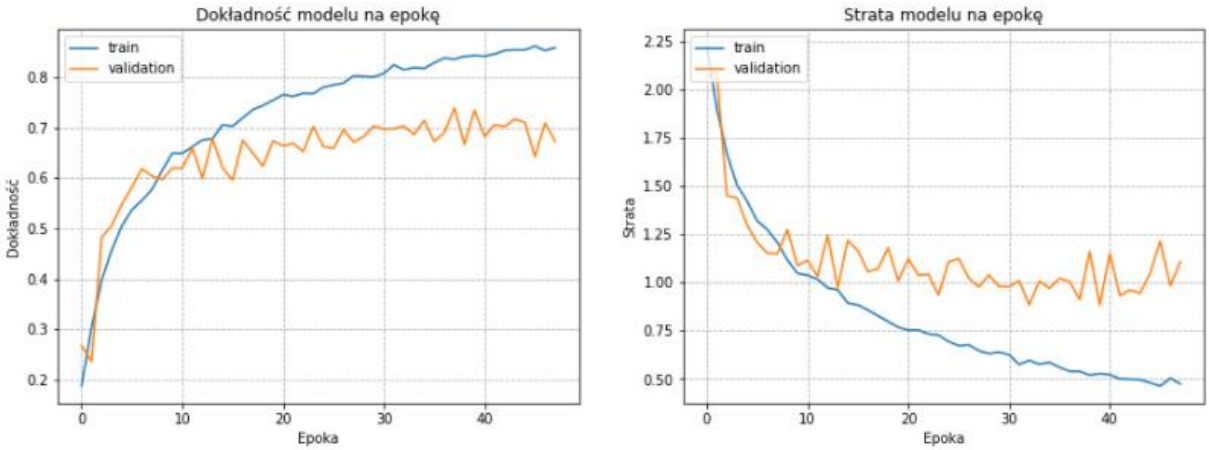
Rys.19: Architektura modelu ulepszonego

W tym przypadku model uzyskał średnią dokładność **69,4%**. Wynik poprawił się o blisko 13,2%. Każdy z trenowanych modeli wykazał w pewnym stopniu przetrenowanie. Zważając na to można rozważyć zmniejszenie liczby epok lub zmianę parametrów modelu czy zastosowanie technik regularyzacyjnych.



Rys.20: Macierz pomyłek dla zbioru danych testowanych na modelu ulepszonym.

Porównując macierze pomyłek z Rys.17 i Rys.20 można zauważyć, że dla prawie wszystkich klas, z pominięciem dźwięku wentylatora (Rys.17 – 377 prawidłowo sklasyfikowanych próbek), ogólna liczba prawidłowo przyporządkowanych etykiet wzrosła. Do przypadków wyróżniających się w stosunku do poprzedniego modelu należą m.in. znaczny wzrost błędnego klasyfikowania dźwięków klimatyzacji jako dźwięki szczekającego psa, bawiących się dzieci czy wiercenia.



Rys.21: Przykładowy wykres zmiany dokładności i straty w funkcji liczby epok dla modelu ulepszonego.

Jak widać na Rys.21, pomimo tego, że trenowanie modelu miało docelowo trwać 100 epok, w tym przypadku zakończyło się przed osiągnięciem 50 epok. W większości pozostałych przypadków wyglądało to podobnie.

4.5 Syntetyczne powiększenie zbioru uczącego

Dosyć częstym problemem, pojawiającym się w kontekście przetwarzania dźwięków przez sieci neuronowe, jest niedostatek próbek potrzebnych do uczenia modelu. Używany w pracy zbiór danych posiada 8732 pliki audio i jest to jeden z większych zestawów kategoryzowanych dźwięków, które można znaleźć w Internecie, nie uwzględniając udostępnionego przez Google AudioSetu. W ciągu ostatnich lat pojawiło się parę inny zbiorów, takich jak ESC-50 [18], jednak wciąż są one małe w porównaniu do, chociażby, zbiorów danych używanych przy klasyfikacji obrazów [19].

