Urban Sound Classification

Audio Pattern Recognition Course. February 21, 2022

Flavio Furia
Università degli Studi di Milano
flavio.furia@studenti.unimi.it

***Abstract*—Predicting urban sounds has become an interesting field of research in the recent years, also thanks to the increasing quantities of online multimedia content of urban scenes; other than that, it is also a quite challenging task since audio samples are often recorded in a noisy environment. In this work, we face the problem of urban sound classification by applying feature extraction and machine learning techniques on audio data coming from the UrbanSound8K dataset. In particular, statistical features from both the frequency and time domain, other than mel and chroma representation, are computed and used to train Random Forests and Support Vector Machines, while mel-spectrogram images of the raw audio signals are produced and fed into Convolutional Neural Networks. The final aim is not only to obtain good classification scores but also to make a comparison of these two different approaches, keeping an eye on both space and time requirements, since all the experiments are performed on a machine with limited hardware resources.**

***Index Terms*—urban sound classification, audio classification, machine learning, feature extraction, mel-spectrogram, convolutional neural network, multiclass classification**

I. Introduction

In the past years, the higher and higher availability of data has made machine learning a powerful and very useful tool to classify and predict observations coming from many different fields. The audio field makes no exception and it is indeed living a time of great improvement and development in terms of analysis, thanks to the ever increasing availability of massive and well organized audio data. All of this allows to apply machine learning techniques on audio data in order to solve a lot of tasks, ranging from sound event detection to music genre classification and speech recognition. In particular, the activity of predicting from which source an urban sound comes from can be referred as automatic urban sound classification and it can be very useful to assess noise pollution and to perform multimedia retrieval and urban informatics. The research on this particular field is now growing larger and larger and a huge step in this direction has been made thanks to the public availability of urban sounds dataset. One of them, *UrbanSound8K* [1], is the actual object of our study and analysis. The most successful results on urban sound classification has been achieved by using 2D convolutional neural networks on spectrogram-based images of audio or 1D CNNs [3] on raw audio files; other approaches, instead, involve recurrent neural networks like LSTM [2] or GRU in order to exploit their capability of "remembering sequences" (this has shown to be particularly good for speech recognition). While neural networks are capable of automatically extracting the features of audio data and then using them to perform classification, different techniques like ensemble methods or clustering/nearest neighbours classifiers can be equally successful if all the process of feature extraction is performed before feeding the data as input to these algorithms. Our works aims at comparing these two approaches by using convolutional neural networks on mel-spectrogram images obtained directly from the raw audio data and random forests and support vector machines on statistical features computed on time and frequency domain, together with mel and chroma based representations, extracted from the same audio files. The overall results obtained on the 10 predefined splits generated by the authors of the dataset are not comparable to state of the art ones (both for poor hardware and time available resources) but are promising and can surely help to understand how a deep learning environment compares to simpler models when dealing with audio data. Another interesting finding is how features coming from different domains participate in classifying urban sounds.

II. System Overview

As already said, two different approaches were taken into consideration, the first one consisting in the computation of statistical features based on time and frequency domain features and the same for mel and chroma based representations. in 1, the overall process is shown. All the chosen features are computed for each audio sample, giving as return a vector (or an image for the image-based features) on which some statistics are computed: in particular, the minimum, maximum, mean, median, standard deviation of each vector is computed, giving a good statistical-based representation of each feature. Mel scale and chroma representations return arrays (images) as output and all values of a row are related to a coefficient of that representation: just to give an example, an MFCC representation of the signal is an array having one row for each coefficient. For these features, each row/coefficient was considered as a single features, thus the statistics are computed on each row per se. At the end of the feature extraction step, we have a tabular dataset with samples as rows and features as columns. For each column, mean and standard deviation are computed and then subtracted and divided, respectively, in order to have values centered around 0 and with a standard deviation of 1. A brief explanation of all the chosen features will now follow.

