

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI  
I TECHNIK INFORMACYJNYCH



Instytut Informatyki

# Praca dyplomowa inżynierska

na kierunku Informatyka  
w specjalności Sztuczna Inteligencja

Etykietowanie muzyki (music tagging)  
za pomocą metod sztucznej inteligencji

Olga Krupa

Numer albumu 304048

promotor  
prof. dr hab. inż. Przemysław Rokita

WARSZAWA 2023



## **Etykietowanie muzyki (music tagging) za pomocą metod sztucznej inteligencji**

**Streszczenie.** Rozpoznawanie instrumentów muzycznych w plikach audio jest jednym z najtrudniejszych zagadnień dotyczących wyszukiwania informacji muzycznych. Dostępność danych do wytrenowania sieci neuronowych jest ograniczona. Dotychczasowe rozwiązania oparte na sieciach głębokich wykorzystują uczenie nadzorowane dla predefiniowanych klas, co powoduje, że sieci nie radzą sobie z nowymi klasami instrumentów. Rozwiązaniem przedstawionego problemu jest Few-shot learning. Metoda ta opiera się na kształceniu modelu przy użyciu niewielkiej liczby przykładów, który jest zdolny do szybkiego i efektywnego przyswajania nowych pojęć. Przykładem sieci neuronowej wykorzystywanej w metodzie few-shot learning jest sieć prototypowa. W niniejszej pracy zostanie przeanalizowane jak sieci prototypowe uczą się na różnych zbiorach danych i od czego zależy ich jakość. Wynikiem pracy jest aplikacja umożliwiająca tagowanie plików audio na podstawie jedynie kilku próbek z każdej klasy.

**Słowa kluczowe:** Etykietowanie muzyki, Etykietowanie instrumentów, Uczenie z małą liczbą przykładów, Sieci prototypowe, Meta-uczenie, Ograniczone zbiory danych

## **Music tagging using artificial intelligence methods**

**Abstract.** Recognizing musical instruments in audio files is one of the most difficult tasks in music information retrieval. The availability of data to train neural networks is limited. Existing deep neural network solutions are based on supervised learning for predefined classes, which makes the model unable to cope with new classes of instruments. The solution to the presented problem is Few-shot learning. This method is based on training a model using a small number of examples, which is capable of learning new concepts quickly and efficiently. An example of a neural network used in the few-shot learning method is a prototypical network. This paper will analyze how prototypical networks learn on different datasets and what determines their quality. The result of the work is an application that allows tagging audio files using only several samples from each class as a knowledge base.

**Keywords:** Music classification, Instrument classification, Few-shot learning, Prototypical network, Meta-learning, Limited datasets

# Spis treści

<b>1. Wprowadzenie</b>	7
1.1. Tagowanie muzyki	7
<b>2. Obróbka danych muzycznych</b>	9
2.1. Mel spektrogram	9
2.2. Analiza bibliotek	9
2.2.1. Librosa	9
2.2.2. Torchaudio	11
2.2.3. Mirdata	12
2.3. Dane muzyczne w formacie MIDI	13
<b>3. Sieci neuronowe w problemie tagowania muzyki</b>	14
3.1. Konwolucyjne sieci neuronowe	14
3.2. Sieć neuronowa konwolucyjno-rekurencyjna	14
<b>4. Ograniczone zbiory danych</b>	15
<b>5. Metody meta-uczenia</b>	17
5.1. Metoda meta-uczenia oparta na metryce	17
5.2. Uczenie metodą optymalizacji	18
5.3. Meta-uczenie oparte na modelu	19
<b>6. Prototypową sieć neuronową</b>	20
6.1. Trening epizodyczny	20
6.2. Sieć prototypowa	21
6.2.1. Prototyp	21
6.2.2. Algorytm uczenia sieci prototypowej	21
<b>7. Zbiory danych</b>	23
7.1. TinySol	23
7.1.1. Rozkład klas	23
7.2. GoodSounds	24
7.2.1. Rozkład klas	25
7.3. Medley-solos-DB	26
7.3.1. Rozkład klas	26
7.4. IRMAS	27
7.4.1. Rozkład klas	27
<b>8. Eksperymenty</b>	30
8.1. Budowa sieci	30
8.2. Wyznaczenie parametrów	30
8.2.1. Współczynnik uczenia oraz liczba epizodów	30
8.2.2. Liczba przykładów przypadających na klasę	31
8.2.3. Liczba klas	33

8.2.4. Długość próbek . . . . .	33
<b>9. Wyniki . . . . .</b>	<b>35</b>
9.1. Wyniki wytrenowanych sieci na różnych zbiorach danych . . . . .	35
9.1.1. Dla zbioru danych TinySOL . . . . .	36
9.1.2. Dla zbioru danych GoodSounds . . . . .	37
9.1.3. Dla zbioru danych Medley-solos-DB . . . . .	38
9.1.4. Dla zbioru danych IRMAS . . . . .	40
9.2. Analiza przestrzeni osadzenia . . . . .	41
9.3. Test na nowym zbiorze danych . . . . .	43
9.4. Wnioski . . . . .	44
<b>10. Aplikacja . . . . .</b>	<b>45</b>
10.1. Instrukcja dla użytkownika . . . . .	45
<b>11. Podsumowanie . . . . .</b>	<b>47</b>
<b>12. Perspektywy rozwoju projektu . . . . .</b>	<b>47</b>
<b>Bibliografia . . . . .</b>	<b>49</b>
<b>Wykaz symboli i skrótów . . . . .</b>	<b>51</b>
<b>Spis rysunków . . . . .</b>	<b>51</b>
<b>Spis tabel . . . . .</b>	<b>52</b>

# 1. Wprowadzenie

Tagowanie muzyki polega na automatycznym przypisywaniu odpowiednich tagów do plików audio. Przykładem informacji zawartych w utworze są występujące w nim instrumenty. Dużym utrudnieniem w rozwoju tagowania muzyki, w tym instrumentów są ograniczenia związane z dostępnymi zbiorami danych. W niniejszej pracy zostaną poddane analizie dostępne zbiory danych oraz ograniczenia, które się z nimi wiążą. Analizie zostaną poddane takie zbiory jak: TinySol, GoodSounds, Medley-solos-DB oraz IRMAS, przedstawiające różne typy plików audio. Ze względu na ograniczone zbiory danych zostanie przedstawiona metoda etykietowania instrumentów za pomocą metod sztucznej inteligencji, która polega na uczeniu z małą liczbą próbek. Jedną z metod, która przełamuje te ograniczenia, jest prototypowa sieć neuronowa operująca na zbiorach wsparcia oraz zapytań. Celem pracy jest analiza różnych konfiguracji sieci prototypowej wykorzystywanej w uczeniu z małą liczbą próbek na różnych zbiorach danych oraz stworzenie aplikacji umożliwiającej etykietowanie plików audio zawierających nagrania instrumentów dla nowych klas.

## 1.1. Tagowanie muzyki

Potrzeba tagowania oraz analizy muzyki powstała wraz z rozwojem aplikacji związanych ze światem muzyki. Korzystając z takich aplikacji jak Spotify, Youtube Music czy Tidal spotykamy się z różnymi playlistami, gatunkami czy też systemami rekomendacyjnymi. Początkowo głównym celem tagowania muzyki była potrzeba stworzenia algorytmu ułatwiającego automatyczną organizację repozytoriów i wyszukiwanie muzyki według określonych tagów[1]. Aby uniknąć ręcznego wprowadzania oraz aktualizacji danych na temat każdego pliku audio powstało zapotrzebowanie na rozwiązanie z dziedziny sztucznej inteligencji.

Podstawowe tagi:

- Gatunek
- Nastroje
- Instrumenty
- Wokale (lub same ich wystąpienie)
- Brzmienie

Wraz z rozwojem platform streamingowych muzyki powstała potrzeba rozwoju systemu rekomendacji. Tagowanie muzyki stanowi kluczowy element w procesie generowania personalizowanych rekomendacji muzycznych. Rekomendację utworów jest dokonywana na podstawie harmonii, melodii, instrumentów, za pomocą wyszukiwania podobieństwa między artystami czy też obecność utworów na listach odtwarzania. Otrzymane tagi oraz informację pozyskane od użytkownika są wykorzystywane przez algorytm do analizy preferencji, a następnie do rekomendacji utworów muzycznych. W celu doboru właściwej

muzyki dla każdego użytkownika wykorzystywane są profile użytkowników składające się z poprzednich interakcji, jak również potencjalnie każde inne źródło informacji o użytkowniku, takie jak cechy osobowości.

Prezentowana praca skupia się na problemie tagowania instrumentów[2]. Tagowanie instrumentów może być wykorzystywane do organizacji muzyki. Dzięki rozpoznaniu instrumentów jesteśmy w stanie wyszukiwać utwory z określonymi instrumentami, co ułatwia organizację muzyki i odnajdywanie podobnych utworów. Kolejnym zastosowaniem jest rekomendacja muzyczna[3], która polega na analizie preferencji użytkownika i rekomendacji utworów zawierających określone instrumenty. Dzięki rekomendacji na podstawie instrumentów jesteśmy w stanie tworzyć spersonalizowane playlisty lub wyszukiwać podobnych artystów.

Istnieje wiele innych możliwych tagów, a za pomocą dodatkowych danych[4] takich jak np. teksty piosenek, jesteśmy w stanie wykryć nastrój i przekaz tekstu. Dzięki uzyskanym tagom, umożliwiamy użytkownikom odnajdywanie i organizowanie swojej muzyki, a także udostępnianie tych informacji innym użytkownikom.



## 2. Obróbka danych muzycznych

Obróbka danych muzycznych polega na przetwarzaniu informacji zawartych w plikach audio, takich jak dane o utworach, artystach i instrumentach, aby umożliwić ich oznaczanie i organizację. Początkowym etapem jest przetwarzanie pliku audio w celu zgromadzenia informacji, takich jak długość utworu, format pliku oraz metadane. Analiza metadanych pozwala na uzyskanie danych takich jak tytuł utworu, gatunek, instrumenty, tonację i inne.

### 2.1. Mel spektrogram

Mel spektrogram jest często stosowany do klasyfikacji dźwięku[5]. Aby zrozumieć czym jest mel spektrogram, należy zrozumieć czym jest spektrogram i jakie niesie ze sobą informację.

Spektrogram jest graficzną reprezentacją sygnału w czasie i częstotliwości. Jest to analiza widma sygnału w czasie. Z pomocą okna wyznaczamy widmo sygnału, które odzwierciedla składowe dla poszczególnych częstotliwości. Spektrogram pozwala na określenie, jakie częstotliwości są obecne w danym sygnale i jakie są ich amplitudy. Poprzez analizę relacji harmonicznych między różnymi częstotliwościami możemy również otrzymać harmonię dźwięku.

Ponieważ ludzkie ucho nie postrzega częstotliwości w skali liniowej, powstała nowa skala reprezentująca tonację dźwięku, aby równe odległości w tonacji brzmiały dla słuchacza jednakowo daleko. Powstała skala nazywana jest skalą Melową. Skala Mel naśladuje sposób działania ludzkiego ucha. Odzwierciedla ona naturalną odpowiedź układu słuchowego na pobudzenie dźwiękami, czyli jest to odtworzenie odbioru dźwięku przez ludzkie ucho. Mel-spektrogram jest to spektrogram, w którym częstotliwości są przekształcane na skalę melową. Jest wykorzystywany w uczeniu maszynowym dotyczącym plików audio.

### 2.2. Analiza bibliotek

Poniżej zostaną przedstawione podstawowe metody obróbki pliku audio za pomocą języka Python. Przedstawione biblioteki pozwolą na wczytywanie, analizowanie i przetwarzanie pliku audio.

#### 2.2.1. Librosa

Librosa jest pakietem Pythona służącym do analizy muzyki i dźwięku, który dostarcza elementy niezbędne do tworzenia systemów wyszukiwania informacji o muzyce.

- Wczytanie pliku

Ładowanie i dekodowanie pliku audio jako szereg czasowy (jednowymiarowa tablica audio)

```
y, sr = librosa.load('trumpet.wav')
```

y - Audio buffers (bufor audio)

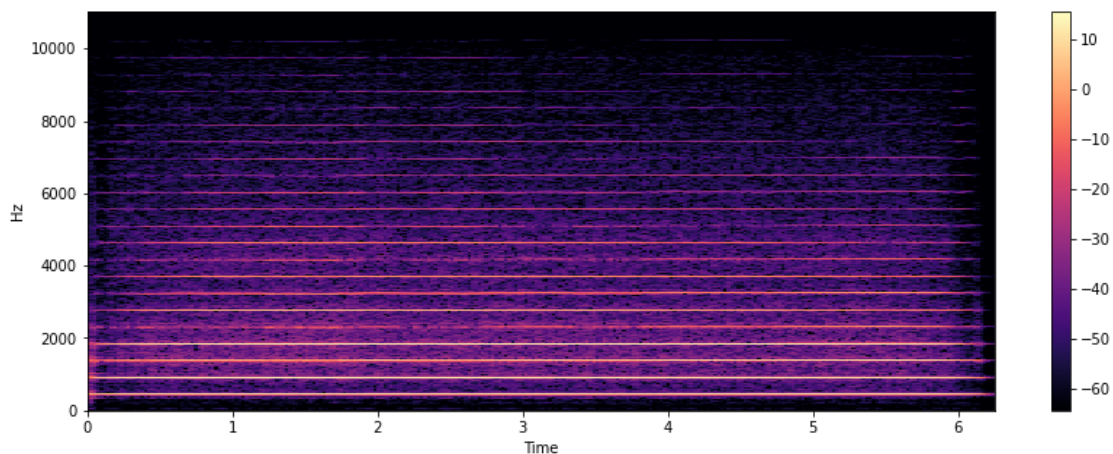
sr - Sampling rate (Częstotliwość próbkowania, liczba próbek na sekundę)

- Czas trwania pliku audio w sekundach

```
duration = librosa.get_duration(y=y, sr=sr)
```

- Spektrogram

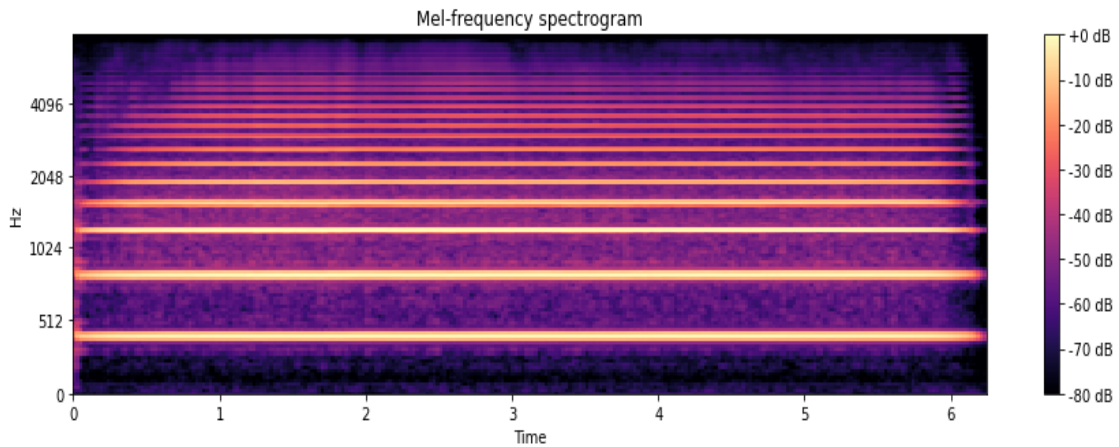
```
Y = librosa.stft(y) # Short-time Fourier transform (STFT)
# Konwersja spektrogramu amplitudy na spektrogram ze skalą dB
Ydb = librosa.amplitude_to_db(abs(Y))
librosa.display.specshow(Ydb, sr=sr, x_axis='time', y_axis='hz')
```



**Rysunek 2.1.** Spektrogram dla próbki dźwięku trąbki - librosa

- Mel spektrogram

```
S = librosa.feature.melspectrogram(y=y, sr=sr)
fig, ax = plt.subplots()
S_db = librosa.power_to_db(S, ref=np.max)
img = librosa.display.specshow(S_db, x_axis='time',
                                y_axis='mel', sr=sr,
                                fmax=8000, ax=ax)
fig.colorbar(img, ax=ax, format='%+2.0f dB')
```



Rysunek 2.2. Mel spektrogram dla próbki dźwięku trąbki - librosa

### 2.2.2. Torchaudio

Torchaudio to wbudowana w PyTorch biblioteka do przetwarzania plików audio. Jest ściśle zintegrowane z PyTorch, co oznacza, że umożliwia korzystanie z danych audio w połączeniu z sieciami neuronowymi i innymi narzędziami PyTorch.

