Date of publication 21 August 2021, date of current version 20 August 2021.

Improvement of alarm and video surveillance systems by incorporating audio analysis

Alberto Menendez Ruiz de Azua[1], Álvaro López Lopez[2], and Lucia Güitta Lopez[2]

[1]Universidad Pontificia Comillas, Escuela Técnica Superior de Ingeniería – ICAI, C/ Alberto Aguilera, 23 - 28015 Madrid

[2]Instituto de Investigación Tecnológica Calle de Santa Cruz de Marcenado, 26 - 28015 Madrid, España

Corresponding author: Alberto Menendez Ruiz de Azua (e-mail: albemene97@gmail.com).

ABSTRACT The study behind this paper stems from a project that seeks to create a reliable model that allows current video surveillance systems to use audio classification to complement their visual capability, i.e. to allow the cameras to "listen". The project succeeded in creating, training and implementing various artificial intelligence models for audio detection and classification in the context of security and video surveillance. State-of-the-art results have been achieved and a study has been carried out comparing different architectures and types of models in order to select those that best fit the specific challenges and problematic of the original project.

INDEX TERMS Spectrogram, Artificial Intelligence, Neural Networks, Transfer Learning, Audio Classification, Security, Video Surveillance

I. INTRODUCTION

This paper and the project behind it arose as a response to a limitation in the security industry, more specifically in the alarm and video surveillance sector. The need for a solution is based on the need to offer a better service to customers in this sector and to increase the reliability and efficiency of the systems that currently exist.

The central idea, and the response to it, came from an exhaustive analysis by experts in the security sector of the cases of failure of their systems and the limitations they usually encounter. After a study of the cases in which these systems fail or do not achieve adequate efficiency, it was determined that a fraction of the thefts or incidents that were omitted by video surveillance could have been detected and, consequently, avoided if an analysis of the audio at the specific moment of the incident had simply been carried out. This analysis would be complementary to the CCTV video and in no way a substitute for it.

Therefore, the main purpose was to satisfy this need and to develop an audio classification method that works in a complementary way to the current security cameras, providing them with a capacity that they have lacked until now and allowing them to improve their efficiency and give them the capability to “listen”.

Finally, it was also decided not to limit the project to the exclusive field of security and to take advantage of the enormous possibilities that audio classification offers in terms of comfort for the end user of the security system. These possibilities are limited only by the classes in which the audio is classified, as there are classes such as a child crying or a fire crackling, which can alert the user to incidents that do not necessarily imply a burglary, but make the system completer and more attractive for the end consumer.

The paper will cover the study and comparison of the performance of different audio classification state-of-the-art algorithms for the proposed problem. Metrics such as inference speed, accuracy and scalability will be measured and compared among the different architectures in order to make a final decision and choose one of them to be pushed into production phase for the solving of the security problem.

II. STATE OF THE ART

The most pioneering techniques in the state of the art regarding audio classification are mostly *machine learning (ML)* technologies and more specifically the ones in the field of *deep learning* and neural networks. [1] and [2] represent the current state of the art in audio classification, achieving the best KPIs in the GITZAN, ESC50 and

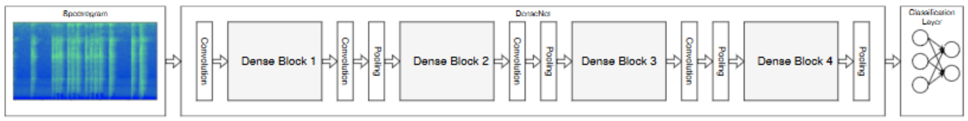


Figure 1: Classic architecture for an Audio Classification Deep Learning Model

UrbanSound8K datasets. From the state-of-the-art study another important point was detected, this was the fact that working with audio signals is complicated and that is why most of the state-of-the-art solutions are based on the transformation of audio waves into images and the subsequent application of state-of-the-art image classification techniques, usually *Deep Learning*, and the corresponding state-of-the-art architectures, such as *inception*, *vgg19* or *Alexnet* models. This idea represents the central point of the two above mentioned methods and is the one used throughout this project.

CONVOLUTIONAL NEURAL NETWORKS

Convolutional networks are one of the most robust and consolidated technologies in the field of machine learning and represent the state of the art in solving image classification or segmentation problems. The central idea of this type of network was created by the French computer scientist Yann Lecun and has given rise to all kinds of architectures (such as *Resnet*, *Vgg*, *Alexnet*...) and a wide variety of applications in many different fields, including audio classification.

The main characteristic of this type of networks is that the neurons are formed by "matrices" or kernels and the weights that are trained using the backpropagation algorithm are the values taken by the cells of these structures. Several kernels of the same dimension are grouped together to form a convolutional layer, which is accompanied by a stage known as pooling in order to compress the output of these layers, which will constitute the input for the next convolutional stage and so on. Several of these combinations of convolution plus pooling give rise to the hidden layers of a convolutional neural network. In its most traditional variant, and for most of the applications for which these types of networks are typically used, these hidden layers are followed by a combination of layers corresponding to a dense neural network that allows for the classification of the output of the convolutional layers [8].