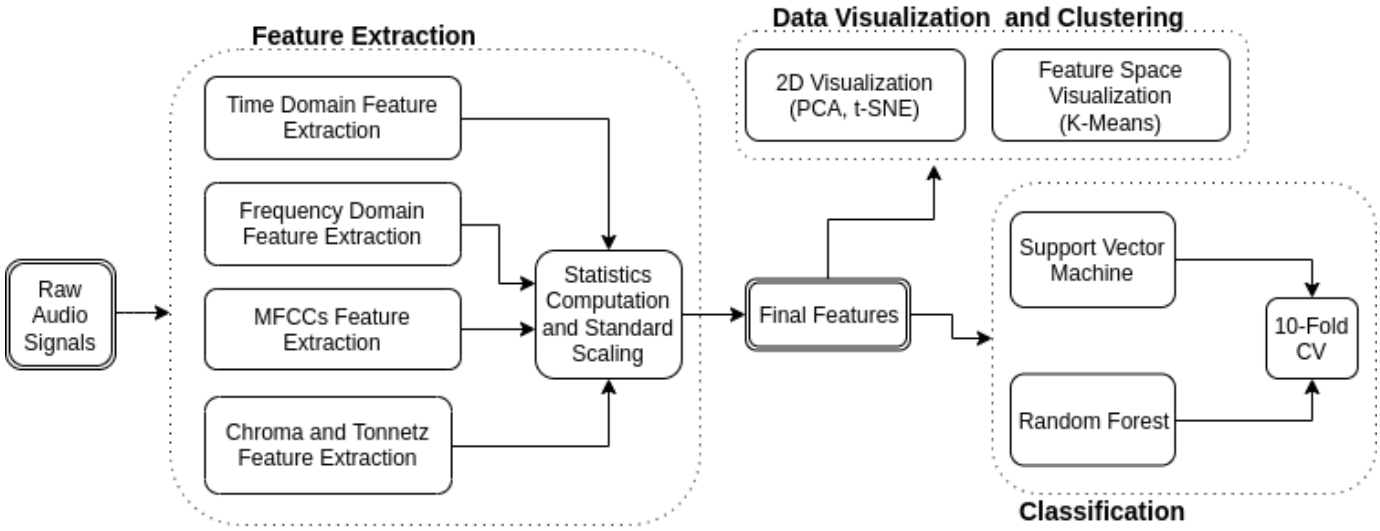

Fig. 1. Block Diagram of Classification using Statistical Features.

A. Time Domain

Relying on the raw audio signal, these kind of features focus on giving information about how the amplitude of the signal changes with respect to time. The first quantity considered is *amplitude envelope*, measuring the maximum amplitude value of all samples in a single frame of an audio signal; being that it focuses on max value, it aims at giving a rough idea of the loudness of the signal but, at the same time, it is sensible to outliers. The second one is the *root-mean square energy*: the energy of a sample is the square of its amplitude, thus we just take the root-mean square of all samples in a frame, obtaining another indicator of loudness but less sensitive to outliers than the amplitude envelope. *Zero crossing rate*, the final one, measures the number of times a signal crosses the horizontal axis in a frame; it is often related to the pitch of sound. In the end, after computing statistics, a vector of $3 \times 5 = 15$ (3 is the chosen amount of features and 5 is the amount of statistics computed for each of them) values for a single audio sample is produced.

B. Frequency Domain

These features rely on a spectrogram, which is obtained by applying Fourier Transform on the raw audio signal. *Spectral centroid* is the first chosen feature, measuring at which frequency value the magnitude is centered, computed for each frame. It is often related to the "brightness" of sound. The next one, the *spectral bandwidth*, is related to the first one, as it measures the distances of frequency bands from the spectral centroid. The *spectral rolloff* tells under which value of frequency some percentage of the total energy of the spectrum is contained, while the *spectral flatness* quantifies how much noise-like a sound is; they're both related to noise and are the last two spectral features employed. In this case, 4 features lead to $4 \times 5 = 20$ total features.