- Wczytanie pliku

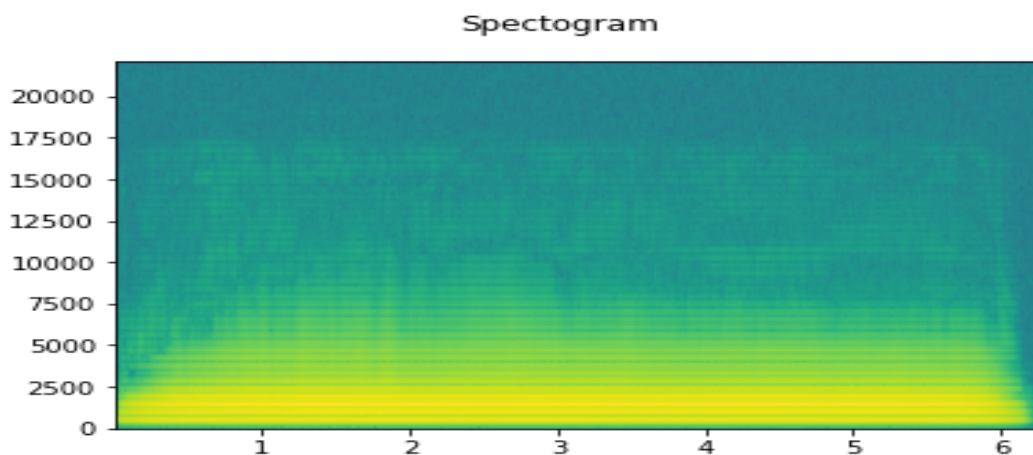
Umożliwia załadowanie tensora z pliku audio

```
waveform, sample_rate = torchaudio.load('trumpet.wav')
```

- Spektrogram

Zwraca spektrogram w formie tensora

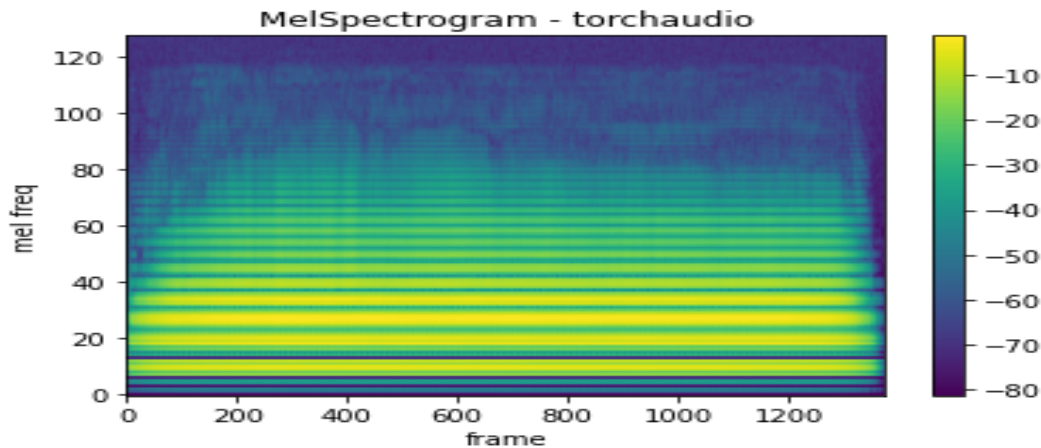
```
specgram = torchaudio.transforms.Spectrogram()(waveform)
```



Rysunek 2.3. Spektrogram dla próbki dźwięku trąbki - torchaudio

- Mel spektrogram

```
mel_spectrogram = torchaudio.transforms.MelSpectrogram()  
                    (waveform)
```



**Rysunek 2.4.** Mel spektrogram dla próbki dźwięku trąbki - torchaudio

### 2.2.3. Mirdata

Mirdata to biblioteka, która ma na celu standaryzację dostępu do zestawów danych audio w Pythonie, eliminując potrzebę pisania niestandardowych modułów do ładowania danych ze źródeł w każdym projekcie i poprawiając odtwarzalność. Umożliwia pobierania zbiorów danych do wspólnej lokalizacji i wspólnego formatu.

#### **Lista wszystkich dostępnych zbiorów danych.**

Wybieramy jeden z dostępnych zbiorów danych w za pomocą zainicjowania go. Wszystkie moduły do ładowania danych zawierają funkcję `download()`, która pozwala pobrać kanoniczną wersję zbioru danych.

```
tinysol = mirdata.initialize('tinysol')  
tinysol.download()
```

W omawianej pracy początkowo korzystałam z powyższej biblioteki, lecz konieczność pobierania zbioru danych za każdym razem podczas uruchamiania kodu była dość uciążliwa dla dużych zbiorów danych. Zdecydowałam się więc na pobranie oryginalnych zbiorów danych oraz korzystanie z wersji lokalnej lub wersji udostępnionej na chmurze. Pozwoliło to na szybszą inicjalizację zbioru danych, jednak wymagało własnej analizy plików i wyszukiwanie metadanych.

Biblioteka *mirdata* jest dobrym rozwiązaniem dla użytkowników, którzy mają dostęp do szybkiego internetu lub potrzebują jedynie dostępu do zbioru danych sporadycznie,

gdyż pobieranie zbioru danych za każdym razem przy uruchomieniu kodu w celu wytre-  
nowania sieci jest nieefektywne i czasochłonne.

### 2.3. Dane muzyczne w formacie MIDI

MIDI jest cyfrowym interfejsem instrumentów muzycznych. Umożliwia komputerom, kartom dźwiękowym i synteзаторom wymieniać informację o dźwiękach i komendach dotyczących ich odtwarzania między sobą. MIDI nie posiada danych audio, a jedynie informację o dźwięku. Popularne notacje:

- MusicXML

Uniwersalny format dla powszechnie stosowanej zachodniej notacji muzycznej. Jest to plik w formacie XML.

- Notacja ABC

Notacja ABC to prosty, ale wydajny format notacji muzycznej ASCII. Istnieje wiele pakietów oprogramowania umożliwiających konwersję zapisu ABC na MIDI, tworzenie nut, odtwarzanie pliku przez głośnik komputera itp.

- GUIDO

Format tekstowy, zdolny do reprezentowania wszystkich informacji zawartych w konwencjonalnych partyturach muzycznych.

Powyższy format przetrzymywania oraz przesyłania informacji o dźwięku i jego cechach nie jest wykorzystywany w omawianej pracy, przez co nie został szczegółowo omówiony. Jest on jednak wykorzystywany w dziedzinie badań MIR (Music Information Retrieval) w celu wytwarzania muzyki za pomocą nauczania maszynowego.

### **3. Sieci neuronowe w problemie tagowania muzyki**

W dziedzinie badań zajmujących się analizą i wyszukiwaniem informacji o muzyce stosuje się różne metody sztucznej inteligencji, takie jak uczenie maszynowe, algorytmy genetyczne czy sieci neuronowe. Dla problemu tagowania muzyki wykorzystywane są między innymi sieci neuronowe, w większości sieci bazujące na sieciach konwolucyjnych[6].

#### **3.1. Konwolucyjne sieci neuronowe**

Jednym z najczęściej stosowanych rodzajów sieci neuronowych jest sieć konwolucyjna[6]. Jest to sieć neuronowa stosowana głównie do przetwarzania obrazów oraz do wykrywania ich cech[7]. Sieci te składają się z neuronów zorganizowanych w 3 wymiarach, które są optymalizowane podczas uczenia. Model przyjmuje na wejściu wektor obrazu, który następnie przechodzi przez wiele warstw konwolucyjnych, poolingu, a na samym końcu przez warstwę w pełni połączoną służącą do klasyfikacji. Warstwy konwolucyjne stosujące filtry o zadanym rozmiarze, odpowiedzialne są za uczenie sieci pewnych wzorców i cech w obrazie. Nauka cech polega na filtracji danych wejściowych, czyli przesuwania się filtra przez dane wejściowe oraz obliczaniu iloczynu skalarnego dla każdej wartości w jądrze. Następnie przefiltrowany obraz zostaje przekazany do warstwy w pełni połączonej, która ma na celu wytworzenia wyników klasy z aktywacji do użycia do klasyfikacji. Dodatkowo dane mogą zostać znormalizowane oraz zredukowane w celu redukcji liczb parametrów. Po przejściu przez warstwy konwolucyjne dane obrazu zostają "spłaszczone" do jednego wymiaru. Kolejnymi warstwami sieci neuronowej są warstwy ukryte, które pełnią funkcję klasyfikatora. Ich zadaniem jest rozpoznanie różnych klas, w przypadku omawianej pracy są to różnego rodzaju instrumenty.

#### **3.2. Sieć neuronowa konwolucyjno-rekurencyjna**

Sieć neuronowa konwolucyjno-rekurencyjna może być opisana jako połączenie konwolucyjnej sieci neuronowej i sieci rekurencyjnej[8]. CRNN jest zmodyfikowaną siecią konwolucyjną, przez zastąpienie ostatnich warstw konwolucyjnych przez rekurencyjne. Ideą CRNN jest wykorzystanie sieci konwolucyjnych do ekstrakcji cech oraz do czasowego podsumowania wyodrębnionych cech przez sieci rekurencyjne. Sieci rekurencyjne umożliwiają globalną analizę danych, która umożliwia wydobywanie takich tagów jak nastrój utworu.

## 4. Ograniczone zbiory danych

Wykorzystując architektury uczenia głębokiego mamy do czynienia z uczeniem nadzorowanym[9] z predefiniowanymi klasami. Aby dana sieć uzyskiwała zadowalające wyniki, zbiór danych musi być wystarczająco obszerny, aby sieć neuronowa była w stanie nauczyć się pewnych wzorców. Zbiory danych są również ograniczone co do występujących w nich klas, co oznacza, że model wytrenowana na określonym zbiorze danych, jest ograniczona do klas występujących w nim. Sieć neuronowa nie radzi sobie z nienapotkanymi wcześniej klasami. Dostępność zbiorów danych dla problemu tagowania muzyki, w tym instrumentów, jest ograniczona[2]. Jest to spowodowane kilkoma utrudnieniami:

- Powstawanie nowych gatunków oraz instrumentów muzycznych  
Muzyka charakteryzuje się ciągłymi zmianami oraz rozwojem. Na rozszerzanie zbiorów muzycznych ma wpływ powstawanie nowych gatunków muzycznych, ewolucje gatunku oraz powstawanie nowych instrumentów.
- Subiektywność  
Podczas tagowania utworu mamy do czynienia z subiektywną oceną człowieka. Muzyka nie jest jednoznaczna. Spowodowane jest to różnym odbiorem oraz wiedzą w zależności od słuchacza. Na muzykę mają wpływ uczucia, co może powodować zmianę oceny nastroju utworu w zależności od aktualnego samopoczucia oceniającego.
- Prawa autorskie  
Aby móc wykorzystać utwory muzyczne objęte prawami autorskimi należy uzyskać pozwolenie od autora. Wiąże się to z dużymi kosztami, co powoduje zaniechanie tworzenia nowych, większych zbiorów danych lub udostępnianie ich tylko na licencję oraz wykorzystywanie ich tylko w określony sposób.
- Czasochłonność wraz z kosztami  
Stworzenie zbioru danych wymaga wielu godzin pracy. Opracowanie rozległego zbioru danych wymaga przesłuchania setek godzin plików audio. Proces tagowania utworu wymaga niekiedy wielokrotnego przesłuchania nagrania przez wykwalifikowanych specjalistów, co wiąże się z wysokimi kosztami oraz wymaga znacznego zaangażowania czasowego.
- Braki w zbiorach danych  
Niestety, często spotykamy się z niepełnymi danymi muzycznymi, co skutkuje brakiem szczegółowych informacji na temat utworu, wykonawcy czy albumu.

Powyższe ograniczenia zbiorów danych, jak i trenowanych na nich sieci głębokich uniemożliwiają nam rozszerzanie klasyfikacji o nowe instrumenty, artystów czy gatunki. Dlatego poszukujemy metody, która pozwoli naszej sieci na uczenie się nowych klas na podstawie nielicznych przykładów, niewystępujących w zbiorze treningowym.

Rozwiązaniem przedstawionego problemu jest **Few-shot learning**[10]. Metoda ta opiera się na kształceniu modelu przy użyciu niewielkiej liczby przykładów dla danej klasy. Efektem jest zdolność rozpoznawania nowych klas na podstawie nielicznej liczby przykładów, dzięki czemu jest to bardziej elastyczne i skuteczne podejście dla tagowania klas o małej liczbie danych. Ideą tego podejścia jest opracowanie modelu, który jest zdolny do szybkiego i efektywnego przyswajania nowych pojęć. Innym podejściem jest uczenie bez użycia przykładów (**Zero-shot learning**)[11][12]. Jest to metoda, która umożliwia przewidywanie nowych, dotąd niespotkanych wcześniej klas. Umożliwia to wykorzystywanie informacji pobocznych o charakterystyce klasy, pewnych jej cech, co daje możliwości modelowi na odkrywanie związków między klasami i na podstawie tych związków, na dokonywanie wniosków.

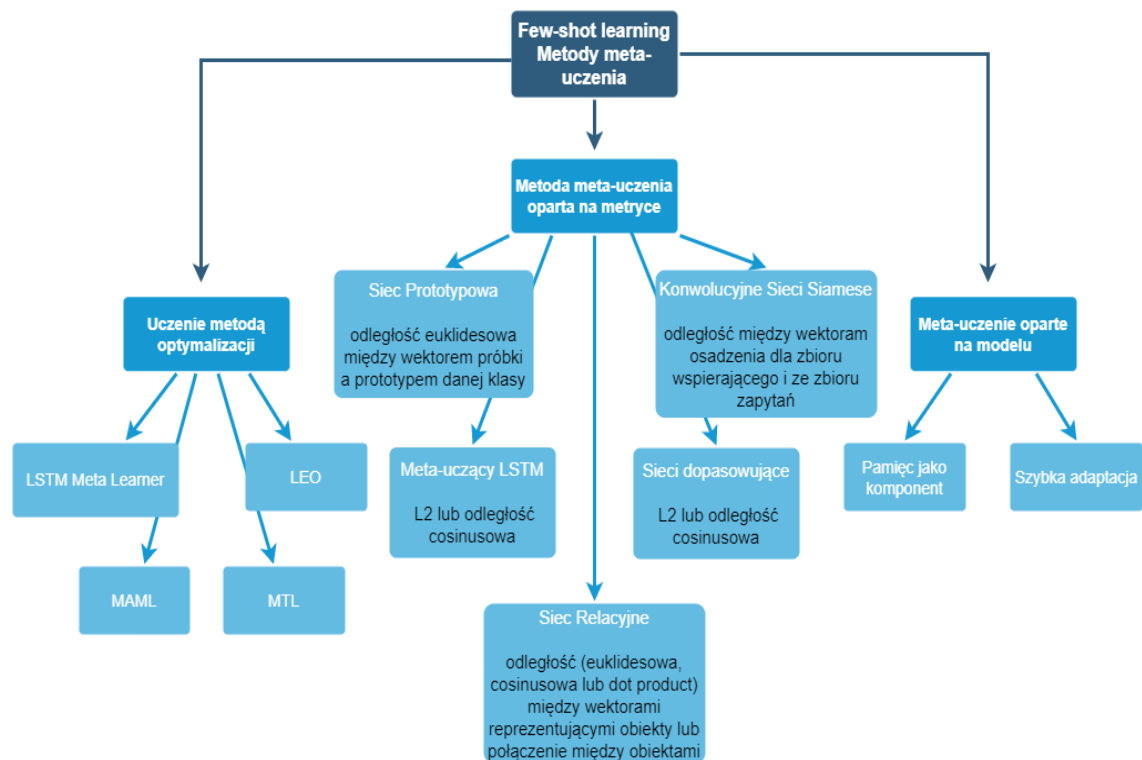
FSL (Few-shot Learning) oraz ZSL (Zero-shot Learning) pozwalają nam na pozbycie się ograniczeń wynikających z konieczności posiadania stałego i wcześniej określonego zbioru klas, dla których posiadamy odpowiednią liczbę etykietowanych danych. Dzięki tym metodom jesteśmy w stanie rozszerzać nasze modele o nowe klasy, bez konieczności posiadania dużej liczby przykładów dla każdej z nich. Niniejsza praca koncentruje się przede wszystkim na metodzie FSL, dlatego w dalszej zostanie przedstawiona ta metoda oraz jej wykorzystaniu.



## 5. Metody meta-uczenia

Few-shot learning wykorzystuje Meta-Learning lub Learning to Learn[10]. **Meta-learning** to podejście z dziedziny uczenia maszynowego, które polega na wydobywaniu istotnych informacji z dotychczasowych doświadczeń, co prowadzi do skuteczniejszego i szybszego uczenia się nowych zadań. Gdy model oparty na metodzie meta-learningu spotyka nowe zadanie, dzięki zgromadzonym dotychczasowym doświadczeniom, jest w stanie szybciej i lepiej je rozwiązać niż standardowe podejście.. Meta-learning polega na gromadzeniu doświadczeń z wielu podobnych zadań, a następnie wykorzystywaniu ich do rozwiązywania nowych zadań. **Learning to Learn** to podejście, które pozwala modelowi na uczenie się samego procesu uczenia. Dzięki temu model jest w stanie dostosowywać swoje metody uczenia, aby były one bardziej skuteczne dla zadanego problemu.

