“Analysis and comparison of the main techniques designed for Environmental Sound Classification”

Mattia Pujatti [†]

Abstract—**Nowadays, the interest of the scientific community about the relation between machine learning and sound is mainly focused on speech recognition tasks, and less relevance is left to problems like environmental sound classification, also due to the scarcity of publicly large available datasets. In this paper we will present the main techniques developed so far for this kind of tasks, applied over an environmental sounds collection known as ESC-50. We are going onto two parallel directions, showing how those audio files can be represented as vectors of high-level sound features, and then showing how the usage of Convolutional Neural Networks has revolutionized the classification framework, proposing itself as an alternative to the traditional methods. In particular, regarding the first approach, we propose a preprocessing pipeline together with a nested cross-validation, aimed at gaining the highest accuracy possible, together with a reliable statistics of the results. This proposal will be successful, in our implementation, since it will be shown to guarantee better performances than the actual (non-CNN) state of the art.** [1]

Index Terms—**Supervised Learning, Neural Networks, Audio Classification, Nested Cross Validation, Sound Features.**

I. Introduction

The actual state of the art for Environmental Sound Classification foresees the usage of Convolutional Neural Networks over datasets constructed collecting all the spectrograms derived from the audio clips. The main reason is that several architectures have been proven to outperform all the approaches based on high-level audio feature extraction.
In this work, however, we have developed a feature extraction technique that allows to reach quite nice scores that, merged with the higher speed of the method and the less memory consumption, could represent a gratifying alternative to CNNs and maybe deserve a closer analysis.
Lots of effort was dedicated, during the years, in order to develop always more precise and efficient audio classification models, via the analysis of Mel coefficients and spectrograms. However, these kind of researches were almost all dedicated to speech recognition tasks, that, nowadays, have much more relevance in our technological society, mainly because of the interactions among humans and devices. But, since most of the interest of the scientific community was ”limited” on making machines to learn human language, a definitive and efficient approach for generic sound classification assignments is still missing. More precisely, environmental sound classification was so put aside, that there are only a few public available datasets of a decent dimension ([1], [2]).
The purpose of this project will be to provide an efficient way, using machine learning techniques, to classify environmental sound clips belonging to one of those collections, called ESC-50 [1]. Several approaches have been tested over it during the years, but only a few of them were able to reproduce, or even overcome, the human classification accuracy, estimated around 81.30%.
The analysis will be organized in the following way: since the very first approaches were mainly focused on the examination of audio features, that one could extract from raw audio files, we will provide a way to collect and organize all those ”vectors of features”, and use them to distinguish among different classes. Then, different classification architectures and techniques will be implemented and compared among each other, in order also to show how they effectively react to different data manipulation (in term of overfitting, numerical stability, etc.). In the second part, instead, it will be shown that all those feature classifiers, without exceptions, underperform when compared to the results provided by the usage of Convolutional Neural Networks directly on audio signals and relative spectrograms (so without any kind of feature extraction), and how this new approach opened for a large number of opportunities in term of models with high accuracy in sound classification.

II. Related Work

In recent years, audio recognition systems have gained so much popularity in the machine learning world, mainly because of the developments in the artificial intelligence field. However, the research has focused mostly on speech recognition tasks, music classification and acoustic scene classification, while identification of environmental sounds was took less in consideration. One of the main reasons for this, as anticipated, was the scarcity of suitable and publicly available datasets. But when, in 2015, K. Piczak realized and made public a collection of 2000 environmental short sound clips, under the name of ESC-50 dataset [1], people’s interest started rising again, together with a sort of ”competition”, in which many classification methods were applied in order to obtain the higher accuracy over it. In the github page of the author [3] there is, in fact, a table summarizing more than 30 papers written on top of ESC-50, each of them proposing

[†] Physics of Data, Department of Physics and Astronomy, University of Padova, email: mattia.pujatti.1@studenti.unipd.it

[1] For the practical Python implementation, and for a step-by-step analysis of the problem, please refer to the official Github repository: https://github.com/MattiaPujatti/Environmental-Sound-Classification.

a different approach to classify those sounds. Piczak itself, in its first papers [1], suggested the usage of a few "canonical" machine learning classifiers (mostly based on features extraction), and compared the results with the ones obtained from a set of human participants. Moreover, in a second paper [4], proposed instead the employment of convolutional neural networks, to be applied over the spectrograms derived from the clips: apparently, this approach outperformed the baseline implementation and, in fact, defined the new state of the art for environmental sound classification.

In the table mentioned earlier, collecting all the recent developments for this specific task and exploiting ESC-50, several different approaches are proposed: many of them are based on speech recognition-like models (MFCC coefficients, [1], [5]), while many others are improved version of the CNN implementations (different representations, architectures, preprocessing phases [6]–[10]). The head of the ranking is entirely covered by the second type of technique, that apparently outperforms the first one from any point of view. Also, many of the methods listed, rely on pretrained models, ad-hoc data augmentation and very complex architectures, mostly realized combining different machine learning prototypes. Actually, only a few approaches, all based on CNNs, are able to overcome the human classification accuracy (81.30%) over ESC-50, with the best model reaching even the 94% of correctly classified samples. Features extraction models, instead, appear much less frequently in the table (people started losing interest versus them) and they can all be found under the humans' accuracy threshold (the best result, 64.30%, was achieved exploiting the combination of a RNN autoencoder and a Multi-Layer Perceptron [5]).

III. Processing Pipeline

In order to provide a complete overview of the approaches that can be followed during this kind of tasks, at a certain point the project will encounter two parallel branches:

- one following the more canonical path of classification according to high-level audio features extracted from the clips (used also in speech recognition tasks);
- a second one instead moving onto an "unconventional" direction and exploiting Convolutional Neural Networks over the audio spectrograms, that, in recent years have, been found to outperform the canonical way for environmental sounds, even replacing it as the state of the art.

Splitting the analysis into two parts, one is effectively able to compare the two methods in term of speed, stability and accuracy reached, but its implementation requires also the generation of two different new datasets.