C. Mel Cepstrum Representations

The mel scale measures pitches as a function of frequency; it is a logarithmic scale thus it well approximates the way human perceives sound. *Mel-frequency cepstral coefficients* (MFCCs) realize this intuition by converting frequency to mel scale and giving a sort of time/mel representation of sound; we consider the first 13 coefficients, as they have proven to be the most informative, concatenated with the delta and delta-delta, i.e. the first and second-order frame-to-frame difference. In the end, as already explained, we take each row/coefficient vector and compute statistics on it, obtaining a total of $13 \times 3 \times 5 = 195$ feature values. In general, Mel based representations are related to how we as humans perceive sounds, thus they have been employed with particular succesful results in speech recognition and musical analysis: our choice is based on the desire of looking at how they can perform in a different environment.

D. Chroma Representations

A *Chromagram* is based on the aggregation all the spectral information related to given pitch classes into a given number of bins, usually 12, representing harmonic characteristic of music. *Tonnetz* (also called tonal centroids) features are very similar, as they represents the chromas by projecting them onto a 6-dimensional basis. Chroma features are, of course, deeply used in musical analysis and music genre recognition and, how it is for MFCCs, it could be interesting to look at how well they do in urban sound classification. Regarding the feature extraction, both chromagrams and tonnetz features give image representations, thus on each row/coefficient vector statistics are computed, obtaining $((12 + 6) \times 5) = 90$ feature values.

E. Mel-spectrogram images

The second approach is based on computing mel-spectrogram images of audio signals and training CNNs with

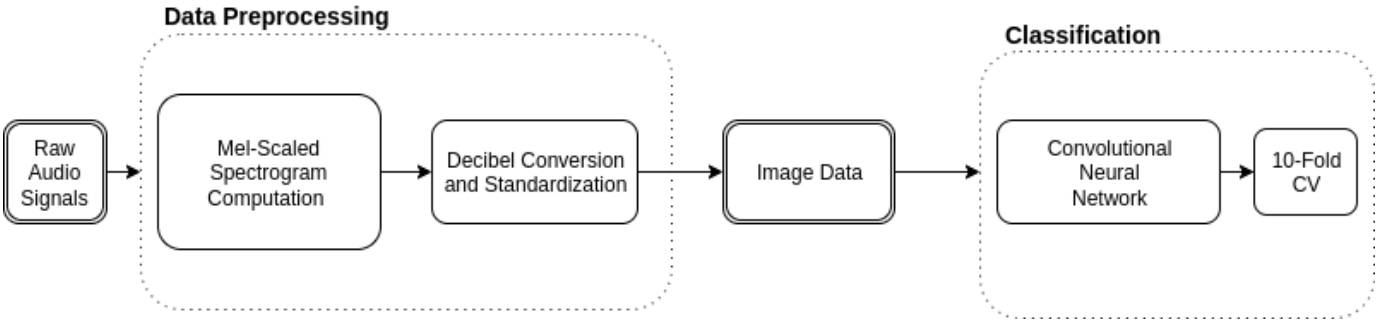

Fig. 2. Block Diagram of Classification using Mel-spectrogram Images.

them. They are obtained by applying, as usual, a Fourier Transform (an STFT in this case) to move into the frequency domain, together with the conversion of amplitude to decibels in order to obtain a logarithmic representation (we can also call this a *log-mel-spectrogram*). Next, a number of mel bands is chosen: in our case, the choice falls on 64 mel bands, since higher ones did not provide great improvements on the results but, at the same time, they made experiments harder to be performed in terms of time and space requirements. In the end, mel filter banks are constructed and applied to the spectrogram. This kind of representation has been widely used in speech recognition together with both 1D and 2D CNNs and recurrent networks.

