"Environmental Sound Classification" a convolutional approach applied to audio signals

Ian Solagna, Michele Zanatta

***Abstract*—Environmental Sound Classification (ESC) is becoming more and more popular finding multiple applications in the fields of surveillance systems and noise cancelling. Previous studies has shown how the log-mel features extracted from the audio clips can be analysed alongside the raw waveforms to achieve good results. In this paper, we propose a different preprocessing of the features as well as a new approach to the training of the Neural Networks in the sound classification framework. We then use different Convolutional Neural Networks to analyse both the mel-spectograms (logmel CNN, AUG logmel CNN) and the raw data (RAW CNN). Finally, the results are combined using Dempster Shafer (DS) theory to reach similar results to those of humans in the ESC-50 dataset: a dataframe containing examples from 50 different categories of sounds. Overall, this paper is intended to serve as a valuable resource for researchers and practitioners working in the field of environmental sound classification: we believe that our approach to the feature extraction and training can be applied to any other architecture to improve the performance and generalise better.**

***Index Terms*—Supervised Learning, Optimization, Neural Networks, Machine Learning, Data Processing, Convolutional Neural Networks.**

I. Introduction

Environmental sound classification is a rapidly growing area of research that aims to identify and classify specific sounds such as a *Crying Baby*, *Car Horn* or *Rain*.

Many different audio features like the Mel Spectogram, the Mel frequency cepstral coefficients have been widely used by researchers ([1], [2]) to tackle this problem. However, given the low number of datasets in this type of framework the work has usually been conducted on segments of sounds extracted from the audio clips ([3], [4]).

In this paper we present some ideas to extract the features and train the Deep Learning models, specifically:

- Instead of computing the Mel Spectogram on the segments we use the full clip.
- We augment the training data with new examples by adding random time delays and pitch shift.
- We develop a new training approach to tackle a sound classification problem that can help with overfitting and can be applied to any neural network architecture.

Our work is structured as follows: in Section II we present the work of other researchers that guided our analysis. Next, in Section III we give a high level introduction to our work, while in Section IV we present the feature extraction process. Then, in V we describe in detail the neural networks architectures and finally in VI we present the results we obtained.

II. Related Work

In [1], *Piczak* presents, for the first time ever, the *ESC-50* dataset along with some Machine Learning models such as K-Nearest Neighbors, Support Vector Machine and Random Forest. Furthermore, it is shown that the Mel-Spectrogram (in particular the Mel-Frequency Cepstral Coefficients) and the Zero-Crossing rate of an audio clip are, in fact, able to capture different patterns of a given sound. In our project we replicate this work just to have a baseline accuracy and then focus on the more advanced Deep Learning techniques.

Convolutional Neural Networks (CNNs) were indeed proven to be quite effective when dealing with audio recognition tasks such as speech ([5]) and music analysis ([6]). This is because the log-mel feature of audio signals has locality in both the time and the frequency, thus it can be treated as an image and given in input to some convolutional layers for an efficient classification.

Piczak in [2] introduces a CNN for environmental sound classification that is able to outperform the simpler models by almost 20% accuracy. This study was based on the idea that learning on the mel-spectograms of the full clips was too limiting. Because of this, data was augmented by adding random time delays and dividing every single five-second clip into multiple overlapping frames of one second. Each frame was then provided along with its delta (local estimate of the derivative of the spectrogram) to the networks as a 2-channel input. We show that using the full mel-spectrogram is not limiting at all and the use of full clips on a deeper but less complex network architecture achieves the same results. Then, we augment the data with pitch shifts and the already mentioned time delays and deltas (although we added the second order one) to increase the accuracy even more.

Tokozume et al. ([4]) showed that it is not necessary to manually extract the log-mel features from the data but that this can be automatically done by a convolutional layer applied on the raw waveform. By doing this, the study achieved better results than those reached by *Piczak* proving the robustness of this kind of approach. In our project we also build a similar network but we use a different sample strategy that we strongly believe can help with overfitting and data augmentation.

Finally, in [3], *Li et al.* use Dempster-Shafer evidence theory to build an ensemble model constituted by two different neural networks. We apply the same procedure to our nets showing that this is a powerful technique capable of improving the predictive power of the models in this type of framework.

III. Processing Pipeline

Data was acquired through the publicly available *ESC-50* ([1]) dataset, which is a collection of 2000 short (5 seconds) environmental recordings comprising 50 equally balanced classes of sound events in 5 major groups (animals, natural soundscapes and water sounds, human non-speech sounds, interior/domestic sounds, and exterior/urban noises), prearranged into 5 folds for comparable cross-validation.