So, for the first part, a vector of high-level audio features is extracted from the clips, and will represent them during the learning phase for the machine learning models implemented: the main advantage of this method is that each audio file can be simply replaced by a vector with just a few dozen of parameters, resulting in much less memory consumption; so, data are easier to handle and manipulate, at the expenses of some additional preprocessing steps (extraction, dimensionality reduction, regularization). For the second part, instead, the sounds are represented by their power spectrograms, and so the collection of audios leaves space to a set of images, that unfortunately occupy much more memory (obviously depending on their resolution). However this second approach seems to guarantee better performances with very high accuracies, and so it is the preferred one, even if it usually takes much more time to train.

Just as a final remark for this part, it can be useful to say that the spectrograms in the second approach can be replaced by their *Melspectrograms*, that basically are the same power spectra, but represented in the Mel scale, which better approximates the human auditory system; however, in the results section it will be shown that both choices lead to similar performances over the dataset studied.

The whole procedure is possible thanks to the Python library *Librosa* [11], that provides all the building blocks necessary for music and audio analysis.

IV. Signals and Features

As anticipated before, one of the main obstacles in research activities focused on environmental sound classification was the scarcity of large public datasets to analyze and exploit for training models. But it is for this reason that Karol J. Piczak [1], realized a collection of sound clips that became one of the two main datasets used to define the state of the art for this research field.

The **ESC-50 dataset** is a labeled collection of 2000 environmental audio recordings suitable for benchmarking methods of environmental sound classification. The dataset consists of 5-second-long recordings [2] organized into 50 semantical classes (with 40 examples per class) loosely arranged into 5 major categories: animals, natural soundscapes and water, human non-speech, interior/domestic sounds and exterior/urban noises. All of these clips have been manually extracted from public field recordings gathered by the Freesound.org project [3].

As presented in the previous section, one of the peculiarity of this work will be proceeding into two parallel directions, in order to analyze and compare the two main approaches developed and applied so far to solve the problem. For the method exploiting the CNNs, the only preparation required is normalizing the entries of the spectrogram images given in input [4] in $[-1, 1]$, that is a common step when working with neural networks. For the features' vector approach, instead, the biggest part of the work consist into defining an entire preprocessing procedure aimed to the construction of a reliable high-level representation of the clips. Those steps will be deepened in the following subsections.

[2] 44.1 kHz, mono, 431 frames, cropped or padded with zeros to reach exactly 5 seconds of duration

[3] A more thorough description of the dataset is available in the original paper [1], while the main folder of the project can be found on github [3].

[4] (300×600) .png images

A. Features extraction

The objective is to define an high-level representation of each sound clip using just a few values, summarizing all the necessary information stored in each audio file. In particular, for this project, the choice fell on the following list of features:

- **spectral centroid**: indicates at which frequency the energy of a spectrum is centered upon or, in other words, where the ”center of mass” for a sound is located;
- **spectral roll off**: measure of the signal’s shape, represents the frequency at which high frequencies decline to zero;
- **spectral bandwidth**: represents the portion of the spectrum which contains most of the energy of the signal;
- **zero-crossing rate**: rate at which a signal changes from positive to zero to negative or from negative to zero to positive;
- **MFCC**: the Mel-Frequency-Cepstral-Coefficient of a signal are a small set of features (usually about 10/20) which concisely describe the overall shape of a spectral envelope modeling the characteristics of the human voice (according to the Mel scale); only twelve of them are kept (discarding the first one, conventionally) together with their first (**delta**) and second derivatives (**delta-delta**);
- **chromagram**: typically a 12-element features vector indicating how much energy of each pitch class, {C, C#, D, D#, E, ..., B}, is present in the signal (usually, it provides a way to describe a similarities between music pieces);
- **energy**: root-mean-square-energy (RMSE) value for each frame, together with the first (**delta-energy**) and second derivatives (**delta-delta-energy**) of their logarithms.

One could argue that some of them are redundant or even unsuitable for the main objective of the implementation, but later on a Principal Component Analysis will discard all the contribution that effectively are useless, and so computing them at the beginning results in just a negligible increment in the computational time. Moreover, looking at the correlation matrix among those features, and reported in appendix A, one can find additional justifications for this choice.

Notice that the list above is constituted by *frame features*, in the sense that each quantity will contribute with one (or more) values for every frame in the clip (431), and so in the end one will have to deal with entire distributions, rather than single vectors. The first and also the most immediate solution, that one can think about, is representing each feature with its mean across all the frames, leading to vectors of overall size 55. But this will be shown to be a suboptimal attempt because there are many audio files whose features distributions are not very regular neither symmetric, like shown in figure 1, and so the mean alone is no more representing a good estimator for them: to take into account the dispersion of the data, one should rely also on other statistical estimators like the standard deviation. So the final choice fell on constructing, for each clip, a vector containing the mean and the standard deviation across the frames, for all the features listed above, for a total of 110 elements.

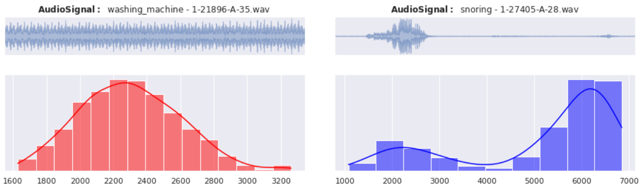

Fig. 1: Spectral centroid distributions for two different sound clips.

B. Silent Elimination

A further preprocessing step that can be implemented is the detection of silent windows in the clips. As can be noticed in picture 2, there are some sounds which have been padded with zeros to reach the five second length, or maybe they just alternate parts of noises and parts of silence. This can be a problem in the final classification, because the features distributions, represented just by their mean and standard deviation, will result strongly shifted towards the zeros measured during those periods of silence, losing, then, much of their regularity. One possibility to deal with this problem could be ”forgetting” about zeros when computing the statistical variables of the features: in this way the mean would result just re-scaled, while the standard deviation would be affected much more. Hence the final accuracy is effectively increased, but the problem is that we are improving the capability of the model when distinguishing among short sounds, but at the same time we are losing information about repetitive sounds, like some animal noises. A possible trade-off between the two situations has been found defining a further step, not on the clips, but directly on the features, according to which if the program is able to identify fix-sized windows of silence, using the energy of the spectrum as a discriminator, those (non-overlapping) windows can be ”forgotten” during the calculation. After several tests, it turned out that the optimal windows’ size was 0.5 seconds.