F. Visualization and Clustering

Dimensionality reduction is a feature selection or extraction tool often used to deal with a smaller but not less informative variant of the original dataset. In our case, we reduce the dimensionality of the dataset to 2 dimensions only for visualization aims, in order to have a look at the data from an higher point of view. In particular, we apply *Principal Component Analysis* and *t-Distributed Stochasting Neighbour Embedding* on the scaled tabular dataset obtained after the statistical features extraction. PCA is a linear transformation that maps high-dimensional points into a lower-dimensional representation trying to preserve the maximal amount of variance, with the idea that greater variance is followed by a greater amount of information; t-SNE, on the other hand, considers every data point as the center of a Gaussian probability distribution in the original space and the center of a Cauchy probability distribution in the lower-dimensional space, with the aim of minimizing their Kullback-Leibler divergence. In this way, close points in the first space will be also close in the reduced space. We use both of them in order to have an idea of the level of correlation (linear or non linear) between samples and labels.

As another data analysis step, we apply *K-means*, a clustering algorithm that given a set of points in an Euclidean space and a number of possible clusters, groups points together by assigning each one to the closest cluster, in an unsupervised way. We did it for different values of K and for different sets of features: in this way we can compute *Silhouette scores*, values that for different K allow to have an idea of how much the data are clustered correctly; *elbow method*, a heuristic that determines the optimal number of clusters in a dataset, is also performed. We also use the labels obtained by clustering the data with $K = 10$ and visualize them with t-SNE reduced representation: KMeans labels are not directly related to the true 10 classes, but it helps at showing how the feature space is distributed.

G. Classification

As already stated, we are in a supervised multi-class classification environment, as there are 10 possible classes. To face this task, the statistical features dataset is used as input of two different models, a *random forest* and a *support vector machine*, while the mel-spectrogram image dataset is used as input to a *convolutional neural network*.

Random forests are ensemble learning methods that consist in gathering together many decision tree and classifying by majority vote or similar strategies. They're powerful, faster to train with respect to deep neural networks and are robust to outliers and noisy data. If we assume a dataset is a collection of points belonging to a d-dimensional space, with d being the total number of features, we can say that a support vector machine is an algorithm that aims at computing the hyper-plane that well separates two classes. In a multi-class environment, the problem is faced by moving to a *one-vs-rest* or *one-vs-one* approach, in which several binary classification models are built and compared together in order to make a single prediction. In case of non-linearly separable dataset, a kernel function can be applied on the dataset to transform it to an higher dimensional space. In this way we can obtain a classifier that is linear in the new space, even if it is non-linear in the original one. This is a very powerful technique that is often applied when dealing with complex datasets.

In the recent years deep learning has received more and more interests thanks to latest discoveries and increasing availability of data and hardware resources: deep neural networks are very powerful techniques that are capable of extracting feature for raw data and learning them to perform classification. In our case, the chosen architecture for the mel-spectrogram image dataset are convolutional neural networks: CNNs are feedforward networks with the capability of adaptively learn

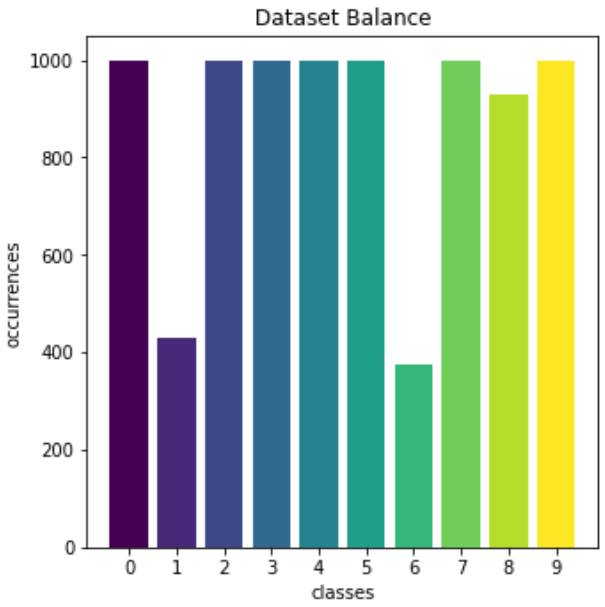

Fig. 3. Class Balance.

the spatial hierarchies of features, making them well suited to deal with image data. The building block of these networks are convolutional layers, which take a kernel matrix and by using the convolution linear operation learn the features of small portion of the given input; then, pooling layers are used to downsample the input. In the end, all the computed features are usually flattened and given as input to a sequence of fully connected layers. The output is computed by using one neuron for each class with softmax activation functions in case of a multi-class classification problem. Deep learning is of course one of the most powerful techniques to deal with regression and classification but, at the same time, it is very costly and requires lots of data and hardware resources.