TECHNIQUES FOR MODEL TRAINING

DROPOUT

Dropout is a regularization technique for neural networks developed in Geoffrey Hinton's lab in 2012 [6]. It consists of randomly deactivating certain connections or neurons in the hidden layers of the network. This modulates the approximation power of the network, which effectively prevents it from over-training (i.e., the famous over fitting).

BATCHNORAMALIZATION

Batch-normalization is a technique that consists of adding an extra stage between the neurons and the activation function, with the aim of normalizing the output activation. This regularization technique is a training aid. This technique can be combined with the idea of a moment, so that the mean and standard deviation used to normalize a sample are not very different from those used to normalize the previous sample [13].

EARLY STOPPING

This technique was used during the training process to avoid over-fitting. It consists of establishing a threshold of epochs for some of the KPIs of the training (test accuracy, test loss...) so that, if this KPI has not improved its value for more than the established epochs, the training is stopped and the model for the 'last epoch that this KPI improved' is chosen as the final model.

CLASS WEIGHT

This technique is used to reduce the possible effect on our model of having strongly unbalanced data across classes, which is exactly the case in this project. The main idea of class weight is to give weights to the predictions of the classes, so that errors in those classes that have less data in the training set are penalized more (proportionally) than errors in those classes that have more data.

TRANSFER LEARNING

Tranfer Learning is a *Machine Learning* technique, which uses a model which was pre-trained and a applied for a task as starting point for another model developed for a different task.

This technique allows the usage of pre-trained convolutional hidden layers trained by the *Google* researches on big datasets and with really powerful hardware to create the *vggish* model [5]. These layers are used in the convolutional part shown in Figure 1 allowing to leverage all the knowledge that this network has acquire while being trained for a different audio classification problem.

MEL SPECTROGRAM

The Mel Spectrogram is a way of representing an audio signal in a more visual way than the usual amplitude approaches. It uses Fourier Transformations and Logarithmic Scales to plot Time, Frequency and Amplitude in a two-dimensional scale and visual colors [1].

III. MODEL ARQUITECTURES

For this paper two different main models with two different ways to approach the problem were compared and then several architectures where built for these models to test and compare. Bothe models are bases on the standard architecture for Audio Classification problems shown in Figure 1, but with several differences in the approach to its usage.

MODEL 2DCNN

This model was entirely built from scratch and every single layer of it was coded for this project. This means higher control and possibilities to experiment, but lower scalability. Inside this approach three main architectures were considered after doing some tests: 2DCNN_1, 2DCNN_2 and 2DCNN_3, with the following characteristics:

- 2DCNN_1 and 2DCNN_3 only differ in the type and size of *kernels* that they use for the convolutional steps.
- 2DCNN_2 is a variation of the 2DCNN_1 architecture which applies *Dropout* to avoid overfitting.

MODEL ENCODER + CLASSIFIER (E+C)

This model leveraged on *Transfer Learning* from a very powerful *Deep Leaning* model for audio classification. This other model is *vggish* [5]. A classifier was then trained on top of *vggish* 128 output vectors to adapt it to the security problem. For this approach only, control for the classifier architecture was obtained. The used of *transfer learning* resulted in a more powerful and scalable model, but allowed less room for innovation and experimentation in the model, as well as higher training and inference times. For this model two different solutions were tried in the classifier side: Classifier and Classifier_Dropout.

The Classifier is a fully connected network with a *softmax* output layer that allows to perform classification on the outputs from the *vggish* network. The Classifier_Dropout network has the same architecture as the Classifier, but applying *Dropout* in the fully connected layers.

IV. PRE-PROCESSING

A big important part of the project was the pre-processing and changes made to data as a result of the different feedback gathered from some of the first training processes of the different architectures. The two main phases of this step were the following:

CREATION OF THE SPECTROGRAMS

In order to be fed in to the models the audio signals needed first to be processed and turned into 2D images which could actually be processed by the image classification models. The size and some features of the images came fixed by the use of the pre-trained *vggish* model, which had already been trained with 960 ms long audio frames, a jump of 10 ms for each Fourier transformation and with 64 mels to split the frequency in. This yield to the images having a size of 96x64 frames and it was decided to keep it like that for all models in order to make them comparable.