The data then was pre-processed in different ways depending on the Machine and Deep Learning model to be used. These methods are described in detail in Section IV, and they include the computation of the log-scaled-mel-spectogram, of the Mel-Frequency-Cepstral-Coefficients (MFCC) and of the spectogram's first and second derivative (deltas). Moreover, in one case data augmentation was applied, by adding a random time delay and applying a slight pitch-shift.

At this point, for ML models the cross validation technique was applied, while for DL architectures the dataset was split into train, validation and test sets (60-20-20).

Thus, training with the learning framework of choice was carried out to obtain a multi-label classifier which output describe the class the sound belongs to.

Results were analyzed through the classification accuracy and the computational cost of each model.

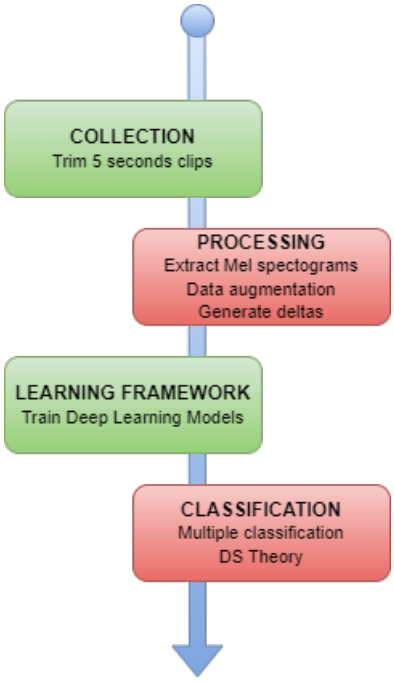

Fig. 1: Processing pipeline.

IV. Signals and Features

Every single audio in the dataset contains a 5 second-long clip belonging to one of the 50 different categories. All samples were also converted to a unified format: 44.1 kHz, single channel, Ogg Vorbis compression at 192 kbit/s (fig. 2).

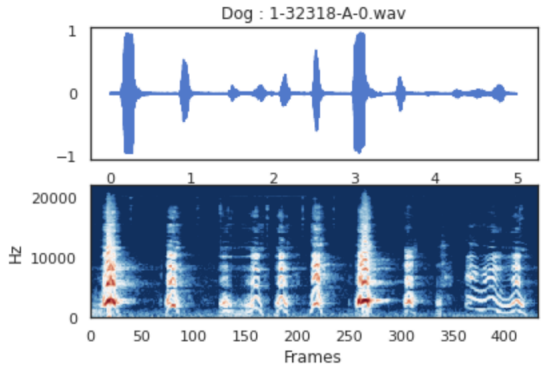

Fig. 2: Raw data and log-mel spectogram representation of a dog barking.

However, since we propose different models and Deep Learning architectures, the features were extracted in multiple ways and pre-processed differently according to the method used.

A. Machine Learning Techniques

As already mentioned in Section II , the procedure we applied is the same as *Piczak* in [1] so we will not focus too much on it:

- Each clip was sampled at 44100 Hz and the log-scaled mel-spectogram was computed with a window size of 1024, hop length of 512 and 128 mel-bands.
- MFCC (excluding the 0th coefficient which is not very informative) and zero crossing rate were also calculated. The sizes of these features are respectively (431×12) and (431×1).
- Given the high number of parameters the features were summarized with their mean and standard error resulting in 26 total features. This was done because the models chosen are not very efficient in high dimension and are not able to exploit the 2D structure of the data.

The clips in this format were then given in input to four different machine learning techniques: K-Nearest Neighbors, Support Vector Machine, Gradient Boosting and Random Forest. Now we move on with the feature extraction for the Neural Network architectures.

B. Logmel CNN

This is the first and simplest CNN that we have tested, therefore the feature extraction is minimal:

- The clips were resampled at 22050 Hz and normalized, resulting in a shorter but still very informative raw data vector of size (110250×1).
- Log-scaled mel-spectogram was extracted from the different recordings with window size of 1024, hop length of 512 and 60 mel-bands. The matrix obtained has size (60×216) corresponding to (frequency $\times$ time).
- We reshape the matrix to a tensor of size $(60 \times 216 \times 1)$ adding a new dimension corresponding to the channel.

The features extracted this way were given in input to the logmel CNN.