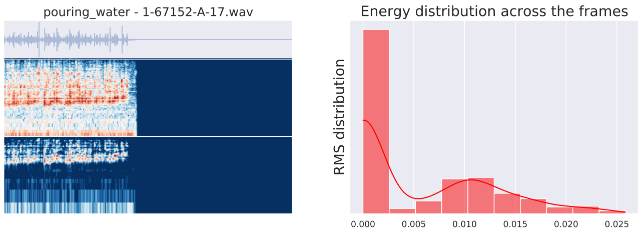

Fig. 2: Example of a clip with a large fraction of silence.

C. Audio Augmentation

The dataset used in this analysis is composed just by 2000 clips for 50 different classes: not enough to train a good classifier nor to construct a reliable statistics of the results. One possible solution to this is data augmentation, i.e. increasing the amount of clips available, adding slightly modified copies of already existing data or newly created ones.

In this way, at least theoretically, the possibility of overfitting will be reduced, making the models able to generalize better. Starting from the original audio files, four new folders, each containing 2000 clips, have been generated, applying one of the available augmentation techniques (inspired by the ones suggested by Salomon and Bello [12], and by the work of Nanni, Maguolo and Paci [13]):

- *Background Noise*: mix the audio sample with some gaussian noise multiplied by a regularization factor, that is calibrated to avoid such noise to overcome too much the original sound.
- *Time Shifting*: sounds should be invariant under time shifting, and so new clips are generated in this way, padding with zeros the part that has been moved.
- *Pitch Shifting*: sound recording technique in which the original pitch of a sound is raised or lowered and, according to Salomon and Bello [12], is particularly beneficial for the augmentation and should deserve a separate analysis.
- *Time Stretching*: just slowing down or speeding up the audio samples.

However, even if the data augmentation step has been reported and implemented, its efficiency will be seriously questioned in the results section, since it will be shown that traditional audio augmentation techniques are not compatible with the approaches followed for environmental sound classification.

V. Learning Framework

Again, this section is divided into two parts, following the two different approaches implemented: the first one exploiting several machine learning classifiers trained with the vectors constructed in section IV-A, and the second one utilizing CNNs to classify spectrograms.

A. Features Classification

There is a "problem" with the ESC-50 dataset: it composed by just 2000 clips, which is never enough to train properly a 50-classes classifier! Also splitting the clips in a train-validation-test sets is not so recommended, because there isn't enough data to construct a significative statistics. A possible solution to all these problems have been found in a technique (suggested also inside the library *sklearn*) called **nested cross-validation**, presented in appendix B.

Then, it's all about figuring out which kind of machine learning models can effectively be trained with success over the clips collected in ESC-50 (or, better, on the features vectors extracted from them). This project started studying the application of four different "canonical" classifiers (provided by *sklearn*) for some reasonable combinations of hyperparameters:

- a **Random Forest**;
- a **Multi-Layer Perceptron**;
- a **K-Neighbors Classifier**;
- a **Support Vector Machine**.

Moreover, before feeding them with the (nested) folds created splitting the dataset, the vectors needed a final preprocessing step, made possible via a data-*pipeline* implementing the following three operations:

1) standardization of the features (via a *StandardScaler*);
2) Principal Component Analysis (whose effects will be studied in appendix D1);
3) label encoding (numerical or one-hot depending on the model).

In addition to the four models listed above, also a feed-forward neural network has been built, with the purpose of having a more customizable architecture and the possibility of monitoring the behavior of the loss across the epochs. Such network is structured in the following way:

- 3 layers with $1024 \rightarrow 512 \rightarrow 50$ nodes;
- *PreLU* as activation function;
- Dropouts and BatchNormalizations;
- *Categorical Cross-Entropy* as loss function.

And now it's just about running the training procedure for several parameters/methods/models and complete all the studies necessary to find the combination that will guarantee the higher accuracy and stability.

B. Spectrograms Classification

One of the main differences between the previous methods and the spectrograms classification is the memory required: in the latter case, in fact, we need to store one image per clip, and an image is much bigger than a vector with just one hundred of samples. We will, usually have to deal with datasets larger than 1 Gb, with the consequent difficulties in loading and manipulating them. For this reason, using a binary file format for data storage can bring to a significant improvement in term of performances, and consequently also on the model's training time. Binary data, in fact, takes up less space on disk, especially when compressed, and can be read in a much more efficient way. *Tensorflow* has developed its own binary storage format, that is the *TFRecord* [5] format, gifted of a lot of preprocessing functionalities and the possibility of loading from the disk and processing, only the data required at that particular time, allowing for an efficient way to deal with large datasets. Unfortunately, this way of proceeding forbid the usage of cross validation, but at the expenses of a small further preparation step (simply generating k tfrecord files splitting the clips by label before loading them) one can still exploit the method to compensate the scarcity of data.

After having determined how to handle such large quantity of images, is time to select the machine learning classifier, and the choice fell on Convolutional Neural Networks, whose high performances has already been proved by several other studies. Apparently, in fact, the possibility of learning local structures from a bidimensional input outperforms the previous approaches, mainly based on the extraction of vectors of features, directly from the raw audio files.

Then it's all about selecting a proper architecture to fit

[5] Tensorflow Records? What they are and how to use them

the spectrograms of the clips. The following choices have been dictated by a long research phase to achieve the best performances in term of accuracy and stability:

- labels are one-hot encoded;
- the loss function is the *Categorical Crossentropy* while the first metric is the *Categorical Accuracy*;
- the optimizer is *Adamax*, that apparently works better than *Adam* for almost any architecture studied;
- the learning rate is quite small, $lr = 0.00005$, that turned out to represent a good tradeoff between training speed, stability and accuracy; an adaptive learning rate is possible, but apparently does not affect very well the stability of the convergence.

The architecture used is represented in figure 3, in which we can distinguish 3 convolutional blocks, characterized by an increasing number of filters and a progressive reduction of the kernel size, in order to simplify the learning of features that will be smaller and smaller as they pass through the network. In fact, at the end of each block there is a *MaxPooling* layer with the purpose of reducing the size of the input images: particularly relevant is the first pooling layer, which strides are set to $(2, 4)$, in such a way that images can be reshaped from a rectangular to a square form. Then, there is a fourth fully-connected block that ends with a layer composed by exactly 50 neurons, corresponding to the 50 possible classes of the images. As activation function the *PReLU* has been selected for all the layers, since it helps the network to prevent vanishing gradients and does not increase too much the time necessary to process the epochs. This architecture has been inspired by several Convolutional models and canonical structures that have been used for this kind of problems, and in particular to AlexNet [14] and VGG16 [15], even if those ones were mainly designed for the dataset ImageNet [16].

