ZZSN - Dokumentacja końcowa

Uczenie klasyfikatorów dźwiękowych niewielką liczbą przykładów z wykorzystaniem sieci prototypowych

Sebastian Pietras, Arkadiusz Sadza

Opis zadania

W ramach projektu stworzono model sieci prototypowej do uczenia klasyfikatorów dźwiękowych niewielką liczbą przykładów (ang. few-shot learning). Wyniki tego podejścia zostały porównane ze standardową metodą uczenia klasyfikatorów.

Zbiór danych

Wykorzystany został zbiór danych FSD50K. Jest to zbiór krótkich (od 0.3 do 30 sekund) fragmentów audio w postaci plików .wav. Do każdego z nich przypisane są klasy, oznaczające czym jest dany dźwięk (np. gitara, alarm). W większości do próbek przypisana jest tylko jedno klasa. Jako że sieci prototypowe w oryginalnym zamyśle służą do klasyfikacji jednoetykietowej to próbki z przypisanymi wieloma klasami zostały odrzucone.

Każdy fragment audio został przekształcony do określonej długości, a następnie zamieniony na spektrogram. Jeśli dźwięk był dłuższy niż ustalony czas trwania to został przycięty, a jeśli krótszy to powielony do wypełnienia całej długości.

Dla celów uczenia i testów sieci neuronowej dwa komponenty zbioru FSD50K (dev i eval) zostały złączone w jeden zbiór, z którego losowano próbki do zbioru trenującego, walidacyjnego oraz testowego.

Opis rozwiązania

Do implementacji wykorzystano bibliotekę PyTorch Lightning, w szczególności pozwalającą na automatyzację procesu uczenia dzięki klasie Trainer.

Podstawą rozwiązania (SoundEmbedding) jest splotowa sieć neuronowa o architekturze zbliżonej do części Audio Subnetwork modelu Look, Listen and Learn. Sieć na wejściu otrzymała spektrogram dźwięku i na wyjściu generowała dla niego odpowiedni wektor zanurzeń o ustalonej wielkości.

Następnie zostały zaimplementowane dwa klasyfikatory: standardowy (SoundClassifier) oraz prototypowy (ProtoSoundFewShotClassifier). Klasyfikator standardowy na SoundEmbedding nakładał warstwę liniową, która mapuje przestrzeń zanurzeń na przestrzeń o wielkości takiej jak liczba wszystkich klas oraz “warstwę” LogSoftmax, która zapewnia że każde wyjście będzie logarytmicznym prawdopodobieństwem dopasowania do danej klasy (i jest numerycznie stabilniejsza niż sam Softmax). Klasyfikator prototypowy

również korzysta z SoundEmbedding do uzyskania zanurzeń i przeprowadza klasyfikację zgodnie z algorytmem opisanym w artykule o sieciach prototypowych.

Uczenie modelu zostało przeprowadzone na kilka sposobów:

- ucząc standardowy klasyfikator na całym zbiorze danych z wykorzystaniem *Negative Log Likelihood* jako funkcji straty
- jak wyżej, lecz z ograniczoną liczbą próbek per klasa
- ucząc klasyfikator prototypowy dla różnej liczby próbek per klasa
- ucząc klasyfikator prototypowy dla różnej liczby klas

Wyniki klasyfikacji na zbiorze testowym modeli wyuczonych powyższymi sposobami zostały porównane ze sobą. W celu zbadania umiejętności klasyfikacji przez sieci prototypowe nieznanych wcześniej klas, zbiory treningowe, walidacyjne oraz testowe zostały losowo podzielone według klas w przypadku uczenia sieci prototypowych.