C. Aug logmel CNN

For the validation and test set the setup is identical to the one of the *logmel CNN* with a little variation:

- The clips were resampled at 22050 Hz and normalized.
- Log-scaled mel-spectogram was extracted from the different recordings with window size of 1024, hop length of 512 and 60 mel-bands.
- We reshape the matrix to a tensor of size $(60 \times 216 \times 1)$ adding a new dimension corresponding to the channel.
- For each spectogram, we calculate the first temporal derivative and the second temporal derivative (deltas) and use them as second and third channel of the input obtaining a tensor of size $(60 \times 216 \times 3)$.

On the other hand, since the number of examples is very limited, we decided to augment the training set with new examples. This way, we believe that this model will be able to generalize better and achieve better results than the one presented in *subsection B*. The procedure is the following:

- The clips were resampled at 22050 Hz and normalized.
- For each recording we create two new training samples by adding a random time delay (between 0 and 1.4 seconds at the beginning of the clip) and truncating the new example to 5 seconds.
- On these new samples we apply a slight pitch-shift by randomly raising it, lowering it or keeping it the same.
- Then we calculate the log-scaled mel-spectrogram and the corresponding deltas as for the validation and test set (fig. 3).

Because of this processing we now have at our disposal three times the training examples we had before and also the new delta features to help us classify the environmental sounds.

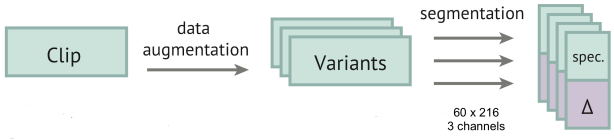

Fig. 3: Augmentation procedure applied to the training set.

D. Raw CNN

For the final neural network architecture we try to use the vector of raw data as an input to the neural network. In this case the feature extraction is even more diversified and is unique across the training, validation and test set. Let's start with the validation set:

- As for the other methods, the clips were resampled at 22050 Hz and normalized.
- Each single clip was split in 50% overlapping segments with a time window of 2 seconds. This results in 5 different segments for each clip of size (44000, 1).

For the test set the procedure is similar:

- The clips were resampled at 22050 Hz and normalized.
- Each single clip was split 75% overlapping segments with a time window of 2 seconds.
- We decide to drop silent frames, i.e those segments whose maximum amplitude is no bigger than 0.25. This comes from the fact that for different classes silence sounds basically the same. Therefore, it's easier for the model to recognise the relevant frames.

This results, on average, in more segments than the validation set (between 1 and 7 for every recording). This is because, to classify each single clip we will apply a probability voting to predict the category of the full original audio. therefore, taking more segments results in a more robust voting in the end.

As for the training set we decide to apply a strategy that, to our knowledge, was not applied by other researchers:

- The clips were resampled at 22050 Hz and normalized.
- Instead of creating a static training set that is the same for the whole training we decide to create a dynamic one that changes after a number n of epochs.
- To do so, we create a function that for each single clip picks a random segment of 2 seconds from it and puts it in the training set.

We believe that this technique can help in reducing overfitting and learn more from the clips in input.

E. Fuse CNN

Lastly, for our ensemble model we don't need to create new features, we simply take the predictions of the *Aug logmel CNN*, the predictions of the probability voting obtained from the *Raw CNN* and fuse them obtaining the final probabilities.

V. Learning Framework

In this section we are going to describe how the learning process is carried out by the already mentioned different neural networks, which are represented in Figure (4).

For each CNN, we chose the architecture based on the type of data we wanted to feed into (Section IV). The hyperparameters, such as the number of filters, the dropout and the batch size, were tuned using the validation set. Here we report only the finals architectures.

Notice that for every CNN implemented we used:

- a 50 units output layer with *softmax* as activation function, which gives us the desired output in the form of a probability distribution;
- the *categorical cross-entropy* loss function to properly match the multiple classification nature of the task;
- the *Adam* optimization algorithm to train the model.

A. Logmel CNN

The *Logmel CNN* is composed by three 2-dimensional convolutional layers and an hidden dense layer. The convolutional layers have filter size of 3×3 and a different numbers of filters: 64 the first two, 128 the last one.

After each convolutional layer we added:

- a *max pool* layer to preserve local features;

- a *batch normalization* layer in order to reduce the effects of vanishing gradient.

Moreover, three *dropout* layers have been added to improve the generalization ability of the model and reduce overfitting.

We trained the network for 60 *epochs*, using a *batch size* of 50 samples.