III. Experimental Setup

All the experiments[1] have been carried out using the Python3 programming language on a low-end PC without a GPU Card: this caused some issues in terms of the amount of trials that could be performed, especially the ones involving CNNs. Some help has been brought by the Google Colaboratory Platform since it provides free access to GPU computation but, the limited amount of available main memory together with the decaying performances made it impossible to totally rely on the platform. All of this has made a big impact on the design choices especially in terms of CNN architecture and mel-spectrogram images dimensions. An overview of all the experiments will now follow.

[1]All the code is available on our Github repository: https://github.com/flaviofuria/Urban-Sound-Classification

A. Dataset and Features

As already introduced, the *UrbanSound8K*[2] is a collection of 8732 annotated urban sounds, which the authors provide in 10 predefined splits to allow reproducibility and comparison, together with others metadata. The vast majority of the samples have a duration of exactly 4 seconds and can belong to 10 possible classes: *air conditioner*, *car horn*, *children playing*, *dog bark*, *drilling*, *engine idling*, *gun shot*, *jackhammer*, *siren*, *street music*. The dataset has a just slight imbalance (3) in terms of occurrences of car horn and gun shot classes, as they appear in 429 and 374 samples, respectively, while there are 929 samples labelled as siren and all the other classes have exactly 1000 occurrences.

All the part of preprocessing and feature extraction was carried out by using Librosa[3], a huge Python package for music and audio analysis: all the samples were loaded with a single channel and a *sampling rate* of 22050, a popular chosen quantity since it is a good compromise between detail and space complexity. The resulting size of each sample was $22050 \times 4 = 88100$, applying zero padding to samples with a duration of less than 4 seconds. All the statistical feature extraction was done using tools made available by Librosa, with the exception of the amplitude envelope which is not provided: the *frame size* was set to 1024, while the chosen value for the *hop length* was 256; in this way, all the feature vectors had $\lceil 22050 \times 4/256 \rceil = 345$ values to extract statistics from. A similar thing stands for the spectrogram representation: having chosen to keep the first 13 coefficients, the resulting MFCC representation was a 13×345 image, while the chroma and tonnetz representation were 12×345 and 6×345 images. In the end, after all the statistics computation of the features, a tabular dataset of size 8732×320 was obtained, on which we applied standard scaling.

As already said, the CNN was fed with mel-spectrogram images of data: for matters of time and space complexity, we chose to keep the first 64 mel bands, and the spectrogram was computed with a frame size of 1024 and a hop length of 512, obtaining $64 \times 173 \times 1$ grayscale images (higher values would have slowed too much the CNN training). After computing the mel-spectrogram and moving to a decibel scale, every image was standard scaled in order to be centered around 0 with a standard deviation of 1.

B. Visualization and Clustering

After scaling the statistical feature data, PCA and t-SNE with the Sklearn implementation were applied on the dataset, in order to obtain 2D representations of data; the default Sklearn parameters were chosen, with one only change for t-SNE: it was computed after pre-reducing the dimension to 50 with PCA, which is a popular heuristic to use when using this algorithm. In the first two images of 4 the result of dimensionality reduction is shown: as one could expect, t-SNE gives a better representation, even if there is no clear

[2]The dataset is easily downloadable from this website: https://urbansounddataset.weebly.com/urbansound8k.html

