Università degli Studi di Milano

Data Science and Economics

Machine Learning Project

Urban Sound Classification with Neural Networks

Antonella D'Amico
961150

Academic year 2021/2022

Declaration

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 our 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.

Contents

1 Dataset Description **3**

1.1 Variables Description 3

1.2 Features extraction 4

1.3 Data Preparation 5

2 Modeling **7**

2.1 Model 1 7

2.2 Model 2 7

2.3 Model 3 8

3 Results **9**

4 Conclusion **12**

5 Appendix **12**

Abstract. The aim of this project is to implement a Neural Network able to classify urban sound contained in UrbanSound8K dataset. Three different Neural Network architecture was developed with three different features extracted from audio files (MFCCs, Mel-Spectogram and Chromagram) as input.

Dataset Description

1.1 Variables Description

In this experimental project we will use UrbanSound 8k dataset[1]. It contains 8732 labeled sound excerpts ($\leq$ 4 sec) of urban sounds from 10 different classes:

Class	Conut
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

Table 1: Classes and their count

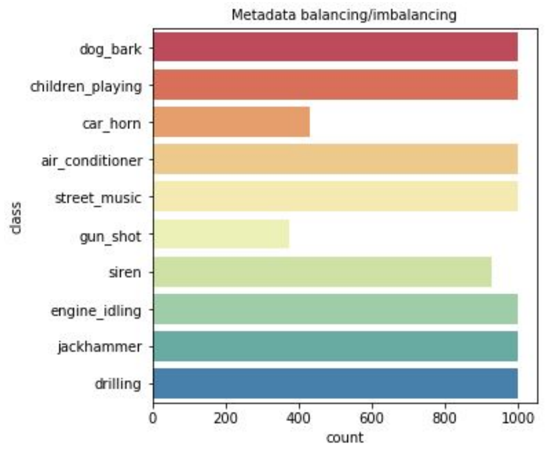
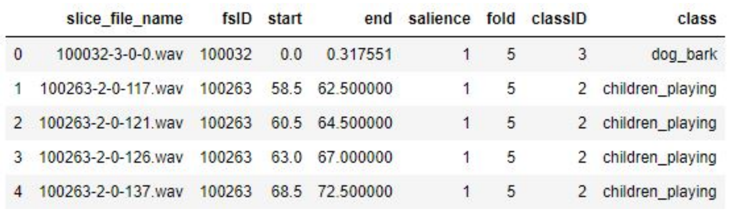


Figure 1.1: Classes distribution

Each sound sample is labeled with the class to which it belongs.
After downloading the dataset we can see that contains two main parts:

- **Audio files** stored in *audio* folder. This folder contains a total of 10 sub-folds labeled from fold1 to fold10. In each of them there are stored a variable amount of audio files in .wav format.
- **Metadata** stored in *metadata* folder. It contains a .csv files where main audio information are listed, such as: filename, numeric classID and the corresponding class according to those showed above.

[1]https://urbansounddataset.weebly.com/urbansound8k.html



	slice_file_name	fsID	start	end	salience	fold	classID	class
0	100032-3-0-0.wav	100032	0.0	0.317551	1	5	3	dog_bark
1	100263-2-0-117.wav	100263	58.5	62.500000	1	5	2	children_playing
2	100263-2-0-121.wav	100263	60.5	64.500000	1	5	2	children_playing
3	100263-2-0-126.wav	100263	63.0	67.000000	1	5	2	children_playing
4	100263-2-0-137.wav	100263	68.5	72.500000	1	5	2	children_playing

Figure 1.2: Metadata file

1.2 Features extraction

From a visual inspection we can see, in **Fig.1.3**, how a children playing wave sound look like.