B. Aug logmel CNN

The *Aug logmel CNN* has the same architecture as the previous network, but we used different data to train it, adding two channels given by the spectograms' first and second temporal derivative.

For this reason, even if the structure remains the same, we trained the model for 70 *epochs* to obtain relevant results.

C. Raw CNN

For the *Raw CNN*, as we have already said, we used the vector of raw data as input to the neural network. Thereby, we can divide the structure into two parts, where the first part aims to extract relevant features from the raw vector, while the second one is focused on mapping the input vector to the corresponding class, through the relevant features given by the first part.

1) Raw Extracting Features: First, we apply two time-convolutional layers with a small filter size to the input raw waveform in order to extract local features. Each convolutional layer has 40 filters, which is the same as the typical dimension of log-mel features. The filter size is 8 in both layers, and we stride the filter by 1. We apply non-overlapping max pooling to the output of the convolutional layers with a pooling size of 275, which corresponds to 15 ms. A *batch normalization* layer is added after each convolutional layer.

Each 40-dimensional vector can be thought of as representing frequency-like features of the corresponding 15 ms area, because of time convolution and pooling. Since we apply a convolutional layer in the direction of the components in the next step, we assume that the order of the components of the 40-dimensional vector will be optimized to maximize the classification performance. In this manner, the output of this section has a locality, and we can treat it as an image ([4]). We reshape the output to obtain a 40×160 image.

2) Processing on Feature Map: Next, we apply five convolutional layers and one fully connected layer to classify the feature-map, treating it as an image. The first two convolutional layers have 24 filters with a size of 6×6 and a stride of 1×1. After the first layer we apply a we apply non-overlapping max pooling to the output with a pooling size of 3×3.

These layers are followed by two convolutional layers with 48 filters of size 5×5 and a 2×2 stride. The last convolutional layer has 64 filters with size 4×4 and stride 2×2.

We apply a global average pooling and we feed the output into a 200 units fully connected layer, with a 0.5 dropout.

We train the model for 200 *epochs* with a batch size of 20, selecting a different set of 2-seconds clips every 20 epochs as training set, as we have already explained in Section (IV).

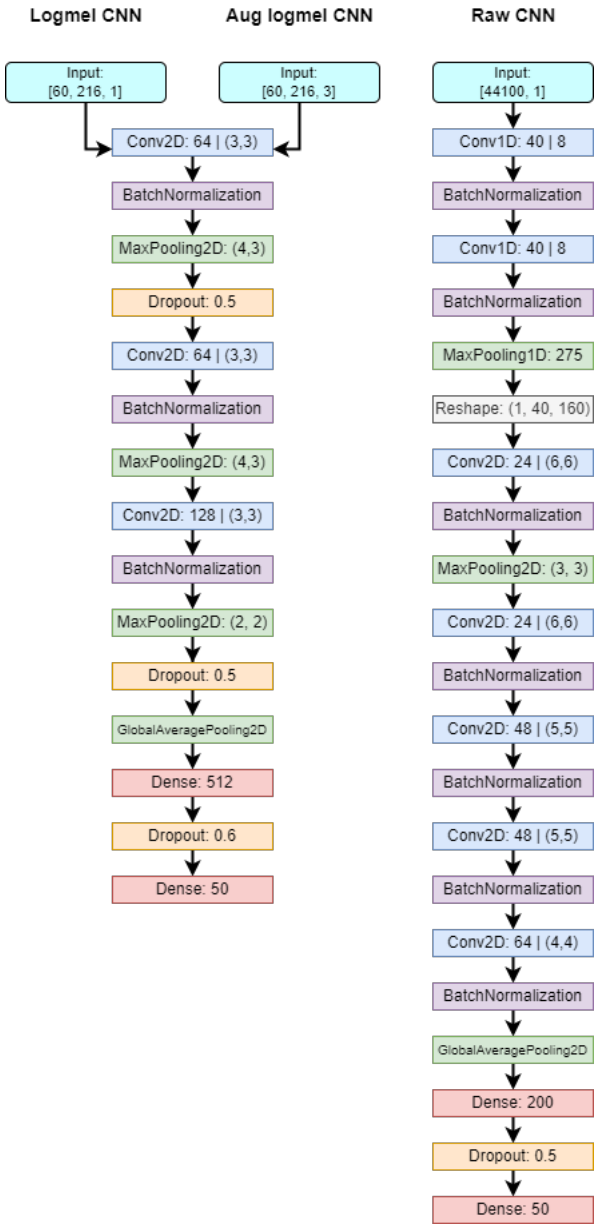

Fig. 4: Neural Networks Architectures.