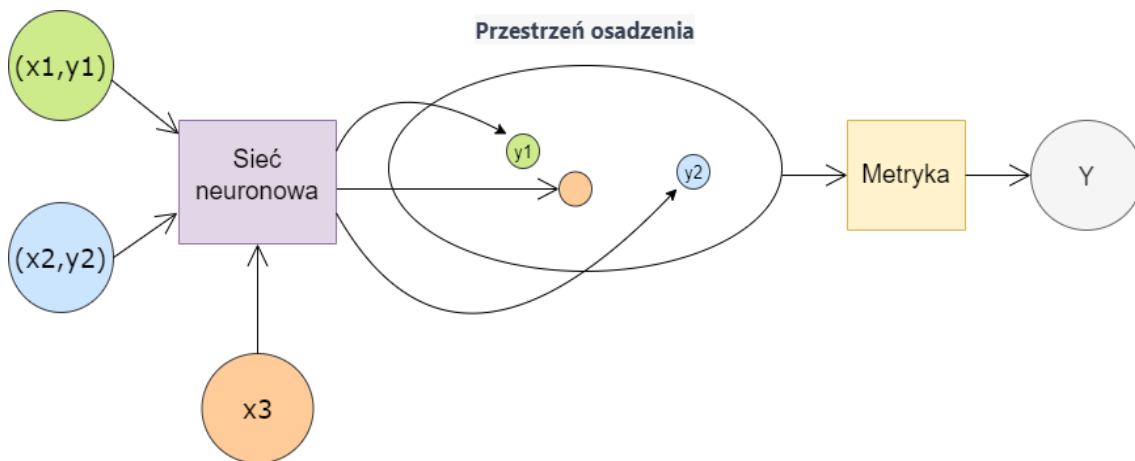
FSL oparte na meta-uczeniu można podzielić na trzy podejścia oparte na metryce, oparte na optymalizacji i oparte na modelu[10].



Rysunek 5.1. Podział modeli dla problemu uczenia z małą ilością przykładów

### 5.1. Metoda meta-uczenia oparta na metryce

Uczenie oparte na metryce polega na znajdowaniu odległości między próbkami danych. Sieci neuronowe są w stanie nauczyć się przestrzeni osadzenia, w której przykłady tej samej klasy są blisko siebie według określonej metryki. Na podstawie metryki porównujemy dane ze zbioru zapytań do przykładów osadzonych w przestrzeni ze zbioru wspierającego.



Rysunek 5.2. Przestrzeń osadzenia

Rozważamy 2 przykłady, para obraz-klasa  $(x1, y1)$  oraz  $(x2, y2)$ , należące do zbioru wspierającego oraz daną  $x3$  ze zbioru zapytań. Za pomocą sieci neuronowej umieszczamy dane w przestrzeni osadzenia. Dla osadzonych punktów obliczamy według pewnej metryki podobieństwo danej  $x3$  do przykładów ze zbioru wspierającego. Następnie przypisujemy klasę ze zbioru klas  $K$ .

Ideą uczenia metrycznego jest nauczenie sieci neuronowej przestrzeni osadzenia, w której przykłady tej samej klasy są blisko siebie, a przykłady różnych klas są daleko od siebie. Trening polega na losowym wyborze  $K$ -way  $N$ -shot dla danego epizodu. Każdy epizod ma oddzielny zestaw danych wpierających i zapytań. Model jest uczony za pomocą zbioru zapytań i ich rzeczywistych wartości, zwanych "ground truth", w celu obliczania straty sieci neuronowej i aktualizacji parametrów.

## 5.2. Uczenie metodą optymalizacji

Optymalizacja gradientowa jest metodą polegającą na stopniowym zmienianiu parametrów modelu, tak aby minimalizować funkcję straty. W przypadku uczenia, w którym mamy do czynienia z ograniczoną liczbą przykładów treningowych, trudno jest znaleźć odpowiednie parametry modelu  $\theta$ . Podejścia oparte na optymalizacji skupiają się na uczeniu parametrów, które pozwolą modelowi na łatwe dostosowanie się do nowych zadań, poprzez wykorzystanie metod meta-uczenia i szkolenia epizodycznego.

Uczenie dzielimy na dwa etapy:

1. Learner. Jest on uczony specyficznie dla danego zadania. Jednak sam model uczony za pomocą metody gradientu prostego nie jest w stanie generalizować.
2. Meta-learner. Nie jest on wyspecyfikowany dla danego zadania, zamiast być skoncentrowany tylko na jednym zadaniu jest on trenowany na zbiorze różnych zadań  $T$ . Podczas treningu epizodycznego, meta-learner  $\phi$  aktualizuje parametry  $\theta$  mo-

delu specjalizowanego, znanego jako Learner, za pomocą małego zbioru danych dla danego zadania  $D^{train}$ .

$$\theta^* = g_\phi(\theta, D^{train}) \quad (1)$$

Celem modelu meta-uczenia jest zaktualizowanie Lenera o parametrach  $\theta^*$ , który jest lepiej dostosowany do nowych zadań niż model trenowany tylko na standardowym zbiorze danych.

Podczas uczenia proces optymalizacji polega na dostosowywaniu parametrów meta-lenera  $\phi$  oraz Learnera  $\theta$  dla poszczególnych zadań. Podczas treningu, meta-lerner jest w stanie zgromadzić wiedzę z różnych zadań i wykorzystać ją do dostosowywania parametrów Learnera dla nowego zadania. Dążymy, aby parametr  $\theta$  był adaptowany do nowych zadań na podstawie kilku przykładów etykietowanych dla nowego zadania. Celem jest stworzenie parametrów, które są nie tylko skuteczne dla aktualnego zadania, ale również łatwe do adaptacji dla nowych zadań.

### 5.3. Meta-uczenie oparte na modelu

Metody oparte na modelu nie obliczają prawdopodobieństwa wyjścia  $y$  dla wejścia  $x$  przy użyciu parametrów modelu  $\theta$ . Zamiast tego, metoda ta polega na skonstruowaniu specjalnie zaprojektowanych architektur modelu, które pozwalają na szybkie uczenie. Modele te możemy podzielić na podstawie ich architektur: oparte na pamięci oraz na szybkiej adaptacji.

#### Pamięć jako komponent

Integracja komponentu pamięci zewnętrznej jest kluczowym elementem rodziny architektur modeli, które umożliwiają skuteczne procesy uczenia się. Ten rodzaj pamięci jest znany jako bank pamięci, macierz pamięci lub po prostu pamięć. Pamięć ta działa jak bufor, do którego sieci neuronowe mogą zapisywać nowe informacje i odzyskiwać wcześniej przechowywane informacje. Pamięć ta pozwala na odciążenie szkolenia sieci neuronowych przy niedużej ilości danych, a także na szybszą generalizację.

#### Szybka adaptacja

Szybka adaptacja wykorzystuje technikę szybkich wag w celu szybkiej adaptacji parametrów modelu dla danego zadania. W przeciwieństwie do tradycyjnej metody gradientu prostego, która jest czasochłonna, ta metoda opiera się na przewidywaniu parametrów sieci neuronowej za pomocą innej sieci. Wagi, które są generowane w ten sposób, nazywane są szybkimi wagami, ponieważ pozwalają na szybką adaptację modelu do nowych danych.

## 6. Prototypową sieć neuronową

W niniejszej pracy zastosujemy prototypową sieć neuronową[13], która jest specjalnie przeznaczona do rozwiązywania problemów uczenia się na małych zbiorach danych. Ta sieć opiera swoje działanie na metodzie meta-uczenia, wykorzystując metryki jako narzędzie do oceny jej skuteczności[10].

### 6.1. Trening epizodyczny

Proces szkolenia sieci neuronowej dla metod meta-uczenia opartych na metryce wykorzystuje metodę treningu epizodycznego. Trening epizodyczny polega na tworzeniu epizodów, w których wybieramy określoną liczbę  $K$  klas ze zbioru klas oraz przykładów dla każdej z nich z głównego zbioru treningowego, tworząc tym samym zbiór wsparcia oraz losowe dane dla każdej klasy, które tworzą zbiór zapytań. Błąd, który jest obliczany dla danego epizodu, jest wykorzystywany do aktualizacji parametrów wag sieci.

---

**Algorithm 1** Trening epizodyczny

---

**Wejście:**  $D = (x_1, y_1), \dots, (x_i, y_i)$

**Dane:**  $D$ : zbiór treningowy,  $i$ : liczba danych,  $K_{train}$ : liczba klas w zbiorze treningowym,  $K_{way}$ : liczba klas użytych w treningu,  $N_{shot}$ : liczba danych w zbiorze wspierającym,  $Q_{volume}$ : liczba danych w zbiorze zapytań,  $n_{episodes}$ : liczba epizodów,  $Model$ : model sieci neuronowej,  $d$ : odległość

---

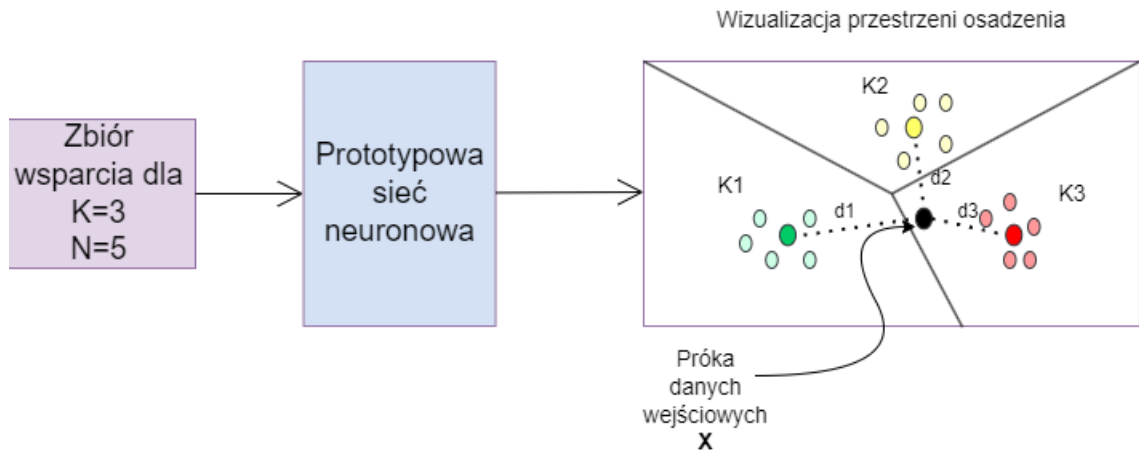
```
1: function EPISODIC-TRAINING( $D$ )
2:   for  $episode$  in  $n_{episodes}$  do
3:      $K \leftarrow RandomSample(K_{train}, K_{way})$ 
4:     for  $k$  in  $K$  do
5:        $SupportSet_k \leftarrow RandomSample(D_k, N_{shot})$ 
6:     end for
7:      $QuerySet \leftarrow RandomSample(K_{train} \setminus SupportSet, Q_{volume})$ 
8:      $TrainNeuralNetModel(Model, SupportSet \cup QuerySet)$  Wytrenuj model,
       aby stworzyć przestrzeń osadzania, w której przykłady tej samej klasy są zgrupowane
       blisko siebie, a przykłady różnych klas są osadzone daleko od siebie.
9:     for  $(x_s, y_s)$  in  $SupportSet$  do
10:      for  $(x_q, y_q)$  in  $QuerySet$  do
11:         $d_{sq} \leftarrow ComputeDistance(x_s, x_q)$ 
12:      end for
13:    end for
14:     $ComputeLoss(d)$ 
15:     $UpdateNeuralNetModelParameters()$ 
16:  end for
```

---

Celem treningu epizodycznego jest stworzenie epizodów, które naśladują problem uczenia na małych zbiorach danych poprzez selekcję podpróbki klas i danych, tworząc dla każdej iteracji epizod jako indywidualne zadanie. W trakcie treningu model jest stale zmuszony do radzenia sobie z nowymi wyzwaniem klasyfikacji  $K$ -shot,  $N$ -way..

## 6.2. Sieć prototypowa

Sieć prototypowa jest siecią neuronową opartą na metryce wykorzystywaną w uczeniu z małą liczbą przykładów[13]. Sieci te wykorzystują przestrzeń osadzenia oraz skupianie się punktów należącej do tej samej klasy blisko siebie. Na podstawie zbioru wspierającego uczymy sieć osadzania tych punktów w przestrzeni, a następnie dla każdej klasy obliczamy **prototyp**. Prototyp danej klasy jest średnią z osadzeń danych ze zbioru wsparcia. Klasyfikacji dokonujemy przez wyszukania najbliższego prototypu dla osadzenia punktu etykietowanego.



Rysunek 6.1. Rozłożenie punktów zbioru wsparcia oraz ich prototypy.

### 6.2.1. Prototyp

Prototyp jest średnią wszystkich osadzeń danych ze zbioru wsparcia dla danej klasy  $k \in K$ . Zbiór wsparcia dla danej klasy oznaczamy  $D_k$ , a funkcję osadzenia z możliwymi do nauczania parametrami  $\phi$  oznaczamy jako  $f_\phi$ . Prototyp  $c$  dla danej klasy  $k$  obliczamy następująco:

$$c_k = \frac{1}{|D_k|} \sum_{(x_i, y_i) \in D_k} f_\phi(x_i) \quad (2)$$

Podobieństwo danej próbki wejściowej  $X$  obliczamy za pomocą kwadratowej odległości euklidesowej  $d$  między osadzeniem próbki a prototypem dla każdej klasy  $k \in K$ .

$$P(y = k|X) = \frac{\exp(-d(f_\phi(X), c_k))}{\sum_{k \in K} \exp(-d(f_\phi(X), c_k))} \quad (3)$$

Następnie jako predykcję wybieramy klasę o największym prawdopodobieństwie.

### 6.2.2. Algorytm uczenia sieci prototypowej

Sieć neuronowa ma na celu nauczyć się przestrzeni osadzenia, w której przykłady tej samej klasy są blisko siebie według określonej metryki. Metryką wykorzystaną w sieci prototypowej jest odległość euklidesowa. Sieć prototypowa wykorzystuje trening epizodyczny,

**Algorithm 2** Uczenie sieci prototypowej dla jednego epizodu

**Dane:**  $K$ : klasy w zbiorze wspierającym,  $D_k$ : zbiór treningowy dla klas  $k$  należących do zbioru  $|K|$ ,  $SupportSet$ : zbiór wspierający,  $QuerySet$ : zbiór zapytań,  $Q_{volume}$ : liczba danych w zbiorze zapytań,  $Model$ : model sieci neuronowej,  $c$ : zbiór prototypów,  $distances$ : odległość między próbką wejściową a prototypem

**Wejście:**  $SupportSet = (SupportSet_1, \dots, SupportSet_k)$  dla  $k = |K|$ ,  $QuerySet = RandomSample(D_k \setminus SupportSet, Q_{volume})$

**Wyjście:**  $logits$

---

```

1: function TRAINPROTOTYPICALNETWORK( $D$ )
2:    $SupportSetEmbeddings \leftarrow ComputeEmbeddings(SupportSet, Model)$ 
3:    $QuerySetEmbeddings \leftarrow ComputeEmbeddings(QuerySet, Model)$ 
4:   for  $k$  in  $K$  do
5:      $c_k \leftarrow ComputePrototype(QuerySet)$ 
6:   end for
7:    $distances \leftarrow ComputeEuclideanDistanceForEachQuery(QuerySet, c)$ 
8:    $logits \leftarrow -(distances^2)$ 

```

---

czyli spojrzenie na problem uczenia sieci podczas każdego epizodu jak na nowy, odrębny problem. Każdy epizod przyjmuje nowy zbiór wsparcia i zbiór osadzenia. Następnie dokonujemy uczenia sieci dla nowych zbiorów, tak jak przedstawiono w algorytmie powyżej. Na podstawie wyjścia sieci obliczamy funkcję straty oraz dokładność sieci w celu dalszego treningu. Aby uzyskać przewidzianą klasę wykorzystujemy funkcję softmax.

$$class = \max(\text{softmax}(logits)) \quad (4)$$

Funkcja softmax jest funkcją aktywacji sieci neuronowej, która służy do obliczania wyjścia sieci neuronowej. Jest stosowana do zadań klasyfikacji, a jej wyjściami są prawdopodobieństwa przynależności do danej klasy. Wybieramy klasę o największym prawdopodobieństwie i przypisujemy jako przewidzianą klasę próbki.