Popularnym rozwiązaniem tego problemu jest syntetyczne powiększenie zbioru uczącego (*ang. data augmentation*). Polega ono na zaaplikowaniu różnorakich deformacji na próbki dźwięków i potraktowaniu ich jako nowe dane treningowe. Zmiany te nie wpływają na semantyczne znaczenie plików, tj. dźwięk wiercenia będzie wciąż oznaczał i będzie przez słuchających rozpoznawany jako dźwięk wiercenia. Z ową metodyką można spotkać się w większości prac korzystających z tego lub podobnego zbioru danych, a nawet jeśli generowanie syntetycznych danych nie odbywa się wprost, to jest robione pośrednio, np. dzieląc jeden dźwięk na parę oddzielnych [20]. Zwiększenie liczby próbek uczących może również wpłynąć na zmniejszenie efektu przetrenowania (*ang. overfitting*).

Do popularnych deformacji zaliczają się przede wszystkim:

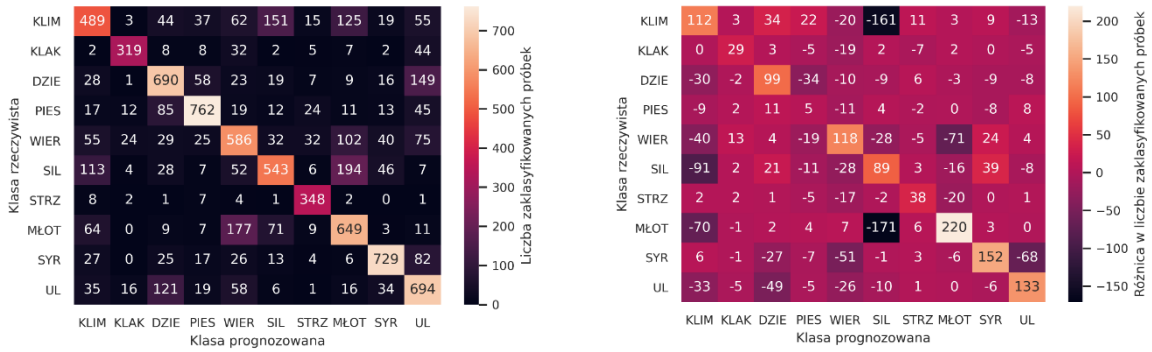
- **Rozciąganie czasu** (*ang. Time Stretching*) – spowalnianie lub przyśpieszenie pliku audio.
- **Pitch Shifting** – podniesienie lub obniżenie wysokości tonów sygnału dźwiękowego.
- **Kompresja dynamiki** (*ang. Dynamic range compression*) – zmniejszenie dynamiki sygnału.
- **Szum otoczenia / szum biały** – nałożenie na próbkę dźwięku innej próbki, zawierającej dźwięki otoczenia / szum biały.
- **Przycinanie ciszy** *(ang. Silence trimming*) – cicha część pliku audio zostaje wycięta, a zapisana zostaje jedynie ta zawierająca dźwięki.

Cztery pierwsze zostały użyte w pracy z następującymi parametrami: rozciąganie czasu – współczynnik 1,23; pitch shifting – wartość 2 (w półtonach); kompresja dynamiki – parametr „film standard”; szum otoczenia – pracownicy uliczni. Wybór parametrów sugerowany był wcześniejszymi badaniami [15] [17].

Każda z wymienionych deformacji została użyta na każdym dźwięku i zapisana, co zwiększyło liczbę próbek 4-krotnie, ostatecznie dostarczając zbiór danych o łącznej wielkości 34 928 próbek. W tym przypadku zbiór testowy stanowi jeden z folderów ze zbioru podstawowego, zbiór walidacyjny stanowi jeden z pozostałych 9 folderów, natomiast zbiór treningowy stanowią syntetycznie wygenerowane dane, których oryginalne dźwięki nie należą do wcześniej użytych dwóch folderów.