D. Fuse CNN

Finally, we describe how we combined the *Aug logmel CNN* and *Raw CNN* predictions for improving the accuracy using the DS evidence theory. Here, each class of sounds in the dataset can be regarded as one element in $\Theta = \{A_1, A_2, \ldots, A_{50}\}$. All elements are exclusive and independent. Meanwhile, the output values of the activation functions (softmax) of the *Aug logmel net* and the *Raw net* models are used as the basic probability assignments m_1 and m_2, respectively, under the same frame of discernment, Θ.

Then, we use an orthogonal operation to effectively synthesize the basic probability assignments of m_1 and m_2 generated by these two models. For any $A \in \Theta$, the fusing formula is

as follows:

$$(m_1 \oplus m_2)(A) = \frac{1}{1-K} \sum_{A_i \cap A_j = A \neq \varnothing} m_1(A_i) \cdot m_2(A_j),$$

where $K = \sum_{A_i \cap A_j = \varnothing} m_1(A_i) \cdot m_2(A_j)$ is a measure of the amount of conflict between the two mass sets.

We use $(m_1 \oplus m_2)(A)$ as the final prediction result of the fusion.

VI. Results

Due to the high number of classes we will not be able to report the confusion matrices. Instead we will focus on the different accuracies reached by the different models and for the best one we will go more in depth looking at the results for some categories of sound. The results are summarised in *table* 1.

Model	Accuracy
Human	81.3%
K-NN	30.6%
SVM	37.0%
XGBoost	29.7%
Random Forest	42.3%
Logmel CNN	62.7%
AUG Logmel CNN	72.25%
RAW CNN	62.0%
FUSE CNN	75.0%

TABLE 1: Accuracy reached by the models

From the table we can see that the dataset was also tested on humans ([1]) reaching an accuracy of 81.3%. It may seem low but the classification task is far from being easy: some of the sounds are very similar, in fact the class on which humans struggled the most was *Engine*, which was confused with *Washing Machine* and *Wind*. Same for *Helicopter* sounds which were confused with a *Washing Machine* almost 20% of the time.

From this point of view the results obtained by the Machine Learning techniques are not bad. The best one is *Random Forest* which is able to reach an accuracy of 42.3% that is already a great improvement if compared to the dummy classifier (i.e a random classifier that on average has 2% accuracy). Note that KNN, SVM, XGBoost and Random Forest were tested with a *Cross-Validation* regime, thus the results are robust. On the other hand, the Deep Learning models, due to their higher complexity (see Section VI-A) were trained using a *Hold Out* approach.

The first network we tested was the *Logmel CNN* and even with a very simple architecture and preprocessing of the data it was able to outperform the *Random Forest* classifier by 20%. This proves that the convolutional layers are, in fact, able to extract the most important features from the spectograms and use them to improve the classification. Playing with a different number of layers did not improve the classification, more layers resulted in too much overfitting, while less layers led to a significant drop in accuracy. On the other hand, the kernel size did not change much, so we just picked the combination that got the highest accuracy on the validation set.

As we already mentioned, we did not have much data at our disposal, so if a not-so-complex architecture was able to reach such high numbers in 50-class problem made us believe that adding some examples could really help the training of the model. This assumption was right and the *AUG logmel CNN* achieved 72.25% accuracy, an improvement of almost 10%. Here we decided to not test different hyperparameters because we simply added training examples and we wanted a direct and fair comparison with the *Logmel CNN*.

For the model using raw data from the audio clips, the structure is way more complex and, at first, we were not able to go past 50% accuracy: we tried adding more layers, more filters and stronger regularization, but with no significant improvement. Then we decided to apply the strategy presented in Section IV-D. This helped a lot with overfitting and forced the neural net to learn the most significant features of the data. The final result was 62% which proves that this kind of approach is powerful and not to be neglected.

Finally, using *Dempster-Shafer Theory* we fused the probabilities obtained by the *AUG logmel CNN* and *RAW CNN* to obtain an accuracy of 75.0%. This result indicates that the two approaches complement each other. Taking a look at the outputs of the model we can see that the *FUSE CNN* really struggles with some classes while, for the others is often a very good classifier (34 of the out of 50 categories have at least 70% accuracy, 15 of these have 100%). In particular, the model performed very poorly when classifying *Helicopter*, *Fireworks* and *Water Drops*, which where mainly confused with, respectively, *Airplane*, *Clapping* and *Can hopening*.