## 7. Zbiory danych

Tagowanie instrumentów muzycznych jest jednym z ważniejszych problemów tagowania muzyki, niestety istnieje niewiele zbiorów danych dedykowanych dla tego typu zadania[2]. Niestety nie istnieje kanoniczny zbiór danych dla problemu tagowania instrumentów, jednak istnieje kilka zbiorów wartych uwagi. W niniejszej pracy zostało opracowane i przeanalizowane działanie sieci neuronowej prototypowej dla jednych z głównych zbiorów danych dla problemu tagowania instrumentów. Następnie dla każdego ze zbiorów został wytrenowany model o różnych parametrach  $K$ -way  $N$ -shot. Poniżej znajduje się przeprowadzona analiza zbiorów danych wykorzystanych do trenowania sieci prototypowej.

### 7.1. TinySol

TinySol jest zbiorem danych składających się z 2913 próbek plików dźwiękowych w formacie **.wav** należących do 14 klas[14]. Każda próbka reprezentuje pojedynczą nutę muzyczną o różnych wysokościach dźwięku i różnych dynamikach. Pliki dźwiękowe w zbiorze danych są nagraniami o różnej długości.

Rozkład instrumentów:

Średnia długość dźwięku - 6.67 s.

Najkrótsza długość pliku dźwiękowego - 1.32 s.

Najdłuższa długość pliku dźwiękowego - 16.24 s.

Odchylenie standardowe - 1.48 s.

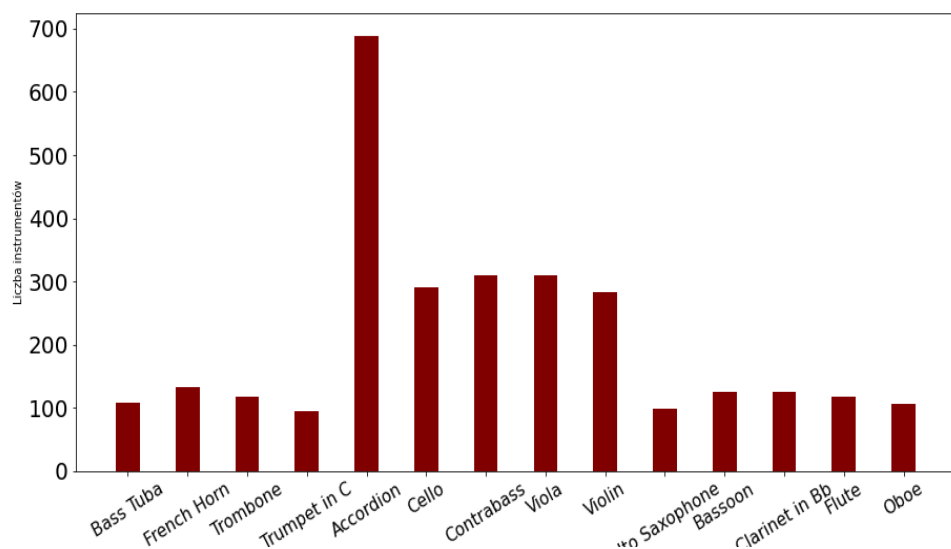
#### 7.1.1. Rozkład klas

Zbiór danych składa się z 14 różnych instrumentów z 4 różnych rodzin: rodzina instrumentów klawiszowych, smyczkowych, dętych blaszanych oraz dętych drewnianych. Poniżej przedstawiono rozkład instrumentów w zbiorze danych.

**Tabela 7.1.** Rozkład instrumentów w zbiorze danych TinySOL

ID	Instrument	Rodzina instrumentów	Ilość próbek
1	Tuba basowa (Bass Tuba)	dętych blaszanych	108
2	Waltornia (French Horn)	dętych blaszanych	134
3	Puzon (Trombone)	dętych blaszanych	117
4	Trąbka w stroju C (Trumpet in C)	dętych blaszanych	96
5	Akordeon (Accordion)	klawiszowych	689
6	Wiolonczela (Cello)	smyczkowych	291
7	Kontrabas (Contrabass)	smyczkowych	309

8	Viola	smyczkowych	284
9	Skrzypce (Violin)	smyczkowych	289
10	Altówka (Alto Saxophone)	smyczkowych	99
11	Fagot (Bassoon)	dętych drewnianych	126
12	Klarnet w stroju Bb (Clarinet in Bb)	dętych drewnianych	126
13	Flet (Flute)	dętych drewnianych	118
14	Obój (Oboe)	dętych drewnianych	107
	<b>Suma:</b>		2913



**Rysunek 7.1.** Wizualizacja rozkładu instrumentów w zbiorze danych TinySol

TinySol zawiera nieduże ilości danych o średniej ilości próbek przypadającej na klasę równej 208. Jest to ograniczony zbiór danych, który mógłby być niewystarczający dla sieci głębokich, jednak jest on bardzo dobrym zbiorem danych dla sieci wykorzystujących metodę uczenia z małą ilością próbek (FSL).

## 7.2. GoodSounds

Zbiór danych GoodSounds jest zbiorem nagrań instrumentów zawierających pojedynczą nutę o różnej jakości wykonania[15]. Każda nuta została uchwycona kilkakrotnie z różnymi charakterystykami tonalnymi, atakami, zanikaniem. Nagrania zostały ocenione przez profesjonalistów w dziedzinie muzyki, a następnie zostały oznaczone jako zagrane dobrze lub źle. Dla treningu zostały wykorzystane nagrania z mikrofonu 'neumann'.

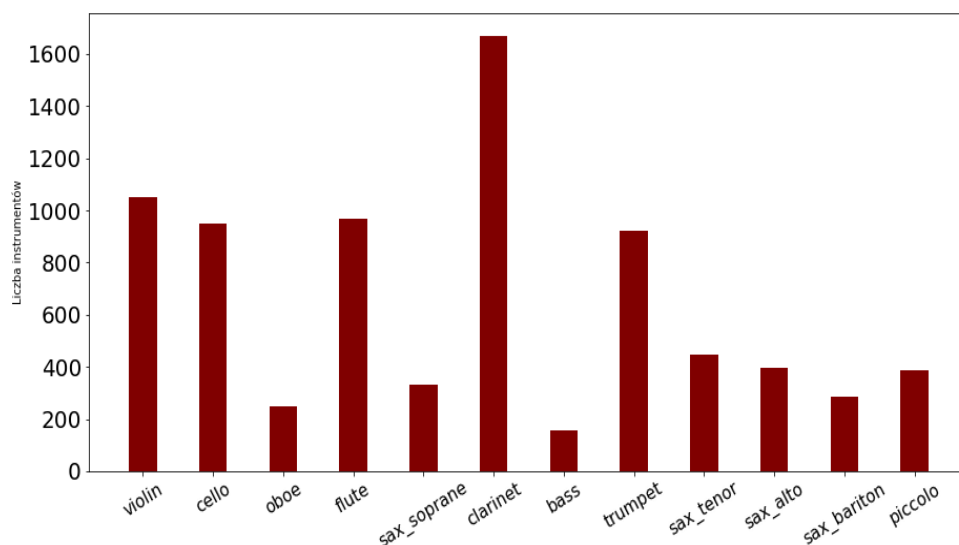


### 7.2.1. Rozkład klas

Zbiór danych składa się z 12 różnych instrumentów z 5 różnych rodzin instrumentów oraz wokalu. Poniżej przedstawiono rozkład instrumentów w zbiorze danych.

**Tabela 7.2.** Rozkład instrumentów w zbiorze danych GoodSounds

ID	Instrument	Rodzina instrumentów	Ilość próbek
1	Wiolonczela (cello)	strunowych smyczkowych	951
2	Bas(bass)	strunowych	159
3	Skrzypce (violin)	strunowych smyczkowych	1053
4	Klarnet (clarinet)	dętych drewnianych	1671
5	Flet (flute)	dętych drewnianych	968
6	Flet piccolo (piccolo)	dętych drewnianych	388
7	Obój (oboe)	dętych drewnianych	247
8	Saksofon tenorowy (sax tenor)	dętych blaszanych	449
9	Saksofon altowy (sax alto)	dętych blaszanych	395
10	Trąbka (trumpet)	dętych blaszanych	920
11	Saksofon barytonowy (sax bariton)	dętych blaszanych	288
12	Saksofon sopranowy (sax soprane)	dętych blaszanych	334
	<b>Suma:</b>		7823



**Rysunek 7.2.** Wizualizacja rozkładu instrumentów w zbiorze danych GoodSounds

GoodSounds zawiera nieduże ilości danych o średniej ilości próbek przypadającej na klasę równej 652. Jest on większy od zbioru TinySOL oraz zawiera szersze reprezentacje nut. Jest on jednak niezrównoważony, klasa o najmniejszej liczbie próbek wynosi 159, a klasa o największej liczbie posiada 1671 próbek. Odchylenie standardowe wynosi 435.

### 7.3. Medley-solos-DB

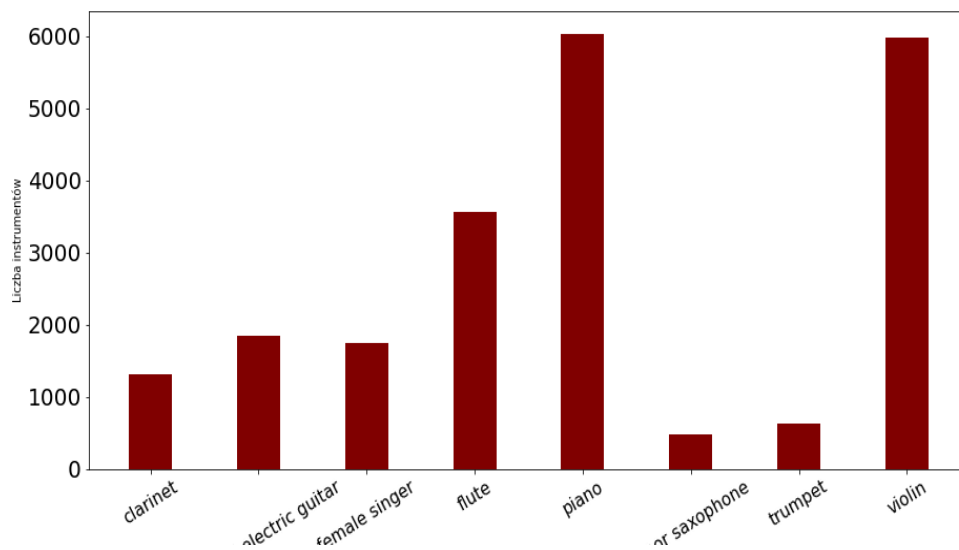
Medley-solos-DB jest zbiorem danych przeznaczonym dla problemu klasyfikacji instrumentów[16]. Zbiór danych składa się z plików audio wydobytych ze zbiorów danych MedleyDB[17] oraz solosDB. W odróżnieniu do powyższych zbiorów danych, Medley-solos-DB jest zbiorem krótkich, 3-sekundowych utworów dla pojedynczego instrumentu muzycznego, a nie nagraniem pojedynczej nuty. Zbiór danych składa się z 21572 plików **.wav** dla 8 klas.

#### 7.3.1. Rozkład klas

Zbiór danych składa się z 8 różnych instrumentów z 5 różnych rodzin instrumentów oraz wokalu. Poniżej przedstawiono rozkład instrumentów w zbiorze danych.

**Tabela 7.3.** Rozkład instrumentów w zbiorze danych Medley-solos-DB

ID	Instrument	Rodzina instrumentów	Ilość próbek
1	Klarnet (clarinet)	dętym drewnianych	1311
2	Przesterowana gitara elektryczna (distorted electric guitar)	strunowych szarpanych	1854
3	Wokalistka (female singer)	—	1744
4	flet (flute)	dętych drewnianych	3555
5	Pianino (piano)	klawiszowych	6032
6	Saksofon tenorowy (tenor saxophone)	dętym drewnianym	477
7	Trąbka (trumpet)	dętych blaszanych	627
8	Skrzypce (violin)	smyczkowych	5971
	<b>Suma:</b>		21572



**Rysunek 7.3.** Wizualizacja rozkładu instrumentów w zbiorze danych Medley-solos-DB

Jest to zbiór danych bogaty w różne przykłady jednak dla ograniczonej liczby 8 klas. Przez znacznie większy rozmiar niż zbiór danych TinySol czy GoodSounds, jest wykorzystywany w treningu sieci przeznaczonych do klasyfikacji instrumentów[18].

#### 7.4. IRMAS

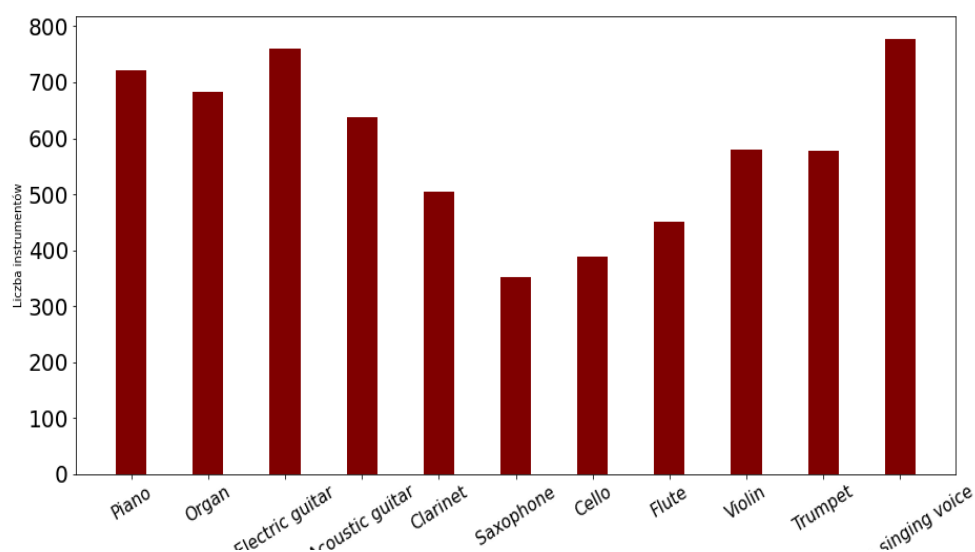
Zbiór danych IRMAS jest najbardziej złożonym zbiorem danych z wyżej wymienionych. Zawiera on fragmenty muzyczne z adnotacjami instrumentów[19]. We wcześniejszych zbiorach mieliśmy do czynienia z pojedynczymi nutami lub fragmentami muzycznymi dla pojedynczego instrumentu. IRMAS proponuje zbiór plików audio zawierający utwory muzyczne z dominującym instrumentem. Ma on na celu wytrenowanie modelu do rozpoznawania dominujących instrumentów w plikach audio.

##### 7.4.1. Rozkład klas

Zbiór danych składa się z 11 różnych klas, w tym głos ludzki. Poniżej przedstawiono rozkład instrumentów w zbiorze danych. Zbiór trenujący zawiera 6705 plików **.wav**. Każdy plik jest 3-sekundowym nagraniem z ponad 2000 różnych nagrań z różnych gatunków muzycznych. Zbiór danych zawiera fragmenty melodii. Pliki audio pochodzą z różnych gatunków muzycznych z różnymi towarzyszącymi instrumentami.

**Tabela 7.4.** Rozkład instrumentów w zbiorze danych treningowych IRMAS

ID	Instrument	Rodzina instrumentów	Ilość próbek
1	Organy (Organ)	klawiszowych	682
2	Głos ludzki (Human singing voice)	—	778
3	Flet (Flute)	strunowych smyczkowych	451
4	Gitara akustyczna (Acoustic guitar)	strunowych	637
5	Pianino (Piano)	klawiszowych	721
6	Wiolonczela (Cello)	strunowych smyczkowych	388
7	Trąbka (Trumpet)	dętych drewnianych	577
8	Saksofon (Saxophone)	dętych blaszanych	352
9	Skrzypce (Violin)	strunowych smyczkowych	580
10	Klarnet (Clarinet)	dętych blaszanych	505
11	Gitara elektryczna (Electric guitar)	strunowych	760
	<b>Suma:</b>		6431

**Rysunek 7.4.** Wizualizacja rozkładu instrumentów w zbiorze danych treningowych IRMAS

Dla plików ze zbioru testowego możemy mieć podane więcej niż jeden instrument jako instrument występujący w utworze ze zbioru danych. Każdy z plików ma takie same instrumenty w danym fragmencie. Poniżej przedstawiono liczbę występujących klas dla 2874 plików audio.

**Tabela 7.5.** Rozkład instrumentów w zbiorze danych testującym IRMAS

ID	Instrument	Ilość próbek
1	Organy (Organ)	361
2	Głos ludzki (Human singing voice)	1044
3	Flet (Flute)	163
4	Gitara akustyczna (Acoustic guitar)	535
5	Pianino (Piano)	995
6	Wiolonczela (Cello)	111
7	Trąbka (Trumpet)	167
8	Saksofon (Saxophone)	326
9	Skrzypce (Violin)	211
10	Klarnet (Clarinet)	62
11	Gitara elektryczna (Electric guitar)	942
	<b>Suma:</b>	4917

Dla 2874 plików testowych mamy podane 4917 instrumentów jako instrumenty występujący w utworze, czyli ponad prawie dwukrotnie większą od ilości utworów.