[3]https://librosa.org/

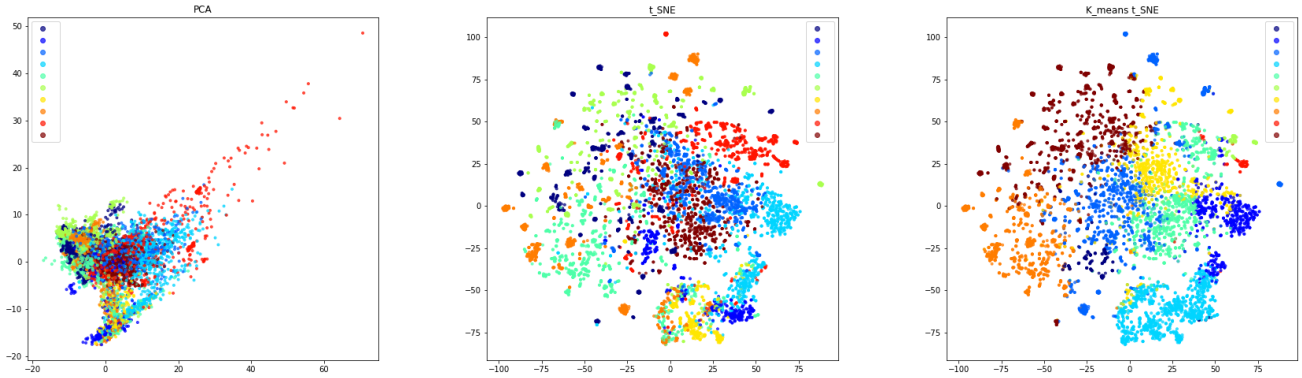

Fig. 4. Data visualization.

separation among the data, probably leading to difficulty in classification.

Clustering was applied on the scaled data using the Sklearn implementation of KMeans on different combination of features and with K ranging from 2 to 12: the third image in 4 shows the result of using the clustering labels of KMeans for all the features with $K = 10$ on the t-SNE representation of data. In 5 silhouette scores and inertia (which measures how internally coherent clusters are and is used for the elbow heuristic) are plotted for all the values of K used and for all the combination of features: the two plots shows that KMeans is not capable at capturing all the relation among the data, as we would like to have 10 clusters but we don't have a clear elbow around $K = 10$.

C. Models

The models built to deal with the statistical features data were implemented using their Sklearn implementation, which combine both ease of use and a lot of available hyperparameters. For the random forest, we chose to use 1000 decision trees, each one with a maximum depth of 15, trying to keep this value as low as possible to avoid overfitting. The best split was considered by using the square root of the amount of available features and the *gini* criterion to measure the quality of each split. We also chose to balance the weight of classes in order to face the slight imbalance issue of the dataset. The support vector machine was built in order to learn a decision boundary that could well adapt to the data: for this reason, a *radial basis function* kernel was used with $C = 7$, which is a regularization parameter to avoid overfitting and, also in this case, we chose to balance the class weights.

The CNN was built by relying on the Keras/Tensorflow2 libraries: it is made of a first 2D convolutional layer with a kernel size of 3 and 32 filters followed by a 2D max pooling layer with a stride of 2 and a dropout of 0.4 to avoid overfitting; next, there is another triplet of 2D convolutional layer (kernel size of 5 and 64 filters), 2D max pooling layer and dropout with the same hyperparameters. A last 2D convolutional layer with a kernel size of 5 and 64 filter is followed by a 2D global average pooling layer. In the end, all the input is flattened and pass through 10 output neurons with a softmax activation function (for all the other layers the relu activation functions is used instead). The final output is a 10-valued vector of probabilities that can be interpreted as the likelihood that the sample belong to each of the 10 classes, with the higher value being the final predicted label of the network. *Adam* with a learning rate of 0.001 and *categorical crossentropy* are the chosen optimizer and loss function, respectively. The training part is performed for 100 epochs with a batch size of 128, with two additions: Keras automatically take a validation set of size 0.1 with respect to the training size and use it to stop the training if for 20 epochs the validation accuracy does not increase; at the end of the training, the weights according to the epoch with the best validation accuracy are saved and used for the evaluation, with the idea that an higher validation accuracy is followed by a better capability of generalizing, since validation set is not used to adjust the weights of the network.