A. Complexity

An important aspect of Machine Learning that is often overlooked is the complexity of the models used in the analysis: in particular, here, we will focus on the training time needed to reach the results presented in the paper. Regarding KNN, SVM, XGBoost and Random Forest we calculated the time elapsed on a single fold, this was done to have a fair comparison with the Deep Learning techniques that are only trained with a Hold Out approach. Notice that the neural networks were trained on high end GPUs made available by *Google Colab Pro*. The results are summarised in *fig.* 5:

- The four baseline classifiers take only a bunch of seconds to compile, which is very good, since the results they obtain are not trivial.
- *Logmel CNN* takes almost a minute to compile, and is able to significantly outperform the previous models.
- *AUG Logmel CNN* is the same structure as the previous network, but the number of samples tripled. As a consequence the training time also tripled, but this led to a 10% improvement in accuracy.

- *RAW CNN* is the one that has the most parameters and because of this it takes the longest time. It takes more than 30 minutes to train it and the results are not bad but are worse than the spectogram-based models.
- *FUSE CNN* needs no training, however, to be able to use it we need *AUG Logmel CNN* and *RAW CNN*, so we just took the sum of the time elapsed in those models.

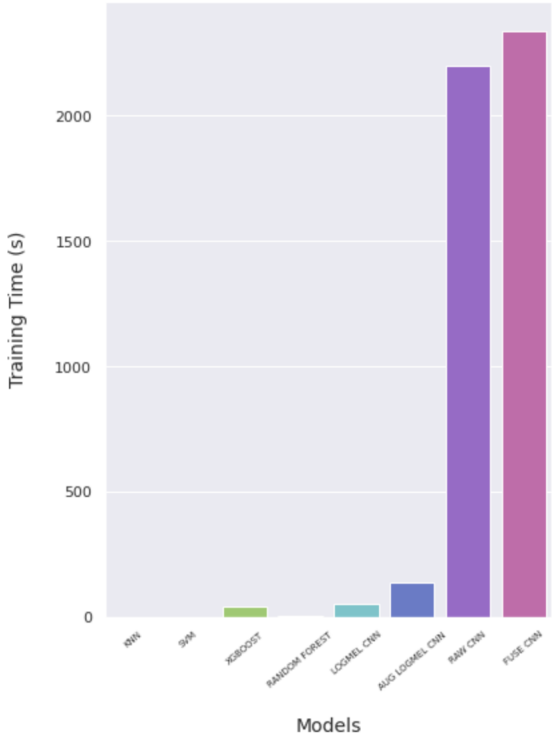

Fig. 5: Time in seconds needed to train each model

In this framework, one may think that the best model is always the one that gives the highest accuracy. However, sometimes the hardware we have at our disposal is not advanced and some compromises are needed. Taking this into account the best model we presented is for sure the *FUSE CNN*, but the *AUG logmel CNN* reaches almost the same accuracy but using much less resources.

VII. Concluding Remarks

In this paper we have dealt with *Environmental Sound Classification* reaching some good results that almost match those of humans. The task was not easy: audio files need a lot of preproccesing and the number of classes in the *ESC-50* dataset makes it a challenging multiclass problem. We developed two main Neural Network architectures that complement each other: *AUG logmel CNN* that works on the Mel Spectogram of the audio clips and *RAW CNN* that functions on the raw data. We then fused their outputs to obtain the final predictions. However, what we focused on the most was the feature engineering and the approach to the training of the networks. In fact, we used the features in a different way compared to similar work and we also developed a new framework to use when analysing audio files. We believe that our work can help other researchers: augmenting the data with pitch shift and the second order derivative of the Spectogram as well as using a "dynamic" training set that changes over time can be applied to more complex architectures that, because of hardware constraints, we could not test.

To conclude, we present some of our considerations regarding this project:

- It was the first time we worked on audio files, it was interesting but at the same time challenging when we first approached the dataset.
- This work made us realize how actually important is feature extraction and especially regularization when working with Neural Networks.
- This project taught us that the work of others can help when developing your own framework. Every reference paper was interesting and added something to our work.

References

[1] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1015–1018, 2015.

[2] 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, IEEE, 2015.

[3] 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, p. 1152, 2018.

[4] 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, IEEE, 2017.

[5] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[6] S. Dieleman, P. Brakel, and B. Schrauwen, "Audio-based music classification with a pretrained convolutional network," in *12th International Society for Music Information Retrieval Conference (ISMIR-2011)*, pp. 669–674, University of Miami, 2011.