The characteristics of the transformation process were the following:

- Sampling frequency: 44100 Hz
- Audio duration: 960 ms (set by the sizes by the pre-trained Vggish architecture)
- Fourier transform window duration: 25ms
- Moving window jump for the Fourier transform: 10 ms
- Number of mels to separate the frequency space into: 64

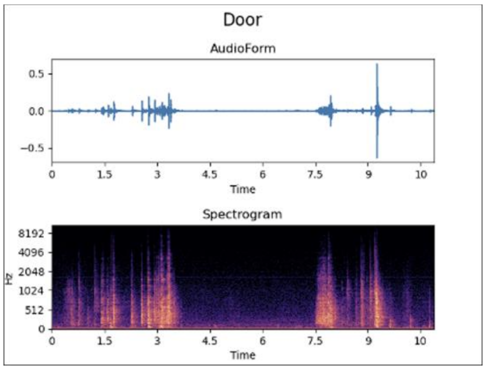


Figure 2: Audio spectrogram of a gate showing the lack of information for any sample whose 960 ms span the range from 3.5 to 7.5 seconds (own elaboration).

CLEANING OF CLASSES

The class cleaning carried out in the project was a continuous feed backwards process from the model trainings that lead to better data quality for the next training processes.

The main cleaning was a clustering approach carried out on the different samples from the classes to detect those samples that were not correctly labeled because of the way that the audio files where being processed before feeding them in to the models. The problem can be shown in Figure 2 where some of the 960 ms samples extracted from the audio between the 3.5 and 7 seconds will not have a characteristic part of their labeling class (Door).

To clean them a clustering algorithm was performed on the output vector of the *vggish* computing Euclidean distances from every class vector to other classes representative vectors. An example of how this algorithm was used can be seen in Figure 3, where the samples that correspond to Silences instead of Doors or Steps are identified.

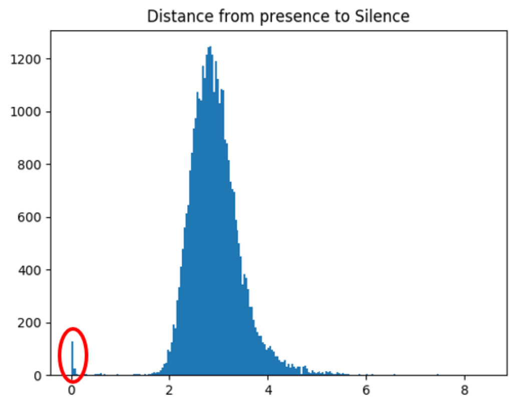


Figure 3: Euclidean distance of the vectors of the classes Steps and Doors to an example vector of the class silence.

V. TRAINING AND EXPERIMENTS

All the five architectures discussed in III were trained and compared in the original datasets and the following datasets that arose from the different data cleaning processes mentioned in IV.

The training process was carried out using the *Google Colab* cloud processing tool, as it allows the free use of GPUs, which enables faster training of any architecture that requires matrix computations such as those of the full 2D convolutional network of the first methodology. For the pre-trained encoder architecture, we opted to transform all samples into their corresponding characteristic vectors first and use these vectors to train the dense neural network directly. This saved a lot of time in the training section, which allowed for more iterations and faster model improvement.

For both methods, regularisation techniques were used such as *Dropout*, or *Batchnormalization* for the 2DCNN method and *Early Stopping* for both. Class weighting was also used in both solutions to deal with the problems arising from having an unbalanced dataset.

All training was done using the Adam optimiser with a learning rate of 0.001 and a categorical cross-entropy as a loss function for the model. The accuracy and loss curves of the training process can be seen in XII.

The architectures were trained and compared using their accuracy KPIs and their confusion matrices until there were only two of them left with the best KPIs, the 2DCNN_1 model and the Classifier model. Then a more extensive analysis was carried out between these two models to be able to determine which of them fit better for the security audio classification problem. For this comparison the following metrics were compared: accuracy, precision, recall, specificity in the different classes, inference and training times and scalability and overall performance.

VI. RESULTS

Accuracy, Precision, Recall and Specificity

Table 1 shows the final general results of the final two candidate models. It captures both the accuracy and loss for training and validation sets after the early stopping had been applied.

Method	**Accuracy (%)**	**Loss**	**Val acc (%)**	**Val Loss**
2DCNN_3	93,86	0,124	89,83	93,86
Classifier	92,77	0,132	91,75	0.251

Table 1: Table for the general final comparison

This results where computed from the confusion matrices of the models in the test sets of their definitive trainings, this confusion matrices can be seen in Table 2 and Table 3.

Classes	**Accuracy**		**Precision**	
	2DCNN	E + C	2DCNN	E + C
Alarms	99.10%	99.58%	67.65%	80.95%
Glass	98.99%	99.07%	58.89%	50.70%
Cry	100.00%	99.50%	100.00%	94.60%
Dog	99.15%	97.35%	95.18%	86.23%
Fire	100.00%	99.87%	100.00%	91.38%
Others	97.83%	98.94%	94.14%	96.82%
Presences	96.02%	98.52%	81.69%	93.97%
Shotgun	98.74%	98.91%	84.72%	86.10%
Total (macro average)	98.73%	98.97%	85.28%	85.09%
Total (micro average)	89.83%	91.73%	89.83%	91.73%

Table 2: Table for accuracy and precision results

These results were computed because depending on the class a different KPI will be more relevant to decide which of the models performs better. These results will allow for a better analysis and decision making in section VII.