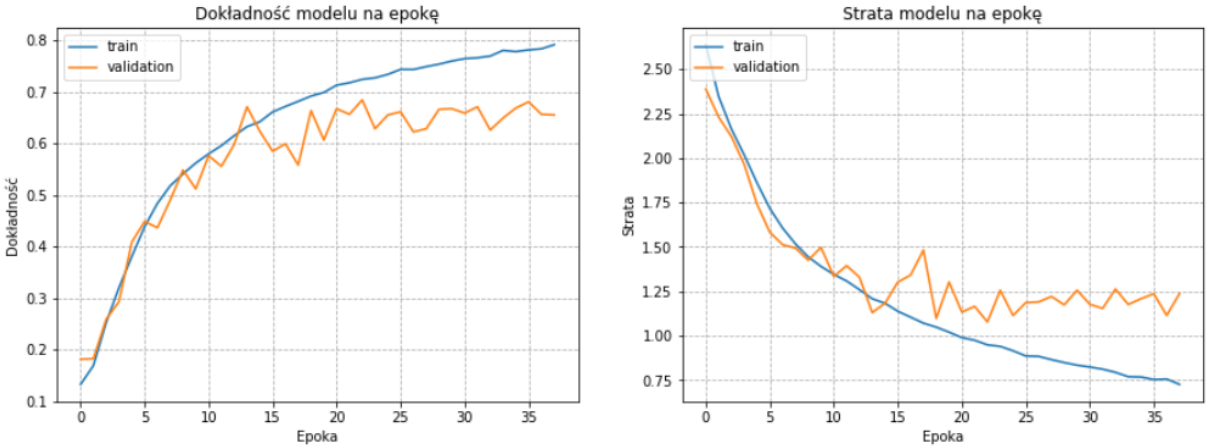
Zaobserwowano niewielki wpływ większej liczby próbek treningowych na średnią dokładność modelu ulepszonego. Za każdym razem wynik wahał się +/- 1% względem wyniku uzyskanego dla podstawowych danych. Nie jest to aż tak zadziwiające, zważając na to, że model wykazał wysoką dokładność już z bazowym zbiorem. Podobne wyniki uzyskali autorzy proponowanego modelu, tj. średnia dokładność wzrosła jedynie o 1% [17].

Wpływ zwiększonej liczby próbek uczących na dokładność zmierzony względem modelu bazowego, bez użycia syntetycznych danych treningowych, to wzrost o ponad 11% efektywności. Osiągnięta średnia dokładność to **67,3%.**



Rys.22: Macierz pomyłek dla modelu bazowego z syntetycznymi danymi (lewa strona) oraz różnica w macierzach pomyłek dla modeli trenowanych na danych syntetycznych i oryginalnych (prawa strona).

Rys. 22 obrazuje ogólny wzrost liczby prawidłowo zaklasyfikowanych próbek dla wszystkich klas. Istnieją jednak przypadki, dla których wzrost danych treningowych spowodował błędy, które wcześniej nie pojawiały się w takich ilościach. Przykładowo na Rys.17 dźwięki, które nie zostały rozpoznane jako klimatyzacja, a nią były, najczęściej były etykietowane jako dźwięki silnika, młota lub wiercenia. Patrząc jednak na Rys.22 (prawa strona) widać, że o ile odnotowano znaczy spadek w przypadku pierwszej i trzeciej klasy, a nieznaczny wzrost w przypadku drugiej, tak model zaczął przejawiać skłonności do klasyfikowania dźwięków klimatyzacji, jako bawiących się dzieci i szczekającego psa. Podobne błędy można było zaobserwować na Rys.20.



Rys.23: Przykładowy wykres zmiany dokładności i straty w funkcji liczby epok dla modelu bazowego z syntetycznymi danymi.

Analizując Rys.23 można zauważyć, że wykresy zaczęły bardziej przypominać te z Rys.21, niż z Rys.18. Pojawia się przetrenowanie modelu, przejawiające się zatrzymaniem wzrostu dokładności, jak i zatrzymaniem spadku straty dla danych walidacyjnych. Efekt ten nie występował w przypadku modelu bazowego bez syntetycznych danych. Ponadto trenowanie zakończyło się chwilę po przekroczeniu 35 epoki.