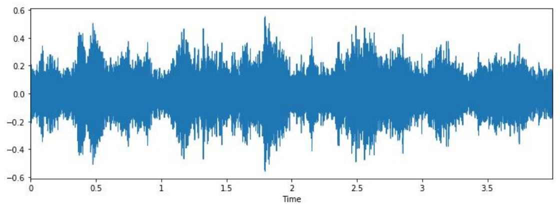
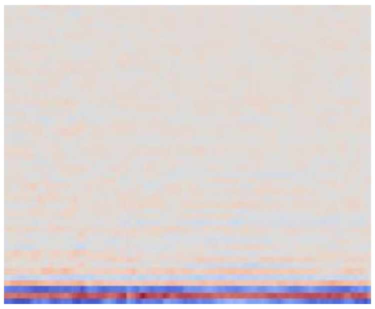


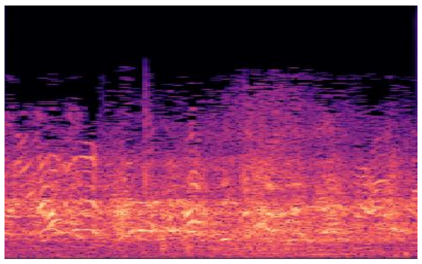
Figure 1.3: Children_playing wave sound

For the purpose of our analysis we extract three main features from each raw time series file. Analyzing the different sound waves of the ten classes contained in the dataset, it is possible to notice how some are very different from each other (the sound of a gunshot is very short if it is compared to a drill); others have repetitive sounds, such as the pneumatic hammer and the drill, and for this reason they are similar in shape. The fact that some sounds may have similar shapes may not be great for a classification analysis point of view. For a more detailed and precise analysis, we decided to extract three of the main features contained in sounds: **MFCCs**, **Mel-Spectogram** and **Chromagram**.

We use librosa library to extract those features by developing three different functions, that were then applied to predefined fold in order to construct our training and test set. For all functions the value for corresponding n_mfcc, n_mel and n_chroma parameters are set eqaul to 50.

MFCCs, Mel-Spectogram and Chromagram can be displayed in plots. In **Fig.1.4** an example of how the raw time series file *17973-2-0-21.wav*, classified as children_playing, can be represented.


(a) MFCCs


(b) Mel-Spectogram

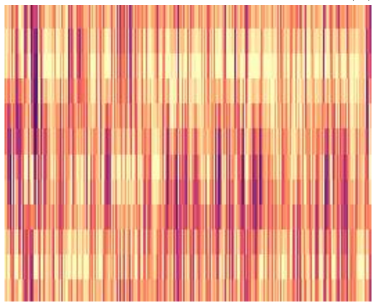

(c) Chromagram

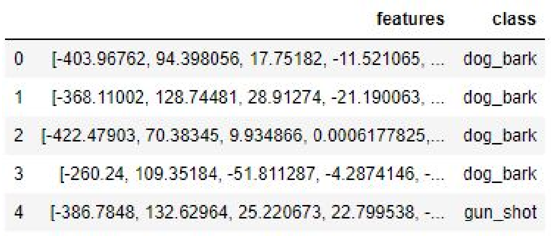
Figure 1.4: Children playing features

1.3 Data Preparation

For the aim of this project we will use folds 1, 2, 3, 4 and 6 as *training set* and folds 5, 7, 8, 9, 10 as *test set*. We extract features in one time[2] using the three functions developed before. We store the obtained results in lists divided for extracted feature and training/test set.

After each extraction and before starting our analysis, features and classes are combined in a dataframe, where the column **features** contains a list of values which corresponds to statistics of the extracted features and column **class** contains the label associated to that file. In **Fig.1.5** an example of a dataframe created with features and class obtained from the use of the function extract_mfcc:

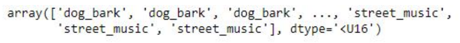
[2]Please note that this operation will take about 30min. It could seems lot of time, but we also performed an experiment by extracting them separately and, at the end, they take the same time. We preferred to put all together so we avoid repetition of code.



	features	class
0	[-403.96762, 94.398056, 17.75182, -11.521065, ...	dog_bark
1	[-368.11002, 128.74481, 28.91274, -21.190063, ...	dog_bark
2	[-422.47903, 70.38345, 9.934866, 0.0006177825,...	dog_bark
3	[-260.24, 109.35184, -51.811287, -4.2874146, -...	dog_bark
4	[-386.7848, 132.62964, 25.220673, 22.799538, -...	gun_shot