Classes	Recall		Specificity	
	2DCNN	E + C	2DCNN	E+C
Alarms	98.57%	95.77%	99.08%	99.57%
Glass	84.13%	81.82%	98.97%	99.06%
Cry	100.00%	97.37%	100.00%	99.45%
Dog	89.34%	92.33%	98.98%	96.82%
Fire	100.00%	100.00%	100.00%	99.87%
Others	91.30%	88.45%	96.68%	98.43%
Presences	82.53%	94.38%	95.17%	98.07%
Shotgun	89.79%	89.75%	98.64%	98.83%
Total (macro average)	91.96%	92.48%	98.44%	98.76%
Total (micro average)	89.83%	91.73%		

Table 3: Table for recall and specificity results

Inference times

The below table shows the times that it took for each of the final models to process an audio file of the different classes. This inference was carried on the same hardware (a GPU in the *Google Colab* online tool) and the same conditions for both models, measuring the time since an audio file is received until the prediction for that file is obtained. The results can be seen in Table 4.

	Samples (#)	2DCNN		E+Cs	
		Time (s)	T/sample (s)	Time (s)	T/sample (s)
Cry	13	0.2778	0.0213	0.6028	0.0463
Dog	9	0.0456	0.0035	0.6973	0.0774
Door	21	0.0921	0.0043	0.5014	0.0238
Fire	1	0.0171	0.0171	0.2299	0.2299
Glass	5	0.1102	0.0220	0.3497	0.0699
Gun	2	0.0103	0.0051	0.2756	0.1378
Others	7	0.0418	0.0059	0.2946	0.0420
Steps	59	0.2056	0.0034	0.9035	0.0153
Avg	117	0.0068		0.0329	

Table 4: Results for inference time comparison

It must also be noticed that the variation in time per sample shown in the tablel is not proportional to the number of samples since the *GPU* allows for parallelization of the calculations.

VII. ANALYSIS OF RESULTS

For the results' analysis the different KPIs were studied and compared among the models depending on the project requirements and keeping always in mind the main objective which was selecting the model that has a better performance for the security audio classification problem.

Accuracy, Precision, Recall and Specificity

Both of the final models have great results and there is no one of the two that outperforms the other in all aspects. The final conclusion of the accuracy analysis is that the E + C model should be the one chosen if only the KPIs of the models are considered, since it is the one that best fits the original and main objective of the project, which is to detect possible theft situations. The Presence class is of vital importance in this aspect and such a big difference in the recall of this class makes it the best option to choose based on the requirements and the final objective of the project. The E+C model's recall outperforms the 2DCNN model 94.38% to 82.53% in this aspect.

It must be noted that for the security classes, such as Presence, Dog or Shotgun the recall is a better KPI, since it shows what percentages of the incidences or thefts have not been detected by the model. Whether for the comfort classes such as Cry or Alarms the precision KPI is a bit better since it shows what percentage of the times that the model detected an occurrence it was right and therefore we are not bothering the final user for no reason.

To sum up this part, even though both models are pretty close in both recall and precision, the difference in Recall in the Presence class in favour of the E+C model made it the chosen model in this category.

Scalability and flexibility

Since the number and type of classes may grow in the future, depending on industry or customer needs, it is necessary that the solution be as scalable as possible. This is more easily achieved with the Encoder + Classifier model, as the 2DCNN model is a more ad hoc solution and would saturate more quickly with an increase in the number of classes. Whereas the vgg model, on which the E+C architecture is based, is designed to be able to classify thousands of classes with ease.

The difference in scalability and flexibility was noted during the entire process, where a change in the number and type of classes was needed throughout the development of the models and this implementation were easier with the E+C model which required no major changes, while the 2DCNN was a bit more of a challenge to get ready for the new requirements.

Inference and training times

From the time results shown in Table 4 it can be concluded that the 2DCNN model is 5 times faster than the E+C model. This was pretty much expected, since the number of parameters and steps in the E+C model is greater than in the 2DCNN model.

The real important analysis in this section is the comparison of this times against the real-world situation were the model will be implemented. Each analyzed sample corresponds to 960 ms of life audio sequences, however each of these samples only contains 480 ms of new audio, since it overlaps half of the previous audio sample and is does not correspond to “unheard” audio for the model. Therefore, it can be concluded that the maximum time that the model may take per sample to be implemented as a real time solution has to be less or equal than 0.48 seconds, which is achieved by both models for their worst time per sample performance (these times are shown in red in Table 4.