Ponadto w ramach rozwiązania dokonano następujących decyzji projektowych:

- dokonano wyboru architektury SoundEmbedding (3 grupy warstw: Conv2d + ReLU + MaxPool2d, na końcu AdaptiveAvgPool2d do wypłaszczenia) - celem było zaprojektowanie racjonalnie dobrego klasyfikatora do porównań, niekoniecznie najlepszego
- metodą optymalizacji funkcji straty był algorytm ADAM
- spektrogramy generowano przy pomocy biblioteki librosa jako Mel Spectrogram (spektrogram z nieliniowo przeskalowanymi wymiarami, standardowo stosowany do zastosowań uczenia na danych audio)
- kod zorganizowano w 3 grupy: moduł danych (`data.py`), moduł modeli sieci (`modules.py`) oraz skrypt do uczenia (`__main__.py`)

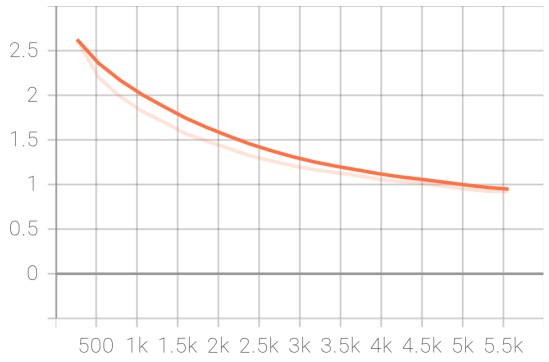
Testy

Testy przeprowadzano dwukierunkowo. Po pierwsze sprawdzano jakość klasyfikacji (ilościowo poprzez pomiar i wizualnie poprzez analizę wykresów) dla standardowego klasyfikatora, standardowego klasyfikatora z ograniczoną liczbą przykładów per klasa i dla klasyfikatora prototypowego. Wyniki tabelaryczne dla porównania jakości dwóch pierwszych sieci pokazuje tabela 1 oraz wykresy 1, 2.

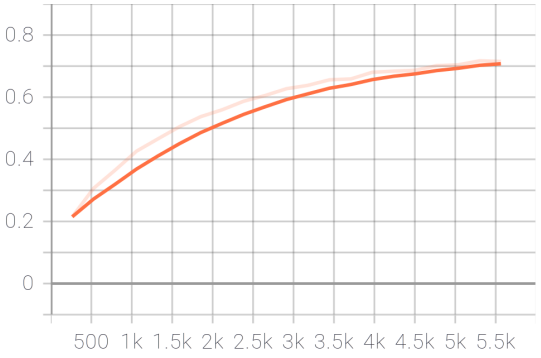
Rodzaj sieci	**Wyniki klasyfikacji na zbiorze:**					
	Treningowym		**Walidacyjnym**		**Testowym**	
	strata	**dokładność**	**strata**	**dokładność**	**strata**	**dokładność**
Standardowy klasyfikator	0.9487	0.7076	1.089	0.6677	1.055	0.6813
Standardowy klasyfikator dla pięciu przykładów per klasa	0.1705	0.9638	5.278	0.3474	3.013	0.21

Tabela 1. Jakość klasyfikacji - porównanie standardowego klasyfikatora w uczeniu na całym zbiorze i w uczeniu z ograniczoną liczbą przykładów per klasa.

Strata na zbiorze treningowym

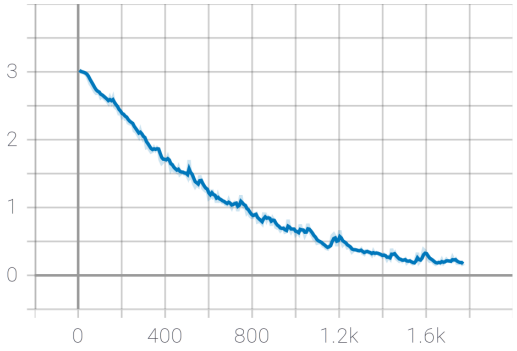


Dokładność na zbiorze treningowym

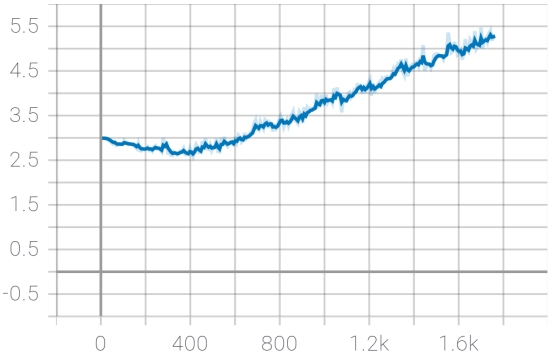


Wykres 1. Klasyfikator standardowy - strata i dokładność osiągnięta na zbiorze treningowym bez ograniczenia liczby próbek.