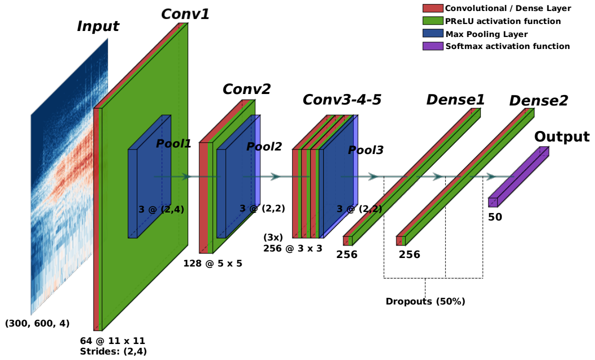

Fig. 3: Architecture of the Convolutional Neural Network.

As last trial to reach a decent accuracy, a more complex architecture has been built merging a feedforward network and the convolutional described above. Its schematic representation is left at figure 4: the idea is using for each clip a double input, made by the features vector and the spectrogram, that will proceed independently one of the other, trough the branches of the network, and the end a unique layer of 50 nodes will predict the class biased by both branches.

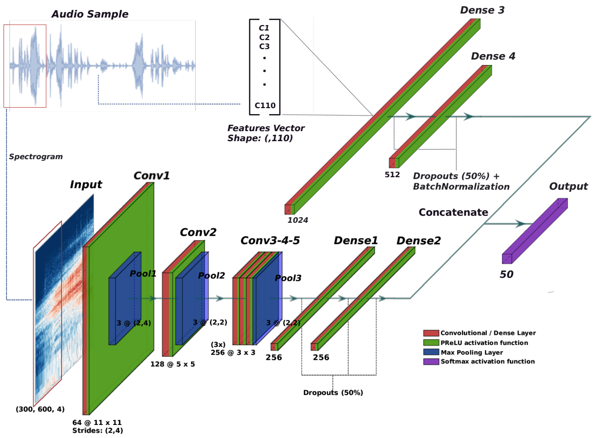

Fig. 4: Architecture of the composite Neural Network (FFN + CNN).

VI. Results

The first series of results that is worth to present are the ones involving the four models implemented in section V-A. All of them were trained exploiting a nested cross validation (appendix B) approach with 5 folds, and, for this reason, the comparison in figure 5 is made via some *violin plots*. The usage of those violin plots seemed the most appropriate choice to represent a *distribution* of accuracy estimates, one for each fold. The results, instead, of the grid search performed over some reasonable combinations of hyperparameters of the models are reported in table 3, in appendix C. So, from this first analysis quite good results have been obtained: getting an average accuracy above the 50% with just 2000 clips, for 50 different classes, and classifiers that are not even complicated, is a nice starting point for this work. Moreover, one can also conclude the top performers, found after several run of the previous analysis, are the Multi-Layer Perceptron and the Support

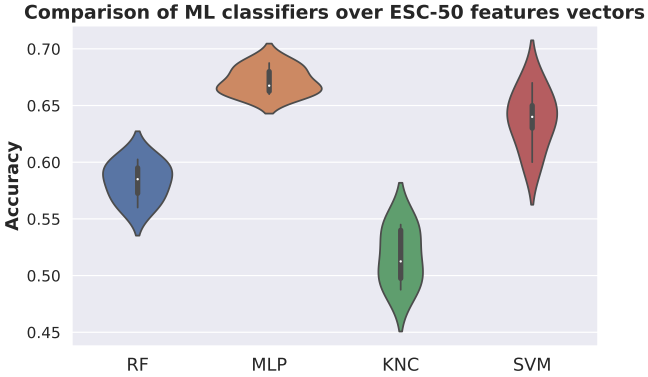

Fig. 5: Comparison of the accuracy reached by different machine learning classifiers.

Vector Machine, with a final accuracy that exceeds the 60%. Random Forests reach intermediate results, but are also by far the model that necessitate of the biggest amount of time to be trained, and this makes them probably the least adapted classifier for this kind of tasks. In the end, there are the K-Neighbor Classifiers, with an accuracy just over the the 50%, but with a much more reasonable training time. For a deeper analysis one could check, for example, directly the confusion matrices of the 4 classifiers, maybe to verify if they behave similarly for different classes, or there is one performing better in a particular field respect to the others and so on. One could also show that, on average, the scores calculated with the nested cross validation are all slightly bigger than the ones obtained with a non-nested approach: this means, as expected, that the overfitting possibility is effectively being reduced, together with the information leakage between the train and validation sets.

To provide, instead, a way to judge the stability of the convergence, one can decide to train from the beginning a neural network, implemented as explained in section V-B, and monitor the loss and the accuracy across the epochs. In this case, since this is just a qualitative test, a nested cross validation is not strictly necessary and one can just split the 2000 clips into a training and validations sets. In figure 6 it can be easily noticed how the model overfits the training set, with the validation loss that reaches a minimum and then starts increasing again: an *early stopping condition* could have been added. Differently from the previous cases, however, this time the validation set is completely made by unseen data, that do not influence the training procedure, and so the final accuracy value will be more credible, at least from a statistical point of view. From the plot it can be noticed that already after some epochs, the training accuracy (and also the error) tends to converge to the ”perfect” classification, while the generalization error, over the validation set (400 clips), converges to an accuracy around the 60/70% (depending on the architecture used). So, since the results obtained are compatible with the ones achieved via nested cross validation, this is a further justification that the approach implemented was statistically corrected.

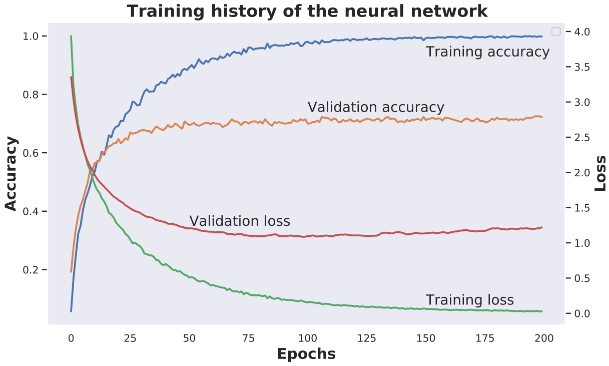

Fig. 6: Training history of a feed-forward neural network over the ESC-50 dataset.

