Urban sound classification

Francesco Tomaselli

July 22, 2021

Contents

1 Introduction

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

The goal of this project is to build a neural network to classify audio files from the *UrbanSound8k* dataset.

This dataset contains audio divided in ten classes, each one representing a different type of city sound, for instance, we can find *car horns*, *dogs barking*, *sirens*, etc. A deeper discussion about the dataset is present in the Subsection 2.1 on the following page.

The presented methodology is composed of three main parts. The first step is to extract relevant features from audio files using the *Librosa* library. This is discussed on Section 2 on the next page.
The next step consists in composing and refining a neural network to classify the data obtained from the previous step. This part is made possible by the *Keras* library and it is discussed in Section 3 on page 5.
Lastly, results from the classification, namely accuracy and standard deviation among test sets, are presented in Section 4 on page 5.

The project is developed in *Python* and the code is structured in a *src* package. Each one of the sub-packages contains code to deal with the different parts of the project. For instance, the *data* folder contains the classes to extract features and to manage a dataset.

In addition to the package there is a *notebook* folder containing the different steps of the project and the various experiments made.

2 Feature extraction

This Sections presents the original dataset structure and the steps followed to create training and test sets from it.

Note that the models mentioned in this section are three layers networks with a reasonable number of neurons and default parameters for the Stochastic Gradient Descent optimizer. Details about models used in the project can be found in the next Section.

2.1 Dataset structure

The dataset contains ten folds of audio samples, each one about four seconds long. The samples are divided in ten classes among ten folds. The following table shows the classes and their relative frequency in the dataset.

Class name	Number of samples
air conditioner	1000
car horn	429
children playing	1000
dog bark	1000
drilling	1000
engine idling	1000
gun shot	374
jackhammer	1000
siren	929
street music	1000

The training set consists of the first four folds plus the sixth, the other folds create five different test sets.

Dataset	Number of samples
Training set	4499
Test set 5	936
Test set 7	838
Test set 8	806
Test set 9	816
Test set 10	837

2.2 First dataset

Choosing the features to extract was difficult as I did not have prior experience working with audio.

The *Librosa* library provides many feature to choose from, for my first try with this dataset I kept it simple by opting for these three ones:

1. *Mel-frequency cepstral coefficients*

2. *Chromagram*

3. *Root-mean-square*

Each feature consists of an array of arrays containing measurements. I applied a series of functions to each sub-array and then concatenated the results in a final feature vector. The functions applied are *minimum*, *maximum*, *mean* and *median*.

This approach resulted in 132 components feature vectors.

Feature scaling After testing some neural networks on the first dataset the results were not promising. One of the reasons is the big difference in ranges among feature vector components.

To mitigate this effect a *StandardScaler* from *sklearn* was applied. This lead to an improvement on the results using the same model as before. The scaler trained on the training set was then used to scale the five different test sets.

2.3 Extended dataset

To improve results on the test sets new features are added to the training one, namely:

1. *Zero-crossing rate*

2. *Roll-off frequency*

3. *Spectral flux onset strength*

After applying the same four functions to these three new arrays, a total of twelve new features are added to the dataset. Scaling yield to promising results on the first dataset, so the same approach is applied to the extended dataset.

After testing a network on the new dataset results improved once again.

PCA Adding new features can be lead to better results in the end but they all need to be useful to the model, so on the extended dataset I experimented with feature selection, in particular I applied the *PCA* algorithm from sklearn.

After some experiments with the number of features to select, 125 out of 144 features were selected. This led to a small improvement in accuracy.