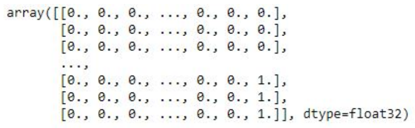
Figure 1.5: Training dataframe for MFCCs features

We perform this operation for all the extracted features. Then, we proceed by splitting features and class, again for all the extracted features, both for training and test set. Since class are categorical we need to transform them into binary variable by using One-Hot Encode techniques. Thanks to techniques we are able to create number of dummy variables equal to the number of class, in our case 10, and for each row mark with 1 the dummy that corresponds to the category associated with that file and 0 otherwise.



```
array(['dog_bark', 'dog_bark', 'dog_bark', ..., 'street_music',
       'street_music', 'street_music'], dtype='<U16')
```

(a) before OHE



```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 0., ..., 0., 0., 1.]], dtype=float32)
```

(b) after OHE

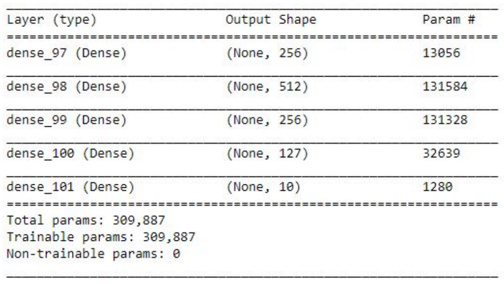
Figure 1.6: y_train data for MFCCs

Modeling

To perform our analysis three different models of Neural Networks were implemented. For all of them **Categorical Cross-entropy** loss function and **Adam** optimizer were used. **Softmax** activation function is used in output layer of all the three model, while for hidden layers **Rectified Linear Unit**, also known as ReLU, activation function is used. The latter is the most communly used activation function in deep learning models. It return 0 if it receives any negative input, but for any positive value x, it returns back that value.
For all models batch size is set to **32** and epochs to **100**.

2.1 Model 1

The first model is composed by three hidden layers with a corresponding number of neurons equal to 512, 256 and 127.

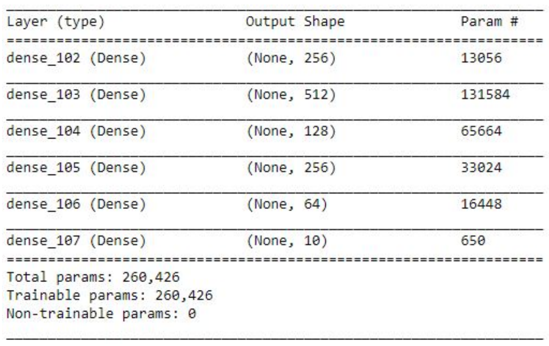


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_97 (Dense)             (None, 256)               13056
_________________________________________________________________
dense_98 (Dense)             (None, 512)               131584
_________________________________________________________________
dense_99 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_100 (Dense)            (None, 127)               32639
_________________________________________________________________
dense_101 (Dense)            (None, 10)                1280
=================================================================
Total params: 309,887
Trainable params: 309,887
Non-trainable params: 0
_________________________________________________________________
```

Figure 2.1: Summary of Model 1

2.2 Model 2

In the second model we add an hidden layers of 64 neurons with respect to Model 1.

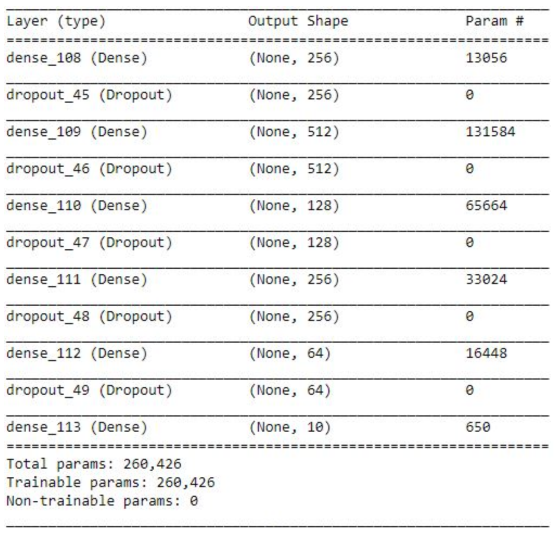


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_102 (Dense)            (None, 256)               13056
_________________________________________________________________
dense_103 (Dense)            (None, 512)               131584
_________________________________________________________________
dense_104 (Dense)            (None, 128)               65664
_________________________________________________________________
dense_105 (Dense)            (None, 256)               33024
_________________________________________________________________
dense_106 (Dense)            (None, 64)                16448
_________________________________________________________________
dense_107 (Dense)            (None, 10)                650
=================================================================
Total params: 260,426
Trainable params: 260,426
Non-trainable params: 0
_________________________________________________________________
```