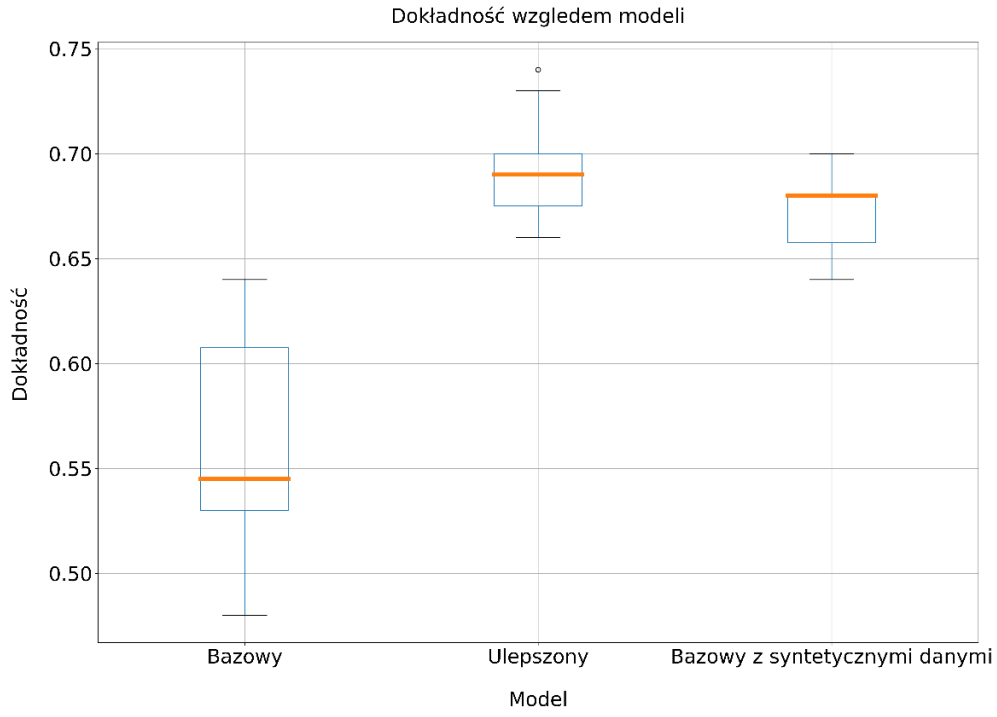
4.6 Porównanie wyników

Uzyskane wyniki można przedstawić w zbiorczej tabeli:

Model	Model bazowy	Model ulepszony	Model bazowy z syntetycznymi danymi
Średnia dokładność	56,2%	69,4%	67,3%
Mediana	54,5%	69%	68%
Odchylenie standardowe	5,2%	2,4%	1,8%

Tabela 1: Tabela przedstawiająca średnie osiągi poszczególnych modeli w zależności od danych.

Analizując Tabelę 1 i Rys.25 można stwierdzić, że o ile zwiększenie liczby próbek miało pozytywny wpływ na średnią dokładność modelu bazowego z syntetycznymi danymi, tak wynik ten i tak był niższy w porównaniu do modelu ulepszonego, choć odznaczał się mniejszym odchyleniem. Ciekawym spostrzeżeniem może być to, że o ile model ulepszony, jak i model bazowy z syntetycznymi danymi uzyskały porównywalnie dobre wyniki, tak wzrost tej dokładności przebiegał względem innych klas. Model ulepszony, w porównaniu do modelu bazowego, bardzo dobrze poradził sobie z klasyfikacją dźwięków bawiących się dzieci czy dźwięków wiercenia. Efektu aż w takiej skali nie obserwujemy względem modelu bazowego z syntetycznymi danymi, natomiast przejawia on inne cechy. Model ten z kolei o wiele lepiej poradził sobie z klasyfikacją dźwięków klimatyzacji czy dźwięków młota pneumatycznego. Osiągnął również lepsze osiągi względem dwóch najrzadszych klas – klaksonu samochodowego i strzału z broni palnej.