For the outcome of the project the really important time is the inference time, since it will be the one that will facilitate a better subsequent production of the model. Having good time and flexibility in training is relevant if you want to have a flexible and quickly adaptable model, however, inference is more determinant in the production environment.

Chosen model

Finally, a decision was made based on the results analysis. The chosen model was the E+C model, since it achieved better KPIs for the problem posed and greater flexibility in case an increase or change in the number and type of output classes is needed. As for the inference time, what is really important in this analysis is the study of these times against the real world where the solution is going to be implemented and although the 2DCNN was faster, speed was no limitation for the implementation of the model.

VIII. FINAL IMPLEMENTATION

Together with the study of the architectures and their results for the proposed problem, a ROC sensitivity study was carried out, as this is a project in which a threshold to reduce the True Positive versus False Positive KPIs is really interesting. This is due to the context in which the project is framed, since in the classes that will later represent a theft, we are interested in not missing any True Positive when the model is active, although this may lead to an increase in False Positives. The philosophy of "Prevention is better than cure" would apply.

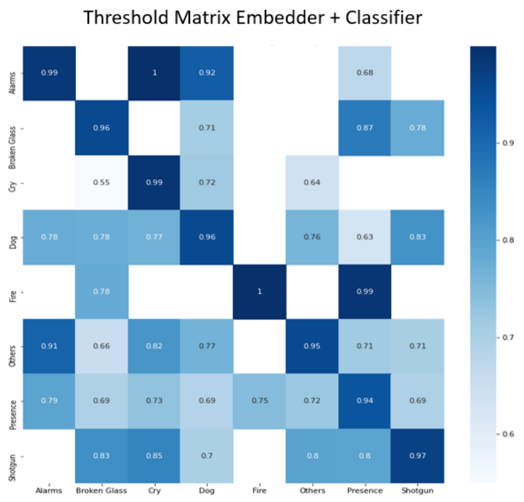


Figure 4: Matrix of average confidence levels of the model for each of the classes

The matrix in Figure 4 shows how the average confidence level in the model's predictions for True Positives (main diagonal) is much higher than for False Positives. This implies that simply adding a threshold as a parameter to the final implementation of the model would give much better results. This threshold can be adjusted depending on the class, but a general acceptable level would be around 90%.

IX. CONCLUSION

The project resulted in a successful and reliable model for audio classification along with a comparison of the two main state-of-the-art solutions for audio classification. It allowed for a pioneering solution for the current video-surveillance technologies and the security sector.

At the time of writing, security industry stakeholders have started to integrate the model into their systems and have conducted their own tests with the model with very satisfactory results. It should be noted that all the requirements requested were met and the scope of the project was extended to include the comfort variable as an added value for the system's end customers.

X. FUTURE WORK

MODEL IMPLEMENTATION

This section will be carried out by security sector stakeholders. The model will work in a complementary way to security cameras by providing them with new capabilities.

STUDY OF ADDITION OF TEMPORTAL VARIABLE

The current model perfectly fulfils the function of recognising audio fragments; however, it would be interesting to study the addition of models that, on top of the current model, are able to understand the context of that fragment. These models could be Transformers-style or NLP models. In the current project this is achieved by creating an overlap between samples, so that each sample is made up of a percentage of the end of the previous sample, a percentage of the beginning of the next sample and a part that does not necessarily belong to either sample.

An example of an application that uses the technique of this project with an extra layer to be able to analyse the temporal variable of the audios can be seen in [17], where the vggish model combined with a BiGru recurrent network has been used to capture this temporality.

HYPERPARAMETER ANALYSIS

Another interesting idea IS a more extensive study of the different variants and possibilities. Although a brief study has been carried out, changing both architectures, and variants have been tested that are not shown in the work because they have given worse results than the chosen ones; it would be possible to go deeper into this aspect and seek to optimise these values for the proposed objective. It is particularly interesting with regard to the 2DCNN model, since of the two, it is the one with the greatest margin for change, having been completely developed for this project.

DYNAMIC TRAINING

This idea is quite common in machine learning applications whose results are easily contrasted against human capability for the same problem is the use of a tool to be able to train the model automatically. With this, the whole process of pre-processing and manual labelling that was done with the data could be done dynamically. The idea would be that for those samples we could hear the most conflicting or mislabelled audios and decide if they belong to another class or if they can be discarded for the project.

XI. References

[1] A. Guzhov, F. Raue, J. Hees and A. Dengel, "Esresnet: Environmental sound classification based on visual domain models," 2020.

[2] K. Palanisamy, D. Singhania and A. Yao, "Rethinking CNN Models for Audio Classification," 2020.