Layers	Units / Kernel	Notes
Input		shape=64 × 173 × 1
Conv2D	32 / 3	activation=ReLU
MaxPooling2D		strides= 2
Dropout		rate=0.4
Conv2D	64 / 5	activation=ReLU
MaxPooling2D		strides=2
Dropout		rate=0.4
Conv2D	64 / 5	activation=ReLU
GlobalAveragePooling2D		strides=2
Flatten		
Dense	10	activation=Softmax

TABLE I
ARCHITECTURE OF THE CNN.

IV. RESULTS

All the three models were evaluated using accuracy metric and confusion matrices, validating the results by the employment of 10-fold cross-validation, with some issues: we have

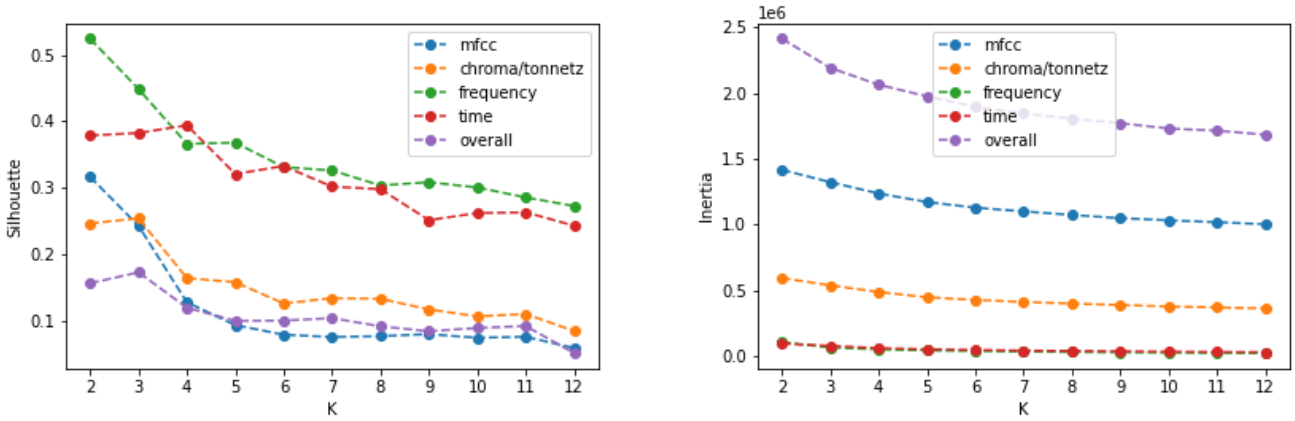

Fig. 5. Silhouette and Elbow.

built 10 random splits and obtained remarkable average scores with all the three models, but we can not say the same thing about the evaluation performed on the 10 predefined folds provided by the authors. We now show the crucial part of the results we have obtained; refer to our Github Repository to have a look at all the scores for each model and each fold.

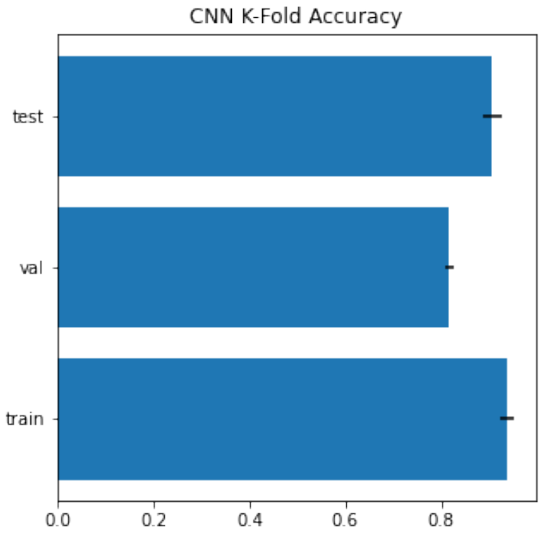

Fig. 6. CNN accuracy with deviation among the 10 random splits.