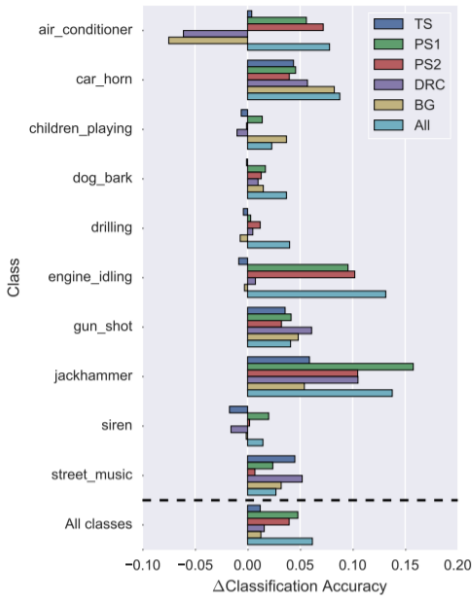
Rys. 25: Wykres pudełkowy dla wyników poszczególnych modeli.

5. Możliwe rozszerzenia

Przedstawione przeze mnie podejścia i modele do trenowania sieci neuronowych kategoryzujących krótkie dźwięki miejskie stanowią jedynie drobną część opcji, z którymi można zapoznać się w innych pracach. Niektóre z nich bardzo się od siebie różnią, a jednocześnie są w stanie osiągać porównywalne wyniki.

5.1 Generowanie dużej liczby syntetycznych próbek

Podejście zastosowane w ostatnim modelu tej pracy operuje na powiększonym zbiorze danych, lecz zbiór ten może zostać wielokrotnie rozszerzony. Autorzy danych w swojej pracy [15] zaproponowali nałożenie 4 różnych deformacji na każdy dźwięk z osobna, jednocześnie powtarzając eksperyment parę razy, używający innych współczynników. Ostatecznie każdy plik audio został poddany 20 różnym przekształceniom, dzięki czemu zbiór danych zawierał około 183 372 pliki, łącznie z podstawowym zbiorem. Taka kolekcja zajmuje 34 GB pamięci.



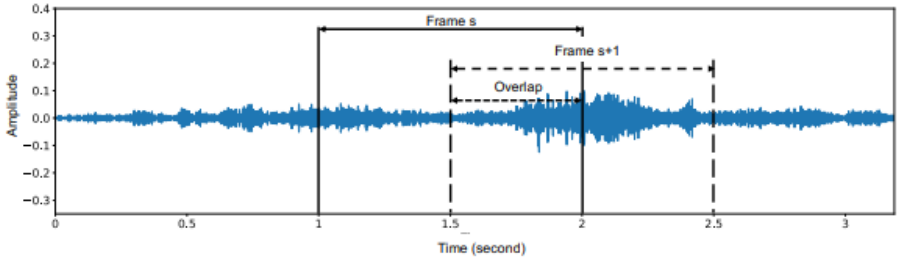
Rys.26: Wpływ poszczególnych deformacji na wynik względem kategorii. [15]

Jak zostało wykazane, nie wszystkie deformacje miały pozytywny wpływ na ostateczny wynik klasyfikacji w poszczególnych kategoriach. Zauważalne jest jednak, że ogólne zwiększenie liczby danych miało pozytywny wpływ na ostateczny wynik. Autorzy proponują by deformacje przydzielać względem kategorii, tak aby niektóre z nich nie wpływały negatywnie na klasyfikację.

5.2 Jednowymiarowa sieć konwolucyjna

Większość prac operuje na dźwiękach przekształconych do odpowiedniego typu, najczęściej sprowadzając reprezentację pliku audio do dwuwymiarowej tablicy, która następnie jest przesyłana do modelu.