[3] S. H. e. at, "CNN architectures for large-scale audio classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

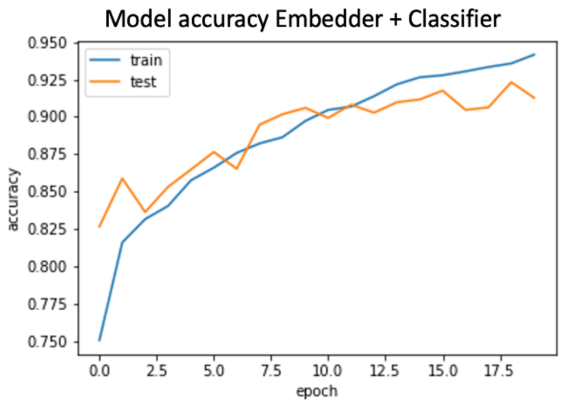
[4] D. Grtzman, "Getting to know the Mel Spectrogram," *Towards Data Science,* August 2019.

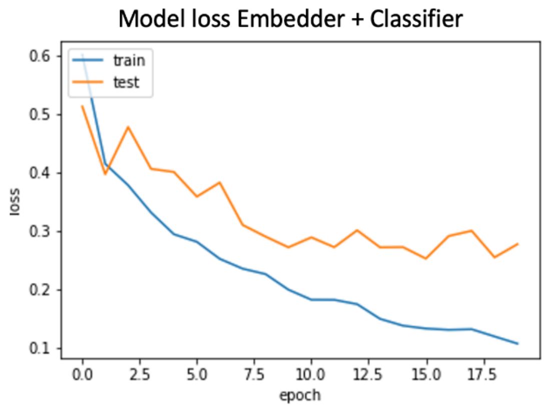
[5] M. Mishra, "Convolutional Neural Networks, Explained," *Towards Data Science*, 26 August 2020.

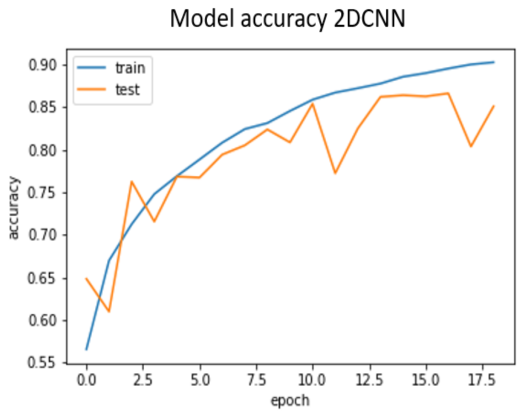
[6] A. Agrawal, "The Current State of the Art in Natural Language Processing (NLP)," *Towards Data Science*, 6 Junio 2020.

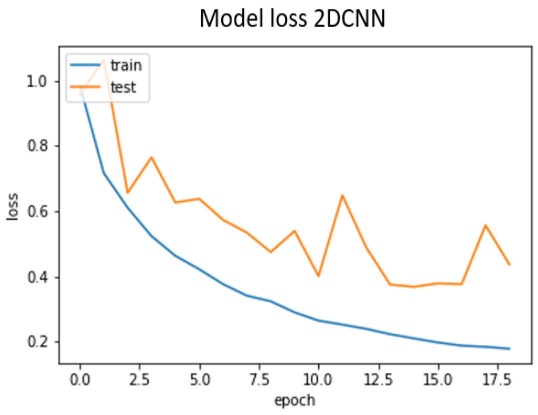
[7] A. Ilyas, A. Mądry, S. Santurkar and D. Tsipras, "How does Batch Normalization Help Optimization?," Gradient Science, 26 November 2018. [Online]. Available: http://gradientscience.org/batchnorm/.