A. Random splits

Using 10-fold cross-validation with random splitting, we have obtained an average test accuracy of 0.897 with the random forests and 0.945 with the support vector machines, meaning that the latter was more capable at generalizing, since both of them reached a training accuracy of 0.99 that is of course caused by overfitting. As for the CNNs, even if they suffered less from overfitting, they performed slightly worse, reaching an average training accuracy of 0.936 a validation accuracy of 0.816, and a test accuracy of 0.906; even if slightly worse, the obtained scores are overall consistent among all the folds, as it can be seen in the deviation of accuracy, in 6.

B. Predefined Splits

Learning data with predetermined splits was much harder, leading to definitely worse scores: Random Forests and Support Vector Machines only managed to reach an average test accuracy of 0.66 and 0.71 respectively, while CNNs stopped at 0.705. Despite the several regularization techniques applied, all the three models ended up overfitting, with the first two reaching an average accuracy of 0.99 and the CNNs 0.92. Just to give an idea, in 7 the confusion matrix of the CNN trained on the 8-th predefined fold, which was the worst in terms of scores, is shown: it is clear that the model was not capable at all at recognising some of the classes and on the best one it did not even managed to well classify the 90% of them.

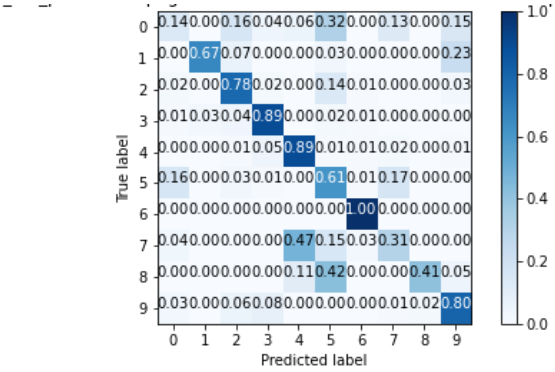

Fig. 7. CNN scores on the 8-th predefined fold.

V. Conclusion and Future Works

In this work we faced the problem of urban sound classification by relying on two slightly different approaches: we used

statistics computed from features coming from both time and frequency domain, together with MFCCs and Chroma representation of audio. Although some of these features are deeply used in speech recognition and music genre classification, they have proven to be useful also when working on an urban sound environment.

Other than that, we computed mel-spectrogram images from audio data and used them to train CNNs, showing that it is possible to move to the deep learning world even when low hardware resources are available; after our experiments, we came to the conclusion that deep learning is not always necessarily better: our Random Forests and Support Vector Machines were definitely less computationally demanding, both in time and space, but still managed to obtain comparable scores, if not even better.

This is of course a starting point: the next step should be taken toward the improvement of the scores on the predefined splits, by refining the chosen features and statistics and, maybe, by applying some data augmentation: this could be done in order to increase the amount of data available for training and, hopefully, to improve the generalization capabilities of our models. Another step in this direction can be probably made by refining the mel-spectrogram images computation: the $64x173$ size was chosen to not make the experiments even slower than what they were: bigger images, together with a deeper convolutional neural network, could lead to better scores.

REFERENCES

[1] Salamon, Justing and Jacoby, Christopher and Bello, Juan Pablo, A dataset and taxonomy for urban sound research, Association for Computing Machinery, 2014.

[2] Joy Krishan Das, Arka Ghosh, Abhijit Kumar Pal, Sumit Dutta, Amitabha Chakrabarty, Urban sound slassification using convolutional neural network and long short term memory based on multiple features, Fourth International Conference On Intelligent Computing in Data Sciences (ICDS), 2020

[3] Mohammed Gamal Ragab, Said Jadid Abdulkadir, Norshakirah Aziz, Hitham Alhussian, Abubakar Bala and Alawi Alqushaibi, An ensemble one dimensional convolutional neural network with bayesian optimization for environmental sound classification, 2021