We wspomnianej przy okazji deformacji pracy [20], autorzy zaproponowali podejście wyróżniające się na tle innych, ponieważ wykorzystujące jednowymiarowe sieci konwolucyjne, bez generacji syntetycznych danych, a jednocześnie odznaczające się wysoką dokładnością. W fazie wstępnego przetwarzania sygnał wejściowy w postaci fali dźwiękowej byłyby dzielony na odpowiednio nakładające się na siebie części. Część taka byłaby reprezentowana przez listę o określonym rozmiarze, która to następnie byłaby przesyłana do modelu.



Rys.27: Dzielenie sygnału wejściowego na kilka ramek z odpowiednim współczynnikiem nakładania części na siebie. [20]

Mechanizm ten pozwala na uzyskanie sporej liczby oddzielnych dźwięków z jednego pliku – przykładowo jeden 3 sekundowy plik audio „wygeneruje” z siebie kilka innych, w zależności od współczynnika nakładania.

Poczynione kroki w celu powtórzenia tego doświadczenia zakończyły się niepowodzeniem. Proces treningu był przy tym bardzo czasochłonny, nawet na tle innych, wymienionych w pracy modeli, a ostateczny wynik wahał się w granicach 30%. Oficjalna implementacja nie została nigdzie zamieszczona, więc porównanie kodu nie było możliwe. Niezależnie od tego faktu, praca przedstawia ciekawy pomysł podejścia do problemu, który omija ogrom prac wstępnego przetwarzania, operuje na obiektywnie prostym założeniu, które powinno skutkować wysokimi wynikami.

6. Podsumowanie

Zagadnienie klasyfikacji dźwięków miejskich stanowi niewątpliwie interesujący temat, stawiający wyzywania w postaci doboru odpowiedniego procesu wstępnego przetwarzaniu, modelu oraz parametrów. Każde z tych zagadnień stanowi równie istotną kwestię i każde z nich podejmowane jest jako potencjalna szansa na udoskonalenie klasyfikatorów.

W pracy miałem okazję zaprezentować etapy wstępnego przetwarzania plików audio, wyjaśnić ich reprezentację oraz przedstawić kilka popularnych modeli służących do klasyfikacji odpowiednich dźwięków. W celu przeprowadzania wysokojakościowych badań użyte zostały konwolucyjne sieci neuronowe oraz zastosowane zostały metody generacji syntetycznych danych. Zważając na to, że celem pracy nie było przedstawienie jak najlepszego modelu, a jedynie prezentacja niektórych podejść, badania były przeprowadzane z podstawowymi parametrami modeli i nie były regularyzowane.

Przedstawione klasyfikatory uzyskały odpowiednio dokładności 56,2%, 69,4% i 67,3%, ukazując pozytywny wpływ zwiększonej liczby próbek na ostateczny wyniki, w zależności od modelu.

Przedstawione zostały zależności pomiędzy poszczególnymi dźwiękami, ukazujące m.in. klasyfikowanie dźwięków klimatyzacji jako dźwięki wiercenia czy dźwięki bawiących się dzieci.

Możliwe koncepcje rozszerzenia badań zostały przedstawione w rozdziale 5 wraz z moimi obserwacjami przy próbach ich zastosowania.

Poczynione badania mogą posłużyć jako wstęp do głębszego zapoznania się z tematem, a omówiona teoria wyjaśniać kwestie transformacji danych i ich wariacji. Przetwarzanie dźwięków w uczeniu maszynowym wciąż stanowi zagadnienie niszowe w porównaniu do problemu przetwarzania obrazów, jednak zważając na rosnącą z roku na rok liczbę prac naukowych zajmujących się tą kwestią oraz potencjalny wpływ na systemy nadzoru, możemy spodziewać się rosnącego zainteresowania tym temat w ciągu najbliższych lat.

Bibliografia

[1] „Urban population (% of total population),” The World Bank, 2019. [Online].

[2] „Ludność. Stan i struktura ludności oraz ruch naturalny w przekroju terytorialnym,” Główny Urząd Statystyczny, 31 12 2020. [Online].