XII. APPENDIX A: TRAINING CURVES

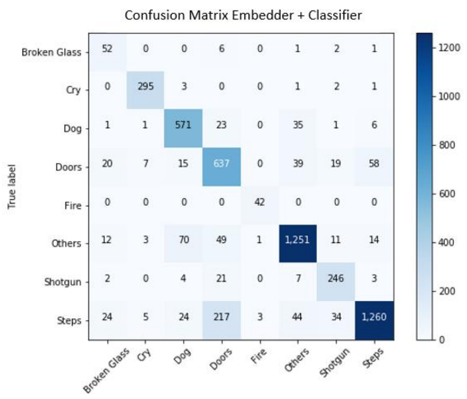






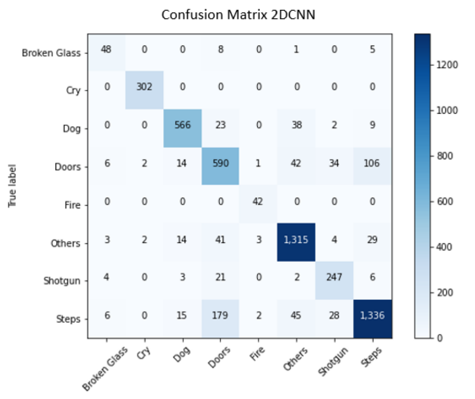


XIII. APPENDIX B: CONFUSION MATRICES



Confusion Matrix Embedder + Classifier

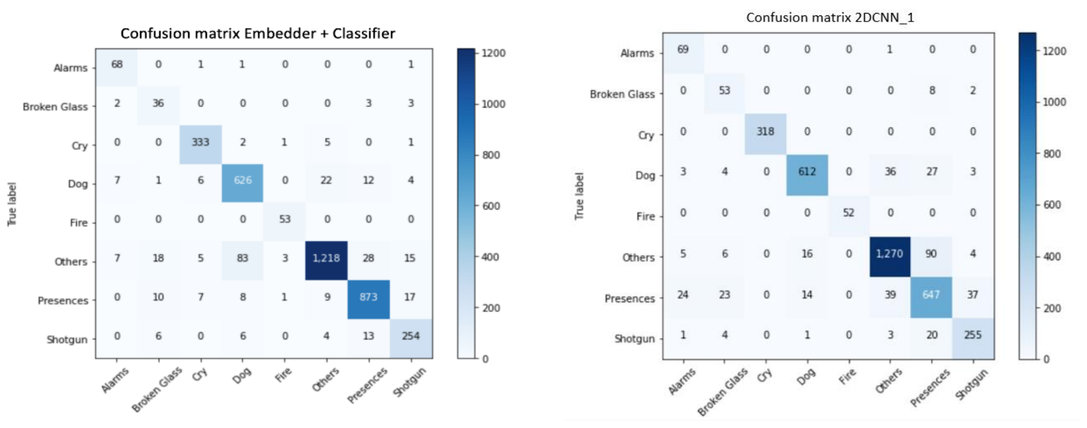
True label	Broken Glass	Cry	Dog	Doors	Fire	Others	Shotgun	Steps
Broken Glass	52	0	0	6	0	1	2	1
Cry	0	295	3	0	0	1	2	1
Dog	1	1	571	23	0	35	1	6
Doors	20	7	15	637	0	39	19	58
Fire	0	0	0	0	42	0	0	0
Others	12	3	70	49	1	1,251	11	14
Shotgun	2	0	4	21	0	7	246	3
Steps	24	5	24	217	3	44	34	1,260



Confusion Matrix 2DCNN

True label	Broken Glass	Cry	Dog	Doors	Fire	Others	Shotgun	Steps
Broken Glass	48	0	0	8	0	1	0	5
Cry	0	302	0	0	0	0	0	0
Dog	0	0	566	23	0	38	2	9
Doors	6	2	14	590	1	42	34	106
Fire	0	0	0	0	42	0	0	0
Others	3	2	14	41	3	1,315	4	29
Shotgun	4	0	3	21	0	2	247	6
Steps	6	0	15	179	2	45	28	1,336

Figure 5: Confusion matrix of the training of the final models on the original data (own elaboration).



Confusion matrix Embedder + Classifier

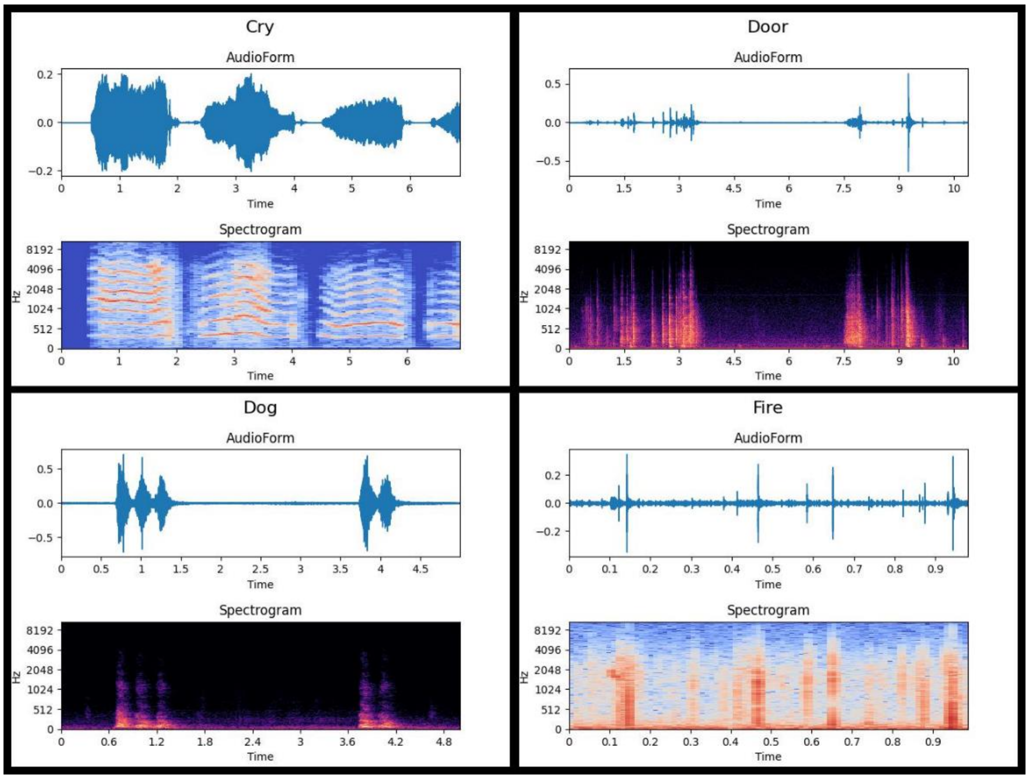
True label \ Predicted	Alarms	Broken Glass	Cry	Dog	Fire	Others	Presences	Shotgun
Alarms	68	0	1	1	0	0	0	1
Broken Glass	2	36	0	0	0	0	3	3
Cry	0	0	333	2	1	5	0	1
Dog	7	1	6	626	0	22	12	4
Fire	0	0	0	0	53	0	0	0
Others	7	18	5	83	3	1,218	28	15
Presences	0	10	7	8	1	9	873	17
Shotgun	0	6	0	6	0	4	13	254