## 8. Eksperymenty

### 8.1. Budowa sieci

Sieć wykorzystana w treningu została zbudowana za pomocą biblioteki `music_fsl`[20]. Sieć prototypowa składa się z 5 bloków konwolucyjnych. Każdy blok składa się z warstwy konwolucyjnej, na której następnie zostaje przeprowadzona normalizacja grupowa *nn.GroupNorm*. Normalizację grupową stosujemy w przypadku małego rozmiaru batcha lub treningu modelu na zbiorze danych z małą liczbą próbek[21]. W kolejnym kroku przekazujemy dane do funkcji aktywacji ReLU, a na sam koniec do warstwy *nn.MaxPool2d*, która służy do zmniejszania rozmiaru obrazu.

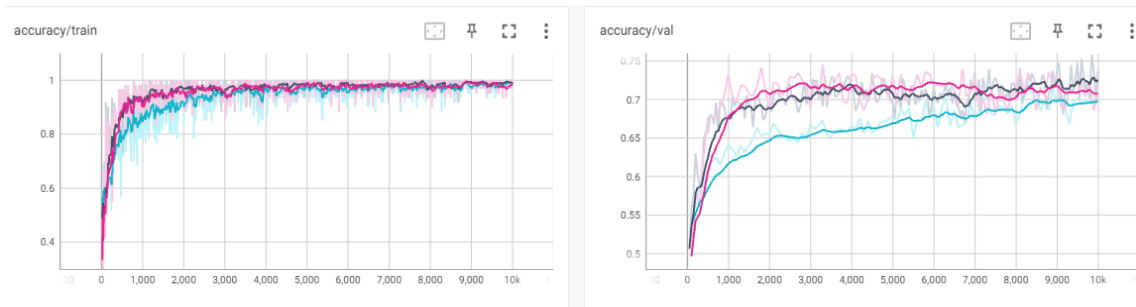
W sieci prototypowej wykorzystujemy trening epizodyczny o zadanej liczbie klas oraz przykładów w zbiorze wsparcia. Funkcja *step* odpowiedzialna za krok uczenia przyjmuje na wejściu zbiór wsparcia oraz zbiór zapytań. Dla danych z obu zbiorów wyznaczamy ich osadzenie za pomocą wyżej opisanej sieci. Następnie, dla osadzeń zbioru wsparcia obliczamy prototypy dla każdej klasy oraz wyznaczamy odległość między prototypami a danymi ze zbioru zapytań. Dla wyjścia funkcji *forward*, czyli funkcji odpowiedzialnej za przepływ sygnałów przez sieć, oraz rzeczywistych wartości, zwanych "ground truth", obliczamy funkcję starty (entropia krzyżowa). Dla obliczonego błędu aktualizujemy wagi sieci. Trening sieci wykonujemy przez wcześniej określoną liczbę epizodów.

### 8.2. Wyznaczenie parametrów

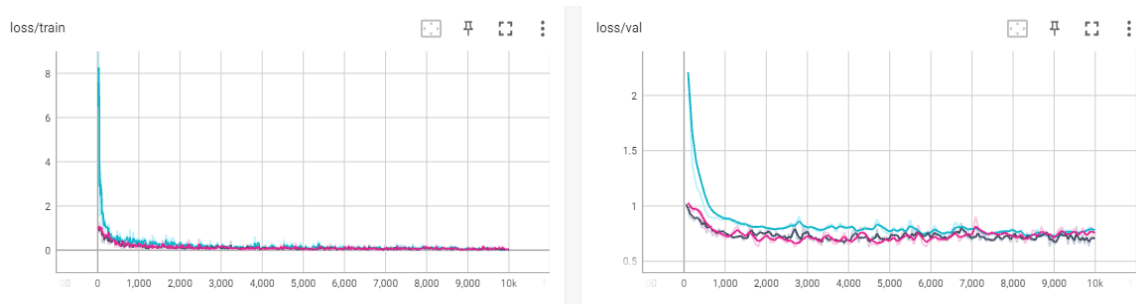
#### 8.2.1. Współczynnik uczenia oraz liczba epizodów

Współczynnik uczenia sieci wpływa na prędkość z jaką sieć się uczy, czyli oddziałuje na korekcję wag sieci. Wpływa on na siłę z jaką wagi będą przesuwane w kierunku przeciwnym do gradientu. Celem wyboru odpowiedniego współczynnika uczenia jest dobranie takiego współczynnika, który nie spowolni sieci oraz nie spowoduje jej przeuczenia.

Poniżej przedstawiono zmianę dokładności, czyli stosunek poprawnie sklasyfikowanych próbek do wszystkich próbek w zbiorze danych, oraz funkcji straty dla 10000 epizodów sieci wytrenowanej na zbiorze danych Medley-solos-DB dla 3 różnych wartości współczynnika uczenia:  $1e-2$ ,  $1e-3$  oraz  $1e-3$ .



Rysunek 8.1. Wizualizacja dokładności w czasie



Rysunek 8.2. Wizualizacja wartości funkcji straty w czasie

Na podstawie powyższych wykresów możemy zaobserwować bardzo szybki wzrost trafności oraz szybki spadek na zbiorze walidacyjnym dla pierwszych 1000 epizodów. Po około 1000 epizodów nie ma już znacznej poprawy dokładności. Uczenie sieci dla 10000 epizodów jest bardzo czasochłonne, dlatego do dalszych testów została wybrana liczba epizodów równa 1000, do której obserwowaliśmy wzrost trafności. Dla współczynnika uczenia została wybrana wartość  $1e-3$ [22].

### 8.2.2. Liczba przykładów przypadających na klasę

Poniżej przedstawiono zachowanie sieci wytrenowanej na zbiorze TinySOL na różnej liczbie próbek w zbiorze wsparcia. Zaleca się, aby liczba próbek dla testów pokrywała się z liczbą próbek, na którym został wytrenowany model[22].

**Tabela 8.1.** Dokładność wytrenowanych sieci w zależności od liczby próbek w zbiorze wsparcia

<b>Budowa sieci</b>	<b>Test dla <i>N-shot=1</i></b>	<b>Test dla <i>N-shot=3</i></b>	<b>Test dla <i>N-shot=5</i></b>	<b>Test dla <i>N-shot=7</i></b>
Dokładność predykcji dla sieci wytrenowanej na <i>K-way=5 N-shot=1</i>	0.62	0.75	0.77	0.79
Dokładność predykcji dla sieci wytrenowanej na <i>K-way=5 N-shot=3</i>	0.61	0.74	0.77	0.79
Dokładność predykcji dla sieci wytrenowanej na <i>K-way=5 N-shot=5</i>	0.62	0.74	0.77	0.79
Dokładność predykcji dla sieci wytrenowanej na <i>K-way=5 N-shot=7</i>	0.62	0.75	0.79	0.82



## 8.2.3. Liczba klas

Tabela 8.2. Dokładność wytrenowanych sieci w zależności od liczby klas

	TinySOL		GoodSounds		IRMAS	
<b>Budowa sieci</b>	<b>Test dla <math>K\text{-way}=3</math> <math>N\text{-shot}=5</math></b>	<b>Test dla <math>K\text{-way}=5</math> <math>N\text{-shot}=5</math></b>	<b>Test dla <math>K\text{-way}=3</math> <math>N\text{-shot}=5</math></b>	<b>Test dla <math>K\text{-way}=5</math> <math>N\text{-shot}=5</math></b>	<b>Test dla <math>K\text{-way}=3</math> <math>N\text{-shot}=5</math></b>	<b>Test dla <math>K\text{-way}=5</math> <math>N\text{-shot}=5</math></b>
Dokładność predykcji dla sieci wytrenowanej na $K\text{-way}=3$ $N\text{-shot}=5$	<b>0.85</b>	0.76	0.74	0.62	0.53	<b>0.39</b>
Dokładność predykcji dla sieci wytrenowanej na $K\text{-way}=5$ $N\text{-shot}=5$	<b>0.85</b>	<b>0.77</b>	<b>0.75</b>	<b>0.63</b>	<b>0.54</b>	<b>0.39</b>

Jak przedstawiono powyżej w zestawieniu dokładności predykcji dla sieci wytrenowanych dla różnej liczby klas, możemy zauważyć, że model uzyskuje lepsze predykcję na modelach wytrenowanych na większej ilości klas. Ponieważ zbiory danych posiadają małą liczbę klas (14,12,8,11), oraz musimy odpowiednio przydzielić liczbę klas do zbioru testowego i walidacyjnego, przyjmujemy liczbę klas równą 5. Dla problemu zadań o małej ilości próbek zaleca się trenować sieć na dużej liczbie klas[22].

## 8.2.4. Długość próbek

Aby poprawnie sklasyfikować instrument należy odpowiednio dobrać długość próbki audio. Należy tak dobrać próbkę, aby sieć była w stanie nauczyć się cech dźwięku. Gdy próbka będzie za krótka, sieć może nie być w stanie nauczyć się specyficznych cech dźwięku instrumentu. Dla próbki o długim czasie trwania, sieć może mieć problemy z analizą dużych rozmiarów danych. Klasyfikując próbkę audio o długim czasie trwania możemy zacząć uczyć się nie tylko charakterystyk instrumentów, ale również melodii.

Poniżej przedstawiono wyniki eksperymentów dla sieci wytrenowanej na zbiorze danych Medley-solos-DB dla 3 różnych klas z 3 próbkami w zbiorze wsparcia. Tak zbudowana sieć

została wytrenowana dla próbek o czasie trwania 1 sekundy oraz czasie trwania 2 sekund. Dokładność sieci została sprawdzona na zbiorze GoodSounds. Na podstawie wyników możemy stwierdzić, że próbka o długości 1 sekundy jest wystarczająca.

**Tabela 8.3.** Analiza długości próbek

<b>Zbiór danych</b>	<b>Test dla próbek trwających 1s</b>	<b>Test dla próbek trwających 2s</b>
Dokładność predykcji dla sieci wytrenowanej na próbkach o długości 1s	0.74	0.77
Dokładność predykcji dla sieci wytrenowanej na próbkach o długości 2s	0.73	0.77

## 9. Wyniki

### 9.1. Wyniki wytrenowanych sieci na różnych zbiorach danych

Na podstawie parametrów wyznaczonych w **rozdziale 8** zostały wytrenowane sieci na zbiorach danych TinySOL, GoodSounds, Medley-Solos-DB oraz IRMAS. Dla powstałych sieci neuronowych zostały przeprowadzone testy oraz zostało sprawdzone jak dla danej sieci zmienia się jej dokładność predykcji nowych klas w zależności od ich pochodzenia. Każdy zbiór danych reprezentuje różne rodzaje danych: pojedyncze nuty o różnej jakości, melodie dla pojedynczego instrumentu oraz utwory muzyczne, w którym występuje wiele instrumentów. Dla sprawdzenia dokładności predykcji w zależności od zbioru danych trenujących zostały wybrane poniższe sieci wytrenowane na zbiorach danych o budowie:

- TinySOL  $K$ -way=5  $N$ -shot=5
- GoodSounds  $K$ -way=5  $N$ -shot=5
- IRMAS  $K$ -way=5  $N$ -shot=5
- Medley-Solos-DB  $K$ -way=3  $N$ -shot=5 (ze względu na mniejszą liczbę klas w zbiorze danych)

Wszystkie sieci zostały wytrenowane na 1000 epizodów oraz na zbiorze zapytań równym 20 próbek.

Następnie dla wyżej wymienionych modeli został przeprowadzony eksperyment, który polegał na dotrenowaniu sieci na nowym zbiorze danych oraz porównaniu wyników dokładności predykcji na danym zbiorze w zależności od zbioru, na którym sieć została wytrenowana. W ramach eksperymentu wszystkie modele zostały dotrenowane przez 100 epizodów dla dwóch różnych liczb nowych klas  $K$ -way - 3 i 5, dla liczby próbek wsparcia  $N$ -shot równej 5 dla każdej z klas. W każdym epizodzie zbiór zapytań składał się z 15 próbek. Poniżej została przedstawiona wizualizacja macierzy błędów oraz dokładność predykcji sieci, czyli stosunek poprawnie sklasyfikowanych próbek do wszystkich próbek w zbiorze danych, dla wszystkich sieci na różnych zbiorach. Z macierzy błędów możemy zaobserwować porównanie liczby poprawnie i błędnie sklasyfikowanych próbek audio. Na osi pionowej zostały przedstawione wartości rzeczywiste, a na poziomej predykcje.