Strata na zbiorze treningowym



Strata na zbiorze walidacyjnym



Wykres 2. Klasyfikator standardowy z ograniczoną liczbą przykładów per klasa - strata na zbiorze treningowym cały czas spada, a na walidacyjnym rośnie.

Z wykresów 1 oraz 2 widać, że o ile klasyfikator standardowy uczy się dobrze na dużej liczbie przykładów, to na ograniczonej liczbie przykładów ulega przeuczeniu. Jest to spodziewany efekt, ponieważ jeśli danych jest zbyt mało to model się do nich nadmiernie dopasowuje.

Następnie porównywano jakość uczenia się klasyfikatora prototypowego dla różnej liczby przykładów per klasa i liczby klas. We wszystkich przypadkach proces uczenia się osiąga dobre wyniki i stabilizuje się, co potwierdzają poniższe tabele 2, 3 i wykresy 3, 4.

Badany parametr	k	Wyniki klasyfikacji na zbiorze:					
		Treningowym		Walidacyjnym		Testowym	
		strata	dokładność	strata	dokładność	strata	dokładność
Liczba przykładów per klasa (k - tzw. support)	3	0.6977	0.7077	0.7395	0.662	0.6581	0.725
	5	0.4661	0.8158	0.64	0.7305	0.5263	0.79
	10	0.6128	0.75	0.5076	0.8078	0.7618	0.7

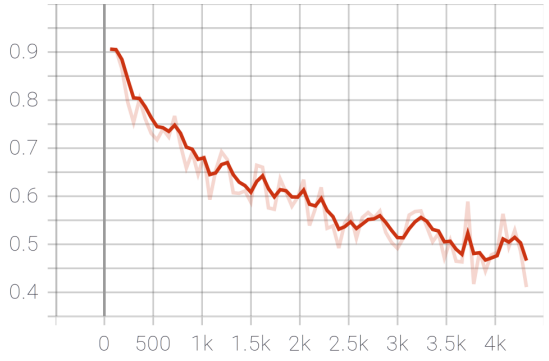
Tabela 2. Porównanie jakości sieci prototypowych dla różnej liczby przykładów per klasa.

Badany parametr	n	Wyniki klasyfikacji na zbiorze:					
		Treningowym		Walidacyjnym		Testowym	
		strata	dokładność	strata	dokładność	strata	dokładność
Liczba klas w jednej iteracji (n)	3	0.5645	0.7423	0.911	0.5975	0.5926	0.732
	4	0.7804	0.694	0.7169	0.7382	0.7125	0.724
	5	0.8632	0.667	0.7444	0.7332	0.9476	0.684

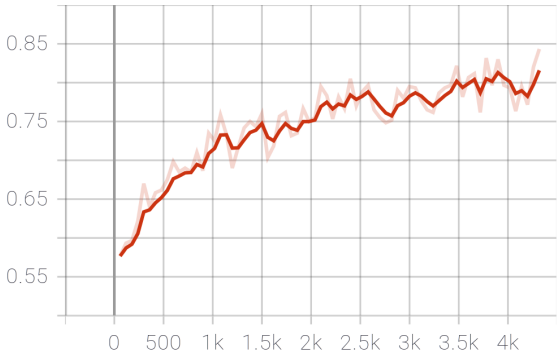
Tabela 3. Porównanie jakości sieci prototypowych dla różnej liczby klas w iteracji.