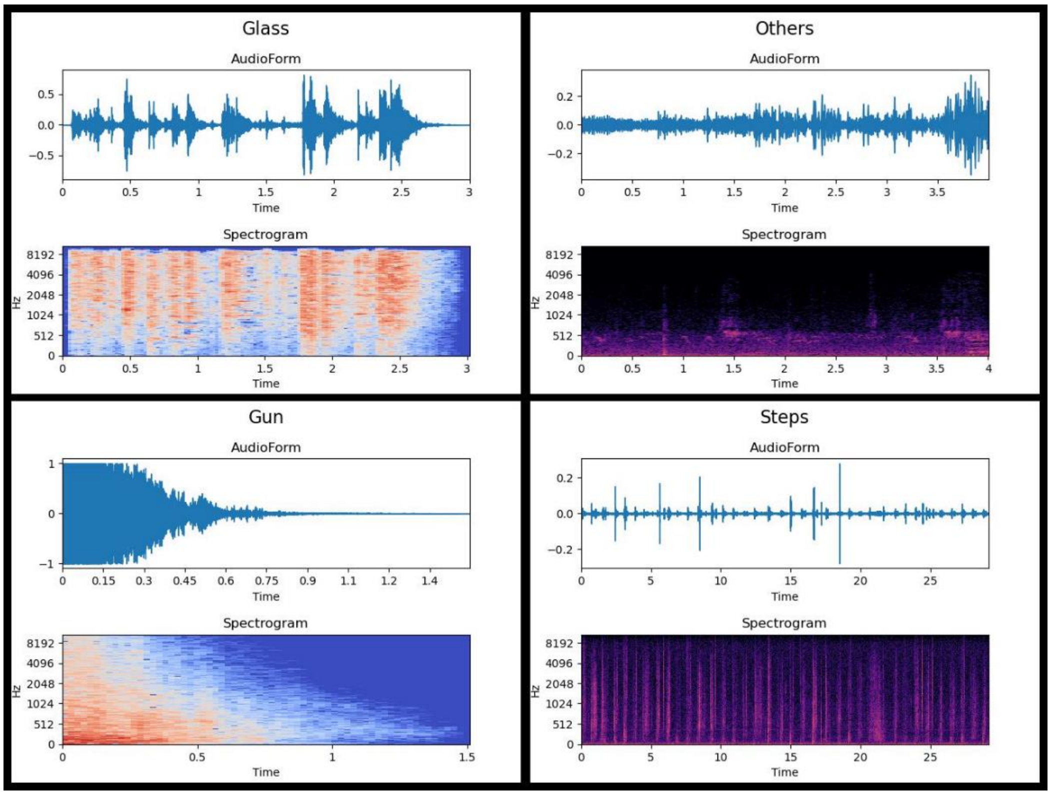
Confusion matrix 2DCNN_1

True label \ Predicted	Alarms	Broken Glass	Cry	Dog	Fire	Others	Presences	Shotgun
Alarms	69	0	0	0	0	1	0	0
Broken Glass	0	53	0	0	0	0	8	2
Cry	0	0	318	0	0	0	0	0
Dog	3	4	0	612	0	36	27	3
Fire	0	0	0	0	52	0	0	0
Others	5	6	0	16	0	1,270	90	4
Presences	24	23	0	14	0	39	647	37
Shotgun	1	4	0	1	0	3	20	255

Figure 6: Confusion matrix of the training of the final models on the final data (own elaboration).

XIV. APPENDIX C: SPECGTROGRAM EXAMPLES




Glass
AudioForm
Spectrogram
Others
AudioForm
Spectrogram
Gun
AudioForm
Spectrogram
Steps
AudioForm
Spectrogram
Time
Hz