## 9.1.1. Dla zbioru danych TinySOL

Tabela 9.1. Dokładność wytrenowanych sieci dla zbioru danych TinySOL

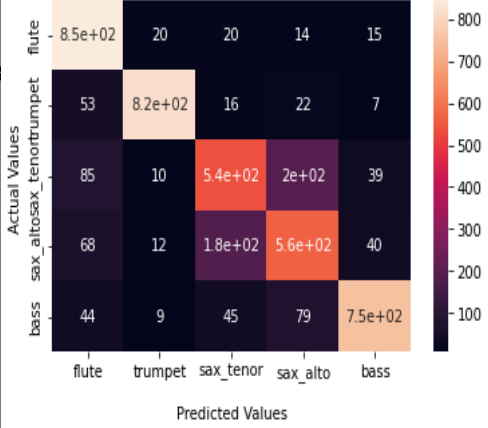
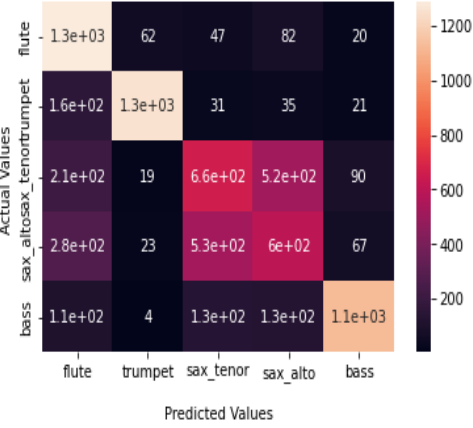
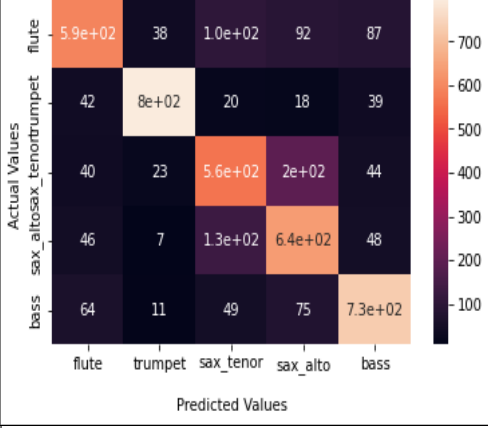
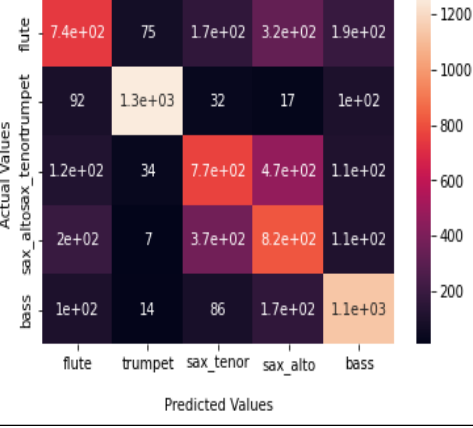
Sieć	$K\text{-way}=3$ $N\text{-shot}=5$	$K\text{-way}=5$ $N\text{-shot}=5$																																																																								
TinySOL	<table><tr><th></th><th>Bassoon</th><th>Viola</th><th>Trumpet in C</th><th>Bass Tuba</th><th>Alto Saxophone</th></tr><tr><th>Bassoon</th><td>7.8e+02</td><td>8</td><td>19</td><td>20</td><td>92</td></tr><tr><th>Viola</th><td>18</td><td>8.3e+02</td><td>10</td><td>18</td><td>42</td></tr><tr><th>Trumpet in C</th><td>8</td><td>42</td><td>7.7e+02</td><td>12</td><td>38</td></tr><tr><th>Bass Tuba</th><td>13</td><td>0</td><td>0</td><td>8.3e+02</td><td>27</td></tr><tr><th>Alto Saxophone</th><td>1.3e+02</td><td>49</td><td>43</td><td>99</td><td>6.1e+02</td></tr></table>		Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone	Bassoon	7.8e+02	8	19	20	92	Viola	18	8.3e+02	10	18	42	Trumpet in C	8	42	7.7e+02	12	38	Bass Tuba	13	0	0	8.3e+02	27	Alto Saxophone	1.3e+02	49	43	99	6.1e+02	<table><tr><th></th><th>Bassoon</th><th>Viola</th><th>Trumpet in C</th><th>Bass Tuba</th><th>Alto Saxophone</th></tr><tr><th>Bassoon</th><td>1.2e+03</td><td>14</td><td>37</td><td>55</td><td>2e+02</td></tr><tr><th>Viola</th><td>31</td><td>1.2e+03</td><td>32</td><td>54</td><td>1.4e+02</td></tr><tr><th>Trumpet in C</th><td>18</td><td>1.1e+02</td><td>1.2e+03</td><td>3</td><td>1.2e+02</td></tr><tr><th>Bass Tuba</th><td>46</td><td>6</td><td>0</td><td>1.3e+03</td><td>1.2e+02</td></tr><tr><th>Alto Saxophone</th><td>3.2e+02</td><td>66</td><td>1.8e+02</td><td>2e+02</td><td>7.4e+02</td></tr></table>		Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone	Bassoon	1.2e+03	14	37	55	2e+02	Viola	31	1.2e+03	32	54	1.4e+02	Trumpet in C	18	1.1e+02	1.2e+03	3	1.2e+02	Bass Tuba	46	6	0	1.3e+03	1.2e+02	Alto Saxophone	3.2e+02	66	1.8e+02	2e+02	7.4e+02
	Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone																																																																					
Bassoon	7.8e+02	8	19	20	92																																																																					
Viola	18	8.3e+02	10	18	42																																																																					
Trumpet in C	8	42	7.7e+02	12	38																																																																					
Bass Tuba	13	0	0	8.3e+02	27																																																																					
Alto Saxophone	1.3e+02	49	43	99	6.1e+02																																																																					
	Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone																																																																					
Bassoon	1.2e+03	14	37	55	2e+02																																																																					
Viola	31	1.2e+03	32	54	1.4e+02																																																																					
Trumpet in C	18	1.1e+02	1.2e+03	3	1.2e+02																																																																					
Bass Tuba	46	6	0	1.3e+03	1.2e+02																																																																					
Alto Saxophone	3.2e+02	66	1.8e+02	2e+02	7.4e+02																																																																					
Dokładność	0.85	0.77																																																																								
GoodSounds	<table><tr><th></th><th>Bassoon</th><th>Viola</th><th>Trumpet in C</th><th>Bass Tuba</th><th>Alto Saxophone</th></tr><tr><th>Bassoon</th><td>6.6e+02</td><td>34</td><td>18</td><td>77</td><td>1.2e+02</td></tr><tr><th>Viola</th><td>32</td><td>5.9e+02</td><td>1.4e+02</td><td>54</td><td>98</td></tr><tr><th>Trumpet in C</th><td>28</td><td>1.7e+02</td><td>5.7e+02</td><td>19</td><td>84</td></tr><tr><th>Bass Tuba</th><td>99</td><td>28</td><td>9</td><td>6.6e+02</td><td>72</td></tr><tr><th>Alto Saxophone</th><td>1.6e+02</td><td>1.3e+02</td><td>61</td><td>1.7e+02</td><td>4.1e+02</td></tr></table>		Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone	Bassoon	6.6e+02	34	18	77	1.2e+02	Viola	32	5.9e+02	1.4e+02	54	98	Trumpet in C	28	1.7e+02	5.7e+02	19	84	Bass Tuba	99	28	9	6.6e+02	72	Alto Saxophone	1.6e+02	1.3e+02	61	1.7e+02	4.1e+02	<table><tr><th></th><th>Bassoon</th><th>Viola</th><th>Trumpet in C</th><th>Bass Tuba</th><th>Alto Saxophone</th></tr><tr><th>Bassoon</th><td>7.9e+02</td><td>60</td><td>19</td><td>3.7e+02</td><td>2.6e+02</td></tr><tr><th>Viola</th><td>33</td><td>7.4e+02</td><td>3.9e+02</td><td>1.3e+02</td><td>2.1e+02</td></tr><tr><th>Trumpet in C</th><td>15</td><td>5.5e+02</td><td>7.6e+02</td><td>18</td><td>1.6e+02</td></tr><tr><th>Bass Tuba</th><td>3e+02</td><td>67</td><td>5</td><td>9.3e+02</td><td>2e+02</td></tr><tr><th>Alto Saxophone</th><td>2.4e+02</td><td>2.4e+02</td><td>1.5e+02</td><td>3.7e+02</td><td>5e+02</td></tr></table>		Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone	Bassoon	7.9e+02	60	19	3.7e+02	2.6e+02	Viola	33	7.4e+02	3.9e+02	1.3e+02	2.1e+02	Trumpet in C	15	5.5e+02	7.6e+02	18	1.6e+02	Bass Tuba	3e+02	67	5	9.3e+02	2e+02	Alto Saxophone	2.4e+02	2.4e+02	1.5e+02	3.7e+02	5e+02
	Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone																																																																					
Bassoon	6.6e+02	34	18	77	1.2e+02																																																																					
Viola	32	5.9e+02	1.4e+02	54	98																																																																					
Trumpet in C	28	1.7e+02	5.7e+02	19	84																																																																					
Bass Tuba	99	28	9	6.6e+02	72																																																																					
Alto Saxophone	1.6e+02	1.3e+02	61	1.7e+02	4.1e+02																																																																					
	Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone																																																																					
Bassoon	7.9e+02	60	19	3.7e+02	2.6e+02																																																																					
Viola	33	7.4e+02	3.9e+02	1.3e+02	2.1e+02																																																																					
Trumpet in C	15	5.5e+02	7.6e+02	18	1.6e+02																																																																					
Bass Tuba	3e+02	67	5	9.3e+02	2e+02																																																																					
Alto Saxophone	2.4e+02	2.4e+02	1.5e+02	3.7e+02	5e+02																																																																					
Dokładność	0.64	0.50																																																																								
Medley-Solos-DB	<table><tr><th></th><th>Bassoon</th><th>Viola</th><th>Trumpet in C</th><th>Bass Tuba</th><th>Alto Saxophone</th></tr><tr><th>Bassoon</th><td>7e+02</td><td>29</td><td>50</td><td>47</td><td>91</td></tr><tr><th>Viola</th><td>52</td><td>7e+02</td><td>37</td><td>28</td><td>1e+02</td></tr><tr><th>Trumpet in C</th><td>64</td><td>74</td><td>6.4e+02</td><td>29</td><td>62</td></tr><tr><th>Bass Tuba</th><td>82</td><td>33</td><td>20</td><td>6.4e+02</td><td>93</td></tr><tr><th>Alto Saxophone</th><td>1.2e+02</td><td>1.5e+02</td><td>57</td><td>1.2e+02</td><td>4.8e+02</td></tr></table>		Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone	Bassoon	7e+02	29	50	47	91	Viola	52	7e+02	37	28	1e+02	Trumpet in C	64	74	6.4e+02	29	62	Bass Tuba	82	33	20	6.4e+02	93	Alto Saxophone	1.2e+02	1.5e+02	57	1.2e+02	4.8e+02	<table><tr><th></th><th>Bassoon</th><th>Viola</th><th>Trumpet in C</th><th>Bass Tuba</th><th>Alto Saxophone</th></tr><tr><th>Bassoon</th><td>9.4e+02</td><td>53</td><td>1.1e+02</td><td>2e+02</td><td>2e+02</td></tr><tr><th>Viola</th><td>1.1e+02</td><td>1e+03</td><td>77</td><td>44</td><td>2.4e+02</td></tr><tr><th>Trumpet in C</th><td>1.6e+02</td><td>1.7e+02</td><td>9.6e+02</td><td>36</td><td>1.8e+02</td></tr><tr><th>Bass Tuba</th><td>2.6e+02</td><td>62</td><td>3</td><td>9.5e+02</td><td>2.3e+02</td></tr><tr><th>Alto Saxophone</th><td>1.8e+02</td><td>4e+02</td><td>1.2e+02</td><td>2.7e+02</td><td>5.3e+02</td></tr></table>		Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone	Bassoon	9.4e+02	53	1.1e+02	2e+02	2e+02	Viola	1.1e+02	1e+03	77	44	2.4e+02	Trumpet in C	1.6e+02	1.7e+02	9.6e+02	36	1.8e+02	Bass Tuba	2.6e+02	62	3	9.5e+02	2.3e+02	Alto Saxophone	1.8e+02	4e+02	1.2e+02	2.7e+02	5.3e+02
	Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone																																																																					
Bassoon	7e+02	29	50	47	91																																																																					
Viola	52	7e+02	37	28	1e+02																																																																					
Trumpet in C	64	74	6.4e+02	29	62																																																																					
Bass Tuba	82	33	20	6.4e+02	93																																																																					
Alto Saxophone	1.2e+02	1.5e+02	57	1.2e+02	4.8e+02																																																																					
	Bassoon	Viola	Trumpet in C	Bass Tuba	Alto Saxophone																																																																					
Bassoon	9.4e+02	53	1.1e+02	2e+02	2e+02																																																																					
Viola	1.1e+02	1e+03	77	44	2.4e+02																																																																					
Trumpet in C	1.6e+02	1.7e+02	9.6e+02	36	1.8e+02																																																																					
Bass Tuba	2.6e+02	62	3	9.5e+02	2.3e+02																																																																					
Alto Saxophone	1.8e+02	4e+02	1.2e+02	2.7e+02	5.3e+02																																																																					
Dokładność	0.70	0.59																																																																								

IRMAS		
Dokładność	0.70	0.59

### 9.1.2. Dla zbioru danych GoodSounds

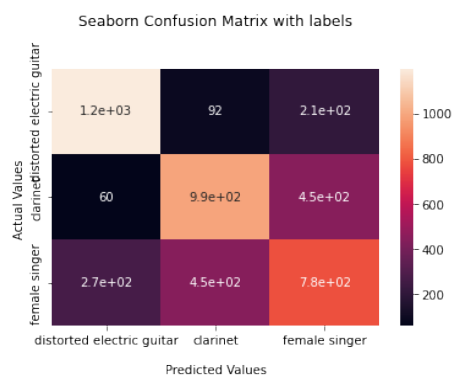
Tabela 9.2. Dokładność wytrenowanych sieci dla zbioru danych GoodSounds

Sieć	$K\text{-way}=3$ $N\text{-shot}=5$	$K\text{-way}=5$ $N\text{-shot}=5$
TinySOL		
Dokładność	0.71	0.58
GoodSounds		
Dokładność	0.75	0.63

Medley-Solos-DB		
Dokładność	0.78	0.66
IRMAS		
Dokładność	0.74	0.63

### 9.1.3. Dla zbioru danych Medley-solos-DB

Dla zbioru Medley-solos-DB nie zostało wykonane porównanie dla sieci wytrenowanej na Medley-solos-DB, ponieważ zbiór treningowy składał się jedynie z 8 klas, z czego 5 klas zostało przeznaczone na trening a 3 na walidację. Obiektywne porównanie sieci na 5 klasach było więc niemożliwe, ponieważ model był na nich trenowany. Poniżej znajduje się wizualizacja predykcji dla 3 klas.



Rysunek 9.1. Wizualizacja dokładności w czasie dla Medley-solos-DB

Tabela 9.3. Dokładność wytrenowanych sieci dla zbioru danych Medley-solos-DB

Sieć	$K\text{-way}=3$ $N\text{-shot}=5$	$K\text{-way}=5$ $N\text{-shot}=5$																																																																								
TinySOL	<table><tr><th></th><th>clarinet</th><th>electric guitar</th><th>female singer</th><th>trumpet</th><th>violin</th></tr><tr><th>clarinet</th><td>5.9e+02</td><td>32</td><td>1.3e+02</td><td>66</td><td>1e+02</td></tr><tr><th>electric guitar</th><td>27</td><td>6.6e+02</td><td>83</td><td>60</td><td>89</td></tr><tr><th>female singer</th><td>82</td><td>71</td><td>5.4e+02</td><td>99</td><td>75</td></tr><tr><th>trumpet</th><td>30</td><td>16</td><td>84</td><td>6.8e+02</td><td>57</td></tr><tr><th>violin</th><td>68</td><td>97</td><td>98</td><td>62</td><td>6.0e+02</td></tr></table>		clarinet	electric guitar	female singer	trumpet	violin	clarinet	5.9e+02	32	1.3e+02	66	1e+02	electric guitar	27	6.6e+02	83	60	89	female singer	82	71	5.4e+02	99	75	trumpet	30	16	84	6.8e+02	57	violin	68	97	98	62	6.0e+02	<table><tr><th></th><th>clarinet</th><th>electric guitar</th><th>female singer</th><th>trumpet</th><th>violin</th></tr><tr><th>clarinet</th><td>8.2e+02</td><td>61</td><td>2.9e+02</td><td>1.5e+02</td><td>1.8e+02</td></tr><tr><th>electric guitar</th><td>35</td><td>9.2e+02</td><td>1.7e+02</td><td>1.4e+02</td><td>2.3e+02</td></tr><tr><th>female singer</th><td>1.8e+02</td><td>1.8e+02</td><td>6.6e+02</td><td>1.8e+02</td><td>3e+02</td></tr><tr><th>trumpet</th><td>86</td><td>67</td><td>1.5e+02</td><td>1.1e+03</td><td>91</td></tr><tr><th>violin</th><td>1.3e+02</td><td>2.8e+02</td><td>2.4e+02</td><td>86</td><td>7.6e+02</td></tr></table>		clarinet	electric guitar	female singer	trumpet	violin	clarinet	8.2e+02	61	2.9e+02	1.5e+02	1.8e+02	electric guitar	35	9.2e+02	1.7e+02	1.4e+02	2.3e+02	female singer	1.8e+02	1.8e+02	6.6e+02	1.8e+02	3e+02	trumpet	86	67	1.5e+02	1.1e+03	91	violin	1.3e+02	2.8e+02	2.4e+02	86	7.6e+02
	clarinet	electric guitar	female singer	trumpet	violin																																																																					
clarinet	5.9e+02	32	1.3e+02	66	1e+02																																																																					
electric guitar	27	6.6e+02	83	60	89																																																																					
female singer	82	71	5.4e+02	99	75																																																																					
trumpet	30	16	84	6.8e+02	57																																																																					
violin	68	97	98	62	6.0e+02																																																																					
	clarinet	electric guitar	female singer	trumpet	violin																																																																					
clarinet	8.2e+02	61	2.9e+02	1.5e+02	1.8e+02																																																																					
electric guitar	35	9.2e+02	1.7e+02	1.4e+02	2.3e+02																																																																					
female singer	1.8e+02	1.8e+02	6.6e+02	1.8e+02	3e+02																																																																					
trumpet	86	67	1.5e+02	1.1e+03	91																																																																					
violin	1.3e+02	2.8e+02	2.4e+02	86	7.6e+02																																																																					
Dokładność	0.54	0.38																																																																								
GoodSounds	<table><tr><th></th><th>clarinet</th><th>electric guitar</th><th>female singer</th><th>trumpet</th><th>violin</th></tr><tr><th>clarinet</th><td>9.4e+02</td><td>21</td><td>3e+02</td><td>92</td><td>1.5e+02</td></tr><tr><th>electric guitar</th><td>35</td><td>1.2e+03</td><td>1.2e+02</td><td>65</td><td>88</td></tr><tr><th>female singer</th><td>3e+02</td><td>1e+02</td><td>6.2e+02</td><td>2.6e+02</td><td>2.3e+02</td></tr><tr><th>trumpet</th><td>91</td><td>1.2e+02</td><td>1.9e+02</td><td>9e+02</td><td>1.9e+02</td></tr><tr><th>violin</th><td>1.2e+02</td><td>1.6e+02</td><td>2e+02</td><td>1.7e+02</td><td>8.5e+02</td></tr></table>		clarinet	electric guitar	female singer	trumpet	violin	clarinet	9.4e+02	21	3e+02	92	1.5e+02	electric guitar	35	1.2e+03	1.2e+02	65	88	female singer	3e+02	1e+02	6.2e+02	2.6e+02	2.3e+02	trumpet	91	1.2e+02	1.9e+02	9e+02	1.9e+02	violin	1.2e+02	1.6e+02	2e+02	1.7e+02	8.5e+02	<table><tr><th></th><th>clarinet</th><th>electric guitar</th><th>female singer</th><th>trumpet</th><th>violin</th></tr><tr><th>clarinet</th><td>9.8e+02</td><td>24</td><td>2.9e+02</td><td>93</td><td>1.2e+02</td></tr><tr><th>electric guitar</th><td>47</td><td>1.2e+03</td><td>1.1e+02</td><td>64</td><td>94</td></tr><tr><th>female singer</th><td>3.1e+02</td><td>1e+02</td><td>6.3e+02</td><td>2.6e+02</td><td>2e+02</td></tr><tr><th>trumpet</th><td>1e+02</td><td>1.1e+02</td><td>2e+02</td><td>9.1e+02</td><td>1.8e+02</td></tr><tr><th>violin</th><td>1.2e+02</td><td>1.7e+02</td><td>2.1e+02</td><td>1.8e+02</td><td>8.2e+02</td></tr></table>		clarinet	electric guitar	female singer	trumpet	violin	clarinet	9.8e+02	24	2.9e+02	93	1.2e+02	electric guitar	47	1.2e+03	1.1e+02	64	94	female singer	3.1e+02	1e+02	6.3e+02	2.6e+02	2e+02	trumpet	1e+02	1.1e+02	2e+02	9.1e+02	1.8e+02	violin	1.2e+02	1.7e+02	2.1e+02	1.8e+02	8.2e+02
	clarinet	electric guitar	female singer	trumpet	violin																																																																					
clarinet	9.4e+02	21	3e+02	92	1.5e+02																																																																					
electric guitar	35	1.2e+03	1.2e+02	65	88																																																																					
female singer	3e+02	1e+02	6.2e+02	2.6e+02	2.3e+02																																																																					
trumpet	91	1.2e+02	1.9e+02	9e+02	1.9e+02																																																																					
violin	1.2e+02	1.6e+02	2e+02	1.7e+02	8.5e+02																																																																					
	clarinet	electric guitar	female singer	trumpet	violin																																																																					
clarinet	9.8e+02	24	2.9e+02	93	1.2e+02																																																																					
electric guitar	47	1.2e+03	1.1e+02	64	94																																																																					
female singer	3.1e+02	1e+02	6.3e+02	2.6e+02	2e+02																																																																					
trumpet	1e+02	1.1e+02	2e+02	9.1e+02	1.8e+02																																																																					
violin	1.2e+02	1.7e+02	2.1e+02	1.8e+02	8.2e+02																																																																					
Dokładność	0.72	0.60																																																																								