To summarize the results derived up to now, one can refer to table 1, that contains the final accuracy values achieved with the procedure described above, for different models.

Model	Final accuracy
Random Forest	58.30%
Multi-Layer Perceptron	67.15%
K-Neighbors Classifier	51.65%
Support Vector Machine	63.60%
Feed-Forward Neural Network	70.60%

TABLE 1: Accuracy values reached training several machine learning models over the ESC-50 dataset.

The best solution, achieved exploiting cross-validation, foresees the usage of a simple 3-layers neural network, that guarantees a final accuracy of the 70.60%.

To conclude this part of features classification, the only thing that remains to be analyzed is the effectiveness of the audio augmentation implemented. So the cross validation procedure is run again over the augmented dataset, and a new set of final accuracies have been estimated. In figure 7 are plot the differences between the training procedures over the new and the old datasets: surprisingly, augmenting the clips leads to a drastic worsening of the model performances. One reason could be the fact that, summarizing an entire clip with a vector of just 110 values is barely influenced by traditional audio augmentation procedures, maybe because the distributions of some features remain invariant even after the transformation. In this way, one would end up just adding copies of the vectors already present in the original dataset, with the effect of increasing the overfitting. This ”theory” has been checked gradually removing augmentation steps, like keeping only noise + time_shif, pitch_shift + time_stretch and

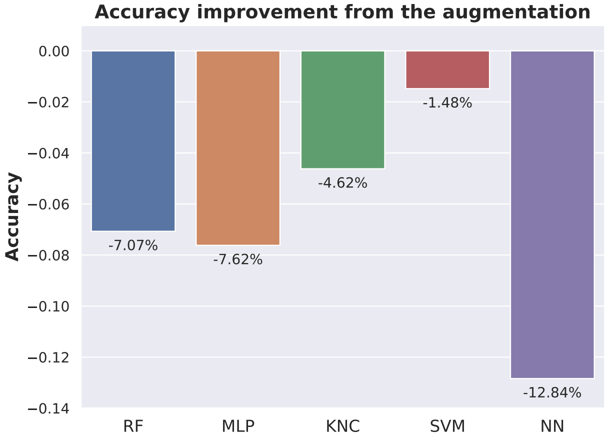

Fig. 7: Effective accuracy improvements after audio augmentation.

so on: in all cases the results are worse than using original data, but it has been noticed that the worsening is reduced when less transformations are considered. This heuristic analysis can be took as an indication that the argument provided is correct and so canonical audio augmentation procedures are not a good idea when combined with features extraction methods (at least in this case).

For what regards the spectrograms classification the situation is similar. Again, data have been split into 5 (stratified) folds and trained over the architecture summarized in figure 3. In figure 8, instead, are reported the trend of the accuracy and the loss for both the training and validation sets as the training proceeds. This behavior have been reported using a lineplot with a shaded area representing the 95% confidence interval for each epoch, in order to summarize the values of the accuracy/loss across different folds. The main purpose of this choice was the possibility of controlling the statistics of the results, in particular if the folds were created nicely and they did not behave too differently, one respect to the others.

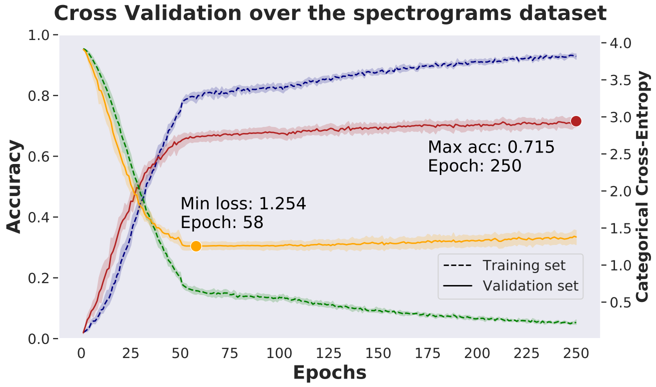

Fig. 8: Training history of a Convolutional Neural Network over the spectrograms extracted from the clips in ESC-50.

The error and accuracy behaviors clearly show that the convergence is effectively reached after some epochs, since the validation error starts fluctuating and rising again already after $\sim$ 60 epochs, and so an early stopping condition could be reasonable. The model chosen obviously overfit the training sets: both the error and the accuracy tend to an optimal classification condition after the 250 epochs, in a stable way, in the sense that the confidence interval is smaller and smaller as the training proceeds. The overall validation accuracy, instead, converges to a value around the 70%, with a similar profile than the one encountered during feature classification (figure 6). In the end, since the interval sizes in the plot are more or less quite regular, one can also be satisfied about the credibility of the statistics of the results, and about the procedure used to create the various folds.

Other alternative plots that can be extracted from the clips, such as the raw signals and the Melspectrograms, can be used as well as inputs to Convolutional Networks. However:

- the usage of Melspectrograms brings to similar results (final accuracy $\sim 68\%$);
- raw signals underperform with those kind of architectures (final accuracy $\sim$ 35%); for them a one-dimensional approach, like the one proposed by Huang and Leanos [9], should be preferable, because it allows to reach surprisingly high accuracies.

Regarding the audio augmentation phase, implemented in section IV-C and proved to be unsatisfactory for the features vectors approach (figure 7), it can be re-evaluated working with convolutional networks: exploiting the same augmented dataset, this time it is possible to derive an effective, even if small, improvement in the final accuracy($+0.5\%/+1\%$). However this comes at the expenses of a dataset that is five times larger than the original one (many Gb of data) and so a much longer computational time, even if, to reach the convergence, fewer epochs are needed.

But no one should forget that those networks work with spectrogram plots, and no more with audio files, and so another kind of data augmentation, that should be kept into account, are traditional image augmentation techniques:

- *Brightness adjustment*;
- *Contrast adjustment*;
- *Random uniform noise addition*;
- *Random left-right flip*.

All the operations listed above where applied randomly to the clips in the ESC-50 dataset, with reasonable parameters, making it five times bigger. Then, a cross validation using the same network as before, was run over the augmented dataset: the results are quite nice, since the network is able now to achieve the 75% average score across the folds, proving definitely that image augmentation outperforms audio augmentation techniques for environmental sounds.