Figure 2.2: Summary of Model 2

2.3 Model 3

Third model is composed by the same numbers and corresponding neurons of Model 2, but in order to prevent overfitting a **Dropout** equal to 0.2 is added for each hidden layers.

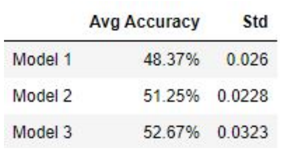


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_108 (Dense)            (None, 256)               13056     
_________________________________________________________________
dropout_45 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_109 (Dense)            (None, 512)               131584    
_________________________________________________________________
dropout_46 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_110 (Dense)            (None, 128)               65664     
_________________________________________________________________
dropout_47 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_111 (Dense)            (None, 256)               33024     
_________________________________________________________________
dropout_48 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_112 (Dense)            (None, 64)                16448     
_________________________________________________________________
dropout_49 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_113 (Dense)            (None, 10)                650       
=================================================================
Total params: 260,426
Trainable params: 260,426
Non-trainable params: 0
_________________________________________________________________
```

Figure 2.3: Summary of Model 3

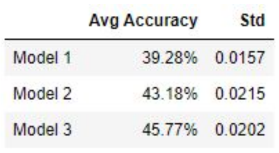
Results

Three different models where used in this project with three different types of features as input (MFCCs, Mel-Spectogram and Chormagram).



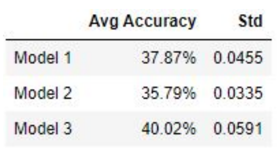
	Avg Accuracy	Std
Model 1	48.37%	0.026
Model 2	51.25%	0.0228
Model 3	52.67%	0.0323

(a) MFCCs



	Avg Accuracy	Std
Model 1	39.28%	0.0157
Model 2	43.18%	0.0215
Model 3	45.77%	0.0202

(b) Mel-Spectogram

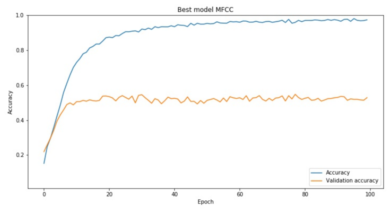


	Avg Accuracy	Std
Model 1	37.87%	0.0455
Model 2	35.79%	0.0335
Model 3	40.02%	0.0591

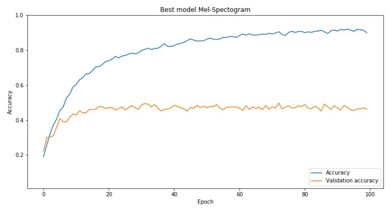
(c) Chromagram

Figure 3.1: Average Accuracy for all the features

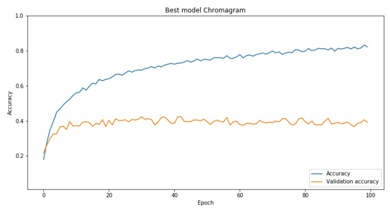
Fig.3.1 reports average accuracy and standard deviation of each model divided for features.
Best model for each feature is **Model 3**, with corresponding accuracy of 52.67% for MFCCs, 45.77% for Mel-Spectogram and 40.02% for Chromagram.
They are very low values, this means that prediction will not be so precise. To go more in depth, if we compare the accuracy and the validation accuracy of those best model for each of the extracted features, we can notice that the models are overfitting. This is due to the fact that there is a strong difference between training accuracy, which is relatively good, and validation accuracy which is very low and so, not good.