Analiza wyników wskazuje, że jakość klasyfikatora prototypowego nie wykazuje jednoznacznej zależności od liczby przykładów per klasa. Natomiast osłabia się, gdy zwiększa się liczbę klas, do których ma być podjęta klasyfikacja.

Strata na zbiorze treningowym

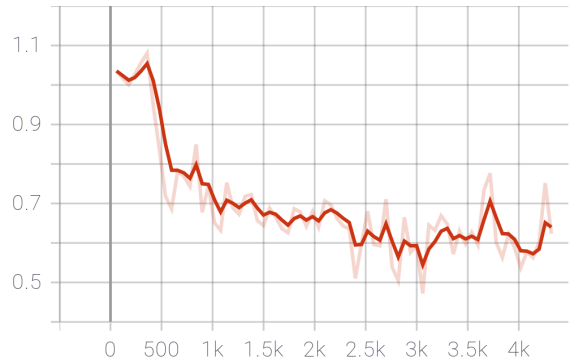


Dokładność na zbiorze treningowym

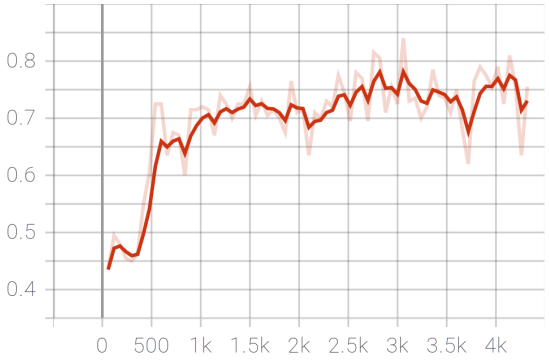


Wykres 3. Klasyfikator prototypowy - strata i dokładność rosną i stabilizują się na wysokim poziomie na zbiorze treningowym.

Strata na zbiorze walidacyjnym



Dokładność na zbiorze walidacyjnym



Wykres 4. Klasyfikator prototypowy - strata i dokładność rosną i stabilizują się na wysokim poziomie na zbiorze walidacyjnym.

Wykres 3 i 4 przedstawiają krzywą uczenia na zbiorze treningowym i walidacyjnym dla klasyfikatora prototypowego. Strata na zbiorze walidacyjnym również z czasem maleje, co wskazuje na to, że model się uczy i jest w stanie dokonywać poprawnych przewidywań dla niewidzianych w czasie uczenia klas.

Wnioski

Standardowy klasyfikator jest w stanie nauczyć się poprawnej klasyfikacji na akceptowalnym poziomie. Oznacza to, że architektura SoundEmbedding jest poprawna i może zostać użyta również w klasyfikatorze prototypowym.

Przy ograniczonej liczbie próbek per klasa standardowy klasyfikator ulega przeuczeniu. Jest to spodziewane, ponieważ przy tak małej ilości danych model się do nich dopasowuje i traci umiejętność generalizacji.

Z uzyskanych wyników wynika, że dla sieci prototypowych liczność *support setu* (*shot*) nie wpływa znacznie na wynik. Widoczne są małe różnice, ale mogą być spowodowane szumem związanym z losowaniem danych w każdej iteracji.

Zauważono natomiast wpływ liczby klas (*way*) na wynik. Im mniej klas tym lepsze wyniki były osiągane. Może to wynikać z tego, że dla mniejszej liczby klas jest większa szansa na to, że są one łatwo separowalne.

Wyników klasyfikatora standardowego i prototypowego nie można bezpośrednio ze sobą porównać, ponieważ sieci te przeznaczone są do nieco różniących się zadań. Standardowy klasyfikator przewiduje dopasowanie do klas w obrębie całego zbioru danych przy pomocy przykładów widzianych w czasie uczenia. Natomiast klasyfikator prototypowy przewiduje dopasowanie do klas w obrębie jednej iteracji przy pomocy niewielkiej liczby próbek każdej z klas dostępnych w czasie dokonywanie predykcji, a uczy się jedynie jak separować klasy.