In the end, the last configuration tested is the composite neural network, presented in figure 4, constructed merging a double input of spectrograms and features vectors: in this way, in fact, considering two independent fluxes of information across the layers, one is able to quantify how much information is effectively carried by both of them, and how much is instead shared. The network has been tested with and without augmentation, providing a tiny improvement respect to the CNN alone, reaching anyway the $75\%/78\%$ of accuracy (with some peaks even overcoming the 80% threshold, for some folds).
The final results achieved following this second approach, and exploiting convolutional neural networks, are collected in table 2

In figure 9, instead, all the different methods studied and tested so far, are put into comparison.

So, to summarize all the results that we have achieved:

- features vectors methods are much faster and less memory demanding with respect to the others, even if they

Model	Final accuracy
CNN	71.55%
CNN + audio augmentation	71.68%
CNN + image augmentation	75.32%
FFN-CNN	74.75%
FFN-CNN + image augmentation	78.10%

TABLE 2: Accuracy values reached with Convolutional networks over the spectrograms of ESC-50.

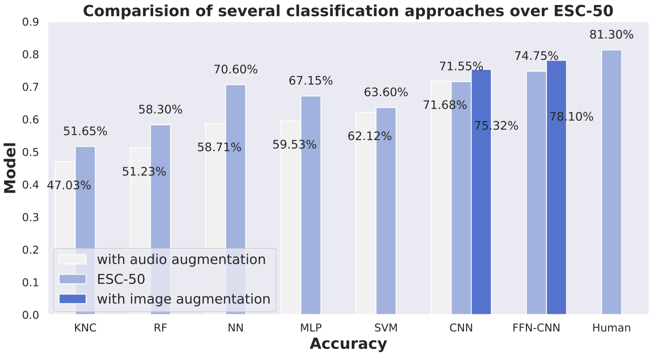

Fig. 9: Training history of a Convolutional Neural Network over the spectrograms extracted from the clips in ESC-50.

are quite limited in term of the classification accuracy that can be achieved;

- thanks to the preprocessing pipeline, and the nested cross validation implemented, we effectively managed to overcome the highest accuracy value reported in Piczak's Github page [3] not exploiting Convolutional networks (FFN 70.60% $>$ 64.30 RNN + MLP [5]);
- we didn't manage to overcome the human classification accuracy (81.30%) even exploiting CNN; however, we definitely proved that traditional image augmentation techniques bring to better results than augmenting directly the audio clips.

VII. Concluding Remarks

The goal of this project was to study and implement the main approaches developed so far in the field of environmental sound classification, especially considering the limited nature of dataset available for our purposes. So, given the ESC-50 dataset, we have presented the main techniques designed and implemented with the objective of classifying those clips. Even if we didn't managed to reach the very high accuracies of several professional works, we can be quite satisfied of several results obtained and summarized in the last paragraph of the previous section.

Further improvements could be simply deepening one of the models analyzed, maybe trying with more complex architectures, different preprocessing steps or even testing pre-trained models. However, actually the biggest problem is the absence of large datasets, since 2000 clips is not enough to construct a credible statistics for generic applications, even exploiting nested cross validation. In any case, the strong interest of the scientific community in speech recognition tasks, and the consequent improvements in sound classification, will for sure benefit also the branch of environmental sounds.

Regarding myself, instead, I can say that I practiced a lot with several Python libraries dedicated to machine learning (*SKLearn, Keras, Tensorflow*), especially for what involves data management and preprocessing. Moreover, this was my first project exploiting sound data, and so I learned a lot about audio features and track manipulation.

The most challenging part of this project was, for sure, determining a way to speed up the training processes (that could take up to many hours for each model), keeping, at the same time, a low memory usage, and ending up with a reliable statistics for the results.

References

[1] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018, ACM Press.

[2] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, (New York, NY, USA), p. 1041â€“1044, Association for Computing Machinery, 2014.

[3] karolpiczak, "Esc-50: Dataset for environmental sound classification," 2015.

[4] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2015.

[5] M. Freitag, S. Amiriparian, S. Pugachevskiy, N. Cummins, and B. W. Schuller, "audeep: Unsupervised learning of representations from audio with deep recurrent neural networks," *CoRR*, vol. abs/1712.04382, 2017.

[6] M. Huzaifah, "Comparison of time-frequency representations for environmental sound classification using convolutional neural networks," *CoRR*, vol. abs/1706.07156, 2017.

[7] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," *CoRR*, vol. abs/1610.09001, 2016.

[8] V. Boddapati, A. Petef, J. Rasmusson, and L. Lundberg, "Classifying environmental sounds using image recognition networks," *Procedia Computer Science*, vol. 112, pp. 2048–2056, 2017.

[9] J. J. Huang and J. J. A. Leanos, "Aclnet: efficient end-to-end audio classification CNN," *CoRR*, vol. abs/1811.06669, 2018.

[10] S. Li, Y. Yao, J. Hu, G. Liu, X. Yao, and J. Hu, "An ensemble stacked convolutional neural network model for environmental event sound recognition," *Applied Sciences*, vol. 8, no. 7, 2018.

[11] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[12] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *CoRR*, vol. abs/1608.04363, 2016.

[13] L. Nanni, G. Maguolo, and M. Paci, "Data augmentation approaches for improving animal audio classification," *CoRR*, vol. abs/1912.07756, 2019.

[14] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F.-F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014.

[17] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie, “Accelerated dual descent for network optimization,” in *American Control Conference*, (San Francisco, CA, US), June 2011.

[18] Y. Tokozume and T. Harada, “Learning environmental sounds with end-to-end convolutional neural network,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2721–2725, 2017.

[19] L. Nanni, G. Maguolo, and M. Paci, “Data augmentation approaches for improving animal audio classification,” *CoRR*, vol. abs/1912.07756, 2019.

[20] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “Specaugment: A simple data augmentation method for automatic speech recognition,” *Interspeech 2019*, Sep 2019.