(a) MFCCs



(b) Mel-Spectogram

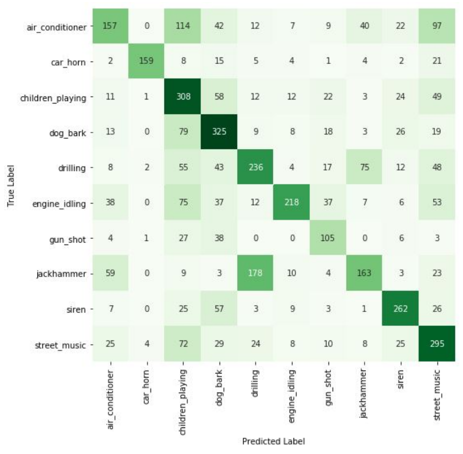


(c) Chromagram

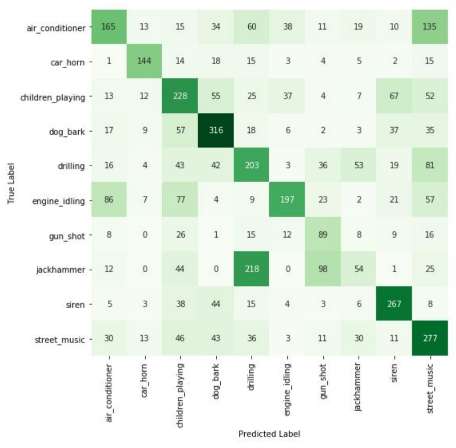
Figure 3.2: History for best model of each feature

This not optimal result can be due to the fact that our models have memorized files contained in the training set, without having learned how to correct map the features given in input to the respective class.

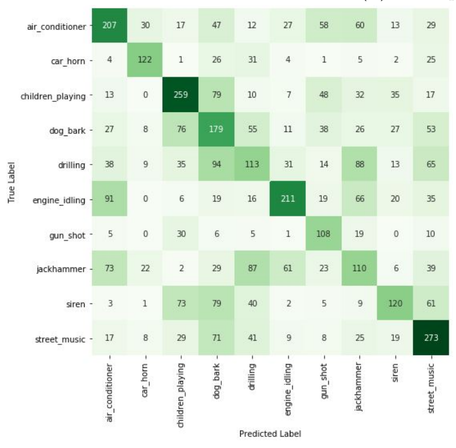
It was earlier stated in data description section, that it is difficult to visualise the difference between some of the classes. In particular, the following sub-groups are similar in shape: repetitive sounds for drilling and jackhammer, spikes for dog bark and gun shot. By plotting the confusion matrix for the best model (Model 3) for each features category we are able to see if also our model struggled to differentiate those classes. In confusion matrix rows represents true labels while columns those predicted by the model.



(a) MFCCs



(b) Mel-Spectogram



(c) Chromagram

Figure 3.3: Confusion matrices

In 3.3.(a) we plot the confusion matrix of Model 3 which takes as input Mel-frequency cepstral coefficients. As we can see, air conditioner is mostly confused with children playing and street music. Jackhammer is confused with drilling, but this is not surprising since they have similar shape (repetitive) if we look at their wave sound.

In 3.3.(b) we have the confusion matrix of Model 3 which takes as input Mel-spectogram. Here, air conditioner is largely confused with street music, as before, and also with drilling and engine. Jackhammer is highly confused with drilling, more than in the previous case where MFCCs were used as input.

In 3.3.(c) confusion matrix of Model 3 with Chromagram as input is displayed. Drilling here is largely confused with dog bark and jackhammer. We see an improvement in air conditioner performance with respect to use MFCCs as input for the same model.

Conclusion

In this experiment three different models were created in order to classify urban sounds contained in the UrbanSound8K dataset. From each audio file three different features were extracted and then given as input to our different models. The performance of models developed in this project are not so high. Despite that, Model 3 is able to classify sound clips to a high degree of accuracy, with respect to the other two models.
MFCCs seems to be the best features that can be used to classify audio data with respect to the other two extracted features.
Overall, the best model in our experimental project for urban sound classification is Model 3 with MFCCs feature as input.

Appendix

Link to Github Repository:

https://github.com/antonelladamico17/Urban-Sound-Classification