## 9. Wyniki

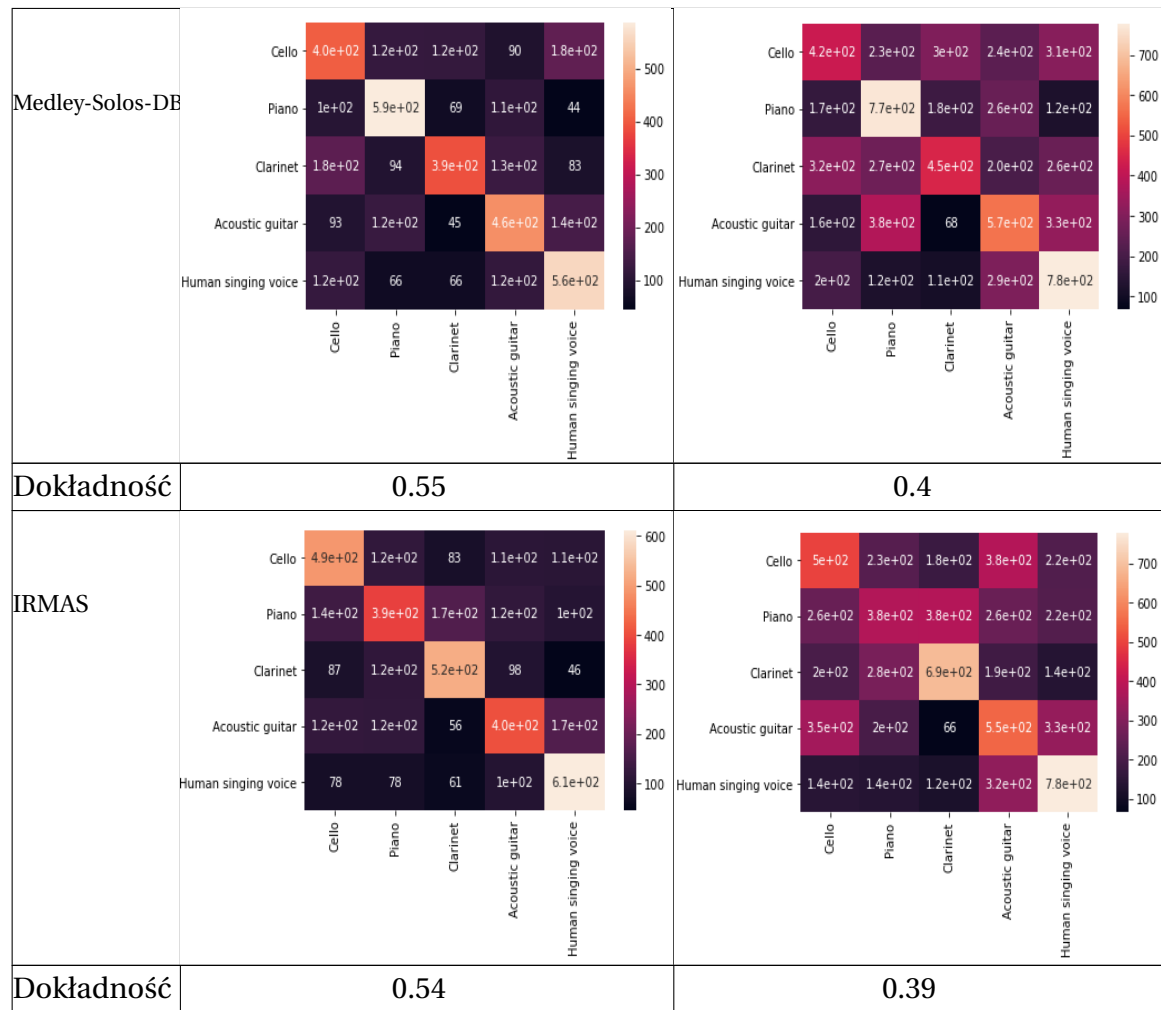
IRMAS		
	Dokładność 0.68	Dokładność 0.57

### 9.1.4. Dla zbioru danych IRMAS

Tabela 9.4. Dokładność wytrenowanych sieci dla zbioru danych IRMAS

Sieć	$K\text{-way}=3 \ N\text{-shot}=5$	$K\text{-way}=5 \ N\text{-shot}=5$
TinySOL		
	Dokładność 0.51	Dokładność 0.36
GoodSounds		
	Dokładność 0.54	Dokładność 0.38

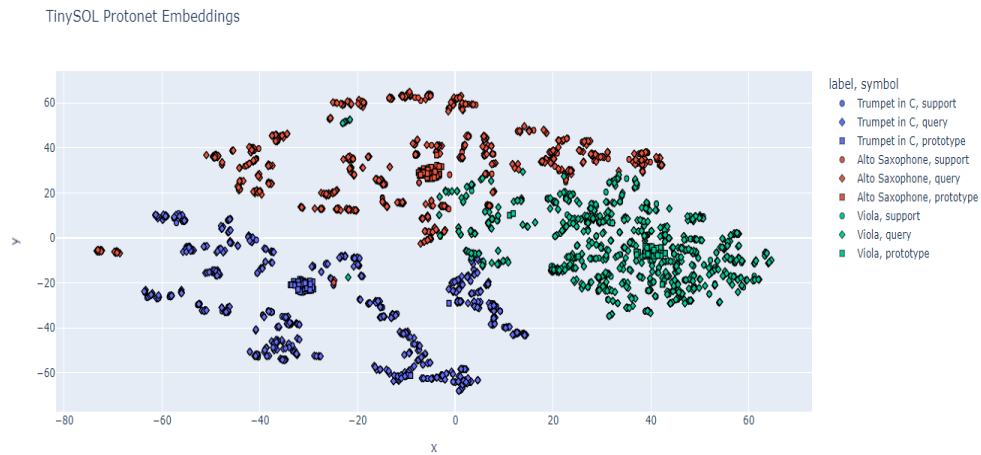




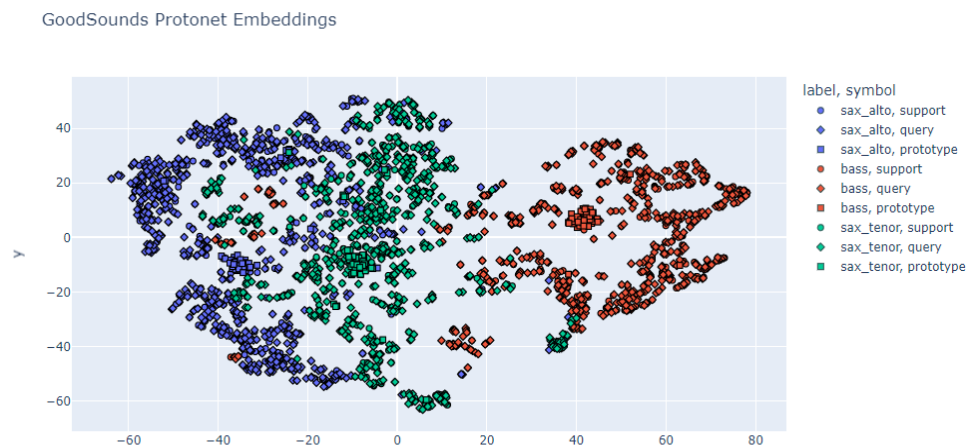
## 9.2. Analiza przestrzeni osadzenia

Dla każdego zbioru danych została wytrenowana prototypowa sieć neuronowa. W artykule[22], w którym została przedstawiona prototypowa sieć neuronowa, sieć składa się z 4 warstw konwolucyjnych. W opisywanej pracy sieć neuronowa została rozszerzona na 5 warstw, aby uzyskać lepszą dokładność rozwiązania. Bloki konwolucyjne składają się z warstw konwolucyjnych z filtrem  $3 \times 3$ , normalizacją grupową wraz z maxpoolingiem  $2 \times 2$  oraz  $4 \times 4$ . Jako funkcja aktywacji wykorzystana została funkcja ReLU. Strata obliczana jest za pomocą entropii krzyżowej, a aktualizacji parametrów jest dokonywana za pomocą optymalizatora Adam. Trening został przeprowadzony dla parametrów  $K - way = 3$  oraz  $N - shot = 5$  przez 1000 epizodów. W celu sprawdzenia jakości wytrenowanych sieci zostały one douczone dla 3 nowych klas oraz  $N=5$  przez kolejne 25 epizodów.

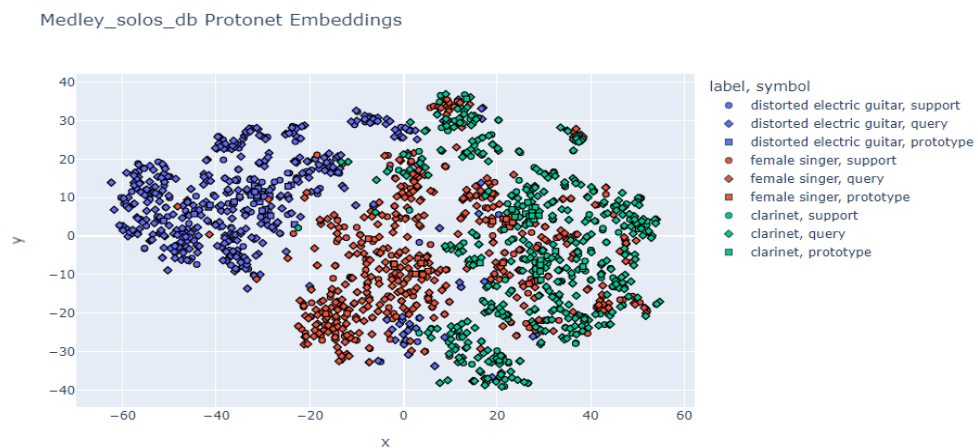
Dla sieci prototypowej, której wyjściem jest wektor osadzenia, została przeprowadzona redukcja wymiarów za pomocą metody UMAP. Poniżej przedstawiona została wizualizacja przestrzeni osadzenia dla sieci wytrenowanych na 4 różnych zbiorach danych.



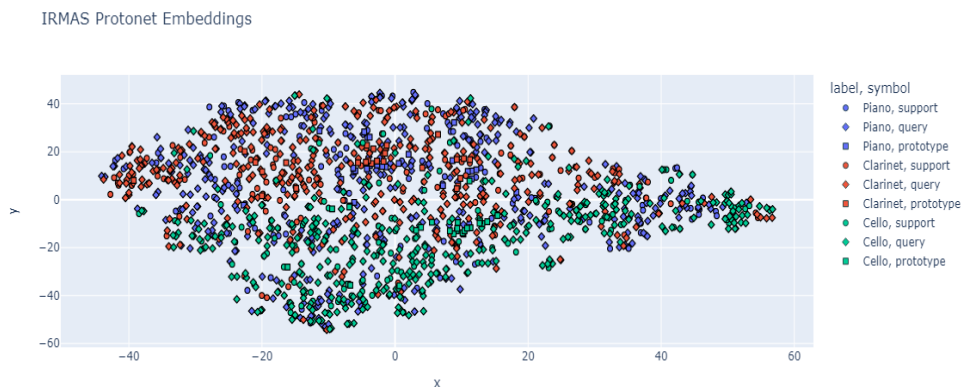
**Rysunek 9.2.** Wizualizacja przestrzeni osadzenia zbioru TinySol



**Rysunek 9.3.** Wizualizacja przestrzeni osadzenia zbioru GoodSounds



**Rysunek 9.4.** Wizualizacja przestrzeni osadzenia zbioru Medley-solos-DB



**Rysunek 9.5.** Wizualizacja przestrzeni osadzenia zbioru IRMAS

Dla instrumentów, które przedstawione są w zbiorze danych za pomocą pojedynczej nuty, widzimy wyraźne rozróżnienie klas. Wraz ze wzrostem zmiany dźwięku, czyli pojawienie się zanikania, ataku, występowania różnych charakterystyk tonalnych, obserwujemy zanikanie wyraźnych granic, co powoduje spadkiem jakości sieci. Dla utworów muzycznych obserwujemy coraz większe trudności z rozpoznawaniem danego instrumentu, punkty osadzone w przestrzeni zaczynają na siebie nachodzić, co odwzorowuje oszacowana jakość sieci. Dla zbioru danych IRMAS obserwujemy nakładające się na siebie punkty w przestrzeni osadzenia. Dzieje się to przez typ danych, które mają postać fragmentów plików muzycznych, w których występuje kilka instrumentów. Dla przykładu, w zbiorze danych oznakowanych jako wiolonczela występują utwory w którym instrumentem dominującym jest wiolonczela w towarzystwie klarnetu. Taki sam przykład mamy dla klasy pianina. Przez taki typ danych obserwujemy więc nakładanie się punktów w przestrzeni osadzenia.

### 9.3. Test na nowym zbiorze danych

Aby jednoznacznie stwierdzić jak powyższe sieci radzą sobie z problemem klasyfikacji instrumentów została przeprowadzona klasyfikacja na nowym zbiorze danych z platformy Kaggle[23] składających się z fragmentów utworów pojedynczych instrumentów. Posiada on 10 różnych klas instrumentów: bas, gitara elektryczna, dzwonki, organy, fortepian, pipa, werbel, smyczki, vintage lead oraz skrzypce. Zbiór ten został wybrany z powodu występowania nowych instrumentów, które pozwolą na określenie dokładności sieci. Dla powyżej opisanych sieci, czyli sieci wytrenowanych na podanych zbiorach danych o następującej budowie:

- TinySOL  $K - way = 5 \ N - shot = 5$
- GoodSounds  $K - way = 5 \ N - shot = 5$
- IRMAS  $K - way = 5 \ N - shot = 5$
- Medley-Solos-DB  $K - way = 3 \ N - shot = 5$

zostało przeprowadzone dotrenowanie sieci na 50 epizodach dla 3 różnych klas:

- Dzwonki
- Pipa
- Skrzypce

Każda klasa posiadała 5 próbek na epizod w zbiorze wsparcia. Poniżej została przedstawiona dokładność każdej sieci na nowym zbiorze danych.

**Tabela 9.5.** Analiza dokładność na nowym zbiorze danych

Sieć dla zbioru danych	Dokładność
TinySOL	0.79
GoodSounds	0.81
Medley-Solos-DB	0.83
IRMAS	0.84

#### 9.4. Wnioski

Na podstawie przeprowadzonych eksperymentów możemy stwierdzić, że sieć należy trenować na większej liczbie klas, wtedy osiąga większą dokładność. Dla problemu tagowania muzyki istotnym parametrem jest również dobranie odpowiedniej długości próbki dźwięku. Długość musi być tak dobrana, aby była w stanie nauczyć się cech dźwięku instrumentu. Sieć prototypową najlepiej trenować dla instrumentów z różnych rodzin, aby sieć była w stanie nauczyć się różnych cech dla danej rodziny, co umożliwi na rozpoznawanie nowych klas. Nie należy ograniczać się jednak do pojedynczych instrumentów dla danej rodziny. Gdy uczymy sieć dla danych z tej samej rodziny instrumentów, uczymy sieć jej możliwych odmian i cech. Należy więc dobrać zrównoważony zbiór danych, o różnych przykładach dla różnych rodzin instrumentu. Na podstawie powyższych wyników możemy zauważyć pewną zależność dla zbiorów danych. Sieci wytrenowane na zbiorach danych o większej złożoności lepiej radzą sobie z problemem klasyfikacji instrumentów w utworach muzycznych oraz dorównują dokładnością klasyfikacji dla pojedynczych nut. W zależności od celu klasyfikacji wybieramy odpowiedni typ zbioru danych dla treningu sieci. Dla problemu klasyfikacji nut, sieci wytrenowane na pojedynczych nutach będą lepiej radziły sobie z klasyfikacją instrumentów. Dla klasyfikacji utworów muzycznych lepsze wyniki osiąga sieć stworzona na zbiorach tego typu danych. Jeśli szukamy rozwiązania dla ogólnej klasyfikacji, czyli klasyfikacji na podstawie losowej próbki dźwięku, lepszym wyborem będzie sieć wytrenowana na fragmentach utworów muzycznych.