Appendix

A. Correlation matrix

In the large figure 10 is represented the correlation matrix among the 55 different high-level audio features (or, better, distributions) that have been extracted from each clip of ESC-50, according to the procedure explained in section IV-A.
Analyzing this correlation matrix we can make several interesting observations about the information that has been effectively extracted from the audio samples:

- as expected, variables that are already ”grouped”, like MFCCs, chroma features, etc. manifest some kind of dependence among each other (especially the variables in the chromagram, that appear to be strongly correlated);
- spectral features and zero-crossing rate, even if computed separately, as ”single” features, exhibit some correlations; the reason is that they simply all come from the analysis of the same spectrogram;
- spectral features and MFCCs seems somehow anti-correlated;
- energy and corresponding derivatives are less likely to reveal any kind of dependence with any other variable;
- an interesting correlation appears between the MFCCs and the chroma features;
- just to precise, many of the anti-correlations involving the deltas that can be evinced from the plot are originated simply by the fact that such features are directly calculated from the others and so the dependence is obvious.

So, to conclude, many of the features extracted exhibit some kind of correlations/anti-correlations among each other, and for this reason they probably share part of the information carried. Further proof of this conclusion can be found in appendix D1, when talking about Principal Component Analysis.

B. Nested Cross Validation

A non-nested approach consists in using the same cross-validation procedure and data both to tune and select a model, but this is likely to lead to an optimistically biased evaluation of the model performance (because of information leakage). Nested Cross-Validation (Nested-CV) nests cross-validation and hyperparameter tuning exploiting two different KFold (or stratified KFold) splitting, such that in the inner loop the score is approximately maximized by fitting a model to each training set, and then directly maximized in selecting (hyper)parameters over the validation set; in the outer loop, instead, the generalization error is estimated by averaging test set scores over several dataset splits. Under this procedure, hyperparameter search does not have the opportunity of overfitting the dataset as it is only exposed to a subset of it, provided by the outer cross-validation. This reduces, if not eliminates, the risk overfitting and should provide a less biased estimate of a tuned model's performance on the dataset. Obviously, this does not come without any additional cost, since you dramatically increase the number of intermediate steps: if $n * k$ models are fit and evaluated as part of a traditional non-nested CV for a given model, then this number is increased to $k * n * k$ as the procedure is then performed k more times for each fold in the outer loop of nested CV.

C. Results of the GridSearch

In this section we leave a brief recap of the combinations of hyperparameters, that we have found thanks to a nested cross validation applied to ESC-50, to guarantee the highest level in term of accuracy over the dataset.

Model	**Best hyperparameters**
Random Forest	• *bootstrap*: True; • *max_depth*: 15; • *n_estimators*: 500.
Multi-Layer Perceptron	• *activation*: relu; • *hidden_layer_sizes*: 512; • *learning_rate_init*: 0.01; • *solver*: adam.
K-Neighbors Classifier	• *algorithm*: auto; • *leaf_size*: 10; • *n_neighbors*: 2; • *weigths*: distance;
Support Vector Machine	• *C*: 0.1; • *kernel*: linear.
Feed-Forward Network	• *batch_size*: 128; • *epochs*: 150; • *lr*: 0.001; • *optimizer*: adamax.

TABLE 3: Summary of the best combinations of hyperparameters for the features classifiers, found thanks to a nested CV.

D. Further Analysis

1) Impact of the dimensionality reduction: Working with large sized vectors of features can be very memory and time demanding, especially when training a complex classifier, maybe with a large number of parameters. For this reason one usually implement, before passing the data to the algorithm, a dimensionality reduction step, that in this case is a *Principal Component Analysis*, with the purpose of reducing the dimension of each input vector, while keeping the ”information” provided by it as untouched as possible. But how much this step will influence the performances of the classifiers? Let's first give a look at figure 11, that allows to

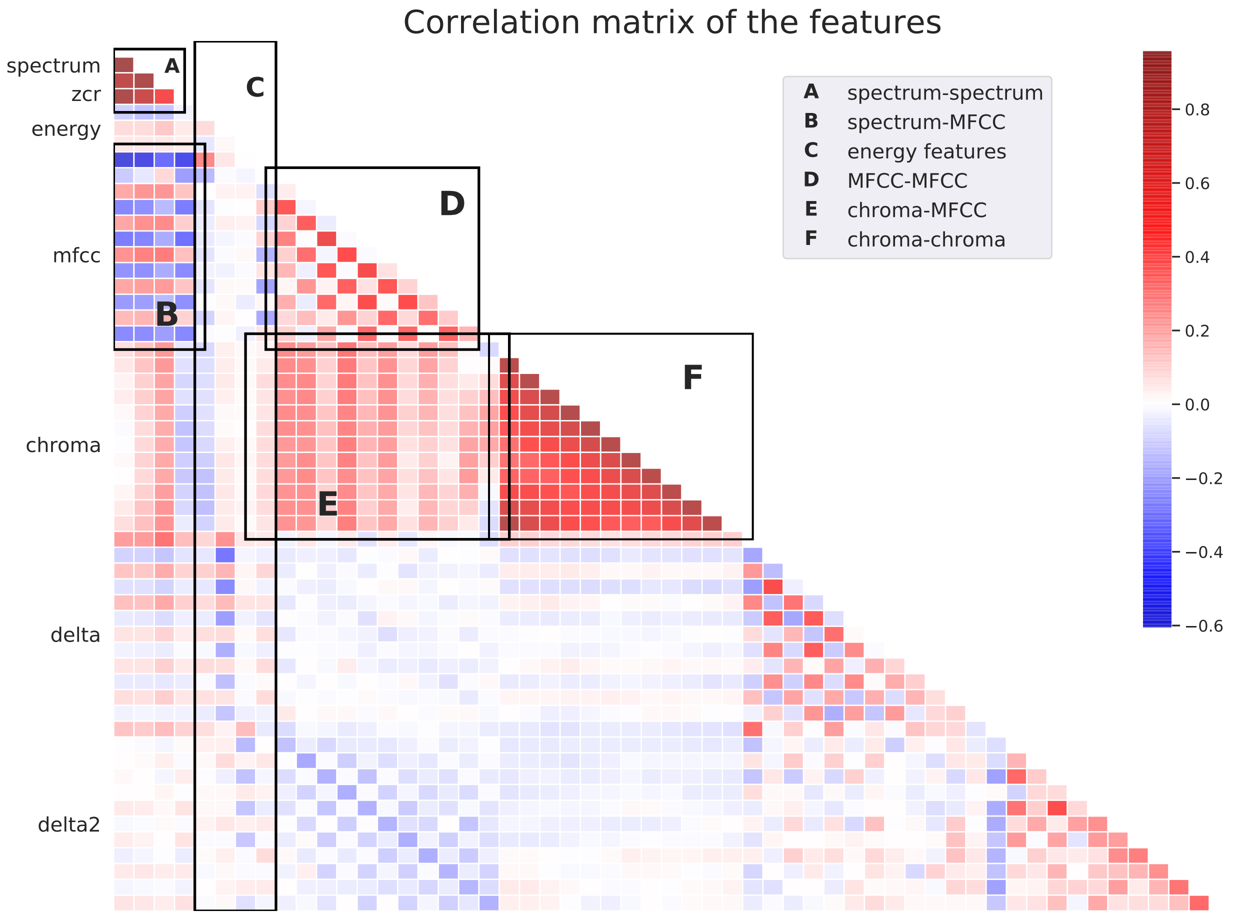

Fig. 10: Correlation matrix of the 55 high-level audio features extracted to represent the clips contained in ESC-50, as explained in section IV-A.