[3] „Kto marzy o życiu na wsi, a kto o życiu w mieście?,” Centrum Badania Opinii Społecznej, luty 2015. [Online].

[4] K. Strittmatter, Chiny 5.0. Jak powstaje cyfrowa dyktatura. Rozdział 9., wydanie pierwsze red., W.A.B., 2020.

[5] J. Salamon, C. Jacoby i J. P. Bello. [Online]. Available: https://urbansounddataset.weebly.com/urbansound8k.html.

[6] https://freesound.org/. [Online].

[7] J. Salamon, C. Jacoby i J. P. Bello, „A Dataset and Taxonomy for Urban Sound Research,” w *22nd ACM International Conference on Multimedia*, Orlando, 2014.

[8] „Fourier transform,” aavos, [Online]. Available: https://aavos.eu/glossary/fourier-transform/.

[9] M. X. Cohen, „Short-time Fourier transform,” [Online]. Available: https://youtu.be/8nZrgJjl3wc.

[10] P. G. Katedra Systemów Multimedialnych, „psychofizjologia słyszenia,” [Online]. Available: https://sound.eti.pg.gda.pl/student/elearning/fizjo.htm.

[11] W. Pitts i W. S. McCulloch, „A logical calculus of the ideas immanent in nervous activity,” 1943.

[12] A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow, wydanie drugie red., O'Reilly Media, 2019.

[13] F. Rosenblatt, „The Perceptron, a Perceiving and Recognizing Automaton,” 1957.

[14] K. J. Piczak, „Klasyfikacja dźwięku za pomocą splotowych sieci neuronowych,” Warszawa, 2018.

[15] J. P. Bello i J. Salamon, „Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification,” 2016.

[16] keras.io, „Why is my training loss much higher than my testing loss?,” [Online].

[17] Z. Mushtaq i S.-F. Su, „Environmental sound classification using a regularized deep convolutional neural network with data augmentation,” 2020.

[18] K. J. Piczak, „ESC: Dataset for Environmental Sound Classification,” Warszawa.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li i L. Fei-Fei, „ImageNet: A Large-Scale Hierarchical,” 2009.

[20] S. Abdoli, P. Cardinal i A. L. Koerich, „End-to-End Environmental Sound Classification using a 1D Convolutional Neural Network,” 2019.

Dodatki

Dodatek A - Biblioteki

Matplotlib – biblioteka służąca do generowania wykresów i rysunków.

Numpy – biblioteka pozwalająca na przechowywanie zbiorów danych i dokonywanie złożonych operacji matematycznych.

Pandas – biblioteka zapewniająca łatwy proces wczytywania i zapisywania zbiorów danych oraz służąca do przechowywania kolekcji.

TensorFlow – biblioteka z interfejsem Keras, uznawana za jedno z najbardziej użytecznych narzędzi przeznaczonych do uczenia maszynowego.

Librosa – biblioteka zapewniająca funkcjonalności do procesu wstępnego przetwarzania dźwięków. Pozwala ona m.in. na zapisanie dźwięków w postaci tablicowej, transformacje do postaci spektrogramów, czy zmianę długości pliku audio.

MUDA – biblioteka implementująca rozszerzanie (deformację) danych muzycznych. Pozwala ona na proste zaaplikowanie perturbacji, wzorowane na transformacjach z biblioteki scikit-learn. Próbki przechowywane są jako tablice oraz odpowiadające im adnotacje w formacie JAMS.

Dodatek B - Oprogramowania

Google Colab – środowisko Jupyter obsługiwane przez Google. Pozwala ono na pisanie programów w Pythonie, zapewnia możliwość pracy z różnymi typami procesorów, udostępniania notatników na GitHubie i wiele innych.

Spyder (Anaconda) – oprogramowanie pozwalające na pracę z bibliotekami jak TensorFlow, czy Keras, z poziomu środowiska lokalnego na systemie Windows. Udostępnia również m.in. funkcje pomagające w zarządzaniu wykresami i wizualizacjami danych.