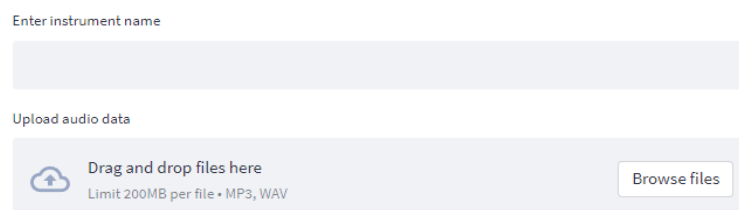
## 10. Aplikacja

Do wykorzystania wytrenowanych sieci przygotowana została aplikacja internetowa. Aplikacja powstała z wykorzystaniem biblioteki Pythona *Streamlit*[24]. Streamlit jest biblioteką umożliwiającą budowę aplikacji webowych dla zadań uczenia maszynowego.

Stworzona aplikacja ma na celu wytrenowanie nowego modelu dla nowych klas oraz otagowanie dostarczonych plików. Użytkownik ma możliwość wyboru modelu wytrenowanego na jednym z powyższych zbiorów danych oraz dostarczenia różnych klas instrumentów wraz z kilkoma próbkami danych dla każdej z nich. Następnie użytkownik podaje zestaw utworów do otagowania. Na podstawie podanych danych wybrany model zostaje douczony przez określoną ilość epizodów dla nowych klas, a następnie program wyznacza klasy dla podanych utworów do otagowania. Wyniki programu zwrócone są w postaci pliku csv. Aplikacji została przedstawiona na rysunku 10.2. Program aplikacji znajduje się na **repozytorium github**.

### 10.1. Instrukcja dla użytkownika

Aplikacja polega na przypisaniu tagów do dostarczonego zbioru danych na podstawie kilku ich próbek. Aby otagować zbiór plików audio, użytkownik powinien podać dla ilu nowych klas mamy dotrenować wybrany model. Od użytkownika zostaje również pobrana liczba próbek na podstawie której model ma za zadanie rozpoznać daną klasę. Aplikacja pobiera również takie informacje jak: liczba próbek dla zbioru wsparcia, liczba epizodów przez które sieć ma się nauczyć oraz wybrany model, na którym zostanie wykonana predykcja. Dostarczenie danych dla wybranej klasy polega na wpisaniu nazwy tej klasy oraz przesłania jej kilku próbek.



The image shows a web application interface. At the top, there is a text input field labeled 'Enter instrument name'. Below this, there is a section titled 'Upload audio data'. Inside this section, there is a light blue box containing a cloud icon with an upward arrow, the text 'Drag and drop files here', and 'Limit 200MB per file • MP3, WAV'. To the right of this box is a button labeled 'Browse files'.

**Rysunek 10.1.** Wprowadzenie plików audio do aplikacji

Po wgraniu wszystkich próbek dla wszystkich klas należy podać zbiór danych dla których zostanie przeprowadzona predykcja. Po wprowadzeniu wszystkich informacji należy nacisnąć przycisk *Learn model and predict*. Model rozpoczął naukę, postęp nauki zostanie przedstawiony na pasku postępu. Po zakończeniu doszkalania modelu zostanie przeprowadzona predykcja dla wprowadzonych plików. Wynik predykcji zostanie wypisany na ekranie oraz program umożliwi pobranie wyników w postaci pliku csv, poprzez naciśnięcie przycisku *Download prediction*.

## Prototypical Network prediction

This application was created to allow the user to train his own prototype network and on its basis to tag instruments in sound files.



In the application we have a choice of pre-trained networks on datasets such as:

- TinySOL
- GoodSounds
- MedleySolosDb
- IRMAS



Number of classes to predict from

2  10



Number of samples in support set of each class (recommended 5, should be smaller than the smallest number of samples per class)

0  


Number of samples in query set of each class (should be not greater than the sum of the smallest number of samples per class minus the data taken for the support set)

0  

Number of epochs to train the model



1  

Select model trained on:

TinySOL 


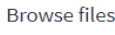
Enter instrument name

Upload audio data


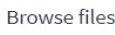
 Drag and drop files here   
Limit 200MB per file • MP3, WAV

Enter instrument name

Upload audio data

 Drag and drop files here   
Limit 200MB per file • MP3, WAV

Upload data to predict

 Drag and drop files here   
Limit 200MB per file • MP3, WAV

Learn model and predict

## 11. Podsumowanie

Zgodnie z założeniem udało się osiągnąć wszystkie cele, które postawiłam sobie w ramach projektu. Dogłębne poznanie zagadnienia metody uczenia na małych próbkach oraz analiza zbiorów danych dedykowanych do klasyfikacji muzyki, w tym instrumentów, pomogło w zrozumieniu oraz wytrenowania najlepszego modelu sieci prototypowej dla problemu klasyfikacji instrumentów dla danych zbiorów. Jeśli poszukujemy najlepszego modelu do tagowania konkretnych plików muzycznych, należy wybrać model trenowany na podobnym zbiorze danych. Jednak jeśli szukamy modelu nadającego się do wszelkiego typu plików audio, najlepszym modelem okazał się model wytrenowany na utworach muzycznych pojedynczego instrumentu. Osiąga on bardzo dobre wyniki w porównaniu z innymi sieciami dla problemu rozpoznawania pojedynczej nuty jak i utworu muzycznego. Do treningu sieci dla problemu rozpoznawania instrumentów należy zapewnić jak najszerszy zbiór danych, z różnych rodzin instrumentów oraz o jak największej liczbie ich przykładów.

Jako efekt pracy powstała aplikacja do tagowania muzyki. Zgodnie z założeniem aplikacja pobiera od użytkownika nowe klasy oraz kilka przykładów audio. Na podstawie zaledwie kilku przykładów od użytkownika, sieć jest w stanie przewidzieć nową klasę z dość wysoką dokładnością, około 80%. Aplikacja pozwala na etykietowanie nieograniczonej liczby nowych klas.

## 12. Perspektywy rozwoju projektu

Mimo że pierwotne założenia projektu zostały spełnione, przedstawioną aplikację można rozwijać na wiele sposobów.

- Rozszerzenie aplikacji o metodę One-shot learning
- Rozszerzenie aplikacji o etykietowanie instrumentów na podstawie wyłącznie metadanych. Jest to metoda Zero-shot learning. ZSL może być również przeprowadzony na sieci prototypowej. Wejście sieci różni się jedynie o typ danych wejściowych, zamiast plików audio wprowadzane są pewne metadane np. rodzina instrumentów,
- Implementacja sieci prototypowej dla problemu klasyfikacji gatunku muzycznego.
- Rozszerzenie sieci prototypowej o system hierarchii[25]. Sieć po wyznaczeniu hierarchii instrumentów umożliwia naukę za pomocą kilku próbek nowej klasy, która ma wspólnego przodka w wyznaczonej hierarchii. Pozwoliłoby to na rozpoznanie instrumentów na różnych poziomach.
- Implementacja innych modeli Meta-uczenia opartych na metryce.





## Bibliografia

- [1] P. Knees, M. Schedl i M. Goto, “Intelligent User Interfaces for Music Discovery”, *Transactions of the International Society for Music Information Retrieval*, t. 3, nr. 1, s. 165–179, 2020. DOI: 10.5334/tismir.60. adr.: <http://doi.org/10.5334/tismir.60>.
- [2] D. Kostrzewa, B. Koza i P. Benecki, “Designing a Training Set for Musical Instruments Identification”, w *Computational Science – ICCS 2022*, D. Groen, C. de Mulatier, M. Paszynski, V. V. Krzhizhanovskaya, J. J. Dongarra i P. M. A. Sloot, red., Cham: Springer International Publishing, 2022, s. 599–610.
- [3] M. Schedl, “Deep Learning in Music Recommendation Systems”, *Frontiers in Applied Mathematics and Statistics*, t. 5, 2019, ISSN: 2297-4687. DOI: 10.3389/fams.2019.00044. adr.: <https://www.frontiersin.org/articles/10.3389/fams.2019.00044>.
- [4] S. Oramas, F. Barbieri, O. Nieto i X. Serra, “Multimodal Deep Learning for Music Genre Classification”, *Transactions of the International Society for Music Information Retrieval*, t. 1, nr. 1, s. 4–21, 2018. DOI: 10.5334/tismir.10. adr.: <https://doi.org/10.5334/tismir.10>.
- [5] M. Dörfler, R. Bammer i T. Grill, “Inside the spectrogram: Convolutional Neural Networks in audio processing”, w *2017 International Conference on Sampling Theory and Applications (SampTA)*, 2017, s. 152–155. DOI: 10.1109/SAMPTA.2017.8024472.
- [6] M. Won, A. Ferraro, D. Bogdanov i X. Serra, *Evaluation of CNN-based Automatic Music Tagging Models*, 2020. DOI: 10.48550/ARXIV.2006.00751. adr.: <https://arxiv.org/abs/2006.00751>.
- [7] K. O’Shea i R. Nash, *An Introduction to Convolutional Neural Networks*, 2015. DOI: 10.48550/ARXIV.1511.08458. adr.: <https://arxiv.org/abs/1511.08458>.
- [8] K. Choi, G. Fazekas, M. Sandler i K. Cho, “Convolutional Recurrent Neural Networks for Music Classification”, eng, 2016.
- [9] P. Cunningham, M. Cord i S. J. Delany, “Supervised Learning”, w *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*, M. Cord i P. Cunningham, red. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, s. 21–49, ISBN: 978-3-540-75171-7. DOI: 10.1007/978-3-540-75171-7\_2. adr.: [https://doi.org/10.1007/978-3-540-75171-7\\_2](https://doi.org/10.1007/978-3-540-75171-7_2).
- [10] A. Parnami i M. Lee, *Learning from Few Examples: A Summary of Approaches to Few-Shot Learning*, 2022. DOI: 10.48550/ARXIV.2203.04291. adr.: <https://arxiv.org/abs/2203.04291>.
- [11] J. Choi, J. Lee, J. Park i J. Nam, *Zero-shot Learning for Audio-based Music Classification and Tagging*, 2019. adr.: <https://arxiv.org/abs/1907.02670>.
- [12] J. Choi, J. Lee, J. Park i J. Nam, *Zero-shot Learning and Knowledge Transfer in Music Classification and Tagging*, 2019. adr.: <https://arxiv.org/abs/1906.08615>.

- [13] J. Snell, K. Swersky i R. S. Zemel, *Prototypical Networks for Few-shot Learning*, 2017. DOI: 10.48550/ARXIV.1703.05175. adr.: <https://arxiv.org/abs/1703.05175>.
- [14] C.-E. Cella, D. Ghisi, V. Lostanlen, F. Lévy, J. Fineberg i Y. Maresz, *TinySOL: an audio dataset of isolated musical notes*, Dostęp: 24.01.2023, 2020. adr.: <https://zenodo.org/record/3685367>.
- [15] O. Romani, H. Parra, D. Dabiri i in., “A real-time system for measuring sound goodness in instrumental sounds.”, Warsaw, Poland: 138th Audio Engineering Society Convention, 2015.
- [16] V. Lostanlen, C.-E. Cella, R. Bittner i S. Essid, *Medley-solos-DB: a cross-collection dataset for musical instrument recognition*, Dostęp: 24.01.2023, 2019. adr.: <https://zenodo.org/record/3464194>.
- [17] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam i J. P. Bello., “MedleyDB: A multitrack dataset for annotationintensive mir research.”, In ISMIR ’14 (Taipei, Taiwan), 2014.
- [18] P. Zinemanas, M. Rocamora, M. Miron, F. Font i X. Serra, “An Interpretable Deep Learning Model for Automatic Sound Classification”, Electronics, 2021.
- [19] J. Bosch, J. Janer, F. Fuhrmann i P. Herrera, “A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals”, *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, s. 559–564, sty. 2012.
- [20] H. F. García, *Few-Shot Learning for Music Information Retrieval*, Dostęp zdalny (20.01.2023): [https://github.com/music-fsl-zsl/music\\_fsl](https://github.com/music-fsl-zsl/music_fsl), 2022.
- [21] Y. Wu i K. He, *Group Normalization*, 2018. DOI: 10.48550/ARXIV.1803.08494. adr.: <https://arxiv.org/abs/1803.08494>.
- [22] J. Snell, K. Swersky i R. Zemel, “Prototypical Networks for Few-shot Learning”, w *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio i in., red., t. 30, Curran Associates, Inc., 2017. adr.: <https://proceedings.neurips.cc/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf>.
- [23] *Wavfiles of Instruments’ audio*, Dostęp zdalny (15.01.2023): <https://www.kaggle.com/datasets/mayur1999/wavfiles-of-instruments-audio>.
- [24] *Streamlit*, Dostęp zdalny (23.01.2023): <https://streamlit.io/>, 2019.
- [25] H. F. Garcia, A. Aguilar, E. Manilow i B. Pardo, *Leveraging Hierarchical Structures for Few-Shot Musical Instrument Recognition*, 2021. DOI: 10.48550/ARXIV.2107.07029. adr.: <https://arxiv.org/abs/2107.07029>.

## Wykaz symboli i skrótów

**EiTI** – Wydział Elektroniki i Technik Informatycznych

**PW** – Politechnika Warszawska

**FSL** – ang. *Few-shot learning*

**ZSL** – ang. *Zero-shot Learning*

## Spis rysunków

2.1	Spektrogram dla próbki dźwięku trąbki - librosa . . . . .	10
2.2	Mel spektrogram dla próbki dźwięku trąbki - librosa . . . . .	11
2.3	Spektrogram dla próbki dźwięku trąbki - torchaudio . . . . .	11
2.4	Mel spektrogram dla próbki dźwięku trąbki - torchaudio . . . . .	12
5.1	Podział modeli dla problemu uczenia z małą ilością przykładów . . . . .	17
5.2	Przestrzeń osadzenia . . . . .	18
6.1	Rozłożenie punktów zbioru wsparcia oraz ich prototypy. . . . .	21
7.1	Wizualizacja rozkładu instrumentów w zbiorze danych TinySol . . . . .	24
7.2	Wizualizacja rozkładu instrumentów w zbiorze danych GoodSounds . . . . .	25
7.3	Wizualizacja rozkładu instrumentów w zbiorze danych Medley-solos-DB . . . . .	27
7.4	Wizualizacja rozkładu instrumentów w zbiorze danych treningowych IRMAS . . . . .	28
8.1	Wizualizacja dokładności w czasie . . . . .	31
8.2	Wizualizacja wartości funkcji straty w czasie . . . . .	31
9.1	Wizualizacja dokładności w czasie dla Medley-solos-DB . . . . .	39
9.2	Wizualizacja przestrzeni osadzenia zbioru TinySol . . . . .	42
9.3	Wizualizacja przestrzeni osadzenia zbioru GoodSounds . . . . .	42
9.4	Wizualizacja przestrzeni osadzenia zbioru Medley-solos-DB . . . . .	42
9.5	Wizualizacja przestrzeni osadzenia zbioru IRMAS . . . . .	43
10.1	Wprowadzenie plików audio do aplikacji . . . . .	45
10.2	Wizualizacja aplikacji. . . . .	46

## Spis tabel

7.1	Rozkład instrumentów w zbiorze danych TinySOL . . . . .	23
7.2	Rozkład instrumentów w zbiorze danych GoodSounds . . . . .	25
7.3	Rozkład instrumentów w zbiorze danych Medley-solos-DB . . . . .	26
7.4	Rozkład instrumentów w zbiorze danych treningowych IRMAS . . . . .	28
7.5	Rozkład instrumentów w zbiorze danych testującym IRMAS . . . . .	29
8.1	Dokładność wytrenowanych sieci w zależności od liczby próbek w zbiorze wsparcia . . . . .	32

8.2	Dokładność wytrenowanych sieci w zależności od liczby klas . . . . .	33
8.3	Analiza długości próbek . . . . .	34
9.1	Dokładność wytrenowanych sieci dla zbioru danych TinySOL . . . . .	36
9.2	Dokładność wytrenowanych sieci dla zbioru danych GoodSounds . . . . .	37
9.3	Dokładność wytrenowanych sieci dla zbioru danych Medley-solos-DB . . . . .	39
9.4	Dokładność wytrenowanych sieci dla zbioru danych IRMAS . . . . .	40
9.5	Analiza dokładność na nowym zbiorze danych . . . . .	44