determine the number of principal components to keep without losing too much of that "information". Basically, during the PCA, you are projecting your data into a smaller dimensional vector space, and each principal component corresponds to an eigenvalue of the covariance matrix of the dataset: reducing the size of the problem means keeping only the first K eigenvalues, i.e. the ones with the higher variance. In the plot it is represented the *cumulative explained variance* as a function of the number of components kept: the closer to 1 is the value, the more will be the information represented. Looking at the line, one clearly see that, potentially, the 90% of the information stored in the features can be reduced to just 48 values! The analysis presented in section VI have been computed keeping, by default, the 99% of the explained variance, i.e. 84 principal components. And the results are still quite nice. What would happen, instead, further reducing the number of eigenvalues to keep? The results shown in figure 12 looks like exactly what we expected:

- in the left plot there are the mean accuracies obtained for the models trained: already after 40/50 principal components the lines start becoming flat, meaning that no significant further improvements are possible. And this is coherent with the explained variance ratio presented before (figure 11). Recalling section IV-A, the vectors aimed to provide an high-level representation of the clips were created using an high number of features: probably so many (55) different quantities were unnecessary, and only half of them were sufficient. However, properly tuning the number of principal components, the problem

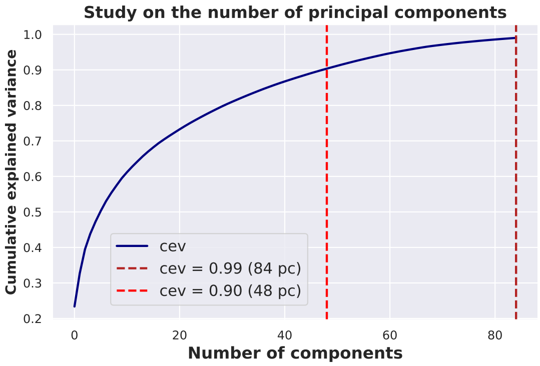

Fig. 11: Cumulative explained variance ratio of the Principal Component Analysis applied to the features dataset.

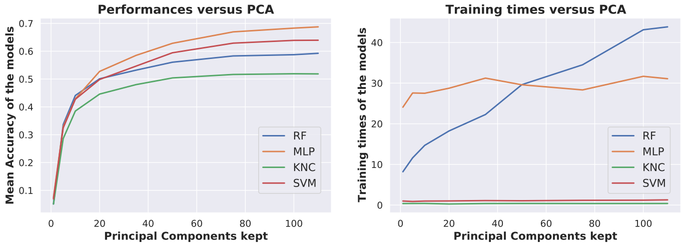

Fig. 12: Comparison of mean accuracy reached and training times of the models for different numbers of principal components kept.

vanishes.

- the right plot, instead, shows the computational times: the bigger are the input vectors, the larger will be the time necessary to make the classifier learn from them. With just 2000 clips the absolute difference is not so relevant, but the scaling behaviors of the lines, instead, are evident.

2) Importance of the statistical estimators for the features distributions: In section IV-A it has been explained that features distributions across frames were "summarized" using also their standard deviation, together with their mean, in order to properly represent also their irregularity and asymmetry. In figure 13 it is shown the effective improvement, in term of accuracy, achieved training the four machine learning classifiers using just the mean or both the mean and the standard deviation. The improvement is obvious: using both the mean and the standard deviation as statistics estimators provides almost a $+20\%$ to the accuracy obtained with all the models used. As expected, in fact, the distributions are, in general, quite asymmetric and non-gaussian, and taking just the average is probably an excess of reductionism.

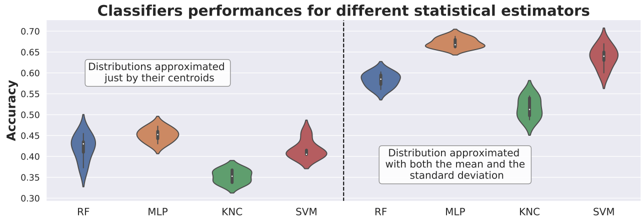

Fig. 13: Comparison of the accuracy reached by the classifiers with one or two statistical estimators representing features distributions.

3) Importance of the silence removal step: The last step of the preprocessing phase explained in section IV-B was the silence removal, i.e. the identification of fixed-size windows of zero energy, and their remotion from the successive calculations. In figure 14 are shown the differences, in term of accuracy, obtained with and without this windowing procedure. And this is not even bad: getting a $+1\%/+5\%$ improvement in accuracy is nice, at the expenses of a small further preprocessing step.

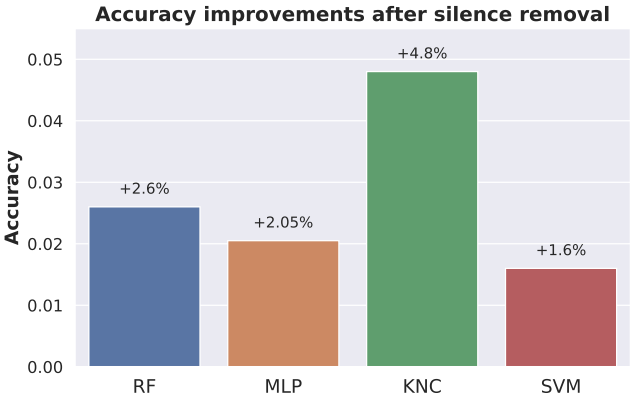

Fig. 14: Effective improvements in accuracy reached training the models with and without the silence removal phase.